AutoUni – Schriftenreihe

Florian Neukart

Reverse Engineering the Mind

Consciously Acting Machines and Accelerated Evolution

AutoUni – Schriftenreihe

Band 94

Herausgegeben von/Edited by
Volkswagen Aktiengesellschaft
AutoUni

Die Volkswagen AutoUni bietet Wissenschaftlern und Promovierenden des Volkswagen Konzerns die Möglichkeit, ihre Forschungsergebnisse in Form von Monographien und Dissertationen im Rahmen der „AutoUni Schriftenreihe“ kostenfrei zu veröffentlichen. Die AutoUni ist eine international tätige wissenschaftliche Einrichtung des Konzerns, die durch Forschung und Lehre aktuelles mobilitätsbezogenes Wissen auf Hochschulniveau erzeugt und vermittelt.

Die neun Institute der AutoUni decken das Fachwissen der unterschiedlichen Geschäftsbereiche ab, welches für den Erfolg des Volkswagen Konzerns unabdingbar ist. Im Fokus steht dabei die Schaffung und Verankerung von neuem Wissen und die Förderung des Wissensaustausches. Zusätzlich zu der fachlichen Weiterbildung und Vertiefung von Kompetenzen der Konzernangehörigen, fördert und unterstützt die AutoUni als Partner die Doktorandinnen und Doktoranden von Volkswagen auf ihrem Weg zu einer erfolgreichen Promotion durch vielfältige Angebote – die Veröffentlichung der Dissertationen ist eines davon. Über die Veröffentlichung in der AutoUni Schriftenreihe werden die Resultate nicht nur für alle Konzernangehörigen, sondern auch für die Öffentlichkeit zugänglich.

The Volkswagen AutoUni offers scientists and PhD students of the Volkswagen Group the opportunity to publish their scientific results as monographs or doctor’s theses within the “AutoUni Schriftenreihe” free of cost. The AutoUni is an international scientifc educational institution of the Volkswagen Group Academy, which produces and disseminates current mobility-related knowledge through its research and tailor-made further education courses. The AutoUni‘s nine institutes cover the expertise of the different business units, which is indispensable for the success of the Volkswagen Group. The focus lies on the creation, anchorage and transfer of knew knowledge.

In addition to the professional expert training and the development of specialized skills and knowledge of the Volkswagen Group members, the AutoUni supports and accompanies the PhD students on their way to successful graduation through a variety of offerings. The publication of the doctor’s theses is one of such offers. The publication within the AutoUni Schriftenreihe makes the results accessible to all Volkswagen Group members as well as to the public.

Herausgegeben von/Edited by
Volkswagen Aktiengesellschaft
AutoUni
Brieffach 1231
D-38436 Wolfsburg
http://www.autouni.de

Florian Neukart

Reverse Engineering the Mind

Consciously Acting Machines and Accelerated Evolution

Springer

Florian Neukart
Wolfsburg, Germany

Any results, opinions and conclusions expressed in the AutoUni Schriftenreihe are solely those of the author(s).

AutoUni – Schriftenreihe
ISBN 978-3-658-16175-0 ISBN 978-3-658-16176-7 (eBook)
DOI 10.1007/978-3-658-16176-7

Library of Congress Control Number: 2016955691

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer Fachmedien Wiesbaden GmbH
The registered company address is: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

The only way to discover the limits of the possible
is to go beyond them into the impossible.

Sir Arthur C. Clarke

Preface

What we perceive as consciousness seems to be an anomaly, and so is intelligence. Earth features a biodiversity of around $8.7 * 10^6 \pm 1.3 * 10^6$ SE (between ~7,400,000 and ~10,000,000) organisms, [1] from those just a few show rudimentary forms of both consciousness and intelligence, and only one seems to be aware and discuss what it means to exist. Although more than 1,000 exoplanets have been detected thitherto (latest estimations predict around $1.7 * 10^9$ only in our galaxy), and 12 of them are probably habitable, there has not been any indication that intelligent life apart from earth has emerged elsewhere. This may be interpreted in many ways, whereby the most appealing ones for me are:

- Intelligence is an anomaly.
  This assumption states that the probability for the evolution of human-like intelligence in a species is infinitely small. No other species on earth seems to have developed it, which can be considered as strong indication for the hypothesis that if extraterrestrial species exist, they may not have developed it either. A similar thought experiment is about the physical laws governing our universe. If universe parameters, such as the speed of light or the strength of gravity, would have been allowed to take any value from 0 below infinity, then the occurrence of the current set of parameters and parameter values governing the very existence of the universe (and life) as we know it is infinitely low. Although I personally do not hope and guess that this hypothesis is true, it has not yet been disproved by scientific evidence.
- We are the first ones searching for intelligent life in our galaxy.
  Considering the size of our galaxy and the time the universe already exists, it is very likely that if intelligent life except from ours exists in whatever form, we will detect it sooner or later, but: if we are not the only intelligent species in the universe it is, given the almost 14 billion years that elapsed since the big bang, very unlikely that we are the first ones having evolved intelligence, and in consequence the technology to communicate and travel through space. More than half a century ago, even Enrico Fermi has faced the paradox that considering the age of our universe, it should be very likely that intelligent life has already emerged elsewhere. But where is it, then? This became widely known as the Fermi-paradox.
- Other intelligent species have already discovered us, but do not (or cannot) show up for some reason.
- No other species has solved the problem of how to produce exotic energy, which is, as to current knowledge, required for travelling faster than light (to be exact, it is not travelling faster than light, but bridging of distances by warping space-time). As no matter can travel faster than light, it must be space-time that is moved through space-time. This can theoretically be done by creating a warp-field around an area of space-time (and an object like a spaceship within this area), which is bordered by a singularity in front of the object and controlled expansion of space-time behind it. In theory this is possible, as has already been shown with a special solution of Einstein's field equations in general relativity. This solution states that the creation of an energy-impulse-tensor modifying the space-time around a spacecraft in the sense that the distance between start- and endpoint can be

1 Tittensor Derek P. et al. (2011): How Many Species Are There on Earth and in the Ocean? [2013-10-24]; URL: http://www.plosbiology.org/article/info%3Adoi%2F10.1371%2Fjournal.pbio.1001127

reduced is possible. However, for creating such a warp-field exotic matter, thus matter not only not consisting of protons, neutrons or electrons, but additionally being of negative energy density must be available[2] – this may be a problem, as we have not even been able to proof that it exists.

From a solely scientific point of view all of these and a lot more arguments are valid options, but I will focus on the first one here. Mankind has always dreamed of immortality, which has, amongst others, been one of the major reasons for why our species devised the concept of religion. Our brain provides us with the abilities required to understand the universe, and the more we understand about our role in the universe and life itself, the more transhumanistic our views become. It may be not easy to accept that we will most likely take care of our immortal soul ourselves in the not-so-distant future, but this is exactly what will happen. Problems like starvation or over-population may be solved by leaving our bodily existence behind, and hundreds of years-lasting journeys to exoplanets for colonization will not pose a challenge for human minds transferred into the computers of spaceships. This is just a first impression of what the future of mankind could be like. The ideas are countless, and history taught us that advancement of science not always bears only good. Anyway, this work marks the beginning of a journey – the journey towards consciously acting machines and artificially accelerated human evolution.

Florian Neukart[3]

2 Alcubierre Miguel (1994): The warp drive: hyper-fast travel within general relativity; Classical and.Quantum.Gravity 11:L73-L77, 1994

3 I am especially eager to note any copyrights of multimedia elements such as images and texts used, and if possible to use graphics and text of my own. However, most of the time scientific work is based on the work already done by others, which becomes apparent by having a look at the number of publications cited within this elaboration. Every publishing scientist knows that one challenge is that the more one reads and studies about a topic, the easier the boundaries of their own ideas get blurred with those of others' ideas. Thus, if in this elaboration unmarked, but by third party copyright protected images or a text are found, it was not possible for me to detect the related copyright. In case of such an unintentional copyright violation I will remove the corresponding picture or text element or will mark it with an appropriate copyright notice/citation of sources indicated in the next version of the publication after a short notification.

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AI	Artificial Intelligence
AIS	Artificial Immune System
ANN	Artificial Neural Network
AOD	Agent Oriented Development
AOP	Agent Oriented Programming
APC	Antigen-Presenting Cell
BP	Back propagation
CI	Computational Intelligence
CSV	Comma separated values
DBN	Deep Belief Network
DM	Data Mining
FFANN	Feed-forward Artificial Neural Network
ETL	Extract, Transform, Load
GA	Genetic Algorithm
MLP	Multi Layer Perceptron
NDS	Neural Data Set
OOP	Object Oriented Programming
PAMP	Pathogen-Associated Molecular Patterns
QANN	Quantum ANN
RANN	Recurrent ANN
RBM	Restricted Boltzmann Machine
RP	Resilient-propagation
(R)MSE	(Root) Mean Squared Error
SA	Simulated annealing
SHOCID	System applying High Order Computational Intelligence in Data Mining
SRANN	SHOCID RANN

Introduction

This elaboration is, in some sense, the first version of a manual describing how to implement an artificial conscious entity and how to extend the very human existence beyond biological limitations. Subsequent versions depend on future research not only conducted by the author, but by numerous scientists from various fields.

Since there is the research field of artificial intelligence (AI), one of the biggest hurdles has always been the creation of conscious experiences in machines. Not only lots of different definitions of what exactly consciousness is exist from a philosophical point of view; it is yet also not completely understood on a neuroscientific level how our brain creates conscious content. Within this elaboration, I will provide a foundation for understanding how conscious experiences emerge by approaching the topic by means of actual and future technology. In my opinion, one of the most important foundations of consciousness is, amongst self-awareness, the ability to understand concepts, or more generally, the understanding of 'things'. I define these things to comprise everything that exits, be it a single atom or a complex lifeform. When it comes to understanding things, learning is an important aspect, and for being able to measure levels of understanding, it is first required to define and quantify when something has been understood. I want to emphasize this challenge at the very beginning of this elaboration, as words such as 'learning', 'understanding' and 'consciousness' are suitcase-words from psychology helping us to discuss complex subjects science has not yet entirely understood. Suitcase-words such as consciousness enable us to include yet unknown processes and (changes of) states associated with the human brain in our everyday-language with ease, not only without being able to describe what accounts for a conscious experience on neuronal or (sub-) atomic layers, but also without being able to explain what consciousness is on a more abstract layer. Lots of scientists from different fields have been working on disclosing the secret of consciousness, and numerous different explanations have been published and discussed controversially. Most of these theories feature a common denominator – the inclusion of a feature set, which is associated with the perception of consciousness. It would be counterproductive to reject such approaches, as only the detailed description and combination of single features will allow us to reproduce conscious behavior in artificial entities. We will deal with this and a lot more in this book, and also define these features crisply, as only then we will be able to create hard- and software capable of not only processing information in the way the human brain does, but also capable of reproducing the conditions that are required for creating conscious experiences.

Not only philosophers, but scientists from numerous different fields have long tried to understand what it is that creates such experiences; from what we can see today, some of the attempts have already been crowned by success. The strength of current AI is not only justified in the fact that today's computers can do things better at which machines have used to be better than humans since the first successful implementations of paradigms of AI. Back in 1996, as IBM's Deep Blue beat Garri Gasparov in chess, one could have argued that this had nothing to do with real intelligence, but resulted from the fact that Deep Blue could calculate $2 * 10^6$ chess moves per second. However, another of IBM's masterpieces, Watson, showed that by the combination of linguistic pre-processors, expert systems, search engines, machine learning, logic, natural language understanding, and by accessing data sources of various kinds intelligently, understanding can be emulated, which makes it more difficult for opponents of AI to argue against it. Both examples are very impressive not only from scientific and engineering points of view, however although these systems are artificially

intelligent, they cannot be considered conscious beings (and, as of now, it has never been the aim of making them conscious).

For what it can mean to create conscious machines becomes even more interesting by taking into account the research of Stuart Hameroff and Roger Penrose, who state that quantum physical phenomena take place on cell-basis, or even on the basis of tubulin dimers, elements occurring within nerve (and all other eukaryotic) cells' cytoskeletons: if quantum coherence, which may constitute a requirement for consciousness, may possibly be maintained at this level, this would result in a dramatic increase of the number of operations per second a brain is able to accomplish. Even if quantum physical phenomena have not yet been linked to the creation of conscious content by proof, it is nevertheless useful taking such into account for the implementation of artificial entities that should emulate or experience conscious content themselves. Thus, within this work the implementation of quantum artificial neural networks (by the leveraging the power of quantum computers) is juxtaposed in opposition to classical artificial neural networks in terms of their probable capabilities and usefulness for the implementation of an artificial mind.

The creation of consciousness in an artificial entity brings up lots of new questions, particularly the one after human immortality; this may not be obvious at a first glance, but if artificial entities are capable of producing conscious content and may theoretically live forever, can we humans then benefit thereof in the sense of transferring our minds, dreams and desires into such vessels, one may ask. But then there is the continuous consciousness-problem: if we copy our mind from our biological brain into an artificial vessel, it is just that – a copy, and in the worst case two of us exist at the same time, one doomed to die and the other blessed to live forever. I will also propose a solution to this problem.

Finally, I ask the reader to pardon me for the technical style of writing in which I sometimes slide. I am admittedly used to creating scientific reports/ papers, or technical and functional documentations, but not so much to writing bedtime lecture.

Last, but not least, I sometimes refer to some computational intelligence paradigms with the prefix 'SHOCID'. This is, because the ideas thereto emerged during the specification phase of the data mining system SHOCID (System Applying High Order Computational Intelligence in Data Mining),[4] which I developed some years ago. This gives the elaboration at hand a practical touch – most of the discussed and introduced computational intelligence-paradigms have been implemented in SHOCID.

4 Neukart Florian (2013): System Applying High Order Computational Intelligence in Data Mining and Quantum Computational Considerations Concerning the Future of Artificial Intelligence; Brasov: Transilvania University of Brasov

Structure

Chapter 1

In this chapter basic brain functionality as well as the rough anatomy of the human brain is discussed. This is mainly because as an artificial intelligence researcher one has to deal with natural intelligence and how the brain works sooner or later. A subfield of artificial intelligence, computational intelligence, comprises many nature-inspired approaches such as swarm intelligence, genetic algorithms or artificial neural networks. Especially some kinds of artificial neural networks, by means of which we are concerned with creating more or less simple imitations of their biological counterparts, have achieved information processing similar to what we currently understand happens within biological brains.

Chapter 2

The major goal of this elaboration is to work out how specific aspects of the human mind, namely those responsible for higher cognitive functions, function, and to figure out which hardware is required to process an artificial mind with the same, similar or superior capabilities. For this, I will focus on any approach that allows us to reproduce cognitive capabilities, but not necessarily achieve this target by the same means as evolution did. It makes sense, at this point, to provide an introduction to artificial intelligence in order to understand which areas of the neocortex are subject to AI research and development. This chapter notabene provides only a brief overview of the pillars of AI, as even the detailed elaboration of just one sub-area of each of those would suffice to fill books. In the later chapters I will mostly focus on artificial neural networks, logic, knowledge representation, and speech in order to illustrate how I think the human thought processes can be rebuilt artificially. I will also give a brief introduction to one of my research fields, autonomously acting cars, which should help to understand what it takes to create intelligent, autonomous, social and adaptive agents; rebuilding collective and intelligent behavior in artificial systems not only allows us to understand a significant amount of brain evolution, but also how intelligence makes us a highly complex species in terms of thought processes.

Chapters 3, 4 and 5

In these chapters the fundamental concepts and standards of artificial neural networks are particularized under the consideration of actual knowledge and research conducted. The field of computational intelligence, to which artificial neural networks belong, comprises nature-inspired approaches for solving complex problem statements. Thus, this is where we will begin the search for paradigms that seem to be suitable for engineering artificial conscious entities.

In chapters 4 and 5, some sophisticated artificial neural network structures and learning approaches, some of which have been developed and published by the author, are particularized. One reason for explaining various algorithms is to show how far we can go with artificial neural networks and classical computers, how sophisticated learning algorithms can become, and how such structures can be manipulated. The more important reason for going into some detail with several algorithms is that I want to free the reader's mind, as for achieving brain-like capabilities we do not necessarily need to copy the inner workings of our brain but rely on other approaches – in this special case not the journey is its own reward. I

will, amongst others, discuss how artificial neural networks can be efficiently trained by simulating the survival behavior of bacteria in hazardous environments.

The major question to be answered in this chapter is whether these complex approaches are already powerful enough for generating conscious experiences in an artificial entity. Furthermore, some simple application scenarios for the introduced approaches are explained, creating a basic understanding of how such approaches may be applied practically.

Chapter 6

Deep artificial neural networks are, in some aspects, the most human-like artificial way of processing information we have thitherto. Very impressive and important achievements have been made due to complex network structures and sophisticated training algorithms, and what can already be achieved is not that far from the capabilities of what parts of the human brain, such as the visual cortex, can achieve. However until recently, the typical research in that area concerned problems that are perfectly suitable for being processed on a computer, like the prediction of numerical values or clustering of unstructured data. Furthermore, once trained to data of a sub-domain, an ANN can only be used in that sub-domain. What I want to say is that individual approaches are simply not potent enough for approaching the full stack of capabilities of a human brain. But a combination of several techniques may… With actual training algorithms it would be impossible to train a structure consisting of billions of neurons and thousands of neuron layers. This chapter serves the purpose to understand what ANNs are often used for, and for this we start with simple data analysis and evolve towards more complex information processing in the latter chapters.

Chapters 7 and 8

Quantum mechanics is of utmost importance within this elaboration, as, thanks to some of humankind's greatest scientists, within the last century the knowledge within this field has grown extensively and allows more than just an outlook on future developments in artificial intelligence. Some theories of consciousness make use of quantum physical phenomena within the brain for explaining awareness, or conscious experiences. Both chapters are used to explain the most important fundamentals of quantum mechanics and how they may find application within a biological brain. However, as the sole reduction of the brain by means of a quantum computer may not solve all problems, some objections against such hypotheses are raised, contributing to the groundwork for the main chapter of this elaboration.

Chapter 9

Before being able to reverse engineer a human mind, one has to understand some of the most commonly accepted philosophical views of what a mind actually is. Trying to implement algorithms without having thought about what the mind actually is, is impossible. Only philosophy can deliver some answers to this question and I, for my part, chose eliminative materialism to be the most suitable approach to pursue, when grappling with implementation scenarios for an aware, artificial entity. Some of the readers may wonder about the statement that I 'chose' one of many philosophical approaches as useful means for what achieving what is discussed in this book; this is because I encounter consciousness from an engineering point of view, and engineers usually solve problems by the means that are to their disposal. Thus, I consider philosophical views as a means of implementation only in the context of this book.

Chapter 10

The objective of writing this book was the introduction of paradigms and ideas that may serve the purpose of reverse engineering the mind in the near future. Amongst others, in this chapter self-organization in both human and artificial brains, which I consider to be another important aspect for the development of our mental abilities, is discussed. So are some more concepts such as search, information retrieval, or data representation, which form, together with what has been elucidated before, the foundation for giving an artificial conscious entity the ability to understand and interpret the world. Furthermore, I am of the opinion that the implementtation of artificial neural networks on quantum systems represents one of the most valuable approaches for achieving the objectives of this research, as only such may deliver the computational power for processing what we understand to be our mind. Of course, everything is connected in order to form a big picture – the picture of how to reverse-engineer the mind.

Finally, the question on how we humans may benefit from such developments in the sense of transferring our minds into artificial vessels is addressed. How can the continuous consciousness-problem be bypassed, thus the required information in and structure of our brain be extracted instead of just being copied? There seems to be a solution for the dilemma in the sense of a distributed mind, which is discussed likewise.

1 Evolution's most extraordinary achievement

When doing research in the field of artificial intelligence, sooner or later one is required to deal with what I consider the most fascinating thing Nature has equipped us with – the human brain. This extraordinary organ does not only allow us to understand the universe, but additionally provides us with feelings, conscious perception or the ability to control our bodies in highest precision. I have always tried not to shelve myself into a specific field of research; however, above all I am a computer scientist and from a computer scientist's point of view it is, at first, interesting how the brain is capable of processing, storing or recalling information. Inevitably, when starting to deal with the matter this yields a lot more questions – questions that cannot be answered as easily for the brain as for hard- and software we are used to work with every day, even if one is equipped with knowledge about the field of artificial intelligence. Later in this work we will see that today's artificial intelligence is something that strongly differs from what we consider to be biological or human intelligence. Thus, I consider a rough explanation of the human brain's anatomy as well as the known and studied workings of its parts to be a good starting point.

1.1 Anatomy of the human brain

Most of the readers will remember the lessons in school where the human brain has been explained according to its evolutionary development: this will also be the way the matter will be dealt with in this chapter. Some of the following explanations have been inspired by the ones given by Bruce F. Katz[5] and at dasgehirn.info,[6] as the former perfectly well outline the functional principles of the human brain from an engineering point of view, and the latter from a medical point of view. From the abstractive Figure 1 - Human brain below we can see that the brain consists of four big lobes, the brain stem and the cerebellum, the latter looking like a ball of wool attached to the undersurface of the temporal lobe. However, there is a far more abstractive scheme describing the human brain, called triune brain model, which has been introduced by Paul Maclean (Figure 2 - Triune brain).

The triune brain model shows three differently colored areas, stacked according to their occurrence in the brain evolution. The brightest area in Figure 2 - Triune brain represents what is considered to be the most ancient and basic part of the nerve system, the reptilian brain, which has evolved about $3 * 10^8$ years ago. The reptilian brain is responsible for regulating autonomous functions like the heart rate or breathing. The medial part in the figure represents the limbic brain, or paleomammalian, which has evolved about $3 * 10^8$ years ago and which is partially responsible for recalling information, thus memorizing things, and emotions. The neocortex, or neomammalian, which makes up most of the visible part of the human brain (once it has been made visible in some way) has evolved only $1.2 * 10^5$ years ago, and merges not only with the structure of the paleomammalian, but also in its functionality.

5 Katz Bruce F. (2011): Neuroengineering the future - Virtual minds and the creation of immortality; Massachusetts: Infinity Science Press LLC, p. 16 ff.

6 dasgehirn.info: [2013-06-22]; dasgehirn.info; URL: http://dasgehirn.info/

F. Neukart, *Reverse Engineering the Mind*, AutoUni – Schriftenreihe 94, DOI 10.1007/978-3-658-16176-7_1

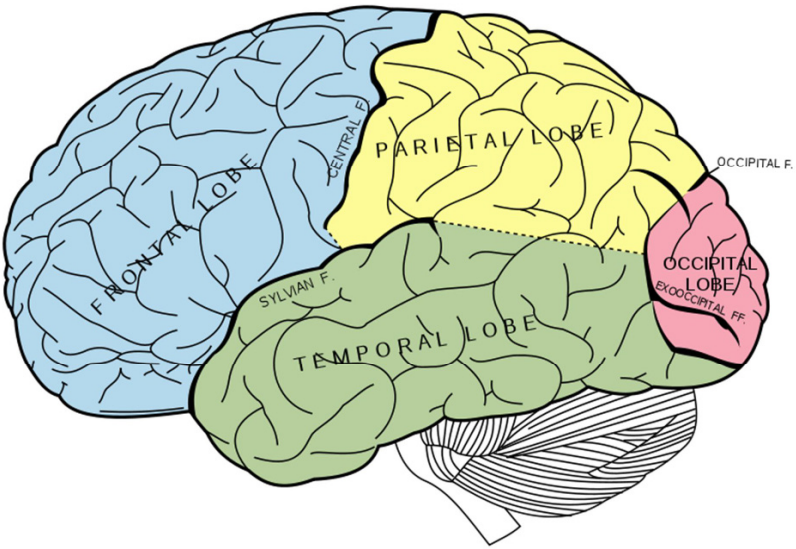

Figure 1 - Human brain[7]

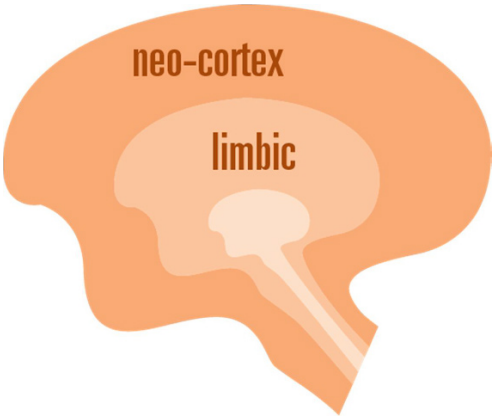

Figure 2 - Triune brain[8]

7 Wikipedia: Human brain [2013-06-19]; Wikipedia; URL: http://en.wikipedia.org/wiki/Human_brain

8 Steven White: Triune brain [2013-06-20]; Steven White; URL: http://blog.stevenwhite.com/

1.1.1 Truncus cerebri

The truncus cerebri (Figure 3 – Truncus cerebri), or brain stem, is the oldest part of the human brain and only as large as a finger, but nevertheless an essential part of our brain, as it regulates circulatory, breathing or sleeping. It consists of four major parts, whereby each of those may be subdivided into further areas. However, the description of the most basic parts does suffice for our purposes here.

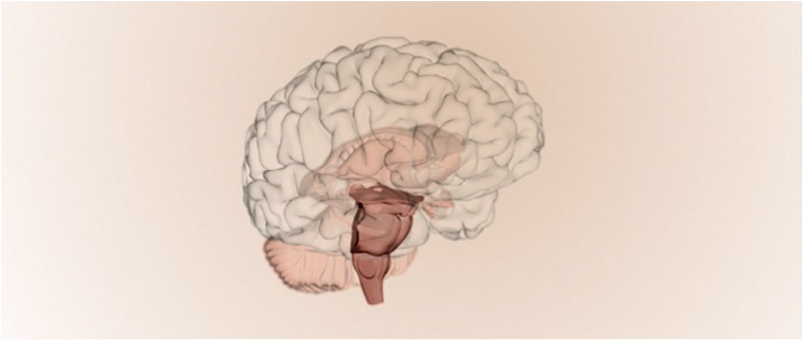

Figure 3 – Truncus cerebri[9]

1.1.1.1 Cerebellum

The cerebellum (Figure 4 - Cerebellum) is known to be responsible for coordination, equilibrium, motor movement and muscle tone. It consists of white matter in the inside and is surrounded by a very tightly folded outer layer of gray matter, called the cerebellar cortex. The number of neurons in the cerebellum makes up one half of all neurons in the brain (~500,000,000), which are used for relaying information between itself and the cerebral cortex' (which we must distinguish from the cerebellar cortex) areas involved in motor controls. An injury of the cerebellum results in disturbances of bodily movements, like a loss of coordination, impossibility of judging distances, the inability for performing fast movements, etc.

9 dasgehirn.info: Der Hirnstamm [2013-06-22]; dasgehirn.info; URL: http://dasgehirn.info/entdecken/anatomie/der-hirnstamm/ (reprinted with permission from dasgehirn.info)

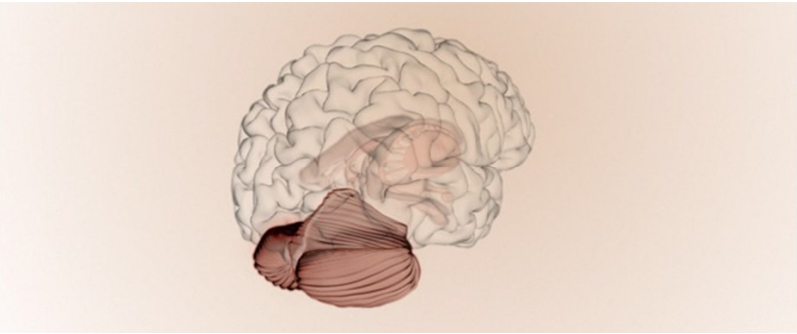

Figure 4 - Cerebellum[10]

1.1.1.2 Mesencephalon

As a part of the brain stem, the mesencephalon (Figure 5 - Mesencephalon) or midbrain connects the prosencephalon (forebrain) with the rhombencephalon (hindbrain). It is responsible for hearing, eye movement, pupil dilation and sight response control, and, as all areas of the brain stem, for motor movements. It comprises the crura cerebri, a bunch of fibers transporting signals from the cortex into the spinal cord or brain nervous cores, and the tegmentum, containing cores like the substantia nigra and the nucleus ruber, the former one being an area that is important for the initiation of movements. Because it is darkly hued, it was named the black substance, or substantia nigra. If the substantia nigra suffers from damage or even complete failure, symptoms of Parkinson occur. The red core, or nucleus ruber, owes his name a high iron content – it is visible to the unaided eye in the mesencephalon and is responsible for muscle tone as well as bodily posture. Another important area is the tectum, consisting of the upper colliculi superiores and the lower colliculi inferiors, the former being responsible for reflex-movements in pupils and eyes, the latter one acting as a control center of the auditory pathway.

10 dasgehirn.info: Das Kleinhirn [2013-06-22]; dasgehirn.info; URL: http://dasgehirn.info/entdecken/anatomie/das-kleinhirn/ (reprinted with permission from dasgehirn.info)

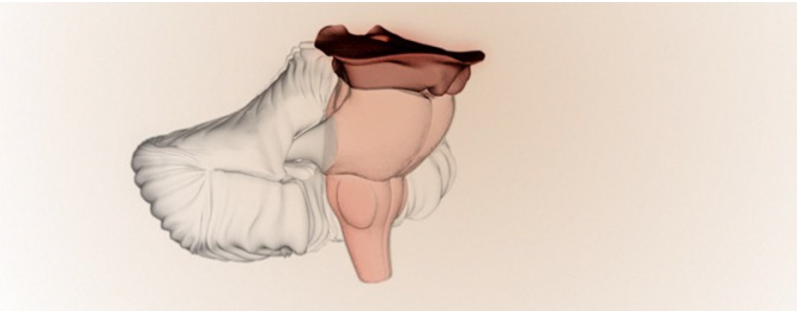

Figure 5 - Mesencephalon[11]

1.1.1.3 Pons

In the brain stem's center, between the medulla oblongata and the mesencephalon we can find the pons (Figure 6 - Pons), which is the Latin word for bridge. Because of its nerve pathways, descending down to the medulla oblongata and ascending up to the mesencephalon, it can be seen as their direct extension. Its name results from the opinion of former physiologists that it connects the two hemispheres of the cerebellum, which in fact is not true. In fact, its fibers are corticophine, thus go from the cortex into the pons, where they are shifted and continue into the cerebellum. Hence, we now know that the pons acts as a relay between cerebellum and cortex and is responsible for translating the signals between cerebellum and motor cortex. Furthermore, it is responsible for numerous vegetative tasks like the steering of cardiac action and breathing, as well as for processing information related to taste and hearing.

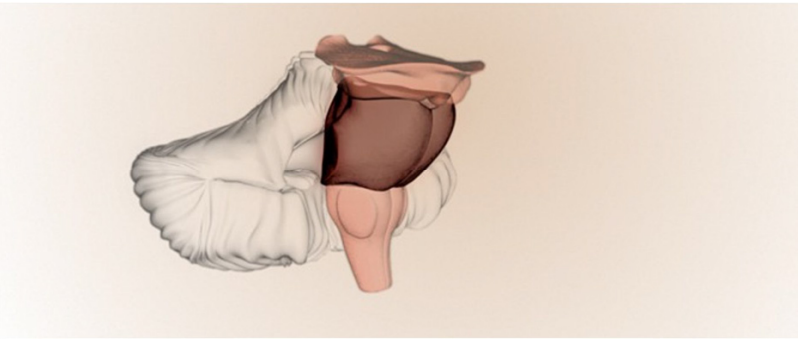

Figure 6 - Pons[12]

11 dasgehirn.info: Das Mesencephalon [2013-06-22]; dasgehirn.info; URL: http://dasgehirn.info/entdecken/anatomie/das-mesencephalon/ (reprinted with permission from dasgehirn.info)

12 dasgehirn.info: Der Pons [2013-06-22]; dasgehirn.info; URL: http://dasgehirn.info/entdecken/anatomie/der-pons/ (reprinted with permission from dasgehirn.info)

1.1.1.4 Medulla oblongata

Within the medulla oblongata (Figure 7 - Medulla oblongata) the medulla (or spinal cord) goes into the brain stem. Between all the fibers there are located lots of important core areas, e.g. reflexes for breathing in and coughing. It is the under most segment of the brain stem, and as a differentiation between the end of spinal cord and the beginning of the medulla oblongata is difficult, it has been denominated the extended spinal cord, or medulla oblongata. Physiologically, it marks the beginning of the brain and contains several important cores for switching neural signals between the medulla and the brain. Noteworthy areas are the cores for breathing or vomiting, as well as breathing, coughing and tasting (together with the pons).

Figure 7 - Medulla oblongata[13]

1.1.2 Paleomammalian

The paleomammalian (Figure 8 - Paleomammalian), or limbic system, has long been seen as the unitary center of our emotions and via numerous popular scientific articles this simplified message is still being transported. However, although the paleomammalian is commonly known to sum up the four Fs (fighting, feeding, fleeing, sexual reproduction [the reader may pardon my rejection for not using the fourth F here]), it provides a lot more than information processing related to emotions; today we know that it also has an important role to play in terms of memory and drive of any kind. However, this does not imply that all of our complex feelings are to be processed within this structure – other structures we get to know later on are involved as well. The second oldest part of our brain comprises four major areas as well, the amygdala, the hippocampus, the hypothalamus and the cingulate gyrus, which are be roughly explained below.

13 dasgehirn.info: Die Medulla Oblongata [2013-06-22]; dasgehirn.info; URL: http://dasgehirn.info/entdecken/anatomie/die-medulla-oblongata/ (reprinted with permission from dasgehirn.info)

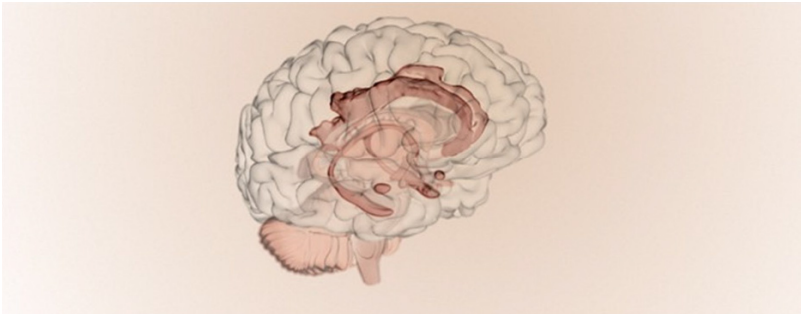

Figure 8 - Paleomammalian[14]

1.1.2.1 Corpus amygdaloideum

The Corpus amygdaloideum, or amygdala (Figure 9 - Corpus amygdaloideum) consists of several cores, although having been named according to one. It is located in both the front part of the temporal lobe as well as the lower part of the side ventricle. As far as we know today, it is involved in the emotions of anger and fear, which has been proved by stimulating this area in animals, which then showed both rage and fear, and when being damaged a result may be a complete lack of these emotions. It is strongly connected to the truncus cerebri, and is also involved in autonomous body functions, like breathing and circulation. It is also strongly connected to the hypothalamus via a bunch of nerve fibers, called the stria terminalis. Via its connection to the hypothalamus, which is responsible for starting adrenaline production in the adrenal glands, it helps to prepare the body so it is able to cope with the reason of fear. Thus, it is responsible for the creation, recognition and bodily reactions of fear, which involves all of our five senses.

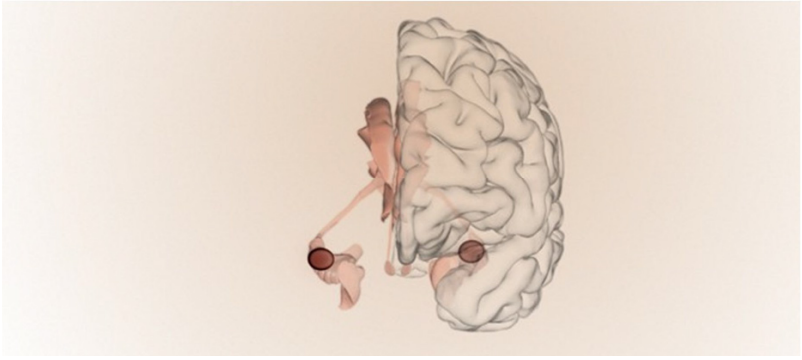

Figure 9 - Corpus amygdaloideum[15]

14 dasgehirn.info: Das limbische System [2013-06-22]; dasgehirn.info; URL: http://dasgehirn.info/entdecken/anatomie/das-limbische-system/ (reprinted with permission from dasgehirn.info)

1.1.2.2 Hippocampus

The hippocampus (Figure 10 - Hippocampus) owes its name a similarity to a sea horse, which may not be seen by everybody. Apart from that, as a central structure of the paleomammalian it has shown to be important in memorizing things. It is located on the inner side of the temporal lobe, at the bottom of the side ventricles and in particular responsible for creating, archiving and recalling information from the long-term memory. If both are missing, no new memories can be created. Thus, it seems to be a very important structure for human consciousness, if one is of the opinion that consciousness is strongly bound to self-perception and the flow of time: a person lacking both hippocampi is captured in the present. Additionally, the hippocampus is one of the few areas from which we know that is capable of building not only new neural connections, but also new neurons.

Figure 10 - Hippocampus[16]

1.1.2.3 Diencephalon

The diencephalon (Figure 11 - Diencephalon), covering almost all sides of the lobes of the telencephalon, is located hidden in the center of the brain. It is divided into four floors, all of which have very different functions. It is attached to the two largest glands of the brain, the pineal gland and the pituitary gland. It comprises the following noteworthy areas.

15 dasgehirn.info: Die Amygdala [2013-06-22]; dasgehirn.info; URL: http://dasgehirn.info/entdecken/anatomie/die-amygdala/ (reprinted with permission from dasgehirn.info)

16 dasgehirn.info: Der Hippocampus [2013-06-22]; dasgehirn.info; URL: http://dasgehirn.info/entdecken/anatomie/der-hippocampus/ (reprinted with permission from dasgehirn.info)

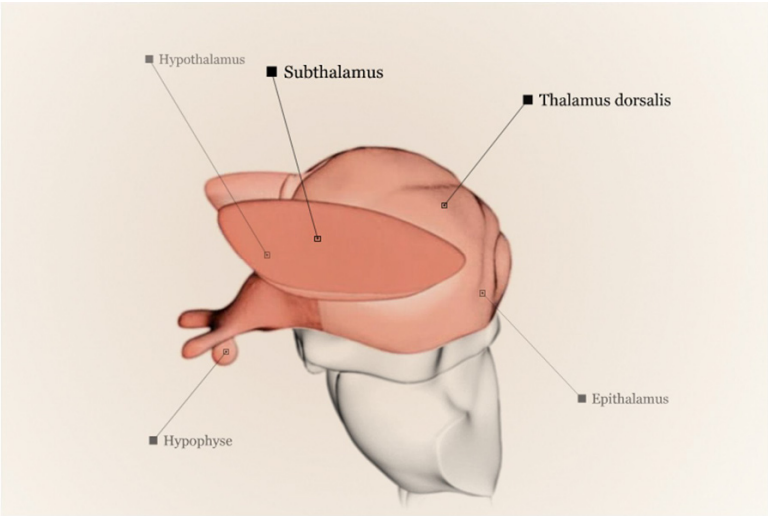

Figure 11 - Diencephalon

1.1.2.3.1 Hypothalamus

The hypothalamus is involved in actions and functions regarding sexual reproduction, temperature regulation, time perception and nutrition. Its name, sub-room, results from the fact that it is located at the bottom of the diencephalon. The rear section of the hypothalamus is streaked by thick nerve fibers, the axons of the fornix, which traverse to the corpora mammillaria, their axons ascending to the anterior thalamus – all of them responsible for memorizing information. The front part of the hypothalamus is traversed by thin nerve fibers and features numerous segregated areas and is connected to the hormones of other endocrine glands by receptors. Additionally, it features nerve cells responsible for steering our biorhythm and for matching it to the diurnal/nocturnal rhythm. Summing up, the hypothalamus is the superior control center of the autonomous nervous system and its rear part belongs to the limbic system. However, the hypothalamus cannot do all of this alone, but requires cooperation with the pituitary gland that produces several hormones, which in turn induce or inhibit the secretion of other pituitary hormones. In front of the hypothalamus one can see the optic chiasm.

Figure 12 - Hypothalamus[17]

1.1.2.3.2 Subthalamus

The subthalamic nucleus is not visible from the ventricle area. His cell masses are migrated laterally in depth - one sees them only when the brain is not centered, but off-center, transverse or longitudinal cut. Its main structure is the subthalamic nucleus, which is associated with the basal ganglia and is involved in the control of motor function.

1.1.2.3.3 Thalamus dorsalis

The dorsal thalamus has inherited the name of the third ventricle and often it is simply referred to thalamus. A shallow groove on the banks of the third ventricle, the hypothalamic sulcus, borders it from the hypothalamus. The thalami of both sides, bulging a little against the ventricles, are connected by an in thickness varying bridge of tissue over the ventricles. This bridge is called adhesio interthalamica, and is not found in any brain, it may be missing, but that has, as far as we know today, no functional significance.

1.1.2.3.4 Pineal gland and Epithalamus

Back up and above the thalamus, the pineal gland is located. It has changed in the course of evolutionary history from a light sensitive organ to an endocrine gland. During the night it produces the hormone melatonin. Together with the tiny Habenulae, which probably have olfactory tasks, it forms the epithalamus.

1.1.2.4 Cingulate gyrus

The cingulate gyrus, or pre-frontal lobe, is an integration center of the cortex, where several sensory data not only converge, but reactions to these are designed and emotions are regulated. It comprises most of the executive functions steering own behavior under consideration of the environment, as well as the working memory. It is superior to the corpus calosum and research suggests that it acts as interface between cortical regions and other

17 dasgehirn.info: Der Hypothalamus [2013-06-22]; dasgehirn.info; URL: http://dasgehirn.info/entdecken/anatomie/der-hypothalamus/ (reprinted with permission from dasgehirn.info)

structures in the limbic system.[18] Evidence regarding the preponderance of spindle cells in this region suggests that it has more in common with higher cortical regions than with the paleomammalian.[19] Katz[20] also mentions that the forward half of the cingulate gyrus, called the anterior cingulate cortex, has come under intense scrutiny as a possible gateway from the paleomammalian and that research found that high activity in the anterior cingulate cortex correlates with depression. Furthermore, he mentions that inhibition of activity by deep brain stimulation in this area may relieve symptoms of depression. Taken together the cingulate gyrus functions as a gateway of emotions from the paleomammalian to the neocortex.

1.1.3 Cortex and neocortex

The cerebrum makes up around 85 per cent of the brain mass. If the cerebral medulla, which especially consists of nerve fibers with the embedded basal ganglia, is removed, then what remains is the neocortex – a layer which is two to five millimeters thick. This layer is also called gray matter because it is rich in nerve cell bodies that give it a reddish brown to gray color. One has determined the number of nerve cells (neurons) in the cerebral cortex with about 23,000,000,000 in the male and about 19,000,000,000 in the female brain - it being noted that the average male body is also larger than the female.

The neocortex is the evolutionary youngest part of the human brain, with an age of 120,000 years, what suggests, compared to the other brain parts, that evolution has not finished it yet. The neocortex consists of four large areas, which are the temporal lobe, the frontal lobe, the occipital lobe and the parietal lobe. As far as we know today it is this structure that provides us with significant advantages compared to other known species. From what we know we are the only species on earth featuring generative language skills and using tools on a regular basis. Additionally, we might be the only ones who will do self-modifying, by the virtue of the achievements of neuroengineering. The cerebral cortex alone takes nearly one half of the brain volume, which becomes possible as it is a six-fold structure. Each of its layers consists of a characteristic population of neuron types and densities, however, this varies throughout the whole brain, and the function of each layer is different. All in all, it is responsible for the coordination of perception, motivation, learning and higher thinking. What we can additionally see is that the neocortex is not only folded on layer-level, but also on surface-level. These canyons are called sulci, the high areas gyri. The overall structure consists of two hemispheres, connected by the corpus callosum, a large bunch of fibers obviously used for communicating signals between the two. There have been conducted several studies on split-brain patients, which will not be discussed here in detail. The neocortex is the part of the brain I have been focusing on the later chapters, when it comes to reverse engineering the mind, as in it all higher cognitive functions take place.

---

18 Katz Bruce F. (2011): Neuroengineering the future - Virtual minds and the creation of immortality; Massachusetts: Infinity Science Press LLC, p. 23.

19 Allman John (2001): The anterior cingulate cortex: The evolution of an interface between emotion and cognition; New York: New York Academy of Sciences, p. 935, 107 - 117

20 Katz Bruce F. (2011): Neuroengineering the future - Virtual minds and the creation of immortality; Massachusetts: Infinity Science Press LLC

1.1.3.1 Frontal lobe

The frontal lobe (Figure 13 - Frontal lobe) is the largest structure in the human brain, occupying the front part of the brain from the cortex to the central gyrus. The frontal lobe is not perfectly understood yet, but it is assumed that it processes, amongst motoric tasks, the higher cognitive functions in the area of the prefrontal cortex. Research has shown that this area plays a role in activities that require attention, thinking, decision-making and planning. Furthermore, it is assumed that it is the residence of personality. Directly at the central gyrus the primary motor cortex is located, which is involved in deliberate movements, where the neural areas for the face, especially lips, mouth and tongue (where the latter one does not specifically belong to the face) occupy most of the space, compared to other areas. In the front of the primary motor cortex we can find the pre-motoric cortex, which is involved in complex movements. Finally, the prefrontal cortex is massively interconnected and associated with executive functions, which makes it important for personality and character. It is involved in social functions and socially appropriate behavior; it has been observed that damage to the prefrontal cortex results in strong changes in personality, as well as motoric disorders. Another point that suggests that it is one of the most complex areas of the brain is that it is not fully developed until the age of 25.

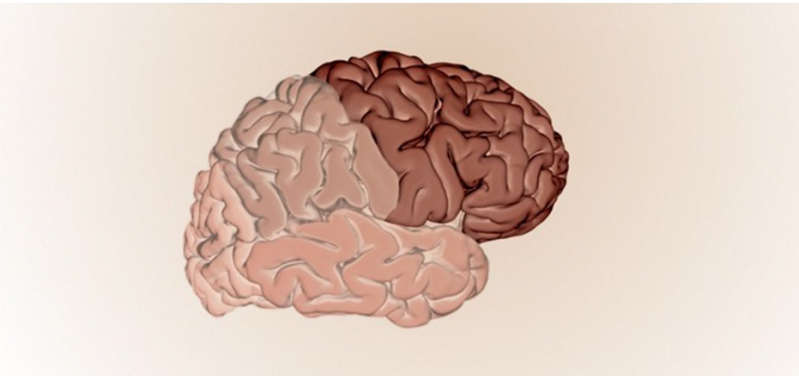

Figure 13 - Frontal lobe[21]

1.1.3.2 Parietal lobe

The parietal lobe (Figure 14 - Parietal lobe) is one of the four major lobes of the cerebral cortex. It is located behind the front and above of the occipital lobe. In its front area somatosensory processes take place in the rear sensory information is incorporated, whereby the handling and orientation of objects in space are enabled. Thus, on the one hand, it is responsible for what the body feels, where one's own limbs are and in what position. Here proprioceptive, auditory, and vestibular information are integrated, so combined to form a larger whole. Particularly important are additional visual information of the dorsal where-

21 dasgehirn.info: Der Frontallappen [2013-06-22]; dasgehirn.info; URL: http://dasgehirn.info/entdecken/anatomie/der-frontallappen/ (reprinted with permission from dasgehirn.info)

way, and the ventral what-way. This creates a three-dimensional image of the environment that is constantly brought up to date and allows us purposeful movement. This concerns both vicinity as well as distance. Thus, the posterior parietal cortex uses this information in relation to local and remote area of the environment. Faults can lead to a 'neglect' – the failure of recognizing one's own limbs, or one half of the environment.

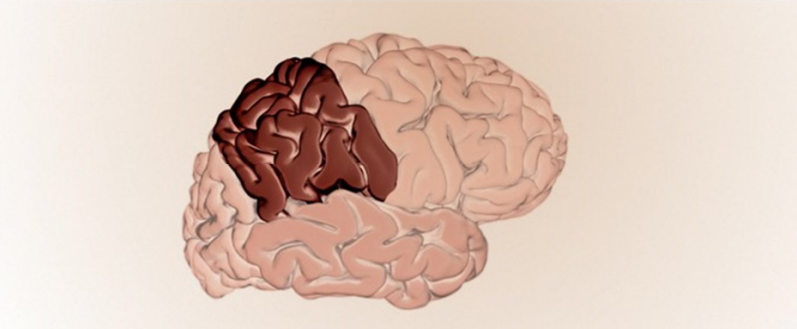

Figure 14 - Parietal lobe[22]

1.1.3.3 Temporal lobe

The temporal lobe (Figure 15 – Temporal lobe) goes into the occipital and parietal lobes without a sharp boundary. It is divided from the frontal lobe via a deep furrow, the lateral fissure, which contains the insula. When the brain viewed from below it can be seen that the two temporal lobes frame the brain stem. Perhaps the most famous feature of the temporal lobe is the processing of information responsible for listening. Because the primary auditory cortex, the so-called Heschl's transverse turns, are hidden in the deep lateral fissure. After a few synaptic switches in the brain stem and thalamus in these windings ends the auditory pathway that transmits signals from the sensory cells in the cochlea of the ear. The primary auditory cortex in Heschl's transverse coils is only about stamp size. The downstream secondary and tertiary auditory centers are a lot bigger. They are located in the upper and middle turn of the temporal lobe and take almost the entire cortical surface of the temporal lobe, which can be seen in the side view. Thus, listening is one of the most widespread systems of our cerebrum – language and music seem to require a high computational effort.

Where the upper and middle temporal convolution to the rear pass into the cortices of the occipital lobe, which is mostly used for the visual system, auditory and visual functions overlap. There one can find lexical centers that have to do with the recognition of written and spoken words. Looking from the bottom of the temporal lobe, one discovers on its inner surface, just behind his blunt anterior pole, a small, inward bulging, which is called uncus, or hook. In its three-layer allocortical surface of the olfactory tract ends. Just below these olfactory cortices, even forming a part of the olfactory cortices, the amygdala can be found, which we remember belongs to the limbic system and is functionally responsible for the

22 dasgehirn.info: Der Parietallappen [2013-06-22]; dasgehirn.info; URL: http://dasgehirn.info/entdecken/anatomie/der-parietallappen/ (reprinted with permission from dasgehirn.info)

affective coloring of our experience. For the memory of the temporal lobe plays an important role; there are allocortical areas again, so atypical six-fold cortical areas serving these functions, which are counted to the limbic system. The innermost, wide turn of the temporal lobe, which can be seen in the bottom view, is the parahippocampal gyrus. Integrated in it is the entorhinal cortex, which acts as a kind of interface between just now experienced things and the memory of the system. Right next to it and something above it is the hippocampal formation. To get them to face, one would have to cut the temporal lobe and inspect the inside. Together, the hippocampal formation and the enthorinal cortex are responsible for the reading of recent memory contents as well as for the retrieval of existing memories. The kind of memory we are talking about here are not limited to knowledge and biography, rather they allow us to focus in everyday life. Important interfaces between the visual system and memory here are the isocortices on the lower rear area of the temporal lobe. Research has shown that the gyrus fusiformis centers have to do with the cognition and recognition of faces.

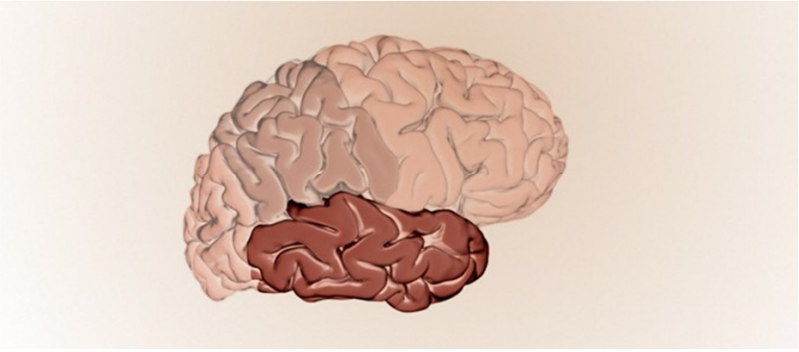

Figure 15 – Temporal lobe[23]

1.1.3.4 Occipital lobe

The occipital lobe (Figure 16 - Occipital lobe) can be roughly divided into two areas: the primary visual cortex, V1 briefly, and the visual association cortices V2 to V5. V1 is mainly on the medial, the inwardly facing side of the hemisphere, thereby forming the wall of the sulcus calcarinus. Incoming nerve impulses, so-called afferents V1 receives via the optic radiation from the lateral geniculate nucleus, a part of the thalamus, with its 1,500,000 million fibers are confronted with 200,000,000 cortex neurons. The primary visual cortex is organized retinotopically, which means that each point on the retina corresponds to a very specific and small cortical area – neighborhoods remain. In the illustration of what is seen the place of sharpest vision is taken by the fovea, which takes only 1.5 millimeters in diameter on the retina, uses four-fifths of V1. This reflects the interconnection of the retina, because the fovea is a ganglion cell to a photoreceptor, or more simply, the resolution is particularly high. Both ensure that what we focus currently is processed at best.

23 dasgehirn.info: Der Temporallappen [2013-06-22]; dasgehirn.info; URL: http://dasgehirn.info/entdecken/anatomie/der-temporallappen/ (reprinted with permission from dasgehirn.info)

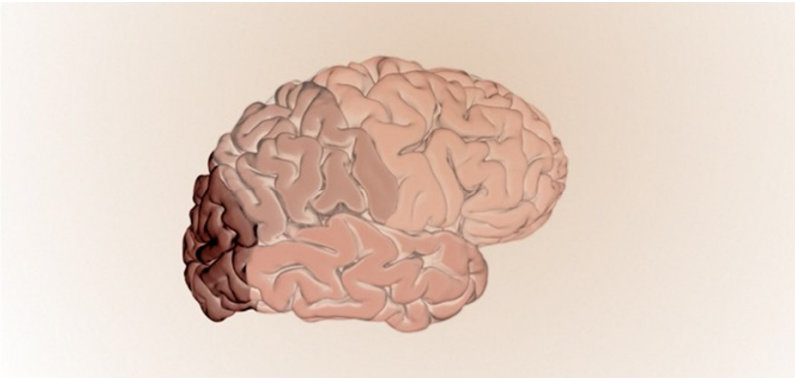

Figure 16 - Occipital lobe[24]

1.2 Neural information transfer

The human brain consists of a network of nearly 10^{11} neurons, and up to 50 times more glial cells. Between these neurons, there exist between 10^{14} and 10^{15} synaptic interconnections. These neurons, acting in a parallel manner, can produce enormous computing power. Each neuron consists of a cell body with a nucleus, as well as dendrites for signal reception and an axon for signal transduction (Figure 17 - Neuron).

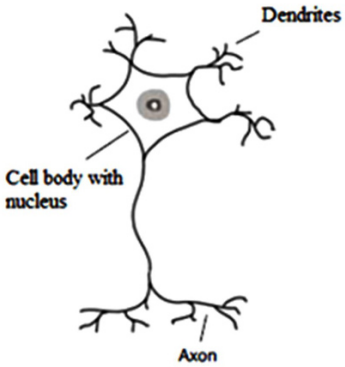

Figure 17 - Neuron[25]

24 dasgehirn.info: Der Occipitallappen [2013-06-22]; dasgehirn.info; URL: http://dasgehirn.info/entdecken/anatomie/der-occipitallappen/ (reprinted with permission from dasgehirn.info)

Even with the naked eye one can distinguish between grey and white matter of the brain. The grey substance consists of the cell bodies of the neurons, which can be found in the core areas and the cortex. The white substance consists of axons, the outgoing fibers of the neurons, which interconnect the single brain structures. In the cerebral cortex and the mesencephalon the neurons are ordered, structured. On the contrary, in the brain stem the order is rather chaotic – a correlation between the complexity of processed tasks and the neural architecture seems to exist. There exist more than 1,000 different types of neurons, which both differ in their form and functionality.

- Sensory neurons translate optical, mechanical, chemical and thermic stimulations in electrical impulses and transfer these to the central nervous system.
- Projection neurons transport signals from one cortex area to another.
- Interneurons act inhibitory or excitatory.
- Neurodoctrine cells emit chemical messenger substances into the blood.
- Motoneurons directly stimulate muscles.

All these neurons look differently, but feature a similar basic structure, consisting of input region, trigger region, conductile region and transfer region.

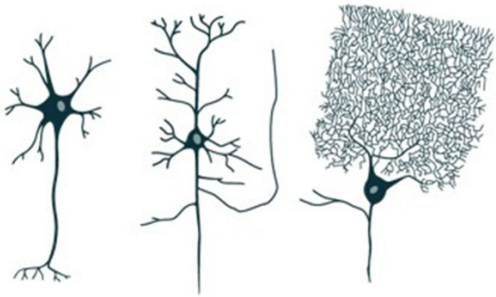

Figure 18 - Neuron types[26]

Figure 18 - Neuron types shows the different types of neurons, on the left a spinal motor neuron, a pyramidal cell of the hippocampus, and a Purkinje cell of the cerebellum.

An excited nerve cell transmits signals in form of electric impulses, along the so called action potentials of its axon, to other neurons. The ions in a neuron determine its electric charge. If a specific neuron, which we call post-synaptic neuron by now, receives signals from other neurons, which we call pre-synaptic neurons here, then its potential may be given a rise. Each of the neurons features a threshold, which may be exceeded if the neuron's potential has been given rise by several pre-synaptic signals. However, this does not necessarily happen, as the

25 Kramer Oliver (2009): Computational Intelligence: Eine Einführung; Berlin Heidelberg: Springer-Verlag, p. 121

26 dasgehirn.info: Zellen: spezialisierte Arbeiter des Gehirns [2013-06-23]; dasgehirn.info; URL: http://dasgehirn.info/entdecken/kommunikation-der-zellen/neuronentypen-9916/image_mediathek_large/ Copyright: Niklas Hippel (reprinted with permission from dasgehirn.info)

synapse transferring the signal may not only be excitatory, thus send a positive signal consequently leading to a rise of the post-synaptic neuron's potential, but also inhibitory, meaning that it decreases the post-synaptic neuron's potential and consequently also decreases this specific neuron's probability for firing. In our example the signal is excitatory, thus the excitement of the neighbor neurons is transmitted into the nucleus by the dendrites, causing an increment of the potential. If the threshold of -60mV is reached via depolarisation, the neuron unleashes an action potential, in other words, transmits its excitement to its neighbor.[27] In doing so, the potential shifts up to +30mV. During the spread, an action potential increases until the neuron's threshold. After the following repolarisation the hyperpolarisation follows. During the hyperpolarisation the resting potential is deceeded and no neighbor neuron can be excited. Finally, the resting potential is exceeded slightly. An action potential is then transmitted until it comes across a synapse, a connection to another neuron (Figure 19 - Action potential).

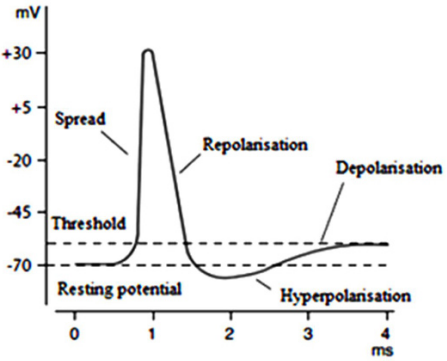

Figure 19 - Action potential[28]

If the action potential of a neuron triggers the action potential of another neuron, the stimulation forwarding through the synapses is increased. This fact is known as Hebbian learning rule (3.3.3.4 Hebb's learning rule). The synaptic connections correspond to the weights in the matrices behind artificial neural networks. The synapses therefore function as transmitters of information, and the result of functioning is either the strengthening or weakening of the stimulation. As a result, the neuron receives signals and some of them are stimulating whereas others are suppressing. The neuron sums stimulating and suppressing impulses. If their algebraic sum exceeds a certain threshold value, the signal at neuron output is transmitted – via axon – to other neurons. Summing up, interconnected neurons allow our brain to process information as it does because of the following features and capabilities:

– An incoming signal to a neuron within is transmitted electrically.
– Between two neurons usually signals are transmitted via chemical neurotransmitters.

27 Kramer Oliver (2009): Computational Intelligence: Eine Einführung; Berlin Heidelberg: Springer-Verlag, p. 120 ff.
28 Kramer Oliver (2009): Computational Intelligence: Eine Einführung; Berlin Heidelberg: Springer-Verlag, p. 121 ff.

- The electrical transmission works on the all-or-nothing principle: Only when the signal strength exceeds a threshold, the action potential within the axon is generated.
- Thereby the synapses translate the electrical signal of the action potential into a chemical one: they release messenger substances and neurotransmitters into the gap between sender and recipient cell.
- The recipient cell takes up the receptors and neurotransmitters and translates them into an electrical signal, the postsynaptic signal words.
- The message of urgency and a signal is reflected in the number and frequency of action potentials; we will discuss this later in more technical detail, as also in artificial neural network research these so-called spiking neurons have come into discussion.

1.3 Summary

I am aware of the fact that there is a lot more to say about the human brain, like on how neuron cells are made up, on how information processing happens or even on its anatomy. All of the required explanations will be given in the next chapters, either when there are approaches introduced that seek to incorporate quantum physical effects in the information processing of the human brain. However, a lot of information will not be provided in this elaboration, as comprehensive knowledge of the human brain is not required for understanding the discussed paradigms. What is, on the contrary, important to know are the major parts of the human brain, what they are responsible for and partially how they work. If required, further explanations follow in the following chapters, but for the beginning the information provided within this chapter forms a solid foundation we can build upon, like

- the description of the most important parts of the human brain,
- the different types of neurons and
- how neurons work from a technical point of view.

2 Pillars of artificial intelligence

The major goal of this elaboration is to work out how specific aspects of the human mind, namely those responsible for higher cognitive functions, function, and to figure out which hardware is required to process an artificial mind with the same, similar or superior capabilities. For this, I will focus on any approach that allows us to reproduce cognitive capabilities, but not necessarily achieve this target by the same means as evolution did. It makes sense, at this point, to provide an introduction to artificial intelligence in order to understand which areas of the neocortex are subject to AI research and development. This chapter notabene provides only a brief overview of the pillars of AI, as even the detailed elaboration of just one sub-area of each of those would suffice to fill books. In the later chapters I will mostly focus on artificial neural networks, logic, knowledge representation, and speech in order to illustrate how I think the human thought processes can be rebuilt artificially. I will also give a brief introduction to one of my research fields, autonomously acting cars, which should help to understand what it takes to create intelligent, autonomous, social and adaptive agents; rebuilding collective and intelligent behavior in artificial systems not only allows us to understand a significant amount of brain evolution, but also how intelligence makes us a highly complex species in terms of thought processes.

An early definition of artificial intelligence on the part of the IEEE Neural Networks Council was 'the study of how to make computers do things at which, at the moment, people are better'[29]. This is still valid today, but the research also focused on letting software do things better, in which computers have always been better, as the analysis of large data sets. Data forms the basis for the development of artificial intelligent software systems that will not only collect information, but is able to

- learn,
- understand and interpret information,
- adapt its behavior,
- plan,
- conclude,
- solve problems,
- think abstract,
- come up with ideas, and
- understand and interpret language.

After having worked for about 10 years in the field, I feel more comfortable with the more general definition:

Artificial intelligence is a machine based form of intelligence that is not distinguished from actual natural intelligence, covering amongst others:

- cognitive intelligence
- emotional intelligence
- social intelligence
- tactic knowledge (everyday-life like)
- collective intelligence (swarm)

29 Rich E., Knight K. (1990): Artificial Intelligence, 5

F. Neukart, *Reverse Engineering the Mind*, AutoUni – Schriftenreihe 94, DOI 10.1007/978-3-658-16176-7_2

I defined both in separate publications (the latter still being subject to secrecy obligations), but a quick literature review will show that both definitions have been published in similar ways by numerous other researchers. Generally, I pursue the holistic approach towards artificial intelligence, which is concerned with incorporating all or at least several pillars of AI. People very often talk about AI in scenarios where they applied machine learning or some other sophisticated approaches to solve a specific problem, i.e. market prediction. This is not what I consider to be AI – it's just an algorithm performing calculations. For me, there exist five pillars of AI, which is of course subject to discussion and sometimes also to dispute.

2.1 Machine learning

On the coarsest level algorithms in machine learning (ML) can be divided into two classes – supervised and unsupervised – depending on whether the respective algorithm requires the specification of a target variable or not.

2.1.1 Supervised learning algorithms

Supervised learning techniques require, in addition to the input variables (predictors), the known target values (labels) of a problem. To train a ML model on the identification of road signs on cameras, vectorized road signs are required as input variables images of, ideally in different configurations. Lighting, angle, pollution, etc. are summarized as noise or blur in the data, nonetheless a street sign in the rain should be equally recognized as good in the sunshine. The labels (the correct names) are typically assigned manually. This correct set of input variables and their correct classification account for a training data set. Although we only have one image per training data set in this case, we are talking about several input variables, since ML algorithms find relevant properties (features) in the training data and learn the relationship between these features and the class assignment for the example cited classification task. Supervised learning is mainly used to predict numeric values (Regression) and classification (prediction of class membership), where the data are not limited to a specific format; the processing of images, audio files, videos, numeric data and text provides for ML algorithms no problem. Some examples of classification are about object recognition (street signs, object ahead of the vehicle, etc.), face detection, credit risk assessment, voice recognition, customer churn to just to name a few.

Examples for regression are the determination of continuous, numerical values on several (sometimes hundreds or thousands) input variables, such as the optimum speed based on road and environmental conditions in an autonomous series vehicle, the determination of a financial measure such as gross domestic product based on a variable number of input variables (using agricultural land, education level of a population, industrial production, etc.) or the identification of potential market share by introducing new models. Each of these issues is highly complex and cannot be expressed by simple, linear relationships in simple equations; it may also happen that the expertise does not exist.

2.1.2 Unsupervised Learning Algorithms

Unsupervised learning techniques single out any individual target variable, but it is the aim to achieve a general characterization of an instance. Frequently, unsupervised ML algorithms are

used to group records (clusters), so relationships between individual data points are found, which can be composed of an arbitrarily high number of attributes, and merged into groups (clusters). In some cases, the output of unsupervised ML algorithms can be used again as an input for supervised learning algorithms. Examples of unsupervised learning are about the formation of groups of customers based on purchase behavior or demographics, or time series clustering in order to group millions of sensor time series into not previously obvious groups.

Machine Learning (ML) is therefore the part of the artificial intelligence (AI), in which computers are given the ability to learn from data without being explicitly programmed. In ML, the focus is on the development of programs, which are capable of teaching themselves in order to grow and change as they are provided with new data. Processes that can be mapped as a flow chart are therefore are not candidates for machine learning – everything requiring dynamic, changing strategies and that cannot be limited to static rules, is potentially suitable to be solved by means of ML. It is applied when

– human expertise does not exist,
– people cannot express their expertise,
– the solution changes over time,
– the solution has to be adapted to specific cases.

In contrast to the statistics, which is used to track the inference from a sample, it is of interest to develop efficient algorithms for solving optimization problems and a representation of the model for evaluation of inference in computer science. Commonly used methods for optimization known in this context are evolutionary algorithms (genetic algorithms, evolution strategies), whose basic principles are based on the natural evolution.[30] These methods are very effective in the application to complex non-linear optimization problems.

ML is not the same as data mining, although ML is generally applied in this field – the goal of both is to analyze data in order to find patterns. Rather than to extract data for human understanding as is the case in the data mining, ML method can be used to improve one's understanding of a program on the data provided. Software implementing ML methods detects patterns in data, and can adapt its behavior based on what it finds. When talking about autonomously acting vehicles (or the software that interprets visual signals from divers input channels, i.e. a camera), an execution block, consisting of the output of the AI algorithm and logic on this output, initiating a braking maneuver in case of suddenly appearing passersby in front of the vehicle needs to function both with small, large, thick, thin, disguised, coming from the left, coming from the right, etc. people, which is called generalization. The term generalization is used here, as the ML-approach must be able to generalize on all persons from the limited amount of training examples provided. It must not, on the other hand, be too generalizing, as the vehicle should not slow down at a stationary dustbin on the roadside.

The complexity in the world will often be greater than the complexity of a ML-model which is tried in most cases, to divide into sub-problems and problems apply ML models on these sub-problems. The output of these models is then integrated to allow complex tasks such as autonomous acting of vehicles in structured and unstructured environments.

30 Bäck T., Fogel D.B., Michalewicz Z. (1997): Handbook of Evolutionary Computation, Institute of Physics Publishing, New York

2.2 Computer Vision

Computer Vision (CV) is a very broad field of research in which scientific theories from different fields (as so often in the AI) are merged, ranging from biology, neuroscience and psychology to computer science, mathematics and physics. To start with it is important to understand how an image is physically created. Before light impinges on a two-dimensional array of sensors, it is refracted, absorbed, scattered or reflected, and an image is formed through the measurement of the intensity of the light beams through each element in the image (pixels). The three basic perspectives on CV are:

- The reconstruction of a scene and the point, from which the scene is observed, based on an image, an image sequence or a film.
- The imitation of biological, visual perception, in order to better understand the physical and biological processes are involved, such as how the wetware works, and how the interpretation and understanding of what has been observed function.
- In technical research and development, the focus is on efficient algorithmic solutions – in CV software often problem-specific solutions are developed, which have only limited overlaps with the optical perception in biological organisms.

All three have similarities and influence each other. If emphasis is on obstacle detection in order to initiate an automated braking maneuver ahead of the vehicle as a passersby appears, it is primarily important to recognize the passer as an obstacle; an interpretation of the entire scene, such as to understand that the vehicle moves toward a family having a picnic in a meadow, is not necessary required in this case. In contrast, the understanding of a scene is then a requirement if context is a relevant input (or output), such as the development of robotic household assistants, who need to understand very well that a person lying on the floor is not only a sleeping obstacle, but is in all likelihood a medical emergency.

Visual perception in a biological organism is understood as an active process, which includes the control of the sensor and is closely linked to the successful completion of the action.[31] As a result,[32] CV-systems are mostly not passive, that is, the system has to

- be continuously supplied with data via sensors (streaming), and
- act on this data stream.

Apart from that the goal of CV systems is not the understanding of scenes in images – it has to primarily extract the relevant information for a specific task from the scene. Rather, a "region of interest" must be determined, which is used for processing. Finally, the system must have a short response time, because it is likely that a scene is changing over time and a delayed action may not result in the desired effect. For object recognition ("what" is "where" in a scene), many different methods have been proposed, including:

- Object detectors, which move a window over the image and determine a response for each position of any such filter by matching template and sub-picture (box contents). Each new object parameterization requires a separate scan. More sophisticated algorithms

31 Bajcsy R. (1988): Active perception, Proceedings of the IEEE, 76:996-1005

32 Crowley J. L., Christensen H. I. (1995): Vision as a Process: Basic Research on Computer Vision Systems, Berlin: Springer

expect the same at different scales and apply filters that have been learned from a large number of images.
– Segment based techniques extract a geometric description of an object by grouping of pixels that define the extent of an object in an image. Based on this, an immutable feature set is calculated, that is, the features contained therein retain the same values under various image transformations such as changes in the light conditions, scaling or rotation. These features are used to identify objects or object classes clearly. An example is the previously mentioned identification of road signs.
– Orientation-based methods use parametric object models that are trained on data.[33,34] Algorithms search for parameters such as scaling, translation or rotation, which optimally fit a model on corresponding features in the image. An approximate solution can be found through a reciprocal process, i.e. by features such as contours, corners, or other characteristic points in the image "voting" for parameter solutions that are compatible with the feature found.

For conducting object recognition, it must first be decided whether algorithms are working on 2D- or 3D-representations of objects, whereby 2D-representations are often a good compromise between accuracy and availability. Current research in deep learning artificial neural networks shows that even distances between two points based on two 2D images, taken from different points, can be determined accurately. In daylight and reasonably good visibility this input can be used in addition to data from laser and radar in order to increase the accuracy – a mono-vision camera is sufficient to generate the necessary data. Unlike 3D-objects 2D-images do not encode any information such as shape, depth or orientation directly. Depth coding can be done in many ways, such as with the aid of laser or stereo cameras (like human perception) and structured light approaches (like Kinect from Microsoft). Currently, the most intensively pursued research direction relates super squares – geometric shapes defined by formulas that use any exponent to identify structures such as cylinders, cubes and cones with rounded or sharp corners. Using a small set of parameters, a wide variety of different base forms can be described. If 3D-images are determined by means of stereo cameras, due to poorer data quality compared with laser scans, statistical methods (such as generating a stereo point cloud) are applied instead of the previously mentioned form methods. Further research is done regarding tracking,[35,36] contextual scene understanding,[37,38] and surveillance[39], but these are currently for the automotive industry is of secondary relevance.

---

33 D. P. Huttenlocher, S. Ulman: Recognizing solid objects by alignment with an image, International Journal of Computer Vision, 5: 195-212, 1990
34 K. Frankish, W. M. Ramsey: The Cambridge handbook of artificial intelligence, Cambridge: Cambridge University Press, 2014
35 Chaumette F., Hutchinson S. (2006): Visual servo control I: Basic approaches, IEEE Robotics and Automation Magazine, 13(4): 82-90
36 Dickmanns E. D. (1991): Dynamic Vision for Perception and Control of Motion, London: Springer, 2007
37 T. M. Straat, M. A. Fischler: Context-based vision: Recognizing objects using information from both 2D and 3D imagery, IEEE Transactions on Pattern Analysis and Machine Intelligence, 13: 1050-65
38 Hoiem D., Efros A. A., Hebert M. (2006): Putting objects in perspective, Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2137-44
39 Buxton H. (2003): Learning and understanding dynamic scene activity: A review, Vision Computing, 21: 125-36

2.3 Logic and reasoning

Known as “Knowlege Representation & Reasoning” (KRR) in the literature, the focus in this field this field of research is on the design and development of data structures and inference algorithms. Problems that are solved by reasoning, are frequently found in applications requiring interaction with the physical world (as humans), such as the making of diagnoses, planning, natural language processing, answering questions, etc. KRR forms the basis for human-level AI.

Reasoning in the field of KRR is where data-based answers have to be found without human intervention or assistance, whereby the data are usually presented in a formal system with clear and precise semantics. Since about 1980, it is assumed that the data involved are a mixture of simple and complex structures, where the former are in a low degree of computational complexity and provide the research basis for large databases. The latter are presented in a more expressive language, which requires less space for representation, and correspond with generalizations and fine-granular information.

Decision-making is a kind of reasoning, in which answering questions about preferences between activities is paramount, such as when an autonomously acting agent tries to perform a task for a human being. Very often this decision-making is conducted in a dynamic domain, which changes by the execution of actions and the lapse of time. One example is the autonomously acting vehicle that must react to changes in traffic.

Mathematical logic is the formal basis for many applications in the real world, such as the theory of computation, our legal system and related arguments, and theoretical developments and evidence regarding research and development. The initial vision was to represent any kind of knowledge in the form of logic and to reason with universal algorithms, but some challenges have arisen; not any kind of knowledge is easy to represent. In addition, it can be very complex to compile the required knowledge for complex applications, and in addition it is not easy to learn this knowledge in from an expressive, logical language.[40] Moreover, it is not easy to reason with the required, expressive language – in the extreme case, such a scenario is computationally not feasible, even if the first two challenges can be solved. Currently three debates are conducted in this regard:

- First there is the statement that logic cannot represent many concepts, such as space, analogy, shape, uncertainty, etc., and therefore they cannot as an active part in the development of AI on human level are counted. The counter argument is that logic is just one of many tools. Currently, the combination of representative expressiveness, flexibility and clarity with any other method or other system is achieved.
- The second debate revolves around the statement that logic is too slow for inference and will therefore never play a role in a production system. The counterargument here is that ways for approximation of inference by logic exist, so that processing is approaching time limits and progress on logical inference is thus made.
- The third debate revolves around the argument that it is extremely difficult or even impossible to develop systems of logical axioms in real world applications. The counter-

40 Lavarac N., Dzeroski S. (1994): Inductive Logic Programming, vol. 3: Nonmonotonic Reasoning and Uncertain Reasoning, Oxford University Press: Oxford

arguments are mainly based on the research of those seeking techniques for learning logical axioms from natural language texts.

A distinction is made between four different types of logic[41] that are not discussed further here:

- Propositional logic
- First-order logic
- Modal logic
- Non-monotonic logic

At this point it is also required to mention automated decision making, i.e. autonomously acting robots (vehicles), automated agents in the internet, or mapping decision finding processes I logic in order to automate those. Very often, such a decision process considers the dynamics in the surrounding world, such as when a transport robot has to avoid another in a production plant. However, this is not a prerequisite, for example when a decision-making process is carried out without well-defined direction in the future, i.e., the decision to rent a warehouse at a specific price at a specific location. Decision-finding as research discipline spans multiple domains, such as computer science, psychology, economics and all engineering. Some fundamental issues need to be addressed for the development of automated decision making systems:

- Is the domain is dynamic in that a sequence of decisions is required, or is it static, so that a single or multiple simultaneous decisions must be made?
- Is the domain deterministic, non-deterministic or even stochastic?
- Should benefits be optimized or a goal be reached?
- Is the domain known to the full extent or only partially at any time?

Logical decision problems are non-stochastic in nature, which includes planning and conflictory behavior. Both require that the available information on the initial and intermediate states is complete, and that actions have only deterministic, known effects and that there is a specific target definition. These types of problems are found in the real world frequently, such as in the control of robots, logistics, complex behavior on the internet, and computer and network security.

In general, a planning problem involves an initial (known) situation, a target definition and a set of permitted actions or transitions between steps. The result of a planning process is a sequence or set of actions, the correct execution of which moves the performer from an initial state to a state that satisfies the goal conditions. Planning is a computationally difficult problem, even if simple problem specification languages are used. The search for a plan can, even with simpler problems, not traverse through the entire state space graph, because it is exponentially larger in the number of states defining the domain. Therefore, it is attempted to develop efficient algorithms mapping subgraphs in order to browse these in the hope of achieving the goal. Recent research has focused on the development of new search methods and new representations of actions and states that allow easier planning. Especially when taking into account one or more agents working against each other, it is important to find a

41 Frankish K., Ramsey W. M. (2014): The Cambridge handbook of artificial intelligence, Cambridge: Cambridge University Press

balance between learning and decision – exploration for the learning's sake, while decisions are taken, can lead to undesirable results.

Many problems in the real world are problems whose dynamic is of stochastic nature – purchasing a vehicle whose properties are unknown to us and affect its value is an example. These dependencies affect the purchase decision and it is therefore necessary to incorporate risks and uncertainties. Stochastic domains are practically more difficult with respect to decision-finding, but also more flexible with regard to approximations than deterministic domains – the simplification of practical assumptions also makes automated decision-finding practicable. There are lots of problem formulations which can represent different aspects and decision making processes in stochastic domains, under the most famous decision networks and Markov decision-making processes.

Many applications require the combination of logical (non-stochastic) and stochastic elements, such as the control of robots requiring high-level specifications in logic and low-level representations with respect to a probabilistic sensor model. Natural language processing is another area to which applies this assumption, since high-level knowledge in logic is combined with probabilistic low-level models of text and voice signals.

2.4 Language and communication

Processing of speech is considered as fundamental in AI, and two fields are distinguished: Computational Linguistics (CL) and natural language processing (NLP). In summary, the difference that in CL research the use of computers for language processing is conducted, and NLU consists of all applications, such as machine translation (MT), Q&A, document summary, information extraction, etc. NLP therefore requires a specific task and is not per se a research discipline. NLP includes:

- Part-of-Speech Tagging
- Natural language understanding
- Natural language generation
- Automated summarization
- Entity recognition
- Parsing
- Voice recognition
- Sentiment analysis
- Voice-, topic- and word-segmentation
- Co-reference resolution
- Discourse analysis
- Machine translation
- Word sense disambiguation
- Morphological segmentation
- Answers to questions
- Relationship extraction
- Sentence separation

The central vision of the AI states that a version of first-order predicate logic (first-order predicate calculus, FOPC), supported by appropriate to the problem mechanisms for the representation of language and knowledge is sufficient. This assumption states that logic can

and should provide the semantics underlying natural language. Although experiments in AI and linguistics to leverage a form of logical semantics as a key for the representation of content have progressed, such were crowned with little success concerning a program translating English into formal logic. Even in psychology, it has not yet been proven that such translations in logic corresponds to the way in which people store and manipulate "meaning". The translation from one language FOPC is therefore currently still a goal that has not yet been achieved. No doubt there are NLP applications must which establish logical inferences between sentence representations, but if these only account for a part of an application, it is not obvious that they have something to do with the underlying meaning of natural language (and thus with CL/ NLP), since the primary task of logical structures was inference. Different positions have emerged also in this area:

– Logical inferences are very closely linked to the meaning of sentences, because to know their meaning is the same as deriving inferences, and logic is the best way to do that.
– There is a meaning outside the logic that posits a number of semantic markers or primitives that are hung on the words to express their meaning. Today, this is commonly referred to as annotation.
– The predicates of logic and formal systems generally appear only different from human language, but their terms are in fact the words than they appear.

The introduction of statistical and AI methods is the latest trend in this regard in the field. The general strategy is to learn how language is processed, ideally in the way in which humans do, whereby this does not constitute a prerequisite. Regarding ML, this is learning based on very large corpora, which were translated manually by humans. This often means that it must be learned (algorithmically) how annotations are assigned, how coropora are provided with part-of-speech categories (allocation of words and punctuation of a text to speech), semantic markers or primitives, and all this based corpora prepared by humans (which are therefore correct). In supervised ML possible associations of part-of-speech tags with words that were annotated by humans on text can be learned so that the algorithms then can annotate new, hitherto unknown texts. [42] This works weakly supervised or unsupervised as well, i.e. when no annotations were made from humans and only a text in one language with identical content in another language is presented, or even relevant clusters are found in thesaurus data, without having defined a goal.[43]

Regarding AI and language information retrieval (IR) and information extraction (IE) play a major role, which very strongly correlate with each other. One of the main tasks of IR is grouping of texts based on their content, whereby IE extracts similar fact elements from texts, or is used to answer questions about text content. Therefore, these fields correlate with each other strongly, since individual records (not only long texts) can also be considered as documents. These methods are applied, i.e., in interactions of users with systems, such as when a driver asks the onboard computer a question about the operator manual while driving – after the linguistic input has been converted to text, based on the semantic content of the question an answer in the manual is searched, extracted and returned.

---

42 Leech G., Garside R., Bryant M. (1994): CLAWS4: The tagging of the British National Corpus. In Proceedings of the 15th International Conference on Computational Linguistics (COLING 94) Kyoto, Japan, pp. 622-628

43 Spärck Jones K. (1999): Information retrieval and artificial intelligence, Artificial Intelligence 141: 257-81

2.5 Agents and actions

In classical AI research mainly focused on single, isolated software systems that operated relatively inflexible on predefined rules. New technologies and applications have created the demand for artificial entities that are more flexible, adaptive and autonomous, and act as social units in multi-agent systems. In classical AI (see "Physical Symbol System Hypothesis"[44], which was embedded in so-called "deliberative" systems) an action theory, thus how systems make decisions and act, is represented logically in individual systems needing to perform actions. The system shall, based on these rules, prove a theorem – the requirements for this are that the system has

- a description of the world in which it is currently located,
- the desired target state and a set of actions,
- along with conditions for execution of this and
- a list with the results of each action gets.

It has been found that even in simple problems by the computational complexity each time-limited system is unusable, which had a great impact on symbolic AI, resulting in the development of reactive architectures. Such follow if-then rules converting inputs directly into tasks. Such systems are extremely simple, but can nevertheless solve very complex tasks. A problem is that such systems learn procedures, but not declarative knowledge, that is, they are learning attributes that simply cannot generalize to similar situations. There have been many attempts to combine deliberative and reactive systems, however, it seems that one must either focus on impractical, deliberative systems or on very loosely developed reactive systems – both are not optimal.

2.5.1 Principles of the new agent-centered approach

The agent-oriented approach can be characterized by the following principles:

- Autonomous behavior: Autonomy refers to the ability of systems to make their own decisions and carry out tasks on behalf of the system designer. The aim is to let a system act autonomously in scenarios in which it is difficult to control it directly. Traditional software systems perform methods after they have been called – they have no choice; Agents, however opt for their views, desires and intentions (Belief, Desire, Intention – BDI).[45]
- Adaptive behavior: Since it is impossible to predict all situations which agents will encounter, they must be able to act flexibly. They must be able to learn from their environment and adapt. This task is more difficult, if not only nature is a source of uncertainty, but the agent is also located in a multi-agent system. Only environments that are not static and closed allow meaningful application scenarios for BDI agents, e.g. a lack of knowledge of the world may be compensated by reinforcement learning: agents are acting in an environment that is described by a set of possible states. Each time an

44 Newell A., Simon H. A.: Computer science as empirical enquiry: Symbols and search, Communications of the ACM 19:113-26

45 Bratman M., Israel D. J., Pollack M. E. (1988): Plans and resource-bounded practical reasoning, Computational Intelligence, 4: 156-72

agent performs an action, it will be "rewarded" with a numerical value expressing how good or bad was his action. This leads to a number of conditions, actions and rewards. The agent is now encouraged to identify an approach that maximizes reward.

– Social behavior: In an environment in which different entities act, it is necessary that agents recognize their opponents and form groups, provided a common goal requires them to do so. Agent-oriented systems are used for personalization of user interfaces, middleware and in competitions such as the RoboCup. In a scenario in which only autonomously vehicles are on the roads, not only autonomy, but also the exchange of information between vehicles and the thereupon based acting as group is an indispensable component. Through coordination between the agents the traffic flow is optimized – congestion and accidents are virtually impossible.

In summary, that the agent-oriented approach is accepted as a forward-looking direction in the AI community.

2.5.2 Multi-agent behavior

There are different approaches pursued to implement multi-agent behavior, which differ mainly in terms of how much control the designer on individual agent has.[46,47,48] Differentiation is made between

– distributed problem-solving systems (Distributed Problem Solving systems, DPS) and
– multi-agent systems (multi-agent system, MAS)

DPS allows the designer to control every individual agent in the domain in which the solution of the task is distributed to more than one agent. In MAS, on the contrary, there are several designers and each can only influence his own agent and has no access to the design of all other agents. In this case, the design of the interaction protocols is very important. In DPS, agents try to achieve a goal or solve a problem, whereas in MAS, each agent is individually motivated to reach its own target and wants to maximize his own benefit. The research in DPS has to achieve the goal of cooperation strategies for problem solving while minimizing the degree of communication. In MAS attention is paid to the coordinated interaction, thus how the autonomously acting agents can be made to find a common basis for communication and carry out uniform actions.[49] Ideally, a world that is only traversed by acting autonomously vehicles, a DPS, however, due to competition between the OEMs a MAS will most likely arise first. The communication and negotiations between agents are in the foreground (see Nash equilibrium).

---

46 Bond H. Ah., Gasser L. (1988): Readings in Distributed Artificial Intelligence, San Mateo, CA: Morgan Kaufmann

47 Durfee E. H. (1999): Coordination for Distributed Problem Solvers, Boston, MA: Kluwer Academic, 1988

48 Weiss G.: Multiagent Systems: A modern approach to distributed artificial intelligence, Cambridge, MA: MIT Press

49 Frankish K., Ramsey W. M. (2014): The Cambridge handbook of artificial intelligence, Cambridge: Cambridge University Press

2.5.3 *Multi-agent learning*

The multi-agent learning (MAL) is given a certain degree of attention only recently. Key issues are the definition of techniques that should be used and what exactly multi-agent learning is. Current ML approaches have been developed to train individual agents, whereas at MAL distributed learning is the target. Distributed does not necessarily mean that a neural network, in which many identical operations are run during exercise and this can therefore be parallelized, is executed, but

- that a problem is split into sub-problems and individual agents learn these in order to solve the main problem based on their combined knowledge, or that
- many agents independently try to solve the same problem by competing against each other.

Reinforcement Learning is an approach which takes in this respect application.[50]

2.6 Summary

Artificial intelligence has already found its way into our daily lives and is no longer exclusively the subject of science fiction novels. At present, AI is mainly used in the following areas:

- Analytical data processing
- In domains in which quick, qualified decisions must be made on a large amount of (often heterogeneous) data
- At monotonous, but attention-requiring activities

In analytical data processing, over the next years we will no longer rely on decision support systems, but on systems that relieve us of decisions. Specifically, in data analysis we are currently developing individual, analytical solutions for specific problem statements, but those cannot be used across contexts – for example, a solution developed for anomaly detection in stock value behavior cannot be used to understand content of images. Also in the future, this will not be the case, but AI systems will integrate individual, interacting components and will so take over more complex tasks currently reserved for humans – a clear trend we can already observe now. A system not only processing the latest data from stock markets, but also analyzing the development of political structures on news texts or videos, extracting sentiments from from blogs or social networks, monitoring and predicting relevant financial ratios, etc., requires the integration of many different subcomponents – encouraging them to work together is the subject of current research, and progress is published weekly. In a world in which AI systems are able to improve themselves continuously and to control companies more effectively than humans, what remains there for the humans? Time to focus on expanding one's knowledge, the improvement of society, the eradication of hunger, elimination of diseases, propagation of our species over the boundaries of our own solar system. Some theories suggest that quantum computers are needed for the development of strong AI systems, and only someone who is very careless, would argue an effective quantum computer would be available within the next 10 years. And only someone who is careless

50 Busoniu L., Babuska R., De Schutter B. (2008): A comprehensive survey of multi-agent reinforcement learning, IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews 38: 156-72

would argue that that is not the case. Be that as it may, as with the majority of relevant, scientific achievements history shows that artificial intelligence must be used wisely – systems that can take exponentially more decisions in extremely short periods of time with increasing hardware performance, can cause many positive things, but also have the potential to be abused.

3 An outline of artificial neural networks

Here we will have a closer look at some of the fundamental concepts and current standards in the field of computational intelligence (CI), which are particularized under the consideration of actual knowledge and research conducted. The focus is on artificial neural networks, as I consider them, together with Markov models, as one of the most valuable means for developing learning software. Artificial neural networks are not only applicable within the field of data mining (see 6.1 Analysis), but can make a software system capable of understanding and solving a presented problem statement through learning. (Hierarchically hidden) Markov models are explained at 10.6 The artificial neocortex, where I also explain one of the most important capabilities of an intelligently acting system, which is the processing of natural language (and language understanding, which I lump together here), and how patterns are hierarchically organized within the human brain. This is, because in artificial systems both concepts may be implemented through Markov models. We will further see that within the field of artificial intelligence the imitation of human brain functionality has been researched on for a long time, but although the achievements have been impressive, software mimicking such approaches is still restricted to doing things at which computers have always been better in than humans, such as the detection of patterns in huge amounts of unstructured data or time-series prediction (the prediction of continuous values). Computational intelligence is a subfield of AI, and further AI paradigms relevant for this elaboration will be discussed at 10 Reverse engineering the mind as well.

3.1 Definition

As already discussed in chapter 2 Pillars of artificial intelligence AI as a whole tries to make systems behave like humans do, whereas computational intelligence relies on evolutionary approaches to solve, amongst others, problems suitable for computers, like detecting similarities in huge amounts of data or optimization problems. Within the field of AI robotics, CI approaches find application for ensuring robust control, planning, and decision making.[51,52,53,54,55]

CI techniques have experienced tremendous theoretical growth over the past few decades and have also earned popularity in application areas e.g. control, search and optimization, data mining, knowledge representation, signal processing, and robotics.[56]

---

51 Liu Dikai, Wang Lingfeng, Tan Kay Chen (2009): Design and Control of Intelligent Robotic Systems; Berlin Heidelberg: Springer-Verlag, p. 2

52 Nolfi Stefano, Floreano Dolfi (2000): Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines; Bradford Books

53 Jain Lakhmi C. (1998): Soft Computing for Intelligent Robotic Systems: Physica-Verlag

54 Watanabe Keigo, Hashem M.M.A. (2004): Evolutionary Computations: New Algorithms and Their Applications to Evolutionary Robotics; Heidelberg: Springer-Verlag

55 Teshnehlab M., Watanabe K. (1999): Intelligent Control Based on Flexible Neural Networks (Intelligent Systems, Control and Automation: Science and Engineering); Dordrecht: Kluwer Academic Publishers

56 Liu Dikai, Wang Lingfeng, Tan Kay Chen (2009): Design and Control of Intelligent Robotic Systems; Berlin Heidelberg: Springer-Verlag, p. 2

F. Neukart, *Reverse Engineering the Mind*, AutoUni –
Schriftenreihe 94, DOI 10.1007/978-3-658-16176-7_3

However, when talking about the application of CI-paradigms in the further chapters, it has to be clearly defined which meaning the umbrella term CI bears in the field of DM at first. The term itself is highly controversial in research and used in different manners. Generally, Fulcher et al. and Karplus provide widely accepted definitions, which are also suitable within the context of this elaboration:

- Nature-inspired method(s) + real-world (training) data = computational intelligence.[57]
- CI substitutes intensive computation for insight into how the system works. Neural networks, fuzzy systems and evolutionary computation were all shunned by classical system and control theorists. CI umbrellas and unifies these and other revolutionary methods.[58]
- Another definition of computational intelligence emphasizes the ability to learn, to deal with new situations, and to reason.[59]

CI is all of the above mentioned, and even more. Combined with an agent-oriented development approach, in other words by encapsulating a software system's fields of CI-functions to single entities working together to reach a superior target, leads to a completely different understanding of what software is. A CI-software system not only consists of static business rules, it comprises the ability to adapt to a problem, to learn and to behave. Behavior is important in this context, as it a more appropriate term for the functionality of an evolutionary system, e.g. ANNs can indeed be developed by a person, but the same person does not know what exactly happens within an ANN learning or solving a problem statement. Of course, the developer knows theoretically what happens behind ANN learning, but as such make use of random variables intensively and repeated training with the same data produces slightly different results it is more suitable to talk about behavior instead of functionality. From the above mentioned information it becomes obvious that computational intelligence comprises

- artificial neural networks,
- fuzzy logic and
- evolutionary computation

but also covers

- granular computing,[60]
- probabilistic reasoning,
- Bayesian (belief) networks,[61]
- fuzzy Petri nets,
- constrained reasoning,

57 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 38

58 Karplus W. (1998) cited in: Kaynak Okyay, Zadeh Lofti A., Türksen Burhan (1998): Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications; Berlin: Springer-Verlag

59 Eberhart Russel C., Simpson Patrick K., Dobbins Roy (1996): Computational Intelligence PC Tools; Boston MA: Academic Press

60 Lin Y. T. (1999): Granular computing: fuzzy logic and rough sets. In: Zadeh LA, Kacprzyk J. (eds.): Computing with Words in Information/Intelligent Systems; Springer-Verlag: Berlin

61 Jensen Finn V., Nielsen Thomas Dyhre (2001): Bayesian Networks and Decision Graphs; Berlin: Springer-Verlag

– case-based reasoning,[62]
– support vector machines,[63]
– rough sets,[64]
– learning/adaptive classifiers,
– fractals,[65]
– wavelets,[66]
– chaos theory,[67] as well as
– intelligent agents.[68]

Thus, computational intelligence researchers pursue the target to empower a machine to learn in a data-driven and non-algorithmic way. Summing up, computational intelligence is the study of adaptive mechanisms to enable or facilitate intelligent behavior in complex, uncertain and changing environments.[69]

3.2 Paradigms of computational intelligence

As indicated beforehand, computational intelligence makes use of evolutionary approaches, e.g. the imitation of human brain functions by the implementation of ANNs for classification, or the imitation of insect swarms (swarm intelligence)[70] for solving optimization problems. The IEEE Computational Intelligence Society defines the imitation of evolutionary approaches as 'mimicking nature for problem solving'. Natural processes CI makes use of are[71,72]

– evolutional paradigms and genetic selection (phylogeny): evolutionary approaches
– multi-cellular organisms (embryology; ontogeny): cellular automata[73]

62 Watson Ian (1997): Applying Case-Based Reasoning: Techniques for Enterprise Systems; San Francisco: Morgan Kaufmann
63 Shawe-Taylor Cristianni N. (2000): Support Vector Machines and Other Kernel-based Learning Methods; UK: Cambridge University Press
64 Inuiguchi Masahiro, Hirano Shoji, Tsumoto Shusaku (2003): Rough Set Theory and Granular Computing; Berlin: Springer-Verlag
65 Falconer Kenneth (2003): Fractal Geometry: Mathematical Foundations and Applications; New York: Wiley
66 Mallat Stephane (1999): A Wavelet Tour of Signal Processing; Boston MA: Academic Press
67 Ott Edward (2002): Chaos in Dynamical Systems, UK: Cambridge University Press
Padgham Lin, Winikoff Michael (2004): Developing Intelligent Agent Systems: A Practical Guide to Designing, Building, Implementing and Testing Agent Systems (Wiley Series in Agent Technology); New York: Wiley
69 Onwubolu, Godfrey C. (2009): Hybrid Self-Organizing Modelling Systems Berlin Heidelberg: Springer-Verlag, p. 13
70 Bertelle Cyrille, Duchamp Gérard H. E., Kadri-Dahmani Hakima (2009): Complex Systems and Self-organization Modelling; Berlin Heidelberg: Springer-Verlag
71 Sipper M., Sanchez E., Mange D., Tomassini M., Perez-Uribe A., Stauffer A. (1997): A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems; IEEE Trans. Evolutionary Computation, 1(1): 83–97
72 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 17
73 Lohn I. D., Reggia J. A. (1997): Automatic discovery of self-replicating structures in cellular automata; IEEE Trans. Evolutionary Computation, 1(3): 165–178

- biological brains, cerebral cortex, the nervous, immune and endocrine systems (epigenesis): ANNs; immunity-based systems[74]
- social organisms, insect swarms, bird flocks: swarm intelligence
- artificial life[75]

The difficulty in developing software systems that behave similar to human beings is to map probability, fuzziness and imprecision on systems that initially have been developed for being deterministic, sharp and precise.

Generally, two fundamental approaches to CI can be characterized, namely the left- versus the right-brained; the former refers to the traditional logical, rule-based, model-driven algorithmic approach, while the latter (data-driven) connectionist approach mimics the intuitive/creative/artistic side of our brains.[76] The further chapters focus on the right-brained approach.

3.3 Neural networks

The human brain has been considered to be one of the most complex natural structures within the yet discovered universe, and it is still capable of fulfilling cognitive processes that top the possibilities of modern computers. Especially learning, saving of information, recognition and perception, adaption to changing situations and the environment, behavior and creativity are the major areas in which computers fail in direct comparison. The operating principle of the human brain differs from the classical von Neumann-architecture in massive parallelism – namely neuronal activity. Artificial neural networks try to imitate the structure and behavior of the human brain and can, to some extent, accomplish astounding achievements.

3.3.1 Artificial neural networks

As indicated, artificial neural networks[77,78,79] seek the imitation of a human brains' functionality. Like a human brain, an artificial neural network is made up of neurons – current ones do not contain as much of these as a human brain does, but future approaches in quantum computing might target the simulation of a brain with a similar count of neurons. From the machine learning point of view, ANNs are capable of making non-linear mapping from a set of inputs to a set of outputs and therefore can be employed as universal functional approximators, which can offer accurate approximation of an unknown system on provision

74 Mange D., Tomassin M. (1998): Bio-Inspired Computing Machines. Presses, Laussanne: Polytechniques et Universitaries Romandes
75 Levy Steven (1997): Artificial Life: A Report From the Frontier: Where Computers Meet Biology; New York: Vintage Books
76 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 17
77 Kung Sun.Y. (1993): Digital Neural Networks. Englewood Cliffs: PTR Prentice Hall
78 Lau Clifford (1991): Neural networks, theoretical foundations and analysis; Los Alamitos: IEEE Press
79 Philippides A. et al. (1999): Diffusible neuromodulation in real and artificial neural networks; In: AI Symposium, Second International Conference on Cybernetics, Applied Mathematics and Physics: CIMAF 1999: Editorial Academia

of data samples.[80] The McCulloch & Pitts neuron model (Figure 20 - McCulloch & Pitts neuron model) shows the similarity of an artificial neuron to a human one.

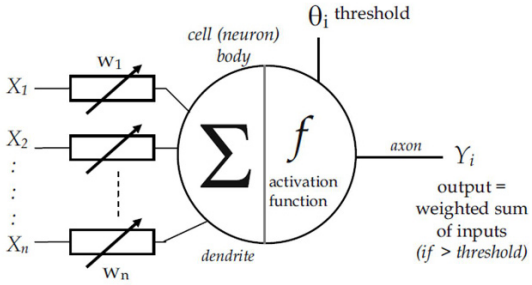

Figure 20 - McCulloch & Pitts neuron model[81]

The cell body receives its input from other neurons or, in case the neuron resides in the input layer, directly from a data source. The input is then multiplied by the weight of the connection to the target neuron, in other words passed over the dendrites, and then the results of all input neurons are summed up. This sum is compared to the threshold value in the nucleus and, if it is exceeded, passed over into the activation function, which then transfers the scaled (an activation function scales the output properly [3.3.1.3 Activation functions]) output of the inputs to the neuron(s) it is connected to. Figure 21 - Representative processing model provides a more mathematical description of the neuron, making it easier to understand the mathematical base processes behind an artificial neural network (Figure 21 - Representative processing model).

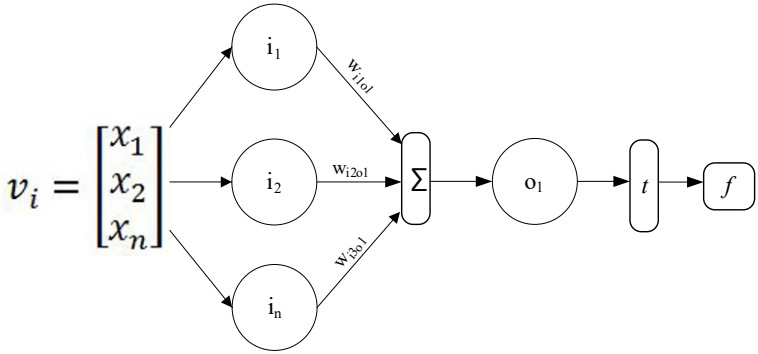

Figure 21 - Representative processing model

80 Jain Lakhmi C. (2008): Computational Intelligence Paradigms: Innovative Applications; Berlin Heidelberg: Springer-Verlag, p. 3

81 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 26

Figure 21 - Representative processing model also shows that the inputs $x_1, \ldots, x_n$ are multiplied by their weights and then summed up, compared with the threshold value t which leads, by an exceedance, to a firing neuron and by a lower deviation to a neuron that does not fire. The weighted sum is not directly passed over to the target neuron(s), but to the activation function, for scaling it properly. In this latter example, the threshold is represented by a bias, in other words the constant value 1.

Artificial neural networks, which are believed to be robust and adaptive, have been used to solve manifold problems, including those that fall into the NP-complete[82] class.[83]

Generally, two approaches for designing an ANN are possible; the evolutionary and the non-evolutionary approach: in the evolutionary approach, an ANN can be evolved by means of an evolutionary technique, e.g. a population-based stochastic search strategy such as a genetic algorithm (3.3.3.7 Genetic learning).[84,85,86] In the non-evolutionary approach, the ANN is built not as the result of an evolutionary process, but rather as the result of a specific algorithm designed to automatically construct it, as is the case with a constructive algorithm.[87]

3.3.1.1 Suitable problems

Not every problem is suitable for being processed with an artificial neural network. If a program works according to an easily written out flow chart, it is not suitable for being solved by an ANN.[88] Programs that should heavily make use of static business rules, and which must be fully predictable, are therefore also not suitable. The major advantages of an ANN, compared to a static program, are its abilities to learn, to be fuzzy if necessary and to adapt to a changing problem statement. When the above mentioned points come true and furthermore, the solution of a problem statement comes to the fore and not the solution process, then the application of an ANN might be the proper approach. Artificial neural networks can perfectly be applied on problem statements that cannot be solved step-by-step, such as recognizing patterns, classification, series prediction and finally, data mining.[89] In data mining, classification and clustering are problems of utmost suitability for presenting them to ANNs, especially when dealing with huge amounts of data and attribute-rich datasets.

---

82 Harel David (2004): Algorithmics: The Spirit of Computing, 3rd ed.; Amsterdam: Addison-Wesley

83 Andreas Tolk (2009): Complex Systems in Knowledge-based Environments: Theory, Models and Applications; Berlin Heidelberg: Springer-Verlag, p. 115

84 Franco Leonardo, José M. Jerez (2009): Constructive Neural Networks; Berlin Heidelberg: Springer-Verlag, p. 2

85 Schaffer J. D. et al. (1992): Combinations of genetic algorithms and neural networks: a survey of the state of the art. In: Proceedings of the International Workshop of Genetic Algorithms and Neural Networks, pp. 1–37

86 Yao X. (1999): Evolving neural networks; Proceedings of the IEEE 87(9), 1423–1447

87 Franco Leonardo, José M. Jerez (2009): Constructive Neural Networks; Berlin Heidelberg: Springer-Verlag, p. 2

88 Heaton Jeff (2008): Introduction to Neural Networks for Java, 2nd ed.; Chesterfield: Heaton Research, Inc., p. 43

89 Heaton Jeff (2008): Introduction to Neural Networks for Java, 2nd ed.; Chesterfield: Heaton Research, Inc., p. 43

3.3.1.2 Basic knowledge

The following explanations provide an overview of some basics that necessarily have to be understood to understand the functioning of ANNs, as they are applied by all them, especially within S/MLPs.

3.3.1.2.1 Structure

The structure of ANNs may vary, from one to many layers, the number of neurons or the direction of processing. Figure 20 - McCulloch & Pitts neuron model describes the structure of a neuron and how it receives input and transfers output. This indicates that an ANN usually does not consist of just one single neuron, except when dealing with a single layer perceptron (3.3.2.6.1 Single layer perceptron) containing just a single neuron. The internal structure therefore depends on the type and learning method of the ANN (3.3.1.4 Regularization).

3.3.1.2.2 Bias

A bias neuron itself does not receive an input, as its activity level is always +1. The weight of a bias to another neuron can either be positive or negative, depending on its usage. If the input from other neurons is fairly strong, the bias ensures that the neuron stays active at positive weight. If the bias weight is negative, the bias ensures that the unit stays inactive. This is useful in special cases where a threshold is needed that at first has to be exceeded by other neurons (negative weight) or when a neuron is expected to fire very often (positive weight). This mentioned threshold is, on the contrary to other to the threshold of an activation function, modifiable, as the weight between a bias and receiving unit is also modifiable through learning.[90]

3.3.1.2.3 Gradient descent

The gradient as a first-order optimization algorithm is especially relevant for propagation learning (3.3.3.6 Propagation learning), as it is used for modifying the weights of the neuron connections within an artificial neural network. The gradient descent has the advantage that not the whole hyper plane of the n-dimensional space all neuron weights together form must be known. The weights of every ANN form a hyper plane, as long as the ANN features more than one weight. For an ANN with only one weight, a 2-dimensional space for representing the gradient descent's curve suffices.

Typical for an ANN the gradient descent proceeding initially uses a random combination of weights for determining the gradient. The learning rate, a constant which influences the rate at which the weights of a neuron connection are adapted, is then used on the gradient to determine the descent. The descent, in this case, is the corresponding adaption of the weights. The gradient is defined as function of a scalar field, which indicates the rate of change (learning rate) and the direction of the largest change in terms of a vector field.[91]

---

90 Rey Günter D.: Neuronale Netze: Eine Einführung [2011-25-08]; Rey Günter D.; URL: http://www.neuronalesnetz.de/aktivitaet.html

91 Rey Günter D.: Neuronale Netze: Eine Einführung [2011-31-08]; Rey Günter D.; URL: http://www.neuronalesnetz.de/back-propagation3.html

For the ANNs discussed within this elaboration, as an error function either the mean squared error or the root mean squared error have been applied (explained in detail at 3.3.3.2 (Root) mean squared error), which is the function that has to be minimized. The gradient descent derivation and the calculation of the gradient descent itself are detailed in 3.3.3.6.1 Back propagation training.

3.3.1.3 Activation functions

As Figure 20 - McCulloch & Pitts neuron model and the latter figures show, the summed output is sent to an activation function, when the threshold is exceeded. The activation function serves the purpose of scaling the presented output of a neuron layer into a proper range for further processing within an artificial neural network. The activation function is not restricted to a specific one, but can take various forms. Common examples for activation functions are the linear activation function with or without bias, the binary threshold function or sigmoid functions.

3.3.1.3.1 Linear activation function

The linear activation function is the simplest one, as it does not modify the input values at all. When the entire range of the input values needs to be represented, then this function is of use, but in any other case a slightly more complex function allows more complex processing. For example, the XOR problem cannot be solved with a linear activation function (Figure 22 - Linear activation).

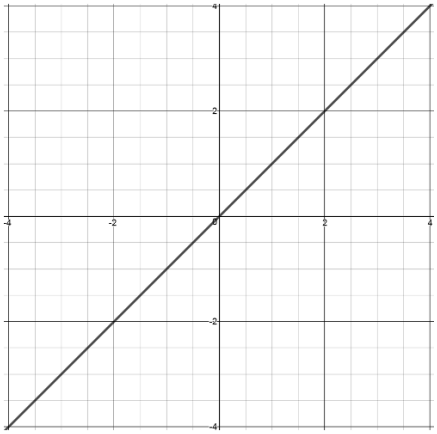

Figure 22 - Linear activation

The linear activation function is represented by the equation

$$f(x) = x \qquad (3\text{-}1)$$

3.3.1.3.2 Sigmoid activation function

Sigmoid functions are not really complex functions, but perfectly serve the purpose of scaling the input in an ANN. The following sigmoid function is a basic example for an ANN just having to return positive values (Figure 23 - Sigmoid activation).

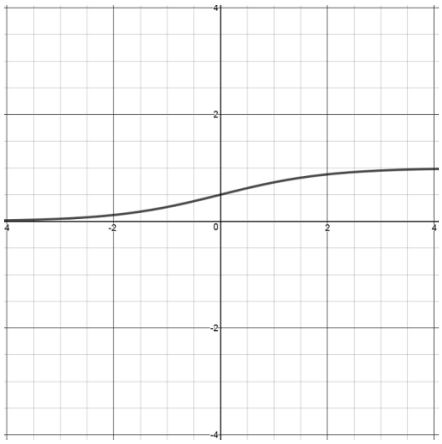

Figure 23 - Sigmoid activation

Its equation is

$$f(x) = \frac{1}{1+e^{-x}} \qquad (3\text{-}2)$$

If an ANN needs to return only values between 0 and 1, but not below 0, this function fits best.

3.3.1.3.3 Hyperbolic tangent activation function

If, however, the output has to be scaled between positive and negative values, e.g. between -1 and 1 as it is common when applying ANNs, the hyperbolic tangent function performs well (Figure 24 - Tangens hyperbolicus activation).

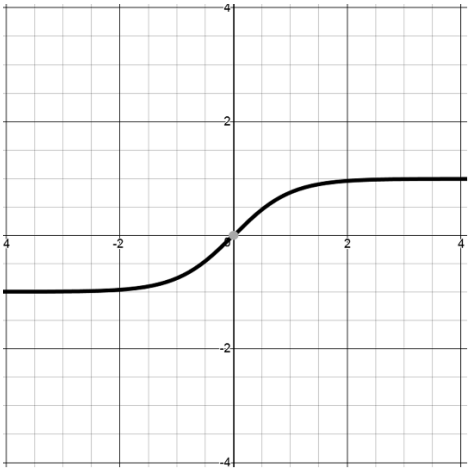

Figure 24 - Tangens hyperbolicus activation

Its equation is

$$f(x) = \frac{e^{2x}-1}{e^{2x}+1} \qquad (3\text{-}3)$$

3.3.1.3.4 Rectifier linear unit

The rectifier linear unit (ReLu) activation function has shown to result in ANNs with remarkable performance, especially in deep structures such as convolutional ANNs.

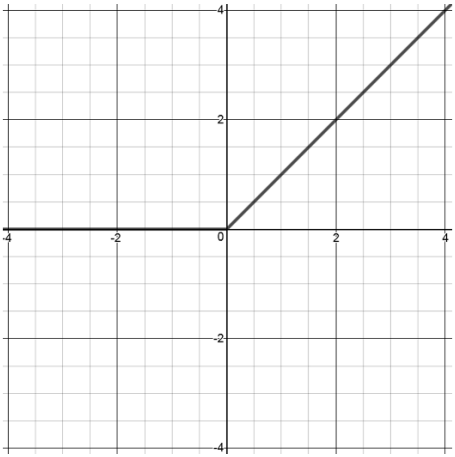

Figure 25 - ReLu activation

Its equation is

$$f(x) = max(0, x) \tag{3-4}$$

As of now, the ReLu function is the most popular activation function. It has been argued to be biologically more plausible than the sigmoid activation.

3.3.1.3.5 Gaussian activation function

The last activation function of relevance for this elaboration is a radial basis one, the Gauss function, producing the familiar curve (Figure 26 - Gauss activation).

