Layered Neural Nets for Pattern Recognition

BERNARD WIDROW, FELLOW, IEEE, RODNEY G. WINTER, AND ROBERT A. BAXTER

***Abstract*—Adaptive threshold logic elements called ADALINES can be used in trainable pattern recognition systems. Adaptation by the LMS (least mean squares) algorithm is discussed. Threshold logic elements only realize linearly separable functions. To implement more elaborate classification functions, multilayered ADALINE networks can be used.**

A pattern recognition concept involving first an "invariance net" and second a "trainable classifier" is proposed. 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, however. When these outputs are fed to a 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 insensitivity to certain aspects of speech signals and at the same time sensitivity to other aspects of speech signals will be required.

The entire recognition system is a layered network of ADALINE neurons. The ability to adapt a multilayered neural net is fundamental. A new adaptation rule is proposed for layered nets which is an extension of the MADALINE rule of the 1960's. The new rule, MRII, is a useful alternative to the back-propagation algorithm.

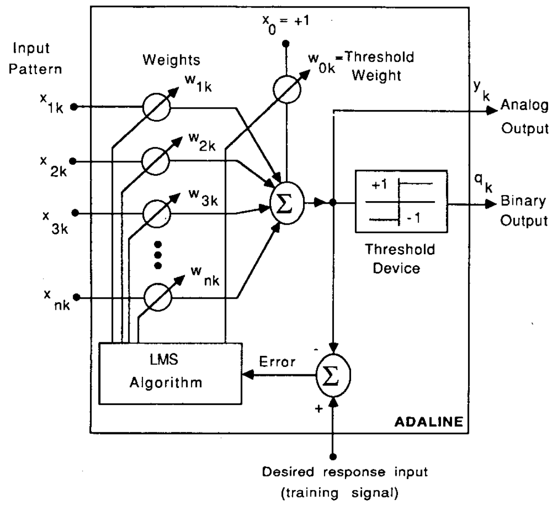

Fig. 1. An adaptive linear neuron (ADALINE).

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 Fig. 1. This is an adaptive threshold logic element. In a digital implementation, this element has at time k an input signal vector or input pattern vector $X_k = [x_0 x_{1_k} x_{2_k} \cdots x_{n_k}]^T$ whose components are weighted by a set of coefficients. The weight vector is $W_k = [w_{0_k} w_{1_k} w_{2_k} \cdots w_{n_k}]^T$. The element produces an analog output, the inner product $y_k = X_k^T W_k = W_k^T X_k$. The bias weight w_{0_k} is connected to a constant input, $x_0 = +1$, and it controls the threshold level. The element also produces a binary output, q_k. Decisions are made by a 2-level quantizer. The binary ± 1 output is $q_k = \text{SGN}(y_k)$.

The "desired response" is a special input signal used to train the neuron. During the training process, input patterns and corresponding desired responses are fed to this element. An adaptation algorithm automatically adjusts the weights so that the output 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]). This algorithm minimizes the sum of the squares of the errors over the training set, where the error is defined as the difference between the desired response and the analog output.

The desired response and the components of X_k could be analog or binary. In neural networks, however, inputs and outputs are often 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.

Once the weights are adjusted and the neuron is trained, its responses can be tested by applying various input patterns. If the neuron responds correctly, with high probability, to input patterns that were not included in the training set, it is said that generalization has taken place. The capability of generalization is a highly significant attribute of neural nets.

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 the field of telecommunications, adaptive equalizers [4] for high-speed digital modems, and adaptive echo cancellers [5] for long-distance telephone circuits and satellite channels. The LMS algorithm is widely used in adaptive signal processing and telecommunications.

Manuscript received February 22, 1988.

B. Widrow and R. A. Baxter are with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305.

R. G. Winter is with the U.S. Air Force and the Department of Electrical Engineering, Stanford University, Stanford, CA 94305.

IEEE Log Number 8821364.

Linear Separability

With n binary inputs and one binary output, a single neuron of the type shown in Fig. 1 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 [6]. These are the set of logic functions that can be obtained with all possible settings of the weight values.

