STOCHASTIC OPTIMIZATION, STOCHASTIC APPROXIMATION AND SIMULATED ANNEALING

Optimization problems are central to human existence. Individuals and organizations are often faced with making tradeoffs between different factors in order to obtain desirable outcomes, and the problem of choosing these factors in some "best" way is the essence of the optimization problem. Some of the earliest manifestations occurred at the dawn of civilization when humans developed strategies for societal organization and for obtaining food and shelter. Optimization continues today throughout society in the design, operation, control, and evaluation of modern systems.

Formal optimization is associated with the specification of a mathematical objective function and a collection of factors (or parameters) that can be adjusted to optimize the objective function. In particular, one can formulate an optimization problem as follows:

$$\text{Find } \theta^* \text{that solves } \min_{\theta \in C} L(\theta) \tag{1}$$

where $L : R^p \to R^1$ represents some *loss* function to be minimized, θ represents the p-dimensional vector of adjustable parameters, and $C \subseteq R^p$ represents a constraint set defining the allowable values for the parameters θ. (Note that a *maximization* problem can be trivially recast as the minimization problem in Eq. (1) by applying a minus sign to the objective function.) Our focus in this article will be problems for which θ represents a vector of continuous parameters; this is in contrast to discrete problems such as "how many items X do we need to optimize performance?" Discrete optimization is a large subject unto itself, and will not be considered in any detail here (see, e.g., Ref. 1 for a detailed discussion of this subject from a deterministic perspective). Further, we are in-

terested in the case where L is sufficiently complicated that it is not possible to obtain a closed-form analytical solution to Eq. (1). This is far and away the most common setting for large-scale optimization problems encountered in practice. Hence to solve for θ^* in Eq. (1), one uses an iterative algorithm: a step-by-step method for moving from an initial guess at θ^* to a final value that is expected to be closer to the true θ^* than the initial guess.

One of the major distinctions in optimization is between global and local optimization. All other factors being equal, one would always want the global optimal solution to Eq. (1), that is, the θ^* that provides a lower value of L than any other value of $\theta \in C$. However, in practice this solution is often not available and one must be satisfied with obtaining a local solution. For example, L may be shaped so that there is a clearly defined (local) minimum point over a broad region of the allowable θ space C while there is a very narrow spike at a distant point such that the trough of this spike is lower than any point in the broad region. Hence, the local solution is better than any nearby θ, but may not be the best possible θ. Because of the inherent limitations of the vast majority of optimization algorithms, it is usually only possible to ensure that an algorithm will approach a local minimum with a finite amount of resources being put into the optimization process. However, since the local minimum may still yield a significantly improved solution (relative to no formal optimization process at all), the local minimum may be a fully acceptable solution for the resources available (human time, money, computer time, etc.) to be spent on the optimization. Most of this article will focus on algorithms that are only guaranteed to yield a local optimum; however, we will also consider one type of algorithm (simulated annealing) that aims to find a global solution from among multiple local solutions.

Our focus in this article is the study of iterative algorithms where

1. there is random noise in the measurements of L (and its derivatives if relevant) and/or
2. there is a random choice in computing the search direction as the optimization algorithm iterates toward a solution.

The above two characteristics contrast with classical deterministic optimization, where it is assumed that one has perfect information about the loss function (and its derivatives, if relevant) and that this information is used to determine the search direction in a deterministic manner at every step of the algorithm. In many practical problems one will not have perfect information about the loss function due to inevitable noise effects. A common example is where it is desired to minimize some mean squared error in the performance of some system (e.g., the tracking error in a robot manipulation problem). Only rarely will one be able to compute the mean squared error (or its derivatives); rather, one might be able to get a specific observation of the squared error, but this differs (sometimes very significantly) from the *mean* squared error.

Relative to point 2 above, it is sometimes beneficial to deliberately introduce randomness into the search process as a means of speeding convergence and making the algorithm less sensitive to modeling errors. Although the introduction of randomness may seem counterproductive, it is well known to have beneficial effects in some settings (another case where this is true is in numerical integration, where Monte Carlo methods can be much more efficient in high-dimensional problems than deterministic quadrature approaches).

However, deterministic optimization—which includes linear and nonlinear programming and such well-known methods as steepest descent, Newton–Raphson, and conjugate gradient—provides a useful starting point for the study of stochastic methods. In particular, many (though certainly not all) of the methods in both deterministic and stochastic optimization for the continuous problems of interest in this article rely in some manner on the gradient vector of the loss function with respect to the parameters:

$$\boldsymbol{g}(\theta) \equiv \begin{bmatrix} \dfrac{\partial L}{\partial \theta_1} \\ \vdots \\ \dfrac{\partial L}{\partial \theta_p} \end{bmatrix}$$

Then for local unconstrained optimization (i.e., $C = R^p$), a necessary condition for optimization when L is a continuously differentiable nonlinear function is that θ^* satisfies

$$\boldsymbol{g}(\theta^*) = 0 \tag{2}$$

(constrained problems can also be formulated in this way through, e.g., the use of a penalty function added to the "basic" loss function to penalize violations of the constraints). The proof of this result follows from simple Taylor series arguments showing that if Eq. (2) is not true then one can move θ in some direction that reduces the value of L. Many optimization algorithms are based on Eq. (2), thereby converting the general optimization setting of Eq. (1) to the problem of finding a root to the equation $g(\theta) = 0$.

This article will focus on two broad—and popular—classes of stochastic optimization algorithms: stochastic approximation and simulated annealing. There are many other stochastic optimization algorithms that we are not considering: notably, genetic algorithms, evolutionary strategies, evolutionary programming, and various types of iterative random search. Many references on these other approaches are available to the interested reader. Among these are Ref. 2 for genetic algorithms, Ref. 3 or the journal *Evolutionary Computation* for genetic algorithms and other evolutionary methods, and Refs. 4, 5, or 6 for random search.

The next section of this article reviews the core stochastic approximation (SA) algorithm that is based on direct (but usually noisy) measurements of the gradient vector $g(\theta)$; this is the well-known Robbins–Monro SA algorithm. Then follows an overview of SA when only measurements of the loss function $L(\theta)$ are available [not measurements of the gradient $g(\theta)$]. The subsequent section analyzes in greater detail one of the gradient-free SA methods—simultaneous perturbation SA—introduced previously. Then a review of simulated annealing is given. The final section is a brief summary putting these algorithms into perspective.

ROBBINS–MONRO STOCHASTIC APPROXIMATION

Background and Algorithm Form

We now discuss the well-known Robbins–Monro stochastic approximation (RMSA) algorithm, which is a gradient-based

stochastic optimization algorithm (sometimes referred to as a "stochastic gradient method") for a wide variety of nonlinear problems. This subsection will introduce the basic algorithm form. The following two subsections discuss some of the theoretical properties related to convergence and asymptotic distributions. Then we will summarize three extensions to the basic algorithm form, and finally summarize how the RMSA algorithm is implemented in several different nonlinear applications.

The prototype stochastic optimization application for the RMSA algorithm is the problem of finding a root to the equation $\boldsymbol{g}(\theta) = 0$ based on (noisy) measurements of $\boldsymbol{g}(\theta)$. Let $Y(\theta)$ represent the measurement of $\boldsymbol{g}(\theta)$. [Although this article is written in the language of optimization, many of the ideas carry over directly to the general root-finding context as well, where $\boldsymbol{g}(\theta)$ represents the function for which a zero is to be found.]

RMSA was introduced in a famous 1951 paper (10) and has spawned a large number of follow-on papers. The algorithm has the form

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k Y(\hat{\theta}_k) \tag{3}$$

where a_k is a nonnegative gain sequence that must satisfy certain conditions (discussed below) and $\hat{\theta}_k$ represents the estimate of θ^* at the kth iteration. Since the deterministic term $\partial L / \partial \theta$ does *not* equal the stochastic term Y, the SA algorithm is fundamentally different from the well-known deterministic steepest descent algorithm. However, there is an intuitive connection, since $E(Y) = \partial L / \partial \theta$ under standard RMSA conditions [typically under the relatively modest regularity conditions justifying the interchange of a derivative and an (expectation) integral] (see, e.g., Ref. 7 or 8).

A variation on the basic form in Eq. (3) is to include a projection operator, say Π_C, that automatically maps solutions outside the constraint set C back to the "nearest" point within C. Kushner and Yin (98) treat this approach extensively. In such a case, Eq. (3) becomes

$$\hat{\theta}_{k+1} = \Pi_C[\hat{\theta}_k - a_k Y(\hat{\theta}_k)]$$

We will not treat this form further in this article.

Convergence of the Robbins–Monro Stochastic Approximation Algorithm

As with any optimization algorithm, it is of interest to know whether the iterate $\hat{\theta}_k$ will converge to θ^* as k gets large. In fact, one of the strongest aspects of SA is the rich convergence theory that has been developed over many years. Researchers and analysts in many fields have noted that if they can show that a particular stochastic optimization algorithm is a form of SA algorithm, then it may be possible to establish formal convergence where otherwise that might have remained an open question. In neural networks, for example, White (9) was apparently the first to use this idea to show convergence of certain forms of the well-known backpropagation algorithm. Note that since we are in a stochastic context, convergence is in a probabilistic sense. In particular, the most common form of convergence established for SA is in the *almost sure* (a.s.) (or "with probability one") sense. [A historical note: Robbins and Monro (10) showed conditions for *mean-square* convergence of the iterate, which implies convergence in probability. Blum (11) was apparently the first to give conditions for a.s. convergence. Neither of a.s. and mean-square convergence is implied by the other, but a.s. is stronger than in probability.]

Many sufficient conditions have been given over the years for a.s. convergence of the SA algorithm in Eq. (3). Ruppert (12) and Rustagi (13, Chap. 9), for example, discuss conditions that have largely evolved out of the statistics perspective. Ljung (14), Kushner and Clark (15), Kushner and Yin (98), and Benveniste et al. (16, Chap. I.2) discuss a somewhat different set of conditions that have largely grown out of the engineering perspective. Central to the latter approach is the definition of an underlying ordinary differential equation (ODE) that roughly emulates the SA algorithm in Eq. (3) for large k and as the random effects disappear. It turns out that the convergence properties of this *deterministic* differential equation are closely related to the *stochastic* convergence properties of Eq. (3). Lai (17, Sec. 2) provides a nice intuitive explanation of this differential-equation-based approach.