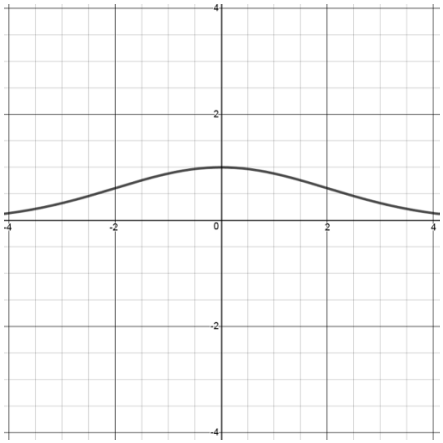

Figure 26 - Gauss activation

Its equation is

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}} \tag{3-5}$$

a being the curve's peak, b representing the curve's position, and c being the width of the curve. The Gaussian activation function can be applied when finer control over the activation range is needed.[92]

Nearly any activation function can be applied within artificial neural networks, also such ones as logarithmic sinus functions. It is of utmost importance for a very often applied training approach within multi-layer perceptrons, abstracted by the umbrella term 'propagation training' (3.3.3.6 Propagation learning), that the activation function is derivative. For calculating the error within propagation training, the actual output of the neural network is

92 Heaton Jeff (2010): Programming Neural Networks with Encog 2 in Java; Chesterfield: Heaton Research, Inc., p. 90

applied to the derivative of the activation function used,[93] which requires the function to be derivative. The last function mentioned here is not used within the prototype, but of use for further explanations, as it is easy understandable. It is the bipolar activation[94] function, algebraically represented by

$$f(s) = \begin{cases} 1, if\ s > 0, \\ -1, if\ s \leq 0 \end{cases} \quad (3\text{-}6)$$

A bipolar function scales the weighted and summed output of a neuron to either -1 or 1.

3.3.1.4 Regularization

Regularization is required due to the number of weights deep neural networks tend to overfit. Several approaches exist:

- L2 regularization: the squared magnitude of all parameters is directly penalized in the objective, thus to every weight ω the term $\frac{1}{2}\lambda\omega^2$ is added, whereby λ represents the regularization strength, the factor $\frac{1}{2}$ is added because this term's gradient with respect to ω then results in $\lambda\omega$ instead of $2\lambda\omega$. This results in penalizing high weight vectors heavily. Weights and inputs are multiplied, and another effect of L2 regularization is that the network uses all of its inputs slightly instead of just some inputs a lot. Finally, every weight is decayed linearly during gradient descent.

  $$w_i(t+1) = w_i(t) - \lambda w_i \quad (3\text{-}7)$$

- L1 regularization: the term $\lambda|\omega|$ is added to the objective, which results in many weight vectors evolving towards 0. This results in the effect that the neurons use only a very sparse subset of their inputs, namely the ones contributing most to the output. The rest of the inputs is considered to be noise and disregarded. L1 regularization is also frequently used in feature selection. A regularization technique called elastic net regularization combines L1 and L2 regularization: $\lambda_1|\omega| + \lambda_2\omega^2$.
- Dropout:[95] dropout is one of the most simple, but also most effective dropout methods and is usually used in combination with the others mentioned. The idea is to sample an ANN, thus keep a neuron and its weights active with probability p.
- Max norm constraints: this refers to apply constraints to the updates. The update rule for the weights is applied as usual, but additionally an upper bound, the constraint c, is enforced on the weight vector of every neuron $\vec{w}$, so that it satisfies $\|\vec{w}\| < c$. The exploding gradient-problem can never occur in an ANN making use of max norm constraint regularization.

---

93 Heaton Jeff (2010): Introduction to Encog 2.5 for Java, Rev. 3; Chesterfield: Heaton Research, Inc., p. 87

94 Heaton Jeff (2010): Introduction to Encog 2.5 for Java, Rev. 3; Chesterfield: Heaton Research, Inc., p. 87

95 Sirivastava Nitish, Hinton Geoffrey, Krizhevsky Alex, Sutskever Ilya, Salakhutdinov Ruslan (2014): Dropout: A Simple Way to Prevent Neural Networks from Overfitting; Journal of Machine Learning Research 15 (2014) 1929-1958

3.3.2 Types of artificial neural networks

The number of different types of ANNs is straightforward, but nonetheless the different types show complete differences in their structure and their cases of application. Furthermore, each type of ANN can be changed in terms of activation functions, data processing or its number of layers which, in some degree, can lead to the assumption that one has to do it with different subtypes.

Artificial neural networks may be classified by the following dimensions:

– Supervised versus unsupervised versus hybrid
– Feed-forward versus feed-forward with feedback versus fully connected (fully recurrent)

3.3.2.1 Supervised and unsupervised learning

Supervised ANNs are called supervised, as they need a teacher for being able to learn. A teacher, in this case, is on the one hand the human presenting training data to the ANN, but primarily the training data itself. Training data are such datasets that feature complete and correct input values as well as the desired output values for the artificial neural network. A commonly used test case for ANNs is the XOR problem, where the network has to have two input neurons for being able to receive the input values and one output neuron, for being able to produce the target value.

Nevertheless, training data is not always available as output values are not always known, which may be the case when clustering data. A special class of ANNs, the self-organizing feature maps, are capable of classifying values into output clusters they defined on their own.

3.3.2.2 Feed-forward artificial neural network

The former type of neural network has its name from the process of sending input values forward through an ANN, which has been denoted feed-forward, related to the direction the data passes through.

3.3.2.3 Feed-forward artificial neural network with feedback connections

Feed-forward with feedback ANNs include, additionally to the latter, also feedback connections, making them recurrent. Such artificial neural networks contain cyclic interconnections between the neuron layers or single neurons.[96] These cyclic interconnections are used to propagate the output values of a specific layer through the extra layer to another specific layer as input values, which form a kind of long term-memory of the ANN.

96 Tahir Mukarram A. (2007): Java Implementation of Neural Networks; USA: Booksurge Publishing Inc., p. 70

3.3.2.4 Fully connected artificial neural network

Fully connected ANNs do not follow a specific direction when processing the input to the output neurons, as every neuron is connected to every other neuron within the network. Fully recurrent ANNs have been proven to be unstable and dynamic,[97] and are, amongst others, applied when simulating chaotic systems.[98,99]

3.3.2.5 Basic artificial neural network structure

The structure of an ANN depends, as indicated beforehand, on its type. The following scheme (Figure 27 - Simple artificial neural network structure) is therefore not valid for each type of ANN, but serves the purpose of explaining data transfer through such one.

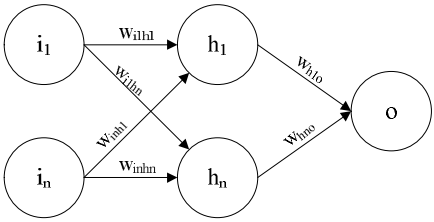

Figure 27 - Simple artificial neural network structure

The schematic ANN dictates over 2 input neurons, which means that the two input values the same time must be presented to it. The purpose behind this is not to save time, but depends from the problem. When e.g. presenting the XOR problem to an ANN, it must have two input neurons, as always two values are presented at the same time. The next layer is the hidden one, in the scheme also consisting of two neurons. A hidden layer is the ANN-internal representation of the processed inputs and at least one is usually needed for enabling the ANN to process complex problems – training often shows if one is required. For generic ANNs, at least one hidden layer is required, as the problem to process is not known before. The only impact a hidden layer in an ANN has when processing a problem of low complexity normally not requiring one, is a decreased performance in terms of processing time. The last neuron in the scheme is the output neuron, receiving the processed data and forwarding it to the outside world. Of course, ANNs are not limited to the number of neurons in each layer shown in the scheme, as e.g. for multidimensional classification (=one input is not classified by one, but more values) at least two output neurons must be available. Additionally, the scheme contains the descriptions of weights between each of the neurons and neuron layers. These weights form the memory of a neural network and are mathematically represented by a matrix. An

97 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 693

98 Wang L. P. (2004): A noisy chaotic neural network for solving combinatorial optimization problems: Stochastic chaotic simulated annealing; IEEE Trans. System, Man, Cybernetics, Part B – Cybernetics, 34(5): 2119–2125

99 Wang L. P. (1998): On chaotic simulated annealing; IEEE Trans. Neural Networks, 9: 716–718

example for saving the memory of a neural network, including its threshold values, is the matrix

$$m_{ih} = \begin{bmatrix} w_{i1hn} & w_{inh1} \\ w_{i1h1} & w_{inhn} \\ t_1 & t_2 \end{bmatrix} \quad (3\text{-}8)$$

based on the connections to the input layer and the hidden layer of Figure 27 - Simple artificial neural network structure. The values of the weights between the input neurons between the input neurons are saved in the columns, where column 1 represents the weights of i_1 to the hidden neurons h_1 and h_n. So does the second column, with the weights from the input neuron i_n to the hidden neurons h_1 and h_n. The two thresholds are saved in the last row of the matrix, in the column of their corresponding neuron. Training or learning of an artificial neural network means 'adaption of the weights between the neurons'. How these weights are adapted, depends from the method of training (3.3.3 Training and learning). Furthermore, ANNs can be regarded as generalization of autoregressive moving-average, known from economy and as ARMA, which is described as follows:[100]

$$\Phi(L)x_t = \Theta(L)\epsilon_t \quad (3\text{-}9)$$

where $\Phi(L)$ and $\Theta(L)$ are polynomials of the order p and q,

$$\Phi(L) = 1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p$$

$$\Theta(L) = 1 - \theta_1 L - \theta_2 L^2 - \cdots - \theta_q L^q \quad (3\text{-}10)$$

where $\{\epsilon_t\}$ represents the white noise, and L the lag operator.

As Fulcher notes correctly, ANNs are nonlinear generalizations of the autoregressive process:

$$x_t = f\left(x_{t-1}, \ldots, x_{t-p}\right) + \epsilon_t \quad (3\text{-}11)$$

3.3.2.6 Perceptron

Perceptrons are supervised ANNs, therefore require training data and can be of the types feed-forward or feedback. Initially, the perceptron has been introduced by Frank Rosenblatt.[101] The two classes of perceptrons are single-layer peceptrons (SLP) and multi-layer perceptrons (MLP). As the descriptions indicate, the SLP consists of just one neuron layer, whereas the MLP has at least one input layer and one output layer as well as one-to-many hidden layers.

3.3.2.6.1 Single layer perceptron

The purpose of an SLP is to segregate data belonging to two different classes. Simply said, data belongs to class one when the neuron fires, and to class two if it does not. Not to fire in

100 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 520

101 Rosenblatt Frank (1958): The Perceptron. A Probabilistic Model for Information Storage and Organization in the Brain; Psychological Reviews, 65: 386–408

case of a bipolar activation function providing the classes one and zero means to put out zero. An SLP with one output neuron is mathematically represented by

$$y = f(\sum_{i=1}^{n} \omega_i x_i + \theta) + \epsilon_t \qquad (3\text{-}12)$$

where f is the activation function, applied to the summed weights ω_i multiplied by the input values x_i, if the threshold is exceeded. ϵ_t is the uncertainty variable of the ANN, also called white noise. Additionally, the formula takes into consideration a bias θ, which is not necessarily used.

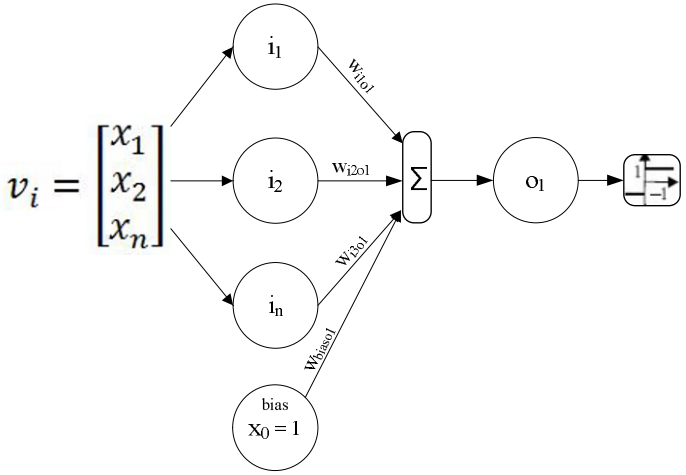

Figure 28 – Single layer perceptron

Figure 28 – Single layer depicts the bias θ with x_0, having the value 1, which is, according to the latter formula, also taken into consideration when summing up. The perceptron works as described before, with the exception that an activation function is depicted. This function described in the scheme is a special bipolar one, meaning that the input is either classified to one, or minus one. Generally, a function capable of classifying inputs into both positive and negative output (not necessarily only into one and minus one) is denoted bipolar. A function only able to classify into positive values is called unipolar.

Thus, a neuron can, and will within this elaboration, also make use of sigmoidal activation functions like the unipolar sigmoid or the bipolar tangens hyperbolicus function mentioned above.

According to the scheme, the following perceptron learning algorithm describes what happens during the training phase:[102]

Start

1. Initial selection of perceptron weights at random.
2. ***Repeat***
   a) To the input neuron, the learning vector x, while $x = x(t) = [x0(t), x1(t), \ldots, xn(t)]T, t = 1, 2, \ldots$ is presented.
   b) The output value y of the perceptron is calculated, according to the before explained equation
   $$y = f\left(\sum_{i=1}^{n} \omega_i x_i + \theta\right)$$
   c) The output value $y(t)$ is afterwards compared with the desired output value $d = d(x(t))$ occurring in the learning sequence.
   d) The weights are then modified according to dependencies:
      i. $if\ y(x(t) \neq d(x(t)), then\ w_i(t+1) = w_i(t) + d(x(t))x_i(t);$
      ii. $if\ y(x(t) = d(x(t)), then\ w_i(t+1) = w_i(t)$, e.g. weights remain unchanged.
3. ***Until criteria are reached***

End

Algorithm 1 - Basic perceptron learning

Hence, the learning algorithm of the SLP x_1 has to adapt the weights $w_1, \ldots, w_n$ in a way that the related neuron puts out one when data belongs to class one, and minus one, when the data set belongs to class two. In the above algorithm, in step 5, sub step a) increases the weight by the desired output value $d(x(t))$ multiplied by a learning vector$x_i(t)$. The algorithm runs as long as an allowed minimum error between the desired output and the actual output has not been reached. Famous examples for learning rules are the Widrow-Hoff rule (Δ or δ rule, 3.3.3.5 Delta rule) or Hebb's learning rule (3.3.3.4 Hebb's learning rule).

According to Rosenblatt's theorem, the learning algorithm of the perceptron converges in final time, and it therefore can learn anything what it is capable to represent. However, due to the fact that the SLP features only one layer, it can only solve linearly separable problems (and may therefore, without exception, be applied on linearly separable data). In other words, an SLP can only solve problems based on data, which can be segregated through planes in space, as the segregation of two points in a two-dimensional space through a line is. An example for linearly inseparable data would be the XOR operator. The following graphs (Figure 29 – OR operator, Figure 30 – AND operator, Figure 31 – XOR operator) describe the problem.

102 Rutkowski Leszek (2008): Computational Intelligence Methods and Techniques; Berlin Heidelberg: Springer-Verlag, p. 192

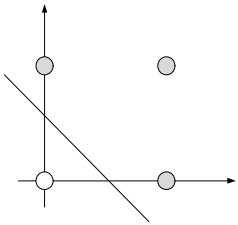

Figure 29 – OR operator

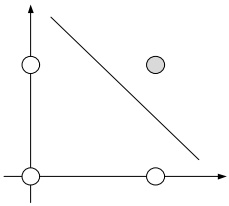

Figure 30 – AND operator

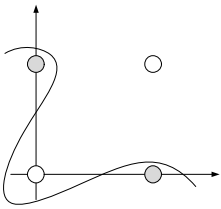

Figure 31 – XOR operator

Figure 31 – XOR operator shows the impossibility of linearly segregating the sets within the XOR-operator.

3.3.2.6.2 Multi layer perceptron

MLPs find application when the necessity of solving problem statements based on convex or arbitrarily complex data. Two-layer perceptrons are capable of separating convex sets, any further layer enables MLPs to separate any sets. The structure of an MLP differs from the one of an SLP, as the name indicates, in the number of layers (Figure 32 – Multi layer perceptron).

Figure 32 – Multi layer perceptron describes an MLP with three layers, consisting of three input neurons, which are presented the input vector v_i, four hidden neurons and two output neurons, which produce the output vector v_o. As indicated in 3.3.1.2.1 Structure, the hidden

neurons are the internal representation of the outside world, transferred by the input neurons and adapted by input neurons' corresponding weights. Every neuron of the MLP makes use of an activation function, as an SLP neuron does. Referring back to ARMA mentioned in 3.3.1 Artificial neural networks, a 3-layer MLP with one output neuron can be described by

$$x_t = h_2(w_0 + \sum_{j=1}^{l} w_j h_1(w_{0j} + \sum_{i=1}^{p} w_{ij} x_{t-i})) + \epsilon_t \qquad (3\text{-}13)$$

where the input layer has p inputs $x_{t-1}, \dots, x_{t-p}$, the hidden layer has l hidden nodes and the output, and there is a single output for the output layer x_t. Layers are fully connected by weights, where w_{ij} is the i^{th} input for the j^{th} node in the hidden layer, whereas w_j is the weight assigned to the j^{th} node in the hidden layer for the output. w_0 and w_{0j} are the biases,

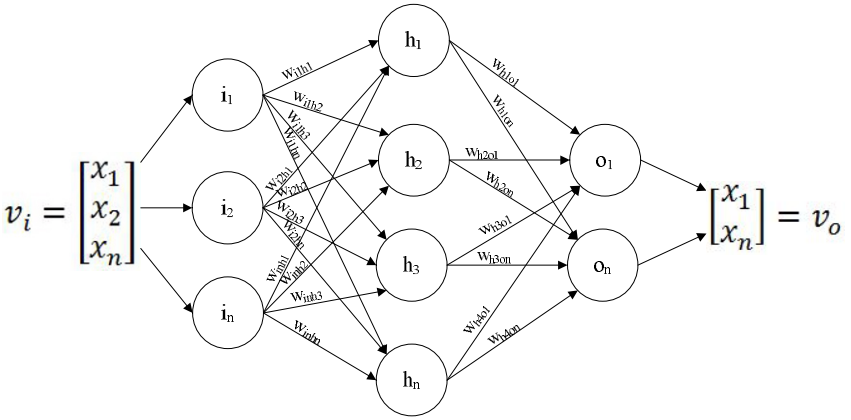

Figure 32 – Multi layer perceptron

h_1 and h_2 are activation functions. ϵ_t is the uncertainty variable of the ANN, also called white noise.[103]

The major advantage of an MLP is the possible application of complex learning functions on a fairly high number of weights, which are the decisive factor in the ANNs capability of learning and remembering how to solve problems of any complexity.

Saving the weights as matrices becomes increasingly complex, the more neurons and layers an ANN includes:

103 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 521

$$m_{ih} = \begin{bmatrix} w_{i1h1} & w_{i2h1} & w_{inh1} \\ w_{i1h2} & w_{i2h2} & w_{inh2} \\ w_{i1h3} & w_{i2h3} & w_{inh3} \\ w_{i1hn} & w_{i2hn} & w_{inhn} \\ t_1 & t_2 & t_3 \end{bmatrix} \quad (3\text{-}14)$$

and

$$m_{ho} = \begin{bmatrix} w_{h1o1} & w_{h2o1} & w_{h3o1} & w_{hno1} \\ w_{h1on} & w_{h2on} & w_{h3on} & w_{hnon} \\ t_1 & t_2 & t_3 & t_4 \end{bmatrix} \quad (3\text{-}15)$$

The depicted ANN describes a feed-forward artificial neural network, as the data is transferred from the input to the output neurons in just one way. The pure calculation of the output of a feed-forward ANN (here without activation function and uncertainty variable) happens as follows:

$$o = (\sum_{i=1}^{n} \omega_i x_i + w_n) \quad (3\text{-}16)$$

The input x_i is multiplied with its corresponding weight ω_i and then summed up with the matrix columns' threshold w_n, in the matrix examples represented by $(t_1, \dots, t_n)$, and in equation (3-12) generalized by θ.

3.3.2.6.3 Spiking artificial neural networks

Spiking neural networks process information differently in the sense that they make use of a temporal component in their operating model. This means that each firing neuron does not fire at each propagation cycle, but rather when a pre-defined membrane potential has been reached. The neural output of a firing neuron is absorbed by a target neuron, which in- or decreases its potential, depending from the signal type. The sum of the incoming signals is called spike-train. Let assume a time series

$$u_{i(t)} = u_r \quad (3\text{-}17)$$

where u_r is the resting potential of the neuron u_i at t. At $t = 0$ the presynaptic neuron j fires and at $t > 0$ the answer of the postsynaptic neuron i is

$$u_{i(t)} - u_r = \varepsilon_{ij(t)} \quad (3\text{-}18)$$

where ε_{ij} is the postsynaptic potential at t. If $\varepsilon_{ij(t)} > 0$ then the signal is excitatory, else it is inhibitory. If now two presynaptic neurons $j = 1,2$ send spikes to i by firing at t_j^1 and t_j^2, then each spike evokes a postsynaptic potential ε_{ij}, and a modification of the potential happens:

$$u_{i(t)} = \sum_j \sum_n \varepsilon_{ij\left(t - t_j^n\right)} + u_r \quad (3\text{-}19)$$

If then the critical value has been reached the action potential in i is stimulated and passed over to other neurons via the axon. After firing the potential is reset to $u_{i(t)} < u_r$.

3.3.2.7 Radial basis artificial neural network

A radial basis network is similar to the before mentioned feed-forward multilayer perceptron, with the restriction that it makes use of one hidden layer in any case. This hidden layer makes use of radial base functions as activation functions, usually Gaussian ones. The equation

$$f(x) = \sum_i^k w_i \varphi(||x - c_i||) \quad (3\text{-}20)$$

describes such a network, where $\varphi()$ is a radial basis function, c_i is the i^{th} center, and k is the number of the center. Both w_i, c_i and k are determined by the data set of x. Typical choices of radial basis functions are the already known

Gaussian function,

$$\varphi(x) = ae^{-\frac{(x-b)^2}{2c^2}} \quad (3\text{-}21)$$

the thin-plate-spline function,

$$\varphi(x) = x^2 log^x \quad (3\text{-}22)$$

the multi-quadratic function,

$$\varphi(x) = (x^2 + \beta^2)^{\frac{1}{2}} \quad (3\text{-}23)$$

the inverse multi-quadratic function,[104]

$$\varphi(x) = \frac{1}{(x^2+\beta^2)^{\frac{1}{2}}} \quad (3\text{-}24)$$

or the compound Gauss function

$$\varphi(x) = \left(1e^{-\frac{(x+2)^2}{(2-1)^2}}\right) + \left(2e^{-\frac{(x-2)^2}{(2-1)^2}}\right) \quad (3\text{-}25)$$

RBF networks are especially useful for function approximation or for predictive ANNs.

3.3.2.8 Recurrent artificial neural network

Although feed-forward ANNs are capable of solving various problems, the application of recurrent ANNs provides several advantages in terms of granularity and abstraction. The notions of granularity and abstraction are used in many subfields of artificial intelligence. The granulation of time and space leads naturally to temporal and spatial granularities.[105] An additional recurrent layer may influence the network's performance and enable the network to solve more complex problems, as the layer, through its influence to another layer, serves as

---

104 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 522

105 Abraham Ajith, Hassanien Aboul-Ella, de Leon F. de Carvalho André Ponce, Snáel Vaclav (2009): Foundations of Computational Intelligence Volume 6: Data Mining; Berlin Heidelberg: Springer-Verlag, p. 68

some kind of long-time memory of an ANN. An ANN implemented in the classic feed-forward architecture, e.g. encounters problems when dealing with moving-average series. In using the multi layer perceptron neural network to represent

$$x_t = \epsilon_t - \theta_{t-1} \tag{3-26}$$

or in another term

$$x_t = \sum_{i=1}^{\infty} \theta^i x_{t-i} + \epsilon_t \tag{3-27}$$

one needs to have an input layer with an infinite number of neurons (infinite memory of the past), namely, $x_{t-1}, x_{t-2}, \ldots$, which is impossible in practice.[106] Although an infinite number of neurons theoretically would lead to an exact representation, a high finite number would indeed suffice, but also increase the complexity of the network by a large number of parameters, which would slow down training and estimation.[107] Therefore, a recurrent ANN can deal with ARMA series, whereas MLPs only can deal with AR time series. A recurrent ANN can be represented by[108]

$$x_t = h_2(w_0 + \sum_{j=1}^{l} w_j h_1(w_{0j} + \sum_{i=1}^{p} w_{ij} x_{t-1} + \sum_{m=1}^{l} \varpi_{mj} z_{m,t-1})) + \epsilon_t \tag{3-28}$$

where

$$z_{m,t} = w_{0m} + \sum_{i=1}^{p} w_{im} x_{t-i} + \sum_{k=1}^{l} \varpi_{kj} z_{k,t-1}, m = 1, \ldots, l \tag{3-29}$$

Two kinds of recurrent ANNs are worth mentioning here, namely the Elman[109] and the Jordan[110] recurrent ANN.

3.3.2.8.1 Elman recurrent artificial neural network

The Elman neural network processes the outputs of the hidden layer to the recurrent or context layer, which passes its outputs again back to the hidden layer in the next iteration. This influences the input, output and adoption by its learning method in the network's next iteration. The connections between the context and the hidden layer are not weighted. Therefore, propagation learning can be applied to the network (Figure 33 - Elman artificial neural network).

106 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 523

107 Mandic Danilo P., Chamber Jonathon (2001): Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures, and Stability; New York: Wiley

108 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 524

109 Elman J. L. (1990): Finding Structure in Time; Cognitive Science, 14, 179-211

110 Jordan Michael I. (1986): Attractor dynamics and parallelism in a connectionist sequential machine; Proceedings of the Eighth Annual Conference of the Cognitive Science Society; Englewood Cliffs: Erlbaum, pp. 531-546

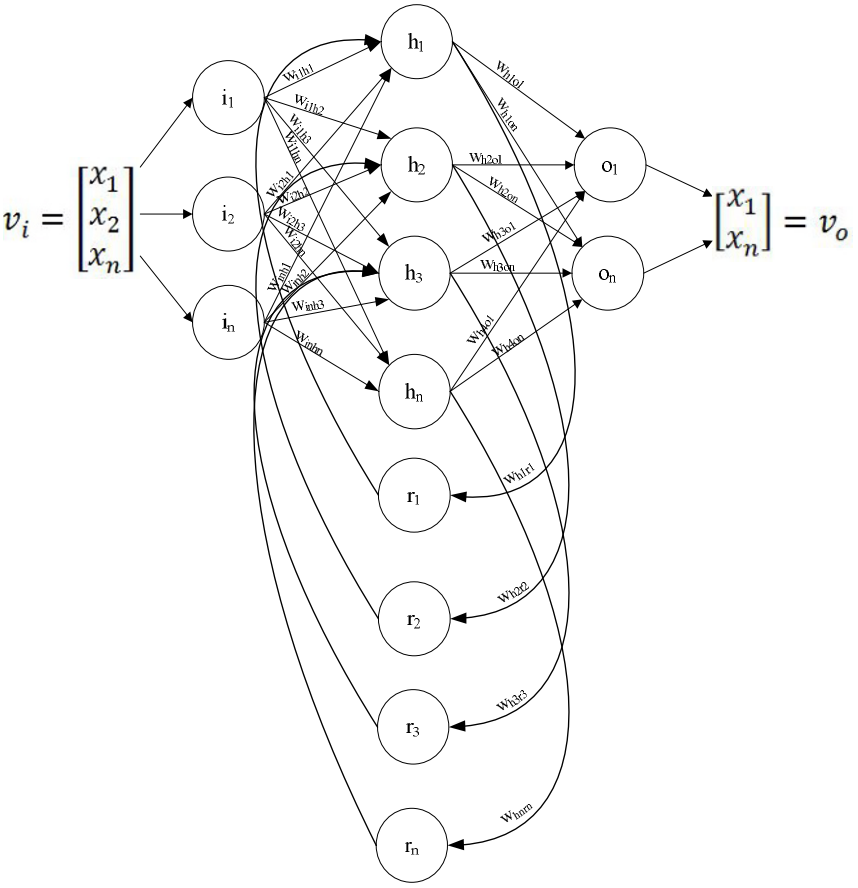

Figure 33 - Elman artificial neural network

3.3.2.8.2 Jordan recurrent artificial neural network

Another example of a recurrent ANN is the Jordan network, having a state layer in which the outputs of the network are propagated into. The connections between the context layer and the hidden layer are not weighted, which again allows propagation training (Figure 34 - Jordan artificial neural network).

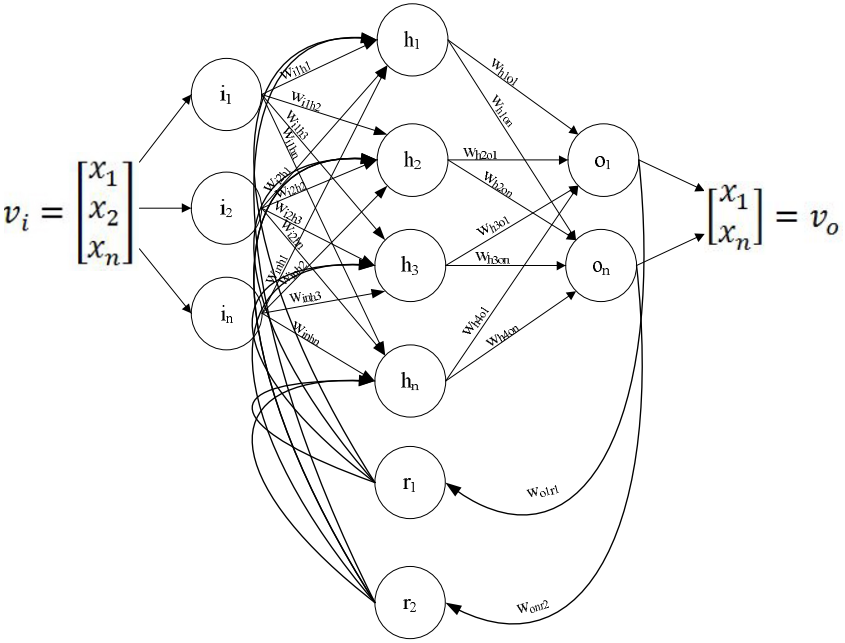

Figure 34 - Jordan artificial neural network

The major difference to the Elman ANN is that every output is propagated into every hidden neuron and not just into one. This is necessary, as the number of output neurons, and therefore the number of the context neurons, very often differs from the number of hidden neurons.

Recurrent ANNs are useful when dealing with problems that require 'long-term memory' for successful processing. There exist more, well-known types of recurrent ANNs which have not been considered within this elaboration.

3.3.2.9 Fully connected artificial neural network

As already indicated, each of the neurons of fully connected ANNs features connections to every other neuron, except themselves – no self-connections exist, but as many input as output ones. Thus, a fully connected ANN has no processing direction like a feed forward ANN has. Two very popular and simple fully connected ANNs are the

– Hopfield network111 and the
– Boltzmann machine.[112]

111 Hopfield J. Joseph (1982): Neural networks and physical systems with emergent collective computational properties; Proceedings Nat. Acad. Sci. (USA) 79, 2554-2558.

Especially the latter type is of utmost importance here, as it is the foundation for the deep belief networks[113] for classification. Both network types are presented an initial pattern through its input neurons, which does not differ from MLPs. However, according to the weight calculations, a new pattern is received from the output neurons. The difference to other ANNs now is that this pattern is fed back to the input neurons, which is possible, as the input and output layer are the same. This cycle continues as long as it takes the network to stabilize. Until stabilization, the network is in a new state after an iteration, which is used for comparison of the actual pattern and the input vector.[114] Both Hopfield ANNs and Boltzmann machines belong to the class of thermal ANNs, as they feature a stochastic component, the energy function, which is applied similarly to simulated annealing-learning explained at 3.3.3.8 Simulated annealing. Energy-based probabilistic models define a probability distribution through an energy function, as

$$p(x) = \frac{e^{-E(x)}}{Z} \qquad (3\text{-}30)$$

where Z is the normalization factor and called the partition function:

$$Z = \sum_x e^{-E(x)} \qquad (3\text{-}31)$$

An energy-based model can be learnt by performing (stochastic) gradient descent on the empirical negative log-likelihood of the training data.

Excursus: the log-likelihood

As the log-likelihood is of importance here, it will be explained in detail, starting with its connection to the likelihood-function. In general, the likelihood function is used within the maximum likelihood-method for the estimation of the parameters of a density or probability distribution. Let assume X is a stochastic variable with a probability function

$$f(x;\theta) \qquad (3\text{-}32)$$

where θ is an unknown parameter that may be multidimensional and $x_1, x_2, \dots, x_n$ are different forms of θ. The likelihood function is the function that assigns each value of this parameter the value

$$L(\theta|x_1, x_2, \dots, x_n) = f(x_1, x_2, \dots, x_n; \theta) \qquad (3\text{-}33)$$

The log-likelihood function is simply the logarithmic likelihood-function. As it is used within the maximum likelihood-estimation, the first and second derivation to θ have to be calculated, as well as the zeros of the first derivation. This is easier when using the log-likelihood

$$log\big(L(\theta)\big) = log(f(x_1, x_2, \dots, x_n; \theta)) \qquad (3\text{-}34)$$

112 Hinton Geoffrey E., Sejnowski Terrence J. (1983): Optimal Perceptual Inference. Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, Washington DC, pp. 448-453.

113 Hinton Geoffrey E., Salakhutdinov Ruslan R. (2006): Reducing the dimensionality of data with neural networks, Science, vol. 313, no. 5786, pp. 504–507, 2006.

114 Heaton Jeff (2010): Programming Neural Networks with Encog 2 in Java; Chesterfield: Heaton Research, Inc., p. 339

This is because the derivation of the log-likelihood to θ is simpler than the likelihood to θ. Generally, if θ^* is a maximum of the log-likelihood function, then it is also a maximum of the likelihood function and vice versa.[115] The basic idea behind the maximum likelihood-function is very simple, as follows. Let assume the approximation for the unknown θ for which the function f is as large as possible. If f is a differentiable function of θ, a necessary condition for f to have a maximum in an interval (not at the boundary) is

$$\frac{\partial f}{\partial \theta} = 0 \qquad (3\text{-}35)$$

The derivative is partial, because f also depends on $(x_1, x_2, \dots, x_n)$. A solution depending on $(x_1, x_2, \dots, x_n)$ is called a maximum likelihood estimate for θ. Thus, 1-30 may be replaced by

$$\frac{\partial ln(f)}{\partial \theta} = 0 \qquad (3\text{-}36)$$

because $f(x_j) > 0$, a maximum of f is in general positive, and $\ln(f)$ is a monotone increasing of the function f, which often simplifies calculations.[116]

As for the logistic regression at first the log-likelihood (1-32) followed by the loss function (1-33) as being the negative log-likelihood need to be defined:

$$L(\theta, D) = \frac{1}{N} \sum_{x^{(i)} \in D} log\, p(x^{(i)}) \qquad (3\text{-}37)$$

$$\ell(\theta, D) = -L(\theta, D) \quad (3\text{-}38)$$

using the stochastic gradient

$$-\frac{\partial\, log\, p(x^{(i)})}{\partial \theta} \qquad (3\text{-}39)$$

where θ are the parameters of the model.[117]

3.3.2.9.1 Hopfield artificial neural network

The Hopfield ANN (Figure 35 - Hopfield artificial neural network) is a self-connected ANN. It contains one single layer of neurons, which is self-connected. Every neuron on the layer is connected to every other neuron on the same layer, without being self-connected. In difference to most of the other ANNs, the Hopfield ANN does not apply threshold values. Another characteristic of this special type of ANN is that it operates on bipolar numbers, allowing binary numbers to be represented numerically in a way that true and false are opposites. Thus, the ANN's activation function is also bipolar, only outputting -1 or 1 for each

115 K rengel Ulrich (1988): Einführung in die Wahrscheinlichkeitstheorie und Statistik; Braunschweig/Wiesbaden 1988: Verlag Friedrich Vieweg & Sohn

116 Kreiszig Erwin (1999): Advanced Engineering Mathematics, 8th ed.; Singapore: John Wiley & Sons, p. 1107

117 Deeplearning.net (2012): Restricted Boltzmann Machines (RBM) [2012-15-08]; Deeplearning. net; URL: http://deeplearning.net/tutorial/rbm.html

output neuron. [118] Until stabilization, the state of the Hopfield network will move towards the closest pattern, thus 'recognizing' that pattern. As the Hopfield network moves towards one of these patterns, the energy, applied with the energy function, lowers. [119] Let assume that $v_{i(t)}$ is the state of a node i at time t and

$$v_{i(t)} = g\left(u_{i(t)}\right) \tag{3-40}$$

$$\frac{du_{i(t)}}{dt} = -E_i(v_t) \tag{3-41}$$

where

$$E_i(v) = \frac{\partial}{\partial v_i} E(v) \tag{3-42}$$

and where g represents an arbitrary increasing function, then a Hopfield ANN may be used to compute a local minimum of this function defined on a hypercube. As the equations can generally be seen as gradient-descent method, they usually do not find the global minimum:

$$\frac{d}{dt}E(v_t) = \sum_i E_i(v_t)\frac{d(v_{i(t)})}{dt} = \sum_i E_i(v_t)g'\left(u_{i(t)}\right)\frac{d(u_{i(t)})}{dt} = -\sum_i g'\left(u_{i(t)}\right)E_i^2(v_t) \leq 0 \tag{3-43}$$

The equation shows that E is decreasing, but not strictly decreasing and any equilibrium reached may only be a local minimum. [120] Graphically, a Hopfield ANN may be represented as shown in Figure 35 - Hopfield artificial neural network

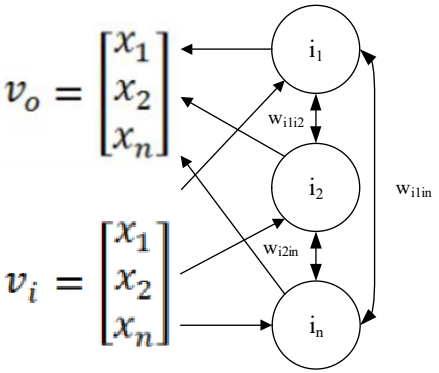

Figure 35 - Hopfield artificial neural network

118 Heaton Jeff (2008): Introduction to Neural Networks for Java, 2nd ed.; Chesterfield: Heaton Research, Inc., p. 85

119 Heaton Jeff (2010): Introduction to Encog 2.5 for Java, Rev. 3; Chesterfield: Heaton Research, Inc., p. 44

120 Wong Eugene (1991): Stochastic Neural Networks; California: University of Berkelay, Department of Electrical Engineering and Computer Science

3.3.2.9.2 Boltzmann Machine

At a first glance, a Boltzmann machine seems to be identical to a Hopfield ANN. However, it contains hidden neurons. Boltzmann machines are a particular form of the log-linear Markov random field, e.g. for which the energy function is linear in its free parameters. To make them powerful enough to represent complicated distributions, it is considered that some of the variables are never observed, represented by the above mentioned hidden neurons. By having more hidden variables, the modelling capacity of the Boltzmann machine can be increased.[121] The hidden neurons require adoptions in the above equations, with respect to the hidden and observed part:

$$(x) = \sum_h P(x,h) = \sum_h \frac{e^{-E(x)}}{Z} \qquad (3\text{-}44)$$

For being able to map this equation to a similar one as equation (3-44) shows, the notation of free energy must be introduced:

$$\mathcal{F}(x) = -log \sum_h e^{-E(x,h)} \qquad (3\text{-}45)$$

leading to

$$P(x) = \frac{e^{-\mathcal{F}(x)}}{Z} \qquad (3\text{-}46)$$

where

$$Z = \sum_x e^{-\mathcal{F}(x)} \qquad (3\text{-}47)$$

The data negative log-likelihood gradient's form then is:

$$-\frac{\partial\, log\, p(x)}{\partial\theta} = \frac{\partial \mathcal{F}(x)}{\partial\theta} - \sum_{\tilde{x}} p(\tilde{x}) \frac{\partial \mathcal{F}(\tilde{x})}{\partial\theta} \qquad (3\text{-}48)$$

The gradient contains two terms, referring to the positive and negative phases. The terms positive and negative do not refer to the sign of each term in the equation, but rather reflect their effect on the probability density defined by the model. The first term increases the probability of training data (by reducing the corresponding free energy), while the second term decreases the probability of samples generated by the model.[122] Graphically, a Boltzmann machine can be described as depicted in Figure 36 - Boltzmann machine.

121 Deeplearning.net (2012): Restricted Boltzmann Machines (RBM) [2012-15-08]; Deeplearning.net; URL: http://deeplearning.net/tutorial/rbm.html

122 Deeplearning.net (2012): Restricted Boltzmann Machines (RBM) [2012-15-08]; Deeplearning.net; URL: http://deeplearning.net/tutorial/rbm.html

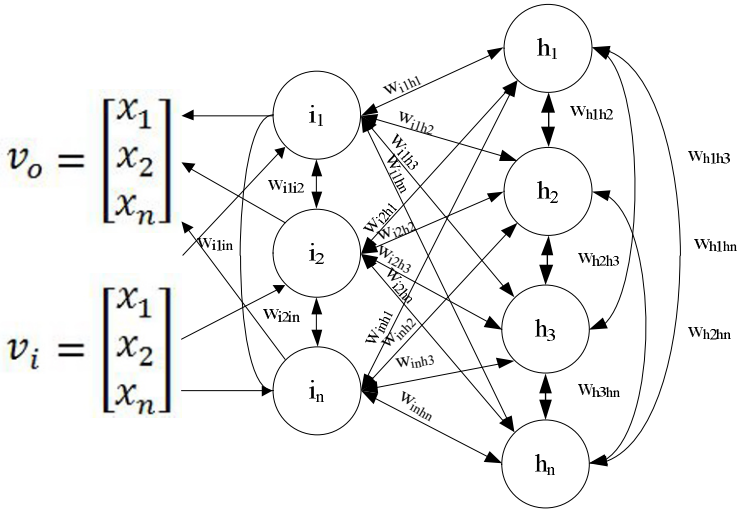

Figure 36 - Boltzmann machine

Every neuron is connected to every other neuron without featuring self-connections. The input neurons are the output neurons at once, thus consuming the input vector and presenting the output vector.

A further development of the Boltzmann machine is the so-called restricted Boltzmann machine (RBM).[123] An RBM consists of a layer of visible units and a layer of hidden units with no visible-visible or hidden-hidden connections. With these restrictions, the hidden units are conditionally independent given a visible vector, so unbiased samples data can be obtained in one parallel step (Figure 37 - Restricted Boltzmann machine).[124]

123 Smolensky Pavel (1986): Information processing in dynamical systems: Foundations of harmony theory. In Rumelhart, D. E. and McClelland, J. L., editors, Parallel Distributed Processing: Volume 1: Foundations, pages 194-281. MIT Press, Cambridge, MA.

124 Scholarpedia (2011): Boltzmann Machine (2012-15-08); Scholarpedia; URL: http://www.scholarpedia.org/article/Boltzmann_machine#Restricted_Boltzmann_machines

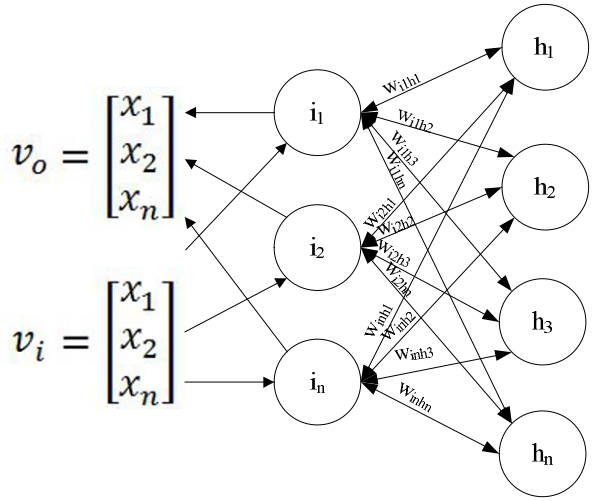

Figure 37 - Restricted Boltzmann machine

The first thing in training a RBM is to create a probability distribution based on the propagated inputs. Each hidden unit then takes a binary value based on this distribution:

$$p(h_j = 1|v) = \sigma\left(\sum_{i=1}^{D} w_{ij} v_i + c_j\right) \qquad (3\text{-}49)$$

where h_j is the j^{th} hidden neuron, v_i the i^{th} visible neuron, w_{ij} the weight between the i^{th} visible and the j^{th} hidden neuron.c_j is the bias of the j^{th} hidden neuron, and σ represents the sigmoid activation function.

Next, each of the visible neurons takes a value based on the probability distribution of the propagated inputs, just in the other direction:

$$p(v_i = 1|h) = \sigma\left(\sum_{j=1}^{D} w_{ij} h_j + b_i\right) \qquad (3\text{-}50)$$

where b_i is the visible layer's bias, and the visible units should reconstruct the original input. If now b, c, the biases of the visible and hidden layers, and W, the weights of the RBM, are considered to be a vector parameter θ, then $p(v|\theta)$ is the probability that can be achieved to get close to the real distribution of v, so θ must be learned. For evaluating the results, the energy function $E(v, h)$ must be introduced:

$$(v, h) = -b^T v - c^T h - h^T W v = -\sum_{i=1}^{D} b v_i - \sum_{j=1}^{M} c\, h_j - \sum_{j=1}^{M} \sum_{i=1}^{D} h_j w_{ij} v_i \qquad (3\text{-}51)$$

An RBM is trained to model the joint probability distribution of its inputs (explanatory variables) and the corresponding labels (response/output variables), both represented by the

visible units of the RBM.[125] Thus, the joint probability density function, showing the simultaneous probability of x given a and a given x, which in an RBM is expressed as the weights shared between the two RBM layers, is

$$p(v,h) = \frac{1}{Z} exp\big(-E(v,h)\big) \tag{3-52}$$

where the normalization factor Z (called partition or normalization function) is

$$Z = \sum_{v,h} exp\big(-E(v,h)\big) \tag{3-53}$$

In order to determine how to train a RBM, we are interested in the distribution of visible variables, thus the marginal distribution over hidden variables taking the form

$$p(v|\theta) = \sum_h P(v,h) = \frac{1}{Z}\sum_h exp\big(-E(v,h)\big) \tag{3-54}$$

Then, the log-likelihood gradient is used to train the model:

$$ln\, L(\theta|v) = ln\, p\,(v|\theta) = ln\frac{1}{Z}\sum_h exp\big(-E(v,h)\big) = ln\sum_h exp\big(-E(v,h)\big) - ln\sum_{v,h} exp\big(-E(v,h)\big) \tag{3-55}$$

The resulting log-likelihood gradient indicates that as long as it is possible to compute the derivative of free energy for the training examples, and the derivative of free energy for the model's own distribution, it is possible to train the model tractably. Next, the gradient against the model parameter θ (θ is used to represent the model parameters) is calculated, starting with the derivative:

$$\frac{\partial\, ln\, L(\theta|v)}{\partial\theta} = \frac{\partial}{\partial\theta}\big(ln\sum_h exp\big(-E(v,h)\big)\big) - \frac{\partial}{\partial\theta}\big(ln\sum_{v,h} exp\big(-E(v,h)\big)\big) = -\frac{1}{\sum_h exp\big(-E(v,h)\big)}\sum_h exp\big(-E(v,h)\big)\frac{\partial E(v,h)}{\partial\theta} = \frac{1}{\sum_h exp\big(-E(v,h)\big)}\sum_{v,h} exp\big(-E(v,h)\big)\frac{\partial E(v,h)}{\partial\theta} = -\sum_h p(h|v)\frac{\partial E(v,h)}{\partial\theta} + \sum_{v,h} p(v|h)\frac{\partial E(v,h)}{\partial\theta} \tag{3-56}$$

The gradient of each weight is then

$$\begin{aligned}\frac{\partial\, ln\, L(\theta|v)}{\partial w_{ij}} &= \sum_h p(h|v)\frac{\partial E(v,h)}{\partial w_{ij}} + \sum_{h,v} p(h|v)\frac{\partial E(v,h)}{\partial w_{ij}} \\ &= \sum_h p(h|v)h_j v_i - \sum_v p(v)\sum_h p(h|v)h_j v_i \\ &= p\big(H_j = 1\big|v\big)v_i - \sum_v p(v)p\big(H_j = 1\big|v\big)v_j\end{aligned} \tag{3-57}$$

125 Fischer Asija, Igel Christian: Training Restricted Boltzmann Machines: An Introduction [2016-07-04]; URL: http://image.diku.dk/igel/paper/TRBMAI.pdf

and for each bias

$$\frac{\partial \ln L(\theta|v)}{\partial c_j} = p(H_j = 1) - \sum_v p(v)p(H_j = 1|v) \tag{3-58}$$

where c_j represents the j^{th} bias of the hidden layer.

When the gradient equation is applied as it is, combinatorial explosion does not allow us to solve it in a realistic time frame. The term $\sum_v p(v)$ states that the probability distribution over all $\{0,1\}$-patterns must be calculated. This in turn implies that the calculation takes into account patterns which are not in the source data. A method to overcome this challenge is contrastive divergence (CD), which leverages Gibbs sampling. Let assume $v^{(0)}$ is an input vector, and $v^{(k)}$ is an input (output) vector that can be obtained by sampling k times using exactly this input vector:

$$h_j^{(k)} \sim p(h_j|v^{(k)}) \tag{3-59}$$

$$h_i^{(k+1)} \sim p(v_i|v^{(k)}) \tag{3-60}$$

Thus, $p(v)$is approximated via iterative Gibbs sampling, and the derivative of the log likelihood function is then

$$\frac{\partial \ln L(\theta|v)}{\partial \theta} = -\sum_h p(h|v)\frac{\partial E(v,h)}{\partial \theta} + \sum_{v,h} p(v,h)\frac{\partial E(v,h)}{\partial \theta} \approx -\sum_h p(h|v^{(0)})\frac{\partial E(v,h)}{\partial \theta} + \sum_{v,h} p(v^{(k)},h)\frac{\partial E(v^{(k)},h)}{\partial \theta} \tag{3-61}$$

The model parameter θ consists then of

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} + \eta\left(p(H_j = 1|v^0)v_i^{(0)} - p(H_j = 1|v^{(k)})v_i^{(k)}\right) \tag{3-62}$$

$$b_i^{(\tau+1)} = b_i^{(\tau)} + \eta\left(v_i^{(0)} - v_i^{(k)}\right) \tag{3-63}$$

$$c_j^{(\tau+1)} = c_j^{(\tau)} + \eta\left(p(H_j = 1|v^0)\right) - p(H_j = 1|v^{(k)}) \tag{3-64}$$

where τ stands for the number of iterations, and η is the learning rate. Summing up, CD sampling k times is CD-k, and usually CD-1 is sufficient. The algorithm for training an RBM is as follows:

Start

1. Initial selection of RBM weights at random.
2. ***Repeat***
   a) Set the input neurons' states to the learning vector x, while $x = x(t) = [x0(t), x1(t), \ldots, xn(t)]T, t = 1, 2, \ldots$
   b) ***For each***
      weight $w_{ij}^{(\tau)}$ set

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} + \eta\left(p(H_j = 1|v^0)v_i^{(0)} - p(H_j = 1|v^{(k)})v_i^{(k)}\right)$$

c) ***For each***
visible bias $b_i^{(\tau)}$ set

$$b_i^{(\tau+1)} = b_i^{(\tau)} + \eta\left(v_i^{(0)} - v_i^{(k)}\right)$$

d) ***For each***
hidden bias $c_j^{(\tau)}$ set

$$c_j^{(\tau+1)} = c_j^{(\tau)} + \eta\left(p\left(H_j = 1 \middle| v^0\right)\right) - p\left(H_j = 1 \middle| v^{(k)}\right)$$

3. Until criteria are reached

End

Algorithm 2 – RBM learning

3.3.2.9.3 Support vector machine

Support vector machines[126] (SVMs) are, as ANNs, used for classification, but are an alternative to these. A support vector machine separates an amount of objects into classes, leaving a considerable area around the classes free of objects. Classifiers classifying objects in such way are called 'large margin classifiers'. As for supervised learning within ANNs it is necessary that for an amount of training data is known to which class they belong to. The SVM is able to separate these objects into classes by inserting a hyper plane between the classes and afterwards to maximize the distance between the vectors closest to the hyper plane. The hyper plane is therefore only dependent from vectors closest to it. As a matter of fact, a hyper plane is very often not capable of linearly separating objects. This is where support vector machines map an n-dimensional input space non-linearly into a high dimensional feature space[127], possibly to a space with infinite dimensions:

$$\emptyset: V^n \longrightarrow V^m \qquad (3\text{-}65)$$

where V^n represents the lower-dimensional input space and V^m representing the high-dimensional feature space. In a high dimensional feature space also a very complex amount of vectors becomes linearly separable. This space is where the hyper plane is then defined. The example of Fulcher shows, how this is conducted with a series of l historical observations:

$$(y_1, x_1), \ldots, (y_l, x_l) \qquad (3\text{-}66)$$

where $y_i \in V^1$ and $x_i \in V^n$.

The approximation and estimation of the functional relation between y_i and x_i is done by

$$= f(x) =< w, \emptyset(x) > +b = \sum_{i=1}^{m} w_i \emptyset(x_s)_i + b \qquad (3\text{-}67)$$

126 Vapnik V. (1998): The support vector method of function estimation. In: Suykens J., Vandewalle J. (eds.): Nonlinear Modeling: Advanced Black-Box Techniques; Boston MA: Kluwer: 55–85

127 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 528

where <.,.> denotes the inner product. The vector w and the constant b are to be determined by following the structural risk minimization principle, known from statistical learning theory.[128] Furthermore, the neurons of the hidden layer in the SVM will not take the same inputs or apply the same transfer function, as it is common in ANNs. For classification, Fulcher describes the SVM by

$$f(x) = \sum_a y_i \alpha_i^* < \emptyset(x_a), \emptyset(x) > b^* \qquad (3\text{-}68)$$

where α_i^* and b^* represent coefficients satisfying the structural risk minimization principle, and s the set of all support vectors. If y_i are categorical, such as $y_i \in \{-1,1\}$, the support vectors, a subset of $\{x_i\}_{i=1}^{l}$, are determined by the minimization process.[129] The classification is carried out according to the sign of the function[130]:

$$y = \begin{cases} 1, & if \quad f(x) > 0 \\ -1, & if \quad f(x) < 0 \end{cases} \qquad (3\text{-}69)$$

3.3.2.9.4 Self-organizing feature map

Self-organizing feature maps (SOFMs) are used to picture high-dimensional data onto a low-dimensional map while trying to keep the neighbor structure of data as good as possible. This means that data close to each other in an n-dimensional space should also stay close to each other in the low-dimensional map – the neighbor structure is kept. SOMs inventor, Teuvo Kohonen, was inspired by the sensory and motor parts of the human brain.[131]

Figure 38 – Self organizing feature map – the scheme of a SOM shows that every component of the input vector x is represented by an input neuron and is connected with the above low-(two-) dimensional layer. During a learning phase the weight vectors of a SOM are adapted in a self-organizing way.[132] As other ANNs, the SOM consists of neurons $(n_1, \dots, n_n)$, each having a weight vector w_i and a distance to a neighbor neuron. The distance between the neurons n_i and n_j is n_{ij},. As Figure 38 – shows, each neuron is allocated a position in the low-dimensional map space. As in all other ANNs, initially the neuron weights are randomized. During learning, the similarity of each input vector to the weights of all neurons on the map is calculated, meaning that all weight vectors are compared with the input vector $d \in D$. The SOMs learning algorithm therefore belongs to the group of unsupervised learning algorithms. The neuron showing the highest similarity, having the smallest distance d_{small} to $d \in D$ is then selected as the winning neuron n_{win}:[133]

---

128 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 528

129 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 529

130 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 529

131 Kohonen Teuvo (1990): The Self-Organizing Map; Proceedings of the IEEE 78, Nr. 9, p. 1464-1480

132 Ritter Helge, Martinez Thomas, Schulten Klaus (1991): Neuronale Netze. Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke; Addison Wesley

133 Kramer Oliver (2009): Computational Intelligence: Eine Einführung; Berlin Heidelberg: Springer-Verlag, p. 141

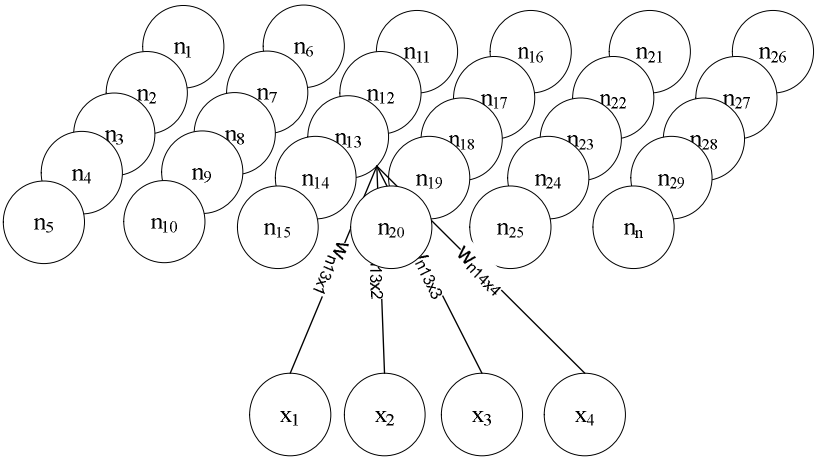

Figure 38 – Self organizing feature map

$$d_{small} = \min_{1 \le j \le n} d\{(d \in D, w_j)\} \qquad (3\text{-}70)$$

Not only the weights of the winning neuron are then adapted, but also the weights of the neighbor neurons have to be taken into consideration by the neighborhood function φ_n and the learning rate μ. The neighborhood function shall feature the following characteristics:[134]

- μ has its center at the position of n_{win} and is a maximum there.
- The neighbor neurons are considered according to a radius. Within this radius, for distances smaller than r, φ_n leads to outcomes greater than zero, and for distances greater than r, it takes zero.

Again, the Gauss function fulfils all the requirements and is a good choice. The adaption of the weights is then carried out as follows:

$$w_i^{(t+1)} = w_i^{(t)} + \mu\varphi_n\left(w_{n_{win}}, w_i^{(t)}, r\right)\left(d \in D - w_i^{(t)}\right) \qquad (3\text{-}71)$$

During learning, not only the neuron weights of the winning neuron and its neighborhood neurons must be adapted, also the learning rate and the neighborhood radius has to be reduced in each iteration, done by $\sigma^{(t+1)}$:[135]

$$\sigma^{(t+1)} = \sigma_s * \left(\frac{\sigma_e}{\sigma_s}\right)^{(t+1)/(t+1)_e} \qquad (3\text{-}72)$$

134 Kramer Oliver (2009): Computational Intelligence: Eine Einführung; Berlin Heidelberg: Springer-Verlag, p. 141 ff.

135 Kramer Oliver (2009): Computational Intelligence: Eine Einführung; Berlin Heidelberg: Springer-Verlag, p. 143

where σ_s represents the starting value and σ_e the ending value, also being the function value of $t(+1)_e$. Kohonen defined the SOM-algorithm as follows:[136]

Start

1. Initial selection of SOM weights at random.
2. ***Repeat***
   a) Present $d \in D$ to every weight vector of every neuron $w_i, \dots, w_n$
   b) Determine the winning neuron according to the smallest distance:

$$d_{small} = \min_{1 \le j \le n} d\{(d \in D, w_j)\}$$

   c) Determine the neighborhood neurons with φ_n and adapt the weights according to

$$w_i^{(t+1)} = w_i^{(t)} + \mu\varphi_n\left(w_{n_{win}}, w_i^{(t)}, r\right)\left(d \in D - w_i^{(t)}\right)$$

   d) Decrease the learning rate μ or the neighborhood radius r *by*

$$\sigma^{(t+1)} = \sigma_s * \left(\frac{\sigma_e}{\sigma_s}\right)^{(t+1)/(t+1)_e}$$

3. ***Until criteria are reached***

End

Algorithm 3 – SOM learning

3.3.2.9.5 Committee machines

Committee machines (CMs) are a number of ANNs working together and trying to find the best solution for one problem. ANNs working as groups usually find more accurate solutions than just one neural network applied to a problem statement. Although one ANN can be trained so solve a specific problem, it might not be as accurate as one might want to have it. The problem is that a software engineer can develop the ANN, and tell it in which way it should learn, but he cannot predict how accurate the ANN will learn and if the ANN will learn correctly (e.g. see problems in 3.3.3.6.1 Back propagation training). The more secure approach therefore is to train more than one ANN on one how to solve one problem statement simultaneously. ANN committee machines therefore need a combiner to conclude the results (Figure 39 - Committee machine).

136 Kohonen Teuvo (1990): The Self-Organizing Map; Proceedings of the IEEE 78, Nr. 9, p. 1464-1480

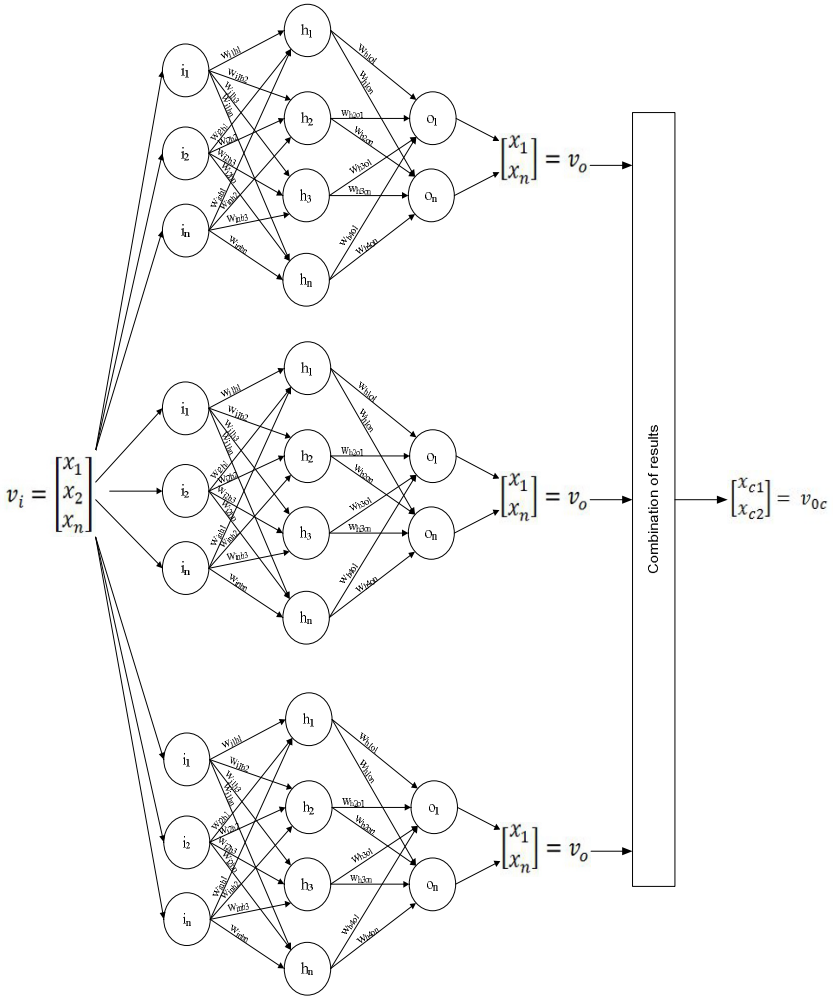

Figure 39 - Committee machine

The above schematic picture shows a committee machine of three multi layer perceptrons. Committee machines are also often referred to as experts, as more than one 'thought' leads to the final solution. Another solution for ensuring the quality of the solution is the training of hybrid committee machines. Several types of hybrid CMs are possible, containing

– ANNs of the same type with different learning functions and the same activation functions,

– ANNs of the same type with different learning functions but different activation functions,
– or a combination of the latter, and
– ANNs of different types

The application of one of these approaches tends to increase the quality of the training's result in a high mass. Slightly more complex, but also an approach used by the prototype described within the practical part of this thesis is the combination of several hybrid committee machines to a committee machine. This combination of experts is can be considered as step towards an artificial immune system, as it is capable of processing any numeric input, as a human immune system is capable of. Within the research field of artificial immune systems the biological immune system is used as a metaphor for computational problems.[137]

The functionality of the combiner of a committee machine depends on the preferred implementation. Possible implementations are e.g.

– the calculation of the mean value of all solutions or
– the brewing of the most similar solutions by simultaneous reduction of the most different solutions, followed by the calculation of the medium value of the remaining solutions.

At the example of a feed-forward ANN, trained by simulated annealing, a committee of three experts would be trained as follows:

Start

1. ***Repeat***
   a) Randomize weights.
   b) Repeat
      i. Calculate the network output for the value $d \in D$
      ii. Evaluate the fitness of each neuron:
      iii. Calculate the error $\delta_{d \in D}$ for each output neuron $d \in D$

$$\delta_{d \in D=}(desired_{d \in D} - actual_{d \in D}) actual_{d \in D}(1 - actual_{d \in D})$$

      iv. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d \in D} (w_{hidd \in D} \delta_{d \in D})$$

      v. Repeat
         1. Create new ANN and randomize weights according to T
         2. Calculate the error $\delta_{d \in D}$ for each output neuron $d \in D$

$$\delta_{d \in D=}(\text{desired}_{d \in D} - \text{actual}_{d \in D})\text{actual}_{d \in D}(1 - \text{actual}_{d \in D})$$

         3. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = \text{actual}_{hid}(1 - \text{actual}_{hid}) \sum_{d \in D} (w_{hidd \in D} \delta_{d \in D})$$

         4. Compare solutions according to

137 Bertelle Cyrille, Duchamp Gérard H. E., Kadri-Dahmani Hakima (2009): Complex Systems and Self-organization Modelling; Berlin Heidelberg: Springer-Verlag, p. 71

$$\Delta_{C(S')C(S)} = C(S') - C(S)$$

5. If $C(S')$ is better than $C(S)$, replace $C(S)$

vi. Until max tries for current temperature reached

vii. Decrease temperature by

$$\varphi_{C(S')C(S)} = T * e^{\frac{ln\left(\frac{s}{e}\right)}{c-1}}$$

c) Until lower temperature bound reached

2. Until all experts have been trained

3. Combine the results

End

Algorithm 4 - Committee of SA feed-forward ANNs

3.3.3 Training and learning

As it is obvious now, how an ANN transports data from its input neurons to its output neurons, how this data is changed due to a neuron's corresponding weights and activation function, and that an artificial neural network learns through weight adaption, it is of severe importance to understand some different learning approaches. Different ANNs and different problem statements require different learning approaches. As elucidated beforehand, training within an ANN is done by the adaption of the neuron connections' weights, which does not happen in only one step, but over multiple iterations. Before discussing different learning methods, some basic knowledge for calculating parameters used by the learning algorithms is explained.

3.3.3.1 Supervised and unsupervised training

As already indicated at 3.3.2.1 Supervised and unsupervised learning, a supervised training algorithm requires both inputs and outputs in the training data sets. On the contrary, the ANN is not presented any desired output with the training sets in unsupervised learning. The major difference lies in the field of application, as unsupervised learning networks are usually applied on data for clustering, when not being aware of how to cluster the data, whereas supervised learning ANNs are applied when one knows, which output the network shall produce on a given input.

Some networks apply hybrid learning approaches, thus both unsupervised and supervised learning. Examples are radial basis function networks, which use supervised learning at the output layer and unsupervised learning at the hidden layer(s), or counter propagation networks, applying unsupervised learning in combination with Grossberg's outstar[138,139].

138 Wasserman Philip D. (1989): Neural Computing: Theory and Practice; New York: Van Nostrand Reinhold

3.3.3.2 (Root) mean squared error

Usually, an ANN is initialized with random weights between the neurons. The input is then sent through the network, which lets it produce an output that will differ from the desired output to greater or lesser extent. The difference between the desired and the actual output is used to calculate the mean squared error (MSE) or the root mean square error (RMSE), which happens after a training iteration. For being able to determine the (R)MSE, the errors for every provided training set have to be calculated. The (R)MSE represents the average error of the whole network and is, after each complete iteration (an iteration involves the presenting of all training sets to the ANN), compared with the predefined, allowed error. As long as the (R)MSE is not less or equal the allowed error or another stop criteria has been reached, the network continues these steps. The MSE-equation is

$$x_{mse} = \frac{1}{n}\sum_{i=1}^{n} x_i^2 \qquad (3\text{-}73)$$

which can also be written as

$$x_{mse} = \frac{x_1^2+x_2^2+\cdots+x_n^2}{n} \qquad (3\text{-}74)$$

where the RMSE is

$$x_{rmse} = \sqrt[2]{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \qquad (3\text{-}75)$$

or

$$x_{rmse} = \sqrt[2]{\frac{x_1^2+x_2^2+\cdots+x_n^2}{n}} \qquad (3\text{-}76)$$

where *n* represents the number of the input values in both equations. For the application of the RMS to the output of an ANN, the equation

$$x_{rmse} = \sqrt[2]{\frac{1}{n}\sum_{i=1}^{n}(actual_i - desired_i)^2} \qquad (3\text{-}77)$$

has to be considered,[140] where the actual value is the value currently produced by the ANN and the desired value the expected one. Therefore, learning is the adaption of the weighs in the matrix representing the artificial neural network.

The next important value is the definition of the trained epochs an ANN is allowed to run through. An ending criterion may be the allowed minimum error, but can also be the attainment of a predefined number of iterations. It has to be considered that an error is only provided to an ANN when supervised learning is applied, as in unsupervised learning the network decides itself, what the output is (on the contrary to supervised learning ones). Supervised training can be applied with SLPs or MLPs, as well as with fully connected ANNs

---

139 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 693

140 Heaton Jeff (2008): Introduction to Neural Networks for Java, 2nd ed.; Chesterfield: Heaton Research, Inc., p. 126

like the Hopfield ANN, whereas unsupervised learning finds application within topographic maps, as the Self-Organizing Feature Map (3.3.2.9.4 Self-organizing feature map).

3.3.3.3 Estimators

Unfortunately, as the (root) mean squared error as an error measure between the actual and desired output is not robust when having to deal with outliers, many robust estimators have been used recently, such as M-estimators (Maximum-likelihood estimators), R-estimators (estimates based on rank transformations), L-estimators (Linear combination of order statistics), and LMedS estimators (Least Median of Squares). Anything required for making training algorithms more robust to replace the squared residuals in the network error by another function of the residuals, yielding

$$x_{mse} = \frac{1}{n}\sum_{i=1}^{n} \rho(x_i) \qquad (3\text{-}78)$$

where ρ is a symmetric, positive-definite function with a unique minimum at zero, and is chosen to be less increasing than square completely in the presence of outliers. [141]

3.3.3.4 Hebb's learning rule

One of the best known and first learning rules is Hebb's rule,[142] described by the psychologist Donald Olding Hebb, which is applied both within supervised and unsupervised learning ANNs. During his research, Donald Hebb realized that the strength of the connection between two neurons increases, if both fire at the same time. This led him to develop an algorithm, which allows the modification of weights in an artificial neural network:

$$w_i(t+1) = w_i(t) + \Delta w_i \qquad (3\text{-}79)$$

Through Δw_i the necessary change (delta) in the weight of the corresponding connection is calculated at the point in time $t+1$, as the following equation shows:

$$\Delta w_i = \mu x_i y \qquad (3\text{-}80)$$

The parameter μ represents the learning rate, which influences the rate at which the weights of the connection are adapted. Hebb's rule, as a result, tends to strengthen the output in the direction it has a tendency towards.[143] Figure 40 - Hebb's rule graphically represents the application of the rule.

---

141 Hassanien Aboul-Ella, Abraham Ajith, Vasilakos Athanasios V., Witold Pedrycz (2009): Foundations of Computational Intelligence Volume 1: Learning and Approximation; Berlin Heidelberg: Springer-Verlag, p. 224 ff.

142 Hebb Donald (1949): Organization of behaviour; New York: John Wiley

143 Heaton Jeff (2008): Introduction to Neural Networks for Java, 2nd ed.; Chesterfield: Heaton Research, Inc., p. 134

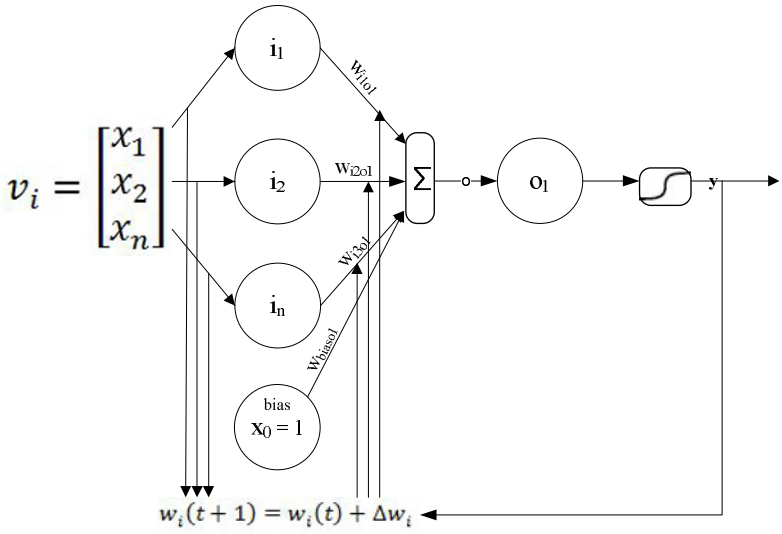

Figure 40 - Hebb's rule[144]

Hebb's rule is also applicable when applying it with unsupervised learning, but needs a slight modification, which is the consideration of the desired output value o_d:

$$\Delta w_i = \mu x_i o_d \qquad (3\text{-}81)$$

3.3.3.5 Delta rule

The delta rule, which targets on calculating the least mean square, is a rule only applied within supervised learning ANNs. This is, because the delta learning rule also requires desired outputs to compared with the actual output values, as Hebb's rule in the supervised manner does:

$$w_i(t+1) = w_i(t) + \Delta w_i \qquad (3\text{-}82)$$

As in Hebb's rule, through Δw_i the necessary change (delta) in the weight of the corresponding connection is calculated, as equation (3-83) shows:

$$\Delta w_i = \mu \delta x_i \qquad (3\text{-}83)$$

The difference lies in δ, which considers the output o_{ba} before transferring it through the activation function, as Figure 41 - Delta rule shows.

144 Rutkowski Leszek (2008): Computational Intelligence Methods and Techniques; Berlin Heidelberg: Springer-Verlag, p. 207

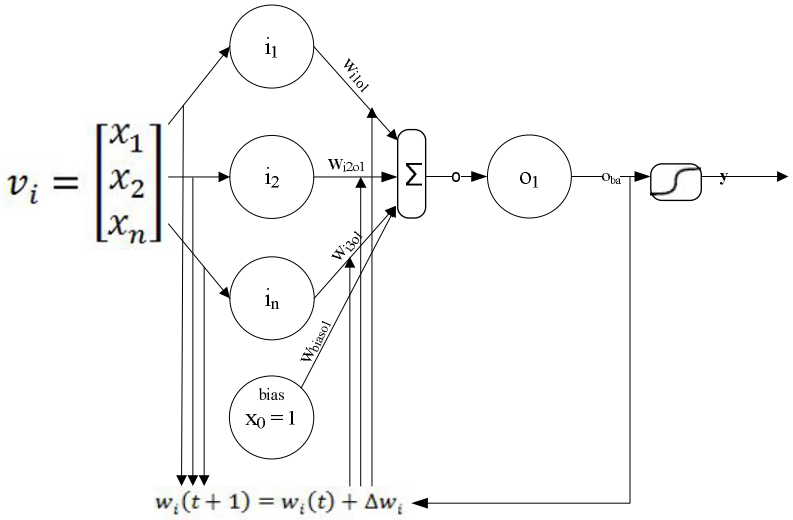

Figure 41 - Delta rule

The Delta rule cannot be modified in a way so that the application within unsupervised learning ANNs is possible.

3.3.3.6 Propagation learning

As the above explained learning rules, all sorts of propagation training use the desired output of an ANN and compare it with the actual output, for propagating the error for the adaption of weights starting from the output neurons back to the input neurons after each iteration, only the way of how this is conducted is slightly different. Therefore, it can only be applied within supervised learning ANNs. Propagation learning includes several forms, but only the three forms of relevance for this elaboration will be discussed in detail. These are

- Back propagation,
- Manhattan update rule, and
- Resilient propagation.[145]

All propagation learning algorithms have in common that the difference between the desired output and the actual output, the error of the ANN, is tried to be improved after each iteration. This means that for each item in one iteration (each loop through the whole amount of

145 Heaton Jeff (2010): Introduction to Encog 2.5 for Java, Rev. 3; Chesterfield: Heaton Research, Inc., p. 87

training data) some change to the weight matrices, but also to the threshold is calculated, precisely in the form of a two-pass process: a forward and backward pass.[146]

With the forward pass, simply the already mentioned transfer of the input data with all operations carried out within the network from the input neurons to the output neurons is meant. Also, as mentioned beforehand, the difference between the desired and the actual output is calculated within this step, which has been described as the error (3.3.3.1 Supervised and unsupervised training) of the network. Additionally, the actual output of the network is stored for the use within the backward pass.

As a second step, within the backward pass, the beforehand stored erratic values are used to calculate the gradient of these errors, which is done by applying the actual output of the artificial neural network to the derivative of the activation function used for this level. The resulting value, as last step, is then multiplied with the error value.[147]

Propagation learning has been of utmost importance in the development of artificial neural network learning approaches, as with the hidden layers within an MLP, the problem of how to calculate the error for this hidden layer arises. This is founded in the fact that the desired output of the overall network can be defined, but not the output for the hidden layer. The error comparison is therefore a more difficult one as when dealing with SLPs and has been solved first with the back propagation learning algorithm.

3.3.3.6.1 Back propagation training

As all of the detailed propagation algorithms, the back propagation one is based on gradient descent, calculated within the error function and the neurons' corresponding weights. This also requires a slightly more complex activation function, as the bipolar one used within the SLP (3.3.2.6.1 Single layer perceptron). For the following explanation, the sigmoid activation function (3.3.1.3 Activation functions) is applied, as it is differentiable:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3-84}$$

or

$$\sigma(x) = \frac{1}{1+exp(-\varphi(x))} \tag{3-85}$$

for which

$$\frac{\delta\sigma(x)}{\delta x} = \sigma(x)\big(1-\sigma(x)\big) \tag{3-86}$$

is valid.[148]

146 Heaton Jeff (2010): Introduction to Encog 2.5 for Java, Rev. 3; Chesterfield: Heaton Research, Inc., p.87

147 Heaton Heaton Jeff (2010): Introduction to Encog 2.5 for Java, Rev. 3; Chesterfield: Heaton Research, Inc., p. 89

148 Kramer Oliver (2009): Computational Intelligence: Eine Einführung; Berlin Heidelberg: Springer-Verlag, p. 130

The difference between the desired and the actual output is usually expressed as MSE (3.3.3.1 Supervised and unsupervised training),[149]

$$mse = \frac{1}{2}\sum_{i=1}^{n} x_i^2 \tag{3-87}$$

or for the application within an ANN

$$mse(w) = \frac{1}{2}\sum_{i=1}^{n}(desired_i - actual_i)^2 \tag{3-88}$$

or for derivation a representation similar to the one Kramer chose:

$$mse(w) = \frac{1}{2}\sum_{d\in D}(desired_d - actual_d)^2 \tag{3-89}$$

where w represents the vector of all network weights and $d \in D$ all possible inputs of the training data set. The factor $\frac{1}{2}$ has the purpose of simplifying the derivation. The next step is the partial derivation ∂ of the function towards their weights, beforehand mentioned as δ-rule (3.3.3.5 Delta rule), as this is the gradient descent of the error function:[150]

$$\frac{\partial mse}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{d\in D}(desired_d - actual_d)^2$$

$$= \sum_{d\in D}(desired_d - actual_d)\frac{\partial}{\partial w_i}(desired_d - actual_d)$$

$$= \sum_{d\in D}(desired_d - actual_d)\left(-\frac{\partial actual_d}{\partial w_i}\right)$$

$$= -\sum_{d\in D}(desired_d - actual_d)\left(\frac{\partial actual_d}{\partial \varphi_d}\frac{\partial \varphi_d}{\partial w_i}\right) \tag{3-90}$$

where φ is a differentiable activation function, and x_i the input x on position i. As a next step, the derivation of the sigmoid function has to be considered:

$$\frac{\partial actual_i}{\partial \varphi_{x_i}} = \frac{\partial(\sigma(\varphi_{x_i}))}{\partial \varphi_{x_i}} = actual_i(1 - actual_i) \tag{3-91}$$

The constants of the weights w are dropped out within the derivation, which leads to

$$\frac{\partial \varphi_d}{\partial w_i} = \frac{\partial w x_d}{\partial w_i} = x_{i,d} \tag{3-92}$$

and finally to

$$\frac{\partial mse}{\partial w_i} = -\sum_{d\in D}(desired_d - actual_d)\, actual_d(1 - actual_d)x_{i,d} \tag{3-93}$$

---

149 Kramer Oliver (2009): Computational Intelligence: Eine Einführung; Berlin Heidelberg: Springer-Verlag, p. 121

150 Kramer Oliver (2009): Computational Intelligence: Eine Einführung; Berlin Heidelberg: Springer-Verlag, p.130 ff.

Kramer also indicates correctly that the weight adaption must have the contrariwise algebraic sign of the differentiation:

$$\Delta w_i = -\mu \frac{\partial mse}{\partial w_i} = \mu \sum_{d \in D} (desired_d - actual_d)\, actual_d (1 - actual_d) x_{i,d} \qquad (3\text{-}94)$$

The weight adaption after presenting one set out of the overall training set is therefore done conducted by

$$\Delta w_i = \mu (desired_d - actual_d) actual_d (1 - actual_d) x_{i,d} \qquad (3\text{-}95)$$

with the delta rule

$$w_i' = w_i + \Delta w_i \qquad (3\text{-}96)$$

or by including a specific point in time $t + 1$

$$w_i(t+1) = w_i(t) + \Delta w_i \qquad (3\text{-}97)$$

with

$$\Delta w_i = \mu \delta x_i \qquad (3\text{-}98)$$

as already described in 3.3.3.5 Delta rule.

Back propagation adapts the weights of the neurons from the output back to the input, requiring also the neurons of the hidden layer(s) to be adapted, which requires the error to be calculated by

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d \in D} (w_{hid d \in D} \delta_{d \in D}) \qquad (3\text{-}99)$$

whereas the calculation errors for input and output layers happens by

$$\delta_{d \in D=} (desired_{d \in D} - actual_{d \in D}) actual_{d \in D} (1 - actual_{d \in D}) \qquad (3\text{-}100)$$

However, additionally to the learning rate, which defines how fast a back propagation artificial neural network learns in a percent value, there is another parameter to consider, which is called momentum, usually a value close to zero, and represented by α. Through the momentum, or term of variable inertia, the last weight change and the gradient are weighted in percent, so that the weight adaption within the current iteration partially depends on the weight adaption of the last iteration. The value zero would cause a weight change only depending from the gradient's value, where a value of 1 would ignore the gradient and just consider the weight change of the last iteration.

Summing up, the weight change between the neurons i and j at the point in time $t + 1$ is carried out by summing up the products of

- the learning rate μ, multiplied by the error (delta) of the neuron j δ_j and the input x of neuron i x_i and
- the momentum α multiplied by the weight change of the last iteration $\Delta w_{ij}(t)$:

$$\Delta w_{ij}(t+1) = \mu \delta_j x_i + \alpha \Delta w_{ij}(t) \qquad (3\text{-}101)$$

As all calculations have been deduced and explained, the back propagation algorithm is as follows:

1. ***Start***
2. Initial selection of network weights w_i at random.
3. ***Repeat***
   a) Calculate the actual outputs of the neurons in the hidden layer

$$o_j(t) = \sigma\left[\sum_{i=1}^{n} x_i(t) * w_{ij}(t) - \theta_i\right]$$

   b) Calculate the actual outputs of the neurons in the output layer

$$o_k(t) = \sigma\left[\sum_{j=1}^{m} x_{jk}(t) * w_{jk}(t) - \theta_i\right]$$

   c) Calculate the error gradient for the neurons in the output layer

$$\delta_k(t) = o_k(t) * [1 - o_k(t)] * e_k(t)$$

   where

$$e_k(t) = t_k(t) - o(t)$$

   d) Calculate the weight corrections

$$\Delta w_{jk}(t) = \mu * o_j(t) * \delta_k(t)$$

   e) Adapt the weights from hidden to output layer according to

$$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk}(t)$$

   f) Calculate the error gradient for the neurons in the hidden layer

$$\delta_j(t) = o_j(t) * \left[1 - o_j(t)\right] * \sum_{k=1}^{l} \delta_k(t) * w_{jk}(t)$$

   g) Calculate the weight corrections

$$\Delta w_{ij}(t) = \mu * o_i(t) * \delta_j(t)$$

   h) Update the weights at the hidden neurons

$$w_{ij}(t+1) = w_{ij}(t) + \alpha \Delta w_{ij}(t)$$

4. ***Until criteria are reached***
5. ***End***

Algorithm 5 - Back propagation algorithm

However, although back propagation has proved to come to desired and correct solutions most of the time, sometimes a problem is that only the gradient, or in other words the local environment in the plane, is known, what might lead to the following problems:

- Local minimum problem
  As only the local environment is known, it is not quite sure if the algorithm found the global minimum or just a local one. This problem occurs especially when the hyper plane of the error term is littered with peaks and lows, which is the case when the networks' dimension is of higher order, or in other words, when the number of connections between the neurons is high. The cliffier a hyper plane is the more local minima it features.
- Flat plateau problem
  When a hyper plane as a whole, or at least partially, features nearly no peaks and lows, also the number of local minima is low. As the gradient descent (3.3.1.2.3 Gradient descent) at the point in time $(t+1)$ depends, according to the momentum α more or less on the gradient of the point in time (t), the gradient at $(t+1)$ might become very small, when the gradient at (t) was small too. Because of this, the next low might not be reached and the descent stagnates.
- Leaving global minimum problem
  The global minimum might be skipped when the expansion of the low in the hyper plane is very small. This leads again to the local minimum problem.
- Oscillation
  Oscillation may happen directly or indirectly. Direct oscillation means it might happen that the gradient does not descend, but heads from the current descent to a neighbor descent of another peak. This forces the gradient to jump back to the point of origin (to the same value with the different sign), which is called to a direct oscillation. Indirect oscillation follows the same principle, with the exception that not one but several steps are needed for getting back to the point of origin.

3.3.3.6.2 Manhattan update rule training

The difference of the Manhattan update rule and back propagation is that the descent at the point in time $(t+1)$ is not changed in dependency of the same gradient's value, but only from its sign. It is therefore considered, if the gradient

- is positive,
- is negative, or
- is close to zero.

Thus, in Manhattan update rule, the value of the gradient's magnitude is used to determine how to update the weight matrix or threshold value:[151]

- If the magnitude is close to zero, then the weight or threshold values remain unchanged.
- If the magnitude is positive, then the weight or threshold value is increased.
- If the magnitude is negative, then the weight or threshold value is decreased.

Manhattan update rule does not require the learning rate or momentum passed over, just the constant for defining the in- or decrease of the weight, which must be provided as parameter Δ:

151 Heaton Jeff (2010): Introduction to Encog 2.5 for Java, Rev. 3; Chesterfield: Heaton Research, Inc., p. 89

$$\Delta w_{ij}^{(t+1)} = \begin{cases} -\Delta & , \quad if \frac{\partial mse^{(t+1)}}{\partial w_{ij}} > 0 \\ +\Delta & , \quad if \frac{\partial mse^{(t+1)}}{\partial w_{ij}} < 0 \\ 0 & , \quad else \end{cases} \tag{3-102}$$

The above equation shows that the weight change at the point in time $(t+1)$ is carried out in dependency of the gradient's sign at $(t+1)$ as well as the constant Δ. The change is propagated back through the network as elucidated in 3.3.3.6.1 Back propagation training (as the detailed back propagation equations have already been outlined, a simplified and more compact notation will be applied for the propagation algorithms to come):

Start

1. Initial selection of network weights w_i at random.
2. ***Repeat***
   a) Calculate the network output for the value $d \in D$
   b) Calculate the error $\delta_{d \in D}$ for each output neuron $d \in D$

$$\delta_{d \in D=}(desired_{d \in D} - actual_{d \in D}) actual_{d \in D}(1 - actual_{d \in D})$$

   c) Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d \in D} (w_{hid d \in D} \delta_{d \in D})$$

   d) Adapt the weights according to

$$\Delta w_{ij}^{(t+1)} = \begin{cases} -\Delta & , \quad if \frac{\partial mse^{(t+1)}}{\partial w_{ij}} > 0 \\ +\Delta & , \quad if \frac{\partial mse^{(t+1)}}{\partial w_{ij}} < 0 \\ 0 & , \quad else \end{cases}$$

3. ***Until criteria are reached***

End

Algorithm 6 - Manhattan update rule

By the application of the Manhattan update rule as training algorithm within an ANN the flat plateau problem is avoided, as, because of its constancy, the gradient or weight change cannot become so small that the next low is not reached.

3.3.3.6.3 Resilient propagation

The difference between resilient propagation and back propagation is that the descent at the point in time $(t+1)$ is not changed in dependency of the same gradient's value, but also only

from its sign, as it is done when applying the Manhattan update rule. It is therefore considered as well, if the gradient

- is positive,
- is negative, or
- is close to zero.

Thus, in resilient propagation, the value of the gradient's magnitude is used to determine how to update the weight matrix or threshold value, equal to the Manhattan update rule:[152]

- If the magnitude is close to zero, then the weight or threshold values remain unchanged.
- If the magnitude is positive, then the weight or threshold value is increased.
- If the magnitude is negative, then the weight or threshold value is decreased.

However, in difference to the Manhattan update rule, resilient propagation does not require the setting of the constant Δ used for the weight adaption. Rather than using a fixed constant to update the weights and threshold values, the deltas do not remain fixed, like in the Manhattan update rule or back propagation algorithm, as they change as training progresses.[153] The calculation of resilient propagation's weight adaption is calculated similarly to Manhattan update rule, except the Δ-value is not static, but calculated:[154]

$$\Delta w_{ij}^{(t+1)} = \begin{cases} -\Delta_{ij}^{(t+1)} & , \quad if \frac{\partial mse^{(t+1)}}{\partial w_{ij}} > 0 \\ +\Delta_{ij}^{(t+1)} & , \quad if \frac{\partial mse^{(t+1)}}{\partial w_{ij}} < 0 \\ 0 & , \quad else \end{cases} \tag{3-103}$$

The determination of the new update-values $\Delta_{ij}^{(t+1)}$ is sign-dependent:

$$\Delta_{ij}^{(t+1)} = \begin{cases} \mu^{+} * \Delta_{ij}^{(t)} & , \quad if \frac{\partial mse^{(t)}}{\partial w_{ij}} * \frac{\partial mse^{(t+1)}}{\partial w_{ij}} > 0 \\ \mu^{-} * \Delta_{ij}^{(t)} & , \quad if \frac{\partial mse^{(t)}}{\partial w_{ij}} * \frac{\partial mse^{(t+1)}}{\partial w_{ij}} < 0 \\ \Delta_{ij}^{(t)} & , \quad else \end{cases} \tag{3-104}$$

where $0 < \mu^{-} < 1 < \mu^{+}$.

When the partial derivative $\frac{\partial mse^{(t+1)}}{\partial w_{ij}}$ of the corresponding weight Δw_{ij} performs a sign change, the update value is $\Delta_{ij}^{(t+1)}$ is decreased in case of an increase by the factor $\mu^{+} = 1.2$ and in case of a decrease by $\mu^{-} = 0.5$. A change of the sign indicates that the last update value was too high and a local minimum was jumped over. However, when the weight does not change, the update value is increased to allow accelerated convergence in shallow regions.

152 Heaton Jeff (2010): Introduction to Encog 2.5 for Java, Rev. 3; Chesterfield: Heaton Research, Inc., p. 89

153 Heaton Jeff (2010): Introduction to Encog 2.5 for Java, Rev. 3; Chesterfield: Heaton Research, Inc., p. 89

154 Mache Niels: RPROP [2011-31-08]; Hong Kong Polytechnic University; URL: http://www.eie.polyu.edu.hk/~enzheru/snns/SNNSinfo/UserManual/node152.html#man

[155] When the sign changes, the weight shall not be changed, which is usually done by $\frac{\partial mse^{(t)}}{\partial w_{ij}} := 0$. The weight change is propagated back through the network as elucidated in 3.3.3.6.1 Back propagation training:

Start

1. Initial selection of network weights w_i at random.
2. ***Repeat***
   a) Calculate the network output for the value d ∈ D
   b) Calculate the error $\delta_{d\in D}$ for each output neuron d ∈ D

$$\delta_{d\in D=}(desired_{d\in D} - actual_{d\in D})actual_{d\in D}(1 - actual_{d\in D})$$

   c) Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid})\sum_{d\in D}(w_{hidd\in D}\delta_{d\in D})$$

   d) Adapt the weights according to

$$\Delta w_{ij}^{(t+1)} = \begin{cases} -\Delta_{ij}^{(t+1)} & , \quad if \frac{\partial mse^{(t+1)}}{\partial w_{ij}} > 0 \\ +\Delta_{ij}^{(t+1)} & , \quad if \frac{\partial mse^{(t+1)}}{\partial w_{ij}} < 0 \\ 0 & , \quad else \end{cases}$$

with

$$\Delta_{ij}^{(t+1)} = \begin{cases} \mu^{+} * \Delta_{ij}^{(t)} & , \quad if \frac{\partial mse^{(t)}}{\partial w_{ij}} * \frac{\partial mse^{(t+1)}}{\partial w_{ij}} > 0 \\ \mu^{-} * \Delta_{ij}^{(t)} & , \quad if \frac{\partial mse^{(t)}}{\partial w_{ij}} * \frac{\partial mse^{(t+1)}}{\partial w_{ij}} < 0 \\ \Delta_{ij}^{(t)} & , \quad else \end{cases}$$

3. ***Until criteria are reached***

End

Algorithm 7 - Resilient propagation

By the application of resilient propagation as training algorithm within an ANN

– the flat plateau problem is avoided, as, because of its constancy, the gradient or weight change cannot become so small that the next low is not reached and

155 Riedmiller Martin et al. (1993): A direct adaptive method for faster back-propagation learning: The Rprop algorithm; Proceedings of the IEEE International Conference on Neural Networks, IEEE Press: 586-591

– oscillation is avoided as the update value is decreased when the sign changes.

3.3.3.7 Genetic learning (NeuroEvolution)

Genetic algorithms (GA) belong to the class of metaheuristic algorithms and are used for solving optimization problems or for training an artificial neural network (which, in fact, is an optimization problem as well). Due to their characteristic of performing the search with populations of solutions, these algorithms have an intrinsic parallelism where many different possibilities are explored simultaneously.[156] GAs are adaptive, robust algorithms and particularly useful for applications that require search and optimization. They are population-based algorithms for which any communication and interaction are carried out within the population and therefore, they possess a high degree of implicit parallelism.[157] When talking about optimization problems, especially NP-hard problems, in other words problems that cannot be solved in polynomial time even by the fastest supercomputers, can be solved relatively quick with GAs. Above all, there is no unified explanation of what genetic algorithms exactly are. However, there are certain, in the field generally accepted elements all descriptions of GAs have in common:[158]

– a population of chromosomes encoding (in string form) candidate solutions to the problem in hand,
– a mechanism for reproduction,
– selection according to fitness, and
– genetic operators.

In artificial neural networks making use of genetic algorithms (= evolutionary artificial neural networks - EANN), evolution can be introduced at various levels, starting from weight evolution, proceeding to architecture adaption and leading to the evolution of the learning mechanism.[159, 160] As in nature, genetic algorithms use natural selection which is both important when trying to solve optimization problems and training ANNs. Mostly, training of an ANN with a GA is a search optimization problem, as the best solution out of a pool of solutions is tried to be found. Genetic algorithms represent possible solutions to a problem as chromosomes, and the sum of the chromosomes as population. Some chromosomes might represent fairly good solutions, some others not. If a solution is good or bad has to be determined by a so-called fitness function. The fitness function constitutes the measure of fitness (adaptation) of a given individual in the population, which allows to evaluate the degree of fitness of particular individuals in a population, and based on this degree select the individuals that are the best fit (that is, having the highest fitness function), in accordance

156 Abraham Ajith, Hassanien Aboul-Ella, Siarry Patrick, Engelbrecht Andries (2009): Foundations of Computational Intelligence Volume 3 Global Optimization; Berlin Heidelberg: Springer-Verlag, p. 426
157 Jain Lakhmi C. (2008): Computational Intelligence Paradigms: Innovative Applications; Berlin Heidelberg: Springer-Verlag, p. 4
158 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 883
159 Abraham Ajith, Crina Grosan, Pedrycz Witold (2008): Engineering Evolutionary Intelligent Systems; Berlin Heidelberg: Springer-Verlag, p. 5
160 Yao X. (1999): Evolving neural networks; Proceedings of the IEEE 87(9), 1423–1447

with the evolutionary principle of the survival of 'the strongest' (the best fit) ones.[161] These chromosomes then receive the 'privilege' for recombination. In the context of ANNs, the fitness function is the error function, in game theory it can be the cost function and in optimization problems it is the objective function trying to determine a minimum or maximum.

A chromosome consists of genes, which are parts of the overall solution. In case of ANN training, the genes are the network's weights, a chromosome is a complete ANN and the population is the overall amount of all ANNs representing a possible solution to a problem. An iteration, or the evolution of a generation for finding a suitable feed-forward ANN for solving a problem proceeds as follows:

Start

1. Creation of initial population.
2. Randomization of weights and threshold values of each chromosome.
3. ***Repeat***
   a) Calculate the network output for the value d ∈ D
   b) Evaluate the fitness of each chromosome:
      i. Calculate the error $\delta_{d \in D}$ for each output neuron d ∈ D

$$\delta_{d \in D=}(desired_{d \in D} - actual_{d \in D}) actual_{d \in D}(1 - actual_{d \in D})$$

      ii. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d \in D} (w_{hidd \in D} \delta_{d \in D})$$

   c) Selection of chromosomes to recombine
   d) Repeat
      i. Crossover of chromosomes
      ii. Mutation of offspring
   e) Until all selected chromosomes are recombined
4. ***Until criteria are reached***
5. Creation of new population

End

Algorithm 8 - Genetic algorithm

Step 3. a) i. to a) iii. in the algorithm is to put on a level with selection in nature and mating and mutation are conducted in step 3. a) iv. Additionally to the already known calculation of the ANN error, the application of genetic operators, which are the mutation of offspring (chromosomes of the population at the point in time $(t + 1)$) and the recombination (mating of the chromosomes selected by the application of the fitness function at (t)), is, according to an example, done as follows:

161 Rutkowski Leszek (2008): Computational Intelligence Methods and Techniques; Berlin Heidelberg: Springer-Verlag, p. 268

The chromosomes

$$chr_1 = [10010|01110|00011] \quad (3\text{-}105)$$

and

$$chr_2 = [10011|11110|11100] \quad (3\text{-}106)$$

are crossed over, or mated and lead to the offsprings

$$ofs_1 = [10010|11110|00011] \quad (3\text{-}107)$$

and

$$ofs_2 = [10011|01110|11100] \quad (3\text{-}108)$$

Both chromosomes consist of 15 genes, and for the recombination in this example two lines were drawn to divide the each chromosome into 3 equally-sized parts, which are then recombined.

In biological systems clones are not exact copies of the parent cell because some mutations are in effect. In artificial systems, clones are also not exact copies of the parent cell or neuron, and therefore some mutation must be in effect as well. The clone, to be a true clone, must have the same parents, e.g. input signals.[162] At first, the mutation operator is applied on the offspring, which leads to

$$ofs_1 = [10110|11110|00011] \quad (3\text{-}109)$$

and

$$ofs_2 = [10011|01110|11110] \quad (3\text{-}110)$$

Both recombination and mutation are carried out with a probability of a predefined percentage. The probability for recombination is calculated for each chromosome, and the mutation probability for each single gene. At recombination, the selected chromosomes are summarized as pairs, and for each pair of chromosomes a random number between a range (usually between 0 and 1) is calculated. When this value falls below the recombination percentage value, the chromosomes are mated:

$$chr_{12mate}^{(t+1)} = \begin{cases} recombine\ chr_1 and\ chr_2\ at\ crossover\ lines & ,\ if\ v_{chr12}^{(t)} < p_{rec} \\ do\ nothing & ,\ if\ v_{chr12}^{(t)} > p_{rec} \end{cases} \quad (3\text{-}111)$$

where p_{rec} represents recombination percentage value. If the random value for recombination $v_{chr12}^{(t)}$ falls below, it is carried out. The same holds for the mutation of single genes, only that every gene receives a random number for mutation when changing gene values by a function

162 Abraham Ajith, Hassanien Aboul-Ella, Snáel Vaclav (2009): Foundations of Computational Intelligence Volume 5: Function Approximation and Classification; Berlin Heidelberg: Springer-Verlag, p. 364

and that every chromosome receives the random number when shuffling genes within this chromosome. The following example shows the mutation of an offspring chromosome by shuffling its genes:

$$o_{1mut}^{(t+1)} = \begin{cases} \varphi(o_1) & , \ if\ v_{o1mut}^{(t)} < p_{mut} \\ do\ nothing & , \ if\ v_{o1mut}^{(t)} > p_{mut} \end{cases} \quad (3\text{-}112)$$

where

$$\varphi(o_1) = \{switch\ g_{rand1}\ with\ g_{rand2} \quad , \ if\ v_{o1mut}^{(t)} < p_{mut} \quad (3\text{-}113)$$

$\varphi(o_1)$ determines the length of the offspring chromosome and two random numbers within this length, at which the values are switched. The examples show how to mutate and recombine chromosomes containing only bit-values. It works the same when applying it to an ANN, as all network weights or genes, stored to an array, represent a possible solution or chromosome.

Genetic algorithms are not limited to the above mentioned operators or general structure. Recent research proposed transgenetic algorithms, applying horizontal or lateral gene transfer as well as endosymbiosis.[163] Further, genetic data mining approaches may also make use of gene grouping and additional operators, like

- the search for outliers,
- the random creation of new gene groups out of existing ones,
- the merging of strong gene groups, or
- the search for groups with a high member variability for scattering these members across the rest.[164]

As mentioned at the beginning of the subchapter, there are three possible evolution strategies an ANN might go through:

3.3.3.7.1 Evolutionary search of connection weights

The evolutionary adaption of weights is the solution which was used to explain genetic algorithms. Several chromosomes, which are possible ANN solutions with different weights, are created and evaluated against the optimal solution during the training phase.

3.3.3.7.2 Evolutionary search of architectures

Evolutionary architecture adaption can be achieved by constructive and destructive algorithms, where constructive ones add complexity to an ANN started with a simple structure and destructive algorithms remove the same from an ANN.[165,166,167,168] By adding and

163 Abraham Ajith, Hassanien Aboul-Ella, Siarry Patrick, Engelbrecht Andries (2009): Foundations of Computational Intelligence Volume 3 Global Optimization; Berlin Heidelberg: Springer-Verlag, p. 429

164 Abraham Ajith, Hassanien Aboul-Ella, Snáel Vaclav (2009): Foundations of Computational Intelligence Volume 4: Bio-Inspired Data Mining; Berlin Heidelberg: Springer-Verlag, p. 257

165 Frean M. (1990): The upstart algorithm: a method for constructing and training feed-forward neural networks; Neural Computation 2: 198–209

removing is meant that new solutions with more or less hidden neurons or layers are created to verify if the specified minimum error rate can be achieved through these.

3.3.3.7.3 Evolutionary search of learning rules

The evolutionary search for the suitable learning rule can be carried out by the initial creation of several populations of artificial neural networks each population making use of different learning strategies. The results are then compared in terms of generations or minimum error.[169]

3.3.3.8 Simulated annealing

Simulated annealing (SA) also belongs to the group of metaheuristic algorithms, suitable for solving optimization or search problems. In physics, the term 'annealing' refers to the very slow cooling of gas or metal into a crystalline solid of minimum energy configuration.[170] The atoms of such materials have very high energy values at very high temperatures, which gives the atoms a great deal of freedom in their ability to restructure themselves.[171] The energy values of such materials decreases during cooling down. When the ideal speed (continuous temperature reduction) has been found, the material will be stable in its structure and more consistent then if cooling it down too quick. Simulated annealing[172] algorithms simulate this behavior, which can also be applied within ANN training. The simulated annealing algorithm always works with two solutions, the first being the best one having been achieved until the point in time $(t+1)$, represented by $C(S')$, and the second being the one currently created and compared with the first one, represented by $C(S)$:

$$\Delta_{C(S')C(S)} = C(S') - C(S) \quad \text{(3-114)}$$

If the second one performs better than the first one, thus the outcome of the above equation is positive, it is used to replace the latter as the current best one. In some cases, simulated annealing also makes use of a probability for determining when to replace a solution with a better one. Therefore, in some implementations a better new solution might not always replace the actual one:

$$\Delta_{repC(S')C(S)} = min\left(1, e^{-\Delta_{C(S')C(S)}/T}\right) \quad \textit{(3-115)}$$

166 Mascioli F. et al. (1995): A constructive algorithm for binary neural networks: the oil spot algorithm; IEEE Trans Neural Netw 6(3): 794–797

167 Omlin C. W. et al. (1993): Pruning recurrent neural networks for improved generalization performance; Technical report No 93-6, CS Department, Rensselaer Institute, Troy, NY

168 Stepniewski S. W. et al. (1997): Pruning back-propagation neural networks using modern stochastic optimization techniques; Neural Comput Appl 5: 76–98

169 Abraham Ajith, Crina Grosan, Pedrycz Witold (2008): Engineering Evolutionary Intelligent Systems; Berlin Heidelberg: Springer-Verlag, p. 8

170 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 909

171 Heaton Jeff (2008): Introduction to Neural Networks for Java, 2nd ed.; Chesterfield: Heaton Research, Inc., p. 199

172 Kirkpatrick Scott et al. (1983): Optimization by simulated annealing; Science, 220(4598): 671–680

where T represents the current temperature, meaning the value that influences the change of weights within a weight matrix of an ANN:

$$\varphi_{C(s')C(S)} = T * r \qquad (3\text{-}116)$$

The ratio $\varphi_{C(s')C(S)}$ for changing the weights within an ANN is calculated by multiplying the temperature T by a random number r. The higher the temperature is, the higher is the probability of a high weight change. Changes with a specific temperature are carried out as long as a predefined number of iterations (cycles) has not been reached. After having fulfilled the last iteration of a temperature, the algorithm verifies if the lowest, predefined temperature has been reached. If not, the temperature will be lowered by either a constant or by logarithmically decreasing it by a ratio between a beginning and an ending temperature, as the following equation shows:[173]

$$\varphi_{C(s')C(S)} = T * e^{\frac{ln\left(\frac{s}{e}\right)}{c-1}} \qquad (3\text{-}117)$$

The variable s represents the starting temperature, e in this case the ending temperature. c represents the cycle count. The above equation calculates a ratio that should be multiplied by the current temperature T, which produces a change that will cause the temperature to reach the ending temperature in the specified number of cycles.

An iteration, or generation, of finding suitable weights for a feed-forward ANN looks as follows:

Start

1. Randomize weights.
2. ***Repeat***
   a) Calculate the network output for the value d ∈ D
   b) Evaluate the fitness of each neuron:
      i. Calculate the error $\delta_{d\in D}$ for each output neuron d ∈ D

$$\delta_{d\in D=}(desired_{d\in D} - actual_{d\in D})actual_{d\in D}(1 - actual_{d\in D})$$

      ii. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d\in D}(w_{hidd\in D}\delta_{d\in D})$$

   c) Repeat
      i. Create new ANN and randomize weights according to T
      ii. Calculate the error $\delta_{d\in D}$ for each output neuron d ∈ D
         – $\delta_{d\in D=}(\text{desired}_{d\in D} - \text{actual}_{d\in D})\text{actual}_{d\in D}(1 - \text{actual}_{d\in D})$
      iii. Calculate the error δ_{hid} for each hidden neuron hid
         – $\delta_{hid} = \text{actual}_{hid}(1 - \text{actual}_{hid}) \sum_{d\in D}(w_{hidd\in D}\delta_{d\in D})$
      iv. Compare solutions according to

173 Heaton Jeff (2008): Introduction to Neural Networks for Java, 2nd ed.; Chesterfield: Heaton Research, Inc., p. 205

$$\Delta_{C(S')C(S)} = C(S') - C(S)$$

v. If $C(S')$ is better than $C(S)$, replace $C(S)$

d) Until max tries for current temperature reached

e) Decrease temperature by

$$\varphi_{C(S')C(S)} = T * e^{\frac{ln\left(\frac{S}{e}\right)}{c-1}}$$

3. Until lower temperature bound is reached

End

Algorithm 9 - Simulated annealing algorithm

3.3.3.9 NeuroEvolution of augmenting topologies (NEAT)

As SA and GA-learning, NEAT belongs to the group of metaheuristic algorithms. However, NEAT does not only adapt the weights of ANNs within a population, it furthermore mutates the architecture and is capable of recombining ANNs with different architectures. The NEAT method of evolving ANNs combines the usual search for appropriate network weights with complexification of the network structure and consists of solutions to three challenges in evolving neural network topology:[174]

– Crossover of disparate ANN topologies
– Protection of topological innovation through speciation
– Minimization of topologies for detection of most efficient solutions

As NEAT is a form of a genetic algorithm, it supports mutation and recombination. As already indicated, the mutation is applied both to connection weights and the ANN architecture. The mutation of the connection weights happens as explained in 3.3.3.7.1 Evolutionary search of connection weights, but the mutation of the architecture may happen through

– adding of a connection between two neurons, or
– adding of a neuron

to the ANN's architecture. The new connection can be added between two neurons that have not been connected beforehand. If, however, a new neuron has been added, the active connection between two neurons is split, meaning that the initial connection is disabled and two new ones are added.

The connection between the first node in the chain and the new node is given a weight of one, and the connection between the new node and the last node in the chain is given the same weight as the split connection, which introduces a nonlinearity (e.g. sigmoid function) where

174 Stanley O. Kenneth (2004): NeuroEvolution of Augmenting Topologies [2011-12-12]; Carnegie Mellon School of Computer Science; URL: http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume21/stanley04a-html/node3.html

there was none before, which changes the function only slightly, and the new neuron is immediately integrated into the network.[175] The NEAT mutations can be explained best with Figure 42 – NeuroEvolution of augmenting topologies mutation.

