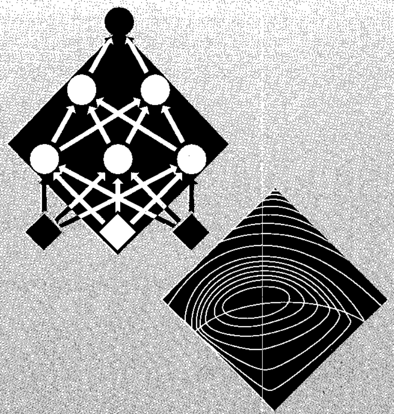

Fuzzy Parameter Adaptation in Optimization:

Some Neural Net Training Examples

Payman Arabshahi, *University of Alabama in Huntsville*
Jai J. Choi, *Boeing Computer Services*
Robert J. Marks II, *University of Washington*
Thomas P. Caudell, *University of New Mexico*

Parameters of certain neural net training algorithms and classification procedures are often chosen or adapted using heuristics that contain fuzzy descriptors such as "smooth," "steep," and "smaller." Such heuristics, quantified into a fuzzy inference engine, can take [illegible]

MANY NONLINEAR OPTIMIZATION ALGORITHMS, INCLUDING THOSE used to train various types of artificial neural networks, strive to optimize some performance measure through judicious selection of one or more parameters. For instance, in the backpropagation-trained multilayer perceptron,[1] the performance measure is convergence speed. This speed is affected by the choice of learning and momentum parameters. Similarly, in the Adaptive Resonance Theory (ART 1) network,[2] the choice of a vigilance parameter affects the number of classes into which the data are classified. The values of these parameters can be adapted during training to improve the performance measure(s) of the neural network. Table 1 summarizes the performance measures and parameters associated with several neural net architectures. The theme introduction on pp. 36–42 provides some background on these various methods.

Training parameters are typically chosen and adapted by a "neural smith," using human judgment, experience, and heuristic rules. For example, a smooth error surface in the backpropagation training of a layered perceptron suggests use of a long step, whereas a steep surface suggests smaller steps. Note that this description is fuzzy: the terms "smooth," "long," "steep," and "smaller" are each fuzzy linguistic variables.

Rather than choosing and optimizing these parameters manually, however, we take advantage of the fact that the linguistic variables used in human judgment can in many cases be quantified into a rule-based fuzzy inference engine. This fuzzy controller then replaces the neural smith. This methodology for choosing training parameters can be applied to other neural networks, including Kohonen's self-organizing maps[3] and layered perceptrons trained by other methods, such as random search.[4] But beyond neural nets, this research has led us to adopt the principles of fuzzy logic in a way that can potentially be broadly applied to a wide variety of algorithms used in adaptation and optimization.

1070-9924/96/$5.00 © 1996 IEEE

Table 1. Performance measures and parameters for different neural architectures.

Training / Architecture	Performance measure	Parameter(s)
Backpropagation/Multilayer perceptron	Convergence speed	Learning parameter η
		Momentum parameter α
ART 1	Number of classes	Vigilance parameter ρ
Random search/Multilayer perceptron	Convergence speed	Search variance σ
Kohonen	Convergence and labeling error	Learning rate α

Fuzzy parameter adaptation

A fuzzy controller consists of a set of fuzzy implications of the type "If A Then B." Consider for example the case of a single-input, single-output system and suppose there are N such implications. Each of these rules associates a fuzzy input subset to a fuzzy output subset, represented by their membership functions. Fuzzy set theory can be used to "quantify" such rule-based descriptions, and can serve as an interface between the imprecise descriptive nature of control and the control actions that need to be taken.

How does this fuzzy controller work? It takes as its input the difference between the desired and actual performance measures of the network, forming an error measure. In backpropagation learning, the actual output of the network is compared to the target network output, forming the error. In ART 1, the network classifies the input data space into various classes. The number of classes actually created is then compared to a desired number of classes, forming an error measure.

The on-line fuzzy controller uses this value and its rules for parameter adjustment to adapt the value of the parameters. This in turn changes the network output and performance attribute, resulting in a new error value which is fed back to the controller. By adaptively updating the parameter in this way, an improved value of the performance attribute results. Details of operation of fuzzy controllers are given elsewhere.[5]

Fuzzy control of backpropagation

The backpropagation learning algorithm[1] has been successfully applied to the training of multilayer feedforward neural networks in a number of practical problems. Although many arguably better training methods exist, BP has the advantages of (1) being performed totally within the neural network structure and (2) intense popularity. Backpropagation is a gradient-descent search in the space of weights of the neural network, and aims to minimize an energy function which is normally defined as the sum of squared errors. Each "error" is the difference between desired (target) values at the output of the network and actual values obtained during each iteration of the algorithm. Weight changes are performed according to

$$\Delta \mathbf{w}_n = -\eta \nabla E(\mathbf{w}_n) + \alpha \Delta \mathbf{w}_{n-1} \tag{1}$$

where $\mathbf{w}_n$ is the vector of weight values after the nth iteration; $\Delta\mathbf{w}_n$ is the change in these weights; $\nabla E(\mathbf{w}_n)$ is the gradient of the error function $E(\mathbf{w}_n)$ at the nth iteration; η is the learning rate (or step size); and α is the momentum parameter.