Probably the most famous of the convergence conditions for RMSA are those on the gain sequence $\{a_k\}$. The conditions provide a careful balancing between wanting to damp out the noise effects as we get near the solution θ^* ($a_k \to 0$) and avoiding premature (false) convergence of the algorithm ($\sum_{k=0}^{\infty} a_k = \infty$). The scaled harmonic sequence $\{a/(k+1)\}$, $a > 0$, is the best-known example of a gain sequence that satisfies the gain conditions (and, as discussed in the next subsection, is an optimal choice with respect to the theoretical rate of convergence, although in practice other decay rates may be superior in finite samples). Usually some numerical experimentation is required to choose the best value of the scale factor that appears in the decaying gain sequence. Other conditions important for convergence relate to the smoothness of $g(\theta)$, the relative magnitude of the noise, and the position of the initial condition.

Asymptotic Normality of Robbins–Monro Stochastic Approximation and Choice of Gain Sequence

We discussed above the issue of convergence of the iterate $\hat{\theta}_k$. This is of central importance in any optimization algorithm. Also of importance is the probability distribution of the iterate (which, recall, is a random vector in our *stochastic* optimization context). Having knowledge of the distribution provides key insight into two main aspects of the algorithm: (1) error bounds for the iterate and (2) guidance in the choice of the optimal gain a_k so as to minimize the likely deviation of $\hat{\theta}_k$ from θ^*.

Unfortunately, the theory governing the asymptotic distribution of the SA iterate is rather difficult. This is to be expected, given the nonlinear transformations arising from the recursion (3): a value $Y(\hat{\theta}_k)$ forms the basis for $\hat{\theta}_{k+1}$, which in turn is the point of evaluation for $Y(\hat{\theta}_{k+1})$ (generally a nonlinear mapping) in the next iteration.

General results on the asymptotic distribution of SA iterates are given by Fabian (18). His work is a generalization of previous asymptotic distribution results for SA by Sacks (19). He shows that, under appropriate regularity conditions,

$$k^{\alpha/2}(\hat{\theta}_k - \theta^*) \xrightarrow{\text{dist}} N(0, \Sigma) \tag{4}$$

as $k \to \infty$, where $\overset{\text{dist}}{\to}$ denotes *convergence in distribution* (see Ref. 20, Chap. 3, or any other graduate-level probability text for a formal definition of this type of convergence), Σ is some covariance matrix that depends on the Hessian matrix of $L(\theta)$ (at $\theta = \theta^*$), $N(0, \Sigma)$ represents a multivariate normal distribution with mean 0 and covariance Σ, and α governs the decay rate for the SA gain sequence a_k [e.g., $a_k = a/(k + 1)^\alpha$]. Ruppert (12) also discusses this result. Various special cases of this result dealing with the situation $\alpha = 1$ are presented in Refs. 13 (pp. 258–259), 21 (pp. 71–78), and 12. The intuitive interpretation of Eq. (4) is that $\hat{\theta}_k - \theta^*$ will be approximately normally distributed with mean 0 and covariance matrix Σ/k^α for k reasonably large.

Equation (4) implies that the rate at which the iterate $\hat{\theta}_k$ approaches θ^* is $k^{-\alpha/2}$. This follows because a random vector with the distribution $N(0, \Sigma)$ on the right-hand side of Eq. (4) is "well behaved" (i.e., not degenerate 0 or ∞ in magnitude), and $\hat{\theta}_k - \theta^*$ must be decaying at a rate $k^{-\alpha/2}$ to balance the $k^{\alpha/2}$ "blowup" factor on the left-hand side of Eq. (4). Under standard conditions on $\{a_k\}$ (see preceding subsection), the rate of convergence of $\hat{\theta}_k$ to θ^* is maximized at $\alpha = 1$.

In practical finite-sample problems, however, the choice of $\alpha = 1$ is not generally recommended. Most analysts and researchers find a lower value of α yields superior finite-sample behavior. (This is a fact that is well known, but not widely documented in the literature because it contradicts the theoretical result suggesting the optimality of $\alpha = 1$; nevertheless, in the author's experience and in the experience of all SA implementers he has consulted, a lower value of α is generally preferable. Other—effectively equivalent—ways exist for slowing down the decay rate of a_k so that it acts like $\alpha < 1$ in finite samples; see, e.g., the practically oriented paper of Okamura et al. (22) for estimation in finite-impulse-response adaptive filters.) The intuitive reason for the desirability of $\alpha < 1$ is that a slower decay provides a larger step size in the iterations with large k, allowing the algorithm to move in bigger steps towards the solution. This observation is a practical *finite-sample* result, as the asymptotic theory showing optimality of $\alpha = 1$ is unassailable.

In fact, in many applications, a *constant* step size ($\alpha = 0$) is used. Typical applications involve adaptive tracking or control problems where θ^* is changing in time. The constant gain provides enough impetus to the algorithm to keep up with the variation in θ^*, whereas if a decaying gain were used, too little weight would be applied to the current input information to allow for the algorithm to track the solution. Such constant-gain algorithms are also frequently used in neural network training even when there is no variation in the underlying θ^* (9,23). Constant-step-size SA algorithms will generally not formally converge. (However, a partial convergence theory *is* possible for constant gains. This is typically based on limiting arguments as the gain magnitude gets small. Essentially, one is able to show that the iterate from a constant-gain algorithm will approach the optimal θ to within some error that decreases as the gain magnitude is made smaller. See, e.g., Refs. 24–28.) Also note that the limiting distribution for the standardized SA iterate [analogous to the left-hand side of Eq. (4)] is *not* generally normal with constant step size (29,26).

Extensions of Standard RMSA

This section discusses some extensions to the basic RMSA framework presented in Eqs. (3)–(4). In particular, we consider: (1) the setting where the observed gradient Y includes a state vector that evolves as θ is being updated, (2) methods of algorithm acceleration by improved choice of the gain sequence, (3) iterate averaging for SA as a means for accelerating convergence, and (4) the setting where the loss function L may change with time.

Joint Parameter and State Evolution. In our first generalization, we replace $Y(\hat{\theta}_k)$ by $Y(\hat{\theta}_k, x_k)$, where x_k represents a state vector related to the system being optimized. The book by Benveniste et al. (16) is devoted to this framework, which was apparently first considered by Ljung (14) and extended by Metivier and Priouret (30). It is typically assumed that x_k evolves online according to a set of certain Markov transition probabilities. The convergence properties of $\hat{\theta}_k$ [as given by Eq. (3)] then depend on the properties of these transition probabilities. Benveniste et al. (16) discuss several applications of this framework in the context of telecommunications, fault detection, and signal processing. One of the common Markov representations of the evolution of the state is via the linear state equation $x_{k+1} = A(\hat{\theta}_k)x_k + B(\hat{\theta}_k)w_k$, where A and B are appropriately dimensioned matrices and w_k is a sequence of independent, identically distributed random vectors (see, e.g., Ref. 14). Then the behavior of the Markov chain, and the associated convergence of $\hat{\theta}_k$, can be tied directly to the stability of this linear process using well-known results in linear systems theory.

A specific application of this form is in temporal difference learning for function approximation (99). The goal here is to approximate the "cost-to-go" function associated with many time-series prediction and control problems. The cost-to-go function measures the expected future cost associated with specific policies (e.g., control strategies) for a dynamic process, given that the process is currently in a particular state. When the state evolves according to a Markov chain, temporal difference learning can be cast as an RMSA form with joint parameter and state evolution.

Adaptive Estimation and Second-Order Algorithms. There are a large number of methods for adaptively estimating the gain sequence (or a multivariate analog of the gain) to enhance the convergence rate of RM-type SAs. Some of these are built on stochastic approximation analogs of the famous Newton–Raphson deterministic optimization algorithm (e.g., Ref. 31). It is known—see, e.g., Ref. 16, Sec. 3.2—that the asymptotically optimal gain for RMSA is a matrix given by $H(\theta^*)^{-1}/k$, where $H(\theta)$ represents the Hessian matrix of $L(\theta)$. This is identical to deterministic optimization, except that we now have the decay factor k included to cope with the stochastic effects. Unfortunately, this is largely of theoretical interest only, since in practice one does not know either θ^* or the Hessian as a function of θ. It also is an asymptotic result, and, as discussed in the preceding subsection, optimality for practical finite-sample analysis may impose other requirements. Nevertheless, this asymptotic result provides a type of ideal in designing adaptive SA algorithms.

Perhaps the first adaptive technique is that of Kesten (32), where one looks at the signs of the difference $\hat{\theta}_{k+1} - \hat{\theta}_k$ in a scalar θ process as a means of designing an adaptive gain sequence a_k (unlike the approaches described below, this approach does not explicitly use the connection with the Hessian matrix as mentioned above). If there are frequent sign

changes, this is an indication that the iterate is near the solution, while if signs are not changing we have an indication that the iterate is far from the solution. This forms the basis for an adaptive choice of the gain a_k, whereby a larger gain is used if there are no sign changes and a smaller gain is used if the signs change frequently. Kesten (32) established a.s. convergence with such a scheme. A multivariate extension of the Kesten idea is described in Delyon and Juditsky (33), including theoretical justification of the extension via a.s. convergence of the iterates.

Ljung and Soderstrom (34, Sec. 2.4, Sec. 3.4, and Chap. 4) and Wei (35) discuss stochastic analogs of the Newton–Raphson search in the context of parameter estimation for particular (possibly linear) models. [These represent extensions of the first paper on adaptive Hessian estimates for *scalar* problems (100).] In so doing, Refs. 34 and 35 demonstrate, via the method of minimizing prediction error, that a batch version of the problem of finding a θ^* satisfying Eq. (2) (where all the data are processed simultaneously) can be converted to the recursive form (where the data are processed one at a time) using only the current (instantaneous) input. The scalar gain a_k is then replaced in their formulation by a matrix that approximates the (unknown) true inverse Hessian matrix corresponding to the current data point's contribution to the loss function.

