12

PATTERN-RECOGNIZING CONTROL SYSTEMS

BERNARD WIDROW AND FRED W. SMITH

Stanford University, Stanford, Calif.

INTRODUCTION

For the past several years the properties and applications of adaptive threshold-logic elements and means of physical realization of these elements have been under study at Stanford University. It is the purpose of this paper to present an up-to-date summary of this work, and in particular, to show how adaptive logic networks have been used in an automatic control system.

ADALINE, AN ADAPTIVE LOGIC ELEMENT

A basic building block of the systems to be considered is an adaptive threshold element, sometimes called an adaptive "neuron." For the past several years, we at Stanford University have called this element Adaline (adaptive linear neuron). A functional diagram of this element is shown in Fig. 1. It includes an adjustable threshold element and the adaptation machinery which automatically adjusts the variable weights. It has been demonstrated experimentally and theoretically that this element can be trained to react specifically to a wide variety of binary input signals and that it can be trained to generalize in certain ways, i.e., to react as desired with high reliability to inputs that it has not been specifically trained on.

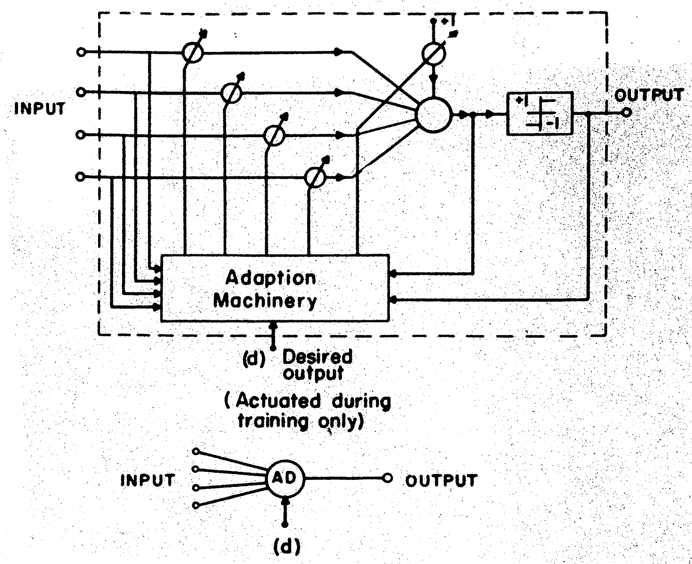
In Fig. 1 the binary input signals on the input lines have values of $+1$ or -1 rather than the usual values of 1 or 0. Within the neuron shown, a linear combination of the input signals is formed. The weights are the gains $w_1, w_2, \ldots$, which could have both positive and negative values. The output signal is $+1$ if this weighted sum is greater than a certain threshold, and -1 otherwise. The threshold level is determined by the setting of w_0, whose input is permanently connected to a $+1$ sorce. Varying w_0 varies a constant added to the linear combination of input signals.

For fixed gain settings, each of the 2^n possible input combinations would cause either a $+1$ or a -1 output. Thus, all possible inputs are classified into two categories. The input-output relationship is determined by choice of the gains $w_0, \ldots, w_n$. In the adaptive neuron, these gains are set during the training procedure.

In general, there are 2^{2^n} different input-output relationships or truth functions by which the n input variables can be mapped into the single output variable. Only a subset of these, the linearly separable logic functions, can be realized by all possible choices of the gains. Although this subset is not all inclusive, it is a useful subset, and it is "searchable," i.e., the "best" function in many practical cases can be found iteratively without trying all functions within the subset. An iterative search procedure has been devised and is described below. This procedure is quite simple to implement, and can be analyzed by statistical methods that were originally developed for the analysis of adaptive sampled-data systems.[1]

An adaptive pattern classification machine has been constructed for the purpose of illustrating adaptive behavior and artificial learning. A photograph of this machine, which is an adjustable threshold element (called "KNOBBY ADALINE"), is shown in Fig. 2.

During a training phase, simple geometric patterns are fed to the machine by setting the toggle switches in the 4 × 4 input switch array. All



ADALINE (Adaptive Linear "neuron")

Figure 1

gains, including the threshold level, are to be changed by the same absolute magnitude such that the analog error (the difference between the desired meter reading and the actual meter reading) is brought to zero. This is accomplished by changing each gain in the direction which will diminish the error by $\frac{1}{17}$. The 17 gains may be changed in any sequence, and after all changes are made, the error for the present input pattern is zero. The weights associated with switches up (+1 input signals) are incremented by rotation in the same direction as the desired meter needle rotation, the weights connected to switches in the down position are incremented opposite to the desired direction of rotation of the meter needle. The next pattern and its desired output is then presented, and the error is read. The same adjustment routine is followed and the error is brought to zero. If the first pattern were reapplied at this point, the error would be small but not necessarily zero. More patterns are inserted in like manner. Convergence is indicated by small errors (before adaption), with small fluctuations about stable weights. A least-mean-square adaption procedure (LMS) requires that adaption be made even if the quantized neuron output is correct. If, for example, the desired response is +1, the neuron is adapted to bring the analog response closer to the desired response, even if the analog response is more positive than +1.

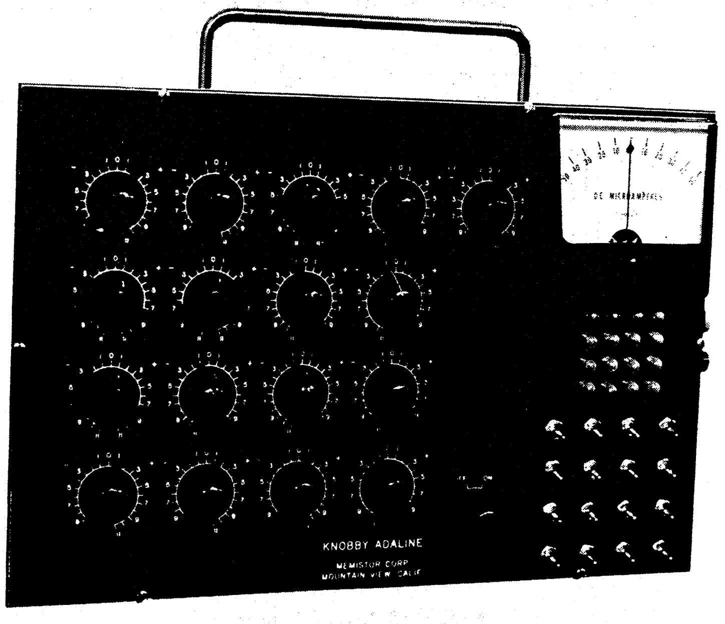


Figure 2

The iterative training routine is purely mechanical. Electronic automation of this procedure will be discussed below.

The results of a typical adaption on six noiseless patterns is given in Fig. 3. During adaption, the patterns were selected in a random sequence, and were classified into 3 categories. Each T was to be mapped to $+30$ on the meter dial, each G to 0, and each F to -30. As a measure of performance, after each adaptation, all six patterns were read in (without adaptation) and six analog errors were read. The sum of their squares denoted by Σe^2 was computed and plotted. Figure 3 shows the learning curve for the case in which all gains were initially zero.

