[9] H.E. Kyburg, Jr., "Bayesian and non-Bayesian evidential updating," *Artificial Intell.* vol. no. 31, 3, pp. 271–294, 1987.
[10] H.T. Nguyen, "On random sets and belief functions," *J. Math. Anal. Appl.*, vol. 65, pp. 531–542, 1978.
[11] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufmann, 1988.
[12] J. Pearl, "Reasoning with belief functions: An analysis of compatibility," *Int. J. Approx. Reasoning*, vol. 4, pp. 363–389, 1990.
[13] G.C. Rota, "Theory of Möbius functions," *Z. für Wahrscheinlichkeitstheorie und Verw. Gebiete*, vol. 2, pp. 340–368, 1964.
[14] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton Univ. Press, 1976.
[15] G. Shafer, "Perspectives on the theory and practice of belief functions, *Int. J. Approx. Reasoning.*, vol. 4, pp. 323–362, 1990.
[16] P. Smets, "Belief functions and their combinations," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-12, pp. 447–458, 1990.
[17] M. Spies, "Combination of evidence with conditional objects and its application to cognitive modeling," to appear in *Conditional Logic in Expert Systems*, Goodman *et al.*, Eds. Amsterdam: North Holland, 1990.
[18] V. Strassen, "Messfehler and information," *Z. für Wahrscheinlichkeitstheorie und Verw. Gebiete*, vol. 2, pp. 273–305, 1964.
[19] L.A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets Syst.*, vol. 1 pp. 3–28, 1978.
[20] L. Zhang, "A new proof to Theorem 3.2 of Fagin and Halpern's paper," unpublished memorandum, Univ. Kansas, Business School, 1989.

Error Surfaces for Multilayer Perceptrons

Don R. Hush, Bill Horne, and John M. Salas

***Abstract*— The paper explores the characteristics of error surfaces for the multilayer perceptron neural network. These characteristics help explain why learning techniques that use hill climbing methods are so slow in these networks. They also help provide insights into techniques that may help speed learning. Several important characteristics are revealed. First, the surface has a stair-step appearance with many very flat and very steep regions. In fact, when the number of training samples is small there is often a one-to-one correspondence between individual training samples and the steps on the surface. As the number of training samples is increased the surface becomes smoother. In addition the surface has flat regions that extend to infinity in all directions making it dangerous to apply learning algorithms that perform line searches. The magnitude of gradients on the surface is found to span several orders of magnitude, strongly supporting the need for floating point representations during learning. The consequences of various weight initialization techniques are also discussed.**

I. Introduction

Multilayer perceptrons (MLP's) are arguably the most popular neural network model. They have been used successfully in a variety of information processing problems including pattern recognition, nonlinear control, and the prediction of chaotic time series. Their strength lies in their ability to perform continuous nonlinear mappings of arbitrary complexity from $\mathcal{R}^n$ to $\mathcal{J}^m$ ($\mathcal{J} = [a.b]$, usually $a = 0$ and $b = 1$) [3], [5], [8]. Their major weakness lies in their extremely slow learning rates, that is in the time complexity of training the network to perform the desired mapping. The severity of this problem is exemplified in the recent results of [1], [6], [12]. In [1] and [6] it has been shown that the problem of loading a set of training examples onto a neural network is NP-complete. Thus it is unlikely that existing algorithms (like backpropagation) can be guaranteed to learn the optimal solution in polynomial time. In [12] it has been shown that the asymptotic rate of convergence of the backpropagation algorithm is very slow, on the order of $1/t$ at best. Our goal in this paper is to reveal characteristics of the error surface that help explain why learning is so slow.

The learning problem for MLP's can be viewed as a nonlinear optimization problem in which the goal is to find the set of network weights that minimize a performance function. The performance function, which is usually a function of the network mapping errors, describes a surface in the weight space, often referred to as the error surface. Learning algorithms can be viewed as methods for searching this surface for a minimum. The complexity of the search is governed by the nature of the surface. Error surfaces for MLP's have several well known characteristics that make them difficult to search. For example they have many flat regions where learning is slow, and long narrow troughs that are flat in one direction and steep in surrounding directions [11]. In this paper we will examine these characteristics in more detail. We will also discover new characteristics that provide an even better understanding of the surfaces. In addition we will make ties between specific characteristics of the surface and corresponding characteristics of the problem being mapped onto the network.

The class of problems we are interested in are pattern recognition problems where the MLP is used as a classifier. In these problems the desired outputs of the network are usually 1's and 0's so that the desired mapping is from $\mathcal{R}^n$ to $\mathcal{J}^m$ ($\mathcal{J} = \{0.1\}$). The threshold function used in the network is the standard sigmoidal function:

$$f(\alpha) = \frac{1}{1 + e^{-\alpha}} \tag{1}$$

and the performance criterion used is the standard total-squared-error function defined here.

The MLP is trained to perform a specific task by presenting the network with examples of the desired mapping and applying a learning algorithm that adjusts the weights of the network to reproduce this mapping. More formally, the weights of the network are adjusted to minimize the total sum-of-squared-error between the actual and desired mapping at the example points. The total sum-of-squared-error criterion is given by

$$E(\mathbf{W}) = \sum_{p=1}^{P} \sum_{j=1}^{m} (d_j^p - u_j^p)^2 \tag{2}$$

where u_j^p and d_j^p are the actual and desired network outputs for pattern p. The outer sum is over the P training patterns and the inner sum is over the m nodes in the output layer; $\mathbf{W}$ is a vector containing all the weights in the network.

Many learning algorithms have been proposed for this network. Most of them are based on descent techniques that start with an initial guess and then search for the optimal weights using a recursive update of the form,

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \mu(k)\mathbf{d}(k) \tag{3}$$

where $\mathbf{d}(k)$ is a vector that determines the direction of search and $\mu(k)$ is a scalar that controls the size of the step. The most popular learning algorithm for the MLP network is the back-propagation

Manuscript received July 28, 1990; revised May 31, 1991, and February 28, 1992. This work was supported in part by Sandia National Laboratories, Albuquerque, NM, under contract number 05-8801.

The authors are with the Department of Electrical Engineering and Computer Engineering, University of New Mexico, Albuquerque, NM 87131.

IEEE Log Number 9201393.

algorithm [13,9,10]. In this algorithm the weight update is of the form

$$\mathbf{W}(k+1) = \mathbf{W}(k) - \mu\hat{\nabla}_{\mathbf{W}} \tag{4}$$

where μ is fixed and $\hat{\nabla}_{\mathbf{W}}$ is an estimate of the gradient of $E(\mathbf{W})$ with respect to $\mathbf{W}$. Often a momentum term is added to speed learning.

$$\mathbf{W}(k+1) = \mathbf{W}(k) - \mu\hat{\nabla}_{\mathbf{W}} + \alpha(\mathbf{W}(k) - \mathbf{W}(k-1)). \tag{5}$$

If we apply the gradient operator to (2) we find that the true gradient involves a summation over all of the patterns in the training set. Typically, however, estimates of the gradient that are computed from individual samples are used. To compensate for the use of gradient estimates (rather than true gradients) μ is chosen to be relatively small.

Because μ is fixed, the actual step size for each update is determined by the magnitude of $\hat{\nabla}_{\mathbf{W}}$. This means that progress on flat parts of the surface where the gradient is small will be relatively slow compared to that on steep parts of the surface where the gradient is large. Unfortunately error surfaces for the MLP tend to have a large number of flat spots. Because of this, learning with back-propagation can be very slow [4], [11]. The purpose of this paper is to help explain why learning in these networks is such a difficult task. We do this by examining the characteristics of the error surface that make it difficult to search.