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

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

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

This linear relation is graphed in Fig. 3. It comprises a separating line which has slope and intercept of

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

The three weights determine slope, intercept, and the side of the separating line that corresponds to a positive output. The opposite side of the separating line corresponds to a negative output.

As sketched in Fig. 3, the binary inputs are classified as follows:

$$\begin{aligned} (+1, +1) &\rightarrow +1 \\ (+1, -1) &\rightarrow +1 \\ (-1, -1) &\rightarrow +1 \\ (-1, +1) &\rightarrow -1. \end{aligned} \tag{4}$$

This is an example of a linearly separable function. An example of a nonlinearly separable function with two inputs is the following.

$$\begin{aligned} (+1, +1) &\rightarrow +1 \\ (+1, -1) &\rightarrow -1 \\ (-1, -1) &\rightarrow +1 \\ (-1, +1) &\rightarrow -1. \end{aligned} \tag{5}$$

No single line exists that can achieve this separation of the input patterns.

With two inputs, almost all possible logic functions can be realized by a single neuron. With many inputs, however, only a small fraction of all possible logic functions are linearly separable. Since the single neuron can only realize linearly separable functions and generally cannot realize most functions, combinations of neurons or networks of neurons can be used to realize nonlinearly separable functions.

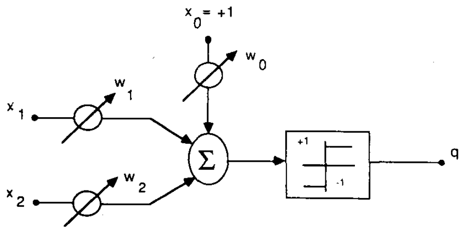

Fig. 2. A two-input neuron.

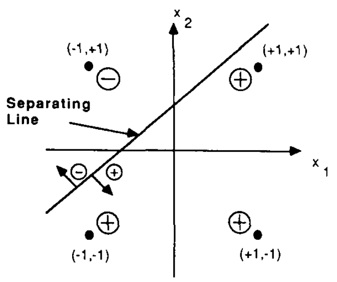

Fig. 3. Separating line in pattern space.

Before discussing networks of neurons, a simple means for achieving nonlinear separability with a single neuron with nonlinearities in its input signal path is shown next.

Nonlinear Separability—Nonlinear Input Functions

Nonlinear functions of the inputs applied to the single neuron can yield nonlinear decision boundaries. Consider the system illustrated in Fig. 4. The threshold condition is

$$\begin{aligned} y = w_0 + x_1 w_1 + x_1^2 w_{11} + x_1 x_2 w_{12} \\ + x_2^2 w_{22} + x_2 w_2 = 0. \end{aligned} \tag{6}$$

With proper choice of the weights, the separating boundary in pattern space can be established as shown, for example, in Fig. 5. The nonlinearly separable function (5) can be realized by this configuration. 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 more inputs than two and for higher degree polynomial functions and cross product functions of the inputs. One of the first works in this area was done by Specht [7], [8] at Stanford in the 1960's, and later by Ivankhnenko [9] in the 1970's.

Nonlinear Separability—MADALINE Networks

Another approach to the implementation of nonlinearly separable logic functions was initiated at Stanford by Hoff

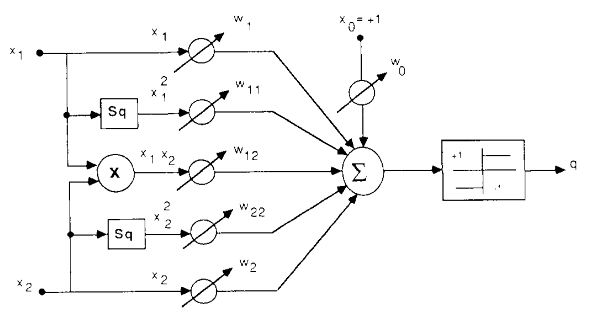

Fig. 4. A neuron with inputs mapped through nonlinearities.

