3.1 Global-Descent-Based Error Backpropagation

Here, a learning method is described in which the gradient-descent rule in batch backprop is replaced with a *global-descent rule* (Cetin et al., 1993a). This methodology is based on a global optimization scheme, acronymed TRUST (terminal repeller unconstrained subenergy tunneling), that formulates optimization in terms of the flow of a special deterministic dynamical system (Cetin et al., 1993b).[1]

Global descent is a gradient descent on a special criterion function $C\left(\mathbf{w},\mathbf{w}^*\right)$ given by

$$C\left(\mathbf{w},\mathbf{w}^*\right)=\ln\left(\frac{1}{1+e^{-\left[E(\mathbf{w})-E\left(\mathbf{w}^*\right)+\sigma\right]}}\right)-\frac{3k}{4}\sum_i\left(w_i-w_i^*\right)^{4/3}u\left[E\left(\mathbf{w}\right)-E\left(\mathbf{w}^*\right)\right] \tag{3.14}$$

where $\mathbf{w}^*$, with component values w_i^*, is a fixed-weight vector that can be a local minimum of $E(\mathbf{w})$ or an initial weight state $\mathbf{w}^0$, $\mathrm{u}[\bullet]$ is the unit step function, σ is a shifting parameter (typically set to 2), and k is a small positive constant. The first term in the right-hand side in Equation (3.14) is a monotonic transformation of the original criterion function (e.g., SSE criterion may be used) that

[1] This method was originally designed for implementation in parallel analog VLSI circuitry, allowing implementation in a form whose computational complexity is only weakly dependent on problem dimensionality.

preserves all critical points of $E(\mathbf{w})$ and has the same relative ordering of the local and global minima of $E(\mathbf{w})$. It also flattens the portion of $E(\mathbf{w})$ above $E\left(\mathbf{w}^*\right)$ with minimal distortion elsewhere. On the other hand, the term $\sum_i \left(w_i - w_i^*\right)^{4/3}$ is a "repeller term," which gives rise to a convex surface with a unique minimum located at $\mathbf{w} = \mathbf{w}^*$. The overall effect of this energy transformation is schematically represented for a one-dimensional criterion function in Figure Error! No text of specified style in document.**-1.**

Performing gradient descent on $C\left(\mathbf{w}, \mathbf{w}^*\right)$ leads to the *global-descent update rule:*

$$\Delta w_i = -\rho \frac{\partial E(\mathbf{w})}{\partial w_i} \frac{1}{1 + e^{+\left[E(\mathbf{w}) - E\left(\mathbf{w}^*\right) + \sigma\right]}} + \rho k \left(w_i - w_i^*\right)^{1/3} \mathrm{u}\left[E(\mathbf{w}) - E\left(\mathbf{w}^*\right)\right] \tag{3.15}$$

The first term on the right-hand side of Equation (3.15) is a *subenergy gradient,* while the second term is a *non-Lipschitzian terminal repeller* (Zak, 1989). Upon replacing the gradient descent in Equations (3.2) and (3.4) by Equation (3.15), where w_i represents an arbitrary hidden-unit or output-unit weight, the modified backprop procedure may escape local minima of the original criterion

function $E(\mathbf{w})$ given in Equation (3.13). Here, the batch training is required because Equation (3.15) necessitates a unique error surface for all patterns.

The update rule in Equation (3.15) automatically switches between two phases: a tunneling phase and a gradient-descent phase. The tunneling phase is characterized by $E(\mathbf{w}) \geq E(\mathbf{w}^*)$. Since for this condition the subenergy gradient term is nearly zero in the vicinity of the local minimum $\mathbf{w}^*$, the terminal repeller term in Equation (3.15) dominates, leading to the dynamical system

