STRUCTURAL LEARNING IN CONNECTIONIST SYSTEMS*

Andrew G. Barto and Charles W. Anderson
Department of Computer and Information Science
University of Massachusetts, Amherst MA 01003

Abstract

Although networks of neuron-like computing elements can be constructed to implement *any* function or operation one desires, it is a highly nontrivial problem to devise algorithms that permit networks to *learn reliably and efficiently* how to realize desired nonlinear functions without being provided with implementation details. In particular, learning algorithms that work for single layers of adaptive elements cannot be extended easily to multilayer or recurrently connected networks where structural changes must be produced. We describe an approach to this problem which uses stochastic search as does the Boltzmann learning procedure (Ackley, Hinton, and Sejnowski, 1985), but which is otherwise quite different from that method. We present several simulations of layered adaptive networks to illustrate our method. In addition, we briefly review a variety of previous approaches to this general problem in order to place these various methods in perspective and to suggest a range of alternatives with which the performance of novel methods should be compared.

INTRODUCTION

One goal of connectionist modelling as pursued by researchers in Cognitive Science and Artificial Intelligence is to bridge the gap between the behavior of networks of neuron-like computing elements and complex forms of behavior that appear at higher levels (Hinton and Anderson, 1981; Feldman, 1985). Learning is likely to play an important role in this research since it may be necessary in order to take advantage of the representational potential of connectionist networks (Hinton, 1984), and since these networks contain obvious parameters that can be adjusted through experience—the connection weights. However, although networks of neuron-like computing elements can be constructed to implement *any* function or operation one desires, it is a highly nontrivial problem to devise algorithms that permit networks to *learn reliably and efficiently* how to realize desired nonlinear functions without being provided with implementation details. In particular, learning algorithms that work for single layers of adaptive elements cannot be extended easily to multilayer or recurrently connected networks.

A significant advance in this area is the Boltzmann learning procedure recently described by Hinton and Sejnowski (1983) and Ackley, Hinton, and Sejnowski (1985) that is based on the analogy between thermodynamic systems and networks of neuron-like elements pointed out by Hopfield (1982). In this paper we describe an approach to this problem which uses stochastic search as does the Boltzmann procedure, but which is otherwise quite different from that method. We present several simulations of layered adaptive networks to illustrate our method. In addition, we briefly review a variety of previous approaches to this general problem in order to place these various methods in perspective and to suggest a range of alternatives with which the performance of novel methods should be compared. Most of these studies are quite old, but we think they are relevant given the renewed interest in networks of this kind.

1. PARAMETRIC VERSUS STRUCTURAL LEARNING

One of the problems with learning systems using the single-layer learning procedures is that learning proceeds up to a certain point and then stops. When the parameters that are adjusted by the learning algorithm—in a network, usually the connection weights—reach optimum values, the degrees of freedom of the system are exhausted even though the problem facing the system may be far from solved. Somehow, this *parametric* learning should be augmented with *structural* learning by which the roles of the parameters

*This research was supported by the Air Force Office of Scientific Research and the Avionics Laboratory (Air Force Wright Aeronautical Laboratories) through contract F33615-83-C-1078. We thank Rich Sutton, P. Anandan, John Moore, and Harry Klopf for their theoretical contributions and helpful discussions.

in determining behavior, and not just their values, are altered by the learning process. Since one can always regard structures as being parameterized, so that adjusting structures amounts to adjusting more parameters, this distinction is not completely straightforward. However, what we mean by structural learning generally involves a space of parameters that is so large, and a performance evaluation surface that is so complex, that the usual algorithms for parametric adaptation do not work. Structural learning is intimately related to the problem of adaptively developing new representations, for example, by the creation of "new terms," since it is the representation that determines the roles of the parameters.

One can view the adjustment of a connection weight in a complex network as a structural adjustment since it affects the roles of other weights in generating network behavior. A complex network will have very many adjustable weights, and the relationship between changes in a weight and changes in network performance (i.e., the gradient of the network performance index with respect to the weight) will be complicated by nonlinear dependencies on the weights of other elements—dependencies that do not make themselves known through information *locally* available to the connection in question. Additionally, even if this gradient could be determined locally, following it can lead to network performance that is only locally optimal. Global searches that do not suffer from this deficiency may be too slow for the large search spaces that arise in structural learning.

2. LAYERED AND RECURRENT NETWORKS

Our work to date has been restricted to the study of layered networks that do not have recurrently connected elements. The Boltzmann learning method, on the other hand, is restricted to symmetrically connected, hence totally recurrent, networks. In layered networks the entire stage corresponding to the running of a Boltzmann network or Harmony system (Smolensky, 1983) to "thermal equilibrium" using simulated annealing appears in a degenerate form: it is just the process of evaluating the input/output function realized by the network, and no iterative relaxation procedure is required. Hence, layered networks do not solve nontrivial constraint satisfaction problems. On the other hand, once a layered network has learned, its performance in computing this function is essentially instantaneous.

