LETTER Communicated by Tom Heskes

Learning Curves for Stochastic Gradient Descent in Linear Feedforward Networks

Justin Werfel
jkwerfel@mit.edu
Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.

Xiaohui Xie
xhxie@mit.edu
Broad Institute of Massachusetts Institute of Technology and Harvard University, Cambridge, MA 02141, U.S.A.

H. Sebastian Seung
seung@mit.edu
Howard Hughes Medical Institute, Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.

Gradient-following learning methods can encounter problems of implementation in many applications, and stochastic variants are sometimes used to overcome these difficulties. We analyze three online training methods used with a linear perceptron: direct gradient descent, node perturbation, and weight perturbation. Learning speed is defined as the rate of exponential decay in the learning curves. When the scalar parameter that controls the size of weight updates is chosen to maximize learning speed, node perturbation is slower than direct gradient descent by a factor equal to the number of output units; weight perturbation is slower still by an additional factor equal to the number of input units. Parallel perturbation allows faster learning than sequential perturbation, by a factor that does not depend on network size. We also characterize how uncertainty in quantities used in the stochastic updates affects the learning curves. This study suggests that in practice, weight perturbation may be slow for large networks, and node perturbation can have performance comparable to that of direct gradient descent when there are few output units. However, these statements depend on the specifics of the learning problem, such as the input distribution and the target function, and are not universally applicable.

1 Introduction

Learning in artificial systems can be formulated as optimization of an objective function that quantifies the system's performance. A typical approach to this optimization is to follow the gradient of the objective function with respect to the tunable parameters of the system. Frequently this is accomplished directly, by calculating the gradient explicitly and updating the parameters by a small step in the direction of locally greatest improvement.

In many circumstances, however, attempts at direct gradient following can encounter problems. In VLSI and other hardware implementations, computation of the gradient may be excessively unwieldy, if not impossible, due to unavoidable imperfections in manufacturing (Widrow & Lehr, 1990; Jabri & Flower, 1992; Flower & Jabri, 1993; Cauwenberghs, 1993, 1996). In some cases, as with many where the reinforcement learning framework is used, there may be no explicit form for the objective function and hence no way of calculating its gradient (Fiete, Fee, & Seung, 2004). And in biological systems, any argument that direct gradient calculation might be what the system is actually doing typically encounters severe obstacles. For instance, backpropagation, the standard method for training artificial neural networks (ANNs), requires two-way, multipurpose synapses, units with global knowledge about the system that are able to recognize different kinds of signals and treat them in very different ways, and (in the case of trajectory learning) the ability to run backward in time, all of which strain the bounds of biological plausibility (Widrow & Lehr, 1990; Bartlett & Baxter, 1999). For reasons such as these, there has been broad interest in stochastic methods that approximate the gradient on average.

Compared to a method that follows the true gradient directly, we might intuitively expect a stochastic gradient-following approach to learn more slowly. In this study (based on analysis of ANNs, for which the tunable parameters are the network weights), the stochastic algorithms use a reinforcement learning framework with a single reward signal, which is assigned based on the contributions of all the network weights. That single reward is all that is available to evaluate how every one of the weights should be updated, in contrast to a true gradient method where the optimal updates are all calculated exactly. If calculation of the gradient is not computationally expensive enough to represent a bottleneck, and the error landscape is sufficiently well behaved that following the gradient is typically the quickest way to decrease error, then the significant advantage that explicit gradient methods have in general in terms of the amount of information available to them for each update could be expected to allow much faster learning. Moreover, if the network is made larger and the number of weights thereby increased, the problem of spatial credit assignment becomes still more difficult; thus, we would tend to expect the performance of stochastic gradient methods to scale up with network size more poorly than that of deterministic methods. However, under some circumstances,

stochastic methods can be equally as effective as direct ones in training even large networks, generating nearly identical learning curves (see, e.g., Figure 3 below). Under what circumstances, then, will stochastic gradient descent have performance comparable to that of the deterministic variety? And how good can that performance be?

In this letter, we investigate these issues quantitatively by analytically calculating learning curves for a linear perceptron using a direct gradient method and two stochastic methods, node perturbation and weight perturbation. We find that the maximum learning speed for each algorithm scales inversely with the first power of the dimensionality of the noise injected into the system; this result is in contradiction to previous work, which reported maximum learning speed scaling inversely with the square root of the dimensionality of the injected noise (Cauwenberghs, 1993). Weight perturbation, which depends on the use of higher-dimensional noise, scales more poorly than node perturbation, which in turn scales more poorly than the noiseless direct gradient method. Further, parallel variation of the network weights in the stochastic algorithms allows learning to take place at a higher speed than does sequential variation, by a constant factor. We also consider how uncertainty in quantities used to calculate the weight updates affects learning speed and lowest mean error attainable by a stochastically trained network.

These exact results depend on the specifics of the learning model, including the linearity of the network, the distribution of inputs it receives, the target function we train it to approximate, and the objective function that quantifies its performance. Under other conditions, the results may be qualitatively different, as we discuss. (Some of the results in this letter were presented in preliminary form in Werfel, Xie, & Seung, 2004.)

