Human Neurobiol (1985) 4:229–256

Human Neurobiology

Learning by statistical cooperation of self-interested neuron-like computing elements

A. G. Barto

Department of Computer and Information Science, University of Massachusetts, Amherst, MA 01003, USA

Summary. Since the usual approaches to cooperative computation in networks of neuron-like computating elements do not assume that network components have any "preferences", they do not make substantive contact with game theoretic concepts, despite their use of some of the same terminology. In the approach presented here, however, each network component, or adaptive element, is a self-interested agent that prefers some inputs over others and "works" toward obtaining the most highly preferred inputs. Here we describe an adaptive element that is robust enough to learn to cooperate with other elements like itself in order to further its self-interests. It is argued that some of the long-standing problems concerning adaptation and learning by networks might be solvable by this form of cooperativity, and computer simulation experiments are described that show how networks of self-interested components that are sufficiently robust can solve rather difficult learning problems. We then place the approach in its proper historical and theoretical perspective through comparison with a number of related algorithms. A secondary aim of this article is to suggest that beyond what is explicitly illustrated here, there is a wealth of ideas from game theory and allied disciplines such as mathematical economics that can be of use in thinking about cooperative computation in both nervous systems and man-made systems.

Key words: Learning – Cooperativity – Adaptive neural networks – Statistical cooperation

Introduction

Cooperative and *competitive* are among the adjectives commonly used to describe the style of neural computation. They express the perception that subtle aspects of brain function are produced through the parallel activity of large numbers of communicating decision-making entities, variously identified with neural subsystems such as columns, modules, netlets, or neurons. One of the earliest models making this explicit is the reticular formation model of Kilmer et al. (1969) in which regions of neuropil are cast in the role of decision-making agents interacting to reach a "concensus" that directs the animal's action. Since then, many systems employing similar principles have been studied (e.g. Amari and Arbib 1977; Dev 1974; Didday 1976; Grossberg 1978, 1980; Julesz 1971; Marr and Poggio 1976). The general theme of "cooperative computation" has been elaborated in Amari and Arbib (1982) and is extensively discussed in Arbib's article in this issue.

There is also growing interest in cooperative computation, and other network approaches, in the allied fields of Cognitive Science and Artificial Intelligence (AI), where the term "connectionism" has been revived to refer to these approaches (see, for example, Feldman 1985; Hinton and Anderson 1981, for collections of relevant papers).[1] Not only have advances in microelectronics made the physical realization of brain-like hardware more of a possibility, but advances in our understanding of some of the problems involved in vision, motor control, and knowledge representation suggest that such hardware offers advantages over conventional computational architectures and may be necessary for real-time performance.

The sense of cooperation and competition most common in these studies is derived from social or ecological analogy: cooperative processes are ones that enhance each other's survival; competitive processes do the opposite. In neural terms, cooperation and competition are mediated, respectively, by excitatory and inhibitory interactions, with mutually excitatory sets of units forming "stable coalitions", and mutually inhibitory sets of units forming, for example, "winner-take-all" circuits, to use Feldman's (1982) terminology. Modelling formalisms tend to be derived from population dynamics of mathematical ecology: systems of nonlinear ordinary differential equations with cooperation and competition embodied in the form of the coupling functions. Analysis and computer simulation of these systems provide insight into the dynamics of the interacting network components. Other studies concern forms of cooperativity that are related to physical phenomena, such as ferromagnetism, in which the interaction of large numbers of units can be studied using statistical methods (e.g., Ackley et al. 1985; Allanson 1956, Amari 1974; Beurle 1956; Cragg and Temperley 1954; Geman and Geman 1983; Harth et al. 1970;

[1] There is much research within AI dealing with interacting parallel processes that does not fall under the connectionist label (e.g., Lesser and Corkill 1981). This research is distinguished from connectionism chiefly because the individual components are complex symbolic information processors rather than simpler neuron-like units. This approach has been called Distributed Artificial Intelligence and is a much less radical departure from conventional computation than is connectionism

Hopfield 1982; Hinton and Sejnowski 1983; Smolensky 1983; Wilson and Cowan 1972). Some of these studies are discussed in the Section "Relationship of the A_{R-P} element to other adaptive elements."

