Neural Networks, Vol. 3, pp. 707–711, 1990
Printed in the USA.

0893-6080/90 $3.00 + .00

ORIGINAL CONTRIBUTION

An Unsupervised Learning Technique for Artificial Neural Networks

AMIR F. ATIYA

California Institute of Technology

(*Received* 24 *April* 1989; *revised and accepted* 4 *June* 1990)

Abstract—*A new artificial neural model for unsupervised learning is proposed. Consider first a two-class pattern recognition problem. We use one neuron (possibly higher order) with a sigmoid in the range from* -1 *to* 1. *Positive output means class* 1 *and negative output means class* 2. *The main idea of the method is that it iterates the weights in such a way as to move the decision boundary to a place of low pattern density. Constraining the length of the weight vector, if the neuron output is mostly near* 1 *or* -1, *then this means that the patterns are mostly far away from the decision boundary and we have probably a good classifier. We define a function which measures how close the output is to* 1 *or* -1. *Training is performed by a steepest-ascent algorithm on the weights. The method is extended to the multiclass case by applying the previous procedure in a hierarchical manner (i.e., by partitioning the patterns into two groups, then considering each group separately and partitioning it further and so on until we end up with the final classifier).*

Keywords—Neural networks, Unsupervised learning, Pattern recognition, Hierarchical classification, Clustering, High-order networks.

1. INTRODUCTION

One of the most important features in neural networks is its learning ability, which makes it in general suitable for computational applications whose structures are relatively unknown. Pattern recognition is one example of such kinds of problems. For pattern recognition there are mainly two types of learning: supervised and unsupervised learning (refer to Duda & Hart, 1973). For supervised learning the training set consists of typical patterns whose class identities are known. For unsupervised learning, on the other hand, information about the class membership of the training patterns is not given, either because of lack of knowledge or because of the high cost of providing the class labels associated with each of the training patterns.

A number of neural network models for unsupervised learning have been proposed, for example, the competitive-learning (Grossberg, 1976) and the self-organizing maps (Kohonen, 1984). In pattern-recognition problems the pattern vectors tend to form clusters, a cluster for each class, and therefore the first step for a typical unsupervised learning technique is the estimation of these clusters. The clusters are usually separated by regions of low pattern density. We present here a new method for unsupervised learning in neural networks (see also Atiya, 1988). The basis idea of the method is to update the weights of the neurons in such a way as to move the decision boundaries in places sparse in patterns. This is one of the most natural ways to partition clusters. It is more or less similar to the way humans decompose clusters of points in three or less dimensions visually. This method contrasts with the traditional approach of the competitive-learning and the self-organizing maps, whereby the estimation of the membership of a pattern vector depends only on, respectively, the inner products and the Euclidian distances between this vector and suitably estimated class representative vectors.

Acknowledgements: The author wishes to gratefully acknowledge Dr. Yaser Abu-Mostafa for the useful discussions. This work is supported by the Air Force Office of Scientific Research under grant AFOSR-88-0213.

Requests for reprints should be sent to Amir Atiya, Electrical Engineering 116-81, Caltech, Pasadena, CA 91125.

2. THE MODEL

Let $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$ represent the training pattern vectors. Let the dimension of the vectors be n. The training patterns represent several different classes. The class identities of the training patterns are not known. Our purpose is to estimate the number of classes available as well as the parameters of the neural classifier. Let us first consider the case of having two classes. Our neural classifier consists of one neuron (see Figure 1) with n inputs correspond-

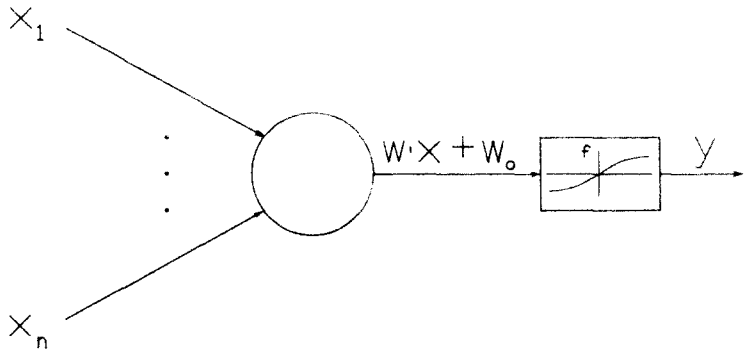

FIGURE 1. The model of a first-order neuron with a sigmoid-shaped function.