As mentioned, the error function E is normally defined as the sum of squared output errors,

$$E(\mathbf{w}) = \sum_{i=1}^{P} \sum_{k=1}^{L} \left(t_{ki} - y_{ki}\right)^2 \tag{2}$$

where t_{ki} and y_{ki} are the target and actual outputs of the kth output unit for the ith training vector, respectively. L represents the total number of output units, and P the total number of training vectors per epoch. This error, used to adapt the weights of the layered perceptron, can also be used to adapt the parameters—the step size and momentum of the BP algorithm.

Despite the effectiveness of BP, its speed of convergence can be painfully slow. It has furthermore been shown that many feedforward neural net training problems may not be solvable by higher-order or more sophisticated optimization methods,[6] thus increasing the need for overcoming the speed limitations of BP.

Reasons for the slow convergence of BP have been discussed in detail by Jacobs.[7] Jacobs also presented heuristics for accelerating this convergence: he suggested that each weight be given its own learning rate, and that this learning rate be allowed to change over time during the learning process. He also suggested how the learning rate should be adjusted, and incorporated these heuristics into the *delta-bar-delta rule*. This says that if the error gradient possesses the same sign over several consecutive time steps, the value of η should be increased; and if the sign of ∇E alternates over consecutive time steps, the value of η should be decreased. Other acceleration techniques have also been proposed.[8–10]

There are still no general guidelines for choos-

Table 2. In a fuzzy controller, rules are coded in the form of a decision table. Each entry here represents the value of the fuzzy variable $\Delta\eta$ for given values of error (E) and change in error (CE). NS is negative small, ZE is near zero, and PS is positive small. For example, if the error is low and the change in error is high, then the incremental update to the learning rate is near zero.

E \ CE	Low	Medium	High
Low	PS	PS	ZE
Medium	PS	PS	ZE
High	ZE	ZE	NS

ing specific (fixed) values of η and α that give fast convergence. Fuzzy control of the learning rate η, although a straightforward procedure, can be remarkably effective as a solution to this problem.[11]

The central idea behind fuzzy control of BP is the implementation of heuristics in terms of fuzzy If–Then rules. This is done by considering the error E and the change in error $CE = E_n - E_{n-1}$ (where n is the iteration number) to be variables with values E_n and CE_n at iteration n. These values can in turn be categorized as *low*, *medium*, or *high*. We define a variable $\Delta\eta$, which takes on values of $\Delta\eta_n$ at each iteration n, as the incremental update to the learning rate η. This change in the learning rate can take on the values *negative small* (NS), *near zero* (ZE), and *positive small* (PS). All of these values are expressed in terms of membership functions, as shown in Figure 1. Based on the actual (crisp) values of E and CE, we can thus arrive at a crisp value for $\Delta\eta$, if we can express the relationship among these variables through fuzzy conditional statements. During each iteration of the BP algorithm, therefore, the value of the learning rate η is incremented by $\Delta\eta_n$ based on current values of the error and change in error. The incremental updates to η may thus be different in every iteration.

Evaluation of rules is best illustrated by an example. The rules chosen for fuzzy control of BP are shown in Table 2. Consider the following rule, which can be read from the table:

If E is high, and CE is low, then $\Delta\eta$ is near zero.

Having obtained (crisp) values for E and CE at the nth iteration, we evaluate their degree of membership in the membership functions of fuzzy sets defining their "values" (*high* and *low* in this case, as shown in Figure 1). The minimum value of these two evaluations is chosen and multiplied by the membership function of the consequent fuzzy set (*near zero* in our case), resulting in a *modified*

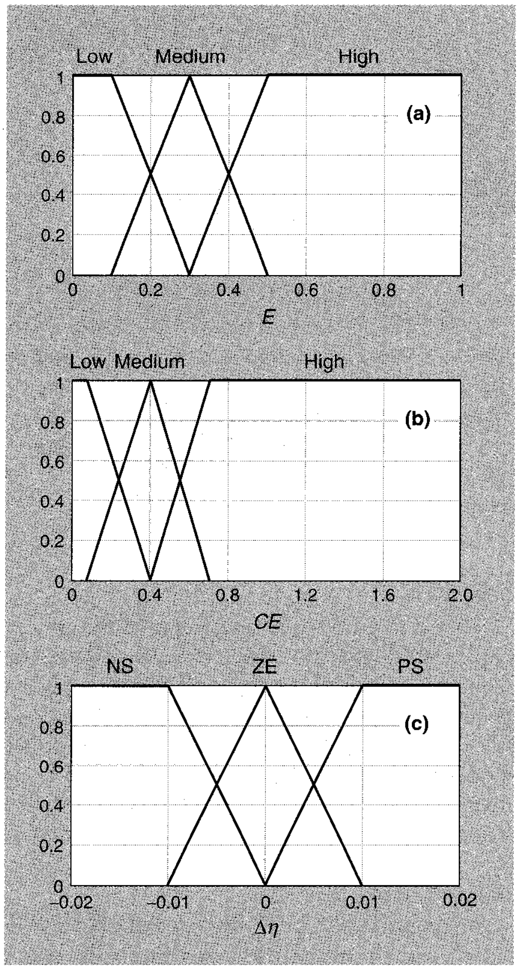

Figure 1. Membership functions in fuzzy sets for several variables involved in training a neural net with backpropagation. Functions are shown for (a) the error E; (b) the change in error CE; and (c) $\Delta\eta$, the incremental update to the learning rate parameter η.

membership function which we choose to represent by $\mu_k(x)$ for rule k, where x is the actual numerical value of either the error or the change in error. This is repeated for every rule in the rule base, and the modified membership functions $\mu_k(x)$ are summed together to form a composite function $\mu(x)$. The centroid of $\mu(x)$ is then chosen as the deterministic incremental update to η:

$$\Delta\eta_n = \frac{\int x\,\mu(x)\,dx}{\int \mu(x)\,dx} \tag{3}$$

There is also a computationally less intensive way of evaluating the centroid. The mathematics is trivial and we refer the interested reader elsewhere[5] for details.