The research described in this article, on the other hand, focuses on learning rather than network dynamics, and the sense of cooperation and competition derives from the theory of games as formulated by von Neumann and Morgenstern (1953). A central tenet of game theory is that the players are "selfish" agents that attempt to maximize their individual utility or payoff. With this assumption, competition exists insofar as there are conflicts of interests among the agents, and cooperative interaction only occurs if it furthers the self-interests of the participants. Since the usual approaches to cooperative computation do not assume that network components have any "preferences", they do not make substantive contact with game theoretic concepts, despite their use of some of the same terminology. In our approach, however, each network component, or adaptive element, is a self-interested agent that prefers some inputs over others and "works" toward obtaining the most highly preferred inputs. This follows the basic idea of the theory of the "hedonistic neuron" due to Klopf (1972, 1982), which we discuss below, and allows us to illustrate network behavior that is cooperative in the game-theoretic sense. It is argued that some of the long-standing problems concerning adaptation and learning by networks might be solvable by this approach, and computer simulation experiments are described that show how networks of self-interested components can solve rather difficult learning problems.

A secondary aim of this article is to suggest that beyond what is explicitly illustrated here, there is a wealth of ideas from game theory and allied disciplines such as mathematical economics that can be of use in thinking about cooperative computation in both nervous systems and man-made systems. This is not a new suggestion, having been made in the Russian literature best known in the West through the work of Testlin (1973) and Varshavsky (1968, 1972), but here it has been overshadowed by other theoretical approaches. Current research in the West that continues this tradition is that of Klopf (1972, 1982), Crane (1978), and the "theory of learning automata" (reviewed by Narendra and Thathachar 1974), although the latter has largely remained an engineering discipline without making significant contact with theories of biological information processing. Related research in mathematical psychology is the statistical learning theory begun by Estes (1950) and Bush and Mosteller (1951a, 1955), but this work has not made significant contact with theoretical studies of neural networks. Similarly, more recent research involving economic analyses of animal behavior (e.g., Staddon 1983) appears not to have been applied at the more microscopic level of the behavior of neural subsystems. Additional relevant current research, also not being related to neural information processing, concerns the "selfish optimization" methods that are being studied by computer scientists for their applications in distributed computer systems. For example, methods for resource allocation in decentralized economies developed by mathematical economists are being adapted to manage the use of shared resources, such as communication channels, in networks of computers (Brooks 1983; Kurose et al. 1985; Yemini 1982; Yemini and Kleinrock 1979). These methods illustrate that analogies between societies of utility-maximizing individuals and networks of computing devices can yield useful algorithms. The research described in this article provides additional examples of the utility of this methaphor.

An essential feature of the mechanistic process our adaptive elements use for furthering their self-interests is random variation;[2] hence the term *statistical* cooperation. We borrow aspects of algorithms developed by engineers studying stochastic learning automata to provide adaptive elements with learning abilities robust enough for them to function effectively as adaptive network components. In these algorithms random variation is the source of behavioral variety from which effective solutions are selected according to their consequences in altering the elements' input. Each of our network components has an endogenous noise process without which cooperative phenomena emerge only in the most simple cases.

Despite the suggestions our research makes regarding the role of noise in nervous systems, and despite parallels between our research and theories of the neural basis of learning, the major source of constraint has been the problem-solving capabilities of our systems rather than faithfulness to current neurophysiological and neuroanatomical data. Our position is that it is appropriate to adopt this engineering methodology as long as the problems with which one is concerned are important and nontrivial to solve, and as long as one refrains from making unjustified claims about the validity of the resulting constructions as models of specific neural systems. Consequently, we do not strive to make our simulated networks conform as closely as possible to presumed neural constraints. For example, the adaptive units are basically linear threshold elements that are only crudely similar to neurons. We could simulate units that more closely resemble neurons, but the added complexity would obscure the theoretical issues we are addressing. It is these issues, which we believe are relevant to learning in both man-made and biological networks, rather than superficial resemblance to actual neural networks, that our research stresses.

The problem

We have focussed on the problem of obtaining directed learning by adaptive networks that are more than one layer deep. Although multilayered networks of linear threshold elements can be constructed to implement *any* input/output function, 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 easily be extended to multilayer networks. If elements more complex than linear threshold elements were to be studied, then one could obtain more complicated processing with fewer elements, but the problem of obtaining effective learning algorithms for networks would remain.