Ruppert (36) describes an approach where the Hessian is estimated by taking finite differences of a gradient measurement. The gradient used by Ruppert differs slightly from the standard RM gradient in Eq. (3) in that he converts the basic problem from one of minimizing $L(\theta)$ to one of minimizing $\|\partial L/\partial\theta\|^2$ (note that this yields the same θ^* when there is a unique minimum). Spall (102) presents a more efficient approach to general Hessian estimation based on the simultaneous perturbation idea discussed below.

Iterate Averaging. An important and relatively recent development in SA is the concept of iterate averaging. Like many good ideas, this one is simple and, in some problems, can be effective. The idea was jointly described by Polyak and Juditsky (37) and Ruppert (Ref. 12, based on a 1988 internal technical report). There are several variations, but the basic idea is to replace $\hat{\theta}_k$ as our current "best" estimate of θ^* after k iterations with the average

$$\overline{\theta}_k \equiv (k+1)^{-1}\sum_{j=0}^{k}\hat{\theta}_j \tag{5}$$

where each of the $\hat{\theta}_j$ summands is computed as in Eq. (3). The singular achievement of the above references was to show that $k^{1/2}(\overline{\theta}_k - \theta^*)$ is asymptotically normally distributed with mean 0 and a covariance matrix that is as small as possible in a matrix sense. These references establish this optimality of Eq. (5) for gain sequences satisfying the standard conditions plus the conditions $a_{k+1}/a_k = 1 + o(a_k)$ [with $o(\cdot)$ implying a term that goes to 0 faster than the argument]. This important additional condition implies that a_k must decay at a rate *slower* than the optimal rate of $1/k$ for the individual iterates $\hat{\theta}_k$ (as discussed in the preceding subsection). The implications of this result are quite strong: namely, that one can use the standard algorithm in Eq. (3) together with the simple averaging in Eq. (5) to achieve the same optimal rate of convergence that previously was possible only with knowledge of the Hessian $H(\theta^*)$, as discussed under "Adaptive estimation and second-order algorithms" above. This result is available for any valid gain satisfying the above-mentioned additional condition. Hence, in principle, iterate averaging greatly reduces one of the traditional banes of SA implementation—namely, choosing the gain sequence in some optimal way.

Variations on the above basic iterate averaging formulation are readily available. One obvious variation is to not average in the first few iterations, but rather start the average at some $N > 0$ or else use only a sliding window of the last $n - N$ (say) measurements. In practical finite-sample implementations, such modifications are likely to help, since the first few iterations tend to produce the poorest estimates. In the sliding-window approach, formal asymptotically optimal normality can be retained if the window length grows with time (see, e.g., Ref. 38). A further modification to the basic approach is to use the averaged value $\overline{\theta}_k$ (together with $\hat{\theta}_k$) in a modified form of the RMSA iteration [instead of $\hat{\theta}_k$ alone on the right-hand side of the basic form (3)]. This is referred to as the feedback approach in Ref. 39, and can be shown to sometimes yield further improvement.

In practice, however, the results on iterate averaging are more mixed than the above would suggest. Numerical studies by the author have shown that a well-chosen a_k sequence will yield results superior to that possible by averaging (see also Ref. 40, p. 57, and Ref. 41 for some cautionary notes). Once again, this appears to be a consequence of the finite-sample properties of practical problems. More study is required to understand the full capabilities of iterate averaging in practice.

Time-Varying Loss Functions. A final generalization of the Robbins–Monro recursion that we discuss is one where the loss function (and corresponding gradient with respect to θ) varies with k. This problem is treated in Refs. 42 and 43. The basic idea is that, while the loss function may change shape with k, it is assumed that the underlying minimum θ^* is either constant for all k or fixed in the limit as $k \to \infty$ (even though the loss function may change shape indefinitely). The two references mentioned above show a.s. convergence in these cases for the scalar-θ setting; the proofs have to be changed from those commonly seen in SA to accommodate the time-varying loss.

Figure 1 depicts a time-varying loss function $L_k(\theta)$ for a scalar parameter. This figure shows a case where the loss function and the minimizing parameter value both change with time, but one where the optimal parameter value converges to a limiting point θ^*_∞. This is a situation for which the above-mentioned theory would apply.

Applications of Robbins–Monro Stochastic Approximation

RMSA has been applied in a large number of engineering (and other) problems, often under another name. This subsection provides a summary of several applications together with some references for further study.

Neural network (NN) training via the well-known backpropagation algorithm has long been recognized as an application of the RMSA algorithm, two of the publications discussing this connection being Ref. 9 and Ref. 44 (pp.

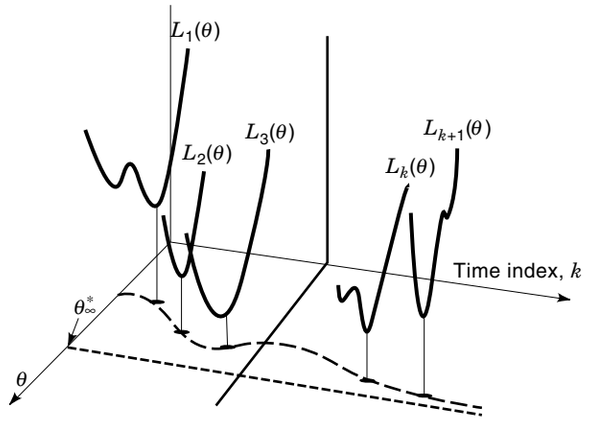

Figure 1. Example of time-varying loss functions with limiting minimum θ^*_∞.

190–199). The essence of the backpropagation algorithm is to systematically compute (via the chain rule applied to the neural network structure) the gradient $g(\theta)$ [or its noisy measurement $Y(\theta)$] for use in an algorithm identical to RMSA (it appears that the authors of some of the earlier publications in this area—e.g., Ref. 45—were unaware of the connection with RMSA).

A popular area for application of RMSA is in recursive estimation for linear models, especially as applied in signal processing, fault detection, and adaptive control [e.g., Benveniste et al. (16, Chap. I.1), Ljung et al. (21, Chap. III), and Solo and King (101, Chap. 5)]. Several variations of the resulting algorithms exist, most notably the least mean squares (LMS) and recursive least squares (RLS) algorithms. The essential idea in such algorithms is that the data are processed sequentially as they arrive according to a formula that weights each new data point in light of the assumed linear relationship between the input and output on the one hand and the data that have already arrived on the other. This contrasts with traditional linear regression methods where all the data are assumed to be available at the outset and the processing is done in batch form. Given the dynamic nature of problems in signal processing, fault detection, adaptive control, and certain other areas, this recursive processing has obvious advantages. Further, it can be shown that the recursive algorithms yield an estimate at time t that is close to that which would result from batch processing if all the data up to time t were available at the outset (when t is reasonably large).

Another popular area for application of RMSA is in simulation-based optimization, especially in the context of discrete-event dynamic systems such as queuing networks. Here the goal may be to optimize some design aspect of a system (e.g., the position of certain machines on a factory floor) by running experiments via computer simulations. A systematic approach to simulation-based optimization has been adapted under the rubric of *perturbation analysis,* which was introduced in modern form by Y. C. Ho and his colleagues in the late 1970s. In this method, one aims to get a gradient estimate $Y(\theta)$ at any θ value based on only one or a small number of simulation runs. Given the stochastic nature of the simulation, this gradient estimate will also only be a stochastic estimate of the true gradient $\partial L/\partial\theta$. Some references on simulation-based optimization in the context of RMSA are Refs. 46, 47, 8, 48, and 49.

The final application we mention here is in image restoration. Here the task is to recover a true image of a scene from a recorded image of the scene where the recorded image is typically corrupted by noise and/or otherwise degraded from the original image. The essential reason for using RMSA rather than more conventional signal processing and deconvolution techniques is the ability to adapt to nonlinearities in the process. Some recent references that discuss implementations of RMSA in the context of image restoration are Refs. 50 and 51. Abreu et al. (50) are concerned with removing impulse noise that may be due, for example, to noisy sensor or transmission errors in collecting the image data. They note that nonlinear methods have often proven superior to linear methods for this problem. Cha and Kassam (51) describe an approach to image restoration based on a type of NN called the radial basis function network (e.g., Ref. 6, Sec. 9.5), where RMSA is used to train the weights of this form of function approximator. Their use of the radial basis function network is as a spatially invariant filter taking measurement data as input and producing estimates of the pixel-by-pixel gray levels in the image.

STOCHASTIC APPROXIMATION WITH GRADIENT APPROXIMATIONS BASED ON FUNCTION MEASUREMENTS

Introduction and Contrast of Gradient-Based and Nongradient Algorithms

There has been a growing interest in stochastic optimization algorithms that do not depend on direct gradient information or measurements. Rather, these algorithms are based on an *approximation* to the gradient formed from (generally noisy) measurements of the loss function. This interest has been motivated, for example, by problems in the adaptive control and statistical identification of complex systems, the optimization of processes by large Monte Carlo simulations, the training of recurrent neural networks, the recovery of images from noisy sensor data, and the design of complex queuing and discrete-event systems.

Overall, such algorithms exhibit certain convergence properties of the Robbins–Monro gradient-based algorithms considered above while requiring only loss-function measurements. A main advantage of such algorithms is that they do not require the detailed knowledge of the functional relationship between the parameters being adjusted (optimized) and the loss function being minimized that is required in gradient-based algorithms [in particular, they do not need the $Y(\theta)$ term in the RM recursion of Eq. (3)]. Such a relationship can be notoriously difficult to develop in some areas (e.g., nonlinear feedback controller design or system optimization via large-scale simulation), while in other areas (such as high-dimensional statistical parameter estimation) there may be large computational saving in calculating a loss function rather than a gradient. In contrast, the approaches based on gradient approximations require only conversion of the basic output measurements to sample values of the loss function, which does not require full knowledge of the system input–output relationships. Examples of approximation-based methods using loss-function measurements only are given below; such methods include, as an early prototype, the Kiefer–Wolfowitz finite-difference SA algorithm (52).

