GENERALIZATION IN LAYERED CLASSIFICATION NEURAL NETWORKS

R. Jackson Marks II, Les E. Atlas and Seho Oh
Interactive Systems Design Lab, FT-10
University of Washington, Seattle, WA 98195

ABSTRACT

Artificial neural networks (ANN's) have been shown to be a potentially powerful approach to classification. As in nature's neural networks, training is performed by example rather than with rules. In this paper, we demonstrate how the use of arbitrary nonlinearities can improve the storage capacity of a class of layered classification ANN's (L-CANN's). The network's storage capacity is on the order of the number of neurons used to stimulate the response. L-CANN's can be trained by viewing the training data only once. Classification boundaries corresponding to maximum points of confusion, if known, can also be learned. Iteration is not required in the recall mode. The manner in which a network responds to data outside the training set can be straightforwardly evaluated. The L-CANN also has the ability to recognize the unfamiliarity of stimuli for which it was not trained.

INTRODUCTION

Artificial neural networks (ANN's) used as classifiers have a stimulus layer of neurons to provide the data input into the system and a response layer which provides the corresponding classification. In general, *linear* classifiers have the inability to separate complex classification regions. The observation that the perceptron was unable to perform a simple XOR operation was a major factor in the decrease in research into artificial neural networks (ANN's) in the sixties [1]. Specifically, visualize a unit square with its lower left corner at the origin. The XOR operation would assign a value of one to the origin and the (1,1) corner. The remaining two corners are assigned a value of zero. There exists no straight line (and therefore no linear classifier) with the ability to separate the ones from the zeros. ANN's ,of course, were never considered to be potential commercially viable architectures for XOR gates. The point is that if the linear ANN was unable to perform a simple XOR, there is little chance that it will perform successfully in more complex classification problems.

Linear classifier ANN's can be augmented to perform nonlinear classification by the use of a hidden layer of neurons. The states of the hidden neurons are typically some nonlinear function of the network's stimulation states such as a product [2], a weighted sum of states followed by a memoryless nonlinearity such as a sigmoid [3] or logic operations.

This developmental approach is used as a presentation order in this paper. First, we review a linear classifier that used *linear discriminant functions* [4-6] (also called *composite matched filters* or *linear combination filters*). The number of stimuli-response pairs that can be stored in such a network is shown to be on the order of the vector length of the data. Using arbitrary nonlinearities, we show that the storage capacity of the network can be arbitrarily increased by the use of hidden neurons the states of which are arbitrary nonlinear functions of the stimulus. The performance of the resulting *layered classification artificial neural network* (L-CANN) is favorably compared to those of other ANN's. Attributes considered include training dynamics, recall dynamics, capacity and generalization.

LINEAR DISCRIMINANT FUNCTION CLASSIFIERS

A set of stimuli vectors $\{s_n \mid 1 \leq n \leq N\}$ is to be made to correspond to a set of response vectors $\{r_n \mid 1 \leq n \leq N\}$. That is, we wish to design a classifier that will output, say, r_3 when the input is s_3. We define the stimulus and response matrices respectively as

$$\mathbf{R} = [\, r_1 \mid r_2 \mid \dots \mid r_N \,]$$

and

$$\mathbf{S} = [\, s_1 \mid s_2 \mid \dots \mid s_N \,].$$

Classification is then accomplished through the synthetic discriminant function matrix

$$\mathbf{C} = \mathbf{R}\,[\,\mathbf{S}^T \mathbf{S}\,]^{-1}\,\mathbf{S}^T \tag{1}$$

Note that $\mathbf{C}\ \mathbf{S} = \mathbf{R}$ and that, as a consequence,

$$\mathbf{C}\, s_n = r_n \,;\ 1 \leq n \leq N. \tag{2}$$

We have thus achieved our desired relational mapping.