ing to the n components of the pattern vector $\mathbf{x}$ to be classified. The output of the neuron is given by

$$y = f(\mathbf{w}^T\mathbf{x} + w_0)$$

where $\mathbf{w}$ is a weight vector, w_0 is a threshold and f is a sigmoid function from -1 to 1 with $f(0) = 0$; for example,

$$f(u) = \frac{1 - e^{-u}}{1 + e^{-u}}.$$

Positive output means class 1, negative output means class 2, and zero output means undecided. The output can be construed as indicating the degree of membership of the pattern to each of the two classes; thus we have a fuzzy classifier.

The decision boundary is the hyperplane

$$\mathbf{w}^T\mathbf{x} + w_0 = 0$$

(see Figure 2). Patterns near the decision boundary will produce outputs close to zero while patterns far away from the boundary will give outputs close to 1 or -1. Assuming $\|\mathbf{w}\| \leq a$, where a is a constant, a good classifier is one that produces an output close to 1 or -1 for the training patterns, since this indicates its "decisiveness" with respect to the class memberships. Therefore, we design the classifier so as to maximize the criterion function

$$J = \frac{1}{N}\sum_j f^2(\mathbf{w}^T\mathbf{x}^{(j)} + w_0) - \left(\frac{1}{N}\sum_j f(\mathbf{w}^T\mathbf{x}^{(j)} + w_0)\right)^{2q},$$

subject to $\|\mathbf{w}\| \leq a$, where q is a positive integer (for best results we take $q = 2$). The first term is a measure of the decisiveness of the classifier with the given test pattern set. Regarding the second term, it has the following purpose. The first term is maximized if the decision hyperplane is very far away from the patterns resulting in the output being very close to 1 for all test patterns (or close to -1 for all patterns), in other words, all patterns are assigned to one class only, which is a trivial solution. Incorporating the second term will prevent this trivial solution because if $f(\mathbf{w}^T\mathbf{x}^{(j)} + w_0) \simeq 1$ for all j (or -1 for all j), then J will be nearly zero. However, J is nonnegative for any $\mathbf{w}$ and w_0 because of the following:

$$J = \frac{1}{N}\sum_j f^2(\mathbf{w}^T\mathbf{x}^{(j)} + w_0) - (\bar{\mathbf{f}})^{2q}$$

$$\geq \frac{1}{N}\sum_j f^2(\mathbf{w}^T\mathbf{x}^{(j)} + w_0) - (\bar{\mathbf{f}})^2,$$

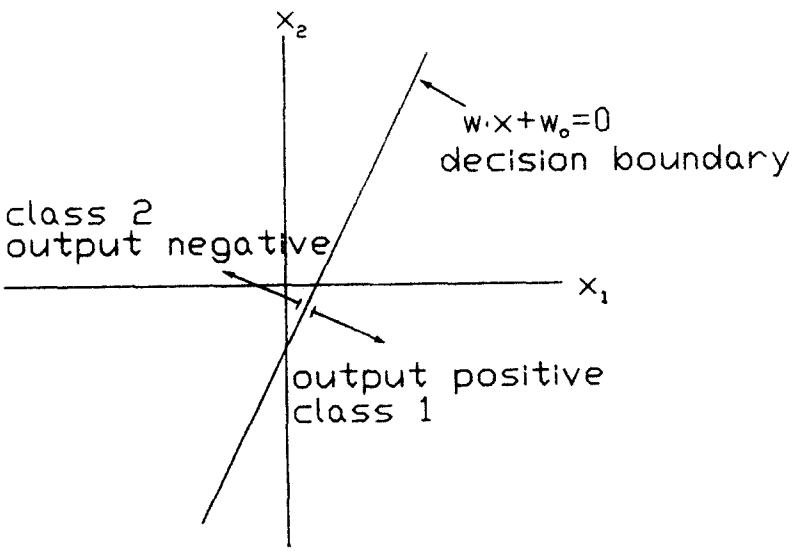

FIGURE 2. The decision regions and the decision boundary of a first-order neuron example.