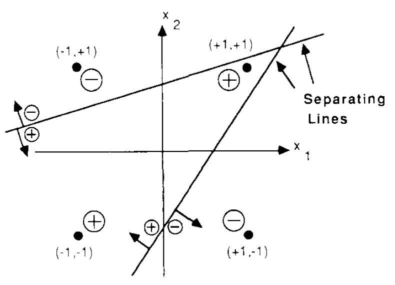

Fig. 7. Separating boundaries for MADALINE of Fig. 6.

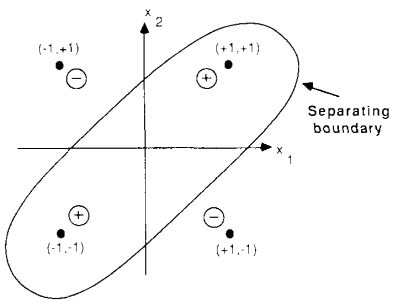

Fig. 5. An elliptical separating boundary for nonlinearly separable function realization.

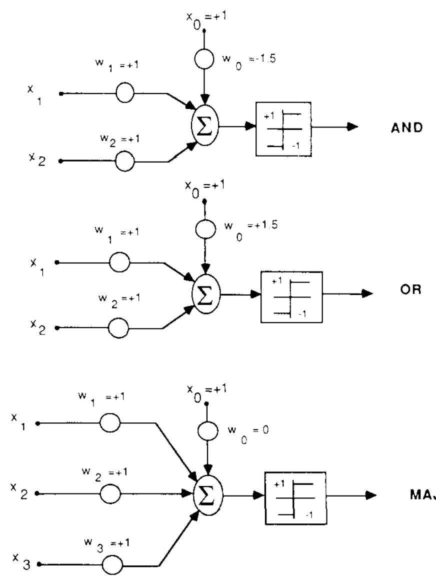

Fig. 8. Neuronal implementation of AND, OR, and MAJ logic functions.

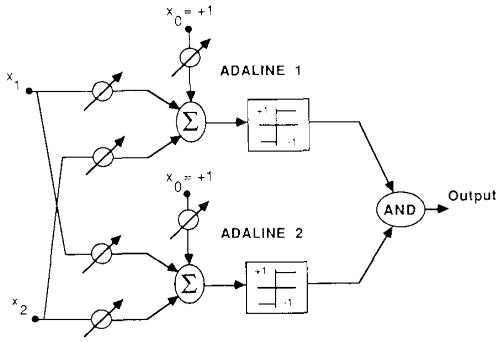

Fig. 6. A two-neuron form of MADALINE.

[10] and Ridgway [11] in the early 1960's. Retinal inputs were connected to adative neurons in a single layer. Their outputs in turn 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 Fig. 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.

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

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 AND, OR, and MAJority vote-taker. These three functions are in themselves threshold logic functions, as illustrated in Fig. 8. The weights given will implement these functions, but the weight choices 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 best-known multilayer work is by Rumelhart *et al.* [2]. A 3-layer adaptive network is illustrated in Fig. 9.

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 the desired responses, adaptation of the output layer can be a straightforward exercise of the LMS algorithm. The fundamental difficulty associated with adaptation of a layered network lies in obtaining desired responses for

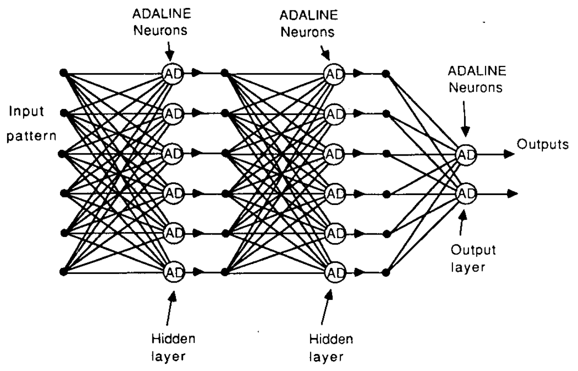

Fig. 9. A three-layer adaptive neural network.

the neurons in the layers other than the output layer. The back-propagation algorithm (reported earliest by Werbos [12] then discovered by Parker [13], and again discovered by Rumelhart *et al.* [2]) 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 Fig. 9).