What is the capacity of this linear classifier? That is, what is the maximum number of relational vector pairs that can be stored before the classifier no longer works as predicted in (2)? Note in (1) that we have made the assumption that the correlation matrix $\mathbf{S}^T\mathbf{S}$ is not singular. If the stimulus training vectors are of length L, then $\mathbf{S}$ is $L \times N$ and $\mathbf{S}^T\mathbf{S}$ is $N \times N$. If $N \leq L$ and $\mathbf{S}$ is of full column rank, then $\mathbf{S}^T\mathbf{S}$ is not singular. If, however, $N > L$, then $\mathbf{S}^T\mathbf{S}$ must be singular and the classifier is overdetermined. Thus, under the full rank assumption, we conclude that

$$N_{max} = L \tag{3}$$

That is, the number of relational pairs that can be stored faithfully cannot exceed the length of one of the stimulus vectors or, in the parlance of ANN's, the number of stimulus neurons.

For the XOR, we would require the $N = 4$ combinations of the $L = 2$ input bits. From (3), the linear synthetic discriminant function classifier will clearly not work.

CH2458-8/88/0000-0503$1.00 © 1988 IEEE

INCREASING THE CAPACITY WITH NONLINEARITIES

The capacity of the synthetic discriminant function classifier can be increased by increasing the length of the stimulus vectors. This can be done artificially. A stimulus vector s_n cannot be thus successfully lengthened by adding new elements that are linear combinations of previous elements. The rank of the corresponding augmented S matrix would not increase. We can, however, choose most any *non*linear combination and affect a rank increase. The first new element to be addended to s_n, for example, could be the square of the first element of s_n. The second new element could be the cosine of the sum of all the elements of s_n. For stimulus matrices containing only zeros and ones, logic operations can be used. Our only concern is that the rank of the augmented stimulus matrix increase with each nonlinearity. For any chosen set of nonlinearities, there does exist, of course, a training data set for which the newly generated states will not increase the matrix rank. The possibility of this happening, however, can be reduced by using extra neurons in the hidden layer for insurance.

Let η denote the nonlinear operation performed on the stimulus vector to increase its length from L to L_+. The augmented vector is:

$$s_{n+} = [\, s_n \mid a_n \,]^T$$

where

$$a_n = \eta\, s_n\,; \quad 1 \le n \le N.$$

The augmented stimulus matrix follows as

$$\mathbf{S}_+ = \begin{bmatrix} \mathbf{S} \\ \text{---} \\ \mathbf{A} \end{bmatrix}$$

where

$$\mathbf{A} = [\, a_1 \mid a_2 \mid \dots \mid a_N \,]$$

The augmented synthetic discriminant matrix follows as

$$\mathbf{C}_+ = \mathbf{R}\,[\, \mathbf{S}_+^T \mathbf{S}_+ \,]^{-1} \mathbf{S}_+^T.$$

If the number of stored responses, N, does not exceed L_+, then an input of s_{n+} produces and output of r_n.

EXAMPLE 1: Consider an XNOR with stimulus matrix:

$$\mathbf{S} = \begin{bmatrix} -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \end{bmatrix}$$

and response matrix

$$\mathbf{R} = [\; 1\; -1\; -1\; 1 \;].$$

Since $N = 4$, we require a minimum of two neurons in the hidden layer to make $L_+ = 4$. We arbitrarily choose the nonlinearities to be:

$$a_1 = 1 + \cos(s_1 + s_2)$$

and

$$a_2 = 1 + \cos(s_1 - s_2).$$

Therefore

$$\mathbf{A} = \begin{bmatrix} \phi & 2 & 2 & \phi \\ 2 & \phi & \phi & 2 \end{bmatrix}$$

where $\phi = 1 + \cos(2) = 0.5839$. It follows that

$$\mathbf{C} = [\; 0 \;\; 0 \; -\theta \;\; \theta \;]$$

where $\theta = 0.7061$. Therefore, in the neural network architecture shown in Figure 1, the interconnect values from the input to the output neurons are zero. As we could expect, multiplying the first or last column of $\mathbf{S}_+$ by $\mathbf{C}$ gives one. Use of either of the remaining two columns gives -1.