[2] The term random is not entirely satisfactory here since it is often taken to mean produced according to a uniform probability distribution. However, here it is taken to mean probabilistic but not necessarily uniform. Further, note that by mechanistic we do not necessarily mean deterministic

Understanding how learning can occur in complicated networks is not only important for what it would suggest about the operation of nervous systems, but it is also central to establishing the utility of adaptive networks for AI applications. One of the problems with learning machines 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.[3] Somehow, this *parametric* learning should be augmented with *structural* learning by which the roles of the parameters in determining behavior, and not just their values, are altered by the learning process. Since structures can always be regarded as being parameterized, so that adjusting structures amounts to adjusting more parameters, the distinction between these types of learning 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.

One can view the adjustment of a weight connecting two units in a complex network as a structural adjustment since it affects the role of other weights in generating network behavior. A complex network will have very many adjustible 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 complex due to nonlinear dependencies on the weights of other units – 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 – there may be other solutions, reachable only by coherent macro-mutations of sets of connection weights, that are much better. This is the classical "false-peak" problem: local searches that follow gradient information find only local performance peaks, and global searches that do not suffer from this deficiency are too slow for the large search spaces that arise in structural learning. A large number of different approaches to this problem have been studied, none of which provides a universally satisfactory solution. Minsky and Selfridge (1961) provide a now classic account of this and other problems, and a partial review of efforts to solve them can be found in Barto (1984).

These fundamental problems for learning systems can also be viewed as manifestations of the *credit-assignment problem* (Minsky 1961). This is the problem of correctly assigning credit or blame to each of the actions and internal decisions that contributed to the overall evaluation received. This problem can become exceedingly difficult either as overall evaluations become more infrequent, making it less clear which overt actions were responsible for changes in performance, or as the learning system becomes more complex, making it less clear which internal decisions were responsible. It is useful to divide the credit-assignment problem into two subproblems. One subproblem is to determine how the individual actions making up an action sequence should be credited for the evaluation generated by the entire sequence. The other subproblem is to use the evaluation credited to each step in order to assign credit to the internal processes of the learning system that determined the action selected on that step. Sutton (1984) calls these subproblems the *temporal* and the *structural* credit-assignment problems respectively. The cause of either type of credit-assignment problem is initial uncertainty about the causal microstructure of the interacting system and environment. Unless one is willing to assume the existence of sufficient a priori knowledge either built into the system or into an external teacher (which we are *not* willing to assume), this uncertainty is unavoidable, and mechanisms must be devised that reduce it. This article primarily concerns structural credit assignment. We have extensively studied temporal credit assignment and results are reported elsewhere (Barto 1984; Barto et al. 1983; Sutton 1984).

We believe that these difficulties in obtaining learning by adaptive networks arise due to fundamental properties of the problem and not of particular solution methods. Consequently, it would be very surprising if in some form they are not faced – and solved – by the adaptive mechanisms of real neural networks. For example, how are multisynaptic pathways established in the absence of a knowledgeable agency that can instruct individual neurons? Genetic specification plays a role, but it cannot account for the remarkable adaptability of neural networks in the face of unforeseen circumstances resulting from individual experience. It is highly doubtful, for example, that detailed instructive information is available for many types of sensorimotor learning tasks that animals routinely perform.

Selfish network components

Although similar ideas had been related to biological information processing by the research on the collective behavior of automata such as that described by Tsetlin (1973) and Varshavsky (1968, 1972), the idea of selfish neuron-like components for adaptive networks is due to Klopf (1972, 1982). In a 1972 monograph, Klopf presented the hypothesis that neurons are self-interested "hedonistic" agents that direct their firing activity so as to cause certain types of input patterns to appear on their input fibers and to prevent the occurrence of other types. In order to be successful, they must incorporate knowledge about the feedback loops in which they are embedded. Although this is a rather unorthodox hypothesis, there are several reasons to consider it seriously. First, it makes explicit the idea of the self-interested network component and thereby allows a host of novel ideas to be brought to bear on problems of neural organization. Second, the type of algorithm Klopf proposed by which the adaptive components pursue their goals has not been extensively studied. Whereas the network components typically studied theoretically, such as those based on Hebb's (1949) hypothesis of synaptic plasticity, can be