II. Error Surface Characteristics

First it is important to note that the surface defined by $E(\mathbf{W})$ in (2) depends on the example patterns (training patterns), and will therefore differ from one problem to the next. Although we will be using specific problems for illustration, it is easily argued that the characteristics revealed hold true for a more general class of problems.

It is difficult to characterize $E(\mathbf{W})$ without visualizing the surface. However, visualization is difficult to do in dimensions higher than three (i.e. for networks with more than two weights), and even the smallest MLP's have more than two weights. Thus, we will use two problems for illustration. The first is a one-node network with two weights. In this problem the surface can be viewed in its entirety in three dimensions. This problem is useful not only because it can be visualized, but also because it reveals characteristics that are true of one-node problems in general. One-node problems are important because many of their characteristics are also found in MLP's. This may be due in part to the fact when viewed solely as a function of the output layer weights, error surfaces for MLP's are exactly that of a one-node network. The second problem that we use as an example is a two-layer network with 13 weights. Since it will be impossible to view this error surface directly, we will look at 2-D subspaces of the surface, that is error surfaces as a function of selected weight pairs formed by keeping the other weights in the network fixed.

The characteristics of the error surface differ depending on the nature of the training data. We begin by examining characteristics of $E(\mathbf{W})$ for the case where the two classes of training data are small in number and nonoverlapping. We then investigate the effects of overlapping the data and increasing the number of samples.

A. Visual Characteristics

The one-node problem has two weights, one is the bias (or threshold), and the other is applied to the input, a 1-dimensional data pattern. The samples that make up the training set, four from each class, are shown in Fig. 1(a) where an "X" represents a training sample from class 1 and and "O" represents a sample from class 2. The error surface for this case is shown in Fig. 1(b). It is interesting

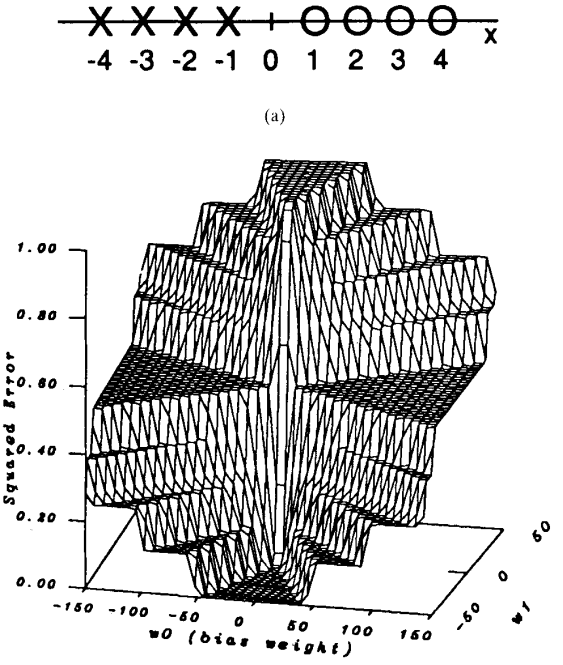

Fig. 1. Example problem 1. (a) Training samples for the one-node network. (b) The corresponding error surface.

to note that without the sigmoid nonlinearity $E(\mathbf{W})$ would be a quadratic function of the weights and the surface in Fig. 1 would be bowl-shaped. But the existence of the sigmoid nonlinearity changes the shape of the surface dramatically.

There are several characteristics of this surface worth noting. The surface has a stair-step appearance with many very steep and very flat spots. There are four large plateaus, one at the bottom, two in the middle, and one at the top. There are also four sets of stairs, two leading from the bottom plateau to the plateaus in the middle, and two continuing from the middle plateaus to the one at the top. Each feature of the surface can be explained by examining the position of the decision boundary with respect to the data in Fig. 1.

Each pair of weights determines a decision boundary in the data space (i.e. a "point" on the x-axis) in the top part of the figure. The decision boundary is given by the point

$$x^* = -w_0/w_1. \tag{6}$$

All values of x that are greater than x^* are assigned to one class and all that are less than x^* to the other. On the bottom plateau where $E(\mathbf{W})$ is at a minimum, all of the weight combinations are such that the decision boundary lies between the two classes and has the proper sign so that all samples are classified correctly. The two plateaus in the middle correspond to cases where the decision boundary lies on the far right or far left of the data so that all samples are assigned to one class (i.e. half are misclassified). The plateau at the top corresponds to the case where the decision boundary lies correctly between the two classes, but this time has the wrong sign so that all samples are misclassified.

As we climb the stairs from the bottom plateau to one of the plateaus in the middle, each upward step corresponds to a weight change that pushes the decision boundary past one of the data samples. This causes the sample to be misclassified and results in an increase in $E(\mathbf{W})$. The steps leading from the middle plateaus

to the top can be explained in a similar fashion. They correspond to the case where the decision boundary starts at the far right (or left) and (with the wrong sign) is moved in toward the center where all samples are misclassified.

It is important to note the symmetry of the surface and its position with respect to the four quadrants of the w_0, w_1 weight plane. The bottom plateau corresponds to a set of solutions $\mathbf{W}^*$ that give perfect classification. The top plateau is in the opposite quadrant and therefore represents the set of solutions $-\mathbf{W}^*$ that give perfect misclassification. The middle plateaus (which also lie in opposing quadrants) correspond to a set of solutions that assigns all the samples to a single class.

Although it is not clear from the figure, the bottom plateau is not completely flat. The minimum of the surface occurs at the outer extreme of this plateau as the weights approach ∞ along a line $-w_0/w_1 = c$, where c is a decision boundary that lies between the two classes in Fig. 1. To see this let $c = -w_1/w_0$ correspond to a solution that correctly classifies the samples. The desired mapping for each sample is either 0 or 1 depending on its class membership. The actual mapping for a sample x is given by

$$\left(1 + e^{-(w_1 x + w_0)}\right)^{-1}$$

which can only approach 0 or 1 in the limit as the exponent approaches ∞ or $-\infty$ respectively. Thus, we can reduce the total mapping error to zero by scaling the weights on the bottom plateau by an arbitrarily large value so that the mapping

$$\lim_{k \to \infty} \left(1 + e^{-k(w_1 x + w_0)}\right)^{-1}$$

approaches the desired mapping.

While multilayer perceptrons are noted for their noise tolerance, it is clear from this example that they are not equally sensitive to weight perturbations in all directions. All solutions on the bottom plateau give perfect classification. If we choose one of these solutions and scale it by a constant k that varies from $+1$ to -1 we follow a path that climbs straight up from the bottom plateau to the top plateau via the steepest part of the surface in the center. In fact any change in the weights (on the bottom plateau) that moves them in the direction of the origin will give rise to a noticeable increase in $E(\mathbf{W})$. On the other hand, $E(\mathbf{W})$ is virtually unaffected by movements away from the origin. This characteristic is universally true of classification problems, in both single and multiple layer networks. Any time the solution weights are allowed to approach the origin the outputs of all nodes in network will approach 0.5 (since the weighted sums will approach zero) resulting in a large mapping error. If the weights are allowed to continue to pass through the origin, changing their sign, the logic of each node will be reversed. This usually results in poor classification and an even larger mapping error.

The fact that the region near the origin represents a transition region as described previously gives strong support for initializing the weights to small random values. In doing so we have a high probability of positioning the initial weights near the origin on the steepest part of the surface where the downhill direction points toward a solution. Note, this is only true for the one-node case. In the multilayer case the steepest part of the surface is not necessarily near the origin. This will be discussed in more detail later in this section.