We have restricted attention to layered networks because it seemed to us that obtaining structural learning in this case would be easier than, and a prerequisite for, obtaining it in the recurrent case. Boltzmann learning shows that this is not true, but the principle employed there does not appear to extend to asymmetric networks. We have not yet decided on the best way to extend our approach to the recurrent case, but we do not think it is inherently limited to nonrecurrent networks. Future research will concern the case of recurrent but asymmetric networks. We have also been influenced by the extensive history of attempts to extend single-layer learning results to nonrecurrent networks having multiple layers, and we briefly review some of these studies in order to place our approach in its proper context.

3. REVIEW OF LAYERED NETWORK STUDIES

Assume that a multilayered network has been designed by defining the output of each element as a parameterized function of its input, and by specifying the interconnections among the elements. What values should be assigned to the parameters of the network in order to implement some desired input/output function? The most straightforward approach is to directly search the space of the network's parameters for those values that maximize some measure of the network's overall performance. By a direct search we mean one in which successive sets of parameter values (i.e., weights) are evaluated by seeing how the network performs with those values in its required task. Relevant gradient information is not obtained locally by the adaptive elements, and a centralized search control mechanism is required. Gilstrap (1971), who has pursued this direct-search approach, used guided random search methods that can be effective under conditions encountered with multilayered networks. Whatever the search method used, however, directly searching a space of dimension equal to the number of weights in an entire network is an extremely time-consuming process for all but the simplest cases.

Most results from early adaptive network research concern layered networks in which only the elements of the final (output) layer adapt. Examples of such networks are the Perceptron of Rosenblatt (1962) and networks of "adalines" (***ada**ptive **lin**ear **e**lements*) studied by Widrow (1962). In these cases the learning algorithms are variants of the now well-known error-correction procedures that adjust weights based on the discrepancy between a unit's response and its desired response supplied by some agency in the network's environment (a "teacher"). There are obvious difficulties in extending these error-correction techniques to layered networks. Adapting the terms of Hinton and Sejnowski (1983), let us distinguish a network's *visible elements* from its *hidden elements*. Visible elements are those whose activity is directly available to the network's environment and that are required to assume certain values for various input patterns provided to the network. Hidden elements are those whose activity is not directly visible and that are somehow to provide an encoding of input signals that will allow the visible elements to respond correctly. Although in many tasks it may be possible for the network's teacher to provide error signals to the network's visible elements (since it is these that define the network's response), it may not be possible for this teacher to provide analogous error signals to the hidden elements without *a priori* knowledge of the implementation details of the desired input/output function. If this knowledge were available, then the problem would be quite different from the ones in which we are interested: it would be more of a programming problem than a learning problem.

Some methods rely on the generation of new elements rather than on the adjustment of the parameters of existing elements. Methods that generate new elements generally divide the learning process into two stages. In the first stage, the weights of the first layer (i.e., the layer that directly receives the external input signals) are held constant while one of the familiar single-layer learning algorithms is used by the second layer. In the second stage, the second layer is held constant while new elements are added to the first layer. Those elements whose outputs are not significantly influencing the second layer might be discarded to limit the number of elements. Selfridge's Pandemonium provides an example of this two-stage learning process (Selfridge, 1959), where new elements are created by "mutated fission" and "conjugation" of existing elements. Although it is not described in network terms, the classifier system of Holland (1980) is probably the most highly developed example of generating new elements via this type of "genetic recombination" process. Uhr and Vossler (1961) presented another system that effectively adds new elements to the first layer. Each new element is set to respond to a subpattern of the current input pattern. Reilly, Cooper, and Elbaum (1982) recently proposed a somewhat related method that incorporates input patterns as "prototypes."

Methods for generating new elements are attempts to avoid the combinatorial explosion that would result from having an element for each of the possible combinations of available signals. The heuristic employed is that useful higher-order features will tend to be compositions of useful lower-order features. A technique using this heuristic may be viewed as a type of *beam search* (see Barr and Feigenbaum, 1981). The search is conducted by forming all pairwise (for example) combinations of lower-order features at each stage, and then removing from consideration all but a certain number of them before forming the next stage's combinations. At each stage, the number of features remaining is the *beam width* of the search. A beam search is not guaranteed to result in an optimal solution but can be efficient if the beam is sufficiently narrow. Although not usually associated with networks of adaptive elements, beam search can obviously be related to layered networks. Ivanhenko's (1971) Group Method of Data Handling, for example, can be regarded as a method for constructing a layered network using a beam search.

Another approach is to train the first layer of a two-layered network in isolation, independently of the second layer and of the network's performance on the required task. Such open-loop procedures are referred to as *clustering* methods. Clustering methods are based on the assumptions that the system's input patterns tend to fall into natural clusters due to their intrinsic structure, and that detecting these clusters is significant in some way for the performance of the system. Block, Nilsson, and Duda (1964) used a clustering algorithm to train the first layer of a two-layered network. Fukushima's Cognitron (1973) and Neocognitron (1980) are other examples of clustering algorithms implemented as networks. In these networks an element is selected for adjustment if its output is sufficiently in excess of the outputs of neighboring elements, and its weights are adjusted so as to cause it to be more vigorously activated by the current input pattern. This type of learning has also been discussed by Grossberg (1976a, 1976b) and Rumelhart and Zipser (1985).