[3] One of the points of Minsky and Papert's book (1969) criticizing the perceptron, an early network using a single-layer learning rule (see Section "Relationship of the A_{R-P} element to other adaptive elements"), is that the degrees of freedom of single-layer learning procedures will always be exhausted before certain problems are solved that are quite easy to pose. In other words, it does not require pathological learning problems to thwart such systems

regarded as single-unit analogs of animal behavior in classical conditioning experiments, Klopf proposed that components ought to behave as analogs of instrumental conditioning behavior where responses are selected according to their consequences. As we shall see in the Section "Relationship of the A_{R-P} element to other adaptive elements", learning under these conditions requires algorithms different from those usually studied. And finally, as we hope our results show, Klopf makes valid suggestions about how some of the classical theoretical problems in obtaining learning in multilayered networks might be solved by using self-interested components.

What makes the idea of self-interested neurons unorthodox is that it is an inversion of the order by which certain high-level properties are usually seen to emerge from underlying neural machinery. According to Klopf's hypothesis, the sophisticated goal-directedness of animal behavior is not seen as emerging at a high level from non-goal-seeking components; rather, the cooperative activity of selfish, goal-seeking neurons gives rise to more sophisticated goal-directed behavior at higher levels. It is beyond the scope of this article to attempt a thorough discussion of the implications of this hypothesis – the questions raised are not simple to resolve. In fact, some of these questions have parallels with those raised by the theory popularized by Dawkins' book *The Selfish Gene* (1976), a theory that has generated considerable debate among evolutionary biologists. A sizable literature already exists about this debate, included in which is an extensive defense by Dawkins (1982), (see also Brandon and Burian 1984). Although evolution and learning in neural networks are definitely not different instances of a single process, the concepts of the selfish gene and the selfish neuron involve a similar inversion of orthodoxy.

Some questions about cooperativity via self-interested components, and about our theoretical work reported here, can be easily addressed to avoid unnecessary misunderstanding. For example, if goal-directed behavior is the result of the cooperative interaction of self-interested adaptive components, what determines the organizational grain at which such self-interested adaptive behavior first appears? In Klopf's neural theory, the neuron is chosen as a starting point because it is the obvious candidate, but the important point is that component self-interest, together with sufficient means for furthering it, first appears at a relatively fine structural grain. Our research is an attempt to study networks of units possessing *enough but not more* behavioral sophistication to allow them to *learn* to productively enter into cooperative relationships with other units like themselves. Whether or not neurons satisfy this criterion is an empirical question that our simulation experiments obviously do not address. Our research does suggest, however, that the adaptive capacity required for units to learn how to cooperate is significantly more sophisticated than that possessed by the adaptive elements considered in most theoretical studies of adaptive networks.

[4] See the discussion of this issue in Szentágothai and Arbib (1974, pp. 357–364. Harth et al. (1970) consider the case in which there is "randomness in-the-small" but "design in-the-large" by studying organized networks whose components are randomly connected netlets.

Another question our approach raises concerns the role of centralized mechanisms in both man-made intelligent systems and in brain function. Our view appears to be radically decentralized to the extent that there is no role for centralized controls of any kind. For example, our view appears to deny the existence of higher-level brain structures such as centralized reinforcement areas. Although we stress parallel distributed processing, it is not at all a consequence of our approach that every function is thoroughly distributed. Similarly, although the networks presented in this article learn to implement mappings from input signals to output signals without the intervention of any form of internal memory, it is by no means our intention to suggest that this is an adequate view of how intelligent behavior arises from underlying network dynamics. The degree of abstraction inherent in our approach should not be underestimated – in particular, one should not identify the boundary between one of our simulated networks and its environment with the boundary between an organism and its environment. We have simply not commented upon high-level organization and how our approach fits into an elaborate overall model of intelligent behavior because we are focussing on specific issues involved in decentralized computation.

Finally, although we occasionally find anthropomorphic language most expressive when describing the behavior of adaptive elements (and use the term "self-interested" throughout), we do not ascribe conscious intentionality to the elements. The adaptive elements are completely described in mechanistic terms (Section "A self-interested adaptive element") and can be regarded as purposive in the same way that regulators and servomechanisms can. One of our major points, however, is that mechanistic adaptive behavior can be much more robust than that exhibited by these classical negative feedback devices.