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 may be needed for the adaptation of multilayered networks to produce required generalizations. Without generalization, neural nets will be of little engineering significance. Merely 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 Fig. 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. However, it 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 [14], [15] and is called MADALINE rule II or MRII. 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 for the Single Neuron

The LMS algorithm applied to the adaptation of the weights of a single neuron embodies a minimal disturbance principle. A self-normalizing form of this algorithm can be written as

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

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 binary ± 1 input vectors, $|X_k|^2$ equals the number of weights.

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

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

The error is changed (reduced) by changing the weights.

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

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

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

Combining (9) and (10), we obtain

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

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 practical range for α is

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

Fig. 10 gives a geometric picture of how the LMS rule works. W_{k+1} equals W_k plus ΔW_k in accord with (10), and ΔW_k is parallel with the input pattern vector X_k also in accord with (10). By (9), the change in the error will be equal to the negative dot product of X_k with ΔW_k. Since the LMS algorithm selects ΔW_k to be collinear with X_k,

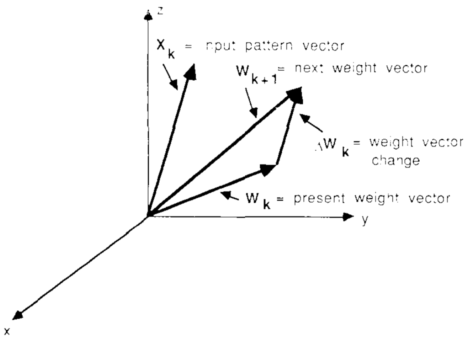
z
X_k = input pattern vector
W_{k+1} = next weight vector
ΔW_k = weight vector change
W_k = present weight vector
y
x

Fig. 10. Weight correction by the LMS rule.

the needed error correction is achieved with the smallest magnitude of weight vector change. When adapting to respond properly to a new input pattern, the responses to previous training patterns are therefore minimally disturbed, on the average. The algorithm also minimizes mean square error [3], for which it is best known.

Adaptation of Layered Neural Nets by the MRII Rule

The minimal disturbance principle can be applied to the adaptation of the layered neural network of Fig. 9 in the following way. Present a retinal pattern vector and its associated desired response vector. The training objective is to reduce the number of output errors (the Hamming distance between the net's actual output and desired response vectors) 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. A check is made to see if this reduces the number of output errors. If so, the trial change is accepted. If not, the weights are restored to their previous values and the first-layer neuron whose analog response is next closest to zero is trial adapted, reversing its response. If this reduces the number of output errors, the change is accepted. If not, the weights are restored and one goes on to adaptively switch the neuron with an analog response next closest to zero, and so on, disturbing the neurons as little as possible. After adapting all of the first-layer neurons whose output reversals reduced the number of network output errors, neurons are then chosen in pair combinations and trial adaptations are made 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 is 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 output-layer desired responses are the desired responses for the network. After adapting the output layer, the responses will be correct. The next input pattern vector and its associated desired response vector 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 MRII algorithm has been tested extensively, and appears to converge and behave robustly. 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. Simulation 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. It should not be necessary to train 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 exists having these properties. The next step is to obtain training algorithms to achieve the desired objectives.