For many types of problems, the need is not just to form clusters of input patterns but to form clusters that are *useful* in terms of the network's interaction with its environment. In order to accomplish this, the initial layers must use the information contained in an error or evaluation signal that is provided by the network's environment. The problem, as discussed above, is that this error or evaluation is directly a function only of the network's visible elements. Somehow this information must be used to tune the hidden elements. Rosenblatt (1962) reported experiments with a stochastic back-propagation scheme for generating error signals for interior elements, but the simplest way of doing this is to adjust a randomly chosen element when an error is made by an output element. This approach was analyzed by Alder (1975) who proved an extension of the Perceptron Convergence Theorem for layered networks. As he pointed out, however, the algorithm was "less than efficient."

Another method for selecting elements to adjust is to select those elements that would require the least amount of change to correct the network's error. Widrow used this method in networks consisting of two layers of adalines (1962). This algorithm and similar ones (e.g., Stafford, 1963) require a rather sophisticated agent to conduct the training of the interior elements. In some cases, the sophistication required by this agent can be reduced if the network structure is sufficiently constrained. Widrow's (1962) study of "madalines" (*m*ultiple *adalines*) can be interpreted in this way. Here the final layer implements a *fixed* logical function, and only the initial layer learns in a manner that depends on this logical function. Systems like this have been called "committee machines." Methods similar to this have been discussed recently by Reilly, Cooper, and Elbaum (1982), mentioned above, and Hampson and Kibler (1982) and seem to offer promising ways of using networks for nonlinear pattern classification.

Some of the aforementioned methods represent attempts to extend error-correction methods to all elements of the network, for example, by restricting network operation so that desired responses for interior elements might be deduced. Another approach is to use elements that do not require desired responses or error signals but that implement reinforcement-learning algorithms. Such elements are capable of improving performance with respect to an evaluation signal that assesses the *collective activity* of the network components. This method differs from what we called direct search since activity patterns rather than weight settings are directly evaluated and the elements locally estimate the relevant gradient information. Our own approach and that suggested by Klopf (1972, 1982) fall into this category, and we know of only a few earlier studies that are similar. In his Ph.D. thesis, Minsky (1954) described the SNARC (Stochastic Neural-Analog Reinforcement Calculator) which he constructed in 1951. It used components implementing a simple stochastic reinforcement-learning procedure. Farley and Clark (1954) experimented with stochastic adaptive elements that are similar in principle to the adaptive elements we have developed. In simulation experiments, recurrently connected networks of these elements were able to solve some simple discrimination tasks. Widrow, Gupta, and Maitra (1973) presented an extension of the adaline algorithm to allow it to do a form of reinforcement learning which they called "selective bootstrap adaptation." They remarked that this extension may permit the elements to learn as components of layered networks. We now describe the reinforcement-learning approach in more detail.

4. LAYERED NETWORKS OF REINFORCEMENT LEARNING ELEMENTS

Consider an adaptive network operating in an environment that can evaluate the behavior of the network, that is, of the collective behavior of the network's elements, but cannot specify the desired behavior of each individual component.[1] Suppose that the environment evaluates each of the network's overt actions by generating a reinforcement signal that is made available each element of the network.[2] If we view the problem from the perspective of an individual element embedded in the interior of this network, we can gain some understanding of the type of learning capability such an element might have to possess. Even if the environment deterministically evaluates the network's actions, the relationship between this element's actions and the evaluation signal will not be deterministic because it also depends on the behavior of other elements. In addition to this, the contingencies faced by the element will vary with time as the other elements adapt. Thus, even if the overall task faced by the network involves only fixed deterministic contingencies, the task faced by an individual element will involve nonstationary random contingencies.

If all the elements in the network are able to improve their individual levels of performance under these conditions, then the collection will also tend to improve its performance. This type of process involves cooperative behavior more closely related to that discussed in game theory and economics than it is to the cooperative phenomena of physics to which Boltzmann learning is related. One can regard the elements as self-interested agents and a network as a "team." This perspective on connectionist learning is due to Klopf

[1]By an agency in a network's environment, we do not necessarily mean an agency outside of the device in which the network resides; this agency may be another component of the overall learning system, such as a module specialized for delivering reinforcement to other modules.

[2]In the research reported here, we have not focussed on problems created by delayed evaluation. We have extensively studied these problems and results are reported elsewhere (Barto, Sutton, and Anderson, 1983; Sutton, 1984).

(1972, 1982), whose theory of the "hedonistic neuron" suggests that many aspects of learning, memory, and intelligence may arise from this type of cooperativity.

