Learning Phenomena In Layered Neural Networks

By
Prof. Bernard Widrow, Dept. of Electrical Engineering, Stanford University
Capt. Rodney G. Winter, USAF, Dept. of Electrical Engineering, Stanford University
Robert A. Baxter, Dept. of Electrical Engineering, Stanford University

Published in the Proceedings of the IEEE First Annual International Conference on Neural Networks, June 1987.

IEEE Catalog #87TH0191-7

LEARNING PHENOMENA IN LAYERED NEURAL NETWORKS

By

Prof. Bernard Widrow, Dept. of Electrical Engineering, Stanford University
Capt. Rodney G. Winter, USAF, Dept. of Electrical Engineering, Stanford University
Robert A. Baxter, Dept. of Electrical Engineering, Stanford University

Abstract

A concept for pattern recognition is proposed involving the use of an "invariance net" followed by a trainable classifer. The invariance net can be trained or designed to produce a set of outputs that are insensitive to translation, rotation, scale change, perspective change, etc., of the retinal input pattern. The outputs of the invariance net are scrambled. When fed to the trainable classifier, the final outputs are descrambled and the original patterns are reproduced in standard position, orientation, scale, etc. It is expected that the same basic approach will be effective for speech recognition, where a recognition system will need to be insensitive to certain aspects of speech signals and at the same time sensitive to other aspects of speech.

The entire recognition system is a layered MADALINE structure. The ability to adapt a multilayered neural net is fundamental. A new MADALINE adaptation rule is proposed for layered nets which is an extension of the ancient (1960's) MADALINE rule.

Introduction

Networks of neural elements can be utilized to construct trainable decision-making systems. The basic building block is the "adaptive linear neuron" or ADALINE [1] shown in Figure 1. The neuron has an input signal vector $X_k = [x_0 x_{1_k} x_{2_k} \cdots x_{n_k}]$, whose components are weighted by a set of coefficients. The weight vector is $W = [w_0 w_1 w_2 \cdots w_n]^T$. The bias weight w_0, connected to a constant +1 input, controls the threshold level. An analog output is produced as the inner product $y_k = X_k^T W = W^T X_k$. The pattern number is k.

Decisions are made by a 2-level quantizer providing a binary ± 1 output $q_k = SGN(y_k)$. The components of X_k could be analog or binary. The desired response, another input which is only used during the training process, could also be analog or binary. In neural networks, all inputs and outputs are generally binary and are preferred to be ± 1 rather than the unsymmetrical 0,1. The weights are essentially continuously variable, and can take on negative as well as positive values. An adaptive algorithm automatically adjusts the weights, thereby controlling the decision making function of the neuron.

Assume that the input patterns are binary, and that the desired response is binary. During the training process, the weights are adjusted so that the neuron responses to the input patterns will be as close as possible to their respective desired responses. Often this is done by adjusting the weights in accord with a least squares adaptation algorithm (the LMS algorithm [1], often called the Widrow-Hoff Delta Rule [2]) that minimizes the sum of the squares of the errors over the training set. The error is defined as the difference between the desired response and the analog output. Each training pattern is inserted along with its desired response, the known binary classification for this pattern. Once the neuron is trained, its responses can be tested by applying unknown input patterns. Generalization is considered to be successful if the neuron responds correctly, with high probability, to input patterns that were not included in the training set.

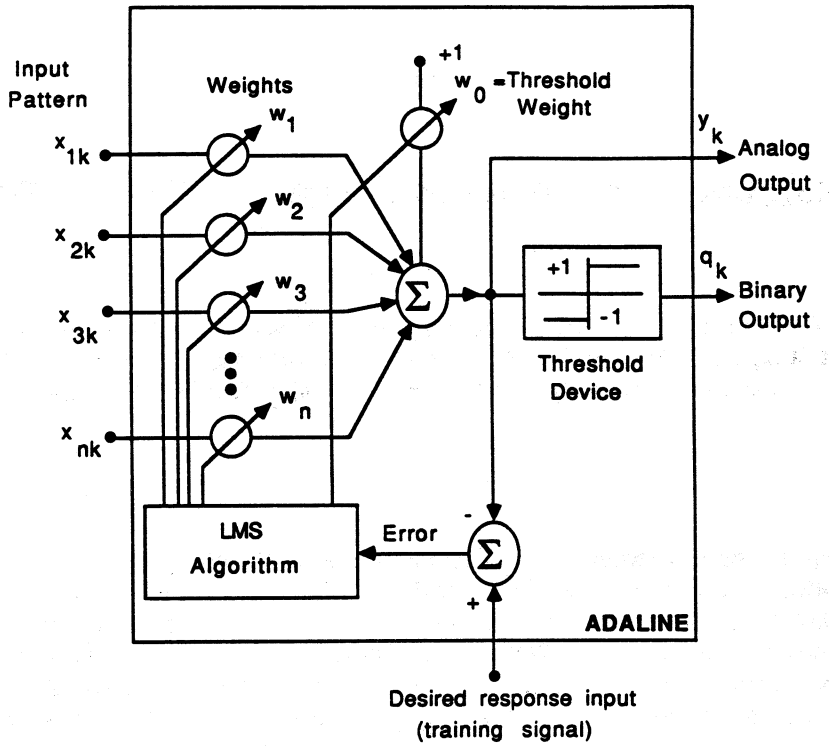

Figure 1. An adaptive linear neuron (ADALINE)

Although the LMS algorithm originated in the field of neural nets, its greatest impact today is in the field of adaptive signal processing [3]. Commercial applications are in adaptive equalizers for high-speed digital modems, and in adaptive echo cancellers for long-distance telephone circuits and satelite channels. The LMS algorithm is the most widely used adaptive algorithm in the world.

Linear Separability

With n binary inputs and a single binary output, a neuron is capable of implementing certain logic functions. There are 2^n possible input patterns. A general logic implementation would be capable of classifying each pattern as either +1 or -1, in accord with the desired response. Thus, there are 2^{2^n} possible logic functions connecting n inputs to a single output. A single neuron is capable of realizing only a small subset of these functions, known as the linearly separable logic functions [4]. These are the set of logic functions that can be obtained with all possible settings of the weight values.

