Biol. Cybern. 59, 257–263

Biological Cybernetics

Accelerating the Convergence of the Back-Propagation Method

T. P. Vogl[1], J. K. Mangis[1], A. K. Rigler[2], W. T. Zink[1], and D. L. Alkon[3]

[1] Environmental Research Institute of Michigan, 1501 Wilson Boulevard, Arlington, VA 22209, USA
[2] Computer Science Department, University of Missouri, Rolla, MO 65401, USA
[3] Neural Systems Section, National Institute of Neurological and Communicative Disorders and Stroke, NIH, 9000 Rockville Pike, Bethesda, MD 20892, USA

Abstract. The utility of the back-propagation method in establishing suitable weights in a distributed adaptive network has been demonstrated repeatedly. Unfortunately, in many applications, the number of iterations required before convergence can be large. Modifications to the back-propagation algorithm described by Rumelhart et al. (1986) can greatly accelerate convergence. The modifications consist of three changes: *1*) instead of updating the network weights after each pattern is presented to the network, the network is updated only after the entire repertoire of patterns to be learned has been presented to the network, at which time the algebraic sums of all the weight changes are applied: *2*) instead of keeping η, the "learning rate" (i.e., the multiplier on the step size) constant, it is varied dynamically so that the algorithm utilizes a near-optimum η, as determined by the local optimization topography; and *3*) the momentum factor α is set to zero when, as signified by a failure of a step to reduce the total error, the information inherent in prior steps is more likely to be misleading than beneficial. Only after the network takes a useful step, i.e., one that reduces the total error, does α again assume a non-zero value. Considering the selection of weights in neural nets as a problem in classical nonlinear optimization theory, the rationale for algorithms seeking only those weights that produce the globally minimum error is reviewed and rejected.

The Back-Propagation Method

The back propagation method of Rumelhart et al. (1986) is a gradient descent method that will establish the weights in a multi-layer, feed-forward adaptive "neural" network. Small arbitrary weights are chosen to initialize the system. Learning is accomplished by successively adjusting the weights based on a set of input patterns and the corresponding set of desired output patterns. During this iterative process, an input pattern is presented to the network and propagated forward to determine the resulting signal at the output units. The differences between the actually resulting output signal and the predetermined desired output signal in each output unit represents an error that is back-propagated through the network in order to adjust the weights. The learning process continues until the network responds with output signals the sum of whose root-mean-square errors from the desired output signals is less than a preset value. The equations governing Rumelhart's method are reproduced in (1–4).

When an input pattern p is applied to the network, the activation of each unit is dynamically determined using the logistic activation function:

$$o_{pj}=\frac{1}{1+\exp\left\{-\left(\sum_i w_{ji}o_{pi}+\theta_j\right)\right\}}, \tag{1}$$

where o_{pj} is the activation of unit j as a result of the application of pattern p, w_{ji} is the weight from unit i to unit j, and θ_j is the bias for unit j. Back propagation is then invoked to update all of the weights in the network according to the following rule:

$$\Delta w_{ji}(n+1)=\eta\cdot\delta_{pj}\cdot o_{pi}+\alpha\cdot\Delta w_{ji}(n), \tag{2}$$

where n is the presentation number i.e., the number of times the system has been presented a pattern, η is the learning rate, δ_{pj} is the error signal for unit j, and α is the momentum factor. The error signal δ_{pj} for an *output* unit j is calculated from the difference between the target value and the actual value for that unit:

$$\delta_{pj}=(t_{pj}-o_{pj})\cdot o_{pj}\cdot(1-o_{pj}). \tag{3}$$

The error signal δ_{pj} for a *hidden* unit$j$ is a function of the error signals of those units in the next higher layer connected to unit j and the weights of those connections:

$$\delta_{pj}=o_{pj}\cdot(1-o_{pj})\cdot\sum_k\delta_{pk}w_{kj}. \tag{4}$$

In practice this algorithm is currently employed as follows: after each input pattern is presented to the network, the consequent error vector across output units is determined and back-propagated through the network to update the weights [1]. The next pattern is then presented and the process repeated. Two parameters η and α, respectively an adjustment of step size and a weight on the "memory" of previous steps, are at the disposal of the user. Assuming η and α are appropriately chosen, the back-propagation process will generally converge to a minimum that satisfies the criterion imposed by the user, usually that the sum of the squares of the error of the output signal, $\sum(t_{pj}-o_{pj})^2$ will be less than a predetermined value for all p.