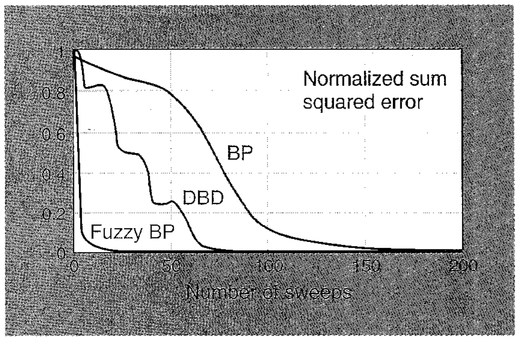

Figure 2. Fuzzy adaptation of the learning parameter speeds up backpropagation-based neural net learning of the three-bit parity problem. Shown are typical speeds of error convergence for plain BP, Jacobs's delta-bar-delta rule, and fuzzy-controlled BP.

Table 3. The contents of this fuzzy controller decision table represent the value of the fuzzy variable $\Delta\alpha$ for a given choice of values for the change in the error gradient (*CE*) and for the change in this change (*CCE*). "—" denotes no adaptation. The maximum value that α can take on is set to 1.

CCE \ *CE*	NB	NS	ZE	PS	PB
NB	–.01	–.01	—	—	—
NS	–.01	—	—	—	—
ZE	—	.01	.01	.01	—
PS	—	—	—	—	–.01
PB	—	—	—	–.01	–.01

Results

To test this approach we compared the convergence speed of standard BP, of Jacobs's delta-bar-delta rule,[7] and of fuzzy-controlled BP, as applied to a three-bit parity problem.[11] In this simple problem, the neural net is trained to ouput 1 if the 1's parity is even, and 0 otherwise. Figure 2 shows that the fuzzy-controlled BP algorithm is much faster than either of the other two methods for this example.

Improving the heuristics

As mentioned earlier, the central idea behind fuzzy control of the BP algorithm is the use of fuzzy If-Then rules that lead to faster convergence. The heuristics we used in the experiment above are mostly those of Jacobs. They are driven by the behavior of the error E (see Equation 2).

To improve the heuristics we set up rules that not only take into account the first derivative of the error, but the second derivative as well.[12] Here the change of error *CE* is an approximation of the gradient, and the change of *CE* (*CCE*) is a second-order gradient. In addition to the learning rate, the momentum parameter is controlled as well (see Table 3). The momentum parameter is used to give some momentum to each weight change so that learning accelerates in the average downhill direction instead of fluctuating with every change in the sign of the associated gradient. There are three rules:

- ♦ If *CE* is small with no sign changes in several consecutive time steps, then the value of the learning parameter should be increased.
- ♦ If sign changes occur in *CE* for several consecutive time steps, then the value of the learning parameter should be reduced with no regard to the value of *CCE*.
- ♦ If *CE* is very small and *CCE* is very small, with no sign change for several consecutive time steps, then the value of the learning parameter η as well as the momentum gain α should be increased.

Notice that the sign change of the gradient is identical to the sign change of *CE*. For example, if $E_{n-1} \le E_n$ and $E_n > E_{n+1}$ then $CE_n = E_n - E_{n-1} \ge 0$ and $CE_{n+1} = E_{n+1} - E_n < 0$. This means there has been a sign change enroute from the $(n-1)$th iteration to the $(n+1)$th iteration. Let us therefore introduce the sign change parameter,

$$SC_n = 1 - \left| \tfrac{1}{2}\left(\operatorname{sgn}(CE_{n-1}) + \operatorname{sgn}(CE_n)\right) \right| \qquad (4)$$

where the hard limiter $\operatorname{sgn}(x)$ equals 1 if $x \ge 0$ and is -1 otherwise. The factor 1/2 is to ensure that *SC* is either 0 (no sign change) or 1 (one sign change). The cumulative sum of *SC* (or *CSC*) thus can reflect the history of the sign changes, that is,

$$CSC_n = SC_n + SC_{n-1} + SC_{n-2} + \cdots$$

The bigger the *CSC*, the more frequently the sign changes have occurred. We use a five-step tracking of the sign changes, and thus define

$$CSC_n = \sum_{m=n-4}^{n} SC_m$$

The heuristic rules are shown in Table 4. The fuzzy sets defined on *CE* and *CCE*—NS, ZE, PS, NB (negative big), and PB (positive big)—are defined by their membership functions in Figure 3. From this table, for instance, one can read the following rule:

If *CE* is negative small, and *CCE* is near zero, then $\Delta\eta$ is positive small.

The domain for both *CE* and *CCE* is [–0.3, 0.3]. Values outside of these limits are clamped to –0.3 and 0.3, respectively.