It is shown in Ref. 2 and 3 that making full correction with each adaption using the LMS procedure is in effect a stable "performance feedback" process having an adaptive time constant equal to the number of weights. In the experiment of Fig. 3, the time constant is 17 adaptions. It is also shown that changing each weight by the same magnitude in the appropriate

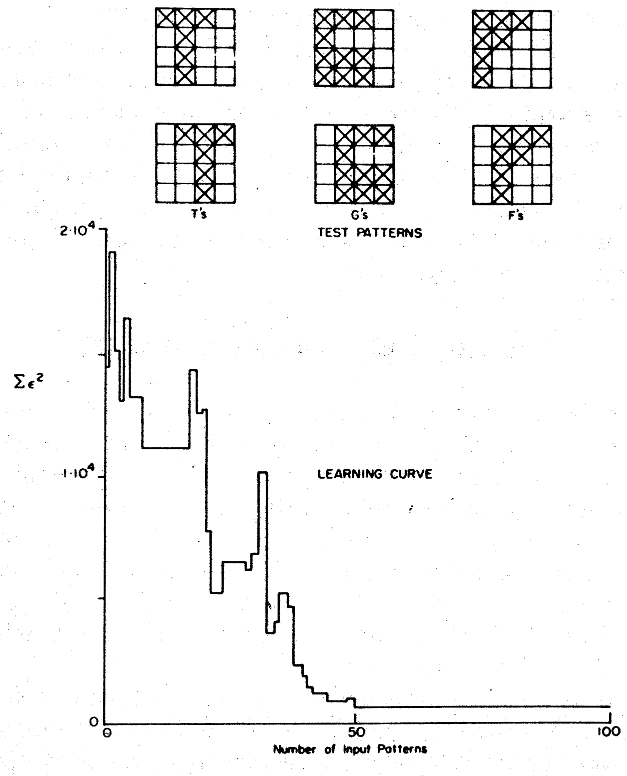


Figure 3

directions is equivalent to utilization of the method of steepest descent on a mean square error surface. A number of other steepest descent adaption procedures have been devised by W. C. Ridgway III[4] and C. H. Mays.[5] [6] These proceedures have been analyzed by Mays with regard to proofs of convcroence and bounds on the number of adaptions required for convergence. The decision to adapt may be based on one of the following rules: adapt only if the response is incorrect; or adapt only if the response is incorrect or within a "dead zone." If the decision to adapt is made, then the increment size might be fixed or might be proportional to the analog error, the difference between the analog sum and the desired output. These procedures are described in detail in Mays' Ph.D. thesis, along with bounds on the number of adaptions needed for convergence.

The effects of adaptive feedback in Adaline networks on their ability to self-heal by adapting around internal defects are analogous to the effects of feedback in amplifiers and control systems in making system performances insensitive to gain changes and nonlinearities. P. R. Low has studied by simulation and by analysis what he calls "defective" Adalines. One such Adaline has a set of weights whose integration speeds vary over a 5-to-1 ratio. These speeds are randomly selected from a uniform distribution of speeds. It was found that sometimes the nonuniformity in the adapt rates hinder and sometimes help, but on the average, this wide variation among the speeds increases the total number of adaptions required to achieve convergence, but by only 5 percent. The resultant weight values are essentially unaffected by this, as are the functions realizable and the statistical memory capacity.

THE ADALINE MEMORY CAPACITY

An important question is, how many patterns or stimuli can the single adaptive neuron be trained to react to correctly at a time? This is a statistical question. Each pattern and desired output combination represents an inequality constraint on the weights. It is possible to have inconsistencies in sets of simultaneous inequalities just as with simultaneous equalities. When the patterns (i.e., the equations) are picked at random, the number which can be picked before an inconsistency is created is a random variable. As few as 4 patterns can form a nonlinearly separable set, regardless of the pattern size.

A series of experiments was devised by J. S. Koford and R. J. Brown where patterns containing unbiased random bits and random desired responses were applied to Adalines with varying numbers of inputs. It was found that the average number of random patterns that can be absorbed by an Adaline is equal to twice the number of weights. This is one basic meas-

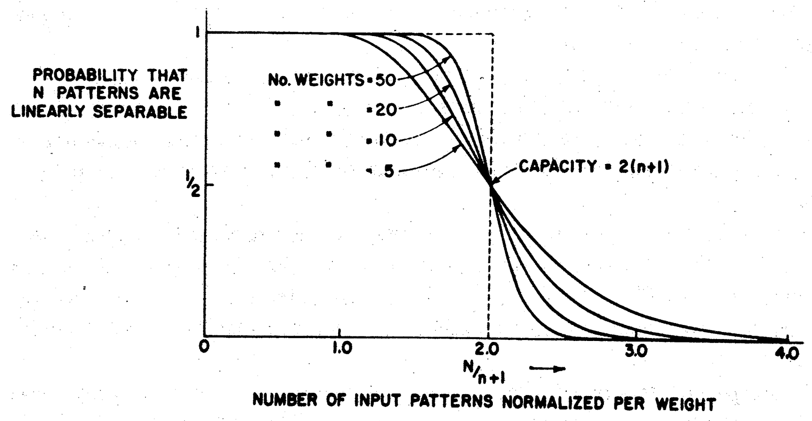


Figure 4

ure of memory capacity. It was recently proven by Brown that this experimental result is rigorously correct. Analytical curves showing the probability of being able to train-in N patterns as a function of $N/(n + 1)$ are presented in Fig. 4. Notice the sharpening of the break point of these curves at exactly the average capacity as the number of inputs to the Adaline increases.

Derivation of the capacity formula and of the curves in Fig. 4 will be presented in a Ph.D. thesis by Brown.

MADALINE, A PARALLEL NETWORK OF ADALINES

Storage capacity in excess of that of a single Adaline can be readily achieved by use of parallel multi-Adaline networks. Several Adalines can be used to assist each other in solving problems by automatic load-sharing.

The configuration in Fig. 5 shows a Madaline (multiple Adalines) of 5 Adalines with parallel-connected inputs in the first layer. In the second layer of fixed logic the Adaline outputs are connected to a majority-rule element whose output is the system output. The "job assigner," a purely mechanical device, automatically decides which Adalines if any need adaption. There are a variety of fixed-logic schemes that could be used on the second layer. M. E. Hoff, Jr., in his doctoral thesis,[7] described convergent adaption procedures that can be used with all possible fixed-logic second layers.

One procedure for training these networks is to use the "minimum-change" rule. Under this rule:

(*i*) No adaption is performed if the system output is correct.

(*ii*) If the system output is in error, a minimum number of the incorrect Adalines are adapted. The Adalines chosen for adaption are those whose analog responses require the least amount of change to give the proper response.

When adaption is performed according to the minimum-change rule, various Adalines tend to take "responsibility" for certain parts of the training problem. Thus, this rule produces load sharing among the Adalines by assigning responsibility to the Adaline or Adalines that can most easily assume it.

The adaptive system of Fig. 5 was suggested by common sense, was tested by simulation, and was found to work very well. It was subsequently proven by Ridgway in his doctoral thesis that this system will converge on a solution if a set of weights exists that will solve the training problem. The essence of the proof lies in showing that the probability of a given Adaline taking responsibility for adaption to a given pattern, desired-response pair is greatest if that Adaline had taken such responsibility during the previous adapt cycle in which the pattern was presented. The division of responsibility stabilizes at the same time that the responses of the individual Adalines to their share of the load stabilizes. In the case that the training problem is not perfectly separable by this system, it can be shown that the adaptation process tends to minimize error probability.

The memory capacities of Madaline structures utilizing both the majority element and the OR element have been measured by Koford. Although the logic functions that can be realized with these output elements are different, both types of elements yield structures with the same statistical storage capacity. The average number of patterns that a Madaline can be adapted to equals the capacity per Adaline multiplied by the number of Adalines. The memory capacity is therefore equal to twice the number of weights.

GENERALIZATION EXPERIMENTS WITH ADALINES AND SIMPLE NETWORKS OF ADALINES

With suitable pattern-response examples and the proper training procedures, generalizations can be trained into Adalines. The kinds of generalizations to be considered here are concerned with the training of Adalines to be statistically insensitive to noise, and to be sensitive or insensitive to translation, rotation, and size. Adalines can be forced to react consistently to a training set of patterns for all possible positions, for example, and then they will react consistently in all positions with high reliability on new patterns.

GENERALIZATION WITH RESPECT TO NOISE

Statistical separation of patterns consisting of a finite set of basic "prototypes" and noisy versions of these basic patterns can be readily accomplished by the single Adaline after training on the basic patterns and/or samples of the noisy patterns. A new pattern would be associated with one of the prototype classes by proximity in a Hamming distance sense.