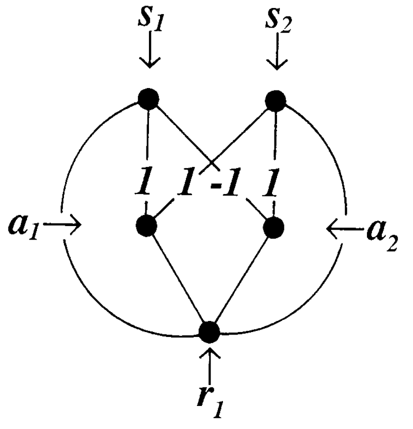

FIGURE 1: A NEURAL NETWORK ARCHITECTURE INTERPRETATION OF THE XNOR PROBLEM IN EXAMPLE 1. THE INTERCONNECTS TO THE OUTPUT NEURON ARE GIVEN BY THE $\mathbf{C}_+$ MATRIX.

A LAYERED CLASSIFIER NEURAL NETWORK

The number of relations that can be stored in the layered neural network in the previous section cannot exceed the number of neurons in the input and hidden layer. Indeed, if the number of neurons in the hidden layer is sufficiently large, we need not even consider connecting the input neurons to the output neurons. We denote the hidden states in such a L-CANN by $\{\, h_n \mid 1 \le n \le N \,\}$ where

$$h_n = \eta\, s_n\,; \quad 1 \le n \le N.$$

We form the hidden layer matrix

$$\mathbf{H} = [\, h_1 \mid h_2 \mid \dots \mid h_N \,].$$

The interconnects from the hidden to output neurons are then

$$\mathbf{C} = \mathbf{R}\,[\, \mathbf{H}^T \mathbf{H} \,]^{-1} \mathbf{H}^T. \tag{4}$$

The number of stored relations cannot exceed the number of hidden neurons.

EXAMPLE 2: We again consider the XNOR of *Example 1*. A minimum of N = four hidden neurons are required. We choose the following nonlinearities:

$$h_1 = \exp(s_1 + 2s_2)$$
$$h_1 = \exp(s_1 - 2s_2)$$
$$h_1 = \exp(-s_1 - 2s_2)$$
$$h_1 = \exp(-s_1 + 2s_2)$$

The corresponding neural network architecture is shown on Figure 2.

For the XOR training data, we have

$$\mathbf{H} = \begin{bmatrix} e^{-3} & e^{1} & e^{-1} & e^{3} \\ e^{1} & e^{-3} & e^{3} & e^{-1} \\ e^{3} & e^{-1} & e^{1} & e^{-3} \\ e^{-1} & e^{3} & e^{-3} & e^{1} \end{bmatrix}$$

from which we compute

$$\mathbf{C} = [\; 0.0585 \;\; -0.0587 \;\; 0.0586 \;\; -0.0577 \;].$$

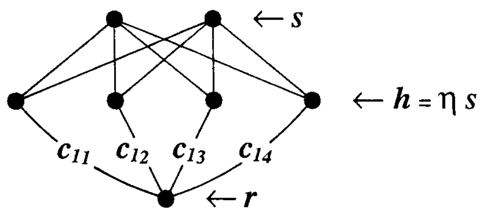

FIGURE 2: THE L-CANN DESCRIBED IN EXAMPLE 2 FOR GENERATING THE XNOR. THE STATES IN THE HIDDEN LAYER ARE EXPONENTIALS OF A WEIGHTED SUM OF THE INPUT STATES.