$$\Delta w_i \approx \rho k \left(w_i - w_i^* \right)^{1/3} \qquad (3.16)$$

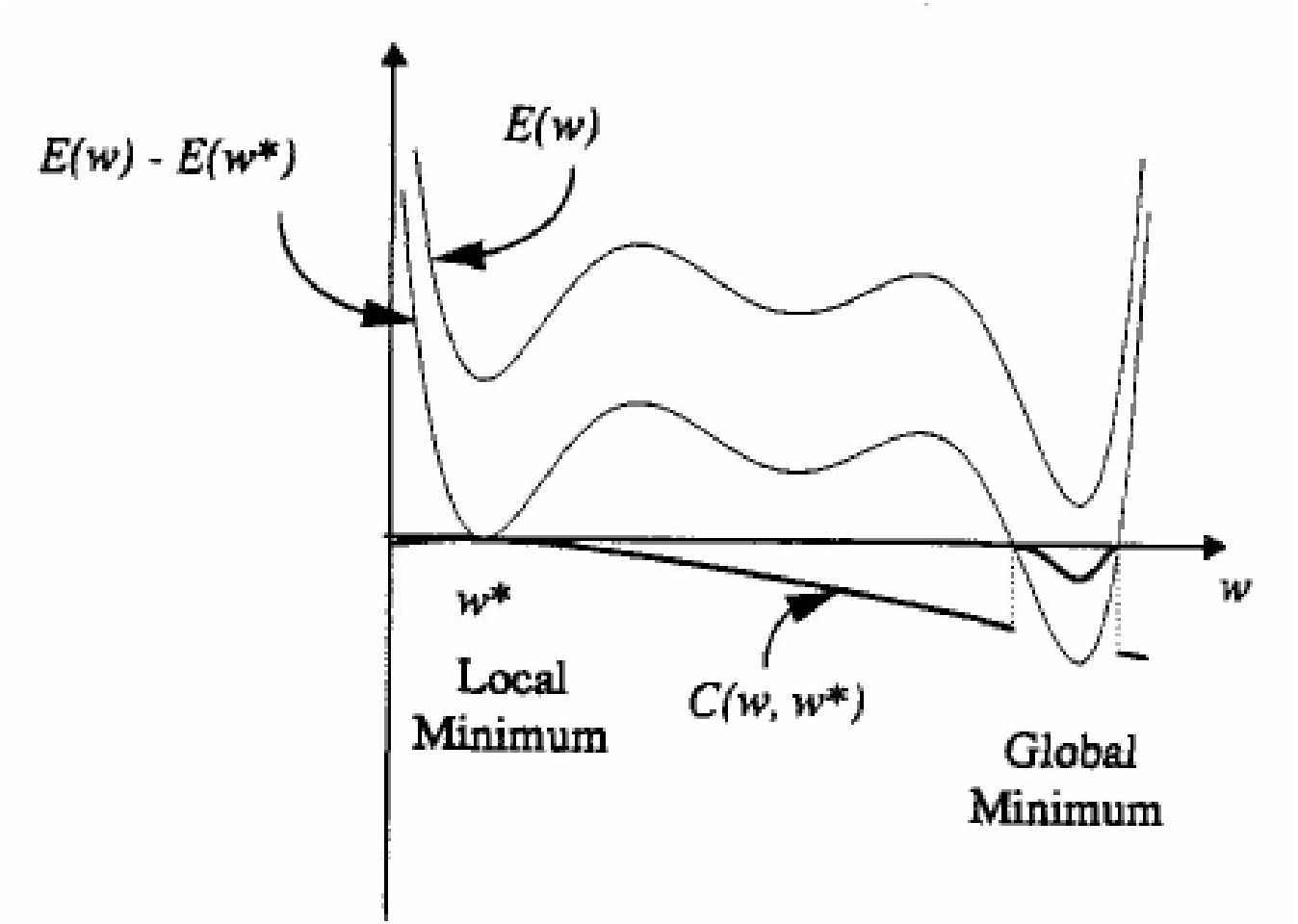

Figure Error! No text of specified style in document.**-1** A plot of a one-dimensional criterion function $E(\mathbf{w})$ with local minimum at w^*. The function $E(w)-E(w^*)$ is plotted below, as well as the global-descent criterion function $C(w,w^*)$.

This system has an unstable repeller equilibrium point at $w_t = wf$, i.e., at the local minimum of $E(\mathbf{w})$. The "power" of this repeller is determined by the constant k. Thus the dynamical system given by Equation (3.15), when initialized with a small perturbation from $\mathbf{w}^*$, is repelled from this local minimum until it reaches a lower-energy region $E(\mathbf{w}) < E(\mathbf{w}^*)$; i.e., tunneling through portions of $E(\mathbf{w})$ where $E(\mathbf{w}) \geq E(\mathbf{w}^*)$ is accomplished. The second phase is a gradient-descent minimization phase characterized by $E(\mathbf{w}) < E(\mathbf{w}^*)$. Here, the repeller term is identically zero. Thus Equation (3.15) becomes

$$\Delta w_i = -\rho(\mathbf{w}) \frac{\partial E(\mathbf{w})}{\partial w_i} \qquad (3.17)$$

where $\rho(\mathbf{w})$ is a dynamic learning rate (step size) equal to $\rho\left\{1 + \exp\left[E(\mathbf{w}) - E(\mathbf{w}^*) + \sigma\right]\right\}^{-1}$. Note that $\rho(\mathbf{w})$ is approximately equal to ρ when $E(\mathbf{w}^*)$ is large compared to $E(\mathbf{w}) + \sigma$.

Initially, $\mathbf{w}^*$ is chosen as one comer of a domain in the form of a hyperparallelepiped of dimension $J(n + 1) + L(J + 1)$, which is the dimension of $\mathbf{w}$ in the architecture of **Error! Reference source not**