With the objective of minimizing the probability of incorrect classification, there is an optimum set of weights that would result from training on a

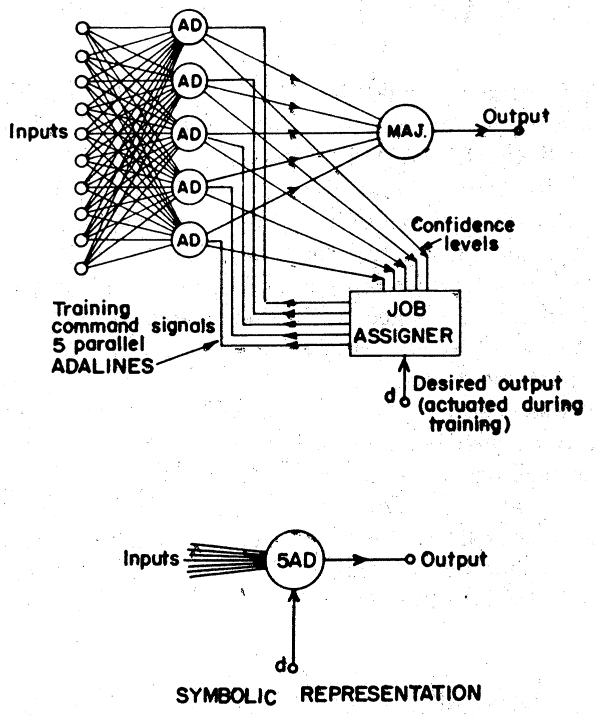


Figure 5

very large sample. The effect of training on a small sample set can be summarized with the following formula, derived in Ref. 2 and 3.

$$M = \frac{(n+1)}{N} \tag{1}$$

The number of training samples is N, randomly selected from all possible samples, and the total number of weights is $(n + 1)$. The quantity M is called the "misadjustment." It is the per unit increase in error probability, based on a minimum error probability attainable by training on a very large sample. This formula leads directly to the idea that the number of patterns required to train an Adaline to discriminate noisy patterns is about five times (making M only 20 percent) the number of weights. The number of training patterns required to produce this form of generalization is of the order of twice the statistical memory capacity.

GENERALIZATION WITH RESPECT TO ROTATION OF PATTERNS

Insensitivity to rotation by 90° is a characteristic that can be perfectly trained into an Adaline. An experiment was made as depicted in Fig. 6 by using the 4 × 4 KNOBBY ADALINE shown in Fig. 2. C's rotated in all four positions were trained-in to give the $+1$ response, while T's were trained-in to give the -1 response in all four rotations. The initial weights were set to zero, and during training, the minimum mean-square error adaption procedure with an adaptive time constant of 32 patterns was utilized. The process converged with the desired responses trained-in precisely, and the set of weights shown in Fig. 6 resulted. Without further training, new patterns totally unrelated to the training patterns were inserted, and it was observed that not only were the decisions made by the Adaline perfectly consistent for each pattern over the four rotations, but the four meter readings (confidence levels or analog outputs) for each pattern were identical. The reason for this is simple: Rotation of the weights by 90° yields an identical set of weights. Let the a-matrix represent the set of weights (not including the threshold weight). The threshold weight remains the same for all rotations. The superscript R represents rotation by $+90°$.

$$[a] = [a]^R = \left[[a]^R\right]^R = \left[\left[[a]^R\right]^R\right]^R \tag{2}$$

Other training patterns and other numbers of training patterns were used in this experiment, and in each case, after convergence, the same

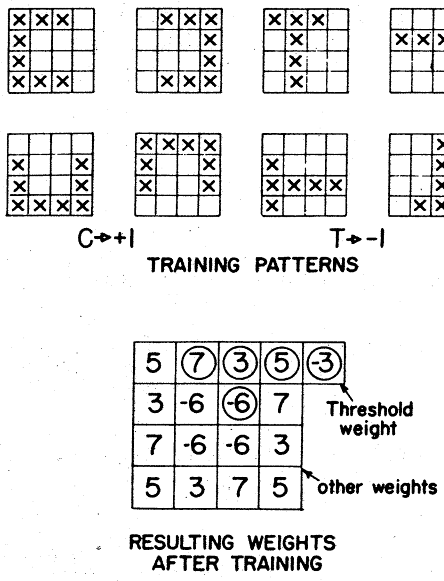


Figure 6

symmetry expressed in Eq. (2) resulted automatically. Adaptation with a time constant, long compared to the number of training patterns, allows the neuron to retain responses to all the training patterns essentially equally. Minimization of mean-square error forces the response voltage to each training pattern in all four rotations to be consistent even when it might not be possible for this voltage to be precisely +1 or −1. This forces the symmetry of Eq. (2).

An interesting question is, how many specific responses on the average can be trained in and yet have the neuron trained to be insensitive to 90° rotation for all patterns. The 4 × 4 neuron has a capacity of 32 patterns. Eight basic patterns on the average can be trained in, since each basic pattern must be inserted in all four rotations. Another point of view on this question was suggested by Hoff. The four encircled weights and the threshold shown in Fig. 6, once chosen, set the rest of the weights when the constraint of Eq. (2) is followed. There are 4 "degrees of freedom" plus the threshold freedom. The number of basic patterns that can be discriminated therefore corresponds to the capacity of a 4-input neuron which is 8 patterns.

The same training procedure could be used to train-in a direct sensitivity to rotation, rather than an insensitivity. The experiment was remade, with a C mapped $+1$ and a T mapped -1, however with a rotated C mapped -1, and a rotated T mapped $+1$, etc., the following set of weights resulted.

$$\begin{array}{rrrrl}
-11 & +9 & -2 & +11 & 0 \nwarrow \\
+2 & 0 & 0 & -9 & \\
-9 & 0 & 0 & +2 & \text{threshold} \\
+11 & -2 & +9 & -11 & \text{weight}
\end{array}$$

The symmetry in the weights can be described by

$$[a] = -[a]^R = -\left[[a]^R\right]^R = -\left[-\left[-[a]^R\right]^R\right]^R \tag{3}$$

Rotation of any input pattern by 90^0 causes a sign reversal in the confidence level, and therefore an opposite decision.

GENERALIZATION WITH RESPECT TO LEFT-RIGHT TRANSLATION

Perfect solutions to the problem of training an Adaline to be insensitive to left-right pattern translation exist. A solution requires the columns of the a-matrix to be identical. On a 4×4 input array, there is a choice of 4 independent weights, each choice setting a row of weight values. It follows that the statistical discrimination capacity subject to the constraint of insensitivity to left-right translation is that of a 4-input Adaline or 8 basic patterns. The total capacity of the 4×4 Adaline is 32 patterns, and this corresponds to the four positional possibilities for each of the 8 basic patterns. Patterns can be placed in four positions by considering the input pattern space to be continuous and folded over a cylinder having a vertical axis.

By symmetry, the same training procedures apply to training for insensitivity to up-down motion. If both left-right and up-down insensitivity is desired, the only perfect solution is the relatively trivial one, all weights in the a-matrix being equal. Discrimination is based on pattern "area," the number of $+1$ pattern bits. More sophisticated discrimination based on pattern features other than area has been made by using two Adalines and an OR output element in the form of a simple Madaline.

An experiment was made to train a KNOBBY ADALINE to give a sign reversal for left-right motion and, at the same time, to give a sign reversal for rotation by 90°. The pattern T on a 3×2 grid was trained in to produce $+1$ in the vertical left position, -1 in the vertical right position, etc. The following set of weights resulted. Notice the symmetries and sign alternations.

8	−5	5	−8	−2
5	−1	1	5	
5	1	−1	−5	
−8	5	−5	8	

It was found that approximately 85 percent of new patterns would consistently produce sign alternation for all possible left-right, up-down, and rotation by 90° motions. The remaining 15 percent of patterns would be consistent in most situations, perhaps be incorrect in only 2 out of 16 cases. Symmetrical patterns would be perfectly consistent.

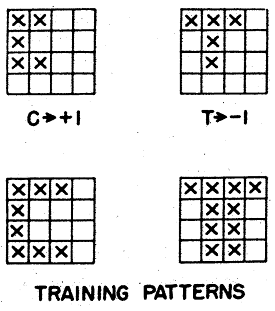
GENERALIZATION WITH RESPECT TO PATTERN SIZE

An Adaline can be trained to be highly insensitive to pattern size. The training procedure again requires slow minimum mean-square-error adaption. In Fig. 7, a set of "small" and "large" patterns is shown that comprised examples for the training experiment. On a 3 × 3 array in the upper left hand corner of a 4 × 4, a *T* and a *C* were inserted as shown. In the full 4 × 4 array, expanded versions of these patterns were trained-in to give corresponding responses. After training, it was found that new patterns gave widely fluctuating analog responses. For about 90 percent of new pattern inputs, the same binary response resulted for the small as for the large versions, and the corresponding confidence levels were extremely close.