Because of the fundamentally different information needed in implementing these gradient-based (RM) and gradient-free

algorithms, it is difficult to construct meaningful methods of comparison. As a general rule, however, the gradient-based algorithms will converge faster than those using loss-function-based gradient approximations *when speed is measured in number of iterations*. Intuitively, this is not surprising given the additional information required for the gradient-based algorithms. In particular, based on asymptotic theory, the optimal rate of convergence measured in terms of the deviation of the parameter estimate from the true optimal parameter vector is of order $k^{-1/2}$ for the gradient-based algorithms (see the subsection on asymptotic normality above) and of order $k^{-1/3}$ for the algorithms based on gradient approximations, where k represents the number of iterations (52). (Exceptions to this maximum rate of convergence for the nongradient algorithms are discussed in Refs. 53–55 and Ref. 56, where special cases are presented that achieve a rate arbitrarily close to, or equal to, $k^{-1/2}$.)

In practice, of course, many other factors must be considered in determining which algorithm is most appropriate for a given circumstance. Three examples of why this is true are: (1) in cases where it is not possible to obtain reliable knowledge of the system input–output relationships, the gradient-based algorithms may be either infeasible (if no system model is available) or undependable (if a poor system model is used); (2) the total cost to achieve effective convergence depends not only on the number of iterations required, but also on the cost needed per iteration, which is typically greater in gradient-based algorithms (this cost may include greater computational burden, additional human effort required for determining and coding gradients, experimental costs for model building such as labor, materials, and fuel, etc.); and (3) the rates of convergence are based on asymptotic theory, and may not be representative of practical convergence rates in finite samples. For these reasons, one cannot say in general that a gradient-based search algorithm is superior to a gradient-approximation-based algorithm even though the gradient-based algorithm has a faster asymptotic rate of convergence (and with simulation-based optimization such as perturbation analysis, as discussed above, requires only one system run per iteration, while the approximation-based algorithm may require multiple system runs per iteration). As a general rule, however, if direct gradient information is *conveniently and reliably* available, it is generally to one's advantage to use this information in the optimization process. The focus in this section is on the case where such information is not readily available.

Spall (57) presents a summary of historical contributions in the area of optimization with algorithms based on gradient approximations using only loss-function measurements. The summary below is a much condensed version of that review.

Background

As in the introduction consider the problem of minimizing a (scalar) differentiable loss function $L(\theta)$, where $\theta \in R^p$, $p \geq 1$, and where the optimization problem can be translated into finding the minimizing θ^* such that the gradient $g(\theta) = 0$. It is assumed that measurements of $L(\theta)$ are available at various values of θ (actually, most of the algorithms have the weaker requirement of only requiring measurements of the *difference* of two values of the loss function, as opposed to measuring the loss functions themselves). These measurements may or may not include added noise. No direct measurements of $g(\theta)$ are assumed available, in contrast to the RM framework.

The recursive procedure we consider is in the general SA form:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k(\hat{\theta}_k) \tag{6}$$

where $\hat{g}_k(\hat{\theta}_k)$ is the estimate of the gradient $\partial L/\partial\theta$ at the iterate $\hat{\theta}_k$ based on the above-mentioned measurements of the loss function. Under appropriate conditions, the iteration in Eq. (6) will converge to θ^* in some stochastic sense, usually a.s. (see, e.g., Ref. 53 or 15). Typical convergence conditions are similar to those mentioned above for the RMSA algorithm.

The essential part of Eq. (6) is the gradient approximation $\hat{g}_k(\hat{\theta}_k)$. We discuss below two forms that have attracted attention. We let $y(\cdot)$ denote a measurement of $L(\cdot)$ at a design level represented by the dot [i.e., $y(\cdot) = L(\cdot) + noise$], and c_k be some (usually small) positive number [if the noise has mean 0, then $y(\cdot) = Q(\cdot)$ as defined in the section on RMSA]. One-sided gradient approximations involve measurements $y(\hat{\theta}_k)$ and $y(\hat{\theta}_k + perturbation)$ while two-sided gradient approximations involve measurements of the form $y(\hat{\theta}_k \pm perturbation)$. The two general forms of gradient approximations are:

Finite Difference (FD) (52,58). Each component of $\hat{\theta}_k$ is perturbed one at a time, and corresponding measurements $y(\cdot)$ are obtained; each component of the gradient estimate is formed by differencing the corresponding $y(\cdot)$ values and then dividing by a difference interval. This is the standard approach to approximating gradient vectors and is motivated directly from the definition of a gradient as a vector of p partial derivatives, each constructed as the limit of the ratio of a change in the function value over a corresponding change in one component of the argument vector. Typically, the ith component of $\hat{g}_k(\hat{\theta}_k)$ ($i = 1, 2, \ldots, p$) for a two-sided FD approximation is given by

$$\hat{g}_{ki}(\hat{\theta}_k) = \frac{y(\hat{\theta}_k + c_k e_i) - y(\hat{\theta}_k - c_k e_i)}{2c_k}$$

where e_i denotes a vector with a one in the ith place and zeros elsewhere (an obvious analog holds for the one-sided version; likewise for the SP form below).

Simultaneous Perturbation (SP) (59,60). All elements of $\hat{\theta}_k$ are randomly perturbed together to obtain two measurements $y(\cdot)$, but each component of $\hat{g}_k(\hat{\theta}_k)$ is formed from a ratio involving the individual components in the perturbation vector and the difference in the two corresponding measurements. For two-sided SP, we have

$$\hat{g}_{ki}(\hat{\theta}_k) = \frac{y(\hat{\theta}_k + c_k \Delta_k) - y(\hat{\theta}_k - c_k \Delta_k)}{2c_k \Delta_{ki}}$$

where the distribution of the user-specified random perturbations for SP, $\Delta_k = (\Delta_{k1}, \Delta_{k2}, \ldots, \Delta_{kp})^T$, satisfies conditions mentioned in the next section.

The algorithm [Eq. (6)] with one of the gradient approximations will be referred to as FDSA or SPSA, as appropriate. [An approach in the same spirit as SPSA, called random di-

rections SA, is discussed in Refs. 61, 15, and 62, but it is shown in Ref. 63 that SPSA will generally have a lower asymptotic mean squared error than RDSA for the same number of measurements $y(\cdot)$.] Note that the number of loss-function measurements $y(\cdot)$ needed in each iteration of FDSA grows with p, while with SPSA only two measurements are needed, independent of p. This, of course, provides the *potential* for SPSA to achieve a large saving (over FDSA) in the total number of measurements required to estimate θ when p is large. This potential is only realized if the number of iterations required for effective convergence to θ^* does not increase in such a way as to cancel the measurement savings per gradient approximation at each iteration. The next section will discuss this efficiency issue further, demonstrating when this potential can be realized.

SIMULTANEOUS PERTURBATION STOCHASTIC APPROXIMATION

Introduction

The preceding section motivated the interest in techniques for recursive optimization that rely on measurements of the loss function only, not on measurements (or direct calculations) of the gradient (or higher-order derivatives) of the loss function. The focus was on two such techniques in the stochastic approximation setting: finite-difference (Kiefer–Wolfowitz) SA, and simultaneous perturbation SA. This chapter will focus on SPSA for reasons of its relative efficiency.

Recent applications of SPSA are described in Refs. 64 and 65 (queuing systems), 66 (industrial quality improvement), 67 (pattern recognition), 68 (neural network training), 69 and 70 (adaptive control of dynamic systems), 71 (statistical model parameter estimation and fault detection), 72 (sensor placement and configuration), and 73 (vehicle traffic management).

As discussed in the preceding section, SPSA is based on a highly efficient and easily implemented *simultaneous perturbation* approximation to the gradient: this gradient approximation uses only two loss-function measurements, independent of the number of parameters being optimized. This contrasts, for example, with the standard (two-sided) finite-difference approximation associated with the well-known Kiefer–Wolfowitz SA algorithm, which uses $2p$ function measurements to approximate the gradient. The fundamental (and perhaps surprising) theoretical result in Ref. 60 (Sec. 4) is:

> Under reasonably general conditions, SPSA and Kiefer–Wolfowitz finite-difference-based SA (FDSA) achieve the same level of statistical accuracy for a given number of iterations even though SPSA uses p times fewer function evaluations than FDSA (since each gradient approximation uses only $1/p$ the number of function evaluations).

This theoretical result has been confirmed in many numerical studies, even in cases where p is on the order of several hundred or thousand. The subsection below discusses this result further. Of course, this result will not always hold in finite-sample practice, because it is derived from asymptotic theory; further, the asymptotic theory is based on conditions that may be violated in some practical applications (the general conditions are similar to those for the RMSA algorithm).

The next subsection summarizes the problem setting and discusses some of the theory associated with the convergence and efficiency of SPSA. The subsection after that summarizes a reference dealing with guidelines for practical implementation of the algorithm. The subsection after that discusses some extensions to the basic SPSA algorithm, including some relatively recent results on a second-order version of SPSA that emulates the Newton–Raphson algorithm of deterministic optimization while still only requiring loss-function measurements (i.e., no gradient or Hessian information). Numerical studies of SPSA are available in many of the references (see, e.g., Ref. 60, 66, 64, 69, 70, or 63).

Basic Assumptions and Supporting Theory

Once again, the goal is to minimize a loss function $L(\theta)$ over $\theta \in C \subseteq R^p$. The SPSA algorithm works by iterating from an initial guess of the optimal θ, where the iteration process depends on the above-mentioned simultaneous perturbation approximation to the gradient $g(\theta)$. The form of the SPSA gradient approximation was presented above.

Spall (74,60) presents sufficient conditions for convergence of the SPSA iterate ($\hat{\theta}_k \to \theta^*$ a.s.) using the differential equation approach discussed in the RMSA section in the context of the RM algorithm. Because of the different form of the input, the conditions here are somewhat different from those of the RM approach. In particular, we must impose conditions on *both* gain sequences (a_k and c_k), the user-specified distribution of Δ_k, and the statistical relationship of Δ_k to the measurements $y(\cdot)$. We will not repeat the conditions here, since they are available in Refs. 74 and 60 (with later extensions in Refs. 75–77). The main conditions are that a_k and c_k both go to 0 at rates neither too fast nor too slow, that $L(\theta)$ is sufficiently smooth (several times differentiable) near θ^*, and that the $\{\Delta_{ki}\}$ are independent and symmetrically distributed about 0 with finite inverse moments $E(|\Delta_{ki}|^{-1})$ for all k, i. One particular distribution for Δ_{ki} that satisfies these latter conditions is the symmetric Bernoulli ± 1 distribution; two common distributions that do *not* satisfy the conditions (in particular, the critical finite-inverse-moment condition) are the uniform and the normal.