2 Perceptron Comparison

Direct and stochastic gradient approaches are general classes of training methods. We study the operation of exemplars of both on a feedforward linear perceptron, which has the advantage over the nonlinear case that the learning curves can be calculated exactly (Heskes & Kappen 1991; Baldi & Hornik, 1993; Biehl & Riegler 1994; Mace & Coolen 1998). We have N input units and M output units, connected by a weight matrix w of MN elements; outputs in response to an input x are given by $y = wx$. For the ensemble of possible inputs, we want to train the network to produce desired corresponding outputs $y = d$; in order to ensure that this task is realizable by the network, we assume the existence of a teacher matrix w^* such that $d = w^*x$. For objective function, we use the squared error

$$E = \frac{1}{2}|y - d|^2 = \frac{1}{2}|(w - w^*)x|^2 = \frac{1}{2}|Wx|^2, \tag{2.1}$$

where we have defined the matrix $W \equiv w - w^*$. We train the network with an online approach, choosing at each time step an input vector x with components drawn from a gaussian distribution with mean 0 and variance γ^2, and using it to construct a weight update according to one of the three prescriptions below.

The online gradient-following approach explicitly uses the gradient of the objective function for a given input to determine the weight update,

$$\Delta W_{\mathrm{OL}} = -\eta \nabla E,$$

where $\eta > 0$ is the learning rate. This is the approach taken, for example, by backpropagation.

In the stochastic algorithms, the gradient is not calculated directly; instead, some noise is introduced into the system, affecting its error for a given input, and the difference between the error with and without noise is used to estimate the gradient. The simplest case is when noise is added directly to the weight matrix:

$$E'_{\mathrm{WP}} = \frac{1}{2} |(W + \psi)x|^2 .$$

Such an approach has been termed "weight perturbation," frequently with only one weight being varied at a time (Jabri & Flower, 1992; Cauwenberghs, 1993). We choose each element of the noise matrix ψ from a gaussian distribution with mean 0 and variance σ^2. Intuitively, if the addition of the noise lowers the error, that perturbation to the weight matrix is retained, which will mean lower error for that input in future. Conversely, if the noise leads to an increase in error, the opposite change is made to the weights; the effect of small noise on error can be approximated as linear, and the opposite change in weights will lead to the opposite change in error, again decreasing error for that input in future. These two cases can be combined into the single weight update,

$$\Delta W_{\mathrm{WP}} = -\frac{\eta}{\sigma^2}(E'_{\mathrm{WP}} - E)\psi.$$

A more subtle way to introduce stochasticity involves adding the noise to the output of each output unit rather than to every weight:

$$E'_{\mathrm{NP}} = \frac{1}{2}|Wx + \xi|^2.$$

Such an approach is sometimes called "node perturbation," though that term has traditionally referred to a serial approach where noise is added to one output unit at a time (Widrow & Lehr, 1990; Flower & Jabri, 1993). Here,

if the addition of the noise ξ leads to a decrease in error, the weights are adjusted in such a way as to move the outputs in the direction of that noise. The degree of freedom for each output unit corresponds to the adjustment of its threshold, making the unit more or less responsive to a given pattern of input activity. The elements of ξ are again chosen independently from a gaussian distribution with variance σ^2; here, ξ has M elements, whereas ψ in the previous case had MN. The REINFORCE framework (Williams, 1992) gives for the weight update

$$\Delta W_{\mathrm{NP}} = -\frac{\eta}{\sigma^2}(E'_{\mathrm{NP}} - E)\xi x^T.$$

These stochastic frameworks produce weight updates identical to that of direct gradient descent on the objective function when averaged over all values of the noise (Williams, 1992; Cauwenberghs, 1993), which is the sense in which they constitute stochastic gradient descent. This result is easy to verify in the particular forms taken by ΔW_{NP} and ΔW_{WP} here, shown below. It is worth emphasizing that not only will they give a decrease in error on average, but every update will decrease the error, so long as the noise is small (compared to Wx for node perturbation, W for weight perturbation).

2.1 Reducing the Dimensionality of the Space of Learning Constants. Three constants affect the course of learning for the system as formulated above: learning rate η, variance of input distribution γ^2, and variance of injected noise σ^2. We can simplify the problem by rewriting the expressions in the previous section. For true gradient descent, $\Delta W_{\mathrm{OL}} = -(\eta\gamma^2)W\frac{x}{\gamma}\frac{x}{\gamma}^T$, where $\frac{x}{\gamma}$ is drawn from a gaussian distribution with variance 1. Hence, any change in γ in the original formulation can be offset by a corresponding change in η, and the relevant space of learning constants is only one-dimensional.

For node perturbation, $\Delta W_{\mathrm{NP}} = (\frac{\eta}{\sigma^2}\gamma^4)(\frac{\xi}{\gamma}^T W\frac{x}{\gamma} + \frac{1}{2}\frac{\xi}{\gamma}^T\frac{\xi}{\gamma})\frac{\xi}{\gamma}\frac{x}{\gamma}^T$, where $\frac{\xi}{\gamma}$ is drawn from a gaussian distribution with variance $\frac{\sigma}{\gamma}$, and $\frac{x}{\gamma}$ has variance 1. Here too, a change in γ can be compensated for by appropriate changes in the other two parameters, and the relevant learning constant space has two dimensions.

For weight perturbation, $\Delta W_{\mathrm{WP}} = (\frac{\eta}{\sigma^2}\gamma^2)(\frac{x}{\gamma}^T\psi^T W\frac{x}{\gamma} + \frac{1}{2}\frac{x}{\gamma}^T\psi^T\psi\frac{x}{\gamma})\psi$. Once again changes in γ can be subsumed into changes in the other parameters, and we need consider only a two-dimensional learning constant space.

Without loss of generality, therefore, we set $\gamma = 1$ to simplify the remainder of this discussion.