Invariance to Up–Down, Left–Right Pattern Translation

Fig. 11 shows a planar 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 Fig. 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 or fewer 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

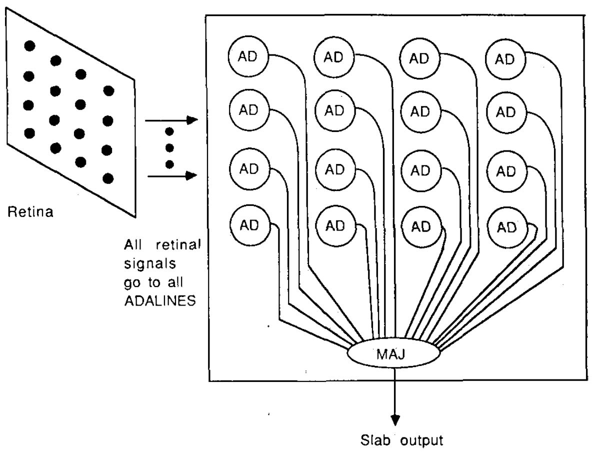

Fig. 11. One slab of a left-right, up-down translation invariant network.

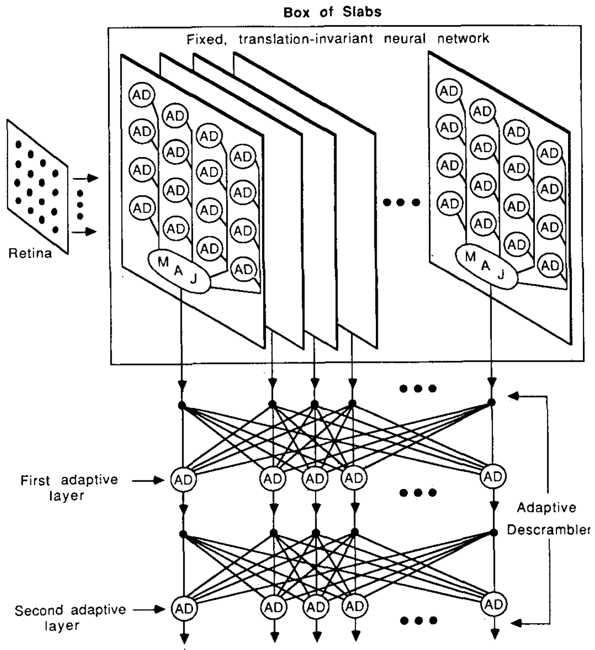

Fig. 12. A translation-invariant neural network and an adaptive two-layer descrambler network.

mapped image in Fig. 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 Figs. 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, as in Fig. 8.

How the weights are structured in the system of Fig. 11 to cause the output to be insensitive to left-right and up-down translation needs some further explanation. Our purpose here is to show that sets of weights exist that will achieve this function. How to adaptively obtain such weights is a separate issue. Consider the diagram of Fig. 13. This system is insensitive to up-down translation. Let the weights of each ADALINE in Fig. 13 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. 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 $T_{D3}(W_1)$. As the input pattern is moved up or down on the retina, the roles of the various ADALINES interchange. 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.

The network of ADALINES of Fig. 13 is replicated on the slab of Fig. 11. Let the first column of weights in Fig. 11 be chosen like the weights of the column in Fig. 13. Let the second column of weights be chosen as

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

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 Fig. 11 is given by

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

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 out-

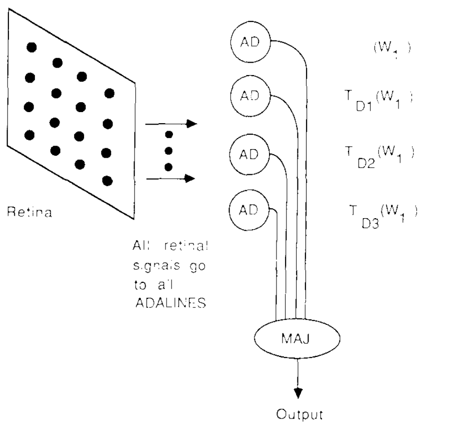

Fig. 13. Training insensitivity to up–down translation.

put of this system will be insensitive to both left–right and up–down translation.

The set of weights (W_1) could be randomly chosen. Once chosen, they can be translated according to (15) to fill out the array of weights for the system of Fig. 11. This array of weights can be incorporated as the weights for the first slab of ADALINES shown in Fig. 12. The weights for the second slab would require the same translational symmetries, but be based on a different 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 Fig. 11 could be manufactured in and fixed, or the ADALINE elements could arrive at such symmetries as a result of a training process. 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 layers of ADALINES of Fig. 12 to be plastic and trainable. Such a preprocessor would definitely work, would provide a 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.