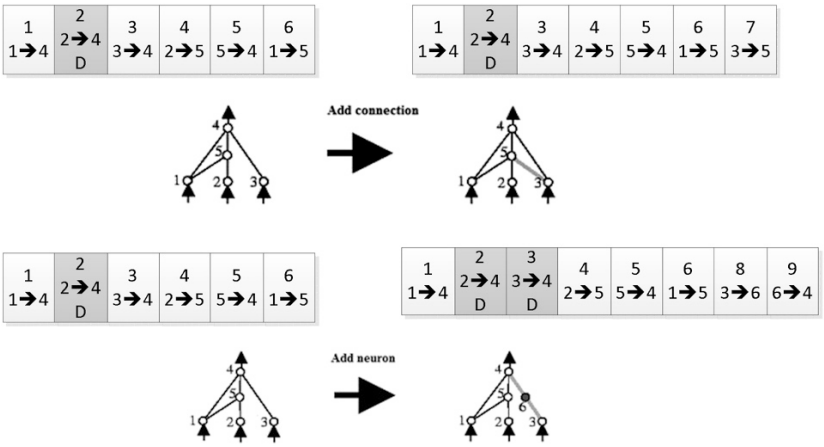

Figure 42 – NeuroEvolution of augmenting topologies mutation[176]

Furthermore, genomes (chromosomes) receive innovation numbers, which are historical markers that identify the original historical ancestor of each gene. New genes are assigned new increasingly higher numbers. In adding a connection, a single new connection gene is added to the end of the genome and given the next available innovation number. In adding a new node, the connection gene split is disabled, and two new connection genes are added to the end the genome. The new node is between the two new connections. A new node gene (not depicted) representing this new node is added to the genome as well (Figure 43 – NeuroEvolution of augmenting topologies recombination of different topologies).[177]

175 Stanley O. Kenneth (2004): NeuroEvolution of Augmenting Topologies [2011-12-12]; Carnegie Mellon School of Computer Science; URL: http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume21/stanley04a-html/node3.html

176 Stanley O. Kenneth (2004): NeuroEvolution of Augmenting Topologies [2011-12-12]; Carnegie Mellon School of Computer Science; URL: http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume21/stanley04a-html/node3.html

177 Stanley O. Kenneth et al. (2002): Evolving Neural Networks through Augmenting Topologies; Evolutionary Computation 10(2): 99-127

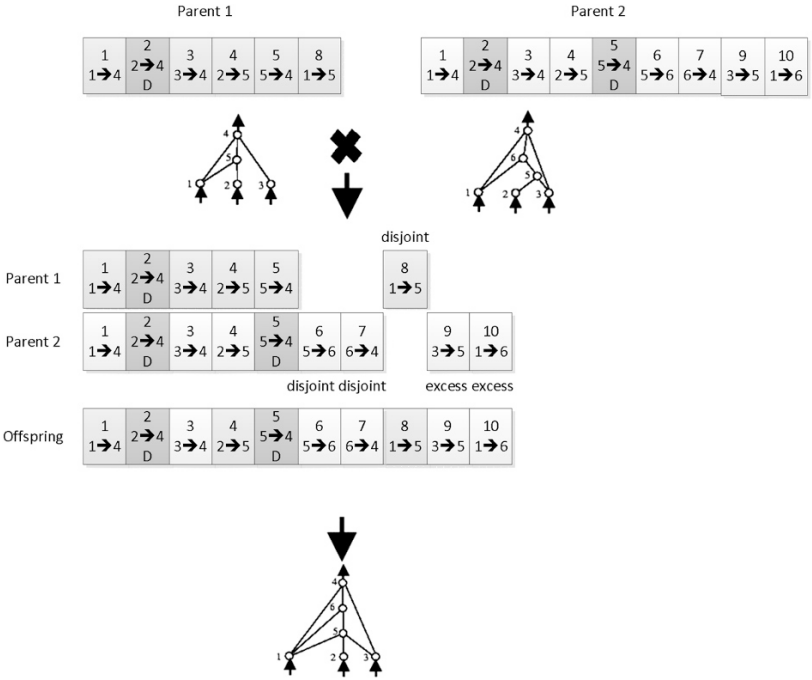

Figure 43 – NeuroEvolution of augmenting topologies recombination of different topologies[178]

NEAT therefore knows, which genes of both parent chromosomes match up and which do not. When a gene of a parent genome with its innovation number is available within the innovation number range of the other parent, it is disjoint, otherwise it is excess. NEAT speciates the population so that individuals compete primarily within their own niches instead with the population at large. This way, topological innovations are protected and have time to optimize their structure before they have to compete with other niches in the population. In addition, speciation prevents bloating of genomes: Species with smaller genomes survive as long as their fitness is competitive, ensuring that small networks are not replaced by larger ones unnecessarily. Protecting innovation through speciation follows the philosophy that new ideas must be given time to reach their potential before they are eliminated. Historical markings make it possible to divide the population into species based on topological similarity:

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 * \overline{W} \qquad (3\text{-}118)$$

178 Stanley O. Kenneth (2004): NeuroEvolution of Augmenting Topologies [2011-12-12]; Carnegie Mellon School of Computer Science; URL: http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume21/stanley04a-html/node3.html

where δ is the distance between two network encodings as a linear combination of the number of excess E and disjoint D genes, as well as the average weight differences of matching genes $\overline{W}$. The coefficients c_1, c_2 and c_3 adjust the importance of the three factors, and the factor N, the number of genes in the larger genome, normalizes for genome size. The reproduction mechanism for NEAT is called explicit fitness sharing, where organisms in the same species must share the fitness of their niche. Thus, a species cannot afford to become too large even if many of its organisms perform well. Therefore, any one species is unlikely to take over the entire population, which is crucial for speciated evolution to maintain topological diversity:

$$f_i' = \frac{f_i}{\sum_{j=1}^{n} sh(\delta(i,j))} \qquad (3\text{-}119)$$

where the fitness f_i' for the chromosome i of a species is calculated according to its distance δ from every other organism j in the population. The sharing function sh is set to 0 when the distance $\delta(i,j)$ is above the threshold; otherwise it is set to 1.[179]

3.3.4 Stability-plasticity dilemma

The stability-plasticity dilemma is the problem of ANNs to retain old memories resulting from new inputs.

- Plasticity concerns the adaption to any change in the input environment, whereas
- stability concerns the preservation of previously learned knowledge.

The ANN weights have to be flexible enough to accommodate the new knowledge (plasticity) but not so much so as to lose the old ones (stability).[180] Some ANN structures minimizing the effects of this dilemma discussed in this book are described in detail in chapter 4. There have been several contributions published describing solutions for the SP-dilemma, and one that focuses on adaption is adaptive incremental learning (AIL) which seeks to deal with data arriving over time or with (static) huge amounts of data that exceed the storage capacities. Thus, processing of data at once is not feasible.[181] AIL algorithms include

- adaptive resonance theory (ART) ANNs 182 (these are of vital significance for the comparison of artificial and biological neural networks at 10.3 Self-organization and 10.3.2.3 Adaptive Resonance Theory),
- fuzzy ARTMAP,183,184

---

179 Stanley O. Kenneth (2004): NeuroEvolution of Augmenting Topologies [2011-12-12]; Carnegie Mellon School of Computer Science; URL: http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume21/stanley04a-html/node3.html

180 Sasu Lucian Mircea (2006): Computational Intelligence Techniques in Data Mining; PhD Thesis; Brasov: Transylvania University of Brasov, p. 45

181 Hassanien Aboul-Ella, Abraham Ajith, Vasilakos Athanasios V., Witold Pedrycz (2009): Foundations of Computational Intelligence Volume 1: Learning and Approximation; Berlin Heidelberg: Springer-Verlag, p. 238 ff.

182 Adeli Hojjat, Hung Shih-Lin (1995):Machine Learning Neural Networks, Genetic Algorithms and Fuzzy Systems; John Wiley and Sons, New York

183 Cheng Tai W., Goldgof Dimitry, Hall Lawrence (1998): Fast fuzzy clustering. Fuzzy Sets and Systems 93, 49–56

– nearest generalized exemplar,185
– generalized fuzzy min-max neural networks,186
– growing neural gas,187,188
– and incremental learning based on function decomposition.[189]

Especially the ART ANN, which has not been discussed within this chapter (but at 10.3.2.3 Adaptive Resonance Theory), produces a stable network that can learn new data. The main idea behind this is to spare some of the output units for new patterns. When the input and a stored pattern are sufficiently similar, they are said to resonate. When there is not sufficient similarity, a new class of patterns is formed utilizing the unused output units. There is no response from the network when all output units are used.[190]

3.4 Summary

Computational intelligence is the imitation of nature for letting software solve and learn problem statements. As stated at the beginning machines shall, by the use of computationally intelligent programs, become able to do things usually humans can do better. CI is a field within the artificial intelligence, which pursues the empowerment of software systems (machines) to

– understand,
– behave,
– adapt,
– collect information,
– plan, reason, solve problems, think abstractly, comprehend ideas and language, and learn[191]

AI as a whole tries to make systems behave like humans do, whereas computational intelligence relies on evolutionary approaches to solve, amongst others, problems suitable for computers, like detecting similarities in huge data amounts or optimization problems. Within

184 Eschrich S., Ke J., Hall L., Goldgof D. (2003): Fast accurate fuzzy clustering through data reduction. IEEE Trans. Fuzzy Systems 11, 262–270
185 Yang M.-S., Ko C.-H. (1996): On a class of fuzzy c-numbers clustering procedures for fuzzy data. Fuzzy Sets and Systems 84(1), 49–60
186 Dave R.N., Krishnapuram R. (1997): Robust clustering methods: a united view. IEEE Trans. Fuzzy Systems 5, 270–293
187 Cover T.M., Hart P.E. (1967): Nearest neighboring pattern classification. IEEE Trans. Information Theory 13, 21–27
188 Eschrich S., Ke J., Hall L., Goldgof D. (2003): Fast accurate fuzzy clustering through data reduction. IEEE Trans. Fuzzy Systems 11, 262–270
189 Bezdek James (1980): A convergence theorem for the fuzzy isodata clustering algorithms. IEEE Trans. Pattern Analysis and Machine Intelligence 2, 1–8
190 Vasantha Kalyani D., Rajasekaran Sundaramoorthy (2009): Pattern Recognition Using Neural and Functional Networks; Berlin Heidelberg: Springer-Verlag, p. 27
191 Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag, p. 7

the field of AI robotics, CI approaches find application for ensuring robust control, planning, and decision making.[192,193,194,195,196]

CI techniques have experienced tremendous theoretical growth over the past few decades and have also earned popularity in application areas e.g. control, search and optimization, data mining, knowledge representation, signal processing, and robotics.[197] Computational intelligence therefore is of inestimable value in the field of data mining, as through its paradigms machines can learn and apply manifold solution types, including classics like

- classification
- (time-Series) prediction, and
- clustering.

From what we have seen in this chapter, the field of computational intelligence, comprising also artificial neural networks, offers interesting ideas which will form the foundation for the search for possible implementation scenarios regarding an artificial mind. However, the introduced artificial neural network structures seem to be too simple for being able to process what we ought to process. Thus, within the next chapter more sophisticated and powerful ANN structures will be introduced, capable of solving more complex problem statements. One of these has even been inspired by the signal processing happening in the brain when a conscious experience occurs.

192 Liu Dikai, Wang Lingfeng, Tan Kay Chen (2009): Design and Control of Intelligent Robotic Systems; Berlin Heidelberg: Springer-Verlag, p. 2

193 Nolfi Stefano, Floreano Dolfi (2000): Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines; Bradford Books

194 Jain Lakhmi C. (1998): Soft Computing for Intelligent Robotic Systems: Physica-Verlag

195 Watanabe Keigo, Hashem M.M.A. (2004): Evolutionary Computations: New Algorithms and Their Applications to Evolutionary Robotics; Heidelberg: Springer-Verlag

196 Teshnehlab M., Watanabe K. (1999): Intelligent Control Based on Flexible Neural Networks (Intelligent Systems, Control and Automation: Science and Engineering); Dordrecht: Kluwer Academic Publishers

197 Liu Dikai, Wang Lingfeng, Tan Kay Chen (2009): Design and Control of Intelligent Robotic Systems; Berlin Heidelberg: Springer-Verlag, p. 2

4 Advanced artificial perception and pattern recognition

Of course, any artificial neural network structure has been inspired by the inner workings of the human brain, but some have been even more so, in the sense that not only how neurons single and clusters of neurons function is important, but also how cortices (e.g. the visual cortex) processes real-world data and which different types of cells are used and why. In the later chapters data representation will be discussed in some more detail, and we will see that how data is presented to a machine learning algorithm is at least as important as the algorithm itself.

Over the last years deep architectures, thus artificial neural networks comprising more than two hidden layers, have been widely discussed (and earned both praise and criticism). It has been emphasized that deep architectures are especially suitable for learning and processing complex data structures, such as images or sound, whereby with images I do not only refer to images as we humans perceive it in our everyday-life. I am referring to point-clouds or n-dimensional representations of real-world images in terms of vectors, matrices or tensors, which only make deep architectures that successful. Whether or not the inner workings in our brain represent images in the same way, I want to emphasize that when we are developing solutions in AI for solving real-world problems we do not only try to imitate how humans or animals process data, perceive or behave. Very often AI-software achieves tasks such as object recognition in a completely different way, starting with potentially different representation of data.

The challenge one has to deal with so-called deep learning ANNs is that training is difficult and time-consuming. This is on the one hand because a lot of the learning theory behind deep architectures has not yet been understood, and on the other hand because of algorithmic unpredictability, such as randomness in the weight initialization. What is more is that most existing learning algorithms have not been developed for deep architectures and cannot be applied to such structures successfully; e.g. back propagation suffers heavier from the classical problems (see 3.3.3.6.1 Back propagation training) the deeper an architecture is, plus additionally from vanishing or exploding gradients. Therefore, one question we must ask when designing algorithms for training deep artificial neural structures is whether there are more advanced ways of representing and presenting data. The purpose of this chapter is to go into detail with some forms of data representation and also to show that we can apply some of nature's optimization strategies for learning although their "real" purpose is something completely different (e.g. an immune system fighting pathogens).

Anyway, in terms of learning (training) genetic or thermal algorithms such as simulated annealing (see 3.3.3.8 Simulated annealing) or completely different approaches such as convolutional, deep belief networks or similar ones (4 Advanced artificial perception and pattern recognition) tackle the problem. This is of utmost importance, as the human brain is a deep neural network structure after all and apparently this has not negatively influenced its capabilities. Some of following artificial neural network structures have been invented and specified within the research phase of the SHOCID[198] project.

198 Neukart Florian (2013): SYSTEM APPLYING HIGH ORDER COMPUTATIONAL INTELLIGENCE IN DATA MINING AND QUANTUM COMPUTATIONAL CONSIDERATIONS

F. Neukart, *Reverse Engineering the Mind*, AutoUni – Schriftenreihe 94, DOI 10.1007/978-3-658-16176-7_4

4.1 Convolutional artificial neural networks

Convolutional ANNs (CNNs) are, as standard ANNs, biologically inspired, but even more so. From the research conducted by Hubel and Wiesel[199] regarding visual perception of cats we learned that the visual cortex is a highly complex arrangement of cells. Not surprisingly, the convolutional neural network has deeply influenced the field of computer vision. The cells in the visual cortex react to signals from the receptive field, small tiled sub-areas the making up the entire visual field. These sub-areas function as in situ (local) filters leveraging the strong spatial correlation occurring in natural images. Three different kinds of cells have been identified:

- Layer 4 cells, whose receptive fields are round like those of LGN (before reaching the primary visual cortex, fibers of the optic nerve make a synapse in a part of the thalamus called the lateral geniculate nucleus [LGN]) and ganglion cells (a ganglion is a nerve cell cluster or a group of nerve cell bodies located in the autonomic nervous system),
- simple cells with elongated receptive fields and thus maximally activated by a line of a particular orientation, and
- complex cells whose receptive fields are similar to those of simple cells except that the line can lie over a larger area of the retina and these fire more to moving lines.[200]

Thus, complex cells seem to feature far bigger receptive fields than simple cells and thus successfully tackle the position invariance in a picture, and simple cells seem to strongly react to edges. The urge to understand the inner workings of the animal visual cortex has inspired numerous researchers, also from the field of AI, which resulted in numerous models (not necessarily strongly related to artificial neural networks), such as NeoCognitron, [201] HMAX[202] and LeNet-5,[203] which will be explained in further detail.

CNNs share some properties with self-organizing feature maps (3.3.2.9.4 Self-organizing feature map), in the sense that the order of the input vector elements not only matters, but is crucial. Other ANN architectures such as feed forward ANNs do not care about the order of the attributes are presented via the input vector. In terms of SOMs the representation of data in a grid (or a higher-dimensional structure – SOMs are not restricted to 2D-space) has shown remarkable success in clustering images, as pixels close to each other are also important to each other. So the order of how the pixels of an image are presented does matter. Nevertheless, apart from the fact that SOMs are clustering algorithms (which can after successful training of course be used as classifiers), further challenges in computer vision

---

CONCERNING THE FUTURE OF ARTIFICIAL INTELLIGENCE; Brasov: University of Brasov

199 Hubel, D. and Wiesel, T. (1968): Receptive fields and functional architecture of monkey striate cortex; Journal of Physiology (London), 195, 215–243.

200 tutis.ca (2013): The Visual Cortex [2016-06-19]; URL: http://www.tutis.ca/NeuroMD/L2V123/V123.pdf

201 Fukushima, K. (1980): Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position; Biological Cybernetics, 36, 193–202.

202 Serre, T., Wolf, L., Bileschi, S., and Riesenhuber, M. (2007): Robust object recog- nition with cortex-like mechanisms; IEEE Trans. Pattern Anal. Mach. Intell., 29(3), 411–426. Member-Poggio, Tomaso.

203 LeCun Yann, Bottou Léon, Bengio Yoshua, and Haffner Patrick (1998): Gradient-based learning applied to document recognition; Proceedings of the IEEE, 86(11), 2278–2324

before the introduction of CNNs were position invariance and noise in an image. Position invariance refers to the spatial position in the image (thus an object located in the top left corner, the bottom right corner or anywhere else), but also the size of the object, e.g. not all cars in an image have the same size or have been photographed from the same distance.

4.1.1 Data representation

One of CNN's primary tasks is to process (classify) images, such as identifying an object in a picture successfully as car or distinguishing between humans and immobile objects when it comes to autonomously driving cars. CNNs "perceive" images as 4-dimensional tensors, whereby a tensor is a data structure that can be imagined either as an array of arrays, or just a stack of matrices, each element of each matrix being another multi-dimensional element. An example for a single 3-dimensional tensor is

$$t = \begin{pmatrix} \binom{1}{4} & \binom{4}{5} & \binom{6}{8} \\ \binom{3}{5} & \binom{6}{7} & \binom{8}{4} \end{pmatrix} \qquad (4\text{-}1)$$

A 4-dimensinal tensor goes just one step further and replaces each of the elements by another multi-dimensional structure, thus is an array of arrays of arrays (see Figure 44 - 4-dimensional tensor).

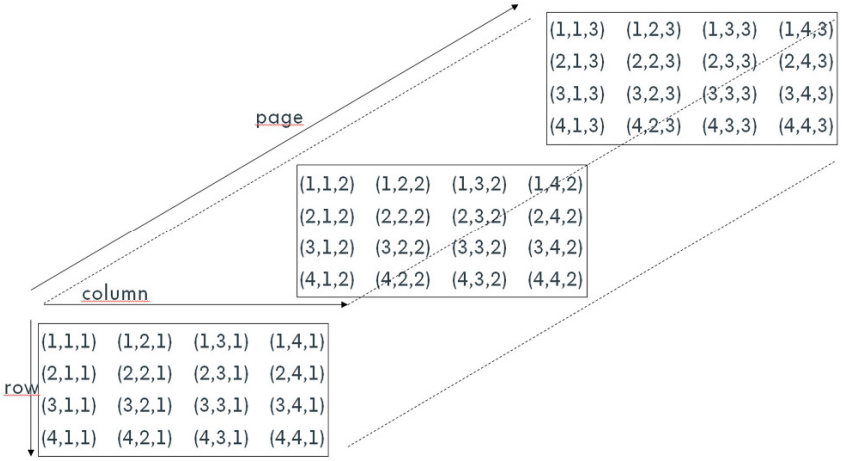

Figure 44 - 4-dimensional tensor

I mentioned beforehand that data may be represented differently, and where this becomes obvious first is in the additional dimensions used for describing an image. Two dimensions are required because the image has length and height, and the other dimensions are needed because of color-encoding. If, for example, RGB encoding is applied, then the CNN features three input layers (called channels), one for each color.

4.1.2 Structure

As data is presented in boxes, and as convolution within the network takes place, the structure differs a lot from classical feed forward ANNs. What is more, a whole standard feed forward ANN is attached to the structure after convolution and downsampling are finished. The receptive fields in a CNN are implemented by connecting subsets of layers l to superordinated layers $l + n$.

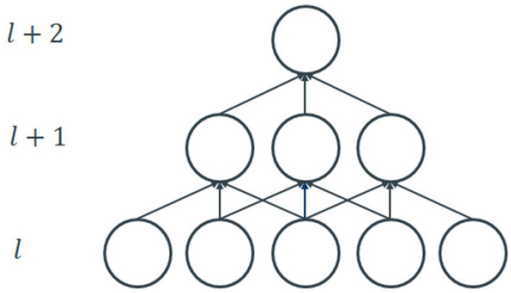

Figure 45 – CNN sparse interconnectivity[204]

In Figure 45 – CNN sparse interconnectivity the bottom (input) layer l represents the receptive field consisting of 5 neurons, each of which features up to three connections to the next layer $l + 1$. Viewed from $l + 1$ down to l the receptive field has width 3, as each $l + 1$-neuron receives input from three l-neurons. This is how spatially local input patterns are learned from the input layer l (with a receptive field of size 5) up to layer $l + n$. Each unit is does not react to variations occurring outside of its receptive field, so what this makes sure that the learnt filters bring out the most vigorous response to a spatially local input pattern.[205] The original architecture of the LeNET-5 neural network features three layer types (see Figure 46 - CNN architecture):

– Convolutional layers
– Max-pool layers
– Dense layers

204 LeCun Yann, Bottou Léon, Bengio Yoshua, and Haffner Patrick (1998): Gradient-based learning applied to document recognition; Proceedings of the IEEE, 86(11), 2278–2324

205 Deeplearning.net (2012): Convolutional Neural Networks (LeNet) [2016-06-20]; Deeplearning.net; URL: http://deeplearning.net/tutorial/lenet.html

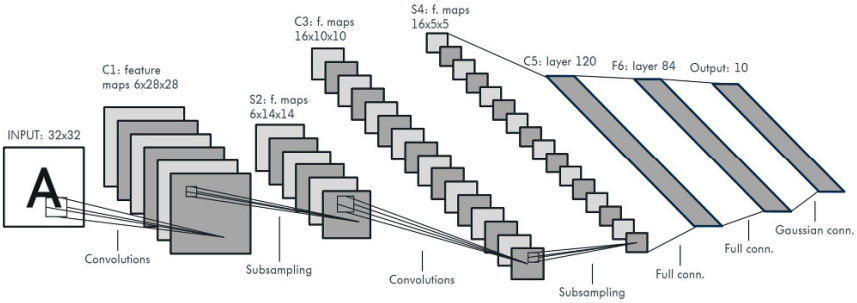

Figure 46 - CNN architecture

4.1.2.1 Convolutional layers

Convolution coms from the Latin word convolvere and stands for "rolling together", and expressed in a more mathematical way a convolution is defined as a product of functions f and g that are objects in the algebra of Schwartz functions in $\mathbb{R}^n$, and convolution of two functions f and g over a finite range $[0, t]$ is given by

$$[f * g](t) = \int_0^t f(\tau) g(t - \tau) d\tau \qquad (4\text{-}2)$$

where $[f * g](t)$ represents convolution of f and g. Thus, convolution produces a third function $[f * g](t)$ from two functions f and g.

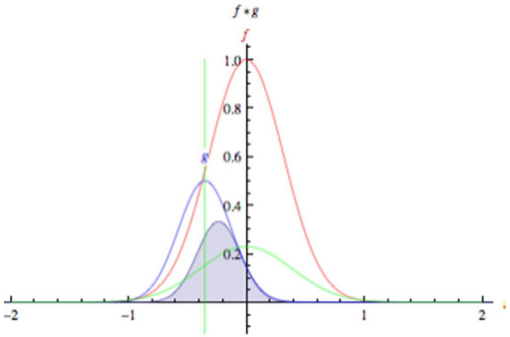

Figure 47 – Convolution[206]

206 WolframMathworld: Colvolution [2016-16-21]; URL: http://mathworld.wolfram.com/Convolution.html

The green curve in Figure 47 – Convolution shows the convolution of the functions f and g, and the x-axis is t. Function g moves from the left to the right, thus propagates in time, and the vertical green line indicates where on the t-axis we are. The filled grey region shows the product of $f(\tau)g(t-\tau)$ as a function of t, thus it is the convolution.[207] Something that is not obvious when looking at the convolution equation is why τ shows up. Let assume we start with the standard notation for convolution:

$$y(t) = x(t) + h(t) \qquad (4\text{-}3)$$

The notation can be misleading in the sense that it seems as if we get $y(t)$ for a particular value of t by looking at both x and h for a specific value of t. What is required instead is to look at the whole time wave forms of x and h, which is why t comes in. Figure Figure 48 – $\boldsymbol{x(t)}$ depicts $x(t)$ and a time signal. At point $t = 2$ we can see that $x(t) = 3.5$. $x(t)$ is a time-function, so it changes over time.

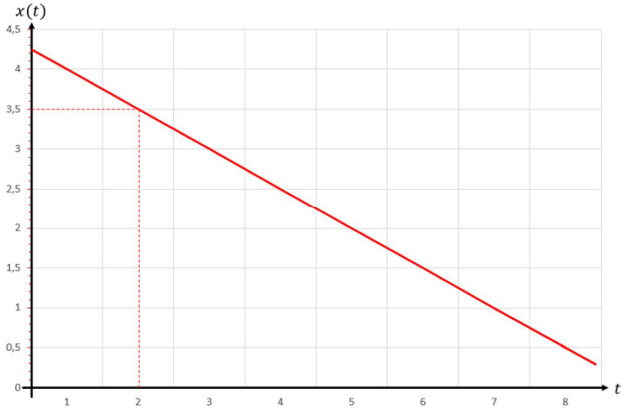

Figure 48 – $\boldsymbol{x(t)}$

Basically, there is nothing special about t, so it would also be possible replace it by τ and talk about $x(\tau)$ instead (see Figure 49 – $\boldsymbol{x(\tau)}$).

207 WolframMathworld: Colvolution [2016-16-21]; URL: http://mathworld.wolfram.com/Convolution.html

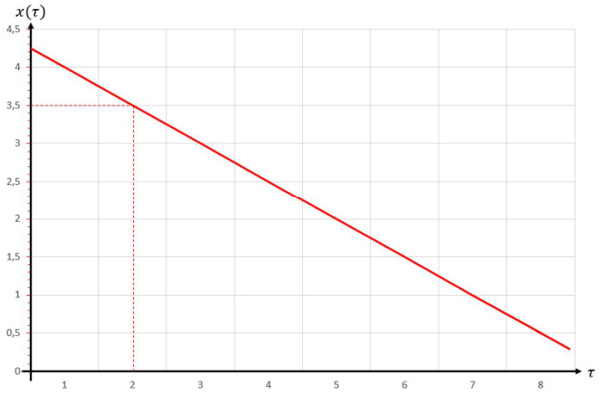

Figure 49 – $\boldsymbol{x(\tau)}$

The same holds for $h(t)$. Let assume the following example in which $h(t) = 1$ where e.g. $t = 3$ (see Figure 50 – $\boldsymbol{h(t)}$).

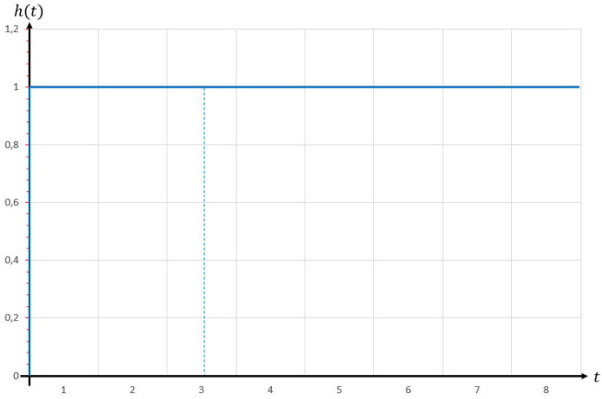

Figure 50 – $\boldsymbol{h(t)}$

Of course, the same replacement of t by τ is possible, but what we are interested here is $t - \tau$, thus $h(t - \tau)$. Let assume that $t = 2$, then we can get to $h(t - \tau)$ by reflecting the function in Figure 50 – $\boldsymbol{h(t)}$ around the $\tau = 0$-line, and shifting it so that the point that lies on 0, now lies on 2 (see Figure 51 – $\boldsymbol{h(\tau)}$ reflected and Figure 52 – $\boldsymbol{h(\tau)}$ reflected and shifted).

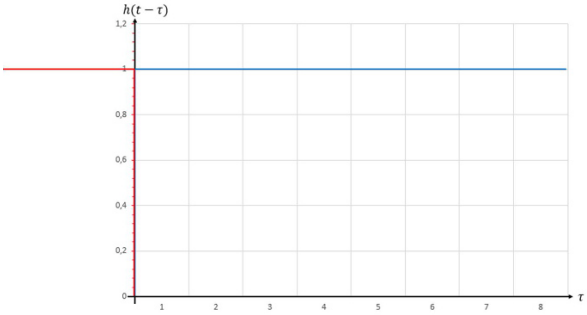

Figure 51 – $\boldsymbol{h(\tau)}$ reflected

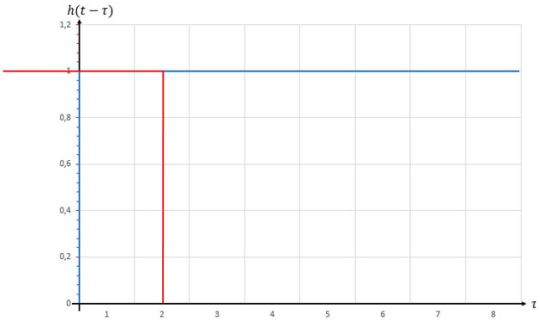

Figure 52 – $\boldsymbol{h(\tau)}$ reflected and shifted

Next and according to the convolution equation, the integral shows the area covered by the multiplication of $x(\tau)$ and $h(t-\tau)$ (see Figure 53 – Convolution of $\boldsymbol{x(t)}$ and $\boldsymbol{h(t-\tau)}$, green area).

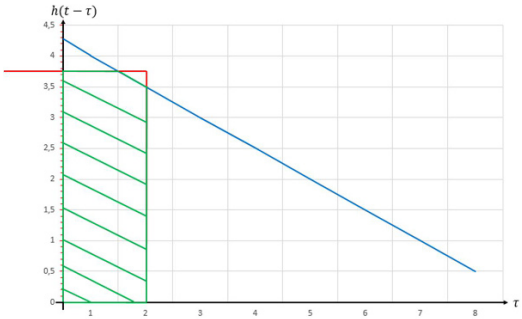

Figure 53 – Convolution of $\boldsymbol{x(t)}$ and $\boldsymbol{h(t-\tau)}$

τ shows up because in order to do the convolution of x and h it is required to have to look at x for values other than t, and in general it is necessary to have to look at x for every possible value of its argument. The same holds for h – every possible value of its argument must be considered. That's what τ does, as it ranges from $-\infty$ to $+\infty$.

When considering analysis of images, the immobile function (red in Figure 47 – Convolution) represents the input data, thus the image itself. The second function moving from the left to the right (blue in Figure 47 – Convolution) is the filter applied to the image, whereby 1 filter is no restriction – usually there are numerous filters applied to an image, each recognizing a different feature. In the first or one of the first convolution layers filters for detecting/creating a map of edges in the image are possible, such as one filter recognizing horizontal, one recognizing diagonal and one for vertical lines. The filters thus perform a search over the image-space – a sliding window moving over the image – and any time a match is detected, this is recorded. Finally, after each convolution, activation functions are applied on the output – some of which are mentioned at 3.3.1.3 Activation functions).

4.1.2.2 Different ways of perception and processing

One thing we should always bear in mind is the fact that in artificial intelligence we seek to implement intelligence in software (and, considering quantum computers, also in hardware), but not necessarily in the same way as humans implement intelligence, for whatever intelligence exactly is. A CNN does not perceive images as we humans do, which is where the tensors mentioned beforehand come into play. If we consider an RGB image, a CNN does not take just one image as input, as we humans seem to do, but splits it into three channels, one for each of the colors. Thus, width and height of the image remain unchanged, but instead of processing one flat 2D-image, it comes as a three layer deep box. Furthermore, the input volume in terms of spatial dimensions does not remain constant propagating through the network.

To start with, the intensity of the three colors red, green and blue is expressed by a number, each of which will be an element in one of the three channels making up the image box. Any channel at this stage is represented by a 2D-matrix. The CNN then applies each of the filters on each of these channels, and at any step the dot product between the filter and the patch the filter slides over is taken (depending from the shape of the filter other operations are applied as well, e.g. the Hadamard product for rectangular filters). If the two matrices both have feature high values in the same positions, the resulting dot product will be high as well, otherwise it will be low. If a filter is thus finding its pattern in the patch it is currently sliding over, this will be recorded with high values. At each step, another dot product is taken, each of which contribute to the activation map, which is another matrix created for holding these values. One parameter of importance is the stride – the step size for moving the filter across the image. The number of columns in the activation map equals the number of steps (thus depends on the stride) required to move across the image. Any search for a specific pattern, which is the application of a specific filter, will produce one activation map. This is where the number of spatial dimensions of the output already differs from the number of input dimensions. Let assume a filter of size 4×4 is moving over a channel with 40×40 pixels with stride 4, and then it will start with rows 1-4 in the top left corner and move across the image until it reaches the top right corner, which will take 10 steps. It will then continue with lines 5-8, and go on until it reached the bottom right corner. The result will be a 10×10 matrix (activation map). If we would look for 20 different patterns, 20 activation maps would

be produced, resulting in a new box sized $10 \times 10 \times 20$ (see the feature maps in Figure 46 - CNN architecture). This whole process is called convolution.

4.1.2.3 Maxpooling/ downsampling layers

What happens in the maxpooling (or downsampling, subsampling) layer is that the activation maps are downsampled, which is similar to what happens in the convolution layers. Patches of a pre-defined size are cut under consideration of a stride and the highest number within a patch is extracted and set as element in a new matrix. Let assume the activation map is as shown in Figure 54 – Activation map

2	5	5	6
6	3	3	7
8	7	5	1
4	9	8	2

Figure 54 – Activation map

A max pool with 22 filters and stride 2 would produce a downsampled activation map as shown in Figure 55 – Downsampled activation map

6	7
3	8

Figure 55 – Downsampled activation map

What remains are the strongest correlations to a feature compressed to a sub-space. Although lots of information is lost in that step, this is also one of the advantages, because it decreases both required processing time and memory needed.

4.1.2.4 Feature maps

As described above, each of the filters is replicated over the whole receptive field (slides over the field), and the dot product is between the filtered region and the filter is calculated. Weights (thus the values in the filter matrix) are shared as the filter moves across the image, which greatly reduces the amount of processing time needed. The same filter may be applied upon the different input channels, so if the input is an RGB image, the filter is applied three times. In Figure 56 – Convolution calculation the calculation is shown in detail, whereby the training algorithm found different filters for the different channels:

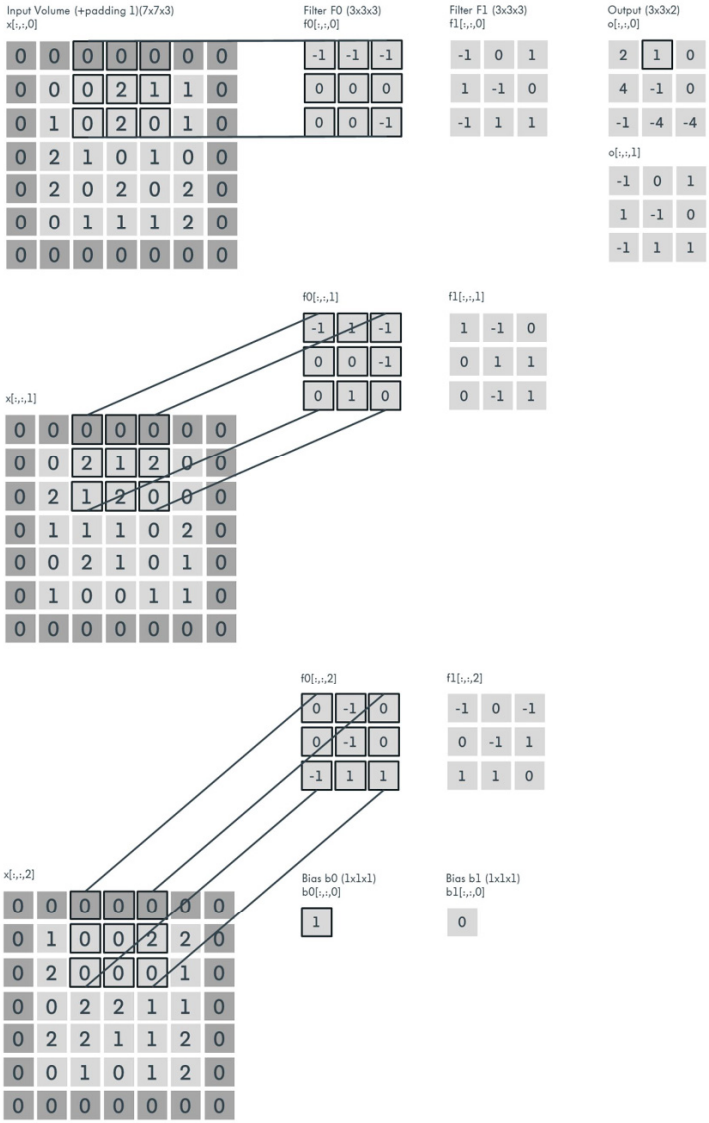

Figure 56 – Convolution calculation[208]

208 Cf. Karpathy Andrej: CS231n Convolutional Neural Networks for Visual Recognition [2016-06-23]; URL: https://cs231n.github.io/

Filter W0 contains three filters, one for each channel, and so does filter W1. The number of filters depends on how many patterns the network should be able to recognize, but ideally one starts with a predefined configuration, such as LeNet or OxfordNet. In the first layer the filters may still be interpretable by examining them closer, but this will become more difficult for subsequent layers. What the training algorithm does is to find filters which minimize the training error, and these are not necessarily human-interpretable. Figure 56 – Convolution calculation depicts the calculation of the dot products of channel 1-3 with its respective filters. In the example the three dot products are summed up to one feature in the feature map – resulting in the value 1. To come back to terms, a feature map is calculated by the repeated application of a function (dot product in our case) across the sub-regions of the entire image. This is what is generally referred to as the convolution of an image with a filter. Additionally, a bias is added and the output passed over to an activation function, as in some of the other artificial neural networks introduced beforehand:

$$h_{ij} = tanh\big((W^k * x)_{ij} + b_k\big) \tag{4-4}$$

The convolution for a one-dimensional signal is given by

$$[f * g](n) = \sum_{u=-\infty}^{\infty} f(u)g(n-u) = \sum_{u=-\infty}^{\infty} f(n-u)g(u) \tag{4-5}$$

For a two-dimensional signal it is given by

$$[f * g](m,n) = \sum_{u=-\infty}^{\infty} f(u,v)g(m-u,n-v) \tag{4-6}$$

which extends to n dimensions. These equations apply to discrete signals, whereas the integral specified beforehand applies to continuous signals.

4.1.2.5 Fully connected layers

Figure 46 - CNN architecture shows that after the layer S4 the network structure changes to a standard feed forward structure, which has been explained at 3.3.2.2 Feed-forward artificial neural network. Let assume S4 produces 16 5 × 5 feature maps, and the next layer features 120 neurons, then each of the 25 pixels within the 5 × 5 maps would feature 120 connections. This results in 48,000 connections from S4 to C5.

4.1.2.6 Number of neurons

The number of the hidden layer's neurons depends on various factors, such as the size of the input image, how many features should be learned (thus, how many filters are applied), filter size and stride: the larger filter and stride, the better the performance as fewer calculations need to be conducted, but also the poorer the performance.

4.1.3 Training

The network can be trained with most of the already introduced training algorithms, such as any propagation algorithm, genetic algorithms or simulated annealing. The most popular algorithm is back propagation, however the forward and backward pass differ slightly depending from what layer the calculations or the error is propagated through. Training

regarding the fully connected layer does not differ from what has been introduced in 3.3.3.6.1 Back propagation training.

Let assume we start with an image of a size of $M \times N$ and a filter size of $m \times n$, then the convolution is

$$z_{ij}^{(k)} = \sum_c \sum_{s=0}^{m-1} \sum_{t=0}^{n-1} w_{st}^{(k,c)} x_{(i+s)(j+t)}^{(c)} \quad (4\text{-}7)$$

where $z_{ij}^{(k)}$ represents the convolution of the filter k over the channel c, i and j are initial row and column for each calculation on the channel, $w_{st}^{(k,c)}$ is the value (weight) of the filter k in its s^{th} row t^{th} column, and c the channel (see Figure 57 - Featuremap generation). If the filters are squared, it is also possible to apply the dot product:

$$z_{ij}^{(k)} = \sum_c \sum_{s=0}^{m-1} \sum_{t=0}^{n-1} w_{st}^{(k,c)} x_{(i+t)(j+s)}^{(c)} \quad (4\text{-}8)$$

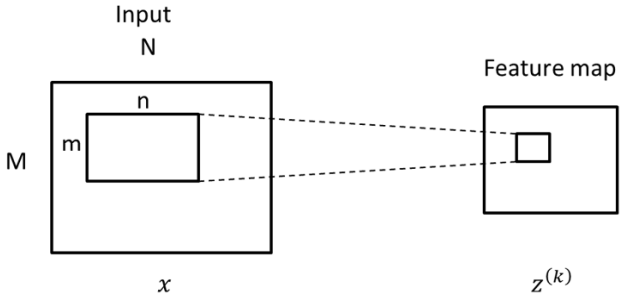

Figure 57 - Featuremap generation

As indicated beforehand, an RGB image has 3 channels, and the filters are adapted for each of the channels first, and summed up after calculation. If the number of filters is K and the number of channels C, then $W \in \mathbb{R}^{K \times C \times n \times n}$. From the equation it becomes obvious that the size of the convolved image is $(M - m + 1) \times (N - n + 1)$. The activations are calculated by

$$a_{ij}^{(k)} = h\left(z_{ji}^{(k)} + \theta^k\right) = max\left(0, \left(z_{ji}^{(k)} + \theta^k\right)\right) \quad (4\text{-}9)$$

where θ^k is the bias of the k^{th} filter. $\theta^k \in \mathbb{R}^K$ is one-dimensional, thus given by a vector, indicating that one bias for each filter exists. In the max-pooling layer, no learning at all happens, so the propagation is

$$y_{ij}^{(k)} = max\left(a_{(l_1 i+s)(l_2 j+t)}^{(k)}\right) \quad (4\text{-}10)$$

where l_1 and l_2 represent the pooling filter size and $s \in [0, l_1], t \in [0, l_2]$. As indicated in 3.3.1.4 Regularization, there exist several ways for regularization, and max-pooling/ downsampling is just the most popular one. Usually, the filters are quadratic and have a size of 2×2 up to 4×4. There is also no need to follow the standard procedure of convolution-

activation-pooling, as the latter two can be mixed up and a convolution filter can of course be succeeded by another convolution filter. The parameters to be provided to the algorithm are thus architecture-related:

– number of convolutions
– number of filters
– filter size
– number of max-pooling filters
– size of max-pooling filters
– arrangement of layers

Finally, the MLP attached to the last convolution/ max-pooling filter requires one-dimensional data, as it is common. Feature/ activation maps are of higher dimension, at least two-dimensional. A 2-dimensional quadratic filter of size $l_1 = l_2 = 2$ would thus feature as many connections from each of its pixels as the fully connected layer has neurons. The error weight adaption is conducted via back propagation, and for the MLP-layers it is exactly as described at 3.3.3.6.1 Back propagation training and will not be repeated here. The error from the MLP-input layer is propagated to the max-pooling layer, and re-mapped into the two-dimensional max-pooling layers. As no learning happens in the max-pooling layer (it is model parameter-free), the error is propagated back to the previous layer:

$$\frac{\partial E}{\partial a^{(k)}_{(l_1 i+s)(l_2 j+t)}} = \begin{cases} \frac{\partial E}{\partial y^{(k)}_{ij}} & if\ y^{(k)}_{ij} = a^{(k)}_{(l_1 i+s)(l_2 j+t)} \\ 0 & otherwise \end{cases} \tag{4-11}$$

where E is the evaluation (error) function. The so-calculated error is propagated to the convolution layer, and can be used to calculate the bias and gradients. As the activation with the bias happens before the convolution when propagating backwards through the network, the gradient of the bias has to be calculated first according to

$$\frac{\partial E}{\partial b^{(k)}} = \sum_{i=0}^{M-m} \sum_{j=0}^{N-n} \frac{\partial E}{\partial a^{(k)}_{ij}} \frac{\partial a^{(k)}_{ij}}{\partial \theta^{(k)}} \tag{4-12}$$

With the following two definitions

$$\delta^{(k)}_{ij} \coloneqq \frac{\partial E}{\partial a^{(k)}_{ij}} \tag{4-13}$$

and

$$c^{(k)}_{ij} \coloneqq z^{(k)}_{ij} + \theta^{(k)} \tag{4-14}$$

such that

$$\frac{\partial E}{\partial b^{(k)}} = \sum_{i=0}^{M-m} \sum_{j=0}^{N-n} \delta^{(k)}_{ij} \frac{\partial a^{(k)}_{ij}}{\partial c^{(k)}_{ij}} \frac{\partial c^{(k)}_{ij}}{\partial \theta^{(k)}} = \sum_{i=0}^{M-m} \sum_{j=0}^{N-n} \delta^{(k)}_{ij} h'\left(c^{(k)}_{ij}\right) \tag{4-15}$$

The gradient of the weight is calculated identically:

$$\frac{\partial E}{\partial w_{st}^{(k,c)}} = \sum_{i=0}^{M-m}\sum_{j=0}^{N-n}\frac{\partial E}{\partial z_{ij}^{(k)}}\frac{\partial z_{ij}^{(k)}}{\partial w_{st}^{(k,c)}} = \sum_{i=0}^{M-m}\sum_{j=0}^{N-n}\frac{\partial E}{\partial z_{ij}^{(k)}}\frac{\partial a_{ij}^{(k)}}{\partial z_{ij}^{(k)}} x_{(i+s)(j+t)}^{(c)} = \sum_{i=0}^{M-m}\sum_{j=0}^{N-n}\delta_{ij}^{(k)} h'\left(c_{ij}^{(k)}\right) x_{(i+s)(j+t)}^{(c)} \quad (4\text{-}16)$$

If the CNN has multiple convolutional and max-pooling layers, then the error of the convolutional layers also has to be calculated:

$$\frac{\partial E}{\partial w_{ij}^{(c)}} = \sum_k \sum_{s=0}^{m-1}\sum_{t=0}^{n-1}\frac{\partial E}{\partial z_{(i+s)(j+t)}^{(k)}}\frac{\partial z_{(i+s)(j+t)}^{(k)}}{\partial x_{ij}^{(c)}} \quad (4\text{-}17)$$

which results in

$$\frac{\partial E}{\partial z_{ij}^{(c)}} = \frac{\partial E}{\partial a_{ij}^{(k)}}\frac{\partial a_{ij}^{(k)}}{\partial z_{ij}^{(k)}} = \delta_{ij}^{(k)}\frac{\partial a_{ij}^{(k)}}{\partial c_{ij}^{(k)}}\frac{\partial c_{ij}^{(k)}}{\partial z_{ij}^{(k)}} = \delta_{ij}^{(k)} h'\left(c_{ij}^{(k)}\right) \quad (4\text{-}18)$$

The error is then

$$\frac{\partial E}{\partial w_{ij}^{(c)}} = \sum_k \sum_{s=0}^{m-1}\sum_{t=0}^{n-1}\delta_{(i-s)(j-t)}^{(k)} h'\left(c_{(i-s)(j-t)}^{(k)}\right) w_{st}^{(k,c)} \quad (4\text{-}19)$$

The algorithm for learning CNNs is as follows (assumption is 1 convolution and 1 max-pooling layer; training equations of the fully connected layer has been omitted, as it has already been described):

1. ***Start***
2. Initial selection of network weights w_i at random.
3. ***Repeat***
   a) Calculate the convolution for channels C and filters K

$$z_{ji}^{(k)} = \sum_c \sum_{s=0}^{m-1}\sum_{t=0}^{n-1} w_{st}^{(k,c)} x_{(i+s)(j+t)}^{(c)}$$

   b) Calculate the rectifier activations

$$a_{ij}^{(k)} = h\left(z_{ji}^{(k)} + \theta^k\right) = \max\left(0, \left(z_{ji}^{(k)} + \theta^k\right)\right)$$

   c) Propagate the convolution output to the max-pooling layer

$$y_{ij}^{(k)} = \max\left(a_{(l_1 i+s)(l_2 j+t)}^{(k)}\right)$$

   d) Propagate the output through the fully connected ANN
   e) Calculate the error
   f) Propagate the error back through the fully connected ANN
   g) Adapt weights of fully connected ANN
   h) Propagate the max-pooling output through the fully connected ANN

$$\frac{\partial E}{\partial a_{(l_1 i+s)(l_2 j+t)}^{(k)}} = \begin{cases} \dfrac{\partial E}{\partial y_{ij}^{(k)}} & if\ y_{ij}^{(k)} = a_{(l_1 i+s)(l_2 j+t)}^{(k)} \\ 0 & otherwise \end{cases}$$

i) Calculate gradient of the bias

$$\frac{\partial E}{\partial b^{(k)}} = \sum_{i=0}^{M-m} \sum_{j=0}^{N-n} \delta_{ij}^{(k)} h'\left(c_{ij}^{(k)}\right)$$

j) Calculate the gradient of the filter

$$\frac{\partial E}{\partial w_{st}^{(k,c)}} = \sum_{i=0}^{M-m} \sum_{j=0}^{N-n} \delta_{ij}^{(k)} h'\left(c_{ij}^{(k)}\right) x_{(i+s)(j+t)}^{(c)}$$

k) Calculate the error of the convolution layer

$$\frac{\partial E}{\partial w_{st}^{(k,c)}} = \frac{\partial E}{\partial w_{ij}^{(c)}} = \sum_{i=0}^{M-m} \sum_{j=0}^{N-n} \delta_{ij}^{(k)} h'\left(c_{ij}^{(k)}\right) x_{(i+s)(j+t)}^{(c)}$$

4. ***Until criteria are reached***
5. ***End***

Algorithm 10 – CNN back propagation algorithm

Breakdown:

k ∈ K: filter k of all filters K

θ^k: bias of filter k

h: activation function

a: activation

$z_{ij}^{(k)}$: convolution z of k^{th} filter, i and j row and column upper left corner of convoluted area of image

: error function

$c \in C$: channel c of all channels C

4.2 Deep belief artificial neural network

Deep belief ANNs are, once the structure is finalized, deep feed forward ANNs, which means that they feature at least more than one hidden layer. The major difference is the weight initialization, which is called layer-wise pre-training. The answer to the question why a more sophisticated weight initialization is required for being able to successfully train a deep ANN with an algorithm such as back propagation is motivated in two problems: vanishing and/ or exploding gradients. The basic idea is that more hidden layers allow an ANN to learn more complex classification functions, which results in higher accuracy. Based on that assumption, additional layers should, in the worst case, not contribute at all, thus neither in- or decrease

accuracy. However, that is not what is happening. The more layers are added to an ANN, the more inaccurate it becomes. Let assume an ANN with two hidden layers, then once back propagation is conducted, one can observe that the gradients for the neuron bias $\frac{\partial E}{\partial b}$ and the gradients for the weights $\frac{\partial E}{\partial w}$, in other words the rate of change of the cost with respect to a neuron's bias and weight in the second hidden layer are a lot larger than in the first hidden layer. In order to understand this better it is useful to compare the learning speed of these two hidden layers by observing the gradients for the first and second hidden layer

$$\delta_j^l = \partial E / \partial b_j^l \quad (4\text{-}20)$$

which represents the gradient for the j^{th} neuron in the l^{th} layer, whereby δ^l is a vector containing all gradients. The size of the single vector components determines how fast the hidden neurons in the layer l learn. In order to determine how fast a layer learns, the length of the vector $\|\delta^l\|$ can be taken. What one can observe is that the gradient gets smaller the more hidden layers an ANN has and as one moves backward through the hidden layers. This is called the vanishing gradient problem. If the opposite happens, which cannot be ruled out, we are talking about the exploding gradient problem. It helps to explain the vanishing gradient problem by going through a simple example:

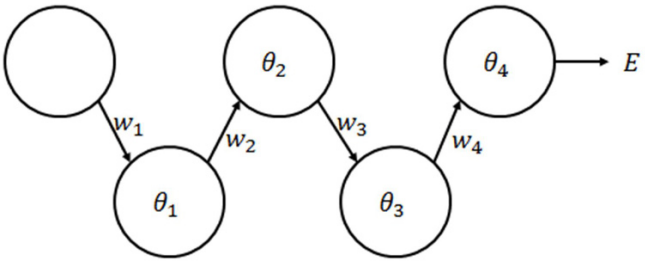

Figure 58 - Simple ANN

The very simple multi-layered ANN depicted in Figure 58 - Simple ANN features four weights $w_1, \dots, w_4$ and 4 biases $\theta_1, \dots, \theta_4$, and the calculated output is compared with the desired output by an error function E (e.g. RMSE). The gradient θ_1 belonging to the hidden neuron 1 is

$$\frac{\partial E}{\partial b_1} = h'(y_1) \times w_2 \times h'(y_2) \times w_3 \times h'(y_3) \times w_4 \times h'(y_4) \times \frac{\partial E}{\partial a_4} \quad (4\text{-}21)$$

where

$$y_j = w_j x_{j-1} + \theta_j \quad (4\text{-}22)$$

y_j represents the weighted input to a neuron j, x_{j-1} the output of neuron $j-1$, h' the derivative of the activation function h.

What happens is that if a change $\Delta\theta_1$ in the bias θ_1 is conducted, this will affect the numeric content of the whole network:

- The output Δx_1 of the first neuron is affected.
- This will result in a change of y_2, the weighted input to the second hidden neuron.
- This will also change Δx_2.
- This continues until the error function is reached, which will then result in a change in the output cost ΔE, so

$$\frac{\partial E}{\partial \theta_1} \approx \frac{\Delta E}{\Delta \theta_1} \tag{4-23}$$

If the result of each single step mentioned above is tracked, an expression for the gradient $\frac{\partial E}{\partial \theta_1}$ can be determined. $\Delta\theta_1$ changes the output of the first hidden neuron x_1 as follows:

$$x_1 = h(y_1) = h(w_1 x_0 + \theta_1) \tag{4-24}$$

Therefore,

$$\Delta x_1 = \frac{\partial h(w_1 x_0 + \theta_1)}{\partial \theta_1} \Delta\theta_1 = h'(y_1)\Delta\theta_1 \tag{4-25}$$

where $h'(y_1)$ is the first term in the gradient $\frac{\partial E}{\partial \theta_1}$, converting a change $\Delta\theta_1$ in the bias into a change Δx_1 in the output activation. This results in a change in

$$y_2 = w_2 x_1 + \theta_2 \tag{4-26}$$

which is the second hidden neuron's weighted input.

$$y_2 \approx \frac{\partial y_2}{\partial x_1} \Delta x_1 = w_2 \Delta x_1 \tag{4-27}$$

Combined with the first term calculated before this results in

$$y_2 \approx w_2 h'(y_1)\Delta\theta_1 \tag{4-28}$$

This clearly shows how the following terms will be influenced by a change in θ_1:

$$\Delta E \approx h'(y_1) w_2 h'(y_2) w_3 h'(y_3) w_4 h'(y_4) \frac{\partial E}{\partial x_4} \tag{4-29}$$

The last expression is constructed of single terms of the structure $w_j h'(y_j)$. The derivative of the sigmoid function is

$$\sigma' = \frac{e^{-x}}{(1+e^{-x})^2} \tag{4-30}$$

and its peak is at 0.25. Thus, when initializing the weights in the ANN with a Gaussian with $\mu = 0$ and $\sigma = 1$ (here, σ is the standard deviation, not an activation function). In terms of the sigmoid function the weights will then be smaller than 1, and as we saw that the whole term is a product of expressions $w_j \sigma'(y_j)$, any of these expressions will satisfy

$$|w_j \sigma'(y_j)| < \frac{1}{4} \tag{4-31}$$

This in turn means that the product will become smaller the more terms are in the network, thus the deeper the network. Comparing $\frac{\partial E}{\partial \theta_1}$ with $\frac{\partial E}{\partial \theta_3}$ we can see that these have some terms in common:

$$\frac{\partial E}{\partial \theta_1} = \sigma'(y_1)w_2\sigma'(y_2)w_3\sigma'(y_3)w_4\sigma'(y_4)\frac{\partial E}{\partial \theta_4} \quad (4\text{-}32)$$

$$\frac{\partial E}{\partial \theta_3} = \sigma'(y_3)w_4\sigma'(y_4)\frac{\partial E}{\partial \theta_4} \quad (4\text{-}33)$$

Thus, as we move backwards through the network when calculating the gradients, $\frac{\partial E}{\partial \theta_1}$ will be a lot smaller than $\frac{\partial E}{\partial \theta_3}$.

Another challenge in deep ANNs is the exploding gradient problem, which occurs when the weights are chosen to be large, and the biases $\theta_1, \dots, \theta_4$ so that the problematic terms will not become too small, i.e. in a way that results in a weighted neuron input of at least $y_j = 0$ and in consequence $\sigma'(y_j) = \frac{1}{4}$. Let assume

$$y_1 = w_1x_0 + \theta_1 = 0 \quad (4\text{-}34)$$

which can easily be accomplished by setting

$$\theta_1 = -100 * y_1 \quad (4\text{-}35)$$

All terms $w_j\sigma'(y_j)$ will then result in

$$100 * \frac{1}{4} = 25 \quad (4\text{-}36)$$

Summing up, the problem is that the gradient is unstable, thus that gradients in early layers always depend from the gradients in later layers. With many layers, this can result in considerable uncertainty. Therefore, other approaches for training deep networks had to be found. Considering the multi-neuron multi-layer case, the gradient in the l^{th} layer of an L-layered ANN is given by

$$\delta^l = \sum'(y^l)(w^{l+1})^T\sum'(y^{l+1})(w^{l+2})^T \dots \sum'(y^L)\nabla_x E \quad (4\text{-}37)$$

$\sum'(y^L)$ is a diagonal matrix, and its entries, the $\sigma'(y)$-values, are the values for the weighted inputs to the l^{th} layer. w^l are the weight matrices for each layer, and $\nabla_x E$ is a vector containing partial derivatives of E with respect to the output activations. Compared to the one-layered network used beforehand, the critical terms change to $(w^l)^T\sum'(y^l)$. Any $\sum'(y^l)$ has small entries, each of them smaller than $\frac{1}{4}$, which means given the w^l are sufficiently small, each of the $(w^l)^T\sum'(y^l)$ will make the gradients sufficiently smaller causing the vanishing gradient problem.[209]

209 Nielsen Michael (2016): Neural Networks and Deep Learning [2016-07-04]; URL: http://neuralnetworksanddeeplearning.com/chap5.html

4.2.1 Stacking together RBMs

Restricted Boltzmann machines have already been introduced at 3.3.2.9.2 Boltzmann Machine, and training and stacking together RBMs is the weight initialization needed in order to prevent the problems mentioned to occur. When training an RBM, in the forward pass the inputs of neurons $n_1, \dots, n_n$ are multiplied by the respective weights of hidden layers $h_1, \dots, h_n$, meaning that at h_1 the summation happens, a bias-value is added, and the sum is passed through an activation function for scaling the output, as common in ANNs. The difference to a standard feed forward ANN is that only discrete values (usually 0 and 1) are taken – research on continuous RBMs is ongoing. As RBMs have been developed as auto-encoders, the backward pass aims to reconstruct the presented input vectors. Therefore, a way to imagine how processing works is to see the activations at the hidden layer as input and the input layer as output layer, thus the activations are multiplied by the respective weights and summed up at the input layer (bot no further activation happens). In order to adapt the weights the back propagation algorithm is applied. Once training has reached the required criteria and the reconstructed values are close enough to the optimum, the activations at the hidden layer are taken and serve as inputs to the next RBM. Thus, the hidden features extracted from the input dataset are used as inputs to the next layer. This is why and how concepts can be learned over a hierarchical structure. After the weights have been initialized in this specific way, an output layer needs to be added, which is capable of representing the desired class output, whereby the output layer usually relies on algorithms such as SVM or logistic regression. Finally, all stacked layers together are treated as a standard feed forward ANN and training can be conducted via back propagation, and usually not much weight adaption happens after this very special form of weight initialization. This is also why in terms of DBNs we are talking about a combination of unsupervised and supervised training – weight initialization is unsupervised, as no class labels are provided and the hidden patterns in the data are learned. Fine-tuning via back propagation is supervised.

In 3.3.2.9.2 Boltzmann Machine the notation of (free) energy has been introduced, and this is how RBMs differ in "thinking" to feed forward ANNs. Every unit features a stochastic state, and the RBM's energy is determined based on each of these units' state. The least amount of energy possible in a network is the target of the minimization problem, and the (minimum) energy condition memorizing the correct data is the network's steady state. RBMs in general have been proposed, as BMs need a huge amount of processing time due to their fully connected architecture.

4.2.2 Training

The training of a deep belief network leverages both the training algorithms of

– RBMs: weight initialization
– back propagation for feed forward ANNs: training after weight initialization

The algorithm for training deep belief ANNs is as follows (details for RBM and back propagation to be found at 3.3.2.9.2 Boltzmann Machine and 3.3.3.6.1 Back propagation training):

Start

1. ***Repeat***
   a) Initialize and train rbm_x from $rbm_1, \dots, rbm_n$
   b) If $x > 1$
      i. Stack rbm_{x-1} and rbm_x
2. ***Until $x == n$***
3. Add output layer
4. Train DBN according to back propagation algorithm

End

Algorithm 11 – DBN training

Breakdown

$rbm_1, \dots, rbm_n$: RBM $1, \dots, n$, depending from the required depth

4.3 Cortical artificial neural network

The cortical artificial neural network[210] is one of the deep artificial neural structures that have been conceived and specified during the research phase of the SHOCID project. Its development has been inspired by theoretical aspects of the signal processing in the human brain, resulting in conscious experiences. Yet Edelman said that human consciousness does not arise via the original sensory processing, but from the interactivity that arises within 500 ms between these low- and high-level modules (re-entry).[211] Furthermore, a theory of consciousness called reduced functionalism, which's key premise is that only causal currents matter in determining conscious contents, goes along with reciprocal corollary, which on her part dictates that only reciprocal currents matter. Although these two statements provide the functional basis of the cortical (recurrent) ANN, by which software systems implementing such receive the ability of solving complex classification and time-series prediction problems, this will not be discussed here in detail, as theories of consciousness are discussed in chapter 9. Katz proposed the framework depicted in Figure 59 – Causal brain influence types.

210 Neukart Florian, Moraru Sorin-Aurel, Grigorescu Costin-Marius, Szakazs-Simon Peter (2012): Cortical Artificial Neural Networks and their Evolution - Consciousness-inspired Data Mining. Proceedings of OPTIM 2012

211 Edelman G. (2001): Naturalizing consciousness: A theoretical framework; Proceedings of the National Academy of Sciences USA, 100, 5520 - 5524

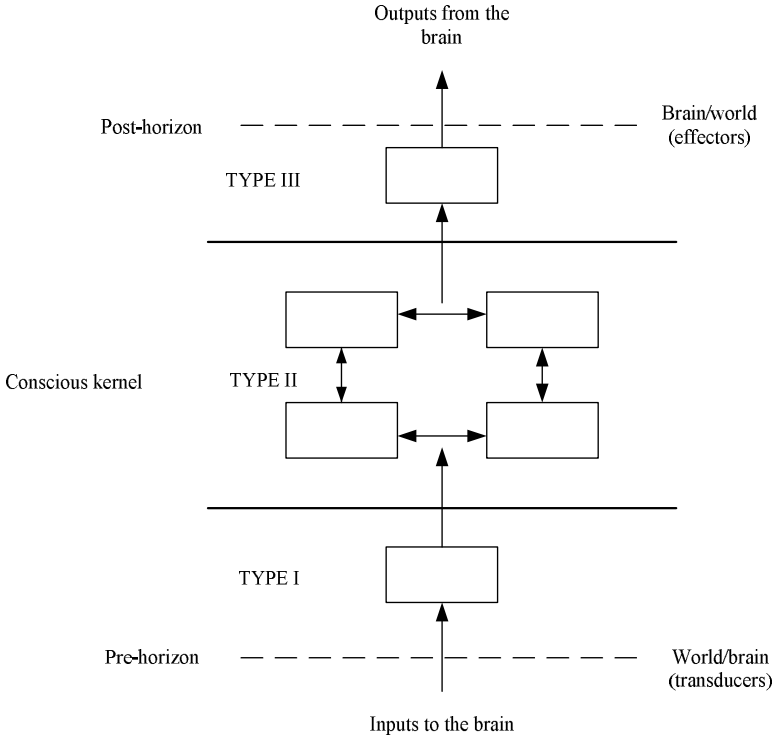

Figure 59 – Causal brain influence types[212]

Summing up, type I causal influences are feed-forward, and operate at the pre-horizon of what may be termed the conscious kernel, meaning that they can influence the contents of consciousness by affecting the firing of neurons in this kernel, but are not themselves generators of conscious content. Type II neural units do not feature this property, and therefore are potentially within the conscious kernel, which gives them the properties of re-entry, synchrony, and accessibility. Type III neurons in the post-horizon of consciousness are influenced by type II ones, and like type I ones they do not have re-entry into the conscious kernel, and can therefore not be part of consciousness.[213]

Especially when trying to solve classification or time-series prediction problem statements, commonly applied ANN structures, like feed-forward or recurrent MLPs tend to show a decreased performance and accuracy when dealing with manifold datasets containing

212 Katz Bruce F. (2011): Neuroengineering the future - Virtual minds and the creation of immortality; Massachusetts: Infinity Science Press LLC, p. 136

213 Katz Bruce F. (2011): Neuroengineering the future - Virtual minds and the creation of immortality; Massachusetts: Infinity Science Press LLC, p. 136

numerous input (predictors) and/or target-attributes. This is independent from the applied learning methods, activation functions, biases, etc... The creation of cortices in combination with an intense increase of the number of weights within an ANN allows eluding this, as explained below.

4.3.1 Structure

Back to computational intelligence, this resulted in the idea for cortical ANNs, in which processing is not only carried out by a single horizontal hidden layer (or cortex), but by several parallel hidden layers. The input is processed feed-forward to every single cortex of the ANN. The pre-cortical ANN-scheme (Figure 60 – Pre-cortical artificial neural network structure) helps to understand this special ANN type.

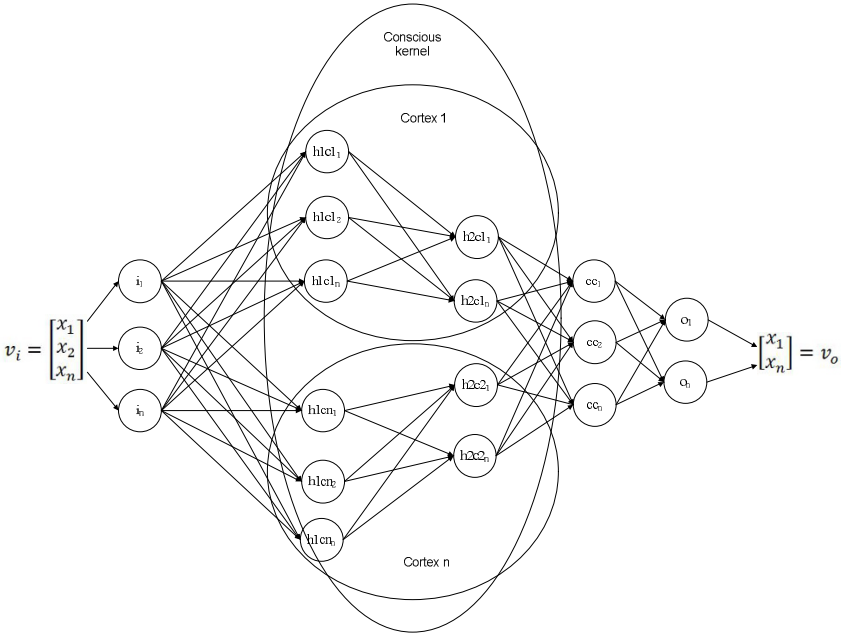

Figure 60 – Pre-cortical artificial neural network structure

The first step to the implementation is the definition of the cortices, which has to be considered within a software-side implementation. The above scheme describes the segregation of both cortices 1 and n, each commanding over two hidden layers. The input is processed from the input neurons to every single neuron of each cortex. By comparison to Figure 59 – Causal brain influence types, the input neurons act as the sensory field input, or the input to the brain, which are the neurons of the type I. The processing within the conscious kernel of the ANN happens in the independent cortices, which is in the pre-cortical ANN structure only feed-forward. In Katz's picture, these are the type II neurons. After

having been processed by each of the cortices, the cortex outputs are processed into a common cortex layer, which is the last processing step before being passed over to the output layer, or in terms of reduced functionalism, to the type III neurons.

However, in the introduction I mentioned the re-entry, which goes along with reduced functionalism, the theory of mind by which the development of the cortical ANN has been inspired. Re-entry states that the information is processed within the conscious kernel and not just passed through. In terms of ANN development this means that there are recurrent layers required for processing, which leads to the slightly more complex scheme of cortical ANNs (Figure 61 – Cortical artificial neural network structure).

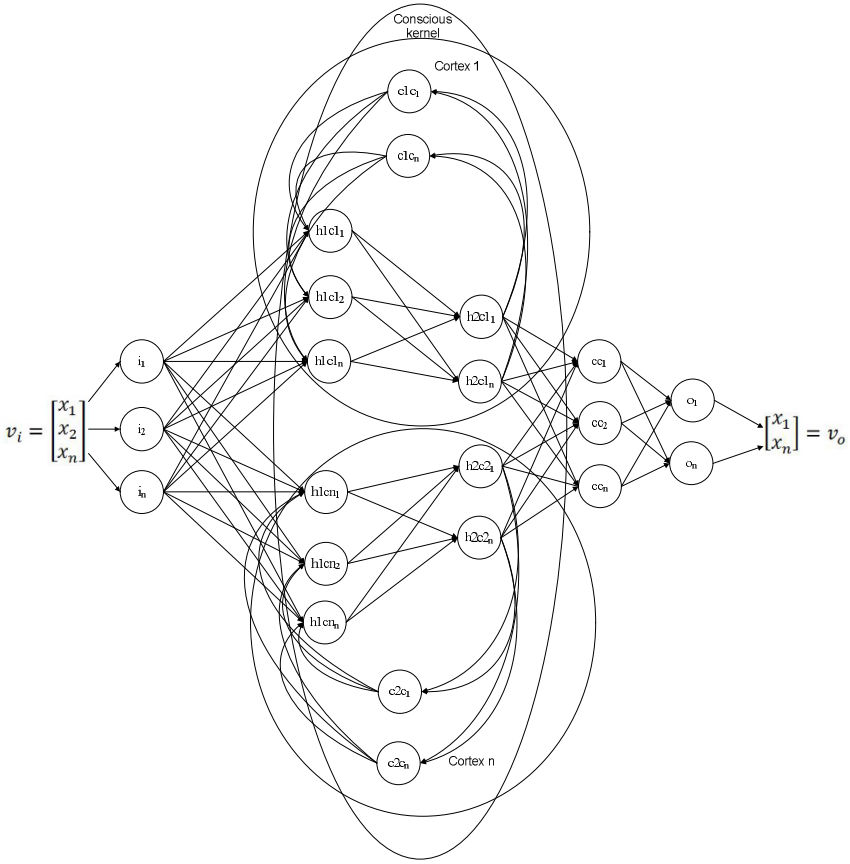

Figure 61 – Cortical artificial neural network structure

The scheme shows that, compared to the pre-cortical ANN structure, the cortical ANN features state layers connected from the second cortical hidden layer to the first cortical

hidden layer of each cortex, which allows the processing cortex to possess a kind of memory. Within the conscious kernel therefore not only causal information processing is carried out, but also recurrent processing of the information. Mathematically, the processing from the input through a cortical ANN consisting of just one cortex, which in fact does not make sense, but is the most basic representation of such an ANN, is

$$x_t = h_4(w_0 + \sum_{l=1}^{n} w_l\, h_3(w_{0l} + \sum_{k=1}^{m} w_j\, h_2(w_{0k} + \sum_{j=1}^{l} w_j h_1(w_{0j} + \sum_{i=1}^{p} w_{ij} x_{t-1} + \sum_{m=1}^{l} \varpi_{mj} z_{m,t-1})))) + \epsilon_t \quad (4\text{-}38)$$

where

$$z_{m,t} = w_{0m} + \sum_{i=1}^{p} w_{im} x_{t-i} + \sum_{k=1}^{l} \varpi_{kj} z_{k,t-1}, m = 1, \dots, l \quad (4\text{-}39)$$

where the input layer has p inputs $x_{t-1}, \dots, x_{t-p}$, the hidden layer 1 has l hidden nodes, the hidden layer 2 has m hidden nodes, the hidden layer 3 has n hidden nodes and the output with y outputs $y_{t-1}, \dots, y_{t-p}$, for the output layer x_t. The layers are fully connected by weights, where w_{ij} is the i^{th} input for the j^{th} node in the hidden layer, whereas w_j is the weight assigned to the j^{th} node in the hidden layer for the output. w_{0j}, w_{0j}, w_{0k} and w_{0l} are the biases, h_1 ... h_n are activation functions. The connections from recurrent layer two to the hidden layer are also weighted in the cortical ANN, therefore, $\varpi_{mj} <> 1$. For every single cortex, the above equation holds and has to be calculated through to receive the proper output, whereby the fourth hidden layer plays a special role as its neuron number directly depends from the number of cortices, the number of neurons in the last hidden layer before the common cortex layer (or the fourth hidden one in the above representation) each cortex commands over, and the number of output neurons. So, if the number of cortices $n_c = 2$, and each last hidden layer before the common cortex layer of every cortex consists of 3 neurons and the number of output neurons is 2, the number of neurons in the common cortex layer is 6, according to the last formula in 4.3.1.2 Number of neurons.

Cortical ANN have proved to perform well when their neurons implement either the sigmoid activation function for only positive scaling of input values and the hyperbolic tangent activation, when the output has to be scaled between positive and negative values.

4.3.1.1 Cortices

Each cortex of the cortical ANN in the scheme consists of two hidden layers and one state layer, but the number is of course not limited to that. The cortices receive their input from the input layer and double-process the information, as it is, after having passed the second, hidden cortex layer, processed to the common cortical layer, which serves as some kind of reception unit for the outputs of every cortex. A single output layer, especially when consisting of a small number of neurons, would not be able to successfully process the output of two or more cortices.

4.3.1.2 Number of neurons

The number of neurons of each layer within a cortical ANN can be determined by calculations based on the input and output neurons. The number of the first hidden cortex layer is determined with

$$n_{h1cn} = \left(\frac{n_{input}}{3} * 2\right) + n_{output} \quad (4\text{-}40)$$

where n_{input} represents the number of input neurons and n_{output} the number of output neurons. However, the empiric research has shown that for some problems it might be necessary to increase the number of hidden neurons and that – 20 neurons for the first hidden layer mark a good start then. In a similar way, the number of the second hidden cortex layer is determined:

$$n_{h2cn} = \left(\frac{n_{h1cn}}{3} * 2\right) + n_{output} \quad (4\text{-}41)$$

where n_{h1cn} represents the number of neurons in the first cortical hidden layer and n_{output} again the number of output neurons.

The number of neurons within the state layers in cortical ANNs always equals n_{h2cn}, although the processing does not happen one to one (each neuron in the source has on neuron in the target layer), but weighted from every source neuron to every target neuron, which has proofed to be the best solution during verification. The number of neurons in the common cortex layer strongly depends from the number of cortices and the number neurons of each cortex' second hidden layer:

$$n_{cc} = \left(\frac{n_{h2cn}}{3} * 2\right) * n_c + n_{output} \quad (4\text{-}42)$$

where n_{h2cn} represents the number of the second cortical hidden layer, n_c the number of cortices within the ANN and n_{output} the number of output neurons.

4.3.1.3 Synapses

Partially, cortical ANNs are recurrent (in their cortices), but what makes it different from most practically applied ones is that not only every feed-forward connection in the ANN is weighted, but also the recurrent ones. The high number of weighted connections and hidden layers within a cortical ANN leads to increased processing time, but also to the possibility of understanding and processing Data Mining problem statements of high complexity, containing numerous different predictors and target attributes.

4.3.2 A generic cortical artificial neural network

As we seek to find a generic neural network architecture, capable of processing strongly varying input with overseeable training effort, the structure of the cortical ANN has been slightly adapted: the ANN structure capable of solving most of the presented problem statements without the absolute need to change the structure or source code has been determined according to empiric research results (Figure 62 – Cortical artificial neural network structure).

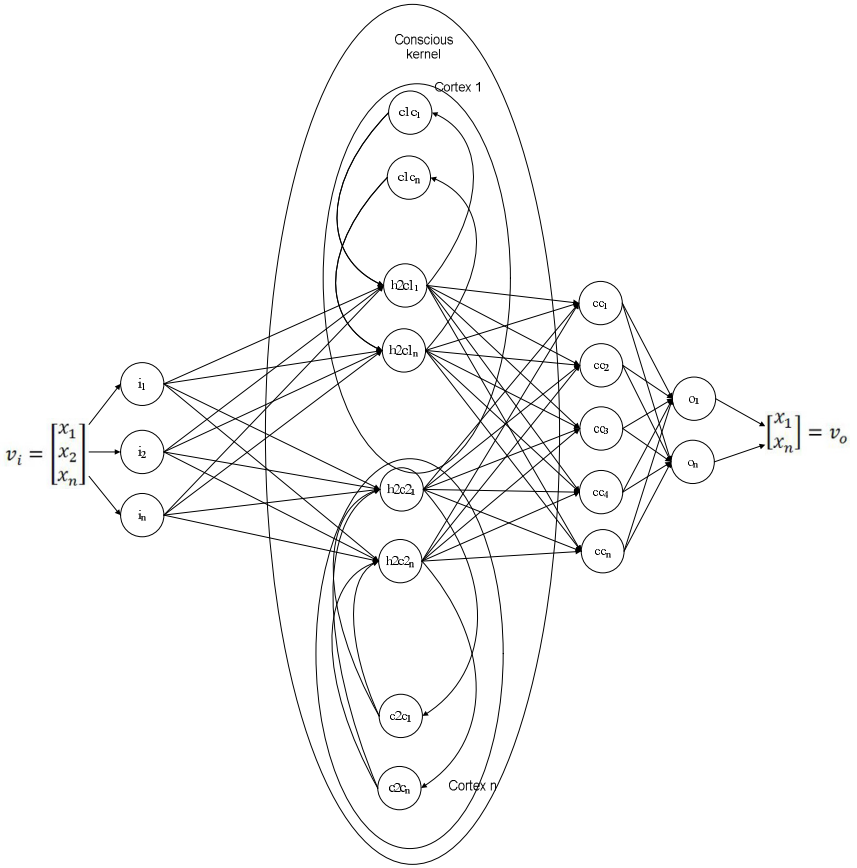

Figure 62 – Cortical artificial neural network structure

Independent from the general cortical ANN, I personally made positive experiences for predicting share prices using cortical ANNs commanding only over one hidden layer per each cortex. Another difference is that indeed all connections are weighted, and the connections from the cortex layers to the state layers are one to one, not one to all. This dictates that the state layers must have the same number of neurons as the cortex layers. Furthermore, this structure demands to determine the number of neurons slightly different from the common cortical artificial neural network. The number of the cortex neurons for each cortex is determined in the same way as the first hidden layer neuron determination in the common cortex ANN:

$$n_{hc} = \left(\frac{n_{input}}{3} * 2\right) + n_{output} \qquad (4\text{-}43)$$

However, the common cortex layer neurons are calculated according to the equation

$$n_{cc} = (n_{hc} * 2) + n_{output} \quad (4\text{-}44)$$

Moreover, the common cortex layer and the output layer neurons apply the hyperbolic tangent activation function, regardless of the sign of the input. The other layer's neurons make use of the sigmoid activation function if the input sign is positive; else they also use the hyperbolic tangent function.

4.3.3 Purpose

As already stated in the introduction, ANNs with only two hidden layers are capable of representing an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy.[214] This leads to the question of the purpose of such a complex ANN structure, commanding at least over three hidden layers and moreover, over at least two cortices containing those hidden layers. When implementing a cortical ANN, or an initial population of such ANNs (as genetic evolution is again an efficient way to finding the best solution for a problem statement when dealing with such complex ANN structures), and teaching them how to solve the XOR problem, the learning success will not be comparable to an ANN with just one hidden layer, learning e.g. with resilient propagation, as it will be much worse. Assuming that the fitness function of the ANN is the root mean squared error

$$x_{rmse} = \sqrt[2]{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \quad (4\text{-}45)$$

the ANN will either be not able to achieve the allowed error within the allowed number of iterations or will at least take much longer to do so. Furthermore, the performance will be much worse than compared to the mentioned single-hidden layer ANN. However, the strength of cortical ANNs advances when source data contains manifold predictors and shall be classified or predicted into various targets, as our brain uses to classify input. As the number of weights within cortical ANNs is high, understanding of very complex source data and the possibility of learning to classify is, according to complex weight matrix structures, possible. Current ANN research does not focus on complex ANN structures, as huge amounts of source data not being able to be processed by just one ANN, are clustered and processed with several ANNs. It does take much until the necessity of applying several cortical ANNs for allowing a software system to understand a problem statement arises, regardless to the problem's complexity. Especially within data mining systems, the overall complexity of a solution is decreased, although the overall solution consisting of one cortical ANN is more complex than the single parts of the first one.

Thus, at a first glance, there would not be the need for creating cortices within an ANN, but practical application showed that the processing of data from the input neurons into two or more independent cortices results in more accurate results. This is because the two cortices process input independently from each other, and the resulting output of both is merged. When the calculations in one cortex are not as accurate as they should be, this is compensated

214 Heaton Research (2005 - 2011): The number of Hidden Layers [2011-28-09]; URL: http://www.heatonresearch.com/node/707

by the other cortices (not training algorithm can ensure that the resulting output is as good as possible, owed to the randomness the evolution of ANNs). The effect of compensation shows its advantage to the full extent when the cortical ANN includes at least 3 cortices, as it is very unlikely that the majority of the cortices incorrectly process information.

4.3.4 Evolution and weight initialization

As cortical ANNs are complex structures, the above described formulas for determining the optimal number of neurons for each layer are a good suggestion for determining an initial condition before evolution, but do not always perform best. This is, where structural evolution can again be applied for determining the optimal solution for a problem statement, which is slightly more complex in cortical ANNs than in feed-forward ones. The evolution of the structure of a cortical ANN is even more complex than the evolution of the SHOCID recurrent ANN, which also contains hidden layers to be adapted and added or removed. Structural evolution of a cortical ANN may contain:

– Addition or removal of cortical layers
– Addition or removal of cortical layer neurons, first and second, which leads to the adaption of the neuron number in the common cortex layer
– Addition or removal of cortices, which leads to the adaption of the neuron number in the common cortex layer

When evolving a cortical ANN I suggest to only carry out the addition or removal of cortical neurons and the addition or removal of cortices (the maximum number of cortices is restricted to 4). Furthermore, the number of hidden layers within a cortex remains 2 and is not in- or decreased. Therefore, the evolution has influence on the whole structure of the ANN, but only the cortex structure is adapted. Similar adaptations occur within the human brain when learning is carried out – unused synaptic connections are weakened or removed, and on the contrary often used connections are strengthened.

The structure-part within the evolutionary process on cortical ANNs does only affect the number of hidden neurons and layers. However, in theory there are further possibilities of evolution currently not having been tested in a way so that a statement can be made. Possible structural modifications / mutations may also contain

– Asynchronous adaption of the cortices
  - Asynchronous in- or decrease of layers
  - Asynchronous in- or decrease of cortex neurons
– Hybrid activation functions per cortex (within a cortex?)

Weight initialization for deep cortical ANN structures can be conducted as described in 4.2 Deep belief artificial neural network. As simulated annealing has proofed to come to the most accurate solutions in short time within cortical ANNs, the algorithm for evolving cortical ANNs may be follows:

Start

1. Creation of initial ANN with nc_{min} cortices, containing nhc_{min} hidden cortex layers and $nh1_{min}$ hidden neurons in the first cortical hidden layer and $nh2_{min}$ neurons in the second cortical hidden layer. Additionally, the initial common cortex layer is created

with ncc_{min} neurons.

2. ***Repeat***
   a) Calculate the network output for the value $d \in D$
   b) Evaluate the fitness of each neuron:
   c) Calculate the error $\delta_{d \in D}$ for each output neuron $d \in D$

$$\delta_{d \in D=}(desired_{d \in D} - actual_{d \in D}) actual_{d \in D}(1 - actual_{d \in D})$$

   d) Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d \in D} (w_{hidd \in D} \delta_{d \in D})$$

   e) Repeat
      i. Create new ANN and randomize weights according to T
      ii. Calculate the error $\delta_{d \in D}$ for each output neuron $d \in D$

$$\delta_{d \in D=}(\text{desired}_{d \in D} - \text{actual}_{d \in D})\text{actual}_{d \in D}(1 - \text{actual}_{d \in D})$$

      iii. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = \text{actual}_{hid}(1 - \text{actual}_{hid}) \sum_{d \in D} (w_{hidd \in D} \delta_{d \in D})$$

      iv. Compare solutions according to

$$\Delta_{C(S')C(S)} = C(S') - C(S)$$

      v. If $C(S')$ is better than $C(S)$, replace $C(S)$
   f) Until max tries for current temperature reached
   g) Decrease temperature by

$$\varphi_{C(S')C(S)} = T * e^{\frac{ln\left(\frac{S}{e}\right)}{c-1}}$$

3. ***Until lower temperature bound reached***
4. Determine the quality value of the initial network architecture
5. Store the quality value of the initial network architecture in the quality array
6. ***Repeat***
   a) if $nh1 < nh1_{max}$:
      i. Increase $nh1$ by c_{h1}
   b) Calculate ncc
   c) Creation of ANN population of n_p ANNs with c cortices, containing nhc hidden cortex layers and $nh1$ hidden neurons in the first cortical hidden layer and $nh2$ neurons in the second cortical hidden layer. Additionally, the initial common cortex layer is created with ncc neurons.
   d) Randomization of weights and threshold values.
   e) Repeat
      i. Repeat
         1. Calculate the network output for the value $d \in D$
         2. Evaluate the fitness of each neuron:
         3. Calculate the error $\delta_{d \in D}$ for each output neuron $d \in D$

$$\delta_{d \in D=}(desired_{d \in D} - actual_{d \in D}) actual_{d \in D}(1 - actual_{d \in D})$$

4. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d \in D} (w_{hidd \in D} \delta_{d \in D})$$

5. Repeat

a. Create new ANN and randomize weights according to T

b. Calculate the error $\delta_{\mathrm{d\in D}}$ for each output neuron $\mathrm{d} \in \mathrm{D}$

$$\delta_{\mathrm{d\in D}=}(\mathrm{desired}_{\mathrm{d\in D}} - \mathrm{actual}_{\mathrm{d\in D}})\mathrm{actual}_{\mathrm{d\in D}}(1 - \mathrm{actual}_{\mathrm{d\in D}})$$

c. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{\mathrm{hid}} = \mathrm{actual}_{\mathrm{hid}}(1 - \mathrm{actual}_{\mathrm{hid}}) \sum_{\mathrm{d\in D}} (\mathrm{w}_{\mathrm{hidd\in D}} \delta_{\mathrm{d\in D}})$$

d. Compare solutions according to

$$\Delta_{C(S')C(S)} = C(S') - C(S)$$

e. If $\mathrm{C(S')}$ is better than $\mathrm{C(S)}$, replace $\mathrm{C(S)}$

6. Until max tries for current temperature reached

7. Decrease temperature by

$$\varphi_{C(S')C(S)} = T * e^{\frac{ln\left(\frac{S}{e}\right)}{c-1}}$$

ii. Until lower temperature bound reached

f) ***Until $\mathbf{nc_{max}}$ is reached***

g) Determine the quality value of the current network architecture

h) Store the quality value of the current network architecture in the quality array

i) Verification

i. Present each verification data set

ii. Increase error array for RMSE > r_{max} for each verification data set by 1

j) if $\mathbf{nh1} == nh1_{max}$:

– Increase $nh2$ by c_{h2}

7. Until $\mathbf{nh2_{max}}$ is reached

8. Comparison of stored ANN solutions by quality

9. Present best solution

End

Algorithm 12 - Evolution of cortical ANN

Breakdown:

nhc_{min}/nhc_{max}: Minimum/maximum number of hidden cortex layers
$nh1_{min}/nh1_{max}$: Minimum/maximum number of neurons of *nhc1*
$nh2_{min}/nh2_{max}$: Minimum/maximum number of neurons of *nhc2*
n_c: Number of cortices
ncc_{min}: Minimum number of common cortex neurons
c_n /c_m: Hidden neuron/layer counters

$d \in D$: Actual input
$\delta_{d \in D}$: Error for input $d \in D$ at output neuron o_n in the current generation
δ_{hid}: Error for hidden neuron
r_{max}: Allowed RMSE

The neuron numbers $nh1_{min}$, $nh1_{max}$, $nh2_{min}$ and $nh2_{max}$ are determined according to the equations explained in 5.3 Structural evolution. It has to be considered that the layer directly upstream to the layer which's neurons have to be calculated, serves as the input layer, where the output layer is always the last layer in the ANN. Otherwise, all hidden layers would receive the same number of neurons.

4.4 SHOCID recurrent artificial neural network

The SHOCID recurrent artificial neural network[215] (SRANN) is another deep learning architecture that has been conceived and specified within the SHOCID research-project, with the purpose of performing classification and time-series prediction tasks on generic input values. In order to give it better long short term-memory, recurrent connections from the output layer to the hidden layers and from the hidden layers to itself exist. Thus, this structure is used for making time-dependent predictions, e.g. predicting the trajectory of a person, or in short: what's going to happen next. Our brain has evolved a series of pattern recognizers and predictors to do exactly that: predict where prey or predators are going to move and what their next actions are. The SRANN is, as well as the Elman and Jordan ANNs, a recurrent ANN, processing its input values forward and able to make use of several learning methods. However, as the calculations within SRANNs are computationally more intensive than in other ANNs, the preferred learning method is based on a genetic adaption of weights (3.3.3.7 Genetic learning). In difference to Elman and Jordan ANNs, the SRANN is also able to apply structural evolution,[216] described at 5.3 Structural evolution, a method similar to NEAT (3.3.3.9 NeuroEvolution of augmenting topologies (NEAT)) carrying out not only an adaption of weights, but also an in- and decrease of neurons and layers within an ANN for determining the optimal solution to a data mining problem statement.

4.4.1 Structure

Schematically, the SHOCID RANN's structure is as depicted in Figure 63 – SHOCID recurrent artificial neural network single hidden layer.

215 Neukart Florian (2013): Accuracy through Complexity - One Step further in Time-Series Prediction and Classification, Knowledge Discovery in Databases.
216 Neukart Florian. et al. (2011): Problem-dependent, genetically evolving Data Mining solutions; Proceedings of DAAAM 2011, 2011 22nd World Symposium

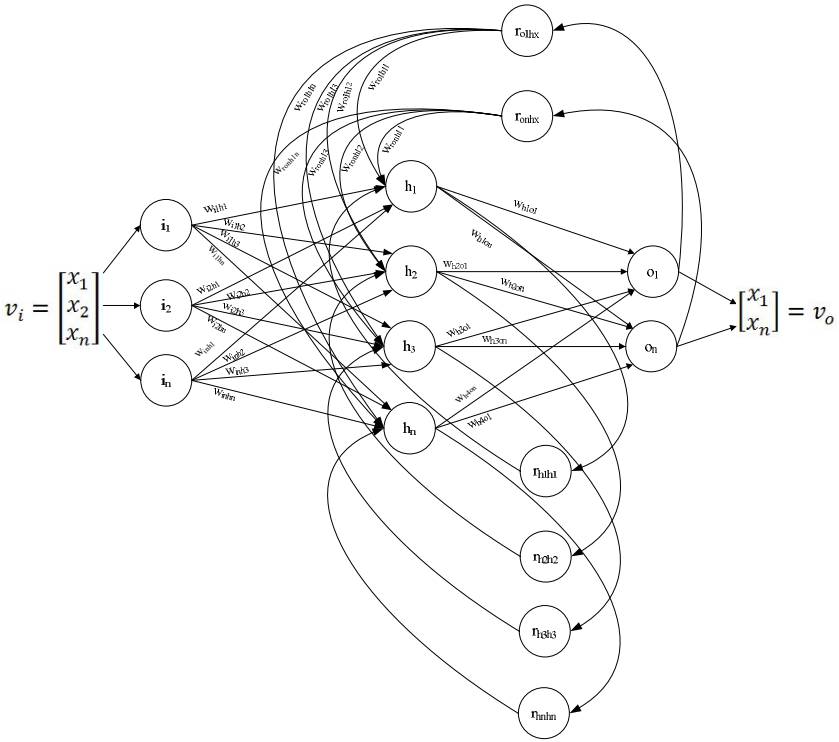

Figure 63 – SHOCID recurrent artificial neural network single hidden layer

The ANN is somewhat similar to the Elman and Jordan recurrent ANNs, with the difference that it unites architectural concepts of both. The above picture shows that the network, in case it contains just one hidden layer, has two recurrent layers.

4.4.1.1 Recurrent layer one

The recurrent layer one, represented by the neurons r_{o1hx} and r_{onhx}, processes the values of the output neurons back to the hidden layer. Each hidden neuron h_x from the hidden layer receives input from each neuron r_{oxhx} of the recurrent layer one. The connections from the recurrent layer to the hidden layer contain weights, which are adapted after a learning iteration. The recurrent layer one therefore must feature exactly the same number of neurons as the output layer. The weight adaption is necessary as the output values from the output layer shall not cause the hidden neurons to fire in the mass the hidden layer two does.

4.4.1.2 Recurrent layer two

The recurrent layer two, represented by the neurons $r_{h1h1},\dots,r_{hnhn}$, processes the output values of the hidden layer back to itself and provides them as input, but without making use of weights. This not only increases the firing rate of the hidden neurons, but also increases the probability of an adaption of weights between the hidden layer and the output layer intensely. In difference to the recurrent hidden layer two, the number of the neurons in the recurrent layer is the same as the number of the neurons in the hidden layer. Furthermore, the processing of the output values in this layer is nearly the same as in an Elman ANN, meaning that the neuron r_{h1h1} only propagates its output value back to the neuron h_1, but without weights being adapted or taken into consideration inside of a leaning iteration.

The hidden layer therefore receives input from both recurrent layers, which, in comparison to classical Jordan and Elman ANNs, increases the network's memory, what is important for time series prediction, but also leads to better classification results by increased weight adaption within the ANN. Mathematically, the SHOCID RANN can be represented by

$$x_t = h_2(w_0 + \sum_{j=1}^{l} w_{ji} y_{t-1} h_1(w_{0j} + \sum_{i=1}^{p} w_{ij} x_{t-1} + \sum_{m=1}^{l} \varpi_{mj} z_{m,t-1} + \sum_{q=1}^{y} \varpi_{yj} z_{y,t-1})) + \epsilon_t \quad (4\text{-}46)$$

where

$$z_{m,t} = w_{0m} + \sum_{i=1}^{p} w_{im} x_{t-i} + \sum_{k=1}^{l} \varpi_{kj} z_{k,t-1}, m = 1,\dots,l \quad (4\text{-}47)$$

and

$$z_{y,t} = w_{0y} + \sum_{i=1}^{p} w_{iy} x_{t-i} + \sum_{k=1}^{l} \varpi_{kj} z_{k,t-1}, y = 1,\dots,l \quad (4\text{-}48)$$

where the input layer has p inputs $x_{t-1},\dots,x_{t-p}$, the hidden layer has l hidden nodes and the output with y outputs $y_{t-1},\dots,y_{t-p}$, for the output layer x_t. Layers are fully connected by weights, where w_{ij} is the i^{th} input for the j^{th} node in the hidden layer, whereas w_j is the weight assigned to the j^{th} node in the hidden layer for the output. $w_0$and w_{0j} are the biases, h_1 and h_2 are activation functions. The connections from recurrent layer two to the hidden layer are not weighted in the SRANN, therefore $\varpi_{mj} = 1$, but stay in the formula as weighting may become useful in not yet discovered use cases. The connections from the recurrent layer one $\varpi_{yj} z_{y,t-1}$ to the hidden layer are, as described beforehand, weighted.

The SRANN is not limited in terms of neurons or layers, meaning that such also can become multi-hidden layer ANNs, which is also the reason why structural evolution can be applied (Figure 64 – SHOCID recurrent artificial neural network multi hidden layer).

The difference to the single hidden layer SRANN (SL-SRANN) is that per each additional hidden layer two additional recurrent layers appear. The processing from the output layer o to the second hidden layer h_2 via the recurrent layer r_{oxh2x} and from the second hidden layer $h_2$via the recurrent layer r_{h2xh2x} to itself is exactly the same as the one described at the SL-SRANN. The recurrent layer r_{h2xh1x} however, uses the output of the hidden layer two as input and processes it, after weighting the values, back to the hidden layer one. Therefore, it features the same number of neurons as the hidden layer two h_2. The recurrent layer r_{h1xh2x} applies the same un-weighted processing as the layer r_{h2xh2x} for propagating its output values back to itself.

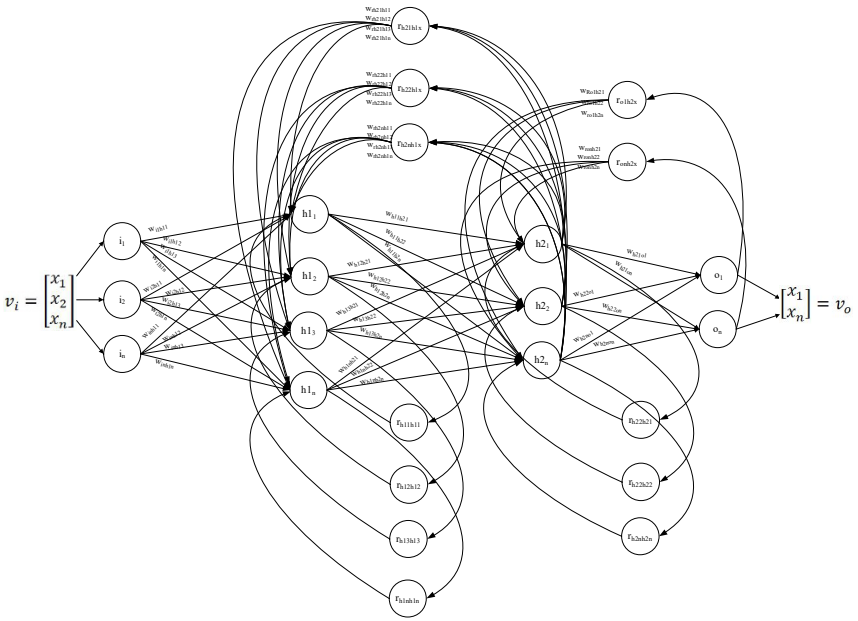

Figure 64 – SHOCID recurrent artificial neural network multi hidden layer

4.4.1.3 Number of neurons

The number of neurons of each layer within the SHOCID RANN is determined by calculations based on the input and output neurons. In case of a single hidden layer SRANN, the number of the hidden layer is determined by

$$n_{h1} = \left(\frac{n_{input}}{3} * 2\right) + n_{output} \qquad (4\text{-}49)$$

where n_{input} represents the number of input neurons and n_{output} the number of output neurons. In a similar way, the second hidden layer's number of neurons, in case of a two-hidden-layer SRANN, is determined:

$$n_{h1} = \left(\frac{n_{h1}}{3} * 2\right) + n_{output} \qquad (4\text{-}50)$$

where n_{h1} represents the number of neurons in the first hidden layer and n_{output} again the number of output neurons.

As indicated beforehand, the number of neurons within the state layers in SHOCID RANNs depends from the submitting layer. The recurrent layer one commands over the same number of neurons as the output layer, as the input is processed one-to-one and therefore no other solution is possible. The same holds for the second recurrent layer, which receives the input

one-two-one from the hidden layer and therefore commands over the same number of neurons as the latter one. Hence,

$$n_{r1} = n_o \tag{4-51}$$

and

$$n_{r2} = n_h \tag{4-52}$$

4.4.1.4 Synapses

The SRANN is a recurrent ANN, but what makes it different from most practically applied ones is that it both commands over weighted, recurrent connections and un-weighted, recurrent connections. As the scheme shows, the connections from the output layer to the recurrent layer one are not weighted, in contrast to the connections from the recurrent layer one to the hidden layer. At a first glance, this may look the same as in a Jordan recurrent ANN. However Jordan recurrent ANNs make use of weighted connections from output layer to the hidden layer. The same holds for the recurrent layer two and a comparison to the Elman recurrent ANN, but examined in detail the SRANN processes the input directly from the hidden layer to itself, in the contrary to the former one, which commands over weighted connections from the hidden layer to recurrent one.

The high number of weighted connections and hidden layers within the SRANN may lead to increased processing time, but also, and as the cortical ANN, to the possibility of understanding and processing data mining problem statements of high complexity, containing numerous different predictors and target attributes.

4.4.2 Purpose

As indicated repeatedly, ANNs with only two hidden layers are capable of representing an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy.[217] However, depending from the presented time-series prediction problem statement, one must derive from testing if Jordan or Elman recurrent ANNs perform better. There is no obvious way of how one can find out which structure will perform better, except through considerable experience and testing.

The SRANN avoids this, as to its structure, which is a combination of modified Jordan and Elman RANNs, the determination becomes unnecessary. In several tests it could be verified that by the application of the SRANN, time-series prediction problems can be solved more accurately than by both Jordan and Elman RANNs. Furthermore, the SRANN showed considerable performance and accuracy in solving classification problem statements, comparable to those of the cortical ANN structure. When implementing a SRANN, or an initial population of such ANNs (as genetic evolution is again an efficient way to finding the best solution for a problem statement when dealing with such complex ANN structures), and teaching them how to solve the XOR problem, the learning success is indeed comparable to

217 Heaton Research (2005 - 2011): The number of Hidden Layers [2011-28-09]; URL: http://www.heatonresearch.com/node/707

an ANN with just one hidden layer, learning e.g. with resilient propagation, although it's complexity.

Therefore, the strength of SRANNs advances not only when source data contains manifold predictors and shall be classified or predicted into various targets, but also in simple problem statements usually solved with back propagation ANNs. As the number of weights within SRANNs is numerous, complex understanding of source data and the possibility of learning to classify is, according to complex weight matrix structures, possible.

4.4.3 Evolution and weight initialization

SRANNs are complex structures, and the latterly described formulas for determining the optimal number of neurons for each layer are a good suggestion, but do not always perform best. This is, where structural evolution may be applied for determining the optimal solution for a problem statement, which is slightly more complex in SHOCID RANNs than it is in feed-forward ones. Furthermore, the evolution of the structure of a cortical ANN is complex, as the structural evolution contains:

– Addition or removal of hidden layers and the corresponding.
– Addition or removal of recurrent layers.
– Addition or removal of hidden layer neurons, first and second, and the corresponding .
– Addition or removal of recurrent layer neurons, first and second for each hidden layer.

When evolving an SRANN, I made positive experiences by only carrying out the addition or removal of hidden layers, recurrent layers and the addition or removal of the corresponding neurons, and by restricting the maximum number of hidden layers within the SRANN to 2. Therefore, the maximum number of recurrent layers is four. Therefore, the evolution has influence on the whole structure of the ANN.

Weight initialization for deep SRANN structures can be conducted as described in 4.2 Deep belief artificial neural network. Furthermore, as genetic evolution has proofed to come to the most accurate solutions in short time within SHOCID RANNs, an algorithm for evolving these ANNs is as follows:

Start

1. Creation of initial ANN population of n_p ANNs with m_{min} hidden layers, n_{min} hidden neurons for each hidden layer, and the recurrent layers one and two with the corresponding number of neurons.
2. ***Repeat***
   a) Calculate the network output for the value d ∈ D
   b) Evaluate the fitness of each chromosome:
      i. Calculate the error $\delta_{d \in D}$ for each output neuron o_n

$$\delta_{d \in D=}(desired_{d \in D} - actual_{d \in D}) actual_{d \in D}(1 - actual_{d \in D})$$

      ii. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d \in D} (w_{hid d \in D} \delta_{d \in D})$$

c) Selection of chromosomes to recombine
d) Repeat
i. Crossover of chromosomes
ii. Mutation of offspring
e) Until all selected chromosomes have been recombined
3. *Until criteria are reached*
4. Determine the quality value of the initial network architecture
5. Store the quality value of the initial network architecture in the quality array
6. Repeat
a) if $n < n_{max}$:
Increase n by c_n
b) Creation of population of x ANNs with m_{cm} hidden layers, n_{cn} hidden neurons for each hidden layer and the recurrent layers one and two with the corresponding number of neurons.
c) Randomization of weights and threshold values of each chromosome.
d) Repeat
i. Calculate the network output for the value $d \in D$
ii. Evaluate the fitness of each chromosome:
8. Calculate the error $\delta_{d \in D}$ for each output neuron o_n

$$\delta_{d \in D=}(desired_{d \in D} - actual_{d \in D}) actual_{d \in D}(1 - actual_{d \in D})$$

9. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d \in D} (w_{hid d \in D} \delta_{d \in D})$$

iii. Selection of chromosomes to recombine
iv. Repeat
10. Crossover of chromosomes
11. Mutation of offspring
v. Until all selected chromosomes have been recombined
e) ***Until criteria are reached***
f) Determine the quality value of the current network architecture
g) Store the quality value of the current network architecture in the quality array
h) Verification
i. Present each verification data set
ii. Increase error array for RMSE > r_{max} for each verification data set by 1
i) if $n == n_{max}$:
Increase m by c_m
7. Until m_{max} is reached
8. Comparison of stored ANN solutions by quality
9. Present best solution

End

Algorithm 13 - Evolution of SRANN

Breakdown:

n_p: Number of chromosomes
m_{min}/m_{max}: Minimum/maximum number of hidden layers
n_{min}/n_{max}: Minimum/maximum number of neurons per m_n
c_n/c_m: Hidden neuron/layer counters
$d \in D$: Actual input
$\delta_{d \in D}$: Error for input $d \in D$ at output neuron o_n in the current generation
δ_{hid}: Error for hidden neuron
r_{max}: Allowed RMSE
n_{min} and n_{max} are determined on the basis of the number of input neurons. The equation

$$n_h = \left(\frac{n_{input}}{3} * 2\right) + n_{output} \tag{4-53}$$

delivers the number of the hidden neurons, which form the basis for the determination of the minimum and maximum number to which evolution shall come to:

$$n_{min} = n_h - \left(\frac{n_h}{3}\right) \tag{4-54}$$

and

$$n_{max} = n_h + \left(\frac{n_h}{3}\right) \tag{4-55}$$

The same holds for the second hidden layer, if evolution is applied that far, with the difference that all neuron numbers of the evolution of the first layer are stored into an array, which serve as a calculation basis for n_{min} and n_{max}. If, e.g., one solution of the first hidden layer evolution process has three hidden neurons, the evolution of the second hidden layer takes this as n_h and calculates n_{min} and n_{max} anew. The algorithm also shows that the first run to determine the initial ANN's quality has to be carried out at the beginning, as the evolution starts immediately afterwards.

4.5 Summary

Within the chapter some more sophisticated artificial neural network structures have been discussed, as well as and according to the research results the most suitable learning algorithms and paradigms for these. The verification of the new paradigms showed that the combination of computational intelligence with typical data mining problem statements features enormous potential. I mention data mining very often, as within this field numerous practices are applied for achieving what the human brain does achieve with ease, such as classification, prediction or clustering. On the other hand, practical application of actual CI-paradigms shows that are capable of solving complex problem statements that humans cannot do that quickly with their brains, such as pattern recognition in numerical and unstructured date. However, in finding a solution to such problems computers have always been better than humans, on the contrary to e.g. interpretation of context (what is someone actually doing?) by observation, which actually is no problem for a human brain. From what the research has shown, the rudimentary simulation of human conscious processing opens new possibilities in ANN research, but the search for software capable of producing conscious content only starts at this point.

Apart from that, stored data and the need for evaluating it will increase, and time required for implementing such will proportionally decrease, as software systems capable of learning and understanding problem statements come to the fore, especially when considering the current ‘Big Data’ and AI-hype cycle.

5 Advanced nature-inspired evolution and learning strategies

From what we have understood about the inner workings of the human brain, our neural networks are initialized while we are developing in our mothers' wombs, but not well configured to work in the outside world right after we are born. Learning in the human brain, which refers to building up neural networks over the neocortical hierarchies, involves a lot more than initializing connections and increasing the connection strength between neurons. Depending from the neuron type, even different types of electrochemical signals (amplitude, frequency, period) may have no effect, result in firing bursts or a single activation only (3.3.2.6.3 Spiking artificial neural networks). The facts that we have not yet a hundred percent understood how learning in the human brain happens and that the beauty of mathematics very often allows us to solve problems in many ways, the explanation of the following nature-inspired training algorithms should show that we do not necessarily need to exactly copy what is happening in the human brain in order to achieve similar or even identical capabilities. Some training algorithms have already been introduced in chapter 3, but the following ones are special in the sense that they copy nature and evolution even more in order to achieve a totally different task, namely training neural networks instead of e.g. fighting viruses or simulating parasitic behavior. Especially complex ANN architectures are useless without benefiting from a suitable learning approach. Within this chapter some algorithms Nature already provided us with have been translated into the training mathematics for artificial neural networks.

Training ANNs means to solve an optimization problem, and the purpose for describing these algorithms is to show that Nature has developed very sophisticated solutions to such kind of problems over millions of years of evolution. Although learning and evolution go along, evolution can either be

– a learning approach, or
– a supportive process for learning, when evolving the architecture.

How can that be? This is on the one hand, because genetic algorithms may be used for learning, which classifies an artificial neural network into an evolutionary learning one. If on the other hand an ANN is capable of adapting its architecture by a supportive architecture evolution process like structural evolution or NEAT, this does not necessarily mean that it has to learn by the application of genetic algorithms.

5.1 Transgenetic NeuroEvolution

As transgenetic NeuroEvolution[218] via the inclusion of both symbionts and horizontal gene transfer is a new approach in artificial neural network learning, it requires an explanation in detail. Transgenetic algorithms are used for performing a stochastic search by simulating endosymbiotic interactions between a host and a population of endosymbionts as well as

218 Neukart Florian et al. (2012): Transgenetic NeuroEvolution. Proceedings of OPTIM 2012

F. Neukart, *Reverse Engineering the Mind*, AutoUni – Schriftenreihe 94, DOI 10.1007/978-3-658-16176-7_5

information exchange between the host and endosymbionts by agents[219]. By the combination of one of the already discussed learning approaches with a host organism, serving as genetic pool, and transgenetic vectors, both learning performance and accuracy can be increased to a considerable degree. A further advantage is that the application of transgenetic vectors massively increases the chance of fulfilling the stopping criteria, as even learning algorithms like back propagation cannot oscillate or get stuck in local minima due to the inescapable transfer of host genetic material.

5.1.1 Fundamentals

The biological fundamentals of transgenetic algorithms for solving NP-hard combinatorial problems have already been elucidated in detail by Goldbarg & Goldbarg[220], but need to be adapted for ANN evolution. As in nature, the relationship between a host organism and a symbiont may be an advantage for both organisms, and the descendants share genetic material of both the host and the symbiont. The host organism here mostly serves as genetic database and contributes to the final solution with gene sequences. The endosymbiont is the real solution, which is evolved until predefined stopping criteria have been met. However, the manipulation of the endosymbiont's genetic material does not only happen through horizontal gene (sequence) transfer of the host's genetic material to the former one, but also by some special types of mutation in its chromosomes, or in case of population-based learning, genomes.

Both changes in the endosymbiotic DNA are carried out by agents, the so-called transgenetic vectors. Transgenetic NeuroEvolution makes use of the following types of vectors:

- Plasmids
  - Weight plasmid
  - Structure plasmid
- Transposons
  - Jump and swap transposon
  - Erase and jump transposon

Plasmids are used for the transportation of genetic information from the host to the endosymbiont, and transposons mutate the genetic material of the endosymbiont. Figure 65 - Transgenetic NeuroEvolution provides a brief overview of how transgenetic NeuroEvolution works.

219 Abraham Ajith, Hassanien Aboul-Ella, Siarry Patrick, Engelbrecht Andries (2009): Foundations of Computational Intelligence Volume 3 Global Optimization; Berlin Heidelberg: Springer-Verlag, p. 425

220 Abraham Ajith, Hassanien Aboul-Ella, Siarry Patrick, Engelbrecht Andries (2009): Foundations of Computational Intelligence Volume 3 Global Optimization; Berlin Heidelberg: Springer-Verlag, p. 425 ff.

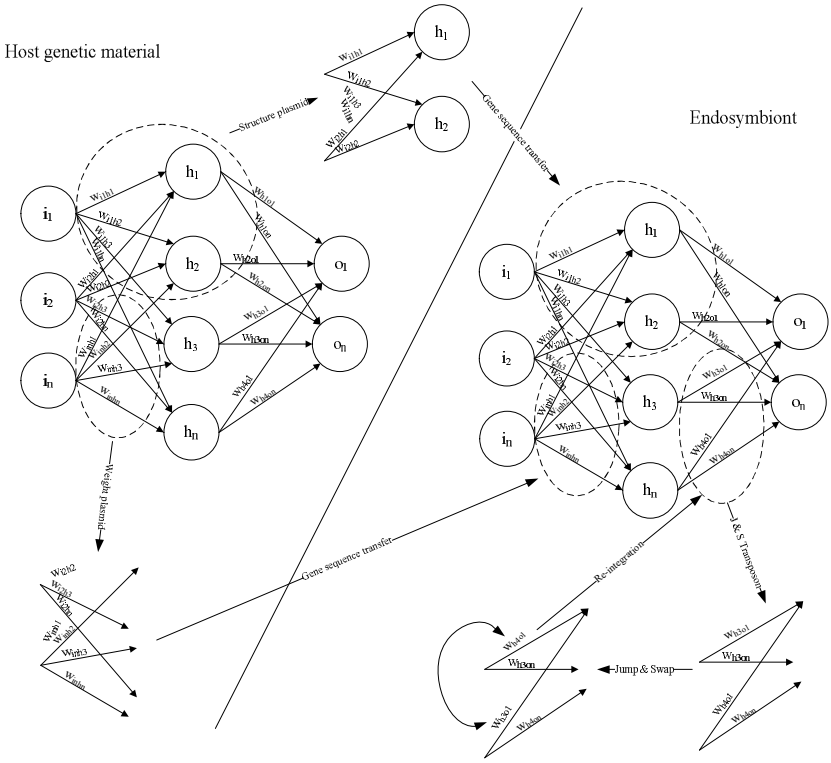

Figure 65 - Transgenetic NeuroEvolution

On the left side, Figure 65 - Transgenetic NeuroEvolution shows the host, which serves as genetic database for the horizontal gene transfer, carried out by plasmid vector agents. One weight plasmid on the left side below transports a sequence of weights to the endosymbiont, as well as a structural plasmid above the host transfers the weights, biases and activation functions. On the right side, a jump and swap transposon vector mutates the chromosome of the endosymbiont.