Although the convergence result for SPSA is of some independent interest, the most interesting theoretical results in Ref. 60, and those that best justify the use of SPSA, are the asymptotic efficiency conclusions that follow from an asymptotic normality result. In particular, under some minor additional conditions in Ref. 60 (Proposition 2), it can be shown that

$$k^{\beta/2}(\hat{\theta}_k - \theta^*) \xrightarrow{\text{dist}} N(\mu, \Sigma) \qquad \text{as} \quad k \to \infty \tag{7}$$

where $\beta > 0$ depends on the choice of gain sequences (a_k and c_k), μ depends on both the Hessian and the third derivatives of $L(\theta)$ at θ^*, and Σ depends on the Hessian matrix at θ^* (note that in general $\mu \neq 0$, in contrast to many well-known asymptotic normality results in estimation). Given the restrictions on the gain sequences to ensure convergence and asymptotic normality, the fastest allowable value for the rate of convergence of $\hat{\theta}_k$ to θ^* is $k^{-1/3}$. This contrasts with the fastest allowable rate of $k^{-1/2}$ for the RMSA algorithm. Hence, one measure of the value of the gradient information in RM is the increase in rate of convergence. There are exceptions to this result, and cases arise where the SPSA rate can be made either arbitrarily close to the RM rate of $k^{-1/2}$ (following the logic of Ref. 53) or the same as the RM rate (e.g., the use in simulation-

based optimization with "common random numbers"—see Ref. 56).

Spall (60, Sec. 4) uses the asymptotic normality result in Eq. (7) (together with a parallel result for FDSA) to establish the relative efficiency of SPSA. This efficiency depends on the shape of $L(\theta)$, the values for $\{a_k\}$ and $\{c_k\}$, and the distributions of the $\{\Delta_{ki}\}$ and measurement noise terms. There is no single expression that can be used to characterize the relative efficiency; however, as discussed in Refs. 60 (Sec. 4) and 63, in most practical problems SPSA will be asymptotically more efficient than FDSA. For example, if a_k and c_k are chosen as in the guidelines mentioned in the subsection below, then by equating the asymptotic mean squared errors $E(\|\hat{\theta}_k - \theta^*\|^2)$ in SPSA and FDSA, we find

$$\frac{\text{no. of measurements of } L(\theta) \text{ in SPSA}}{\text{no. of measurements of } L(\theta) \text{ in FDSA}} \to \frac{1}{p} \tag{8}$$

as the number of loss measurements in both procedures gets large. This result implies that the p-fold saving per iteration (gradient approximation) translates directly into a p-fold saving in the overall optimization process.

Figure 2 is an illustration of the relative efficiency of SPSA and FDSA in an adaptive control problem related to wastewater treatment. The specific scenario is close to that described in Ref. 70. The plot is showing the mean deviation (in a root-mean square, RMS sense) of the output of a treatment plant from some specified target values for water cleanliness and methane gas byproduct, the goal, of course, being to minimize the RMS error. We see that on an *iteration-by-iteration* basis, the SPSA and FDSA approaches yield similar RMS values. However, the cost per iteration in FDSA [measured in number of $L(\theta)$ measurements] is 412 ($= p$) times that of SPSA. This leads to the very large overall cost savings shown in the upper right box of the figure. In fact, *one* iteration of FDSA takes over five times the number of loss measurements than are taken in all 80 iterations of SPSA. The performance of the two algorithms in Fig. 2 is very consistent with the theoretical result in Eq. (8) above.

Implementation of Simultaneous Perturbation Stochastic Approximation

Reference 78 includes a step-by-step summary of the implementation of SPSA together with a listing of MATLAB® code (the standard algorithm can be coded in 12 or fewer lines). In addition, this reference provides some practical guidelines for choosing the gain coefficients a_k and c_k. The values recommended in these guidelines differ from the asymptotically theoretical optimal values in a reflection of the realities of finite-samples analysis (e.g., it is recommended in practice that the gains a_k and c_k decay at rates *slower* than the asymptotically optimal $1/k$ and $1/k^{1/6}$, respectively). Furthermore, theoretical guidelines, such as discussed in Fabian (53) and Chin (63), are not generally useful in practical applications, since they require the very information on the loss function and its gradients that is assumed unavailable.

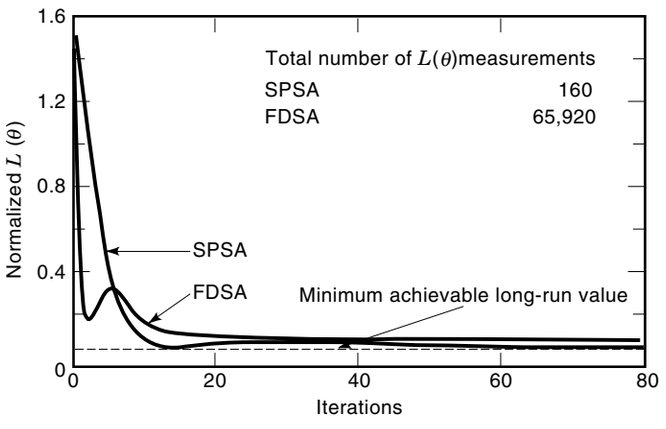

Figure 2. Relative performance of SPSA and FDSA in wastewater treatment system.

Further Results and Extensions to the Basic Algorithm

Sadegh and Spall (79) consider the problem of choosing the best distribution for the vector Δ_k. Based on asymptotic distribution results, it is shown that the optimal distribution for the components of Δ_k is symmetric Bernoulli. This simple distribution has also proven effective in many finite-sample practical and simulation examples.

Some extensions to the basic SPSA algorithm above are reported in the literature. For example, its use in feedback control problems, where the loss function changes with time, is given in Refs. 69, 70. The former reference also reports on a gradient smoothing idea (analogous to "momentum" in the neural network literature) that may help reduce noise effects and enhance convergence (and gives guidelines for how the smoothing should be reduced over time to ensure convergence). Alternatively, it is possible to average several SP gradient approximations at each iteration to reduce noise effects (at the cost of additional function measurements); this is discussed in Ref. 60. An implementation of SPSA for *global* minimization is discussed in Ref. 80 [for the case where there are multiple minimums at which $g(\theta) = 0$]; this approach is based on a stepwise (slowly decaying) sequence c_k (and possibly a_k). The problem of constrained (equality and inequality) optimization with SPSA is considered in Refs. 81 and 65 using a projection approach. A one-measurement form of the SP gradient approximation is considered in Ref. 82; although it is shown in that reference that the standard two-measurement form will usually be more efficient (in terms of the total number of loss-function measurements to obtain a given level of accuracy in the θ iterate), there are advantages to the one-measurement form in real-time operations where the underlying system dynamics may change too rapidly to get a credible gradient estimate with two successive measurements (and, analogously to the perturbation analysis and related approaches in simulation-based optimization for discrete-event systems, this would allow for a one-run gradient estimate, but *without* requiring the gradient information needed in the IPA-type approaches).

An accelerated form of SPSA is reported in Refs. 83 and 102. This approach extends the SPSA algorithm to include second-order (Hessian) effects with the aim of accelerating convergence in a stochastic analog to the deterministic Newton–Raphson algorithm. Like the standard (first-order) SPSA algorithm, this second-order algorithm is simple to implement and requires only a small number—independent of p—of loss-function measurements per iteration (no gradient measurements, as in standard SPSA). In particular, only four measurements are required to estimate the loss-function gradient and inverse Hessian at each iteration (though one additional

measurement is recommended as a check on algorithm behavior). The algorithm is implemented with two simple parallel recursions: one for θ and one for the Hessian of $L(\theta)$. The recursion for θ is a stochastic analog of the well-known Newton–Raphson algorithm of deterministic optimization. The recursion for the Hessian matrix is simply a recursive calculation of the sample mean of per-iteration Hessian estimates that are formed using SP-type ideas.

SIMULATED ANNEALING

Introduction and Motivation from the Physics of Cooling

This section continues in the spirit of the preceding two in considering algorithms that do not require direct gradient information. One of the algorithms that has attracted considerable attention is simulated annealing (SAN). SAN was originally developed for discrete optimization problems, but more recently has found application in continuous optimization problems of the type emphasized in the previous sections.

One of the main virtues of SAN is that, in principle, this algorithm addresses the difficult global optimization problem discussed in the introduction. The algorithm is designed to traverse local minima en route to a global minimum to $L(\theta)$. Further, since the method can address both discrete and continuous optimization problems, there is no need to assume the existence of a loss-function gradient (much less compute it).

The term "annealing" comes from analogies to the cooling of a liquid or solid. A central issue in statistical mechanics is analyzing the behavior of substances as they cool. At high temperatures, molecules have much mobility, but as the temperature decreases this mobility is lost and the molecules *may* tend to line themselves in a crystalline structure. This structure is the minimum-energy state for the system. Note the qualifier "may": temperature alone does not govern whether the substance has reached a minimum-energy state. To achieve this state, the cooling must occur at a sufficiently slow rate. If the substance is cooled at too rapid a rate, an amorphous (or polycrystalline) state may be reached that is not a minimum-energy state of the substance. The principle behind annealing in physical systems is the *slow* cooling of substances to reach the minimum-energy state.

In optimization, the analogy to a minimum-energy state for a system is a minimized value of the loss function. The technique of SAN attempts to capture mathematically the process of controlled cooling associated with physical processes, the aim being to reach the lowest value of the loss function in the face of possible local minima. As with the physical cooling process, whereby temporary higher-energy states may be reached as the molecules go through their alignment process, SAN also allows temporary increases in the loss function as the learning process captures the information necessary to reach the global minimum. A more thorough explanation of the analogy between SAN and physical cooling is given in, e.g., Ref. 84.

It is clear that the critical component of an SAN algorithm is the mathematical analog of the rate of cooling in physical processes. As with all other stochastic optimization algorithms, the choice of this algorithm- and problem-specific cooling schedule (analogous to the gains in stochastic approximation) has a strong effect on the success or failure of SAN.