In Figure 2, a two-input neuron is shown. In Figure 3, all possible binary inputs for a two-input neuron are shown in pattern vector space. The neuron separates the input patterns into two categories, depending on the values of the input-signal weights and the bias weight. The critical thresholding condition occurs when the analog response y equals zero:

$$y = x_1 w_1 + x_2 w_2 + w_0 = 0 \tag{1}$$

$$\therefore x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2} x_1 \ . \tag{2}$$

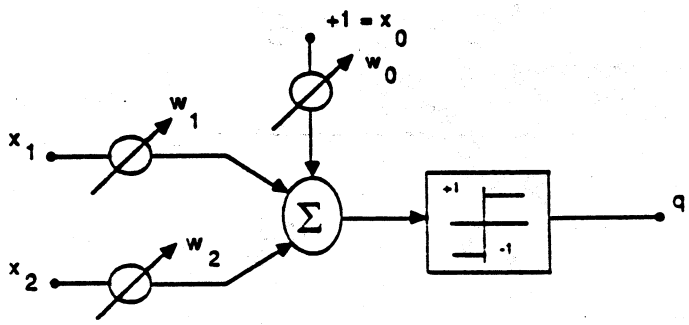

Figure 2. A Two-Input Neuron

In Figure 3, the separating line has slope and intercept of

$$slope = -\frac{w_1}{w_2} \; ; \; intercept = -\frac{w_0}{w_2} \tag{3}$$

The three weights determine slope, intercept, and the side of the separating line that corresponds to a positive output. As sketched in Figure 3, the binary inputs are classified as follows:

$$\begin{array}{l}(+1,+1) \rightarrow +1 \\ (+1,-1) \rightarrow +1 \\ (-1,-1) \rightarrow +1 \\ (-1,+1) \rightarrow -1\end{array} \tag{4}$$

This is an example of a linearly separable function. An example of a non-linearly separable logic function with two inputs is the following:

$$\begin{array}{l}(+1,+1) \rightarrow +1 \\ (+1,-1) \rightarrow -1 \\ (-1,-1) \rightarrow +1 \\ (-1,+1) \rightarrow -1\end{array} \tag{5}$$

With two inputs, almost all possible logic functions can be realized by the neuron. With many inputs, the pattern vector space is dichotomized by a hyperplane. However, only a small fraction of all possible logic functions are linearly separable. Since the single neuron cannot realize most functions, combinations of neurons can be used to achieve more intricate separators than simple hyperplanes.

Nonlinear Separability

Nonlinear functions of the inputs applied to the single neuron can also yield more intricate decision boundaries. Consider the system illustrated in Figure 4. The thresholding condition is:

$$y = w_0 + x_1^2 w_{11} + x_1 x_2 w_{12} + x_2 w_2 + x_2^2 w_{22} = 0 \tag{6}$$

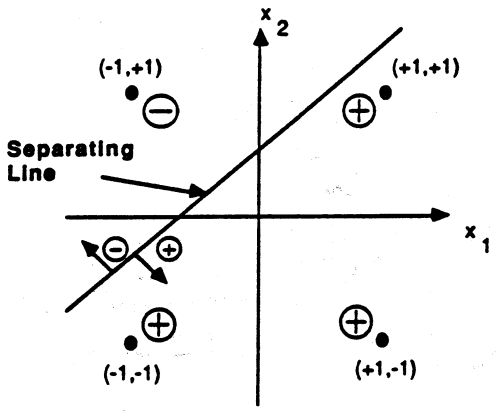

Figure 3. Separating line in pattern space

With proper choice of the weights, the separating boundary in pattern space can be established as shown for example in Figure 5. The non-linearly separable function (5) can be realized by this configuration of neuron with nonlinearities applied to the inputs. Of course, with suitable choice of the weight values, all of the linearly separable functions are also realizable. The usage of such nonlinearities can be generalized for high degree polynomial functions and cross product functions of the inputs. The idea can be further generalized for more inputs than two. One of the first works in this area was done by D. F. Specht [5,6] at Stanford in the 1960's, and later by Ivankhnenko [7] in the 1970's.

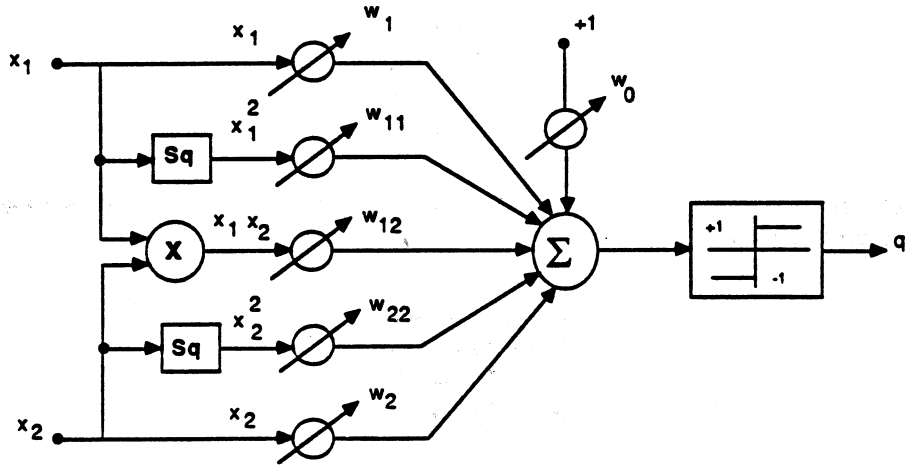

Figure 4. A neuron with inputs mapped through nonlinearities.

MADALINE Networks

Another approach to the implementation of non-linearly separable logic functions was initiated at Stanford by M. E. Hoff, Jr. [8] and W. C. Ridgway III [9] in the early 1960's. Retinal inputs were connected to adaptive neurons in a layer. Their outputs were connected to a fixed logic device providing the system output. Methods for adapting such nets were developed at that time. An example of such a network is shown in Figure 6. Two ADALINES are connected to an AND logic device to provide an output. Systems of this type were called MADALINES, (many ADALINES). Today such systems would be called neural nets.

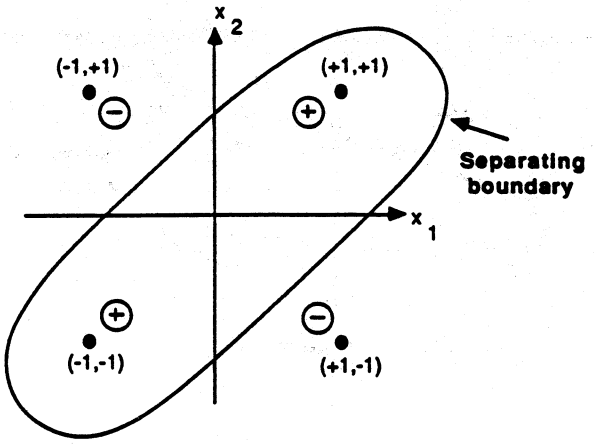

Figure 5. An elliptical separating boundary for non-linearly separable function realization.

With weights suitably chosen, the separating boundary in pattern space for the system of Figure 6 would be as shown in Figure 7. This separating boundary also implements the non-linearly separable logic function (5).

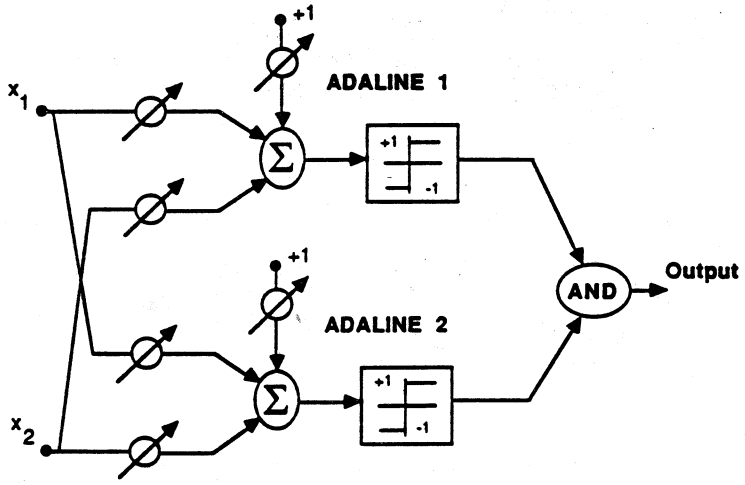

Figure 6. A two-neuron form of MADALINE

MADALINES were constructed with many more inputs, with many more neurons in the first layer, and with various fixed logic devices in the second layer such as OR, MAJORITY, and AND. These three functions are in themselves threshold logic functions, as illustrated in Figure 8. The weights given will implement these functions, but they are not unique.