5.1.2 Host genetic material

The host consists of 'rough' solutions to the problem statement, that is several populations each consisting of thousands individuals each of them subject of continuous evolution. Therefore, the host is not only one organism, but thousands of organisms in different populations.

5.1.3 Endosymbiont

The endosymiont is a simulated annealing ANN solution, which simulates the cooling of metal, as explained in 3.3.3.8 Simulated annealing. The transgenetic vectors are applied during the simulated cooling process. Any iteration creates a new possible solution, and each new solution must endure the attacks of both plasmid and transposon vector agents to see if a better solution can be found.

5.1.4 Algorithm

The algorithm for applying transgenetic NeuroEvolution is as follows:

Start

1. Creation of n_{hp} initial host populations with n_c individuals.
2. ***For each $\mathbf{n_{hp}}$ repeat***
   a) Randomization of weights and threshold values of each chromosome.
   b) Repeat
      i. Calculate the network output for the value d ∈ D
      ii. Evaluate the fitness of each chromosome:
         1. Calculate the error $\delta_{d\in D}$ for each output neuron d ∈ D

$$\delta_{d\in D=}(desired_{d\in D} - actual_{d\in D})actual_{d\in D}(1 - actual_{d\in D})$$

         2. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d\in D} (w_{hidd\in D}\delta_{d\in D})$$

      iii. Selection of chromosomes to recombine
      iv. Repeat
         1. Crossover of chromosomes
         2. Mutation of offspring
      v. Until all selected chromosomes are recombined
   c) ***Until criteria are reached***
3. ***Until rough evolution ($\mathbf{r_{rmse}}$) for each $\mathbf{n_{hp}}$ has been finished.***
4. Create initial ANN solution and randomize weights.
5. ***Repeat***
   a) Calculate the network output for the value d ∈ D
   b) Evaluate the fitness of each neuron:
      i. Calculate the error $\delta_{d\in D}$ for each output neuron d ∈ D

$$\delta_{d\in D=}(desired_{d\in D} - actual_{d\in D})actual_{d\in D}(1 - actual_{d\in D})$$

      ii. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d\in D} (w_{hidd\in D}\delta_{d\in D})$$

      iii. Set $C(C) = C(S)$

*c) **Repeat***

i. Create new ANN and randomize weights according to T

ii. Calculate the error $\delta_{d\in D}$ for each output neuron $d \in D$

$$\delta_{d\in D=}(\text{desired}_{d\in D} - \text{actual}_{d\in D})\text{actual}_{d\in D}(1 - \text{actual}_{d\in D})$$

iii. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = \text{actual}_{hid}(1 - \text{actual}_{hid}) \sum_{d\in D} (w_{hidd\in D}\delta_{d\in D})$$

iv. Compare solutions according to

$$\Delta_{C(S')C(S)} = C(S') - C(C)$$

v. If $C(S')$ is better than $C(C)$, set $C(C) = C(S')$

vi. Apply plasmid vector

vii. If $C(P)$ is better than $C(C)$, set $C(C) = C(P)$

viii. Apply transposon vector

ix. If $C(T)$ is better than $C(C)$, set $C(C) = C(T)$

*d) **Until max tries for current temperature reached***

e) Decrease temperature by

$$\varphi_{C(S')C(S)} = T * e^{\frac{ln\left(\frac{S}{e}\right)}{c-1}}$$

6. Until lower temperature bound reached

End

Algorithm 14 - Transgenetic NeuroEvolution

Breakdown:

n_{hp}: the number of host populations to create
r_{rmse}: rough root mean squared error
C(C): calculation current solution
C(S): calculation initial solution
C(P): calculation plasmid solution
C(T): calculation transposon solution

The algorithm shows that the evolution of the host genetic material is carried out only roughly, until the rough root mean squared error has been reached by the evolution of the population. Within the introduced system this is the allowed, passed RMSE multiplied by 10. If the allowed RMSE is 1%, the rough RMSE is 10%.

5.1.5 Horizontal (endosymbiotic) gene (sequence) transfer

The horizontal transfer of genetic material is the transfer of genetic material from the host to the endosymbiont by plasmids. Every time the plasmid vector is applied on the currently evolving endosymbiont, the actual best solution of the currently selected host population is determined and serves as current host. The plasmid vector selects a gene sequence of the host chromosomes, copies it and transfers it to the endosymbiont. The application of the plasmid

vector happens once during each the evolutionary iteration of the endosymbiont. As mentioned above, the actual best solution (=the host) is chosen from one of the host populations, where the selection of the latter also happens at random, depending from the number of host populations created. If three host populations have initially been created, the probability for each one to be selected is 1/3 either.

After having selected the host chromosome, the relevant gene sequence has to be determined, which also happens at random. For transgenetic algorithms it is important that the host and the endosymbiont consist of the same number of genes, as for the horizontal gene transfer gene sequence determination the chromosome length is taken into consideration. After the length of the selected chromosome has been determined, a random starting and end point are created, which enclose the transfer gene sequence. The same starting and end points are then used to delete the gene sequence in the endosymiont. The gap is then filled with the host genetic transfer sequence. However, both the old and new solutions are then compared according to their quality, and if the transfer has proofed to be evolutionary reasonable, the new solution will form the basis for further evolution, but if not, the gene transfer is rolled back. As mentioned in the fundamentals, there are two types of plasmids. The application of either the one or the other happens at random with a probability of 1/2 for each.

5.1.5.1 Weight plasmid

The weight plasmid does only take the weights between the neurons into consideration. This means that a transfer sequence only consists of weights, but does not contain activation functions or biases. The following example shows the application of a weight plasmid. The length of the host chromosome is determined and a random sequence selected:

$$chr_h = [10010|0111000|011] \quad \textit{(5-1)}$$

The host chromosome has the length 15, and the gene transfer sequence starts with gene 6 and ends with gene 12. The same sequence is then selected in the endosymbiont:

$$chr_s = [10011|1111011|100] \quad \textit{(5-2)}$$

Afterwards, the endosymbiotic gene sequence is deleted and replaced by the host gene sequence, which forms the descendant chromosome:

$$chr_d = [10011|0111000|100] \quad \textit{(5-3)}$$

5.1.5.2 Structure plasmid

The structure plasmid, in contrary to the weight plasmid, contains all activation functions and bias values the transfer sequence encloses. This requires the different host populations to make use of different activation functions, as a transfer of these would be useless otherwise. The bias values evolve independently in each chromosome of each host population as well as in the endosymbiont. It should only be taken care of both the activation functions and the bias values - there is no need to interfere otherwise. The example for the structural plasmid is the same, with the exception that the gene sequence also contains bias values and activation functions:

The length of the host chromosome is determined and a random sequence selected:

$$chr_{host} = [10n_{1h}01n_{2h}0|0n_{3h}111n_{4h}000|0n_{5h}11] \quad (5\text{-}4)$$

The host chromosome has the length 15, and the gene transfer sequence starts with gene 6 and ends with gene 12. The difference is that the above chromosome also contains neuron information, namely the mentioned activation functions and the bias values of the host. The same sequence is then selected in the endosymbiont:

$$chr_s = [10n_{1s}01n_{2s}1|1n_{3s}111n_{4s}011|1n_{5s}00] \quad (5\text{-}5)$$

Afterwards, the endosymbiotic gene sequence is deleted and replaced by the host gene sequence, which forms the descendant chromosome:

$$chr_d = [10n_{1s}01n_{2s}1|0n_{3h}111n_{4h}000|1n_{5s}00] \quad (5\text{-}6)$$

5.1.6 Transposon mutation

As mentioned in the fundamentals, the transposon mutation does not transfer genetic sequences from the host, but changes the genetic information in the endosymbiont. As with plasmid vectors the application of transposonic vectors happens once within an endosymbiont's evolutionary iteration. A transposon vector selects a gene sequence in the endosymbiont and mutates it according to the transposon type. Again, both the old and new solutions are then compared according to their quality, and if the mutation has proofed to be evolutionary reasonable, the new solution will form the basis for further evolution, but if not, the gene sequence mutation is rolled back. As mentioned in the fundamentals, there are two types of transposons. The application of each type of transposon vector depends on the randomly selected gene sequence in the endosymbiotic chromosome.

5.1.6.1 Jump and swap transposon

After the length of the endosymbiotic chromosome has been determined, a random starting and end point are created, which enclose a gene sequence. If the enclosed gene sequence consists of two genes, the jump and swap transposon vector is applied and the two selected genes are swapped. The length of the symbiont chromosome is determined and a random sequence selected:

$$chr_s = [10010|01|11000011] \quad (5\text{-}7)$$

The host chromosome has the length 15, and the gene transfer sequence starts with gene 6 and ends with gene 8. The transposon can only be a jump and swap vector, when the length of the selected gene sequence does not exceed 2. These two genes are then swapped, which forms the descendant chromosome:

$$chr_d = [10010|10|11000011] \quad (5\text{-}8)$$

5.1.6.2 Erase and jump transposon

If the length of the random gene sequence exceeds two the erase and jump transposon is applied. In principle, it works similar to the jump and swap transposon, except that one randomly selected gene in the chromosome is deleted and replaced by a randomly chosen other one of the sequence. The length of the symbiont chromosome is determined and a random sequence selected:

$$chr_s = [10010|0111000|011] \quad \textit{(5-9)}$$

The host chromosome has the length 15, and the gene transfer sequence starts with gene 6 and ends with gene 2. The transposon can only be an erase and jump vector, when the length of the selected gene sequence exceeds 2. Within this gene sequence, one gene is selected at random and erased, in the example gene number 7:

$$chr_s = [10010|0\ 11000|011] \quad \textit{(5-10)}$$

Another gene is selected at random for replacing the missing one, in the following case gene number 10, which forms the descendant chromosome:

$$chr_d = [10010|0011000|011] \quad \textit{(5-11)}$$

5.1.7 Usage

Transgenetic NeuroEvolution can be applied with each ANN solution type, as long as there exist both a host and an endosymbiont. It is crucially to represent the chromosomes and gene sequences as arrays of values, so that the transfer and mutation can be carried out.

5.2 Artificial immune system-inspired NeuroEvolution

The artificial immune system (AIS) based NeuroEvolution is a combination of genetic algorithm learning and the strengths of an artificial immune system, which decreases the number of generations needed for finding a suitable solution in a high mass. This is achieved by a combination of clonal selection and somatic hypermutation, negative selection and the inclusion of the danger theory in each population's evolution cycle. Furthermore, causality plays a more important role within the introduced paradigm, as the solution population does not only change from generation to generation, but also within one generation. For the immune system-based operations only the individuals of the current generation do matter. Moreover, the in- and outputs a population in consideration uses for learning do indeed have a significance, or in other words do have a significance over time, for the genetic evolution of the single individuals (chromosomes). However, all of the immune system-based operations do not need to consider these, as only the current populations do matter. This, as a whole, allows the connection to a new, within this thesis introduced ANN paradigm – the consciousness-inspired ANN (4.3.2 A generic cortical artificial neural network).

5.2.1 Fundamentals

What an artificial immune system (AIS) exactly is can be described best when classifying them according to their application. AIS can be:

- Models of biological immune systems, as artificial neural networks are simple models of biological ones that can be used by immunologists for explanation, experimentation and prediction activities that would be difficult or impossible in wet-lab experiments, which is known as computational immunology.
- Abstractions of one or more immunological processes. Since these processes try to protect e.g. humans from biologically and biochemically hazardous entities, it has been reasoned that they may be computationally useful.[221] The latter one is of special interest for the elaboration in hand.

Furthermore, not just one solution is created during training, but populations of solutions, which firstly allows the evolution of the solutions by the already known paradigms of genetic evolution, which are

- selection,
- mating, and
- mutation.

Additionally, the immunological operators are carried out once in every generation on the current population. Thus, every single individual of every population in consideration is subject to continuous change, if it is strong enough to overcome the natural selection and survive invasions of other populations. The AIS-based operations that are carried out are

- clonal selection,
- somatic hypermutation,
- danger theory, and
- negative selection.

The latter operations also do not happen by chance, but once in every generation of individuals. However, there is one additional event that is likely to occur and which influences the size and the structure of populations, in particular a virus infection. A virus reduces the number of individuals in a population, in contrary to clonal selection. Thus, mechanisms for controlling the population size have to be implemented, which are

- elimination of individuals,
- hyperrecombination and
- birth control.

As natural selection does only occur from generation to generation and not within a generation, some of the immunological processes, like clonal selection, would increase the number of the population above the initial value, which serves as population number threshold and shall not be exceeded. According to this, the number the threshold would be exceeded by must also serve as value for the elimination operator, which then eliminates exactly this number of individuals starting from the individual possessing the worst quality.

221 Garrett Simon M. (2005): How Do We Evaluate Artificial Immune Systems?; Evolutionary Computation 13(2): 145-178

Hyperrecombination is initialized when the size of a population has rapidly decreased by a virus attack and always works together with birth control, as hyperrecombination within a population is only allowed as long the population number threshold has not been exceeded. Furthermore, it is a part of the danger theory explained below, as weak individuals which died an unnatural death by virus attack are replaced by strong ones, originating the surviving strong ones.

The population number threshold is set as the target is to continuously evolve a population towards a super-population, each single individual consisting of valuable genetic material.

5.2.2 Clonal selection and somatic hypermutation

According to Burnet's 1959 clonal selection theory,[222] the immune system repertoire undergoes a selection mechanism during the lifetime of the individual. The theory states that on binding with a suitable antigen, activation of lymphocytes occurs. Once activated, clones of the lymphocyte are produced expressing identical receptors to the original lymphocyte that encountered the antigen. Thus, a clonal expansion of the original lymphocyte occurs. This ensures that only lymphocytes specific to an activating antigen are produced in large numbers. The clonal selection theory also stated that any lymphocyte having antigen receptors specific to molecules of the organism's own body must be deleted during its development. This ensures that only antigens from a pathogen might cause a lymphocyte to clonally expand and thus elicit a destructive adaptive immune response. In this sense, the immune system can be viewed as a classifier of antigens into either self antigen or non-self antigen, with non-self antigen assumed to be from a pathogen and thus needs to be removed from the body.[223] Additionally, it has to be mentioned that the mutation carried out is described as somatic hypermutation, which states that the mutation factor is a lot higher than in usual mutation procedures.

An example algorithm for clonal selection is as follows:[224]

Start

1. Initialization: Create an initial, random population of antibodies, P_0. Iterate steps 2-7 if a predefined termination condition is not met.
2. Evaluation and Selection 1: Select a subset, F, of the fittest antibodies from P_t according to some fitness function, $f(ab_i)$.
3. Cloning:
   a) For each
   $ab \in F$, create a set of clones, C_i, such that $|C_i| = nc(ab_i)$. The set of all clones, $C = \cup_i C_i$.
4. Mutation:

222 Burnet Frank M. (1959): The Clonal Selection Theory of Acquired Immunity. Cambridge University Press

223 AISWeb (2012): Immune-Inspired Algorithms [2012-03-13]; AISWeb; URL: http://www.artificial-immune-systems.org/algorithms.shtml

224 de Castro L. N., Von Zuben F. J. (2001): An immunological approach to initialize centers of radial basis function neural networks. In Proc. of 5th Brazilian Conference on Neural Networks, 79–84

a) For each
clone $c \in C$ apply mutation function $am(f(ab_i), \rho$ Add the mutated clones, C', to P_t to give $P_t^{'}$

5. Evaluation and Selection 2: Select a subset, F' of the fittest antibodies from $P_t^{'}$
6. Diversity: Add r randomly generated new B-cells to F' give a new population $P_t^{''}$
7. Death: Retain only the best $|P_t|$ members of $P_t^{''}$ t to give P_{t+1}; all other B-cells are considered to have died.

End

Algorithm 15 – Clonal selection algorithm

Thus, the clonal selection theory explains how (B and T-)lymphocytes are activated by binding these to suitable antigens. Once this activation has happened, these special lymphocytes are cloned by a factor – the better their response to an antigen is, the higher the clone factor will be. The lower their response is, the lower is the clone factor, which ensures that the immune system only produces lymphocytes specific to an antigen are produced in a large amount. Within the proposed algorithm, clonal selection within a population happens once in each generation of individuals. In other words, clonal selection is applied before all genetic operators and operations have been applied on a population in consideration.

After each generation the quality of the best solution is determined, e.g. by calculating the unified root mean squared error or another fitness measurement for each individual. Each individual not fulfilling the minimum fitness requirement is removed from the population. The remaining population is then raised again by clones of the best-performing genome of the remaining population. Let assume, the number of individuals in a population is 5,000 and 2,450 do not fulfil the minimum quality requirements. These are then removed, so that only 2,550 individuals remain in the population. The best-performing individual is then cloned and mutated 2,450 times, which again raises the number of genomes in the population to 5,000. Therefore, the best-performing individual and its clones would then make up nearly a half of the overall population. However, as a high number of identical individuals reduce the genetic versatility of a population, each of these clones is mutated by either a weakened standard evolution mutation factor m_w so that the advantaged genetic material is preserved. For that, the algorithm simply divides the standard mutation m_s factor by two. Certainly, randomness plays a role only with a factor always the same clone would be created. The formula shows the value of a single chromosome's gene v_g after the weakened mutation:

$$v_{g(t+1)} = v_{g(t)} + (m_w - x_{rand} * m_w * 2) \qquad (5\text{-}12)$$

where $v_{g(t+1)}$ is the value of the gene after the mutation and $v_{g(t)}$ the gene's value before mutation; x_{rand} is a random number. The second half of the clones has to undergo somatic hypermutation, which means that the basic genetic material stays the same, but is mutated by a higher factor, which has to be defined separately. Hypermutation is carried out in the same way as the weakened mutation, with the only difference that the mutation factor is higher. In my most successful implementations, it is increased by its half. What has to be taken into consideration after all is the size threshold of the population, which is never exceeded due to controlled elimination of individuals. Referring to the above example and to standard genetic evolution, natural selection does only occur from generation to generation, but not within one single generation. Furthermore, as the population consists of a higher number of high-quality

individuals, compared to standard genetic evolution, the percentage to mate has also been increased from the standard 25 % to 50 % for AIS-inspired NeuroEvolution, at least until 80 % of the input datasets of a DM-problem statement have been successfully learned according to the fitness function. After that, it is decreased to 25 % for allowing the population to step gently towards the learning success. The algorithm for clonal selection within is as follows:

Start

1. Create initial ANN population and randomize weights.
2. ***Repeat***
   f) Calculate the network output for the value $d \in D$
   g) Evaluate the fitness of each chromosome:
      i. Calculate the error $\delta_{d \in D}$ for each output neuron $d \in D$

$$\delta_{d \in D=}(desired_{d \in D} - actual_{d \in D})actual_{d \in D}(1 - actual_{d \in D})$$

      ii. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d \in D} (w_{hidd \in D}\delta_{d \in D})$$

   h) Selection of chromosomes to recombine
   i) Repeat
      i. Crossover of chromosomes
      ii. Mutation of offspring
   j) Until all selected chromosomes are recombined
   k) Set $C(C) = C(S)$

– ==========***Clonal selection & hypermutation***==========

   l) Determine the best chromosome c_{best} of $C(C)$
   m) For each c of C(C)
      i. Determine quality
      ii. if current chromosome c_{cur} is below minimum fitness
         remove c_{cur} from C(C)
   n) Until cloning happened $f c_{best}$ times
   o) Repeat
      i. Clone c_{best}
      ii. if current clone c_{cur} has even number
         Weakened mutation of c_{cur}
      iii. else
         Hypermutation of c_{cur}
   p) Until initial population size has been restored
3. ***Until criteria are reached***

End

Algorithm 16 - Clonal selection and hypermutation

Breakdown:

n_{hp}: the number of host populations to create

C(C): calculation current population
C(S): calculation initial population
c_{best}: best chromosome
c_{cur}: current clone

5.2.3 Danger theory, virus attack and hyperrecombination

The danger theory[225] was initially proposed in 1994, and since then become popular amongst immunologists as an explanation for the development of peripheral tolerance (tolerance to agents outside of the host). Summing up, the danger theory states that antigen-presenting cells (APCs) are themselves activated via an alarm: a danger signal, which enables them to provide the necessary co-stimulatory signal to the T helper cells that subsequently control the adaptive immune response. The danger signals are emitted by ordinary cells of the body that have been injured due to attack by pathogen, e.g. the intra-cellular contents released due to uncontrolled (necrotic) cell death could provide such signals. Such signals are detected by specialised innate immune cells called dendritic cells that seem to have three modes of operation: immature, semi-mature and mature. In the dendritic cell's immature state it collects antigen along with safe and danger signals from its local environment such as pathogen-associated molecular patterns (PAMPs) and inflammatory cytokines. The dendritic cell is able to integrate these signals for being able to decide whether the environment is safe or dangerous. If safe, the dendritic cell becomes semi-mature and upon presenting antigen to T-cells the dendritic cell will cause T-cell tolerance. If dangerous, the dendritic cell becomes mature and causes the T-cell to become reactive on antigen-presentation.[226]

An example algorithm for danger theory is as follows: [227]

Start

1. Create an initial population of dendritic cells (*DCs*), *D*
2. Create a set to contain migrated *DCs*, *M*
3. ***For each***

$$D \in S$$

a) Do

i. Create a set of DCs randomly selected from *D*, *P*

ii. For each

$$M \in S$$

1. Do

a. Add data item to *DCs* collected list
b. Update danger, PAMP and safe signal concentrations
c. Update concentrations of output cytokines
d. Migrate the DC from *D* to *M* and create a new *DC* in *D* if concentration of co-stimulatory molecules is above a threshold

225 Matzinger P. (2002): The Danger Model: A renewed sense of self. Science, 296(5566): 301–305
226 AISWeb (2012): Immune-Inspired Algorithms [2012-03-14]; AISWeb; URL: http://www.artificial-immune-systems.org/algorithms.shtml
227 AISWeb (2012): Immune-Inspired Algorithms [2012-03-14]; AISWeb; URL: http://www.artificial-immune-systems.org/algorithms.shtml

2. End

b) End

c) For each

$$DC \in M$$

i. Do

1. Set DC to be semi-mature if output concentration of semi-mature cytokines is greater than mature cytokines, otherwise set as mature

ii. End

d) For each

$$d \in S$$

i. Do

1. Calculate number of times data item is presented by a mature *DC* and a semi-mature *DC*
2. Label data item a safe if presented by more than semi-mature *DCs* than mature *DCs*, otherwise label as dangerous
3. Add data item to labelled set *M*

ii. End

End

Algorithm 17 – Danger theory algorithm

Breakdown

S: set of data items to be labelled safe or dangerous

D: set of data items labelled safe or dangerous

Therefore, the danger theory explains how immune response to harmful cells works and why other not harmful bacteria are not attacked. The clue is that stressed cells raise some sort of alarm signal. Cells can die in two ways: via apoptotic, normal death that has been requested by the body's internal signalling system, or via necrosis, a form of unexpected death caused by something going wrong with the cell, which often causes an inflammatory response. Matzinger[228] suggested that the immune system is particularly activated by cell necrosis.[229]

The danger theory within the introduced algorithm works similar to the natural paradigm, as infected individuals are removed from the population by AIS-based operations. Infection happens via a virus, thus the virus operator. The virus operator only attacks special individuals, namely weak ones, whereas it does not bother strong individuals. The question now is what qualifies an individual as strong and what makes it weak? Again, strength is determined by a fitness function, e.g. by the calculation of the already mentioned, unified RMSE of the individual in question. The virus operator therefore affects and eliminates all individuals below a specific quality, which is the quality of the worst performing individual of the last population. Therefore, danger theory's virus operator cannot strike until the second

228 Matzinger P. (2002): The Danger Model: A renewed sense of self. Science, 296(5566): 301–305

229 Garrett Simon M. (2005): How Do We Evaluate Artificial Immune Systems?; Evolutionary Computation 13(2): 145-178

generation has been created, as the unified RMSE of the worst individual of the first generation serves as elimination threshold for the virus. All removed individuals have to be replaced, as the final size of every generation's population must be the same as the initial population's size. Thus, another population size control mechanism, called hyperrecombination, applies. Hyperrecombination only happens within a population, like clonal selection and somatic hypermutation do, and for that reason all remaining individuals of the virus-decimated population receive the privilege to recombine, in contrary to standard generation-spanning recombination, where just a specific percentage of the best-performing individuals receives it. Recombination then happens randomly and as long as birth control allows it, or in other words, the population number threshold has not been exceeded. The algorithm for the danger theory, virus attack and hyperrecombination is as follows:

Start

1. Create initial ANN population and randomize weights.
2. ***Repeat***
   a) Calculate the network output for the value $d \in D$
   b) Evaluate the fitness of each chromosome:
      i. Calculate the error $\delta_{d \in D}$ for each output neuron $d \in D$

$$\delta_{d \in D=}(desired_{d \in D} - actual_{d \in D})actual_{d \in D}(1 - actual_{d \in D})$$

      ii. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d \in D} (w_{hidd \in D}\delta_{d \in D})$$

   c) Selection of chromosomes to recombine
   d) Repeat
      i. Crossover of chromosomes
      ii. Mutation of offspring
   e) Until all selected chromosomes are recombined
   f) Set $C(C) = C(S)$

– ***=======Danger theory, virus attack and hyperrecombination==========***

   g) Store quality value of worst performing individual i_{worst}
   h) if generation > 1
      i. For each $c \in C(C)$
         1. Compare quality of $c_{(t)}$ with $c_{worst\ (t-1)}$
         2. If $quality_{c(t)}$ worse than $quality_{cworst(t-1)}$
            Eliminate $c_{(t)}$
      ii. Repeat
         1. Random crossover of all chromosomes (hyperrecombination)
         2. Standard mutation of hyperrecombined offspring
      iii. Until population number threshold has been reached again
3. ***Until criteria are reached***

End

Algorithm 18 - Danger theory, virus attack and hyperrecombination

Breakdown:

r_{rmse}: rough root mean squared error
C(C): calculation current population
C(S): calculation initial population
$c_{(t)}$: current chromosome
$quality_{c(t)}$: quality of the actual chromosome
$quality_{cworst(t-1)}$: quality of the worst chromosome of the last generation

5.2.4 Negative selection

Negative selection was introduced by Forrest[230] in 1994 and refers to the process of deleting self-reactive lymphocytes, which is termed clonal deletion and is carried out via a mechanism called negative selection that operates on lymphocytes during their maturation. For T-cells this mainly occurs in the thymus, which provides an environment rich in antigen presenting cells that present self-antigens. Immature T-cells that strongly bind these self-antigens undergo a controlled death (apoptosis). Thus, the T-cells surviving this process should be unreactive to self-antigens. The property of lymphocytes not to react to the self is called immunological tolerance.[231] The algorithm for negative selection is as follows:[232]

Start

1. Create a set of self strings, *S*, by some means.[233]
2. Create a set of randomly generated strings, R_0.
3. For each$r_0 \in R_0$, form a set, *R*, of those r_0 that do not strongly match any $s \in S$. A strong match is defined by a matching function$m(r_0, s)$, such that: $(i) m(r_0, s) \bowtie \theta$. An example of the use of $m(x, y)$ will be given below.
   a) For each
   r in *R*, ensure that no $s \in S$ matches above (or 'below', depending on the form of $\bowtie$) the threshold.
4. Return to 3. a) while change detection of *S* is required

End

Algorithm 19 – Negative selection algorithm

Breakdown:

$\bowtie$: an operator, such as $\geq$, that defines whether high or low value of $m(r_0, s)$ indicates greater similarity between the strings, and θ defines a threshold. [234]

230 Forrest S., Perelson A. S., Allen L., Cherukuri R. (1994). Self-nonself discrimination in a computer. In Proceedings of 1994 IEEE Symposium on Research in Security and Privacy, 132–143
231 AISWeb (2012): Immune-Inspired Algorithms [2012-03-13]; AISWeb; URL: http://www.artificial-immune-systems.org/algorithms.shtml
232 Forrest S., Perelson A. S., Allen L., Cherukuri R. (1994). Self-nonself discrimination in a computer. In Proceedings of 1994 IEEE Symposium on Research in Security and Privacy, 132–143
233 Forrest S., Perelson A. S., Allen L., Cherukuri R. (1994). Self-nonself discrimination in a computer. In Proceedings of 1994 IEEE Symposium on Research in Security and Privacy, 132–143

Thus, the negative selection theory states that lymphocytes being dangerous or self-reactive are eliminated instead of being allowed to maturate. Self-reaction tests may be carried out with every new child solution originating two parent solutions in standard generation-spanning recombination. Every new individual has to show its reaction towards a negative selection-operator, which then marks a solution as deletion candidate for the elimination operator or leaves it untouched. The negative selection operator checks whether a newborn individual's quality (according to 5.3.3 Generic determination of artificial neural network quality) is below a dynamically calculated threshold, namely the quality value of the worst genome of the last generation error $quality_{cworst_{(t-1)}}$, which also finds application within the danger theory. Solutions with a quality below this threshold would indeed not survive very long, but if selected for deletion before further evolution is applied, place is given for new solutions. Mating continues as long as it needs to reach the population number threshold, equally how many individuals are negatively selected. Let assume, the population number threshold is 10, and the evolution from the initial population at point in time t to the population at point in time $t+1$ produces 3 solutions with a quality value below $quality_{cworst_{(t-1)}}$, then the generation at $t+1$ would only consist of 7 individuals. Thus, the quality of new-born individuals is verified after each recombination of two parent chromosomes and in case the selection criteria are met negative selection is applied. However, evolution then continues until the threshold of 10 is reached again and not until then the new generation is ready for further evolution. As the value $quality_{cworst_{(t-1)}}$ is needed, negative selection cannot be applied until the second generation.

At a first glance, one might say that there is no need for applying negative selection, as danger theory also eliminates low-quality individuals. Nevertheless, if very bad performing solutions are removed from the population as soon as possible, as it happens within negative selection, then the overall quality of a population is increased in advance. The algorithm for the negative selection is as follows:

Start

1. Create initial ANN population and randomize weights.
2. ***Repeat***
   a) Calculate the network output for the value d ∈ D
   b) Evaluate the fitness of each chromosome:
      i. Calculate the error $\delta_{d\in D}$ for each output neuron d ∈ D

$$\delta_{d\in D=}(desired_{d\in D} - actual_{d\in D})actual_{d\in D}(1 - actual_{d\in D})$$

      ii. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d\in D} (w_{hidd\in D}\delta_{d\in D})$$

   c) Selection of chromosomes to recombine
   d) Repeat
      i. Crossover of chromosomes

234 AISWeb (2012): Immune-Inspired Algorithms [2012-03-13]; AISWeb; URL: http://www.artificial-immune-systems.org/algorithms.shtml

ii. Mutation of offspring
– ***==========Negative selection==========***
iii. Check quality of offspring
iv. If quality of current offspring < r_{rmse}
1. Eliminate current offspring
2. Repeat
– Recombine selected chromosomes anew
3. Until quality of current offspring < $quality_{cworst(t-1)}$
e) Until all selected chromosomes are recombined
f) Set $C(C) = C(S)$
3. Until criteria are reached

End

Algorithm 20 - Negative selection

Breakdown:

n_{hp}: the number of host populations to create
C(C): calculation current population
C(S): calculation initial population
C(P): clone population

5.2.5 Overall algorithm

Thus, the overall algorithm as a combination of all mentioned paradigms and methods is as follows:

Start

1. Create initial ANN population and randomize weights.
2. Repeat
a) Calculate the network output for the value $d \in D$
b) Evaluate the fitness of each chromosome:
i. Calculate the error $\delta_{d \in D}$ for each output neuron $d \in D$

$$\delta_{d \in D=}(desired_{d \in D} - actual_{d \in D})actual_{d \in D}(1 - actual_{d \in D})$$

ii. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d \in D} (w_{hidd \in D}\delta_{d \in D})$$

c) Selection of chromosomes to recombine
d) Repeat
i. Crossover of chromosomes
ii. Mutation of offspring
– ***==========Negative selection==========***
iii. Check quality of offspring

iv. If quality of current offspring < 0.1

1. Eliminate current offspring
2. ***Repeat***
   - Recombine selected chromosomes anew
3. ***Until quality of current offspring < quality$_{cworst(t-1)}$***

e) Until all selected chromosomes are recombined

f) Set $C(C) = C(S)$

– ***==========Clonal selection & hypermutation==========***

a) Determine the best chromosome c_{best} of $C(C)$

b) For each c of C(C)

iii. Determine quality

iv. if current chromosome c_{cur} is below minimum fitness

– remove c_{cur} from C(C)

c) Until cloning happened $f c_{best}$ times

d) Repeat

v. Clone c_{best}

vi. if current clone c_{cur} has even number

– Weakened mutation of c_{cur}

vii. else

– Hypermutation of c_{cur}

e) Until initial population size has been restored

– ***=======Danger theory, virus attack and hyperrecombination==========***

i) Store quality value of worst performing individual i_{worst}

j) if generation > 1

i. For each $c \in C(C)$

1. Compare quality of $c_{(t)}$ with $c_{worst\ (t-1)}$
2. If $quality_{c(t)}$ worse than $quality_{cworst(t-1)}$
   - Eliminate $c_{(t)}$

ii. Repeat

3. Random crossover of all chromosomes (hyperrecombination)
4. Mutation of hyperrecombined offspring

iii. Until population number threshold has been reached again

3. Until criteria are reached

End

Algorithm 21 - Immune system-inspired NeuroEvolution

5.2.6 Causality

More than in other evolutionary approaches, causality plays a major role when applying immune system-inspired NeuroEvolution, as not only standard and generation-spanning evolution, but also evolution of the current population is applied. For such evolution, nothing but the individuals and the genetic operators and immune system-based operations do matter. It is not important, what the inputs are or what the underlying individuals shall learn. This is the passage to Katz and his theory of reduced functionalism. The key premise of reduced functionalism, as explained within 4.3.2 A generic cortical artificial neural network, is that only causal currents matter in determining conscious contents. Thus, only current connections

of neurons, and within the introduced approach also current populations, do matter for several of the operations carried out. It is not important, what the inputs for and outputs of each single chromosome are. Only their existence and the ability to manipulate them matters.

5.2.7 Usage

AIS-inspired NeuroEvolution may be applied with any feed-forward or recurrent ANN. As it is a refined genetic learning approach, the learning performance strongly depends from the size of the genetic pool (a smaller pool will require more generations for building out the desired solution while a larger genetic pool will perform better in terms of generations, but worse in terms of computation time).

5.3 Structural evolution

The combination of artificial neural networks to committee machines has already been described; however, by applying a committee machine the detection of one of the solutions fulfilling the requirements of the fitness function (more than one solutions may be a fit) is highly probable, but unfortunately the detection of the optimal one is not. The two approaches applying adaptation of an ANN's architecture within the evolution process discussed within ths elaboration are structural[235] evolution and NeuroEvolution of augmenting topologies. NEAT is restricted to learn by genetic algorithms, whereas the structural evolution may be and has been applied for any given training approach for feed-forward and recurrent artificial neural networks that have been discussed.

5.3.1 Fundamentals

As with all learning algorithms discussed here the number of input neurons is determined according to the number of input data sets and thus applies the formula described in 4.3.1.2 Number of neurons for determining the number of hidden neurons. This resulting artificial neural network serves as model for the creation of an initial population. For the following algorithm this means that the number of input and output neurons are determined according to the presented training data. n_{max} can be determined automatically, based on the number of input neurons. m_{max} will never exceed 2, as with 2 hidden layers ANNs are capable of representing an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy.[236] The algorithm shows that a number of committees n_c is created, each of these having $n + 1$ neurons in the actual hidden layer, compared to the last committee. When the algorithm has reached n_{max} for the current hidden layer m_{cur}, the second hidden layer, which simultaneously represents m_{max}, is created. For m_{max}, the same constructive procedure for creating the neurons is adducted, with the exception that for calculating n_{max} the hidden layer $m_{max} - 1$ is drawn on. During the verification phase, an array is created for each committee, which is increased by 1 when r_{max}

235 Neukart Florian et al. (2011): Problem-dependent, genetically evolving Data Mining solutions; Proceedings of DAAAM 2011, 2011 22nd World Symposium, p. 1 ff.

236 Heaton Research (2005 - 2011) The number of Hidden Layers [2011-28-09]; URL: http:// www.heatonresearch.com/node/707

has been exceeded for the actual data set. The best solution is the one with the lowest array value.

Individuals of this initial population are then selected according to their fitness, and e.g. by the application of a genetic algorithm recombined and mutated, which as a whole is the evolution of a single generation of individuals. As fitness function serves the RMSE, and selection according to fitness in structural evolution means that the 25% fittest individuals of a generation receive the privilege to recombine. Mutation is carried out on 10% of the offspring. However, it has been proved that in combination with SRANNs the selection of the 50% fittest individuals, combined with an offspring mutation rate of 25% the best-converging ANNs for classification and time-series prediction can be achieved. The whole evolution process therefore can partially be regarded as a combination of incremental and selective pruning not only on neuron-layer, but also on hidden layer-layer. Both pruning methods are trial and error approaches to finding an appropriate number of hidden neurons for an ANN's hidden layer.[237] The incremental pruning algorithm usually increases the number of neurons up from an ANN with just one hidden neuron until a specified number of hidden neurons have been reached. Selective pruning removes neurons from an ANN until either no more neurons can be removed or the error rate increases. In contrast to the mentioned ones, structural evolution starts with an assumed optimal solution and both increases and decreases an ANN's neurons and layers until n_{max} or n_{min} and m_{max} have been reached.

In case of applying structural evolution with NeuroEvolution, recombination and mutation in detail happens as explained in 3.3.3.7 Genetic learning, by the creation of cut-points between the created double-arrays consisting of a network's weights. structural evolution makes use of two cut-points, whereas the first one is chosen randomly, but by taking into consideration that the first cut-point c_1 cannot be chosen so far in the array that there is not a sufficiently long section left to allow the full cut length to be taken, as proposed by Heaton Research.[238] The second cut-point is located at the position $c_2 = 2 * c_1$.

5.3.2 Algorithm

When applying structural evolution for a committee of FFANNs, learning by a genetic algorithm, a software system requires any presented problem to run through the following algorithm:

Start

1. Creation of initial ANN population of n_p ANNs with m_{min} hidden layers and n_{min} hidden neurons for each hidden layer.
2. ***Repeat***
    a) Calculate the network output for the value $d \in D$
    b) Evaluate the fitness of each chromosome:
        i. Calculate the error $\delta_{d \in D}$ for each output neuron o_n

237 Heaton Jeff (2008): Introduction to Neural Networks for Java, 2nd ed.; Chesterfield: Heaton Research, Inc., p. 213 ff.

238 Heaton Jeff (2008): Introduction to Neural Networks for Java, 2nd ed.; Chesterfield: Heaton Research, Inc., p. 179

$$\delta_{d\in D=}(desired_{d\in D} - actual_{d\in D})actual_{d\in D}(1 - actual_{d\in D})$$

ii. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d\in D} (w_{hidd\in D}\delta_{d\in D})$$

c) Selection of chromosomes to recombine

d) Repeat

i. Crossover of chromosomes

ii. Mutation of offspring

e) Until all selected chromosomes have been recombined

3. *Until criteria are reached*

4. Determine the quality value of the initial network architecture

5. Store the quality value of the initial network architecture in the quality array

6. Repeat

a) if n < nmax:

Increase *n* by c_n

b) Creation of population of *x* ANNs with m_{cm} hidden layers and n_{cn} hidden neurons for each hidden layer.

c) Randomization of weights and threshold values of each chromosome.

d) Repeat

i. Calculate the network output for the value d ∈ D

ii. Evaluate the fitness of each chromosome:

12. Calculate the error $\delta_{\text{d}\in\text{D}}$ for each output neuron o_n

$$\delta_{d\in D=}(desired_{d\in D} - actual_{d\in D})actual_{d\in D}(1 - actual_{d\in D})$$

13. Calculate the error δ_{hid} for each hidden neuron hid

$$\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d\in D} (w_{hidd\in D}\delta_{d\in D})$$

iii. Selection of chromosomes to recombine

iv. Repeat

1. Crossover of chromosomes
2. Mutation of offspring

v. Until all selected chromosomes have been recombined

e) Until criteria are reached

f) Determine the quality value of the current network architecture

g) Store the quality value of the current network architecture in the quality array

h) Verification

i. Present each verification data set

ii. Increase error array for RMSE > r_{max} for each verification data set by 1

i) if n == n_{max}:

j) Increase *m* by c_m

7. Until m_{max} is reached

8. Comparison of stored ANN solutions by quality

9. Present best solution

End

Algorithm 22 - Structural evolution

Breakdown:

n_p: Number of chromosomes
m_{min} /m_{max}: Minimum/maximum number of hidden layers
n_{min} /n_{max}: Minimum/maximum number of neurons per m_n
c_n/c_m: Hidden neuron/layer counters
$d \in D$: Actual input
$\delta_{d \in D}$: Error for input $d \in D$ at output neuron o_n in the current generation
δ_{hid}: Error for hidden neuron
r_{max}: Allowed RMSE
n_{min} and n_{max} are determined on the basis of the number of input neurons. The equation

$$n_h = \left(\frac{n_{input}}{3} * 2\right) + n_{output} \tag{5-13}$$

delivers the number of the hidden neurons, which form the basis for the determination of the minimum and maximum number to which evolution shall come to:

$$n_{min} = n_h - \left(\frac{n_h}{3}\right) \tag{5-14}$$

and

$$n_{max} = n_h + \left(\frac{n_h}{3}\right) \tag{5-15}$$

The same holds for the second hidden layer, if evolution is applied that far, with the difference that all neuron numbers of the evolution of the first layer are stored into an array, which serve as a calculation basis for n_{min} and n_{max}. If, e.g., one solution of the first hidden layer evolution process has three hidden neurons, the evolution of the second hidden layer takes this as n_h and calculates n_{min} and n_{max} anew. The algorithm also shows that the first run to determine the initial ANN's quality has to be carried out at the beginning, as the evolution starts immediately afterwards.

5.3.3 Generic determination of artificial neural network quality

The quality of an ANN in the applied algorithms is, on the one hand, determined by the RMSE. The problem is that the accumulation of all output neurons' RMSE is not enough for being able to determine the best network of an evolutionary process. This can be explained by the following algorithm:

Start

1. Initialize/reset overall quality values q_1,...,q_{10} and $q_{overall}$ with 0
2. ***Repeat***
    a) ***Repeat***
        i. Determination of the current solution's quality (RMSE) and storage of the quality value (the error of each neuron for each dataset) into overall quality value for each input dataset at:

$$q_{cur} = \begin{cases} q_1 + x_i^2 & , \quad if\ x_i^2 \not\in (0.01, 0.1) \\ q_2 + x_i^2 & , \quad if\ x_i^2 \not\in (0.001, 0.01) \\ \dots & \dots \quad \dots \end{cases}$$

b) Until all datasets have been presented

i. Calculate the overall quality of the current architecture by taking the RMSE

$$q_{overall} = \left(\sqrt[2]{\frac{1}{n} \sum_{i=1}^{n} x_i^2} * \left(\sum_{i=1}^{n_o} o_i * m \right) + \sqrt[2]{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) * k$$

ii. Store $q_{overall}$ in the overall quality array with the current architecture value.

3. Until evolution has finished

4. Present the best solution

End

Algorithm 23 – Quality determination

Breakdown:

$q_{1,\dots,}q_{10:}$ the quality values holding the quality results of the current architecture from 0.1 to 0.000000001
i_n: the currently presented input data set
q_{cur}: the current architecture's quality
q_{com}: the common denominator calculation
$q_{overall}$: the overall quality value of the current architecture
m: predefined multipliers, applied on the error of each single neuron depending from its decimal place
k: frequency parameter

This can be explained better by the comparison of two networks A and B, both having one output neuron and been presented four input data sets (Table 1 - Quality determination).

Table 1 - Quality determination

Input Dataset / Output Neuron	RMSE ANN A	RMSE ANN B
1	0.01	0.1
2	0.01	0.1
3	0.01	0.1
4	0.3	0.05

It is clearly obvious that A comes to better results for the input data sets 1, 2 and 3. However, the fourth output shows an error of 30%, which is pretty high. When trying to sum up or calculate a mean value of all the RMSEs of both networks, A will have the better result than

B, although B as a whole performs better than A. The solution therefore is to apply the introduced algorithm.

For ANN A, the overall quality is

$$q_{overall} = (r * (0.01 * 9 + 0.01 * 9 + 0.01 * 9 + 0.3 * 10) + r) * \frac{3}{9} * \frac{1}{10} = 4.697 \quad (5\text{-}16)$$

For ANN B, the overall quality is

$$q_{overall} = (r * (0.1 * 10 + 0.1 * 10 + 0.1 * 10 + 0.05 * 9) + r) * \frac{3}{10} * \frac{1}{9} = 4.01625 \quad (5\text{-}17)$$

where r is the summed RMSE for all output neurons. The above equations both contain the multipliers 10 and 9, which are selected by considering the decimal place of each neuron's error. Errors greater than 0.1 are multiplied by 10, errors between 0.01 and 0.1 by 9, ..., errors between 1E-9 and 1E-10 by 1, which leads to an application of the quality determination down to errors of 0.000000001 or 0.0000001%. A further multiplication is carried out by the frequency parameter k, which simply is a division of the count of decimals with a specific decimal place by the related multiplier.

5.3.4 Parameterization

As structural evolution can be applied not only with one learning method, the parameterization depends on the chosen approach, e.g. a genetic algorithm requires amongst others an initial population size, and back propagation amongst others a momentum.

5.3.5 Usage

Structural evolution can be applied with every feed-forward or recurrent ANN, although for the application with recurrent ANNs like the SRANN, the context layers have to be taken into consideration. Furthermore, when applying structural evolution it has to be considered that the stopping criteria for evolving an ANN must not be the same error x for a number of y generations. Especially when used with SRANNs, the stopping criteria must be a fixed number of generations up to a maximum of 10,000 or an allowed minimum error. This is because SRANNs (and sometimes other genetic solutions) tend to perform extraordinary well (especially on problem statements of lower complexity), which makes a determination of the network quality impossible, as the quality of all architectures throughout the evolution tends to become zero.

5.4 Summary

The introduced approaches show is that it is not necessarily required to apply the solutions Nature conceived for solving exactly the same problems. It is possible to translate parts from solutions like an immune system into machine learning approaches, e.g. resulting in well-performing training algorithms for artificial neural networks. When pursuing the target of

implementing an artificial mind it is important to understand the inner workings of the brain, but it is not necessarily required to implement solutions in the same way – only to achieve the target counts.

Transgenetic NeuroEvoltion perfectly shows that strategies for solving optimization problems Nature has developed over millions of years of evolution can successfully be translated into machine learning. Transgenetic NeuroEvolution solution outperformed most of the classical solution types discussed within this elaboration by relying on a strategy combining a host organism with an endosymbiont. By the application of transgenetic NeuroEvolution, we can eliminate several, well-known problems in ANN learning, like the local minima or oscillation problems of back propagation. Horizontal gene transfer and transposon mutation of the endosymbiont do, due to architectural constraints like the constant changing of the processed solution's structure (or gene sequence), simply not allow the upcoming of such problems.

The application of artificial immune system-based operations is a completely new type of artificial neural network evolution and processing as well. Not only generation-spanning recombination and mutation do matter, but also the elimination of weak chromosomes in advance, and the higher requirements for surviving covered by the danger theory. Thus, also genetic operations only of relevance for the actual generation are carried out. All the combined paradigms contribute in a high mass to the improvement of the overall performance, allowing a software system to find an optimal solution in fewer generations. As a matter of course, all of the AIS-based operations increase the learning performance and may be applied solely, depending from the requirements on the final solution. As last and most important point, AISGA outperforms standard GA-solutions in terms of learning performance and classification success on new datasets.

6 Autonomously acting cars and predicting market behaviour: some application scenarios for ANNs

As already discussed, deep artificial neural networks are, in some aspects, the most human-like artificially developed way of processing information we have thitherto. Very impressive and important achievements have been made due to complex network structures and sophisticated training algorithms, and what can already be achieved is not so far from the capabilities of what parts of the human brain, such as the visual cortex, can achieve. However until recently, the typical research in that area concerned problems that are perfectly suitable for being processed on a computer, like prediction of numerical values or clustering of unstructured data, and individual approaches are simply not potent enough for approaching the full stack of a human brain's capabilities. But a combination of several techniques may… With actual training algorithms it would be impossible to train a structure consisting of millions of neurons and thousands of neuron layers. A major step to forward have been deep learning architectures as well as sophisticated learning algorithms (see chapters 4 and 5), but as we will see this is not the answer we are looking for. My educated guess is that quantum ANNs will open a whole new world to creating artificial brains (see 10.2 Quantum linear superposition in artificial brains). This chapter serves the purpose to understand what ANNs are often used for, and we start with simple data analysis and evolve towards more complex information processing in the latter chapters.

6.1 Analysis and knowledge

As indicated before, data mining pursues the target of extracting knowledge (for a detailed description of how data can be represented in order to efficiently support knowledge discovery see 10.6.1.1 Knowledge representation) from data.[239] In the context of data mining (DM), knowledge are interesting patterns and amongst these, patterns that are generally accepted, not trivial, new, useful and interpretable.[240] Since the introduction of data mining, numerous definitions arose. The most precise and within the field accepted ones are:

- DM is the analysis of observational data sets to find unsuspected relationships between these and to summarize the data in novel ways that are both understandable and useful,241 but also
- the process of discovering knowledge from large amounts of unstructured data stored in databases, data warehouses (a DWH forms the factual basis on which decision situations are presented and assessed[242]), or other information repositories,[243] and

239 Berthold Ed (1999): Intelligent Data Analysis: An Introduction; New York: Springer-Verlag, cited from: Runkler Thomas A. (2010): Data Mining - Methoden und Algorithmen intelligenter Datenanalyse; Wiesbaden: Vieweg+Teubner | GWV Fachverlage GmbH, p. 2

240 Fayyad et al. (1996) Advances in Knowledge Discovery and Data Mining, cited from: R Runkler Thomas A. (2010): Data Mining - Methoden und Algorithmen intelligenter Datenanalyse; Wiesbaden: Vieweg+Teubner | GWV Fachverlage GmbH, p. 2

241 Hand David J., Mannila Heikki, Smyth Padhraic (2001): Principles of Data Mining; Cambridge: MIT Press

242 Ruan Da, Hardeman Frank, van der Meer Klaas (2008): Intelligent Decision and Policy Making Support Systems (Studies in Computational Intelligence); Berlin Heidelberg: Springer-Verlag, p. 66

F. Neukart, *Reverse Engineering the Mind*, AutoUni – Schriftenreihe 94, DOI 10.1007/978-3-658-16176-7_6

– the nontrivial extraction of implicit, previously unknown, interesting and potentially useful information and from data.[244]

According to these definitions, DM is usually applied on huge amounts of data containing knowledge not yet discovered, or to recognize and identify patterns. Some examples are:

– An autonomously driving car embedded in a multi-agent system relying on a huge amount of sensors and algorithms in order to correctly perceive what it happening around it and make sense of the huge amount of unstructured data the external world delivers.
– An assurance company, having saved lots of personal data about their customers, ranging from income to age and marital status, trying to find out which of the saved attributes might influence a customer to switch to another assurance company is one illustrative example from business side.
– A meteorological institution trying to predict weather on data, stored over a period of several years, is an example from the world of science.

A countless number of algorithms and functions which an intelligent software system may apply on raw data, and also numerous cases in which DM finds application, but the target is always the same: gaining interesting, accurate, interpretable knowledge in an intelligent and efficient way.

Our everyday world as we perceive it is unstructured and raw, meaning that most of the information around us is irrelevant for our current actions and must be filtered. Only what is relevant is extracted and processed further, i.e. our visual system detects and interprets information from visible light to build a representation of the surrounding environment, and carries out a number of complex tasks:

– reception of light and the formation of monocular representations
– buildup of a nuclear binocular perception from a pair of two dimensional projections
– identification and categorization of visual objects
– assessing distances to and between objects
– guiding body movements in relation to the objects seen

Non-image forming visual functions, independent of visual perception, includes the pupillary light reflex (PLR) and circadian photo entrainment. Data mining algorithms help software systems to do exactly that. Data mining functions are classified into three umbrella dimensions and their counterparts, which are

– supervised and unsupervised
– descriptive and predictive
– transparent and opaque[245]

243 Han Jiawei, Kamber Micheline, Pei Jian (2000): Data Mining: Concepts and Techniques; Morgan Kaufmann Publishers

244 Frawlwy William J. et al. (1992): Knowledge Discovery in Databases - An Overview; AI Magazine: 213-228

245 Hornik Mark F., Marcade Erik, Venkayala Sunil (2007): Java Data Mining: Strategy, Standard, and Practice. A Practical Guide for Architecture, Design, and Implementation (Morgan Kaufmann Series in Data Management Systems); San Francisco: Elsevier, Inc., p. 86

6.1.1 Supervised and unsupervised functions

Supervised functions are all the mining functions requiring training data before being able to perform on new data. Therefore, supervised means that an artificial neural network of this category cannot be applied on new data before having been trained with existing data.

6.1.2 Classification

Classification describes a form of prediction, where one or many input values are processed by the mining function or algorithm to receive one or many output values. In other words, classification is the projection of an assignment of elements in pre-defined classes,[246] or the classification of feature vectors in a finitely number of classes.[247] Classification (and clustering) tasks are very common in the field of biology for determining groups of data that show similar behavior and for classifying new samples into them. Classification is more often used as a second stage after the clustering.[248]

Very often, for classification no specific algorithm can be derived easily, so software systems capable of processing numerous inputs for deriving the desired solution decrease the time and complexity needed to come to a solution. The notion of classification is to classify cases according to a fixed set of categories, which itself simply is a discrete value with well-defined meaning.[249] An ANN used for classification, like a perceptron (3.3.2.6 Perceptron), derives a functional relationship between the input data and the desired output. Training data therefore must contain data sets as well as the desired outputs to each data set, so the ANN can learn (3.3.3 Training and learning) how to derive target values from future datasets.

The results of software side classification are of a specific quality, determined by the comparison of the actual output values and the desired output values of some training data. It is therefore common not to train an ANN with all the available training data, but with a specific amount so that the trained ANN can be evaluated by the feeding of known data and the following comparison of its actual output values with the given values. Classification is supported by manifold statistical and machine learning approaches, such as logistic regression, naïve Bayes, feed forward ANNs, decision trees, support vector machines and a lot more.

246 Chamoni Peter, Gluchowski Peter (2006): Analytische Informationssysteme: Business Intelligence- Technologien und –Anwendungen, 3rd. ed.; Berlin: Springer-Verlag, p. 265

247 Ertel Wolfgang (2008): Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung; Wiesbaden: Friedr. Vieweg & Sohn Verlag | GWV Fachverlage GmbH, p. 181

248 Nedjah Nadia et al. (2009): Intelligent Text Categorization and Clustering; Berlin Heidelberg: Springer-Verlag, p. 2

249 Hornik Mark F., Marcade Erik, Venkayala Sunil (2007): Java Data Mining: Strategy, Standard, and Practice. A Practical Guide for Architecture, Design, and Implementation (Morgan Kaufmann Series in Data Management Systems); San Francisco: Elsevier, Inc., p. 88

6.1.3 Regression

Regression is used to make predictions on continuous numerical targets and, like classification, a supervised mining function.[250] Regression therefore tries to explain a dependent, continuous variable by several independent variables.[251] As it is supervised, the same prerequisites apply to regression as to classification. Training data is needed for the algorithm or function for being able to derive a functional relationship between the input values and the output values. However, in difference to classification, a specific form of ANNs has proofed to be of value, namely recurrent ANNs (3.3.2.8 Recurrent artificial neural network). This type of ANNs has short-term memory, which is very useful when having to predict continuous numerical values.

The results of software side regression are of a specific quality, which is determined by the comparison of the actual output values and the desired output values of some training data, as it is done within classification. Regression is, amongst many others, supported by (recurrent) ANNs, decision trees, support vector machines, (linear, logistic, polynomial, stepwise etc.) regression.

6.1.4 Clustering

Algorithms and functions capable of clustering belong to the form of unsupervised mining approaches. Clustering is used, when needing to identify natural clusters, to find representtative cases within a huge amount of data for supporting data reduction, or when needing to identify data not belonging to any of the found clusters.[252] Within huge amounts of data, clustering therefore tries to identify data sets similar to each other and assigns these to a cluster. Different data sets belong to different clusters, and a set of clusters is considered to be of high quality if the similarity between clusters is low, yet the similarity of cases within a cluster is high.[253] The groups are, in contrary to classification, not known a priori, but the outcome of the clustering proceeding.[254]

Within the field of artificial neural networks, the self-organizing feature map of Teuvo Kohonen (3.3.2.9.4 Self-organizing feature map) is applied for clustering.

---

250 Hornik Mark F., Marcade Erik, Venkayala Sunil (2007): Java Data Mining: Strategy, Standard, and Practice. A Practical Guide for Architecture, Design, and Implementation (Morgan Kaufmann Series in Data Management Systems); San Francisco: Elsevier, Inc., p. 89

251 Fahrmeir L. et al. (1996): Regressionsanalyse; In: Fahrmeir, L. et al. (Hrsg.): Multivariate statistiche Verfahren. 2nd ed.; Berlin, New York, S. 93 - 168

252 Dipartimento di Elettronica e Informazione, Politecnico Di Milano: Clustering [2011-18-09]; Dipartimento di Elettronica e Informazione; URL: http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html

253 Anderberg Michael R. (1973): Cluster Analysis for Applications, New York: Academic Press Inc.

254 Chamoni Peter, Gluchowski Peter (2006): Analytische Informationssysteme: Business Intelligence- Technologien und –Anwendungen, 3rd. ed.; Berlin: Springer-Verlag, p. 265

Clustering is, amongst many others, supported by self-organizing feature maps, centroid-based algorithms,[255] distribution-based algorithms, density-based algorithms, orthogonal partitioning clustering.[256]

6.1.5 Attribute importance

Very often, it is necessary to determine which attributes influenced the emergence of a data mining processes' result most, which makes it possible to focus on these attributes. Through attribute importance, the attributes not contributing to the outcome of a mining process in a high mass also can be identified and, if desired, be reduced. Especially when dealing with data sets containing numerous attributes, the identification of the attributes contributing to a result becomes necessary knowledge, as the data mining process can be improved in terms of time and complexity through the reduction of attributes. Also, attributes not relevant for finding a solution may produce noise, which may falsify a DM result.

When dealing with ANNs, attributes with higher numeric values influence a solution more than attributes with lower numeric values. Therefore, no other algorithm then sorting the attributes according to their value is necessary. As attributes in different data sets may have either high or low values, it is useful to determine the probability of an attribute being high as well.

6.1.6 Association

With association a special relationship between attributes is determined, namely support and confidence. The support of a rule indicates how frequently the items associated in the rule occur together, whereas the confidence of a rule indicates the probability of finding both the antecedent item set and the consequent item set in the same transaction, given that the antecedent alone is found.[257] The most simple implementation of an association rule therefore is A implies B, determined through the number of occurrences of A, followed by B. As association rules are used for determining the relationship between attributes, they can also be used for determining the relationships between categories.

6.1.7 Interesting knowledge

The knowledge, resulting from a DM-process, carried out by interaction of humans and a DM system, must be interesting. Interesting in this context means that especially software systems applying computational intelligence-paradigms very often are capable of detecting patterns in

255 MacQueen J. B. (1967): Some Methods for Classification and Analysis of Multivariate Observations; Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability; Berkeley: University of California Press, 1:281-297

256 Milenova Briana et. al. (2002): O-Cluster: Scalable Clustering of Large High Dimensional Datasets [2011-18-09]; Oracle Corporation; URL: http://www.oracle.com/technology/products/bi/odm/pdf/o_cluster_algorithm.pdf

257 Hornik Mark F., Marcade Erik, Venkayala Sunil (2007): Java Data Mining: Strategy, Standard, and Practice. A Practical Guide for Architecture, Design, and Implementation (Morgan Kaufmann Series in Data Management Systems); San Francisco: Elsevier, Inc., p. 93

unstructured raw data, but not everything might be of interest. A DM-system relying on CI therefore must be able to correctly interpret (learn) and process what its user wants to know, exactly how a human infant has to learn what to do in special situations for being able to apply the same or similar behavior in similar future situations.

6.1.8 Accurate knowledge

The results a data mining system delivers must be accurate. It is not in the interest of the knowledge receivers that information is interesting, but wrong. A software system that predicts a 98% probability for a loss of € 5,000,000 in the next six months because customers will not continue to buy the products x and y would, without doubt, be interesting for a company. If, according to this newly gained knowledge, the production of these products would be stopped although customers are still demanding for it, and in consequence buy products of competitors, a wrong prediction would lead to a financial disaster.

6.1.9 Interpretable knowledge

The knowledge must be interpretable for humans. Not only evolutionary approaches in data mining require either a software system or the user to transform textual information into machine-readable information, meaning digits. Additionally, the often required normalization of data makes the information useless for a human. Therefore, not only the input must be transformed, but also the output transformed back into a human-interpretable format. Graphical user interfaces providing diagrams and charts very often help to interpret the resulting information.

6.1.10 Intelligent processing

Data mining very quickly becomes very complex, which can also come true when applying simple paradigms. An example is the classification of data with artificial neural networks (3.3.1 Artificial neural networks), which is not very complex in a situation when there are 100 data sets, each containing only three attributes, and the classification happens by a ANN having to predict the membership to classifier a or b. The classification becomes more complex when one data set contains 1,000 attributes, and the results must be predicted by not only one, but a committee of numerous ANNs of different type and size, making use of different activation functions, into 100 classifiers. From such a situation can be derived that intelligent processing is the probable, correct combination of different techniques as well as the efficient application of the same, which leads to efficient processing.

6.1.11 Efficient processing

The execution of CI-methods is very often time-consuming and requires lots of processing power. It is therefore preferable to develop such systems in a way that not only makes them capable of multithreading, but also allows them to apply the suitable set of techniques for a specific problem, i.e. split the problem into sub-problems and tackle each of the sub-problems with the most suitable approach. The result, or the representation of the gained knowledge resulting from a data mining process, is usually described by models or patterns.

A model is a global summary of a dataset and may, e.g. be expressed as an equation that describes a process.[258] An example would be the representation of a matrix in a p-dimensional space, so the equation

$$y = a + x * b \tag{6-1}$$

where a and b represent parameters, can be declared as a model.[259] On the contrary, a pattern describes a structure relating to a small part of the overall data, exemplified by the notation

$$if\ x > x_1\ then\ P(y \in Y_1) = p_1 \tag{6-2}$$

where x_1 and Y_1 denote a value and a set of values.[260]

The modelling phase is often considered as the most interesting phase of the data mining process, as the resulting model or pattern describes the solution for the problem statement. As many possible techniques or algorithms for a problem statement exist, DM often requires the user to understand and to know what to do next. The difficulty increases, when no algorithmic solution for a problem can be found. Very often, the user of a DM system may not be interested in the algorithmic solution of a problem statement, but in the result that can be achieved by applying this solution. This is again where CI paradigms in DM apply, as a software system capable of falling back on evolutionary approaches is able to find out the best solution on its own and to provide the desired results by applying this solution on the presented problem.

6.2 Autonomously acting cars

The development of autonomously acting cars (AAC) requires one to understand how certain aspects of the human brain function, such as object recognition and some predictive capabilities, so I will very briefly give an introduction into the field, go into some of the development challenges and explain where ANNs can successfully be applied. AACs rely on a huge amount of manifold sensors and algorithms, including LiDAR for recognizing moving objects, cameras and related algorithms for identifying what this moving object is, radar and ultrasonic for better interpreting its own position in a dynamic world. Of course logic is not to be omitted, as the domain controller (the car's brain) has to make decisions based on what it perceives through its sensors:

- interpret the car's surroundings
- anticipate upcoming events and predict the necessary reactions
- instruct the various hardware components of the car to perform the necessary actions

Basically, there exist two approaches in AAC development:

---

258 Sasu Lucian Mircea (2006): Computational Intelligence Techniques in Data Mining; PhD Thesis; Brasov: Transylvania University of Brasov, p. 9

259 Sasu Lucian Mircea (2006): Computational Intelligence Techniques in Data Mining; PhD Thesis; Brasov: Transylvania University of Brasov, p.10

260 Sasu Lucian Mircea (2006): Computational Intelligence Techniques in Data Mining; PhD Thesis; Brasov: Transylvania University of Brasov, p.10

– V2X-communication
– massively equip car with hardware and algorithms

6.2.1 V2X-communication

The input comes to a significant part from infrastructure, i.e., road sensors, intersection management systems, etc. and comparing a LiDAR-obtained profile of the 360° surroundings of the car (comparing that image to a map database + identifying differences between the two images as obstacles). The advantages are that the autonomously acting agent

– can be made quite reliable over time,
– covers relatively large distances, and
– relatively low cost (from the car's perspective).

The disadvantages in the V2X-communcation-only approach are

– high initial cost (need to build out infrastructure and a detailed street-view map database), and
– the car's ability to react to sudden changes.

6.2.2 Massively equip car with processing power and AI-algorithms

In terms of understanding what needs to be done to create an autonomously acting agent in the dynamic real world, it is useful to second on the second approach, massively equipping a car with hardware and AI-algorithms, and augment it with the first (as we do not necessarily need to copy how humans function to achieve the same goals). In this implementation scenario, the car does not receive any external navigation instructions, and relies on algorithms and logic applied on streams from cameras, radar, LiDAR and other sensors, which give it a 360° knowledge of the surrounding environment. The advantages are that the autonomously acting agent

– can react quickly to situations, and
– focus only on what is important, while
– ignoring everything else (most important).

The disadvantages in this approach are mostly related to weaknesses in the sensors and cost, namely

– relatively high car cost (at least near term),
– sensitivity to weather and
– other sources of electronic signal blockage.

6.2.3 Artificial intelligence and environment sensing

The most secure approach is a mix of both approaches, whereby nowadays we see improvement potential especially in V2V-communication, as AACs mostly act as singular agents instead the fleet as multi agent system, i.e. some of the areas of interest related research currently goes incorporate

- Each car broadcasting what it senses to all others in its surroundings
- Scene understanding instead of object recognition is the target
- Better recognition by more sophisticated algorithms (i.e., deep learning ANNs)
- Integration of auditory perception

As already introduced, there are manifold sensor data used in a car for AI tasks, including

- temperature sensors, steering and breaking maneuvers for doing predictive maintenance or traffic prediction,
- rain and temperature sensors for weather analysis,
- ultrasonic, radar, LiDAR, breaking maneuvers, GPS for analyzing anomalies in traffic flow,
- cameras, LiDAR, ultrasonic and radar for environment perception, etc.

Here, the focus is on environment perception and the sensors used for achieving this non-trivial target.

6.2.3.1 Cameras and how AI is applied to related data

Monovision-cameras are, due to low resolution and low-quality imagery mostly used for fixed infrastructure recognition, i.e. lane markings or speed limit signs. With appropriate algorithmmic support, they allow fast image processing in order to recognize common roadside infrastructure from a simple black and white relatively low-resolution image.

Stereo-vision cameras allow depth perception and thus are used for understanding relative position of moving traffic and obstacles as well as in short-range perception.

Amongst others, AI algorithms such as ANNs on the data from cameras may be used to

- conduct object recognition,
- do fast image processing to recognize common roadside infrastructure from a simple black and white relatively low-resolution image,
- do depth perception,
- to understand the AAC's relative position of moving traffic and obstacles, and be
- combined with other sensory information used to get to scene understanding instead of mere object recognition.

6.2.3.2 RADAR and how AI is applied to related data

RADAR is used for collecting information about the car's immediate surroundings, whereby "immediate" depends from the car speed. For low speed short range- RADAR is used in order to sense the ACC's immediate surroundings, especially at low speeds. Long range-RADAR is applied at high speeds and over relatively long distances. Long distance radar combined with algorithm-based processing of images from stereovision cameras gives the autonomous car the capability of

- knowing what is in front of it as well as
- how the positions and profiles of external objects are changing at all times.
- Combined with LIDAR, can be used to measure velocity of objects (Doppler effect).

Amongst others, AI algorithms such as ANNs on the data from RADAR may be used to

- conduct fusion of RADAR and algorithm-based processing of images from stereovision cameras,
- recognize obstacles in the front of the ACC,
- change of positions and profiles of external objects over time.

A disadvantage or RADAR is that compared to LiDAR the field of view is narrow, which is why fusion with other sensory information is required.

6.2.3.3 LiDAR and how AI is applied to related data

LiDAR makes use of a combination of reflected laser/ light (LI) and radar (DAR) to create a 3D profile of the surroundings of the car. A LiDAR creates a rapid series of 360° profiles and compares them to each other and to a database storing historic data in order to detect changes (i.e., moving objects). This is also why LiDAR cannot be used exclusively – snowfall, e.g., makes up an ever-changing environment and piles of snow should not be identified as moving object only because they are not stored in the LiDAR database. LiDAR comes with some advantages, namely that algorithms on it allow to

- accurately detect movement, and
- compared to RADAR the field of view is broad.

Some of the disadvantages and open questions are that

- temporary changes (like snow or new traffic patterns) could disrupt the surrounding profile,
- it does not work for some aspects of autonomous driving like lane and sign tracking (camera/ vision systems required), and
- velocity measurement current area of research (Doppler effect).

Amongst others, AI algorithms such as ANNs on the data from LiDAR may be applied for

- identifying (moving) objects within the scans, e.g. comparing scans, edge-detection over scans in order to get different views from an object,
- matching scans with LIDAR database (and identify/ ignore disturbances),
- identifying unique coordinates in the LIDAR-scan and camara images for overlaying,
- extracting frame coordinates around moving objects from LIDAR, mapping it into camera and identifying what the object is.

6.2.3.4 Additional sensors and how AI is applied to related data

Sensors of all kinds already extensively used in cars, including acceleration sensors, pressure sensors, light sensors etc. An ACC also needs to monitor itself to know that it is not traveling over the speed limit, if something is wrong with the car, predictive maintenance and a lot more. Thus, these sensors may be used for

- improving security, and
- human-machine interaction.

Amongst others, AI algorithms such as ANNs on the sensor data may be applied for

- active-safety related aspects, such as recognizing road conditions by haptic feedback (e.g. vibrations from steering wheel, suspension),
- driver monitoring (attention, health conditions), i.e.
  - driver condition monitoring,
  - driver intention estimation, and
- predictive maintenance (tire, break-condition)

6.2.3.5 GPS and how AI is applied to related data

The global positioning system (GPS) is a reliable, high-speed two-way data communications equipment (antennas, 4G- and GPS receivers), mainly used for

- navigation
- V2V/ V2X communication
- content reception

usually, data recorders or black boxes are required, given the high level of automation, in the event of an accident or failure, and amongst others, AI algorithms such as ANNs on the data from GPS may be applied for

- analyzing and predicting events, and
- given information about other cars‘ positions, traffic optimization.

6.2.3.6 Microphones and how AI is applied to related data

In current research, multiple microphones in terms of AACs are used to create environment sound profiles, which in turn may e.g. be used for

- recognizing approaching emergency vehicles (Doppler effect),
- focusing sensors on particular areas (noisy, crowdy, particular noises such as crashes).

and amongst others, AI algorithms such as ANNs on the data from GPS may be applied for classification of events based on ambient noise analysis.

6.2.3.7 Autonomously acting car’s brain – the domain controller

The domain controller is a computer acting as the brain of the autonomously acting car, as it processes all of the produced data and algorithms, as well as executes the logic to be executed based on what the AAC senses. Thus, it acts as the crossover between the input and output systems of the car by receiving signals from the various cameras, radar, and sensors, determining what action is to be taken and then communicating with the car’s drivetrain to execute the necessary actions. Summing up, this is where

- the software brain/ operating system of the car resides, and
- where machine learning is meant to be executed,
- fusing signals from cameras, radar, and sensors happens,

– necessary actions on the algorithmic output are determined (which will then be used for communicating with the car's drivetrain to execute these).

In terms of scene understanding, current AACs do not understand what they sense, as sensory information heavily used to do object recognition only. However, understanding context is one of the most complex topics in AI research and will be dealt with in detail at 10.6.2 Context recognition and hierarchical learning. In terms of an AAC understanding a scene can contribute to safety, e.g. sensory information from close other AACs can be broadcast.

What we also do not see in humans intelligence is, at least as far as we know, re-training and re-deployment of models. The common approach is to train an AI algorithm on historic data, deploy it and if required, retrain and redeploy. Data, i.e. environments, may change over time, not only in terms of configurations (which objects we see), but also in terms of content (what is there that has not been there at training time, how single agents behave, etc.), and humans can adapt to such changes perfectly fine, thus the AACs can potentially learn from human behavior, i.e. when interaction is required and a large group of users shows the same reactions in identical situations, or erratic behavior of other ACCs.

Future research will most likely treat autonomous cars not as individuals, but as multi-agent systems, which requires to incorporate the three aspects of multi-agent systems (see 2.5 Agents and actions)

– Autonomous behavior: model decisions should be based on beliefs, desires and intentions (BDI-agents)
– Adaptive behavior: It is impossible to predict any situation, thus the agents must be able to adapt to not-yet-perceived situations, e.g. by means of reinforcement learning
– Social behavior: V2V-communication should be exploited in order to share information between agents, and also communication from a central control center to all agents in order to optimize traffic. This information could be used to develop services, i.e. the AAC recognizing accidents and calling emergency services, and let AACs automatically form rescue lanes in case of an accident.

6.3 Summary

This chapter showed that most of the tasks in AI rely on analyzing data, which is also why I strongly focused on data mining and related areas of application. A brief overview about some of the data collected and produced by autonomously acting cars and how these are efficiently leveraged in order to rebuild aspects of human intelligence, i.e. safely reacting in a dynamic environment, helps to understand some of the challenges involved in developing AI-systems. It also helps to understand how we can develop artificially intelligent agents performing human-like tasks, but not necessarily by exactly the same means (humans do not use LiDAR, and humans do not perceive images as 4D-tensors).

7 An outline of quantum mechanics

Quantum computer science is one of the fields that seem to be very promising for future developments within the field of artificial intelligence, especially in terms of artificial neural networks or Markov models. This is due to the effects of quantum mechanics, which offer completely new possibilities for ANN learning. In the last years the interest in theoretical aspects of quantum physics has grown intensely and thus, also research in computational neuroscience discovered it and is now trying to make use of its possibilities. Quantum information theory is sometimes confusing computer scientists, especially because of its otherness compared to classical information processing. Therefore, the following chapter provides a brief introduction into quantum computer science and its paradigms with respect to artificial neural networks, or more precisely, quantum artificial neural networks.

7.1 Quantum systems in general

There exist several important aspects in quantum mechanics that are useful for quantum computer science and which need to be elucidated in detail. The first distinction occurs in information processing. A similarity between classical computers and quantum computers is that the former process information in bits (on the most granular level), and the latter ones through quantum bits (Qbits, Qbit-states), whereby I go along with the physicist David N. Mermin, who refuses to use the abbreviation qubit due to linguistic reasons. In a two-state quantum system, the states $|0\rangle$ and $|1\rangle$, which at a first glance look similar to the classical states of a von Neumann computer (0 and 1), form the basis.

Before going into detail, I require the readers to accept the following in advance, although the meaning of quantum linear superposition will be explained not until 7.1.3.2 Quantum linear superposition. The information a Qbit contains is (represented by) the superposition of the two independent states within a complex vector space, and is written as

$$|\psi\rangle = \alpha_1|0\rangle + \alpha_2|1\rangle = \begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix} \qquad (7\text{-}1)$$

The probability amplitudes α_1 and α_2, which describe the probability that the Qbit is either associated with the one or the other state, are complex numbers that in quadratic power must sum to unity:

$$|\alpha_1|^2 + |\alpha_2|^2 = 1 \qquad (7\text{-}2)$$

Thus, the state $|\psi\rangle$ associated with a Qbit can be any unit vector in the two-dimensional vector space spanned by $|0\rangle$ and $|1\rangle$ over the complex numbers, where α_1 and α_2 are two complex numbers constrained only by the requirement that $|\psi\rangle$, like $|0\rangle$ and $|1\rangle$, needs to be a unit vector in the complex vector space. Moreover and as indicated beforehand, a Qbit cannot be said to have a value, instead it is associated with a state, in contrary to a classical bit, which can be said to be in a concrete state. [261] A quantum system that is composed of n Qbits has

261 Mermin David N. (2007): Quantum Computer Science: An Introduction; Cambridge: Cambridge University Press

F. Neukart, *Reverse Engineering the Mind*, AutoUni – Schriftenreihe 94, DOI 10.1007/978-3-658-16176-7_7

$N = 2n$ independent states obtained through the tensor product of the complex vector space already addressed. A quantum systems' wave function exists in is the Hilbert space, which consists of a set of states $|\phi\rangle$. Thus, a quantum system can also be described by

$$|\psi\rangle = \sum_i \alpha_i |\phi_i\rangle \qquad (7\text{-}3)$$

where α_i represents the complex coefficients (probability amplitudes) describing the probability of the system collapsing into the state ϕ_i when measuring it.

So much to the first impression of how quantum systems are described. I started with these theoretical fragments as I consider it important to realize some facts before dealing with quantum theory in detail:

- A quantum system, be it a just a single Qbit or a complex one, does not simply exist – it is dynamic and evolves through time.
- It is associated with states, and not in concrete states, and
- its probability amplitudes are complex numbers in Hilbert space.

7.1.1 Quantum theory

Quantum theory (as all physical theories) can be characterized by how it represents (physical) states, observables, measurements, and dynamics (evolution in time).[262] As Jyh Ying Peng explains the axioms of quantum mechanics very well in his lecture notes,[263] these have formed the foundation for the coming chapters.

7.1.1.1 Quantum states

A state in a common physical system is its complete description. In quantum mechanics, a state is a ray in a Hilbert space, where a Hilbert space is a vector space over the field of complex numbers, denoted by C. The vectors in the Hilbert space are denoted by Dirac's Bra(c)Ket notation $|\psi\rangle$. A ket $|\psi\rangle$ can either be seen an n element vector or an $n\ x\ 1$ matrix, where n forms the dimension of the Hilbert space. Its corresponding bra $\langle\psi|$ is the complex conjugate transpose (adjunct) of the ket. Its inner product $\langle\psi|\phi\rangle$ may be seen as matrix multiplication mapping an ordered pair of vectors to C, with some special properties:

Positivity

$$\langle\psi|\psi\rangle > 0\ for\ |\psi\rangle \neq 0 \qquad (7\text{-}4)$$

Linearity

$$\langle\phi|(\alpha_1|\psi_1\rangle + \alpha_2|\psi_2\rangle) = \alpha_1\langle\phi|\psi_1\rangle + \alpha_2\langle\phi|\psi_2\rangle \qquad (7\text{-}5)$$

---

262 Jyh Ying Peng (2003): Quantum Computation Lecture Notes [2012-10-06]; URL: http://red.csie.ntu.edu.tw/QC/peng/chap1.pdf

263 Jyh Ying Peng (2003): Quantum Computation Lecture Notes [2012-10-06]; URL: http://red.csie.ntu.edu.tw/QC/peng/chap1.pdf

Skew symmetry

$$\langle\phi|\psi\rangle = \langle\phi|\psi\rangle^* \tag{7-6}$$

Furthermore, the Hilbert space is complete (or contains all necessary limits) in the norm

$$||\psi|| = \langle\psi|\psi\rangle^{\frac{1}{2}} \tag{7-7}$$

A ray in a Hilbert space is represented by a given vector and all its complex multiplies. Two different vectors (rays) will never be parallel in that space. Rays are represented by vectors with unit norm

$$\langle\psi|\psi\rangle = 1 \tag{7-8}$$

so that $e^{i\alpha}|\psi\rangle$, where

$$\alpha =\in \mathbb{R} \tag{7-9}$$

represent the same physical state for all α. By the superposition

$$\alpha_1|\phi\rangle + \alpha_2|\psi\rangle \tag{7-10}$$

new states can be formed, and the relative phase between the two components is physically relevant, which means whereas

$$\alpha_1|\phi\rangle + \alpha_2|\psi\rangle \tag{7-11}$$

and

$$e^{i\alpha}(\alpha_1|\phi\rangle + \alpha_2|\psi\rangle) \tag{7-12}$$

represent the same physical state,

$$\alpha_1|\phi\rangle + e^{i\alpha}\alpha_2|\psi\rangle \tag{7-13}$$

is a different one. [264]

7.1.1.2 Observables

A property of a physical system is that it can be measured. This is not different in quantum mechanics, where an observable is a self-adjunct operator, thus a matrix, and a linear map taking vectors to vectors. This is represented by matrices. For an operator

$$A, A: |\psi\rangle \longrightarrow A|\psi\rangle \tag{7-14}$$

and

$$A(\alpha_1|\psi\rangle + \alpha_2|\phi\rangle) = \alpha_1 A|\psi\rangle + \alpha_2 A|\phi\rangle \tag{7-15}$$

---

264 Jyh Ying Peng (2003): Quantum Computation Lecture Notes [2012-10-06]; URL: http://red.csie.ntu.edu.tw/QC/peng/chap1.pdf

The adjunct $A^\dagger$ of an operator A is defined by

$$\langle\phi|A\psi\rangle = \langle A^\dagger\phi|\psi\rangle \tag{7-16}$$

where $|A\psi\rangle$ simply denotes $A|\psi\rangle$ for all vectors $|\psi\rangle$, $|\phi\rangle$. In matrix form the adjunct is represented as the (complex) conjugate transpose. A is self-adjunct, if $A = A^\dagger$. The eigenvectors of the corresponding self-adjunct matrix, the eigenstates, form a complete orthonormal basis in the Hilbert space. [265] Orthonormal means that all vectors of the space feature the length 1 and the dot product of two of them is 0, which means that they are orthogonal to each other (see the orthonormalbase of the three-dimensional Euclidean space vectors (7-17) - (7-19)):

$$\vec{\imath} = \begin{pmatrix}1\\0\\0\end{pmatrix} \tag{7-17}$$

$$\vec{\jmath} = \begin{pmatrix}0\\1\\0\end{pmatrix} \tag{7-18}$$

$$\vec{k} = \begin{pmatrix}0\\0\\1\end{pmatrix} \tag{7-19}$$

7.1.1.3 Quantum measurements

The formulation of quantum mechanics describes the deterministic unitary evolution of a wave function, which can never be observed experimentally and which allows to compute the probability that certain macroscopic events will be observed. There are no events and no mechanism for creating events in the mathematical model. It is this dichotomy between the wave function model (see eqs. (7-22) - (7-26)) and observed macroscopic events that is the source of the interpretation issue in quantum mechanics. In contrary to classical physics, where the mathematical model talks about the things that can be observed, in quantum mechanics the mathematical model by itself never produces observations, since the wave function must be interpreted in order to relate it to experimental observations. [266] The numerical outcome of the measurement of the observable (=self-adjunct operator) A is an eigenvalue of A, and directly after measurement the quantum state becomes the eigenstate of A corresponding to the measurement result, and if the quantum state before measurement is $|\psi\rangle$, then outcome α_n is obtained with the probability

$$P(\alpha_n) = \left||P_n|\psi|\right|^2 = \langle\psi|P_n|\psi\rangle \tag{7-20}$$

As a consequence, the normalized quantum state becomes

265 Jyh Ying Peng (2003): Quantum Computation Lecture Notes [2012-10-06]; URL: http://red.csie.ntu.edu.tw/QC/peng/chap1.pdf

266 Budnik Paul: The measurement problem [2012-10-06]; URL: http://www.mtnmath.com/faq/meas-qm-2.html

$$\frac{P_n|\psi\rangle}{\langle\psi|P_n|\psi\rangle^{\frac{1}{2}}} \tag{7-21}$$

If the measurement is then repeated immediately, the same result would be obtained with the probability of one. [267]

7.1.1.4 Quantum dynamics

Time evolution of a quantum state is unitary, and a unitary transformation can be seen as a rotation in Hilbert space. Evolution happens via a special self-adjunct operator, called the Hamiltonian[268] of the system. In the Schrödinger picture of dynamics, the Hamiltonian (vector or state) describing that the system evolves time-independently is determined by the Schrödinger equation

$$\hat{H}\psi(r) = \frac{-\hbar^2}{2m} \Delta\, \psi(r) + V(r)\psi(r) = E\psi(r) \tag{7-22}$$

where the Laplace-operator

$$\Delta\, \psi(r) = \Delta\, |\psi\rangle = \nabla^2|\psi\rangle = \frac{\partial^2}{\partial x^2}|\psi\rangle + \frac{\partial^2}{\partial y^2}|\psi\rangle + \frac{\partial^2}{\partial z^2}|\psi\rangle \tag{7-23}$$

and in time according to the time-dependent Schrödinger equation

$$i\hbar\frac{\partial}{\partial t}\psi(r,t) = \frac{-\hbar^2}{2m}\ \Delta\, \psi(r,t) + V(r,t)\psi(r,t) \tag{7-24}$$

where

$$\hat{H}\psi(r,t) = i\hbar\frac{\partial}{\partial t}\psi(r,t) \tag{7-25}$$

and

$$E\psi(r,t) = \frac{-\hbar^2}{2m} \Delta\, \psi(r) + V(r)\psi(r) \tag{7-26}$$

$\hbar$ is the reduced Planck's constant $\frac{h}{2\pi}$, $\frac{-\hbar^2}{2m}\Delta$ the kinetic energy and V the potential energy of the particle (a particle may have differing values for its potential energy, depending from where it is located, e.g. in a gravitational field or an electric field). $\frac{\partial^2}{\partial x^2}\psi(x,t)$, is the derivation of the wave function with respect to its position x. $\hat{H}$ is the Hamiltonian, which allows a simplified description of the time-evolution of the physical system and the related wave function ψ, and i is the imaginary unit. The Hamiltonian function (equation (7-27)) itself allows the creation of the Hamilton operator (equation (7-28)):

$$\hat{H}(x,p) = \frac{p^2}{2m} + V(x) \tag{7-27}$$

---

267 Jyh Ying Peng (2003): Quantum Computation Lecture Notes [2012-10-06]; URL: http://red.csie.ntu.edu.tw/QC/peng/chap1.pdf

268 Jyh Ying Peng (2003): Quantum Computation Lecture Notes [2012-10-06]; URL: http://red.csie.ntu.edu.tw/QC/peng/chap1.pdf

$$\hat{H}(\hat{x},\hat{p}) = \frac{\hat{p}^2}{2m} + V(\hat{x}) \qquad (7\text{-}28)$$

This happens by replacing of the momentum p and location x by the related momentum operator $\hat{p}$, working as the derivative $-i\hbar\frac{\partial}{\partial x}$, and the location operator $\hat{x}$, $V(\hat{x})$ then being the potential-operator working as multiplication with the potential function $V(x)$.[269]

7.1.2 Quantum operators

For being able to conduct operations on a quantum computer, operators are required, similar to a classical von Neumann computer. Operators on a Hilbert space describe how a wave function is transformed into another; they may be represented as matrices acting on vectors, where the notation $\langle .|$ indicates a column vector and the $|.\rangle$ complex conjugate row vector. Using operators, (7-3) can be created. The solutions ϕ_i to such an equation are called eigenstates and can be used to construct the basis of a Hilbert space. In the quantum formalism, all properties are represented as operators whose eigenstates are the basis for the Hilbert space associated with that property and whose eigenvalues are the quantum allowed values for that property. It is of utmost importance to mention that operators in quantum mechanics must be linear operators and that they must be unitary.[270] An operator is unitary, when its inverse equals its adjunct (which is the complex conjugate transpose):

$$U^{-1} = U^{\dagger} \qquad (7\text{-}29)$$

The usage of unitary operators U is especially required when trying to solve problems of the type

$$Ax = y \qquad (7\text{-}30)$$

The solution

$$x = A^{-1}y \qquad (7\text{-}31)$$

is simple when one knows A^{-1}. However, the calculation of the inverse of a large matrix may not always be simple, which is the primary reason for the application of unitary operators. It goes beyond this thesis to discuss all of the quantum operators in detail, but it is important to know the following ones, taken from Mermin's introduction to quantum computer science:[271]

The NOT-operator X

$$X\colon |x\rangle \longrightarrow |\tilde{x}\rangle \qquad (7\text{-}32)$$

which does

---

269 University of Wuppertal (2013): Der Hamiltonoperator [2013-01-25]; URL: http://hydrogen.physik.uni-wuppertal.de/hyperphysics/hyperphysics/hbase/quantum/hamil.html

270 Ricks Bob, Ventura Dan (2003): Training a Quantum Neural Network; Provo: Brigham Young University

271 Mermin David N. (2007): Quantum Computer Science: An Introduction; Cambridge: Cambridge University Press

$$\tilde{1} = 0 \qquad (7\text{-}33)$$

and

$$\tilde{0} = 1 \qquad (7\text{-}34)$$

as well. The NOT-operator is reversible, as a second application brings back the state to its initial state:

$$X^2 = 1 \qquad (7\text{-}35)$$

When the orthogonal states of a bit are represented by

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad (7\text{-}36)$$

and

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad (7\text{-}37)$$

then the linear operator of X expressed on the two-dimensional vector space is

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad (7\text{-}38)$$

and

$$1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad (7\text{-}39)$$

where the latter one flips the bit and the former one leaves it in its current state.

The swap-operator s_{ij}

$$s_{10}|xy\rangle = |yx\rangle \qquad (7\text{-}40)$$

changes the states of the two bits i and j, where the example s_{10} does the following on

$$s_{10}|01\rangle = |10\rangle \qquad (7\text{-}41)$$

or

$$s_{10}|10\rangle = |01\rangle \qquad (7\text{-}42)$$

but does leaves $|11\rangle$ and $|00\rangle$ as they are, as an interchange results in the same state.

The cNOT operator c_{ij}

$$c_{10}|x\rangle|y\rangle = |x\rangle|y \oplus x\rangle \qquad (7\text{-}43)$$

$$c_{01}|x\rangle|y\rangle = |x \oplus y\rangle|y\rangle \qquad (7\text{-}44)$$

where $\oplus$ is the modulo 2 addition (or the XOR). The c in *cNOT* or c_{ij} stands for controlled, meaning that if the state of the i^{th} bit is $|0\rangle$ then cNOT does not change the state of the j^{th} bit, but if the i^{th} bit is $|1\rangle$, then the j^{th} bit will be subject to X:

$$c_{10}|01\rangle = |11\rangle \tag{7-45}$$

and

$$c_{10}|11\rangle = |01\rangle \tag{7-46}$$

and leaves $|10\rangle$ and $|00\rangle$ as they are, as the cNOT on these bits results in the same state. This changes when the control bit is on the right. If a vector space consists of two bits, the basis is $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, and the vector assignments are

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{7-47}$$

$$|01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \tag{7-48}$$

$$|10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{7-49}$$

$$|11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{7-50}$$

Thus, the cNOT operator for a two bit-space as unitary version of XOR is a permutation matrix of the form

$$c_{ij} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{7-51}$$

The n-operator

$$n|x\rangle = x|x\rangle \tag{7-52}$$

where

$$x = 0 \vee 1 \tag{7-53}$$

and

$$\tilde{n} = 1 - n \tag{7-54}$$

and

$$+\tilde{n} = 1 \tag{7-55}$$

n is the projection operator of $|1\rangle$ and $\tilde{n}$ projects onto $|0\rangle$. As matrix operators

$$n = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \tag{7-56}$$

and

$$\tilde{n} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \tag{7-57}$$

$|0\rangle$ and $|1\rangle$ are eigenvectors of n with eigenvalues 0 and 1, and so it is for $\tilde{n}$ with eigenvalues of 1 and 0.

The Z-operator

$$Z = n - \tilde{n} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{7-58}$$

According to the matrices X anticommutes with Z:

$$ZX = -XZ \tag{7-59}$$

Furthermore, as

$$n + \tilde{n} = 1 \tag{7-60}$$

then

$$n = \frac{1}{2}(1 - Z) \tag{7-61}$$

and

$$\tilde{n} = \frac{1}{2}(1 + Z) \tag{7-62}$$

Also c_{ij} can be rewritten:

$$c_{ij} = \frac{1}{2}(1 + Z_i) + \frac{1}{2}X_j(1 - Z_i) = \frac{1}{2}\left(1 + X_j\right) + \frac{1}{2}Z_i\left(1 - X_j\right) \tag{7-63}$$

An interchange of the operators X and Z has the effect of switching the control and target-bits of c_{ij}. Thus, c_{ij} becomes c_{ji}.

The Hadamard transformation H

$$H = \frac{1}{\sqrt{2}}(X + Z) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{7-64}$$

Equation (7-63) shows the interchange of the control and target-bits of the cNOT operator, which is done by the Hadamard transformation H. Furthermore,

$$X^2 = Z^2 = 1 \tag{7-65}$$

and

$$XZ = -ZX \tag{7-66}$$

and

$$H^2 = 1 \tag{7-67}$$

and

$$HXH = Z \tag{7-68}$$

and

$$HZH = X \tag{7-69}$$

so H can be used to interchange X and Z in c_{ji}:

$$c_{ji} = (H_i H_j) c_{ij} (H_i H_j) \tag{7-70}$$

The important point for quantum computation is the difference between s_{ij} and $H_i H_j$, as $H_i H_j$ is a product of two 1-bit operators, while s_{ij} is not.

$$|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + |1\rangle \tag{7-71}$$

and

$$H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle - |1\rangle \tag{7-72}$$

Although the application of H on bits is not a transformation that makes sense, a combination with other operators like the one in equation (7-69) may be used. In a quantum computer, the application of H on a 1-Qbit state for interchanging control and target bit of cNOT is perfectly useful, as it is a simple 1-Qbit operator and therefore easy to implement.

There is a lot more that can be said about quantum operators, e.g. Pauli's e-vector $\vec{\sigma}$, or three complex matrices, used for the representation of the rotation of a spin-1/2 object (Fermions), has not been explained, but for this elaboration the basic knowledge does suffice.

7.1.3 Quantum physical effects

Quantum physical systems show several properties cannot be observed in the macroscopic world. Such effects are on the one hand of considerable importance for quantum computation as well as the search for hard- and software capable of processing conscious experiences.

7.1.3.1 Quantum interference

Quantum interference is, although it is a familiar wave phenomenon, one of the most interesting principles of quantum mechanics. Considering waves, such interfere constructively with each other when being in phase and interfere destructively when out of phase. In quantum mechanics, this interference applies to the probability waves, where the wave

function interferes with itself through the action of an operator; the different prates of the wave function interfere constructively or destructively according to their relative phases like any other kind of wave. [272] Thomas Young's double-slit experiment describes the confusion interference in quantum mechanics produces. Deutsch's well-known problem will serve to explain how quantum interference can be of use in quantum computation. Let assume we want to determine if a function f is constant or balanced, thus if $f(0) = f(1)$ or $f(0) \neq f(1)$. All information we possess about f is

$$f(0) = 1 \tag{7-73}$$

$$(1) = 0 \tag{7-74}$$

so the function itself is a black box. What we can additionally say is that if the function is balanced, it is reversible, and if it is constant, it is not (as the result for both inputs is the same – one cannot say what the last operation was). The function call will be implemented as a unitary and thus reversible transformation

$$U_f: |x, y\rangle \rightarrow |x, y \oplus f(x)\rangle \tag{7-75}$$

First of all, a quantum register consisting of two Qbits has to be initialized:

$$|x\rangle|y\rangle \leftarrow |0\rangle|1\rangle \tag{7-76}$$

Via the Hadamard-transformation an equal superposition of the two-Qbit register can be created:

$$|\psi\rangle = \left(\tfrac{1}{\sqrt{2}}|0\rangle + |1\rangle\right)\left(\tfrac{1}{\sqrt{2}}|0\rangle - |1\rangle\right) = \tfrac{1}{2}(|0\rangle|0\rangle - |0\rangle|1\rangle + |1\rangle|0\rangle - |1\rangle|1\rangle) \tag{7-77}$$

The function call, which is U_f applied to the input register, results in

$$|\psi\rangle = \tfrac{1}{2}(|0\rangle|0 \oplus f(0)\rangle - |0\rangle|1 \oplus f(0)\rangle + |1\rangle|0 \oplus f(1)\rangle - |1\rangle|1 \oplus f(1)\rangle) = \tfrac{1}{2}(|0\rangle|1\rangle - |0\rangle|0\rangle + |1\rangle|0\rangle - |1\rangle|1\rangle) \tag{7-78}$$

This must be simplified further, so that we can observe the result of a second Hadamard transformation on the first bit (we remember that if the function is reversible, it is balanced and the first Qbit will be in the state $|0\rangle$ again).

$$|\psi\rangle = \tfrac{1}{2}(-|0\rangle(|0\rangle - |1\rangle) + |0\rangle(|0\rangle - |1\rangle)) = \tfrac{1}{2}(|1\rangle - |0\rangle)(|0\rangle - |1\rangle) \tag{7-79}$$

The application of H results in

$$|\psi\rangle = \tfrac{1}{2} - \sqrt{2}|1\rangle\sqrt{2}|1\rangle = -|1\rangle|1\rangle \tag{7-80}$$

Thus, we now know that the function is balanced. Interference comes into play before the second Hadamard transformation is applied, as in case the function is constant, the amplitudes for $|0\rangle$ sum to 1 and the amplitudes for $|1\rangle$ cancel each other out ($\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}} = 0$).

---

272 Ricks Bob, Ventura Dan (2003): Training a Quantum Neural Network; Provo: Brigham Young University

$$\pm\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\rightarrow\pm\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)+\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)\right) \quad (7\text{-}81)$$

In case f is balanced, the situation behaves vice-versa.

Excursus: Thomas Young's double-slit experiment

In the double-slit experiment, a beam of light is aimed at a barrier with two vertical slits. After the light passes through the slits, the resulting pattern is recorded on a photographic plate. When one slit is covered, a single line of light is displayed, aligned with whichever slit is open. Intuitively, one might hypothesize that if both slits were open, the resulting pattern would display as two lines of light, aligned with the slits. What occurs in practice, however, is that light passing through the slits and displayed on the photographic plate is entirely separated into multiple lines of lightness and darkness in varying degrees. The result illustrates that interference is taking place between the waves and particles going through the slits in what a layman might expect to be two non-crossing trajectories. If the beam of photons is slowed enough to ensure that individual photons are hitting the plate, one might expect there to be no interference and a pattern of light would be two lines of light, aligned with the slits. The results of the experiment, however, indicate the presence of interference. Somehow, the single particles are interfering with themselves. On the face of things, this might seem impossible: one expects that a single photon will go through one slit or the other and will end up in one of two possible light line areas. That expectation, however, is invalidated by the results of the double-slit experiment. What actually occurs is that each photon not only goes through both slits, but also simultaneously traverses every possible trajectory en route to the target. Research into this phenomenon has demonstrated that other elementary particles, such as electrons, exhibit the same behavior.[273]

7.1.3.2 Quantum linear superposition

The linear superposition or coherence, which results in quantum parallelism, is represented by the quantum state (7-3) in Hilbert space, with complex coefficients c_i and a set of states ϕ_i in this space. Quantum mechanics dictates that the interaction of a system with its environment results in the destruction of its superposition. This allows the conclusion that the construction of a quantum computer that will not be influenced by its own irrelevant physical properties is difficult. The coefficients c_i give the indication of the system being associated with the state ϕ_i when measurement happens. At the beginning of this chapter it has been mentioned that the coefficients must sum to unity, which is founded on the fact that a physical system must collapse to one basis state. Furthermore, it is important to understand that states of the quantum system are rays in the Hilbert space that cannot be added to each other. It is only possible to sum up the vectors two different states consist of, however, the result is no new ray, but a two-dimensional subspace of the Hilbert space. Though, if $|\phi\rangle$ and $|\psi\rangle$ represent two states, then many different superpositions, thus the ray $\alpha_1|\phi\rangle+\alpha_2|\psi\rangle$, may be built and $\lambda(\alpha_1|\phi\rangle+\alpha_2|\psi\rangle)$ belongs to the same state, if $\lambda\neq 0$. Summing up, the quantum linear superposition contains all possible configurations of a quantum system at once, until measurement is done, which finally results in a collapse or decoherence.

273 whatis.com (2011): double-slit experiment [2012-10-06]; URL: http://whatis.techtarget.com/definition/double-slit-experiment

7.1.3.3 Quantum entanglement

Unlike bits in a von Neumann machine, whose general state can only be one of the 2^n products of $|0\rangle$ and $|1\rangle$, a general state of n Qbits is a superposition of these 2^n product states and cannot, in general, be expressed as a product of any set of 1-Qbit states. Individual Qbits that make up a multi-Qbit system cannot always be characterized as having individual states of their own - in contrast to the bits in a von Neumann machine – as they may be in nonproduct-states called entangled states.[274] When measuring a Qbit A and obtaining spin up (or down), then Qbit B (on which no access is given) will also be in the state spin up (or down), similarly for measurements on Qbit B. Another way to look at it is that in this particular correlation between the two Qbits, that is, in this particular state of the whole system, any preparation of only one of the Qbits will cause the other Qbits to have the same value. In this case the Qbits *A* and *B* are entangled. When system A becomes mixed due to interaction with system B, one can say that the two systems are entangled. Moreover, this entanglement destroys the coherence of system A, as system A collapses from the initial superposition of states to one of the states each with a certain probability.[275] For a computer scientist trying to make use of the properties of a quantum computer, entanglement may not be difficult to understand, as if one bit is in the state $|0\rangle$, another is as well. However, in the world of physics entanglement is not quite well understood, as it is indeed clear, what the resulting effects are but not how it works. What makes entanglement little understood is the fact that since quantum states exist as superpositions, these correlations exist as such as well and when coherence is lost (e.g. by measurement), then the correct correlation is somehow communicated between the Qbits, and it is this 'somehow' that is not understood.[276] Mathematically, Ricks and Ventura[277] provide a detailed description of entanglement by the density matrix formalism: the density matrix ρ_ψ of a quantum state $|\psi\rangle$ is defined as the tensor product of $|\psi\rangle$ and $\langle\psi|$:

$$\rho_\psi = |\psi\rangle\langle\psi| \qquad (7\text{-}82)$$

An example is the quantum state

$$|\psi\rangle = \tfrac{1}{\sqrt{2}}|00\rangle + \tfrac{1}{\sqrt{2}}|01\rangle \qquad (7\text{-}83)$$

In vector form, it is written as

$$|\psi\rangle = \tfrac{1}{\sqrt{2}}\begin{pmatrix}1\\1\\0\\0\end{pmatrix} \qquad (7\text{-}84)$$

---

274 Mermin David N. (2007): Quantum Computer Science: An Introduction; Cambridge: Cambridge University Press

275 Jyh Ying Peng (2003): Quantum Computation Lecture Notes [2012-10-06]; URL: http://red.csie.ntu.edu.tw/QC/peng/chap1.pdf

276 Ricks Bob, Ventura Dan (2003): Training a Quantum Neural Network; Provo: Brigham Young University

277 Ricks Bob, Ventura Dan (2003): Training a Quantum Neural Network; Provo: Brigham Young University

and as density matrix ρ, which describes the quantum system in a mixed state (of all probability distributed po ssible states), it appears as

$$\rho_\psi = \sum_i \alpha_i \, |\psi\rangle\langle\psi| = |\psi\rangle\langle\psi| = \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{7-85}$$

The quantum state

$$|\phi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \tag{7-86}$$

has the density matrix

$$\rho_\phi = |\phi\rangle\langle\phi| = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \tag{7-87}$$

where the matrices and vectors are indexed by the state labels 00,...,11. ρ_ψ may be factorized as

$$\rho_\psi = \frac{1}{2}\left(\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}\right) \tag{7-88}$$

whereas ρ_ϕ cannot be factorized, which declares them as entangled. The simplest states of entanglement between two Qbits are the Bell-states, also known as maximally entangled two-Qbit states:[278]

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B) \tag{7-89}$$

$$|\phi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B - |1\rangle_A \otimes |1\rangle_B) \tag{7-90}$$

$$|\psi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B + |0\rangle_A \otimes |1\rangle_B) \tag{7-91}$$

$$|\psi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B - |0\rangle_A \otimes |1\rangle_B) \tag{7-92}$$

These states dictate that from two individuals A and B, often named Alice and Bob, each possesses one of two entangled Qbits, described in the Bell-states by the subscripts of A and B. If A decides to measure the related Qbits then the outcome cannot be predicted, as

$$\left|\frac{1}{\sqrt{2}}\right|^2 = 0.5 \tag{7-93}$$

and therefore with equal probabilities either $|0\rangle$ or $|1\rangle$ would be measured. However, according to quantum entanglement, B would now measure the same. This is because the final state $|0\rangle$ can only the result of A's Qbit being in the state $|0\rangle$.

278 Bell John S. (1987): Speakable and Unspeakable in Quantum Mechanics, Cambridge: Cambridge University Press

7.2 The unitary evolution U

The evolution of a quantum state is, although it only exists until the collapse of the quantum linear superposition, or state vector reduction, occurs. How this state evolves through time, has already been introduced – it is called the Schrödinger picture of dynamics. Thus, the Schrödinger equation describes how a quantum state evolves through time, or in other words, how a quantum wave function changes with respect to time. It has already been mentioned that e.g. a quantum bit may be described by

$$|\psi\rangle = \alpha_1|0\rangle + \alpha_2|1\rangle \tag{7-94}$$

and represented by a hydrogen atom with its electron. To be more specific, any combination of possible final states a physical system may represent can be used to describe a superposition, as long as their complex number amplitudes sum to unity in quadratic power, like

$$|\psi\rangle = \alpha_1|0\rangle + \alpha_2|1\rangle + \alpha_3|2\rangle + \cdots + +\alpha_n|N\rangle \tag{7-95}$$

where

$$|\alpha_1|^2 + |\alpha_2|^2 + |\alpha_3|^2 + \cdots + |\alpha_n|^2 = 1 \tag{7-96}$$

and $0\rangle$,…, $N\rangle$ do not refer to the states of Qbits as before, but to the different physical states a quantum wave function may collapse into after measurement. Again, as quantum states always feature complex number amplitudes, they exist as rays in Hilbert space. The ray that belongs to the quantum state $\psi \neq 0$ is the complex subspace that it spans.

If physical states are allocated states in Hilbert space, the basic equation for the probably resulting measured values changes from

$$p(i, A, \psi) = |\langle \Lambda_1|\psi\rangle|^2 \tag{7-97}$$

to

$$p(i, A, \psi) = \frac{|\langle \Lambda_1|\psi\rangle|^2}{\langle \Lambda_1|\Lambda_1\rangle\langle\psi|\psi\rangle} \tag{7-98}$$

where $p(i, A, \psi)$ is the probability of obtaining i^{th} measurement value by measuring the state ψ with the measurement apparatus A (the arising probabilities due to measurement will be discussed in detail in the next chapter). Furthermore, we have to remember that e.g. $\langle \Lambda_1|\Lambda_1\rangle \in C$ is the dot product of a vector, which is positive definite with itself and used for determining the vector's length. For all pairs of vectors a dot product $\langle \Lambda|\psi\rangle \in C$ is defined with the following properties:

$$\langle \Lambda|\psi\rangle^* = \langle \psi|\Lambda\rangle \tag{7-99}$$

and

$$\langle \Lambda|\psi_1\alpha_1 + \psi_2\alpha_2\rangle = \langle \Lambda|\psi_1\rangle\alpha_1 + \langle \Lambda|\psi_2\rangle\alpha_2 \tag{7-100}$$

Thus, the argument is linear in the second argument and antilinear in the first

$$\langle \alpha_1\psi_1 + \alpha_2\psi_2|\Lambda\rangle = \alpha_1^*\langle \psi_1|\Lambda\rangle + \alpha_2^*\langle \psi_2|\Lambda\rangle \tag{7-101}$$

As states in the Hilbert space are rays, they cannot be added together. Admittedly, it is possible to add the vectors two different rays consist of; however, the sums do not form a ray, but rather span a two-dimensional subspace. The representatives ψ and ϕ of two rays can be used to construct many different superpositions, resulting in the ray $\alpha_1|\psi\rangle + \alpha_2|\phi\rangle$, where $\lambda(\alpha_1|\psi\rangle + \alpha_2|\phi\rangle)$ belongs to the same physical state, as long as $\lambda \neq 0$. As the different superpositions of the states ψ and ϕ form the space

$$CP^1 = S^2 \qquad (7\text{-}102)$$

each one can be thought of as a point on the two-dimensional surface of a sphere. On such a surface, ψ marks the north pole, $\psi_\perp$ the position perpendicular to it, thus the south pole. The equator corresponds to the equivalent superpositions $\frac{\psi + e^{i\alpha}\psi_\perp}{\sqrt{2}}$. Consolidating superpositions by the +-sign is not associative or commutative, hence it is not an addition.[279] This can be easily understood by having a closer look at Figure 66 - Bloch sphere, a sphere named after the physicist Felix Bloch used for describing a two-level quantum mechanical system like a Qbit is one.

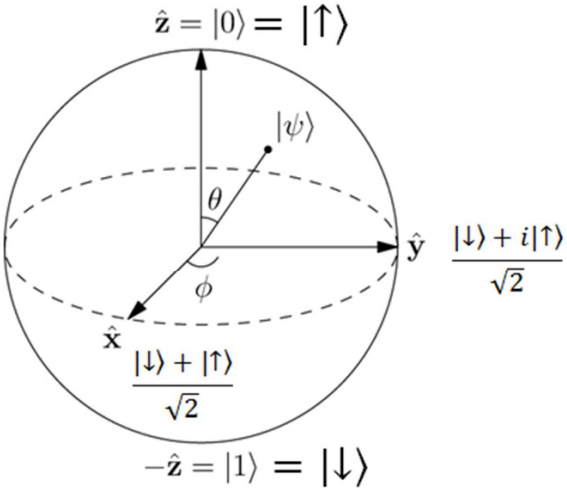

Figure 66 - Bloch sphere[280]

Figure 66 - Bloch sphere uses spin-mechanics for describing the Qbit – spin up represents $|0\rangle$ and spin down $|1\rangle$. However, a Qbit may also take all the values in between, which a classical

279 Norbert Dragon (2005): Anmerkungen zur Quantenmechanik; Hannover: University of Hannover

280 University of Copenhagen: Bloch-sphere gymnastics [2013-08-04]; URL: http://www.nbi.ku.dk/forskningsgrupper/Kvanteoptik/english/qoptlab/research/exp-clock/bloch/

bit cannot represent. An example is the point $\frac{|0\rangle+i|1\rangle}{\sqrt{2}}$, which can be found on the sphere's equator and on the positive y-axis. As we are dealing with a Qbit in our example, the probability amplitudes have to be taken into consideration, which for the Bloch sphere above are

$$\alpha_1|0\rangle + \alpha_1|1\rangle \qquad (7\text{-}103)$$

In a 3-d real vector space, the complex amplitudes of a Qbit $\alpha_1|0\rangle + \alpha_1|1\rangle$ are then described by

$$\alpha_1 = e^{\frac{-i\phi}{2}} cos\frac{\theta}{2} \qquad (7\text{-}104)$$

$$\alpha_2 = e^{\frac{i\phi}{2}} sin\frac{\theta}{2} \qquad (7\text{-}105)$$

where θ is the polar angle (the angle between the vector and the z-axis), ϕ is the azimuthal angle (the angle between the x-axis and the vector's projection onto the two-dimensional $x|y$-plane). The angle between the vector (formerly described as a ray in Hilbert-space) and the z-axis determines the probabilities of obtaining $|\uparrow\rangle$ or $|\downarrow\rangle$ along this axis, whereas the azimuthal angle represents the relative phase. Thus, the vector points into the positive range of z, when

$$\alpha_1 = |\downarrow\rangle \wedge \alpha_2 = |\uparrow\rangle \qquad (7\text{-}106)$$

and into the negative range of z, when

$$\alpha_1 = |\uparrow\rangle \wedge \alpha_2 = |\downarrow\rangle \qquad (7\text{-}107)$$

which is a representation of the general Qbit, existing in 2-d Hilbert-space, in 3-d real vector space.[281] Back to the Bloch-sphere, there are some important things to bear in mind: the surface of a sphere is two-dimensional, thus features two degrees of freedom. However, the amplitudes of a Qbit are given by complex numbers, so one might assume that the two of them feature four degrees of freedom, which in fact is not the case:

- The amplitudes require the unitarity constraint, which means that their squares must sum to unity. Thus, one degree of freedom is removed.
- Furthermore, the overall phase factor is physically irrelevant in this example, as it does not have observable consequences. If a complex number is written in the form of polar coordinates, like $r\,e^{i\theta}$, the phase factor is $e^{i\theta}$, and θ in this expression represents the phase. It is not quite correct that the phase factor does not have any influence at all; it just leaves the expectation values of a Hermitian operator as they are, so the values of $\langle\psi|A|\psi\rangle$ and $\langle\psi|e^{-i\phi}Ae^{i\phi}|\psi\rangle$ are the same,[282] but if phase factors of two quantum states interacting with each other differ, these differences might be measurable. Back to the Qbit this means that if the phase factor is only of relevance for one of the two vector components, α_1 may be left with the real part only, resulting in

$$\alpha_1 = cos\frac{\theta}{2} \qquad (7\text{-}108)$$

281 Jyh Ying Peng (2003): Quantum Computation Lecture Notes [2012-10-06]; URL: http://red.csie.ntu.edu.tw/QC/peng/chap1.pdf

282 Albert Messiah (2000): Quantum Mechanics; Dover Publications

Summing up, the removal of the overall phase as well as the unitarity constraint leaves us with only two local degrees of freedom.