where

$$\bar{f} = \frac{1}{N} \sum_j f(\mathbf{w}^T \mathbf{x}^{(j)} + w_0).$$

The inequality follows because $\bar{f}$, the average of the outputs, is less than 1 in magnitude and $q \geq 1$. Then we get

$$J \geq \frac{1}{N} \sum_j [f(\mathbf{w}^T \mathbf{x}^{(j)} + w_0) - \bar{f}]^2 \geq 0,$$

and hence the trivial solution of assigning all patterns to one class only is ruled out because it corresponds to a minimum, not a maximum, for J.

We iterate the weights in a steepest-ascent manner in an attempt to maximize the defined measure

$$\begin{aligned} \Delta \mathbf{w} &= \rho \frac{\partial J}{\partial \mathbf{w}} \\ &= \rho \frac{2}{N} \sum_j [f(\mathbf{w}^T \mathbf{x}^{(j)} + w_0) - q(\bar{f})^{2q-1}] \\ &\quad \times f'(\mathbf{w}^T \mathbf{x}^{(j)} + w_0) \mathbf{x}^{(j)} \end{aligned}$$

and

$$\begin{aligned} \Delta w_0 &= \rho \frac{\partial J}{\partial w_0} \\ &= \rho \frac{2}{N} \sum_j [f(\mathbf{w}^T \mathbf{x}^{(j)} + w_0) - q(\bar{f})^{2q-1}] \\ &\quad \times f'(\mathbf{w}^T \mathbf{x}^{(j)} + w_0), \end{aligned}$$

where ρ is the step size. The iteration is subject to the condition $\|\mathbf{w}\| \leq a$. Whenever after some iteration this condition is violated, $\mathbf{w}$ is projected back to the surface of the hypersphere $\|\mathbf{w}\| = a$ (simply by multiplying by $a/\|\mathbf{w}\|$). The term $f'(\mathbf{w}^T \mathbf{x}^{(j)} + w_0)$ in the update expression gives the patterns near the decision boundary more effect than the patterns far away from the decision boundary. This is because $f'(u)$ is high whenever u is near zero and goes to zero when the magnitude of u is large. Thus, what the method is essentially doing is iterating the weights in such a way as to move the decision boundary to a place sparse in patterns. The relative importance of patterns near the decision boundary increases for large a. This is because $\mathbf{w}$ tends to go to the surface of the ball $\|\mathbf{w}\| \leq a$. Large a will therefore result in the argument of f' being in general higher than for the case of small a, thus resulting in a smaller strip with high f' around the decision boundary. Of course, without the constraint $\mathbf{w}$ could grow in an unbounded fashion, resulting in $f(\mathbf{w}^T \mathbf{x}^{(j)} + w_0)$ being very near 1 or -1 for all j for most possible partitions, and we obtain therefore possibly bad solutions.

3. EXTENSIONS

We have shown in the previous section a method for training a linear classifier. To extend the analysis to the case of a curved decision boundary we use a higher-order neuron; in other words, the output is described by

$$y = f(w_0 + \sum_i w_i x_i + \cdots + \sum_{i_1 \leq \cdots \leq i_L} w_{i_1 \cdots i_L} x_{i_1} \cdots x_{i_L})$$

where x_i denotes the ith component of the vector $\mathbf{x}$ and L represents the order. Higher-order networks have been investigated by several researchers (refer, for example, to Psaltis and Park (1986), Psaltis, Park, & Hong (1988), and Giles & Maxwell (1987). They achieve simplicity of design while still having the capability of producing fairly sophisticated nonlinear decision boundaries.