Table 4. A decision table for a fuzzy controller with improved heuristics. Table contents represent the value of the fuzzy variable $\Delta\eta$ for a given choice of values for *CE* and *CCE*.

CCE \ *CE*	NB	NS	ZE	PS	PB
NB	NS	NS	NS	NS	NS
NS	NS	ZE	PS	ZE	NS
ZE	ZE	PS	PS	PS	ZE
PS	NS	ZE	PS	ZE	NS
PB	NS	NS	NS	NS	NS

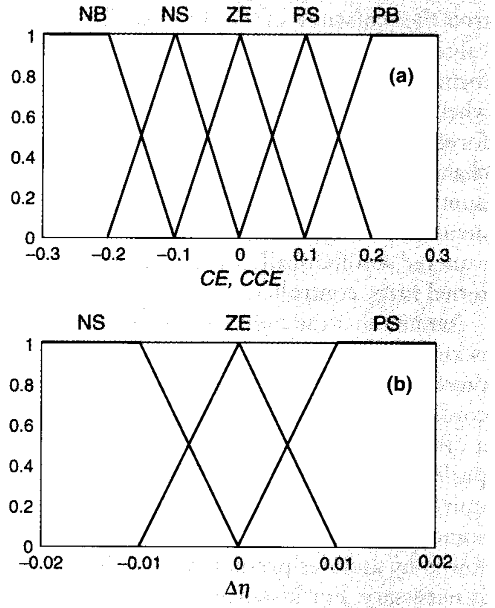

Figure 3. Membership functions in fuzzy sets for the improved heuristics. Shown are functions for *CE* and *CCE*; and $\Delta\eta$. NB is negative big, NS is negative small, ZE is near zero, PS is positive small, and PB is positive big.

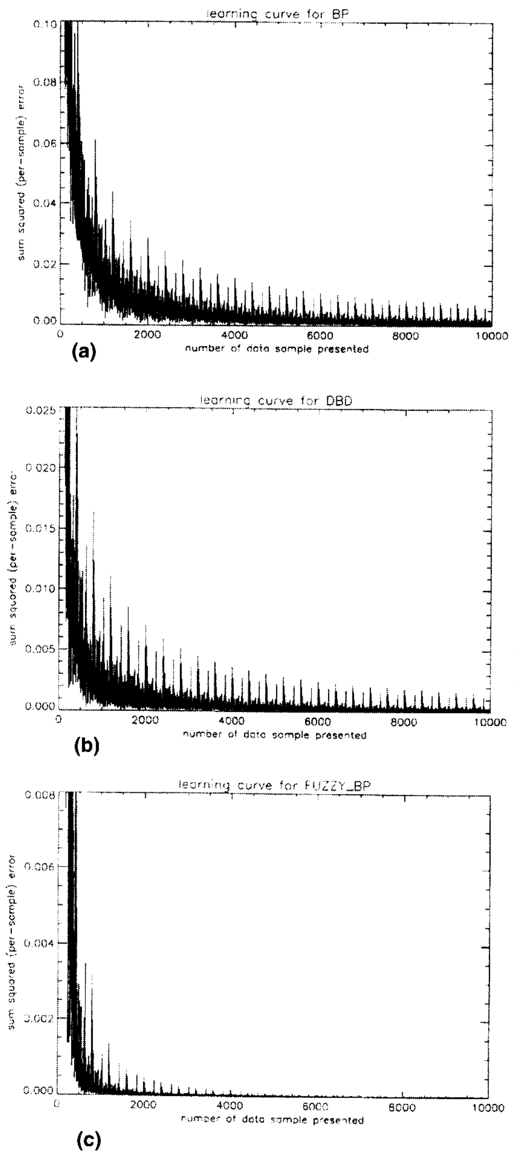

Figure 4. A typical learning curve for the Laplace noise detection problem with four input neurons, one output, one 15-unit hidden layer, and 800 training data: (a) using regular BP, $\eta = 0.9$, and $\alpha = 0.5$; (b) using Jacobs's DBD rule, initial $\eta = 0.9$, and $\alpha = 0.5$; (c) using fuzzy BP, initial $\eta = 0.9$, and initial $\alpha = 0.5$. Fuzzy control of backpropagation results in dramatically accelerated convergence. Note that the scales of the plots differ.

Results

We compared regular BP, Jacobs's delta-bar-delta rule, and fuzzy BP with the improved heuristics just discussed, as applied to a detection problem: We trained a neural network to distinguish between Laplace noise and a constant signal corrupted with Laplace noise.[13] (In this problem, the layered perceptron is taught to distinguish between Laplace noise **n** and a constant signal s corrupted by Laplace noise (that is, s + **n**). The probability density function for Laplace noise is $(\gamma/2)e^{-\gamma|n|}$. In training, the value of noise is changed for each weight update.)

Typical training curves for the three methods are shown in Figure 4. Fuzzy BP results in dramatically faster convergence, and has a significantly smaller "tail" than regular BP. Note that the scales of the plots differ. After just 500 iterations fuzzy BP reaches an error value of .0006 whereas error values for DBD and regular BP are .005 and .035 respectively.

Although we have not explicitly implemented Jacobs's heuristics by a fuzzy control system, nonetheless the control rules can be interpreted as being derived from the general guidelines he proposed.[7] While our approach in this implementation differs from that of Jacobs's delta-bar-delta rule, it is only natural to assume that whenever Jacobs's heuristics fail in a specific problem, the same will happen with the fuzzy BP technique.