2.2 Learning Curves. The appendix gives derivations for the following learning curves and convergence conditions on η, where the parenthesized

superscript is a time index, and the angle brackets indicate a mean taken over both noise and inputs at every time step. For the online gradient method,

$$\left\langle E_{\mathrm{OL}}^{(t)}\right\rangle = (1-2\eta+(N+2)\eta^2)^t E^{(0)}$$
$$\eta_{\mathrm{OL}} < \frac{2}{N+2}.$$

In a single online learning run, $E^{(t)}$ would depend on the particular values of x that were randomly chosen; averaging over the ensemble of possible inputs x removes this variation. We therefore use this averaged error $\langle E^{(t)}\rangle$ as the learning curve measuring the performance of the system.

The limit on η depends on N because of the randomness inherent in an online training regimen; the exact gradient for error due to a given single input x will not in general match that for error averaged over the entire ensemble of inputs. (More details appear in the appendix.) This "gradient noise" (Widrow & Lehr, 1990) is common to all three algorithms considered here.

For node and weight perturbation:

$$\left\langle E_{\mathrm{NP}}^{(t)}\right\rangle = \left(E^{(0)} - \frac{\eta\sigma^2(M+2)(M+4)MN}{8(2-(N+2)(M+2)\eta)}\right)$$
$$\cdot(1-2\eta+(M+2)(N+2)\eta^2)^t + \frac{\eta\sigma^2(M+2)(M+4)MN}{8(2-(M+2)(N+2)\eta)}$$
$$\eta_{\mathrm{NP}} < \frac{2}{(M+2)(N+2)}$$
$$\left\langle E_{\mathrm{WP}}^{(t)}\right\rangle = \left(E^{(0)} - \frac{\eta\sigma^2 MN(MN(M+2)(N+2)+12(MN+2))}{8(2-(N+2)(MN+2)\eta)}\right)$$
$$\cdot(1-2\eta+(N+2)(MN+2)\eta^2)^t$$
$$+\frac{\eta\sigma^2 MN(MN(M+2)(N+2)+12(MN+2))}{8(2-(N+2)(MN+2)\eta)}$$
$$\eta_{\mathrm{WP}} < \frac{2}{(MN+2)(N+2)}$$

3 Comparison of Learning Curves

All three of the above learning curves $\langle E^{(t)}\rangle$ take the form

$$\bar{E}(\alpha(\eta))^t + \beta(\eta,\sigma),$$

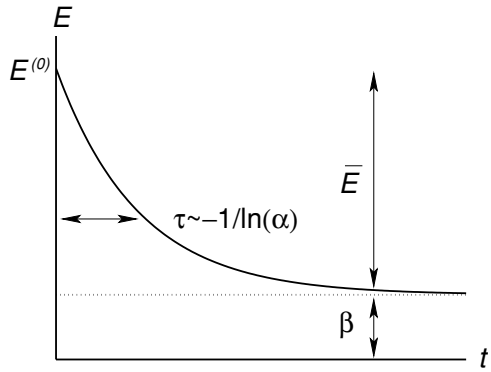

Figure 1: Sketch of a sample learning curve, converging from initial error $E^{(0)}$ to residual error β at speed $-\ln(\alpha)$.

where β is the residual error that the network will approach as $t \to \infty$ if learning converges, $\bar{E} \equiv E^{(0)} - \beta$ is the transient error, and α is a multiplicative factor by which $\bar{E}$ changes at each time step. The magnitude of α, which depends on the parameter η but not on σ, determines whether the average error will converge and the speed at which it will do so.[1] Figure 1 illustrates schematically these quantities, by which we will be comparing the algorithms.

For the online gradient method, $\beta = 0$; a network trained this way, if it converges, will approach zero error as $t \to \infty$. The stochastic algorithms have positive β, which is a result of the noise: when W is far from the minimum of the objective function, the noise will typically be small in comparison to the term to which it is added, but close to the minimum, the noise will prevent the system from attaining arbitrarily low error. The residual error depends on both η and σ; in the limit $\sigma \to 0$, this residual error vanishes. Of course, σ cannot be set directly to 0, or the stochastic algorithms will cease to function.

3.1 Equal Average Updates. One way to compare these different algorithms with respect to performance is to choose learning rates η such that all three have the same weight update on average. As noted above, choosing the same value of η in all three cases will ensure this condition. That common value of η must be small enough that all three algorithms converge. If we take $\eta \ll \frac{1}{MN^2}$, the learning curves, to highest order in η, M, and N,

[1] Note the important distinction between *learning rate* η, which is a constant affecting the magnitude of individual weight updates, and *speed of learning*, a property of a given learning curve associated with this multiplicative factor $\alpha(\eta)$. The latter is the relevant measure of performance when comparing learning curves.

become

$$\left\langle E_{\mathrm{OL}}^{(t)}\right\rangle = \bar{E}(1-2\eta)^t$$

$$\left\langle E_{\mathrm{NP}}^{(t)}\right\rangle = \bar{E}\,(1-2\eta)^t + \frac{1}{16}\eta\sigma^2 M^3 N$$

$$\left\langle E_{\mathrm{WP}}^{(t)}\right\rangle = \bar{E}\,(1-2\eta)^t + \frac{1}{16}\eta\sigma^2 M^3 N^3.$$

In section 1, we outlined an argument that because of the problem of determining updates to many weights based on a single reward signal, a stochastic gradient-following approach might be expected to learn more slowly than a direct one, which has no such credit assignment problem. However, for equal small η, the average error for all three algorithms converges at the same speed. Weight perturbation approaches a larger value of residual error than does node perturbation, unless a value of σ at least a factor of N larger for the latter than for the former is chosen; however, in the $\sigma \to 0$ limit, the residual error vanishes for both.