The decision boundary for the higher-order neuron is given by the equation

$$w_0 + \sum_i w_i x_i + \cdots + \sum_{i_1 \leq \cdots \leq i_L} w_{i_1 \cdots i_L} x_{i_1} \cdots x_{i_L} = 0.$$

The higher-order case is essentially a linear classifier applied to augmented vectors of the form

$$(x_1, \cdots, x_n, \cdots, x_1^L, x_1^{L-1} x_2, \cdots, x_n^L)$$

(the superscripts denote here powers) using an augmented weight vector

$$(w_1, \cdots, w_n, \cdots, w_{1 \cdots 11}, w_{1 \cdots 12}, \cdots, w_{n \cdots nn}).$$

We can therefore apply basically the same training procedure described for the linear classifier case on the set of augmented vectors; in other words, we update the weights according to

$$\Delta w_{i_1 \cdots i_L} = \rho \frac{2}{N} \sum_j [f(u_j) - q(\bar{f})^{2q-1}] f'(u_j) x_{i_1}^{(j)} \cdots x_{i_L}^{(j)},$$

$$\Delta w_0 = \rho \frac{2}{N} \sum_j [f(u_j) - q(\bar{f})^{2q-1}] f'(u_j),$$

where $f(u_j)$ is the output of the neuron when applying the pattern $\mathbf{x}^{(j)}$, in other words,

$$u_j = w_0 + \sum_i w_i x_i^{(j)} + \cdots + \sum_{i_1 \leq \cdots \leq i_L} w_{i_1 \cdots i_L} x_{i_1}^{(j)} \cdots x_{i_L}^{(j)}$$

and

$$\bar{f} = \frac{1}{N} \sum_j f(u_j).$$

We have considered so far the two-class case. The extension to the multiclass case is as follows. We apply the previously described procedure (preferably the curved-boundary one)—partition the patterns into two groups. One group of patterns S^+ corresponds to positive outputs and represents an estimate of a collection of classes. The other group S^- corresponds to negative outputs and represents an estimate of the remaining classes. Then we consider the group S^+ separately (i.e., we use only the patterns in S^+ for training), and partition it further into two groups S^{++} *and* S^{+-} corresponding to positive and negative outputs respectively. Similarly, we par-

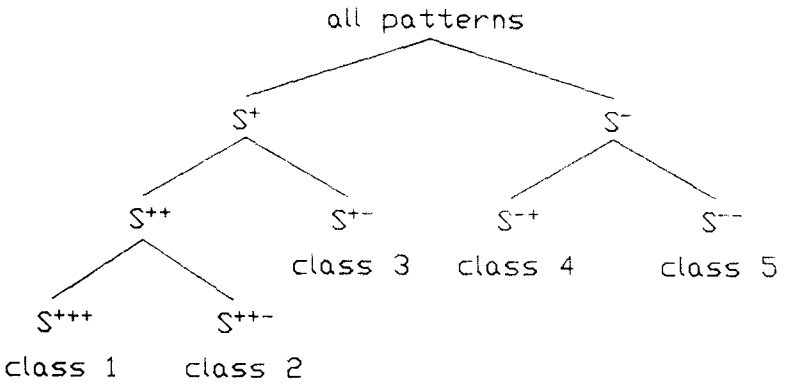

FIGURE 3. An example of the hierarchical design of the classifier.

tition S^- into S^{-+} and S^{--}. We continue in this hierarchical manner until we end up with the final classifier (see Figure 3 for an example). We stop the partitioning of a group whenever the criterion function J associated with the partition falls below a certain threshold. (One practically observes that J in general decreases slightly after each partition until one cluster remains whose partitioning results in a relatively abrupt decrease of J.) The final classifier consists now of several neurons, the sign pattern of whose outputs is an encoding of the class estimate of the presented pattern. Note that the output of the neuron responsible for breaking up some group plays a role in the encoding of only the classes contained in that group. Therefore, the encoding of a class could have a number of "don't cares." Refer to the next section for a constructive example of the extension to the multiclass case.