Layered Neural Nets

The MADALINES of the 1960's had adaptive first layers and fixed threshold functions for the second (the output) layers. The neural nets of the 1980's have many layers, and all layers are adaptive. The most well-known multilayer work is by Rumelhart, et al. [2]. A sample network architecture is shown in Figure 9, with 6 inputs, 3 layers, and 2 outputs.

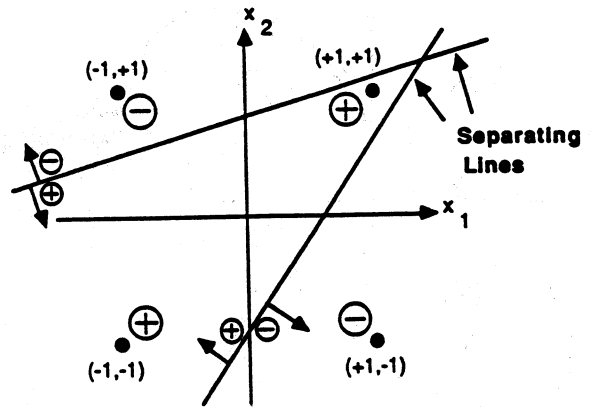

Figure 7. Separating boundaries for MADALINE of Figure 6.

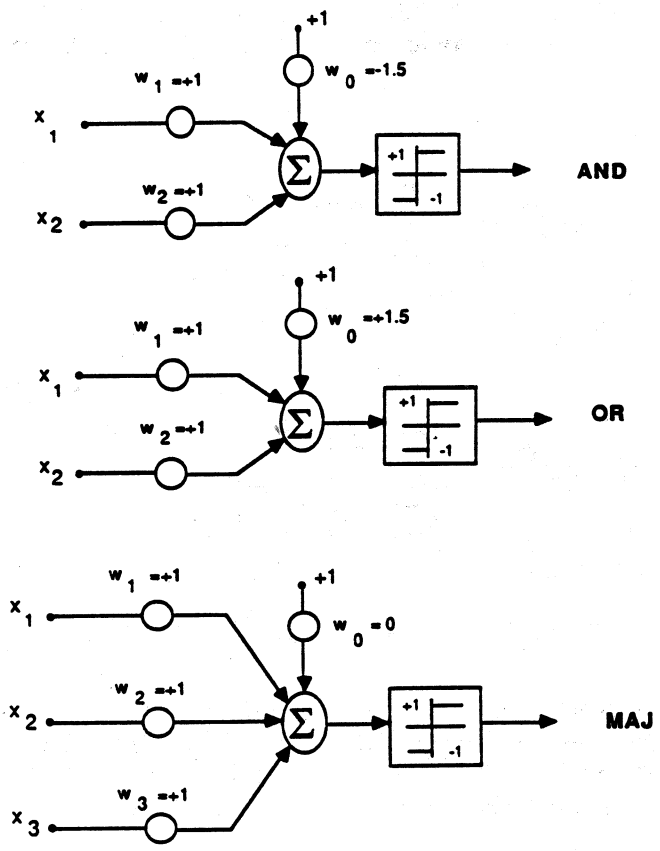

Figure 8. Neuronal implementation of AND, OR, and MAJ logic functions.

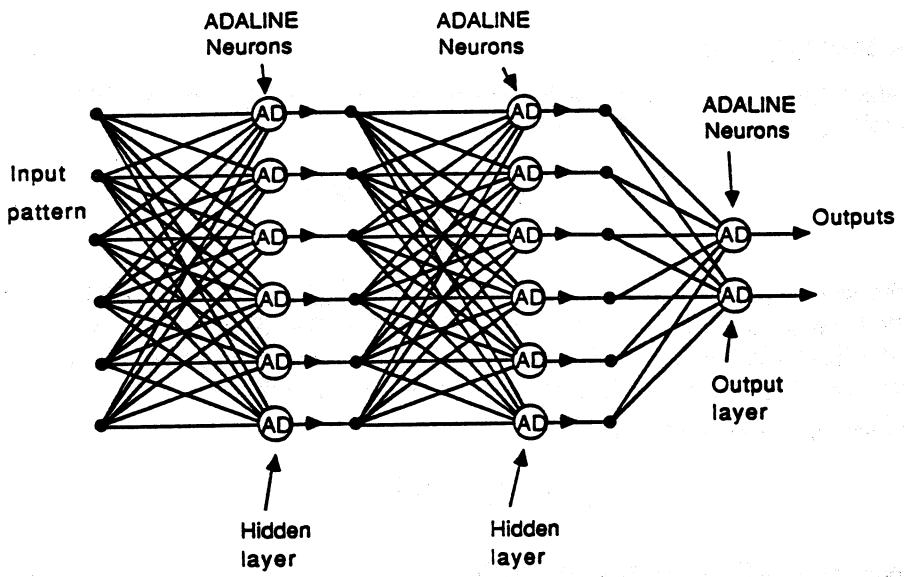

Figure 9. A three-layer adaptive neural network.

It is a simple matter to adapt the neurons in the output layer, since the desired responses for the entire network (which are given with each input training pattern) are the desired responses for the corresponding output neurons. Given inputs and desired responses, adaptation is a straightforward exercise of the LMS algorithm. The fundamental difficulty associated with adaptation of a layered network lies in obtaining desired responses for the neurons in the layers other than the output layer. The back propagation algorithm (reported earliest by David Parker [10]) is one method for establishing desired responses for the neurons in the "hidden layers," those layers whose neuronal outputs do not appear directly at the system output (refer to Figure 9). There is nothing unique about the choice of desired outputs.

Generalization in layered networks is a key issue. The question is how well do multilayered networks perform with inputs that were not specifically trained in. Most of the work in the field deals with learning the training patterns. The question of generalization will be important and some good examples are being developed where useful generalizations take place. Many different algorithms will be needed for the adaptation of multilayered networks to produce required generalizations. Without generalization, neural nets will be of little engineering significance. Meerly learning the training patterns can be accomplished by storing these patterns and their associated desired responses in a look-up table.

The layered networks of Parker and Rumelhart et al., utilize neuronal elements like the ADALINE of Figure 1, except that the quantizer or threshold device is a soft limiting "sigmoid" function rather than the hard limiting "signum" function of the ADALINE. The various back propagation algorithms for adapting layered networks of neurons require differentiability along the signal paths of the network, and cannot work with the hard limiter of the ADALINE element. The sigmoid function has the necessary differentiability, but presents implementational difficulties if the neural net is to be ultimately constructed digitally. For this reason, a new algorithm was developed for adaptation of layered networks of ADALINE neurons with hard limiting quantizers. The new algorithm is an extension of the original MADALINE adaptation rule [11,12]. The idea is to adapt the network to properly respond to the newest input pattern while minimally disturbing the responses already trained in for the previous input patterns. Unless this principle is practiced, it is difficult for the network to simultaneously store all of the required pattern responses.