5. THE A_{R-P} LEARNING RULE

Together with R. Sutton, we have studied several types of adaptive elements capable of reinforcement learning (Anderson, 1982; Barto and Sutton, 1981; Barto, Sutton, and Anderson, 1983; Sutton, 1984), but the one used in the simulations described here was developed by Barto and Anandan (in press) who called its learning algorithm the *associative reward-penalty*, or A_{R-P}, *algorithm* and proved a convergence theorem. Details of this learning algorithm are provided in the appendix. Here there is only space to point out the following. An element implementing this algorithm is a linear threshold device with a randomly varying threshold. We use the logistic distribution function so that an element fires with probability $1/(1+e^{-s/T})$, where s is the total stimulus strength. Thus, the input/output behavior of the element is identical to that of the elements used in Boltzmann learning, where T is the "computational temperature." None of our results, however, require the use of this specific distribution function. After each action, the element receives a reinforcement signal taking values $+1$ and -1 to respectively indicate "reward" and "penalty." By updating its weights at each step (see the appendix) the element is able to improve its performance when its environment provides stimulus patterns and reinforcement feedback according to the following probabilistic scheme. At each step the environment presents the element with an input pattern $x \in X \subset \Re^n$ (where $\Re$ denotes the real numbers). For each pattern x in X and each of the element's actions, $y = 0$ or 1, the environment returns "reward" with probability $d(x,y)$ when the element emits action y in the presence of input pattern x. It delivers "penalty" with probability $1 - d(x,y)$. The element would maximize its probability of receiving reward if it responded to each x in X with the action y for which $d(x,y)$ is largest. Learning tasks like this one are related to instrumental, or cued operant, tasks used by animal learning theorists (where an input pattern x corresponds to a discriminative stimulus) and to "two-armed bandit" tasks studied by mathematicians and engineers (e.g., Cover and Hellman, 1970; Narendra and Thathachar, 1974)

Since it need not be true that $d(x,1) + d(x,0) = 1$, for a given input pattern x it might be true that no matter what action the unit produces, it usually receives reward (i.e., $d(x,1) > .5$ *and* $d(x,0) > .5$); or it might be true that no matter what action the element produces, it usually receives penalty (i.e., $d(x,1) < .5$ *and* $d(x,0) < .5$). Given these possibilities, the feedback received from producing one action provides *no information about the suitability of the other action.* This property makes this task significantly more difficult than the tasks usually solvable by neuron-like adaptive elements, yet it is an unavoidable feature of tasks faced by the hidden elements. Barto and Anandan (in press) prove that the A_{R-P} algorithm is asymptotically optimal[3] for arbitrary probabilistic contingencies if the set X of stimulus patterns is linearly independent. Interestingly, when the temperature T is zero, the A_{R-P} algorithm reduces to the Perceptron algorithm modified to accept reward/penalty feedback instead of training information in the form of desired responses. So restricted, however, optimal performance is obtainable only for deterministic environments ($d(x,y) = 0$ or 1, for all x and y), and such elements are not able to learn reliably when embedded in networks when a reinforcement-learning paradigm is used.

6. A MINIMAL CASE OF COOPERATIVE LEARNING

Fig. 1 shows a network of two A_{R-P} elements, e_1 and e_2. Only e_1 receives input patterns from the environment, and only the action of e_2 is available to the environment (e_1 is hidden; e_2 is visible). Suppose the network's output, the output of e_2, affects the probability of reward for both elements in a manner that depends on the stimulus pattern presented to e_1. If there were no means for e_1 to communicate with e_2, the elements would be capable of achieving only limited reward frequencies. The action of e_2 influences the reinforcement received by both elements, but in the absence of a communication link, e_2 remains blind to the discriminative stimulus x. On the other hand, in the absence of a communication link, e_1 can sense the discriminative stimulus but cannot influence the network's actions. The complementary specialties of the two elements have to be combined in order for each to attain optimal performance. If the weight connecting

[3]More precisely, it is ϵ-optimal for each $x \in X$, to use the terminology of learning automata theory (see Narendra and Thathachar, 1974).

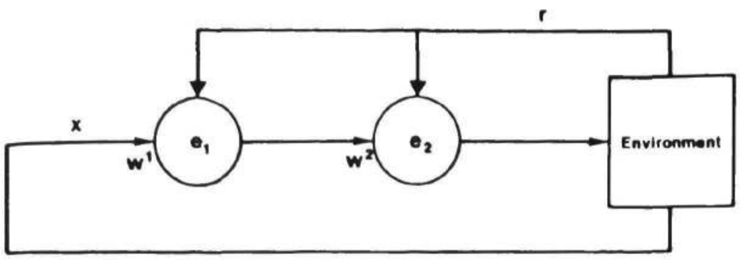

Figure 1: A two-element series in a simple discrimination task

e_1 to e_2 can be adjusted properly, the network can respond correctly. However, the correct value of the interconnecting weight depends on how e_1 has learned to respond to x. Conversely, the correct behavior of e_1 depends on the value of the interconnecting weight. Thus the two elements must adapt simultaneously in a tightly-coupled cooperative fashion.

To be more specific, we set up the simulation in the following way. A training step consists of presenting a randomly chosen input signal to the network, computing the network's output, determining the reinforcement signal, and then updating the weights.[4] The stimulus signals, $x = 0$ and $x = 1$, are equally likely to occur at each step, and the success probabilities implemented by the network's environment are given by the following table:

x	$d(x,0)$	$d(x,1)$
0	.9	.1
1	.1	.9

Thus it is optimal for the network as a whole to respond to $x = 0$ with visible action 0 to obtain reward with probability .9, and to respond to $x = 1$ with action 1 to obtain reward with probability .9, yielding an overall reward probability of .9. If the discrimination is not made so that the network responds identically to all input patterns, the overall success probability is $(.9 + .1)/2 = .5$. Since each element also adaptively adjusts its threshold (more precisely, the mean value of its threshold by adjusting a "threshold weight"), there are two ways the network can solve this problem—both elements can implement the identity map, or both can invert their input signal—and there are many ways it can fail.