Stochastic networks

There are at least two distinct ways in which networks can be stochastic. In the first, the attributes of elements and their connection structure are randomly established, but the resulting networks operate according to deterministic rules. For example, connections between elements may be determined by a probabilistic rule based on the spatial distance between the elements (e.g., Beurle 1956; Uttley 1965). This *anatomical* randomness was assumed by many early neural-network modellers, and the term "random network" generally refers to this type of system. While there is reason to reject total specificity, the prevalence of this assumption in that early research may be attributable to mistaking complexity for randomness.[4] A major feature of the study of randomly connected networks has been the use of "macrostate" descriptions of network dynamics that ignore the behavior of individual units in the same way that a statistical mechanical description of a gas abstracts away from the individual molecules (e.g., Allanson 1956; Amari 1974; Harth et al. 1970; Wilson and Cowan 1972). In approaches to pattern classification using networks, anatomical randomness has been used to provide a set of sensory units with randomly chosen receptive fields in the hope that some of them would be sufficiently useful (Rosenblatt 1962).

The second kind of stochastic network is more *physiologically* stochastic because the computational units use some

kind of random process in their behavior, such as randomly varying thresholds. Thus, independently of whether or not the network's structure is specified randomly, its operation has a random component. Contrary to what one's intuition may say, this kind of randomness can be positively useful rather than a nuisance. There are man-made devices and algorithms that use noise, sometimes called "jitter" or "dither", for a variety of purposes: to counteract effects of quantization error in digital processes; to prevent a mechanism from remaining in an unstable equilibrium state; and, most importantly for us, to facilitate a search process by providing the variety that drives the search and possibly prevents convergence to false performance peaks. The networks to be described here are stochastic in this sense.

Although there have been many theoretical studies of the stochastic activity of single neurons (MacGregor and Lewis 1977; Moore et al. 1966, discuss several of them), and there have been studies of how to obtain reliable computation *despite* the presence of noise (von Neumann 1956), we know of relatively few approaches in which randomness is *purposefully* introduced into a network in order to facilitate computation. The recent studies of Ackley et al. (1985), Hopfield (1982), Hinton and Sejnowski (1983), and Smolensky (1983) fall into this category (see the Section "Relationship of the A_{R-P} element to other adaptive elements"). Harth's (1976) theory of visual perception uses randomness to provide variety in a search for appropriate feature-specific enhancement of visual input. In his Ph. D. thesis, Minsky (1954) described the SNARC (Stochastic Neural-Analog Reinforcement Calculator) which he constructed in 1951. It was an adaptive network of stochastic components that roughly corresponded to synapses rather than to entire neurons. Farley and Clark (1954) experimented with adaptive elements that are similar to the adaptive elements we have been studying, and they used the term "statistical cooperation" to describe the behavior of their networks. There are other studies relevant to biological information processing in which randomness is purposefully used but which do not explicitly deal with networks of neuron-like elements, notably the stochastic learning automata research referred to above and the genetic algorithms of Holland (1975).

A self-interested adaptive element

It is difficult to define precisely what a self-interested adaptive element is. Although we have attempted a careful characterization of this type of adaptive capability (Barto and Sutton 1981a), any definition raises innumerable questions that would take us far afield to address adequately. It is hoped that the following informal definition together with a concrete example will suffice. By a self-interested adaptive element we mean one that works toward causing its input to rank as highly as possible according to its own measure of preference. In order to do this the element must interact with its environment in a closed-loop manner so that its actions exert a causal influence on its input. Behavior is then selected according to its consequences in altering the preference of its input. The measure of preference need not be defined on separate sensory "snapshots" but can indicate the relative desirability of extended time sequences of sensory experiences. For example, the adaptive element to be described here receives a signal at each time step indicating "reward" or "penalty", and it learns to generate actions so as to maximize the probability of receiving the reward signal. It therefore works toward maximizing the frequency of reward over its "lifetime".

Although it is relatively easy to interpret the behavior of a variety of systems as self-interested in this sense (for example, a thermostat might be said to "prefer" sensing temperatures close to its set point), it is important to distinguish between such systems according to what they require of their environments in order to successfully further their interests. Some systems are successful only when interacting with very restricted types of environments (for example, a thermostat fails if its wires are crossed), whereas others can be successful in a wide range of environments. We say a self-interested system is more "robust" than another if it can further its interests when interacting with a wider range of environments.