found.. A slightly perturbed version of $\mathbf{w}^*$, namely, $\mathbf{w}^* + \varepsilon_{\mathbf{w}}$, is taken as the initial state of the dynamical system in Equation (3.15). Here $\varepsilon_{\mathbf{w}}$, is a small perturbation that drives the system into the domain of interest. If $E\left(\mathbf{w}^* + \varepsilon_{\mathbf{w}}\right) < E\left(\mathbf{w}^*\right)$, the system immediately enters a gradient-descent phase that equilibrates at a local minimum. Every time a new equilibrium is reached, $\mathbf{w}^*$ is set equal to this equilibrium, and Equation (3.15) is reinitialized with $\mathbf{w}^* + \varepsilon_{\mathbf{w}}$ which ensures a necessary consistency in the search flow direction. Since $\mathbf{w}^*$ is now a local minimum, $E\left(\mathbf{w}\right) \geq E\left(\mathbf{w}^*\right)$ holds in the neighborhood of $\mathbf{w}^*$. Thus the system enters a repelling (tunneling) phase, and the repeller at $\mathbf{w}^*$ repels the system until it reaches a lower basin of attraction where $E\left(\mathbf{w}\right) < E\left(\mathbf{w}^*\right)$. As the dynamical system enters the next basin, the system automatically switches to gradient descent and equilibrates at the next lower local minimum. Then $\mathbf{w}^*$ is set equal to this new minimum, and the process is repeated. If, on the other hand, $E\left(\mathbf{w}^* + \varepsilon_{\mathbf{w}}\right) \geq E\left(\mathbf{w}^*\right)$ at the onset of training, then the system is initially in a tunneling phase. The tunneling will proceed to a lower basin, at which point it enters the minimization phase and follows the behavior discussed above. Training can be stopped when a

minimum $\mathbf{w}^*$ corresponding to $E(\mathbf{w}^*)=0$ is reached or when $E(\mathbf{w}^*)$ becomes smaller than a preset threshold.

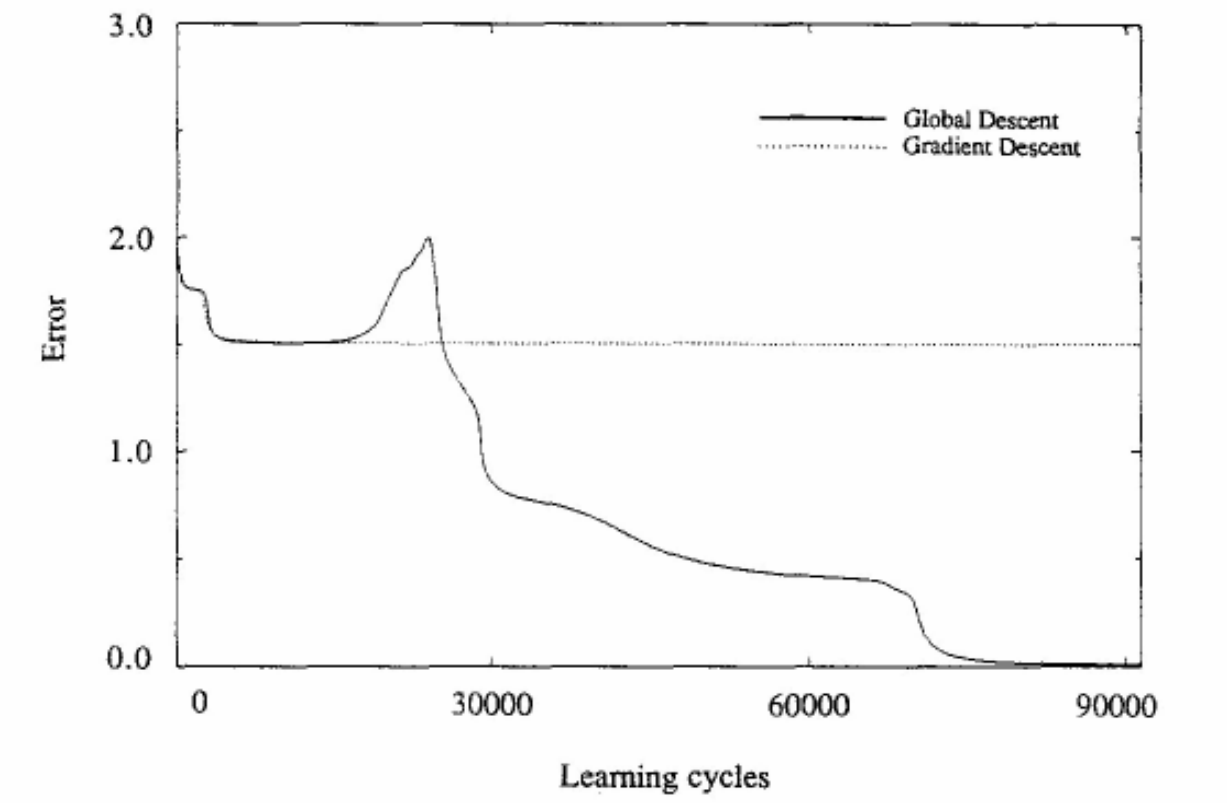

Figure Error! No text of specified style in document.**-2** Learning curves for global-descent- and gradient-descent-based batch backprop for the 4-bit parity.