3.2 Maximal Learning Speeds. The usual way to choose the learning rate in applications of training networks is to use the value of η for which the error experimentally turns out to have the fastest speed of convergence. In this light, the previous comparison may be of more theoretical than practical interest. Arguably a more practical way to compare the algorithms, then, is to choose the "optimal" learning rate for each, here defined as that value of η for which the average error converges most quickly.[2] The learning curves, to highest order in M and N, are then

$$\left\langle E_{\mathrm{OL}}^{(t)}\right\rangle = \bar{E}\left(1-\frac{1}{N}\right)^t$$

$$\left\langle E_{\mathrm{NP}}^{(t)}\right\rangle = \bar{E}\left(1-\frac{1}{MN}\right)^t + \frac{1}{8}\sigma^2 M^2$$

$$\left\langle E_{\mathrm{WP}}^{(t)}\right\rangle = \bar{E}\left(1-\frac{1}{MN^2}\right)^t + \frac{1}{8}\sigma^2 M^2 N.$$

Direct gradient descent, then, can train a network faster than can node perturbation, which in turn is faster than weight perturbation.

[2] Other definitions for what is to be considered optimal are possible, for example, the amount of time taken for error to fall below some specified threshold, or the level to which error falls within a specified number of updates. We use the present definition because the rate of convergence of error to its asymptotic value seems arguably the most relevant quantity, and is analytically tractable under this framework.

The noise takes different forms in the two stochastic variants. For node perturbation, ξ_i is added directly to the ith output unit; for weight perturbation, the quantity added to the same output unit is $\sum_{ij} \psi_{ij} x_j$. By the central limit theorem, the latter approaches a gaussian with mean 0 and variance $N\sigma^2$ for large N. For the most direct comparison of the two stochastic variants, therefore, the variance of ξ should be chosen a factor N larger than that of ψ. With this choice, the residual error for the two stochastic variants becomes identical, and the learning curves differ only in their speeds of convergence.

3.3 Parallel vs. Sequential Update. The terms *node* and *weight perturbation* have been most often used to refer to sequential variation of the tunable system parameters. We can apply the node perturbation approach as described here, but add noise to only a single output unit at a time and update only the corresponding weights; or apply weight perturbation, varying and adjusting only one weight at a time, where the one output unit or one weight is chosen at random with uniform probability. Similar analysis then gives for the convergence condition and learning curves

$$\eta_{\mathrm{NP}} < \frac{2}{3(N+2)}$$

$$\left\langle E^{(t)}_{\mathrm{NP}} \right\rangle = \bar{E}\left(1 - \frac{2}{M}\eta + \frac{3(N+2)}{M}\eta^2\right)^t + \frac{15\eta\sigma^2 MN}{4(2-3(N+2)\eta)}$$

$$\eta_{\mathrm{WP}} < \frac{2}{3(N+2)}$$

$$\left\langle E^{(t)}_{\mathrm{WP}} \right\rangle = \bar{E}\left(1 - \frac{2}{MN}\eta + \frac{3(N+2)}{MN}\eta^2\right)^t + \frac{45\eta\sigma^2 MN}{4(2-3(N+2)\eta)}$$

for this serial update strategy. Choosing the optimal learning rates makes the learning curves

$$\left\langle E^{(t)}_{\mathrm{NP}} \right\rangle = \bar{E}\left(1 - \frac{1}{3M(N+2)}\right)^t + \frac{5}{4}\sigma^2\frac{MN}{N+2}$$

$$\left\langle E^{(t)}_{\mathrm{WP}} \right\rangle = \bar{E}\left(1 - \frac{1}{3MN(N+2)}\right)^t + \frac{15}{4}\sigma^2\frac{MN}{N+2}.$$

3.4 Uncertainty in Weight Updates. In practice, any of the quantities used to calculate weight updates may have some associated uncertainty, due, for instance, to poor estimation or VLSI manufacturing imperfections (G. Cauwenberghs, personal communication, November 2003). Treating these uncertainties as gaussian random variables, a very general case can

be written

$$\Delta W = -\frac{\eta}{\sigma^2}\left(E(x+a,\zeta+b) - E(x+c,0+e) + d\right)\epsilon(x+f,\zeta+g),$$

where E is the squared error; ϵ is the eligibility; ζ is either ξ or ψ according to whether node or weight perturbation is being considered; and $a \ldots g$ are tensors, with dimensions matching those of their addends, whose components are normally-distributed variables with means $\mu_{a,i} \ldots \mu_{g,i}$ and variances $\sigma^2_{a,i} \ldots \sigma^2_{g,i}$ (where the second index is over components). Expanding this full expression according to the treatment above gives a set of equations too lengthy to report here, as well as a set of constraints on $a \ldots g$ that must hold in order for the recursive approach described here to be applicable (e.g., a must be identical to c; certain pairs of variables must not both have nonzero mean). Because these general results are not particularly illuminating, we will discuss only two special cases here.