LMS or Widrow-Hoff Delta Rule

The LMS algorithm applied to the adaptation of the weights of a single neuron embodies the minimal disturbance principle. This algorithm can be written as

$$W_{k+1} = W_k + \frac{\alpha}{|X_k|^2} \varepsilon_k X_k \ , \tag{7}$$

where W_{k+1} is the next value of the weight vector, W_k is the present value of the weight vector, X_k is the present input pattern vector, and ε_k is the present error (i.e., the difference between the desired response and the analog output before adaptation). With each adapt cycle, the above recursion formula is applied, and the error is reduced as a result by the fraction α. This can be demonstrated as follows.

At the kth interaction cycle, the error is

$$\varepsilon_k = d_k - X_k^T W_k \ . \tag{8}$$

The error is changed (reduced) by changing the weights

$$\Delta\varepsilon_k = \Delta(d_k - X_k^T W_k) = - X_k^T \Delta W_k \ . \tag{9}$$

In accord with the LMS rule (7), the weight change is

$$\Delta W_k = W_{k+1} - W_k = \frac{\alpha}{|X_k|^2} \varepsilon_k X_k \ . \tag{10}$$

Combining (9) and (10) we obtain

$$\begin{aligned} \Delta\varepsilon_k &= - X_k^T \frac{\alpha}{|X_k|^2} \varepsilon_k X_k \\ &= - X_k^T X_k \frac{\alpha}{|X_k|^2} \varepsilon_k \\ &= - \alpha\varepsilon_k \ . \end{aligned} \tag{11}$$

Therefore, the error is reduced by a factor of α as the weights are changed while holding the input pattern fixed. Putting in a new input pattern starts the next adapt cycle. The next error is then reduced by a factor of α , and the process continues. The choice of α controls stability and speed of convergence. Stability requires that

$$2 > \alpha > 0 \tag{12}$$

Making α greater than 1 generally does not make sense, since the error would be overcorrected. Total error correction comes with $\alpha = 1$. A generally useful range for α is

$$1.0 > \alpha > 0.1 \tag{13}$$

Figure 10 gives a geometric picture of how the LMS rule works. One can see that W_{k+1} equals W_k plus ΔW_k in accord with (10), and that ΔW_k is parallel with the input pattern vector X_k also in accord with (10). The change in the error will be equal to the negative dot product of X_k with ΔW_k, in accord with (9).

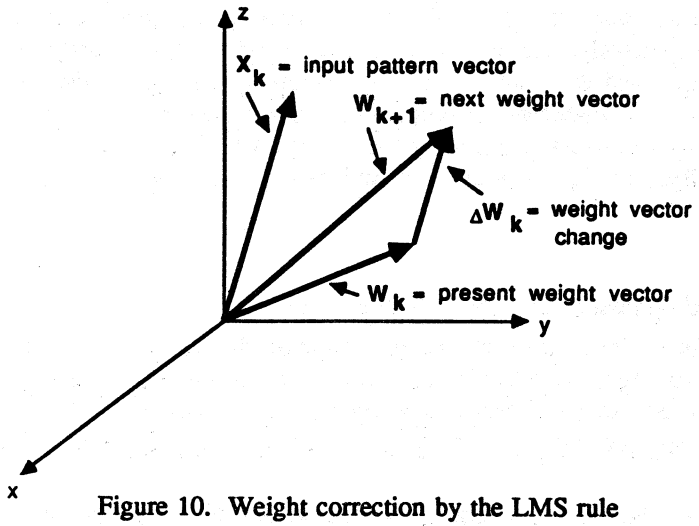

Figure 10. Weight correction by the LMS rule

By making the weight change vector ΔW_k colinear with the input pattern X_k, it is clear that one accomplishes the needed error correction with the smallest weight vector change. Thus, when adapting to respond properly to a new input pattern, the responses to previous training patterns are, on the average, minimally disturbed. The algorithm also minimizes mean square error [3].

Adaptation of Layered Neural Nets

Applying the minimal disturbance principle to the adaptation of the layered neural network of Figure 9 can be accomplished in the following manner. Insert a retinal pattern and its associated desired responses to the input nodes. Let all input signals and output signals be ±1 binary. The training objective is to reduce the number of errors to as low a level as possible. Accordingly, when the first training pattern is presented to the neural network, the first layer will be adapted as required to reduce the number of response errors at the final output layer. In accord with the minimal disturbance principle, the first layer neuron whose analog response is closest to zero is given a trial adaptation in the direction to reverse its binary output. When the reversal takes place, the second layer inputs change, the second layer outputs change, and consequently the network outputs change. Check to see if this reduced the number of errors. If so, accept the trial change. If not, restore the weights to their previous values and trial adapt the neuron whose analog response is next closest to zero, reversing its response. If this reduces the number of output errors, accept the change. If not, restore the weights and go on to adaptively switch the neuron with analog response next closest to zero, and so on. Only adapt neurons whose analog responses are close to zero, disturbing the neurons as little as possible. After adapting all of the first layer neurons whose analog responses were within a specified closeness to zero (for example, within the range ±.25) and whose output reversals reduced the number of network output errors, choose these neurons in pair combinations, and make trial adaptations

which can be accepted if output errors are reduced. After adapting the first layer neurons in singles, pairs, triples, etc., up to a predetermined limit in combination size, the second layer can be adapted to further reduce the number of network output errors. The method of choosing the neurons to be adapted in the second layer is the same as that for the first layer. If further error reduction is needed, the output layer can be adapted. This is straightforward, since the network desired responses are the desired responses for the corresponding output layer neural elements. After adapting the output layer, the responses will be correct. The next input pattern vector and its associated desired responses are then applied to the neural network and the adaptive process resumes.

When training the network to respond correctly to the various input patterns, the "golden rule" is: *give the responsibility to the neuron or neurons that can most easily assume it.* In other words, *don't rock the boat* any more than necessary to achieve the desired training objective. This minimal disturbance algorithm has been tested extensively, and appears to converge and behave robustly in all cases. It appears to be a very useful algorithm and does not require differentiability throughout the net. A great deal of effort will be required to derive its mathematical properties. To simulate it and make it work is straightforward and gives insight into its behavioral characteristics. A parallel effort is contemplated, (a) to analyze the algorithm mathematically, and (b) to improve it and explore its application to practical problems on an empirical basis.

Application of Layered Networks to Pattern Recognition

It would be useful to devise a neural net configuration that could be trained to classify an important set of training patterns as required, but have these responses be invariant to left-right, up-down translation within the field of view, and to be invariant to rotation and scale change, without necessitating the training of the system with the specific training patterns of interest in all combinations of translation, rotation, and scale.

The first step is to show that a neural network structure exists having these properties, and then we will determine training algorithms to achieve the desired objectives.