The global-descent method is guaranteed to find the global minimum for functions of one variable but not for multivariate functions. However, in the multidimensional case, the algorithm will always escape from one local minimum to another with a lower or equal functional value. Figure Error! **No text of specified style in document.**-2 compares the learning curve for the global-descent-based backprop with that of batch backprop for the 4-bit parity problem in a feedforward net with four hidden units and a single output unit. The same initial random weights are used in both cases. The figure depicts one tunneling phase for the global-descent algorithm before convergence to a (perfect) global-minimum solution. In performing this simulation, it is found that the choice of the direction of the perturbation vector $\varepsilon_{\mathbf{w}}$, is very critical in regard to reaching a global minimum successfully. On the other hand, batch backprop converges to the first local minimum it reaches. This local solution represents a partial solution to the 4-bit parity problem (i.e., mapping error is present). Simulations using incremental backprop with the same initial weights as in the preceding simulations are also performed but are not shown in the figure. Incremental backprop was able to produce both solutions

shown in Figure Error! **No text of specified style in document.**-2; very small learning rates (ρ_0 and ρ_n,) often lead to imperfect local solutions, while relatively larger learning rates may lead to a perfect solution.

3.2 Backprop Enhancements and Variations

In general, learning with backprop is slow (Huang and Lippmann, 1988). Typically, this is due to the characteristics of the error surface. The surface is characterized by numerous flat and steep regions. In addition, it has many troughs that are flat in the direction of search. These characteristics are particularly pronounced in classification problems, especially when the size of the training set is small.

Many enhancements of and variations to backprop have been proposed. These are mostly heuristic modifications with goals of increased speed of convergence, avoidance of local minima, and/or improvement in the network's ability to generalize. This section presents some common heuristics that may improve these aspects of backprop learning in multilayer feedforward neural networks.

3.2.1Weight Initialization

Owing to its gradient-descent nature, backprop is very sensitive to initial conditions. If the choice of the initial weight vector $\mathbf{w}^0$ (here $\mathbf{w}$ is a point in the weight space being searched by backprop) happens to be located within the attraction basin of a strong local minima attractor (one where the minima is at the bottom of a steep-sided valley of the criterion/error surface), then the convergence of backprop will be fast and the solution quality will be determined by the depth of the valley relative to the depth of the global minima. On the other hand, backprop converges very slowly if $\mathbf{w}^0$ starts the search in a relatively flat region of the error surface.

An alternative explanation for the sensitivity of backprop to initial weights (as well as to other learning parameters) is advanced by Kolen and Pollack (1991). Using Monte Carlo simulations on simple feedforward nets with incremental backprop learning of simple functions, they discovered a complex fractal-like structure for convergence as a function of initial weights. They reported regions of high sensitivity in the weight space where two very close initial points can lead to substantially different learning curves. Thus they hypothesize that these fractal-like structures arise in backprop due to the nonlinear nature of the dynamic learning equations, which exhibit multiple attractors;

rather than the gradient-descent metaphor with local valleys to get stuck in, they advance a many-body metaphor where the search trajectory is determined by complex interactions with the systems attractors.

In practice, the weights are normally initialized to small zero-mean random values (Rumelhart et al., 1986). The motivation for starting from small weights is that large weights tend to prematurely saturate units in a network and render them insensitive to the learning process (this phenomenon is known as *flat spot*). On the other hand, randomness is introduced as a symmetry-breaking mechanism; it prevents units from adopting similar functions and becoming redundant.

A sensible strategy for choosing the magnitudes of the initial weights for avoiding premature saturation is to choose them such that an arbitrary unit *i* starts with a small and random weighted sum net_i. This may be achieved by setting the initial weights of unit *i* to be on the order of $1/\sqrt{f_i}$ where f_i is the number of inputs (fan-in) for unit *i*. It can be easily shown that for zero-mean random uniform weights in $[-r,+r]$ and assuming normalized inputs that are randomly and uniformly distributed in the range [0,1], net_i has zero mean and has standard deviation $\sigma_{net_i} = (r/3)\sqrt{f_i}$. Thus,

by generating uniform random weights within the range $\left[-3/\sqrt{f_i}, +3/\sqrt{f_i}\right]$,the input to unit i (net_i) is a random variable with zero mean and a standard deviation of unity, as desired.

In simulations involving single-hidden-layer feedforward networks for pattern classification and function approximation tasks, substantial improvements in backprop convergence speed and avoidance of "bad" local minima are possible by initializing the hidden unit weight vectors to normalized vectors selected randomly from the training set (Denoeux and Lengelle, 1993).

3.2.2Learning Rate

The convergence speed of backprop is directly related to the learning rate parameter ρ [ρ_o and ρ_h Equations (3.2) and (3.9), respectively]; if ρ is small, the search path will closely approximate the gradient path, but convergence will be very slow due to the large number of update steps needed to reach a local minima. On the other hand, if ρ is large, convergence initially will be very fast, but the algorithm will eventually oscillate and thus not reach a minimum. In general, it is desirable to have large steps when the search point is far away from a minimum, with decreasing step size as the

search approaches a minimum. This section gives a sample of the various approaches for selecting the proper learning rate.