Invariance to Rotation

The system represented in Fig. 12 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°.

Refer to Fig. 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 Fig. 14, 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 Fig. 14, 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).

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

Presenting a pattern to the retina causes an immediate response from the output majority element in Fig. 14. 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 Fig. 14 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

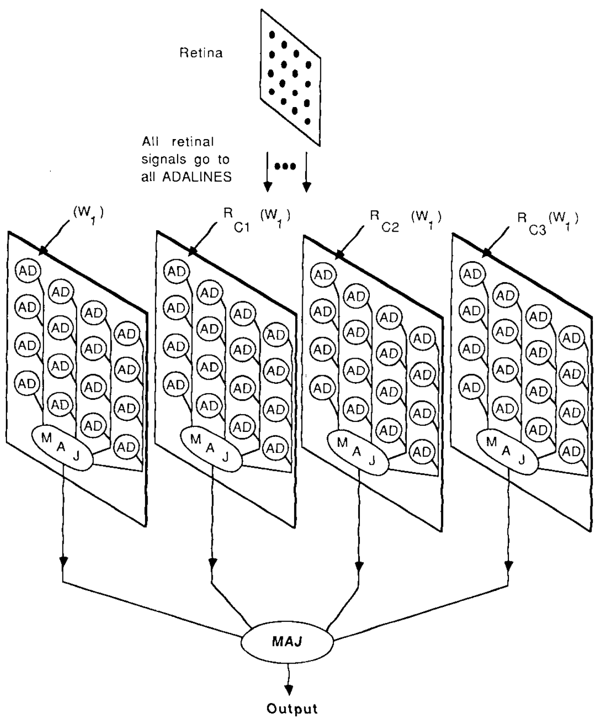

Fig. 14. A network for translational and rotational invariance.

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 Figs. 12 and 14. Each slab of Fig. 12 would need to be replaced by the multiple slab and majority element system of Fig. 14.

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

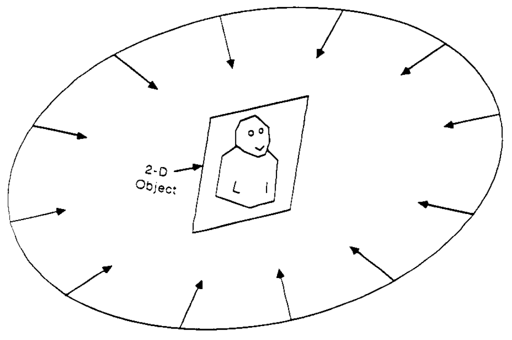

Fig. 15. Two-dimensional perspectives of a two-dimensional object.

Fig. 15 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.

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 Fig. 12 will have the proper attributes for this application. This will soon be tested and reported.

Simulation Experiments

The system of Fig. 12 was computer simulated. The training set consisted of 36 patterns each arranged on a 5×5 pixel retina in "standard" position. Twenty-five slabs, each with twenty-five ADALINES, having weights fixed in accord with the symmetry patterns of (15), were used in the translation-invariant preprocessor. The preprocessor output represented a scrambled version of the input pattern. The nature of this scrambling was determined by the choice of the upper-left ADALINE weight matrices $(W_1), \cdots, (W_{25})$. These weights were chosen randomly, the only requirement being that the input pattern to preprocessor output map be one-to-one. (This choice of weights produced a very noise intolerant mapping. Methods of training-in these weights using MRII to customize them to the training set are being investigated.)