To be perfectly insensitive to size, the weights of an Adaline must be such that an element of area of the small pattern "sees" the same total weight (input patterns are thought of as continuous two-dimensional functions and weights are thought of as continuous distribution functions) as the corresponding area element of the large pattern that it maps into. It can be shown that perfect solutions result when the weight function radiates from a point and has an intensity that decays with an inverse-square law. These general effects can be detected in the weights of Fig. 7.

APPLICATIONS OF PATTERN CLASSIFICATION PRINCIPLES TO SYSTEMS PROBLEMS

In addition to an application to automatic control systems which will be described in detail in the next section, the above principles have been applied to weather forecasting, speech recognition, and diagnosis of EKG waveforms.



2	1	-5	-1	1
7	-7	-1	-1	
8	-1	-1	-1	
1	1	1	-1	

RESULTING WEIGHTS

Figure 7

WEATHER FORCASTING

One of the highly successful applications of Adaline-type trainable threshold networks has been to weather forecasting. This work has been performed mainly by M. J. C. Hu, a graduate student at Stanford, with the cooperation of Mr. H. E. Root of the U.S. Weather Bureau at the San Francisco International Airport.

Measurements of sea-level barometric pressure at a number of points around San Francisco were applied, after appropriate encoding, to the inputs of a network of Adalines. The desired outputs to which the network was trained was whether or not it rained during some future interval at San Francisco. Both two-level and multilevel input Adalines have been used for these experiments.

Consider the results obtained from an illustrative experiment. A three-Adaline system was used to give three separate forecasts, covering three successive 12-hour periods in the future. One Adaline was trained to indicate whether or not it rained from 8 a.m. to 8 p.m. on the day when the pressure map was made (the map was made at 4 a.m.). The other two Adalines were trained, using the same data, to forecast whether or not it rained from 8 p.m. of the same day to 8 a.m. of the next day, and from 8 a.m. to 8 p.m. of the next day, respectively.

The experiment utilized three types of weather information. The three-Adaline system was trained to recognize 33 patterns of each of the following types.

1. Today's 4 a.m. PST (Pacific Standard Time) Surface Pressure Map.
2. Today's 4 a.m. PST and yesterday's 4 a.m. PST Surface Pressure Maps.
3. Today's 4 a.m. PST Surface Pressure Map and the difference between today's and yesterday's pressure (*dp*/*dt*).

The three-Adaline system was then tested on 18 patterns from each of the types mentioned above. The performance of the three-Adaline system was then compared with the official forecast for those 18 days. The results are tabulated below:

		Percent Right	
	Today 8 a.m.–8 p.m.	Tonight 8 p.m.–8 a.m.	Tomorrow 8 a.m.–8 p.m
Official forecast	78	89	67
Adaline forecast using			
1. 4 a.m. PST Map	72	67	67
2. Today's and yesterday's 4 a.m. PST Maps	78	78	78
3. Today's 4 a.m. PST Map and *dp*/*dt*	78	89	83

The success of this work, particularly considering that only barometric pressure was used for generating the Adaline forecasts, has resulted in growing interest, both at Stanford and among local meteorologists, in this technique.

SPEECH RECOGNITION

A small-scale, real-time, trainable speech recognition system has been built and studied extensively by L. R. Talbert, G. F. Groner, and J. S. Koford, P. R. Low, and R. J. Brown, with the advice of Dr. Dorothy A. Huntington of the Speech Pathology and Audiology Department at Stanford. This system consists of a microphone-input speech preprocessor, which feeds a speech waveform of normalized amplitude into eight bandpass filters spaced throughout the audio spectrum. The detected outputs of these filters (proportional to spectral energy) are then quantized, digitally coded, and sampled for application directly to a simulated Adaline network.

In a typical experiment the output of each of the bandpass filters is quantized into four levels, and is then represented by a simple three-bit linearly separable code (this will be explained in the next section). Ten samples of each quantized output are then taken at equal intervals throughout the duration of a spoken word. Thus, each spoken word is represented by 8 × 3 × 10, or 240, bits. k-simulated Adalines are used to provide classification for 2^k words. Thus, for recognition of the spoken digits 1 to 10, four simulated Adalines of 240 inputs each were used.

In use, training samples of the particular voice to be recognized are first taken, and the Adalines are trained to correctly classify these training samples. After convergence on the training samples, new spoken samples can be used to test the ability of the network. After training on ten samples of each spoken word from one individual, the network can typically classify new samples of the same spoken word by that individual with 98 percent accuracy or better. If tested on a new individual, this accuracy averages 90 percent or better.

VECTORCARDIOGRAM DIAGNOSIS

Networks of Adalines have been applied with encouragingly good success by D. F. Specht with help, advice, and data supplied by Drs. J. G. Toole and J. Von der Groeben of the EKG Department, Stanford University, School of Medicine to the diagnosis of heart defects from examination of vectorcardiograms. A vectorcardiogram differs somewhat from the usual 12-track electrocardiogram in that only three sets of data are recorded, but they are recorded simultaneously, so that significant phase information among them is preserved. The input patterns to an Adaline network are formed by sampling the vectorcardiograms at 5-millisec intervals. During training on approximately 100 sample patterns, the desired outputs were based on an electrocardiologist's diagnosis. The following table indicates the sort of success which has been obtained in preliminary experiments.

Recognition Rate on the Testing Set

	True Normals, % (27 cases)	*True Abnormals,* % (30 cases)
Clinical EKG	95	54
Generalized adaptive approach	89	73

APPLICATION TO CONTROL SYSTEMS

The state of a dynamic system can be completely described at any instant by the values of the state variables of the system. (The state variables of a control system are such quantities as the error, the error derivative, etc.) A control decision therefore need depend only on the present values of the state variables. The value of each state variable can be encoded as a sequence of binary digits. The collection of these encoded state variables forms a pattern. Proper control of a dynamic system by an Adaline or Madaline becomes a matter of the proper classification of the patterns which represent the different states of a dynamic system. Just as an Adaline can be taught to classify patterns into two groups, it can also be taught to control a dynamic system in a "bang-bang" or $+1$, -1 manner.

When the state variables are encoded using what has been called a "linearly independent code," the task of learning control strategies is quite natural for an Adaline.

i. The large sets of patterns representing the control strategy for all possible regions of state space are often either linearly separable, or separable with simple Madaline structures. Then umberof patterns which the Adaline is able to correctly classify is generally a norder of magnitude or more greater than its statistical capacity.

ii. The Adaline generalizes in a known and predictable way. Namely, the Adaline can correctly classify all the patterns of a control strategy after learning to correctly classify only the patterns bordering on the switching surface.

Because of this strong generalizing property and because of the special interrelationships among the many patterns, the Adaline is much easier to train than it would be for a similar number of random or near random patterns.

THE TRAINABLE CONTROLLER

Figure 8 shows in block diagram form the general situation in which a Madaline would be used as a trainable controller for a dynamic system. The state variable $y_1, \ldots y_m$ are assumed to be the system error.

The teaching controller supplies the desired output to the Adaline during the training process. This controller could be an automatic controller or possibly a human. The Adaline controller and the teaching controllre need not have the same inputs, provided both receive the same or related information. For instance, the Adaline controller could be receiving the state variables as electronic signals while a human teacher could be receiving information about the system by actually watching its motions.

For the purposes of discussion the teacher will be assumed to be represented by a function $f(y_1, \ldots, y_m)$. The switching surface $f(y_1, \ldots, y_m) = 0$ describes the transition where the teacher changes his reaction from "force plus" to "force minus." During the training, the Adaline analog output $\hat{f}(y_1, \ldots, y_m)$ is adjusted so that its switching surface $\hat{f}(y_1, \ldots, y_m) = 0$ is made to approximate the switching surface of the teacher.

The Adaline controller consists of an encoder and an Adaline. For simplicity, a single Adaline is shown here in the controller; more typically a Madaline might be used. The Adaline with its encoder is basically a trainable function generator which forms the function $\hat{f}(y_1, \ldots, y_m)$. The pattern inputs to the Adaline change continually as the state variables change. The encoder produces patterns by quantizing or dividing the range over which each of the state variables varies into a finite number of zones. Each zone of a state variable y_i is represented by binary number or *partial pattern*. The m partial patterns make up the total pattern which represents a particular hypercube of state space.