One early proposed heuristic (Plaut et al., 1986) is to use constant learning rates that are inversely proportional to the fan-in of the corresponding units. The increased convergence speed of backprop as a result of using this method of setting the individual learning rates for each unit inversely proportional to the number of inputs to that unit has been theoretically justified by analyzing the eigenvalue distribution of the Hessian matrix of the criterion function, $\nabla^2 E$ (Le Cun et al., 199l). Such learning rate normalization can be thought of intuitively as maintaining balance between the learning speed of units with different fan-in. Without this normalization, after each learning iteration, units with high fan-in have their input activity (*net*) changed by a larger amount than units with low fan-in. Thus, and due to the nature of the sigmoidal activation function used, the units with large fan-in tend to commit their output to a saturated state prematurely and are rendered difficult to adapt. Therefore, normalizing the learning rates of the various units by dividing by their corresponding fan-in helps speed up learning.

The optimal learning rate for fast convergence of backprop/gradient-descent search is the inverse of the largest eigenvalue of the Hessian matrix **H** of the error function *E* evaluated at the search point **w**. Computing the full Hessian matrix is prohibitively expensive for large networks with thousands of parameters involved. Therefore, finding the largest eigenvalue $\lambda_{\max}$, for speedy convergence seems rather inefficient. However, one may employ a shortcut to efficiently estimate $\lambda_{\max}$ (Le Cun et al., 1993). This shortcut is based on a simple method of approximating the product of **H** by an arbitrarily chosen (random) vector **z** through Taylor expansion: $\mathbf{Hz} = (1/\alpha)\left[\nabla E(\mathbf{w}+\alpha\mathbf{z}) - \nabla E(\mathbf{w})\right]$, where α is a small positive constant. Now, using the power method, which amounts to iterating the procedure

$$\mathbf{z} \leftarrow \frac{\mathbf{Hz}}{\|\mathbf{z}\|} = \frac{1}{\alpha}\left[\nabla E\left(\mathbf{w} + \alpha\frac{\mathbf{z}}{\|\mathbf{z}\|}\right) - \nabla E(\mathbf{w})\right],$$

the vector **z** converges to $|\lambda_{\max}|\mathbf{c}_{\max}$, where $\mathbf{c}_{\max}$ is the normalized eigenvector of **H** corresponding to $\lambda_{\max}$. Thus the norm of the converged vector **z** gives a good estimate of $|\lambda_{\max}|$, and its reciprocal may

now be used as the learning rate in backprop. An on-line version of this procedure is reported by Le Cun et al. (1993).

Many heuristics have been proposed so as to adapt the learning rate automatically. Chan and Fallside (1987) proposed an adaptation rule for $\rho(t)$ that is based on the cosine of the angle between the gradient vectors $\nabla E(t)$ and $\nabla E(t-1)$ (here, *t* is an integer that represents iteration number). Sutton (1986) presented a method that can increase or decrease $\rho_i(t)$ for each weight w_i, according to the number of sign changes observed in the associated partial derivative $\frac{\partial E}{\partial w_i}$. Franzini (1987) investigated a technique that heuristically adjusts $\rho(t)$, increasing it whenever $\nabla E(t)$ is close to $\nabla E(t-1)$ and decreasing it otherwise. Cater (1987) suggested using separate parameters $\rho^k(t)$ one for each pattern $\mathbf{x}^k$.

Silva and Almeida (1990) used a method where the learning rate for a given weight w_i, is set to $^{a\rho_i(t)}$ if $\frac{\partial E(t)}{\partial w_i}$ and $\frac{\partial E(t-1)}{\partial w_i}$ have the same sign, with $a > 1$; if the partial derivatives have different signs,

then a learning rate of $b\rho_i(t)$ is used, with $0 < b < 1$. A similar, theoretically justified method for increasing the convergence speed of incremental gradient-descent search is to set $\rho(t) = \rho(t-1)$ if $\nabla E(t)$ has the same sign as $\nabla E(t-1)$, and $\rho(t) = \rho(t-1)/2$ otherwise (Pflug, 1990).

When the input vectors are assumed to be randomly and independently chosen from a probability distribution, we may view incremental backprop as a stochastic gradient-descent algorithm. Thus simply setting the learning rate ρ to a constant results in persistent residual fluctuations around a local minimum $\mathbf{w}^*$. The variance of such fluctuations depends on the size of ρ, the criterion function being minimized, and the training set. Based on results from stochastic approximation theory, the "running average" schedule $\rho(t) = \rho_0/(1+t)$ with sufficiently small ρ, guarantees asymptotic convergence to a local minimum $\mathbf{w}^*$. However, this schedule leads to very slow convergence. Here, one would like to start the search with a learning rate faster than $1/t$ but then ultimately converge to the $1/t$ rate as $\mathbf{w}^*$ is approached. Unfortunately, increasing ρ_0 can lead to instability for small t. Darken and Moody (1991) proposed the "search then converge" schedule $\rho(t) = \rho_0/\left[1+(t/\tau)\right]$, which allows for faster convergence without compromising stability. In this

schedule, the learning rate stays relatively high for a "search time" τ during which it is hoped that the weights will hover about a good minimum. Then, for times $t \gg \tau$, the learning rate decreases as $\tau\rho_0 / t$, and the learning converges. Note that for $\tau = 1$, this schedule reduces to the running average schedule. Therefore, a procedure for optimizing τ is needed. A completely automatic "search then converge" schedule can be found in Darken and Moody (1992).