With uncertainty only on the offset error (only d nonzero), for both node and weight perturbation, only the residual error is affected, not the maximum rate of convergence or bound on η. The new learning curves are

$$\begin{aligned}\left\langle E^{(t)}_{\mathrm{NP}}\right\rangle &= \bar{E}(1-2\eta+(M+2)(N+2)\eta^2)^t \\ &\quad + \frac{\eta MN\left(\sigma^4(M+2)(M+4)+4\left(\sigma_d^2+\mu_d^2+\mu_d\sigma^2(M+2)\right)\right)}{8\sigma^2(2-(M+2)(N+2)\eta)} \\ \left\langle E^{(t)}_{\mathrm{WP}}\right\rangle &= \bar{E}(1-2\eta+(N+2)(MN+2)\eta^2)^t \\ &\quad + \left[\eta MN\left(\sigma^4(MN(M+2)(N+2)+12(MN+2))+4\left(\sigma_d^2\right.\right.\right. \\ &\quad \left.\left.\left.+\ \mu_d^2+\mu_d\sigma^2(MN+2)\right)\right)\right]/[8\sigma^2(2-(N+2)(MN+2)\eta)].\end{aligned}$$

With uncertainty on the injected noise used to calculate the eligibility (only g nonzero), both the residual error and the bound on η are affected. For node perturbation, the new learning curve and condition on η are

$$\begin{aligned}\left\langle E^{(t)}_{\mathrm{NP}}\right\rangle &= \bar{E}\left(1-2\eta+\eta^2(N+2)\left(M+2+\frac{1}{\sigma^2}(|\sigma_g|^2+|\mu_g|^2)\right)\right)^t \\ &\quad + \frac{\eta\sigma^2 MN(M+2)((M+4)\sigma^2+|\sigma_g|^2+|\mu_g|^2)}{8(2\sigma^2-\eta(N+2)((M+2)\sigma^2+|\sigma_g|^2+|\mu_g|^2))} \\ \eta_{\mathrm{NP}} &< \frac{2\sigma^2}{(N+2)((M+2)\sigma^2+|\sigma_g|^2+|\mu_g|^2)}.\end{aligned}$$

With weight perturbation, in order for the recursive method used here to be applied, $\mu_{g,ij}$ must be zero for all $\{i, j\}$, but $\sigma_{g,ij}$ may still be nonzero in

general. To highest order in M and N, the learning curve and condition on η become

$$\left\langle E_{\mathrm{WP}}^{(t)} \right\rangle = \bar{E}\left(1 - 2\eta + \frac{\eta^2}{\sigma^2} N(MN\sigma^2 + \|\sigma_g\|^2)\right)^t + \frac{\eta\sigma^2 M^2 N^2 (MN\sigma^2 + \|\sigma_g\|^2)}{8(2\sigma^2 - \eta N(MN\sigma^2 + \|\sigma_g\|^2))}$$

$$\eta_{\mathrm{WP}} < \frac{2\sigma^2}{N(MN\sigma^2 + \|\sigma_g\|^2)}.$$

4 Discussion

In a linear feedforward network of N input and M output units, in terms of the maximum possible speed of convergence of average error, online gradient descent on a squared error function is faster by a factor of M than node perturbation, which in turn is faster by a factor of N than weight perturbation. The difference in the speed of convergence is the dimensionality of the noise. Weight perturbation operates by explicit exploration of the entire MN-dimensional weight space; only one component of a particular update will be in the direction of the true gradient for a given input, while the other components can be viewed as noise masking that signal. That is, an update can be written as $\Delta W = \langle \Delta W \rangle$ (the "learning signal," the actual gradient) + $(\Delta W - \langle \Delta W \rangle)$ (the "learning noise"), where the average is taken over all values of ψ. This learning noise will typically have magnitude $\sqrt{MN}$ larger than the learning signal, and so MN samples are required in order to average it away. Direct gradient descent gives weight updates that are purely signal in this sense; while still occurring in an MN-dimensional space, they are by definition exactly in the direction of the gradient for a given input. Thus, no exploration of the weight space or averaging over multiple samples is necessary, and the maximum learning speed is correspondingly greater. Node perturbation is a stochastic algorithm like weight perturbation, but it explores the M-dimensional output space rather than the larger weight space; the learning noise is of lower dimension, and correspondingly fewer samples need to be averaged to reveal a learning signal of a given size.

It has previously been argued that the maximum learning speed should scale not with the dimensionality of the update, as shown here, but with the square root of that dimensionality (Cauwenberghs, 1993). That claim is based on the fact that the squared magnitude of the update goes as the number of dimensions, and for a given error landscape and position in weight space, there will be a maximum update size, greater than which instability will result. However, a more quantitative approach is to examine the

conditions under which error will decrease, as we have done above. Rather than stopping with the statement that the size of the weight update scales as the square root of the number of dimensions, we have shown that this fact implies that the restriction on convergence scales with the first power of the dimensionality. Numerical simulations of error curves, averaged over many individual trials with online updating, support these conclusions with respect to both the quantitative shapes of the learning curves and the scaling behavior of the conditions on convergence (see Figure 2).

4.1 Parallel vs. Sequential Update. Serial perturbation allows the use of a larger η than does the parallel approach first discussed, by a factor of D, the dimensionality of the noise (M for node perturbation, MN for weight perturbation). However, the learning curves at optimal η scale with M and N in the same way for serial as for parallel updates. Intuitively, by the argument of the preceding section, a perturbation with only one component (and correspondingly smaller learning-signal-to-learning-noise ratio) allows the step size to be increased by a factor of D, but the algorithm must then cycle through all D components, so the net learning speed is no faster.

Further, while the fastest learning curves scale the same way with network size for serial as for parallel update, the speed differs by a constant factor: the network can learn three times faster with parallel update than with serial. This factor comes about from a term in the parallel equations $(D + 2)$. With parallel updates and large networks, the constant 2 is negligible compared to D; but that term becomes 3 for serial updates, where the effective dimensionality of the noise is 1. The 2 represents, in effect, the "overhead cost" of the learning noise; the parallel approach minimizes the effect of this overhead by updating all components at once, while the serial approach encounters it with each successive component.

The serial curves also have a residual error smaller by a factor of D than parallel. However, if the variance σ^2 is scaled by D to make the total noise injected into the system directly comparable in the two cases, as in the discussion of the difference between the residual error with node versus weight perturbation in equation 3.2 above, this difference vanishes. A constant difference remains, but this can also be adjusted for by the choice of variance.