Back-propagation, as described above, encounters two difficulties in practice. These are:

1) a step (a change of weights and activations based on the corrections produced by the back-propagation) that reduces the error with respect to one pattern will not, in general, produce a network with reduced errors with respect to all the other patterns which the system is to learn. Such a step may misdirect the optimization path and thus may increase considerably the number of iterations required for convergence;

2) the value of η, which modulates the step size, is sensitive to the local shape of the hyper-dimensional terrain which is being traversed in the optimization. If a steep *V*- shaped valley is being followed (particularly if that valley has twist and turns), too large a value of η will cause steps that bounce between the two opposite sides of the valley rather than following the contour of its bottom. On the other hand, too small a value of η will prevent the system from making reasonable progress across a long flat slope. Choosing a suitable learning rate for a particular problem thus involves experimenting with different values to see how the convergence reacts. Rumelhart et al. (1986) state that for the most rapid learning, η should be as large as possible without leading to oscillation. Unfortunately, even though one value of η may be optimum at one stage of the learning, there is no guarantee that the same learning rate will be appropriate at any other stage of the learning process.

The optimum value of η depends on the topography of the domain being traversed. If the contours of constant error are circular, then clearly neither the step size nor the acceleration factor will influence the outcome and gradient descent will converge. If the contours are elliptical and/or bent, and local minima do exist, then convergence to the global minimum cannot be guaranteed or expected. Only relatively slow convergence to some local minimum, that may fortuitously also be the global minimum, will be achieved. However, if that local minimum satisfies the criterion originally imposed for stopping the back-propagation method, that minimum will be indistinguishable from the global minimum unless the search is repeated with a fortuitously well-chosen different starting point. We return to this issue in the discussion below.

A number of recent papers have featured higher order methods. A few remarks on the relative merits of Newton-type methods, gradient, and conjugate direction [e.g. Parker (1987); Pineda (1987)] are appropriate at this point in order to set our suggested algorithm in a proper context with respect to other methods.

First, the second order convergence of Newton's method occurs only in the neighborhood of a local solution and the size of that neighborhood diminishes with increasing number of variables. The inclusion in the algorithm of higher order derivatives will, therefore, generally not accelerate convergence far from a solution. This statement does not imply that higher order derivatives are not of value in other approaches.

Secondly, the conjugate direction algorithms are superlinearly convergent in the neighborhood of a local solution, and several of these conjugate direction algorithms, such as the Polak-Ribiere variant of the Fletcher-Reeves method and the Broydon-Fletcher-Goldfarb-Shanno variant of Davidon's algorithm have been used successfully for simply conducting a descent search. (See Luenberger (1984) for references.) Muneer (1988) offers empirical evidence for the superiority of conjugate direction algorithms over Newton's method.

Finally, the gradient descent method (of which Back Propagation is an example) is notorious for its slowness. The reason for its tendency to stagnate was explained by Akaike (1959). In essence, its behavior is dominated by the two-dimensional subspace determined by the eigenvectors associated with the largest and least eigenvalues of the Hessian matrix. In contrast, conjugate direction methods explicitly construct their searches using linearly independent vectors that span the space. Our variation of the gradient search changes learning parameters from time to time, thus "bumping" the current solution from the two-dimensional "cage" which is characteristic of the pure gradient scheme.