3.2.3Momentum

Another simple approach to speed up backprop is through the addition of a momentum term (Plaut et al., 1986) to the right-hand side of the weight update rules in Equations (3.2) and (3.9). Here, each weight change Δw_i, is given some momentum so that it accelerates in the average downhill direction instead of fluctuating with every change in the sign of the associated partial derivative $\frac{\partial E}{\partial w_i(t)}$. The addition of momentum to gradient search is stated formally as

$$\Delta w_i(t) = -\rho \frac{\partial E}{\partial w_i(t)} + \alpha \Delta w_i(t-1) \qquad (3.18)$$

where α is a momentum rate normally chosen between 0 and 1 and $\Delta w_i(t-1) = w_i(t) - w_i(t-1)$. Equation (3.18) is a special case of multistage gradient methods that have been proposed for accelerating convergence and escaping local minima.

The momentum term also can be viewed as a way of increasing the effective learning rate in almost-flat regions of the error surface while maintaining a learning rate close to ρ (here $0 < \rho \ll 1$) in regions with high fluctuations. This can be seen by employing an *N*-step recursion and writing Equation (3.18) as

$$\Delta w_i(t) = -\rho \sum_{n=0}^{N-1} \alpha^n \frac{\partial E}{\partial w_i(t-n)} + \alpha^N \Delta w_i(t-N) \qquad (3.19)$$

If the search point is caught in a flat region, then $\frac{\partial E}{\partial w_i}$ will be about the same at each time step, and Equation (3.19) can be approximated as (with $0 < \alpha < 1$ and *N* large)

$$\Delta w_i(t) \approx -\rho \frac{\partial E}{\partial w_i(t)} \sum_{n=0}^{N-1} \alpha^n = -\frac{\rho}{1-\alpha} \frac{\partial E}{\partial w_i(t)} \qquad (3.20)$$

Thus, for flat regions, a momentum term leads to increasing the learning rate by a factor $1/(1-\alpha)$. On the other hand, if the search point is in a region of high fluctuation, the weight change will not gain momentum; i.e., the momentum effect vanishes. An empirical study of the effects of ρ and α on the convergence of backprop and on its learning curve can be found in Tollenaere (1990).

Adaptive momentum rates also may be employed. Fahlman (1989) proposed and extensively simulated a heuristic variation of backprop, called *quickprop,* that employs a dynamic momentum rate given by

$$\alpha(t) = \frac{\dfrac{\partial E}{\partial w_i(t)}}{\dfrac{\partial E}{\partial w_i(t-1)} - \dfrac{\partial E}{\partial w_i(t)}} \qquad (3.21)$$

With this adaptive $\alpha(t)$ substituted in Equation (3.18), if the current slope is persistently smaller than the previous one but has the same sign, then $\alpha(t)$ is positive, and the weight change will accelerate. Here, the acceleration rate is determined by the magnitude of successive differences between slope values. If the current slope is in the opposite direction from the previous one, it

signals that the weights are crossing over a minimum. In this case, $\alpha(t)$ has a negative sign, and the weight change starts to decelerate. Additional heuristics are used to handle the undesirable case where the current slope is in the same direction as the previous one but has the same or larger magnitude; otherwise, this scenario would lead to taking an infinite step or moving the search point backwards or up the current slope and toward a local maximum. Substituting Equation (3.21) in Equation (3.18) leads to the update rule

$$\Delta w_i(t) = -\rho \frac{\partial E}{\partial w_i(t)} - \frac{\dfrac{\partial E}{\partial w_i(t)}}{\left[\dfrac{\dfrac{\partial E}{\partial w_i(t)} - \dfrac{\partial E}{\partial w_i(t-1)}}{\Delta w_i(t-1)}\right]} \qquad (3.22)$$

It is interesting to note that Equation (3.22) corresponds to steepest gradient-descent-based adaptation with a dynamically changing effective learning rate $\rho(t)$. This learning rate is given by the sum of the original constant learning rate ρ and the reciprocal of the denominator of the second term in the right-hand side of Equation (3.22).