Figure 9 illustrates the quantization of a two-dimensional state space. Each square in the figure is represented by a particular pattern for the Adaline. The continuous curve $\hat{f}(y_1, y_2) = 0$ represents a typical desired switching surface (a curved line in this case). The jagged curve $\hat{f}(y_1, y_2) = 0$ is the switching curve that an Adaline controller might use to approximate the teaching controller.

The system has two modes of operation:

i. During the training mode, the teaching controller controls the dynamic system. The adapt logic in the Adaline continuously compares the binary output of the Adaline with that of the teacher. Whenever they differ, the Adaline is adapted in the direction which would make them agree. Because the patterns change rapidly, there may not be time for a full

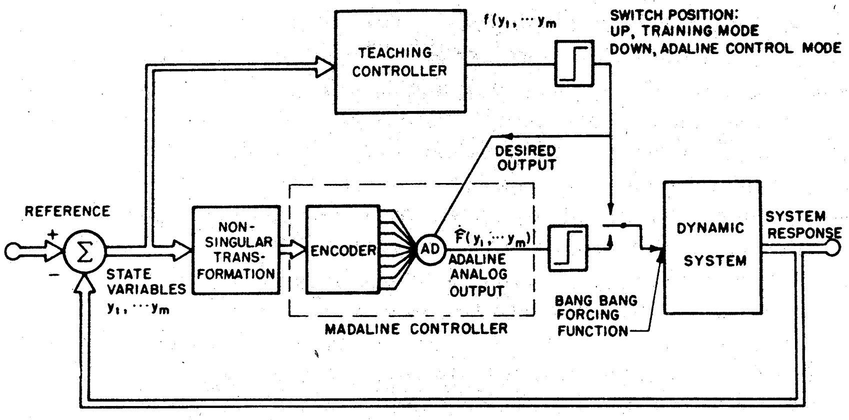


Figure 8

correction. However, the pattern is bound to recur, at which time adaption can be continued. During the training mode the Adaline controller "watches" the teacher zero the error after various large disturbances or initial contions.

ii. During the Adaline control mode the teaching controller is not used and may be completely removed from the system.

CODING

The choice of codes used to represent the values of the state variables largely determines how well the Madaline controller will be able to imitate its teacher. Figure 10 illustrates two possible "linearly independent codes." A linearly independent code is any code which has a nonsingular *partial pattern matrix*. This matrix has the partial patterns as rows plus a column of ones (if necessary). The partial pattern matrix for the codes of Figs. 10*a* and *b* are respectively:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Both matrices are obviously invertible. When linearly independent coding is used, the Adaline will be able to exactly imitate (except for quantization effects) any teacher whose function does not contain cross-product terms, i.e., terms of the form y_iy_j, $i \neq j$, regardless of the number of patterns.

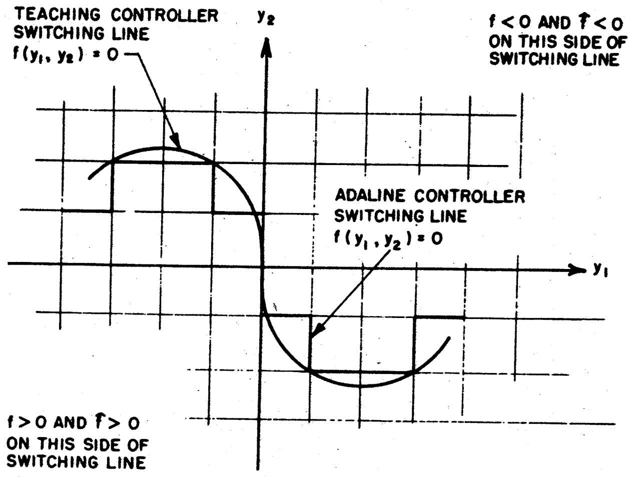

TEACHING CONTROLLER SWITCHING LINE $f(y_1, y_2) = 0$
y_2
$f < 0$ AND $\hat{f} < 0$ ON THIS SIDE OF SWITCHING LINE
ADALINE CONTROLLER SWITCHING LINE $\hat{f}(y_1, y_2) = 0$
y_1
$f > 0$ AND $\hat{f} > 0$ ON THIS SIDE OF SWITCHING LINE

Figure 9

The proof that an Adaline using linearly independent coding has such classifying power can be given by demonstrating of how the Adaline matches its function to that of the teacher. The analysis an Adaline is much simpler for the class of classification problems in which a decision is based on the encoded values of state variables than for most problems. The simplification occurs because the Adaline can be fully described in terms of the m state variables instead of the n binary inputs, ($n \gg m$ generally). The weights for the inputs used to encode a particular state variable are considered not as separate entities but as a single function of the state variable. The abilities and limitations of a single Adaline in pattern classification and generalization become apparent. Also, the weights can often be calculated in many seemingly complicated problems.

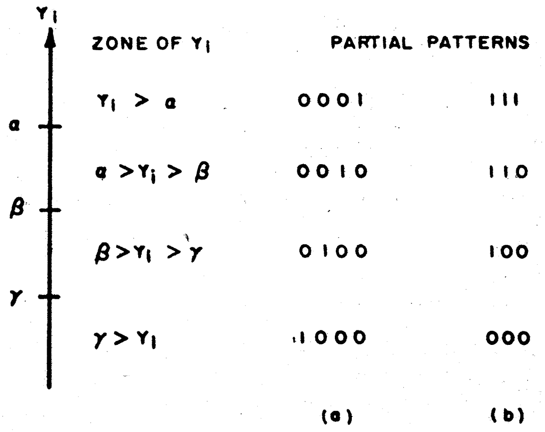
This new interpretation of the Adaline is for analytical purposes only. The Adaline is trained in the usual way. The function matching to be described goes on automatically "inside" the Adaline.

Let the schematic of the Adaline be redrawn as in Fig. 11. Only the method of summing has been changed so as to allow the quantities $f_i(y_i)$, $i = l, \ldots, m$, to be defined. The threshold weight w_0 is figuratively considered to be divided into m thresholds w_{i0}, where

$$\sum_{i=1}^{m} w_{i0} = w_0 \tag{4}$$

Each partial sum $\hat{f}_i(y_i)$ is a function of only the state variable y_i. The switching surface for the Adaline is

$$\hat{f} = \sum_{i=1}^{m} f_i(y_i) = 0 \tag{5}$$



ZONE OF Y_I	PARTIAL	PATTERNS
$Y_I > \alpha$	0001	111
$\alpha > Y_I > \beta$	0010	110
$\beta > Y_I > \gamma$	0100	100
$\gamma > Y_I$	1000	000
	(a)	(b)

Figure 10

This function has no cross product terms and cannot approximate teaching functions with cross-product terms. Thus, the Adaline can imitate only teaching functions of the form

$$f = \sum_{i=1}^{m} f_i(y_i) = 0 \tag{6}$$

This is a consequence of encoding each state variable independent of the others.

If the switching surface of the Adaline controller is to imitate that of the teaching controller, $\hat{f}$ must be proportional to f, and furthermore, because the y_i can vary independently, each $\hat{f}_i(y_i)$ must be proportional to $f_i(y_i)$. (Assume a proportionality constant of one here.) Thus the coding of the state variables into patterns can be studied by examining how a *single* variable is encoded.