HIDDEN LAYER INTERCONNECTS

Thus far we have required only that the input neuron layer be connected to the hidden layer and that the hidden layer be connected to the output layer. We will, in addition, use interconnect values among the hidden neurons for two reasons. First, it allows the neural network to be easily trained and, second, lets the neural network evaluate whether it has been trained on a given input vector. An example of our fully trained L-CANN is shown in Figure 3 for two input and output neurons and three hidden neurons. The interconnects among the hidden layers are given by the projection matrix [7-11]

$$\mathbf{P} = \mathbf{H}\,[\,\mathbf{H}^T\mathbf{H}\,]^{-1}\,\mathbf{H}^T. \quad (5)$$

Multiplying any vector h by $\mathbf{P}$ results in a vector that is the linear combination of $\{h_n \mid 1 \le n \le N\}$ that is closest to h in the mean square sense or, equivalently, $\mathbf{P}\,h$ is the orthogonal projection of h onto the column space of $\mathbf{H}$.

The hidden layer interconnects can be used to ascertain whether an excitation vector s has been previously learned. First, we let the input tos

hidden layer interconnects and nonlinearities compute $h = \eta\, s$. The hidden layer interconnects can then perform an iteration to generate $\mathbf{P}\,h$ internal to the hidden layer. The states $\mathbf{P}\,h$ are then, on a node by node basis, subtracted from the previous states to compute

$$\varepsilon = (\mathbf{I} - \mathbf{P})\,h \quad (6)$$

If h had been previously seen in training data, then then ε is identically zero. Thus, if an appropriate metric on ε, e.g.

$$\|\,\varepsilon\,\|^2 = \varepsilon^T\varepsilon,$$

exceeds a prescribed threshold, then we conclude that the network has not been trained on the network's stimulus, s.

TRAINING

The hidden layer interconnects are invaluable when training the neural network. In establishing the neural interconnects, we with to avoid the matrix inversion operations explicitly used in the interconnect equations in (1) and (5). We, rather, would prefer that the interconnects be trained internally by exposing the network to the training data in a sequential manner. In this section, we will show that this indeed can be done and that, furthermore, the network has only to see each training vector only once.

For a given set of nonlinearities, assume that we have hidden layer interconnects $\mathbf{P}$ and hidden to output interconnects $\mathbf{C}$. We are presented with a new relational pair, s and r. We wish to update the interconnects so that a stimulus of s will result in a response s. We assume that the network's capacity is large enough to learn the new data. The interconnect updating requirement can be shown to be [11]

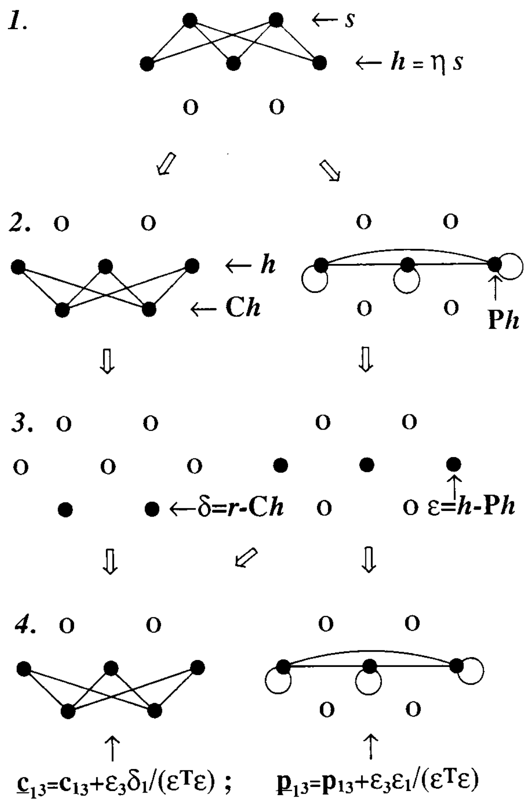
1.
$\leftarrow s$
$\leftarrow h = \eta\, s$
2.
$\leftarrow h$
$\leftarrow \mathbf{C}h$
$\mathbf{P}h$
3.
$\leftarrow \delta = r - \mathbf{C}h$
$\varepsilon = h - \mathbf{P}h$
4.
$\underline{c}_{13} = c_{13} + \varepsilon_3\delta_1/(\varepsilon^T\varepsilon)$;
$\underline{p}_{13} = p_{13} + \varepsilon_3\varepsilon_1/(\varepsilon^T\varepsilon)$