Suppose now that the two classes of training data are overlapped. For example, consider the training set and the corresponding error surface shown in Fig. 2. The nature of the surface is basically the same as shown previously with one exception. As we move upward from the bottom plateau we find a downward step among the upward steps. The upward steps correspond to transitions from correct to incorrect classifications as before, and the downward step to a transition from

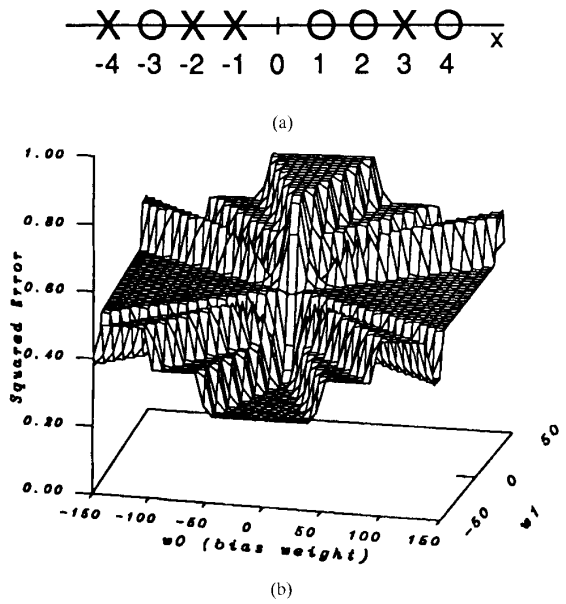

Fig. 2. Example problem 2: overlapping data. (a) Training samples for the one-node network. (b) Corresponding error surface.

an incorrect to correct classification. This happens because samples from the two classes are intermixed.

Let us now look at the types of surfaces formed by a two-layer network. The problem we wish to examine is one in which the training data are drawn from a four-dimensional, two-class Gaussian distribution with the mean vectors and the covariance matrices given by

$$m_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad m_2 = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

$$C_1 = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix} \qquad C_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The optimal decision boundary for this problem is quadratic. This boundary can be approximated using a two layer perceptron network with two nodes in the hidden layer. This network has a total of thirteen weights. As we did in the one-node problem we begin with a relatively small number of training samples, ten for each class. Although there is overlap between the underlying data distributions, the training samples drawn in this example turn out to be separable by the two-layer MLP.

To examine the error surface for this problem the network was first trained using back propagation to obtain a set of weights that correspond to a minimum of the surface. The plots shown in Fig. 3 correspond to plots of $E(\mathbf{W})$ as a function of two selected weights in the network, while keeping the other eleven weights fixed at the minimum. The weights in these figures are labeled so that $w(i, j, k)$ corresponds to weight k of node j in layer i. As in the previous section we observe regions that are both very flat and very steep. For the most part however we see only troughs, not the stairs seen in the previous section. This can be explained as follows. Fixing eleven of the weights at the minimum and varying the other two is analogous to fixing one of the weights in the one-node problem and varying the other. In the one-node problem the 1-D surface that results will

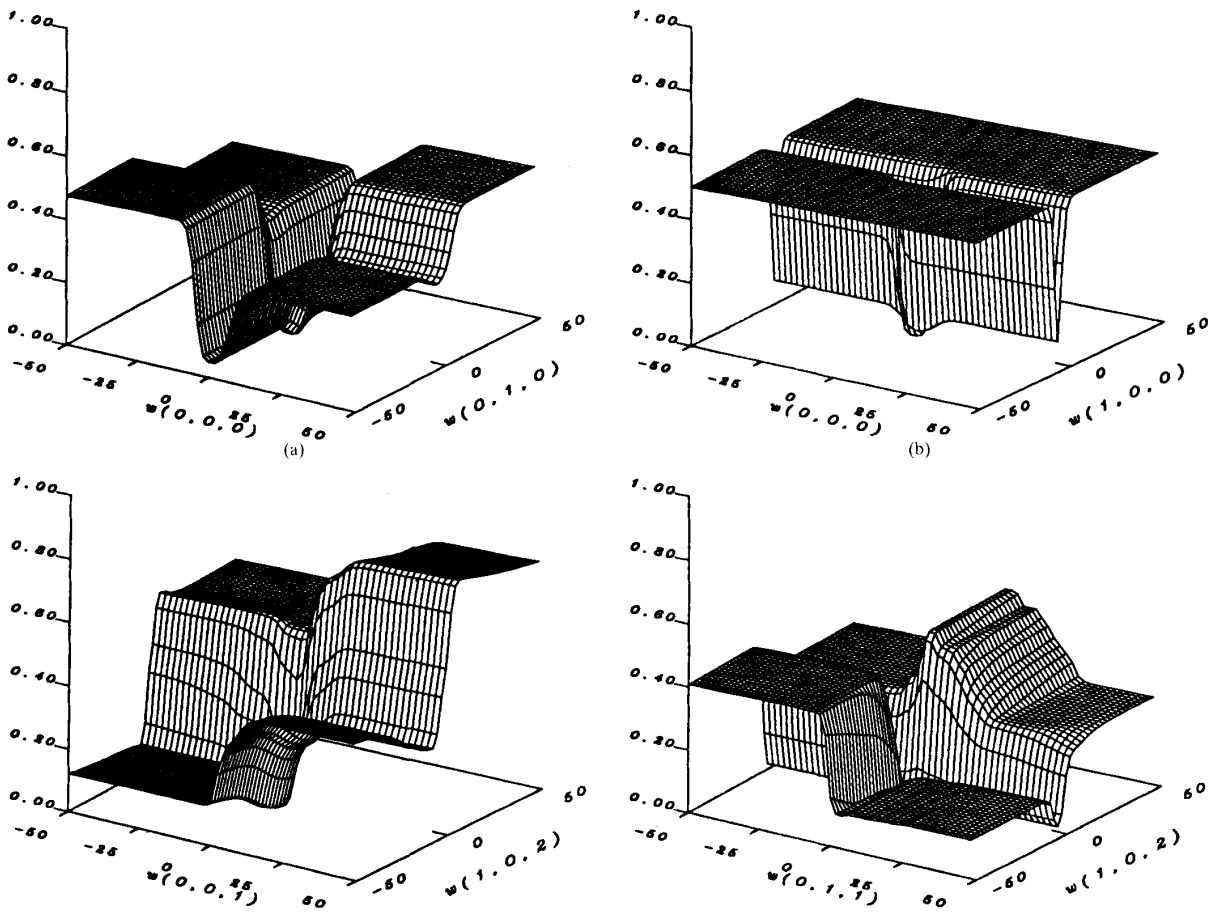

Fig. 3. Two-dimensional projections of the error surface for the 13-weight network with 10 samples per class formed by fixing 11 weights at the minimum and varying the other two. (a)$E(w(0,0,0), w(0,1,0))$. (b)$E(w(0,0,0), w(1,0,0))$. (c)$E(w(0,0,1), w(1,0,2))$. (d)$E(w(0,1,1), w(1,0,2))$.

have a "V" shape as we move from one plateau down through the minimum and back up to the other plateau. The troughs seen in Fig. 3 are analogous to the "V" shaped curve obtained in the one-node problem. To see the stair step effect in the current problem we must move the eleven fixed weights to a position on the surface away from the minimum. This is accomplished by scaling the previous weight values by a factor of four. Fig. 4 shows the results. As expected the stairs are now visible and the surface characteristics bear a closer resemblance to those seen in the smaller problem. The stairs appear here for the same reason they did in the one-node problem. As the weights of the MLP change, the decision boundary is moved past the data samples. In this case however the decision boundary is no longer linear (a hyperplane) as it was in the one-node problem. In the MLP the boundary changes both its shape and location as the weights change.