Consider the state variable y_i. When y_i is in a particular quantum zone the partial sum $\hat{f}_i(y_i) = (\vec{a})^T \cdot (\vec{W}_i)$. The vector $(\vec{a})$ is the partial pattern associated with that zone, augmented by a $+1$ "threshold input" as its first entry. The vector $(\vec{W}_i)$ contains the weights associated with y_i in Fig. 12. The threshold weight w_{i0} is the first entry. If $\hat{f}_i(y_i)$ is to match $f_i(y_i)$, then they should be equal somewhere within each quantum zone. Thus:

$$(\vec{a})^T \cdot (\vec{w}_i) = f_i(y_i)$$

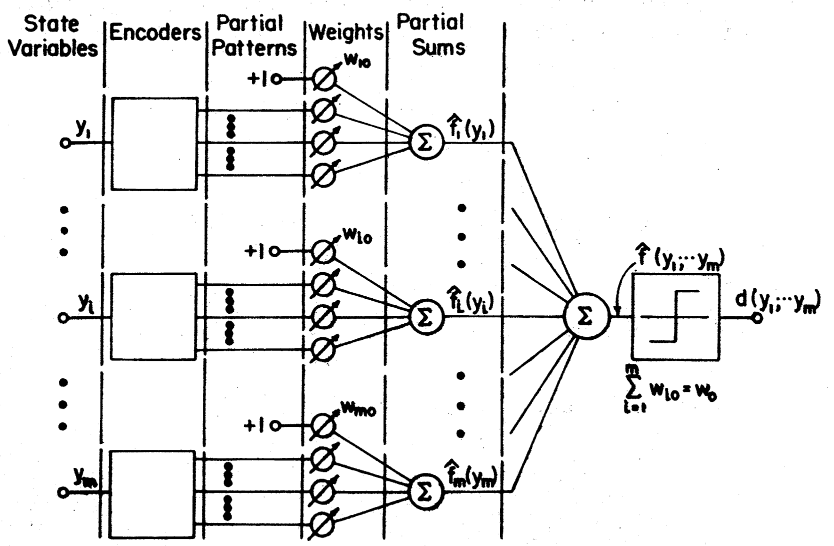


Figure 11

somewhere within each quantum zone. These equations (one from each quantum zone) must have a simultaneous solution. They can be rewritten more compactly

$$[A]\,(\vec{w}_i) = (\vec{f}_i) \tag{7}$$

$[A]$ is the partial pattern matrix described above. The $(\vec{f}_i)$ is a vector containing values of $f_i(y_i)$ at which $\hat{f}_i(y_i) = f_i(y_i)$.

When the equations $[A](\vec{w}_i) = (\vec{f}_i)$ are consistent $\hat{f}_i(y_i) = f_i(y_i)$ is possible and the Adaline partial sum will be able to "exactly" imitate the partial sum of the teaching function, $f_i(y_i)$. "Exactly" is meant in the sense that $\hat{f}_i(y_i) = f_i(y_i)$ for at least one value of y_i in each zone of y_i. If each of the $\hat{f}_i(y_i)$ is "exactly" equal to its corresponding $f_i(y_i)$ for all i, then $\hat{f}(y_1, \ldots, y_m)$ will equal $f(y_1, \ldots, y_m)$ somewhere within each hypercube of state space. Furthermore, the hypercube, and its pattern, will have the same sign as $f(y_1, \ldots, y_m)$ at this point. (These points of equality are indicated in Fig. 13.)

If $f_i(y_i)$ is an arbitrary function, then the only way in which the equations $[A]\,(\vec{w}_i) = (\vec{f}_i)$ can be guaranteed to be consistent is to choose partial patterns so that $[A]$ has a left inverse. To minimize the number of weights $[A]$ must also have a minimum number of columns. The only form of $[A]$ which satisfies both of these criteria is an $[A]$ which is square and invertible. Thus, for an arbitrary $f_i(y_i)$ the partial patterns representing y_i must be such that $[A]$ has an inverse. Then, $\vec{w}_i = [A]^1{}_1(\vec{f}_i)$ for any $f_i(y_i)$. Obviously, there are many possible $[A]$'s.

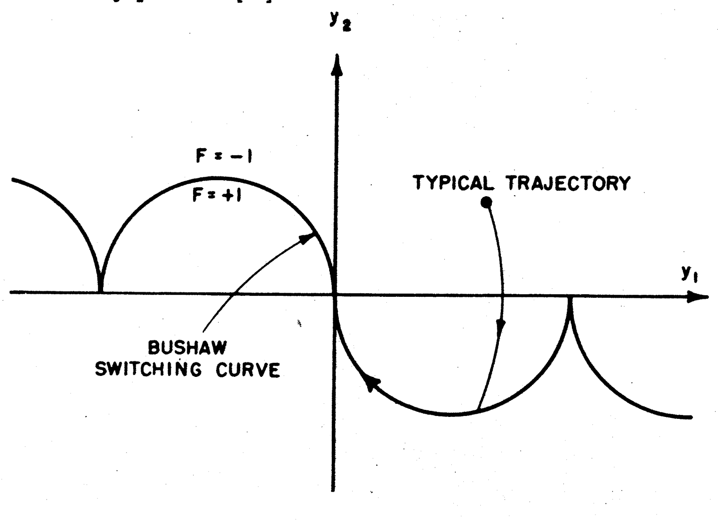


Figure 12

Two possible ways of encoding the state variables have been illustrated in Fig. 10. The "single spot" code of Fig. 10*a* is easy to analyze mathematically because most of the weights would have zero coefficients. If the threshold weight is not used, the weights are:

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = (\vec{f}_i)$$

The "multispot" code of Fig. 10*b* is illustrated because it is usually quite easy to instrument and also because this code usually allows the weights of the Adaline to be quite small. Other authors[5] have shown that, in general, the smaller the magnitudes of the weights (after proper normalization) the easier it will be to train the Adaline.

The use of linearly independent coding shows that it is sufficient to guarantee that the Adaline function generator will be able to "exactly" imitate a teaching function that has no cross-product terms. By a more involved argument, it can be shown that linearly independent coding is necessary if the Adaline controller is to have a minimum number of weights while "exactly" imitating a teaching function with no cross terms. A proof of necessity is hardly needed, however, when the nonstatistical capacity of an Adaline using linearly independent coding is considered. For in-

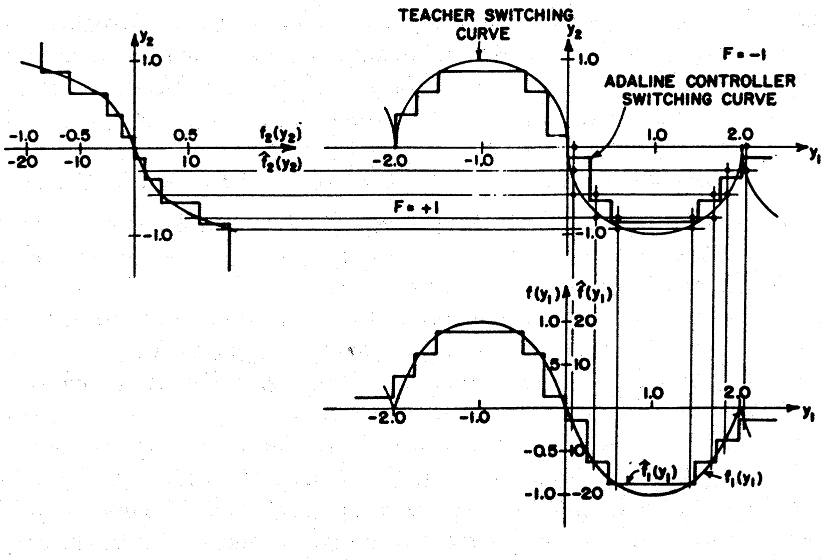


Figure 13

stance, when each of the state variables is quantized into n' zones, there are $(n')^m$ possible patterns. The statistical capacity of this Adaline is approximately $2mn'$.

Consider the values $n' = 5$; $m = 4$. Then $(n')^m = 625$ patterns can be correctly classified while the statistical capacity is only $2mn' = 40$. The actual capacity in this typical case is more than an order of magnitude greater than the statistical capacity.

AN EXAMPLE OF AN ADALINE CONTROLLER

The above ideas can best be illustrated by showing how an Adaline controller would control the oscillatory undamped second-order system with differential equation

$$\ddot{y}_1 + y_1 = F, \quad F = \pm 1, \quad y_2 = \dot{y}_1 \tag{8}$$

The minimum-time optimum-switching curve is the well-known Bushaw[8] switching curve shown in Fig. 12. This switching surface was chosen for an example because it gives appreciably faster response than a linear switching line, and because it is highly nonlinear.[9] A typical trajectory of the minimum time optimum controller is also shown in Fig. 12. The optimal controller makes no "wrong" decisions but instead moves right to the origin with one reversal of F. A controller containing one Adaline is capable of closely approximating the nonlinear function of the optimum controller. The switching surface of the Adaline controller is shown in Fig. 13 with functions $f_l(y_l)$, $f_1(y_1)$, $\hat{f}_1(y_1)$, and $f_2(y_2)$. The weights needed to realize these functions are shown in Fig. 14. The weights of (a) are for the "single spot" coding of Fig. 10a, while those of (b) are for the "multispot" coding of Fig. 10b. The porportionality constant relating $\hat{f}(y_1, y_2)$ and $f(y_1, y_2)$ is 20.