Fig. 2a and 2b show the behavior of the network for a typical sequence of 500 training steps with $\lambda = .04$ and $\rho = 1.5$ (see the appendix). Fig. 2a shows the evolution of the behavior of e_1 in terms of two graphs. The first shows the conditional probability that e_1 fires ($y^1 = 1$) given that its input is 0, and the second shows the same thing for input 1. Both of these probabilities start at .5 since the weights are initially zero. Fig. 2b shows the evolution of the mapping implemented by e_1 and e_2 acting together by showing the probability that e_2 fires ($y^2 = 1$) for the different values of the network input x. Fig. 2c shows the evolution of the overall reward probability. Fig. 2d is a histogram of the number of steps required to reach a criterion of 98% of optimal performance for each of 100 sequences of trials. In all of the sequences the network reached this criterion before 1,500 steps. In about half of the sequences both elements learned to implement the identity map, and in the other half, both became inverters.

A series of two elements in a discrimination task provides one of the simplest examples we could devise to demonstrate cooperative reinforcement learning. We interpret the result as illustrating cooperativity in the literal game-theoretic sense, with the interconnecting link representing a "binding agreement" by which the elements form a coalition for mutual benefit.

[4]Note that in contrast to Boltzmann learning, the weights are updated after the each presentation of a single input pattern.

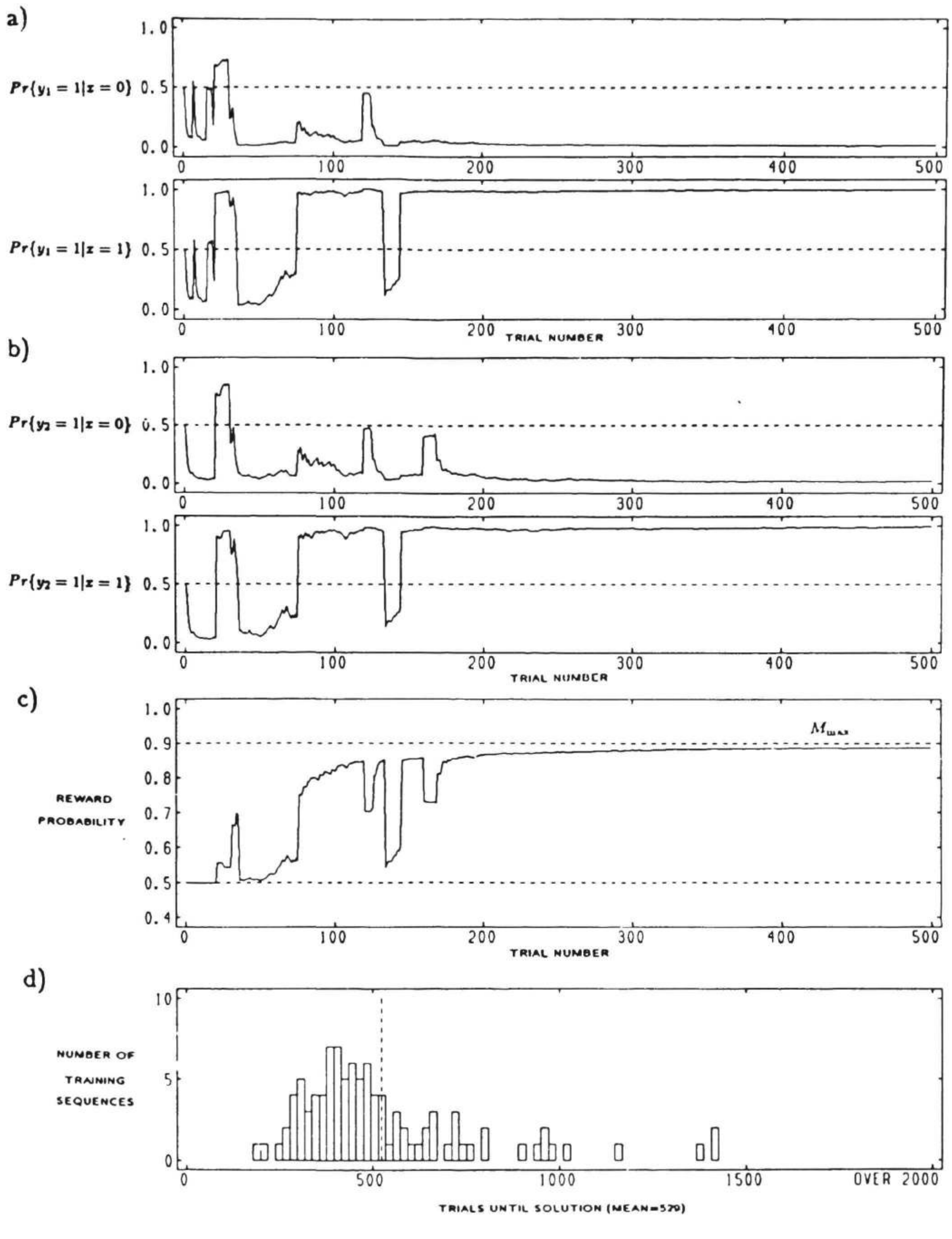

Figure 2: Simulation results for the two-element series. See text for explanation.