The Modified Back-Propagation Method

To accelerate the convergence of back-propagation, the Rumelhart algorithm was modified based on experience gained with other non-linear optimization

[1] Rumelhart et al. (1986), in their footnote 1, note that "the values of the bias can be learned just like any other weights. We simply imagine that the bias is the weight from a unit that is always on"

algorithms (Pegis et al. 1966) that have been used for the design of a variety of engineering problems with non-linear properties and constraints, including optical design problems, mechanical design problems, and piping flow problems. The following modifications were made to the Back Propagation method:

1. The network weights are not updated after each pattern is presented. Rather, the weights are modified only after *all* input patterns have been presented. After each pattern is presented, all weight changes are calculated as usual through back-propagation, but these changes are not immediately applied. Instead, the changes for each weight are summed over all of the input patterns and the sum is applied to modify the weight after each iteration over all the patterns. Thus (2) is modified as follows:

$$\Delta w_{ji}(m+1)=\eta\cdot\sum_{p}(\delta_{pj}o_{pi})+\alpha\cdot\Delta w_{ji}(m). \tag{5}$$

Note that m (in contrast to n in (2)) represents the iteration number rather than the presentation number, since the weights are updated only once per iteration through all the patterns.

2. The learning rate η is varied according to whether or not an iteration decreases the performance index (the total error for all patterns). If an update results in reduced total error, η is multiplied by a factor $\phi>1$ for the next iteration. If a step produces a network with a total error more than a few (typically, 1–5) percent above the previous value, all changes to the weights are rejected, η is multiplied by a factor $\beta<1$, α is set equal to zero, and the step is repeated. When a successful step is then taken, α is reset to its original value.

The rationale behind this maneuver is that as long as the topography of the terrain is relatively uniform and the descent is in a relatively smooth line, the memory implicit in α will aid convergence. If, however, a step results in a degradation of the performance of the system, then clearly the topography of the terrain demands a change in the direction of the optimization, and prior experience (incorporated in the term in α) will be more misleading than beneficial. Hence α is set to 0 so that memory from previous steps is lost. Only after the network takes a step that reduces the total error will α again assume a non-zero value.

Comparison of the Two Methods

A simple subject for comparison of the Rumelhart Back Propagation Method and our modification thereof, is the three-layer $T-C$ case examined by Rumelhart et al. (1986) using a 3×3 pixel receptive field and considering all four orientations of the two

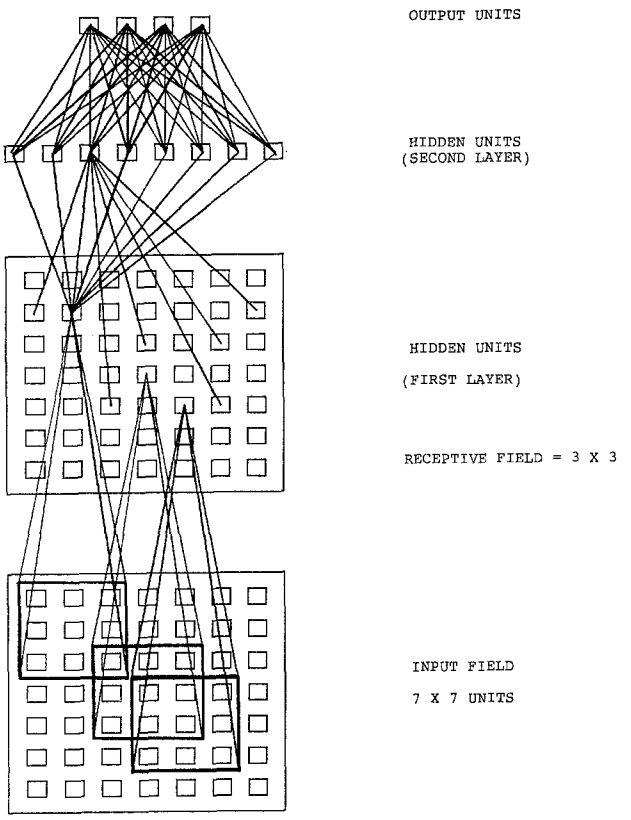

Fig. 1. Network used is like Rumelhart's et al. (1986) $T-C$ network but with one additional layer of hidden units and more than one output unit. Each hidden unit in the first layer connects to a "receptive field" of nine input units. All hidden units in the second layer connect to all hidden units in the first layer and to all of the output units

letters. For this case, the original back propagation method using an $\eta=0.1$ and an $\alpha=0.9$ required 2119 iterations while the modified method required 826 iterations.