IMITATION OF FUNCTIONS WITH CROSS-PRODUCT TERMS

Previously it was shown that a single Adaline controller can imitate "exactly" teaching functions which do not have cross-product terms. Functions containing cross products can be realized in two ways. One way would be to encode additional variables which were the desired cross-product terms. Another and more satisfactory approach would be to use several Adalines together in a Madaline. Encoding additional variables has the disadvantage that there are an extremely large number of possible cross-product terms (even when only low-order terms are considered). With no *a priori* information available to indicate which cross-product terms are necessary, they would all have to be encoded. The operation of a Madaline structure can be described in function-generator terms briefly as follows: The quantizers at the Adaline outputs and the fixed logic element which combines these outputs perform nonlinear operations which can introduce the necessary cross-product terms. A Madaline does not need a weight for every cross-product term because it can organize its total structure in such a way as to take into account the most significant cross-product terms while ignoring the rest.

The situation illustrated in Figs. 15*a*, *b*, and *c* demonstrates the ability of a Madaline structure to imitate a teaching function with cross-product terms. The teaching function is a rotated ellipse with equation:

$$5y_1^2 - 6y_1y_2 + 5y_2^2 - 2 = 0 \tag{9}$$

This curve was chosen as a familiar nonlinear function. Two Adalines are used. Adaline I in Fig. 15*b* has the U shaped switching line which approximates the switching line of half of the teaching function. Adaline II in Fig. 15*c* has the inverted U-shaped switching line which approximates the other half of the teaching function. The Adaline outputs are combined in an OR circuit. The logic of the OR circuit is: both Adaline outputs -1 then Madaline output -1 otherwise Madaline output $+1$. With the polarity of the Adaline outputs as shown on the figure the OR circuit causes the interior of the ellipse to be -1 as desired. The functions approximated by the individual Adalines can be shown to contain no cross terms.

$$\vec{W}_1 = \begin{bmatrix} 1 \\ 5 \\ 11 \\ 16 \\ 16 \\ 16 \\ 16 \\ 11 \\ 1 \\ -1 \\ -11 \\ -16 \\ -16 \\ -16 \\ -16 \\ -11 \\ -5 \\ -1 \end{bmatrix} \quad \vec{W}_2 = \begin{bmatrix} +17 \\ +12 \\ +5 \\ +2 \\ 0 \\ -2 \\ -5 \\ -12 \\ -17 \end{bmatrix}$$

NO THRESHOLD WEIGHT

(a)

$$\vec{W}_1 = \begin{bmatrix} 1 \\ 4 \\ 6 \\ 5 \\ 0 \\ 0 \\ 0 \\ -5 \\ -10 \\ -2 \\ -10 \\ -5 \\ 0 \\ 0 \\ 0 \\ +5 \\ +6 \\ +4 \end{bmatrix} \quad \vec{W}_2 = \begin{bmatrix} +17 \\ -5 \\ -7 \\ -3 \\ -2 \\ -2 \\ -3 \\ -7 \\ -5 \end{bmatrix}$$

$$W_0 = W_{10} + W_{20} = 1 + (-17) = -16$$

(b)

Figure 14

"BROOM-BALANCING MACHINE"

To demonstrate the ideas presented thus far in this chapter a relatively complex dynamic system with an Adaline controller has been assembled. The dynamic system is a motorized cart carrying an inverted pendulum. The controller for the system is required to keep the pendulum balanced and keep the cart within certain bounds by applying a horizontal force to the cart. An actual memistor Adaline is used in the trainable controller. This form of physical realization of adaptive logic circuits will be explained below. Figure 8 gives a block diagram of the dynamic system and its controllers. The nonsingular transformation in Fig. 8 is the identity transformation in this case.

The cart and pendulum system is an undamped and inherently unstable fourth-order dynamic system. The four-state variables are the angle of the pendulum from vertical, θ; the rate of change of angle θ; the position of the cart, x; and the rate of change of position x. These and other relevant

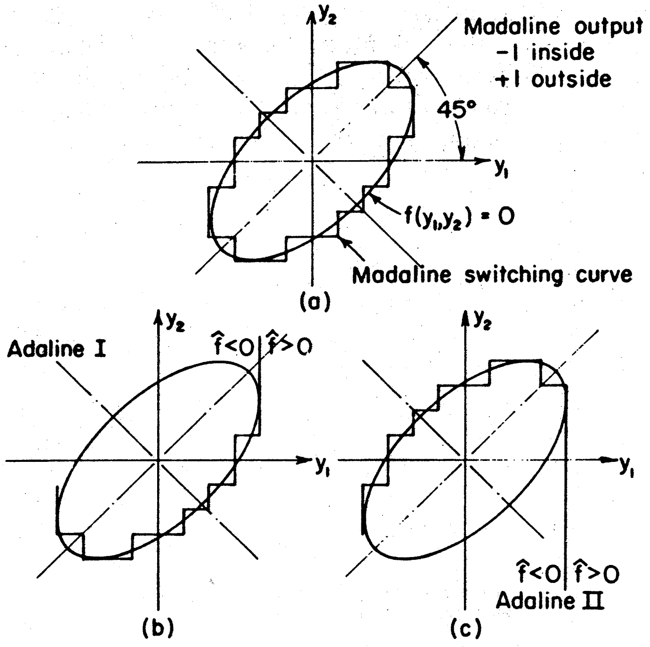


Figure 15

quantities are defined in Fig. 16. The linearized differential equations representing this system are:

$$\ddot{\theta} = \frac{3g}{4l}\theta - \frac{3}{4lM}F$$
$$\ddot{x} = \frac{1}{M}F \qquad (10)$$

It is assumed that there is no damping, and that the reaction of the pendulum motions on the cart is negligible.

The teaching controller being used in these experiments has a linear switching surface of approximately

$$f = -2.0\,\dot{\theta} - 1.0\,\theta + 1.0\dot{x} + 1.0x \qquad (11)$$

The Adaline controller contains one 24-input Adaline. The range of each of the state variables is divided into seven approximately equal zones. The state variables are encoded into 6-bit partial patterns using a linearly independent code similar to the one illustrated in Fig. 10*b*. The controller is taught by having it observe the teacher return the system to the origin of state space after it has received various large disturbances.

"SELECTIVE BOOTSTRAPPING"

The Adaline controller illustrated in Fig. 8 is a trainable adaptive system, but not a self-optimizing one. A teacher must exist in some form to serve as an example for the imitating Adaline. An alternative self-optimizing

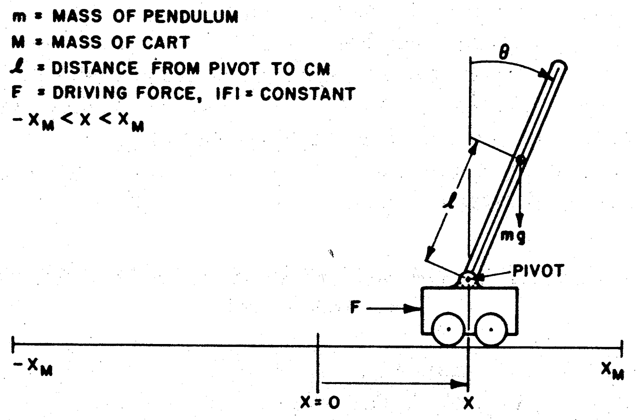


Figure 16

technique for the training of an Adaline controller, that of "selective bootstrapping," has been studied by E. C. Fraser. Using this method, the Adaline controller is allowed to operate with the Adaline weights set to any arbitrary initial values—thus realizing an arbitrary control function. The performance of the system, when operating in this manner, is observed and evaluated. When, in the judgment of the observer, the performance over long chains of decisions is acceptable, the Adaline is adapted to reinforce such decisions; conversely, if the performance is unacceptable, the Adaline is adapted to reverse the decisions as they are being made. If the observer is uncertain, no change is made in the Adaline weights. In this way, training information is obtained directly from the performance and does not depend on any knowledge of the dynamic system configuration. That the system be in a stable operating condition is not a requirement of this technique since information about a system's performance can be obtained even though it is at the moment unstable. This is similar to a person trying to balance a broom on his finger; even though he may drop it (the system undergoes an unstable runaway), he learns something from the experience which aids him to do a better job the next time. Of course, systems to which this technique is applicable must be such that they can be stopped and restarted whenever the output gets out of bounds (i.e., the broom falls over).