In the one-node previous problem we found the origin of the weight space to be a transition region of the surface, and a suitable region for weight initialization. In multilayer nets, however, the situation is more complicated. For weights in the output layer, the surface exhibits characteristics similar to those seen in the previous section. Fig. 5(a) shows how the error surface changes when we fix the hidden layer weights at their minimum and scale the output layer weights incrementally between -1 and 1. Note that the surface is steep when the output weights are zero. However, when we scale the entire weight solution incrementally from -1 to 1, we see that the error surface is quite flat at the origin as shown in Fig. 5(b). The steepest part of the surface is apparently located away from the origin. This suggests that hidden layer weights should be initialized to "moderate size" values, which is consistent with the findings in [7]. However, it is difficult to determine exactly what values constitute a "moderate size" for a particular problem. In fact, it is not just the size of the weights, but the orientation with respect to the origin that is important. In addition, in the appendix we show that for large weights the gradient approaches zero, regardless of the problem at hand. Thus while the origin may not be the ideal region for initialization of hidden layer weights, it remains the safest.

B. Learning Curves

Learning curves (plots of the total squared error verses the iteration index) for the one-node problem previously shown are shown in Fig. 6. We note here that the stair-steps on the surface are reflected as stair-steps in the learning curve. The reason for this is obvious. As the search proceeds from one level of the surface to the next it results in a large decrease in the total squared error. This happens very rapidly

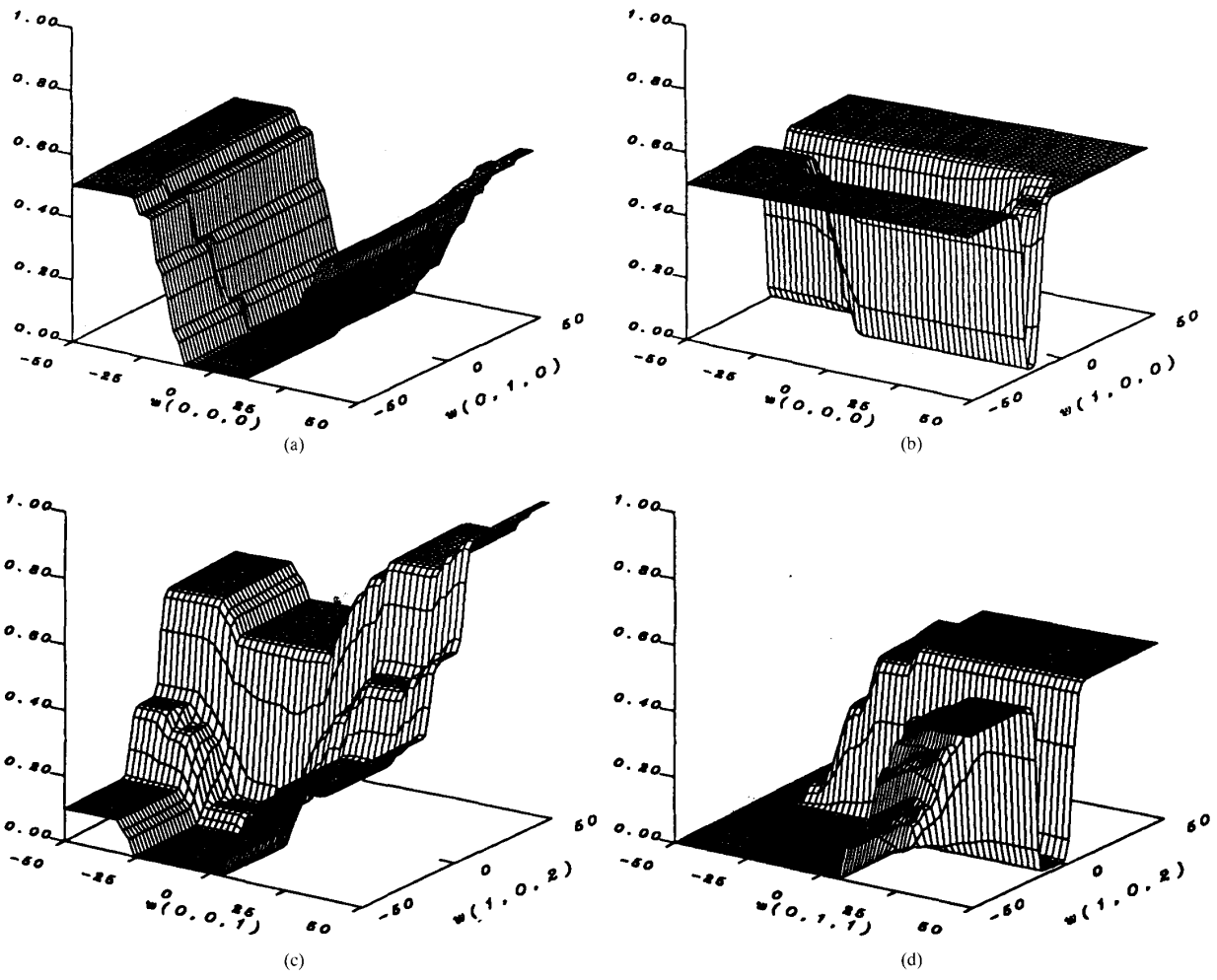

Fig. 4. Two-dimensional projections of the error surface for the 13-weight network with 10 samples per class formed by fixing 11 weights at four times their value at the minimum and varying the other two. (a) $E(w(0,0,0), w(0,1,0))$. (b) $E(w(0,0,0), w(1,0,0))$. (c) $E(w(0,0,1), w(1,0,2))$. (d) $E(w(0,1,1), w(1,0,2))$.

since the gradients are larger in this region (on the edge of the stair). Learning curves for the two-layer network are shown in Fig. 7. As in Fig. 6 we note stair-steps in the curves that correspond to stair-steps on the error surface.

C. Gradient Histograms

To gain an appreciation for the degree of "flatness" and "steepness" of the surface a histogram of $\| \nabla \|$, the magnitude of the gradient, is shown in Figs. 8 (for the one-node network) and 9 (for the two-layer network). The gradient was randomly sampled over a quadrant of the surface that excluded the minimum. This was done to avoid characteristics of the surface that are only encountered near the minimum. We note that the dynamic range of gradients is approximately 10^7, although most of them are concentrated in a region that spans roughly three orders of magnitude. The largest gradients from the steepest part of the surface take on values near 1. The most important observation to be made with regard to learning is that well over 50% of the surface is comprised of flat regions whose gradients are from 1 to 6 orders of magnitude smaller that those in the steep regions. This means that we can expect learning to proceed at 10 to 10^6 times slower in these regions. In addition this makes termination of the search very difficult in practice since flat spots lead to the same type of behavior as that seen near the minimum.

III. Searching the Surface

The large number of very flat and very steep parts of the surface make it difficult to search the surface efficiently using algorithms with a fixed learning rate μ. In these algorithms μ must be chosen to bound the step size on steep parts of the surface to prevent large steps from leading to instabilities. This results in extremely small steps on flat parts of the surface. This is the primary reason that back-propagation is so slow. Using the momentum term in (5) can be of some help. This term tends to keep the search going at the same rate and direction as its recent past. In this way "momentum" can be built up on the steep part of the surface so that the search will continue to make progress when it hits a flat part.