In a more difficult problem, a four-layer network (shown in Fig. 1), consisting of a 2-dimensional input plane of 7×7 input elements, a layer of hidden units of equal size, a second layer of 8 hidden units, and an output layer consisting of 4 units, was posed the problem of learning the characters shown in Fig. 2 in a translationally independent manner using the same constraints described in the $T-C$ problem discussed by Rumelhart et al. (1986). Using the original Rumelhart algorithm described above with $\alpha=0.9$ and various learning rates between 0.01 and 0.25, convergence was not achieved after more than 30,000 iterations (the measure of learning, the performance index, is calculated after each iteration. It is the sum of the squares of the errors on the output units for all patterns; one iteration consists of one set of presentations of each of the different input patterns). Given a small enough η and enough iterations, it is suspected that this problem

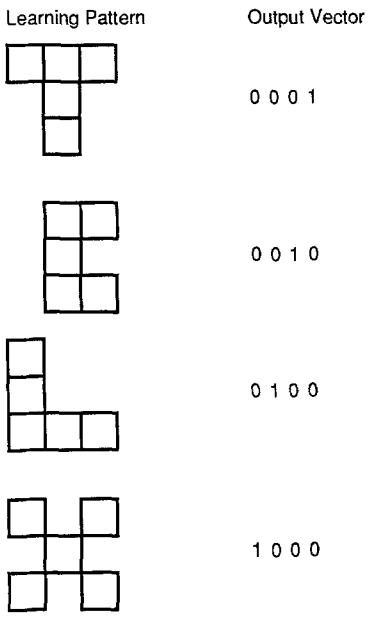

Fig. 2. The four sets of patterns to be learned

will converge to a solution; unfortunately, as the learning rate (η) gets smaller, more iterations are needed before convergence is achieved. Figure 3 shows the total learning error as a function of iteration using the modified method ($\phi = 1.05$, $\beta = 0.7$) from the same starting point and with the same convergence criterion. The revised algorithm converged after 1,000 rather than after 30,000+iterations.

We found that in the initial stages of the optimization, η can be quite large, on the order of 0.1; during

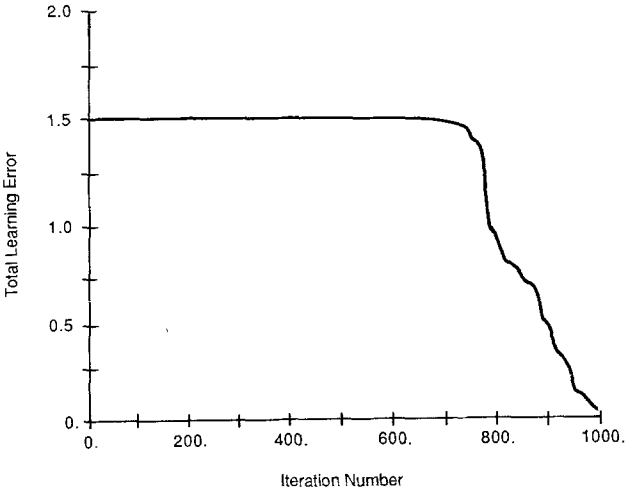

Fig. 3. Convergence behavior for TCLX problem using the modified back-propagation method

the intermediate portions of the optimization, η may have to decrease in size by several orders of magnitude and remains small until the end game when η can again increase in size to 0.1 or more. It is the extreme flatness of the plateau, coupled with a characteristic that may be characterized as "roughness," that makes it desirable to permit the performance index to increase slightly before corrective measures are taken. Finally, in the endgame, the errors are small and therefore η needs to be large in order to make significant changes.