Invariance to Up-Down, Left-Right Pattern Translation

Figure 11 shows a neural network configuration (a "slab" of neurons) that could be used to map a retinal image into a single-bit output such that, with proper weights in the neurons of the network, the response will be insensitive to left-right translation and/or up-down translation. The same slab structure can be replicated, with different weights, to allow the retinal pattern to be independently mapped into additional single-bit outputs, all insensitive to left-right, up-down translation.

The general idea is illustrated in Figure 12. The retinal image having a given number of pixels can be mapped through an array of slabs into a different image that could have the same number of pixels or possibly more pixels, depending on the number of slabs used. In any event, the mapped image would be insensitive to up-down, left-right translation of the original image. The mapped image in Figure 12 is fed to a set of ADALINE neurons that can be easily trained to provide output responses to the original image as required. These output responses would classify the original input images and would at the same time be insensitive to their left-right, up-down translations.

In the systems of Figures 11 and 12, the elements labeled "AD" are ADALINES. Those labeled "MAJ" are majority vote-takers. (If the number of input lines to MAJ is even and there is a tie vote, these elements are biased to give a positive response.) The AD elements are adaptive neurons and the MAJ elements are fixed neurons. Figure 13 shows the structure of an MAJ element. All input signals are weighted equally. The bias weight is small and positive.

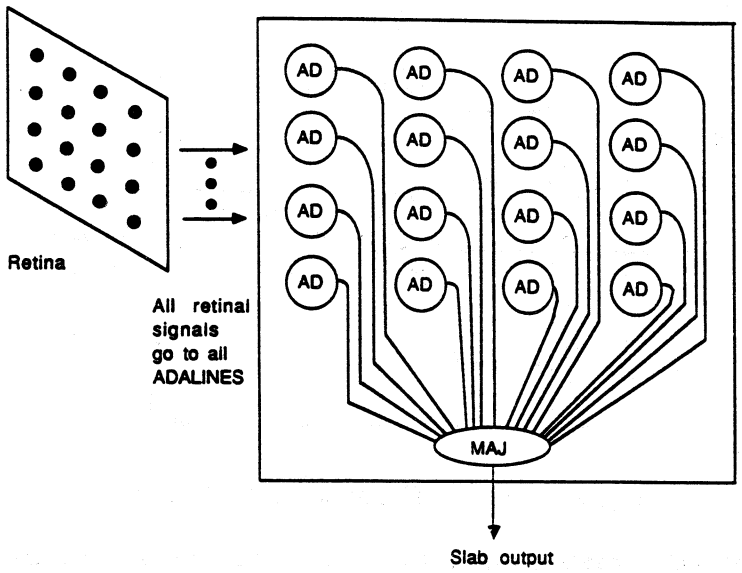

Figure 11. One slab of a left-right, up-down translation invariant network.

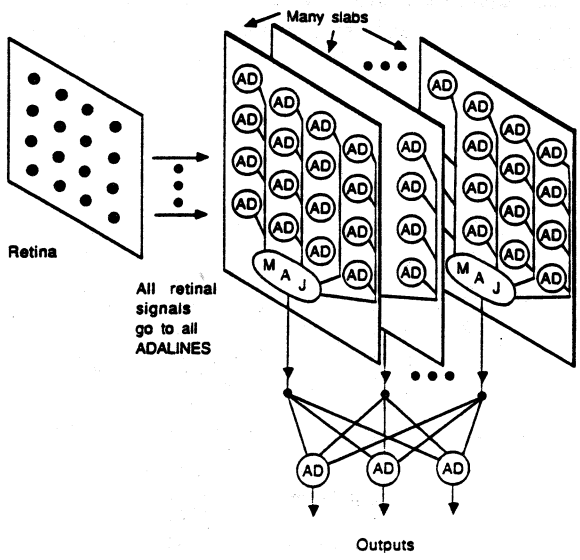

Figure 12. A trainable pattern classification system whose response is insensitive to left-right, up-down translation.

How the weights are structured in the system of Figure 11 to cause the output to be insensitive to left-right and up-down translation needs some further explanation. Consider the diagram of Figure 14. This system is insensitive to up-down translation. Let the weights of each ADALINE in Figure 14 be arranged in a square array. Let the corresponding retinal pixels also be arrayed in a square pattern. Let the array of weights of the topmost ADALINE be designated by the square matrix (W_1). Let the array of weights of the next lower ADALINE be $T_{D1}(W_1)$. The operator T_{D1} represents "translate down one." This set of weights is the same as that of the topmost ADALINE, except that they are en masse translated down by one pixel.

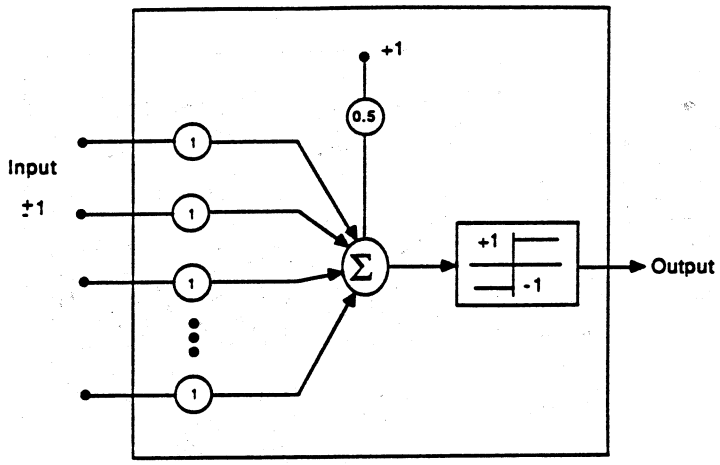

Figure 13. An "MAJ" element - a fixed neuron.

The bottom row is wrapped around to comprise the top row. The patterns on the retina itself are wrapped around on a cylinder when they undergo translation. The weights of the next lower ADALINE are $T_{D2}(W_1)$, and those of the next lower ADALINE are $T_{D3}(W_1)$. Since the outputs of the four ADALINES are all equally weighted by the MAJ element, translating the input pattern up-down on the retina will have no effect on the MAJ element output. As the input pattern is moved up or down on the retina, the roles of the various ADALINES interchange, giving a consistent MAJ output.

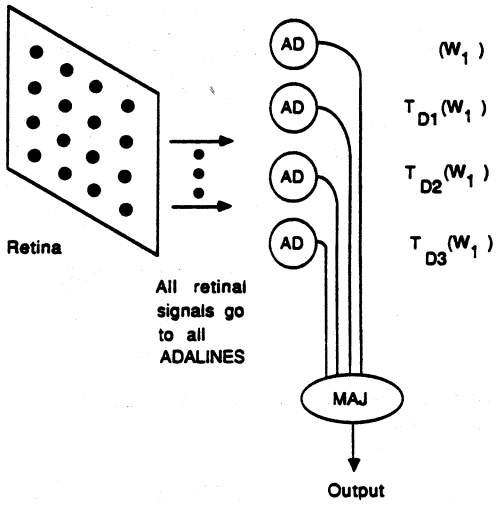

Figure 14. Training insensitivity to up-down translation.