Also, just as BP cannot guarantee convergence to a globally minimum solution, neither can fuzzy-controlled BP. This is a problem inherent to a localized optimization technique such as steepest descent, of which backpropagation is a special case.

Furthermore, although a smaller number of iterations towards convergence is certainly desirable, this is not the only important consideration. This aspect of our solution should be considered jointly with the total number of operations required for each iteration. In this respect, however, the total number of operations of the two techniques (plain and fuzzy BP) are not significantly different from each other, owing to the inherent simplicity of the computations carried out in the fuzzy controller.

Fuzzy control of learning in ART 1 neural nets

ART 1 is a biologically inspired neural net for unsupervised clustering. Here is a real-world application appropriate for an ART 1 network: sheet metal parts for aircraft are clustered into groups with similar geometry to allow the identification of reusable engineering. The parts are represented as a two-dimensional pixel map, or silhouette, with 1's where there is material and 0's otherwise. The pixel map is strung into a binary vector and fed to the ART 1 network for recognition. Given a fixed number of classes into which we wish to classify the parts, the fuzzy vigilance controller always finds the appropriate vigilance parameter.

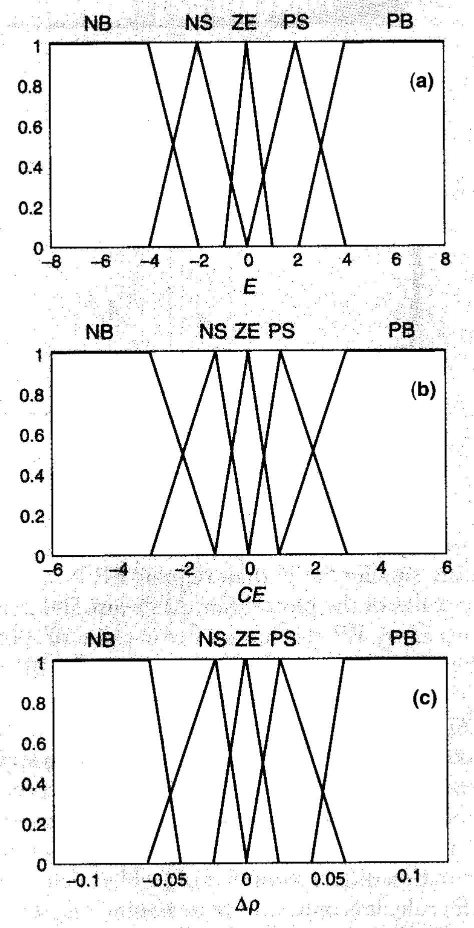

Figure 5. Membership functions for fuzzy sets for several variables involved in training an ART 1 network. Shown are functions for E, CE, and $\Delta\rho$.

In the Adaptive Resonance Theory model[2] the number of clusters formed depends directly on the value of the vigilance parameter ρ, which controls the coarseness of the clusters. The higher the value of ρ, the greater the number of clusters formed to represent the input data. In applications where the number of desired clusters is known beforehand, the value of ρ can be incrementally changed so that complete classification into a fixed number of clusters is achieved. We do this incremental updating of the value of the vigilance parameter automatically by means of a simple external fuzzy controller.

Assume that the number of clusters desired N_d is known a priori. The set of all input vectors is presented to the ART 1 network and classified accordingly into N_a (actual) classes. We wish to find a value for ρ that gives us N_d clusters. Since this preliminary classification does not meet the requirement that N_d clusters should be formed, some adjustment of the vigilance parameter followed by another presentation of the input data is necessary. For instance, if $N_d < N_a$, a decrease in the value of ρ would be desired. If $N_d > N_a$, then we would want to increase ρ. The question naturally arises as to the magnitude of the change in ρ that would make $N_a = N_d$ in a single sweep after the initial classification, or let N_a approach N_d gradually after multiple sweeps.

If one chooses to change ρ in small increments of $\Delta\rho$, then a simple fuzzy controller can be used to arrive at an optimum value for ρ. The controller will seek to regulate the value of the vigilance parameter based on how far we are from achieving N_d classes, having started with N_a clusters. Its adaptation policy can be formulated as a set of heuristic rules of the form shown in Table 5.

Following the controller's updating of the value of ρ by $\Delta\rho$, the ART 1 network performs a second classification with this new value for ρ. If the resulting number of classes is again unsatisfactory, we repeat the process. Eventually, the number of classes N_a formed by the network will approach and become equal to N_d, and we are done.

The fuzzy values shown in Table 5 that E and CE can take on (NB, NS, ZE, PS, PB) are defined

Table 5. A decision table for an ART 1 fuzzy controller. Table contents represent the value of the fuzzy variable $\Delta\rho$, the change in the vigilance parameter, for given values of error $E = N_d - N_a$ and change in error CE.

CE \ E	NB	NS	ZE	PS	PB
NB	—	NS	ZE	PS	—
NS	NB	NS	ZE	PS	PB
ZE	NB	NS	ZE	PS	PB
PS	NB	NS	ZE	PS	PB
PB	—	NS	ZE	PS	—

in terms of their membership functions in Figure 5. Here $E = N_d - N_a$ and CE is the change in E.