4.2 Uncertainty in Weight Updates. For both node and weight perturbation, with uncertainty only in the offset error (nonzero d), the residual error is increased compared to the case without uncertainty. Moreover, in this case, it cannot be made arbitrarily small by making σ arbitrarily small; as $\sigma \to 0$, the residual error now diverges. For a given value of σ_d, an optimal value of σ can be calculated for which residual error is a minimum. Learning speed for a given η, and bounds on η, remain unaffected.

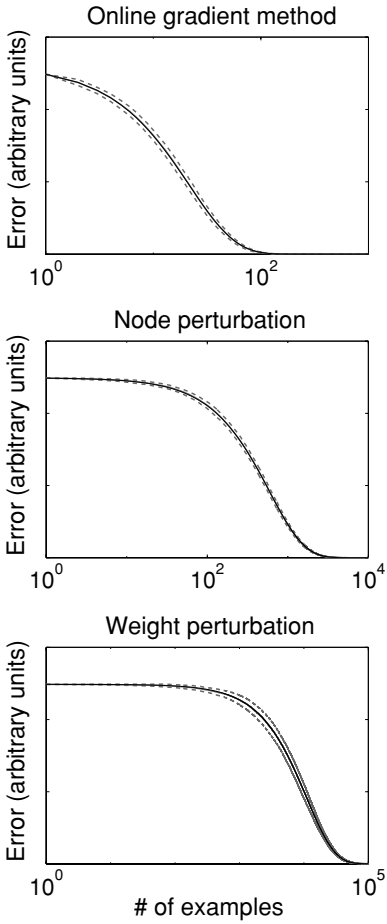

Figure 2: Sample learning curves for the three algorithms applied to a linear feedforward network as described in the text, showing the agreement between theory (black) and experiment (gray). In each case, a network of linear units with $N = 20$, $M = 25$, $\sigma = 10^{-3}$, and optimal η was trained on successive input examples for the number of iterations shown. One hundred such runs were averaged together in each case. The three gray lines show the mean (solid) and standard deviation (dashed) of squared error among those runs.

With uncertainty only in the injected noise used in calculation of the eligibility (nonzero g), the residual error can still be made arbitrarily low by choosing σ sufficiently small. However, smaller σ now means a stricter upper bound on η; there is a trade-off between residual error and learning speed.

Because the uncertainties $a \ldots g$ can be expected to be nonzero in general, we can expect typically to encounter the limitations found in both of these special cases, that is, uneliminable residual error and trade-off between residual error and learning speed.

4.3 Other Issues. The analysis in this letter and the corresponding results cover the case where the objective function and distribution of inputs are isotropic. In a sense, this constitutes a worst case, where every weight must be learned with equal importance in the calculation of the cost function. A companion article (in preparation) discusses the more general case of an anisotropic quadratic cost function, which can result, for example, from a restricted set of input patterns. In such a case, different learning modes progress at different speeds, the stochastic algorithms will in general not scale so poorly, and parallel variation of weights may outperform sequential variation by more than a constant factor.

An issue of considerable importance is that of baseline subtraction in the stochastic learning rules. Both can be written as $\Delta W = -\eta/\sigma^2(\tilde{E} - B)e$, where the scalar B is the baseline. In the analysis above, we have chosen B to be the error in the absence of noise. So long as the baseline is uncorrelated with the noise, the stochastic updates will still match the true gradient on average. However, while the mean updates remain the same, their variance can be very great if the baseline is poorly chosen; individual updates will not necessarily decrease the error, and the learning noise will mask the learning signal to make the algorithm effectively unusable. If the baseline is correlated with the noise, even the mean update will not in general match the true gradient; analysis of such a case is not pursued in this study.

While the analytic approach taken here cannot be extended readily to networks of nonlinear units, these results appear to extend at least qualitatively to more complicated networks and architectures. For instance, Figure 3 shows learning curves that result from applying the three algorithms to a two-layer feedforward network of nonlinear units. All three algorithms give identical learning curves if the learning rate is set small enough; as η is increased, the weight perturbation curve fails to converge to low error, while the other two curves continue to match; increasing η further leads to the node perturbation curve's also failing to converge.

Finally, we have considered only those target functions that can be realized by these networks, and have not treated the class of functions for which no teacher weight matrix w^* exists, where attaining zero error is impossible.

5 Conclusion

We have shown that stochastic gradient descent techniques can be expected to scale with increasing network size more poorly than direct ones, in terms of maximum learning speed. This result may serve as a caution regarding the size of networks they may usefully be applied to. However, with learning