Here we describe an adaptive element that is robust enough to learn to cooperate with other elements like itself. The learning algorithm it employs is one of many with which we have experimented and was first introduced by Barto and Anandan (1985), who called it the *associative reward-penalty*, or A_{R-P}, algorithm. We call the adaptive element the A_{R-P} element. In this section, we describe the element and present a simulation of single element; in the next section, we present examples of the cooperative behavior of these elements. After that, we describe the A_{R-P} element's learning capabilities in detail with respect to its behavior as a network component, and we place it in its proper historical and theoretical perspective through comparison with a number of related algorithms.

The A_{R-P} element

Figure 1 shows an adaptive element having $n + 1$ input pathways and one output pathway. The input pathways are of two kinds. Those labelled x_1 through x_n are "normal" pathways that carry input signals generated by other elements of the network or by the network's external environment; $x_i(t)$ denotes the magnitude of the signal on pathway x_i, $1 \leq i \leq n$, at time t. The remaining pathway is a specialized "reinforcement" pathway r; $r(t)$ denotes the value of its signal at time t. It is often convenient to allow signals on these various pathways to take on positive and negative values. Although the reinforcement pathway is shown as if it were an actual physical pathway, it is not necessary to take this view literally. It is better to regard it as a formally convenient way to indicate the element's preference for input patterns – patterns accompanied by a value of r indicating reward are preferred over those accompanied by a value indicating penalty.[5] To each input pathway x_i, $1 \leq i \leq n$, is associated a parameter, or weight, w_i, with value $w_i(t)$ at time t, that is adjusted by the learning algorithm. The pathway y carries the

[5] One could define a preference ordering directly over the patterns that can appear over the pathways $x_1, \ldots, x_n$; for example, one could let a pattern at time s be preferred over a pattern at time t if, say, $x_3(s) < x_3(t)$ and $x_8(s) > x_8(t)$. By using a specialized input to indicate ordering, we are studying certain aspects of all of these possibilities simultaneously, although here we restrict attention to just two preference classes

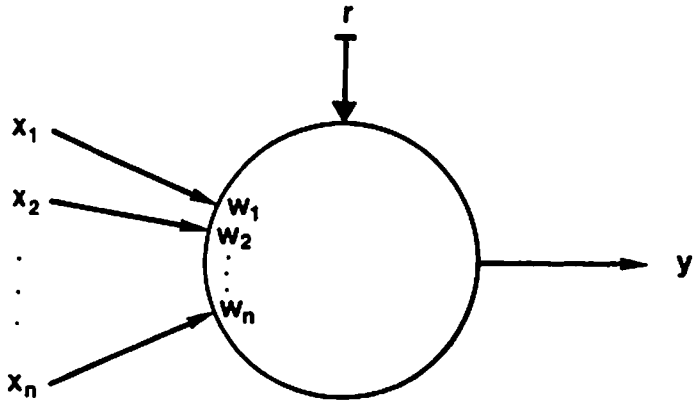

Fig. 1. A neuron-like adaptive element. Input pathways labelled x_1 through x_n carry non-reinforcing input signals, each of which has an associated weight w_i, $1 \leq i \leq n$; the pathway labelled r is a specialized input for delivering reinforcement; the unit's output pathway is labelled y

element's output signal, which is computed from the normal input signals and the weights according to a response mapping rule to be described; $y(t)$ denotes the magnitude of the output signal at time t. The A_{R-P} algorithm is easiest to express if $y(t)$ is either +1 or −1. When we discuss networks, $y(t)$ will be redefined to be either 0 or 1.

The A_{R-P} element operates in discrete time, meaning that $t = 1, 2, \ldots$, where each step is best regarded as a trial rather than a small time increment.[6] Assume that at the start of the t^{th} trial, the environment provides the element with a pattern vector $x(t) = (x_1(t), \ldots, x_n(t))$ of real numbers over its (normal) input pathways. The element then emits an action $y(t)$ that is determined from the weighted sum of the inputs by the following random thresholding process:

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

where $s(t) = \sum_{i=1}^{n} w_i(t)\, x_i(t)$ is the weighted sum of the (normal) input signals and each $\eta(t)$ is a random real number (more precisely, the $\{\eta(t), t \geq 1\}$ are idependent, identically distributed random variables). Let us assume for the moment that each random number is chosen according to a mean-zero normal distribution. According to this response mapping rule, the weighted sum $s(t)$ determines the probability of each response. For example, when $s(t) = 0$, then both output values are equally likely; whereas when $s(t) > 0$, then $y(t) = +1$ is the more likely response, and when $s(t) < 0$, then $y(t) = -1$ is the more likely response. The system becomes deterministic as $s(t)$ approaches $+\infty$ or $-\infty$ for each t. Changing the weights therefore only changes the probabilities of the output values. The random process $\eta(t)$ is an endogenous source of noise, purposefully included in the element, that can be regarded as noise in the membrane potential (to borrow the neural term). Alternatively, the random process $\{-\eta(t)\}$ can be viewed as a random threshold.