The stair-step appearance of the surface also makes it very dangerous to perform line searches that form a quadratic fit along the direction of search. On flat parts of the surface the quadratic approximation can give rise to a very large step taking the search out

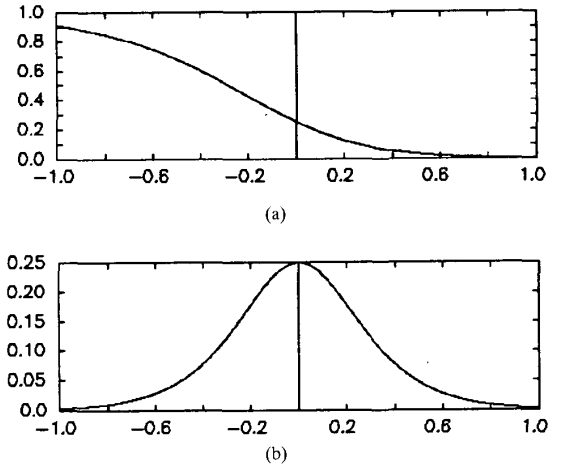

Fig. 5. Slices of the error surface (a) as the output weights are scaled from -1 to $+1$, and (b) as all the weights are scaled from -1 to $+1$.

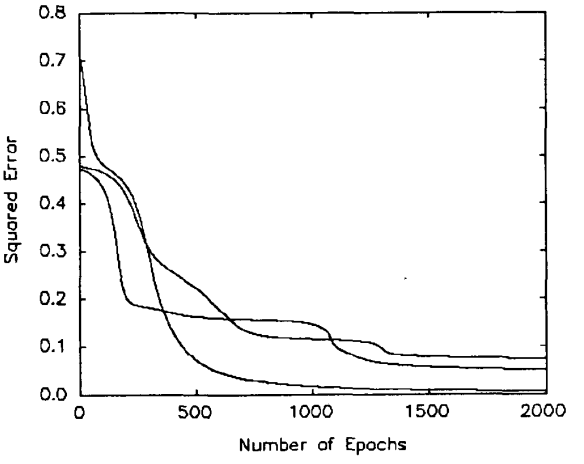

Fig. 7. Typical learning curves for the 13-weight problem with 10 samples per class.

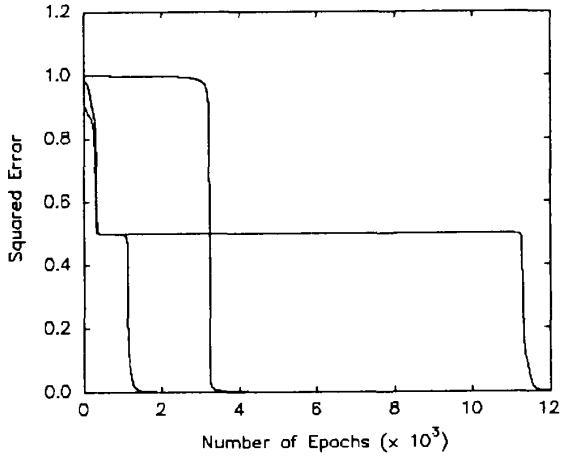

Fig. 6. Typical learning curves for the one-node problem.

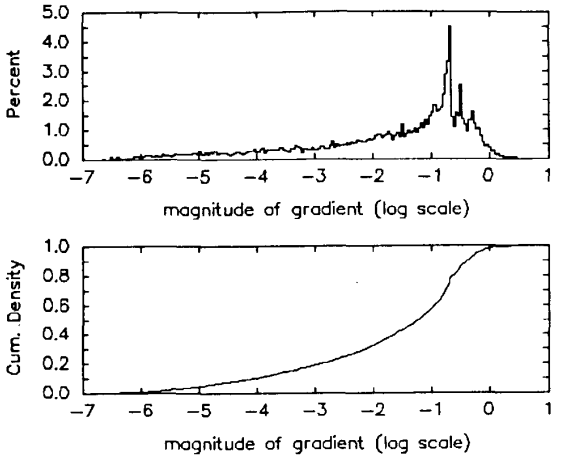

Fig. 8. Histograms of $\| \nabla \|$ for the one-node problem.

onto an even flatter part of the surface far away from the minimum as shown in Fig. 10.

Finally, we note that the flat parts of the surface extend to infinity in all directions. This is shown more formally in the appendix for a general MLP network. This characteristic makes it very dangerous to perform accurate line searches, that is line searches that actually search for a minimum by increasing μ until $E(\mathbf{W}(k) - \mu \mathbf{d}(k))$ stops decreasing. The problem here is that with this type of surface there are many cases where there is no minimum in a given direction, just a flat spot that extends to infinity.

IV. Increasing the Number of Training Samples

We now look at the effect of increasing the number of training samples. For the one-node problem consider samples drawn from two Gaussian distributions with unit variance and with means equal to -1 and 1. A total of 50 samples from each class are used. The $E(\mathbf{W})$ surface is shown in Fig. 11.

The surface appears much smoother in the stair-step region. This is because the steps from adjacent samples have been smeared together. There is no longer a plateau at the bottom or the top, only a narrow trough. This happens because there is no longer a large range of positions where the decision boundary can be placed and still give optimal classification, there is only one point. The trough in the figure corresponds to the class of solutions $k\mathbf{W}^*$ that give the optimal decision boundary. This trough, however, does not represent a distributed minimum. This can be seen on the contour plot in Fig. 12. Even though all the weight values lying along the bottom of the trough give optimal classification, they produce slightly different mapping errors. Consequently the minimum of the surface is no longer at ∞. Because of the overlap it is impossible to have zero mapping error, so the minimum represents a trade-off between the mapping error for misclassified samples (on both sides of the boundary) and the mapping error for correctly classified samples. Weights approaching ∞ as in the previous example would only minimize the mapping error for the correctly classified samples, while increasing the mapping error for incorrectly classified samples.

If the data in this example is translated away from the origin in one direction or the other it has a significant effect on the error surface. Consider for example the surface in Fig. 13. The samples used to generate this surface are identical to those used in the previous example translated to the left by 2. The effect is to produce two very steep walls in place of two of the stair-step regions. Actually, each

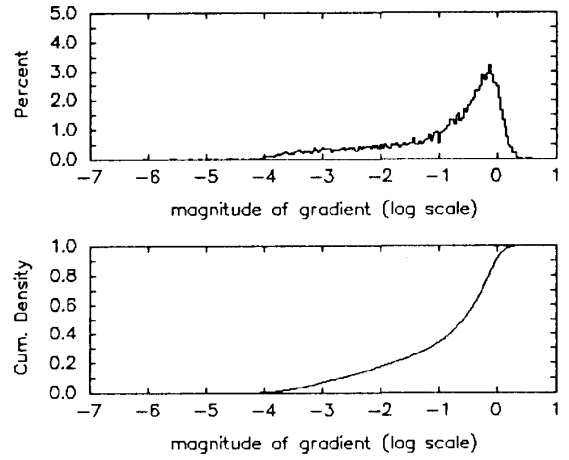

Fig. 9. Histograms of $\| \nabla \|$ for the 13-weight problem with 10 samples per class.

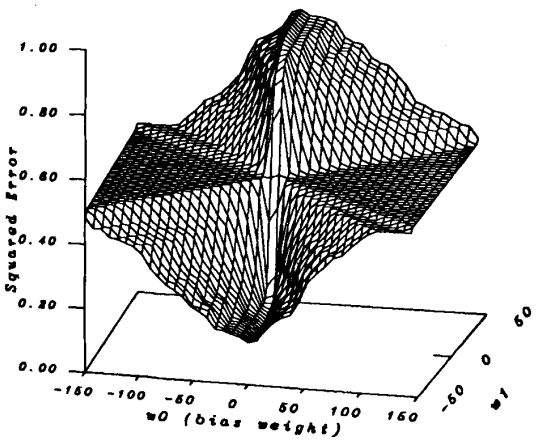

Fig. 11. Example problem 4: error surface using a large number of training samples with overlapping data.