4. IMPLEMENTATION EXAMPLES

The new method is implemented on several examples. In the first three examples the patterns of each class are generated from bivariate Gaussian distributions. The first example is a two-class problem with a large overlap between the two classes. Figure 4 shows the resulting decision boundary when applying the new method using a first-order neuron. One observes that the obtained partition agrees to a large extent to that a human would estimate when attempting to decompose the two clusters visually. Regarding the second example, we have two classes with equal diagonal covariance matrices whose diagonal entries differ much, resulting in two long clusters. Figure 5 shows the results of the application of the proposed method using a first-order neuron. In another example we have a five-class problem with the means being on the corners and the center of a square. We used third-order neurons. Figure 6 shows the results. We ended up with four neurons partitioning the patterns. The first neuron is responsible for partitioning the whole collection of patterns into the two groups S^+ and S^-, separated by the boundary B_1; the second neuron partitions the group S^+ into the groups S^{++} and S^{+-}, separated by the boundary B_2; the third neuron divides S^{++} in S^{+++} and S^{++-} with boundary B_3; finally, the fourth neuron partitions S^- into S^{-+} and S^{--}, the boundary being B_4 (see also Figure 3). One observes that the method partitioned the patterns successfully. As a pattern classifier, the four neurons give the encoding of the class estimate of the presented pattern. The codes of the five classes are $+++X$, $++-X$, $+-XX$, $-XX+$, and $-XX-$, where the $+$'s and $-$'s represent the signs of the outputs and the X means don't care. In the last simulation experiment we test the performance of the method on a 15-di-

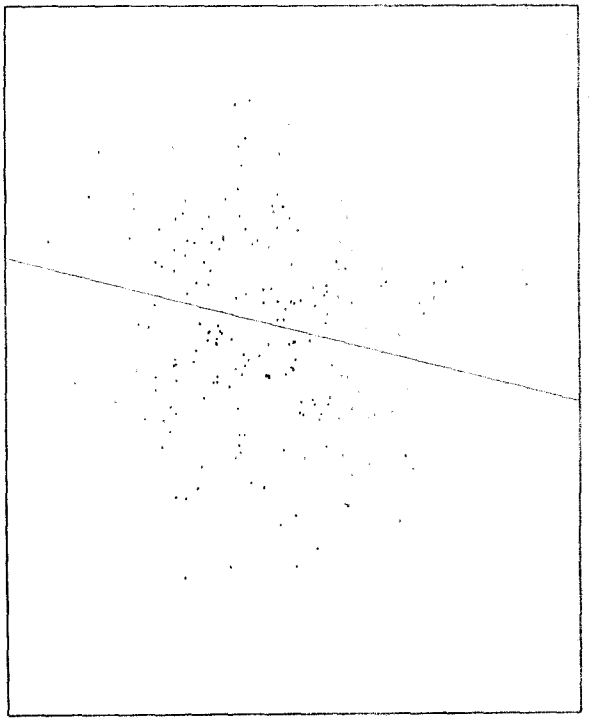

FIGURE 4. A two-cluster example with the decision boundary of the designed classifier.

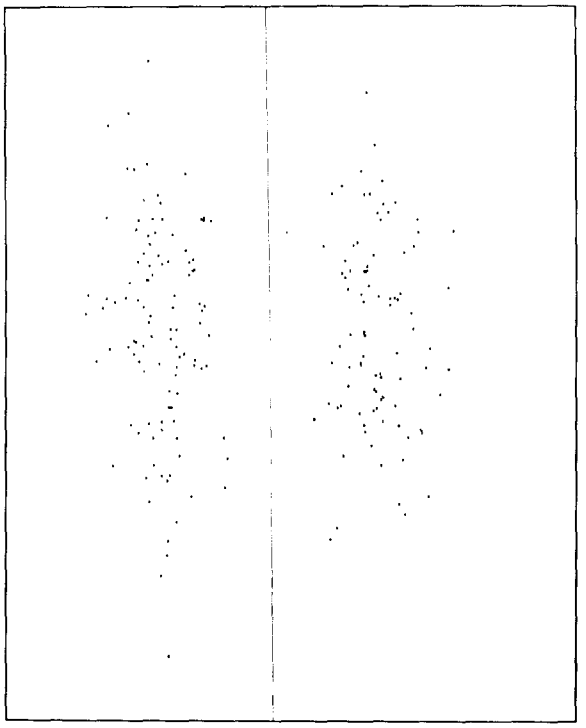

FIGURE 5. A two-cluster example with the decision boundary of the designed classifier.