MRII was used to train the descrambler. The descram-

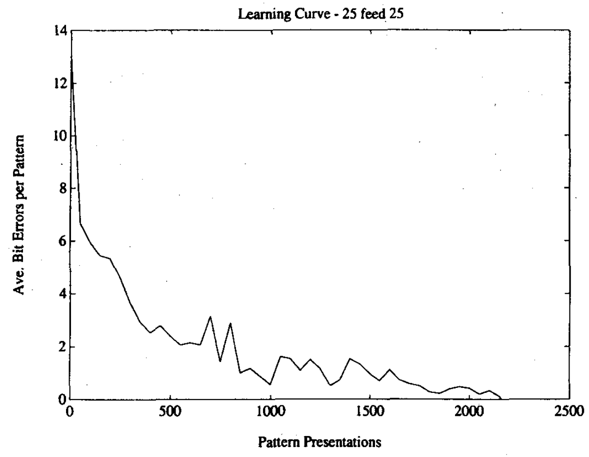

Fig. 16. Learning curve for 2-layer 25 by 25 adaptive descrambler.

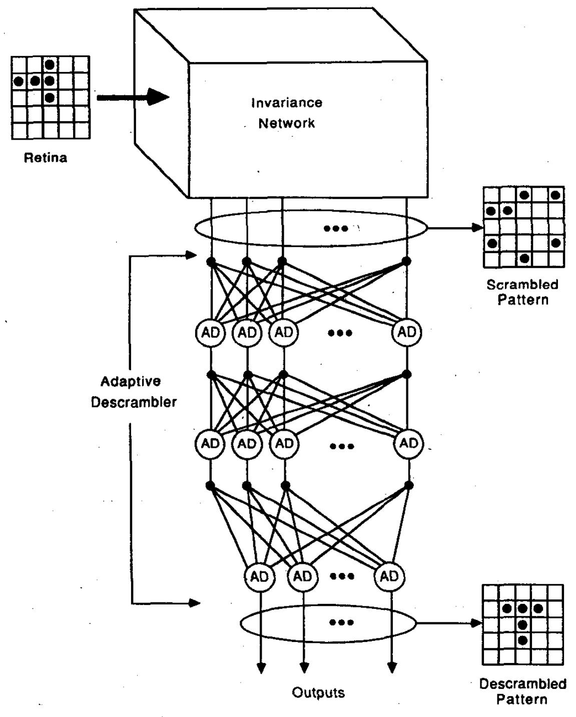

Fig. 17. A MADALINE system for pattern recognition.

bler was a 2-layer system with 25 ADALINES in each layer. The initial weights of the descrambler were chosen randomly. (All random weights in the system were chosen independently, identically distributed uniformly on the interval $(-1, +1)$.) Patterns were presented in random order, each pattern being equally likely of being the next presented. The desired response used was the training pattern in standard position. The system as a whole would then recognize any trained-in pattern in any translated position on the input retina and reproduce it in standard position at the output. A typical learning curve for the descrambler is shown in Fig. 16. The graph shows the number of incorrect pixels at the output, averaged over the training set, every 50 pattern presentations.

Much work on MRII remains to be done, including detailed studies of its convergence properties and its ability to produce generalizations. Preliminary results are very encouraging. Applying the algorithm to problems will lead to insights that will hopefully allow a mathematical analysis of the algorithm.

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 Fig. 17. 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 Fig. 17.