Results

We tested the control of the number of clusters in an ART 1 network using data from our earlier work,[14] which involved the sheet metal example mentioned earlier. Shown in Figure 6 is the number of groups as a function of the vigilance parameter ρ, empirically determined from a set of 47 input designs. Order differences account for the nonmonotonic behavior of the curve.

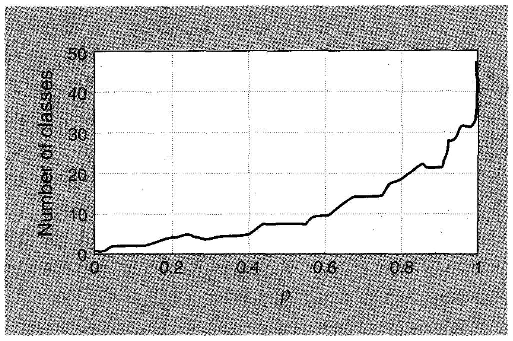

Figure 6. Fuzzy control of the vigilance parameter ρ in training an ART 1 neural net allows the system to classify a set of 47 sheet-metal parts into any desired number of shape-related classes.

Fuzzy control of hierarchical ART

We present here results of applying the fuzzy control technique for ART 1 to a hierarchical ART tree network.[14] This network consists of a tree of ART networks with different levels of abstraction at each level in the tree. Lower levels in the tree have lower ρ while the higher levels have higher ρ. The data patterns clustered by lower networks are treated as the training sets for the next higher networks. Therefore the number of ART networks at one level equals the total number of clusters produced by the immediate lower level in the tree. Different applications may need to control the branching ratios in such a tree structure, that is, the number of clusters per level. The number of clusters produced by an ART network is related to the vigilance parameter ρ, and therefore it is necessary to servo the number of clusters by controlling this parameter.

For example, we consider the problem of a four-level tree with the number of clusters per level equal to N_d (2, 4, 8, 16). The data space consists of the letters of the English alphabet, and the clusters represent groupings of similar-looking letters. Figures 7a and 7b illustrate the different clusterings; ρ_f is the final value of the vigilance parameter ρ at each level which gave the correct number of clusters. The difference in clustering is due to different training data initialization.

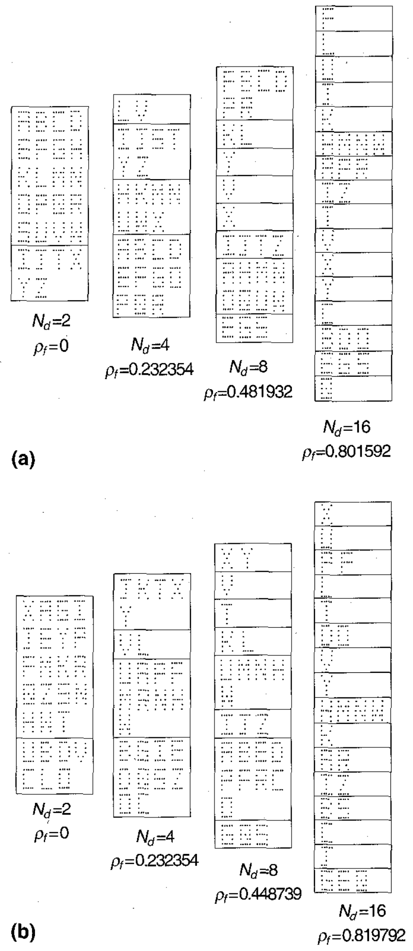

Figure 7. (a) Progressive classification of the letters of the alphabet into different clusters, using fuzzy control of the vigilance parameter in a hierarchical ART network. (b) Same network but with a different initialization.

Fuzzy control of random optimization

Random optimization[15] is an alternative to BP for training of multilayer perceptrons.[16] This technique has the same aim as BP: to minimize an energy function defined as a sum of squared errors in the space of weights of the neural network. Random optimization does this by taking steps in the space of weights, evaluating the energy function at those points, and adjusting the steps taken so as to eventually converge (in probability) to the *global* minimum of the error surface. This means that the probability of the search eventually reaching the global minimum is 1. In the bidirectional random optimization technique, two points are chosen in the vicinity of the starting point. The location of the points is governed by the variance of the random number generator: the higher the value of the variance, the bigger the steps taken. Recall from Table 1 that our desire is to control the variance in order to achieve fast training.

One of the disadvantages of using random-search techniques[16] for optimization problems is their slow convergence speed. In attempting to improve this, the random-search step size (variance) can be regulated via simple fuzzy rules. Similar to the gradient-search (BP) case, once random search is successful for several consecutive trials, one can increase the search step size. Through fuzzy control of step size, the search toward the global minima of the objective function can be accelerated.

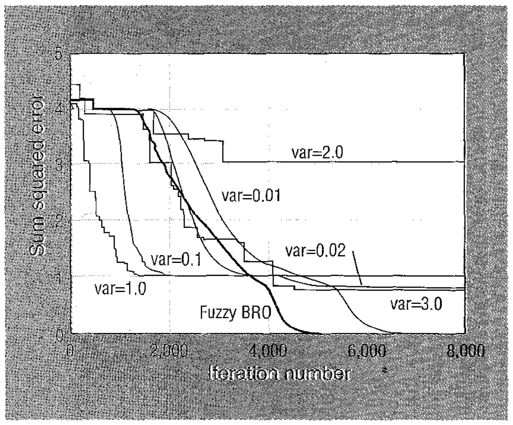

Figure 8. Fuzzy control of neural net training by random optimization: the four-bit parity problem. Fuzzy bidirectional random optimization (BRO) results in better and faster convergence than regular BRO with different fixed choices for the variance. The variance in fuzzy BRO is adaptive; it changes from one iteration to another.