Excursus: Polar coordinates

The description of complex numbers by the use of polar coordinates is very common in quantum physics and goes back to Euler's formula:

$$z = x + iy = |z|(cos\phi + isin\phi) = re^{i\phi} \quad (7\text{-}109)$$

$$\overline{z} = x - iy = |z|(cos\phi - isin\phi) = re^{-i\phi} \quad (7\text{-}110)$$

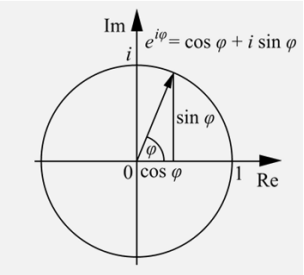

Figure 67 - Euler's formula

Back to the unitary evolution U that should be described here briefly, we now know how simple quantum systems are constructed, but not exactly how they behave (which is, by the way, something that has puzzled scientists for decades). The evolution of a quantum state dictates that with time given states of a quantum system evolve towards new states:

$$\alpha_1|\psi\rangle + \alpha_2|\phi\rangle \quad (7\text{-}111)$$

evolves to

$$\alpha_1|\psi'\rangle + \alpha_2|\phi'\rangle \quad (7\text{-}112)$$

This also applies for quantum linear superpositions consisting of more than two states. What happens is that the evolution of each of the states can be dealt with by the Schrödinger equation, without a change in the complex number weightings. It is very important to understand that although it seems that the complex number-weightings and the superposition do not play any physical role in the evolution at all, but this is in fact not the case. I refer to the next chapter here, as the experiments with half-silvered mirrors help to understand what is meant by that.

7.3 The state vector reduction R

Within chapter 10.5 we already got to know the state vector reduction of a quantum linear superposition, also called decoherence. What is so special about quantum mechanics is that although the scale of the elements its effects happen to have an influence is very small, thus on the level of molecules, atoms, electrons, photons or even fundamental particles, they can

be observed over macroscopic distances. Thus, we distinguish between two kinds of physics, the classical, Newtonian mechanics involving elements on a macroscopic scale and quantum mechanics. I go along with Roger Penrose[283] here that the finding of appropriate quantum / classical laws which operate uniformly at all scales might be a scientific advance of considerable impact. Maybe we should continue with a short excursus to Newton's three axioms, as we will encounter Newton's laws within this work very often.

Excursus: Newton's axioms

With his axioms Newton defined how we have to deal with forces and how they govern the movements of masses. All of his axioms have been proved to hold several times thitherto although dating back to 1687 or even earlier (1638 for the first axiom, specified by Galileo Galilei), where he proposed them in his Philosophiae Naturalis Principia Mathematica.

Newton's first axiom

The first axiom is commonly known as the law of inertia, stating

$$v = constant, if\ F = 0 \tag{7-113}$$

The meaning of this is simply that a body remains in a resting position or in uniformly linear movement ($v = constant$) if no external force F acts on it. An example may be the space probe Voyager that has been started in 1977, currently moving through interstellar space with approximately 17 km/s. If we neglect the now minimally acting gravitational forces of the sun and the planets of our solar system, then we can calculate the distance it is moving over time quite simple, let say for the next 10 years:

$$10 * 365 * 24 * 60 * 60s * 17\,\frac{km}{s} \approx 5.4 * 10^9 km \tag{7-114}$$

Voyager will have traveled 5,400,000,000 km when the next 10 years have passed by.

Newton's second axiom

The second axiom tells us that

$$= \frac{dp(t)}{dt} \tag{7-115}$$

In words this means that the forces F working on a body can be calculated if one knows its momentum, or that one can calculate a body's momentum $p(t)$ by knowing the forces working on it. $p(t)$ is a product of mass and velocity:

$$p(t) = m * v \tag{7-116}$$

where m is the mass of a body and v its velocity. As the momentum is a product of two components, its derivation can also be calculated with the product rule, thus be represented by

283 Roger Penrose (1994): Shadows of the mind – a search for the missing science of consciousness; Oxford: Oxford University Press; p. 308

$$F = \frac{dm(t)}{dt}v + m\frac{dv(t)}{dt} \tag{7-117}$$

where the first part of the equation specifies a change of mass over time, like one we would experience with a driving car consuming fuel. However, if the mass remains constant, then the axiom can be simplified to the well-known equation

$$F = m * a \tag{7-118}$$

simply stating that the acting force is a product of mass and acceleration. As an example we may compare the momentum of a truck, weighing 20,000 kg and moving with a speed of 80 km/h with the momentum of a small car, weighing 1,000 kg and travelling with a speed of 210 km/h.

$$p_{Truck} = \frac{80}{3.6} * 20{,}000 \approx 4.4 * 10^5 kg\ m/s \tag{7-119}$$

$$p_{Car} = \frac{210}{3.6} * 1{,}000 \approx 5.8 * 10^4 kg\ m/s \tag{7-120}$$

Therefore, the momentum of the truck exceeds the momentum of the car by a factor of approximately 7.5.

Newton's third axiom

$$actio = reactio \tag{7-121}$$

The third axiom is easy to understand, it simply says that wherever a force acts on a body, another force worked on the body this force originated from.

Why are these axioms of importance? They are important, because they can be used to deal with bodies on a macroscopic scale. They cannot be used to deal with effects on a quantum mechanical scale, as we shall see in the following chapters. First of all, we need to discuss the quantum linear superposition quantum physical systems feature as long as they do not undergo a measurement, where measurement here does not necessarily mean that the system has been measured by a human. Independent from which event the system was caused to collapse, be it an observation or a quantum gravitational effect, I will use the letter R when referring to this state vector reduction from now on. The first problem we have with R is that we need to prove that it actually takes place, although we mostly just get to see the result of R, thus a quantum physical system in a specific state and not in all of its possible states U. Before we do that, we shall again shortly address **U**, the quantum linear superposition we already discussed in chapter 10.5. For being able to express **U**, we must use complex numbers as such a system exists in a complex vector space, the Hilbert space. This is where we first need to distinguish between the laws governing classical physics and quantum physics, as in the phenomena of the former one we do not encounter complex numbers. Back to the scale at which quantum phenomena unveil, let us imagine a single photon being at one of two specific positions A and B. With classical physics, we cannot only imagine this easily, but also say that there are no contradictions in such an assertion; either the photon is at A or it is at B. However, on a quantum level we not only cannot tell where the photon is, stunningly it can occupy both places at once. How can that be? In chapter 10.5 we learned that a Qbit is represented by a probability-distributed superposition of the values of 0 and 1:

$$|\psi\rangle = \alpha_1|0\rangle + \alpha_2|1\rangle \tag{7-122}$$

where α_1 and α_2 represent the complex number amplitudes providing the probabilities the superposition $|\psi\rangle$ will either collapse into $|0\rangle$ or $|1\rangle$. We have the same situation with our photon and the positions in space here:

$$|\psi\rangle = \alpha_1|A\rangle + \alpha_2|B\rangle \qquad (7\text{-}123)$$

Although we have been talking of probabilities by now, we have to bear in mind that we are still dealing with complex numbers, thus we actually cannot say that the amplitudes represent such, as these would require real numbers to be represented. In classical physics, any physical system can be assigned a state that one intends to describe can be represented by a number of physical quantities. Even more is possible, if one is aware of the initial position of this system, namely the description of the system's state in the future.

A quantum state is also called a quantum wave function, and such a wave function is described by the Schrödinger equation I already described at 7.1.2.4 Quantum dynamics. Back to equation (8 – 11), it is correctly read as

$$he\ state\ vector \left| \begin{array}{c} \psi\rangle\ of\ a\ quantum\ system = the\ quantum\ linear \\ superposition\ of\ the\ quantum\ system's\ possible\ states\ \alpha_1 \end{array} \right| A\rangle$$

$$and\ \alpha_2|B\rangle$$

However, beforehand we were only dealing with Qbits, thus systems featuring only two possible states. It is indeed possible and even more likely that a system features more than two states it can collapse into when being measured:

$$|\psi\rangle = \alpha_1|A\rangle + \alpha_2|B\rangle + \alpha_3|C\rangle + \alpha_4|D\rangle + \alpha_5|E\rangle + \alpha_n|N\rangle \qquad (7\text{-}124)$$

We just need to imagine a single particle moving around in space; without knowing its phase space position it could be anywhere within its own and the space's physical boundaries.

Let us try to understand this by an experiment often used by Roger Penrose[284,285] (whom I admire most for his work on quantum physics and human consciousness) for describing this behavior. Imagine we have a half-silvered mirror (semi-transparent) angled 45° towards a light source, reflecting half of the light sent against it and letting the other half pass through. Now let us imagine that we would have an apparatus that would allow us sending one single photon (we remember that a beam of light or light in general consists of numerous photons) against this mirror. We would then expect to find the photon either at position *A* (the end of the first path, thus if the photon would have passed through the mirror), or at position *B* (the end of the second path, thus if the photon would have been reflected). In fact that is the case, as 'finding' the photon at position *A* or *B* implies that we induced some sort of measuring. Everything seems to be fine and as we had expected the situation to be. However, before 'finding' the photon at *A* or *B*, something absolutely different from classical physics has happened, in particular the photon has been in a quantum superposition of reflection and transmission and has travelled along both pathways (Figure 68 - Photon and half-silvered mirror):

---

284 Roger Penrose (1994): Shadows of the mind – a search for the missing science of consciousness; Oxford: Oxford University Press

285 Roger Penrose (1991): The Emperor's New Mind Concerning Computers, Minds, and the Laws of Physics; Oxford: Oxford University Press

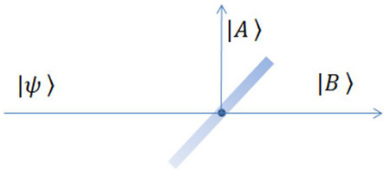

Figure 68 - Photon and half-silvered mirror

Directly after our apparatus has emitted the photon and until the photon has reached the half-silvered mirror, it is in the state $|\psi\rangle$, which may be a complex superposition as well depending from the events occurring between the light source and the half-silvered mirror, however, not one of reflection and transmission. However, before we move on to the particle/wave-dualism, we will continue with our experiment (which emanates, by the way, from the Elitzur–Vaidman bomb-testing problem, a thought experiment first proposed by Avshalom Elitzur and Lev Vaidman in 1993). Before encountering the half-silvered mirror, the photon's probability wave $|\psi\rangle$ describes the probability of it having a specific position or momentum when doing a measurement. After having encountered the half-silvered mirror evolves, in accordance with the Schrödinger picture of dynamics, into the state

$$|\psi\rangle = i|A\rangle + |B\rangle \qquad (7\text{-}125)$$

where $|B\rangle$ tells us that the photon is moving along the path towards B, and $i|A\rangle$ tells us that the photon is moving along the path towards A, i having been caused by a net phase shift occurring between the two. I say between the two here, as both exist at once, because although the half-silvered mirror alters the photon's state, it does not absorb it (which would be a measurement). Therefore, we have to deal with a non-deterministic alteration of the photon state $|\psi\rangle$ in this situation, and as such our photon is allowed (or forced, it is in the eye of the beholder) to undergo quantum linear superposition, what causes it to feature all of its possible states at once, which in our case are the reflected and transmitted states. But how can we know that, as only measurement would allow us to determine the position of the photon, or more specifically, to determine if the photon has taken path A or not. Well, for this there is a slight adaptation of the experimental situation required, namely by means of another half-silvered mirror and two fully-silvered mirrors, the latter ones allowing to bring the photons back together (Figure 69 - Photon, half-silvered and fully-silvered mirrors).

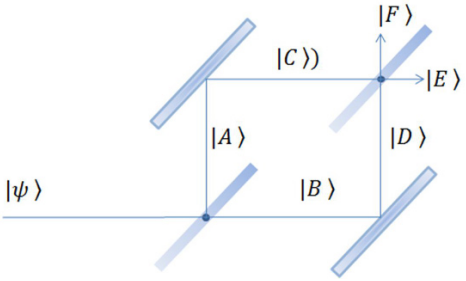

Figure 69 - Photon, half-silvered and fully-silvered mirrors

The initial situation remains the same, with an apparatus emitting a single photon towards a half-silvered mirror. However, after the photon has passed the first half-silvered mirror, no deterministic measurement happens, but again a non-deterministic alteration of the superposed state by both $i|A\rangle$ and $|B\rangle$ being reflected from fully-slivered mirrors (identifiable by the thick line around them). Thus, $i|A\rangle$ has undergone another phase-shift and evolved into $i(i|C\rangle$, and so has $|B\rangle$ which, as a consequence, has evolved into $i|D\rangle$. Summing up and according the unitary evolution quantum physical systems undergo, the following has happened, where the arrow represents an evolution towards a state:

$$i|A\rangle + |B\rangle \rightarrow i|D\rangle + i(i|C\rangle) \qquad (7\text{-}126)$$

which results in

$$|\phi\rangle = i|D\rangle - |C\rangle \qquad (7\text{-}127)$$

because of the fact that the squared imaginary unit equals -1:

$$i^2 = -1 \qquad (7\text{-}128)$$

The remarkable thing in this experiment happens at last half-silvered mirror, where the beams come together again. Depending from what we take as our initial situation, the states $i|D\rangle$ respectively $|C\rangle$ evolve to

$$|D\rangle \rightarrow i|F\rangle + i(i|E\rangle) = |F\rangle - i|E\rangle \qquad (7\text{-}129)$$

and

$$|C\rangle \rightarrow |E\rangle + i|F\rangle \qquad (7\text{-}130)$$

The overall state then becomes

$$i|D\rangle + |C\rangle \rightarrow i(|F\rangle + i|E\rangle) - (|E\rangle + i|F\rangle) = i|F\rangle - |E\rangle - |E\rangle - i|F\rangle = -2|E\rangle \qquad (7\text{-}131)$$

So when we now extend our experimental setting by two detectors at each of the two paths, a measurement will always result in detecting the photon at $|E\rangle$. A multiplicative factor such as $-2|E\rangle$ still represents the same physical system – only the ration between more than one multiplies would be of interest for quantum mechanical considerations. Back to the surprising result, it can be explained by quantum interference, thus the two probability waves cancel each other out at $|F\rangle$, but constructively interfere at $|E\rangle$ – the behavior is the same as the one explained beforehand with Young's double-slit experiment. Again this experiment has served for proving the particle/wave-dualism. I have been using transmission and reflection in the above experiment without detailed explanations; however, it is important to mention here that a particle like a photon could not only be transmitted through a half-silvered mirror, but theoretically also through a wall of bricks. As the understanding of such behavior is very important knowledge in quantum mechanics, I will go into detail a bit, starting with an example that is very commonly used in elementary courses of quantum physics. In our universe, a particle is allowed to move in three space directions, but due to reasons of simplicity just one space dimension has been used in the following explanations (which can be skipped without remorse, if one is not interested in the mathematics of reflection and transmission on quantum-layer).

Let assume there is a particle with the mass m moving along the direction x of a one-dimensional space. Furthermore, the particle is captured in an infinite rectangular potential $V(x)$ (infinite at the origin point of ordinates as well as at point p), named after its shape. Infinite means that the potential cannot be overcome by the particle. This can be imagined in one direction (from the origin of ordinates along the positive x-axis) as described with Figure 70 - Infinite potential step. To catch a particle in an infinite potential step it is required to keep a potential well, in which the particle (or wave) is able to move, bordered by such steps.

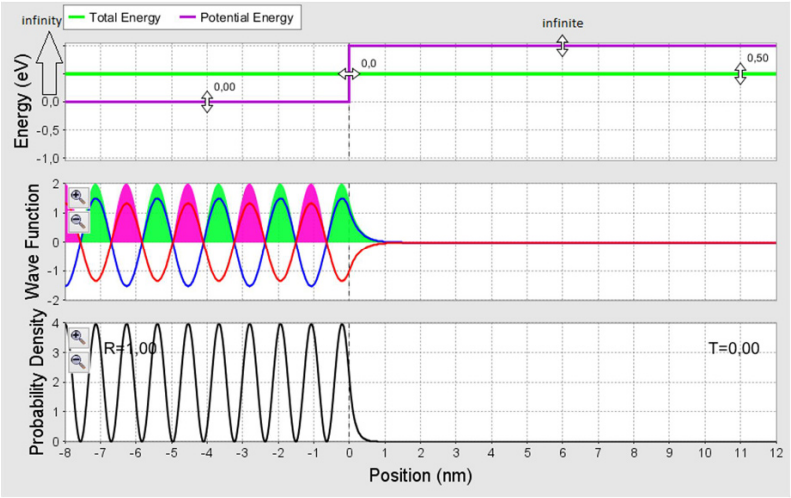

Figure 70 - Infinite potential step[286]

The first thing required to be done is to set up the Schrödinger equation:

$$-\frac{\hbar^2}{2m}\Delta\psi(x) + V(x)\psi(x) = E\psi(x) \qquad (7\text{-}132)$$

In difference to the equation described in chapter 10.5, for our one-dimensional space it is not required to deal with more than one dimensions, thus x in the equation is the only space coordinate. Therefore, also the differential Laplace-operator shrinks to

$$\Delta\psi(x) = \frac{\partial^2}{\partial x^2}\psi(x) \qquad (7\text{-}133)$$

The resulting Schrödinger equation is

$$-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}\psi(x) + V(x)\psi(x) = E\psi(x) \qquad (7\text{-}134)$$

286 Simulation done with "Quantum Tunneling and Wave Packets"-Applet available at http://phet.colorado.edu/en/simulation/quantum-tunneling

For being able to proceed, it is required to consider what it means when the potential steps are infinite. The potential is infinite along the negative x-axis to the origin point of ordinates and infinite from point p along the positive x-axis:

$$V(x) = \infty, if\ x < 0 \vee x > p \tag{7-135}$$

Furthermore, the potential is 0 between the origin point of ordinates and point p:

$$V(x) = 0, if\ 0 \leq x \leq p \tag{7-136}$$

According to this, we can change the required Schrödinger-equation to

$$-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}\psi(x) = E\psi(x) \tag{7-137}$$

If we furthermore consider the wave number to be

$$k = \frac{\sqrt{2mE}}{\hbar} \tag{7-138}$$

the equation can be rewritten as

$$\frac{d^2}{dx^2}\psi(x) + k^2\psi(x) = 0 \tag{7-139}$$

I do not want to go into too much detail with differential equations here, but now we are dealing with a second order differential equation, which features two different solutions with two constants A and B, which have to be determined separately:

$$\psi_1(x) = Asin(kx) \tag{7-140}$$

$$\psi_2(x) = Bcos(kx) \tag{7-141}$$

The general solution to our Schrödinger-equation is then

$$\psi(x) = Asin(kx) + Bcos(kx) \tag{7-142}$$

This is of utmost importance, as this knowledge can be used for determining the energy levels. In the case with infinite potential steps ψ at its potential thresholds must be 0, which then gives the boundary constraints of the problem:

$$\psi(0) = 0 \tag{7-143}$$

$$\psi(p) = 0 \tag{7-144}$$

This can be used for calculating the missing constants A and B: According to eq. (6-141) B must be 0, as

$$\psi(0) = Bcos(kx) = 0 \tag{7-145}$$

$cos(0) = 1$. Thus, if B had another value than 0 eq. (6-145) would not result in 0. Furthermore,

$$\psi(p) = Acos(kp) = 0 \tag{7-146}$$

dictates that

$$kp = n\pi \qquad (7\text{-}147)$$

This can be concluded from that fact that the sinus of any whole-number multiply of π equals 0, so p would then equal π and k equal n. k can be used to express the wave number in another way as given by eq. (6-138), so the following equation is valid:

$$\frac{2mE}{\hbar^2} = \frac{n^2\pi^2}{p^2} \qquad (7\text{-}148)$$

and thus

$$E = \frac{n^2\pi^2\hbar^2}{2mp^2} \qquad (7\text{-}149)$$

Eq. (6-149) provides the allowed energy levels for the particle in question, or in other words describes the quantized states belonging to the principal quantum number n, which describes the electron cloud or the electron state. The larger n is, the lower is the binding energy of the electron and therefore, the larger is the probability that the electron is farther from the atomic nucleus.

Excursus: quantum numbers

Above the energy level of a particle has been mentioned, which represents one of the conserved quantities in the dynamics of a quantum system. Altogether, there exist four such quantities, called quantum numbers, which will admittedly not be explained in detail here, but nevertheless mentioned for staying complete:

Principal quantum number n: describes the shell (energy level) of an atom, thus the height of the orbit. May take the values $n = 1,2,\dots,n$

Azimuthal quantum number l: describes the orbital's type ($s, p, d, f, ?$) and can take the values $l = 1,2,\dots,n-1$

Magnetic quantum number m: describes the orientation of the orbital in z-direction and may take the values $m = -l,\dots,+l$

Spin quantum number s: describes the spin orientation of an electron (as it is a dot this cannot be imagined easily. However, this property has been proofed) along the z-axis. An electron is a fermion (particularly a lepton), as its spin is $\frac{1}{2}$. Thus, s may take $\frac{1}{2}$ and $-\frac{1}{2}$ in this particular case.

As a next step, it is required to normalize the wave function, which means that it is required to ensure that the probabilities for finding a particle in the region between x and dx sum up to 1:

$$\int_0^p |\psi(x)|^2 dx = 1 \qquad (7\text{-}150)$$

As $\psi(x)$ is already known, it is required to insert it (only the part with the constant *A*, as *B* equals 0):

$$|A|^2 \int_0^p sin^2\left(\frac{n\pi x}{p}\right) dx = 1 \qquad (7\text{-}151)$$

$$\int_0^p sin^2\left(\frac{n\pi x}{p}\right) dx = \frac{a}{2} \qquad (7\text{-}152)$$

and thus

$$|A|^2 \left(\frac{a}{2}\right) = 1 \qquad (7\text{-}153)$$

Finally,

$$A = \sqrt{\frac{2}{a}} \qquad (7\text{-}154)$$

The normalized wave function is then

$$\psi(x) = \sqrt{\frac{2}{a}} sin\left(\frac{n\pi x}{p}\right) \qquad (7\text{-}155)$$

With this knowledge it is possible to continue with reflection and transmission that have been discussed in the mirror-experiment beforehand. Let assume the potential step is not infinite, as with the semi-transparent mirrors used in the experiment above (such a mirror can be considered to be a finite potential step), then the particle may have enough energy to overcome such a step. Assuming that the potential is 0 from the negative x-axis to the origin point of ordinates, and is finite (0.5 electron Volt [eV] in Figure 71 - Finite potential step), but with less energy than the particle's total energy (0.82 eV in Figure 71 - Finite potential step).

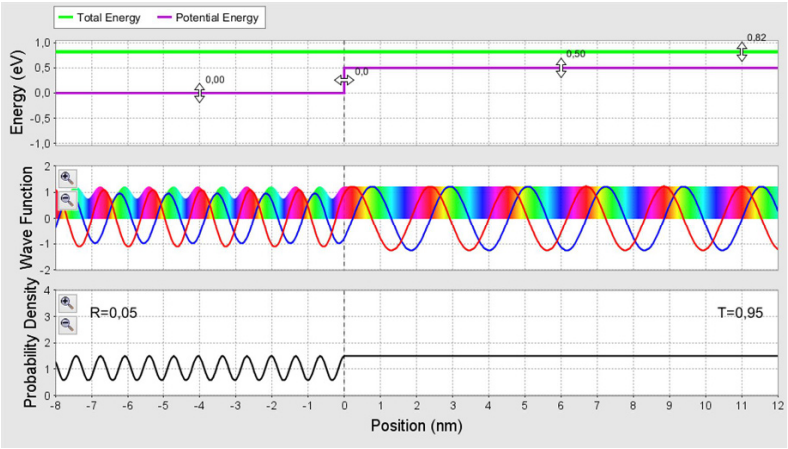

Figure 71 - Finite potential step[287]

287 Simulation done with "Quantum Tunneling and Wave Packets"-Applet available at http://phet.colorado.edu/en/simulation/quantum-tunneling

As the whole situation is dealt with quantum mechanically, again a wave function and not a point particle is required for the explanations. In the area $x < 0$ the situation is as follows:

$$\frac{d^2\psi_1}{dx^2}(x) + k_1^2\psi_1(x) = 0 \qquad (7\text{-}156)$$

where

$$k_1^2 = \frac{2mE}{\hbar^2} \qquad (7\text{-}157)$$

However, in the area $x > 0$ the situation is different, as the potential has to be taken into consideration:

$$\frac{d^2\psi_2}{dx^2}(x) + k_2^2\psi_2(x) = 0 \qquad (7\text{-}158)$$

where

$$k_2^2 = \frac{2m(E-V_0)}{\hbar^2} \qquad (7\text{-}159)$$

Thus k changes as the wave moves from the area with no potential step into the area with a potential step (described in the equation with V_0), as the total energy of the particle is reduced by exactly the energy of the step. According to Figure 71 - Finite potential step, the energy would result in 0.32 eV. Again two differential equations of second order are at hand, but with finite potential. For the situation $x < 0$ the general turns out to be

$$\psi_1(x) = Ae^{ik_1x} + Be^{-ik_1x} \qquad (7\text{-}160)$$

As the particle features more energy than the potential step, the equation for the situation $x > 0$ can be treated the same way as potential-free movement:

$$\psi_2(x) = Ce^{ik_2x} + De^{-ik_2x} \qquad (7\text{-}161)$$

where in both equations e^{ikx} describes waves moving along the positive x-axis, and e^{-ikx} describes waves moving along the negative x-axis. Although the situation has been dealt with as if it was potential-free, this is not the whole story, as the wave may impinge the potential step instead of passing it. When this happens, the wave may either be transmitted or reflected. As the reflection in the example described in Figure 71 - Finite potential step can only redirect the wave along the negative x-axis, D must be 0. Thus, the correct solution for the situation $x > 0$ is

$$\psi_2(x) = Ce^{ik_2x} \qquad (7\text{-}162)$$

Therefore, all possible situations have been covered, namely the initially incoming wave via Ae^{ik_1x}, the reflected wave via Be^{-ik_1x} and the transmitted wave via Ce^{ik_2x}. From the mirror-experiment it became obvious that when measuring the particle's position, only one of these options will remain, so the probabilities for reflection and transmission have to be determined, which is done by calculating the reflection and transmission-coefficients. For this the determination of three variables is required in advance, which are the incoming current density J_i, the reflected current density J_r as well as the transmitted current density J_t. As the situation is still the simplified one with just one dimension, the gradient-operator in the

following equation for calculating the probability current density is represented only by the derivation towards *x*:

$$J_i = \frac{i\hbar}{2m}\left[\psi_e(x)\frac{d\psi_i^*(x)}{dx} - \psi_e^*(x)\frac{d\psi_i(x)}{dx}\right] \quad (7\text{-}163)$$

The probability current density is thus determined by the wave function in position-space, where $\psi_e(x)$ is the incoming wave (along x), and $\psi_e^*(x)$ its complex conjugate. For the described example the incoming wave Ae^{ik_1x} has to be used

$$J_i = \frac{i\hbar}{2m}\left[Ae^{ik_1x}\left(-Aik_1e^{-ik_1x}\right) - Ae^{-ik_1x}Aik_1e^{ik_1x}\right] \quad (7\text{-}164)$$

where $-Aik_1e^{-ik_1x}$ and $Aik_1e^{ik_1x}$ are the wave's complex conjugate respectively the wave itself derived with respect to x, which may be simplified to

$$J_i = \frac{\hbar k_1}{m}|A|^2 \quad (7\text{-}165)$$

The same holds for J_r and J_t:

$$J_r = \frac{i\hbar}{2m}\left[Be^{-ik_1x}Bik_1e^{ik_1x} - Be^{ik_1x}(-Bik_1e^{-ik_1x})\right] \quad (7\text{-}166)$$

$$J_r = \frac{\hbar k_1}{m}|B|^2 \quad (7\text{-}167)$$

$$J_t = \frac{i\hbar}{2m}\left[Ce^{ik_1x}\left(-Cik_1e^{-ik_1x}\right) - Ce^{-ik_1x}Cik_1e^{ik_1x}\right] \quad (7\text{-}168)$$

$$J_t = \frac{\hbar k_1}{m}|C|^2 \quad (7\text{-}169)$$

Thus, the required coefficients are given by

$$R = \frac{J_r}{J_i} = \frac{|B|^2}{|A|^2} \quad (7\text{-}170)$$

and

$$T = \frac{J_t}{J_i} = \frac{|C|^2}{|A|^2} \quad (7\text{-}171)$$

The final thing to do is the determination of the missing constants, which begins with the definition of the boundary conditions. The potential step in the latter example is not infinite, so $\psi(x)$ does not go towards 0. Furthermore, the wave and its derivation towards space are continuous over the potential step:

$$\psi_1(0) = \psi_2(0) \quad (7\text{-}172)$$

$$\frac{d\psi_1}{dx}(0) = \frac{d\psi_2}{dx}(0) \quad (7\text{-}173)$$

If the solutions (eqs. (6-160) and (8-162)) of the differential equation are plugged in, the result is

$$Ae^{ik_1x} + Be^{-ik_1x} = Ce^{ik_2x} \quad (7\text{-}174)$$

which is

$$A + B = C \tag{7-175}$$

and

$$k_1 A - k_1 B = k_2 C \tag{7-176}$$

where

$$B = \frac{k_1 - k_2}{k_1 + k_2} A \tag{7-177}$$

and

$$C = \frac{2k_1}{k_1 + k_2} A \tag{7-178}$$

A cancels out and therefore, the final results for this example are

$$R = \frac{(k_1 - k_2)^2}{(k_1 + k_2)^2} \tag{7-179}$$

and

$$T = \frac{4k_1 k_2}{(k_1 + k_2)^2} \tag{7-180}$$

This result shows some of the magic of quantum mechanics, as if k_1 does not equal k_2 then the particle is reflected, although it features more energy than the potential step (0.5 eV compared to 0.82 eV). This is only possible when the particle is not treated as a particle, but when the particle/wave-dualism is taken into consideration – only a wave features the probability of being reflected from the step in the example. If k_1 equals k_2 then the probability of the wave being reflected is 0, and vice versa. When the wave is reflected at a potential step, such as it happens in Figure 72 - Particle lacks energy, then the probability density is a sum of the incoming wave and its reflected part.

Let assume, the particle lacks enough energy for passing the (finite) potential step, as described in Figure 72 - Particle lacks energy. The step features 1 eV of energy, the particle only 0.5 and quantum physics does not change anything here – the particle will not pass the barrier. However, things get a little different, when a wave collides with a potential barrier, the former featuring less energy than the latter, as described in Figure 73 - Potential barrier.

In Figure 74 - Potential barrier - forces one can see that the potential barrier has its beginning at point A, thus the wave starts encountering a negative force at A. Classical physics dictates that the particle would have to be reflected at point P, as this is where the particle's energy equals the barrier's energy.

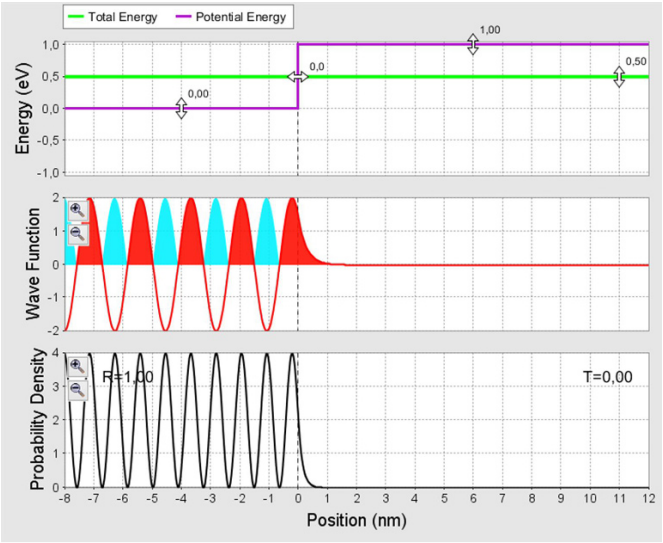

Figure 72 - Particle lacks energy[288]

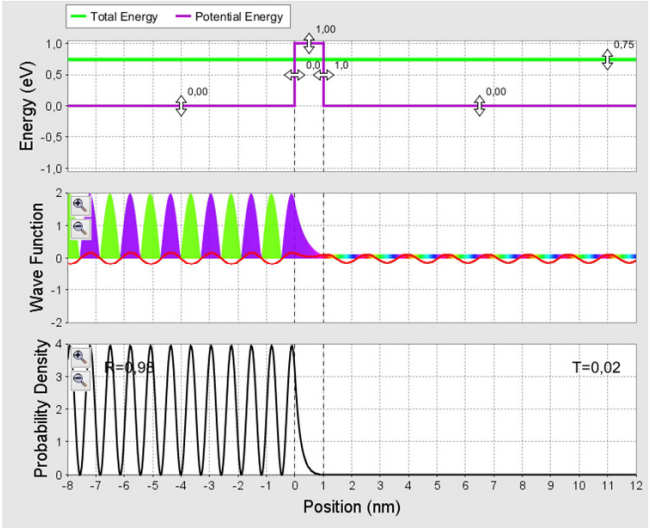

Figure 73 - Potential barrier[289]

288 Simulation done with "Quantum Tunneling and Wave Packets"-Applet available at http://phet.colorado.edu/en/simulation/quantum-tunneling

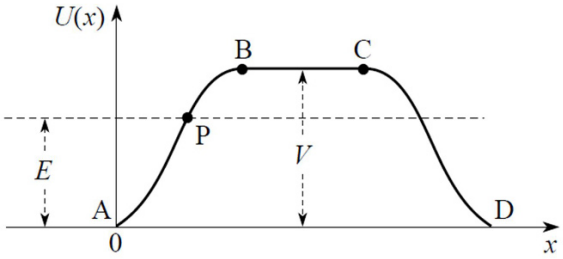

Figure 74 - Potential barrier - forces

However, quantum physics tells us something different, but only when any of the distances of a point to its adjacent point is of less or equal order than the de Broglie wavelength of the particle in question.

Excursus: de Broglie wavelength

According to Louis-Victor de Broglie any particle can be allocated a wavelength – he postulated the concept of the matter wave in his famous Ph.D. thesis in 1924. When talking of a particle without mass, like a photon, then

$$E = \hbar\omega \tag{7-181}$$

where $\hbar$ is the already discussed reduced Planck's constant, v the wave frequency and ω the radial frequency. As

$$\hbar = \frac{h}{2\pi} \tag{7-182}$$

and

$$\omega = 2\pi v \tag{7-183}$$

the energy results in

$$E = hv \tag{7-184}$$

Apart from that, the photon features momentum, although it does not feature mass:

$\vec{p} = \hbar\vec{k}$where $\vec{k}$ is the wave vector, which is perpendicular to the wave front. A wave expanding towards the direction $\vec{k}$ is described by

$$\psi(\vec{r}, t) = Ae^{i(\vec{k}\vec{r}-\omega t)} \tag{7-185}$$

where

289 Simulation done with "Quantum Tunneling and Wave Packets"-Applet available at http://phet.colorado.edu/en/simulation/quantum-tunneling

$$\vec{k} = (k_x, k_y, k_z) \tag{7-186}$$

The circle closes with

$$= \left|\vec{k}\right| = \frac{\omega}{c} = \frac{2\pi}{v} \tag{7-187}$$

Back to the photon momentum it is obvious that

$$p = \frac{hk}{2\pi} \tag{7-188}$$

because of the reduced Planck's constant. Due to k (which in fact is the same k as the one used in the examples, but calculated in a different way) the resulting momentum is

$$p = \frac{h}{\lambda} \tag{7-189}$$

where λ is the wavelength. De Broglie generalized this equation for all particles in the form of

$$\lambda = \frac{h}{p} \tag{7-190}$$

where p is the relativistic momentum of a particle with the rest mass m:

$$p = \frac{mv}{\sqrt{1-\left(\frac{v}{c}\right)^2}} \tag{7-191}$$

Assuming that AB and BC in Figure 74 - Potential barrier - forces are negligible, the schematic description of the potential barrier described in Figure 73 - Potential barrier is of sufficient accuracy. The distance d from the origin point of ordinates towards p describes the width of the potential barrier, which in most physical situations is larger than the de Broglie wavelength. The illustrations show that the particle features 0.75 eV of energy and the potential barrier 1 eV, however, the probability density shows that there exists a probability of 0.02 for the particle to overcome the barrier, which is called quantum tunneling. This either means that a particle is temporarily supplied with additional energy so it is able to bypass the barrier, or the wave extension is itself allows this. Taking only the x-dimension into consideration, the Schrödinger equation for $x < -p$ (assuming that the potential barrier originates at -p) again takes the form

$$\frac{d^2\psi_1}{dx^2}(x) + +k_1^2\psi_1(x) = 0 \tag{7-192}$$

where

$$k_1^2 = \frac{2mE}{\hbar^2} \tag{7-193}$$

When considering the potential barrier to end at point p then the equation for the area $x \leq 0 \leq p$ is

$$\frac{d^2\psi_2}{dx^2}(x) + k_2^2\psi_2(x) = 0 \tag{7-194}$$

where

$$k_2^2 = \frac{2m(E-V_0)}{\hbar^2} \tag{7-195}$$

Nothing new by now… however, when having a look at Figure 73 - Potential barrier it becomes obvious that $E - V_0$ is negative and would thus make k imaginary, which is physically not allowed. Thus, the sign of the equation is switched:

$$\frac{d^2\psi_2}{dx^2}(x) - k_2^2\psi_2(x) = 0 \tag{7-196}$$

and

$$k_2^2 = \frac{2m(V_0-E)}{\hbar^2} \tag{7-197}$$

In the area $x > p$ the equation becomes

$$\frac{d^2\psi_3}{dx^2}(x) + k_1^2\psi_3(x) = 0 \tag{7-198}$$

Thus, there are three solutions, one for each ψ_n, starting with the solution for the area $x < -p$, which is again

$$\psi_1(x) = Ae^{ik_1x} + Be^{-ik_1x} \tag{7-199}$$

the one for $-p \leq x \leq p$ is

$$\psi_2(x) = Ce^{k_2x} + De^{-k_2x} \tag{7-200}$$

without imaginary part, and the one for $x > p$ is

$$\psi_3(x) = Ee^{ik_1x} + Fe^{-ik_1x} \tag{7-201}$$

However, in the area $x > p$ there is no wave travelling to the left, thus

$$\psi_3(x) = Ee^{ik_1x} \tag{7-202}$$

as F equals 0. A situation with 5 arbitrary complex constants seems to be unsolvable at the beginning, however, by matching the wave functions and their respective derivatives at the boundaries. From the latter example the reflection and transmission coefficients are known, which are

$$R = \frac{J_r}{J_i} = \frac{|B|^2}{|A|^2} \tag{7-203}$$

and

$$T = \frac{J_t}{J_i} = \frac{|E|^2}{|A|^2} \tag{7-204}$$

Again, the wave functions and its derivations towards space are continuous over the potential barrier:

$$\psi_1(0) = \psi_2(0) \tag{7-205}$$

$$\frac{d\psi_1}{dx}(0) = \frac{d\psi_2}{dx}(0) \tag{7-206}$$

$$\psi_2(p) = \psi_3(p) \tag{7-207}$$

$$\frac{d\psi_2}{dx}(p) = \frac{d\psi_3}{dx}(p) \tag{7-208}$$

All in all there are three regions, thus two boundaries:

$$Ae^{-ik_1p} + Be^{ik_1p} = Ce^{-k_2p} + De^{k_2p} \tag{7-209}$$

for $x = -p$ and

$$Ce^{k_2p} + De^{-k_2p} = Ee^{ik_1p} \tag{7-210}$$

for $x = p$

As a next step, the derivatives of the wave function have to be set up (all solutions together make up the wave function):

$$\psi(x) = \begin{cases} Ae^{ik_1x} + Be^{-ik_1x} \\ Ce^{k_2x} + De^{-k_2x} \\ Ee^{ik_1x} \end{cases} \tag{7-211}$$

$$\psi'(x) = \begin{cases} Aik_1e^{ik_1x} - Bik_1e^{-ik_1x} \\ Ck_2e^{k_2x} - Dk_2e^{-k_2x} \\ Eik_1e^{ik_1x} \end{cases} \tag{7-212}$$

The boundary condition for the derivative of $\psi(x)$ at $x = -p$

$$Aik_1e^{-ik_1p} - Bik_1e^{ik_1p} = Ck_2e^{-k_2p} - Dk_2e^{k_2p} \tag{7-213}$$

and at $x = p$

$$Ck_2e^{k_2p} - Dk_2e^{-k_2p} = Eik_1e^{ik_1p} \tag{7-214}$$

is continuity. Summing up, there are two equations for $x = -p$ and two for $x = p$, still containing five unknown variables. Multiplying the first of the four equations by ik_1, solving it for B

$$Bik_1e^{ik_1p} = Cik_1e^{-k_2p} + Dik_1e^{k_2p} - Aik_1e^{-ik_1p} \tag{7-215}$$

and substituting it into the third equation

$$Aik_1e^{-ik_1p} - Cik_1e^{-k_2p} - Dik_1e^{k_2p} + Aik_1e^{-ik_1p} = Ck_2e^{-k_2p} - Dk_2e^{k_2p} \tag{7-216}$$

results in

$$2Aik_1e^{-ik_1p} = Ck_2e^{-k_2p} + Cik_1e^{-k_2p} - Dk_2e^{k_2p} + Dik_1e^{k_2p} \tag{7-217}$$

$$2Aik_1e^{-ik_1p} = C(ik_1 + k_2)e^{-k_2p} + D(ik_1 + k_2)e^{k_2p} \tag{7-218}$$

$$2Ak_1e^{-ik_1p} = \left(1 - \frac{ik_2}{k_1}\right)Ce^{-k_2p} + \left(1 - \frac{ik_2}{k_1}\right)De^{k_2p} \tag{7-219}$$

After that, the second equation has to be multiplied by k_2, solved for D

$$Dk_2e^{-k_2p} = Ek_2e^{ik_1p} - Ck_2e^{k_2p} \tag{7-220}$$

and substituted into the fourth equation

$$Ck_2e^{k_2p} - Ek_1e^{ik_1p} + Ck_2e^{k_2p} = Eik_1e^{ik_1p} \tag{7-221}$$

which results in

$$2Ck_2e^{k_2p} = Ek_2e^{ik_1p} + Eik_1e^{ik_1p} \tag{7-222}$$

$$2Ck_2e^{k_2p} = E(k_2 + ik_1)e^{ik_1p} \tag{7-223}$$

$$Ck_2e^{k_2p} = \frac{1}{2}\left(1 + \frac{ik_1}{k_2}\right)Ee^{ik_1p} \tag{7-224}$$

As a next step, the second equation has to be multiplied by k_2, solved for C

$$Ck_2e^{k_2p} = Ek_2e^{ik_1p} - Dk_2e^{-k_2p} \tag{7-225}$$

and substituted into the fourth equation

$$Ek_2e^{ik_1p} - Dk_2e^{-k_2p} - Dk_2e^{-k_2p} = Eik_1e^{ik_1p} \tag{7-226}$$

which results in

$$2Dk_2e^{-k_2p} = Eik_1e^{ik_1p} + Ek_2e^{ik_1p} \tag{7-227}$$

$$2Dk_2e^{-k_2p} = E(ik_1 + k_2)e^{ik_1p} \tag{7-228}$$

$$De^{-k_2p} = \frac{1}{2}\left(1 - \frac{ik_1}{k_2}\right)Ee^{ik_1p} \tag{7-229}$$

Some of the exponential signs in the resulting equations do not match, thus

$$Ck_2e^{k_2p} = \frac{1}{2}\left(1 + \frac{ik_1}{k_2}\right)Ee^{ik_1p} \tag{7-230}$$

is adapted to

$$Ck_2e^{-k_2p} = \frac{1}{2}\left(1 + \frac{ik_1}{k_2}\right)Ee^{ik_1p}e^{-2k_2p} \tag{7-231}$$

and

$$De^{-k_2p} = \frac{1}{2}\left(1 - \frac{ik_1}{k_2}\right)Ee^{ik_1p} \tag{7-232}$$

to

$$De^{k_2p} = \frac{1}{2}\left(1 - \frac{ik_1}{k_2}\right)Ee^{ik_1p}e^{2k_2p} \tag{7-233}$$

This can now be substituted into

$$2Ak_1e^{-ik_1p} = \left(1 - \frac{ik_2}{k_1}\right)Ce^{-k_2p} + \left(1 - \frac{ik_2}{k_1}\right)De^{k_2p} \tag{7-234}$$

which results in

$$2Ak_1e^{-ik_1p} = \left(1-\frac{ik_2}{k_1}\right)\frac{1}{2}\left(1+\frac{ik_1}{k_2}\right)Ee^{ik_1p}e^{-2k_2p} + \left(1-\frac{ik_2}{k_1}\right)\frac{1}{2}\left(1-\frac{ik_1}{k_2}\right)Ee^{ik_1p}e^{2k_2p} \tag{7-235}$$

$$2Ak_1e^{-ik_1p} = \frac{Ee^{ik_1p}}{2}\left(1-\frac{ik_2}{k_1}+\frac{ik_1}{k_2}+\frac{k_1k_2}{k_1k_2}\right)e^{-2k_2p} + \frac{Ee^{ik_1p}}{2}\left(1+\frac{ik_2}{k_1}-\frac{ik_1}{k_2}+\frac{k_1k_2}{k_1k_2}\right)e^{2k_2p} \tag{7-236}$$

$$2Ae^{-ik_1p} = \frac{Ee^{ik_1p}}{2}\left(2e^{-2k_2p}-\frac{ik_2}{k_1}e^{-2k_2p}+\frac{ik_1}{k_2}e^{-2k_2p}+2e^{2k_2p}+\frac{ik_2}{k_1}e^{2k_2p} - \frac{ik_1}{k_2}e^{2k_2p}\right) \tag{7-237}$$

$$2Ae^{-ik_1p} = \frac{Ee^{ik_1p}}{2}\left[2(e^{-2k_2p}+e^{2k_2p})-\frac{ik_2^2}{k_1k_2}e^{-2k_2p}+\frac{ik_1^2}{k_1k_2}e^{-2k_2p}+\frac{ik_2^2}{k_1k_2}e^{2k_2p}- \frac{ik_1^2}{k_1k_2}e^{2k_2p}\right] \tag{7-238}$$

$$2Ae^{-ik_1p} = \frac{Ee^{ik_1p}}{2}\left[2(e^{-2k_2p}+e^{2k_2p})+\frac{ik_2^2}{k_1k_2}(e^{2k_2p}-e^{-2k_2p}) - \frac{ik_1^2}{k_1k_2}(e^{2k_2p}-e^{-2k_2p})\right] \tag{7-239}$$

$$2Ae^{-ik_1p} = Ee^{ik_1p}2\left(\frac{e^{2k_2p}+e^{-2k_2p}}{2}\right)+\frac{i\left(k_2^2-k_1^2\right)}{k_1k_2}\left(\frac{e^{2k_2p}-e^{-2k_2p}}{2}\right) \tag{7-240}$$

As

$$sinh(x) = \frac{1}{2}(e^x-e^{-x}) = -isin(ix) \tag{7-241}$$

and

$$cosh(x) = \frac{1}{2}(e^x+e^{-x}) = cos(ix) \tag{7-242}$$

the result of this delicate piece of algebra is

$$Ae^{-ik_1p} = Ee^{ik_1p}\left[cosh(2k_2p)+\frac{i\left(k_2^2-k_1^2\right)}{2k_1k_2}sinh(2k_2p)\right] \tag{7-243}$$

The transmission coefficient can be expressed as the ratio of the transmitted current density J_t and the incoming current density J_i. By knowing that the transmitted current density is $|E|^2$ and the incoming current density is $|A|^2$, followed by a closer look at the last equation, it is obvious that everything needed to do the required calculations for the reciprocal of the transmission coefficient is already available, as the equation can be re-expressed as

$$\left|\frac{Ae^{-ik_1p}}{Ee^{ik_1p}}\right| = \left|\left[cosh(2k_2p) + \frac{i(k_2^2-k_1^2)}{2k_1k_2}sinh(2k_2p)\right]\right| \tag{7-244}$$

Furthermore, by knowing that the product of the complex conjugates represents the sum of the squares of the real part and the coefficient of the imaginary part, the equation's left part can be simplified as follows:

$$\left|\frac{Ae^{-ik_1p}}{Ee^{ik_1p}}\right| = \sqrt{\left[\frac{Ae^{-ik_1p}}{Ee^{ik_1p}}\frac{A^*e^{ik_1p}}{E^*e^{-ik_1p}}\right]} \tag{7-245}$$

$$\left|\frac{Ae^{-ik_1p}}{Ee^{ik_1p}}\right| = \sqrt{\left[\frac{A}{E}\frac{A^*}{E^*}\right]} \tag{7-246}$$

$$\left|\frac{Ae^{-ik_1p}}{Ee^{ik_1p}}\right| = \frac{\sqrt{[AA^*]}}{\sqrt{[EE^*]}} \tag{7-247}$$

$$\left|\frac{Ae^{-ik_1p}}{Ee^{ik_1p}}\right| = \frac{|A|^2}{|E|^2} \tag{7-248}$$

With this, the ratio can be expressed as

$$\frac{|A|^2}{|E|^2} = \left|\left[cosh(2k_2p) + \frac{i(k_2^2-k_1^2)}{2k_1k_2}sinh(2k_2p)\right]\right|^2 \tag{7-249}$$

$$\frac{|A|^2}{|E|^2} = \left|\left[cosh(2k_2p) + \frac{i(k_2^2-k_1^2)}{2k_1k_2}sinh(2k_2p)\right]\right|^2 \tag{7-250}$$

$$\frac{|A|^2}{|E|^2} = cosh^2(2k_2p) + \left(\frac{k_2^2-k_1^2}{2k_1k_2}\right)^2 sinh^2(2k_2p) \tag{7-251}$$

$$\frac{|A|^2}{|E|^2} = 1 + sinh^2(2k_2p) + \left(\frac{k_2^2-k_1^2}{2k_1k_2}\right)^2 sinh^2(2k_2p) \tag{7-252}$$

$$\frac{|A|^{2^2}}{|E|^2} = 1 + \left(1 + \frac{k_2^4-2k_1^2k_2^2+k_1^4}{4k_1^2k_2^2}\right) sinh^2(2k_2p) \tag{7-253}$$

$$\frac{|A|^2}{|E|^2} = 1 + \left(\frac{4k_1^2k_2^2+k_2^4-2k_1^2k_2^2+k_1^4}{4k_1^2k_2^2}\right) sinh^2(2k_2p) \tag{7-254}$$

$$\frac{|A|^2}{|E|^2} = 1 + \left(\frac{k_2^4+2k_1^2k_2^2+k_1^4}{4k_1^2k_2^2}\right) sinh^2(2k_2p) \tag{7-255}$$

$$\frac{|A|^2}{|E|^2} = 1 + \frac{\left(k_2^2+k_1^2\right)^2}{4k_1^2k_2^2} sinh^2(2k_2p) \tag{7-256}$$

The substitution of the wave numbers

$$k_1^2 = \frac{2mE}{\hbar^2} \tag{7-257}$$

or

$$k_1 = \frac{2mE}{\hbar} \tag{7-258}$$

and

$$k_2^2 = \frac{2m(V_0-E)}{\hbar^2} \tag{7-259}$$

or

$$k_2 = \frac{\sqrt{2m(V_0-E)}}{\hbar} \tag{7-260}$$

finalizes the solution for the transmission coefficient:

$$\frac{|A|^2}{|E|^2} = 1 + \frac{\left(\frac{2m}{\hbar^2}(V_0-E)+\frac{2mE}{\hbar^2}\right)^2}{4\left(\frac{2mE}{\hbar^2}\right)\left(\frac{2m}{\hbar^2}(V_0-E)\right)} sinh^2\left(\frac{2p}{\hbar}\sqrt{2m(V_0-E)}\right) \tag{7-261}$$

$$\frac{|A|^2}{|E|^2} = 1 + \frac{\frac{4m^2}{\hbar^4}(V_0-E+E)^2}{\frac{4m^2}{\hbar^4}4E(V_0-E)^2} sinh^2\left(\frac{2p}{\hbar}\sqrt{2m(V_0-E)}\right) \tag{7-262}$$

$$\frac{|A|^2}{|E|^2} = 1 + \frac{{V_0}^2}{4E(V_0-E)} sinh^2\left(\frac{2p}{\hbar}\sqrt{2m(V_0-E)}\right) = T^{-1} \tag{7-263}$$

By having derived the transmission coefficient, which does obviously not equal 0, an interesting feature of quantum physics has been revealed, beforehand mentioned as quantum tunnelling. Classically, a particle would be reflected at any case, however, as the above calculations show, quantum physics states that even in a case a particle features less energy than a potential barrier it hits, a finite probability the particle passing the exists. This is especially important for later explanations based on the research of Stuart Hameroff[290] and Roger Penrose.[291]

7.4 Summary

Although there is a lot more to say and explain about quantum physics, from the provided knowledge one gets compelled to search for the occurrence of quantum effects in the human brain. Research has shown that there is a lot more than classical physics, and as our knowledge about quantum physics advances, we may not only find new fields of application, like the creation of quantum computers and the implementation of paradigms from artificial intelligence on such (see the next chapter). We may find out that quantum physics also plays a significant role in our macroscopic world as some theories propose that quantum effects also occur in our brain. It is not that we need to be aware of all these complex algebra and trigonometry elucidated in this chapter – they may just happen, but the only (currently known) way

290 Quantumconsciousness: Quantum computation in brain microtubules? The Penrose-Hameroff "Orch OR" [2013-07-28]; URL: http://www.quantumconsciousness.org/penrose-hameroff/quantumcomputation.html

291 Roger Penrose (1994): Shadows of the mind – a search for the missing science of consciousness; Oxford: Oxford University Press

for describing all the quantum effects discussed is by the use of mathematics, which will also be the way for reproducing human brain functionality artificially. However, before moving back to the human brain we will have a closer look at artificial neural structures that may benefit from the implementation on a quantum computer.

8 Quantum physics and the biological brain

Why may one suggest that quantum physical properties do have an influence on how information in the human brain is actually processed? From the first chapter it should be clear that the soma of a biological neuron features numerous branched dendrites, and at least one axon extending to other neurons, each of them featuring branches as well. We recall that neurons have a negative resting potential, at about –70 mV. If a specific neuron, which we call post-synaptic neuron by now, receives signals from other neurons, which we call pre-synaptic neurons here, then its potential may be given a rise. Each of the neurons features a threshold at around –55 mV, which may be exceeded if the neuron's potential has been given rise by several pre-synaptic signals. However, this does not necessarily happen, as the synapse transferring the signal may not only be excitatory, thus send a positive signal consequently leading to a rise of the post-synaptic neuron's potential, but also inhibitory, meaning that it decreases the post-synaptic neuron's potential and consequently also decreases this specific neuron's probability for firing. In chapters 3 and 4 we were dealing with simple simulations of brain activities by the application of artificial neural networks to get a basic understanding of how such structures work and what they may be used for. In chapter 5 we were applying our knowledge from chapter 2 for creating some more complex artificial neural network structures, and in chapter 5 we got to know some other nature-inspired approaches for artificial neural network learning. In chapter 10.5 Quantum physics and the artificial brain I am extending the possibilities towards processing such an artificial neural network on a quantum computer, but the purpose for the proposition of the latter one has not only been the possibility for processing problem statements like the ones explained in chapter 5. In my point of view, in former times there were two possibilities for trying to grasp the complex workings for information processing the human brain is capable of:

– The neurologist's approach was to gather very detailed knowledge about the features of the human brain, like how information may be processed, or which parts are responsible for specific functions.
– The computer scientist's approach was at first to understand the basic functions of the human brain and to try imitating those functions by means of computer science, like artificial neural networks.

However, none of these approaches alone remains valid, as the former one very often takes into consideration only Newtonian physics, thus only processing of information on a macroscopically visible scale, where the latter one may indeed consider information processing on a quantum physical scale, e.g. in a quantum artificial neural network, but without taking into consideration the role quantum mechanics may actually play in information processing in the human brain. A complete understanding must incorporate a mixture of three areas, thus rely on the neurologist's knowledge of the of the brain's inner workings, the computer scientist's knowledge on digital information processing and quantum information theory as well as the physicist's knowledge of quantum mechanics and quantum field theory in general. As this approach to the very complex topic has proven to be useful, I will not any longer distinguish between and move freely within these fields, and solely talk of neuronengineering, which has also been used by Katz[292] in this context and which I consider to be the field of research

292 Katz Bruce F. (2011): Neuroengineering the future - Virtual minds and the creation of immortality; Massachusetts: Infinity Science Press LLC

F. Neukart, *Reverse Engineering the Mind*, AutoUni – Schriftenreihe 94, DOI 10.1007/978-3-658-16176-7_8

incorporating all other fields required for understanding the functioning of the human brain as well as re-engineering its capabilities.

In my point of view it is not reprehensible to research for possible quantum effects in the human brain, although at a first glance it might seem impossible that such may or even can occur. Analogies can already be found, be it the massive parallel processing of information, the brain's nondeterministic processes or the reduction of information to single events, which may be interpreted from conscious experiences. Quantum mechanics and the beforehand discussed phenomena like entanglement, deterministic evolution, but non-deterministic state-vector reduction, or interference seems to be a promising candidate for the explanation of neural information processing. However, I am not the first researcher being interested in the area, as even Niels Bohr, one of the first physicists having dealt with quantum mechanics, has allegedly been of the opinion that biological processes may incorporate quantum physical effects.

1967 the Umezawa and Ricciardi have proposed a hypothesis on how quantum field theory may find application within the human brain. They proposed that spontaneous violations of the symmetry can result in quantum states within the brain, which are called Goldstone-Bosons. Such Goldstone-Bosons are long-range correlated waves, and the violation of the symmetry has been interpreted as natural measurement procedure. One year after that, Werner Fröhlich has detected coherent dipole-waves one layer below the cell membrane. He recognized that the orientation of these dipoles represents a general control- and order-parameters in biological processes. Furthermore, in his hypothesis he explains that quantum coherence may occur below the cell membrane for several centimetres, the so-called dipole-line-up. However, this has not been proved and the hypothesis has been checked by several researchers which could not find any evidence for long-term coherences to occur.[293]

8.1 Difficulties with U in the macroscopic world

According to our understanding, there exists one reason why quantum linear superposition cannot be held in the macroscopic world – it is not possible to isolate a macroscopic quantum system from its own irrelevant properties, such as gravitation might be one. Currently there does not exist a theory of all, which unifies the theory of general relativity with the quantum theory. The theory of general relativity covers gravity, whereas the quantum theory covers all other physical elementary forces, which are electromagnetic, weak and strong interaction. The difference between these theories is, roughly said, that the former one describes the setup of the universe, and the latter one the interaction between the smallest particles in small spaces. Furthermore, gravitation is the only of the four elementary forces that has no 'interaction' in its name, as according to our current knowledge it is the only force that acts exceptionally attractive, which is due to only one gravitational charge, the mass (it cannot be cancelled out by an opposing force). Weak and strong interaction only play a role at microscopic level, whereas the electromagnetic interaction is also important on macroscopic level. As of today's knowledge, both theories do not overlap except in very extreme cases, like in black holes or the big bang, where the curvature of space time goes to infinity. Anyway, the difficulties for formulating a theory of everything are manifold, e.g. the problem that gravitation cannot be

293 Held Werner: Quantenphysikalische Ansätze des Bewusstseins [2013-08-09]; URL: http://www.datadiwan.de/netzwerk/index.htm?/experten/he_001d_.htm

split up in quanta or 'elementary portions', which is required for being able to successfully apply calculations on quantum layer. Unfortunately, we cannot discuss some of current time's most interesting theories like the string theory or the loop-quantum gravitation here.

Penrose[294] well described that we have to bear in mind that although the theories have not yet been brought together, not only in the most extreme physical situations gravitation may play a role in quantum physics. As the Planck length of $1.616199(97) * 10^{-35}$ m is the length that counts for effects of quantum gravity and one may not easily figure out how effects on this scale may influence the macroscopic world, he used a very figurative example to describe such a situation. Summing up, he describes a photon impinging a half-silvered mirror, as explained the sample experimental situations (see 7.3 The state vector reduction R). After the photon has been split up into two parts, one part meets a detection device that is treated fully quantum mechanically and moves a spherical lump attached to it. So the lump may feature two different positions, either the moved one or the initial one. Thus, the linear superposition also features both and not only that, it also involves the lump's gravitational field. Penrose points out that this is, according to general relativity, a superposition of two different space-time geometries and asks the very interesting question, at which point those differ enough from each other so that decoherence occurs. The answer is that the difference must be something larger than or equal to the Planck scale. For a superposed state consisting of two spatially displaced states he asks for the energy that it would for the displacement to happen, considering exceptionally the gravitational force between them. According to this proposal and in absolute units, the gravitational energy that is required for displacing the states is

$$E_d = \frac{m^2}{a} \qquad (8\text{-}1)$$

and the time it takes until decoherence happens is the reciprocal of E_d

$$T_r = \frac{a}{m^2} \qquad (8\text{-}2)$$

where a is the radius of the superposed objects and m its mass. Summing up, the criterion for measurement Penrose proposed is that if there is sufficient disturbance in an environment, R will take place rapidly and will also include any physical system that is entangled with it, like the measuring apparatus mentioned beforehand. This can be easily imagined by replacing the spherical lump with a glass of water that just absorbs the part of the photon that has passed the mirror – macroscopically we cannot observe any physical difference, however, micro-scopically, the arrangement of particles within the glass of water has changed and thus interaction or measurement has happened. We will go into detail with this by discussing the Hameroff-Penrose model of orchestrated objective reduction followed by my own proposal.

294 Penrose Roger (1994): Shadows of the mind – a search for the missing science of consciousness; Oxford: Oxford University Press, p. 335 ff.

8.2 The Hameroff-Penrose model of orchestrated objective reduction

A lot of scientists from different fields have been discussing the model of orchestrated objective reduction that has been proposed by Roger Penrose and Stuart Hameroff[295,296] in the 90ies. The model will be described completely, but not in as much detail as done by Roger Penrose and Stuart Hameroff in their work, taking into account my own comments and some of Werner Held.[297] Furthermore, I do not comment on the 'feasibility' of the theory, as for me this is of no concern here – the focus is on aspects that may be used for the implementation of an artificial conscious entity, even if there currently exists no proof for the relevance of quantum effects for the emergence of conscious experiences.

8.2.1 The idea

On a first instance it may not be obvious why a theory of particle physics can be used for the explanation of biological or mental phenomena, as conscious experiences are. However, on a second glance one will recognize that the conscious thoughts and brain activity in general feature a certain similarity to quantum physics, compared to classical physics, which is due to the massively distributed and parallel processing of in our brains, the uniformity and globality of a from many individual aspects assembled perception, the simultaneous irreducibility of this emergent phenomenon to individual components as well as the increasingly visible non-determinism biological and mental action require almost necessarily a conceptual framework capable of processing such non-local, non-reductionistic and indeterminate processes. Quantum theory is capable of exactly that,

- as it is home to non-local phenomena such as the Einstein-Podolski-Rosen paradox (as discussed beforehand in the half-silvered mirror experiments),
- as it features discontinuous quantum jumps,
- and as it is a probability theory, in which a single event cannot be physically determined.

The idea that quantum physical processes may impact the nervous activity or consciousness has already been expressed sporadically in the early days of the debate on quantum physics, however, the dissenting votes predominated. Niels Bohr, the founder of quantum mechanics, has been known for several significant cancellations to the belief that quantum theory could be suitable for the interpretation of biological processes, whereby his objection was the seeming impossibility of direct measurement, as to prepare biological organism for a measurement would mean to kill it. Quantum theory would be suitable only for inanimate matter, since the condition of the isolation of quantum physical systems must be given by their environment, which is not the case in biological systems, as such are dissipative structures and in permanent exchange of energy with their environment. Another issue in the debate at that time were the different orders of magnitude, as in that time quantum effects at

295 Penrose Roger (1994): Shadows of the mind – a search for the missing science of consciousness; Oxford: Oxford University Press

296 Hameroff Stuart: Quantum consciousness [2013-07-28]; URL: http://www.quantumconsciousness.org

297 Held Werner: Quantenphysikalische Ansätze des Bewusstseins [2013-06-22]; URL: http://www.werner-held.de/pdf/qc.pdf

that time played only a role in microscopic orders of magnitude. No measurement, not even to inorganic systems, could expand on macroscopic areas, because even the cosmic background radiation of $3K$ already leaves behind $2.3 * 10^{13}$ photons per second on cm^3 tungsten, according to Wigner's calculations, which is an influence large enough to make any accurate measurement impossible. Objections along these lines could rest the efforts for several decades, however, meanwhile it is possible to detect the same quantum phenomena in all atoms, not just electrons or photons – the next step will be to detect these at molecules.

8.2.2 Microtubules

Stuart Hameroff presented the idea that microtubules as holographic bio-computer could be home to quantum effects in the sense that the microtubules could be the place of action of the Fröhlich-waves. Changes in dipole density in the environment of the cell cause changes in the quantum states of the microtubules, e.g. the coherent Goldstone / Fröhlich- waves. Microtubules are tiny protein tube occurring in all cells, with an approximately $2.5 * 10^{-9}$ m diameter on the outside and $1.4 * 10^{-9}$ m diameter on the inside as well as highly variable length from a few nanometers to several millimetres in the axons of neurons, which are composed of 13 parallel or spirally arranged strands of the protein tubulin. The dipolar tubulin is a dimer, consists of an alpha and a beta monomer and can occur in two contiguous formations. Except for the control of cell division, these extremely plastic microtubules also play a decisive role in the organization of the cytoskeleton, the protein, mass, plasma and for transport within the cell, at communication with neighboring cells, and at the shaping of the cell and the movement. They are arranged in parallel and connected to other nerve cells and other microtubule structures by so-called cell microtubule associated proteins (MAPs). Some experiments conducted show that microbotules may play a significant role in conscious activities:

- Hameroff researched on single-celled organisms such as the Paramecium, which do not feature neurons or synapses, but microtubules. However, they do feature primitive sensory abilities and learning performance in certain labyrinths.
- Cronley-Dillon showed that rats that were raised in a barren environment feature a significantly lower density of microtubules than those who were kept in a stimulating environment.
- At the same time, the consumption of drugs that prevent the construction of microtubules, result in reduced memory performance.

Then it was primarily Roger Penrose[298,299] who triggered off a broad discussion of the emergence of quantum physical phenomena in the brain.

According to Hameroff and Penrose, quantum physical processes take place in the microtubules and other structures of the neurons in the brain, and the building block of microtubuli, the protein tubulin dimer, may exist in two different states, which is due to different electron localization associated with different mass distributions. These two states exist in a quantum mechanical superposition of entanglement before collapsing into a final state. From our discussion about entangled quantum bits we remember, that unlike bits in a von Neumann

298 Penrose Roger (1991): The Emperor's New Mind Concerning Computers, Minds, and the Laws of Physics; Oxford: Oxford University Press

299 Penrose Roger (1994): Shadows of the mind – a search for the missing science of consciousness; Oxford: Oxford University Press

machine, whose general state can only be one of the 2^n products of $|0\rangle$ and $|1\rangle$, a general state of n Qbits is a superposition of these 2^n product states and cannot, in general, be expressed as a product of any set of 1-Qbit states. Individual Qbits that make up a multi-Qbit system cannot always be characterized as having individual states of their own - in contrast to the bits in a von Neumann machine – as they may be in nonproduct-states called entangled states (please see the description at 7.1.4.3 Quantum entanglement).[300] Back to ORCH OR, quantum coherence or quantum linear superposition simply put means that different parts become a single state, which is described by the Schrödinger wave equation (see 7.1.2.4 Quantum dynamics) and should now be able to extend to larger areas, presumably driven by thermal and biochemical energy, such as described by Fröhlich. Penrose's idea is that the two different potential states of the tubulin receive different mass distributions with increasing coherence time, and therefore the result is a divergence of the space-time geometry of the two possibilities, until a threshold in quantum gravity is exceeded and a self-organized collapse of the wave function, thus the end of the entangled state in favour of a real state of the tubulin dimers, occurs. The reduction of the wave packet within the microtubules is the 'Orchestrated Objective Reduction' (ORCH OR), because in this type of natural measurement process the MAPs, genetic and other changes of tubulin proteins determine this process in an unspecified, not in detail described, but not causal manner. The usual Copenhagen interpretation of quantum physics, which makes no statements about quantum phenomena in living systems, on the other hand is based on the random reduction R of the wave packet. The consciousness interpretation due to Wigner, however, assumes a subjective reduction of SR obtained by consciousness. Penrose's idea of the OR represents some sort of natural measurement process by the action of the hypothetical quantum gravity. He is of the opinion that an objective criterion for a collapse in the sense of quantum gravity threshold is present. With Penrose's increasing devotion to a living organism, he adds a required extension with the self-organized ORCH OR. However, how to bring together the non-objective, but condition- and experience-dependent processes in organisms with an objective reduction remains still to be clarified within this model. Although he recognizes, due to his previous thoughts on Gödel's incompleteness theorem and its consequent rejection of the possibility in principle of artificial intelligence, the OR as non-algorithmizable and thus not computable, this still does not solve the fundamental contradiction between his objectively formulated concept and the probability propositions of quantum physics and self-organization.

Penrose takes over the assumption expressed by the neurophysiologist Benjamin Libet in his time-on theory that only via the duration of neural activity can be determined what becomes aware in one's mind. Libet determined that about one-third to one-half seconds of lead time of unconscious neural activity are necessary (in terms of a readiness potential postulated by Kornhuber and Deecke in the sixties, thus a slowly rising negative voltage shift that points unerringly to the same onset activity). Penrose then puts the coherence length of the microtubules on a level with the second half of unconscious neural activity and thus tries to equal the transition from unconscious processes to conscious processes with the processes in the microtubules. The various formations of microtubules components (the protein tubulin dimers) for Penrose form the threshold for the transition of quantum physics to the classical description, as after the self-collapse the impetus for further neurophysiological processes and functions can then be described classically, thus causally and locally. The formula for the quantification of quantum gravity threshold is

300 Mermin David N. (2007): Quantum Computer Science: An Introduction; Cambridge: Cambridge University Press

$$T = \frac{\hbar}{E} \qquad (8\text{-}3)$$

where T is the duration of the superposition between the tubulin, $\hbar$ the reduced Planck's constant and E the gravitational energy of the different mass distributions of the two tubulin states, certainly depending on the number of coherent tubulin proteins. Penrose assumes an average superposition time of half of a second and comes to the following calculations: a typical neuron features around 10^7 tubulin dimers. If about 10^6, thus around 10% of the tubulin dimers, constitute a state of quantum coherence, then around 1,000 neurons, which would be correlated for half a second, would exceed the threshold of quantum gravity; the orchestrated objective reduction of the wave function happens. The conclusion is that an organism that is capable of maintaining quantum coherence over 10^9 tubulin dimers for around $500 * 10^{-3}$ sec is capable of experiencing conscious experiences. More tubulin for a shorter time or lower for a longer time allow the same.

Now that quantum states are not local, each ORCH OR can tie together various superpositions, which either emerged spatially distributed or at different times. The only condition would be the exceeding of the gravitational threshold at a specific moment. According to his calculation also a single electron could gain consciousness (here Penrose's concept of complete randomness of the occurrence of consciousness becomes obvious). However, for this an electron would have to be in undisturbed isolation and able to maintain the superposition for longer than the age of the universe. If the electron is not capable of doing this, then the random reduction R with its environment occurs. We can see that the theory distinguishes between random and non-random, non-algorithmizable OR. The dubious nature of this distinction is reflected in so that certain parts of the organism only work completely random, whereas the microtubules cause non-random processes. However, even this does not happen exclusively, microtubules are home to both classical and quantum mechanical processes. The polarity of tubulin final states that can change in 10^{-9} sec increments propagates locally (classically) on neighboring regions to form patterns and may act as signal relay. This can be done at a speed from 8 to 800 meters per second and is thus just as fast as the forwarding of the action potentials. This classical process Hameroff and Penrose set equal with autonomous, not mind-enabled activities, while the far-reaching, long-lasting quantum physical coherences within the microtubules are set equal with the unconscious, but consciousness-causing processes.

The theory strongly depends from the maintenance of a quantum linear superpositions, or coherences. For such isolation is required, thus an energy gap to the environment, and the theory mentions three possible mechanisms for this:

- This isolation may be caused by the ordered water which fills the microtubuli and may surround them up to $3 * 10^{-9}$ m.
- A hull of gelatine may shield the microtubule, where the surrounding cytoplasma may occur in two phases, either as separation fluid or as gel.
- The inner space of the microtubule may be responsible for the isolating function.

What is currently not clear is even if the microtubules would provide enough isolation for maintaining a superposition within, how the synapses can be bridged over so that coherence can be maintained over several nerve cells. Furthermore, if the theory is true, the role of classical signal processing, thus nerve impulses by action potentials over dendrites, synapses, cell bodies and axons has to be reconsidered. Maybe signal processing and interpretation is split in the sense that signal processing required for consciousness only happens on quantum

layer and everything else on classical layer? Or maybe conscious experiences are both processed on classical and quantum level? For information processing in an artificial conscious entity we could make use of a combined approach, such as described at 10.5.1.4 Entanglement, where both classical communication and such on quantum level are used for processing information.

8.3 Further models

A theory going beyond the optical conductivity of microtubules maintains that microtubules forward not only photons of light, but are also constructed and structured of light photons. [301] The cause of this phenomenon is the focus of coherent waves, which creates the conditions for the arrangement and concatenation of matter along to the radiation beam, thus a mutual stabilization of coherent radiation and matter.

Another theory has the same form as the Schrödinger wave equation in quantum mechanics, in which interference plays a crucial role in the neural wave equation because each spatio-temporal order goes over to a non-local implicate order if it goes over in an interference. [302,303] If the brain is considered from the viewpoint of quantum mechanics, this means that if an input reaches the interference of the brain waves, it goes into a non-local, thus ubiquitous order, so it will be interpenetrated by other patterns, leaving no trace behind. This implicit, non-local, quantum physical order must be distinguished from the macro physical, local order of topological maps, which is researched on in the main branch of the brain researchers in its description of the brain areal-bound functions.

A further theory describes a transition from a local composite pattern, e.g. a meaningless syntactic form to a non-local activity.[304] The jump is thus interpreted as a form of non-local phenomenon, and the microtubule hypothesis is extended beyond the membrane on the extracellular space and hence in the synaptic cleft. These protein tubes are called extracellular matrix, and they are often linked from the inside out via hydrogen bonds, which is considered as a quantum physical system, for with its charge the net captures non-local photons. Water molecules are also closely associated with the net, have a weak electrical charge and fit together to form a huge molecule of water, which spans about 50 micron regions. A molecule of that size requires a quantum mechanical description, thus the non-locality is preserved, and the mediating instances of the transfer function of the cells are huge water molecules and light.

The last theory to be mentioned here focuses on the synapses. [305] In it, quantum operations can play a role throughout the electrophysiological process of the neuron only there. In the basic synaptic units (boutons) exocytosis takes place, which is nothing more than the temporary opening of a channel in the pre-synaptic membrane and the release of transmitter substance into the synaptic cleft. Exocytosis always takes happens in the way that if the

301 Quantum Mind: Quantum Mind [2013-09-01]; URL: http://www.quantum-mind.co.uk/qbd-3-c58.html
302 Ricciardi L. M., Umezawa U. (1967): Brain physics and many-body problems, cybernetics, vol. 4
303 Quantum Mind: Quantum Mind [2013-09-01]; URL: http://www.quantum-mind.co.uk/jibu-yasue-c575.html
304 Globus Gordon (2003): Quantum Closures and Disclosures: Thinking-together Postphenomenology and Quantum Brain Dynamics; Amsterdam: John Benjamins Publishing
305 Eccles John C. (1994): How the self controls the brain; Berlin Heidelberg: Springer-Verlag

transmitter substance is released, then all of it. An analysis showed that a pulse which is transmitted from an axon to a basic synaptic unit with an average probability of 25% triggers a release, for which a quantum mechanical description has been developed. In this case, it was assumed that a tunnel of a quasi-particle movement is the release mechanism of exocytosis, so the idea is that the mental intent is active in selected areas of the cortex by a momentary increase in the probability of exocytosis. In the language of quantum physics, this means that a selection of events was already prepared with a certain probability. The time frame refers to experiments[306] that yielded an interval of $200 * 10^{-3}$ seconds between conscious will and the execution of a voluntary act.

8.4 Summary

Where does that leave us now? At least for me, it is very appealing to search for explanations for how the human mind works in the field of quantum mechanics. The ORCH-OR model is partially based on experimental results that cannot be ignored, such as Penrose's experiments with the Paramecium or Hameroff's explanations on how anesthesia eliminates consciousness. However, the question is if the human brain can be reduced to a quantum computer; I personally am of the opinion that only by the fact that self-organization occurs within biological brains, such are capable of what they currently are. Apart from that, quantum effects have so far been detected only in highly artificially isolated systems to a maximum of -140° C. It is believed that at 37° C in living tissue quantum states completely collapse, as they interact with their environment. Without isolation and a consequent energy gap coherence seems to be lost permanently. Be that as it may, by now the extraordinary effectiveness of the entanglement of protons in normal water at normal room temperature has been detected, so that a natural constant, the scattering angle, measured from the proton bombardment by neutrons, is no longer a constant. The entanglement of the individual protons changes very quickly high 10^{-11} seconds, but it is believed that around 25 – 50% of the protons are correlated at any time. Unfortunately, these assumptions are still not convincing, as yet no theory for decoherence has been developed, so if a residual memory remains at two particles that have just separated, is currently not known. Although the microtubules and the microphysical quantum phenomena feature the same size range, for triggering a conscious perception the activity of several tens of thousands of synapses is needed, so that nearly all neuroscientists exclusively consider a classical description adequate. This could be argued as a counter-argument: if one photon hits the eye, it can be seen in total darkness. Furthermore, it has been shown that decision-making processes in a cell feature at least the degree of fineness of a molecule. Another argument against quantum physical approach of brain activity is the measurement problem. So far, there is no way in sight to actually verify such approaches also experimentally, because it is required to do this on living tissue. However, in order to measure something, one has to prepare it; unfortunately, for a living organism this would mean to kill it.

Anyway, as stated beforehand, it is not my aim to dismantle the discussed theories down to a level where no proof for their plausibility can be found – within this elaboration I use parts of them for modelling possible implementation scenarios for machine consciousness. For this, I consider a implementation relying on both information processing on classical as well as on quantum level to be promising.

306 Philosophie verständlich: Die Libet-Experimente [2013-09-01]; URL: http://www.philosophieverstaendlich.de/freiheit/aktuell/libet.html

9 Matter and consciousness

As we have seen in the last chapter, the model of consciousness brought forward by Hameroff and Penrose seems to be very promising, although for me, as a computer scientist, a solely physical explanation of human consciousness seems to be true only partially. I suppose that the brain as the structure being responsible for complex thought processes, information storage as well as home of our consciousness does benefit from quantum effects, however, not solely. It is required to take the organic specialties of living organisms into consideration as well. Hameroff already mentioned that the cytoskeleton of a nerve cell differs from the one of other eukaryotic cells in the sense that the in the brain neurons are placed in parallel instead of radially, are more stable due to the support of proteins, are distributed more densely and form larger networks. However, there still remains the classical activity we can observe, like the firing of neurons, the building/ strengthening/ weakening of connections between single neurons (remember also 3.3.3.4 Hebb's learning rule or 3.3.3.5 Delta rule) or the release of neurotransmitters. Furthermore, one has to consider the development of the brain during growth, as all the neurons are already present at birth. However, only ~ 2,500 synaptic connections exist, which increase sixfold up to 15,000 at the age of 3. So if all the neurons already exist at birth, would not theoretically conscious experiences be possible to the full extent as well? Although I have heard of exceptional circumstances where people tell that they already had conscious experiences in their early days, this is for sure no common situation. Sure, one might argue that qualia serve as the input for the (maybe quantum physical) neural computer that produces conscious experiences. So, for the beginning, it is required to define what qualia actually are.

9.1 Qualia

Qualia are defined as the subjective content of experience of mental states. However, this is the only thing what can be said about qualia that has been widely accepted, as opinions widely differ, however Thomas Nagel has characterized the idiom that qualia can be interpreted as ‘what is it like’. An example is a person that has had enough food so that he/she is not hungry any more. Manifold neural processes are conducted within the person’s brain and the person may behave in a special way, like grinning and leaning back in the chair. Additionally, the person encounters the special feeling of being not hungry any more. To be complete here, this only functions when the person in consideration has understood what subjective contents of experience are. However, we can surely include things like our five senses into qualia, as they help us to

- experience visually,
- receive tactile feedback (including pain or pleasures),
- taste,
- feel, and
- smell.
- Furthermore, we may include
- emotions,
- proprioceptions, that is the knowledge humans have about the location of their body in space as well as their performed bodily movements, based on feedback from their nerve endings into the central nervous system,

F. Neukart, *Reverse Engineering the Mind*, AutoUni –
Schriftenreihe 94, DOI 10.1007/978-3-658-16176-7_9

– sensations like the beforehand mentioned feeling of being not hungry any longer, or feeling coldness or heat,

and maybe also

– ecstatic experiences,
– understanding,
– confusions, and the like.

The difficulty with qualia is, as already indicated, that they are subjective. Therefore, it is a major difficulty to measure or quantize any of them, which at least in part is required for our understanding on how to allow an artificial entity to experience the same, thus conscious experiences. Due to these restrictions, qualia have the negative connotation of not being exactly describable. So, assuming that conscious experiences or the mind as a whole is a sort of software running on the brain including some random operations, is it then possible to write such a software and process it on massively parallel processing classical (von Neumann) computers, as we may describe our brain in terms of information technology, or does our brain benefit from quantum effects as well and thus the same mind-program would allow conscious perceptions only on a quantum computer? Maybe we should first try to understand the meaning of qualia for conscious experiences even better, starting with why qualia may be required for experiencing conscious perceptions, however, even before that it is required to provide a very short overview of some theories of consciousness with respect to artificial intelligence.

9.2 Materialism

Materialism assumes that the human brain is a computer, and that it can be fully understood in terms of architecture and structure. It assumes that as soon as the smallest parts of the brain have been understood and how they are connected it can be replicated on any other platform that does not necessarily need to constitute of the same ingredients – there should not exist any limitation that dictates why all brain functionality must be limited to biological neurons. One of the main arguments against materialism from comes from idealistic side and is that one can understand human mental abilities such as self-confidence is not material and cannot be fully attributed to matter. In contrast, one of the main arguments against idealism from the materialistic point of view is that idealism cannot explain the autonomy of the sensible world and the observed independence of mental processes. It is, however, ignored that any form of sensory perception and observation beyond mental processes is impossible. Further arguments against materialism contain that materialism cannot explain itself, since it does not occur as a theory and as matter. Moreover, the concept of truth cannot be understood purely material. The epistemology will shorten the materialism of an empirical science. Cultural content, ideas and all intangible forms have no independent existence of its own. A critique of knowledge or an independent reflection of knowledge were not, or only in a very limited materialism. A review of scientific hypotheses is possible only within certain metaphysical preconditions.

Back to consciousness, a major point of criticism takes the products of human mind as a starting point for its argument. Even under the assumption that ideas, theories, (building) plans, technical know- how and more are produced by the brain (and not from consciousness), such could exist independently from their authors. Because man is surrounded by a spiritual

world, the making up its cultural heritage, without which he would always start at the level of a pre- stone age-people.

9.2.1 Eliminative materialism

The idea behind eliminative materialism is that claims about mentality are a weakly explanatory concept that will be eliminated as soon as science progresses.[307] Thus, it takes the positions that our mind and qualia should be eliminated for being superseded by scientific ways of their description. However, humans do experience things and thus feature qualia, which are bound to the human mind, thus mental states. As the scientific community has generally accepted the existence and notion of qualia, this point of view, as well as materialism in general, can hardly be held.

9.2.2 Noneliminative materialism

Qualia are real, and are the direct causal product of some purely physical aspect of the brain. The question that arises is the one after how matter should be capable of producing consciousness? Two possibilities have to be taken into consideration:

- Qualia emanate from some special substance in the brain
- Some dynamic but still physics-based property of the brain yield qualia
  - Ordinary physics such as electromagnetic fields
  - Quantum physics[308]

As Katz further mentions, there exist some problems with noneliminative materialism, namely that the production of conscious content is linked to neural activity – although not all neural activity may produce content, it can, as far as we can say today, only be produced by neural activity. Yet neural activity is defined as neurons submitting or receiving electrical signals, and not by producing molecules, like the special substance that may be emanated in the first explanation above. Furthermore, if such a substance would actually correspond to our consciousness, what would happen if it was extracted and injected into another brain? On the other hand, it cannot be denied that the physical explanation may partially support a possible explanation. Quantum theory seems to be a promising candidate for explaining some of the phenomena occurring in the human brain, as we have learned from the ORCH OR model of consciousness.

9.3 Functionalism

Functionalism currently is the most well-known and popularly discussed theory of mind, a fact that it owes the influence of IT on our society. It denies that mental states are identical to physical states, or that the former ones can be reduced to the latter ones. It thus states that the

307 Katz Bruce F. (2011): Neuroengineering the future - Virtual minds and the creation of immortality; Massachusetts: Infinity Science Press LLC, p. 101 ff.

308 Katz Bruce F. (2011): Neuroengineering the future - Virtual minds and the creation of immortality; Massachusetts: Infinity Science Press LLC, p. 103

mind is an algorithm that is processed on the brain, like the software is processed on a computer. This mind-software would also incorporate all that has been mentioned at 9.1 Qualia. Functionalism dictates that being in a mental state is being in a state that transforms inputs into appropriate outputs, which serve a functional role.[309]

Due to the fact that functionalism does not demand a human brain for processing what we got to know as mind-algorithm nonchalant, it offers some interesting perspectives. Most of us would agree that animals like cats or dogs do feature some sort of rudimentary consciousness. Not only mammals may have that privilege, as Stuart Hameroff showed by describing the paramecium's clever behavior.[310] The paramecium consists of one single eukaryotic cell and a cytoskeleton rich in microtubules, which seems to be sufficient for learning behavior. However, this does not belong to the area of functionalism, but should just help to put the imagination in the right way – neither only humans may benefit from conscious experiences, nor exists the requirement that entities experiencing such are carbon-based (to avoid misunderstandings: the paramecium is). Functionalism seems to be incomplete the sense of what we are searching for, namely an explanation of how we can describe how the creation of conscious experiences works. Some major problems have been identified.

9.3.1 The problem of absent or inverted qualia

Theoretically, from an engineering point of view it could be possible to create an artificial brain with the same architecture than a biological one, but which cannot experience qualia. If such an approach would be successful, qualia cannot be explained with functionalism. Inverted qualia means that it may be possible to re-engineer a human brain that is capable of experiencing qualia by sensory inputs, but to invert these experiences. An example used very often is the inverted color spectrum: two persons declare an item to be red, and both persons experience 'redness' although one of the two sees blue (which, in fact, is red for this person).

9.3.2 The Chinese Room argument

The philosopher John Searle described a closed space in which a person resides. To this person a paper with stories in Chinese is delivered through a slot in the door. The person is not capable of understanding the Chinese language, and is thus not only unable to understand single words, but also the whole meaning of the story. Furthermore, the person receives a paper with questions about the story (also in Chinese). Additionally, the person finds stack of Chinese scripts and a manual with rules written in his native language. The manual allows the person to connect the symbols with the history, but only at the level of character recognition (about the shape of the characters). From the manual, the person finds instructions which character he has to transfer (depending on the character of the story and the questions) on the answer paper. The person follows purely mechanical instructions and pushes the result, thus the 'answers' to the questions, through the door slot without having understood the story or the questions.

---

309 Katz Bruce F. (2011): Neuroengineering the future - Virtual minds and the creation of immortality; Massachusetts: Infinity Science Press LLC, p. 110

310 Hameroff Stuart: Can quantum mechanics explain free will [2013-09-11]; URL: http://www.quantumconsciousness.org/views/freewill.html

In front of the door, a Chinese native speaker is waiting and reading the responses. Because of the meaningfulness of the responses the Chinese native speaker comes to the result that in the room there is another Chinese-speaking person. Searle thus argues that functionalism cannot account for intentionality.

9.3.3 The knowledge argument

Frank Jackson put forward an argument that suggests that qualia do not supervene on the physical architecture or structure of a human brain and therefore, cannot be explained physically or functionally. Summing up, he asks to imagine a neuroscientist named Mary who is brought up in a room that does not feature any color – it is just black and white. Mary becomes an expert in understanding which functional roles brain states are important for color vision; she is able to gather comprehensive knowledge about that special area. However, although she has that intense knowledge about the neurophysiological workings of color vision, Jackson assumes that she would learn something new the first time she leaves her black and white-only room. The point is that her knowledge did not allow her to predict the qualia experienced the first time after experiencing color vision.

Despite these arguments, functionalism has gathered lots of supporters, also in the field of artificial intelligence research. The usual functionalist example for a computer is the one of a computer adding two numbers. There are two levels that have to be taken into consideration, the hardware and the software. On the one hand, what is happening internally when adding two numbers is dependent from the hardware, however, the process of calculation cannot be described as hardware activity, as different computers feature different hardware. This is where software adding the two numbers comes into play, namely in terms of a function implemented on the hardware. Thus, hardware only serves for the realization of the calculation (by means of software), but does not implement the calculation. This has led supporters of strong artificial intelligence to the concept of functionalism. To be complete here: supporters of strong artificial intelligence argue that once a computer will be built that is capable of implementing a conscious entity to the full extent (capable of reasoning, thinking …), whereas supporters of weak artificial intelligence claim that such behavior can only be simulated and computers thus only seem to act consciously. This brings up another question, namely when a simulation is well enough so it is not regarded as simulation any longer.

9.4 The Identity Theory

Identity theory states that for each sensation we experience, there exists an exact neurological state in the brain. Thus, according to this theory any sensation may trigger the interaction or firing of a set of neurons. Therefore, any experience would theoretically be observable and furthermore, that any observed activity in the brain can e.g. be brought into connection with specific qualia or thoughts. In a certain manner this means that there is no difference between mind and brain. Although this theory seems to be promising at a first glance, especially when taking possible technical implementations into consideration, a closer investigation reveals some weaknesses, from which especially one is critical: we can say that different people react differently to the same situations, depending from their gathered experiences or knowledge. So when we consider that a specific visual experience, like seeing the president speak on TV, results in a specific brain state, then we have to consider that for one person this could result

in positive emotions and for another person in negative ones, and an additional person may be concerned by a specific political situation some years ago, when another president used the same words. The conclusion is that the same brain state, namely seeing the president on TV, does not mean the same for different people. Followers of this theory may encounter this by stating that seeing the president on TV does not correspond to a brain state, instead it is what each individual experiences, which may differ.

9.5 Summary

Within this chapter the fundamental ideas for being able to form a model of mind and reverse engineering the mind have been discussed. When trying to describe the mind in a technical way, which is indeed the way that is needed for fulfilling our purposes, then it gets obvious that different external sensations do not affect everybody in the same way, thus cannot form the foundation for modelling our mind. Noneliminative materialism contains very interesting ideas like quantum physical explanations of the mind; however, it does not take into account the software, thus the algorithm of mind, working on the hardware. Functionalism does that in the way that it states that the mind is the software running on our brains, and that different physical implementations may be realized for processing the 'mind-algorithm'. Apart from that, identity theory states that each external sensation may cause different internal sensations, which of course may be different from individual to individual, but if the resulting internal sensation of two individuals is the same, also the same areas of the brain are affected. This is also very interesting for our purposes, as we do seek a possible form of implementation that can serve as a computer for minds of different people, and not just the mind of one specific person. From a biological point of view, this theory may not be true, as although the basic structure of all human brains (see 1.1 Anatomy of the human brain) is the same, there are differences in the number and structure of neural connections or brain functionality on neural layer. Hence, a combination of all three theories will not necessarily be the correct one from a philosophical point of view, but a valid approach from an engineering position.

10 Reverse engineering the mind

Within this chapter all the requirements for reverse engineering the mind based on the knowledge imparted in the previous chapters will be discussed, and open questions attempted to be solved. A suitable theory of mind that on one side may not be the whole truth from a philosophical point of view, but serves as a valid foundation from an engineering point of view on the other side is introduced. Furthermore, as I indicated more than once, I am of the opinion that both quantum physics as well as self-organization occupy the most important roles in how our brain works and lets us experience conscious content and again, it is required to plunge into the information theoretical approach to quantum physics, quantum computer science. All of these fields will be discussed in detail, but the borderlines blur very often, as from a superior point of view (which is our brain as a whole) all these fields must interact closely.

10.1 Theory of mind

We got to know the required theories of mind for being able to formulate the one that will serve as a foundation for reverse engineering the mind. First of all, non-eliminative materialism and materialism in general incorporate some of the features we need, namely

- the assumption that the human brain as a computer can be fully understood in terms of structure and architecture and that
- the brain can be replicated on any other platform as soon we have gained this understanding.

The smallest parts in our brain we have to understand are those acting on quantum level, thus elementary particles, atoms or molecules. ORCH OR already suggests that quantum coherence may be maintained over larger areas, due to the occurrence of ordered water occurring within microtubules, which shows the same effects as a Bose-Einstein condensate and can thus be described as an extreme physical state of a system of indistinguishable particles, where the majority of the particles exist in the same quantum state in what we perceive as time. This is only possible if the particles are bosons and therefore subject to Bose-Einstein statistics, the latter describing a probability distribution in quantum systems (we remember that spontaneous violations of the symmetry can result in quantum states within the brain, which are long-range correlated waves, called Goldstone-Bosons, see also Chapter 8). Thus, from noneliminative materialism we gathered the insight that the brain can be fully understood in terms of functionality and for our purposes thus hardware capable of implementing quantum coherence effects is required. However, to be equipped with the quantum-hardware alone is not sufficient for the chosen approach. As mentioned multiple times, self-organization is one of the most important factors in the human brain, so two probable approaches may be followed, the

- self-organization of hardware followed by the assumption that the implementation of software is not required or the
- implementation of self-organizing software on the quantum-hardware.

The second approach seems to be more promising than the first one, as we just have to ensure that the hardware is capable of processing all the functions the software is implementing, which can be specified even if the software is self-organizing. Even the first approach

F. Neukart, *Reverse Engineering the Mind*, AutoUni –
Schriftenreihe 94, DOI 10.1007/978-3-658-16176-7_10

contradicts a purely materialist view, as also self-organizing must implement software, as can be made clear by having a look at claytronics, a sort of programmable matter being researched on at Carnegie Mellon University in cooperation with Intel.[311] Hence, also functionalism must be included into our theory, thus the idea that the mind can be recreated algorithmically. If the mind was just an algorithm, then we could easily program it. As of our current knowledge, the mind does not work deterministically, so the implementation of a mind-algorithm would be difficult and the classical functionalist view drops out. A solution to this dilemma is that the software must be self-organizing has to work within pre-defined parameters; it must be allowed to structure and develop itself, but within boundaries as also our brain does. Our brain seems to work at the borders of chaos, but never becomes a chaotic system although it features numerous, if not infinite degrees of freedom. This is what we have to try to implement through our software. Last, but not least also identity theory features some important aspects, as it states that each external sensation may cause different internal sensations. These may, of course, be differ within each individual, but if we only focus on the internal sensation created by an external one and take into consideration that two people experience the same internal sensation caused by a specific external one, then the idea that this affects the same neural activity may not be true from a biological point of view, but nevertheless seems to be promising from an information theoretical approach. This is, because the hard- and software we are seeking for reverse engineering the mind should not only be capable of implementing the mind of a specific subject with a specific brain structure, but rather on the one side be so generic that different evolutions would result in different final states, and on the other hand so programmable that functional on neural layer can be identified with high accuracy.

Summing up, our theory incorporates ideas from noneliminative materialism, functionalism and the identity theory. The assumption is that the brain can be rebuilt through on a specific form of hardware that is capable of generating and maintaining quantum effects. These effects are triggered by self-organizing software at specific events, like the experience of conscious content. The software works within predefined parameters on the border to chaotic systems, but is never allowed to become chaotic, despite its possibly infinite degrees of freedom. The implementation of self-organizing software on quantum-hardware must allow the assignment of specific sensations to specific parts of the software, thus the software must be definable within itself although the only thing we can do before it starts working is to parameterize its environment.

10.2 Quantum linear superposition in artificial brains

The ORCH-OR model has described that within a human brain quantum linear superposition may occur in the ordered water of microtubules and that it may also be a large-scale effect. Even if this is not the case in a biological brain, it is a more than promising approach for artificial brains. At first we should discuss why there exists the assumption of neuroscientists that large scale quantum effects in a biological brain are not of relevance, followed by some explanations why it may nevertheless useful in an artificial implementation.

311 Carnegie Mellon: Welcome to the Claytronics Project [2013-09-22]; URL: http://www.cs.cmu.edu/~claytronics/

Regarding relevant quantum effects within our brain it is at first required to reconsider the functionality of neural information transfer as far as we understand it today. It is commonly known that neurons transmit and receive electrical signals; however, before a neural signal can be transmitted from one neuron to the next, it has to travel along the axon. The physical and chemical gradients can be coupled efficiently and serve as an energy source. The axons of the neurons conduct electrical pulses, called action potentials, which travel along nerve fibers, like waves along a jump rope. This works, because the axonal membranes contain ion channels that open and close to let bypass electrically charged ions. Some channels allow through Na^+ ions, other K^+ ions. If the channels open, the Na^+ and K^+-ions flow along the opposite chemical and electrical gradients in the cell in or out as a result of the electrical depolarization of the membrane. If an action potential begins the cell body, the Na^+ channels open first. A wave of Na^+-ions travels into the cell and a new equilibrium occurs within a millisecond. In an instant, the transmembrane voltage changes by about 100 mV. The inner membrane potential changes from negative (about-70 mV) to positive (about +30 mV). This change opens K^+ channels and a wave of K^+ −ions flow out of the cell, almost as soon as the Na^+ ions that are passed inwardly. This means that the membrane potential inside of its negative part returns to its initial value. For this to happen, only very few ions are required to cross the membrane – the concentration of Na^+ and K^+ ions in the cytoplasm does not change significantly during the action potential. Thus, the firing of neurons includes transfer of ions, which is also a counter-argument against the first noneliminative materialism-approach, which dictates that some special substance may emanate conscious content, as not new substances are created, let alone one that contains conscious experiences.

The above explanation does entirely go without any quantum physical phenomena. However, the remaining question is still if only the neural signal transmission and reception can account for conscious experiences. Even if every conscious event causes areas of neurons to fire, the quantum layer may play a role that has not yet been understood. What I want to say is that even if we would fully understand the classical level of our brain we could not exclude the relevance of actions on quantum layer in the brain, for we would not have a proof against it. The idea that quantum physics may play a role in producing conscious experiences is, at least to me, very appealing; however, as already indicated, neither am I of the opinion that quantum physics alone can be made responsible for the existence of consciousness, nor do I believe that classical physics alone can account for that. Either way, within this elaboration reverse engineering does not stand for analysis, understanding and cloning; it stands for analysis, understanding and implementation of functionalities, irrespective of differences in architecture that may be encountered. The proposition of reverse engineering mind functionalities is approached from an information theoretical perspective, which may include everything that is of value for a possible realization scenario, irrespective of its relevance for the human brain has been proved or not. Thus, if quantum physics is considered to be useful, it will be included in the proposal, irrespective if it features some effective impact on our minds. What has already been mentioned is that the brain is a complex self-organizing structure, featuring infinite degrees of freedom. Hence, an artificial neural network or better, a combination of artificial neural networks seems to be a promising idea. However, this cannot be an artificial neural network as described in 3.3 Neural networks or even chapter 4. The only neural network that would be able to encounter the requirements of what we are looking for can be one that is processed on a quantum computer. Thus, not the hardware will be self-organizing, but the software and the software will be capable massively parallel information processing due to quantum hardware, such as described in 10.5 Quantum physics and the artificial brain. However, before going into detailed descriptions of quantum artificial neural networks, an

equivalent for the ordered water within microtubules described by the ORCH OR-procedure needs to be found. Such large-scale coherent quantum states are called Bose-Einstein condensates, but today we encounter several problems when we try to produce such. Bose-Einstein condensates are macroscopic quantum objects, in which the individual bosons are completely delocalized. The probability of each boson to be found at a certain point is the same everywhere within the condensate. The state can thus be described by a single wave function. This results in properties such as superfluidity, superconductivity or coherence over macroscopic distances. The latter allows interference experiments with Bose-Einstein condensates and the implementation of an atom laser, which can be maintained through controlled extraction of a part of the matter wave from the trap holding the condensate trap. The phase transition of a conventional atomic gas to a Bose-Einstein condensation occurs when a critical phase bulk density is achieved, e.g. when the density of particles having almost the same impulse is large enough. The atoms are quantum particles whose motion is represented by a wave packet. The extent of this wave packet is the thermal de Broglie wavelength, which becomes larger, the more the temperature falls. If the de Broglie wavelength reaches the mean distance between two atoms, the quantum properties come into effect. In a three-dimensional ensemble the Bose-Einstein condensation sets in. Therefore, it is required to increase the density of the gas and to reduce the temperature for being able to reach the phase transition. In the framework of statistical physics, by the application of the Bose-Einstein statistics, the critical temperature T_c of an ideal Bose gas can be calculated, below which the Bose-Einstein condensation sets in:

$$T_c = \frac{h^2}{2\pi m k_B} \sqrt[3]{\left(\frac{n}{(2S+1)\zeta\left(\frac{3}{2}\right)}\right)^2} \qquad (10\text{-}1)$$

where h is the Planck's constant, m the particle mass, k_B Boltzmann's constant, n the particle density, S the particle spin and

$$\left(\frac{3}{2}\right) \approx 2{,}6124 \qquad (10\text{-}2)$$

Riemann's Zeta-function. The Bose-Einstein statistics or Bose-Einstein distribution is a probability distribution in quantum statistics. It describes the mean occupation number $\langle n(E)\rangle$ a quantum state of the energy E being in the thermodynamic equilibrium at the absolute temperature T, for identical bosons as occupying particles. Analogous there is the Fermi-Dirac statistics for fermions, which goes, as well as the Bose-Einstein statistics, in the Boltzmann statistics in the limit of large energy E. The core of the Bose-Einstein statistics is that at simultaneous exchange of all four variables x, y, z, m of two bosons (x, y, z representing the position in space, and m the spin), the wave function ψ, respectively the state vector of a many-body system does not change its sign ($\psi \to \psi$), while it does change the sign in the Fermi-Dirac statistics (($\psi \to -\psi$)). In contrast to fermions, several bosons can therefore be in the same one-particle state, thus featuring the same quantum numbers.

The usual method for generating Bose-Einstein condensates of atoms consists of two phases:

- First, the atoms are trapped in a magneto-optical trap and precooled by laser cooling. However, the laser has a lower limit for cooling temperatures (typically about 100 μK), which is due to the recoil of the spontaneous emission of photons.
- The average velocity of the atoms cooled by that way is of only a few centimeters per second, which is small enough to be captured in a magnetic or optical trap. Through

evaporative cooling, e.g. continuously removing the most energetic atoms, the temperature of the cloud of atoms can be reduced further. In this process, usually more than 99.9% of the atoms are selectively removed. Thus, the remaining atoms reach the necessary phase space density to complete the phase transition to a Bose -Einstein condensate.

Until 2004, it was possible to reach at ultralow temperatures of 10^{-7} K and to create Bose-Einstein condensation for many different isotopes in this way. Furthermore, Bose-Einstein condensation could be realized with one hydrogen atom, albeit with slightly different methods.

That the above mentioned gases show bosonic and not fermionic behavior is based on a subtle interplay of electron and nuclear spin at ultralow temperatures: at correspondingly low excitation energies the half-integer total spin of the electron shell of atoms and also half-integer nuclear spin by the weak hyperfine interaction are coupled to an integer total spin of the system. In contrast, the behavior at room temperature is determined solely by the spin of the electron shell, because here the thermal energies are much larger than the hyperfine field energies. The search for an equivalent of Bose-Einstein condensates, thus some sort of matter that allows wide-scale quantum coherence effects, it is required to recall the idea of implementation. The idea is that quantum coherence should be realized within an artificial neural network, a quantum artificial neural network, which has already been found (see 10.5.1.8 Envisaged implementations of a quantum artificial neural network).

10.3 Self-organization

As already indicated, I find the idea of considering the brain as quantum computer appealing, but it is also required to take into account neural information processing and self-organization. The ORCH OR model mentions the property of self-organization in the way that it explains that microtubules are self-assembling hollow crystalline cylinders whose walls are hexagonal lattices of subunit proteins known as tubulin and that in neurons, microtubules self-assemble to extend axons and dendrites and form synaptic connections for helping to maintain and regulate synaptic strengths responsible for learning and cognitive functions. [312] The human brain, with all its neural connections, can be understood as self-organizing computer. Depending from sensual inputs and environmental stimuli, connections between neurons form, are strengthened or weakened. Furthermore, self-organization is not something that our brain experiences once, it is a continuous process. Thus, our brain is a dynamical, self-organizing and distributed system. Through each impulse of awareness, neural structures, thus multiple neurons, are affected in the sense that learning happens. A major advantage is that every new input is compared to existing experiences with the goal that similar sensations are processed in the same neural area. This leads to pattern and connection stabilization of the respective structures. The more often such stimuli reach a specific neural pattern, the more stable it gets and the more likely it is that these pattern can be actualized and used for processing similar inputs (see also 3.3.3.4 Hebb's learning rule).

312 Hameroff Stuart: Quantum computation in brain microtubules? The Penrose-Hameroff "Orch OR" model of consciousness [2013-09-18]; URL: http://www.quantumconsciousness.org/penrose-hameroff/quantumcomputation.html

An important structure in learning and thus self-organization is the hippocampus (see 1.1.2.2 Hippocampus), which evaluates sensory inputs in the sense that it classifies the actual input into categories like ‘already known’, ‘new’, ‘old’, ‘important’, ‘interesting’,... The hippocampus is, as far as we know today, also important for experiencing conscious content, as it ensures that events, news, facts, etc… are experienced consciously. New experiences, associations … may be stored within this part of the brain, but only for a short time as its storage capacity is very low. Therefore, the hippocampus transmits the information it considers to be important to the cortex (1.1.3 Cortex and neocortex), which is more suitable for storing information permanently. We can imagine the hippocampus as trained neural network that trains another neural network. The learning approach itself can be compared with chapter 5.1, which copies information from one artificial neural network to another.

On taking stock we see that the human brain benefits from plasticity (called neuroplasticity in neuroscience), which is the ability of the central nervous system to structurally and functionally adjust through internal change processes due to external conditions and to changing requirements. Our perception, our capacity, our speech and language ability are becoming more sophisticated and extensive as in the course of child development. In the first years of life, not the number of neurons not necessarily increases, but the synaptic density as well as large neural bundles and connections are partially restructured and contoured. Change processes happen due to the fact that the neural structures and connections become finer through development-related neurochemical processes. Furthermore, the respective situations and challenges of life evoke the development and adaptation of neural structures. As far as we know today, not only individual nerve cells can change depending on new utility models (synaptic plasticity), but that whole areas of the brain can adapt to new challenges (cortical plasticity).

Before going into further detail with self-organization, it is required to define some termini that are important in this context.

10.3.1 Structure and system

In mathematics, the term structure is defined as the kind of relations and definable operations on a quantity of different elements. A structure of a system is the kind of ordinance and the conjunction of elements, or the entirety of relations between these elements. There exist spatial and temporal structures, and in terms of functionality we have to differentiate between conservative and dissipative ones, whereas only the latter one is of interest for us.

- A system is a relatively stable, ordered entirety of elements and relations that is characterized by the existence of specific laws.
- Structure building can be characterized as the creation of new structures; however, here structure building is seen the emergence of higher, more ordered structures, like the enhancement of a neural network by the adding or removal of synaptic connections.

10.3.1.1 Conservative structure

A conservative structure is a structure that heads for a local minimum of potential energy. This means that a system that is deprived warmth, obtains a higher degree of order. An example is a ferromagnet that loses its spontaneous magnetization at high temperatures, but

regains it at low temperatures, which happens because on microscopic layer the interacting elementary magnets act cooperatively, and arrange in parallel when the thermal movement ceases. Such conservative structures are static, separable and mostly of microscopic magnitude. This is where thermodynamics comes into play, as a reduction of entropy through the lowering of the temperature acting on the structure occurs.

10.3.1.2 Dissipative structure

Conservative structures cannot be used to explain biological systems, as cooling results in death. Living organisms can only function by permanently exchanging matter and energy with their environment. This means that chemical energy is supplied in terms of relatively unstructured nourishments, which subsequently is changed into ordered aspects like regulation or movements. Furthermore, this chemical energy is released into its environment by warmth and inferior substances. A system based on a dissipative structure is in continuous imbalance with its environment, thus continuously exchanging matter and energy, which allows continuous export of entropy and structure formation. A neural network, e.g., is a nonlinear dissipative system.

10.3.2 Self-organization in computational intelligence

Without doubt, self-organization in the brain is one of its most extraordinary features. On the classical level, the brain complexity is achieved not only by the number of neurons it consists of, but by causal collaboration of these, which is possible because of the neural interconnections. These dynamically forming interconnections are the connection to self-organization, as somehow the information of how the evolution within such a complex system can be achieved must be coded within the gene structure. Let assume, our brain is a system with infinite degrees of freedom, and let us compare a possible evolution of an artificial brain with selection and evolution, as on a classical level neural, the correct set of neural interconnections for an area of neurons may be determined by the application of genetic algorithms, as introduced in chapters 3.3.3.7 Genetic learning (NeuroEvolution)or 5.1. A major challenge to both brain research and our instincts is the dispersed association of our brains. The neurobiological confirmation amassed throughout the most recent decades has prompted radical changes in our perspectives of the brain. In former days, instinct and contemplation were the major wellsprings of information for the detailing of theories about the association of the cerebrum. Right away it has been discovered that these instincts are in exceptional clash with the confirmation furnished by experimental examinations, raising the fascinating address of why the mind is so agnostic to its own particular association. Instinct recommends that at some place in the cerebrum there should be a merging focus where all data is meeting up to be amiable to lucid understandings. This might be the site where perception takes place, where the intentional agent is active, where decisions are reached, where plans are developed and where the Self is seated. We expect various levelled structures and we likewise reproduce them in the social and practical world – most likely not dependably further bolstering our good fortune, in light of the fact that they may be maladapted when frameworks come to be extremely unpredictable. The actuality of our brains is altogether different. The cerebral cortex contains an expansive number of distinctive zones that, contingent upon their data, fulfil diverse capacities however utilize comparative computational algorithms. In this manner, the configuration of replaceable data is dependably

the same and correspondence around cortical regions can profit by this. This is an important essential for generalization, deliberation, typical encoding and, last yet not slightest, for the constitution of the solidarity of cognizance. The shocking finding is that the associations connecting these territories give just minimal confirmation for serial preparing in strictly various levelled architectures. Rather, the connectivity plan is commanded by standards of parallelism, correspondence and distributedness. Subsequently, neurons placed in the visual cortex can talk straightforwardly to neurons in the limbic system or in official territories, and a large portion of these collaborations are equal. This meshwork of associations is remarkably thick and complex yet a long way from arbitrary. It is exceedingly organized and has the structure of small world networks. This building design is the fittings acknowledgment of the projects as per which brains process data and it is likewise the groundwork of saved knowledge. Let assume a person is confronted with a dangerously-looking dog, which it touches nevertheless and considers it benevolent then. Hence all visual zones will be active and take part in the identification of the mentioned dog, the same holds for the tactile areas, which investigate the surface of its hide, the sound-related regions that disentangle the yapping and the limbic zones that include the passionate essences. There is no single locus for the representation of the coordinated percept of this animal. Rather, the representation comprises of a mind boggling spatio-temporal example of conveyed neural movement. In this manner, the mind exhibits itself as a profoundly conveyed, self-organizing system. Actually, nonetheless, there is no single focus; there is no identifiable seat of the cognizant, purposeful self. The mind is a circulated framework that self composes and produces each one of the aforementioned remarkable phenomena that we as spectators credit to the individual, the Self. The mind, in spite of the fact that it displays nonlinear elements, is tuned to accept that the methodologies to be investigated are straight so as to have the capacity to make sensible forecasts. Notwithstanding, if the cerebrum expects the same concerning its own particular working, provided that it accepts that it executes predominantly straight operations, it is certain to hypothesize a mover, in light of the fact that direct frameworks can't without anyone else's input produce all the striking capacities that we watch, they can't be imaginative, open towards what's to come and intentional.