Figures 4 and 5 show the convergence and η behavior respectively for a yet more difficult problem. Using the modified algorithm, 21 patterns were taught

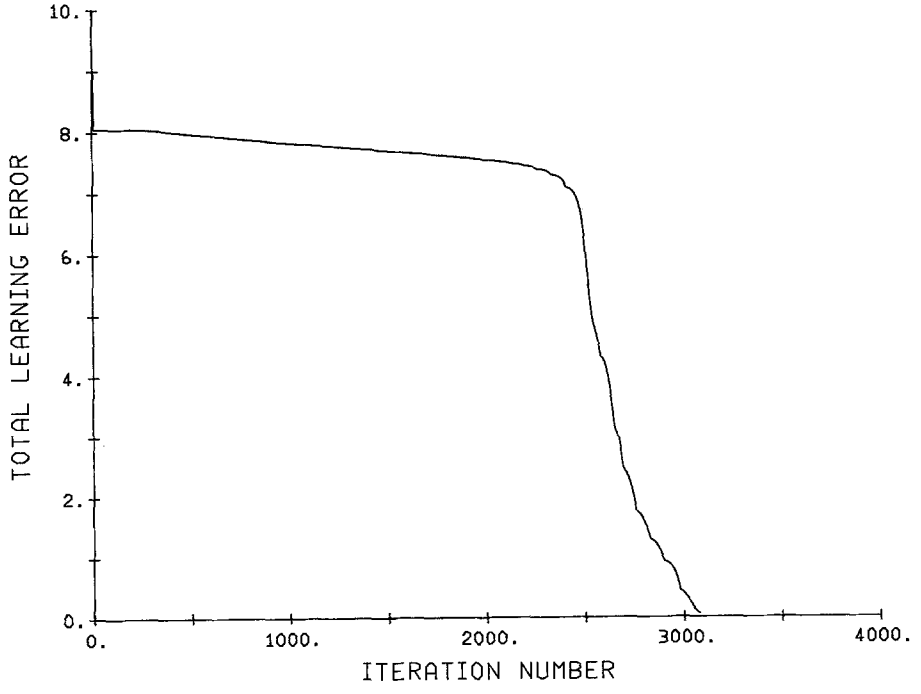

Fig. 4. Convergence behavior for the 21 patterns problem

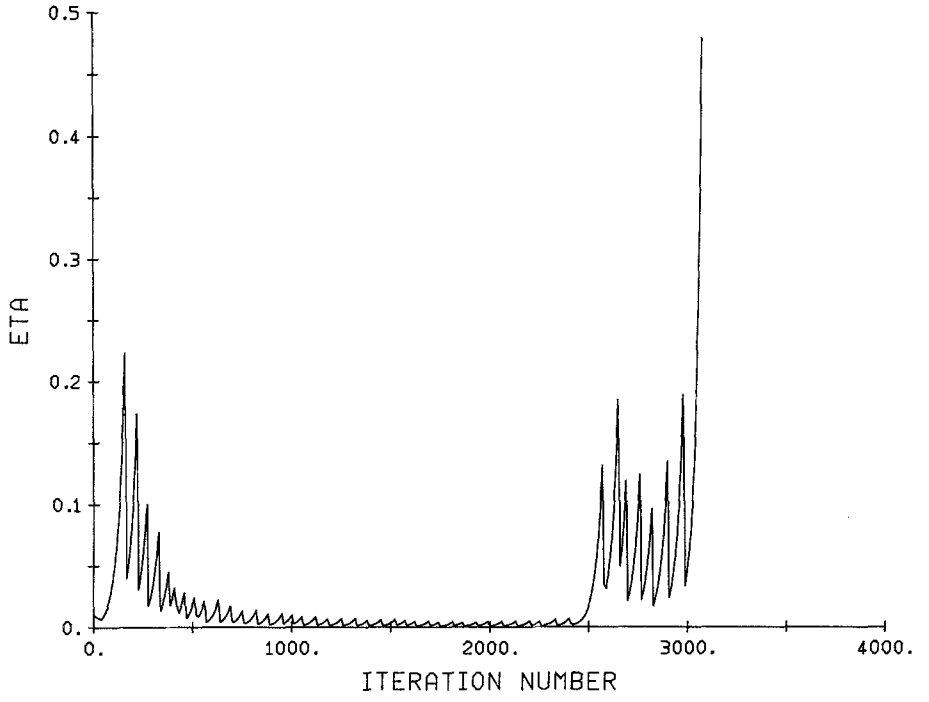

Fig. 5. Learning rate behavior for the 21 patterns problem

to a network with a 9×13 input field incorporating a 5×7 receptive field, a hidden layer also of 9×13 units, a second hidden layer of 24 units, and 6 output units. Figure 5 shows that the system starts with relatively large values of η (about 0.1) that become smaller for the intermediate steps (about 0.005), and then increase again to larger values at the end (about 0.08).

In order to separate the effects of varying η from the effects of replacing the acceptance of each step by the composite step resulting from summing over all the input patterns, β and ϕ were set to 1.0, thus effectively locking η to a constant value; the convention of setting alpha equal to zero for unsuccessful steps remained in force. The value of η was then varied between 0.05 and 0.3 for a simple $T-C$ problem without rotational or translational independence. This problem was chosen to keep the computation time for multiple runs within