The success of the search greatly depends on the choice of random initialization. Figure 8 presents simulation results for a four-bit parity problem with a specific random-seed initialization. This example showed faster convergence. In the simulation, we applied the bidirectional random optimization algorithm,[15,16] a specific manifestation of random optimization, to a multilayer perceptron.

Our result does not necessarily reflect a general trend. Depending on the initialization, the results vary and are at times unstable.

Fuzzy Kohonen networks

Tsao, Bezdek, and Pal[3] have combined the ideas of fuzzy membership values for learning rates, the parallelism of the fuzzy c-means algorithm,[17] and the structure and update rules of Kohonen networks into a new family of neural networks, called fuzzy Kohonen clustering networks. KCNs are heuristic procedures with the following properties:

- Termination of clustering is not based on optimizing any model of the process or its data.
- Final weight vectors usually depend on the input sequence.
- Different initial conditions usually yield different results.
- Several parameters such as the learning rate and the size of the update neighborhood must be varied from one data set to another, as well as during learning, to achieve the best results.

The integrated approach of Tsao et al. is based on a new learning rate: $\alpha_{ik,n}=(\mu_{ik,n})^{m_n}$, where $m_n = (m_0 - 1)/n_{\max}$ and $\mu_{ik,n}$ is the value of the membership of vector $\mathbf{x}_k$ (member of a set X of feature vectors that are to be clustered) in membership function i, for a given value of n. The fuzzy subsets μ_i form a fuzzy c-partition of X.[17] Here m_0 is a positive constant greater than one, and $n_{\max}$ is the iteration limit for fuzzy clustering. The clustering is then performed via an update rule as in fuzzy c-means.

Furthermore, in addition to the learning rate, the size of the update neighborhood in the competitive layer is automatically adjusted during learning. This results in improved convergence as well as reduced labeling errors. The new algorithm is nonsequential, unsupervised, and always terminates in many less iterations, independent of the sequence of feeding data.

NEURAL NET DEVELOPERS TYPICALLY CHOOSE and refine their training parameters heuristically. Our examples have demonstrated how these heuristics can be emulated with fuzzy inference engines. These engines are algorithmically implemented to take the human out of the loop and provide for faster convergence. Other possibilities, outside of neural nets and yet untried, are fuzzy scheduling of annealing and genetic algorithms.

The astute reader will note that the fuzzy inference engine is, itself, parameterized. An obvious example is the shape (the center and width, for instance) of each fuzzy membership function. The designer of the fuzzy inference engine must thus choose parameters that control the fuzzy system. Could not a second system be designed to control the parameter choice of the fuzzy inference engine? The answer is yes, *if* an appropriate performance measure can be assigned to the first fuzzy inference engine.[18] ◆

Acknowledgments

This work was supported in part by Boeing Computer Services, the Washington Technology Center, and the Royalty Research Fund from the University of Washington.

Payman Arabshahi obtained his BSE in electrical engineering from the University of Alabama in Huntsville and his MS and PhD, both in electrical engineering, from the University of Washington. His research focuses on fuzzy systems, digital signal processing, and digital communications. He is a visiting assistant professor at the University of Alabama in Huntsville doing research in fuzzy pattern recognition and digital mobile networking, in conjunction with Neda Communications, Bellevue, Washington. Arabshahi is a member of IEEE and the editor-in-chief of the IEEE Neural Network Council's home page on the World Wide Web. He can be reached at payman@ee.washington.edu.

Jai J. Choi received his BSE and MSE in electronics engineering from Inha University, Korea, and his MSEE and PhD in electrical engineering from the University of Washington, where he is an affiliate assistant professor and graduate faculty member. In 1990 he joined Boeing Computer Services, where he is developing neural networks, fuzzy logic, and adaptive systems for real-time signal processing, intelligent diagnosis, and pattern recognition. He is also associated with the Electrical Engineering and Electronic Technology Department of Cogswell College North, Kirkland, Washington. His current research interests include neural networks, fuzzy logic, adaptive signal processing, machine monitoring, and pattern recognition. He is an associate editor for *IEEE Transactions on Neural Networks* and a member of the IEEE Circuits and Systems Society and Eta Kappa Nu. He can be reached at jai@atc.boeing.com.

Robert J. Marks II is professor of electrical engineering at the University of Washington. An IEEE Fellow, Marks was named an IEEE Distinguished Lecturer in 1992. He chaired the IEEE Neural Networks Committee in 1989 and served as the first president of the IEEE Neural Networks Council in 1990–91. He is editor-in-chief of *IEEE Transactions on Neural Networks* and associate editor of *IEEE Transactions on Fuzzy Systems* and the *Journal on Intelligent Control, Neurocomputing and Fuzzy Logic.* Marks is a member of the Board of Governors of the IEEE Circuits and Systems Society and cofounded that society's technical committee on neural systems and applications. He helped organize the International Symposium on Circuits and Systems, the International Joint Conference on Neural Networks, the IEEE World Congress on Computational Intelligence, and the IEEE/IAFE Conference on Financial Engineering, among other conferences. He owns three US patents in the field of artificial neural networks and signal processing. He can be reached at marks@ee.washington.edu.