The use of error gradient information at two consecutive time steps in Equation (3.21) to improve convergence speed can be justified as being based on approximations of second-order search methods such as Newton's method. Newton's method is based on a quadratic model $\widetilde{E}(\mathbf{w})$ of the criterion $E(\mathbf{w})$ and hence uses only the first three terms in a Taylor series expansion of E about the "current" weight vector $\mathbf{w}^c$:

$$\widetilde{E}\left(\mathbf{w}^c+\Delta\mathbf{w}\right)=E\left(\mathbf{w}^c\right)+\nabla E\left(\mathbf{w}^c\right)^{\mathrm{T}}\Delta\mathbf{w}+\frac{1}{2}\Delta\mathbf{w}^{\mathrm{T}}\nabla^2 E\left(\mathbf{w}^c\right)\Delta\mathbf{w}$$

This quadratic function is minimized by solving the equation $\widetilde{E}\left(\mathbf{w}^c+\Delta\mathbf{w}\right)=\mathbf{0}$, which leads to Newton's method: $\Delta w=-\left[\nabla^2 E\left(\mathbf{w}^c\right)\right]^{-1}\nabla E\left(\mathbf{w}^c\right)=-\left[H\left(\mathbf{w}^c\right)\right]^{-1}\nabla E\left(\mathbf{w}^c\right)$. Here, $\mathbf{H}$ is the Hessian matrix with components $\mathbf{H}_{ij}=\dfrac{\partial^2 E}{\partial w_i \partial w_j}$.

Newton's algorithm iteratively computes the weight changes $\Delta\mathbf{w}$ and works well when initialized within a convex region of E. In fact, the algorithm converges quickly if the search region is quadratic or nearly so. However, this method is very computationally expensive, since the computation $\mathbf{H}^{-1}$

requires $O\left(N^3\right)$ operations at each iteration (here, N is the dimension of the search space). Several authors have suggested computationally efficient ways of approximating Newton's method (e.g. Becker and Le Cun, 1989). Becker and Le Cun proposed an approach whereby the off-diagonal elements of **H** are neglected, thus arriving at the approximation

$$\Delta w_i = -\frac{\partial E}{\partial w_i}\left(\frac{\partial^2 E}{\partial w_i^2}\right)^{-1} \qquad (3.23)$$

which is a "decoupled" form of Newton's rule where each weight is updated separately. The second term in the right-hand side of Equation (3.22) can now be viewed as an approximation of Newton's rule, since its denominator is a crude approximation of the second derivative of E at step t. In fact, this suggests that the weight update rule in Equation (3.22) may be used with $\rho = 0$.

As with Equation (3.22), special heuristics must be used in order to prevent the search from moving in the wrong gradient direction and in order to deal with regions of very small curvature, such as inflection points and plateaus, which cause Δw_i in Equation (3.23) to blow up. A simple solution is

to replace the $\frac{\partial^2 E}{\partial w_i^2}$ term in Equation (3.23) by $\left|\frac{\partial^2 E}{\partial w_i^2}\right| + \mu$ where μ is a small positive constant. The approximate Newton method just described is capable of scaling the descent step in each direction. However, because it neglects off-diagonal Hessian terms, it is not able to rotate the search direction as in the exact Newton's method. Thus this approximate rule is only efficient if the directions of maximal and minimal curvature of *E* happen to be aligned with the weight space axes. Bishop (1992) reported a somewhat efficient technique for computing the elements of the Hessian matrix exactly using multiple feedforward propagation through the network followed by multiple backward propagation.

Another approach for deriving theoretically justifiable update schedules for the momentum rate in Equation (3.18) is to adjust $\alpha(t)$ at each update step such that the gradient-descent search direction is "locally" optimal. In *optimal steepest descent* (also known as *best-step steepest descent),* the learning rate is set at time *t* such that it minimizes the criterion function *E* at time step *t* + 1; i.e., we desire a ρ that minimizes $E\left[\mathbf{w}(t+1)\right] = E\left\{\mathbf{w}(t) - \rho \nabla E\left[\mathbf{w}(t)\right]\right\}$. Unfortunately, this optimal learning step is impractical because it requires computation of the Hessian $\nabla^2 E$ at each time step.

However, one may still use some of the properties of the optimal ρ in order to accelerate the search, as demonstrated next.

When $\mathbf{w}(t)$ is specified, the necessary condition for minimizing $E[\mathbf{w}(t+1)]$ is

$$\begin{aligned}\frac{\partial E[\mathbf{w}(t+1)]}{\partial \rho} &= \nabla E[\mathbf{w}(t+1)]^{\mathrm{T}} \frac{\partial \mathbf{w}(t+1)}{\partial \rho} \\ &= -\nabla E[\mathbf{w}(t+1)]^{\mathrm{T}} \nabla E[\mathbf{w}(t)] = 0\end{aligned} \qquad (3.24)$$

This implies that the search direction in two successive steps of optimal steepest descent are orthogonal. The easiest method to enforce the orthogonal requirement is the Gram-Schmidt orthogonalization method. Suppose that the search direction at time $t-1$ is known, denoted $\mathbf{d}(t-1)$, and that the "exact" gradient $\nabla E(t)$ (used in batch backprop) can be computed at time step t [to simplify notation, we write $E[\mathbf{w}(t)]$ as $E(t)$]. Now, we can satisfy the condition of orthogonal consecutive search directions by computing a new search direction, employing Gram-Schmidt orthogonolization

$$\mathbf{d}(t) = -\nabla E(t) + \frac{\nabla E(t)^{\mathrm{T}}\,\mathbf{d}(t-1)}{\mathbf{d}(t-1)^{\mathrm{T}}\,\mathbf{d}(t-1)}\mathbf{d}(t-1) \qquad (3.25)$$