reasonable bounds. The resulting curve is shown in Fig. 6. With variable η, 137 iterations were required for convergence, a value that intersects at the knee of the curve. Since many experimental computer runs are required to establish the optimum η for each problem, the variable η method makes significant contributions to efficiency. In particular, it should be noted that the lowest number of iterations on this problem was 80, achieved at a $\eta = 0.261$. At $\eta \geqq 0.263$ no convergence was obtained.

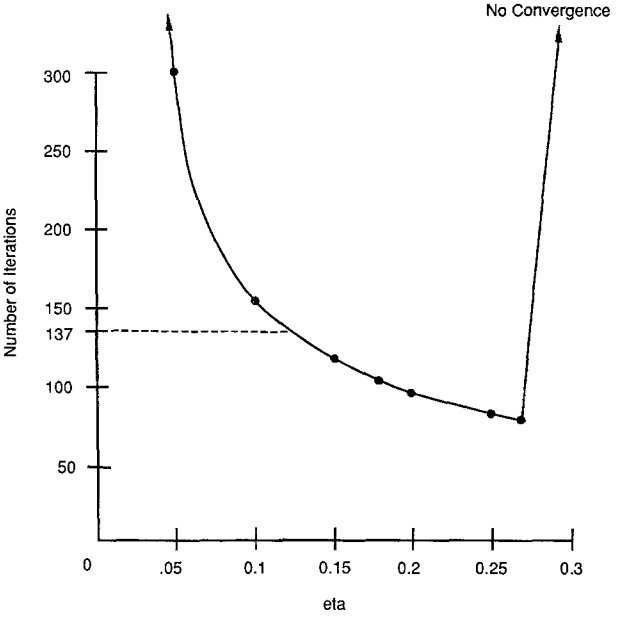

Fig. 6. The number of iterations required for convergence at constant η accepting only composite changes representing the sum of changes from all individual patterns. Use of the variable η scheme described in this paper required 137 iterations for convergence, i.e., a point at the knee of the curve. Value of $\eta > 0.261$ produced no convergence after thousands of iterations

Discussion

The requirements of realistic neural nets to be composed of extremely large numbers of individual elements implicitly demand attention to computational efficiency. We show that simple changes in the details of the back propagation algorithm can significantly improve convergence, at least for a class of problems of moderate complexity.