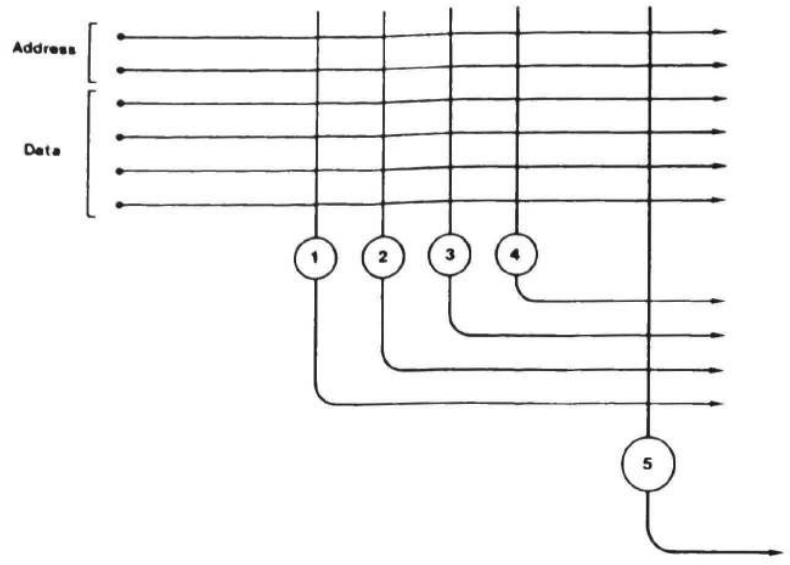

Figure 3: Network for the multiplexer task

7. A NONLINEAR TASK

In the task just described, cooperative learning is required only because the network lacks a direct pathway from input to output. The task itself is easily within the capabilities of a single element. Here we discuss a task that cannot be solved by a single linear threshold element, or any single-layer network of them. The network shown in Fig. 3 has six input components and a single principal output (from element 5). There are 39 weights to adjust, one associated with each of the pathway intersections and one threshold weight for each element. The reward contingencies implemented by the network's environment force the network to learn to realize a multiplexer circuit in order to obtain optimal performance. A multiplexer is a device with n address inputs and 2^n data inputs (here $n = 2$). Given a pattern over the address pathways, i.e., an address, a multiplexer's output is equal to whatever signal (0 or 1) appears on the data line associated with that address. It therefore routes signals from different input pathways to a single output pathway depending on the "context" provided by the pattern over the address pathways. For each of the 64 possible input patterns, we rewarded each element of the network with probability 1 if the visible element (number 5) produced the correct output, and we penalized each element with probability 1 otherwise. The input patterns were chosen randomly for presentation to the net. All of the elements implement the A_{R-P} algorithm with $T = .5$ except for the visible element (number 5) which uses $T = 0$ (and therefore essentially uses the Perceptron algorithm).

This is a highly nonlinear task since the natural generalizations over the set of input patterns tend to be wrong with respect to the required actions of the network. Consequently, it does not show the strengths of distributed representations (see Hinton, 1984), but it represents a rather stringent test of the learning method. The hidden elements (elements 1–4) are necessary in order to create new features to permit the visible element to respond correctly. Fig. 4 is a histogram of the number of steps required for the network to respond 99% correctly for 1,000 consecutive steps for each of 30 sequences of trials with $\rho = 1$ and $\lambda = .01$ (see the appendix). The average number of steps required was 133,149, or about 2,080 presentations of each stimulus pattern. In every sequence the network reached the criterion before 350,000 steps.

8. DISCUSSION

The multiplexer simulation suggests that layered networks of A_{R-P} elements are able to learn to implement associative mappings that are beyond the capabilities of individual elements. More importantly, they are able to do this when being directed by evaluative feedback that is based on knowledge of "what" the network as a whole should accomplish but no knowledge of "how" the network should accomplish it. Although we have not yet proved convergence for networks of A_{R-P} elements, all of our simulations suggest

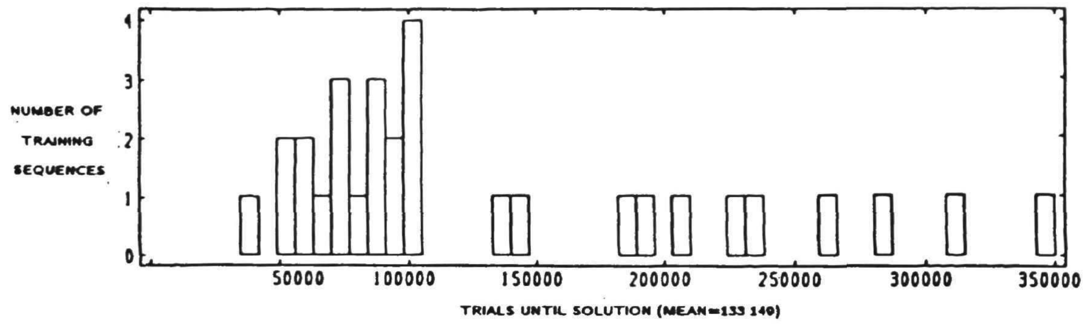

Figure 4: Simulation results for the multiplexer task. See text for explanation.

extremely reliable performance. However, these results also suggest that, like Boltzmann learning, the process may take a considerable amount of time. It is difficult to evaluate the learning rate of A_{R-P} networks without comparing their performance with that of other learning algorithms, and we are currently in the process of performing comparative simulation studies using some of the algorithms mentioned in Section 3. At present we only know that for relatively small networks the simplest direct search generally yields almost no improvement in performance by the time the A_{R-P} networks are performing near optimally. There are also a number of methods for accelerating convergence that have been developed for conventional pattern-classification algorithms with which we have not yet experimented.