The network of ADALINES of Figure 14 is replicated on the slab of Figure 11. Let the first column of weights in Figure 11 be chosen like the weights of the column in Figure 14. Let the second column of weights be chosen as

$$\begin{bmatrix} T_{R1}(W_1) \\ T_{R1}T_{D1}(W_1) \\ T_{R1}T_{D2}(W_1) \\ T_{R1}T_{D3}(W_1) \end{bmatrix} \tag{14}$$

The topmost weights of this column are the weights (W_1) translated right one pixel. The next lower set of weights are translated right one, translated down one, and so forth. The pattern of weights for the entire array of ADALINES of Figure 11 is given by

$$\begin{bmatrix} (W_1) & T_{R1}(W_1) & T_{R2}(W_1) & T_{R3}(W_1) \\ T_{D1}(W_1) & T_{R1}T_{D1}(W_1) & T_{R2}T_{D1}(W_1) & T_{R3}T_{D1}(W_1) \\ T_{D2}(W_1) & T_{R1}T_{D2}(W_1) & T_{R2}T_{D2}(W_1) & T_{R3}T_{D2}(W_1) \\ T_{D3}(W_1) & T_{R1}T_{D3}(W_1) & T_{R2}T_{D3}(W_1) & T_{R3}T_{D3}(W_1) \end{bmatrix} \tag{15}$$

From column to column, the weight patterns are translated left-right. From row to row, the weight patterns are translated up-down. Since the outputs of all of the ADALINE units are equally weighted and dealt with symmetrically by the output MAJ element, it is clear that the output of this system will be insensitive to both left-right and up-down translation.

The set of weights (W_1) can be randomly chosen. Once chosen, they can be translated according to (15) to fill out the array of weights for the system of Figure 11. This array of weights can be incorporated as the weights for the first slab of ADALINES shown in Figure 12. The weights for the second slab would require the same translational symmetries, but be based on the randomly chosen set of weights (W_2) rather than (W_1). The mapping function achieved by the second slab would therefore be distinct from that of the first slab.

The translational symmetries in the weights called for in the system of Figure 11 could be manufactured in and fixed, or the ADALINE elements could arrive at such symmetries as a result of a training process to be described below. If one knew when designing a pattern recognition system for a specific application that translational invariance would be a required property, it would make sense to manufacture the appropriate symmetry into a fixed weight system, leaving only the final output layer of ADALINES of Figure 12 to be plastic and trainable. Thus there would be a fixed neural net used as a pre-processor to achieve insensitivity to left-right and up-down translation. Such a pre-processor would definitely work, would provide very high speed response without requiring scanning and searching for the pattern location and alignment, would be an excellent application of neural nets, and would be a useful practical product.

Use of a Fixed Translational Invariant Neural Net

The system of Figure 12 was computer simulated with all the weights in the slabs fixed in accord with the symmetry patterns of (14) and (15). The slab outputs were invariant to up-down, left-right translation of the input pattern across the retina. However, the slab outputs comprised a binary pattern that was a scrambled version of the input pattern. The nature of the scrambling was determined by random choice of the weights (W_1),(W_2) , $\cdots$, for the first, second, $\cdots$, *kth* slab respectively.

Since the slab outputs were scrambled, a goal was set for the adaptive layer of ADALINES to serve as a descrambler. As such the ADALINE outputs were arrayed in a square pattern, just like the retinal array itself. For each retinal input pattern applied to the system, the desired responses for the ADALINES were chosen to replicate the input pattern. Of course, the system could not be trained to perfectly replicate all possible input patterns, being limited by the finite learning capacity of the trainable output layer. Each ADALINE has an average capacity equal to 2*X* the number of weights [13,14]. Since all the ADALINES needed to respond correctly at the same time to give proper replication of the input pattern, the capacity of the entire layer was somewhat less than the capacity of the individual ADALINE. The capacity was increased by increasing the number of slabs and thus increasing the number of weights per ADALINE. The capacity was also increased by using two adaptive layers, as shown in Figure 15. Using this system, a wide variety of input patterns were able to be trained in to replicate themselves at the output, regardless of up-down, left-right translation on the retina.

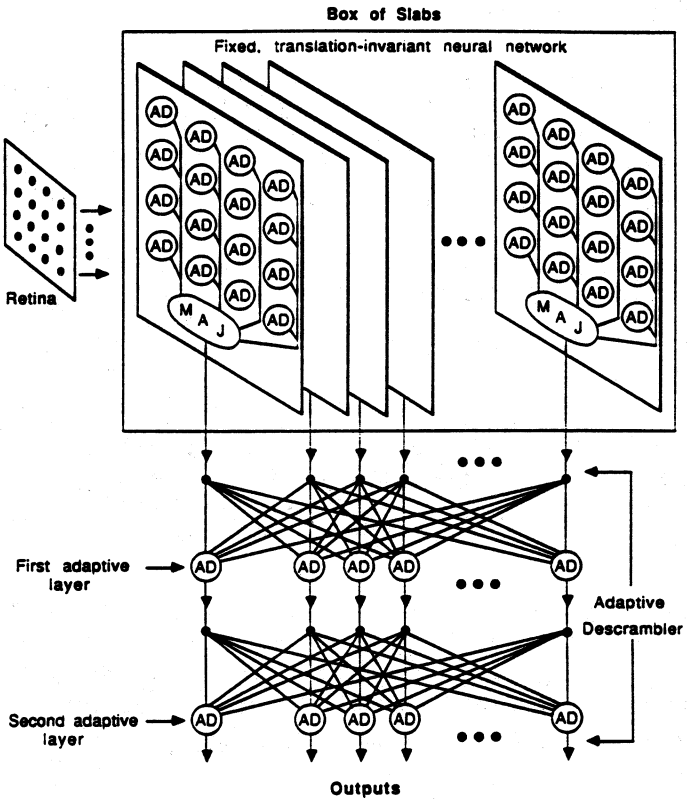

Figure 15. A translational invariant net and an adaptive 2-layer descrambler net.

Invariance to Rotation

The system represented in Figure 15 is designed to preprocess retinal patterns with a translational invariant fixed neural net followed by a two-layer adaptive descrambler net. This system can be expanded to incorporate rotational invariance in addition to translational invariance.

Suppose that all input patterns can be presented in "normal" vertical orientation, approximately centered within the field of view of the retina. Suppose further that all input patterns can be presented when

rotated by 90° from normal, and 180°, and 270° from normal. Thus each pattern can be presented in all four rotations and in addition, in all possible left-right, up-down translations. The number of combinations would typically be large. The problem is to design a neural net preprocessor that is invariant to translation and to rotation by 90°.