A primary distinction between SAN and the majority of other optimization approaches (including those discussed in earlier sections) is the willingness to give up the quick gain of a rapid decrease in the loss function by allowing the possibility of temporarily increasing it. SAN derives this property from the Boltzmann (or Gibbs) probability distribution of statistical mechanics, describing the probability of system having a particular energy state:

$$P(\text{energy} = x) = c_{\mathrm{T}} \exp\left(-\frac{x}{c_{\mathrm{b}} T}\right) \tag{9}$$

where $c_{\mathrm{T}} > 0$ is a normalizing constant, $c_{\mathrm{b}} > 0$ is known as the Boltzmann constant, and T is the temperature of the system. Note that at high temperatures, the system is more likely to be in a high-energy state than at low temperatures. The optimization analogy derives from the fact that even at a low temperature (equivalent to the optimization algorithm having run for some time with a decreasing temperature), there is some nonzero probability of reaching a higher energy state (i.e., higher level of the loss function). So the SAN process sometimes goes uphill, but the probability of this decreases as the temperature is lowered. Hence, there is the possibility of getting out of a local minimum in favor of finding a global minimum, and this possibility is especially prominent in the early iterations when the temperature is high.

It was Metropolis et al. (85) who first introduced the idea of the Boltzmann–Gibbs distribution into numerical analysis through constructing a means for simulation of a system at some fixed temperature. In particular, if a system is in some current energy state E_{c}, and some system aspects are changed to make the system potentially achieve a new energy state E_{n}, then the Metropolis simulation always has the system go to the new state if $E_{\mathrm{new}} < E_{\mathrm{curr}}$. On the other hand, if $E_{\mathrm{new}} \geq E_{\mathrm{curr}}$, then the probability of the system actually going to the new state is

$$\exp\left(-\frac{E_{\mathrm{new}} - E_{\mathrm{curr}}}{c_{\mathrm{b}} T}\right) \tag{10}$$

This expression is known as the Metropolis criterion. After a large number of such decisions and outcomes, the system eventually reaches an equilibrium where the system state is governed by the Boltzmann–Gibbs distribution in Eq. (9). This is predicated on the system being at the fixed temperature T. Kirkpatrick et al. (84) were apparently the first to use this criterion for optimization together with the idea of a *changing* temperature that decays according to an annealing schedule (exponentially in their case). Geman and Geman (86) go beyond the informal annealing guidelines of Kirkpatrick et al. (84) and establish conditions on the annealing schedule for formal convergence of their form of the SAN algorithm iterate to the global minimum.

Algorithm Form

Using the principles discussed in the preceding subsection, we now present the general SAN algorithm form. There are variations depending on how one wants to implement the annealing schedule and how one performs the sampling required for generating a new candidate point. As before, we consider the minimization of some loss function $L(\theta)$, $\theta \in C \subseteq R^p$. Since the user has control over T, one can (without loss of generality) take $c_b = 1$ in the Metropolis criterion (10). Below are listed the general sequence of steps in SAN when noise-free measurements of L are available:

Step 1. Choose an initial temperature T and set of current parameter values θ_{curr}; determine $L(\theta_{\text{curr}})$.

Step 2. Randomly determine a new value of θ, θ_{new}, that is "close" to the current value, and determine $L(\theta_{\text{new}})$.

Step 3. Compare the two L values via the Metropolis criterion (10): Let $\delta = L(\theta_{\text{new}}) - L(\theta_{\text{curr}})$. Accept θ_{new} if $\delta < 0$. Alternatively, if $\delta \geq 0$, accept the new point θ_{new} only if a uniform (0, 1) random variable U (generated by Monte Carlo) satisfies $U \leq \exp(-\delta/T)$.

Step 4. Repeat steps 2 and 3 for some period until either the *budget* of function evaluations allocated for that T has been used or the system reaches some state of equilibrium.

Step 5. Lower T according to the annealing schedule, and return to step 2. Continue the process until the total budget for function evaluations has been used or some indication of convergence is satisfied (analogous to the system being "frozen" in its minimum-energy state).

The specifics of implementation for the five steps above can vary greatly. In the annealing schedule, Kirkpatrick et al. (84), Press et al. (87 p. 452), and Brooks and Morgan (88) discuss the case where T decays geometrically in the number of cooling phases (number of times T is lowered according to step 5). Geman and Geman (86) present conditions such that if T decreases at the rate $1/\log k$ (where k is the number of algorithm iterations) then the probability distribution for the iteration converges to a uniform distribution over all possible global minimum points. Szu and Hartley (89) present some formal arguments justifying a faster rate of decay for T: namely, having T decay at a rate $1/k$. Another area for different implementations is in step 2, where a new θ value is generated randomly. Probably the most common form of perturbation for continuous optimization is to add a p-dimensional Gaussian random variable to the current value θ_{curr} (e.g., Ref. 6, p. 183). Alternative perturbations include the approach based on changing only one component of θ at a time (88), the approach using spherically uniform perturbations (90), and the approach using a multivariate Cauchy distribution to change all components (89,62).

Aside from the general variations in implementation above, SAN is critically dependent on the specific values for various algorithm parameters and decision criteria. In particular, these include the initial temperature T, the specific distribution parameters chosen for the perturbation distribution (e.g., the covariance matrix for a Gaussian perturbation), the specific parameter(s) associated with the decay of T (e.g., the λ in $T_{\text{new}} = \lambda T_{\text{old}}$, where T_{new} and T_{old} are the new and old temperatures if one is using a geometric decay), and the criterion for determining when to lower the temperature (e.g., the maximum allowable number of function evaluations before a lowering of T).

There appears to be no fully satisfactory way to accommodate noisy function measurements in SAN, although some work on this aspect of the *discrete* (combinatorial) optimization problem has been carried out in Ref. 91. (One of the particularly appealing features of the stochastic approximation methods in the preceding three sections is that they handle noisy function measurements in an essentially seamless manner. Namely, the forms of the algorithms do not have to be altered, the convergence theory applies in the noisy case, and there is a *gradual* degradation in performance as the noise level increases from 0, in contrast with the sudden large degradation in approaches such as SAN that use explicit decision criteria based on loss-function measurements). The primary difficulty arises in the critical decision step 3, where θ_{curr} is changed or not changed according to the value of $\delta = L(\theta_{\text{new}}) - L(\theta_{\text{curr}})$ together with the random sampling associated with the Metropolis criterion. With noisy function measurements, the value of δ, now equal to $y(\theta_{\text{new}}) - y(\theta_{\text{curr}})$ instead of $L(\theta_{\text{new}}) - L(\theta_{\text{curr}})$, will be altered from its underlying true value according to noise-free function measurements. In fact, even a modest level of noise will frequently alter the sign of δ, which is likely to change the decision regarding the acceptance or rejection of the new point θ_{new}. The obvious means by which one can attempt to cope with noise is to average several function measurements $y(\cdot)$ at each of the values θ_{curr} and θ_{new} when performing the comparison at step 3. However, this may dramatically increase the cost of optimization (specifically, the total number of function evaluations required), since a large amount of averaging will often be required to effectively remove the noise, especially in the region around a local or global minimum where the function may be relatively flat. An alternative way is to alter the acceptance criterion to accept the inherent errors that will be made with the noisy measurements. Hence we replace the criteria $\delta < 0$ or $\delta \geq 0$ with $\delta < \rho\sigma$ or $\delta \geq \rho\sigma$, where σ is the function measurement noise standard deviation (or, more likely, an estimate of it) and ρ the number of multiples of the standard deviation that will be tolerated. The rationale for such a change is that, while we are willing to accept a θ_{new} that temporarily increases the loss function (a basic aspect of SAN), we are less willing to forgo a θ_{new} that decreases the loss function. Changing the unconditional acceptance criterion from $\delta < 0$ to $\delta < \rho\sigma$ will allow for a greater number of cases where $L(\theta_{\text{new}}) < L(\theta_{\text{curr}})$ even though $y(\theta_{\text{new}}) \geq y(\theta_{\text{curr}})$ due to the noise. With $\rho \approx 2$, one can be sure (through the Chebyshev inequality of probability) that most such cases will be caught, although at the expense of letting some additional θ_{new} values that increase the loss function be accepted.

Evaluation of Simulated Annealing

Although many positive studies regarding SAN have been reported in the literature (e.g., Refs. 92–94), the author is unaware of any formal theoretical analysis comparing SAN with the gradient-free SA algorithms discussed in the preceding two sections. (Note that some connections of SAN with SA have been explored in Refs. 95 and 96. The basic idea there is, beginning with the basic SA recursion using either gradients or function measurements, to add a Monte Carlo-generated Gaussian noise term $b_k W_k$ to the recursion, where $b_k \to 0$ and W_k is the Gaussian random vector. This term is similar to SAN in that it will force the iterate out of local minima under some conditions that are established in the cited papers.) However, Fabian (97), Chin (103), Styblinski and Tang (62), and the author (together with several students in a graduate class he taught) have conducted some numerical comparisons. Fabian (97), Chin (103), and Styblinski and Tang (62) compare different forms of SAN against several random and deterministic search algorithms for problems with noise-free measurements of the loss function. The SAN algorithms generally compared poorly with the competing algorithms in these two papers. The author's studies involved four different loss functions: $p = 2$ and $p = 20$ fourth-order polynomials,

and the Example 1 ($p = 2$, fourth-order polynomial) and Example 6 ($p = 10$, trigonometric) functions in Ref. 62. The studies of the author considered both noise-free and noisy measurements of the four loss functions and compared SAN with simple random search mentioned in Ref. 7 (pp. 186–189) and the global version of SPSA outlined in Ref. 80. Although SAN was sometimes superior to the random search method, the global SPSA method was significantly more efficient than SAN in all cases considered, and, even more dramatically, was sometimes convergent when SAN seemed unable to obtain a solution anywhere near the true optimal (this was most apparent in the Example 6 problem of Ref. 62, which had a very large number of local minima, one of which always seemed to form a trap for SAN). In order to have a fair comparison in performing these studies, the algorithm coefficients (e.g., the gains for the SA algorithms and the decay rate and other coefficients mentioned above for SAN) were tuned to approximately optimize performance for the competing algorithms.