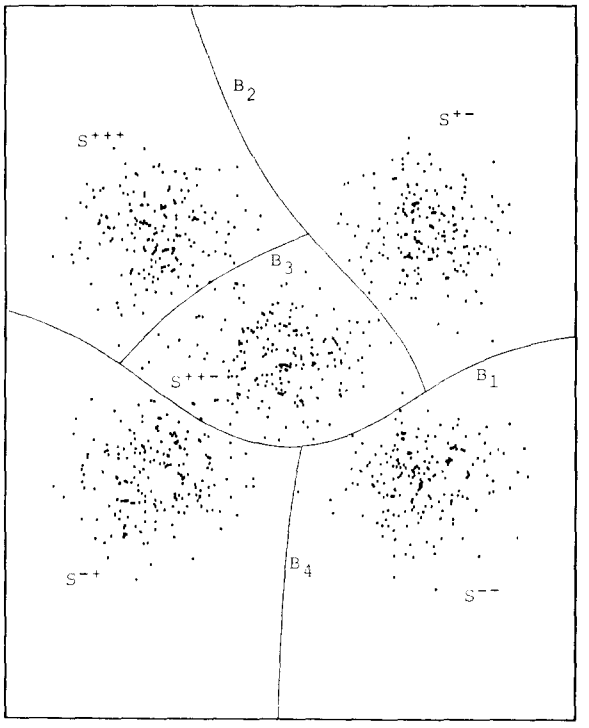

FIGURE 6. A five-class example showing the decision boundaries of the hierarchically designed classifier. There are four neurons associated with the boundaries B_1, B_2, B_3 and B_4.

mensional example having 30 clusters. The means of the clusters are generated using a uniform distribution in $[-1, 1]^{15}$. The inputs are Gaussian and uncorellated; the standard deviation is 0.15 for each input and for each class. Using second-order neurons, the new method resulted in the correct partitioning of the clusters, the number of neurons used for the partitioning being 28.

We remark that in all the previous problems, the weights are initialized by generating them randomly. A final important comment about the method is that when using a high-order neuron and a constant step size, it is imperative to scale the inputs so that their maximum magnitude are neither much higher nor much lower than 1. The reason is that otherwise, the higher-order terms will respectively either dominate or be dominated by the lower-order terms, and unsatisfactory results might be obtained.

5. CONCLUSION

A new neural unsupervised learning technique has been proposed in this work. This technique is based on the hierarchical partition of the patterns. Each partition corresponds to one neuron, which is in general a higher-order neuron. The partition is performed by iterating the neuron weights in an attempt to maximize a defined criterion function. The method is implemented on several examples and is found to give good results. The method is fast, as it takes typically from about 2 to 5 iterations to converge (by one iteration we mean one sweep through the set of training patterns). Although the proposed method is prone to get stuck in local minima, this did happen in our simulations in only very difficult problems and this problem could be solved by using gradient algorithms for searching for the global maximum, like the tunneling algorithm (Levy & Montalvo, 1985).

REFERENCES

Atiya, A. (1988). A neural unsupervised learning technique. In *Neural networks, Abstracts of the First Ann. INNS Meeting*, Boston, MA (Vol. 1, Supp. 1, p. 69).

Duda, R., & Hart, P. (1973). *Pattern classification and scene analysis*. New York: John Wiley and Sons.

Giles, C., & Maxwell, T. (1987). Learning, invariance, and generalization in high-order neural networks. *Applied Optics*, **26**(23), 4972–4978.

Grossberg, S. (1976). Adaptive pattern classification and universal recording: Part I. Parallel development and coding of neural feature detectors. *Biological Cybernetics*, **23**, 121–134.

Kohonen, T. (1984). *Self-organization and associative memory*. Berlin: Springer-Verlag.

Levy, A., & Montalvo, A. (1985). The Tunneling Algorithm for the global minimization of functions. *SIAM J. Sci. Stat. Comput.*, **6**(1), 15–29.

Psaltis, D., & Park, C. (1986). Nonlinear discriminant functions and associative memories. In J. Denker (Ed.), *AIP Conference Proceedings, Snowbird, Utah*. New York: American Institute of Physics.

Psaltis, D., Park, C., & Hong, J. (1988). Higher order associative memories and their optical implementations. *Neural Networks*, **1**(2), 149–163.