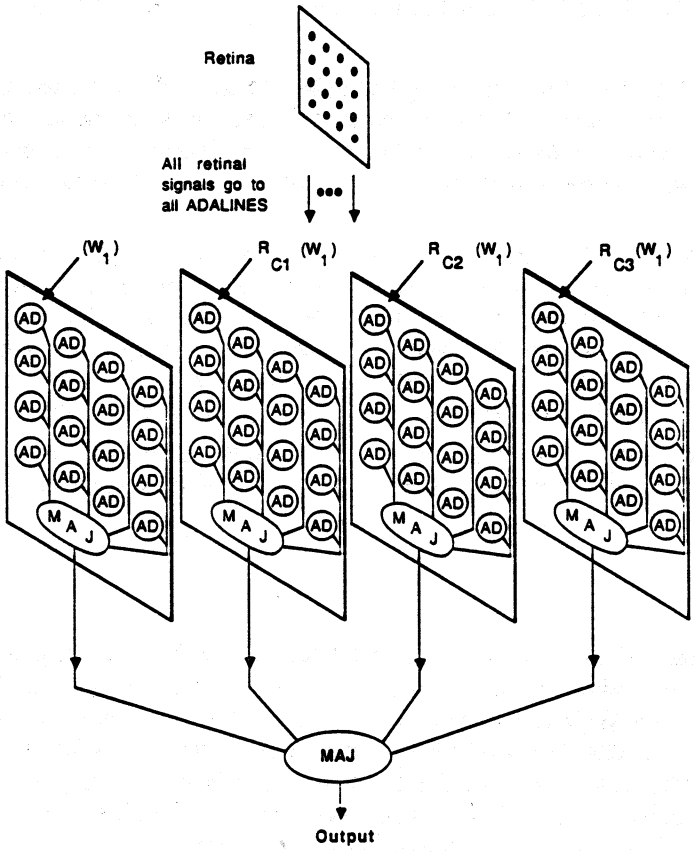

Figure 16. A network for translational and rotational invariance.

Refer to Figure 11, which shows a single slab of ADALINE elements. This slab produces a majority output which is insensitive to translation of the input pattern on the retina. Refer next to Figure 16, which shows four such slabs whose majority outputs feed into a single majority element. In the first slab, the matrix of weights of the ADALINE in the upper left-hand corner is designated by (W_1) . The matrices of weights of all ADALINES in the first slab are shown in (15). In the second slab of Figure 16, the weight matrix of the upper left-hand corner ADALINE corresponds to the weight matrix in the first slab, except rotated clockwise 90°. This can be designated by $R_{C1}(W_1)$. The corresponding upper left-hand corner ADALINE weight matrix of the third slab can be designated by $R_{C2}(W_1)$, and of the fourth slab $R_{C3}(W_1)$. Thus, the weight matrices of the upper left hand corner ADALINES begin with (W_1) in the first slab, and are rotated clockwise by 90° in the second slab, by 180° in the third slab, and by 270° in the fourth slab. The weight matrices of all of these slabs are translated right and down, starting with the upper left-hand corner ADALINES. For example, the array of weight matrices for the second slab is represented by (16).

$$\begin{array}{|cccc|}\hline R_{C1}(W_1) & T_{R1}R_{C1}(W_1) & T_{R2}R_{C1}(W_1) & T_{R3}R_{C1}(W_1) \\ T_{D1}R_{C1}(W_1) & T_{R1}T_{D1}R_{C1}(W_1) & T_{R2}T_{D1}R_{C1}(W_1) & T_{R3}T_{D1}R_{C1}(W_1) \\ T_{D2}R_{C1}(W_1) & T_{R1}T_{D2}R_{C1}(W_1) & T_{R2}T_{D2}R_{C1}(W_1) & T_{R3}T_{D2}R_{C1}(W_1) \\ T_{D3}R_{C1}(W_1) & T_{R1}T_{D3}R_{C1}(W_1) & T_{R2}T_{D3}R_{C1}(W_1) & T_{R3}T_{D3}R_{C1}(W_1) \\ \hline \end{array} \quad (16)$$

Presenting a pattern to the retina causes an immediate response from the output majority element in Figure 16. It is clear that this response will be unchanged by translation of the pattern on the retina. Rotation of the pattern by 90° causes an interchange of the roles of the slabs in making their responses, but since they are all weighted equally by the output majority element, the output response is unchanged by 90° rotation and translation.

If one wished to have insensitivity to 45° rotation, the system of Figure 16 would need eight slabs, and the upper left-hand corner ADALINES would have weight matrices rotated by 45° relative to corresponding neighbors. In each slab, the weight matrices would be left-right, up-down translated. Rotation insensitivity can be achieved for much smaller angular increments by increasing the number of slabs. Rotation of the weight matrices by small angular increments can only be done with large retinas having high resolution. All of this involves neural networks having large numbers of weights.

A complete neural network providing invariance to rotation and translation would involve the structures of both Figures 15 and 16. Each slab of Figure 15 would need to be replaced by the multiple slab and majority element system of Figure 16.

Invariance to Scale

The same principles can be used to design invariance nets to be insensitive to scale or pattern size. Establishing a "point of expansion" on the retina so that input patterns can be expanded or contracted with respect to this point, two ADALINES can be trained to give similar responses to patterns of two different sizes if the weight matrix of one were expanded (or contracted) about the point of expansion like the patterns themselves. The amplitude of the weights must be scaled in inverse proportion to the square of the linear dimension of the retinal pattern. Adding many more slabs, the invariance net can be built around this idea to be insensitive to pattern size as well as translation and rotation.

Invariance to Perspective

Insensitivity to change in perspective is a difficult attribute to attain for three dimensional objects. The following is a simpler problem. Consider the flat object of Figure 17 being photographed from various vantage points indicated by the arrows. Each two dimensional photo will be of a certain perspective relative to the original object. The photos could be spatially quantized and provided as retinal inputs to a recognition system. The recognition problem requires first an insensitivity to perspective. The requirement is similar to insensitivity to scale, except that the vertical scale is fixed and the horizontal scale is variable. The method of approach is similar to that for insensitivity to scale, rotation, and translation.

Simulation Experiments

The MADALINE adaptation rule, with many variations on the theme, has been simulated on a conventional computer and used to adapt multilayered networks. Various examples such as the exclusive OR problem and a pattern edge detection problem have been worked with speed and dispatch. Currently the algorithm is being used to adapt the translational invariance network of Figure 12, with some success. The same algorithm is being used to train the two layer descrambler net of Figure 15. The invariance net is

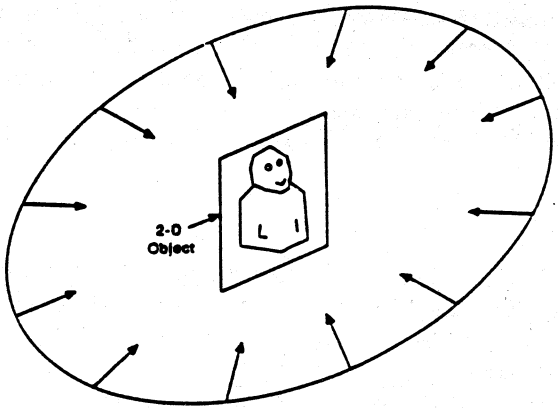

Figure 17. Two-dimensional perspectives of a two-dimensional object.

designed and fixed for translational insensitivity. It contains 25 slabs, all connected to a 5 *X* 5 retina. The adaptive net has 25 ADALINES in each layer. A learning curve for a typical experiment giving error rate versus number of training cycles for the descrambler is shown in Figure 18. The new multilayer MADALINE training rule has been tested many times and has worked without failure. It appears to be a robust, rapidly converging procedure.