Performing descent search in the direction **d**(*t*) in Equation (3.25) leads to the weight vector update rule

$$\Delta\mathbf{w}(t) = -\rho\nabla E(t) + \frac{\nabla E(t)^{\mathrm{T}}\,\mathbf{d}(t-1)}{\mathbf{d}(t-1)}\Delta\mathbf{w}(t-1) \qquad (3.26)$$

where the relation $\Delta\mathbf{w}(t-1) = \mathbf{w}(t) - \mathbf{w}(t-1) = +\rho\mathbf{d}(t-1)$ has been used. Comparing the component-wise weight update version of Equation (3.26) with Equation (3.18) reveals another adaptive momentum rate given by

$$\alpha(t) = \frac{\nabla E(t)^{\mathrm{T}}\,\mathbf{d}(t-1)}{\left\|\mathbf{d}(t-1)\right\|^2} = \rho\frac{\nabla E(t)^{\mathrm{T}}\,\Delta\mathbf{w}(t-1)}{\left\|\Delta\mathbf{w}(t-1)\right\|^2}$$

Another similar approach is to set the current search direction **d**(*t*) to be a compromise between the current "exact" gradient $\nabla E(t)$ and the previous search direction $\mathbf{d}(t-1)$; i.e.,

$\mathbf{d}(t) = -\nabla E(t) + \beta \mathbf{d}(t-1)$, with $\mathbf{d}(0) = -\nabla E(0)$. This is the basis for the conjugate gradient method in which the search direction is chosen (by appropriately setting β) so that it distorts as little as possible the minimization achieved by the previous search step. Here, the current search direction is chosen to be conjugate (with respect to **H**) to the previous search direction. Analytically, we require $\mathbf{d}(t-1)^{\mathrm{T}} \mathbf{H}(t-1)\mathbf{d}(t) = 0$, where the Hessian $\mathbf{H}(t-1)$ is assumed to be positive definite. In practice, β, which plays the role of an adaptive momentum, is chosen according to the Polak-Ribiere rule (Polak and Ribiere, 1969):

$$\beta = \beta(t) = \frac{\left[\nabla E(t) - \nabla E(t-1)\right]^{\mathrm{T}} \nabla E(t)}{\left\|\nabla E(t-1)\right\|^2}$$

Thus the search direction in the conjugate gradient method at time *t* is given by

$$\begin{aligned}\mathbf{d}(t) &= -\nabla E(t) + \beta \mathbf{d}(t-1) \\ &= -\nabla E(t) + \frac{\left[\nabla E(t) - \nabla E(t-1)\right]^{\mathrm{T}} \nabla E(t)}{\left\|\nabla E(t-1)\right\|^2} \mathbf{d}(t-1)\end{aligned}$$

Now, using $\mathbf{d}(t-1) = (1/\rho)\Delta\mathbf{w}(t-1)$ and substituting the preceding expression for $\mathbf{d}(t)$ in $\Delta\mathbf{w}(t) = \rho\mathbf{d}(t)$ leads to the weight update rule:

$$\Delta\mathbf{w}(t) = -\rho\nabla E(t) + \frac{\left[\nabla E(t) - \nabla E(t-1)\right]^{\mathrm{T}} \nabla E(t)}{\left\|\nabla E(t-1)\right\|^2}\Delta\mathbf{w}(t-1)$$

When E is quadratic, the conjugate gradient method theoretically converges in N or fewer iterations. In general, E is not quadratic, and therefore, this method would be slower than what the theory predicts. However, it is reasonable to assume that E is approximately quadratic near a local minimum. Therefore, conjugate gradient descent is expected to accelerate the convergence of backprop once the search enters a small neighborhood of a local minimum. As a general note, the basic idea of conjugate gradient search was introduced by Hestenes and Stiefel (1952). Beckman (1964) gives a good account of this method. van der Smagt (1994) gave additional characterization of second-order backprop (such as conjugate gradient-based backprop) from the point of view of optimization. The conjugate gradient method has been applied to multilayer feedforward neural net training (van der Smagt, 1994) and is shown to outperform backprop in speed of convergence.

It is important to note that the preceding second-order modifications to backprop improve the speed of convergence of the weights to the "closest" local minimum. This faster convergence to local minima is the direct result of employing a better search direction as compared with incremental backprop. On the other hand, the stochastic nature of the search directions of incremental backprop and its fixed learning rates can be an advantage, since they allow the search to escape shallow local minima, which generally leads to better solution quality. These observations suggest the use of hybrid learning algorithms, where one starts with incremental backprop and then switches to conjugate gradient-based backprop for the final convergence phase. This hybrid method has its roots in a technique from numerical analysis known as *Levenberg-Marquardt optimization* (Press et al., 1986).

As a historical note, it should be mentioned that the concept of gradient descent was first introduced by Cauchy for use in the solution of simultaneous equations; the method has enjoyed popularity ever since. For a good survey of gradient search, the reader is referred to the book by Polyak (1987).