In any area of study, problems arise that are of theoretical interest and intrinsically worthy of solution, but that are not necessarily relevant to the

practical application of the theory. The search for a global solution is a conspicuous example of an idea that falls into this category. Global optimization has a large body of literature and several classes of computational methods have been proposed. These include: *(1)* Methods depending upon special properties of the problem; for example convexity and methods for more general problems derived from this property. See, for example, the survey by Pardalos and Rosen (1986); *(2)* Exhaustive search by interval computation (Walster et al. 1985) in the continuously differentiable case and implicit enumeration for large-scale discrete problems; *(3)* Continuation methods that depend on a systematic deformation of the problem, (reviewed by Watson 1986); *(4)* Tunneling methods that find local optima, then find a passage through the wall of the local well to a neighboring sidehill (Levy and Gomez 1985); and *(5)* Stochastic methods such as random adaptive search, multiple restart methods, and simulated annealing that generally are convergent in probability (Rinnooy-Kan and Timmer 1986; Lundy and Mees 1986). Of these, only the first two supply any assurance that the global solution has indeed been found, although the other classes of algorithms can produce an extremum that would convince a jury (beyond reasonable doubt) but fail to satisfy the mathematical guarantee of being global.

In many practical situations, it is not important that the solution be global or even a local extremum. The purpose of the optimization algorithm is to guide the search for a feasible solution point at which some prescribed criterion is satisfied, generally that some measure of error is below a given tolerance. Another reason for eschewing the search for the Holy Grail of a global optimum is that the mathematical model may describe very well the gross characteristics of the system under study, but does not possess the fine detail needed for its global optimum to be practically meaningful. Furthermore, the cost of finding the global solution may not be justified by any subsequent advantage of having obtained it. In the context of the network learning model, satisfaction of the learning criterion requires a successful search for a threshold point, as mentioned above, not the point of minimal error. One might even conjecture that a global minimum may be undesirable. After a system has learned its lesson, a portion of the network may be destroyed. However, when any additional constraint or reduction of size (damage to the network) is imposed upon an optimal feasible solution of a nonlinear program, the most likely consequence is that it is no longer feasible or optimal. Commencing a relearning at a point that satisfied the original constraints and threshold and continuing the search for a satisfactory feasible point from there, makes lesser demands on the optimization algorithm than asking it to recover from a superoptimal extreme point.

Related to this last point is the importance of commencing a search from a propitious starting point. Iterative procedures for solving problems arising in the more traditional areas of science and engineering often depend on a good first guess for arriving at an acceptable solution. This initial guess is usually provided by the user and is most commonly based upon a history of similar problems. In a new area, such as adaptive neural networks, it is perhaps more important to develop a repertoire of suitable first guesses, than it is to be concerned about the global versus the local nature of the solution. A technique for locating the knee of the "Nebraska" curves (Figs. 3 and 4) would be most useful.

These results show how much remains to be learned about the topography of the optimization space and the effects on convergence and performance of varying the connectivity of the elements and the number of layers, both of which are clearly problem dependent. Our results also suggest that in some cases additional layers of elements markedly facilitate convergence. Clearly, a theoretical basis for the organization of elements is crucial for progress.

Perusal of the recent literature reveals an enormous body of work on neural network computing, nonlinear optimization, artificial intelligence, and on areas for their application; it also reveals a parochialism that must be overcome. For example, the 1988 ACM Computer Science Conference contains one brief paper on Parallel Distributed Processing (PDP) (Whitson and Kulkarni 1988). The associated ACM conference on Computer Science in Education also contained but one paper in the field (by the same author Whitson 1988) appealing for a closer tie between AI and PDP. In the same vein, we believe in the importance of closer ties between the computer science community and neurophysiologists in the development of more effective "neural" computer networks.