FIGURE 3: ILLUSTRATION OF A FOUR STEP TRAINING PROCEDURE FOR A L-CANN. THE PROCEDURE FOR UPDATING THE HIDDEN TO OUTPUT INTERCONNECTS IS IN THE RIGHT COLUMN AND THE HIDDEN INTERCONNECTS IN THE RIGHT COLUMN. EXCEPT FOR THE NEEDED COMPUTATION OF THE NORM OF THE HIDDEN LAYER ERROR VECTOR, ε, ALL OPERATIONS ARE PERFORMED BY THE NETWORK.

$$\underline{\mathbf{P}} = \mathbf{P} + \varepsilon\,\varepsilon^T/(\varepsilon^T\varepsilon)$$

and

$$\underline{\mathbf{C}} = \mathbf{C} + \delta\,\varepsilon^T/(\varepsilon^T\varepsilon)$$

where the underline denotes the updated interconnect matrix and

$$\delta = r - \mathbf{C}\,h \quad (7)$$

Except for the normalizing factor $\varepsilon^T\varepsilon$, all training can be performed internal to the neural network. The four step procedure is illustrated in Figure 3. First, the stimulus s is used to generate the hidden neural states. In the second step, the hidden layer interconnects are used to compute $\mathbf{P}h$ while, simultaneously, the hidden-to-output interconnects generate $\mathbf{C}h$ at the output nodes. Next, the computed states are subtracted from the imposed states at both the hidden and output layers. At the hidden layer, $\mathbf{P}h$ is subtracted from h and, at the output layer, $\mathbf{C}\,h$ is subtracted from h. The result is the error vectors defined respectively in (6) and (7). The states of the neurons in the hidden and output nodes are assigned these error values. In the fourth and final step, the neuron interconnects are updated. From (1), the interconnect between hidden neuron 3 and output neuron 1 is updated in accordance to

$$\underline{c}_{13} = c_{13} + \delta_1\,\varepsilon_3/(\varepsilon^T\varepsilon)$$

The interconnect is thus updated proportional to the product of the errors at the neurons that the interconnect connects. Similar updating is simultaneously performed among the hidden layer interconnects. For example

$$p_{13} = p_{13} + \varepsilon_1\ \varepsilon_3/(\varepsilon^T\ \varepsilon).$$

GENERALIZATION

Within the previously discussed constraints, and choice of nonlinearities will result in a neural network that will respond correctly to training data. The fashion in which the classifier responds to stimulus not in its training set is referred to as the manner in which the network *generalizes*. Equivalently, generalization is the manner in which the network interpolates among the trained points. While the choice of nonlinearities does not affect the manner in which the network responds to training data, it does affect the way the network generalizes.

The mathematics describing generalization for the L-CANN is straightforward. For a stimulus s, the response is

$$r = \mathbf{C}\,\eta\, s$$

Generalization over a region of interest is computed by allowing s to range over that region and computing the corresponding response.

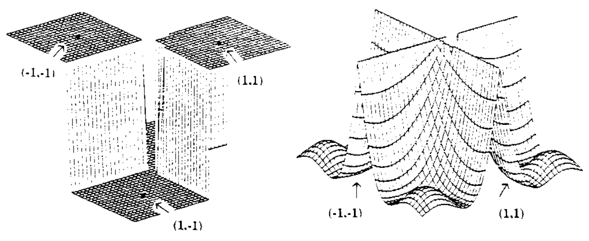

FIGURE 4: (LEFT) THE GENERALIZATION OF THE XNOR IN EXAMPLE 1 AS DESCRIBED IN EXAMPLE 3. (RIGHT) THE GENERALIZATION ERROR. THE ERROR AT THE TRAINING POINTS (THE SQUARE VERTICES) IS ZERO.