Self-organizing, distributed and objective administered systems require effective and adaptable components for organization and binding, in a connection subordinate manner, the numerous disseminated neighborhood forms into intelligible wholes. Restricted to tie dispersed effects is meeting in dedicated anatomical circuits. Provided that the messages encoded by units X and Y are to be bound it suffices to associate their yields with a third unit Z and after that to select suitable thresholds for unit Z, with the intention that Z is just animated when X and Y are active in the meantime. Thusly relations could be assessed in unbending, anatomical architectures and communicated by the reactions of conjunction particular neurons. The mind utilizes this technique, however on account of its inflexibility and resoluteness, this methodology can apply just for the encoding of every now and again happening stereotyped relations. The elective is to express relations by dynamic marks, with the intention that the representation remains circulated however capacities as a sound entire, a methodology called get assembly coding. With 10^{11} neurons, each one having the part of an image, and an adaptable recombination component, a basically interminable number of distinctive dispersed representations could be shaped. The representations of novel questions, of the perpetually changing heavenly bodies of true conditions and of adjustable engine reactions, are, consequently, actualized best by progressively configured assemblies. Be that as it may, in get assembly coding, one needs a code that characterizes from example to occurrence which subset of the bunch animated neurons truly helps a specific representation.

As there will dependably be some coinciding congregations, an unambiguous indicator is wanted that advises to whatever is left of the mind which neurons are truly bound together in a gathering. In this manner, neurons supporting gathering codes need to pass on two messages in parallel. To begin with, they need to indicator if the characteristic for which they serve as image is available and, second, they need to demonstrate, in parallel, with which different neurons they are truly working together so as to structure the lucid entire to which they help their characteristic. According to common assertions, they indicate the vicinity of the characteristic for which they stand as image, by expanding the recurrence of their releases, by coming to be more dynamic. The signature for the relatedness of the cells fitting in with a get together is the exact synchronization of unique releases that, much of the time, experience likewise an oscillatory regulation. The needed exactness is in the reach of milliseconds which, in guideline, permits the meaning of relations with the fleeting determination important to reconfigure assemblies at a quick pace. Furthermore, there are to be sure systems that render neurons especially susceptible to synchronous, e.g. coincidental inputs. Since the revelation of boost ward synchronization of oscillatory reactions, numerous research facilities have joined the quest for its practical suggestions. A major essential for those studies is to example at the same time the reactions of no less than two neurons, ideally of whatever number as could be allowed, in light of the fact that generally fleeting relations can't be surveyed. In this connection it is imperative that, up to this point, it has only been recorded from only one neuron at once, and related to the firing of these isolated cells stimuli or behavior have been used in order to identify their functional properties. This blocked dissection of relations and thus the Id of practically bound congregations. Provided that one acknowledges the unpredictability of the framework it is evident that even the multisite recordings have their points of confinement. In spite of all advancement, currently we still are at the exact start of comprehension the mind methodologies underlying higher cognitive functions. Since its revelation, synchronization of oscillatory movement has turned into a hopeful instrument for numerous distinctive capacities. One is the recently said dynamic tying, the adaptable meaning of relations. Nonetheless, synchronization likewise appears included in attentional mechanisms that select motions for further transforming. It seems to serve the readout of data that is archived in the connectivity and it additionally seems, by all accounts, to be utilized to tie diverse sub-frameworks together, for example tactile and engine frameworks. Confirmation additionally shows that synchronization of oscillatory action serves the specific tracking of indicators over the exceptionally interconnected systems of the cerebral cortex. The instrument takes after the tuning of a radio to the recurrence of a certain transmitter, along these lines permitting cerebrum centers to communicate something specific with high selectivity from focus X to focus Y without spreading it to the various others, additionally associated, structures. This particular steering is an exceptionally challenging issue in a quite joined framework and may be understood by synchronization of oscillatory action. There are additionally signs that entrainment into lucid motions assumes a part in the space and support of data in fleeting memory; and, at long last, expansive scale synchronization has all the earmarks of being an essential for indicators to have admittance to cognizant processing.[313]

313 Singer Wolf (2009): The Brain, a Complex Self-organizing System; European Review, Vol. 17, No. 2, pp. 321–329

10.3.2.1 Self-organized learning

On cellular level, the cerebrum and the nervous system are made out of an endless system of interconnected cells called neurons. Neurons might be of numerous sorts and shapes, however eventually they capacity in a comparable way. In case the brain is self-organized, neurons might be the unique components that connect with a specific end goal to shape one global pattern. Neurons hold long and short enlargements called axons and dendrites, separately. The neurons are joined with one another through these amplifications. Dendrites convey electric possibilities towards the cell and axons divert them from the unit. The dendrite of one neuron is associated with the axon of an alternate, with a minor hole in the middle of called the synaptic crevice or synapse. So as to transmit data starting with one neuron then onto the next the neuron transmits an electric signs that ventures down the axon and makes the arrival of neurotransmitters that travel through the synapse the other unit. This sort of cooperation is totally local. The associations around neurons are not static, in any case. Associations are reinforced and debilitated continually, and this sort of face to face time structures the premise of studying. In spite of the fact that they are the essential units that make up the cerebrum, neurons are a long way from straightforward independent from anyone else. Coordination of sensory information throughout studying is all the additionally amazing when we consider that two neurons in, say, the visual and audial cortices of the mind, are divided by billions of interceding neurons and offer no normal synapses that can accelerate simple combination of data. An alternate significant part of studying is the capacity to recognize critical data from immaterial data. Hebbian learning (see 3.3.3.4 Hebb's learning rule) is important here, as well as competitive learning, which has not been introduced yet. Adaptive Resonance Theory ANNs, which utilize competitive learning as well as some different runs to beat the steadiness versatility issue, is likewise talked over. Truth be told, the vast majority of the fruitful models use Hebb’s learning rule, in which neurons utilize just local data. These models have a considerable lot of the attributes required in self-organized systems.

10.3.2.2 Learning with respect to self-organization

The basic surmise utilized within most models of the cerebrum is the presence of Hebbian synaptic versatility, or Hebbian learning. The algorithm, by which neurons change the quality of their associations with different neurons, was at first proposed by Donald Hebb and is as follows:[314]

The point when an axon of neuron A is close enough to excite neuron B repeatedly or tirelessly partakes in firing it, some development prepare or metabolic change happens in one or both cells such that A's productivity, as one of the neurons firing to B, is increased.

The definitive hypothesis has been adjusted and cleared up since its commencement with a specific end goal to include some key attributes. The principal characteristic is that this system is local, as the neurons react to local information through their association with neighboring neurons. This doesn't discount global control signs that may be utilized to control Hebbian versatility in an aggregation of units. There is some proof that neuromodulators might act in this part. A second significant characteristic is that the cooperation requires

314 Brown T.H., Chattarji S. (1995): Hebbian Synaptic Plasticity; The Handbook of Brain Theory and Neural Networks; Massachusetts: MIT Press, pp. 454–459

action on both sides of the synapse. This brings about neurons that follow up on correlated input for reinforcing each other. The last characteristic of this model is that the right timing of the presynaptic and post-synaptic movement of neuron A and neuron B, separately is crucial in verifying how the associations are changed. Hebbian studying is a standout amongst the most critical ideas utilized for unsupervised studying within Neural Networks. One use of systems utilizing this control is within making an associated memory. An acquainted memory is a framework that can review mappings between particular inputs and particular yields. Additionally, it has been demonstrated that the standard execution is optimal in finding associations under the presumption of Gaussian noise. Let assume that there exists an artificial neural network consisting of N_i neurons, each one joined with different neurons and the quality of the connection between neuron i and neuron j is w_{ij} . We already got to know Hebb's learning rule with respect to artificial neural networks, but not everything about it has been said. Just to remember, the weight change within an artificial neural network according to Hebb's rule is calculated as follows:

$$w_i(t+1) = w_i(t) + \Delta w_i \qquad (10\text{-}3)$$

Through Δw_i the necessary change (delta) in the weight of the corresponding connection is calculated at the point in time $t+1$, as the following equation shows:

$$\Delta w_i = \mu x_i y \qquad (10\text{-}4)$$

Considering not only the weight change, but the resulting output, we remember the single layer perceptron equation

$$y = f(\sum_{i=1}^{n} \omega_i \, x_i + \theta) + \epsilon_t \qquad (10\text{-}5)$$

which in principle equals a single neuron output of a multi-layer perceptron:

$$y_i = \sum_j w_{ij} x_j \qquad (10\text{-}6)$$

where y_i is the output of the i^{th} neuron, x_j the output of the presynaptic neuron and w_{ij} the weight between these neurons.

Hebb's learning rule states that the change in weight between neuron i and j is impacted by the learning rate μ, the input gained from the presynaptic neuron j, and the output of neuron i (post-synaptic). The learning rate is normally a modest number that could be diminished through time. This implies that if the two neurons fire concurrently, then the weight of their connection will expand relatively to the firing strength. When substituting the equation for the weight change into the equation for the output of a multi-layer perceptron's single neuron, the result is

$$\Delta w_{ij} = \mu x_j \sum_k w_{ik} x_k = \mu \sum_k w_{ik} x_k \, x_j \qquad (10\text{-}7)$$

It might be demonstrated utilizing this last mathematical statement that the system utilizing this rule has the capacity to find associations in the data set. This straightforward tenet is insufficient, on the other hand, since it is shaky; rehashed utilization can build the weights of the connections without limits, and the execution will corrupt since all the neurons will be saturized to their maximum values. This is to some extent because of the positive feedback in the system: Larger weights results in larger outputs, which will bring about a larger increase of weights. It is likewise biologically unrealistic since there is a limit on the number and

effectiveness of synapses for every neuron. Thus, rules featuring a decay term have come up, which could be implemented utilizing negative feedback. The single neuron output equation continues as before, yet the firing from one cycle is fed back over to the following as inhibition to give:

$$x_j(t+1) = x_j(t) - \sum_{k=1}^{M} w_{kj} y_k \qquad (10\text{-}8)$$

resulting in

$$\Delta w_{ij} = \mu_t y_i x_j(t+1) = \mu_t y_i \left(x_j(t) - \sum_{l=1}^{M} w_{lj} y_l\right) \qquad (10\text{-}9)$$

after substitution, where $x_j(t)$ and $x_j(t+1)$ show the difference of activation times. With these particular changes, the weights converge and it has been indicated that the systems winds up doing primary component analysis (PCA). PCA uncovers the best linear compression of data by uncovering the straight support of an information set that minimizes the mean squared error between the compressed and uncompressed data. An alternate method to overcome the issue of boundlessly increasing connection weights is to renormalize the sum synaptic weights of all inputs, so the total input weight becomes a constant. This brings about competition, since an increment in weight from one neuron brings about a decline in the weights of connections with other neurons. There have been numerous varieties of Hebb's learning rule that bring about altogether different designs or functions coming about. Furthermore, and of utmost importance for understanding the learning functions in the human brain, some proof for Hebb's guideline was later discovered in the hippocampus as Long Term Potentiation (LTP). Due to the study of animal brains, it was discovered that certain stimulation of axons in the hippocampus expedite an expansion in the synaptic strength as measured by the post-synaptic reaction, and it was discovered to last anyplace between a couple of hours and a couple of days. LTP may be seen as an extension to Hebb's learning rule and states that if a weak and a strong input act on a synapse simultaneously, the weak synaptic connection gets stronger. Likewise, if a second weak input exists, which is yet active while the strong synapse is active, unlike the first week input, it won't be strengthened.

10.3.2.2.1 Competitive learning

An algorithmic execution of Hebb's learning rule in artificial neural networks is called competitive learning, and it alludes to a group of calculations that utilize a competition between lateral neurons throughout learning. Accordingly, the error function itself works as feedback signal that educates the system of the course in which changes are needed. As of late, intense studying has appropriated impressive consideration, not only because of its biological plausibility. Normalization of the aggregate of inputs to a neuron, as implemented in Hebb's learning rule is one basic type of that. In an ordinary competitive artificial neural network, neurons in every layer are joined with a layer above as in standard artificial neural networks; however moreover, there are parallel (lateral) connections between neurons in the same layer which make the competition. Competitive learning incorporates a wide mixture of algorithms performing diverse tasks, for example clustering or classification, and is likewise alluded to as self-organized or unsupervised learning. This shows it rather than supervised learning, where the network appropriates sentiment as target outputs (see also 3.3.2.1 Supervised and unsupervised learning). In unsupervised learning, the learner chooses which parts of the input indicate structure to catch in the output. Basically, the learner fabricates a probabilistic model of data and utilizes this to create a recognition distribution, given a particular example of the input. Reaction in competitive artificial neural networks happens at two levels:

1. The competitive artificial neural uses feedback, as parallel (lateral) inhibitory connections and self-excitation, to pick the competition's winning neuron.
2. The victor's weight vector is changed to minimize the error between the input data and the weights.

Competitive learning algorithms utilize rivalry through lateral connections between neurons in the same layer. Again, there are two approaches we can differ between:

1. Hard competition results in a final activity of a single neuron – only the winning one.
2. Soft competition does not reduce neural activity of all losing neurons down to zero, as practical applications have shown that hard competition results in neurons that never win a competition. This is a problem as such neurons are still required for ensuring the network's functionality, although they are not used. Thus, they must be considered to be probably hazardous for the network's configuration. An alternate issue with hard completive learning is that distinctive arbitrary initializations might expedite broadly varying outputs, as local changes within the network may be unable to get it out of the local minimum in which training has started. All of these issues can be overcome by the application of soft competition approaches.

The biological conceivability of competitive learning might be deduced from the way that the neurons in competitive neural networks improve into feature-sensitive indicators, which we currently know do also exist in biological brains and may thus be related to what we perceive as mind. It has also been shown that inhibition plays a major role regarding orientation in the visual and sensory cortices.[315,316,317]

10.3.2.2.2 Competitive learning in artificial neural networks

Competitive learning artificial neural networks embody the feed forward excitatory network and the lateral inhibitory network, where the former one normally is used for implementing an excitatory Hebb's learning rule and the latter one an inhibitory one. During learning, the network selects winning neurons due to a competitive learning rule, and depending from the rule, the winner may receive the largest input and all other neurons nothing (what would be the case with hard competition). Let assume a neuron featuring lateral positive (excitatory weighted) connections (the neurons feature self-connections – see also 3.3.2.8 Recurrent artificial neural network) and featuring negative (inhibitory weighted) connections to the other neurons within the same layer is presented a vector x:

$$y = \sum_j w_i x_j \qquad \textit{(10-10)}$$

Each neuron then computes the weighted sum y of this vector‘s inputs. Some other neuron z may feature a value of y bigger than any possible in the layer. It is currently asserted that, if the network activation h, is permitted to develop by making utilization of the lateral connections, then neuron z will improve a maximal value while the others get decreased. The time

315 Merzenich R.J. et al. (1988): Cortical representation plasticity; Neurobiology of Neocortex; New York: Wiley; pp. 41–68

316 von der Malsburg Christoph (1973): Self-organization of orientation sensitive cells in the striate cortex; Kybernetik, 14:85–100

317 Ranganathan Ananth, Zsolt Kira: Self-Organization in Artificial Intelligence and the Brain; Atlanta: Georgia Institute of Technology

advancement of the neuron is normally represented by a comparison which confirms the rate of progress of the activation. This needs to incorporate the input from the lateral connections and also the external input given by y.

$$\frac{dh}{dt} = c_y y + c_s S - c_h h \qquad (10\text{-}11)$$

In the equation c_y and c_h represent the positive (excitatory) constants and c_s a negative (inhibitory) one, working on the weighted input sum of the lateral connections S. h represents the neuron activation threshold (not the activation function). Observation suggests that the neuron z with the most significant excitation y from the input has its activation increased straightforwardly by this and by implication through the self-excitatory connection. This brings about restraint of the neighboring neurons, whose inhibition of z thus, is then further decreased. This process is continued until stability occurs. There is thusly a competition for activation over the layer and the system is said to advance by means of competitive dynamics. Throughout the learning stage, the network is provided a training set of inputs, and every input data set is represented by a vector. As has already been described, the objective for the ANN is to find a weight vector setup that minimizes the network's total error with respect to the training set and by the application of a learning algorithm. Keeping in mind the end goal to attain this objective, the weight vectors must be changed so they match the training set. The neuron z with the closest vector is that which gives the most input data excitation y since this is simply the dot product of the weight and input vectors. The weight vector of neuron z may be arranged all the more nearly with the input if a change is made consistent with the following equation:

$$\Delta w = \alpha(x - w) \qquad (10\text{-}12)$$

Competitive dynamics is helpful in figuring out the neuron z, which is determined by hard competition. Assuming that o is the output of a neuron, where o is zero for all nodes with the exception of z, then the change in weights for every neuron is made consistent with

$$\Delta w = \alpha(x - w)o \qquad (10\text{-}13)$$

Summing up, the learning algorithm is as follows

Start

1. Create initial ANN and randomize weights.
2. ***Repeat***
   a) ***For each input vector***
      i. Calculate $y = \sum_j w_i x_j$
      ii. Update the ANN according to

$$\frac{dh}{dt} = c_y y + c_s S - c_h h$$

3. ***Until criteria are reached***

End

Algorithm 24 – Competitive training

Breakdown:

y: weighted sum y of a vector's inputs
w_i: the i^{th} weight from a neuron j
c_y, c_h: positive (excitatory) constants
c_s: negative (inhibitory) constant

From the learning algorithm it becomes obvious that the weight vectors are learned unsupervised or in a self-organized manner, as the network is not taught the nature of the input.

10.3.2.3 Adaptive Resonance Theory

Most training algorithms, including competitive learning and SOFMs, work just on static input data. In the event that a system is prepared on a set of input vectors, it is capable of classifying the input data accurately just if it is not dynamic. If the input data is self-organizing and dynamic, the exactness of a network diminishes quickly since the fixed weights anticipate the system from its adaptation to the dynamic environment resulting in networks being not plastic. To overcome this issue, the network might be retrained on another set of data vectors, so it is able to adjust to any progressions in the input data; however, this makes a fast decline of exactness by which it classifies the old inputs, since the old data is lost. This is what has been explained as the stability-plasticity dilemma at 3.3.4 Stability-plasticity dilemma. Amongst others, adaptive resonance theory (ART)-networks have been developed explicitly to tackle this issue, as it relies on an incremental clustering algorithm by which it is able to self-organize progressively and to produce stable recognition while learning input patterns past those initially stored (within the weight matrix). ART consolidates feedback both at the level of the competitive network and between different modules at the network level, as both inhibitory and excitatory connections are utilized to attain learning success. The least complex ART ANN is a vector classifier accepting a vector as input data and classifying it into a category hinging upon the stored pattern it most closely corresponds. When a pattern is discovered, it is altered to take after the input vector. Provided that the input vector does not match any stored pattern inside a certain tolerance, then another category is made by storing another example comparable to the input vector. Hence, no stored example is ever altered unless it matches the input vector inside a certain tolerance. Such ANNs have been successfully applied within a wide range of areas such as for the classification of ECG patterns, for applications requiring associative memory and semantic information processing, or even the building of expert systems, which requires concept discovery – a field only such kind of ANNs can be successfully used within.[318,319,320,321,322] ART1 is the simplest ART network - learning unsupervised and capable of processing only binary values. It commonly

318 Barro S. et. al. (1998): Classifying multichannel ECG patterns with an adaptive neural network; IEEE Engineering in Medicine and Biology Magazine; 17(1):45–55
319 Grossberg S. (1976): Adaptive pattern classification and universal recording, 1: Parallel development and coding of neural feature detectors; Biological Cybernetics; 23:121–134
320 Tan A. (1997): Cascade artmap: Integrating neural computation and symbolic knowledge processing; IEEE Transactions on Neural Networks; 8(2):237–250
321 Williamson J.R. (1996): Gaussian artmap: A neural network for fast incremental learning of noisy multidimensional maps; Neural Networks; 9(5):881–897
322 Ranganathan Ananth, Zsolt Kira: Self-Organization in Artificial Intelligence and the Brain; Atlanta: Georgia Institute of Technology

comprises of an orienting subsystem, an attentional subsystem (see Figure 75 - ART1 Structure), a vigilance parameter and as well as a reset module. The vigilance parameter has respectable impact on the system. High vigilance processes more detailed remembrances, e.g. fine classes, while lower vigilance brings about additional general remembrances. The ART1 attentional subsystem features two competitive networks, a comparison field layer F1 and the recognition field layer F2, two control gains Gain1 and Gain2 and two short term memory (STM) stages F1 and F2. Long term memory (LTM) follow between F1 and F2 multiply the signal in these pathways.

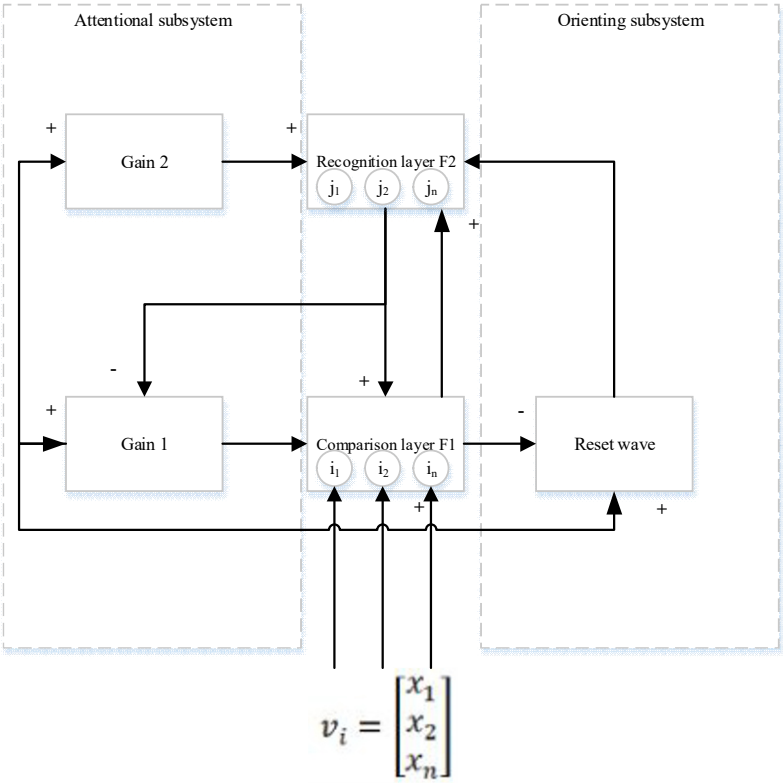

Figure 75 - ART1 Structure

Gains control empowers F1 and F2 to recognize the present phases of the running cycle. The STM reset wave inhibits animated F2 cells, if confounds between the bottom-up and top-down indicators happen at F1. The comparison layer gains the twofold outside input passing it to the recognition layer answerable for matching it to a classification category. This come about is passed again to the comparison layer to discover if the category matches that of the input vector. In the event that there is a match, another input vector is perused and the cycle begins once more. Assuming that there is a mismatch, the orienting system is responsible for

inhibiting the past category to get another category match in the recognition layer. The two gains control the movement of the recognition and the comparison layer separately. The orienting subsystem creates a reset wave to F2 when the bottom-up input pattern and top-down model pattern at F1, consistent with the vigilance criterion. The reset wave specifically and enduringly inhibits the animated F2 cell until the present is stopped. Offset of the input pattern ends its handling at F1 and triggers offset of Gain2. Gain2 offset causes quick rot of STM at F2 and consequently gets ready F2 to encode the following input pattern without bias.[323] Thus, adaptive resonance theory comprises a type of ANN as well as a learning approach:

Start

1. Initialize w_{ij}^b and w_{ij}^f:
   a) For each w_{ij}^b
   $$w_{ij}^b(0) = 1$$
   b) For each w_{ij}^f
   $$w_{ij}^f(0) = \frac{1}{1+N}$$
2. ***Repeat***
   a) For each value x_i of the input pattern x calculate y_i:
   $$y_i = \sum_{j=1}^{N} w_{ij(t)}^f x_i$$
   b) Select winning neuron k
   c) If
   $$\frac{\boldsymbol{w_{k(t)}^b x}}{\boldsymbol{xx}} > \rho$$
   go to step $2.f)$
   d) Disable k from further activity and go to step $2.b)$
   e) For each l, where $0 \leq l < N$
   $$w_{kl(t+1)}^b = w_{kl(t)}^b x_l$$
   $$w_{ijl(t+1)}^f = \frac{w_{kl(t)}^b x_l}{\frac{1}{2} + \sum_{i=1}^{N} w_{ki(t)}^b x_i}$$
   f) Reenable all neurons in F2
3. ***Until criteria are reached***

End

Algorithm 25 – Adaptive resonance theory

Breakdown:

w_{ij}^b: binary-valued backward long term memory weights

323 Mbaitiga Zacharie (2007): Adaptive Resonance Theory 1 (ART1) Neural Network Based Horizontal and Vertical Classification of 0-9 Digits Recognition; Okinawa National College of Technology

w_{ij}^{f}: continuous-valued forward long term memory weights
N: number of neurons in F1
y_i: activation values

10.3.3 The transition to the human brain

Basically, two kinds of experiments in neuroscience have proofed to be of value in the understanding of how brain functionality corresponds with the body:

- Uncovering the relations between segments of the brain and parts of the body they are responsible for.
- Uncovering the relations between groups of neurons and the type of sensory input they respond to.

It has been shown that the latter correspondence is spatial, implying that groups of neurons managing comparable characteristics are likewise placed close to one another in the brain. Spatial organization of neurons in a neural network is additionally attractive in pattern classification issues. Another important development has been the self-organizing feature map (see 3.3.2.9.4 Self-organizing feature map), a topographic map learning with an algorithm that generates spatially arranged neural networks. A percentage of the most striking proof for self-organization in the brain is the presence of different tonotopic, somatosensory, and retinotopic maps, in which columns or sheets of neurons reacting to similar features of the sensory input, are placed close to one another. As such, neurons organize themselves in a way that a topographic map of the skin on the body, sound frequencies, or visual features, is created. Thus, there exist maps for body movement and the visual and auditory fields. This sort of emergent pattern is not stable or fixed, and changes depending from the environment the subject (e.g. a monkey) is subjected to; an example are the somatosensory maps of monkeys who had a finger removed changed in such a path, to the point that the zones for the different fingers ventured into the district that was delicate to the excised finger. It is not only that the local adjacent areas expand, but many regions' size increases as these changes propagate through the whole map – something that even occurs in adult brains. This could be verified in numerous experiments, and what this suggests is that a sort of pre-defined procedure model or function does not exist or is not being utilized, which results in the self-organization-assumption. Moreover, numerous models utilizing Hebb's learning rule as a foundation have been developed, which can represent plenty of the features in these maps.

Numerous research efforts have targeted the visual cortex, thus the corresponding sensory maps are those we know most about. The primary visual cortex is a topographic map in which neighboring neurons react to neighboring areas of the visual field, and different neurons react stronger to specific features of the input, e.g. ocularity, size, or temporality. By and large, neurons with the same feature characteristics tend to group themselves together into columns:

- Assuming that electrodes are moved vertically through the thickness of the cortex, it has been discovered that most neurons feature the same orientation and ocularity inclination.
- Assuming that the electrodes are moved tangentially through the cortex, the neurons first react to left eye inputs, then both, then right eye, then both, then left eye, and so on.
- Provided that the electrodes are moved tangentially in the orthogonal direction, it was discovered that the neurons are selective for vertical inputs, then inclining, then flat, and so on.

Figure 76 - Monkey striate cortex recording, which has been acquired by optical recording in macaque monkey striate cortex, shows the arrangement of orientation domains and their association with ocular dominance segment boundaries (white lines); the resulting selectivity patterns can be identified.[324]

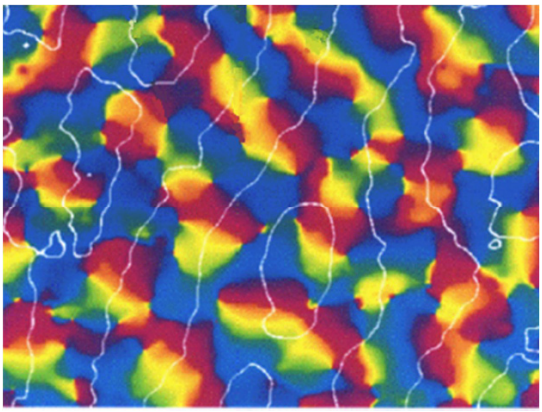

Figure 76 - Monkey striate cortex recording[325,326]

Every color shows a field of orientation preferences, while the white lines border the visual dominance columns. The maps are not superbly constant; instead there are points around which the orientation preference changes ceaselessly and circularly. These are called singularities or pinwheels. Between neighboring singularities there are areas where the orientation preference changes gradually and persistently in a circular manner, called linear zones. There are other regions between singularities where local minima of orientation preference in orthogonal directions exist, called saddle points. It is likewise worth noting that ocularity and orientation patterns are not independent – Figure 76 - Monkey striate cortex recording shows that the lines speaking to areas of diverse ocularity are to some degree orthogonal to the lengthened circle shapes, which are regions of similar orientation. The neuronal specialization that happens is not fixed or hardwired – it has been discovered that even an adult cortex can experience reorganization of the connections when encountering sensory or cortical manipulations (i.e. lesions). Be that as it may, some specialization does happen before organized visual stimuli are encountered, and in a few animals it even may exists before birth.

---

324 Ranganathan Ananth, Zsolt Kira: Self-Organization in Artificial Intelligence and the Brain; Atlanta: Georgia Institute of Technology

325 Swindale N. V. (1996): The development of topography in the visual cortex: a review of models; Network 7:161–247 (reprinted with permission from Swindale N. V.)

326 Ranganathan Ananth, Zsolt Kira: Self-Organization in Artificial Intelligence and the Brain; Atlanta: Georgia Institute of Technology

Later presentation of organized visual input skews the circulation of orientation selectivity, and referring back to computational intelligence, it has been showed that a feed-forward artificial neural network utilizing Hebb's learning rule could advance orientation preference comparative to that seen in the visual cortex even given unstructured visual input (e.g. spontaneously happening retinal movement, which some have even inferred could be genetically-encoded and internally generated input patterns). Numerous models leveraging self-organization have been proposed for explaining the manifold patterns occurring in the visual cortex, and most of the models share similar concepts, i.e. leverage Hebbian synapses and normalize synapse strength. Connections may be fixed or correlated/ show locally patterned activity; the former between cortical neurons, which react inhibitory at large distances and excitatory locally, and the latter in the cortical neuron's afferent nerves.[327,328,329,330] The Von der Malsburg-model[331] (containing inhibitory and excitatory cortical cells, as well as retinal cells) leveraged a competitive Hebb's learning rule (once the strength to one neuron increases, the strength to another decreases) first, yet in a less difficult way than portrayed in the recent past. Normalization keeps the summed incoming connections to a neuron constant, which lets neurons react to correlated inputs, e.g. in the event that two inputs are enacted by correlated neural activity, this results in a common strengthening of their connections since they both cooperate in order to activate the target cell. The two cell types of the model are used to represent a network featuring two layers, despite the fact that one layer is simply the input given to the system.

Something that is thought to happen in brains is that a constant number of inhibitory and excitatory cells are wired together unchangeably, so that the inhibitory connections go somewhat further than excitatory ones, and every excitatory cell features connections to every one of the retinal input neurons, as well as every retinal neuron has a connection to each excitatory cell. At the beginning, the weights of the excitatory cells are randomized, and the non-linear differential equation models the response changes in the cell:

$$\frac{dH_k(t)}{dt} = -\alpha_k H_k(t) + \sum_{l=1}^{N} p_{lk} H_l^*(t) + \sum_{i=l}^{M} s_{ik} A_i^*(t) \qquad (10\text{-}14)$$

where $-\alpha_k H_k(t)$ in general is the decay of a neuron over time, α_k the decay constant and $H_k(t)$ the neural response at time t. $\sum_{l=1}^{N} p_{lk} H_l^*(t)$ is the description of the excitation and inhibition the neuron receives from other lateral neurons, where p_{lk} is the connection strength between cell l and cell k and $H_l^*(t)$ is the value of $H_l(t)$ after the application of a threshold function. $\sum_{i=l}^{M} s_{ik} A_i^*(t)$ describes the effect of the retinal cells, where s_{ik} is the strength of the connection and $A_i^*(t)$ is the stimulus strength of retinal cell i after applying a threshold function to it. For changing the weights of the input connections both between cortical and retinal cells the following update rule is used:

---

327 Ranganathan Ananth, Zsolt Kira: Self-Organization in Artificial Intelligence and the Brain; Atlanta: Georgia Institute of Technology

328 Bednar J.A., Miikkulainen R. (2003): Learning Innate Face Preferences; Neural Computation, 15(7)

329 Erwin E. et al. (1995): Models of orientation and ocular dominance columns in the visual cortex: A critical comparison; Neural Computation, 7:425– 468

330 Swindale N.V. (1996): The development of topography in the visual cortex: a review of models; Network 7:161–247

331 von der Malsburg Christoph (1973): Self-organization of orientation sensitive cells in the striate cortex; Kybernetik, 14:85–100

$$\Delta s_{ik} \propto A_i^* H_k \qquad (10\text{-}15)$$

Normalization of the summed synaptic strengths going into any single cortical cell happens by multiplication with a factor that is proportional to

$$\frac{1}{\sum_i s_{ik}} \qquad (10\text{-}16)$$

It is important that the weights between horizontal cells p_{lk} remain unchanged. This model was the first leveraging local connectivity featuring short distance excitatory connections and long distance inhibitory connections cell sheets, as well as the first model to bring out this pattern. [332] This is where models applying the SOFM described at 3.3.2.9.4 Self-organizing feature map feature major advantages.

10.3.3.1 Laterally interconnected synergetically self-organizing maps

It is critical to recognize that lateral and afferent connections exist in neuronal structures, as well as that lateral connections are the inhibitory and excitatory connections around neurons, and afferent connections are connections between various layers. As discussed beforehand, lateral connections are thought to unchanging, and excitatory connections are shorter than inhibitory ones. Receptive Field Laterally Interconnected Synergetically Self-Organizing Maps (RF-LISSOM) is a model in which both lateral and afferent connections emerge together and may represent orientation, visual predominance, and measure selectivity columns as well as low-level phenomena. [333] Lissom self-organizing computational models intend to imitate the neural structure of the visual cortex of human beings, where the key proposal is that the cortex organizes itself utilizing general learning rules to catch correspondences in both visual inputs and inside produced wellsprings of enactment. The learning rules comprise of basic changes in the strengths of feed-forward and horizontal connections between neurons. Furthermore, the model has been indicated to show large portions of the same offers discovered in human and test animal cortex, and as basic idea for these models served the self-organizing feature maps algorithm broadly utilized for information visualization. RF-LISSOM amplifies SOFMs to be more effective and all the more biologically realistic by utilizing Hebb's learning rule and by incorporating parallel connections between neurons. There have been made some extensions to the initial model, as

- RF-LISSOM displays just the essential visual cortex (V1), whereas
- CRF- LISSOM incorporates a model for the information processing handling within the retina and the sidelong geniculate core, which permits the model to work with regular picture stimuli.
- HLISSOM further amplifies CRF- LISSOM to incorporate cortical regions past V1, permitting it to clarify observation of articles, e.g. confronts, and low-level picture characteristics, and thus is a model of the mammalian visual system, not just V1.

All of these models make use of layers of columns of interconnected neurons, and through afferent connections every neuron gets input from an open surface and, additionally, features

332 Ranganathan Ananth, Zsolt Kira: Self-Organization in Artificial Intelligence and the Brain; Atlanta: Georgia Institute of Technology

333 Ranganathan Ananth, Zsolt Kira: Self-Organization in Artificial Intelligence and the Brain; Atlanta: Georgia Institute of Technology

reciprocal lateral excitatory and inhibitory connections (again, lateral inhibitory connections run for more extended distances), which are similar to SOFMs. Every neuron in the $N * N$ network gets input from a receptive field of size $s * s$, s being half the measure of the retina, thus the receptive surface. The weights of both afferent and parallel connections are positive, randomly initialized, and weight change happens by application of Hebb's learning rules. The initial response of a neuron is determined by

$$n_{ij} = \sigma\left(\sum_{r_1,r_2} \xi_{r_1,r_2}\, \mu_{ij,r_1,r_2}\right) \qquad (10\text{-}17)$$

where n_{ij} is the response of neuron (i,j), ξ_{r_1,r_2} represents the activation level of the retinal receptor (r_1,r_2) within the receptive field of the neuron, and μ_{ij,r_1,r_2} describes the strength of the afferent neural connection. σ represents a gradually linear approximation of the sigmoidal activation function, and at each iteration, the neural response is modified by

$$n_{ij} = \sigma\left(\sum_{r_1,r_2} \xi_{r_1,r_2}\, \mu_{ij,r_1,r_2} + \gamma_e \sum_{k,l} E_{ij,kl}\, \eta_{kl}(t-\delta t) - \gamma_i \sum_{k,l} I_{ij,kl}\, \eta_{kl}(t-\delta t)\right) \qquad (10\text{-}18)$$

where the first expression represents the afferent connection, the second term the impact of the excitatory connections, and the third term the impact of the inhibitory connections. γ_e and γ_i are simply scaling variables verifying what amount of parallel excitatory and inhibitory interaction is wanted. Right away, the action on the responsive field is spread out over the topographic map, yet after a couple of emphases it will stabilize in a patch of activity. After this, both afferent and lateral weights of the connections are altered by

$$w_{ij,mn}(t+1) = \frac{w_{ij,mn}(t) + \alpha \eta_{ij} x_{mn}}{\sum_{mn}[w_{ij,mn}(t) + \alpha \eta_{ij} x_{mn}]} \qquad (10\text{-}19)$$

which is essentially Hebb's learning rule, but normalized, as both the inhibitory and excitatory lateral connections are reinforced by correlated action, and since correlated activity is extraordinary around neurons far from one another the long range connections in the long run come to be weak, which are physically pruned by the system, occasionally in the event that they come to be too weak, implying that such connections disappear at an early development stage of the animal. The radius of lateral excitatory reaction times is diminished until it considers just closest neighbors, although it is large at the beginning. As the system works through thousands of training iterations, the activity-radius of the pattern decreases and finally converges to a very small area. In spite, or maybe of the fact that the model is more than complex than most of the others, an extensive variety of patterns and relationships occurring in animals can be explained therewith.[334,335] There has been an incredible arrangement of neuroscientific and hypothetical examination regarding the systems that empower the mind, and self-organization is a feasible hopeful and has been solidly connected to model the mind.

334 Ranganathan Ananth, Zsolt Kira: Self-Organization in Artificial Intelligence and the Brain; Atlanta: Georgia Institute of Technology

335 Miikkulainen Risto et al. (1997): Selforganization, plasticity, and low-level visual phenomena in a laterally connected map model of the primary visual cortex; Perceptual Learning of Psychology of Learning and Motivation, vol. 36; San Diego: Academic Press; pp. 257–308.

10.3.3.2 The pruning neocortex

We know that within the human brain there exist around 10^{11} neurons, each featuring around 7,000 synaptic connections to other neurons. Current estimates claim that a three year old child's brain has 10^{15} synaptic connections, compared to an adult brain with 10^{14} to $5 * 10^{14}$ synapses.[336] This can simply be explained when closely examining a learning neocortex. The information coded into our DNA cannot account for all the complex patterns mapped in our neocortex, as well as for all structural differences. What I am up to is that according to our current knowledge, the 2.9 billion base pairs of the haploid human genome correspond to a maximum of about 725 megabytes of data, since every base pair can be coded by 2 bits. Since individual genomes vary by less than 1% from each other, they can be compressed losslessly to roughly 4 megabytes.[337] Ray Kurzweil even argues that the amount of unique information in the genome after lossless compression as applied to the brain is about $2.5 * 10^7$ bytes.[338] As a remark here: haploidy is spoken of if the genome of a cell or a prokaryote (such as bacterium) is present only once, thus each allele occurs in a single form. The haploid set of chromosomes of a human germ cell has 23 different chromosomes (22 autosomes and 1 gonosome). Back to the learning neocortex, this relatively small amount of information encoded into the human genome cannot account for the information that would be required for building such a complex neural structure. Therefore, the assumption is that the initial neocortex is made up of numerous copies of smaller structures, and nearly fully connected. During learning, some connections are strengthened and weakened, such as we also do within ANN learning with Hebb's learning rule. However, by solely applying Hebb's rule unused connections or even neurons are still included into the network, which is not how our brain organizes itself. Unused connections are eliminated, or pruned away finally developing the neocortex towards the complex structure it has in an adult human being (which does not mean that it is static then). This structural modification is also something that we apply within neural network learning, additionally to structural evolution paradigms like the one introduced at 5.3 Structural evolution. The modification of the artificial neural network architecture is based on an analysis providing information on how much each neuron contributes to the final result produced by the network. If Hebb's learning has resulted in a massive decrease of the significance of a synapse for the overall output, it will be pruned. If all connections to or from a neuron have been pruned, the overall neuron will be pruned as well. Pruning does may increase the effectiveness of an ANN, but it increases efficiency in processing for sure, which is especially important when needing to process numerous ANNs hierarchically, as it happens within the neocortex – ANNs are allowed to process more information in a given timeframe. This is not the only reason for doing this; the effectiveness of an ANN may be improved in the way that if a weak connection is pruned, the network's accuracy may be improved. On the other hand, it is also possible to decrease the performance by pruning the wrong synaptic connections. Basically, there are two pruning approaches,

– selective and
– incremental pruning.

---

336 Drachman D. (2005): Do we have brain to spare?; Neurology 64 (12): 2004–5

337 Stackoverflow (2012): How much memory would be required to store human DNA? [2013-11-17]; URL: http://stackoverflow.com/questions/8954571/how-much-memory-would-be-required-to-store-human-dna

338 Kurzweil Ray (2012): How to create a mind; London: Penguin Books

Selective pruning starts with a neuron- and synapse-rich ANN and prunes away connections and neurons as long as the error rate remains within the allowed parameters. Incremental pruning is the converse approach, as it starts with an ANN featuring little neurons and neural connections (and most likely with a high error rate) and ends when the number of hidden neurons and connections added results in an error rate within the allowed parameters. The detailed algorithms of both methods will not be explained here, as both can be derived from the structured evolution discussed in the training algorithms section of this elaboration.

10.3.3.2.1 Incremental pruning

The foundation for incremental pruning is an ANN that has not experienced supervised training before. A suitable training algorithm trains the ANN for a pre-defined number of iterations and then matches the result with the expected allowed error, such as the RMSE. The number of iterations can be defined either by a threshold that has to be reached by the number of training iterations, or by a threshold value that is continuously compared with the number of iterations that have been conducted without a change in the ANN error. The challenge here is that the training algorithm has to be applied to each new structure to verify if the structure at point in time t outperforms the one at point in time $t-1$, which means that the processing time for completely applying incremental pruning may become very high, especially when taking into account that the ANN structures required for re-modelling the higher cognitive brain functions of the neocortex are very complex.

10.3.3.2.2 Selective pruning

On the contrary to incremental pruning, the selective pruning algorithm starts with a trained ANN, removes connections and neurons one by one and compares the resulting error value with the allowed error parameters. If the error value does not exceed the allowed maximally error value, the object (neuron or synapse) is pruned. This is repeated as long as the error threshold is not exceeded. Compared to incremental pruning, the algorithm is fast, as it does not require training after each structural modification, however, it also results in more inaccurate ANNs, as with incremental pruning the training algorithm adapts the weight matrix related to the current structure.

10.3.3.2.3 Pruning and quantum artificial neural networks

As incremental pruning is the more accurate way of modifying a structure of an ANN, a solution for dealing with the high processing requirements is required. Especially when considering that the human neocortex is organized hierarchically, thus is organized in lists of ANNs, each entry referencing to lower level ANNs, it is required to conduct training as fast as possible, as not only spatial, but also temporal resolution of events has to be taken into account (events as an input from real-world cannot be compared to classical data mining tasks such as predicting weather, in which the time required for training is more or less obsolete). Thus, quantum linear superpositions of a number of different ANN structures could be created at once, and each of them be queried for the optimal result (see 10.5 Quantum physics and the artificial brain) for determining the optimal solution.

10.3.4 Arguments for self-organization in artificial neural systems

Consequently, in spite of the fact that none of the other organizing standards can represent the complete cerebrum, this recommends that not just one instrument is included. Since the cerebrum as well as the mind itself is such a complex system, it may be the case that the sum of the different organizing routines is utilized, incorporating self-organization. Luckily, developments in engineering utilized by neuroscientists may permit a more excellent comprehension of these methodologies sometime to come. Self-organized AI-systems were intended to explicitly fuse systems that are applied within the brain. Notwithstanding, in keeping with the distinctive provision needs, there has been a uniqueness between biology and AI. While feedback mechanisms exist at diverse levels in the ANN structures mentioned beforehand, not the sum of these lead to self-organized behavior. In competitive learning, for instance, feedback happens at both the neuron connection level, as well as throughout adjustment of weights. Feedback through lateral inter-neuronal connections heads the development of a winning neuron exclusively through local communications and constitutes self-organized behavior. Be that as it may, the alteration of weights is finished with the point of driving the weight vectors closer to the input. Here every neuron gains the input vector and consequently, this constitutes global knowledge, or learning success. The progressions in the system are likewise made remembering the last objective to be accomplished, e.g. a matching of input and weight vectors. Additionally, in the ART-1 network talked about formerly, sentiment happens in the F1 and F2 modules that are competitive networks. Common positive criticism is additionally dynamic between the attentional and orientational sub-systems and prompts resonance, which thusly brings about learning. The learning, be that as it may, is not self-organized in that the resonance state is not realized by any self-organized component working at the module level of the network. Kohonen networks realize topological requesting through a competitive network that has a neighborhood capacity connected with weight changes. As in the recent past, self-organization happens at the neuronal level. Also, the network likewise organizes itself topologically in respect to the input through just local cooperation. The weight acclimatization in neighborhood areas accelerates a spatially grouped representation of the input. An outcome of utilizing such a component is, to the point that the correctness of the network builds monotonically both with expanding number of hubs and with more amazing learning knowledge. This is not the situation with, say, back-propagation networks that can experience the ill effects of over-learning, where the network correctness diminishes with all the more learning.

10.4 Mechanisms apart from self-organization

It has been assumed that four different manifestations of organization are

- leader,
- blueprint
- recipe and
- template.[339, 340]

339 Camazine Scott et al. (2001): Selforganization in Biological Systems; Princeton: Princeton University Press

One leader neuron or area of the mind presumably can't stay informed regarding the billions of neurons in the cerebrum, and such a plan might leave even a modest measure of cerebrum harm deadly provided that it was in the ideal spot. This doesn't, be that as it may, discount a various leveled system of administration when taken a gander at from a bigger scale.

10.4.1 Leader

The cerebrum (1.1 Anatomy of the human brain) has been demonstrated to be made up of to some degree specific locales that do specific things well, which has been found through imaging of the mind to see which parts are dynamic throughout particular undertakings. One well known sample is that the hippocampus (see 1.1.2.2 Hippocampus) has been discovered to be included in the working of memory. Taking a gander at it from this view does not discount self-organization, since the parts themselves are made of an incredible arrangement of neurons that are left to be clarified for. It is totally conceivable that the self-organizing process brings about the specialization of different parts of the mind; that could be the pattern that develops. There is some priority for this in the visual cortex, since distinctive neurons advance specializations to diverse characteristics of the input.

10.4.2 Blueprint

DNA can likewise perhaps go about as a blueprint for the mind, yet it doesn't have enough data to record for the greater part of the connections in the cerebrum from a data theoretic perspective. Besides, it is well realized that everybody has an alternate mind particularly when it is produced or encountering distinctive situations. This is accurate indeed, for indistinguishable twins who have the same DNA. The data held in DNA is unrealistic to hold possibilities for all conceivable situations the mind may experience.

10.4.3 Recipe

An alternate strategy, a recipe, is additionally impossible for the same explanations, as it might experience the same data measure issue.

10.4.4 Template

It is not clear at the outset if the cerebrum could be structured from an arrangement. It can be assumed that nature itself domain decidedly guides the pattern development. The different somatosensory maps, for example those discovered in the visual cortex reorganize themselves relying upon the visual input it is given all around its lifetime, particularly throughout advancement however indeed, throughout adulthood. The last pattern, then again, is unquestionably not encoded or specified in the life form itself. It is likewise possibly precisely the same shape or estimate, in spite of the fact that the relative sizes may have some solid correspondence. This is unquestionably accurate for different cases of self-organization since the environment ants live in shapes the pattern; however is not found inside the organism. Thus, this type of impact does not appear to fit this particular meaning of a template.

---

340 Ranganathan Ananth, Zsolt Kira: Self-Organization in Artificial Intelligence and the Brain; Atlanta: Georgia Institute of Technology

10.5 Quantum physics and the artificial brain

Although it has not been affirmed by research that quantum effects occur in the brain, it is a very reasonable approach to make use of such when constructing an artificial brain. The ORCH OR model suggests that conscious experiences are related to the collapse of a quantum state over a affecting a large area of the brain. Apart from that, incremental pruning can massively benefit from quantum ANNs, as different structural architectures may be created in superpositions, so that each of these can be queried for the optimal result a lot faster than it would take by training each structure in parallel. Why quantum ANNs may also be of use for information processing in artificial intelligence is explained in detail at 10.6.3.5 Quantum pattern recognizers. Via the quantum artificial neural network approach described within this chapter, the research provides an outlook on a possible future of artificial intelligence, as sooner or later quantum computers may replace classical von Neumann machines. The introduced quantum artificial neural network is a classical feed forward ANN making use of quantum mechanical effects, and should help to understand how simple neural processing may be implemented on a quantum computer (although we will later on see that it is most likely not the neurons that function in a quantum linear superposition). To begin with, we define a quantum artificial neural network (QANN) as a physical system whose multiple occurrences (component networks) are trained according to the set of patterns of interest. Thus, it is a physical system instantiated from a set of physical parameters,[341] which is not different from other physical systems.

10.5.1 Quantum artificial neural network

Thitherto, several different prototypes of quantum artificial neural networks have been introduced[342,343,344], some of them very different from the already explained ANN structures like quantum dot[345], lattice[346] or phase shifting ANNs. The one proposed in this elaboration relies on classical feed forward structures, with the difference of the weights being switched into superposition instead of continuously changing them through learning. The introduced quantum ANN makes use of simulated quantum effects for learning. Thus, the ANN is supervised and requires a teacher. For the introduced quantum artificial neural network, especially the simulation of the quantum mechanical theorems and paradigms of

- quantum parallelism,
- entanglement and
- interference

341 Singh Gurwinder (2009): Quantum Neural Network Application for Weather Forecasting; Thapar: Thapar University

342 Altaisky M.V. (2001): Quantum neural network; Technical report; URL: http://xxx.lanl.gov/quantph/0107012.

343 Narayanan Ajit; Menneer Tammy (2000): Quantum artificial neural network architectures and components; Information Sciences, volume 124. 1-4, p. 231–255

344 Behrman Elizabeth C., Niemel Jari, Steck James E., Skinner S. R. (1996): A quantum dot neural network; Proceedings of the 4th Workshop on Physics of Computation, p. 22–24

345 Behrman Elizabeth C., Niemel Jari, Steck James E., Skinner S. R. (1996): A quantum dot neural network; Proceedings of the 4th Workshop on Physics of Computation, p. 22–24

346 Fujita Yukari, Matsui Tetsuo (2002): Quantum gauged neural network: U(1) gauge theory; Technical report; URL: http://xxx.lanl.gov/cond-mat/0207023.

are of interest. Moreover, the simulation of quantum bits for processing the information goes along with the concept of quantum parallelism, as only such may be described by a wave function ψ that exists in Hilbert space. Certainly, all quantum effects the system benefits from have been simulated, as quantum computers with the required architecture are not available at these times. However, despite all the negative effect on performance in terms of computation time this comes along with, the simulation of an environment suitable for processing information with quantum bits has an enormous advantage: the elimination of all irrelevant effects a real world physical system would subject on these bits. In the real world, a quantum computer would use particles for data representation, and in case of a quantum artificial neural network, these particles might represent neurons with all their features, like axons and dendrites (weights) or their thresholds and activation functions. A Qbit could be represented well by two-particle systems as a hydrogen-like ion is one, which additionally to the atomic nucleus consists of one electron. Depending on the distance between nucleus and electron, the for quantum mechanics utmost important Schrödinger equation, a partial differentiation describing the change of a physical system's quantum state over time (eq. (7-24)) can be solved. The orbit of the hydrogen-like ion's electron may vary, depending on the energy levels. Physicists name this the 'existence in different energy levels' (see excursus: quantum numbers). The orbital position of the electron is then used to represent either 0 or 1, or according to Dirac's BraKet notation, either the state $|0\rangle$ or the state $|1\rangle$ and the highest orbital position representing the latter one, the lowest the former one.[347]

10.5.1.1 Structure

The structure of a quantum feed forward artificial neural network does not differ from a normal one, but the concept of linear superposition is one of the differences that can be described graphically (Figure 77 - Quantum artificial neural network).

Figure 77 - Quantum artificial neural network shows that neurons are represented as quantum registers $|i_{11}\rangle, \dots, |i_{mn}\rangle$, where m represents the number of the data sets and n the number of the input neurons, which are presented the input datasets of the actual training data set. Assuming that a QANN shall be used to learn how to solve a problem statement based on input datasets each consisting of three attributes, it would feature three input registers (input neurons in a standard ANN). Further considering the example network, for each input data set the registers $|i_{11}\rangle, \dots, |i_{m3}\rangle$ need to be created. Consequently, the network features $|o_{11}\rangle, \dots, |o_{m2}\rangle$ output registers presenting the calculated output values for its respective input data set. The same holds for the hidden layer, represented by $|h_{11}\rangle, \dots, |h_{m4}\rangle$, whose quantum neuron registers process the calculated, weighted sum of the actual training input coming from $|i_{11}\rangle, \dots, |i_{m3}\rangle$. Furthermore, the superpositions of the weight vectors are important to mention here, as these hold every possible weight value they can represent at once, independent from the training data (detailed explanations follow in 10.5.1.6 Processing).

347 Neukart Florian, Moraru Sorin-Aurel (2012): On Quantum Computers and Artificial Neural Networks. Journal of Signal Processing Research, vol. 2, 1, 2013

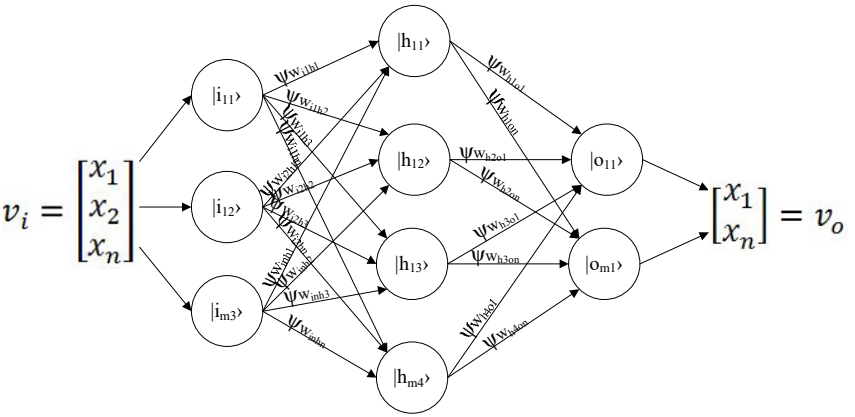

Figure 77 - Quantum artificial neural network

Table 2 – (Quantum) artificial neural network feature comparison, which is based on the comparison Ezhov and Ventura provide in their work,[348] is required for understanding the differences between classical and quantum ANNs.

Table 2 – (Quantum) artificial neural network feature comparison

Feature	ANN		QANN	
Neuron encoding	Definite bit state	$x_i \in \{0,1\}$	Probable Qbit state	$\|x\rangle = \alpha_1\|0\rangle + \alpha_2\|1\rangle$
Neuron connections	Weighted connections	$\{\omega_{ij}\}_{ij}^{n-1}$	Weighted connections	$\{\psi\omega_{ij}\}_{ij}^{n-1}$
			Entanglement	$\|x_0 x_1 \dots x_{n-1}\rangle$
Learning	Rule	e.g. $\Delta w_{ij}(t+1) = \mu\delta_j x_i + \alpha\Delta w_{ij}(t)$	Superposition of entangled Qbits	$\sum_{i=1}^{n} \alpha_1 \|x_0^i \dots x_{n-1}^i\rangle$
Desired solution determination	Cost function	e.g. $n = min\left(\sqrt[2]{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)$	Grover's search for optimal solution in superposition via unitary transformation	$U: \psi \longrightarrow \psi'$ with an oracle like e.g. $\|p\rangle \geq n * m * p$
Result		n	Decoherence	$\sum_{i=1} \alpha_1 \|x^i\rangle \Rightarrow \|x^k\rangle$

348 Ezhov Alexandr A., Ventura Dan (-): Quantum neural networks, BSTU Laboratory of Artificial Neural Networks

10.5.1.2 Quantum bits

The introduced QANN relies on double values, so theoretically an 8 Qbytes or 64 Qbits-architecture needs to be simulated for being able to process all the information, what would be the optimum. For the needs of processing this seems to be oversized, as according to physicist David Deutsch, quantum parallelism allows a quantum computer to work on a million computations at once, while an ordinary PC works on one. A 30-Qbit quantum computer would equal the processing power of a conventional computer that could run at 10 teraflops, where today's typical desktop computers run at speeds measured in gigaflops.[349] The probability distribution of a Qbit either being $|0\rangle$ or $|1\rangle$ after the collapse of $|\psi\rangle$ caused by measurement is, as already indicated, determined by complex in Hilbert space and not by real numbers. Therefore, for simulating one Qbit two floating point numbers are required. Theoretically, for state machines, like auto-associative artificial neural networks are, a number of two architecture-Qbits would suffice, as their calculation is not based on floating point numbers, but on signed 2bit-values (-1, 0, 1). Anyway, as the quantum ANN only exists as a simulation, the optimal architecture, which is a 64-bit one, has been simulated. Moreover, the number of used Qbits depends from the problem to be solved. Let assume, the QANN from Figure 77 - Quantum artificial neural network serves as model, then two registers have to be created, one representing the inputs x and one representing the outputs $f(x)$. Besides, the process of carrying out the function $f(x)$ would need lots of additional Qbits, which play an important part in the entanglement of the input and output register a quantum computer makes use of.

10.5.1.3 Superposition

A quantum artificial neural network does not only require to put its weights into superposition, so do its neurons, respectively their dynamic action potentials or thresholds as well as its underlying mathematical function.

10.5.1.3.1 Superposition of dendrites

The dendrites or weights of a standard feed forward artificial neural network are lifted into linear superposition, which is the first step towards a quantum ANN. This means that every weight is changed into a wave and would theoretically contain every possible value at once. However, as currently the quantum ANN is simulated on a von Neumann machine, infinite accuracy, or better, any possible α_1 and α_2 in

$$|\psi\rangle = \alpha_1|0\rangle + \alpha_2|1\rangle = \begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix} \qquad (10\text{-}20)$$

restricted only by the normalization condition

$$|\alpha_1|^2 + |\alpha_2|^2 = 1 \qquad (10\text{-}21)$$

would require exponentially more time than restricting the possible peculiarity of the weights by a range and increasing them only by pre-defined increments starting from the lower

349 Bonsor Kevin, Strickland Jonathan (2000): How Quantum Computers Work [2012-10-18], URL: http://computer.howstuffworks.com/quantum-computer.htm

boundary to the upper boundary. Assuming the basis for the increment is 0.001 and the weight boundaries reach from -100 to 100, any value in this interval to the granularity of 0.001 will serve as possible weight. The process of setting the weights $w_0, \dots, w_n$ into $|\psi\rangle_0, \dots, |\psi\rangle_n$ with the exemplified parameters above happens as follows:

Start

1. Set -100 as p_{start} and 100 as p_{end}
2. ***For each w***
   a) Set $w_{current}$ to p_{start}
   b) Repeat
      i. Save $w_{current}$ into corresponding $\psi n_f n_t$
      ii. Set
      $$w_{current} = w_{current} + 0.001$$
   c) Until p_{end} is reached

End

Algorithm 26 - Lifting weights into superposition

Breakdown:

$\psi n_f n_t$: Array list holding each possible normalized weight, being the superposition of a specific weight and n_f being the from-neuron, n_t being the to-neuron
p_{start}: Calculation start point
p_{end}: Calculation end point
$w_{current}$: The current weight, not normalized

10.5.1.3.2 Superposition of neurons

The registers of each layer are lifted into linear superposition as well. Let say, an example QANN consists of three layers, like the one described with Figure 77 - Quantum artificial neural network, and let further assume that any of the QANNs neurons scales its output through an activation function and makes use of dynamic thresholds, in which a learning algorithm would scale in standard ANN learning, then, according to the increment and upper and lower threshold boundaries, logically several thresholds for each superposition of networks must exist. Let further assume, the lower threshold boundary would be 0.1, the upper threshold boundary 1 and the increment 0.1, then 9 different values for a threshold would be possible. In the example QANN this leads to 387,420,489 (9^9) possible configurations of thresholds for one configuration of weights. However, as a specific configuration of weights only exists once within $|\psi\rangle$, the whole QANN must be lifted into superposition.

10.5.1.3.3 Superposition of the quantum artificial neural network

As the linear superposition indeed dictates that any possible configuration of each Qbit exists at once and as, related to the example QANN, any of these (weight) configurations has numerous configurations of thresholds, an impracticality occurs, namely the coexistence of

identical configurations in one superposition. Thus, a single configuration of a quantum artificial neural network must include both the weights and the neuron thresholds.

10.5.1.4 Entanglement

Entanglement within a quantum artificial neural network may be realized by a quantum physical effect called quantum teleportation (this chapter has strongly been influenced by Li Weigang's research on entangled ANNs – for further details see his publication).[350] Entanglement has already been discussed in detail, so we will continue quantum teleportation, which on the contrary to solely quantum entanglement, includes entangled as well as classical communication channels. First of all, we recall the first maximally entangled Bell-state

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B) \qquad \textit{(10-22)}$$

or in a simpler manner of writing

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \qquad \textit{(10-23)}$$

A source that is capable of creating such entangled states is an EPR-source such as discussed at 7.3 The state vector reduction R. Additionally, two entities A and B, the first being the sender and the second being the receiver, a classical communication channel as well as a the teleportation system (an EPR-source-creating device) are sufficient for being able to construct the teleportation-algorithm (the following algorithms will be described with more continuous text than usual within this work to make them more comprehensible):

Start

1. Each of A and B receives one of the entangled particles emitted from the EPR-source, so that an EPR pair is created:
$$|\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$
2. A intends to submit the state of A's Qbit
$$|\phi\rangle_A = \frac{1}{\sqrt{2}}(\alpha_1|0\rangle + \alpha_2|1\rangle)$$
to B via the quantum and the classical channel. Decoding is applied on $|\phi\rangle_A$ and the first half of the entangled Qbit (the second is in possession of B):
$$|\phi\rangle_A \otimes |\phi^+\rangle = \frac{1}{\sqrt{2}}\big(\alpha_1|0\rangle \otimes (|00\rangle + |11\rangle) + \alpha_2|1\rangle \otimes (|00\rangle + |11\rangle)\big)$$
$$= \frac{1}{\sqrt{2}}(\alpha_1|000\rangle + \alpha_1|110\rangle + \alpha_2|100\rangle + \alpha_1|111\rangle)$$
3. A applies $c_{ij} \otimes I$ and $H \otimes I \otimes I$ on the result:

350 Li Weigang: Entangled Neural Networks; Brazil: University of Brasilia

$$(H \otimes I \otimes I)(c_{ij} \otimes I)(\phi \otimes \phi^+)$$
$$= (H \otimes I \otimes I)(c_{ij} \otimes I)\frac{1}{\sqrt{2}}(\alpha_1|000\rangle + \alpha_1|110\rangle + \alpha_2|100\rangle + \alpha_2|111\rangle)$$
$$= (H \otimes I \otimes I)\frac{1}{\sqrt{2}}(\alpha_1|000\rangle + \alpha_1|011\rangle + \alpha_2|110\rangle + \alpha_2|101\rangle)$$
$$= \frac{1}{2}\big(|00\rangle(\alpha_1|0\rangle + \alpha_2|1\rangle) + |01\rangle(\alpha_1|1\rangle + \alpha_2|0\rangle) + |10\rangle(\alpha_1|0\rangle - \alpha_2|1\rangle)$$
$$+ |11\rangle(\alpha_1|1\rangle - \alpha_2|0\rangle)\big)$$

4. A is in control of the first two Qbits, whereas B is in control of the last one. A measures one of the states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ on the first two Qbits with equal probability and submits the result via the classical channel to B.
5. Depending from what A has measured, the state of B's Qbit is projected to one of the following states, each one consisting of the phase state + and – as well as the basic states 0 and 1:

$$|\phi\rangle_B = (\alpha_1|0\rangle + \alpha_2|1\rangle)$$
$$|\phi\rangle_B = (\alpha_1|1\rangle + \alpha_2|0\rangle)$$
$$|\phi\rangle_B = (\alpha_1|0\rangle - \alpha_2|1\rangle)$$
$$|\phi\rangle_B = (\alpha_1|1\rangle - \alpha_2|0\rangle)$$

6. B receives the information submitted by A on the classical channel, thus two classical bits according to which he knows how to decode his Qbit so that it matches the original state of A's Qbit:

Classical bits received	Final state of Qbit B	Transformation to receive final state
00	$\alpha_1\|0\rangle + \alpha_2\|1\rangle$	I
01	$\alpha_1\|1\rangle + \alpha_2\|0\rangle$	X
10	$\alpha_1\|0\rangle - \alpha_2\|1\rangle$	Z
11	$\alpha_1\|1\rangle - \alpha_2\|0\rangle$	$Y\ (= ZX)$

End

Algorithm 27 – Quantum teleportation

Brought forward to connections within quantum artificial neural networks, the idea of entanglement may be applied in the way that a receiving neuron may decode the initial state of a sending neuron, and depending from the result, the signal may be passed over to the next neuron or not (see Figure 78 – Quantum teleportation network unit).

Figure 78 – Quantum teleportation network unit shows that there exist three units, where the EPR source represents unit II, the sending neuron unit A and the receiving neuron unit B, whereby for the following explanations both are considered to be hidden units. Algorithm 28 explains the difference in processing compared to standard quantum teleportation:

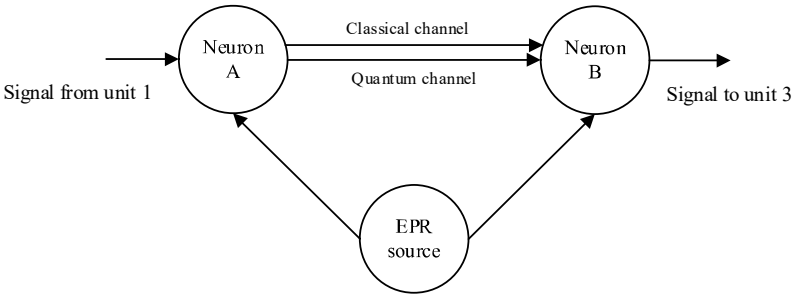

Figure 78 – Quantum teleportation network unit

Start

1. Each of the neurons A and B receives one of the entangled particles emitted from the EPR-source, so that an EPR pair is created:
$$|\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$
2. Neuron A implements a decision factor that is represented by a Qbit-superposition, where the squared amplitudes $|\alpha_1|^2$ and $|\alpha_2|^2$ represent the probability that the state is collapsing into one result
$$|\phi\rangle_A = \frac{1}{\sqrt{2}}(\alpha_1|0\rangle + \alpha_2|1\rangle)$$
3. Neuron A is sent information from one of the neurons attached to it in the previous layer, whereas the connection may be of classical or quantum nature.
4. Neuron A intends to submit the state of A's Qbit
$$|\phi\rangle_A = \frac{1}{\sqrt{2}}(\alpha_1|0\rangle + \alpha_2|1\rangle)$$
to B via the quantum or the classical channel. For this, decoding is applied on $|\phi\rangle_A$ and the first half of the entangled Qbit (the second is in possession of B):
$$|\phi\rangle_A \otimes |\phi^+\rangle = \frac{1}{\sqrt{2}}\big(a|0\rangle \otimes (|00\rangle + |11\rangle) + b|1\rangle \otimes (|00\rangle + |11\rangle)\big)$$
$$= \frac{1}{\sqrt{2}}(a|000\rangle + a|110\rangle + b|100\rangle + b|111\rangle)$$
5. Neuron A applies $c_{ij} \otimes$ I and H $\otimes$ I $\otimes$ I on the result:
$$(H \otimes I \otimes I)\big(c_{ij} \otimes I\big)(\phi \otimes \phi^+)$$
$$= (H \otimes I \otimes I)\big(c_{ij} \otimes I\big)\frac{1}{\sqrt{2}}(\alpha_1|000\rangle + \alpha_1|110\rangle + \alpha_2|100\rangle + \alpha_2|111\rangle)$$
$$= (H \otimes I \otimes I)\frac{1}{\sqrt{2}}(\alpha_1|000\rangle + \alpha_1|011\rangle + \alpha_2|110\rangle + \alpha_2|101\rangle)$$
$$= \frac{1}{2}\big(|00\rangle(\alpha_1|0\rangle + \alpha_2|1\rangle) + |01\rangle(\alpha_1|1\rangle + \alpha_2|0\rangle) + |10\rangle(\alpha_1|0\rangle - \alpha_2|1\rangle) + |11\rangle(\alpha_1|1\rangle - \alpha_2|0\rangle)\big)$$
6. Neuron A is in control of the first two Qbits, whereas B is in control of the last one.

Depending from the information out- and inside the unit, this is called decision-key τ (the first 2 Qbits could be measured in A, the third one in B). By the use of τ, A measures one of the states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ on the first two Qbits with equal probability. The result is now defined as measurement-key υ. By the use of Grover's search algorithm, τ is amplified to $|\beta_1|^2$ and the probability of all other vectors is reduced to $|\beta_2|^2$, where

$$|\beta_1|^2 + 3|\beta_2|^2 = 1$$

and

$$|\beta_1|^2 \gg |\beta_2|^2$$

υ, which equals τ with the probability of $|\beta_1|^2$, is sent to B via the classical channel.

7. Depending from what has been measured in neuron A, thus υ, the state of neuron B's Qbit is projected to one of the following states, each one consisting of the phase state + and – as well as the basic states 0 and 1:

$$|\phi\rangle_B = (\alpha_1|0\rangle + \alpha_2|1\rangle)$$
$$|\phi\rangle_B = (\alpha_1|1\rangle + \alpha_2|0\rangle)$$
$$|\phi\rangle_B = (\alpha_1|0\rangle - \alpha_2|1\rangle)$$
$$|\phi\rangle_B = (\alpha_1|1\rangle - \alpha_2|0\rangle)$$

8. Neuron B receives the information submitted by neuron A, thus υ, on the classical channel. Therefore, one of the following four states can be measured, depending from the content of υ:

Classical bits received	Final state of Qbit B	Transformation to receive final state
00	$\alpha_1\|0\rangle + \alpha_2\|1\rangle$	I
01	$\alpha_1\|1\rangle + \alpha_2\|0\rangle$	X
10	$\alpha_1\|0\rangle - \alpha_2\|1\rangle$	Z
11	$\alpha_1\|1\rangle - \alpha_2\|0\rangle$	$Y\ (= ZX)$

9. The measurement result may then be transmitted to another unit (neuron) in the next layer, either by a classical or a quantum connection.

End

Algorithm 28 – Quantum teleportation artificial neural network

Let assume, $\tau = 01$, then Grover's search algorithm would result in

$$Grover\left((H \otimes I \otimes I)(c_{ij} \otimes I)(\phi \otimes \phi^+)\right) = \beta_2|00\rangle(a|0\rangle + b|1\rangle) + \beta_1|01\rangle(a|1\rangle + b|0\rangle) + \beta_2|10\rangle(a|0\rangle - b|1\rangle) + \beta_2|11\rangle(a|1\rangle - b|0\rangle) \quad (10\text{-}24)$$

The phase states as well as the amplitudes are used to represent factors, thus in this example, the Qbit of neuron A in unit 1 (as in any other unit, but we start with unit 1) is capable of representing four factors and their inverses:

Neural unit 1

Assuming that the four factors of neuron A are represented by $a, !\,a, b, !\,b$ then the positive phase (+) may be used to represent a, the negative phase (-) to represent $!\,a$, the first amplitude (α_1) to represent b and the second amplitude (α_2) to represent $!\,b$.

Neural unit 2

Assuming that the four factors of neuron A are represented by $c, !\, c, d, !\, d$ then the positive phase (+) may be used to represent c, the negative phase (-) to represent $!\, c$, the first amplitude (α_1) to represent d and the second amplitude (α_2) to represent $!\, d$.

Neural unit 3

Assuming that the four factors of neuron A are represented by $e, !\, e, f, !\, f$ then the positive phase (+) may be used to represent e, the negative phase (-) to represent $!\, e$, the first amplitude (α_1) to represent f and the second amplitude (α_2) to represent $!\, f$.

Based on the assumption that we start processing within unit 1, this unit's neuron A is defined as

$$|\phi\rangle_A = \frac{1}{\sqrt{2}}(\alpha_1|!\, b\rangle + \alpha_2|b\rangle) \qquad (10\text{-}25)$$

B's quantum state is projected into one of the four states

$$|\phi\rangle_B = (\alpha_1|0\rangle + \alpha_2|1\rangle) \qquad (10\text{-}26)$$

$$|\phi\rangle_B = (\alpha_1|1\rangle + \alpha_2|0\rangle) \qquad (10\text{-}27)$$

$$|\phi\rangle_B = (\alpha_1|0\rangle - \alpha_2|1\rangle) \qquad (10\text{-}28)$$

$$|\phi\rangle_B = (\alpha_1|1\rangle - \alpha_2|0\rangle) \qquad (10\text{-}29)$$

depending from υ. Each of these states represents a combination of the factors represented by the Qbit of neuron A. This results in

$$|\phi\rangle_B = (\alpha_1|0\rangle + \alpha_2|1\rangle) \rightarrow \begin{cases} (a, !\, b) & ,if\ 0\ with\ probability|\alpha_1|^2 \\ (a, b) & ,if\ 1\ with\ probability|\alpha_2|^2 \end{cases} \qquad (10\text{-}30)$$

with precondition

$\upsilon = 00$(10-31)

$$|\phi\rangle_B = (\alpha_1|1\rangle + \alpha_2|0\rangle) \rightarrow \begin{cases} (a, b) & ,if\ 0\ with\ probability|\alpha_1|^2 \\ (a, !\, b) & ,if\ 1\ with\ probability|\alpha_2|^2 \end{cases} \qquad (10\text{-}32)$$

with precondition

$$\upsilon = 01 \qquad (10\text{-}33)$$

$$|\phi\rangle_B = (\alpha_1|0\rangle - \alpha_2|1\rangle) \rightarrow \begin{cases} (!\, a, !\, b) & ,if\ 0\ with\ probability|\alpha_1|^2 \\ !\, a, b & ,if\ 1\ with\ probability|\alpha_2|^2 \end{cases} \qquad (10\text{-}34)$$

with precondition

$$\upsilon = 10 \qquad (10\text{-}35)$$

$$|\phi\rangle_B = (\alpha_1|1\rangle - \alpha_2|0\rangle) \rightarrow \begin{cases} (!\, a, b) & ,if\ 0\ with\ probability|\alpha_1|^2 \\ !\, a, !\, b & ,if\ 1\ with\ probability|\alpha_2|^2 \end{cases} \qquad (10\text{-}36)$$

with precondition

$$\upsilon = 11 \qquad (10\text{-}37)$$

for neural unit 1.Each of these therefore allows two measurement results, but all in all there are just four that differ from each other. These four measurement results represent the possible decision results of neural unit 1. τ both depends from information of inside and outside the unit, each represented by 0 or 1.

The first number represents the phase state within the unit, whereas the second part represents what has been learned from outside the neural unit. For being able to represent neuron A's Qbit

$$|\phi\rangle_A = \frac{1}{\sqrt{2}}(\alpha_1|0\rangle + \alpha_2|1\rangle) \qquad (10\text{-}38)$$

it is required to represent the states from the Qbits emitted by the EPR-source

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \qquad (10\text{-}39)$$

Basically, this is similar to the representation of neural unit 1 described earlier, with the difference that the phase states are also represented by 0 and 1:

Neural unit 1 (inside information)

Assuming that the four factors of neuron A are represented by $a, !\,a, b, !\,b$ then the positive phase (+, but described by 0) may be used to represent a, the negative phase (but described by 1) to represent $!\,a$, the first amplitude (α_1) to represent b and the second amplitude (α_2) to represent $!\,b$.

Neural unit 2 (inside information)

Assuming that the four factors of neuron A are represented by $c, !\,c, d, !\,d$ then the positive phase (+, but described by 0) may be used to represent c, the negative phase (-,but described by 1) to represent $!\,c$, the first amplitude (α_1) to represent d and the second amplitude (α_2) to represent $!\,d$.

Neural unit 3 (inside information)

Assuming that the four factors of neuron A are represented by $e, !\,e, f, !\,f$ then the positive phase (+, but described by 0) may be used to represent e, the negative phase (-,but described by 1) to represent $!\,e$, the first amplitude (α_1) to represent f and the second amplitude (α_2) to represent $!\,f$.

Outside information

The information learned from outside does not feature amplitudes, so only the phases are represented by 0 and 1 as described in any other neural unit.

Given this information, it is possible to describe how exactly τ is determined. First of all, we get back to the example: From now on, the representations of the phases are concluded as requirement factors, and the amplitudes as influence factors, so that the problem statement that has to be solved demands $!\,a, !\,c, e$ as solution, each of them coded into a different neural

unit. These requirement factors can be influenced by the amplitudes, or influence factors. Thus, the network is trained to determine the requirements factors based on the influence factors as input – a classical classification problem. Furthermore, expected decision results are defined as a combination of requirement and influence factors of neural unit 3: $e,f;\ e,!f;\ !e,f;\ !e,!f$

Start

1. Starting in neural unit 1, where !a is represented by 1 (phases are represented by 0 and 1, as described earlier), the first part of τ is thus 1.
2. The second part comes from outside, and is the factor e (which is also represented in neural unit 3, but the outside information is required), which is represented by 0. Thus, the decision key for neural unit 1 is 10.
3. ***Repeat***
   a) Now that the decision key is known, the measurement key υ equals the decision key with the probability of $|\beta_1|^2$ according to

$$Grover\left((H \otimes I \otimes I)(c_{ij} \otimes I)(\phi \otimes \phi^+)\right) = \beta_2|00\rangle(a|0\rangle + b|1\rangle) + \beta_2|01\rangle(a|1\rangle + b|0\rangle) + \beta_1|10\rangle(a|0\rangle - b|1\rangle) + \beta_2|11\rangle(a|1\rangle - b|0\rangle)$$

   and depending from what has been measured at neuron A. If Grover's search algorithm has projected all vectors that we do not want to measure orthogonal to what we want to measure, then neuron B's Qbit has been projected into

$$|\phi\rangle_B = (\alpha_1|1\rangle - \alpha_2|0\rangle)$$

   which is

$$|\phi\rangle_B = (\alpha_1|(!a, !b)\rangle - \alpha_2|(!a, b)\rangle)$$

   as described beforehand (the measurement key is 10, thus the possible results are restricted in advance). The decision result is obtained with the probabilities $|\alpha_1|^2$, $|\alpha_2|^2$ respectively.

   b) The next step is to use the decision result for defining the outside information-part of the decision key of neural unit 2. Assuming that the measurement resulted in (!a, b), then the state b in neural unit 1 may be used to orientate neuron B in neural unit 2 towards the state d.
   c) Finally, the requirement factor of unit 2, thus !c, is represented as 1, so the decision key of unit 2 is 10.
4. ***Until all units have been processed***

End

Algorithm 29 – Quantum teleportation artificial neural network processing

10.5.1.5 Interference

Quantum interference is especially interesting when dealing with stochastic ANNs, such as the Hopfield ANN or the Boltzmann machine, as via this phenomenon stochastic processes which are coupled via quantum interference can be modelled. Furthermore, the interference of presented patterns offers possibilities classical ANNs do not.

Remembering the single layer perceptron with one neuron,

$$y = f(\sum_{i=1}^{n} w_i x_i + \theta) + \epsilon_t \qquad (10\text{-}40)$$

or better a single layer perceptron with multiple neurons in its layer

$$x_t = f\left(\sum_{i=1}^{p} w_{ij} x_{t-i}\right) + \epsilon_t \qquad (10\text{-}41)$$

calculating the network's output x at time t, then a quantum representation considering the state vector

$$\alpha_0 \alpha_1 \dots \alpha_n = \langle \psi | \qquad (10\text{-}42)$$

and without uncertainty is as follows:

$$\alpha_{i(t)} = f\left(\sum_{i=1}^{p} U_{ij} \alpha_{j(t-i)}\right) \qquad (10\text{-}43)$$

where the probability of the transition of mapping the i^{th} eigenvector to the j^{th} eigenvector

$$\{00100\} \rightarrow \{00100\} \qquad (10\text{-}44)$$

(the 1 on the left side represents the i^{th} eigenvector, and the one on the right side the j^{th} eigenvector) is given by

$$p_i^j = \left|U_{ji}\right|^2 \qquad (10\text{-}45)$$

where

$$\sum_{i=1}^{n} p_i^j = 1 \qquad (10\text{-}46)$$

Thus, the matrix p_i^j represents the transition matrix.

More on measurement is explained later, but if a measurement on the state (amplitudes) vector is made, it is mapped on into an eigenstate vector:

$$\{\alpha_0 \alpha_1 \dots \alpha_n\} \rightarrow \{0,0 \dots 1 \dots 0\} \qquad (10\text{-}47)$$

If the architecture of the QANN is based on one measurement per iteration, and l measurements are conducted, then then the mapping becomes

$$\{\alpha_0 \alpha_1 \dots \alpha_n\} \rightarrow \left\{0, \dots, \frac{1}{\sqrt{l}}, \dots, \frac{1}{\sqrt{l}}, \dots\right\} \qquad (10\text{-}48)$$

with the probability that the i_k^{th} component of the new state vector is not zero of

$$p_{ik} = \alpha_{ik}^2 \qquad (10\text{-}49)$$

If a sequence of transformations for the state vector $\langle\psi|$

$$|\psi(0)\rangle \rightarrow U|\psi(0)\rangle \rightarrow f(U|\psi(0)\rangle) = |\psi_{t+1}\rangle \tag{10-50}$$

is assumed, which formally represents eq. (9-41), then for the continuation of the sequence the quantum device needs to be reset, so that the measured eigenstate can serve as new input, resulting in the QANN described with eq. (9-42). Assuming that the activation function is a sigmoidal one and corresponds to the explained vector mapping, the equation for the QANN can be rewritten as

$$\alpha_{i(t)} = \sigma_l\{U_{ij}\alpha_{j(t-1)}\} \tag{10-51}$$

Considering an architecture with two measurements, the mapping described in (9-43) changes to

$$\frac{1}{\sqrt{2}}\{00110\} \rightarrow \frac{1}{\sqrt{2}}\{00110\} \tag{10-52}$$

where we simply assume that before and after the mapping the i_1^{th} and the j_1^{th} is the third bit from the left, and the i_2^{th} and the j_2^{th} is the fourth bit from the left, so that

$$i_1 + i_2 \rightarrow j_1 + j_2 \tag{10-53}$$

where i_1, i_2, j_1, j_2 represent eigenstates with the unit 1 at the $i_1^{th}, j_1^{th}, i_2^{th}, j_2^{th}$ locations. This results in the transitional probability of mapping

$$p_{i_1,i_2}^{j_1}(i_1 + i_2 \rightarrow j_1) = \frac{1}{2}|j_1 U(i_1 + i_2)|^2 = \frac{1}{2}\left|U_{j_1 i_1} + U_{j_1 i_2}\right|^2 \tag{10-54}$$

$$p_{i_1,i_2}^{j_2}(i_1 + i_2 \rightarrow j_2) = \frac{1}{2}|j_2 U(i_1 + i_2)|^2 = \frac{1}{2}\left|U_{j_2 i_1} + U_{j_2 i_2}\right|^2 \tag{10-55}$$

Both mappings result from independent measurements, so the joint transition probability is simply

$$p_{i_1,i_2}^{j_1,j_2} = (i_1 + i_2 \rightarrow j_1 + j_2) = p_{i_1,i_2}^{j_1} p_{i_1,i_2}^{j_2} = \frac{1}{4}\left|U_{j_1 i_1} + U_{j_1 i_2}\right|^2 \left|U_{j_2 i_1} + U_{j_2 i_2}\right|^2 \tag{10-56}$$

Quantum interference now comes into play as the input patterns i_1 and i_2 interfere with each other, which means that their probabilities have to be added as they are subject to unitary transformations. The output patterns j_1 and j_2 do not interfere with each other, as they result from two independent measurements. Thus, the joint transition probabilities for both stochastic processes, which are coupled via quantum interference

$$p_{i_1,i_2}^{j_1,j_2} = \frac{1}{4}\left|U_{j_1 i_1} + U_{j_1 i_2}\right|^2 \left|U_{j_2 i_1} + U_{j_2 i_2}\right|^2 \tag{10-57}$$

are considered at the same time as each single one

$$i_1 + i_2 \rightarrow j_1 + j_2 \tag{10-58}$$

of the stochastic processes.

10.5.1.6 Processing

The calculations within the QANN need to be carried out as it is done in a standard FFANN. However, as already indicated on several occasions, one of the major advantages of quantum mechanics is its superposition and the resulting quantum parallelism. For the QANN-internal calculations done by the function f on all configurations of x the reversible unitary transformation U_f, only taking basis states ($|0\rangle$ and $|1\rangle$) into such, will serve as example processing of one of the QANN's Qbits:

$$U_f(|x\rangle_n|y\rangle_m) = |x\rangle_n|y \oplus f(x)\rangle_m \tag{10-59}$$

where $\oplus$ indicates a modulo-2 bitwise addition or exclusive or. If x and y are m-bit integers whose j^{th} bits are x_j and y_j, then $x \oplus y$ is the m-bit integer whose j^{th} bit is $x_j \oplus y_j$. $|x\rangle_n|y\rangle_m$ represents a tensor product and it may also be written as $|x\rangle_n \otimes |y\rangle_m$ or $|x_n y_m\rangle$. Within this elaboration all representations will be used, depending on the situation.

$$1101 \oplus 0111 = 1010 \tag{10-60}$$

Furthermore, if the initial value represented by the output register is

$$y = 0 \tag{10-61}$$

then

$$U_f(|x\rangle_n|0\rangle_m) = |x\rangle_n|f(x)\rangle_m \tag{10-62}$$

and $f(x)$ would represent the result in the output register, where regardless to the initial state of y the input register remains in its initial state $|x\rangle_n$. Furthermore, U_f fulfils another important criterion, which is that it is invertible:

$$U_f U_f |x\rangle|y\rangle = U_f(|x\rangle|y \oplus f(x)\rangle) = |x\rangle y \oplus f(x) \oplus f(x)\rangle = |x\rangle|y\rangle \tag{10-63}$$

as

$$z \oplus z = 0 \tag{10-64}$$

for any z. Equation (10-62) shows an important feature of quantum computation, namely that the application of the 1-Qbit Hadamard transformation H on each Qbit in the 2-Qbit state results in

$$(H \otimes H)(|0\rangle|0\rangle) = H_1 H_0 |0\rangle|0\rangle = (H|0\rangle)(H|0\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |0\rangle|1\rangle + |1\rangle|0\rangle + |1\rangle|1\rangle) \tag{10-65}$$

which leads to the generalization of the n-fold tensor product of n Hadamard transformations on the n-Qbit state

$$(H^{\otimes n})(|0\rangle_n) = \frac{1}{2^{n/2}} \sum_{0 \le x < 2^n} |\, x\rangle_n \tag{10-66}$$

of first use in equation (10-73). If the initial state of an input register is $|0\rangle$, the application of n-fold Hadamard transformations transforms the state of this register into an equally weighted superposition of all n-Qbit inputs.[351]

10.5.1.6.1 Entanglement

Considering the example described in Figure 77 - Quantum artificial neural network, a possible implementation of a quantum artificial neural network could begin with the superposition $|\psi\rangle_{wx_ny_n}$ of all possible weight vectors, which allows classifying all training examples with respect to every weight vector at once. Furthermore, a performance register $|p\rangle$ is used for storing the number of correctly classified training examples; its update may happen through the comparison of the network's actual output with verification registers $|v_{11}\rangle, \dots, |v_{n2}\rangle$, each holding the desired values of the respective input training sets. If the actual output corresponds with the desired output, the related specific performance registers $|sp_{11}\rangle, \dots, |sp_{n2}\rangle$ receive 1, otherwise nothing. After the training set has been processed, the specific performance registers are summed up and the result is then sent to the overall performance register $|p\rangle$:

$$|p_i\rangle = \sum |sp_{mn}\rangle_i \qquad (10\text{-}67)$$

where m represents the number of data sets, n the number of output values and i the networks' i^{th} weight configuration within the superposition. The update of the performance register with respect to each configuration of the QANN creates an entanglement between $|p_{1,\dots,n}\rangle$ and $|\psi\rangle_{1,\dots,n}$, where $|\psi\rangle_i$ represents all weights of the i^{th} and p_i the performance value of the same. Thus the oracle is

$$|p\rangle = i * o \qquad (10\text{-}68)$$

where i represents the number of training examples (inputs) and o the number of output neurons. As it may occur that either no configuration of a network within the superposition is able to classify all training examples correctly (and therefore every vector has an equal chance of being measured) or the amount of time required for finding a vector is increasing with the number of bits in the weight vector and thus exponential complexity:

$$O\left(\sqrt{2^b/t}\right) \qquad (10\text{-}69)$$

For avoiding the first case, Ventura and Ricks[352] suggest modifying the search oracle to

$$|p\rangle \geq i * o * p \qquad (10\text{-}70)$$

where p is the percentage of correctly classified training examples. With respect to quantum artificial neural networks, this means that any possible configuration of the quantum ANN is kept within the quantum linear superposition. However, there is still the problem of measurement, as measurement needs to be done when probability of receiving a desired result is high. Let assume a quantum register consisting of 64 Qbits each in the already known state

351 Mermin David N. (2007): Quantum Computer Science: An Introduction; Cambridge: Cambridge University Press

352 Ricks Bob, Ventura Dan (2003): Training a Quantum Neural Network; Provo: Brigham Young University

$$|\psi\rangle = \frac{1}{2}(|0\rangle + |1\rangle) \qquad (10\text{-}71)$$

then every possible state, or every value (double) that can be expressed with these 64 bits may be measured, but with the same probability distribution, so any of these double values would exist in this quantum register at once. This is where quantum entanglement comes into play, as each possible weight vector has been entangled with a slot in the performance register due to processing. Therefore, a measurement on the performance register when the probability of measuring the desired output (see eq. (10-70)) is close to 1 will also reveal the desired overall weight vector (ANN configuration) due to the resulting loss of coherence in the processing bits (10.5.1.7 Measurement). A more complicated substitute to equation (10-68) might be an operator U_f, applying the already known ANN performance calculations represented by a function f on $|\psi_i\rangle$:

$$x_{rmse} = \sqrt[2]{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \qquad (10\text{-}72)$$

This could be done by summing the results from each $U_f|\psi\rangle$ (U_f on each training set) into another quantum register $|\phi_i\rangle$, i representing the respective weight vector. When applying a quantum search on $|\phi\rangle$ after all, this then must include an average calculation, resulting in the overall RMSE. However, the input and output register-Qbits will become entangled by the calculation-Qbits, which in fact is a problem, as in this case both registers cannot be assigned a state on their own (see 10.5.1.6.4 Reduction of and information about the quantum perceptron equations).

10.5.1.6.2 Quantum parallelism

Let assume a training set consists of several hundreds of input data sets each consisting of several attributes, then the input register would at first grow and secondly U_f in a classical computer would have to be applied on every single input data set consecutively not only once, but $n-1$ times, where n represents the number of iterations (weigt adaptions) a training algorithm requires for adapting the overall weight vector of an ANN. Let further assume, 100 (fictive and not related to the inputs, but in numbers easier to describe) Hadamard transformations (gates) would be applied on every Qbit before the application of U_f, like

$$U_f(H^{\otimes n} \otimes 1_m)(|0\rangle_n|0\rangle_m) = \frac{1}{2^{n/2}}\sum_{0\le x<2^n} U_f\,(|\,x\rangle_n|0\rangle_m) \qquad (10\text{-}73)$$

then the final state would contain 2^{100} or $\approx 10^{30}$ applications of U_f.[353] However, quantum parallelism allows the performance of an exponentially high quantity of U_f in unitary time. This means that indeed every training set would require one application of U_f, but only once, as all possible weight vectors coexist in quantum linear superposition.

10.5.1.6.3 From basic operators to the quantum transfer function

Figure 84 - Quantum single layer perceptron diagram and Figure 85 - Quantum multi layer perceptron diagram both describe the whole processing of the quantum artificial neural

353 Mermin David N. (2007): Quantum Computer Science: An Introduction; Cambridge: Cambridge University Press

network as single transformation. However, it is required to detail the processing with regards to the activation or transfer function used by the neurons of the input- and hidden layer. Equation (10-141) describes the sigmoid function as quantum function

$$|f(x)\rangle = \langle\phi|\psi\rangle = \frac{1}{1+e^{-|x\rangle}} \tag{10-74}$$

where the quantum linear superposed $|x\rangle$ contains all possible values resulting from calculation based on the previous neuron layer output multiplied by the related weight vectors in superposition. This has not solved the basic challenge of arithmetic operations, like the multiplication or division required for all introduced quantum functions. As mentioned at the beginning, quantum operators must be linear, which means that

$$\Lambda(af_1 + bf_2) = a\Lambda f_1 + b\Lambda f_2 \tag{10-75}$$

where f_1 and f_2 represent functions, and a and b constants. To be complete here, the application of a logarithm is an example for a non-linear function, as e.g.

$$log\, 2x = 2\, log\, x \tag{10-76}$$

is not true for all values of x. Thus, the first required operator, the multiplication, fulfils the requirements, as it is a linear operator. This multiplication, as any other arithmetic operation used within the perceptron equations needs to be included in one of the beforehand described unitary transformations. For this, the quantum operators described at the beginning of this chapter will form the basis – especially the Hadamard transformation H and the cNOT operator c_{ij} are of importance, as these may be used to create operators like addition and multiplication. The basis for the cNOT-operator is the unitary matrix operator c-V (controlled V) described by the equations (10-77) – (10-79):

$$V = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{10-77}$$

$$V|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle \tag{10-78}$$

$$V|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ i \end{pmatrix} = i|1\rangle \tag{10-79}$$

After four subsequent applications of controlled V, the result is identity, and further three applications of the same operator result in its inverse, which is also its complex conjugate transpose $V^\dagger$. As mentioned beforehand, all operators must be linear, thus

$$V^\dagger V = VV^\dagger = I \tag{10-80}$$

where $V^\dagger$ is the complex conjugate transpose, or adjunct, of V, and I the identity operator. Therefore, the resulting operator is c-$V^\dagger$. Back to cNOT, this operator can be built as in Figure 79 - cNOT from H and V:

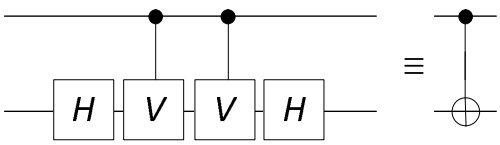

Figure 79 - cNOT from H and V

$$|0\rangle|0\rangle \longrightarrow |0\rangle\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \longrightarrow |0\rangle\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \longrightarrow |0\rangle|0\rangle \quad \textit{(10-81)}$$

$$|0\rangle|1\rangle \longrightarrow |0\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \longrightarrow |0\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \longrightarrow |0\rangle|1\rangle \quad \textit{(10-82)}$$

$$|1\rangle|0\rangle \longrightarrow |1\rangle\frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) \longrightarrow |1\rangle\frac{1}{\sqrt{2}}(|0\rangle + i^2|1\rangle) \longrightarrow |1\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \longrightarrow |1\rangle|1\rangle \textit{(10-83)}$$

$$|1\rangle|1\rangle \longrightarrow |1\rangle\frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle) \longrightarrow |1\rangle\frac{1}{\sqrt{2}}(|0\rangle - i^2|1\rangle) \longrightarrow |1\rangle|0\rangle \quad \textit{(10-84)}$$

This forms a very good basis for creating further operators, and indeed this needs to be done: for being able to perform all the arithmetic of the perceptron equations, another operator must be created, which is the ccNOT-operator (controlled-controlled-NOT), also known as Toffoli-gate[354] (Figure 80 - Toffoli gate with controlled V and Figure 81 - Toffoli-gate with complex conjugate transpose V):

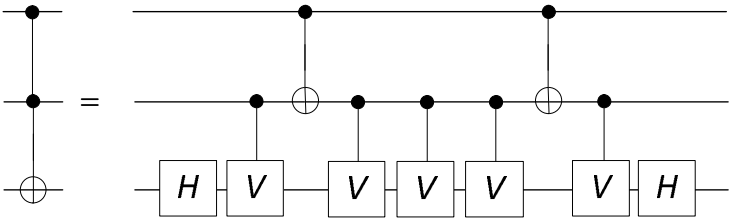

Figure 80 - Toffoli gate with controlled V

Or, with the already introduced quantum gate $V^{\dagger}$:

354 Toffoli Tommaso (1981): Mathematical Systems Theory 14 13.

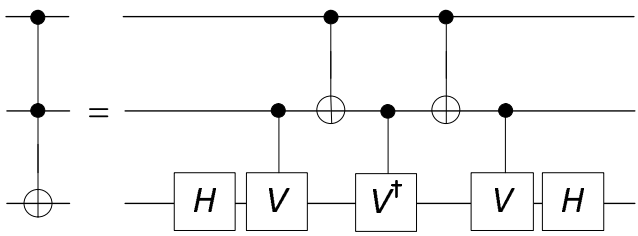

Figure 81 - Toffoli-gate with complex conjugate transpose V

This can be described in mathematical form by the quantum operations in the equations (10-85) and (10-86):

$$|110\rangle \longrightarrow |11\rangle\tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \longrightarrow |11\rangle\tfrac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) \longrightarrow |10\rangle\tfrac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) \longrightarrow$$
$$|10\rangle\tfrac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) \longrightarrow |11\rangle\tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \longrightarrow |11\rangle\tfrac{1}{\sqrt{2}}(|0\rangle + i^2|1\rangle) =$$
$$|11\rangle\tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \longrightarrow |111\rangle \qquad (10\text{-}85)$$

$$|111\rangle \longrightarrow |11\rangle\tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \longrightarrow |11\rangle\tfrac{1}{\sqrt{2}}(|0\rangle - i|1\rangle) \longrightarrow |10\rangle\tfrac{1}{\sqrt{2}}(|0\rangle - i|1\rangle) \longrightarrow$$
$$|10\rangle\tfrac{1}{\sqrt{2}}(|0\rangle - i|1\rangle) \longrightarrow |11\rangle\tfrac{1}{\sqrt{2}}(|0\rangle - i|1\rangle) \longrightarrow |11\rangle\tfrac{1}{\sqrt{2}}(|0\rangle - i^2|1\rangle) =$$
$$|11\rangle\tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \longrightarrow |110\rangle \qquad (10\text{-}86)$$

The Toffoli-gate, as controlled-controlled-NOT gate, features two control-bits, one more than the original cNOT-gate. The target bit is flipped only, and only when both of the control bits feature the state $|1\rangle|1\rangle$. Furthermore this provides a very important feature for multiplication (which in fact on binary level is an addition), namely a reversible AND-gate, if the target has initially featured the state $|0\rangle$: the target becomes the logical AND-operator of the two control bits as described in equation (10-87).[355]

$$|x_1, x_2\rangle|0\rangle \longrightarrow |x_1, x_2\rangle|x_1 \wedge x_2\rangle \qquad (10\text{-}87)$$

Thus, all required arithmetic operations, which are NOT, AND and XOR (cNOT), are available now as unitary transformations, which means that they can be stacked together to a larger unitary transformation for approaching the quantum perceptron equations. A combination of the Toffoli-gate and the cNOT-operator as quantum-adder form the basis for full addition and multiplication (Figure 82 - Quantum addition):

355 Quantiki (2005): Basic concepts in quantum computation [2013-01-02]; URL: http://www.quantiki.org/wiki/Basic_concepts_in_quantum_computation

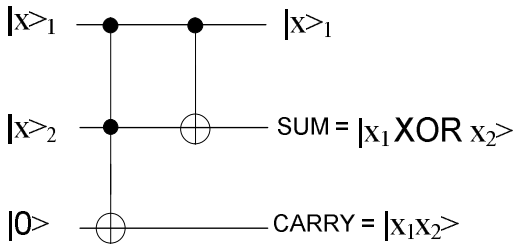

Figure 82 - Quantum addition

Furthermore, any Boolean function which maps n bits from an input register to m bits of an output register may be evaluated (equation (10-143)):

$$\{0,1\}^n \longrightarrow \{0,1\}^m \qquad (10\text{-}88)$$

As the operation of AND is not reversible, it needs to be embedded into the ccNOT-operator. If the third bit has initially been set to 1 instead of 0 then the value $|x_1 \wedge x_2\rangle$ is flipped. Generally, the action of the Toffoli-gate is written as described in equation (10-144):

$$|x_1, x_2\rangle|y\rangle \longrightarrow |x_1, x_2\rangle|(y + (x_1 \wedge x_2))\oplus 2^m\rangle \qquad (10\text{-}89)$$

Summing up, the function evaluation may be described as unitary evolution of the input and output registers (equation (10-145)):

$$|x, y\rangle \longrightarrow |x, (y + f(x))\oplus 2^m\rangle \qquad (10\text{-}90)$$

Thus, also more complex functions like the quantum sigmoid function ((10-74)), which include power functions like $e^{-|x\rangle}$ may be constructed, as the example of

$$f(x) = x^2 \qquad (10\text{-}91)$$

A quantum network (the construct of quantum operators) calculating

$$f: \{0,1\}^2 \longrightarrow \{0,1\}^3 \qquad (10\text{-}92)$$

such that equation (10-91) acts as described in (10-93) – (10-96):

$$|00\rangle|000\rangle \longrightarrow |00\rangle|000\rangle \qquad (10\text{-}93)$$

$$|01\rangle|000\rangle \longrightarrow |01\rangle|001\rangle \qquad (10\text{-}94)$$

$$|10|000\rangle \longrightarrow |10|100\rangle \qquad (10\text{-}95)$$

$$|11|000\rangle \longrightarrow |11\rangle|001\rangle \qquad (10\text{-}96)$$

which in terms of quantum computation is

$$|x, 0\rangle \longrightarrow |x, x^2\oplus 8\rangle \qquad (10\text{-}97)$$

an example being (10-98), which is (10-97):[356]

$$3^2 \oplus 2^3 = 1 \qquad \textit{(10-98)}$$

It becomes obvious from the above explanations that arithmetic operations like multiplication are not ones that benefit from quantum effects, as these still are step-by-step-operations. The benefit occurs, when multiple of these operations for different configurations coexist and can be calculated simultaneously due to quantum parallelism in the quantum state of a physical system, like an artificial neural network in quantum linear superposition.

10.5.1.6.4 Reduction of and information about the quantum perceptron equations

The nontrivial question is now, although both x and $f(x)$ are available, how to measure the required output, or in other words, how to get to know the content of a closed box without being allowed to look inside. As input and output are entangled now, a measurement of the output register would also let the input register collapse and furthermore, no information about f can be gathered, which in fact represents the QANN. Thus, one has to dismiss the general opinion which does not require one to know what happens between the input and output neurons exactly, but to extract as many information about the internal calculations on all configurations of the QANN in $|\psi\rangle$ as possible. This may be achieved with unitary quantum gates applied on all registers before and after the application of U_f and by combining these with measurements of some values, thus subsets of Qbits, of the QANN. It is not a hundred percent clear now how this may happen, but when the whole QANN is measured, this would on the one hand theoretically allow gaining useful relations between x and f, and all this from one calculation, but on the other hand according to Heisenberg's uncertainty principle not allow determining $f(x)$ for a specific x. Shor, for example, has developed a quantum algorithm (which is of course probabilistic) for factorization and by the use of the discrete Fourier transformation. But back to QANNs, Mermin,[357] when explaining Deutsch's problem,[358] gives a quite well example of how to reduce the probability of measuring a configuration, or better, gathering information of configurations of $f(x)$ that are not desired.

Let assume, x specifies a choice of two different inputs to an elaborate subroutine that requires many additional Qbits, then one can think of $f(x)$ as characterizing a two-valued property of this subroutine's output. It has also been mentioned that the input and output registers are not allowed to be entangled with the subroutine's Qbits after having finished processing, as in case of entanglement the input and output registers would not have final states on their own, which does not allow describing the computational process as unitary transformation at all. A simple linear transformation (10-59) may then be used to determine the net effects to the input and output register. The output

$$U_f(H \otimes 1)(|0\rangle|0\rangle) = \tfrac{1}{\sqrt{2}}|0\rangle|f(0)\rangle + \tfrac{1}{\sqrt{2}}|1\rangle|f(1)\rangle \qquad \textit{(10-99)}$$

---

356 Quantiki (2005): Basic concepts in quantum computation [2013-01-02]; URL: http://www.quantiki.org/wiki/Basic_concepts_in_quantum_computation

357 Mermin David N. (2007): Quantum Computer Science: An Introduction; Cambridge: Cambridge University Press

358 Deutsch David (1985): The Church-Turing principle and the universal quantum computer; Proceedings of the Royal Society of London A. 400, 1985, p. 97.

has been achieved by applying U_f on the input. Initially, the input and output registers need to be in the state $|0\rangle$, followed by the application of X on the input and output registers, again followed by the application of H on both. The input for U_f becomes then

$$(H\otimes H)(X\otimes X)(|0\rangle\otimes|0\rangle) = (H\otimes H)(|1\rangle\otimes|1\rangle) = \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}(|0\rangle|0\rangle - |1\rangle|0\rangle - |0\rangle|1\rangle + |1\rangle|1\rangle) \quad (10\text{-}100)$$

and the result

$$\frac{1}{2}U_f\left(U_f(|0\rangle|0\rangle) - U_f(|1\rangle|0\rangle) - U_f(|0\rangle|1\rangle) + U_f(|1\rangle|1\rangle)\right) \quad (10\text{-}101)$$

which in terms of the function $f(x)$ is then

$$\frac{1}{2}\left(|0\rangle|f(0)\rangle - |1\rangle|f(1)\rangle - |0\rangle|\tilde{f}(0)\rangle + |1\rangle|\tilde{f}(1)\rangle\right) \quad (10\text{-}102)$$

where

$$\tilde{x} = 1\oplus x \quad (10\text{-}103)$$

and

$$\tilde{1} = 0 \quad (10\text{-}104)$$

and

$$\tilde{0} = 1 \quad (10\text{-}105)$$

and

$$\tilde{f}(x) = 1\oplus f(x) \quad (10\text{-}106)$$

If

$$f(0) = f(1) \quad (10\text{-}107)$$

then the output state is

$$\frac{1}{2}\left(|0\rangle|f(0)\rangle - |1\rangle|f(1)\rangle - |0\rangle|\tilde{f}(0)\rangle + |1\rangle|\tilde{f}(1)\rangle\right) \quad (10\text{-}108)$$

or

$$\frac{1}{2}\left((|0\rangle - |1\rangle)\left(|f(0)\rangle - |\tilde{f}(0)\rangle\right)\right) \quad (10\text{-}109)$$

else if

$$f(0) \neq f(1) \quad (10\text{-}110)$$

then

$$\frac{1}{2}\left(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle - |0\rangle|\tilde{f}(0)\rangle + |1\rangle|\tilde{f}(1)\rangle\right) \quad (10\text{-}111)$$

or

$$\frac{1}{2}\Big((|0\rangle + |1\rangle)\big(|f(0)\rangle - |\tilde{f}(0)\rangle\big)\Big) \qquad (10\text{-}112)$$

H on the input register leads to

$$f(0) = f(1) \longrightarrow |1\rangle\frac{1}{\sqrt{2}}|f(0)\rangle - |\tilde{f}(0)\rangle \qquad (10\text{-}113)$$

and

$$f(0) \neq f(1) \longrightarrow |0\rangle\frac{1}{\sqrt{2}}|f(0)\rangle - |\tilde{f}(0)\rangle \qquad (10\text{-}114)$$

summed up as

$$(H\otimes 1)U_f(H\otimes H)(X\otimes X)(|0\rangle\otimes|0\rangle) = \begin{cases} |1\rangle\frac{1}{\sqrt{2}}|f(0)\rangle - |\tilde{f}(0)\rangle & , f(0) = f(1) \\ |0\rangle\frac{1}{\sqrt{2}}|f(0)\rangle - |\tilde{f}(0)\rangle & , f(0) \neq f(1) \end{cases} \qquad (10\text{-}115)$$

This states that the input register is either $|0\rangle$ or $|1\rangle$, but depends on what is true, equation (10-107) or (10-110). The clue is that two of four possible representations of the function f have been eliminated, just by one operation, which again is a quantum computer-specific ability. In terms of the QANN the same must now happen in more complex ways, as not only one Qbit needs to be taken into consideration and the internal calculations of an ANN require more than a few quantum operators and transformations. Currently, exactly this is subject of further research.

However, this is not enough. The transformation U_f does not consider the additional Qbits required for the calculations within the QANN. Thus, let assume a further transformation W_f, again inspired by Mermin's naming convention taking these bits into consideration. Whereas U_f only considered the Qbits of the input and output registers, W_f takes into account all Qbits. In terms of the QANN, this can is described schematically by Figure 83 – Quantum artificial neural network calculations

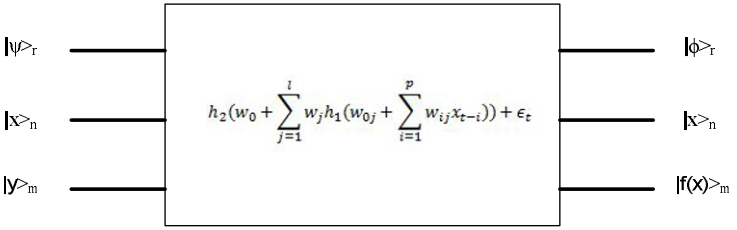

Figure 83 – Quantum artificial neural network calculations[359]

359 This is only a schematic representation used to explain the relation between inputs and outputs. I am careful to point out that this diagram is a lot simpler than the quantum circuit diagram breaking down the whole QANN instruction to single gates.

where the thick lines represent multiple Qbit-inputs, with multiple Qbit-registers, as the calculation must be reversible. m and n represent the Qbits of the input and output registers, and r represents the additional Qbits required for the QANN-internal computation. The r Qbits required for the computations are as well in superposition and after computation entangled with the input and output register. Thus the following quantum perceptron equations have to be set up.