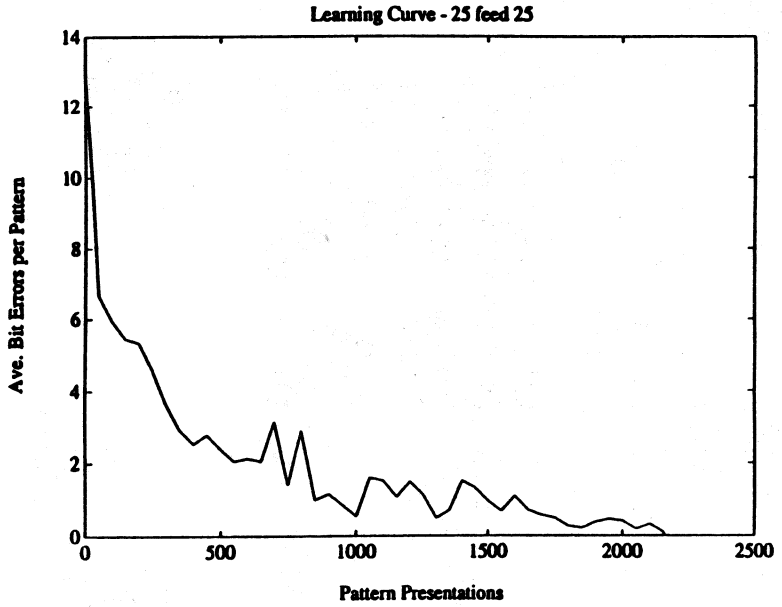

Figure 18. Learning curve for 2-layer 25 by 25 adaptive descrambler.

Speech Recognition

The idea of an invariance net followed by a trainable classifier can, it is believed, be used effectively for speech recognition. Speech could be spectrally analyzed and sampled over time in each of a set of

bandpass ranges or it could be encoded by adaptive linear prediction and the LPC coefficients could be sampled over time, or some other form of preprocessing could be practiced to obtain input patterns for a speech classifier. Speech recognition requires insensitivity to certain aspects of speech and at the same time, sensitivity to other aspects. Trainable sensitivity and insensitivity is needed. The system structure of Figure 15 will have the proper attributes for this application. This will soon be tested and reported.

Summary

A general concept for pattern recognition is described involving the use of an invariance net followed by a trainable classifier. The key ideas are illustrated in Figure 19. The invariance net can be trained or designed to produce a set of outputs that are insensitive to translation, rotation, scale change, perspective, etc., of the retinal pattern. These outputs are scrambled however. The adaptive layers can be trained to descramble the invariance net outputs and to reproduce the original patterns in "standard" position, orientation, scale, etc. The reader is referred once again to Figure 19.

Multilayer adaptation algorithms are essential to making such a scheme work. A new MADALINE adaptation rule has been devised for such purpose, and preliminary experimental results indicate that it works and is effective.

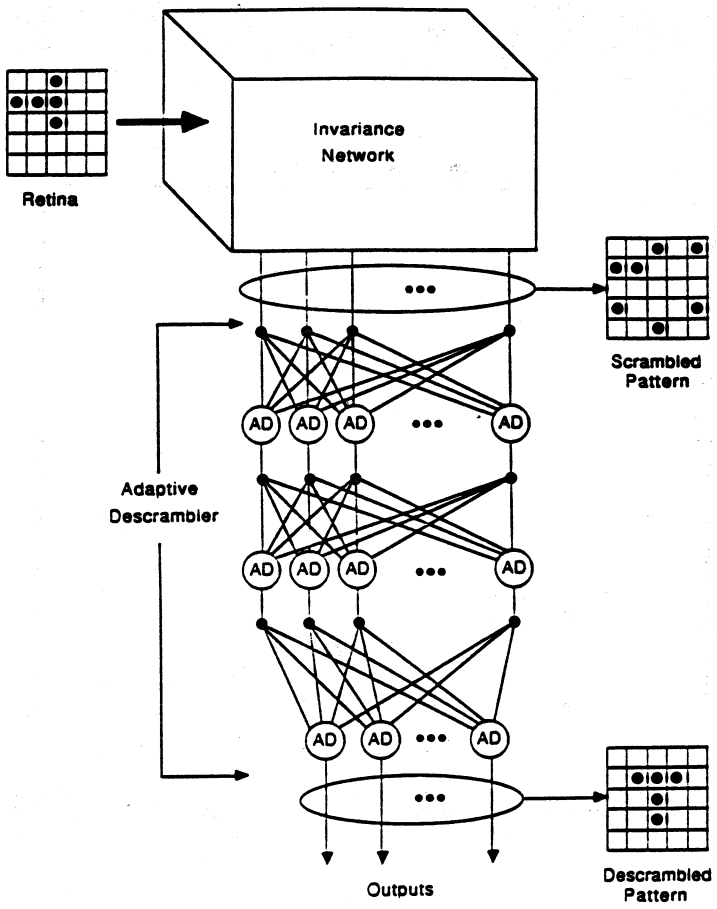

Figure 19. A MADALINE system for pattern recognition.

REFERENCES

[1] B. Widrow and M. E. Hoff, Jr., "Adaptive switching circuits," in *IRE WESCON Conv. Rec.*, pt. 4, pp. 96-104, 1960.

[2] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing*, Cambridge, Massachusetts: MIT Press, 1986, Volumes I and II.

[3] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1985.

[4] P. M. Lewis II and C. L. Coates, *Threshold Logic*, New York: John Wiley and Sons, 1967.

[5] D. F. Specht, "Vectorcardiographic diagnosis using the polynomial discriminant method of pattern recognition," IEEE Trans. on Bio-Medical Engineering, Vol. BME-14, No. 3, April, 1967, pp. 90-95.

[6] D. F. Specht, "Generation of polynomial discriminant functions for pattern recognition," IEEE Trans. on Electronic Computers, Vol. EC-16, No. 3, June, 1967, pp. 308-319.

[7] A. G. Ivakhnenko, "Polynomial theory of complex systems," IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-1, No. 4, pp. 364-378, Oct. 1971.

[8] M. E. Hoff, Jr., "Learning phenomena in networks of adaptive switching circuits," Stanford Electronics Labs, Stanford University, Stanford, CA, Rep. 1554-1, July 1962 (PhD Dissertation).

[9] W. C. Ridgway III, "An adaptive logic system with generalizing properties," Stanford Electronics Labs, Stanford University, Stanford, CA, Rep. 1556-1, April 1962 (PhD Dissertation).

[10] D. B. Parker, "Learning-Logic," Center for Computational Research in Economics and Management Science, Mass. Inst. of Tech., Rep. TR-47, April 1985.

[11] B. Widrow, "Generalization & Information Storage in Networks of Adaline 'Neurons'" *Self Organizing Systems 1962*, Edited by M. C. Yovitz, G. T. Jacobi, & G. D. Goldstein, Washington, DC: Spartan Books, pp. 435-461, 1962.

[12] N. Nilsson, *Learning Machines*. New York: McGraw-Hill, 1965.

[13] R. J. Brown, "Adaptive multiple-output threshold systems and their storage capacities," Stanford Electronics Labs., Stanford University, Stanford, CA, Rep. 6671-1, June 1964 (PhD Dissertation).

[14] T. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," IEEE Trans. on Elect. Comp., Vol. EC-14, pp. 326-334, June 1965.