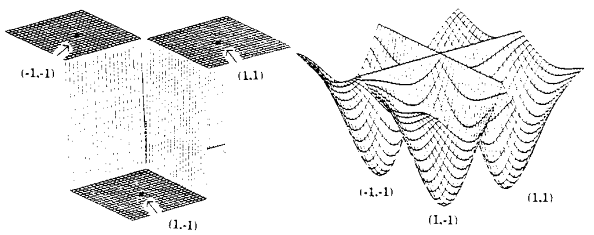

FIGURE 5: (LEFT) THE GENERALIZATION OF THE XNOR IN EXAMPLE 2 AS DESCRIBED IN EXAMPLE 4. (RIGHT) THE GENERALIZATION ERROR. THE ERROR AT THE TRAINING POINTS IS ZERO.

EXAMPLE 3: The generalization of the augmented XNOR L-CANN in *Example 1* is shown at the left of Figure 4. The *sign* of the output neuron state is shown as a function of the stimuli, (s_1,s_2). The right plot is the square of the difference of the top plot from the floating point value of the output neural state, and represents the *generalization error*. The generalization error at the vertices of the unit square are, by our design, zero.

EXAMPLE 4: The generalization of the augmented XNOR L-CANN in *Example 2* is shown at the left in of Figure 5. The *sign* of the output neuron state is shown as a function of the stimuli, (s_1,s_2). The right plot is the corresponding generalization error.

ACKNOWLEDGEMENTS

This work was supported by the Washington Technology Center at the University of Washington and, in part, by the SDIO/IST administered by ONR through the Optical Systems Lab at Texas Tech University in Lubbock. A discretionary gift from PhysioControl for the authors' work in neural networks is also gratefully appreciated. In addition, L.E. Atlas was supported by an NSF Presidential Young Investigator's Award.

REFERENCES

1. M. Minsky and S. Papert, **Perceptrons**, The MIT Press, Cambridge, Massachusetts, 1972.

2. C. Giles and T. Maxwell "Learning, invariance, and generalization in high-order neural networks", **Applied-Optics**, vol. 26, p.4972 (1987).

3. D.E. Rumelhart and J.L. McClelland, **Parallel Distributed Processing**, The MIT Press, Cambridge, Massachusetts, 1986.

4. R.J. Marks II and L.E. Atlas "Composite matched filtering with error correction", **Optics Letters**, vol. 12, pp.135-137 (1987).

5. R.J. Marks II, J.A. Ritcey, L.E. Atlas and K.F. Cheung "Composite matched filter output partitioning", **Applied Optics**, vol. 26, pp.2274-2278 (1987).

6. K.F. Cheung, L.E. Atlas, J.A. Ritcey, C.A. Green and R.J. Marks II "A comparison of conventional and composite matched filters with error correction", **Applied Optics**, vol. 26, pp.4235-4239 (1987).

7. T. Kohonen, **Self-Organization and Associative Memory**, Springer-Verlag, Berlin, 1984.

8. R.J. Marks II "A class of continuous level associative memory neural nets", **Applied Optics**, vol. 26, pp.2005-2009 (1987).

9. K.F. Cheung, R.J. Marks II and L.E. Atlas "Neural net associative memories based on convex set projections", **Proceedings of the IEEE First International Conference on Neural Networks**, San Diego, June 1987.

10. R.J. Marks II, L.E. Atlas, S. Oh and J.A. Ritcey "The performance of convex set projection based neural networks", **Neural Information Processing Systems - Natural and Synthetic**, (American Institute of Physics Press, 1988).

11. G. Eichmann and M. Stojancic "Superresolving signal and image restoration using a linear associative memory", **Applied Optics,** vol. 26, p.1911 (1987).