Despite these important questions regarding learning rate, the learning method we have described has a number of attractive features. First, the training procedure does not require any elaborate or centralized control structures—it is a more or less "natural" consequence of the adaptive elements interacting with one another under contingencies that are simple to implement. Second, if their initial architectures permit, networks of A_{R-P} elements tend to learn the easy parts of a problem quickly so that performance tends to remain relatively high while the hard parts of the problem are being learned. Appropriate architectures are those in which hidden elements are not strictly interposed between layers but rather form auxiliary side networks. This is illustrated by the multiplexer network in which the input pathways to the network connect to the visible element as well as to the hidden elements. Finally, although it is not illustrated by the simulations described here, networks of A_{R-P} elements are able to learn in environments that cannot directly instruct even the visible elements but can only evaluate the consequences of their activity on some other process. As has been suggested elsewhere (Barto, Sutton, and Brouwer, 1981), this may be important for sensorimotor learning tasks where evaluative feedback is a function of the spatial result of a network's actions.

APPENDIX

THE A_{R-P} ALGORITHM

Assume that at the start of the t^{th} step, the environment provides an element implementing the A_{R-P} algorithm with a pattern vector $x(t) = (x_1(t), \ldots, x_n(t))$ of real numbers. The element then emits an action $y(t)$ that is determined by a random thresholding process:

$$y(t) = \begin{cases} +1, & \text{if } s(t) + \eta(t) > 0; \\ -1, & \text{otherwise;} \end{cases} \tag{1}$$

where $s(t) = \sum_{i=0}^{n} w_i(t)x_i(t)$ is the weighted sum of the input signals and the $\{\eta(t), t \geq 1\}$ are independent, identically distributed random variables (we used the logistic distribution with $T = .5$ in the simulations). Let $r(t)$ denote the reinforcement that evaluates the consequences of $y(t)$. It takes the values +1 and −1 to respectively indicate "reward" and "penalty." The weights, w_i, $1 \leq i \leq n$, are updated according to the following equation:

$$\Delta w_i(t) = \begin{cases} \rho[r(t)y(t) - E\{y(t)|s(t)\}]x_i(t) & \text{if } r(t) = +1; \\ \lambda\rho[r(t)y(t) - E\{y(t)|s(t)\}]x_i(t) & \text{if } r(t) = -1; \end{cases} \tag{2}$$

where $\Delta w_i(t) = w_i(t+1) - w_i(t)$, $\rho > 0$, and $0 < \lambda \leq 1$. $E\{y(t)|s(t)\}$, the expected value of the output given the weighted sum, depends on the distribution function used. For the logistic distribution function, it is

$$E\{y(t)|s(t)\} = \frac{e^{s(t)/T} - 1}{e^{s(t)/T} + 1},$$

which is a sigmoidally-shaped function of $s(t)$ with limits of -1 and $+1$ for $s(t)$ respectively approaching $-\infty$ and $+\infty$. In the simulations we recoded the values of $y(t)$ to be 1 and 0 instead of 1 and -1 when the elements communicated with one another.

If one lets the random variable $\eta(t)$ in (1) be identically zero (the deterministic "zero temperature" case) and interprets the term $r(t)y(t)$ in (2) as a training signal giving the element's desired response, then the A_{R-P} algorithm becomes an asymmetric form of the Perceptron algorithm. Additionally, if the input pattern $x(t)$ is held constant and non-zero over t, then the A_{R-P} algorithm reduces to a stochastic learning automaton algorithm (see Narendra and Thathachar, 1974); specifically it reduces to a nonlinear reward-penalty algorithm of the non-absorbing type as defined by Lakshmivarahan (1979). Barto and Anandan (in press) discuss the A_{R-P} algorithm is more detail and prove a convergence theorem.

REFERENCES

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. A learning algorithm for Boltzmann Machines. *Cognitive Science*, 1985, *9*, 147–169.

Alder, M. D. A convergence theorem for hierarchies of model neurones. *SIAM J. Comput.*, 1975, *4*, 192–201.

Anderson, C. W. Feature generation and selection by a layered network of reinforcement learning elements: Some initial experiments. COINS Technical Report 82-12, University of Massachusetts, Amherst, 1982.

Barr, A. & Feigenbaum, E. A. *The handbook of artificial intelligence, Volume 1.* Los Altos, California: Kauffman, 1981.

Barto, A. G. & Anandan, P. Pattern recognizing stochastic learning automata. *IEEE Trans. on Systems, Man, and Cybernetics*, in press.

Barto, A. G., & Sutton. R. S. Landmark learning: An illustration of associative search. *Biological Cybernetics*, 1981, *42*, 1–8.

Barto, A. G., Sutton, R. S., & Anderson, C. W. Neuronlike elements that can solve difficult learning control problems. *IEEE Trans. on Systems, Man, and Cybernetics*, 1983, *13*, 835–846.