To this end, we have embarked on an effort to model the neural connections and activity of the marine mollusk, *Hermissenda crassicornis*. Extensive intersensory convergence within the neural system of this animal, as it relates to a specific associative learning task, has now been mapped in detail and the relevant biophysics at the neuronal level delineated (Alkon 1983, 1984, 1985). Consequently, a detailed model of the system will permit biological comparison with the computer model, particularly as it relates to convergence and adaptive modification, as well as enabling the exploration of questions relating to the effects of the number of neural layers as well as the detailed properties of individual neurons on the ability of the model to learn, store, and recall information.

References

Akaike H (1959) On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method. Ann Inst Statist Math 11:1–17

Alkon DL (1983) Learning in a marine snail. Scientific American (July 1984), pp 70–84

Alkon DL (1984) Calcium-mediated reduction in ionic currents: a biophysical memory trace. Science 226:1037–1945

Alkon DL (1985) Conditioning – induced changes of *Hermissenda* channels: relevance to mammalian brain function. In: Weinberger NM, McGaugh JL, Lynch G (eds) Memory systems of the brain. The Guiford Press, New York

Levy AV, Gomez S (1985) The tunneling method applied to global optimization. In: Boggs PT, Byrd RH, Schnabel RB (eds) Numerical optimization 1984. SIAM, Philadelphia, pp 213–244

Luenberger DG (1984) Linear and nonlinear programming, 2nd ed. Addison-Wesley, Reading, Mass

Lundy M, Mees A (1986) Convergence of an annealing algorithm. Math Prog 34:111–124

Muneer T (1988) Comparison of optimization methods for non-linear least squares minimization. Int J Math Educ Sci Tech 19:192–197

Pardalos PM, Rosen JB (1986) Methods for global concave minimization: a bibliographic survey. SIAM Rev 28:367–379

Parker DB (1987) Optimal algorithms for adaptive networks: second order back propagation, second order direct propagation, and second order Hebbian learning. In: Caudill M, Butler C (eds) Proceedings of the 1st International Conference on Neural Networks, San Diego, Calif., June 1987. IEEE Cat. #87TH0191-7, pp II-593–II-600

Pegis RJ, Grey DS, Vogl TP, Rigler AK (1966) The generalized orthonormal optimization program and its applications. In: Lavi A, Vogl TP (eds) Recent advances in optimization techniques, Wiley, New York, pp 47–60

Pineda FJ (1987) Generalization of back propagation to recurrent and higher order neural networks. Proceedings of the IEEE Conference on Neural Information Processing Systems, Denver, Colorado 1987: (to be published)

Rinnooy-Kan AHG, Timmer GT (1985) A stochastic approach to global optimization. In: Boggs PT, Byrd RH, Schnabel RB (eds) Numerical optimization 1984. SIAM, Philadelphia, pp 245–262

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representation by error propagation. In: Rumelhart DE, McClelland JL and the PDP Research Group (eds) Parallel distributed processing, vol 1, chap 8. MIT Press, Cambridge, Mass

Walster GW, Hansen ER, Sengupta S (1985) Test results for a global optimization algorithm. In: Boggs PT, Byrd RH, Schnabel RB (eds) Numerical optimization 1984. SIAM, Philadelphia, pp 272–287

Watson LT (1986) Numerical linear algebra aspects of globally convergent homotopy methods. SIAM Rev 28:529–545

Whitson GM (1988) An introduction the the parallel distributed processing model of cognition and some examples of how it is changing the teaching of artificial intelligence. In: Dreshem HL (ed) Proceedings of the 19th Annual Technical Symposium on Comp Sci Education. ACM, New York, pp 59–62

Whitson GM, Kulkarni A (1988) A testbed for sensory PDP models. Proceedings of the 16th Annual Comp Sci Conf. ACM, New York, pp 467–468

Received: October 22, 1987

Dr. Thomas P. Vogl
Environmental Research Institute of Michigan
1501 Wilson Boulevard, Suite 1105
Arlington, VA 22209
USA