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

References

[1] B. Widrow and M. E. Hoff, Jr., "Adaptive switching circuits," in *IRE WESCON Conv. Rec.*, pt. 4, 1960, pp. 96–104.
[2] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing*, Vol. I and II. Cambridge, MA: M.I.T. Press, 1986.
[3] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.
[4] R. W. Lucky, "Automatic equalization for digital communication," *Bell Syst. Tech. J.*, vol. 44, pp. 547–588, Apr. 1965.
[5] M. M. Sondhi, "An adaptive echo canceller," *Bell Syst. Tech. J.*, vol. 46, pp. 497–511, Mar. 1967.
[6] P. M. Lewis, II, and C. L. Coates, *Threshold Logic*. New York: Wiley, 1967.
[7] D. F. Specht, "Vectorcardiographic diagnosis using the polynomial discriminant method of pattern recognition," *IEEE Trans. Biomed. Eng.*, vol. BME-14, pp. 90–95, Apr. 1967.
[8] ——, "Generation of polynomial discriminant functions for pattern recognition," *IEEE Trans. Electron. Comput.*, vol. EC-16, pp. 308–319, June 1967.
[9] A. G. Ivakhnenko, "Polynomial theory of complex systems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-1, pp. 364–378, Oct. 1971.
[10] M. E. Hoff, Jr., "Learning phenomena in networks of adaptive switching circuits," Ph.D. dissertation, Stanford Electron. Labs. Rep. 1554-1, Stanford Univ., Stanford, CA, July 1962.
[11] W. C. Ridgway, III, "An adaptive logic system with generalizing properties," Ph.D. dissertation, Stanford Electron. Labs. Rep. 1556-1, Stanford Univ., Stanford, CA, Apr. 1962.
[12] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Harvard Univ., Cambridge, MA, Aug. 1974.
[13] D. B. Parker, "Learning logic," Tech. Rep. TR-47, Center for Comput. Res. Econ. and Manage. Sci., Mass. Inst. Technol., Cambridge, Apr. 1985.
[14] B. Widrow, "Generalization and information storage in networks of adaline 'neurons,'" in *Self-Organizing Systems 1962*, M. C. Yovitz, G. T. Jacobi, and G. D. Goldstein, Eds. Washington, DC: Spartan Books, 1962, pp. 435–461.
[15] N. Nilsson, *Learning Machines*. New York: McGraw-Hill, 1965.

Bernard Widrow (M'58-SM'75-F'76) was born in Norwich, CT, on December 24, 1929. He received the S.B., S.M., and Sc.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 1951, 1953, and 1956, respectively.

He is a Professor of Electrical Engineering at Stanford University, Stanford, CA. From 1951 to 1956, he was a Staff Member of the M.I.T. Lincoln Laboratory and a Research Assistant in the Department of Electrical Engineering of M.I.T. He joined the M.I.T. Faculty in 1956, and taught classes in radar, theory of sampled-data systems, and control theory. In 1959 he joined the Stanford Faculty. He is presently engaged in research and teaching in systems theory, pattern recognition, adaptive filtering, and adaptive control systems. He is an Associate Editor of the journals *Adaptive Control and Signal Processing*, *Neural Networks*, *Information Sciences*, and *Pattern Recognition*. He is co-author with S. D. Stearns of *Adaptive Signal Processing* (Englewood Cliffs, NJ: Prentice-Hall).

Dr. Widrow received the IEEE Alexander Graham Bell Medal in 1986 for exceptional contributions to the advancement of telecommunications. He is a member of the American Association of University Professors, the Pattern Recognition Society, Sigma Xi, and Tau Beta Pi. He is a Fellow of the American Association for the Advancement of Science.

Rodney G. Winter received the B.S.E.E. and M.S.E.E. degrees from Purdue University in 1977.

He is a graduate student attending Stanford University through the Civilian Institutes program of the Air Force Institute of Technology. His research interests include signal processing, pattern recognition, and adaptive systems. His doctoral thesis topic is the application of neural networks to optimal nonlinear filtering. His prior duties were those of a Pilot in the U.S. Air Force, most recently in the F-106 interceptor.

Capt. Winter is a member of Eta Kappa Nu.

Robert A. Baxter was born in Charlotte, NC, on August 9, 1954. He received the B.S.E.E. degree from North Carolina State University, Rayleigh, and the M.Sc. degree from Ohio State University, Columbus. He is currently pursuing the doctorate at Boston University, Boston, MA.

In 1976 he became a Research Associate at the Ohio State University's ElectroScience Laboratory, and in 1979 became a Research Engineer at Lockheed Missiles and Space Co. In 1984 he became an independent consultant to several aerospace companies and started a software development venture in 1985. In January of 1987, he joined the Staff of Stanford University's Information Systems Laboratory. His interests span the fields of electromagnetic field theory, 3-D vision, self-organizing neural networks, and learning phenomena in adaptive systems.