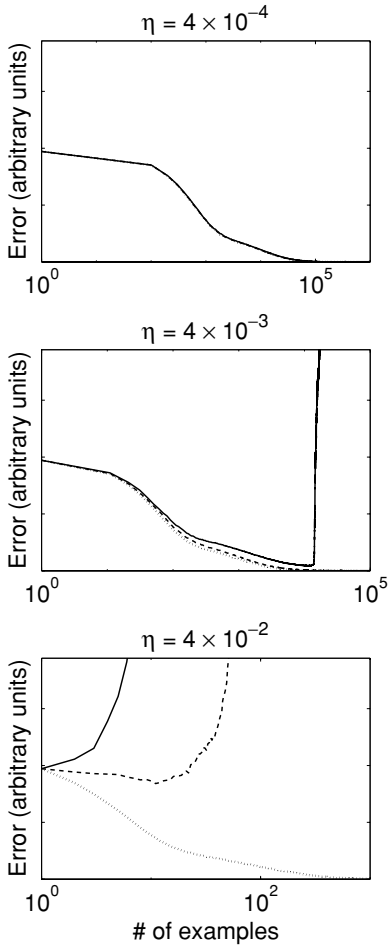

Figure 3: Sample learning curves for the three algorithms applied to a two-layer nonlinear feedforward network (gradient descent, dotted; node perturbation, dashed; weight perturbation, solid). The input, hidden, and output layers each had 10 units, with output equal to the hyperbolic tangent of their weighted input. Inputs and noise were drawn from the same distributions as in the linear case; $\sigma = 10^{-3}$, η had the value shown for all three algorithms in each panel. In each case, the network was trained on successive input examples for the number of iterations shown. Error was evaluated based on the total squared difference between the output of the network and that of a teacher network with randomly chosen weights; the test error was the mean of that for 100 random inputs not used in training. Curves show averages over 100 independent runs.

rates small, equal learning curves in each of the three will follow from equal learning rates, although individual weight updates will typically be considerably different. This is because for correspondingly small adjustments to the weights, only the component parallel to the gradient will have a significant effect on error; orthogonal components will not affect the error to first order. Moreover, node perturbation can have performance comparable to that of direct gradient descent even in training very large networks, so long as the number of output units is small (Fiete et al., 2004). Thus, these stochastic methods may be of considerable utility for training networks in some situations, particularly in reinforcement learning frameworks and those where the gradient of the objective function is difficult or impossible to calculate, for mathematical or practical reasons.

Appendix: Derivations

Here we give in detail the calculation of the learning curves and convergence conditions stated in the text.

A.1 Online Gradient Method. Taking the gradient of the objective function of equation 2.1 gives

$$\Delta W_{\mathrm{OL}} = -\eta W x x^T \tag{A.1}$$

as the individual weight update for particular values of W and x. Given the weight matrix at time $t = 0$ and an input vector x, we can apply one such update and square each element of the result, obtaining

$$W_{ij}^{(1)2} = \left(W_{ij}^{(0)} - \eta \sum_k W_{ik}^{(0)} x_k x_j \right) \left(W_{ij}^{(0)} - \eta \sum_l W_{il}^{(0)} x_l x_j \right). \tag{A.2}$$

Averaging over the distribution of possible inputs,

$$\left\langle W_{ij}^{(1)2} \right\rangle = (1 - 2\eta + 2\eta^2) W_{ij}^{(0)2} + \eta^2 \sum_k W_{ik}^{(0)2}.$$

If we sum over rows (i.e., over all inputs to a given output unit),

$$\sum_j \left\langle W_{ij}^{(1)2} \right\rangle = (1 - 2\eta + (N+2)\eta^2) \sum_j W_{ij}^{(0)2}.$$

Summing over columns as well (i.e., over all output units) gives

$$\sum_{ij} \left\langle W_{ij}^{(1)2} \right\rangle = (1 - 2\eta + (N+2)\eta^2) \sum_{ij} W_{ij}^{(0)2}.$$

This result gives us a recursion relation specifying $\langle \| W \|^2 \rangle$ as a function of time: after t updates,

$$\sum_{ij} \left\langle W_{ij}^{(t)2} \right\rangle = (1 - 2\eta + (N+2)\eta^2)^t \sum_{ij} W_{ij}^{(0)2}.$$

We then have as the condition for convergence of the average error

$$\eta_{\mathrm{OL}} < \frac{2}{N+2},$$

and the learning rate for which the average error converges most quickly on this quadratic landscape is

$$\eta_{\mathrm{OL}}^{*} = \frac{1}{N+2}. \tag{A.3}$$

The learning curve can be written

$$\left\langle E_{\mathrm{OL}}^{(t)} \right\rangle = (1 - 2\eta + (N+2)\eta^2)^t E^{(0)},$$

where the parenthesized superscript indicates the number of updates. Another way to write equation A.2 is explicitly in terms of "gradient signal" (term multiplying $W_{ij}^{(0)}$) plus "gradient noise" (Widrow & Lehr, 1990) (contamination from other components of W due to projection onto x):

$$W_{ij}^{(1)2} = \left(W_{ij}^{(0)} (1 - \eta x_j^2) - \eta \sum_{k \neq j} W_{ik}^{(0)} x_k x_j \right)^2$$

Expanding the multiplication, the cross terms vanish when the average is taken, giving

$$\sum_{ij} \left\langle W_{ij}^{(1)2} \right\rangle = \sum_{ij} W_{ij}^{(0)2} (1 - 2\eta + 3\eta^2) + \eta^2 (N-1) \sum_{ij} W_{ij}^{(0)},$$

where the first term is due entirely to the signal and the second to the noise. Choosing $\eta \lesssim 1/N$ allows the signal to be revealed via averaging over $\gtrsim N$ samples (see also section 4).