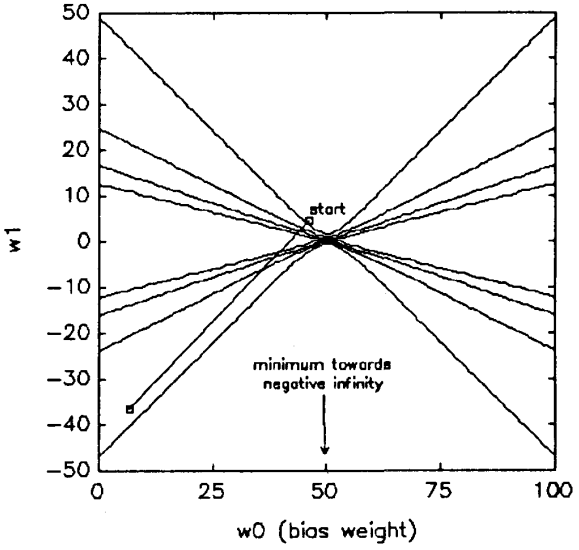

Fig. 10. Contour plot of the error surface in Fig. 1 and search path for line search that searches for the minimum in the current direction by performing a quadratic fit before computing a new direction.

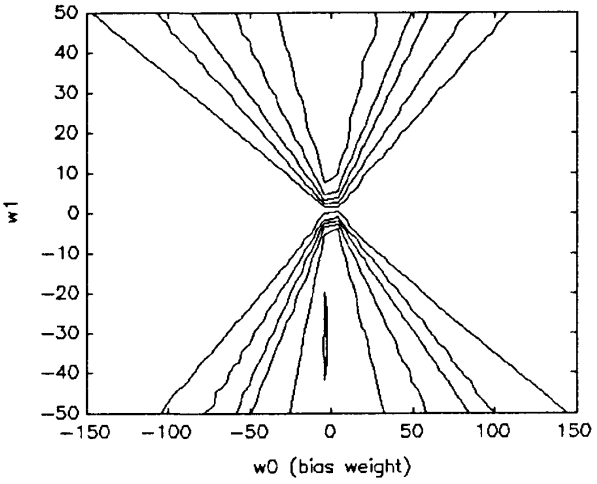

Fig. 12. Contour plot of the surface in Fig. 11.

of these walls is formed as a combination of steps as before, but this is not visible because they are so narrow. They are formed when the decision boundary (which is inversely proportional to w_1, see (6) is *accelerated* past the data samples as w_1 passes through zero.

The presence of this wall can have an unusual effect on the search path. A typical path that follows the gradient direction is shown on the contour plot in Fig. 14. The path finds its way down the stairs to the trough that leads to the minimum. On one side of the trough is the wall and on the other are the stairs. If the learning rate is sufficiently large what will happen (especially if momentum is used) is that the search will proceed across the trough ending up on a very steep part of the wall. At this point the (negative) gradient is very large so that the next step shoots back across the trough out onto the flat part of the surface causing the search to retrace its steps to the trough. In fact if the learning rate is high enough this may happen repeatedly causing the overall search to actually move away from the minimum as illustrated in Fig. 14. This helps explain why a search that starts out doing very well will sometimes appear to oscillate or even go up hill for a time. This problem can be resolved by lowering the learning rate, but this causes the search to proceed very slowly down the trough to the minimum.

The smoothing affect due to an increase in the number of training samples is also seen in the two-layer network. If we increase the number of training samples to 200 per class we obtain plots of $E(\mathbf{W})$ like those in Fig. 15 (formed in the same manner as before). As they did in the one-node problem these surfaces appear much smoother.

We should point out that the changes in the location of the minimum and the effect of data translation noted previously for the one-node problem are also observed in the two-layer network, although it is not possible to illustrate these effects pictorially.

V. The Effect of Using Alternative Desired Outputs

So far we have assumed that the desired output used comes from the set $\{0, 1\}$. However, it is quite common to use $\{0.1, 0.9\}$ as an alternative [10]. One motivation for this choice of target output is

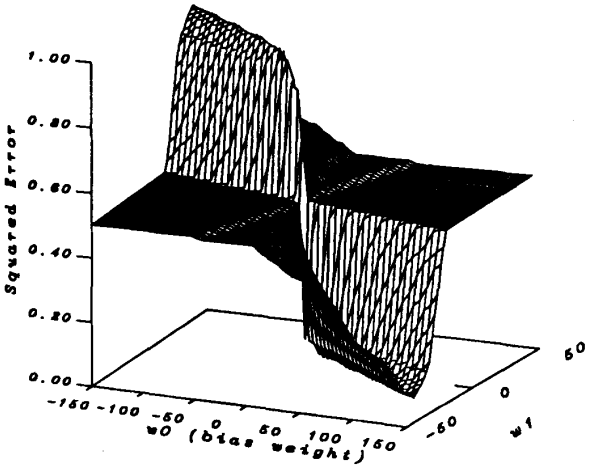

Fig. 13. Example problem 5: error surface for translated data.

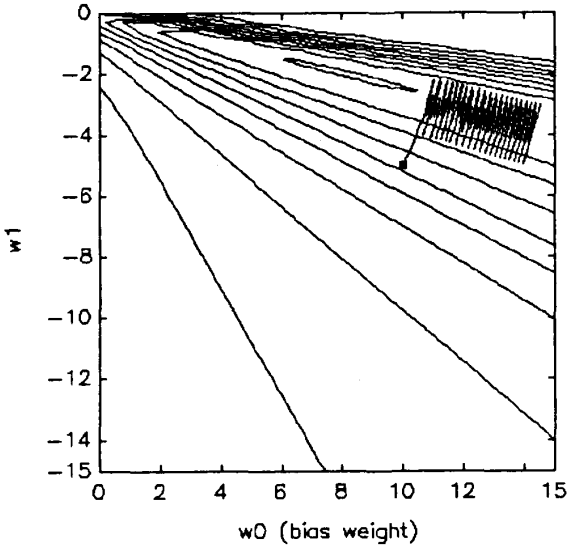

Fig. 14. Contour plot of the error surface in Fig. 13 and search path for gradient search (with momentum).

that the weight solution for separable classes is not longer at infinity as it was in Section II [2]. However, this choice of target outputs introduces slight "dips" near the bottom of each stair step. These dips can be seen in the plot in Fig. 16, which represents a slice of the error surface generated using the example in Section II with target outputs of 0.1 and 0.9. These dips occur because the mapping error for individual data samples is minimum when the perceptron output matches the 0.1 or 0.9 target exactly, and larger when the perceptron output falls on either side of the target values.

One technique for avoiding this problem is to use a modified error criterion, whereby, the error is set to zero if the output of the perceptron is greater than 0.9 when $d = 0.9$ or is less than 0.1 when $d = 0.1$.

VI. Conclusion

In this paper we have observed several distinct characteristics of error surfaces for multilayer perceptrons. In addition we have tied these characteristics to the characteristics of the problem being loaded onto the network. Specifically we have seen that the surface is comprised of numerous flat and steep regions where the magnitude of the gradients vary by several orders of magnitude. There are numerous plateaus that extend to infinity in all directions. In addition the surface has many troughs that are extremely flat in the direction of search. When the number of training samples is small we observe stair steps in the surface, one for each training sample. When the number of training samples is increased the surface becomes smoother as the steps smear together. The surface appears more complex when there is overlap from different classes. Finally, there is strong support for initializing weights to small random values. It should be noted that we have considered only pattern recognition problems in this paper, and that many of our observations are not likely to carry over to other types of applications.