Thomas P. Caudell is associate professor of electrical and computer engineering at the University of New Mexico, where he researches neural networks, pattern recognition, machine vision, neuroanatomy, virtual reality and augmented reality, and optical computing. Before moving to New Mexico, he was senior principal scientist and the principal investigator of Boeing Computer Services's adaptive neural systems R&D project. He received his BS in physics and mathematics from California State Polytechnic University, Pomona, and his MS and PhD in physics from the University of Arizona. Caudell is a member of the IEEE Neural Networks Council, the Association for Computing Machinery, the International Neural Network Society, and the Optical Society of America. He can be reached at tpc@eece.unm.edu.

The authors can also be reached by contacting Payman Arabshahi, Dept. of Electrical and Computer Engineering, University of Alabama, Huntsville, AL 35899.

For Further Reading

S. Haykin, *Neural Networks: A Comprehensive Foundation*, IEEE Press, Piscataway, N.J., 1994. A very readable, comprehensive, yet mathematically detailed introduction to neural computing. An excellent textbook!

R.J. Marks II, ed., *Fuzzy Logic Technology and Applications*, IEEE Press, Piscataway, N.J., 1994. Provides an overview of state-of-the-art technology in fuzzy systems.

T. Ross, *Fuzzy Logic with Engineering Applications*, McGraw-Hill, New York, 1994. Excellent introductory text on fuzzy sets and systems.

L.A. Zadeh, "Fuzzy Sets," *Information Control*, Vol.8, 1965, pp.338–353. Reprinted in *Fuzzy Models for Pattern Recognition*, J.C. Bezdek and S.K. Pal, eds., IEEE Press, Piscataway, N.J., 1992, pp.29–35. Traces the evolution of fuzzy algorithms for feature analysis, clustering and classifier design, and image processing and vision.

IEEE Neural Network Council, http://www.ieee.org/nnc. Links to neural computing research programs worldwide, NNC news, and conferences.

References

1. D.E. Rumelhart and J.L. McClelland, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, MIT Press, Boston, 1986.
2. B. Moore, "ART1 and Pattern Clustering," *Proc. 1988 Connectionist Models Summer School*, Morgan Kaufmann, San Francisco, Calif., 1989.
3. E.C-K. Tsao, J.C. Bezdek, and N.R. Pal, "Fuzzy Kohonen Clustering Networks," *Pattern Recognition*, Vol. 27, No. 5, 1994, pp. 757–764.
4. J.J. Choi, S. Oh, and R.J. Marks II, "Training Layered Perceptrons Using Low Accuracy Computations," *Proc. Int'l Joint Conf. Neural Networks*, IEEE, Piscataway, N.J., 1991, pp. 554–559.
5. D. Driankov, H. Hellendoorn, and M. Reinfrank, *An Introduction to Fuzzy Control*, Springer-Verlag, New York, 1993.
6. S. Saarinen, R. Bramley, and G. Cybenko, "Ill-Conditioning in Neural Network Training Problems," *SIAM J. Scientific Computing*, Vol. 14, No. 3, 1993, pp. 693–714.
7. R.A. Jacobs, "Increased Rates of Convergence Through Learning Rate Adaptation," *Neural Networks*, Vol. 1, 1988, pp. 295–307.
8. R. Battiti, "Accelerated Back-Propagation Learning: Two Optimization Methods," *Complex Systems*, Vol. 3, 1989, pp. 331–342.
9. T. Fukuda et al., "Neuromorphic Control: Adaptation and Learning," *IEEE Trans. Industrial Electronics*, Vol. 39, No. 6, Dec. 1992, pp. 497–503.
10. T.P. Vogl et al., "Accelerating the Convergence of the Back-Propagation Method," *Biological Cybernetics*, Vol. 59, 1988, pp. 257–263.
11. P. Arabshahi et al., "Fuzzy Control of Backpropagation," *Proc. First IEEE Int'l Conf. Fuzzy Systems, FUZZ-IEEE '92*, IEEE, Piscataway, N.J., 1992, pp. 967–972.
12. J.J. Choi et al., "Fuzzy Parameter Adaptation in Neural Systems," *Proc. Int'l Joint Conf. Neural Networks*, IEEE, Piscataway, N.J., 1992, pp. 232–238.
13. R.J. Marks II et al., "Detection in Laplace Noise," *IEEE Trans. Aerospace and Electronic Systems*, Vol. 14, 1978, No. 6, pp. 866–872.
14. T.P. Caudell et al., "NIRS: Large Scale ART-1 Neural Architectures for Engineering Design Retrieval," *Neural Networks*, Vol. 7, No. 9, 1994, pp. 1,339–1,350.
15. F.J. Solis and J.B. Wets, "Minimization by Random Search Techniques," *Math. of Operation Research*, Vol. 6, 1981, pp. 19–30.
16. N. Baba, "A New Approach for Finding the Global Minimum of Error Function of Neural Networks," *Neural Networks*, Vol. 2, 1989, pp. 367–373.
17. J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum, New York, 1981.
18. P. Arabshahi et al., " Pointer Adaptation and Pruning of Min-Max Fuzzy Inference and Estimation," to be published in *IEEE Trans. Circuits and Systems*, 1996.