Since one must be careful in drawing conclusions beyond the specific cases in any numerical analysis, the above should not indicate that SAN is always a poor performer. Rather, these results should be a cautionary note in view of some of the positive results reported elsewhere. Certainly, SAN does have a role in the area of global optimization, as evidenced by its popularity and positive performance in some challenging problems. Furthermore, the above studies were for problems with "smooth" (differentiable) loss functions, and SAN (unlike most SA algorithms) has the ability to deal with nonsmooth loss functions.

CONCLUDING REMARKS

This article has surveyed several important algorithms with stochastic optimization. The focus was on algorithms within the stochastic approximation and simulated annealing classes. Broadly speaking the gradient-based (Robbins–Monro) SA algorithms are most appropriate when enough prior information is available about the system so that it is possible to determine the gradient of the observed loss function. It is expected that when such information is available, this type of algorithm may be the most efficient means for optimization within the stochastic setting, especially when coupled with one of the algorithm acceleration techniques such as iterate averaging or second-order methods. However, it was also pointed out that in many applications, it is difficult or impossible to obtain the gradient information needed for the application of the gradient-based methods. Then one is faced with carrying out the optimization using only the (possibly noisy) measurements of the loss function itself. In that context, we emphasized three different approaches: finite-difference and simultaneous perturbation SA (both of which are based on approximating the unknown gradient vector) and simulated annealing. The FDSA method is the oldest and best-known of these approaches, and can work in a variety of applications. The newer SPSA approach will, however, usually be more efficient than FDSA, with the gain in efficiency being approximately equal to the dimension of the θ vector being optimized (i.e., the SPSA algorithm will use only one-pth the number of function evaluations to obtain the same level of statistical accuracy as FDSA). Unlike the SA algorithms, simulated annealing is focused on global (as opposed to local) optimization problems. Simulated annealing is based on an intriguing analogy to the cooling of materials and the achievement of an optimal state for the material by cooling neither too fast nor too slow. While some positive experience has been reported with optimization by simulated annealing, it appears that there exist more efficient algorithms for many problems.

Although stochastic optimization has the potential for treating a broader class of problems than many traditional deterministic techniques, their application may sometimes be a challenge. A problem common to *all* stochastic optimization techniques (including those not discussed in this article, such as the genetic algorithm, sequential random search, and evolutionary programming) is that values must be specified for the algorithm's tunable coefficients. All stochastic optimization techniques have such coefficients (the gains in SA, terms associated with the cooling schedule and probability of accepting a step in simulated annealing, etc.). These coefficient values are typically problem-dependent and can have a profound effect on the performance of an algorithm.

Stochastic optimization is playing an ever larger role in optimization, as it allows for the treatment of problems such as global optimization and noisy loss-function evaluations that arise frequently in areas such as network analysis, neural network training, image processing, and nonlinear control. It is expected that the role of stochastic optimization will continue to grow as modern systems increase in complexity and as population growth and dwindling natural resources force tradeoffs that were previously unnecessary. Stochastic optimization allows for the treatment of a broader range of problems than possible with only standard deterministic methods. The algorithms of this article provided a sampling of several important stochastic optimization methods.

ACKNOWLEDGEMENTS

This work was partially funded by U.S. Navy Contract N00024-97-C-8119 and the JHU/APL Independent Research and Development Program. The author appreciates the comments of two anonymous reviewers and G. Cauwenberghs, V. Fabian, M. Fu, J. L. Maryak, B. T. Polyak, M. A. Styblinski, and S. Yakowitz on an earlier version of parts of this article.

BIBLIOGRAPHY

1. G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization,* New York: Wiley, 1988.
2. K. S. Tang et al., Genetic algorithms and their applications, *IEEE Sig. Process. Mag.,* November 1996, pp. 22–37.
3. T. Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, and Genetic Algorithms,* Oxford, UK: Oxford University Press, 1996.
4. F. J. Solis and J. B. Wets, Minimization by random search techniques, *Math. Oper. Res.,* **6**: 19–30.
5. A. A. Zhiglavsky, *Theory of Global Random Search,* Boston: Kluwer Academic, 1991.
6. J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-fuzzy and Soft Computing,* Upper Saddle River, NJ: Prentice-Hall, 1997.
7. W. H. Fleming, *Functions of Several Variables,* New York: Springer Verlag, 1977.
8. P. Glasserman, *Gradient Estimation via Perturbation Analysis,* Boston: Kluwer, 1991.

9. H. White, Some asymptotic results for learning in single hidden layer feedforward neural networks, *J. Amer. Statist. Assoc.,* **84**: 1003–1013, 1989.
10. H. Robbins and S. Monro, A stochastic approximation method, *Ann. Math. Statist.,* **22**: 400–407, 1951.
11. J. R. Blum, Approximation methods which converge with probability one, *Ann. Math. Statist.,* **25**: 382–386, 1954.
12. D. Ruppert, Stochastic approximation, in B. K. Ghosh and P. K. Sen (eds.), *Handbook of Sequential Analysis,* New York: Marcel Dekker, 1991, pp. 503–529.
13. J. S. Rustagi, *Optimization Techniques in Statistics,* New York: Academic Press, 1994.
14. L. Ljung, Analysis of recursive stochastic algorithms, *IEEE Trans. Autom. Control,* **AC-22**: 551–575, 1977.
15. H. J. Kushner and D. S. Clark, *Stochastic Approximation Methods for Constrained and Unconstrained Systems,* New York: Springer Verlag, 1978.
16. A. Benveniste, M. Metivier, and P. Priouret, *Adaptive Algorithms and Stochastic Approximations,* Berlin: Springer Verlag, 1990.
17. T. L. Lai, Stochastic approximation and sequential search for optimum, in L. M. Le Cam and R. A. Olshen (eds.), *Proc. Berkeley Conf. Honor of Jerzy Neyman and Jack Kiefer,* Vol. II, Belmont, CA: Wadsworth, 1985.
18. V. Fabian, On asymptotic normality in stochastic approximation, *Ann. Math. Statist.,* **39**: 1327–1332, 1968.
19. J. Sacks, Asymptotic distribution of stochastic approximation procedures, *Ann. Math. Statist.,* **29**: 373–405, 1958.
20. R. G. Laha and V. K. Rohatgi, *Probability Theory,* New York: Wiley, 1979.
21. L. Ljung, G. Pflug, and H. Walk, *Stochastic Approximation and Optimization of Random Systems,* Basel: Birkhäuser, 1992.
22. A. Okamura, T. Kirimot, and M. Kondo, A new normalized stochastic approximation algorithm using a time-shift parameter, *Electron. Commun. Jpn.,* Part 3, **78**: 41–51, 1995.
23. C. M. Kuan and K. Hornik, Convergence of learning algorithms with constant learing rates, *IEEE Trans. Neural Net.,* **2**: 484–489, 1991.
24. O. Macchi and E. Ededa, Second-order convergence analysis of stochastic adaptive linear filtering, *IEEE Trans. Autom. Control,* **AC-28**: 76–85, 1982.
25. H. J. Kushner and H. Huang, Asymptotic properties of stochastic approximations with constant coefficients, *SIAM J. Control Optim.,* **19**: 87–105, 1983.
26. G. C. Pflug, Stochastic minimization with constant step size: Asymptotic laws, *SIAM J. Control Optim.,* **24**: 655–666, 1986.
27. G. G. Yin, Asymptotic optimal rate of convergence for an adaptive estimation procedure, in T. E. Duncan and B. Pasik-Duncan (eds.), *Stochastic Theory and Adaptive Control,* New York: Springer Verlag, 1992, pp. 480–489.
28. G. G. Yin and K. Yin, Passive stochastic approximation with constant step size and window width, *IEEE Trans. Autom. Control,* **41**: 90–106, 1996.
29. S. Mukherjee and T. L. Fine, Online steepest descent yields wieghts with nonnormal limiting distributions, *Neural Comp.,* **8**: 1075–1084, 1996.
30. M. Metivier and P. Priouret, Applications of a Kushner and Clark lemma for general classes of stochastic algorithms, *IEEE Trans. Inf. Theory,* **IT-30**: 1440–1451, 1984.
31. J. E. Dennis and R. B. Schnabel, A view of unconstrained optimization, in G. L. Nemhauser et al. (eds.), *Optimization* (Handbooks in OR and MS, Vol. 1) New York: Elsevier, 1989, pp. 1–72.
32. H. Kesten, Accelerated stochastic approximation, *Ann. Math. Statist.,* **29**: 41–59, 1958.
33. B. Delyon and A. Juditsky, Accelerated stochastic approximation, *SIAM J. Control Optim.,* **3**: 868–881, 1993.
34. L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification,* Cambridge, MA: MIT Press, 1983.
35. C. Z. Wei, Multivariate adaptive stochastic approximation, *Ann. Statist.,* **15**: 1115–1130, 1987.
36. D. Ruppert, A Newton–Raphson version of the multivariate Robbins–Monro procedure, *Ann. Statist.,* **13**: 236–245, 1985.
37. B. T. Polyak and A. B. Juditsky, Acceleration of stochastic approximation by averaging, *SIAM J. Control Optim.,* **30**: 838–855, 1992.
38. H. J. Kushner and J. Yang, Stochastic approximation with averaging: Optimal rates of convergence for general processes, *SIAM J. Control Optim.,* **31**: 1045–1062, 1993.
39. H. J. Kushner and J. Yang, Stochastic approximation with averaging and feedback: Rapidly convergent on-line algorithms, *IEEE Trans. Autom. Control,* **40**: 24–34, 1995.
40. I. J. Wang, Analysis of stochastic approximation and related algorithms, Ph.D. Thesis, Purdue University School of Electrical Engineering, 1996.
41. J. L. Maryak, Some guidelines for using iterate averaging in stochastic approximation, in *Proc. IEEE Conf. Decision Control,* San Diego, CA, 2287–2290, 1997.
42. C. A. Goodsell and D. L. Hanson, Almost sure convergence for the Robbins–Monro process, *Ann. Probab.,* **4**: 890–901, 1976.
43. S. N. Evans and N. C. Weber, On the almost sure convergence of a general stochastic approximation procedure, *Bull. Austral. Math. Soc.,* **34**: 335–342, 1986.
44. B. Kosko, *Neural Networks and Fuzzy Systems,* Englewood Cliffs, NJ: Prentice-Hall, 1992.
45. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning internal representations by error propagation, in *Parallel Distributed Processing,* Vol. 1, Cambridge, MA: MIT Press, 1986, pp. 318–362.
46. R. Suri, Perturbation analysis: The state of the art and research issues explained via the Gl/G/1 queue, *Proc. IEEE,* **77**: 114–137, 1989.
47. P. L'Ecuyer and P. W. Glynn, Stochastic optimization by simulation: Convergence proofs for the Gl/G/1 queue in steady state, *Management Sci.,* **40**: 1562–1578, 1994.
48. G. C. Pflug, *Optimization of Stochastic Models: The Interface between Simulation and Optimization,* Boston: Kluwer, 1996.
49. M. Fu and J.-Q. Hu, *Conditional Monte Carlo: Gradient Estimation and Optimization Applications,* Norwell, MA: Kluwer, 1997.
50. E. Abreu et al., A new efficient approach for the removal of impulse noise from highly corrupted images, *IEEE Trans. Image Proc.,* **5**: 1012–1025, 1996.
51. I. Cha and S. A. Kassam, RBFN restoration of nonlinearity degraded images, *IEEE Trans. Image Proc.,* **5**, 964–975, 1996.
52. J. Kiefer and J. Wolfowitz, Stochastic estimation of a regression function, *Ann. Math. Statist.,* **23**: 462–466, 1952.
53. V. Fabian, Stochastic approximation, in J. J. Rustigi (ed.), *Optimizing Methods in Statistics,* New York: Academic Press, 1971, pp. 439–470.
54. P. Glasserman and D. D. Yao, Some guidelnes and guarantees for common random numbers, *Management Sci.,* **38**: 884–908.
55. P. L'Ecuyer and G. Yin, Budget-dependent convergence rate of stochastic approximation, *SIAM J. Optim.,* **8**: 217–247, 1998.
56. N. L. Kleinman, J. C. Spall, and D. Q. Naiman, Simulation-based optimization with stochastic approximation using common random numbers, *Management Sci.,* submitted.
57. J. C. Spall, Developments in stochastic optimization algorithms with gradient approximations based on function measurements, in J. D. Tew, M. S. Manivannan, D. A. Sadowski, and A. F. Seila (eds.), *Proc. Winter Simulation Conf.,* 1994, pp. 207–214.
58. J. R. Blum, Multidimensional stochastic approximation methods, *Ann. Math. Statist.,* **25**: 737–744, 1954.