Barto, A. G., Sutton, R. S., & Brouwer, P. S. Associative search network: A reinforcement learning associative memory. *Biological Cybernetics*, 1981, *40*, 201–211.

Block, H. D., Nilsson, N. J., & Duda, R. O. Determination and detection of features in patterns. In Tou, J. T. & Wilcox, R. H. (Eds.), *Computer and information sciences: Collected papers in learning, adaptation, and control in information systems.* Washington, D. C.: Spartan Books, 1964, 75–110.

Cover, T. M., & Hellman, M. E. The two-armed bandit problem with time-invariant finite memory. *IEEE Transactions on Information Theory*, 1970, *16*, 185–195.

Farley, B. G., & Clark, W. A. Simulation of self-organizing systems by digital computer. *I.R.E. Transactions on Inf. Theory*, 1954, *4*, 76–84.

Feldman, J. A. (Ed.) Special issue on connectionist models and their applications. *Cognitive Science*, 1985, *9*.

Fukushima, K. A model of associative memory in the brain. *Kybernetic*, 1973, *12*, 58–63.

Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 1980, *36*, 193–202.

Gilstrap, L. O., Jr. Keys to developing machines with high-level artificial intelligence. *Design Engineering Conference, ASME*, New York, 1971.

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

Grossberg, S. Adaptive pattern classification and universal recoding: II. Feedback, expectation, olfaction, illusions. *Biological Cybernetics*, 1976b, *23*, 187–202.

Hampson, S., & Kibler, D. A boolean complete neural model of adaptive behavior. Department of Information & Computer Science Technical Report No. 190, University of California, Irvine, CA, 1982.

Hinton, G. E. Distributed representations. Technical Report CMU-CS-84-157, Carnegie-Mellon University, Pittsburgh, PA, 1984.

Hinton, G. E., & Anderson, J. *Parallel models of associative memory.* Hilsdale, N. J.: Erlbaum, 1981.

Hinton, G. E., & Sejnowski, T. J. Analyzing cooperative computation. *Proceedings of the Fifth Annual Conference of the Cognitive Science Society,* Rochester N.Y., 1983.

Holland, J. H. Adaptive algorithms for discovering and using general patterns in growing knowledge-bases. *International Journal of Policy Analysis and Information Systems,* 1980, *4,* 217–240.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci.* USA, 1982, *79,* 2554–2558.

Ivanhenko, A. G. Polynomial theory of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics,* 1971, *1.*

Klopf, A. H. Brain function and adaptive systems—A heterostatic theory. Air Force Cambridge Research Laboratories Research Report, AFCRL-72-0164, Bedford, MA., 1972 (A summary appears in *Proceedings International Conference on Systems, Man, Cybernetics*). IEEE Systems, Man, and Cybernetics Society, 1974, Dallas, Texas.

Klopf, A. H. *The hedonistic neuron: A theory of memory, learning, and intelligence.* Washington, D.C.: Hemisphere, 1982.

Lakshmivarahan, S. ϵ-optimal learning algorithms—Non-absorbing barrier type, Technical Report EECS 7901, School of Electrical Engineering and Computer Sciences, University of Oklahoma, Norman Oklahoma, 1979.

Minsky, M. L. Theory of neural-analog reinforcement systems and its application to the brain-model problem. Princeton Univ. Ph.D. Dissertation. 1954.

Narendra, K. S., & Thathachar, M. A. L. Learning automata—a survey. *IEEE Transactions on Systems, Man, and Cybernetics,* 1974, *4,* 323–334.

Reilly, D. L., Cooper, L. N., & Elbaum, C. A neural model for category learning. *Biological Cybernetics,* 1982, *45,* 35–41.

Rosenblatt, F. *Principles of neurodynamics.* New York: Spartan Books, 1962.

Rumelhart, D. E., & Zipser, D. Feature discovery by competitive learning. *Cognitive Science,* 1985, *9,* 75–112.

Selfridge, O. G. Pandemonium: A paradigm for learning. *Proceedings of the Symposium on the Mechanisation of Thought Processes.* Teddington, England: National Physical Laboratory, H.M. Stationary Office, London, 2 vols., 1959.

Smolensky, P. Harmony theory: A mathematical framework for stochastic parallel processing. *Proc. Nat. Conf. on Artificial Intelligence AAAI-83,* Washington, DC, 1983.

Stafford, R. A. Multi-layer learning networks. In Garvey, J. E. (Ed.), *Symposium on self-organizing systems.* Office of Naval Research, 1963.

Sutton, R. S. Temporal aspects of credit assignment in reinforcement learning. Univ. of Massachusetts Ph.D. Dissertation, 1984.

Uhr, L., & Vossler, C. A pattern recognition program that generates, evaluates and adjusts its own operators. *Proc. Western Joint Comp. Conf.,* 1961, 555–569.

Widrow, B. Generalization and information storage in networks of adaline "neurons." In Yovits, M., Jacobi, G., & Goldstein, G. (Eds.), *Self-organizing systems.* Spartan Books, 1962.

Widrow, B., Gupta, N. K., & Maitra, S. Punish/reward: Learning with a critic in adaptive threshold systems. *IEEE Transactions on Systems, Man, and Cybernetics,* 1973, *5,* 455–465.