Acknowledgment

The authors would like to thank Guy Smith of the Flinders University in Australia for helpful comments on the initial draft of this paper. We would also like to thank the reviewers and the associate editor for their careful reading of the paper and their helpful comments. Finally, we would like to thank the BSP Group for their enlightening series of Friday afternoon seminars at the UNM Club.

Appendix

In this appendix we show that as the weights of the MLP grow large the error surface becomes extremely flat, regardless of the problem at hand. This is accomplished by showing that the magnitude of the gradient becomes extremely small, and in fact approaches zero as the magnitude of the weights approach infinity.

Let us begin by adopting the following notation. The output for jth node in the lth layer is denoted

$$u_{l,j} = f(y_{l,j}) \tag{7}$$

where $f(\cdot)$ is the sigmoid nonlinearity, and $y_{l,j}$ is the weighted sum for this node, given by

$$y_{l,j} = \sum_{i=0}^{N_{l-1}} w_{l,i,j} u_{l-1,i} = \mathbf{w}_{l,j}^{l} \mathbf{u}_{l-1} \tag{8}$$

For convenience we have defined $u_{l-1,0} = 1$ so that $w_{l,0,j}$ represents the bias weight. The component of the gradient that corresponds to weight $w_{l,i,j}$ is given by [10]

$$g_{l,i,j} = \sum_{p=1}^{P} e_{l,j} f'(y_{l,j}) u_{l-1,i}. \tag{9}$$

where P is the number of training patterns, and $e_{l,j}$ is the "error" term for the l,jth node. At the output layer $e_{l,j}$ represents the difference between the actual output and the target output for the node. For hidden layers $e_{l,j}$ is given by

$$e_{l,j} = \sum_{i=0}^{N_{l+1}} e_{l+1,i} f'(y_{l+1,i}) w_{l+1,j,i}. \tag{10}$$

A key factor in determining the behavior of the gradient as the weights grow large will be the $f'(\cdot)$ term in (9) and (10). This term represents the derivative of the sigmoid nonlinearity, and is easily shown to be of the form

$$f'(y) = f(y)(1 - f(y)) \tag{11}$$

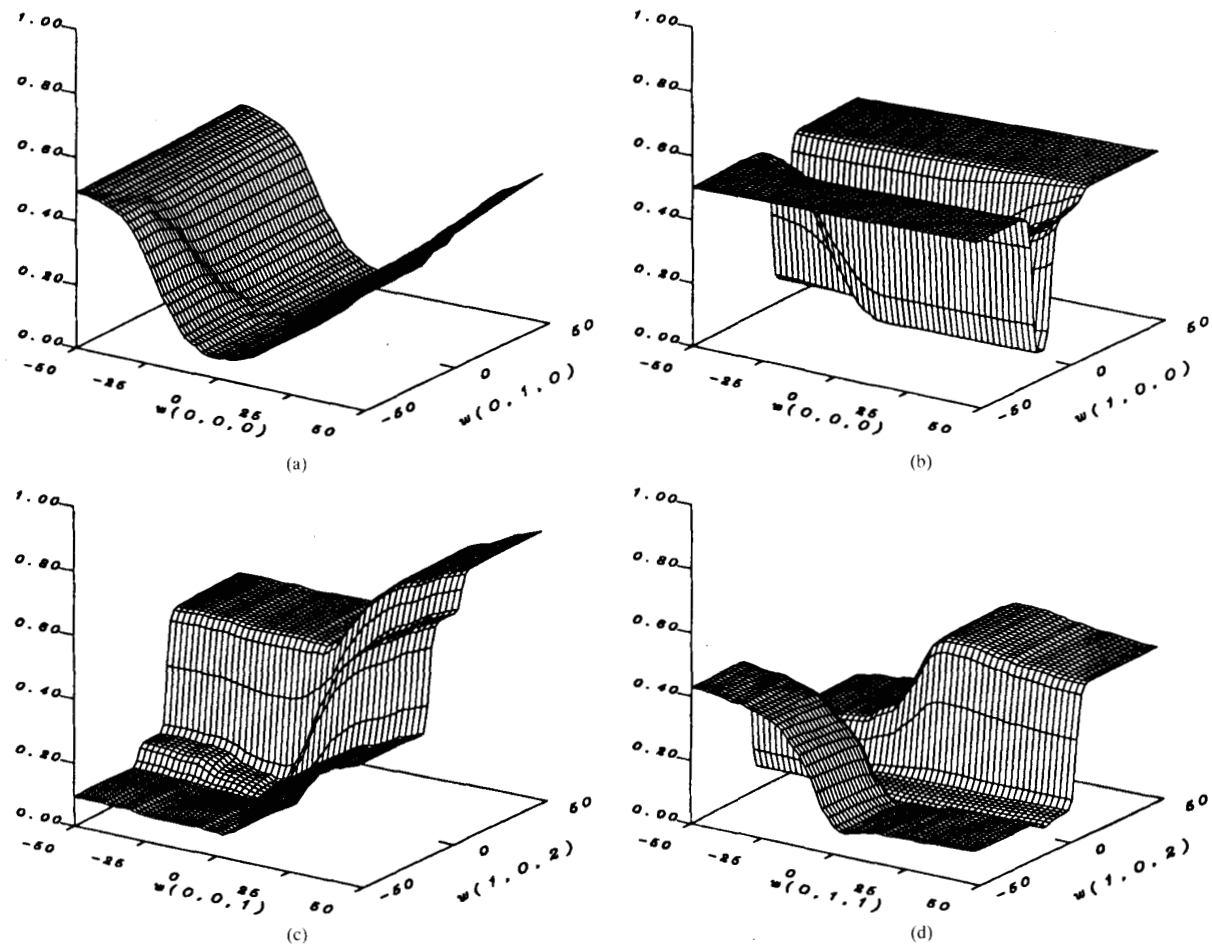

Fig. 15. Two-dimensional projections of the error surface for the 13-weight network with 200 samples per class formed by fixing 11 weights at four times their value at the minimum and varying the other two. (a) $E(w(0,0,0), w(0,1,0))$. (b) $E(w(0,0,0), w(1,0,0))$. (c) $E(w(0,0,1), w(1,0,2))$, (d) $E(w(0,1,1), w(1,0,2))$.

Our goal is to show that as the magnitude of the weights grows (approaches ∞) the gradient term in (9) becomes very small (approaches zero). This will be true for all but an infinitesimally small portion of the weight space.

We begin with the following expression, obtained from (9), which represents a bound on the magnitude of gradient.

$$|y_{l,j}| \leq P|e_{\max}| \cdot |f'_{\max}(y)| \cdot |u_{\max}|. \tag{12}$$

The subscript max is used to indicate the term of largest magnitude in the summation in (9). As the weights grow large we find the following.

1) The last term, $u_{\max}$, remains bounded between zero and one.
2) The middle term, $f'_{\max}(y)$ approaches zero for a large portion of the weight space. This will be shown here.
3) At the output layer the error term, $e_{l,j}$, is bounded between -1 and 1. In the hidden layers we will show that this term approaches zero.

Given 1, 2, and 3 then it must be true that as the weights grow large the gradient term approaches zero. Note that this will be that case regardless of the nature of the problem.