It can be shown that, for second-order systems at least, when this technique is employed, the system will converge to a stable configuration if the performance evaluations of the observer are correct greater than 50 percent of the time. For higher-order systems this limit may be somewhat higher. The average rate of convergence is related to the observer's performance and is maximum when he is 100 percent correct and becomes zero when he is 50 percent correct. For values less than 50 percent, the system tends toward unstable configurations.

Experimental evidence has been collected to verify the convergent properties of this technique. In experiments conducted with computer-simulated systems it was found that an observer familiar with control system theory, but ignorant of the plant configuration, would consistently produce training sequences leading to stable system configurations. The observer was found to range from 55 to 62 percent correct in his evaluations.

REALIZATION OF ADAPTIVE CONTROL CIRCUITS BY MEMISTORS

In large networks of adaptive neurons it is imperative that the adaptive processes be fully automated. The structure of the Adaline neuron and the adaption procedures used with it are sufficiently simple, that it has been possible to develop electronic automatically adapted neurons which are reliable, contain few parts, and are suitable for mass production. In such neurons it is necessary to be able to store weight values, analog quantities which could be positive or negative, in such a way that these values could be changed electronically.

A new electrochemical circuit element called the Memistor (a resistor with memory) has been devised by B. Widrow and M. E. Hoff for the realization of automatically adapted Adalines. The Memistor provides a single variable gain element. Each neuron therefore employs a number of Memistors equal to the number of input lines, plus one for the threshold.

A Memistor consists of a conductive substrate with insulated connecting leads, and a metallic anode, all in an electrolytic plating bath. The conductance of the element is reversibly controlled by electroplating. Like the transistor, the Memistor is a 3-terminal element. The conductance between two of the terminals is controlled by the time integral of the current in the third terminal, rather than by its instantaneous value, as in the transistor. Reproducible elements have been made which are continuously variable, which vary in resistance from 50 to 2 ohms, and cover this range in about 15 sec with several tenths of a milliampere of plating current. Adaptation is accomplished by direct current, while sensing is accomplished nondestructively with alternating current.

Although the Memistor is still an experimental device, it is in limited commercial production. Figure 17 shows how they are made, 21 at a time on a common substrate. Each cell has a volume of about 2 drops. The entire unit is encapsulated in epoxy.

The broom-balancer has been controlled by an adaptive machine called Madaline I, containing 102 Memistors. This machine was constructed hastily over a 1½-month period. The Memistors were not tested before installation in the machine, and some were defective when first made. A number of wiring errors existed; some weights were adapting to diverge rather than converge. There were a number of short circuits, open circuits, cold solder, joints, etc. This machine worked very well when first turned on, and has functioned with very little attention over the past year and a half. After several weeks of experimentation the individual weights were checked. Twenty-five percent of them were not adapting. Yet the machine was able to adapt around these internal flaws and was able to be trained to make very complex pattern discriminations. Self-repairing control systems are a very real and vital possibility.

ACKNOWLEDGMENTS

This work was performed under Office of Naval Research Contract Nonr 225 (24), NR 373 360, jointly supported by the U.S. Army Signal Corps, the U.S. Air Force and the U.S. Navy (Office of Naval Research), under

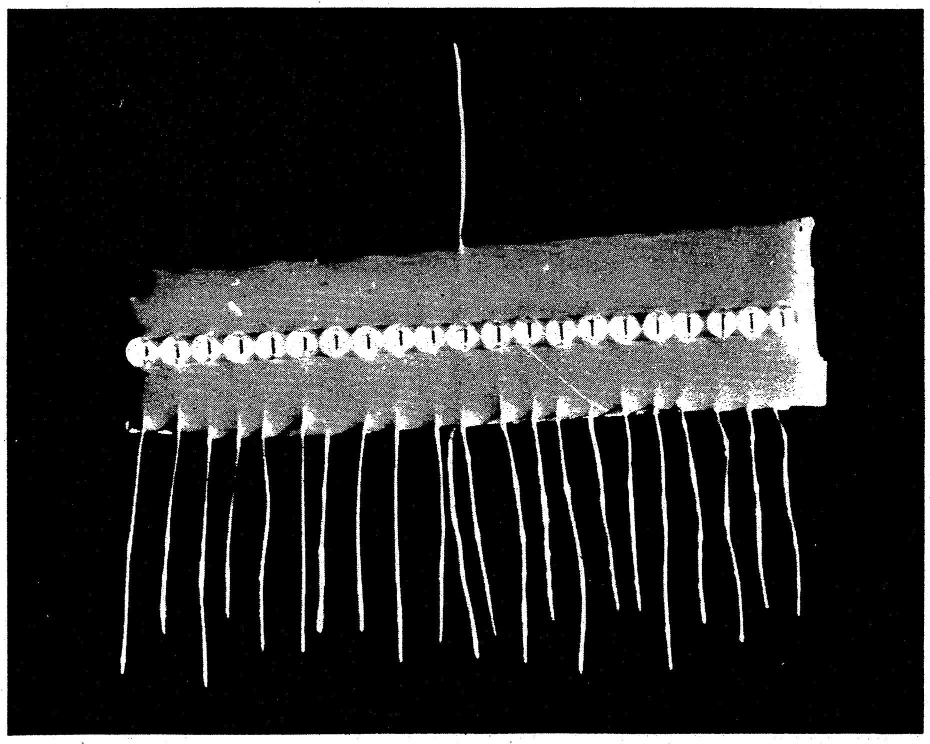


Figure 17

Air Force Contract AF33(616)7726 supported by Aeronautical Systems Division, Air Force System Command, Wright-Patterson Air Force Base, and under Contract DA-04-200-AMC-57(Z) supported by the U.S. Army Zeus Project Office, Redstone Arsenal, Huntsville, Alabama.

REFERENCES

1. Widrow, B., "Adaptive Sampled-Data Systems—A Statistical Theory of Adaption," 1959 WESCON Convention Record, Part 4.
2. ———, and M. E. Hoff, "Adaptive Switching Circuits," 1960 WESCON Convention Record, Part IV, Aug. 23, 1960, pp. 96–104.
3. ————————, "Adaptive Switching Circuits," Technical Report No. 1553–1, Stanford Electronics Laboratories, Stanford University, Stanford, Calif., June 30, 1960.
4. Ridgway, W. C., III, "An Adaptive Logic System with Generalizing Properties," Technical Report No. 1556–1, Stanford Electronics Laboratories, Stanford University, Stanford, Calif., April 1962.
5. Mays, C. H., "Adaptive Threshold Logic," Technical Report No. 1557–1, Stanford Electronics Laboratories, Stanford University, Stanford, Calif., April 1963.

6. ———, "Effects of Adaptation Parameters on Convergence Time and Tolerance for Adaptive Threshold Elements," submitted to *IEEE Trans. Electronic Computers.*
7. Hoff, M. E., Jr., "Learning Phenomena in Networks of Adaptive Switching Circuits," Technical Report No. 1554–1, Stanford Electronics Laboratories, Stanford, Calif., July 1962.
8. Pontryagin, L. S., V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko, "The Mathematical Theory of Optimal Processes," (translation), Interscience, New York 1962.
9. Smith, Fred B., Jr., of Minneapolis-Honeywell Regulator Company has independently done some similar work using a threshold logic element to implement nonlinear minimum time switching surfaces. This work was reported in "A Logical Net Mechanization for Time-Optimal Regulation," NASA Technical Note D–1678; December 1962.