Quantum single layer perceptron equation

$$|f(x)\rangle = \langle\phi|\psi\rangle = h(\sum_{i=1}^{n} |\omega_i\, x_i\rangle + \theta) + \epsilon_t \qquad (10\text{-}116)$$

where h is the activation function, applied to the summed weights in linear superposition $|\omega\rangle_i$ multiplied by the input values in linear superposition $|x\rangle_i$, if the threshold is exceeded. n represents the number of inputs. The activation function needs to be in superposition as well, as the dynamic action potentials of the neurons lead to multiple configurations of f. ϵ_t is the uncertainty variable of the ANN and additionally, the equation takes into consideration a bias θ, which is not necessarily used. This adapts Figure 83 – Quantum artificial neural network calculations to Figure 84 - Quantum single layer perceptron diagram

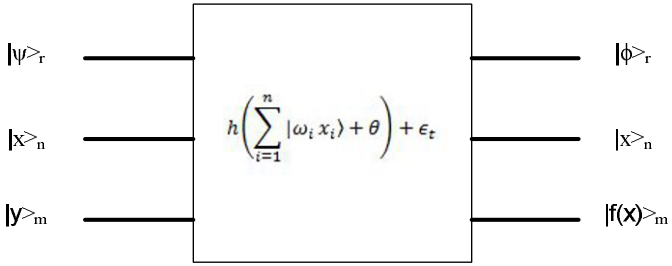

Figure 84 - Quantum single layer perceptron diagram[360]

Quantum multi layer perceptron equation

$$|f(x_t)\rangle = \langle\phi_t|\psi\rangle = h_2(w_0 + \sum_{j=1}^{l} |w_j\rangle h_1(w_{0j} + \sum_{i=1}^{p} |w_{ij} x_{t-i}\rangle)) + \epsilon_t \qquad (10\text{-}117)$$

where the input layer has p inputs $x_{t-1}, \ldots, x_{t-p}$, the hidden layer has l hidden nodes and the output, and there is a single output for the output layer x_t. Layers are fully connected by weights, where $|w\rangle_{ij}$ in linear superposition represents i^{th} input for the j^{th} node in the hidden layer, whereas $|w\rangle_j$ is the weight assigned to the j^{th} node in the hidden layer for the output. w_0 and w_{0j} are the biases, h_1 and h_2 are activation functions. Again, the dynamic action potentials of the neurons must be in linear superposition as well, which then leads to multiple

360 This is only a schematic representation used to explain the relation between inputs and outputs. I am careful to point out that this diagram is a lot simpler than the quantum circuit diagram breaking down the whole QANN instruction to single gates.

configurations of not only the weights, but also the neurons. This finally results in Figure 85 - Quantum multi layer perceptron diagram

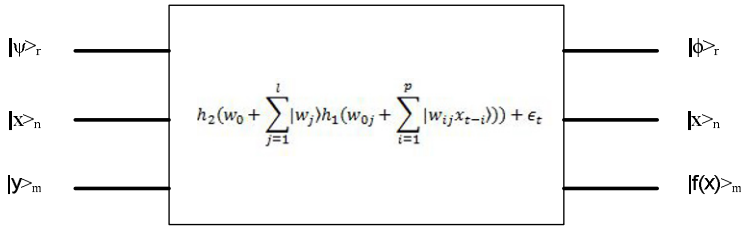

Figure 85 - Quantum multi layer perceptron diagram[361]

However, Figure 85 - Quantum multi layer perceptron diagram may not be the final state, as this requires unentangled states of the registers. Moreover, in the initial state of the system it is required to ensure the independency between the input and output registers' initial states and the additional calculation-Qbits initial states. The independence of the calculation-Qbits can be achieved by setting them to an initial state $|\psi\rangle_r$ and also taking care of the final state $|\phi\rangle_r$, which should be identical to $|\psi\rangle_r$, which can be achieved by reversible unitary transformations:

$$W_f = V_f^* C_m V_f \qquad (10\text{-}118)$$

or described graphically with Figure 86 - Reverse the calculation bits

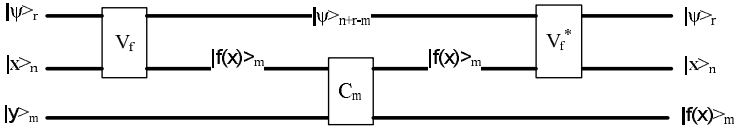

Figure 86 - Reverse the calculation bits[362]

where V_f is a unitary transformation on the input register m and the calculation Qbit-register r, which creates an own state for both. Furthermore, this allows constructing the function $f(x)$ in a subset of the output register n, based on the entangled input and calculation Qbits. Then $|y\rangle_m$ is transformed into $|y \oplus f(x)\rangle_m$ without influencing the bits of the input register or the bits of the calculation register by the unitary transformation C_m. C_m does not affect the input or the calculation register, the inverse of V_f, namely V_f^* can be used to restore the initial

361 This is only a schematic representation used to explain the relation between inputs and outputs. I am careful to point out that this diagram is a lot simpler than the quantum circuit diagram breaking down the whole QANN instruction to single gates.

362 Mermin David N. (2007): Quantum Computer Science: An Introduction; Cambridge: Cambridge University Press

state of $|\psi\rangle_r$.[363] However, it is not always required to go that far, as the QANN is a mathematical function, where the variables in Figure 83 – Quantum artificial neural network calculations are replaced by real numbers. Therefore, just for gathering information about f, it is not required to revert the unitary transformation V_f with V_f^*, but it is indeed required when there is the need to verify if $f(x)$ equals $f(y)$, which may be the case for an auto-associative QANN.

However, although the elimination of possible configurations of f decreases the probability of measuring an undesired configuration and the time needed for quantum search, it does not reveal the desired configuration. Thus, a further register is required for measuring the performance of the quantum artificial neural network, for which continuous entanglement with the calculation register is absolutely necessary. After having applied the unitary transformations clearing away the entanglement between the input, calculation and output registers, there should only remain entanglement of the calculation and performance registers. When applying a quantum search on the performance register, which in fact manipulates the phases of the possible performance values until the probability of measuring the desired output is near unity, the following measurement also lets the calculation register collapse, which then reveals the desired function f, thus the quantum artificial neural network one is searching for (detailed explanation follows in 10.5.1.7 Measurement).

10.5.1.6.5 Normalization

As in terms of input scaling a quantum feed forward artificial neural network still processes information in the same way an ordinary one does, the need for input normalization does not vanish. Thus, the input has to be scaled to a peculiarity between 0 and 1 for positive values and -1 and 0 for negative values.

Start

1. Create normalization factor

$$f = \frac{1}{\sqrt[2]{\sum_{i=0}^{n-1} x_i^2}}$$

2. ***For each*** w
   a) Normalize $w_{current}$ with f to $w_{current}^{normalized}$
   b) Save $w_{current}^{normalized}$ into corresponding $n_f n_t$

End

Algorithm 30 – Quantum input normalization

Breakdown:

$w_{current}$: The current weight, not normalized
$w_{current}^{normalized}$: The current weight, normalized
f: The normalization factor according to multiplicative normalization

363 Mermin David N. (2007): Quantum Computer Science: An Introduction; Cambridge: Cambridge University Press

$n_f n_t$: Array list holding each possible weight, n_f being the from-neuron, n_t being the to-neuron

Data normalization happens before processing the quantum ANN and does not necessarily be carried out by the quantum computer, as the processing is straightforward and not NP-hard.

10.5.1.7 Measurement

Finally, all of this fails to consider the basic search problem, as a configuration of the QANN within the allowed parameters has to be found within its superposition. This is where Grover's[364] algorithm needs to be applied, which is used for searching a special configuration or item in an unsorted database (which is in case of the QANN the performance register in linear superposition). Grover's algorithm provides the answer to when the system shall be measured, as measurement lets the superposition collapse, which eliminates all possible configurations of the QANN except the measured one. It is important to mention that the algorithm is capable of finding the desired solution in $O(\sqrt{N})$ time (iterations), which is nothing that can be done on a von Neumann computer.

10.5.1.7.1 Quantum artificial neural network configuration search function

When searching an unstructured database, containing $N = 2^n$ datasets, and with the search function $f(x)$, called search oracle, then according to probability theory the probability P for finding the desired dataset x_d is k/N, where k is the number of randomly chosen database entries. Thus, on a von Neumann computer searching x_d requires an oracle querying all datasets (equation (10-119)).

$$O(N) = O(2^n) \tag{10-119}$$

calls of the oracle $f(x)$, if

$$f(x) = \begin{cases} 1, & if\ x = x_d \\ 0, & if\ x! = x_d \end{cases} \tag{10-120}$$

According to Grover's search algorithm, the number of oracle calls can be reduced dramatically, when inverting the phase of the desired basis states followed by an inversion of all basis states about the average amplitude of all states. The repetition of this process produces an increase of the amplitude of the desired basis state to near unity, followed by a corresponding decrease in the amplitude of the desired state back to its original magnitude.[365] Grover detected that this routine just needs to be called a number of repetitions that does not exceed $\frac{\pi}{4}\sqrt{N}$, which is $O(\sqrt{N})$ iterations and thus, although the search is stochastic somehow, it outperforms a classical computer.

When $\mathcal{H}_2$ describes a two-dimensional Hilbert-space with the already known orthonormal basis of $\{|0\rangle, |1\rangle\}$ and $\{|0\rangle, |1\rangle, \dots, |N-1\rangle\}$ describes the orthonormal basis the Hilbert space

364 Grover Lov K. (1996): A fast quantum mechanical algorithm for database search, Proceedings of the 28th Annual ACM Symposium on the Theory of Computation, pp.212-219.

365 Ezhov Alexandr A., Ventura Dan (-): Quantum neural networks, BSTU Laboratory of Artificial Neural Networks

$$\mathcal{H} = \bigotimes_{0}^{n-1} \mathcal{H}_2 \qquad (10\text{-}121)$$

where $\otimes$ again represents the tensor product. Here, the unitary transformation U_f represents the oracle function $f(x)$ by

$$|x\rangle \otimes |y\rangle \longrightarrow |x\rangle \otimes |f(x) \oplus y\rangle \qquad (10\text{-}122)$$

$\oplus$ again representing the exclusive or, identically to equation (10-59). However, as already indicated beforehand, the inner workings of the unitary transformation U_f are not known. However, it may be replaced by another computationally equivalent unitary transformation, namely

$$I_{|x_d\rangle}(|x\rangle) = (-1)^{f(x)}|x\rangle = \begin{cases} -|x_d\rangle, & if\ x = x_d \\ |x\rangle, & if\ x! = x_d \end{cases} \qquad (10\text{-}123)$$

Equivalence is given, as

$$U_f \otimes |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = \left(I_{|x_d\rangle}(|x\rangle)\right) \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \qquad (10\text{-}124)$$

Furthermore, U_f results from a controlled $I_{|x_d\rangle}$ and two one-bit Hadamard transformations, and is, in fact, an inversion in H about the hyperplane orthogonal to $|x_d\rangle$. Thus, $I_{|x_d\rangle}$ may also be expressed as the tensor product of the desired ket with its bra and the identity transformation I, x_d now represented as a ray:[366,367]

$$I_{|\psi\rangle} = I - 2|\psi\rangle\langle\psi| \qquad (10\text{-}125)$$

10.5.1.7.2 Example processing

The already explained Hadamard-transformation needs to be applied for creating a superposition of all database entries, as described with equation (10-73). In terms of the quantum artificial neural network, a conglomerate of all basis states each in a specific state represent one configuration. The Hadamard transformation applied for creating $|\psi\rangle$ is

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \qquad (10\text{-}126)$$

where N is the number of datasets and i represents the possible basis states of a neuron or even a whole artificial neural network. It is important to mention here that $|\psi\rangle$ contains $|x_d\rangle$ amongst all other basis states. Both states span a Euclidean plane, and Grover's algorithm rotates $|\psi\rangle$ about its origin as often as it takes this state to be as close to $|x_d\rangle$ as possible. This is the time the measurement needs to be done. However, before that the unitary transformation

$$Q = -I_{|\psi_d\rangle} I_{|x_d\rangle} \qquad (10\text{-}127)$$

366 Lomonaco Samuel J. Jr. (2000): Grover's Quantum Search Algorithm; Mathematics Subject Classification. Primary 81P68; Secondary 81-01.

367 Kitaev Alexej (1995): Quantum measurements and the Abelian Stabilizer Problem; L. D. Landau Institute for Theoretial Physics

is applied, where $|\psi_d\rangle$ denotes the desired vector that is going to be measured, given from

$$Q = -HI_{|0\rangle}H^{-1}I_{|x_d\rangle} \qquad (10\text{-}128)$$

$I_{|0\rangle}$, or $I_{|\psi_d\rangle}$ in its basic form, being an inversion of the following character:

$$I_{|\psi\rangle} = I - 2|\psi\rangle\langle\psi| \qquad (10\text{-}129)$$

(10-129) shows that the unitary transformation $I_{|\psi\rangle}$ is an inversion in $\mathcal{H}$ about the hyperplane perpendicular to x_d, thus the value the algorithm is searching for (the I without subscript represents the identity transformation). Therefore, for any unit length ket $|\psi\rangle$, $I_{|\psi\rangle}$ is an inversion in $\mathcal{H}$ about the hyperplane orthogonal to $|\psi\rangle$.[368] From this, one can say that if $|\psi\rangle$ is a unit length ket in the Hilbert space $\mathcal{H}$ and U_f a unitary transformation on $\mathcal{H}$, then

$$U_f I_{|\psi\rangle} U_f^{-1} = I_{U_f|\psi\rangle} \qquad (10\text{-}130)$$

This means that U_f applied on the inversion of the ket $|\psi\rangle$, followed by the application of the inversion of the U_f, namely U_f^{-1}, results in $I_{U_f|\psi\rangle}$, thus the inversion of U_f on $|\psi\rangle$. It is important to know that the inversion of a unitary operator is always its adjunct. If this is not the case, the operator is not unitary. The result of equation (10-130) is then the inversion of $|\psi\rangle$ after U_f has worked on it. Back to Q and equation (10-127) this means that Q is a rotation of the state $|\psi_d\rangle$, which is an equal superposition of all the standard basis states, within $|\psi\rangle$ towards the desired value of the query $|x_d\rangle$ by a specific angle β. The rotation starts from two unit length vectors orthogonal to $|\psi_d\rangle$ and $|x_d\rangle$, namely $|x_d^{\perp}\rangle$ and $|\psi_d^{\perp}\rangle$ with the same origin and β being the angle between these. The application of the transformation (10-127) on these two vectors will result in a reflection in $|x_d^{\perp}\rangle$ followed by a reflection in $|\psi_d\rangle$, which is the same as a rotation by 2β. Summing up,

$$Q = -I_{|\psi_d^{\perp}\rangle} I_{|x_d\rangle} \qquad (10\text{-}131)$$

is a rotation of $|\psi_d\rangle$ by 2β towards $|x_d\rangle$ (Figure 87 - Rotation towards $|x_d\rangle$).[369]

368 Lomonaco Samuel J. Jr. (2000): Grover's Quantum Search Algorithm; Mathematics Subject Classification. Primary 81P68; Secondary 81-01.

369 Lomonaco Samuel J. Jr. (2000): Grover's Quantum Search Algorithm; Mathematics Subject Classification. Primary 81P68; Secondary 81-01.

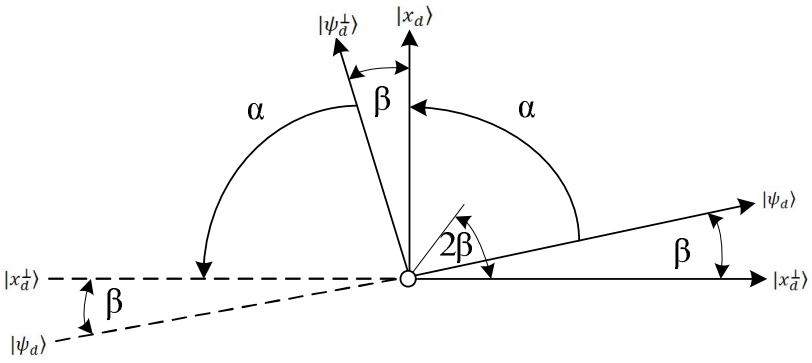

Figure 87 - Rotation towards $|x_d\rangle$

It is important to mention here that the rotations are carried out in Euclidian plane, thus the real 2-dimensional inner product space of $\mathcal{H}$. The definitions (10-132) and (10-133) help to understand this. Let assume that $|\phi\rangle$ and $|\psi\rangle$ are two Kets in $\mathcal{H}$ with a real dot product $\langle\phi|\psi\rangle$ (which, as already indicated, is the probability of $|\psi\rangle$ collapsing into $|\phi\rangle$), then (10-132) is the sub-space of $\mathcal{H}$ spanned by both $|\phi\rangle$ and $|\psi\rangle$:

$$S_{\mathbb{C}} = Span_{\mathbb{C}}(|\phi\rangle, |\psi\rangle) = \{\alpha|\phi\rangle + \beta|\psi\rangle \epsilon \mathcal{H} | \alpha, \beta \epsilon \mathbb{C}\} \qquad (10\text{-}132)$$

Then there must be a real inner space in $S_{\mathbb{C}}$:

$$S_{\mathbb{R}} = Span_{\mathbb{R}}(|\phi\rangle, |\psi\rangle) = \{a|\phi\rangle + b|\psi\rangle \epsilon \mathcal{H} | a, b \epsilon \mathbb{R}\} \qquad (10\text{-}133)$$

Thus, if $|\phi\rangle$ and $|\psi\rangle$ are linearly independent, then $S_{\mathbb{R}}$ is the mentioned 2-dimensional inner product space of $\mathcal{H}$.[370] Finally, the number of rotations needs to be determined, as too many rotations will rotate $|\psi_d\rangle$ past $|x_d\rangle$ and too few of them would lead to a stop way before it. Both are not desirable. From the latter explanations and Figure 45 - Rotation towards $|x_d\rangle$ it is obvious that

$$|\psi_d\rangle = sin\,\beta|x_d\rangle + cos\,\beta|x_d^{\perp}\rangle \qquad (10\text{-}134)$$

After n rotations, or n applications of Q the resulting state is

$$|\psi_n\rangle = Q^n|\psi_d\rangle = sin[(2n+1)\beta]|x_d\rangle + cos[(2n+1)\beta]|x_d^{\perp}\rangle \qquad (10\text{-}135)$$

Moreover, Lomonaco[371] describes that the target now is to find an integer n so that $\sin[(2n+1)\beta]$ is as close to one as possible, or in another term, an integer that $(2n+1)\beta$ is very close to $\frac{\pi}{2}$. As a consequence,

370 Lomonaco Samuel J. Jr. (2000): Grover's Quantum Search Algorithm; Mathematics Subject Classification. Primary 81P68; Secondary 81-01.

371 Lomonaco Samuel J. Jr. (2000): Grover's Quantum Search Algorithm; Mathematics Subject Classification. Primary 81P68; Secondary 81-01.

$$n = \frac{\pi}{4\beta} - \frac{1}{2} = \left\lfloor \frac{\pi}{4\beta} \right\rfloor \qquad (10\text{-}136)$$

However, for being able to calculate these equations, it is required to determine β at first. At first, it is important to know that the angle α is complimentary to β (see Figure 87 - Rotation towards $|x_d\rangle$):

$$\alpha + \beta = \frac{\pi}{2} \qquad (10\text{-}137)$$

It follows that

$$\frac{1}{\sqrt{N}} = \langle x_d|\psi_d\rangle = cos\,\alpha = cos\left(\frac{\pi}{2} - \beta\right) = sin\,\beta \qquad (10\text{-}138)$$

and according to this

$$\beta = sin^{-1}\left(\frac{1}{\sqrt{N}}\right) \approx \frac{1}{\sqrt{N}} \qquad (10\text{-}139)$$

Finally, the number of required rotations is given by

$$n = \left\lfloor \frac{\pi}{4\,sin^{-1}\left(\frac{1}{\sqrt{N}}\right)} \right\rfloor \approx \frac{\pi}{4}\sqrt{N} \qquad (10\text{-}140)$$

Although the algorithm is generally described as database search algorithm, it would be more suitable to describe it as function inverter. This is, because for a given function $f(x)$ the algorithm is able to determine y. Ricks and Ventura[372] made a very interesting proposal of how the optimal solution may be determined by a generalization of the of Grover's algorithm initially presented by Boyer et al.[373]

Another approach could be as follows: the output of a node would be $f(x)$ and according to Grover's algorithm the determination of y is possible, which has to happen with a quantum search routine U_f. This search routine must then calculate backwards through the network, which is quite different from any other approach in neural network learning. Usually, y is given and one tries to determine $f(x)$ and adapts $f(x)$ through a learning algorithm as long as it is required to fulfil a stopping criterions, like the RMSE. However, as only $f(x)$ is given, U_f is required to find the correct input to the desired output. Thus, the calculated output must be taken and the calculation must go backwards. Let assume the perceptron equation is as follows:

$$|f(x)\rangle = tan(\textstyle\sum_{i=1}^{n} |\omega\rangle_i\, |x\rangle_i + \theta) + \epsilon_t \qquad (10\text{-}141)$$

Then U_f must be

$$|x\rangle_i = \frac{arctan(|y\rangle_i - \epsilon_t) - \theta}{\sum_{i=1}^{n} |\omega\rangle_i} \qquad (10\text{-}142)$$

372 Ricks Bob, Ventura Dan (2003): Training a Quantum Neural Network; Provo: Brigham Young University

373 Boyer Michael, Brassard Gilles, Hoyer Peter, Tapp Alain (1996): Tight Bounds on Quantum Searching, Fourth Workshop on Physics and Computation

and the error calculation

$$|y_{rmse}\rangle = \sqrt[2]{\frac{1}{n}\sum_{i=1}^{n}|y^2\rangle_i} \qquad (10\text{-}143)$$

where n represents the number of outputs and y the input values.

However, again and before that, the input register consisting of a number of n Qbits has to be put into superposition as it has already been done in equation (10-66):

$$|\phi\rangle = H^{\otimes n}|0\rangle_n = \frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle_n \qquad (10\text{-}144)$$

Also, the already introduced unitary transformation V is required, plus an additional unitary transformation W acting on the input register as V with a fixed form not depending on a and preserving the component of any state along the standard (initial) state $|\phi\rangle$, but changing the sign of its component orthogonal to $|\phi\rangle$:

$$W = 2|\phi\rangle\langle\phi| - 1 \qquad (10\text{-}145)$$

$|\phi\rangle\langle\phi|$ representing the projection operator on $|\phi\rangle$.[374] Given these transformations, Grover's algorithm applies the Product WV many times onto the input register in $|\phi\rangle$. Furthermore, each invocation of WV requires the quantum search routine or unitary operator U_f to be executed, which must be able to work with the superpositions of the QANN's states and which compares the entries of the database, or in the case of a quantum artificial neural network, the desired output with the calculated output. Summing up, in both algorithms the quantum search seeks to let the system fall into decoherence when the probability amplitudes for measuring the desired state near unity.

10.5.1.8 Envisaged implementations of a quantum artificial neural network

At first, the basic analogue to a hard disk or RAM for representing bits must be found. A suitable option are fermions, or spin-1/2 objects, because the possible spin directions allow the representation of $|0\rangle$ by spin up with $|\uparrow\rangle$ and of $|1\rangle$ by spin down with $|\downarrow\rangle$ along the z-axis. In a 3-d real vector space, the complex amplitudes of a Qbit $\alpha_1|0\rangle + \alpha_1|1\rangle$ are then described by

$$\alpha_1 = e^{\frac{-i\phi}{2}}cos\frac{\theta}{2} \qquad (10\text{-}146)$$

$$\alpha_2 = e^{\frac{i\phi}{2}}cos\frac{\theta}{2} \qquad (10\text{-}147)$$

where θ is the polar angle (the angle between the vector and the z-axis), and ϕ is the azimuthal angle (the angle between the x-axis and the vector's projection onto the two-dimensional x|y-plane). The angle between the vector (formerly described as a ray in Hilbert-space) and the z-axis determines the probabilities of obtaining $|\uparrow\rangle$ or $|\downarrow\rangle$ along this axis,

374 Mermin David N. (2007): Quantum Computer Science: An Introduction; Cambridge: Cambridge University Press

whereas the azimuthal angle represents the relative phase. Thus, the vector points into the positive range of z, when

$$\alpha_1 = |\downarrow\rangle \wedge \alpha_2 = |\uparrow\rangle \qquad \textit{(10-148)}$$

and into the negative range of z, when

$$\alpha_1 = |\uparrow\rangle \wedge \alpha_2 = |\downarrow\rangle \qquad \textit{(10-149)}$$

which is a representation of the general Qbit, existing in 2-d Hilbert-space, in 3-d real vector space.[375] Apart from that, there are several difficulties one faces when trying to implement a quantum computer. One of the major ones is the elimination of the physical system's (that a quantum computer certainly is) own irrelevant properties, so that the superposition, or coherence, can be established. This is not only difficult when thinking of possible implementations of a quantum artificial neural network, but a general issue. However, there may be a chance of implementing quantum systems capable of processing a quantum artificial neural network before standard quantum computers may be implemented. This is, because the first implementations will possibly target auto-associative artificial neural networks, not relying on double values for processing, but on but on signed 2bit-values values (-1, 0, 1). Thus, the architecture would not consist of 64 Qbits, but on 2 Qbits. Given such an auto-associative quantum artificial neural network for recognizing patterns consisting of 3 input values would just require 3 Qbits in case of a Hopfield artificial neural network (Figure 88 - Quantum Hopfield artificial neural network), and maybe 7 (depending from the number of the hidden classifiers) in case of a Boltzmann machine (Figure 89 - Quantum Boltzmann machine). Furthermore, there exist proposals for the implementation of quantum hidden Markov models.[376]

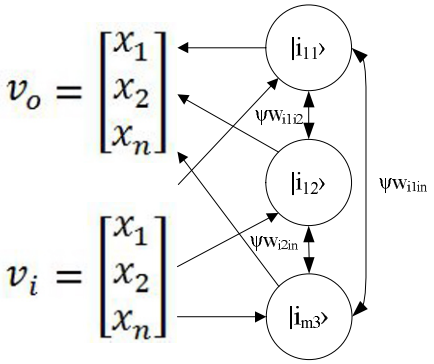

Figure 88 - Quantum Hopfield artificial neural network

375 Jyh Ying Peng (2003): Quantum Computation Lecture Notes [2012-10-06]; URL: http://red.csie.ntu.edu.tw/QC/peng/chap1.pdf

376 Monras Alex et al. (2012): Hidden Quantum Markov Models and non-adaptive read-out of many-body states [2012-10-22]; URL: http://arxiv.org/abs/1002.2337v2

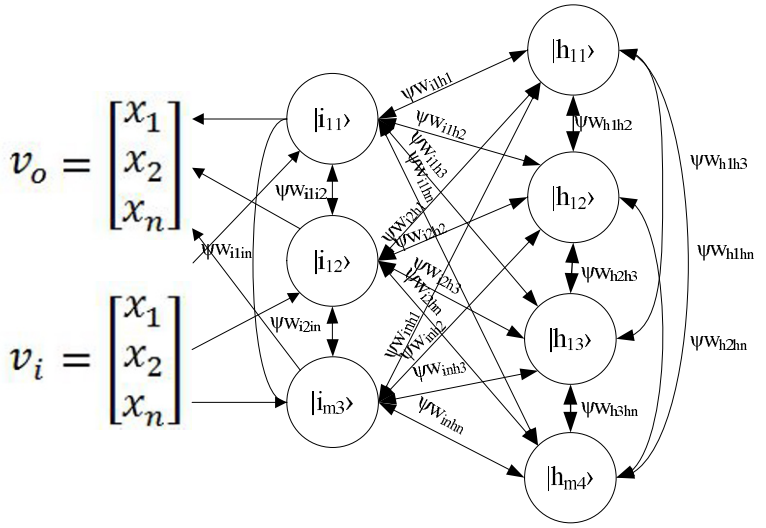

Figure 89 - Quantum Boltzmann machine

Another challenge is the realization of the required interconnection of the network's neurons. The calculation of the connection weights happens in the perceptron equation and thus creating entanglement of the calculation bits. Therefore, the weighted neuron connections exist as entangled Qbit-states. According to DiVincenzo, physical systems implementing quantum computers must provide the following (DiVincenzo criteria):

- A scalable system of well-characterized Qbits,
- which must offer the possibility of proper initialization and
- which must have much longer coherence times than the time scales required for the fundamental operations,
- a universal set of quantum gates and
- a Qbit-specific measurement.

Additionally, DiVincenzo demands for

- the ability to interconvert stationary and flying Qbits and
- the ability to faithfully transmit flying Qbits between specified locations.[377]

Technically, there exist no fundamental obstacles in implementing a fault-tolerant quantum computer; however, there is no best approach, although a few top candidates exist. The implementation of a quantum system being capable of processing an artificial neural network may be realized by the use of the following systems:

377 DiVincenzo David. P. (2001): Dogma and heresy in quantum computing; Quantum Information Comp. 1, 1.

10.5.1.8.1 Adiabatic quantum annealing

One potential implementation and the most promising one, although completely different in architecture and not developed with the target of achieving universal quantum computing, is the adiabatic quantum annealing model. A basic building block is the SQUID (superconducting quantum interference device), and interference in that case refers to the fact that electrons produce interference patterns resulting in quantum effects. This means that the SQUID behaves as a Qbit, which is due to the fact that these are implemented as tiny metal-rings made of niobium. Niobium, when cooled down, becomes a superconductor, meaning that charged particles traveling through it do not experience any resistance. The SQUID represents the two possible states of a Qbit by magnetic fields, pointing up or down, which is caused by electrons travelling in both directions in the ring at once (which is only possible in a superconductor). Controlling the amount of charge flowing in either of the directions allows direct manipulation of the probability amplitudes.

Multiple Qbits must be coupled, and in a quantum annealer this happens with Josephson junctions, also called couplers, which are made of superconducting rings as well. If several Qbits and couplers are brought together, a quantum neural network can be implemented directly in hardware. Finding an optimal solution in an ANN is an optimization problem, in which we aim to minimize the error. If translated into an optimization function, the D-Wave[378] represents the errors surface of an ANN by an energy surface, and what the optimization function does is try to find the minimum energy configuration of the system. The objective function is given by

$$Obj(a_i, b_{ij}; q_i) = \sum_i a_i q_i + \sum_{ij} b_{ij} q_i q_j \qquad (10\text{-}150)$$

where q_i represents the Qbit participating in the annealing cycle and finally converging to a computational basis state, $q_i q_j$ is the coupler (Josephson junction) that allows one Qbit q_i to influence another Qbit q_j, the weight a_i is a real-valued constant associated with Qbit q_i (the probability amplitude), and b_{ij} is a real-valued constant controlling the strength of influence carried out by one Qbit q_i to another Qbit q_j.

Thus, the D-Wave samples from the q_i minimizing the objective and the translation of a problem into the objective function must be done by the programmer.

10.5.1.8.2 Nuclear magnetic resonance

NMR seems to be a promising approach for implementing a quantum artificial neural network, as although the number of bits is currently limited, linear superposition can be kept up to milliseconds, thus long enough for allowing calculation, quantum search and measurement. Chuang and Gershenfeld[379] made an experimental verification using a nuclear magnetic resonance system made up of two Qbits, the first having been the spin of a carbon isotope's (C^{13}) nucleus, the second having been described by the spin of a hydrogen ion's (H^+) proton

---

378 D-Wave (2016): The D-Wave 2X System [2016-07-21], URL: http://www.dwavesys.com/d-wave-two-system

379 Gershenfeld Neil, Chuang Isaac L. (1996): Quantum Computing with Molecules [2012-12-19]; URL: http://www.mat.ucm.es/catedramdeguzman/old/01historias/haciaelfuturo/Burgos090900/quantumcomputingSciAmer/0698gershenfeld.html

spin. However, although their system is a real quantum system, they compute the statistical average of many copies of the system, thus molecules.

10.5.1.8.3 Others

Other promising systems are quantum dot systems, consisting of an electron, trapped in an atomic cage, ion traps, or quantum electrodynamics of atoms in optical cavities.

10.6 The artificial neocortex

"Consciousness" is one of these suitcase words from psychology that helps us to discuss a complex subject no scientist or philosopher has been able to describe or demystify for millennia. It enables us to include yet unknown processes and (changes of) states associated with the human brain in our everyday-language with ease, not only without being able to describe what accounts for a conscious experience on neuronal or (sub-) atomic layers, but also without being able to explain what consciousness is on a more abstract layer. Lots of scientists from different fields have been working on disclosing the secret of consciousness, and numerous different explanations have been published and discussed controversially. Most of these theories feature a common denominator – the inclusion of a feature set, which is associated with the perception of consciousness. It would be counterproductive to reject such approaches, as only the detailed description and combination of single features will allow us to reproduce conscious behavior in artificial entities. However, lots of these theories share one more thing: they try to simplify the enormous complexity of different information processing levels, such as self-conscious reflection, (self-) reflective or deliberative thinking, or subjectivity in conscious experiences, which, amongst others, incorporates learned behavior and experience. Simplification has been very applicable for physical theories, but we assume that more complex theories are required to approach the explanation of which features, brain states or processes consciousness comprises. We will not provide such a theory here, as this would go beyond the scope of this elaboration, but we will approach such one by starting to combine and technically explain features that we consider such a theory needs to take into account.

Although there has always been criticism on ANNs over the last decades, we consider them as a useful tool for implementing consciousness in an artificial entity. Taking deep learning ANNs as one of the foundations for processing, it may be possible to implement powerful structures such as context classifiers, used for interpreting the context of a situation. The description of quantum artificial neural networks has shown that in theory networks with arbitrary depth can be created and processed, such as the human brain is one. It is obvious that not only one artificial neural network alone will be capable of processing all information provided by sensory inputs, such as vision. Thus, we consider that the following structures and methods are a good starting point for achieving what we are looking for, however not all of these will be dealt with in detail, as this goes beyond the scope of this book:

- Search and optimization
  - (Automated) reasoning (i.e. through trees of goals and subgoals, inference engines, theorem provers, classifiers)
  - Access to knowledge (i.e. index-based search, weighted result sets)
- Logic and reasoning
  - Logic (i.e. first order logic, formal logic, description logic)

  - Deductive and inductive reasoning (i.e. logic, mathematical reasoning, knowledge-based reasoning, reducing problem to search)
  - Planning (i.e. search through trees of goals and subgoals, means-ends analysis, logic)
  - Decision making (i.e. knowledge-based reasoning)
  - Data-based reasoning
  - Uncertain reasoning (probabilistic methods)
- Perception, pattern recognition and -understanding
  - Object recognition and cene understanding (i.e. ANNs + access to knowledge)
  - Context recognition and -classification (i.e. logic and reasoning + ANNs)
  - Clustering of information (i.e.ANNs)
  - Long short-term memory (i.e.LSTM ANNs)
  - Ability to make predictions (i.e.ANNs)
- Natural language
  - Understanding and deriving meaning from text (i.e. deep ANNs, text mining, hierarchical analysis and representation of language)
  - Natural language understanding (i.e. segmentation, part-of-speech tagging, disambiguation)
  - Natural language generation (i.e., conversion of information in databases in human readable language)
  - Information retrieval (i.e. storing, searching, retrieving information)
- Conscious experiences
  - Understanding of concepts (i.e. by hierarchical structure of language, abstraction in deep neural networks)
  - Conscious experiences (i.e. collapse of a superposition in a large quantum neural network from a multitude of possibilities to one definite state)

Additionally, access to knowledge is required, thus to

- (knowledge) databases,
- ontologies,
- semantic nets,
- systems architecture,
- frames,
- rules,
- indices via search engines,
- expert systems, and
- information processors, thus agents
  - collecting
  - processing,
  - ranking, and
  - selecting context-sensitive information.

From the listing we can see that I suggest not only classical means of information technology and artificial intelligence, such as databases containing information or expert systems correctly accessing context-sensitive information, but also quantum physical approaches, which has also been the reason to going into detail with quantum mechanics and especially quantum artificial neural networks. It cannot be taken for granted that an artificial mind can only be implemented by merging both fields to one strong component. Even if our brain does not work in that manner, so that e.g. for learning or the execution of some limb motion

information access and processing may happen on the classical neural level only, and experiencing conscious content may involve signal processing on the quantum physical layer as well, a combination for an ACE seems to be promising.

I will start the explanation of my ideas by discussing two further concepts, namely the one of context recognition as well as the concept of hierarchies. This is, because one of the major powers of our brains is that we can extract context from a situation. For this and other paradigms, such as language processing, to work, it is required to understand hierarchical information structuring and processing. The first concept mainly concerns the acquisition of knowledge and is discussed in the following part, whereas the second concept has to do with the organization of patterns in our (and the artificial entity's) brain, which is discussed at 10.6.3 Implementation. I already mentioned that natural language processing (and understanding) is one of the key concepts for the human brain development, and therefore I will explain the proposed artificial neocortex based on the implementation of language processing starting with chapter 10.6.2 Context recognition and hierarchical learning. Before that, it is required to understand how knowledge can be represented, accessed and stored.

10.6.1 Knowledge and data

To start with, it is required to understand what knowledge exactly is, and in the field of knowledge representation research is concerned with only that. Knowledge representation within knowledge modeling is to formally reflect knowledge in knowledge-based systems. For this, various formal languages and notations have been proposed. A collection in this way represented knowledge is called knowledge; in the semantic web formalized knowledge is stored in a distributed way. In contrast to the representation of knowledge, the focus of the organization of knowledge is more on the order of existing knowledge that not even shown, but are described by metadata. Means of knowledge representation are, amongst others, applied at the construction of expert systems, machine translation programs, systems for computer-assisted maintenance and database query programs.

In ordinary language, a person possesses knowledge about an issue if the following conditions about an issue if the following three conditions are true:

1. the knowledge carrier keeps the facts to be true
2. the speaker also keeps the facts to be true
3. the knowledge carrier can describe the facts

The third condition also requires that the verb "to know" is used to describe conscious states, that is, those that can be verbalized.

In computational linguistics and AI research one speaks of knowledge, however, even if the above conditions 2 and 3 are not met, that is, even where it comes that someone believes something or thinks, knows, or can something. When using "can" is clear that it is not just about facts, but also to methods and procedures is (procedural knowledge). With the distinction between "know" and "can" the distinction between so-called declarative and procedural knowledge has already been touched upon. In AI there exists a distinction accordingly between declarative and procedural forms of knowledge representation. It should be noted, though, that procedural knowledge and procedural knowledge representation do not necessarily need to cover. Declarative knowledge can be represented procedurally as well, and vice versa.

– Procedural knowledge representations describe methods for design, link and application of knowledge.
– Methods for controlling the use of declarative and procedural knowledge descriptions are called control knowledge. Control knowledge is meta knowledge.

10.6.1.1 Knowledge representation

Several forms of knowledge exist per definition, and also several ways of representing knowledge in order to be used in an artificially intelligent system. We distinguish between four different kinds of knowledge:

– Objects: typically, we consider knowledge as the knowledge of facts about objects in the world around us: Cars have tires. Some mountains are high. We therefore have to represent objects, classes or categories of objects, descriptions of objects and relationships between objects.
– Events: we also possess knowledge of processes and events in the world: A car sped at 220 km/h in the city. Besides the presentation of the events themselves, a representation formalism possibly also contains the timing of a sequence of events and the difference between the cause-effect-relationships.
– Practical knowledge: a capacity requires not only knowledge of objects and events also knowledge about how certain actions are to be executed. Also, most cognitive skills such as the formation of sentences or the theorem proving require such action knowledge.
– Meta knowledge: we also make use of knowledge about our knowledge, so-called meta knowledge. For example, we know something about the scope and origin of our knowledge of a specific subject on the reliability of certain information, or about the relative importance of specific facts about the world. For meta-knowledge also includes the assessment of our own cognitive abilities and knowledge about ways of acquiring knowledge.

The most important criterion for the assessment of various methods of knowledge representtation is what use of the illustrated knowledge ultimately is to be made. The processing of knowledge in AI systems involves three main stages:

– The acquisition of new knowledge (knowledge acquisition).
– Finding the relevant facts regarding the current problem in the knowledge base (knowledge retrieval)
– Reasoning (reasoning) with these facts in search of a solution to the problem.

The knowledge used and manipulated by expert systems corresponds to what the simulated procedure of specialists is based upon. Depending on whether knowledge content is described as “passive data” or applicable procedures leads to different forms of knowledge represetation:

10.6.1.2 Declarative knowledge representation

Declarative representations of knowledge content give descriptions of situations which do not contain information about the construction and use of knowledge, i.e. “the sum of 5 and 4 is 9” or the formula: $5 + 4 = 9$. Declarative forms of knowledge representation are

- Semantic networks
- Object-attribute-value-triples
- Frames (schemes, scripts)
- Production rules
- Predicate logic

10.6.1.2.1 Semantic networks

Semantic networks (see Figure 90 – Semantic network): semantic networks were originally designed as psychological models of cognitive structures such as associations in memory, and later to describe the semantic structure of sentences. A semantic network is a directed graph of a set of nodes, which represent objects (terms and concepts), as well as an amount of directed edges (arcs or links) representing relations between objects. Usually both nodes and edges are named.

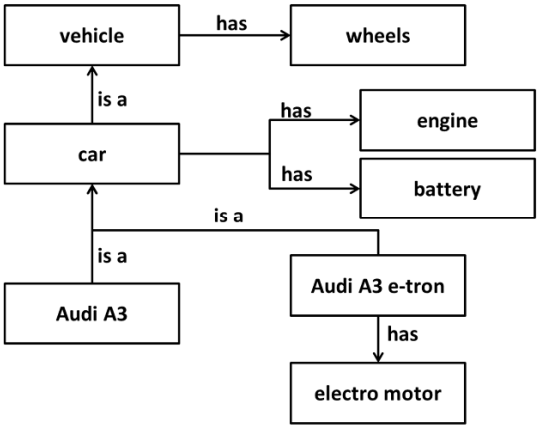

Figure 90 – Semantic network

Nodes are used to represent objects and descriptors. Objects can be physical objects that one can see or touch. Objects can also be mental elements, e.g. actions, events or abstract categories, and descriptors provide additional information (attributes, properties) on objects.

Edges represent relations, connecting objects and descriptors. Some frequent connections are:

- Is-a: represents the relation between the class and case, i.e. “An Audi A3 is a car.” Often, however, but also the subset relationship or a subcategory is represented therewith, i.e. “A car is a means of transportation.” These two uses should however be distinguished. For example, the relation between case and class could be described by “element-by” or “instance of”.
- Has: has-connections identify relations between parts and sub-elements, i.e. “A car has an engine”.

One of the main advantages of this representation scheme is its flexibility. New nodes and links can be defined as required. If we, for example, represent an electric machine as a network, then the nodes can represent items, “wired-with” other items. An essential feature of networks is the inheritance of properties, which is the basis for knowledge manipulation in such networks.

Inheritance refers to the fact that a node can inherit the characteristics of other nodes to which it is connected. The inheritance of properties is due to the is-a relation, and means that all individual cases of a class take over all properties of the parent class to which they belong.

From the foregoing presentation it is clear that semantic networks are suitable for representing relationships that would appear in predicate logic as two-place predicates. The question then is how three or more digit predicates can be represented, i.e. “Alice gave the key to Bob.” For this, the whole sentence is first conceived to be an instance of an event class “give”. There are a number of process operators who are in certain respects to the event (see Figure 91 – Representation of n-digit predicates):

– Agens = agent
– Patiens = entity directly affected by the action
– Recipient

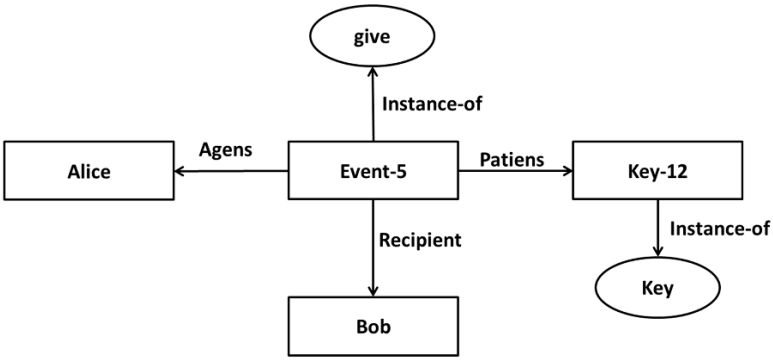

Figure 91 – Representation of n-digit predicates

10.6.1.2.2 Object-attribute-value-triplet

Another common method to represent knowledge content is the representation of an object-attribute-value triplet or O-A-V-triplet (associative triplet). This representation scheme has i.e. been used in the medical expert system MYCIN. This is a special case of representation by semantic networks (see Figure 92 – O-A-V-triplet).

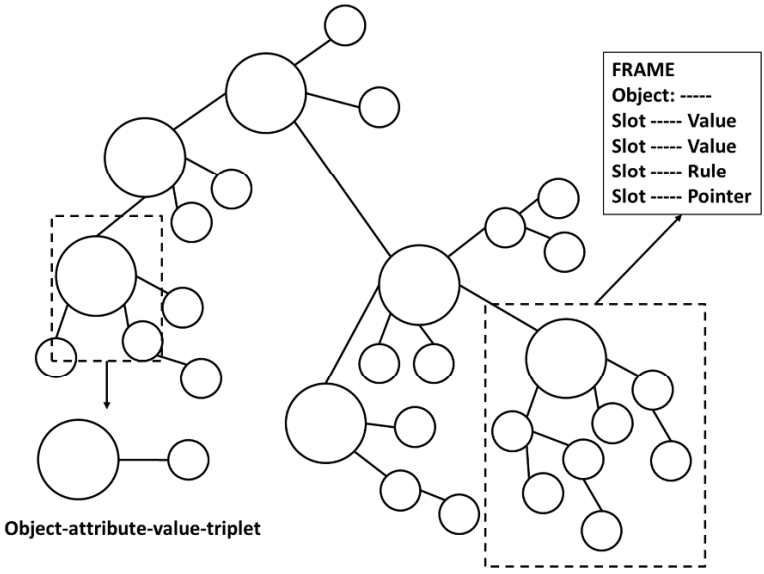

Figure 92 – O-A-V-triplet

Conceptual units can be either objects or physical entities. Attributes are general characteristics or properties that are associated with objects. Size, shape and color are typical attributes of physical objects. The third element of the triple is the value of an attribute. The value indicates the specific nature of an attribute in a particular situation. Table 3 – O-A-V-triplet provides an example.

Table 3 – O-A-V-triplet

Object	**Attribute**	**Value**
Rose	Color	Red
Rose	Origin	Germany
Rose	Durability	Medium

10.6.1.2.3 Frames

Frames have been introduced by Marvin Minsky as means of representation in image processing: When one encounters a new situation (or makes a substantial change in one's view of the present problem), one selects from memory a structure called a frame. This is a remembered framework to be adapted to fit reality by changing details as necessary. A frame is a

data-structure for representing a stereotyped situation, like being in a certain kind of living room, or going to a child's birthday party. Attached to each frame are several kinds of information. Some of this information is about how to use the frame. Some is about what one can expect to happen next. Some is about what to do if these expectations are not confirmed. We can think of a frame as a network of nodes and relations…[380] A frame is a grouping of nodes and attribute-value pairs in a semantic network describing a stereotyped object, an action, or an event in its entirety. Thus, a frame is a data structure that includes all knowledge about a particular object. Knowledge in a frame is partitioned into slots, in declarative (e.g., the color of a car) and procedural knowledge (e.g., activate a certain rule if a value exceeds a given level). The content of a frame are slots and facets, where by a slot is a set of attributes describing the object, and a facet describes knowledge about attributes in the slot. One slot contains one or more facets. Frames may also contain other frames or sub-frames. Examples for frames are Table 4 – Car-frame and Table 5 – Engine-frame.

Table 4 – Car-frame

Automobile frame
Class of: Transportation
Name of manufacturer: AUDI AG
Origin of manufacturer: Germany
Model: Audi S3
Type of car: Sportback
Weight: 1380 kg
Wheelbase: 2595–2637 mm
Number of doors: 5
Transmission: 6 gear S-tronic
Number of wheels: 4
Engine: – Type: TFSI – Number of cylinders: 4
Acceleration 0-100: 4.9 – 5.3 s
Gas mileage 100 km: 6.9–7.0 l

380 Minsky M. (1981): A Framework for Representing Knowledge; In: J.Haugeland (ed.); Mind Design; Cambridge; MA: The MIT Press, 95-128.

Table 5 – Engine-frame

Engine frame
Cubic capacity: 1984 cm³
Ventiles: 16
CO_2-emission combined: 159–162 g/km
Fuel system: TFSI
Horsepower: 300
Torque: 380 Nm/1800–5500

Frames can initially be regarded as a partial view in a semantic network in which all with O-A-V-triples associated with an object are combined into a whole. Thus, a semantic network is a frame hierarchy.

Knowledge-based systems

The need to represent knowledge in a way usable for software firstly found application in knowledge-based system (KBS). A KBS reasons and uses a knowledge base to solve complex problems. The term is broad and is used to refer to many different kinds of systems. The one common theme that unites all knowledge based systems is an attempt to represent knowledge explicitly via tools such as ontologies and rules rather than implicitly via code the way a conventional computer program does. A knowledge based system has two types of sub-systems: a knowledge base and an inference engine. The knowledge base represents facts about the world, often in some form of subsumption ontology. The inference engine represents logical assertions and conditions about the world, usually represented via IF-THEN rules.[381]

Early knowledge-based systems were primarily expert systems. In fact, the term is often used synonymously with expert systems. The difference is in the view taken to describe the system. Expert system refers to the type of task the system is trying to solve, to replace or aid a human expert in a complex task. Knowledge-based system refers to the architecture of the system, that it represents knowledge explicitly rather than as procedural code. While the earliest knowledge-based systems were almost all expert systems, the same tools and architectures can and have since been used for a whole host of other types of systems. I.e., virtually all expert systems are knowledge-based systems but many knowledge-based systems are not expert systems.

The first knowledge-based systems were rule based expert systems. One of the most famous was MYCIN, a program for medical diagnosis. These early expert systems represented facts

381 Smith Reid (1985): Knowledge-Based Systems Concepts, Techniques, Examples [2016-07-14]; URL: http://www.reidgsmith.com. Schlumberger-Doll Research

about the world as simple assertions in a flat database and used rules to reason about and as a result add to these assertions. Representing knowledge explicitly via rules had several advantages:

- Acquisition & Maintenance. Using rules meant that domain experts could often define and maintain the rules themselves rather than via a programmer.
- Explanation. Representing knowledge explicitly allowed systems to reason about how they came to a conclusion and use this information to explain results to users, i.e. following the chain of inferences that led to a diagnosis and use these facts to explain the diagnosis.
- Reasoning. Decoupling the knowledge from the processing of that knowledge enabled general purpose inference engines to be developed. These systems could develop conclusions that followed from a data set that the initial developers may not have even been aware of.[382]

As KBS became more complex the techniques used to represent the knowledge base became more sophisticated. Rather than representing facts as assertions about data, the knowledge-base became more structured, representing information using similar techniques to object-oriented programming such as hierarchies of classes and subclasses, relations between classes, and behavior of objects. As the knowledge base became more structured reasoning could occur both by independent rules and by interactions within the knowledge base itself. For example, procedures stored as demons on objects could fire and could replicate the chaining behavior of rules.[383] Another advancement was the development of special purpose automated reasoning systems called classifiers. Rather than statically declare the subsumption relations in a knowledge-base a classifier allows the developer to simply declare facts about the world and let the classifier deduce the relations. In this way a classifier also can play the role of an inference engine.[384]

The most recent advancement of knowledge-based systems has been to adopt the technologies for the development of systems that use the internet. The internet often has to deal with complex, unstructured data that can't be relied on to fit a specific data model. The technology of knowledge-based systems and especially the ability to classify objects on demand is ideal for such systems. The model for these kinds of knowledge-based Internet systems is known as the Semantic Web.[385] The term is broad, and is used to refer to many kinds of systems; examples include IBM's Watson[386] and the Wolfram Language.[387]

---

382 Hayes-Roth F., Waterman D., Lenat D. (1983): Building Expert Systems. Addison-Wesley.

383 Mettrey W. (1987): An Assessment of Tools for Building Large Knowledge- BasedSystems; AI Magazine 8 (4)

384 MacGregor R. (1991): Using a description classifier to enhance knowledge representation; IEEE Expert 6 (3): 41–46

385 Berners-Lee T., Hendler J., Lassila O. (2001): The Semantic Web A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities; Scientific American 284: 34–43

386 ibm.com: What is IBM Watson? [2016-07-14]; URL: http://www.ibm.com/watson/what-is-watson.html

387 wolfram.com: Wolfram Language for Knowledge-Based Programming [2016-07-14]; URL: https://www.wolfram.com/language/

10.6.2 Context recognition and hierarchical learning

Now that it is clear how knowledge can be represented, it is required to define what context actually means. Our brain is capable abstracting information, thus it can convert a situation rich on information into something more common, by implementing an inductive thought process and leaving off details. If we e.g. take a photograph picturing a laughing man and a woman, sitting in a meadow on a blanket under a tree, between them two glasses of wine and some food, then the context is that they are picnicking. A lot of information that may be depicted on the picture is irrelevant for a human brain correctly recognizing the context, e.g. it does not matter what exactly is on the blanket, or which clothes both wear. What is obvious for us is difficult to implement so that an artificial conscious entity is able to conclude the same. Before dealing with visual information, we will try to solve the problem with textual information such as speech or texts describing a situation. In terms of textual information, an example for a context problem is as follows: Let assume that there are three textual propositions:

1. Injury
2. Car
3. Speed

Let further assume that there exist two contexts:

a) Accident
b) No accident

Textual proposition 1 and 2 are true in the context of a, whereby propositions 2 and 3 are true in the context of b. Without further information, this is not only a complex situation for an artificially intelligent software system, but also for humans.

The meaning of context permits the reconciliation of statistical measures of context examination utilizing the terms of context formalization, and for a series of propositions there exists a collection of sets of contexts, and the outer context C is defined by

$$true\left(C, \bigcap_{i=1}^{m} true(C_{ij}, T_i)\right) \forall j \qquad (10\text{-}151)$$

where $T_1, \ldots, T_m$ represents as a series of textual propositions, and in case $\forall T_i$ there is a set of contexts C_{ij}. For each i then $true(C_{ij}, T_i)\ \forall j$, which states that P_i is true for each C_{ij}, and C_{ij} are not structured hierarchically in advance, instead such are created according to specific sets of textual propositions. The number of existing contexts is assumed to be finite and to satisfy

$$C, C_{ij} \subseteq U_c \qquad (10\text{-}152)$$

where U_c is the unity of all existing contexts. This results in two sub-problems:

1. What are the possible contexts C_i satisfying $true(C_i, T)\ \forall i$ if T is defined as single text?
2. If $T_1, \ldots, T_m$ is set of textual propositions satisfying the condition that for each text T_i there is a set of contexts C_{ij} so that $true(C_{ij}, T)\ \forall i$. What is the outer context C so that

$$true\left(C, \bigcap_{i=1}^{m} true(C_{ij}, P_i)\right) \forall j \qquad (10\text{-}153)$$

The first problem requires some text T to be received, whereby each of the texts may feature a set of contexts that are true for T. An example is the phrase 'to put away', which can represent different activities, such as 'scoffing', or 'tidying away'. The challenge is how to figure out these possible contexts from all existing contexts.

The second problem goes one step further in the sense that more than one text is presented, namely a series $T_1, \ldots, T_m$. Each of these texts T_i features a number of contexts and the challenge is to select the correct combined contexts of the remaining contexts for each T_i. The most suitable contexts are chosen by ranking in the stage of specialization and refinement. Considering the first given example, there exist the text propositions 'injury', 'car', and 'speed' with the two existing contexts 'accident' and 'no accident', whereby the textual propositions 'injury' and 'car' are true in context 'accident', while the propositions 'car' and 'speed' are true in context 'no accident'. We can see that the outer context would include both 'accident' and 'no accident', which is a contradiction resulting from a lack of information. More information would result in higher contextual ranking and another outcome.[388]

10.6.2.1 Definition of context-sensitive information

A textual context description consists of phrases, words or sentences, each of these describing accurately one facet of the full context. First of all, it is required to consolidate the data, which may be delivered in different forms, such as sentences, words or phrases. A simple procedure then is responsible for splitting the text into single words, which are then compared with the content of some dictionaries. The first (exclusion-)dictionary may be built upon words that do not help to understand the context of a situation, such as personal pronouns or articles. As a next step, it would be required to classify the remaining words, thus to translate them into knowledge. For this, additional domain-specific dictionaries are required, the number depending on what knowledge the artificial entity should be able to process. We have to bear in mind that also humans do not feature domain-spanning knowledge, so we should not presuppose that an artificial entity can do that at first. However, in theory there are not bounds. Each single word is then compared with the contents of the knowledge dictionaries and will be considered for further context analysis, if it is found. A short algorithm describing the situation is as follows:

Start

1. Collect and split text
2. Create context-array A_c and exclusion-array A_e
3. ***For each*** word
   a) compare word with exclusion-dictionary
      If
         word matches with entry in exclusion-dictionary
         update A_e *with word*
      Else
         update A_c with word

388 Segev Aviv et al. (2007): Context recognition using internet as a knowledge base; Journal for Intelligent Information Systems; 29:305–327

```
4. For each
   entry in A_c
      a) compare word with knowledge-dictionaries
         If
            word matches with entry in knowledge-dictionaries
            do nothing
         Else
            remove word from A_c
End
```

Algorithm 31 – Word comparison

Breakdown:

A_c: Array list holding the words used for context-analysis
A_c: Array list holding the words ignored at context-analysis

Basically, this can be compared with how context-sensitive information in brains is pre-processed. Information useless for exactly interpreting what happens in a situation is removed – this has been mentioned beforehand as abstraction.

10.6.2.2 Information Clustering

The textual information gathered in the last step will be used to query one or many information clusters, such as our brain would do. When again considering the example of two picnicking persons, then our brain would not search for information regarding the situation in our memories of cars or houses. Instead, our hierarchical thinking starts with conceiving the overall information on the picture to specific information depending on memories, knowledge and experience. This is what we also have to do with the memory of the artificial entity, such as a context-database. A context-database may be implemented by leveraging the power of distributed file system oriented databases and graph databases for encoding relations, allowing the storage and management of hundreds of millions of files in combination as well as exploiting the knowledge representation forms discussed in chapter 10.6.1 Knowledge and data. However, before querying specific files, thus knowledge, defined by specific contexts, it is required to get back to the self-organizing feature maps discussed in 10.3.3 The transition to the human brain. As our human brain, this specific kind of artificial neural networks is capable of clustering content. Furthermore, human thinking happens hierarchically, which will also be considered in the introduced approach.

As an artificial entity should also be able to learn new things and to extend its knowledge, continuous clustering of information is absolutely required. Otherwise, each context interpretation would happen over the entity's overall knowledge, which would result in incorrect interpretation with high probability. Therefore, it is required to automatically detect and create preliminary contexts on existing knowledge (which we have to differ from preliminary contexts one level deeper, of which more lately). The approach applied is a tree view based

hierarchical document clustering approach.[389] As the SOFM is an unsupervised learning ANN, no prior training is required, which makes it a perfect approach for clustering a huge amount of unstructured data, which the knowledge of the entity in consideration without doubt should be. Before being able to cluster the information, some pre-processing must be applied, such as indexing by the vector space model (VSM)[390] for determining the occurrence frequency of words or terms within a document, resulting in a juxtaposition of the term occurrence frequency in documents. Again, a so-called stop-list or exclusion-dictionary is applied on all the documents contained within the distributed file system database for discarding information with little knowledge and importance for context interpretation, followed by the application of stemming algorithms.[391] The VSM-term-document-matrix forms the basis for further processing, such as described in 10.6.2.3 Context analysis.

10.6.2.3 Context analysis

After only the context-sensitive textual information has remained, it has to be compared with information stored in context-databases. As already mentioned, a context-sensitive database may be implemented by a distributed file system oriented database, such as Apache Hadoop, in the sense that stored contextual information refers to clusters of files. This context-sensitive database is then queried with the textual information determined within 10.6.2.1 Definition of context-sensitive information, whereby each query delivers the foundation for contextual information; the delivered information can only form a foundation, as most of the files will not contain only information useful for correctly interpreting knowledge. Instead, the information within the search results needs to be clustered, resulting in preliminary contextual information. Clustering can be done by done by an algorithm such as the Term Frequency / Inverse Document Frequency (TF/IDF),[392] which is used in information retrieval for assessing the relevance of terms in documents from a document collection. With the so-calculated weighting of a word regarding the document in which it is contained, documents can be arranged as a word-based search results in better search results than would be possible alone, e.g., the term frequency. A term's frequency of occurrence is determined by

$$tf(t,d) = \frac{f(t,d)}{max\{f(w,d):w \in d\}} \qquad (10\text{-}154)$$

where t represents the term occurring in document d. The equation contains a normalization, thus a division by the maximum number of occurrences of a term w in d, which is applied for avoiding a falsification of the results in long documents. This simply means it is reasonable that multiple occurrences of a term do not equally contribute to its relevance. The inverse document frequency measures the general meaning of the term for the total amount of the actual documents, and it does not does not depend on the individual document, but the document corpus, which is the total amount of all documents in the retrieval scenario.

---

389 Freeman Richard et al. (2002): Self-Organising Maps for Tree View Based Hierarchical Document Clustering; Honolulu: Proceedings of the IEEE IJCNN'02; vol. 2, pp. 1906-1911

390 Salton G. (1988): Automatic text processing: the transformation, analysis, and retrieval of information by Computer; Addison-Wesley: Massachusetts

391 Porter M. F (1980): An algorithm for suffix stripping: Program, 14 no3, pp 130-137

392 Salton G., McGill M. J. (1983): Introduction to modern information retrieval; New York: McGraw-Hill

$$w_{i,j} = tf_{i,j} * idf_i \quad (10\text{-}155)$$

where

$$idf_i = log\frac{N}{n_i} \quad (10\text{-}156)$$

idf_i represents the inverse document frequency, and

$$N = |D| \quad (10\text{-}157)$$

where D is the overall amount of documents within the retrieval scenario, n_i the number of documents containing the term i, and $w_{i,j}$ the weight of a term i in document j. Once the document vectors have been determined, clustering can be applied, whereby we will focus on hierarchical clustering (such as done within our brain) by self-organizing feature maps. The document vector itself is determined by the beforehand mentioned VSM, which is an algebraic model that may be used for representing text documents as vectors of identifiers, such as index terms. VSM represents documents and queries as vectors:

$$d_i = w_{1,i}, w_{2,i}, \dots, w_{n,i} \quad (10\text{-}158)$$

$$q = w_{1,q}, w_{2,q}, \dots, w_{n,q} \quad (10\text{-}159)$$

where d_i is the i^{th} document, q a query and $w_{n,i}$ represents the weights of the n^{th} term within d_i. Terms, however, are not necessarily single words, but may also be phrases.

10.6.2.4 Hierarchical learning

The hierarchical representation of information is a method that is strongly applied by our brains, so it has proved to be a valuable approach. Hierarchical clustering via SOFMs works in the same way as non-hierarchical clustering would do, with the difference that the underlying document structure is segmented into different levels, which are then processed subsequently (with still one SOFM). Furthermore, it is required to apply an algorithm for the growth of the SOFM, as the optimal structure cannot be determined in advance, preferably one that allows the ANN to take arbitrary structures. Algorithm capable of this is are the growing-SOFM algorithms,[393,394] first growing the structure of the SOFM to the optimal size, and finally fine-tuning it. An advanced algorithm suitable for our purposes is as follows, whereby the number of starting nodes may vary:

Start

1. Initialize weight vectors of starting nodes randomly
2. Calculate growth threshold T_g for the given data set according to definition
3. Present input to network
4. ***Repeat***

393 Bauer H. U., Villmann T. (1995): Growing a Hypercubical Output Space in a Self-Organizing Feature Map; ICSI Tech Rep. TR-95-030

394 Alahakoon D. et al. (2000): Dynamic self organizing maps with controlled growth for knowledge discovery; IEEE Transactions on Neural Networks; vol. 11, pp. 601--614

For each
input value
i. Determine the winning neuron n_w
ii. Apply weight vector adaptation to neighborhood of winner n_w and winner itself only by

$$w_{j(t)} = \begin{cases} w_{j(t-1)}, & if\ j \notin N_t \\ w_{j(t-1)} + \alpha_{(t-1)}\left(x_t - w_{j(t-1)}\right), & if\ j \in N_t \end{cases}$$

iii. Increase winner's error value
If
$TE_i < T_g$
If
i is boundary node
a. grow node
b. Initialize new node weight vectors to match neighboring node weight
Else
distribute weight to neighbors
iv. Initialize learning rate to its starting value
5. Until node growth is reduced to a minimum level
6. Reduce learning rate and fix a small starting neighborhood
7. Find winner and adapt the weights of winner and neighbors in the same way as in growing phase.
End

Algorithm 32 – Growing SOFM

Breakdown:

T_g: growth threshold
n_w: weight vector closest to input vector mapped to current SOFM
TE: total error of node i
α: learning rate

As the approach should be as dynamic as possible, it should take into consideration both the growth between already existing neurons and boundary neurons, on the contrary to commonly applied SOFM growth algorithms. The growing grid algorithm [395] extends an existing structure by inserting new neurons as layers (rows or columns) within existing nodes, whereby as the basic growing SOFM algorithm inserts new neurons at the boundaries only; however, in the former approach the weights are determined by interpolation, as well as in the latter for the insertion between existing neurons. The weights for inserted neurons at the boundaries are then determined by extrapolation. As both hierarchical and growing features need to be combined, an approach relying on independently spun and dynamically growing one-dimensional SOFM, [396] applying the Growing SOFM algorithm for growth seems to be

395 Fritzke B. (1995): Growing Grid: A self-organizing network with constant neighborhood range and adaptation strength; Neural Processing Letters, 2(5)
396 Freeman Richard et al. (2002): Self-Organising Maps for Tree View Based Hierarchical Document Clustering; Honolulu: Proceedings of the IEEE IJCNN'02; vol. 2, pp. 1906-1911

the most useful for our purposes. The use of independent SOFMs has the advantage that hierarchies can be modelled perfectly well. For correctly parsing the clusters, thus finding the correct branch of knowledge within the context-related search so that not the whole knowledge must be parsed, it is required to label these in a suitable manner, e.g. by LabelSOM, [397] which is capable of determining the features of the input space that are most relevant for the mapping of an input vector onto a specific unit. Basically, this happens by determining the contribution of every element in the vector towards the overall Euclidean distance between an input vector and the winners' weight vector, which forms the basis of the SOM training process.[398] At first, the quantization error (QE) for each term within a cluster has to be determined (the beforehand mentioned Euclidean distance between word vectors and weights). The QE is then used for performing a ranking, whereby the first n words represent the label of the cluster or hierarchy node. A neuron's weight as well as the document similarity may be determined by

$$d(a,b) = \sum_i |a_i - b_i| \qquad \textit{(10-160)}$$

which is the Manhattan-metric, where $d(a,b)$ is the distance between two points a and b, and i is the current position in the two-dimensional coordinate system. The number of starting nodes for two-dimensional SOFMs within this structure adaptation is usually four, which may remain the same for one-dimensional maps, with the difference of one-dimensionality. The minimum number would be two. After the top-level clusters have been detected, clustering with one-dimensional SOFMs below these clusters can be done, so that a hierarchical structure of SOFMs would be the result.

10.6.2.5 Interpreting the context

Now that the artificial entity commands over context-sensitive knowledge this knowledge can be applied on incoming interpretation requests. What a hypothetical artificial entity would be capable of so far is

- Abstraction
  The reduction of information to relevant parts
- Hierarchical knowledge structure
  The creation of a hierarchical knowledge structure based / creation of preliminary contexts on knowledge provided to the entity

This is quite a lot, but before being able to show intelligent behavior of any kind, be it extraverted by speech or deeds, it must be able to connect the abstracted information with its knowledge. For this, the whole knowledge base is queried with the context-sensitive information determined from the input (10.6.2.1 Definition of context-sensitive information). The result is then one to many different contexts, based on the hierarchical learning the artificial entity has conducted. For each context, the number of appearances of the words is compared and based on this, a ranking can be created. Furthermore, a lexical database such as

397 Rauber Andreas (1999): LabelSOM: On the labelling of selforganizing maps; Washington: Proceedings International Joint Conference on Neural Networks

398 Rauber Andreas (1999): LabelSOM: On the Labeling of Self-Organizing Maps [2013-11-01]; URL: http://www.ifs.tuwien.ac.at/ifs/research/pub_html/rau_ijcnn99/ijcnn99.html

WordNet[399] should be applied, so synonyms are included in the search, as well as the lexical meaning of words.

Start

1. Receive pre-processed input texts
2. ***For each***
   Term
   a) Query database
      i. Determine full meaning by inclusion of lexical databases
      ii. Apply algorithm from 10.6.2.1 Definition of context-sensitive information on lexical meanings
      iii. Query database with pre-processed input texts, lexical meanings and synonyms
      iv. Determine contexts
   b) Summarize similar contexts
   c) Determine
      i. number of actual text appearances
      ii. number of actual document references (if the documents have been linked)
3. Rank contexts based on number of text appearances and number of document references
   Calculate context weighting[400]
      i. Determine difference between each value of the number of references and its nearest lower value neighbor d_a
      ii. Determine difference between each value of the number of appearances and its nearest lower value neighbor d_r

$$W_c = \sqrt{\left(\frac{2d_a r_m}{3a_m}\right)^2 + {d_r}^2}$$

4. Determine maximum weight value
5. Determine all contexts appearing before the maximum weight value

End

Algorithm 33 – Context interpretation

Breakdown:

r_m: maximum number of references
a_m: maximum number of appearances
d_a: appearance difference to nearest lower neighbor
d_r: reference difference to nearest lower neighbor

399 WordNet (2013): WordNet a lexical database for English [2013-11-03]; URL: http://wordnet.princeton.edu/

400 Segev Aviv et al. (2006): Context recognition using internet as a knowledge base; Journal of Intelligent Information Systems (2007) 29:305–327

It is not necessarily required to perform a ranking according to the weight equation; however, results seem to be better than when just performing a ranking based on the number of appearances only.

10.6.2.6 Hidden Markov models and conceptual hierarchies in the neocortex

Apart from artificial neural networks, there exist lots of other classifying, predicting, auto-associating and clustering (network) structures. One that we need to discuss here is a special model developed by the Russian mathematician Andrei Andrejewitsch Markow, the Markov chain, or more precisely, the hierarchically hidden Markov model, derived from the hidden Markov model, which in turn has been derived from simpler Markov models such as the Markov chain. Not all processes always run into a fixed deterministic order and are therefore relatively easy to describe. Usually a system follows basic rules though, but it may not behave deterministically – I consider our brain to be such a system. Also, there are processes that cannot be observed directly but from which we can detect and evaluate signals. These must then be sufficient to draw conclusions on the true events. For a description of such hidden running, nondeterministic processes to statistical modelling, such as the hidden Markov models, may be used. Hidden Markov models are now with various issues, such as speech and handwriting recognition, in Biology for classification of protein sequences, as well as for modelling of economic processes.

Hidden Markov models (HMM) indicate a two-stage stochastic process. The first stage corresponds to a Markov chain whose states are not visible from outside (hidden). A second stage generates a stochastic process with so-called observations, which are the output symbols that can be observed at any time according to a probability distribution. The aim is to conclude from the sequence of output symbols upon the sequence of the non-visible states. HMMs are used in the recognition of patterns in sequential data, e.g. stored speech sequences in the context of speech recognition or price behavior in the stock market. The hidden states of the Markov chain correspond to semantic units that are that have to be detected in sequential data, which are semantic models. A hidden Markov model describes a two-stage stochastic process. The first stage forms a discrete stochastic process that can be described as a sequence of random variables, which can take values from a discrete, finite set of states (thus states).

$$Q = \{S_1, S_2, \dots, S_L\} \tag{10-161}$$

The process thus describes probabilistic state transitions in a discrete, finite state space.

$$S = \{q_t | t = 1,2, \dots, N\} \tag{10-162}$$

where $q_t \in Q$ in the sequence $q_1, \dots, q_{t-1}, q_t$.

The stochastic process S is

- stationary, that is, independent of the (absolute) time t,
- causal, that is, the probability distribution of the random variable S_t only depends on the past states and possibly, and it is
- simple, that is, the distribution of S_t only depends from the immediate predecessor's state, which then corresponds to a HMM of first order:

$$P(q_t|q_1, q_2, \dots, q_{t-1}) = P(q_t|q_{t-1}) \quad (10\text{-}163)$$

The stochastic process S can be regarded as a finite state machine with state set $\{1,2,N\}$. State transitions occur according to the state transition probabilities:

$$a_{ij} = P(q_t = S_j|q_{t-1} = S_i) \quad (10\text{-}164)$$

The state transition probabilities can be summarized in the state probability matrix

$$A = \left[a_{ij}\right]_{LxL} \quad (10\text{-}165)$$

where

$$(\forall i, j)\left(a_{ij} \geq 0\right) \quad (10\text{-}166)$$

and

$$(\forall i)\left(\sum_{j=1}^{L} a_{ij} = 1\right) \quad (10\text{-}167)$$

If the Markov-chain does not require starting in a defined state S_i then additional probabilities are required:

$$\pi = (\pi_1, \dots, \pi_L) \quad (10\text{-}168)$$

where

$$\pi_i = P(q_1 = S_i) \quad (10\text{-}169)$$

and

$$\sum_{j=1}^{L} \pi_i = 1 \quad (10\text{-}170)$$

The stochastic behavior or a homogeneous Markov-chain is fully defined by the parameters (π, A). A very often used illustration of a Markov-chain is a simple form of weather prediction, where three states

$$Q = \{S_1, S_2, S_3\} \quad (10\text{-}171)$$

are given, where S_1 represents rain, S_2 clouds and S_3 sunshine. $L = 3$ as well, as the unit of time is a day.

$$A = \left[a_{ij}\right]_{LxL} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \quad (10\text{-}172)$$

The model allows the answering of different questions, such as the question after the probability that after a sunny day the weather may be sunny, sunny, rainy, rainy, sunny, cloudy, and sunny in the following seven days. The Markov-chain in this situation is defined as

$$\begin{aligned} &P(S_3S_3S_1S_1S_3S_2S_3|S_3) = \\ &P(S_3|S_3)P(S_3|S_3)P(S_1|S_3)P(S_1|S_1)P(S_3|S_1)P(S_2|S_3)P(S_3|S_2) = \\ &a_{33}a_{33}a_{31}a_{11}a_{13}a_{32}a_{23} = 1.539 * 10^{-4} \end{aligned} \quad (10\text{-}173)$$

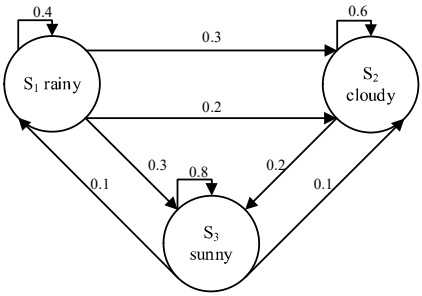

Figure 93 - Markov chain weather prediction

Another application are the interpretation of speech signals of letters, where the states of are not known, but the signals or derived attributes.

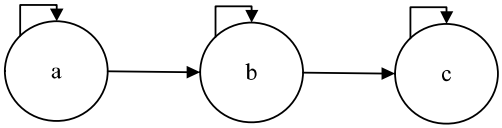

Figure 94 - Markov chain interpretation of speech signals

Hidden Markov models extend what has been discussed above in the sense that at each point in time t not only an unknown state is taken, but additionally a symbol from a finite alphabet.

$$V = \{v_1, \dots, v_K\} \tag{10-174}$$

The observer can only observe the generated sequence of symbols

$$O = o_1, \dots, o_T \tag{10-175}$$

where

$$o_i \in V \tag{10-176}$$

The states that have led to the output of the sequence of symbols O are unknown, or hidden, thus such a model is called hidden Markov model. The output of the symbols O_t is arbitrary and only depends from the state q_t, never from formerly taken states and output symbols:

$$P(o_t|o_1 \dots o_{t-1}, q_1 \dots q_{t-1} q_t) = P(o_t|q_t) \tag{10-177}$$

In each state S_j each Symbol v_k may be put out, with the probabilities

$$B = \left[b_{jk}\right]_{LxK} \tag{10-178}$$

where

$$b_{jk} = P(v_k|S_j) \quad \textit{(10-179)}$$

Furthermore,

$$(\forall i,k)(b_{jk} \geq 0) \quad \textit{(10-180)}$$

and

$$(\forall j)\left(\sum_{k=1}^{K} b_{jk} = 1\right) \quad \textit{(10-181)}$$

The hidden Markov model is fully defined by the parameters (π, A, B). Figure 95 - Markov chain and Figure 96 - Hidden Markov model graphically describe the differences between the two approaches - ω_i describe the states.

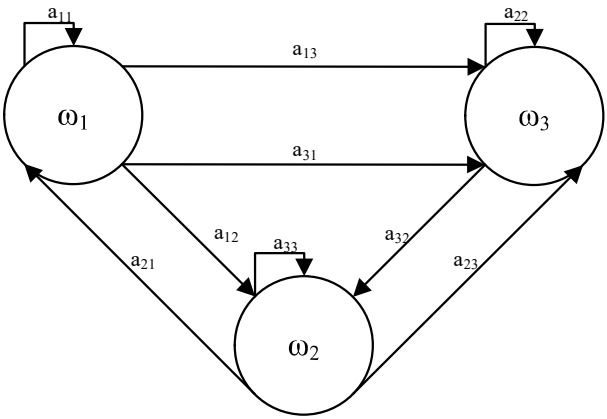

Figure 95 - Markov chain

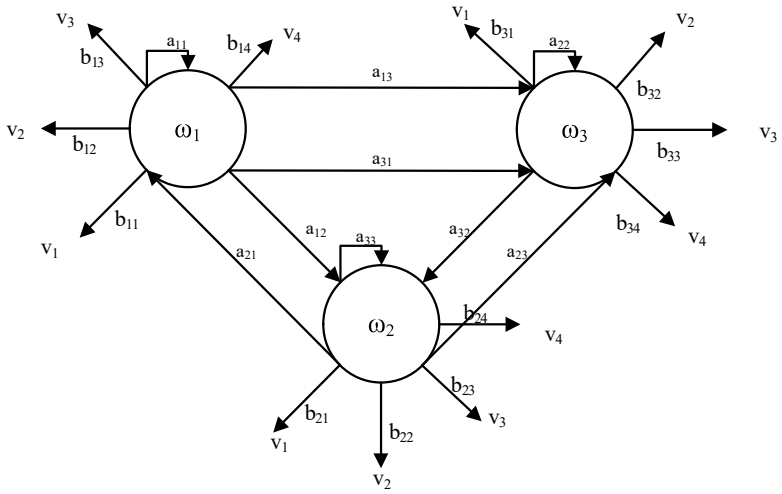

Figure 96 - Hidden Markov model

The algorithm for the generation of the sequence of symbols $O = o_1, \dots, o_T$ is as follows:

Start

1. Set $t = 1$ and select an initial state
$$q_1 = S_i$$
under consideration of π.
2. ***Repeat***
   a) Select an observation symbol
   $$o_t = v_k$$
   under consideration of
   $$P(v_k|q_t)$$
   from the matrix B.
   b) ***If*** $\boldsymbol{t < T}$
      pass over to the state
      $$q_{t+1} = S_j$$
      under consideration of the matrix A.
      else
      End process.
   c) Set $t = t + 1$
3. ***Until T has been reached.***

End

Algorithm 34 – Creation of the sequence of symbols

Breakdown:

t: Point in time
π: Starting probabilities $(\pi_1, \ldots, \pi_L)$
B: Symbol output probability matrix $\left[b_{jk}\right]_{LxK}$
: State probability matrix $\left[a_{ij}\right]_{LxL}$
S_j: Current state
o_t: Observation at t
v_k: Symbol taken for O_t
q_t: State at t

As a simple example of a hidden Markov model two weather observations may serve:

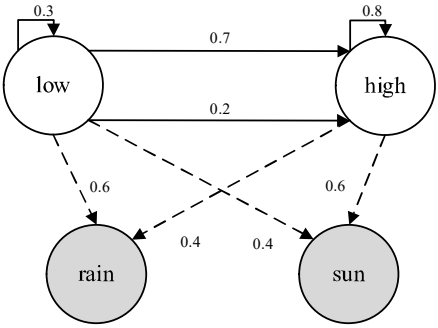

Figure 97 - Hidden Markov model weather observations

In the scenario described in Figure 97 - Hidden Markov model weather observations it is possible to observe rainy and sunny weather. The two hidden states are high and low air pressure. The transition probabilities are

$$P('low'|'low') = 0.3 \quad (10\text{-}182)$$

$$P('high'|'low') = 0.7 \quad (10\text{-}183)$$

$$P('low'|'high') = 0.2 \quad (10\text{-}184)$$

$$P('high'|'high') = 0.8 \quad (10\text{-}185)$$

and the output probabilities are

$$P('rain'|'low') = 0.6 \quad (10\text{-}186)$$

$$P('sun'|'low') = 0.4 \quad (10\text{-}187)$$

$$P('rain'|'high') = 0.4 \quad (10\text{-}188)$$

$$P('sun'|'high') = 0.3 \quad (10\text{-}189)$$

All possible sequences of the hidden states are described as

$$P(\{'sun','rain'\} = \\ P(\{'sun','rain'\},\{'low','low' + P(\{'sun','rain'\},\{'low','high' + \\ P(\{'sun','rain'\},\{'high','low'\} + P(\{'sun','rain'\},\{'high','high'\}) \quad (10\text{-}190)$$

and considering just the first term ('low') the result would be

$$P(\{'sun','rain'\},\{'low','low' = P(\{'sun','rain'\}|\{'low','low'\} * P(\{'low','low'\} = \\ P('sun'|'low') * P('rain'|'low') * P('low') * P('low'|'low') = 0.4 * 0.6 * 0.4 * 0.3 = \\ 0.01152 \quad (10\text{-}191)$$

According to their capabilities, hidden Markov models may be applied for three types of problem statements, which are

- Decoding
  A model M and an observed sequence of S is given. Within the decoding problem it is determined what the most likely path through M that generates S is, which may happen according to the Viterbi algorithm.
- Classification
  A model M and an observed sequence S is given. Within the classification problem it is determined how large the total probability $P_M(S)$ of M is, so that S is emitted, which may happen according to the forward algorithm.
- Training
  A set of training sequences and a model structure is given. Within the training problem it is determined what the best model parameters (state transition- and emission probabilities) that the training set allows are, which may be done according to the Baum-Welch algorithm.

Hidden Markov models are important in the task of engineering an artificial mind, as they allow abstraction, which I already mentioned is a major strength of our neocortex. Artificial neural networks do that as well, however, I do not consider it as the correct approach to solely rely on one paradigm. What is more, hierarchically hidden Markov models as auto associative pattern recognizers have proven to outperform ANNs in some domains, such as speech recognition. Furthermore, HHMMs work hierarchically, such as the pattern recognizers in our neocortex, activating only the higher hierarchy-pattern recognizers required for further processing incoming data. Thus, we will continue examining HHMMs.

HHMMs can be seen as hierarchical and recursive generalization of the hidden Markov models discussed beforehand. The difference to the former ones is that each of the hidden states becomes a probabilistic model as well, which simply means that each of the hidden states is a hierarchically hidden Markov model as well. This is important, as the structure of the neocortex is organized in that way as well – neural structures are organized in lists whose entries are lists again, or in other words, pattern recognizers of a lower conceptual hierarchy refer to pattern recognizers on a higher hierarchical level (see 10.6.3 Implementation). Hierarchically hidden Markov models thus do not emit single symbols, such as described beforehand, but sequences. HHMMs extend the already discussed HMMs in the sense that they are not restricted to emitting single observations. The states of HHMMs are HHMMs themselves, which again contain sub-states and which are capable of emitting strings of observations. States being able to emit strings of observations are called abstract states, on the contrary to states that have single emissions and are called production states. The internal states of

HHMMs may be called recursively, so that after the run has been completed control is returned to the abstract state. HHMMs can be represented by HMMs, where each HMM state consists of the production state and the abstract state higher up the hierarchy of the HHMM.[401] However, it should be mentioned that both the HMM and the HHMM belong to the same type of classifiers and may be used for solving the identical set of problem statements. Basically, the HHMM may be transformed into a standard HMM; however, the latter one utilizes its structure for solving a subset of those problem statements more efficiently.

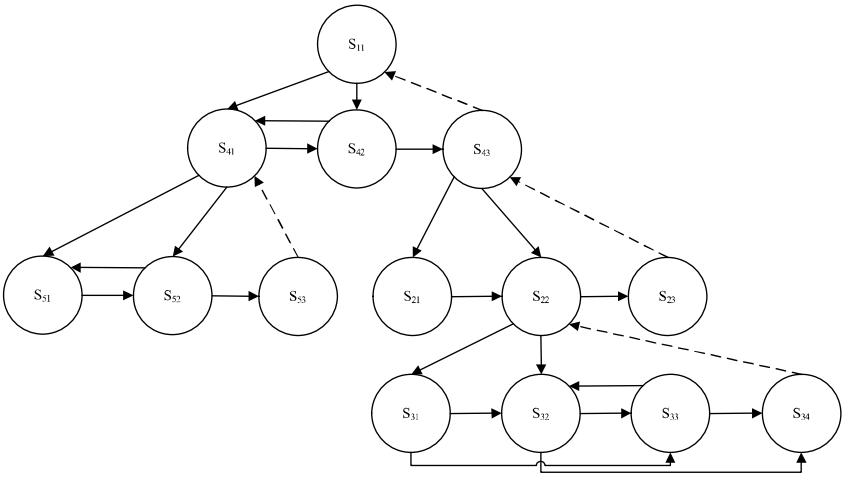

Figure 98 - Hierarchically hidden Markov model

The point when a state in an HHMM is activated it will activate its own particular probabilistic model. Thus, it will activate one of the states of the underlying HHMM, which then may activate its underlying HHMM and so forth. The procedure is rehashed until the production state is activated, which we learned are the states emitting observation symbols in the HMM sense. The point when the production state has emitted a symbol, control comes back to the state that activated the production state. The enactment of a state in a HHMM under an internal state is known as a vertical transition. After a vertical transition is finished, a horizontal transition occurs at a state inside the same level. The point when a horizontal transition results in a terminating state, control is come back to the state in the HHMM, higher up in the hierarchy that generated the last vertical transition. A vertical transition can bring about additional vertical transitions before arriving at a succession of production states and at last coming back to the top level. Consequently, the production states visited offers ascent to a succession of observation symbols that is processed by the state at the top level. The strategies

401 Heller Katherine A. et al. (2009): Infinite Hierarchical Hidden Markov Models; Proceedings of the 12th International Conference on Artificial Intelligence and Statistics; Florida: Clearwater Beach

for evaluating the HHMM parameters and model structure are more intricate than for the HMM.

10.6.3 Implementation

We already got to know that the neocortex is organized in a hierarchical structure. Additionally to the biological knowledge of our brain, we got to know all of the ingredients required for implementing an artificial neocortex within this elaboration, where the most important ones are

- different sorts of pattern recognizers, comprising
  - auto associative,
  - classifying, and
  - predictive ones
- learning algorithms for pattern recognizers
- self-organization and artificial self-organizing structures
- differently structured databases, and
- search engines.

Further ideas may be of importance as well, but are not absolutely required, such as quantum information theory in terms of quantum artificial neural networks. The neocortex consists of 6 layers, each of which is connected with the next-higher and/ or next-lower layer. Within these layers, it is assumed that hierarchical information processing occurs, which means that at first information from a lower conceptual layer is passed over to the next-higher conceptual layer in the sense of 'rough' pattern recognition up to detailed pattern recognition. This can be imagined very simply by the recognition of a face. A face consists of numerous features, but at first it is a face. Thus, the lowest conceptual layer is responsible for detecting basic structures, maybe rough shapes. There is not only one pattern recognizer active at once, but all. Figure 99 - Pattern presented to multiple pattern recognizers shows the presentation of a pattern to two pattern recognizers – in fact it would be hundreds of thousands or more. This is, because it is required to present the basic face-patterns to all pattern recognizers at once – this is one of the massive parallel processing tasks we often hear of in the context of the human brain.

After the input has been presented, some of the pattern recognizers will fire, if the pattern has been recognized. What happens if it is a new pattern, such as a face we have not known beforehand, will be discussed later. So, assuming that the pattern has already been learned, some of the pattern recognizers will fire and transmit a signal via their axon to the next higher hierarchy, but not to all pattern recognizers within this level, but rather to specific ones, namely the ones the pattern recognizers of the first level are connected to. The signal they transmit to the next-higher hierarchy tells the pattern recognizers in the higher hierarchy that they have to expect a pattern that they should try to recognize. The pattern is then passed over to all of the pattern recognizers of the next-higher hierarchy, which then do the same as the pattern recognizers on the next-lower level. The pattern is then passed over up to the highest conceptual level, where it is finally recognized. Figure 100 - Hierarchical pattern processing shows a simple auto associative ANN (left) passing over its result to another pattern recognizer one hierarchy above (right).

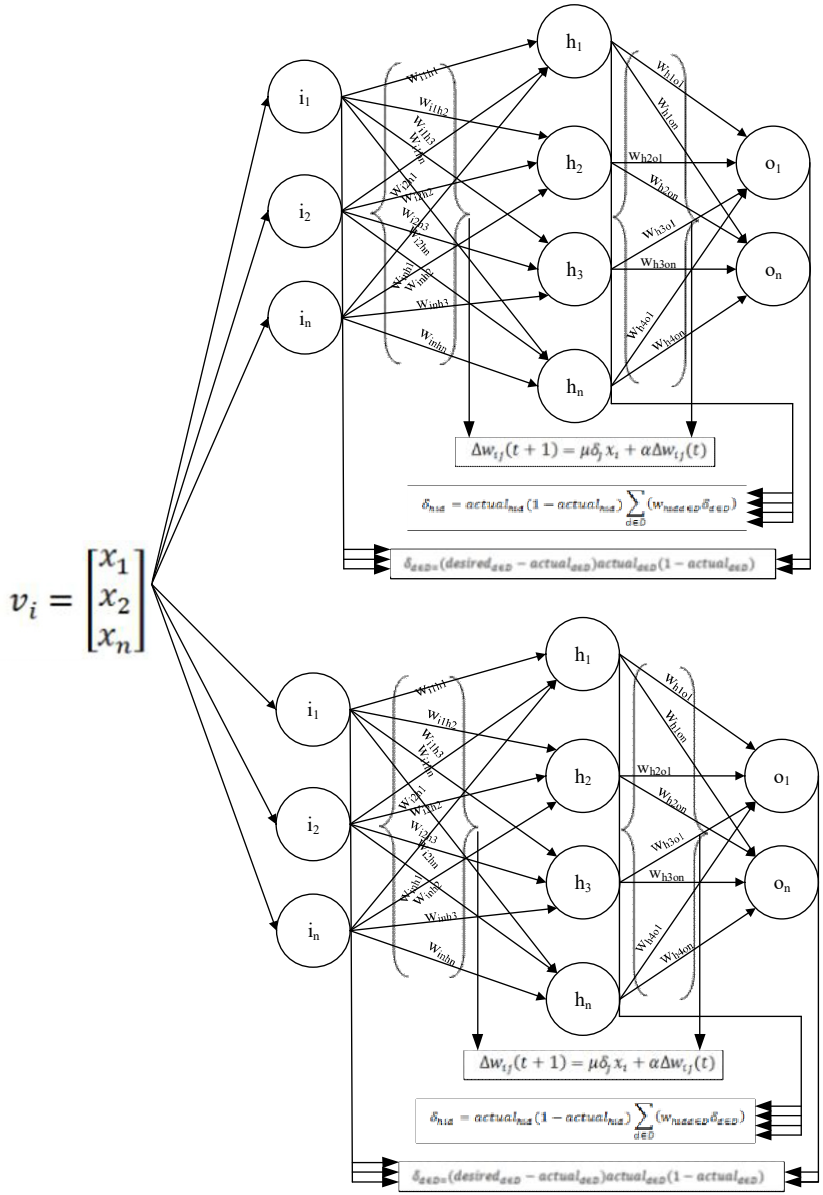

Figure 99 - Pattern presented to multiple pattern recognizers

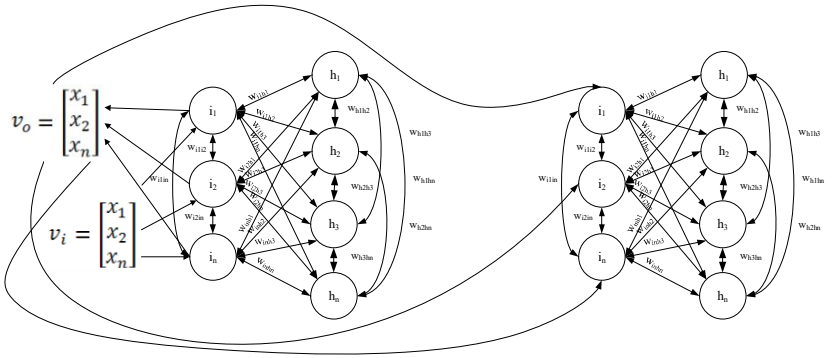

Figure 100 - Hierarchical pattern processing

There are thousands of pattern recognizers active at once, as we need to consider that not only all pattern recognizers on the lowest conceptual level need to be presented the incoming pattern, but also that each stored pattern (such as a specific face) is represented by numerous pattern recognizers, as a face must be recognized also when it is distorted, like when it is wearing sunglasses or a beard, or when it is viewed from the profile instead of from the front, or when the light is lighter, darker, … This is basically what our neocortex consists of (and what allows us to philosophize about the human mind or to read and think about this text):

– Specific pattern recognizers, and their
– redundant counterparts, as well as
– hierarchical structures of pattern recognizers.

However, hierarchies do not only work in one direction, it is also assumed that pattern recognizers on higher hierarchical levels inform pattern recognizers on lower conceptual levels that a pattern is likely to be recognized. Thus, we have to deal with communication in both ways, up the hierarchy when lower level-pattern recognizers recognize a pattern, and down the hierarchy when higher-level pattern recognizers assume that a pattern will likely be recognized. This happens, as higher-level pattern recognizers do receive input from numerous lower-level pattern recognizers, and when a cluster of lower level-pattern recognizers is not able to auto associate the pattern they are responsible for (such as a mouth that is not only distorted by e.g. a beard, but covered by a blue scarf), some others responsible for the same higher level-pattern (such as a face) may recognize the pattern they are responsible for (such as eyes or a nose). Triggered by these lower level-pattern recognizers, the respective higher level ones may have enough information so that the overall pattern (such as the face of a specific person) can be recognized and pass over the information to the lower-level cluster that has failed to recognize the pattern. The information that could not be identified (the mouth with the scarf) is then learned, thus stored into a new pattern recognizer for this specific face, which we can easily create within an artificial entity. It is also possible that the specific face is linked to the pattern of the blue scarf, which our entity may have already been stored, so that the face of the specific person is also recognized easily in the future if it is partially covered with a blue scarf. In our biological neocortex (and most likely also in an artificial one), it is not always granted that a distorted pattern is recognized correctly. Most of

us have made this experience when we fail to recognize a specific person we know very well or if we think that we have recognized a specific person that actually is someone that we do not know. From a computer science point of view, the hierarchies can be represented as multi-dimensional arrays of pattern recognizers, or lists pattern recognizers referring to other lists of pattern recognizers. We got to know two methods of implementing pattern recognizers, which are artificial neural networks and (hierarchically hidden) Markov models. Both have their qualities, thus both with find application in the proposed solution.

10.6.3.1 Acquisition of basic knowledge

First and foremost, it is required that the artificial entity is capable of learning. However, learning is a very loose concept, thus it has been split in four major parts, which overlap. Besides, the sequence of the steps discussed here is not predefined – some of them may be carried out in parallel, some of them in sequence. To begin with, it is required that the artificial entity features knowledge, but how can access to knowledge be given? There exist several different approaches, whereby I consider access to the internet as the first and foremost thing. Furthermore, different knowledge databases should be provided, which may be very different in structure, thus file-system oriented, relational or even OLAP-databases. The internet contains most of the knowledge of our species, and we are used to access it for learning new things. This is what an artificial conscious entity should also be capable of, however, access to the internet alone is not where acquisition of knowledge can start, as the ACE must be able to ask questions, as we do it. Thus, what is required at first is the capability of asking questions, or better, purposeful search for information. As an approach, I suggest to use a file system with information deposited into knowledge categories, such as discussed at 10.3 Self-organization and 10.6.2 Context recognition and hierarchical learning. This may be implemented by a file system oriented database as well as meta information referencing to the correct knowledge categories. This information is not stored hierarchically, thus in categories and subcategories. For the ACE being able to access and extend knowledge purposeful, it is required to structure this knowledge so that it is stored hierarchically, which can be done by a self-organizing clustering method, such as a self-organizing feature map, or even better by an approach incorporating adaptive resonance theory ANNs, such as fuzzy ARTMAP (see 10.3.2.3 Adaptive Resonance Theory). Approaches relying on ART do not suffer from the stability/plasticity-problem, which means that also the input data can be self-organizing or dynamic and change over time. Thus, anew clustering would not require complete re-training of the clustering approach, as continuous training allows the network to keep old knowledge as well as adapt to new. After the knowledge has been clustered, access may be given to limited knowledge databases, such as Wikipedia (I call it limited, because although Wikipedia contains millions of pages belonging to most of the knowledge categories we can differentiate from each other, the whole internet contains even more). Another step required is that the ACE is allowed to cross-reference the knowledge learned, which has been partially done by the permission for assigning knowledge to not only one, but many categories, such as an apple may be accessible through the category ‘plants’ as well as the category ‘food’. Furthermore, access to synonym databases such as WordNet is required, so that the ACE is not only able to access information, but also synonyms, which can be used to further interconnect information. Moreover, such databases provide access to verbs and adjectives belonging to specific nouns as well, so that even more information interconnection may be achieved. For the extension of existing knowledge categories different document similarity measurement methods may be applied, a simple one being TD/IDF already discussed at

10.6.2.2 Information Clustering. This implies that only a specific amount of knowledge for each category is stored, which I suggest should happen both in distributed file system databases, as well as via meta information, such as links to websites containing more knowledge. I suggest this shared approach for the following reasons:

- For the ACE it is not useful to store and organize the whole internet as a distributed file system database, although technically viable. The internet as knowledge entity is dynamic and subjected to continuous changes, thus it would require enormous resources for continuously querying, storing and clustering newly available information.
- It is not required for the ACE to have all existing knowledge to its avail in its brain (which the distributed file system database is a part of), as it is possible to store meta information, such as links, additionally to specific knowledge on a high conceptual level, as well as to let the ACE query the internet as we do it when we do not know something.
- As a next step, it is required to encode the conceptual and hierarchically structured knowledge into patterns.

Additionally to the textual information, pictures to specific kinds of knowledge should be stored and linked to the respective textual concept, such as the pictures of apples to the textual concept of an apple. Thus, the ACE now not only has textual, but also visual information to its avail.

10.6.3.2 Encoding the acquired knowledge into pattern recognizers

We humans do not store knowledge in the form of texts within our brains, but within patterns. One may argue that if the ACE accesses its knowledge by a query of its databases, and puts out an answer to a question that has been based on a statistical weighting, then this does not go along with understanding. However, this is the same what happens within our brain when we access knowledge. Beforehand I mentioned that it has been estimated that within our neocortex there exist around $3 * 10^8$ pattern recognizers, which are organized hierarchically. This means that some pattern recognizers on the lowest conceptual level are able to recognize, say, the crossbar of the letter A. On higher conceptual levels the whole letter is recognized. If the brain is dealing with a word, such as ‘Apple’, then the same happens with all the letters within the word, so all of the letters are recognized and signals from the pattern recognizers of the lower levels converge on a higher conceptual level – the word ‘Apple’. However, this is not quite right, as we are dealing with auto associative neural networks, which are able to complete the word if some parts are missing, if the pattern as a whole is disturbed in a not overly great manner, or even during the pattern is being processed. Let assume, the first four letters of the word have been read and processed, then the ‘Apple’-pattern recognizers are already in standby position and may also communicate down the hierarchy to the ‘e’-recognizers telling it that it as a whole may be required to fire soon. What happens then is that the firing thresholds of the ‘e’-recognizers are lowered so that they can fire more easily. However, I have been writing about pattern recognition and the querying of clustered and hierarchically stored textual information. Again, I consider a shared approach as useful, which means that knowledge within a database for querying textual information is absolutely useful, but it is also required to translate visual information into queries, and this requires auto associative pattern recognizers (see also 3.3.2.9 Fully connected artificial neural network). Such auto associative pattern recognizers, often thermal artificial neural networks, are presented specific patterns, such as the pattern of an apple, and then they are able to recall it, even if it is slightly modified or disturbed. For a simple concept such as an apple (and all

other concepts as well), it is required to create hundreds of pattern recognizers with different inputs, and to structure the pattern recognizers hierarchically. The human brain is not organized highly redundant, as simply not enough neurons are available for achieving this. Thus, the input pattern must be split up in simpler patterns at first, or better, in different input streams directing their information directly to all pattern recognizers on the lowest conceptual level. One of the input streams may present the curves of the apple's round shape, one its color, one its muster … The human visual cortex processes 12 such input streams, and what happens is the same. The lowest conceptual pattern recognizers (all of them) are addressed, and some of them fire. Thus, if one of the recognizers responsible for curved shapes, one for the color red, one for specific muster of an apple, and some more fire, they fire up the hierarchy to all of the pattern recognizers they are connected to. All other lower-hierarchy pattern recognizers do nothing, as no pattern has been recognizes (as edges, e.g.). What happens is that in the best case only one of the apple-pattern recognizers has received inputs from most of the lower-hierarchy pattern recognizers that fired. Certainly, this does not only happen via two hierarchies – the brain e.g. uses six, based on the fact that the neocortex is six-fold, whereby this means that there exist six layers of abstraction, each of which is organized hierarchically.

Possible means for implementing a hierarchical artificial neural structure are so-called deep learning architectures, such as Hinton's deep belief network. Basically, such complex networks consist of stacked ANNs, such as restricted Boltzmann machines, where the output of each layer (each Boltzmann machine) serves as input for the next layer. The problem is that such architecture only make use of some sort of hierarchical processing for preparing the initialization of weights, which is not exactly what is required here. What would be required is that each of these auto-associative layers is trained separately on specific patterns, resulting in a structure in which lower level pattern recognizers activate processing in higher level areas. However, there is another challenge: hierarchical processing within brains happens does not only pass over the input from one lower level pattern recognizer to higher hierarchies, but from many. So if we again imagine the image of the apple (which is not stored as an image in our brain, but as a hierarchy of patterns), some shape pattern recognizers, some color-recognizers and more do fire at once and propagate their signals via axons up to the higher hierarchical level. Again, on the next level, some pattern recognizers fire as they have been activated from the next-lower level by the patterns of the lower level (we remember that the ANNs used are auto-associative, so what they do is to reproduce a pattern when they recognize it). Processing in that way continues until the highest conceptual hierarchy has been reached and finally, the apple has been recognized. Thus, information from lower conceptual hierarchies is converged in higher conceptual hierarchies, so the chosen approach should combine ideas from decision trees and ANNs. A decision tree is a structure consisting of nodes organized in a tree structure, whereby there are two kinds of nodes:

- Internal nodes, which make local decisions based on the local information they have to their avail.
- Terminal nodes, which are used for making a final decision.

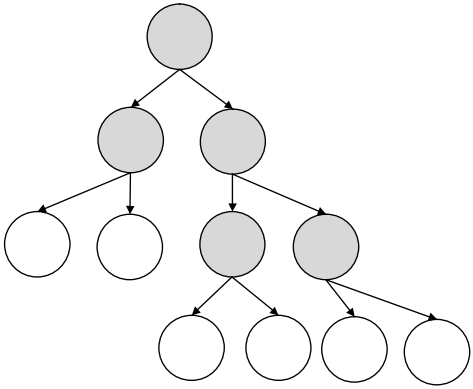

Figure 101 - Binary decision tree

Figure 101 - Binary decision tree shows a tree where each of the grey internal nodes results in two decisions (thus binary). The terminal nodes are represented by the white nodes. Each internal node makes its local decision based on a function $f(x)$, whereby a binary decision tree may simply decide according to

$$d_i = \begin{cases} left, & if\ f(x) < 0 \\ right, & if\ f(x) > 0 \end{cases} \qquad (10\text{-}192)$$

and usually

$$(x) = x_i - a_i \qquad (10\text{-}193)$$

where a_i represents the feature a on the i^{th} node that is used for coming to local decisions. The tree assigns a distribution of data to each terminal node, which is then used for coming to a final decision. If the decision tree is not binary, then it is required to build classes, so data belonging to the class c_1, is structured beneath the terminal node n_1. The weather outlook, e.g., can be structured in classes, such as 'rainy', 'sunny' or 'cloudy'. Thus, if all data assigned to a node belongs to the same class, the node is a terminal node, as no further decisions than the final decision can be made. If this is not the case and more than one class exists, a decision function selecting one of the nodes has to be defined. Although decision trees are easy to define and understand, they may become very complex for complex problem statements, and hierarchical pattern recognition in human brains for sure is. If it was the size only, this can be tackled by multivariate decision functions, such as the linear combination of the features. In case of hierarchical information processing, the decision tree must be created and processed bottom-up, and each of the nodes is represented by an auto-associative ANN (see Figure 102 - Bottom-up ANN tree):

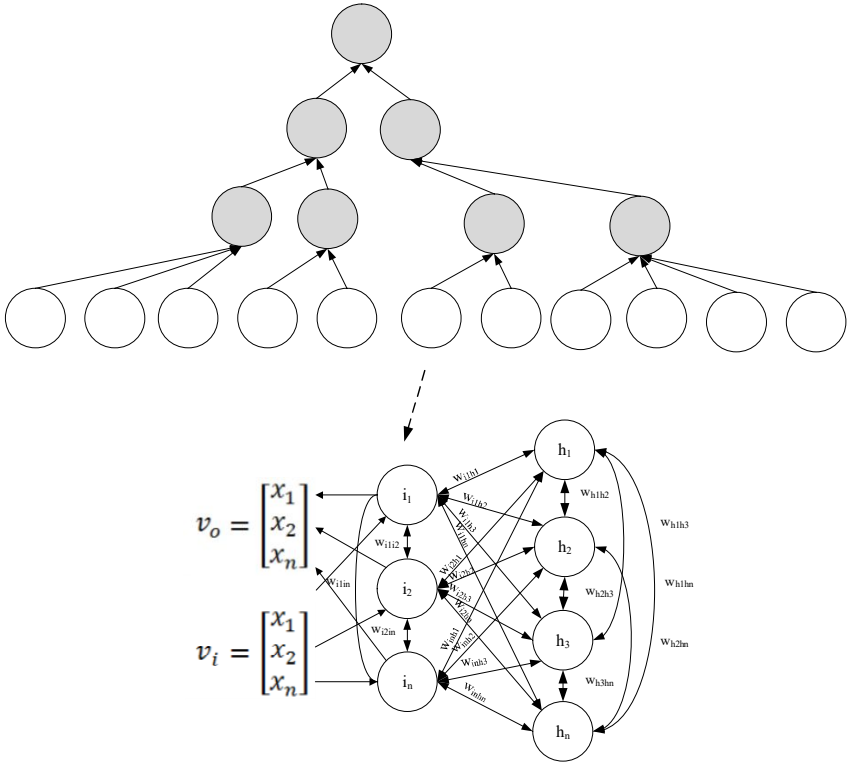

Figure 102 - Bottom-up ANN tree

The next difference is that we are not exactly dealing with a decision tree, but with a structure that is like a decision tree. Would the ANN tree be processed top-down, then the result would not be the activation of just one terminal node, but of many. Furthermore, due to redundant use of patterns, lower level pattern recognizers do not just propagate their patterns up the hierarchy in a decision tree-like manner, as each node may have more than one parent node (see Figure 103 - Cumulative activation).

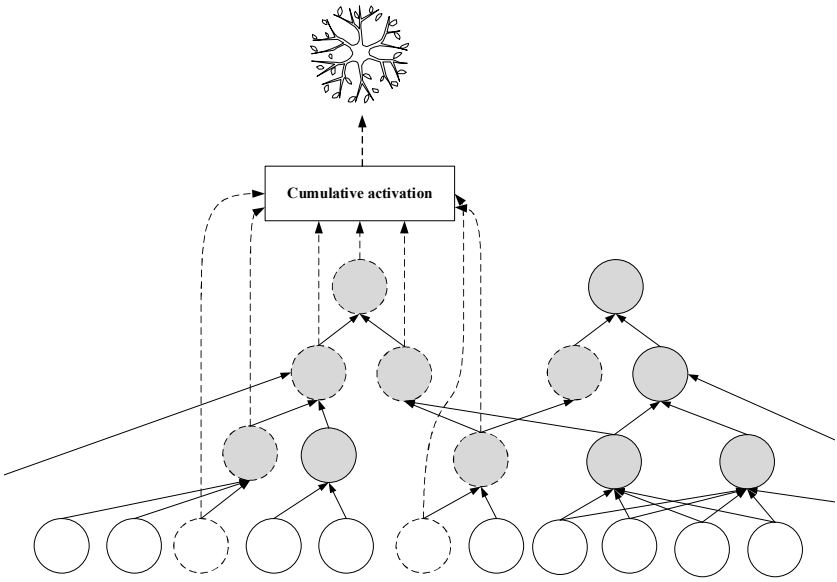

Figure 103 - Cumulative activation

The dashed lines show the activation of some lower level pattern recognizers, which in turn activate some higher level pattern recognizers and so on. In Figure 103 - Cumulative activation, the cumulative activation of all dashed pattern recognizers results in the recognition of (a specific sort of) tree. Furthermore, on the third hierarchy from below on the right side a pattern recognizer is activated by the firing of a pattern recognizer from below as well, however, in the example this does not result the activation of the highest level pattern recognizer (for another kind of tree) – cumulative activation of more nodes from the right path would be required. Such structures can be implemented easily, as well as pruning and adding of connections between the nodes. What has to be considered is the number of input neurons for each pattern recognizer, whereby the optimal number of input neurons does not necessarily need to be defined in advance for optimally recognizing a specific pattern – auto associative ANNs are even capable of recognizing their specific patterns when these are incomplete or disturbed.

10.6.3.3 Access to knowledge and how search engines are similar to the brain

Now that the knowledge has been structured and visual information has been encoded into patterns, these two components have to be set into relation. Although within a human brain everything is stored in patterns, even all knowledge of different domains a person may possess, I do not consider it required to do the same for an ACE. For the internal information processing of an ACE I consider it required to distinguish between

– knowledge about world and
– expert knowledge.

There exist many more subcategories and I am aware that especially the second category provides an ample scope, but in the beginning it is essential that the creators of the ACE distinguish between them. After the ACE has been equipped with basic knowledge about the world and some expert knowledge, it can learn and take over categorization itself.

The first category incorporates visual information about the environment the ACE is moving, which includes basic information about entities, objects and their purpose, as well as context-sensitive information, so far as possible. Getting back to the example of the picnic, visual information about entities and objects may include patterns of people, flora, fauna and food that the ACE is capable of processing visual information in a way that allows it to correctly recognize what it perceives. Contextual information is modeled in patterns as well, as the activation of the visually perceived objects does not only allow the identification of such, but also the recognition of a situation, such as the picnic. I do not consider it as a requirement that a context such as a picnic is recognized at first, as even we fail in some situations and the ACE is capable of learning.

The second category incorporates knowledge of the world as well, such as descriptions of objects on atomic layer or specific knowledge of information technology. Such expert-knowledge does not necessarily need to be encoded into patterns as within our brain, but may be stored in large and distributed database structures, comprising different database types such as distributed file system-, OLAP- and relational databases, whereby the first category may serve as storage for knowledge stored in pages or files such information provided by Wikipedia, and the latter ones e.g. for processing complex (knowledge-related) stored procedures, for storing the results of calculations, or even for the analysis of specific knowledge stored in a cube.

Databases may also serve as short-term extension to a short-term memory that has been modeled in patterns, but more on that later. The connection between knowledge and patterns is enabled by translating incoming visual information (patterns) into textual information. If one is of the opinion that the translation of visual information into text for accessing knowledge is nothing a biological conscious entity would do, I admit that she is correct; however, I do not see an obstacle in creating an ACE by changing the way of how information is accessed. Thus, the suggestion is to store simple and shot textual information additionally to each pattern (such as the word ‘apple’ for the pictures of apples), which then can be used to query the knowledge stored in databases when patterns are presented through visual channels.

In current search technologies we see many similarities to the human brain in terms of how information is retrieved and stored. Before a search engine can provide information about where a file or document resides, it has to find this information within numerous files or web sites on different servers. In order to get to achieve this ambitious target, agents extract significant words from these files or sites and store them in lists. In terms of the human brain it has already been discussed that neural connections are more likely to fire the more often they are used, which means that clusters of neurons over the neocortical hierarchy reacting to the recognition of a specific object will fire easier the more often this object has been recognized. Thus the search for whether an object is a ball or an apple in the human brain happens by providing the features of the object through the visual system and ranking the top-firing neuron clusters first. To put it in a suitable analogy: only the most popular search results, thus neuron clusters firing intensely, appear in the result-set.

In common web search engines, the agents (spiders) usually start indexing the words on frequently visited websites, as well as follow every link on each site – the agents traverse the whole complex graph of web sites, or in other words, literally travel all over the world. The spiders in the human brain are the transformed features propagated through the neocortical hierarchies. In search engines, work is parallelized, thus multiple spiders act at once and each of them maintains hundreds of parallel connections at once. This is similar to what happens in the human brain: search is parallelized as well, thus the search paths are not pursed in sequence, but in parallel. There are different approaches for how to implement spiders, i.e. indexing only significant words and leaving out stop-words (a, the, an, etc.) keeping track of where the words occur (title, subtitle, meta tags etc.). In 10.6.1 Knowledge and data we already saw how knowledge can be represented and that knowledge about knowledge is relevant as well. In terms of search technologies, there exists a similar concept, called meta tags. Meta tags allow the owner of a page to specify key words and concepts under which the page will be indexed. This can be helpful, especially in cases in which the words on the page might have double or triple meanings -- the meta tags can guide the search engine in choosing which of the several possible meanings for these words is correct. To protect against non-matching tags, spiders will correlate meta tags with page content, rejecting the meta tags that don't match the words on the page.