59. J. C. Spall, A stochastic approximation technique for generating maximum likelihood parameter estimates, in *Proc. Amer. Control Conf.,* 1987, pp. 1161–1167.
60. J. C. Spall, Multivariate stochastic approximation using a simultaneous peturbation gradient approximation, *IEEE Trans. Automat. Control,* **37**: 332–341, 1992.
61. B. T. Polyak and Y. Z. Tsypkin, Pseudogradient adaptation and training algorithms, *Automation Remote Control,* **34**: 377–397, 1973.
62. M. A. Styblinski and T. S. Tang, Experiments in nonconvex optimization: Stochastic approximation with function smoothing and simulated annealing, *Neural Netw.,* **3**: 467–483, 1990.
63. D. C. Chin, Comparative study of stochastic algorithms for system optimization based on gradient approximations, *IEEE Trans. Syst. Man. Cybern. B,* **27**: 244–249, 1997.
64. S. D. Hill and M. C. Fu, Transfer optimization via simultaneous perturbation stochastic approximation, in C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman (eds.), *Proc. Winter Simulation Conf.,* 1995, pp. 242–249.
65. M. C. Fu and S. D. Hill, Optimization of discrete event systems via simultaneous perturbation stochastic approximation, *Trans. Inst. Ind. Eng.,* **29**: 233–243, 1997.
66. F. Rezayat, On the use of an SPSA-based model-free controller in quality improvement, *Automatica,* **31**: 913–915, 1995.
67. Y. Maeda, H. Hirano, and Y. Kanata, A learning rule of neural networks via simultaneous perturbation and its hardware implementation, *Neural Netw.,* **8**: 251–259, 1995.
68. G. Cauwenberghs, Analog VLSI autonomous systems for learning and optimization, Ph.D. Dissertation, California Institute of Technology, 1994.
69. J. C. Spall and J. A. Cristion, Nonlinear adaptive control using neural networks: Estimation based on a smoothed form of simultaneous perturbation gradient approximation, *Statist. Sinica,* **4**: 1–27, 1994.
70. J. C. Spall and J. A. Cristion, A neural network controller for systems with unmodeled dynamics with applications to wastewater treatment, *IEEE Trans. Syst. Man. Cybern. B,* **27**: 369–375, 1997.
71. A. Alessandri and T. Parisini, Nonlinear modeling and state estimation in a real power plant using neural networks and stochastic approximation, in *Proc. Amer. Control Conf.,* 1995, pp. 1561–1567.
72. P. Sadegh and J. C. Spall, Optimal Sensor Configuration for Complex Systems, in *Proc. Test Technol. Symp.* (sponsored by U.S. Army TECOM), http://www.atc.army.mil/~tecom/tts/proceed/optsenr.html, 1996.
73. D. C. Chin and R. H. Smith, A traffic simulation for mid-Manhattan with model-free adaptive signal control, in *Proc. Summer Comput. Simulation Conf.,* (sponsored by the Society for Computer Simulation) 1994, pp. 296–301.
74. J. C. Spall, A stochastic approximation algorithm for large-dimensional systems in the Kiefer–Wolfowitz setting, in *Proc. IEEE Conf. Decision Control,* 1988, pp. 1544–1548.
75. H. F. Chen, T. E. Duncan, and B. Pasik-Duncan, A stochastic approximation algorithm with random differences, in *Proc. 13th IFAC World Congr.,* Vol. H, 1996, pp. 493–496.
76. I.-J. Wang and E. K. P. Chong, A deterministic analysis of simultaneous perturbation stochastic approximation, in *Proc. 30th Conf. Inf. Sci. Syst.,* 1996, pp. 918–922.
77. J. Dippon and J. Renz, Weighted means in stochastic approximation of minima, *SIAM J. Control Optim.,* **35**, 1811–1827, 1997.
78. J. C. Spall, Implementation of the simultaneous perturbation method for stochastic optimization, *IEEE Trans. Aerosp. Electron. Syst.,* **34**: in press.
79. P. Sadegh and J. C. Spall, Optimal random perturbations for multivariate stochastic approximation using a simultaneous perturbation gradient approximation, *IEEE Trans. Autom. Control,* **43**: in press.
80. D. C. Chin, A more efficient global optimization algorithm based on Styblinski and Tang, *Neural Netw.,* **7**, 573–574, 1994.
81. P. Sadegh, Constrained optimization via stochastic approximation with a simultaneous perturbation gradient approximation, *Automatica,* **33**: 889–892, 1997.
82. J. C. Spall, A one-measurement form of simultaneous perturbation stochastic approximation, *Automatica,* **33**: 109–112, 1997.
83. J. C. Spall, Accelerated second-order stochastic optimization using only function measurements, *Proc. IEEE Conf. Decision Control,* 1997, pp. 1417–1424.
84. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Optimization by simulated annealing, *Science,* **220**: 671–680, 1983.
85. N. Metropolis et al., Equation of state calculations by fast computing machines, *J. Chem. Phys.,* **21**: 1087–1092, 1953.
86. S. Geman and D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Trans. Pattern Anal. Mach. Intell.,* **PAMI-6**: 721–741, 1984.
87. W. H. Press et al., *Numerical Recipes in C,* 2nd ed., Cambridge: Cambridge University Press, 1992.
88. S. P. Brooks and B. J. T. Morgan, Optimization using simulated annealing, *Statistician,* **44**: 241–257, 1995.
89. H. Szu and R. Hartley, Fast simulated annealing, *Phys. Lett. A,* **122**: 157–162, 1987.
90. I. O. Bohachevsky, M. E. Johnson, and M. L. Stein, Generalized simulated annealing for function optimization, *Technometrics,* **28**: 209–217, 1986.
91. S. Gelfand and S. K. Mitter, Simulated annealing with noisy or imprecise energy measurements, *J. Optim. Theor. Appl.,* **62**: 49–62, 1989.
92. P. J. M. Van Laarhoven, *Theoretical and Computational Aspects of Simulated Annealing,* CWI Tract 51, Amsterdam: Center for Mathematics and Computer Science, 1988.
93. R. Azencott (ed.), *Simulated Annealing: Parallelization Techniques,* New York: Wiley, 1992.
94. R. V. V. Vidal (ed.), *Applied Simulated Annealing,* Berlin: Springer Verlag, 1993.
95. S. Gelfand and S. K. Mitter, Simulated annealing type algorithms for multivariate optimization, *Algorithmica,* **6**: 419–436, 1991.
96. S. Gelfand and S. K. Mitter, Metropolis-type annealing algorithms for global optimization in R^d, *SIAM J. Control Optim.,* **31**: 111–131, 1993.
97. V. Fabian, Simulated annealing simulated, *Comput. Math. Appl.,* **33**: 81–94, 1997.
98. H. J. Kushner and G. G. Yin, *Stochastic Approximation Algorithms and Applications,* New York: Springer-Verlag, 1997.
99. J. Tsitsiklis and B. Van Roy, An analysis of temporal-difference learning with function approximation, *IEEE Trans. Autom. Control,* **42**: 674–690, 1997.
100. J. H. Venter, An extension of the Robbins–Monro procedure, *Ann. Math. Statist.,* **38**: 181–190, 1967.
101. V. Solo and X. Kong, *Adaptive Signal Processing Algorithms,* Englewood Cliffs, NJ: Prentice-Hall, 1995.
102. J. C. Spall, Adaptive stochastic approximation by the simultaneous perturbation method, *IEEE Trans. Autom. Control,* submitted, 1998.
103. D. C. Chin, The simultaneous perturbation method for processing magnetospheric images, *Opti. Eng.,* submitted, 1998.

JAMES C. SPALL
The Johns Hopkins University