A.2 Node Perturbation. The weight update for a given ξ and x is

$$\Delta W_{\mathrm{NP}} = -\frac{\eta}{\sigma^2} \left(\xi^T W x + \frac{1}{2} \xi^T \xi \right) \xi x^T.$$

Averages are taken at each step not only over the inputs but also over the noise. Taking the same approach as before, we have

$$\begin{aligned}\left\langle W_{ij}^{(1)2}\right\rangle &= W_{ij}^{(0)2}(1-2\eta+4\eta^2)\\ &\quad+\eta^2\left(\sum_{kl} W_{kl}^{(0)2}+2\sum_{k} W_{kj}^{(0)2}+2\sum_{l} W_{il}^{(0)2}\right)\\ &\quad+\frac{1}{4}\eta^2\sigma^2(M^2+6M+8).\end{aligned}$$

Summing over all elements, we obtain

$$\begin{aligned}\sum_{ij}\left\langle W_{ij}^{(1)2}\right\rangle &= \sum_{ij} W_{ij}^{(0)2}(1-2\eta+\eta^2(M+2)(N+2))\\ &\quad+\frac{1}{4}\eta^2\sigma^2MN(M^2+6M+8).\end{aligned}$$

The condition for convergence is

$$\eta_{\mathrm{NP}} < \frac{2}{(M+2)(N+2)},$$

so that average error will converge fastest for

$$\eta^*_{\mathrm{NP}} = \frac{1}{(M+2)(N+2)}. \tag{A.4}$$

The learning curve is

$$\begin{aligned}\left\langle E_{\mathrm{NP}}^{(t)}\right\rangle &= \left(E^{(0)}-\frac{\eta\sigma^2(M+2)(M+4)MN}{8(2-(N+2)(M+2)\eta)}\right)(1-2\eta+(M+2)(N+2)\eta^2)^t\\ &\quad+\frac{\eta\sigma^2(M+2)(M+4)MN}{8(2-(N+2)(M+2)\eta)}.\end{aligned}$$

A.3 Weight Perturbation. Here the weight update is

$$\Delta W_{\mathrm{WP}} = -\frac{\eta}{\sigma^2}\left(x^T\psi^T Wx+\frac{1}{2}x^T\psi^T\psi x\right)\psi.$$

The same approach as above gives in this case

$$\begin{aligned}\sum_{ij}\left\langle W_{ij}^{(1)2}\right\rangle &= \sum_{ij} W_{ij}^{(0)2}(1-2\eta+\eta^2((MN+2)(N+2)))\\ &\quad+\frac{1}{4}\eta^2\sigma^2(M^3N^3+2M^2N^3+2M^3N^2+16M^2N^2+24MN).\end{aligned}$$

The condition for convergence is then

$$\eta_{\mathrm{WP}} < \frac{2}{(MN+2)(N+2)},$$

so that the η giving fastest convergence of average error is

$$\eta^*_{\mathrm{WP}} = \frac{1}{(MN+2)(N+2)}. \tag{A.5}$$

The learning curve is

$$\begin{aligned}\left\langle E^{(t)}_{\mathrm{WP}}\right\rangle = &\left(E^{(0)} - \frac{\eta\sigma^2 MN(MN(M+2)(N+2)+12(MN+2))}{8(2-(N+2)(MN+2)\eta)}\right)\\ &\cdot(1-2\eta+(N+2)(MN+2)\eta^2)^t\\ &+\frac{\eta\sigma^2 MN(MN(M+2)(N+2)+12(MN+2))}{8(2-(N+2)(MN+2)\eta)}.\end{aligned}$$

Acknowledgments

We thank Ila Fiete and Gert Cauwenberghs for useful discussions and comments. This work was supported in part by a Packard Foundation Fellowship (to H.S.S.) and NIH grants (GM07484 to MIT and MH60651 to H.S.S.).

References

Baldi, P., & Hornik, K. (1993). Learning in linear neural networks: A survey. *IEEE Transactions on Neural Networks*, *6*(4), 837–858.

Bartlett, P., & Baxter, J. (1999). *Hebbian synaptic modifications in spiking neurons that learn* (Tech. Rep.). Canberra: Research School of Information Sciences and Engineering, Australian National University.

Biehl, M., & Riegler, P. (1994). On-line learning with a perceptron. *Europhys. Lett.*, *28*, 525–530.

Cauwenberghs, G. (1993). A fast stochastic error-descent algorithm for supervised learning and optimization. In C. L. Giles, S. J. Hanson, & J. D. Cowan (Eds.), *Advances in neural information processing systems, 5* (pp. 244–251). San Mateo, CA: Morgan Kaufmann.

Cauwenberghs, G. (1996). An analog VLSI recurrent neural network learning a continuous-time trajectory. *IEEE Transactions on Neural Networks*, *7*(2), 346–361.

Fiete, I. R., Fee, M. S., & Seung, H. S. (2004). *Neural theory of gradient learning with empiric synapses*. Manuscript submitted for publication.

Flower, B., & Jabri, M. (1993). Summed weight neuron perturbation: An $\mathcal{O}(n)$ improvement over weight perturbation. In C. L. Giles, S. J. Hanson, & J. D. Cowan

(Eds.), *Advances in neural information processing Systems, 5* (pp. 212–219). San Mateo, CA: Morgan Kaufmann.

Heskes, T. M., & Kappen, B. (1991). Learning processes in neural networks. *Physical Review A*, *44*(4), 2718–2726.

Jabri, M., & Flower, B. (1992). Weight perturbation: An optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayered networks. *IEEE Transactions on Neural Networks*, *3*(1), 154–157.

Mace, C. W. H., & Coolen, A. C. C. (1998). Statistical mechanical analysis of the dynamics of learning in perceptrons. *Statistics and Computing*, *8*, 55–88.

Werfel, J., Xie, X., & Seung, H. S. (2004). Learning curves for stochastic gradient descent in linear feedforward networks. In S. Thrun, L. Saul, & B. Schölkopf (Eds.), *Advances in neural information processing systems*, *16* Cambridge, MA: MIT Press.

Widrow, B., & Lehr, M. A. (1990). Thirty years of adaptive neural networks: Perceptron, Madaline, and backpropagation. *Proc. IEEE*, *78*(9), 1415–1442.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.

Received November 1, 2004; accepted March 23, 2005.