First we show that the middle term in (12), $f'_{max}(y)$, approaches zero. It is easily verified that this term approaches zero as its argument approaches $\pm\infty$ (see (11)). To show that y approaches $\pm\infty$, let us represent the weight vector as follows,

$$\mathbf{w}_{l,j} = \alpha \hat{\mathbf{w}}_{l,j} \tag{13}$$

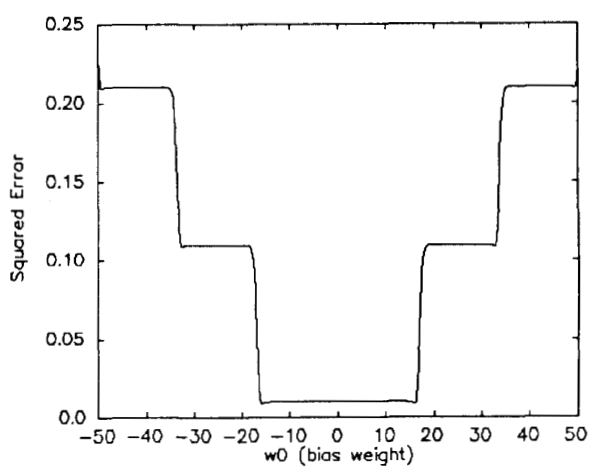

Fig. 16. Slice of the error surface when 0.1 and 0.9 are used as the desired outputs.

where α is a positive scalar and $\hat{\mathbf{w}}_{l,j}$ is a unit vector in the direction of $\mathbf{w}_{l,j}$. With this we can write

$$|y_{l,j}| = \left|\mathbf{w}_{l,j}^T \mathbf{u}_{l-1}\right| = \alpha\left|\hat{\mathbf{w}}_{l,j}^T \mathbf{u}_{l-1}\right|. \tag{14}$$

Now, as the magnitude of the weights increase, the magnitude of $y_{l,j}$ also increases, and in the limit we have

$$\lim_{\alpha \to \infty} |y_{l,j}| = \infty. \tag{15}$$

This is true everywhere except on the hyperplane defined by $\hat{\mathbf{w}}_{l,j}^T \mathbf{u}_{l-1} = 0$. Since this hyperplane occupies an infinitesimally small region of the weight space we conclude that $|y_{l,j}|$ grows arbitrarily large for a large portion of the weight space.

It remains only to show that the error term in (12) approaches zero for hidden layer nodes. Using (10) we can obtain the following bound on the magnitude of this term:

$$|e_{l,j}| \leq (N_{l+1} + 1) \cdot |e\max| \cdot \left|f'(y)w\right|_{\max} \tag{16}$$

where,

$$\left|f'(y)w\right|_{\max} = \max_{l,j,i} \left|f'(y_{l+1,i})w_{l+1,j,i}\right|.$$

As the weights increase the last term in (16) approaches zero. This can be shown by evaluating the following limit: $\lim_{\alpha \to \infty}[f'(y)w]$. Using the fact that both y and w are proportional to α (see (14) and (13)), one can easily verify that this limit is zero (e.g. using L'Hopital's rule).

Finally let us consider the $|e_{\max}|$ term in (16). In the last hidden layer (just before the output layer) $e_{\max}$ represents an error from the output layer and is thus bounded between -1 and 1. This, combined with the previous result tells us that $|e_{l,j}|$ approaches zero for the last hidden layer. By showing that $|e_{l,j}|$ approaches zero for this layer we will have recursively verified that $|e_{max}|$ is bounded (and in fact approaches zero) for all previous hidden layers.

REFERENCES

[1] A. Blum and R. L. Rivest, "Training a 3-node neural network is NP-complete," in *Proc. Computational Learning Theory (COLT) Conf.*, 1988, pp. 9–18.

[2] M. Brady, R. Raghavan, and J. Slawny, "Gradient descent fails to separate," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 1, 1988, pp. 649–656.

[3] K. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, no. 3, pp. 183–192, 1989.

[4] W.Y. Huang and R.P. Lippmann. "Neural net and traditional classifiers," in *Neural Information Processing Systems*. New York: 1988, Amer. Inst. Phys., pp. 387–396.

[5] B. Irie and S. Miyake, "Capabilities of three-layered perceptrons," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 1, San Diego, CA, 1988, pp. 641–648.

[6] S. Judd, "On the complexity of loading shallow neural networks," *J. Complexity*, vol. 4, pp. 177–192, 1988.

[7] H. Lari-Najafi, M. Nasiruddin, and T. Samad, "Effect of initial weights on back-propagation and its variations," in *Proc. 1989 IEEE Int. Conf. Syst., Man, Cybern.*, vol. 1, pp. 218–219, 1989.

[8] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE Acoust., Speech Signal Processing Mag.*, vol. 4, pp. 4–22, Apr. 1987.

[9] D. B. Parker, "Learning-logic," Tech. Rep. TR–47, Center for Comp. Res. in Econ. and Man, MIT, Cambridge, MA, Apr. 1985.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.

[11] R.S. Sutton, "Two problems with back-propagation and other steepest-descent learning procedures for networks," in *Proc.Eighth Annu. Conf. Cognitive Sci. Soc.*, 1986, pp. 823–831.

[12] G. Tesauro and Y. H. S. Ahmad, "Asymptotic convergence of back-propagation," *Neural Computation*, vol. 1, no. 3, pp. 382–391, 1989.

[13] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," doctoral dissertation, applied mathematics, Harvard Univ., Boston, MA, Nov. 1974.

A Semantic Network Representation of Personal Construct Systems

Michael W. Bringmann and Frederick E. Petry

***Abstract*—A method is presented to transform and combine heuristic knowledge gathered from multiple domain experts into a common semantic network representation. Domain expert knowledge is gathered with an interviewing tool based on personal construct theory. The problem of expressing and using a large body of knowledge is fundamental to artificial intelligence and its application to knowledge-based or expert systems. The semantic network is a powerful, general representation that has been used as a tool for the definition of other knowledge representations Combining multiple approaches to a domain of knowledge may reinforce mutual experiences, information, facts, and heuristics, yet still retain unique, specialist knowledge gained from different experiences. An example application of the algorithm is presented in two separate expert domains.**

I. INTRODUCTION

The problem of representing and using a large body of knowledge is fundamental to artificial intelligence and its application to knowledge-based or expert systems [1], [2]. An important instance of knowledge representation research is the semantic network introduced by Quillian [3], [4]. Semantic networks may express knowledge as concepts, their properties and hierarchical relationships among classes (e.g., IS-A hierarchies).

Also, the design and construction of knowledge acquisition tools recently have become areas of intense research and development. The elicitation of knowledge from a skilled individual is the most fundamental step in the development of an expert system. Experts can provide the information themselves by writing out their thoughts or by "thinking aloud" while they solve a specific problem in their field. Alternatively, a knowledge engineering analyst can obtain the required information by interviewing one or several domain specialists. Many dozens of robust tool sets and applications have been developed for the knowledge engineer during the recent commercial application of expert systems [5]–[7] Also a variety of systems have emerged that are intended to interview human experts and encode the resulting knowledge [8]–[10]. However, the majority of such systems address specific aspects, environments or domains with minor attempts to integrate divergent knowledge sources.

A knowledge-based system seeks to emulate the cognitive functions of a human specialist. The subject matter domains for which

Manuscript received October 21, 1990; December 27, 1991.

M. W. Bringman is with QMS Inc., Product Research and Development, P.O. Box 81250, Mobile, AL 36689–1250.

F. E. Petry is with the Center for Intelligent and Knowledge-Based Systems, Department of Computer Science, Tulane University, New Orleans, LA 70118.

IEEE Log Number 9201403.