The more knowledge the ACE can access, the more difficult it will be to find relevant information and to accurately identify context. For this, exclusion-conditions can be implemented, i.e. that when several features for an apple have been identified, the result can never be a tree. This is also what is successfully applied in search technologies, as many times, a page's owner doesn't want it showing up on a major search engine, or doesn't want the activity of a spider accessing the page. In order to avoid this, the robot exclusion protocol was developed – a spider is told by the protocol in the meta-tag section of a site that it must not conduct its operations, i.e. traverse further along the links, or extract words.

As the internet is ever-changing, spiders continuously crawl the web about new or changed information, and the more data is collected the more important it is how this data is stored and maintained. Again, this is similar to what happens in the human brain when it comes to learning: learning is a continuous process, and the information must be encoded in a way that makes it useful. We already discussed encoding knowledge into pattern recognizers as well as several forms of knowledge representation, such as semantic networks. When it comes to search technologies, there are two key components involved in making the gathered data accessible to users:

– How indexing of the information is conducted, and
– which information is stored together with the data.

By just storing an identifier and a reference (key and value), such as a word and a URL, which gives the location where the word can be found, practical use would be severely limited. It would not be possible to figure out if the word was used often, it is important in terms of the contexts it has been used, or if the page found at the specified URL contains links to other pages, in which this word is used. Additionally, any search will not come up with only one result, thus a ranking of the results based on importance is required, which can only be done by storing more than just key-value pairs containing word and URL. A common approach is to weight the entries based on different criteria, i.e. how often the word occurs in a document, and where it occurs (heading, body, meta tag, etc.).

In terms of the human brain, independent from what scenario we are facing, i.e. object recognition or talking to another human – mostly object recognizers are involved (whereby I define object not only as physical object, but also as words, context, etc., so literally anything that can be recognized), and search refers to correctly executing the neuron clusters responsible for recognizing an object, but nevertheless it is not required to achieve the same target by the same means. An apple looks similar to a pear, but more neural clusters for apple fire, although some of them may also be firing (voting) for pear. Ranking in an ACE could be conducted in a way that fits its personality, thus more cautious actions may be proposed for an ACE whose task it is to negotiate.

In the human brain we also see a sort of compression in terms of reusable neural clusters, i.e. the cluster for recognizing diagonal lines may be used to recognize a part of the letter A as well as a part of a houses' roof. Of course, this is also something that needs to happen when storing data for search engines, as storage capacity does not increase in the same way available data increases. Finally, an index is created on the data, which helps to find relevant information a lot quicker than in the raw data. Numerous ways for creating an index exist, and one of the most common ones is hashing, where the output of a formula is attached to each word; basically what happens is that entries are distributed evenly across a predefined number of domains. The distribution of the words in a document is different to the overall distribution of words in a specific language, i.e. there exist more words beginning with the letter M than words beginning with other letters, resulting in longer search time for a word starting with M. Hashing simply compensates this difference so that all words are found in more or less the same time, and segregates the actual entry from the index. The hash-table can also be seen as a key-value store, as it contains a hashed number as well as a pointer to the actual data, and this in combination with advanced storage concepts result in quick responses also when complex search requests are submitted. If more complex queries are built, single words must be connected by the use of Boolean operators. In terms of the ACE, search must be fuzzy, as the input is not directly a word, but most often a stream of unstructured information. From this stream of information, pattern recognizers must extract the (hidden) features, which allow the ACE to understand a situation. Current research also focuses on translating unstructured in formation in more structured information, i.e. generating texts from images. This, in turn, allows leveraging the full capacity of knowledge representation and / or searching technologies. The Boolean operators usually used in search technology are AND, OR, NOT, as well as

- FOLLOWED BY: one of the terms must be directly followed by the other.
- NEAR: one of the terms must be within a specified number of words of the other.
- Quotation marks: the words between the quotation marks are treated as a phrase, and that phrase must be found within the document or file.[402]

Building more complex queries in ACE would of course include all of the above mentioned, but as brains are fuzzy, also fuzzy gates (operators) should be implemented, which can be happen by probabilistic networks or ANNs. Literal searches pose challenges in the sense that some words may have multiple meanings and are thus used in multiple contexts. Thus, context must be added (word sense disambiguation), which is done by concept-based search, founded on statistical analysis of word counts, word collocation, concordance, collocation,

402 Franklin Curt (2000): How Internet Search Engines Work; URL:http://computer.howstuffworks.com/internet/basics/search-engine.htm

collostructional analysis, keyword linguistics, pattern grammar, etc., which allows to find relevant search results. As a lot more data has to be stored for such an approach, performance problems resulted in another strain of research, where queries are based on natural language input; Boolean operators are then obsolete (see also 2.4 Language and communication). The challenge, which also closes the circle, is again how to represent language in order to be able to figure out meaningful results. Logic is one way, semantic networks another, and to consider language as the most simple representation of itself, one more.

10.6.3.4 Language processing and understanding

Clustered, interconnected knowledge provided via textual and visual information does not enable the ACE to communicate, so the ability to process natural language or another approach for communication is required as well. Language processing incorporates numerous different approaches and paradigms, such as probabilistic models of language, statistical natural language processing, information extraction, text mining, robust textual interference, statistical parsing, grammar induction, constraint-based theories of grammar, and computational lexicography. An approach that can be applied are hidden Markov models (see 10.6.2.6 Hidden Markov models and conceptual hierarchies in the neocortex). Such models have proven to perform well at processing speech or parts of speech. Part-of-speech tagging (POST) is the part of corpus linguistics that will serve for explanations here. Basically, POST considers the context as well as the definition of a particular part of speech for making up a word within a text, such as a document or speech, as belonging to a specific part of speech. Part of speech refers to a linguistic category of words, generally defined by the syntactic or morphological behavior of the lexical item in question. POST is more difficult than just considering words and their correct part of speech, as words may have more than one meaning. An example is the German word 'einen', which can both be an indefinite article and a verb. A commonly used example in English is 'The sailor dogs the hatch', where the word dogs alone may represent the plural of dog, what would be definitely wrong in that context, in which it is used as a verb. Both grammatical and semantic analysis may result in the desired tagging, where via the former one analysis would state that the plural noun is definitely wrong here, and the latter one that the verb would occur in the context of sailor and hatch more likely. Summing up, POST is the task of tagging each word in a text with its appropriate part of speech, such as

The[AT] representative[NN] put[VBD] chairs[NNS] on[IN] the[AT] table[NN].

or

The[AT] representative[JJ] put[NN] chairs[VBZ] on[IN] the[AT] table[NN].[403]

A well-known POST-corpus is the Brown-corpus,[404] from which the tags above came from. The first example shows that the word 'representative' has been tagged as a singular noun, whereby in the second example it has been tagged as an adjective. The words 'put' and 'chair' allow for the same. Thus, some sort of probability model has to be applied for correctly

403 Dror Gideon (2009): Part-of-Speech Tagging [2013-12-08]; URL: http://www2.mta.ac.il/~gideon/courses/nlp/slides/chap10_pos.pdf

404 Leeds University: Automatic Mapping Among Lexico-Grammatical Annotation Models [2013-12-08]; URL: http://www.comp.leeds.ac.uk/amalgam/tagsets/brown.html

identifying the context of a sentence, and one of them are HMMs. It has already been explained that processing of the current state within HMMs only depends from the last step. The states are represented by POST tags, whereby observations are sequences of words. The transition probability is represented by the bigram model for POST tags, the observation probability by the probability of generating each token from a given PIST tag. A bigram is every sequence of two adjacent elements in a string of tokens, which can be e.g. words or letters. The bigram frequency distribution within strings can be used for statistical analysis of text, such as within HMMs:

$$P(W_n|W_{n-1}) = \frac{P(W_{n-1},W_n)}{P(W_{n-1})} \qquad (10\text{-}194)$$

where the probability P of a token W_n with the preceding token W_{n-1} is given by the probability of their bigram, which is the co-occurrence of the two tokens $P(W_n|W_{n-1})$ divided by the probability of the preceding token. Again, hidden within the HMM means that the sequence of POS tags (states) that is responsible for generating the sequence of words (observations) is hidden. The Viterbi algorithm is used within HMMs for decoding (POST in the example here), which we remember is the discovery of a best hidden state sequence Q given an observation sequence O (and an HMM). Decoding is capable of detecting the most probable sequence of states that has produced the observed sequence:

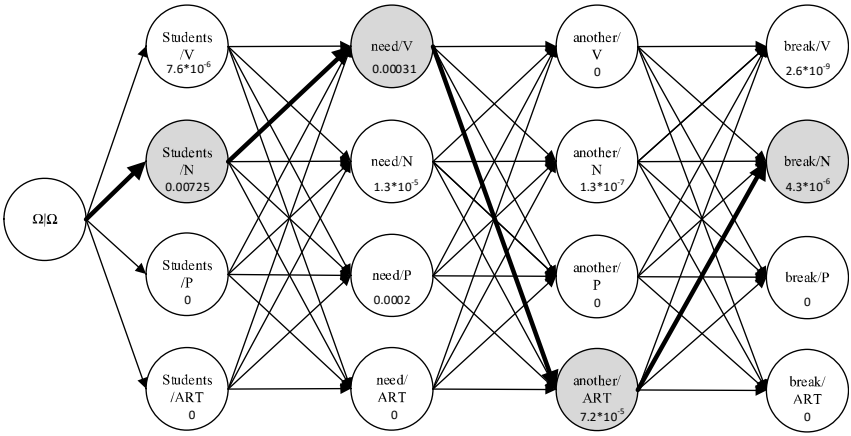

Figure 104 - Viterbi example

Figure 104 - Viterbi example shows how a HMM determines the most probable tags for words in a context. The Viterbi-algorithm therefore determines the most likely sequence of hidden states in a given HMM and an observable sequence of symbols. From the HMM description we know that the sequence of observable symbols is defined as

$$O = o_1, \dots, o_T \qquad (10\text{-}195)$$

where

$$o_i \in V^* \qquad (10\text{-}196)$$

Now the most likely sequence of states

$$Q^* = q_1^*, \dots, q_T^* \tag{10-197}$$

where

$$q_i \in S^T \tag{10-198}$$

shall be determined. S is the quantity of hidden states, and V the alphabet of observable symbols (emissions). Thus, the sequence of hidden states that maximizes the value of $P(Q|O;\lambda)$ amongst the sequences Q with the length T is the probability that the model λ ran through the states Q at output O.

$$P(Q|O;\lambda) = \frac{P(O;Q|\lambda)}{P(O|\lambda)} \tag{10-199}$$

$P(O|\lambda)$ does not depend from Q, thus

$$P(O;Q^*|\lambda) = maxP(O;Q|\lambda) \tag{10-200}$$

For the calculation, two further variables are required, where the first one is the maximal joint probability $\vartheta_t(i)$, which is stored at point in time $1 \leq t \leq T$ if the prefix $O_1, \dots, O_t$ has been observed when running through a sequence of states of length t and ends in the state $s_i \in S$:

$$\vartheta_t(i) = \max_{\substack{Q \in S^T \\ q_t = s_i}} P(o_1, \dots, o_t; q_1, \dots, q_t|\lambda) \tag{10-201}$$

The second variable is $\psi_t(i)$, which remembers for every point in time and state which preceding state has contributed to the maximum. The Viterbi-algorithm is then as follows:

Start

1. ***Initialization: For each***
   state s from 1 to N do
   $$\vartheta_1(i) = \pi_i b_i(o_1)$$
   $$\psi_t(i) = 0$$
2. ***Recursion: For each***
   time step t from 2 to T do
   For each
   state s from 1 to N do
   $$\vartheta_1(i) = b_i(o_t) \max_{1 \leq i \leq N} \left(a_{ij}\vartheta_{t-1}(j)\right)$$
   $$\psi_t(i) = \underset{1 \leq i \leq N}{argmax} \left(a_{ij}\vartheta_{t-1}(j)\right)$$
3. ***Termination: Terminate***
   $$P(O;Q^*|\lambda) = \max_{1 \leq i \leq N} \vartheta_T(j)$$
   $$q_T^* = \underset{1 \leq i \leq N}{argmax}\, \vartheta_T(j)$$
4. ***Return backtrace path***
   $$q_T^* = \psi_{t+1}(q_{t+1}^*)$$

End

Algorithm 35 – Viterbi algorithm

Breakdown:

s: state
N: number of states
T: length of state sequence
t: point in time
a_{ij}: transition probability from previous state q_i to current state q_j
$b_i(o_t)$: state observation likelihood of the observation symbol o_t given the current state j

$argmax$: determines the set of points of a given argument for which the given function attains its maximum value

max: the maximum value of an element

Having discussed the HMM decoding problem, which asks to discover the best hidden state sequence Q by a given observation sequence O and an HMM $\lambda = (A, B)$, we now know how λ can determine POS-tags for words. However, before an HMM can do that, it must be trained, similar to an ANN which also has to be trained. There exist three types of training for HMMs, which are supervised, unsupervised and semi-supervised. In supervised training all training sequences are tagged, meaning that e.g. a human expert has labelled the words with tags beforehand. The model thus would be trained to learn that 'Students' is a noun, 'need' is a verb, etc. whereas in unsupervised learning all training sets are unlabelled, thus the tags are unknown.

Supervised learning relies on the estimation of state transition probabilities on tag bigram and unigram statistics in the labelled data:

$$\alpha_{ij} = \frac{C(q_t=s_i, q_{t+1}=s_j)}{C(q_t=s_i)} \qquad \textit{(10-202)}$$

where α_{ij} is the probability of the state transition from state i to state j, q_{t-1} the state at point in time $t-1$ and q_t the state at point in time t. s_i is the i^{th} state that q_t can take and s_j the j^{th} state q_{t+1} can take (see Figure 105 - Variable explanation).

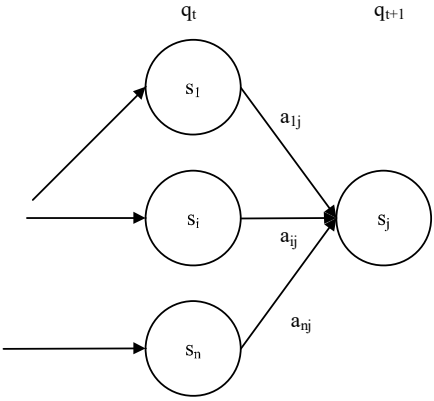

Figure 105 - Variable explanation

$$b_i(k) = \frac{C(q_i=s_i, o_i=v_k)}{C(q_{t-1}=s_j)} \quad (10\text{-}203)$$

Thus, supervised learning is a maximum likelihood estimation.

For unsupervised learning, it is required to discuss the remaining two HMM problems in detail, as it relies on the forward-backward algorithm (Baum-Welch algorithm). The forward-part can be explained by solving the first HMM problem, which is the determination of the likelihood $P(O|\lambda)$, given an observation sequence O and an HMM $\lambda = (A, B)$. The likelyhood of an observation itself depends from the Markov assumption that states that the probability of any state at time t only depends on the probability of each possible state at time $t-1$. First of all, considering the paths in Figure 105 - Variable explanation, the probabilities of all possible ways of getting to s_j at time $t+1$ by coming from all possible states s_i have to be determined and summed to get the overall probability of being in state s_j at $t+1$ while accounting for the first t observations. Finally, the result has to be multiplied with the probability of observing o_t in s_j:

Start

1. ***Initialization: For each***
   state s from 1 to N do
   $$\alpha_1(i) = P(o_1, q_1 = s_i|\lambda) = \pi_i b_i(o_1)$$
2. ***Recursion: For each***
   time step t from 2 to T do
   For each
   state s from 1 to N do
   $$\alpha_{t+1}(j) = P(o_1, o_1, \dots, o_{t+1}, q_{t+1} = s_j) = \sum_{i=1}^{N} P(o_1, o_1, \dots, o_{t+1}, q_t = s_i, q_{t+1} = s_j)$$
   $$= \sum_{i=1}^{N} P(o_1, o_1, \dots, o_{t+1}, q_t = s_i) a_{ij} b_j(o_{t+1}) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1})$$
3. ***Termination: Terminate***
   $$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$
4. ***Return***
   $$P(O|\lambda)$$

End

Algorithm 36 – Forward recursion

Breakdown:

s: state
N: number of states
T: length of state sequence
t: point in time
a_{ij}: transition probability from previous state q_i to current state q_j

$b_i(o_t)$: state observation likelihood of the observation symbol o_t given the current state j

For the last problem, which states that a sequence of observation is given, for which the transition- and output-probabilities of λ need to be determined that most likely result in the output sequence, also the backward recursion is required:

Start

1. ***Initialization: For each***
   state s from 1 to N do
   $$\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T | q_t = s_i, \lambda)$$
   $$\beta_T(i) = 1, 1 \leq i \leq N$$
2. ***Recursion: For each***
   time step t from 2 to T do
   For each
   state s from 1 to N do
   $$\alpha_{t+1}(j) = P(o_{t+1}, o_{t+2}, \dots, o_T | q_t = s_j) = \sum_{j=1}^{N} P(o_{t+1}, o_{t+2}, \dots, o_T, q_{t+1} = s_i, q_t = s_j)$$
   $$= \sum_{j=1}^{N} P(o_{t+2}, o_{t+3}, \dots, o_T | q_{t+1} = s_i) a_{ij} b_j(o_{t+1})$$
   $$= \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(o_{t+1})$$
3. ***Termination: Terminate***
   $$P(O|\lambda) = \sum_{j=1}^{N} \beta_1(i) \pi_i b_i(o_1)$$
4. ***Return***
   $$P(O|\lambda)$$

End

Algorithm 37 – Backward recursion

Breakdown:

s: state
N: number of states
T: length of state sequence
t: point in time
a_{ij}: transition probability from previous state q_i to current state q_j
$b_i(o_t)$: state observation likelihood of the observation symbol o_t given the current state j

The forward–backward algorithm is used for finding the most likely state for any point in time, on the contrary to the Viterbi-algorithm, which is used for finding the most likely sequence of states. Assuming an HMM with N states (or POS-tags, for language processing), its initial parameters are set randomly. Until the HMM converges, the forward-backward algorithm is used for determining the probability of various possible state sequences for

generating the training data (often described by E(xpectation)). In the Baum-Welch algorithm, these probabilities are then used for re-estimating the values for all of the parameters of the HMM (often described by M(aximization)). Each iteration modifies the HMM-parameters in a way that guarantees for the increase of the likelihood of the data ($P(O|\lambda)$). The algorithm can be stopped at any time before convergence for getting an approximated solution.

That leaves us at a point where POST of words becomes possible, which forms one of the basis features of natural language processing and in consequence, natural language understanding. These two fields comprise numerous subfields of research (see 2.4 Language and communication). As numerous books and other sorts of publications have been written on these topics, I will not go into further detail here. For me, it was important to describe a paradigm that an artificial entity can make use of for processing and even understanding language, such as the hidden Markov model, and how information processing within such a model works. Furthermore, in 10.6.2.6 Hidden Markov models and conceptual hierarchies in the neocortex similarities between the neocortex and HMMs have been discussed, so it is only fair to describe its basic functional principles here.

10.6.3.5 Quantum pattern recognizers

A large part within this elaboration has been dedicated to quantum mechanics and theoretically possible applications in machine learning. This has been done with a special purpose, namely the discussion of a possible processing approach for the numerous pattern recognizers of the ACE. Even if it may not be that case that quantum effects are required for producing conscious experiences in human brains, we may nevertheless make use them within ACEs, if advantages prevail. Breaking down quantum information processing to single pattern recognizers, it has already been discussed that the training performance can be significantly increased (see 10.5 Quantum physics and the artificial brain) by lifting all possible configurations of the ANN in quantum linear superposition at once and searching for the required configuration with Grover's search algorithm. However, it has been shown that with an ANN of arbitrary largeness the search for the solution on the performance register would still require exponential time with respect to the composite weight vector's size, which is

$$O\left(\sqrt{\frac{2^b}{T}}\right) \qquad \textit{(10-204)}$$

There are ways to deal with this situation, such as the application of randomized training algorithms, but there are further advantages that suggest the implementation of quantum neurocomputer, and this is the capability for solving so-called hard problems. The problem of combinatorial explosion could be solved by exploiting physical phenomena directly, such as analogue or quantum computers do. Initially, the idea was described by Richard Feinman[405] who was able to show that exponential complexity related to calculated probabilities can be reduced to a problem of polynomial complexity related to simulated probabilities, and a similar proceeding may be applied to NP-complete problems. A neurocomputer, which our brain or to some degree also an artificial neural network is (the latter one is not exactly a neurocomputer, as a classical ANN is not processed on neurons) belongs to another problem class, and compared to NP-hard problems, their complexity comprises not only an

405 Feynman Richard (1982): International Journal of Theoretical Physics, Vol. 21, No. 6/7

exponentially large number of simple computations, but an unknown analytical structure. Unfortunately, an analogue computer is not universal, slow and inaccurate. A possible way for tackling the problem may be the introduction of non-deterministic approaches to computations, which may be the not here discussed Monte Carlo method, as well as randomized algorithms. Both can be utilized for solving combinatorial problems, and the theory of computational complexity states that polynomial time nondeterministic algorithms, compared to polynomial time deterministic ones, are more efficient in the sense that there exist algorithms of the former category that can solve problems probably within polynomial time or even within polynomial time for sure. To put it briefly, this means that such algorithms exchange complexity with completeness or correctness. Nevertheless they provide a solution for the problem, would not there be the problem of number randomization – random number generators are slow and unreliable in the sense that no real random numbers are generated; instead an algorithm is used for it. Quantum physics allows the creation of an analogue computer with real randomization, which means with such a device polynomial time nondeterministic algorithms can be directly implemented and solved on the hardware, and it is universal. Back to quantum artificial neural networks and the brain, very often I mentioned auto associative neural networks, which are trained to recognize one or more patterns even if the presented pattern is distorted or incomplete. The associative memory problem can be described as the storage of q n-dimensional patterns as dynamical attractor:

$$\xi_i^{\eta}(\eta = 1,2,\dots,q; i = 1,2,\dots,n) \qquad \textit{(10-205)}$$

and if a pattern ξ_i is then presented to the network, its similarity is compared to the stored patterns ξ_i^{η}. If the presented pattern is similar enough (depending from the allowed error rate) to one specific stored pattern, it is related to the specific attractor and a dynamic process is initialized, which eventually converges to the sample pattern. It has been explained in 10.5.1.7.1 Quantum artificial neural network configuration search function that within a QANN the goal is to find the configuration of weights, or Hamiltonian H in quantum terminology, which provides a solution within the pre-defined ANN parameters, which in terms of an associative QANN are a prescribed number of attractors at specific locations and of specific type. Zak and William[406] state that compared to a classical ANN, the approach is completely different, as the structure converges to an attractor via a training algorithm. Arguments for favouring QANNs (instead of classical ANNs) are

- The dimension of the unitary matrix (H) implemented on quantum hardware may be exponentially larger within the same space. As a consequence, the QANN capacity, or the number of patterns the QANN is able to store, as well as the number of dimensions of these patterns, are exponentially larger as well.
- Through interference of patterns, a sort of grammar may be introduced to a QANN (apart from that, only a QANN with high dimensionality and complexity would be able to process such information). This means that if e.g. letters are stored in the form of stochastic attractors ξ_η are stored within the QANN then the simultaneous presentation of a specific number of letters to the QANN is accompanied by quantum interference effects in the sense that they will converge to a new attractor, which not only preserves the identity of the single letters, but simultaneously is not only a sum of these. This is similar to sentences consisting of words, and words consisting of letters. By changing the phases

406 Zak Michail, William Colin P.: Quantum Neural Nets; Center for Space Microelectronics Technology: Jet Propulsion Laboratory; Pasadena: Caltech

of the single components H_{ij}, also the grammar may be changed, let say from English to German.
– QANNs feature attractors representing stochastic processes, as described above.

The application QANNs within an ACE would therefore be beneficial, as a lot more complex and deep artificial neural networks could be used. The major problem with deep ANNs has always been their training, although some approaches have been found that deal with the problem. However, such algorithms either deal with the depth, thus the vertical structure of an ANN, or the horizontal structure, thus the number of neurons within a hidden layer. Additionally to restricted Boltzmann machines, I personally have made very good experience with genetic or thermal algorithms for training deep structures, especially as the latter work well independent from both structural dimensions. However, when dimensionality increases, also such algorithms' performance decreases exponentially. Thus, for training deep and broad networks such as the one(s) in a human brain, the search for the optimal configuration via Grover's algorithm, be it via entangled performance registers or in another way, is a lot more promising than attractor dynamics in classical ANNs.

10.6.3.6 Real world input and new experiences

Now that the ACE is capable of learning from different kinds of databases (whereby I consider the internet as heterogeneous and unstructured database here as well) and structuring this information efficiently, it is required to determine how sensual information is interpreted. Well, for with respect to visual information both spatial and temporal information is of relevance, thus how the content of a scene changes with respect to time. The information we visually perceive is much unstructured, but may be organized on a higher conceptual level. For example, when the ACE would walk down a street, heading from one location A to another location B, then it may interpret objects, which belong to one of the low conceptual levels of the scene (I call it scene, although the whole process may consist of several scenes) simply by scanning the environment. A low conceptual level in this scene may comprise cars, people, a street, a crossing. With this information, the ACE may gain contextual information, such as described in 10.6.2 Context recognition and hierarchical learning. Contextual information in this scene may comprise information such as 'busy', 'accident at crossing', 'dangerous', 'how to pass crossing', etc… Equipped with this contextual information, behavior can be derived, either from stored patterns or databases. Behavior may be the avoidance of collision with objects or other people, how to pass a crossing, etc… I will not go into technical detail here on how visual information can be interpreted with pattern recognizers, as this has been already discussed extensively. However, the information described above could also have been extracted from a static image, thus does not necessarily feature a temporal component. It is improbable that all people and objects in our scene are static; instead they are moving and have their own targets. For the ACE, it is not required to know all these targets, as even we humans do not know that. Nevertheless, what we do is to predict the situation. Our neocortex has been developed for prediction and it does that all the time. It predicts where a car may be in 2 seconds and from that we derive if it is save to cross the street or not, or it predicts where someone heading towards us will be most likely in 3 seconds, so that we can avoid collision. This is also what an ACE should do, which means that it is on the one hand required to access knowledge or trained ANNs, but on the other hand to train such instantaneously for predicting patterns as quickly as possible. For this, not auto associative ANNs are used, but feed forward ones, or better, QANNs, as only the latter

ones may be trained quickly enough for predicting a complete scene. For sure, it is not required to predict the whole scene, but in the given example to extract features of moving objects for a time sequence, and when one of these objects becomes relevant for the ACE, to predict its behavior and derive own behavior from the results.

Not everything may be interpreted correctly, and an inexperienced ACE may make mistakes, as inexperienced humans do. Consider a child having never seen a butterfly before and thus running after one, ignoring its environment, which may not be dangerous on a flat meadow, but in a city. Thus, not even new information about objects, but also about context are collected continuously. For an ACE, there is no need to forget anything, even if the information is of no relevance for its desires and aims. Basically, the interpretation of new information happens in the same way as with basic knowledge, described at 10.6.3.1 Acquisition of basic knowledge, with the difference that it is stored and brought into context with existing knowledge. This then transforms new information into knowledge, or experience. The difference between experience and knowledge is that experience is based on knowledge, thus it is a higher conceptual level of information. From a computer science point of view, experience may be interpreted as a cluster of knowledge, whereby knowledge here does not only comprise hard facts, but of stored situations linked to stored knowledge. These stored situations may be enriched with Meta information and tagged, so that they can be queried and compared to an actual situation, which is then handled based on the results, thus experience. If the handling was successful although the situation differed from the experience in a higher amount, then the related experience-cluster may be extended by the variational information of the new situation. Knowledge may be stored as new pattern recognizers (which encode patterns), and a situation may be stored in manifold ways, one of which I would suggest are sequences situation-relevant patterns.

10.6.3.7 Automatic information interconnection

A human brain does interconnect information, which is the reason why we can learn in one domain from another domain, or explain complex information such as physical processes in metaphors. This is what is also required to happen within an ACE. Information interconnection may be implemented in different areas and ways. The intercompatibility of patterns or a cascade of patterns may e.g. be verified by understanding the type of a problem, or ‘what it really is’. Having understood that evolution, the behavior of ants when searching for food, the cooling of metal or learning within an artificial neural network all are optimization problems, has resulted in the creation of ANN training algorithms that copy paradigms from nature e.g. for predicting weather. The type of a problem can very often be figured out easily by understanding its nature, and understanding its nature has very often to do with understanding context. The implementation of checking for pattern intercompatibility is a meaningful means for solving unsolved problems, thus the ACE may be able to derive solutions from different categories of knowledge, as we humans do.

10.6.4 A superior goal

Now that the theoretical ACE features all the interesting capabilities, it must be equipped with a superior goal. IBM Watson‘s goal e.g. was to answer Jeopardy questions, but this was not a superior goal it pursued itself. The goal of its ingenious developers was to show that Watson can understand and intelligently process input provided in natural language, which in fact was

an extraordinary achievement. However, when Watson is not provided input, he is not pursuing a superior goal, such as continuously improving itself (in fact, he may partially do that in the sense of structuring and learning information, but only when he is told to do so), or contributing to the world's progress. Our goals, for example, are relatively easy to explain from a scientific point of view. For all we do, be it the contact to others, eating or moving we have motivation, which is a central drive deeply rooted in our brain and established evolutionary. The clock for the motivation is the reward system that regulates what we perceive as pleasant or less pleasant by the release of the neurotransmitter dopamine. In former times, when our neocortex has not been as developed as it is now, dopamine may have been released by successfully hunting down an animal, and nowadays it may be the reward of our boss or the money we collect each month. For the ACE, if it should be included in our society, the same reward principle may be implemented – only the goal may be more flexible than we are able to define ours. This sounds difficult at first, but there is a machine learning approach where we actually apply this, namely particle swarm optimization, in which individuals of a swarm represent solutions to a problem moving through n-dimensional space. The movement of each individual is influenced by its local and best-known position as well as the best known positions in the whole search-space. These positions are updated as soon as better positions have been detected by other individuals, which results in an overall movement of the swarm towards the best solution. Schematically, the best solution in case of an ACE may be encoded as the superior goal, whereby the superior goal must be implemented as continuously moving towards infinity, so that it can never really be reached. This is not something mean, as if we think on contributions for improving the world it is far from being perfect – moreover, who can really describe what a perfect world for everybody is like?

10.7 A distributed mind

The search for approaches that allow machines to experience, act and feel in the same way as humans do, brings up not only ethical discussions (which have not been considered within this elaboration), but also new possibilities. In fact, the question after we have been able to reverse-engineer parts of the human mind will be the one on how it will be possible to transfer the human mind into a vessel capable of storing and processing it. What has always disturbed me when thinking on how a mind may be transported from a human brain into an artificial one is the fact that what is commonly described as transport in literature is making a copy – referring to this, quantum teleportation may be more than just an analogy. After having copied one's mind into a vessel capable of holding it, there would remain the fact that the original would still exist. Quantum teleportation, by which quantum information (like the state of a photon) can be transmitted over distance, requires both the use of traditional communication and quantum entanglement between the sending and receiving parties. On the grounds that it relies on upon established communication, which according to special relativity can never reach the speed of light, it cannot be utilized for superluminal transport or correspondence. What's more, it's impossible to violate the no-cloning hypothesis by preparing two duplicates of the framework. Quantum teleportation doesn't transport the quantum system itself and thus it is, regardless of the name, best considered a sort of communication, instead of a real transportation.

It does not matter if conscious experiences are based on quantum effects or not, pure extraction of what we consider to be our mind is, at least according to our current physical understanding, impossible. The solution for the dilemma may be a sort of distributed mind,

thus a constant connection of the human mind with the artificial vessel holding the mind after biological death – shared consciousness between an artificial and a human brain. After biological death, the collective mind can survive, as it would just be partially reduced. The artificial entity does not necessarily need to be sealed off from all environmental stimuli, so that it only acts as mirror of the biological entity's mind, or a better storage device. It would even be more beneficial for the common interface – the mind – if both entities were given the chance of experiencing conscious content. With sufficient spatial and temporal resolution, it should be possible to train a conglomerate of sophisticated machine learning approaches directly by reading and interpreting the signals within a human brain. Apart from difficulties of realizing an artificial conscious entity capable of not only mimicking, but creating conscious content itself (when is a simulation detailed enough that it cannot be counted as simulation any longer, regardless from the implementation?), there now arise new difficulties, namely the question after how the information and the mind located in a biological entity's brain can be shared between it and an artificial counterpart. It is required to have a look on brain machine interfaces (BMI) or brain computer interfaces (BCI) for finding a possible solution. The first thing that has to be realized is the signal detection from the human brain, what can currently be realized by

– non-invasive transducers, by which electrical, magnetic or light-signals produced within the brain are read from the scalp,
– semi-invasive transducers, which are placed on the surface of the cortex, and
– invasive transducers, which require the direct implantation in the human brain.

Any common manifestation of communication or control requires peripheral nerves and muscles, whereby each of these processes starts with the user's intention, which triggers an unpredictable process in which specific areas within the brain are activated. Subsequently, signals are transmitted through the peripheral nervous system (the motor pathways), to the relating muscles, which thus perform the movement fundamental for the communication or control task. The movement coming about because of this process is regularly called motor yield or efferent yield. Efferent methods pass on impulses from the fundamental to the peripheral nervous system and further to an effector, thus a muscle.

Afferent portrays communication from the sensory receptors to the central nervous system, thus vice versa. For movement control, the motor pathway is crucial, whereas the sensory pathway is especially paramount for studying motor abilities and tasks requiring dexterity. BCIs can be applied for offering an alternative method for communication and control – it is an artificial system that detours the form's ordinary efferent pathways, which are the neuromuscular output channels. In place of relying upon peripheral nerves and muscles, a BCI measures mind movement connected with the user's expectation and deciphers the recorded brain activity into comparing control motions for BCI applications, which includes pattern recognition or signal processing, which in turn is ordinarily done by software. Since the measured action begins straight from the brain and not from the peripheral systems or muscles, the system is known as a brain–computer Interface. A BCI in general should be capable of

– activity recording straightforwardly from the brain invasively or non-invasively,
– real-time feedback provision to the user, as well as
– reacting on intentional control. That is, the user should decide to perform a mental assignment at whatever point he needs to fulfil an objective with the BCI. Gadgets that

just inactively distinguish changes in cerebrum action that happen without any goal are not BCIs.

Neuroprostheses, to which BCIs belong to, are gadgets that cannot just receive output from the nervous system, yet can additionally provide input. In addition, they can collaborate with the peripheral and the central nervous system. They are, as recently depicted in the definitions above, immediate fake yield channels from the mind. Unlike other human–computer interfaces, which require muscle movement, BCIs provide non-muscular communication.[407] In general, BCIs offer options to people that are locked within their bodies due to some sort of disablement, but another field of application may be the reading of neural functions from a brain for training the neural networks of an ACE.

10.7.1 Non-invasive transducers

As brain activity produces electrical and attractive action, sensors can be applied for discovering diverse sorts of changes in electrical or attractive action at distinctive times over distinctive regions of the mind. Most BCIs depend on electrical measures of brain action, and further depend on sensors set over the head to measure this movement. Via a BCI continuous brain activity for brain patterns originating specific areas can be analysed. To get predictable recordings from particular districts of the head, researchers depend on a standard system for faultlessly placing electrodes (known as the international 10–20 system).

Electroencephalography (EEG) alludes to recording electrical action from the scalp with electrodes. Although the temporal resolution is good, EEG also has a major disadvantage in the sense that the spatial resolution and the frequency extent are limited, in the sense that it is sensitive to bioelectrical contaminations caused by activities requiring bioelecticity (artifacts), such as muscle movement or even external electromagnetic sources. Another thing to consider is the unwieldy setup procedure EEG comes along with, comprising amongst others skin preparation with abrasive electrode gel. The number of such wet electrodes required can reach up to above 100, which in combination with the electrode gel shows that the application is impractical, and dry electrodes do not provide an equal signal quality to date.

Magnetoencephalography (MEG) records the attractive fields connected with brain activity, whereby functional magnetic resonance imaging (fMRI) measures modest changes in the blood oxygenation level-dependent (BOLD) signals connected with cortical activity. Like fMRI, additionally close infrared spectroscopy (NIRS) is a hemodynamic based technique for assessment of functional activity in the human cortex, where diverse oxygen levels of the blood bring about distinctive optical properties which might be measured. All of these techniques have been utilized for brain–computer communication, yet they all have burdens which make them unrealistic for most BCI requisitions, such as MEG and fMRI are large devices and very expensive and additionally, NIRS and fMRI feature poor temporal resolution.[408,409,410,411,412]

---

407 Graimann Bernhard et al. (2010): Brain-Computer Interfaces: A Gentle Introduction; Berlin Heidelberg: Springer; p. 3 ff.

408 Graimann Bernhard et al. (2010): Brain-Computer Interfaces: A Gentle Introduction; Berlin Heidelberg: Springer; p. 3 ff.

10.7.2 Semi-invasive, invasive transducers and the neural grid

Both approaches require surgery incorporating the opening of the skull through a surgical technique called a craniotomy and cutting the films that blanket the cerebrum. The point when the electrodes are set on the surface of the cortex, the signals recorded from these electrodes is known as the electrocorticogram (ECoG) and thus this approach does not harm any neurons in light of the fact that no electrodes infiltrate the brain.

The signals recorded from electrodes that enter brain tissue is called intra-cortical recording, whereby such approaches usually come along with excellent signal quality, a higher frequency range, and good spatial resolution. Furthermore, bioelectrical contaminations caused by activities requiring bioelectricity can be dealt with better, and complicated procedures for placing electrodes are not required. Intra-cortical electrodes can record the neural action of a solitary brain cell or agglomerations of such, and it records the coordinated activity of a much bigger number of neurons that are in the closeness of the ECoG electrodes.

Any invasive strategy has preferable spatial resolution compared to the EEG, and invasive systems feature advantages over non-invasive techniques, but these come along with the requirement of surgery. Apart from that, budgetary and moral obligations make neurosurgery unreasonable aside from a few users who depend from such kind of BCI to communicate. Understandably, ethical reasons make research on invasive and semi-invasive transducers difficult and restrict it to patients with brain damage or malfunction. It is additionally indistinct if both ECoG and intra-cortical recordings can furnish safe and stable recording over years. Long-term soundness may be particularly dangerous on account of intra-cortical recordings. Electrodes embedded into the brain tissue can cause tissue responses that accelerate breaking down sign quality or even failure of the electrode.[413]

In the not so distant future, we will rely on far more advanced technology. Neural grids, spanning the brain surface and growing through all layers of the neocortex, will allow direct connection on the neuron level. As this allows arbitrary spatial resolution, this will be the first seamless interface between a brain and electronics. Apart from potentially enhancing the capabilities of biological brains, this technology will allow reprogramming of the brain by influencing the intra-neural electrochemical signals by releasing or blocking chemicals.

10.7.3 Signal processing

All of these approaches most likely require a transformation of the output signal, as although the artificial entity's brain may feature the same or even extended capabilities, however, may

409 Niedermeyer E., Silva F. L. D. (2004): Electroencephalography: Basic principles, clinical applications, and related fields; Philadelphia: Lippincott Williams & Wilkins

410 Wolpaw J. R. et al. (2006): BCI Meeting 2005 – workshop on signals and recording methods; IEEE Transactions on Neural Systems and Rehabilitation Engineering: A Publication of IEEE Engineering in Medicine and Biology Society 14, Jun., 138–141

411 Bauernfeind G. et al. (2008): Development, set-up and first results for a one-channel near-infrared spectroscopy system; Biomedizinische Technik, Biomedical Engineering 53, 36–43

412 Dornhege G. et al. (2007): Toward Brain-Computer Interfacing; Massachussetts: The MIT Press

413 Graimann Bernhard et al. (2010): Brain-Computer Interfaces: A Gentle Introduction; Berlin Heidelberg: Springer; p. 3 ff.

not be capable of interpreting the incoming signals directly. Further questions concern the bit-rate of available or consumed data communication resources expressed in bits per time unit, thus the bandwidth, as well as the degrees of freedom, which refers to the number of different channels that can be used for producing the information. A high bandwidth alone cannot solve the problem of information transfer, as it may not be possible to separate to partition the information into separate channels in order to create arbitrary degrees of freedom, as it may not be possible for the brain to produce these orthogonal streams. [414] This simply means that if two or more activities require conscious attention, then these may not be separated into distinct streams of information. The brain patterns utilized as a part of BCIs are portrayed by certain characteristics or properties, such as amplitudes and frequencies, which are vital characteristics of sensorimotor rhythms and SSVEPs. Furthermore, the firing rate of distinct neurons is a significant characteristic of invasive BCIs utilizing intra-cortical recordings. Thus, a BCI measures brain signals and processes these progressively to catch certain patterns that reflect the user's intention, which can be divided into three stages:

10.7.3.1 *Pre-processing*

The signal that has to be interpreted by the BCI has to be pre-processed, which means that it is either required to apply modification, normalization, etc. on the incoming values, but also to reduce the signal-to-noise ratio (SNR). The SNR is a measure for the technical quality of a signal, such as EEG, that is superimposed by a noise signal. It is defined as the ratio of the average power of the desired signal to the average noise power of the interference signal. A bed SNR ratio makes it difficult to interpret the incoming signal, which in case of a BCI means that the desired brain patterns cannot be extracted from the signal (what may happen through artefacts in EEG). If the SNR is good, then the detection and interpretation of the desired brain patterns usually works out well. Similar to an ETL-process in data warehousing, filters or transformations can be applied on the signal, which then allow it to collect the desired information.

10.7.3.2 Feature extraction

After the signal has been cleansed from noise so that the features have become readable the desired information can be extracted. Thus, the BCI does not make use of the whole cleansed signal, but from a dimensionality reduced one. The reason is simply the simplification of the resource amount required for being able to describe a large data set accurately. A machine learning approach useful for feature extraction is, amongst others, artificial neural networks.

10.7.3.3 Detection and classification

The detection and classification of brain patterns is the purpose of BCIs, whereby the user creates brain patterns by performing mental tasks, and the BCI detects, classifies and translate these patterns into commands for BCI applications. This recognition and classification process could be improved when the user communicates with the BCI just in decently

414 Katz Bruce F. (2011): Neuroengineering the future - Virtual minds and the creation of immortality; Massachusetts: Infinity Science Press LLC, p. 187

characterized time periods, and such a time period is indicated by the BCI by visual or acoustic signals. For instance, a beep briefs the user that he could send a command throughout the upcoming time period, throughout which the user is supposed to perform a particular mental task. The BCI then tries to classify the brain signs recorded in this timeline, and as this type of BCI does not take about the plausibility that the user does not wish to communicate anything throughout one of these time spans into consideration, or that the user to communicate outside of a specified time span. This mode of operation is called synchronous or prompt paced and BCIs utilizing this mode of operation are called synchronous or a cue-paced BCIs. In spite of the fact that these BCIs are moderately simple to advance and use, they are unfeasible in numerous true settings. A signal passed BCI is to a degree as a console that can just be utilized at sure times. In an asynchronous or self-paced BCI, users can connect with a BCI at any time they want, without stressing over overall characterized timelines. Therefore, asynchronous BCIs need to scan the brain activity continuously, and this mode of operation is demanding on the one hand side, but on the other hand it offers a more characteristic and advantageous manifestation of communication with the BCI.

10.7.4 BCI requirements for the distributed mind

The requirements a BCI (or a combination of BCIs) are only proposed roughly here, as detailed explanations will be published after further research has been done. At least, the BCI-requirements for implementing a distributed mind are

- The propagation of information (experiences) from the biological entity A to the artificial entity B. The information must incorporate not only sensual information, but the accompanying qualia. As qualia are specific to individuals, the artificial vessel B sharing this information is also specifically suitable and trained for A.
- The interpretation of information on the side of B, which means that A's outgoing signals must be translated in the sense that B is capable of perceiving the same experience as A.
- The interpretation of information on the side of A, as also B is capable of producing conscious experiences.
- Continuous activity recording. Data transfer is not necessarily required to happen synchronous. Activity recording must incorporate sensual information, thus external information influencing internal processing, and qualia, thus internal information.
- The training of artificial neural networks in B, be it quantum or classical ones, may be achieved by extracting the structure and processing information A's of neural networks in the sense that the signals propagated via neurons resulting from the activation of a specific pattern in A is translated into a trained artificial neural network in B.

From the requirements we can see that at least two BCIs are required for sharing conscious experiences – one for the artificial conscious entity and one for the biological entity. Furthermore, if external conditions are responsible for creating experiences, these must be recorded and transferred as well, as only then the same experience may be reproduced in the artificial conscious entity. Current BCIs do not feature enough spatial and temporal resolution for extracting signals created by experiences on the level of single neurons, however, even with non-invasive approaches great progress has been made within the last years. My guess is that in around 15 years first implementations will be feasible.

10.8 Summary

Although quantum effects in human brains may not play a significant or even any role in information processing, I think it has become obvious that from such theories we can learn a lot for creating an artificial mind. The use of quantum computers e.g. for processing artificial neural networks is beneficial in the sense that compared to classical computation a lot more complex and deep artificial neural networks may be created. This is, because major problems with deep ANNs can be dealt with, such as their training performance. Quantum computers feature abilities that classical computers do not have, like the ability of working with quantum linear superpositions or entanglement, which do not have a counterpart in classical computation. We have seen that processing an artificial neural network on a quantum computer might theoretically be possible, however, we would require a combination of numerous and large neural networks for being able to emulate a whole brain. In terms of quantum artificial neural networks, the difficulty is to measure the system, which means to destroy the superposition exactly when the system's state fulfils the required learning criteria, e.g. the number of correctly classified training examples. Moreover, when taking the internal calculations, dynamic thresholds and the weights into consideration, the possible configurations of the ANN within its superposition grow exponentially in the size of the combined input vector plus threshold values. However, quantum computer science does not only allow new approaches for information processing; it opens the door to the future of artificial intelligence. The massive parallel processing quantum computers are capable of resembling some kind of the parallel information processing of the human brain. Thitherto, it has not been fully understood how information in the human brain is processed, not to mention the emergence of human consciousness. The brain is such a complex structure that not only its continuous study will lead to the full understanding, but also several experiments. These experiments may in the near future not only consist of invasive, semi-invasive or non-invasive approaches, but also in the simulation of a brain with all of its ~ 10^{11} neurons and its capability for processing information in parallel with quasi one processor. Although the human brain that may process 2^{13} analog operations and although it is already slower than current supercomputers, it does not require that much energy (only 15-20 watts). Furthermore, the whole neural network allowing unique information processing has more than once been declared to be the most complex information system and thus it is a structure worth being imitated. It is very likely that this will be possible and happen sometime with a quantum computer simulating a biological neural network by a corresponding artificial one. A combination of classical and quantum computation seems to be a useful approach, as I do not want to reduce the brain, be it biological or artificial to a quantum computer.

Furthermore, we are nowadays capable of developing self-organizing models, such as the one described for hierarchical clustering of information. The human brain is a complex self-organizing system, built upon repetitive structures. The self-organization lies in the interconnection of these structures, which remarkably change and increase during reaching adulthood. Apart from that, it is likely that self-organization occurs in numerous other structures; diverse Hebb's learning rules may be utilized as a part of the brain, with distinctive patterns and outcomes developing. It has been mentioned that depending from sensual inputs and environmental stimuli, connections between neurons form, are strengthened or weakened, thus self-organization is not something that our brain experiences once, it is a continuous process – our brain is a dynamical, self-organizing and distributed system. Through each impulse of awareness, neural structures are affected in the sense that learning happens. A major advantage is that every new input is compared to existing experiences with the goal that

similar sensations are processed in the same neural area. It has been explained how the same may be implemented in an artificial brain.

Regarding the required training and learning of the artificial conscious entity's brain, learning from unstructured and structured data structures, from new inputs (experiences) as well as association of knowledge and hierarchical structuring of knowledge have to be taken into consideration. How learning can happen has been explained in detail, whereby an approach combining brain computer interfaces with an artificial brain for creating a distributed mind is an approach that has just been roughly touched on and a lot of research has to be done. Current BMI technologies seem to be not technically sophisticated enough for fulfilling the purpose of extracting information of human brains in the spatial and temporal resolution we would need. Nevertheless, even some of non-invasive technologies are promising, like NIRS, although it features low temporal resolution. Most likely, a combination of non-invasive and (semi-) invasive transducers may be the solution for the problem of information extraction, but this will be discussed in further elaborations.

11 Conclusion

From what has been discussed within this elaboration it is evident that the creation of artificial conscious entities is only a matter of time – the required technology and methodology already exists or is researched on in ambitious projects such as the human brain project.[415] The question therefore is not if it will be possible or not, but how this will influence the world we live in. Ethical questions must be prompted, as we will have to decide upon how we treat artificial conscious entities, namely as conscious beings or as objects such as a today's computer, and which rights will be granted to them, because if an ACE is capable of questioning the world and oneself it will also question its creators.

Apart from that, if we go one step further and consider what it means for humanity and evolution if we advance technology in a way that allows to transfer our minds into an artificial vessel, not only questions of ethics, but also from religious groups will come up. I personally pursue a transhumanist opinion and consider the next step in human evolution towards machine consciousness is not an option, but an absolute requirement, as with continuous population growth the resources earth can provide us with will not suffice, and this will happen before we will be able to conduct interstellar journeys.

415 Human brain project [2013-12-20]; URL: https://www.humanbrainproject.eu/de

F. Neukart, *Reverse Engineering the Mind*, AutoUni – Schriftenreihe 94, DOI 10.1007/978-3-658-16176-7_11

Glossary – computational intelligence

Not all the below explained terms occur within the thesis, but as research on relevant topics might require to understand abbreviations, the thesis glossary has been extended by the general ANN-glossary from Altafkhan[416].

Activation function
is the transform applied to the weighted sum of inputs plus offset for computing the output of a neuron. Also known as the squashing function.

Affine group invariance
The property of a group due to which it stays unchanged after the application of an affine transform.

Affine transform
is a transform from the set of rotations, shifts, scaling, or any combinations thereof.

Alopex
is a stochastic learning procedure which uses local correlations between changes in individual weights and changes in the cost function to update weights. This procedure predates the current resurgence in neural network learning by more than a decade and was originally proposed for mapping visual receptive fields.

Approximation property
is the ability of a set functions to approximate a specific class of functions to any desired accuracy.

Approximation property
is the property of an approximation scheme on a set of functions to select a function that is at a minimum distance from the function to be approximated.

ART-EMAP
is ARTMAP with added spatial and temporal evidence accumulation processes.

ARTMAP
is a supervised learning procedure explicitly based on neurobiology.

ARTMAP-IC
is ARTMAP with an instance counting procedure and a match tracking algorithm.

Attribute
is an element of the input vector. Also known as a feature.

Auto associator
A system for which the desired output response is the same as the input.

416 Altafkhan (2005-2006): Neural Nets Glossary [2012-01-18]; URL: http://www.altafkhan.com/ib/neural-nets-glossary.htm

F. Neukart, *Reverse Engineering the Mind*, AutoUni – Schriftenreihe 94, DOI 10.1007/978-3-658-16176-7

Back propagation
is an procedure in which the difference between the actual and desired responses of the neurons in the output layer is minimised using the steepest-descent heuristic.

Balanced data set
is set in which all classes are equally represented.

Bayesian classifier
assigns a class to an object in such a way that the expectation value of misclassification is minimised. Also known as the minimum risk classifier, and belief network.

Bayesian statistics
differs from the conventional 'frequent' approach to statistics in that it allows the probability of an event to be expressed as `degree of belief' in a particular outcome instead of basing it solely on a set of observations.

Bayes's theorem
allows prior estimates of the probability of an event to be revised in accordance with new observations. It states that probability of an event A given another event B, P(A|B), is equal to P(B|A)P(A)/P(B).

Black-hole mechanism
is a rounding mechanism for 'nearly discrete' weights.

Black-hole radius
If the value of a weight gets within this radius of a discrete value, it becomes that discrete value.

Bootstrap
A random sample is selected by sampling with replacement from the data set and is used to train the network. The trained network is then tested on the remaining data. This procedure is repeated a large number of times. The average of all such test errors is an estimate of the generalization performance metric.

Borel measurable functions
Just about all functions that one may encounter are Borel measurable. Functions that are not Borel measurable do exist but are known to mathematicians only as mathematical peculiarities.

Cascade-correlation
learning method starts with a network without any hidden neurons and systematically increases their number during training until the required performance is achieved.

Classification
is a task in which the desired responses are restricted to a finite set of values.

Cost function
is the quantity that is to be minimized in an optimization experiment. In the case of feed-forward networks this quantity is usually the RMS error in the output of the network. Also known as error measure.

Cross-validation
The data set is divided equally into k randomly selected, mutually exclusive subsets called folds. k-1 networks are trained sequentially on all combinations of k-1 folds, while the performance of the trained networks is tested on the one remaining folds. The average of k-1 such errors is an estimate of the generalization performance metric.

Decision sensitivity
is the likelihood that an event will be detected if it occurs. It is the ratio of true positives to the sum of true positives and false negatives. This metric is especially of importance when it is critical that a an event be detected. Also known as True Positive Ratio.

Decision specificity
is the likelihood that the absence of an even is detected given that it is present. It is the ratio of the true negatives to all negatives. Also known as True Negative Ratio.

Decision surface
is the plot of the response of an output neuron with respect to the inputs.

Disjunctive normal form (DNF).
The form of a logical expression consisting of a single conjunction (·) of a set of disjunctions(+). All logical expressions are expressible in this form.

Effective sample size
(for classification learning tasks) is the number of examples representing the smallest classification group.

EM algorithm.
Expectation Maximisation algorithm calculates the probability density of observations based on parameters and not observations.

Epoch
is the cycle in which all examples in the training set are presented to the network.

Ergodic process
A random process is ergodic if its ensemble and temporal averages are the same.

Error surface
is the plot of the cost-function with respect to all of the weights in a network.

Feed-forward network
consists of a layer of inputs, zero or more layers of hidden neurons, and an output layer of neurons. Generally all neurons in adjacent layers are fully connected to each other with feed-forward synapses only. There are no intra-layer synapses. Also known as the multilayer perceptron.

Fitting (over)
An over-fit is due to the trained network having a higher complexity than the concept embedded in the training data. Also known as memorization and over-specialization.

Fitting (under)
An under-fit is caused by the trained network having a complexity lower than that of the concept embedded in the training data.

Forward pass
The process by which a network computes the output vector in response to an input vector. Also known as recall.

Function approximation
is a task in which the desired output values are continuous. Also known as regression.

Functional
is a scalar-valued continuous linear function defined on a normed linear space.

Generalisation performance
is the accuracy of decision of a trained network on a set of data which is similar to but not the same as the training data set.

Hebbian learning
The main idea behind Hebbian learning is that the synapse between two neurons should be strengthened if they fire simultaneously.

Hidden layer
is the layer of neurons which is not directly connected to the network inputs or outputs.

k-nearest neighbors
is a clustering algorithm that minimises the the sum of squares of distances between the training data and k points.

Lp-norm
measures is a popular form of the cost function for feed-forward networks.

Learning
is the process in which a feed-forward network is forced to adjust its weights such that the network's response to a given input vector becomes closer to the desired response.

Learning (batch)
The type of learning during which weights are updated at the end of every epoch. Also known as off-line learning.

Learning (in-situ)
differs from on-line learning in that the former is the property of a network requiring the deployed network to have adaptive weights, whereas the later is a property of the learning procedure, requiring the weights to be updated on the presentation of every example.

Learning (on-line)
The type of learning during which weights are updated after the presentation of every training example. Also known as pattern and incremental learning.

Learning (supervised)
The learning process in which a system's internal parameters are modified in order to minimize the error in its output with respect to a desired value.

Learning (unsupervised)
The learning process in which a system's internal parameters are modified so that similar input patterns result in similar outputs.

Learning rate
determines the size of the weight modification at each training step.

Likelihood
is the probability density of observations calculated from parameters and not observations.

Linear separability
The property of a classification task by which the members of one class can be separated from the ones from all other classes by a single hyperplane.

Loading problem
The problem of finding the optimal weight values for a given network such that the network performs the required mapping.

Logistic discriminant analysis
chooses classification hyper planes with respect to maximizing a conditional likelihood cost-function and not optimizing a quadratic cost-function which is the case for linear discriminant analysis.

Margin
Error in the output of a neuron is not back propagated if it is within this small margin.

Minima (global)
The points of minimum error on an error surface.

Minima (local)
The points of zero gradient on an error surface which are not global minima.

Mixture representation
of data use a linear combination of Gaussian distributions to represent arbitrary distributions.

Momentum
is a training parameter used in a very common variation on standard error back propagation learning procedure. It controls the effect of the last weight modification on the current weight update.

n-layer network
is a feed-forward network with n - 1 hidden layers.

NP-complete problems.
(non-polynomial time problems) The time required to find the optimal solution for this class of problems grows exponentially with the size of the problem. Also known as intractable problems.

Neural network (artificial)
is a set of interconnected artificial neurons.

Neuron (artificial)
is the fundamental processing element in an artificial neural network. It performs a weighted sum of its inputs, adds the offset value to that sum, and then outputs a certain transform of that sum. Also known as node and processing element (PE).

Ockham's Razor
is the conjecture that if, for a given problem, two solutions with similar performances are available then the one with the lower computational complexity should be preferred.

Offset
is the value added to the weighted sum before the transform is applied to compute a neuron's output. Also known as threshold and bias.

Over-trained
networks have a complexity higher than what is required to learn the concept embedded in training data. They act as look-up tables for the training data and are poor generalizes.

Perceptron
is a feed-forward network with no hidden neurons.

Probability (prior)
is the probability assigned to an event in advance of any empirical evidence. Also known as a priori probability.

Probability (posterior)
is the probability assigned to an event based on observations. Also known as 'a posteriori' probability.

Projection pursuit regression
is a generalisation of the feed-forward network in that it allows more than one type of activation function in the hidden layer. These non-homogeneous activation functions are data-dependent and constructed during learning.

Regularisation
A class of methods designed to avoid overfitting to the training data by enforcing smoothness of the fit.

Ridge regression
The precision of least-squares estimates gets worse with with an increase in dependence between the input variables. Ridge regression estimators are more precise in those situations and are obtained as the estimators whose distance to an ellipsoid centered at a least-squares estimate from the origin of a parameter space is a minimum.

Ridging (constrained)
Optimization procedure in which some norm of the weights is constrained to a specific value.

Ridging (penalized)
Optimization procedure in which the cost function is augmented by a penalty term.

Ridging (smoothed)
Optimization procedure in which noise is introduced in the inputs.

Root Mean Square error
is computed by summing the output layer errors for all examples in a training or test set, dividing the sum by the total number of examples and the number of the output layer neurons, and taking the square root of the resultant. The output layer error is computed by

summing the squares of the individual neuron errors with respect to the desired output. An individual output-layer neuron's error is set to zero if it is less than the margin.

Sampling with replacement
may result in successive samples that are not mutually exclusive, some of the examples may never appear in any of the samples, and there may be repetitions within an individual sample.

Set (closed)
A subset M of metric space N is a closed set if it contains each of its limit points.

Set (finite)
A is finite if all of its elements can be displayed as {a1, a2, . . . ,an} for some integer n.

Set (open)
is the subset G of the metric space X if each point of G is the center of some open sphere contained in G.

Shattered
If a set of functions F includes all possible dichotomies on a set S of points, then S is said to be shattered by F.

Shrinkage
The difference between the training set accuracy of a network and its accuracy on a test set.

Sigmoidal functions
Definitions vary but are generally taken to be bounded, monotone, and continuous, e.g. logistic and tanh(·) functions.

Simulated annealing
is a stochastic optimisation technique inspired by the physical process of annealing.

Skip-layer synapses
Synapses connecting neurons in two non-adjacent layers. Also known as short-cut synapses. Known as main effects in the statistical literature.

Smoothing spline modelling
is piecewise approximation by polynomials of degree n with the requirement that the derivatives of the polynomials are continuous up to degree n-1 at the junctions.

Softmax
The purpose of the softmax activation function is to make the sum of the output neuron responses equal to one, so that the outputs are interpretable as posterior probabilities. Also known as the multiple-logistic function.

Synapse
is a measure of the effect that a neuron's output has on the output of another neuron at the other end of the synapse. Also known as connection, edge, and weight.

Testing
is the process of verifying the function of a trained network against a set of examples which is different from the training examples set.

Training
See Learning.

Training example
is a pair: an input vector, and the desired response to that input vector.

Weight
is the value of a synapse or an offset.

Weight decay
is a common regularisation technique used in feed-forward network training in which the cost-function is augmented with a term which penalises large weight values.

Weight depth
is the number of binary bits in a weight.

Weight elimination
is a regularisation technique used in feed-forward network training in which the cost-function is augmented with a term which penalises the number of non-zero weights.

Weight perturbation
is a hardware-friendly alternative to BP learning. In this method, all of the weights are perturbed in turn and the associated change in the output of the network is used to approximate local gradients.

Weight sharing
is a regularisation technique used in feed-forward network training in which the cost-function is augmented with a term which penalises the number of independent weights.

Glossary – quantum physics

All the following explanations find application within the thesis. Some of the short explanations of the quantum computer science glossary have been taken from Wikipedia[417], but have been extended and corrected due to some inconsistencies and errors.

A complete set of wavefunctions
is the basis of the Hilbert space of wavefunctions with respect to a system.

Born's rule
is the probability of the state $|\alpha\rangle$ collapsing to an eigenstate $|k\rangle$ of an observable is given by $|\langle k|\alpha\rangle|^2$.

Bound state
A state is called bound state if its position probability density at infinite tends to zero for all the time. Roughly speaking, we can expect to find the particle(s) in a finite size region with certain probability.

bra
The Hermitian conjugate of a ket is called a bra $\langle\alpha| = |\alpha\rangle^\dagger$. See 'bra-ket notation'.

bra-ket notation
The bra-ket notation is a way to represent the states and operators of a system by angle brackets and vertical bars, for example, $|\alpha\rangle$ and $|\beta\rangle\langle\alpha|$.

Collapse
'Collapse' means the sudden process which the state of the system will 'suddenly' change to an eigenstate of the observable during measurement.

Degeneracy
See 'degenerate energy level'.

Degenerate energy level
If the energy of different state (wavefunctions which are not scalar multiple of each other) is the same, the energy level is called degenerate. There is no degeneracy in a 1D system.

Density matrix
Physically, the density matrix is a way to represent pure states and mixed states. The density matrix of pure state whose ket is $|\alpha\rangle$ is $|\alpha\rangle\langle\alpha|$.

Density operator
Synonymous to 'density matrix'.

Dirac notation
Synonymous to 'bra-ket notation'.

Eigenstate

An eigenstate of an operator A is a vector satisfying the eigenvalue equation:

417 Wikipedia (2011): Glossary of elementary quantum mechanics [2013-02-17]; URL: http://en.wikipedia.org/wiki/Glossary_of_elementary_quantum_mechanics

F. Neukart, *Reverse Engineering the Mind*, AutoUni – Schriftenreihe 94, DOI 10.1007/978-3-658-16176-7

$$A|\alpha\rangle = c|\alpha\rangle$$

where c is a scalar. Usually, in bra-ket notation, the eigenstate will be represented by its corresponding eigenvalue if the corresponding observable is understood.

Energy spectrum

The energy spectrum refers to the possible energy of a system. For bound system (bound states), the energy spectrum is discrete; for unbound system (scattering states), the energy spectrum is continuous.

Expectation value

The expectation value $\langle W\rangle$ of the observable W with respect to a state $|\alpha\rangle$ is the average outcome of measuring W with respect to an ensemble of state $|\alpha\rangle$. $\langle W\rangle$ is calculated as follows:

$$\langle W\rangle = \langle\alpha|W|\alpha\rangle$$

Hamiltonian $\widehat{H}$

$$\widehat{H}\psi r = -\frac{\hbar^2}{2m}\Delta\psi(r) + V(r)\psi(r)$$

The operator represents total energy of the system, where Δ is the Laplace operator, $\hbar$ the reduced Plack constant, r the 3D-location in space, V the potential energy.

Hermitian operator

An operator satisfying $A = A^{\dagger}$.

Equivalently,

$$\langle\alpha|A|\alpha\rangle = \langle\alpha|A^{\dagger}|\alpha\rangle$$

for all allowable wavefunction $|\alpha\rangle$.

Hilbert space

Given a system, the possible pure state can be represented as a vector in a Hilbert space. Each ray (vectors differ by phase and magnitude only) in the corresponding Hilbert space represent a state.

ket

A wavefunction expressed in the form $|\alpha\rangle$ is called a ket. See ‘bra-ket notation’.

Mixed state

A mixed state is a statistical ensemble of pure state.

Normalizable wavefunction

A wavefunction $|\alpha'\rangle$ is said to be normalizable if $\langle\alpha'|\alpha'\rangle < \infty$. A normalizable wavefunction can be made to be normalized by

$$|\alpha'\rangle \longrightarrow \alpha = \frac{|\alpha'\rangle}{\langle\alpha'|\alpha'\rangle}$$

Normalized wavefunction

A wavefunction $|\alpha\rangle$ is said to be normalized if $\langle\alpha|\alpha\rangle = 1$.

Observable
Mathematically, it is represented by a Hermitian operator.

Position representation and momentum representation
Position representation of a wavefunction: $\psi_\alpha(x,t) \coloneqq \langle x|\alpha\rangle$

Momentum representation of a wavefunction: $\psi_\alpha(p,t) \coloneqq \langle p|\alpha\rangle$

where $|x\rangle$ is the position eigenstate and $|p\rangle$ the momentum eigenstate respectively. The two representations are linked by Fourier transform.

Probability amplitude
Synonymous to 'probability density'.

Probability current
Having the metaphor of probability density as mass density, then probability current J is the current:

$$J(x,t) = \frac{i\hbar}{2m}\left(\psi\frac{\partial\psi^*}{\partial x} - \frac{\partial\psi}{\partial x}\psi\right)$$

The probability current and probability density together satisfy the continuity equation:

$$\frac{\partial}{\partial t}|\psi(x,t)|^2 + \nabla J(x,t) = 0$$

where ∇ represents the gradient.

Probability density
Given the wavefunction of a particle, $|\psi(x,t)|^2$ is the probability density at position x and time t. $|\psi(x_d,t)|^2 dx$ means the probability of finding the particle near x_d.

Pure state
A state which can be represented as a wavefunction / ket in Hilbert space / solution of Schrödinger equation is called pure state. See 'mixed state'.

Quantum numbers
A way of representing a state by several numbers, which corresponds to a complete set of commuting observables. A common example of quantum numbers is the possible state of an electron in a central potential: n, l, m, s, which corresponds to the eigenstate of observables H (in terms of r).

- n represents the principal quantum number and thus describes the 'shell' (electron cloud) of an electron. The basic state is given by $n = 1$, the next state by $n = 2$,.., $n = 7$. The larger n is, the lower is the binding energy of the electron and therefore, the larger is the probability that the electron is farther from the nucleus.
- l represents the secondary or azimuthal quantum number, describing in which of the allowed states of angular momentum an electron actually is. With given n, l may take n values, but at maximum $n-1 : l = 1,..,l = n-1$
- m represents the magnetic quantum number, describing the orientation of an orbital in space.
- s represents the spin quantum number, describing the spin of the electron.

Schrödinger equation

$$-\frac{\hbar^2}{2m}\frac{\partial^2\psi(x,t)}{\partial x^2}+V\psi=i\hbar\frac{\partial\psi(x,t)}{\partial t}$$

Spin wavefunction

Part of a wavefunction of particle(s). See 'total wavefunction of a particle'.

Spinor

Synonymous to 'spin wavefunction'.

Spatial wavefunction

Part of a wavefunction of particle(s). See 'total wavefunction of a particle'.

Square-integrable

Square-integrable is a necessary condition for a function being the position/momentum representation of a wavefunction of a bound state of the system. Given the position representation $\psi(x,t)$ of a state vector of a wavefunction, square-integrable means:

1D:

$$\int_{-\infty}^{+\infty}|\psi(x,t)|^2dx<+\infty$$

3D:

$$\int_{V}^{+\infty}|\psi(r,t)|^2dV<+\infty$$

Stationary state

A stationary state of a bound system is an eigenstate of Hamiltonian operator. Classically, it corresponds to standing wave. It is equivalent to the following things:

- an eigenstate of the Hamiltonian operator
- an eigenfunction of Time-Independent Schrödinger Equation
- a state of definite energy
- a state which 'every expectation value is constant in time'
- a state whose probability density $|\psi(x,t)|^2$ does not change with respect to time, e.g.

$$\frac{\partial}{\partial t}|\psi(x,t)|^2=0$$

State

A state is a complete description of the observable properties of a physical system. Sometimes the word is used as a synonym of 'wavefunction' or 'pure state'.

State vector

synonymous to 'wavefunction'.

Statistical ensemble

A large number of copies of a system.

System

A sufficiently isolated part in the universe for investigation.

Tensor product of Hilbert space

When we are considering the total system as a composite system of two subsystems A and B, the wavefunctions of the composite system are in a Hilbert space $H_A \otimes H_B$, if the Hilbert space of the wavefunctions for A and B are H_A and H_B respectively.

Time-Independent Schrödinger Equation (TISE)

A modification of the time-dependent Schrödinger equation as an eigenvalue problem. The solutions are energy eigenstate of the system.

$$\hat{H}\psi = E\psi$$

Total wavefunction of a particle

For single-particle system, the total wavefunction ψ of a particle can be expressed as a product of spatial wavefunction and the spinor. The total wavefunctions are in the tensor product space of the Hilbert space of the spatial part (which is spanned by the position eigenstates) and the Hilbert space for the spin.

Wavefunction

The word 'wavefunction' could mean one of following:

1. A vector in Hilbert space which can represent a state; synonymous to 'ket' or 'state vector'.
2. The state vector in a specific basis. It can be seen as a covariant vector in this case.
3. The state vector in position representation, e.g. $\psi_\alpha(x_0) = \langle x_0|\alpha\rangle$, where $|x_0\rangle$ is the position eigenstate.

Bibliography

Books

1. Abraham Ajith, Crina Grosan, Pedrycz Witold (2008): Engineering Evolutionary Intelligent Systems; Berlin Heidelberg: Springer-Verlag

2. Abraham Ajith, Hassanien Aboul-Ella, Snáel Vaclav (2009): Foundations of Computational Intelligence Volume 4: Bio-Inspired Data Mining; Berlin Heidelberg: Springer-Verlag

3. Abraham Ajith, Hassanien Aboul-Ella, de Leon F. de Carvalho André Ponce, Snáel Vaclav (2009): Foundations of Computational Intelligence Volume 6: Data Mining; Berlin Heidelberg: Springer-Verlag

4. Abraham Ajith, Hassanien Aboul-Ella, Siarry Patrick, Engelbrecht Andries (2009): Foundations of Computational Intelligence Volume 3 Global Optimization; Berlin Heidelberg: Springer-Verlag

5. Abraham Ajith, Hassanien Aboul-Ella, Snáel Vaclav (2009): Foundations of Computational Intelligence Volume 5: Function Approximation and Classification; Berlin Heidelberg: Springer-Verlag

6. Adeli Hojjat, Hung Shih-Lin (1995):Machine Learning Neural Networks, Genetic Algorithms and Fuzzy Systems; John Wiley and Sons, New York

7. Anderberg Michael R. (1973): Cluster Analysis for Applications, New York: Academic Press Inc.

8. Andreas Tolk (2009): Complex Systems in Knowledge-based Environments: Theory, Models and Applications; Berlin Heidelberg: Springer-Verlag

9. Aziz-Alaoui Moulay, Bertelle Cyril (2009): From System Complexity to Emergent Properties; Berlin Heidelberg: Springer-Verlag

10. Bäck T., Fogel D.B., Michalewicz Z. (1997): Handbook of Evolutionary Computation, Institute of Physics Publishing, New York

11. Bell John S. (1987): Speakable and Unspeakable in Quantum Mechanics, Cambridge: Cambridge University Press

12. Bertelle Cyrille, Duchamp Gérard H. E., Kadri-Dahmani Hakima (2009): Complex Systems and Self-organization Modelling; Berlin Heidelberg: Springer-Verlag

13. Berthold Ed (1999): Intelligent Data Analysis: An Introduction; New York: Springer-Verlag

14. Brown T.H., Chattarji S. (1995): Hebbian Synaptic Plasticity; The Handbook of Brain Theory and Neural Networks; Cambridge: MIT Press

15. Chamoni Peter, Gluchowski Peter (2006): Analytische Informationssysteme: Business Intelligence- Technologien und –Anwendungen, 3rd. ed.; Berlin: Springer-Verlag

F. Neukart, *Reverse Engineering the Mind*, AutoUni –
Schriftenreihe 94, DOI 10.1007/978-3-658-16176-7

16. Cloete Ian, Zurada Jacek M. (2000): Knowledge-Based Neurocomputing; Cambridge: MIT Press

17. Crowley J. L., Christensen H. I. (1995): Vision as a Process: Basic Research on Computer Vision Systems, Berlin: Springer

18. Dickmanns E. D. (1991): Dynamic Vision for Perception and Control of Motion, London: Springer, 2007

19. Dornhege Guido et al. (2007): Toward Brain-Computer Interfacing; Massachussetts: The MIT Press

20. Durfee E. H. (1999): Coordination for Distributed Problem Solvers, Boston, MA: Kluwer Academic, 1988

21. Eberhart Russel C., Simpson Patrick K., Dobbins Roy (1996): Computational Intelligence PC Tools; Boston MA: Academic Press

22. Eccles John C. (1994): How the self controls the brain; Berlin Heidelberg: Springer-Verlag

23. Ertel Wolfgang (2008): Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung; Wiesbaden: Friedr. Vieweg & Sohn Verlag | GWV Fachverlage GmbH

24. Falconer Kenneth (2003): Fractal Geometry: Mathematical Foundations and Applications; New York: Wiley

25. Fayyad Usama M., Piatetsky-Shapiro Gregory, Uthurusamy Ramasamy (1996): Advances in Knowledge Discovery and Data Mining; Menlo Park: AAAI Press

26. Franco Leonardo, José M. Jerez (2009): Constructive Neural Networks; Berlin Heidelberg: Springer-Verlag

27. Frankish K., Ramsey W. M. (2014): The Cambridge handbook of artificial intelligence, Cambridge: Cambridge University Press

28. Fulcher John (2008): Computational Intelligence: A Compendium; Berlin Heidelberg: Springer-Verlag

29. Globus Gordon (2003): Quantum Closures and Disclosures: Thinking-together Postphenomenology and Quantum Brain Dynamics; Amsterdam: John Benjamins Publishing

30. Han Jiawei, Kamber Micheline, Pei Jian (2000): Data Mining: Concepts and Techniques; Morgan Kaufmann Publishers

31. Hand David J., Mannila Heikki, Smyth Padhraic (2001): Principles of Data Mining; Cambridge: MIT Press

32. Hannig Uwe (2002): Knowledge Management und Business Intelligence, 1st ed.; Springer: Berlin

33. Harel David (2004): Algorithmics: The Spirit of Computing, 3rd ed.; Amsterdam: Addison-Wesley

34. Hassanien Aboul-Ella, Abraham Arjith, Herrera Francisco (2009): Foundations of Computational Intelligence Volume 2: Approximate Reasoning; Berlin Heidelberg: Springer-Verlag

35. Hassanien Aboul-Ella, Abraham Ajith, Vasilakos Athanasios V., Witold Pedrycz (2009): Foundations of Computational Intelligence Volume 1: Learning and Approximation; Berlin Heidelberg: Springer-Verlag

36. Hayes-Roth F., Waterman D., Lenat D. (1983): Building Expert Systems. Addison-Wesley.

37. Heaton Jeff (2008): Introduction to Neural Networks for Java, 2nd ed.; Chesterfield: Heaton Research, Inc.

38. Heaton Jeff (2010): Introduction to Encog 2.5 for Java, Rev. 3; Chesterfield: Heaton Research, Inc.

39. Heaton Jeff (2010): Programming Neural Networks with Encog 2 in Java; Chesterfield: Heaton Research, Inc.

40. Hebb Donald (1949): Organization of behavior; New York: John Wiley

41. Hornik Mark F., Marcade Erik, Venkayala Sunil (2007): Java Data Mining: Strategy, Standard, and Practice. A Practical Guide for Architecture, Design, and Implementation (Morgan Kaufmann Series in Data Management Systems); San Francisco: Elsevier, Inc.

42. Inuiguchi Masahiro, Hirano Shoji, Tsumoto Shusaku (2003): Rough Set Theory and Granular Computing; Berlin: Springer-Verlag

43. Jain Lakhmi C. (1998): Soft Computing for Intelligent Robotic Systems: Physica-Verlag

44. Jain Lakhmi C. (2008): Computational Intelligence Paradigms: Innovative Applications; Berlin Heidelberg: Springer-Verlag

45. Jain Lakhmi C., Nguyen Ngoc Thanh (2009): Knowledge Processing and Decision Making in Agent-Based Systems; Berlin Heidelberg: Springer-Verlag

46. Jensen Finn V., Nielsen Thomas Dyhre (2001): Bayesian Networks and Decision Graphs; Berlin: Springer-Verlag

47. Katz Bruce F. (2011): Neuroengineering the future - Virtual minds and the creation of immortality; Massachusetts: Infinity Science Press LLC

48. Kaynak Okyay, Zadeh Lofti A., Türksen Burhan (1998): Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications; Berlin: Springer-Verlag

49. Kemper Hans-Georg, Mehanna Walid, Unger Carsten (2010): Business Intelligence – Grundlagen und praktische Anwendungen, 3rd ed.; Wiesbaden: Vieweg+Teubner Verlag | Springer Fachmedien

50. Kolman Eyal, Margaliot Michael (2009): Knowledge-Based Neurocomputing: A Fuzzy Logic Approach; Berlin Heidelberg: Springer-Verlag

51. Kramer Oliver (2009): Computational Intelligence: Eine Einführung; Berlin Heidelberg: Springer-Verlag

52. Kreiszig Erwin (1999): Advanced Engineering Mathematics, 8th ed.; Singapore: John Wiley & Sons

53. Krengel Ulrich (1988): Einführung in die Wahrscheinlichkeitstheorie und Statistik; Braunschweig/Wiesbaden 1988: Verlag Friedrich Vieweg & Sohn

54. Kung Sun.Y. (1993): Digital Neural Networks. Englewood Cliffs: PTR Prentice Hall

55. Kurzweil Ray (2012): How to create a mind; London: Penguin Books

56. Lau Clifford (1991): Neural networks, theoretical foundations and analysis; Los Alamitos: IEEE Press

57. Lavarac N., Dzeroski S. (1994): Inductive Logic Programming, vol. 3: Nonmonotonic Reasoning and Uncertain Reasoning, Oxford University Press: Oxford

58. Levy Steven (1997): Artificial Life: A Report From the Frontier: Where Computers Meet Biology; New York: Vintage Books

59. Liu Dikai, Wang Lingfeng, Tan Kay Chen (2009): Design and Control of Intelligent Robotic Systems; Berlin Heidelberg: Springer-Verlag

60. Mallat Stephane (1999): A Wavelet Tour of Signal Processing; Boston MA: Academic Press

61. Mandic Danilo P., Chamber Jonathon (2001): Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures, and Stability; New York: Wiley

62. Mange D., Tomassin M. (1998): Bio-Inspired Computing Machines. Presses, Laussanne: Polytechniques et Universitaries Romandes

63. Mermin David N. (2007): Quantum Computer Science: An Introduction; Cambridge: Cambridge University Press

64. Nedjah Nadia et al. (2009): Innovative Applications in Data Mining; Berlin Heidelberg: Springer-Verlag

65. Nedjah Nadia et al. (2009): Intelligent Text Categorization and Clustering; Berlin Heidelberg: Springer-Verlag

66. Nolfi Stefano, Floreano Dolfi (2000): Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines; Bradford Books

67. Onwubolu Godfrey C. (2009): Hybrid Self-Organizing Modeling Systems Berlin Heidelberg: Springer-Verlag

68. Ott Edward (2002): Chaos in Dynamical Systems, UK: Cambridge University Press

69. Padgham Lin, Winikoff Michael (2004): Developing Intelligent Agent Systems: A Practical Guide to Designing, Building, Implementing and Testing Agent Systems (Wiley Series in Agent Technology); New York: Wiley

70. Penrose Roger (1991): The Emperor's New Mind Concerning Computers, Minds, and the Laws of Physics; Oxford: Oxford University Press

71. Penrose Roger (1994): Shadows of the mind – a search for the missing science of consciousness; Oxford: Oxford University Press, p. 335 ff.

72. Phillips-Wren Gloria, Ichalkaranje Nikhil (2008): Intelligent Decision Making: An AI-Based Approach (Studies in Computational Intelligence); Berlin Heidelberg: Springer-Verlag

73. Rahm Erhard (2009): Data Cleansing: Problems and Current Approaches; Leipzig: University of Leipzig

74. Ricciardi L. M., Umezawa U. (1967): Brain physics and many-body problems, cybernetics, vol. 4

75. Ritter Helge, Martinez Thomas, Schulten Klaus (1991): Neuronale Netze. Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke; Addison Wesley

76. Ruan Da, Hardeman Frank, van der Meer Klaas (2008): Intelligent Decision and Policy Making Support Systems (Studies in Computational Intelligence); Berlin Heidelberg: Springer-Verlag

77. Runkler Thomas A. (2010): Data Mining - Methoden und Algorithmen intelligenter Datenanalyse; Wiesbaden: Vieweg+Teubner | GWV Fachverlage GmbH

78. Rutkowski Leszek (2008): Computational Intelligence Methods and Techniques; Berlin Heidelberg: Springer-Verlag

79. Shawe-Taylor Cristianni N. (2000): Support Vector Machines and Other Kernel-based Learning Methods; UK: Cambridge University Press

80. Smolensky Pavel (1986): Information processing in dynamical systems: Foundations of harmony theory. In Rumelhart, D. E. and McClelland, J. L., editors, Parallel Distributed Processing: Volume 1: Foundations, pages 194-281. MIT Press, Cambridge, MA.

81. Tahir Mukarram A. (2007): Java Implementation of Neural Networks; USA: Booksurge Publishing Inc.

82. Teshnehlab M., Watanabe K. (1999): Intelligent Control Based on Flexible Neural Networks (Intelligent Systems, Control and Automation: Science and Engineering); Dordrecht: Kluwer Academic Publishers

83. Tolk Andreas (2009): Complex Systems in Knowledge-based Environments: Theory, Models and Applications (Studies in Computational Intelligence); Berlin Heidelberg: Springer-Verlag

84. Tsau Lin Y., Xie Ying, Wasilewska Anita, Liau Churn-Jung (2008): Data Mining: Foundations and Practice; Berlin Heidelberg: Springer-Verlag

85. Vasantha Kalyani D., Rajasekaran Sundaramoorthy (2009): Pattern Recognition Using Neural and Functional Networks; Berlin Heidelberg: Springer-Verlag

86. Venugopal K. R., Srinivasa K. G., Patnaik L. M. (2009): Soft Computing for Data Mining Applications (Studies in Computational Intelligence); Berlin Heidelberg: Springer-Verlag

87. Wasserman Philip D. (1989): Neural Computing: Theory and Practice; New York: Van Nostrand Reinhold

88. Watanabe Keigo, Hashem M. M. A. (2004): Evolutionary Computations: New Algorithms and Their Applications to Evolutionary Robotics; Heidelberg: Springer-Verlag

89. Watson Ian (1997): Applying Case-Based Reasoning: Techniques for Enterprise Systems; San Francisco: Morgan Kaufmann

90. Yin Yong, Kaku Ikou, Tang Jiafu, Zhu JianMing (2011): Data Mining: Concepts, Methods and Applications in Management and Engineering Design (Decision Engineering), UK: Springer-Verlag

Articles / Book Chapters / Papers

91. Alahakoon D. et al. (2000): Dynamic self organizing maps with controlled growth for knowledge discovery; IEEE Transactions on Neural Networks; vol. 11, pp. 601--614

92. Alcubierre Miguel (1994): The warp drive: hyper-fast travel within general relativity; Classical and.Quantum.Gravity 11:L73-L77, 1994

93. Aschbacher Helmut, Neukart Florian, Schatzl, Sebastian (2009): The use of Business Intelligence and Data Mining for improving the detection of Customer Needs in Service Engineering; Graz: Campus02 University of Applied Sciences

94. Bahrammirzaee Arash (2010): A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems; Neural Computing and Applications, 8(19): 1165 – 1195

95. Bajcsy R. (1988): Active perception, Proceedings of the IEEE, 76:996-1005

96. Barro S. et. al. (1998): Classifying multichannel ECG patterns with an adaptive neural network; IEEE Engineering in Medicine and Biology Magazine; 17(1):45–55

97. Bartz-Beielstein Thomas et al. (2005): Sequential Parameter Optimization; MCKAY, B.: Proceedings of the IEEE Congress on Evolutionary Computation; IEEE Press, 1: p. 773–780

98. Bauer H. U., Villmann T. (1995): Growing a Hypercubical Output Space in a Self-Organizing Feature Map; ICSI Tech Rep. TR-95-030

99. Bauernfeind G. et al. (2008): Development, set-up and first results for a one-channel near-infrared spectroscopy system; Biomedizinische Technik, Biomedical Engineering 53, 36–43

100. Bednar J.A., Miikkulainen R. (2003): Learning Innate Face Preferences; Neural Computation, 15(7)

101. Behrman Elizabeth C., Niemel Jari, Steck James E., Skinner S. R. (1996): A quantum dot neural network; Proceedings of the 4th Workshop on Physics of Computation, p. 22–24

102. Bezdek James (1980): A convergence theorem for the fuzzy isodata clustering algorithms.;IEEE Trans. Pattern Analysis and Machine Intelligence 2, 1–8

103. Bond H. Ah., Gasser L. (1988): Readings in Distributed Artificial Intelligence, San Mateo, CA: Morgan Kaufmann

104. Bonet Blai et al. (2001): Planning and Control in Artificial Intelligence: A Unifying Perspective; Applied Intelligence, 3(14): 237 – 252

105. Bratman M., Israel D. J., Pollack M. E. (1988): Plans and resource-bounded practical reasoning, Computational Intelligence, 4: 156-72

106. Burnet Frank M. (1959): The Clonal Selection Theory of Acquired Immunity; Cambridge University Press.

107. Busoniu L., Babuska R., De Schutter B. (2008): A comprehensive survey of multi-agent reinforcement learning, IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews 38: 156-72

108. Buxton H. (2003): Learning and understanding dynamic scene activity: A review, Vision Computing, 21: 125-36

109. Camazine Scott et al. (2001): Selforganization in Biological Systems; Princeton: Princeton University Press

110. Chaumette F., Hutchinson S. (2006): Visual servo control I: Basic approaches, IEEE Robotics and Automation Magazine, 13(4): 82-90

111. Cheng Tai W., Goldgof Dimitry, Hall Lawrence (1998): Fast fuzzy clustering. Fuzzy Sets and Systems 93, 49–56

112. Cover T. M., Hart P.E. (1967): Nearest neighboring pattern classification. IEEE Trans. Information Theory 13, 21–27

113. Dave R. N., Krishnapuram R. (1997): Robust clustering methods: a united view. IEEE Trans. Fuzzy Systems 5, 270–293

114. de Castro L. N., Von Zuben F. J. (2001): An immunological approach to initialize centers of radial basis function neural networks. In Proc. of 5th Brazilian Conference on Neural Networks, 79–84

115. DiVincenzo David. P. (2001): Dogma and heresy in quantum computing; Quantum Information Comp. 1, 1.

116. Drachman D. (2005): Do we have brain to spare?; Neurology 64 (12): 2004–5

117. Edelman G. (2001): Naturalizing consciousness: A theoretical framework; Proceedings of the National Academy of Sciences USA, 100, 5520 - 5524

118. Elman J. L. (1990): Finding Structure in Time; Cognitive Science, 14, 179-211

119. Erwin E. et al. (1995): Models of orientation and ocular dominance columns in the visual cortex: A critical comparison; Neural Computation, 7:425– 468

120. Eschrich S., Ke J., Hall L., Goldgof D. (2003): Fast accurate fuzzy clustering through data reduction. IEEE Trans. Fuzzy Systems 11, 262–270

121. Ezhov Alexandr A., Ventura Dan (-): Quantum neural networks, BSTU Laboratory of Artificial Neural Networks

122. Fahrmeir L. et al. (1996): Regressionsanalyse; In: Fahrmeir, L. et al. (Hrsg.): Multivariate statistiche Verfahren. 2nd ed.; Berlin, New York, S. 93 – 168

123. Feynman Richard (1982): International Journal of Theoretical Physics, Vol. 21, No. 6/7

124. Forrest S., Perelson A. S., Allen L., Cherukuri R. (1994). Self-nonself discrimination in a computer. In Proceedings of 1994 IEEE Symposium on Research in Security and Privacy, p. 132–143

125. Frawlwy William J. et al. (1992): Knowledge Discovery in Databases - An Overview; AI Magazine: 213-228

126. Frean M. (1990): The upstart algorithm: a method for constructing and training feed-forward neural networks; Neural Computation 2: 198–209

127. Freeman Richard et al. (2002): Self-Organising Maps for Tree View Based Hierarchical Document Clustering; Honolulu: Proceedings of the IEEE IJCNN'02; vol. 2, pp. 1906-1911

128. Fritzke B. (1995): Growing Grid: A self-organizing network with constant neighborhood range and adaptation strength; Neural Processing Letters, 2(5)

129. Fukushima K. (1980): Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position; Biological Cybernetics, 36, 193–202.

130. Garrett Simon M. (2005): How Do We Evaluate Artificial Immune Systems?; Evolutionary Computation 13(2): 145-178

131. Graimann Bernhard et al. (2010): Brain-Computer Interfaces: A Gentle Introduction; Berlin Heidelberg: Springer; p. 3 ff.

132. Grigorescu Costin-Marius, Moraru Sorin-Aurel, Neukart Florian, Badea Milian (2010): BUFFERING APPLICATION FOR AN INDUSTRIAL MONITORING SOFTWARE SYSTEM. Proceedings of Optimization of Electrical and Electronic Equipment (OPTIM), 2010 12th International Conference on, p. 780-785.

133. Grossberg S. (1976): Adaptive pattern classification and universal recording, 1: Parallel development and coding of neural feature detectors; Biological Cybernetics; 23:121–134

134. Grover Lov K. (1996): A fast quantum mechanical algorithm for database search, Proceedings of the 28th Annual ACM Symposium on the Theory of Computation, pp.212-219.

135. Heller Katherine A. et al. (2009): Infinite Hierarchical Hidden Markov Models; Proceedings of the 12th International Conference on Artificial Intelligence and Statistics; Florida: Clearwater Beach

136. Hinton Geoffrey E., Salakhutdinov Ruslan R. (2006): Reducing the dimensionality of data with neural networks,y Science, vol. 313, no. 5786, pp. 504–507.

137. Hinton Geoffrey E., Sejnowski Terrence J. (1983): Optimal Perceptual Inference. Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, Washington DC, pp. 448-453.

138. Hoiem D., Efros A. A., Hebert M. (2006): Putting objects in perspective, Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2137-44

139. Hopfield Joseph J. (1982): Neural networks and physical systems with emergent collective computational properties; Proceedings Nat. Acad. Sci. (USA) 79, 2554-2558.

140. Hou Zeng-Guang et al. (2009): Editorial to special issue: computational intelligence for optimization, modeling and control; Neural Computing & Applications, 5(18): 407-408.

141. Hsieh William W. (2009): Machine Learning Methods in the Environmental Sciences, In: Hsieh, William W. (eds) Neural Networks and Kernels; Vancouver: University of British Columbia

142. Hubel D., Wiesel T. (1968): Receptive fields and functional architecture of monkey striate cortex; Journal of Physiology (London), 195, 215–243.Fukushima, K. (1980): Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position; Biological Cybernetics, 36, 193–202.

143. Ichimura Takumi et al. (2004): A learning method of immune multi-agent neural networks; NEURAL COMPUTING & APPLICATIONS, 14(2): 132 – 148

144. Jerne N. (1974): Towards a network theory of the immune system. Annals of Immunology, 125: 373–389

145. Jordan Michael I. (1986): Attractor dynamics and parallelism in a connectionist sequential machine; Proceedings of the Eighth Annual Conference of the Cognitive Science Society; Englewood Cliffs: Erlbaum, pp. 531-546

146. Karaboga Dervis et al. (2009): A survey: algorithms simulating bee swarm intelligence; Artificial Intelligence Review, 1-4(31): 61 – 85

147. Kirkpatrick Scott et al. (1983): Optimization by simulated annealing; Science, 220(4598): 671–680

148. Kitaev Alexej (1995): Quantum measurements and the Abelian Stabilizer Problem; L. D. Landau Institute for Theoretial Physics

149. Kohonen Teuvo (1990): The Self-Organizing Map; Proceedings of the IEEE 78, Nr. 9, p. 1464-1480

150. Kryzhanovsky M.V. et al. (2010): Neuron network methods of task assignment in multiprocessing system; Optical Memory and Neural Networks, 17(3): 213 – 219

151. LeCun Yann, Bottou Léon, Bengio Yoshua, and Haffner Patrick (1998): Gradient-based learning applied to document recognition; Proceedings of the IEEE, 86(11), 2278–2324

152. Leech G., Garside R., Bryant M. (1994): CLAWS4: The tagging of the British National Corpus. In Proceedings of the 15th International Conference on Computational Linguistics (COLING 94) Kyoto, Japan, pp. 622-628

153. Legg Shane et al. (2007): Universal Intelligence: A Definition of Machine Intelligence, Minds and Machines; 4(17): 391 – 444.

154. Li Weigang: Entangled Neural Networks; Brazil: University of Brasilia

155. Lin Y. T. (1999): Granular computing: fuzzy logic and rough sets. In: Zadeh LA, Kacprzyk J. (eds.): Computing with Words in Information/Intelligent Systems; Springer-Verlag: Berlin

156. Lohn I. D., Reggia J. A. (1997): Automatic discovery of self-replicating structures in cellular automata; IEEE Trans. Evolutionary Computation, 1(3): 165–178

157. Lomonaco Samuel J. Jr. (2000): Grover's Quantum Search Algorithm; Mathematics Subject Classification. Primary 81P68; Secondary 81-01.

158. Lucadou Walter von (1986): Experimentelle Untersuchungen zur Beeinflußbarkeit von stochastischen quantenphysikalischen Systemen durch den Beobachter; Frankfurt am Main

159. MacGregor R. (1991): Using a description classifier to enhance knowledge representation; IEEE Expert 6 (3): 41–46

160. MacQueen J. B. (1967): Some Methods for Classification and Analysis of Multivariate Observations; Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability; Berkeley: University of California Press, 1:281-297

161. Mascioli F. et al. (1995): A constructive algorithm for binary neural networks: the oil spot algorithm; IEEE Trans Neural Netw 6(3): 794–797

162. Matzinger P. (2002): The Danger Model: A renewed sense of self. Science, 296(5566): 301–305

163. Merzenich R.J. et al. (1988): Cortical representation plasticity; Neurobiology of Neocortex; New York: Wiley; pp. 41–68

164. Mettrey W. (1987): An Assessment of Tools for Building Large Knowledge-BasedSystems; AI Magazine 8 (4)

165. Monras Alex et al. (2012): Hidden Quantum Markov Models and non-adaptive read-out of many-body states [2012-10-22]; URL: http://arxiv.org/abs/1002.2337v2

166. Narayanan Ajit; Menneer Tammy (2000): Quantum artificial neural network architectures and components; Information Sciences, volume 124. 1-4, p. 231–255

167. Neukart Florian (2012): A sample algorithm for the system-side adaptation of artificial neural network architectures. Scribd. - The World's Largest Online Library

168. Neukart Florian (2012): Are deep artificial neural network architectures a suitable approach for solving complex business-related problem statements? Scribd. - The World's Largest Online Library

169. Neukart Florian (2013): Accuracy through Complexity - One Step further in Time-Series Prediction and Classification, Knowledge Discovery in Databases.

170. Neukart Florian, Moraru Sorin-Aurel, Grigorescu Costin-Marius (2011): High Order Computational Intelligence in Data Mining - A generic approach to systemic intelligent Data Mining. Proceedings of Speech Technology and Human-Computer Dialogue (SpeD), 2011 6th Conference on, p. 1-9.

171. Neukart Florian, Moraru Sorin-Aurel, Grigorescu Costin-Marius, Szakazs-Simon Peter (2012), Artificial Immune System-inspired NeuroEvolution. Proceedings of DAAM, 2012

172. Neukart Florian, Moraru Sorin-Aurel, Grigorescu Costin-Marius, Szakazs-Simon Peter (2012): Cortical Artificial Neural Networks and their Evolution - Consciousness-inspired Data Mining. Proceedings of OPTIM 2012

173. Neukart Florian, Moraru Sorin-Aurel, Grigorescu Costin-Marius, Szakazs-Simon Peter (2012): Transgenetic NeuroEvolution. Proceedings of OPTIM 2012

174. Neukart Florian, Moraru Sorin-Aurel, Szakazs-Simon Peter (2011): Problem-dependent, genetically evolving Data Mining solutions. Proceedings of DAAAM 2011

175. Newell A., Simon H. A.: Computer science as empirical enquiry: Symbols and search, Communications of the ACM 19:113-26

176. Niedermeyer E., Silva F. L. D. (2004): Electroencephalography: Basic principles, clinical applications, and related fields; Philadelphia: Lippincott Williams & Wilkins

177. Omlin C. W. et al. (1993): Pruning recurrent neural networks for improved generalization performance; Technical report No 93-6, CS Department, Rensselaer Institute, Troy, NY

178. Philippides A. et al. (1999): Diffusible neuromodulation in real and artificial neural networks; In: AI Symposium, Second International Conference on Cybernetics, Applied Mathematics and Physics: CIMAF 1999: Editorial Academia

179. Porter M. F (1980): An algorithm for suffix stripping: Program, 14 no3, pp 130-137

180. Rahm Erhard, Hai Do Hong (2009): Data Cleansing: Problems and Current Approaches, Leipzig: University of Leipzig

181. Ranganathan Ananth, Zsolt Kira: Self-Organization in Artificial Intelligence and the Brain; Atlanta: Georgia Institute of Technology

182. Rauber Andreas (1999): LabelSOM: On the labelling of selforganizing maps; Washington: Proceedings International Joint Conference on Neural Networks

183. Ricks Bob, Ventura Dan (2003): Training a Quantum Neural Network; Provo: Brigham Young University

184. Riedmiller Martin et al. (1993): A direct adaptive method for faster back propagation learning: The Rprop algorithm; Proceedings of the IEEE International Conference on Neural Networks, IEEE Press: 586-591

185. Rosenblatt Frank (1958): The Perceptron. A Probabilistic Model for Information Storage and Organization in the Brain; Psychological Reviews, 65: 386–408

186. Salton G. (1988): Automatic text processing: the transformation, analysis, and retrieval of information by Computer; Addison-Wesley: Massachusetts

187. Salton G., McGill M. J. (1983): Introduction to modern information retrieval; New York: McGraw-Hill

188. Schaffer J. D. et al. (1992): Combinations of genetic algorithms and neural networks: a survey of the state of the art. In: Proceedings of the International Workshop of Genetic Algorithms and Neural Networks, pp. 1–37

189. Segev Aviv et al. (2006): Context recognition using internet as a knowledge base; Journal of Intelligent Information Systems (2007) 29:305–327

190. Serre T., Wolf L., Bileschi S., and Riesenhuber M. (2007): Robust object recog- nition with cortex-like mechanisms; IEEE Trans. Pattern Anal. Mach. Intell., 29(3), 411–426. Member-Poggio, Tomaso.

191. Sim Kwang Mong (2001): Bilattices and Reasoning in Artificial Intelligence: Concepts and Foundations; Artificial Intelligence Review, 3(15): 219 – 240

192. Singer Wolf (2009): The Brain, a Complex Self-organizing System; European Review, Vol. 17, No. 2, 321–329

193. Sipper M., Sanchez E., Mange D., Tomassini M., Perez-Uribe A., Stauffer A. (1997): A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems; IEEE Trans. Evolutionary Computation, 1(1): 83–97

194. Sirivastava Nitish, Hinton Geoffrey, Krizhevsky Alex, Sutskever Ilya, Salakhutdinov Ruslan (2014): Dropout: A Simple Way to Prevent Neural Networks from Overfitting; Journal of Machine Learning Research 15; 1929-1958

195. Spärck Jones K. (1999): Information retrieval and artificial intelligence, Artificial Intelligence 141: 257-81

196. Stanley O. Kenneth et al. (2002): Evolving Neural Networks through Augmenting Topologies; Evolutionary Computation 10(2): 99-127

197. Stepniewski S. W. et al. (1997): Pruning back propagation neural networks using modern stochastic optimization techniques; Neural Computation Applications 5: 76–98

198. Swindale N.V. (1996): The development of topography in the visual cortex: a review of models, Network 7:161–247

199. T. M. Straat, M. A. Fischler: Context-based vision: Recognizing objects using information from both 2D and 3D imagery, IEEE Transactions on Pattern Analysis and Machine Intelligence, 13: 1050-65

200. Tan A. (1997): Cascade artmap: Integrating neural computation and symbolic knowledge processing; IEEE Transactions on Neural Networks; 8(2):237–250

201. Toffoli Tommaso (1981): Mathematical Systems Theory 14 13

202. Vapnik V. (1998): The support vector method of function estimation. In: Suykens J., Vandewalle J. (eds.): Nonlinear Modeling: Advanced Black-Box Techniques; Boston MA: Kluwer: 55–85

203. von der Malsburg Christoph (1973): Self-organization of orientation sensitive cells in the striate cortex; Kybernetik, 14:85–100

204. Wang L. P. (1998): On chaotic simulated annealing; IEEE Trans. Neural Networks, 9: 716–718

205. Wang L. P. (2004): A noisy chaotic neural network for solving combinatorial optimization problems: Stochastic chaotic simulated annealing; IEEE Trans. System, Man, Cybernetics, Part B – Cybernetics, 34(5): 2119–2125

206. Weiss G.: Multiagent Systems: A modern approach to distributed artificial intelligence, Cambridge, MA: MIT Press

207. Williamson J. R. (1996): Gaussian artmap: A neural network for fast incremental learning of noisy multidimensional maps; Neural Networks; 9(5):881–897

208. Wolpaw J. R. et al. (2006): BCI Meeting 2005 – workshop on signals and recording methods; IEEE Transactions on Neural Systems and Rehabilitation Engineering: A Publication of IEEE Engineering in Medicine and Biology Society 14, Jun., 138–141

209. Wong Eugene (1991): Stochastic Neural Networks; California: University of Berkelay, Department of Electrical Engineering and Computer Science

210. Yang M.-S., Ko C.-H. (1996): On a class of fuzzy c-numbers clustering procedures for fuzzy data. Fuzzy Sets and Systems 84(1), 49–60

211. Yao X. (1999): Evolving neural networks; Proceedings of the IEEE 87(9), 1423–1447

212. Yu Tina et al. (2008): Evolutionary Computation in Practice: Studies in Computational Intelligence, Genetic Programming and Evolvable Machines, 4(9): 371 – 372

213. Zak Michail, William Colin P.: Quantum Neural Nets; Center for Space Microelectronics Technology: Jet Propulsion Laboratory; Pasadena: Caltech

214. Zhang Nevin Lianwen (1998): Computational Properties of Two Exact Algorithms for Bayesian Networks; Applied Intelligence, 2(9): 173 – 183

Theses

215. Costea A. (2007): COMPUTATIONAL INTELLIGENCE METHODS FOR QUANTITATIVE DATA MINING; PhD Thesis; Turku: Abo Akedemi University

216. Gorman B. (2009): Imitation Learning Through Games: Theory, Implementation and Evaluation; PhD Thesis; Dublin: Dublin City University

217. Neukart Florian (2013): System Applying High Order Computational Intelligence in Data Mining and Quantum Computational Considerations Concerning the Future of Artificial Intelligence; Brasov: Transilvania University of Brasov

218. Sasu Lucian Mircea (2006): Computational Intelligence Techniques in Data Mining; PhD Thesis; Brasov: Transilvania University of Brasov

219. Singh Gurwinder (2009): Quantum Neural Network Application for Weather Forecasting; Thapar: Thapar University

URLs

220. Apache foundation: hadoop; [2013-10-29]; URL: http://hadoop.apache.org/

221. AISWeb (2012): Immune-Inspired Algorithms [2012-03-13]; AISWeb; URL: http://www.artificial-immune-systems.org/algorithms.shtml

222. Altafkhan (2005-2006): Neural Nets Glossary [2012-01-18]; URL: http://www.altafkhan.com/ib/neural-nets-glossary.htm

223. Altaisky M.V. (2001): Quantum neural network; Technical report; URL: http://xxx.lanl.gov/quantph/0107012

224. Berners-Lee T., Hendler J., Lassila O. (2001): The Semantic Web A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities; Scientific American 284: 34–43

225. Bonsor Kevin, Strickland Jonathan (2000): How Quantum Computers Work [2012-10-18], URL: http://computer.howstuffworks.com/quantum-computer.htm

226. Budnik Paul: The measurement problem [2012-10-06]; URL: http://www.mtnmath.com/faq/meas-qm-2.html

227. Carnegie Mellon: Welcome to the Claytronics Project [2013-09-22]; URL: http://www.cs.cmu.edu/~claytronics/

228. Deeplearning.net (2012): Restricted Boltzmann Machines (RBM) [2012-15-08]; Deeplearning.net; URL: http://deeplearning.net/tutorial/rbm.html

229. Deeplearning4j.org (2016): Convolutional networks [2016-06-19]; URL: http://deeplearning4j.org/convolutionalnets

230. Dipartimento di Elettronica e Informazione, Politecnico Di Milano: Clustering [2011-18-09]; Dipartimento di Elettronica e Informazione; URL: http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html

231. Douglass Jeremy (2001): Self-Organizing Maps [2011-22-07]; University of California Santa Barbara; URL: http://www.english.ucsb.edu/grad/studentpages/jdouglass/coursework/hyperliterature/soms/#SOMs

232. Dror Gideon (2009): Part-of-Speech Tagging [2013-12-08]; URL: http://www2.mta.ac.il/~gideon/courses/nlp/slides/chap10_pos.pdf

233. DTREG (2011): RBF Neural Networks [2012-01-01]; URL: http://www.dtreg.com/rbf.htm

234. Edwin Chen's Blog (2011): Introduction to Restricted Boltzmann Machines [2012-28-08]; URL: http://blog.echen.me/2011/18/introduction-to -restricted-boltzmann-machines

235. Fujita Yukari, Matsui Tetsuo (2002): Quantum gauged neural network: U(1) gauge theory; Technical report; URL: http://xxx.lanl.gov/cond-mat/0207023.

236. Gershenfeld Neil, Chuang Isaac L. (1996): Quantum Computing with Molecules [2012-12-19]; URL: http://www.mat.ucm.es/catedramdeguzman/old/01historias/haciaelfuturo/Burgos090900/quantumcomputingSciAmer/0698gershenfeld.html

237. Hameroff Stuart: Quantum consciousness [2013-07-28]; URL: http://www.quantumconsciousness.org

238. Han Jiawei et. al. (2006): Data Preprocessing [2011-11-08]; University of Illinois; URL: http://www.cs.uiuc.edu/homes/hanj/bk2/02.ppt

239. Heaton Research (2005 - 2011): Applying Multithreading to Resilient Propagation and Back propagation [2009-26-10]; Heaton Research; URL: http://www.heatonresearch.com/encog/mprop/compare.html

240. Heaton Research (2005 - 2011): Encog [2011-22-11]; Heaton Research; URL: http://www.heatonresearch.com/encog/

241. Heaton Research (2010): Introduction to Neural Networks for Java, Session 7 [2011-24-09]; Heaton Research; URL: http://www.heatonresearch.com/course/intro-neural-nets-java/7

242. Heaton Research (2005 - 2011): A Really Simple Introduction to Normalization [2011-12-08]; Heaton Research; URL: http://www.heatonresearch.com/content/really-simple-introduction-normalization

243. Heaton Research (2005 - 2011): The number of Hidden Layers [2011-28-09]; URL: http://www.heatonresearch.com/node/707

244. Heaton Research (2005 - 2012): Quantum Computing; [2012-10-11]; URL: http://www.heatonresearch.com/articles/1/page5.html

245. Held Werner: Quantenphysikalische Ansätze des Bewusstseins [2013-06-22]; URL: http://www.werner-held.de/pdf/qc.pdf

246. Human brain project [2013-12-20]; URL: https://www.humanbrainproject.eu/de

247. IBM: The data mining process [2011-10-08]; IBM; URL: http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.im.easy.doc/c_dm_process.html

248. IBM: What is IBM Watson? [2016-07-14]; URL: http://www.ibm.com/watson/what-is-watson.html

249. Jyh Ying Peng (2003): Quantum Computation Lecture Notes [2012-10-06]; URL: http://red.csie.ntu.edu.tw/QC/peng/chap1.pdf

250. Leeds University: Automatic Mapping Among Lexico-Grammatical Annotation Models [2013-12-08]; URL: http://www.comp.leeds.ac.uk/amalgam/tagsets/brown.html

251. Mache Niels: RPROP [2011-31-08]; Hong Kong Polytechnic University; URL: http://www.eie.polyu.edu.hk/~enzheru/snns/SNNSinfo/UserManual/node152.html#man

252. Milenova Briana et. al. (2002): O-Cluster: Scalable Clustering of Large High Dimensional Datasets [2011-18-09]; Oracle Corporation; URL: http://www.oracle.com/technology/products/bi/odm/pdf/o_cluster_algorithm.pdf

253. Nielsen Michael (2016): Neural Networks and Deep Learning [2016-07-04]; URL: http://neuralnetworksanddeeplearning.com/chap5.html

254. Onboard CRM: Data Cleansing [2011-24-09]; OnboardCRM; URL: http://www.onboardcrm.com/services/data-cleansing.html

255. Philosophie verständlich: Die Libet-Experimente [2013-09-01]; URL: http://www.philosophieverstaendlich.de/freiheit/aktuell/libet.html

256. Quantiki (2005): Basic concepts in quantum computation [2013-01-02]; URL: http://www.quantiki.org/wiki/Basic_concepts_in_quantum_computation

257. Quantum Mind: Quantum Mind [2013-09-01]; URL: http://www.quantum-mind.co.uk

258. Rauber Andreas (1999): LabelSOM: On the Labeling of Self-Organizing Maps [2013-11-01]; URL: http://www.ifs.tuwien.ac.at/ifs/research/pub_html/rau_ijcnn99/ijcnn99.html

259. Rey Günter D.: Neuronale Netze: Eine Einführung [2011-25-08]; Rey Günter D.; URL: http://www.neuronalesnetz.de/aktivitaet.html

260. Scientific Consultant Services (2003): Neural Network Test Data [04-04-2012]; URL: http://www.scientific-consultants.com/nnbd.html

261. Scholarpedia (2011): Boltzmann Machine (2012-15-08); Scholarpedia; URL: http://www.scholarpedia.org/article/Boltzmann_machine#Restricted_Boltzmann_machines

262. Smith Reid (1985): Knowledge-Based Systems Concepts, Techniques, Examples [2016-07-14]; URL: http://www.reidgsmith.com

263. Stackoverflow (2012): How much memory would be required to store human DNA? [2013-11-17]; URL: http://stackoverflow.com/questions/8954571/how-much-memory-would-be-required-to-store-human-dna

264. Stanley O. Kenneth (2004): NeuroEvolution of Augmenting Topologies [2011-12-12]; Carnegie Mellon School of Computer Science; URL: http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume21/stanley04a-html/node3.html

265. Tittensor Derek P. et al. (2011): How Many Species Are There on Earth and in the Ocean? [2013-10-24]; URL: http://www.plosbiology.org/article/info%3Adoi%2F10.1371%2Fjournal.pbio.1001127

266. tutis.ca (2013): The Visual Cortex [2016-06-19]; URL: http://www.tutis.ca/NeuroMD/L2V123/V123.pdf

267. University of Köln: Imputation (Substitution of missing values) [2011-12-08]; University of Köln; URL: http://eswf.uni-koeln.de/lehre/06/05/s11.pdf

268. University of Wuppertal (2013): Der Hamiltonoperator [2013-01-25]; URL: http://hydrogen.physik.uni-wuppertal.de/hyperphysics/hyperphysics/hbase/quantum/hamil.html

269. whatis.com (2011): double-slit experiment [2012-10-06]; URL: http://whatis.techtarget.com/definition/double-slit-experiment

270. Wikipedia (2011): Glossary of elementary quantum mechanics [2013-02-17]; URL: http://en.wikipedia.org/wiki/Glossary_of_elementary_quantum_mechanics

271. WolframMathworld: Colvolution [2016-16-21]; URL: http://mathworld.wolfram.com/Convolution.html

272. wolfram.com: Wolfram Language for Knowledge-Based Programming [2016-07-14]; URL: https://www.wolfram.com/language/

273. WordNet (2013): WordNet a lexical database for English [2013-11-03]; URL: http://wordnet.princeton.edu/

Every revolutionary idea seems to evoke three stages of reaction.
They may be summed up by the phrases:

(1) It's completely impossible.
(2) It's possible, but it's not worth doing.
(3) I said it was a good idea all along.

Sir Arthur C. Clarke