[6] We have not yet considered "real-time" versions of this algorithm in which events within trials are represented, although we have done so for other algorithms (Sutton and Barto 1981)

[7] Clearly a temporally refined version of this element would have to accomodate time delays between the action and the receipt of the relevant reinforcement signal

Upon receiving $y(t)$, the environment sends to the element a reinforcement signal $r(t)$ that takes the values +1 and −1 to respectively indicate "reward" (or "success") and "penalty" (or "failure") for producing output $y(t)$ in the presence of $x(t)$.[7] Upon receiving this signal, the A_{R-P} element updates its weights, w_i, $1 \leq i \leq n$, according to the following equation:

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

where $\Delta w_i(t) = w_i(t+1) - w_i(t)$, $\rho > 0$, and $0 \leq \lambda \leq 1$. $E\{y(t)|s(t)\}$, the expected value of the output given the weighted sum, is a specific deterministic function of $s(t)$ that is built into the element and depends on the distribution of the random variables. Here it suffices to regard it as an indication of how the unit *usually* responds to the current input pattern. Note that according to Eq. (2), the A_{R-P} element adjusts its weights based on four types of information: adopting the neural terms, it uses the presynaptic signal x_i, the postsynaptic signal y, the reinforcement signal r that indicates the consequences of the unit's activity, and a function of $s(t)$ that indicates what the element usually does for the given stimulus pattern.

The A_{R-P} algorithm is an embellishment of Thorndike's (1911) "Law of Effect":

> Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur; those which are accompanied or closely followed by discomfort to the animal will, other things being equal, have their connections with that situation weakened, so that, when it recurs, they will be less likely to occur. The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond. (p. 244)

To see how the A_{R-P} element accomplishes something similar to this, suppose for simplicity that the input signals are binary and that at trial t a certain set of input pathways become active (i.e., the signal on each pathway in the active set takes the value 1). Also suppose that the element produces output $y(t) = +1$, either totally by chance or partially as a result of a positive bias due to excitation by the active input pathways. Let us say that there is no excitation so that $E\{y(t)|s(t)\} = 0$. If the environment returns reinforcement $r(t) = +1$ (reward), the product $r(t)\,y(t)$ is +1, and Eq. (2) adds ρ to the weight of each active input pathway. Thus the output $y(t) = +1$ is "more firmly connected with the situation" indicated by the pattern of active input signals. When that pattern, or a similar pattern, recurs, the probability of producing the output +1 will be greater. On the other hand, if the environment returns reinforcement $r(t) = -1$ (penalty), then $r(t)\,y(t) = -1$, and Eq. (2) substracts $\lambda\rho$ from the weight of each active pathway, thus weakening the connection. The case in which the element's output is $y(t) = -1$ can be analysed similarly.

If one assumes that whenever an action is followed by a penalty signal, then the *other* action would have most likely produced reward, it is reasonable to regard the product $r(t)\,y(t)$ as an estimate of what the output *should* have been (since $-1 \times 1 = -1$ and $-1 \times -1 = 1$). We therefore think of $r(t)\,y(t)$ as an estimate of the *desired response* to the input pattern present on trial t. Unfortunately, one cannot

always assume that the other action would have most likely produced reward, and this terminology must not be taken too literally. A major point, to be elaborated in the Section "Theoretical analysis", involves the subtle but critical difficulties this creates for learning.

These difficulties require the learning rule to have two features to ensure its proper functioning: (1) asymmetry with respect to reward and penalty, and (2) dependence of the size of the weight change, and hence of the change in action probabilities, on what the element "usually does" in a situation. The amount of asymmetry depends on the parameter λ: as λ approaches 0, learning upon penalty plays a decreasing role in the process. In our simulations we use a small but nonzero value for λ (e.g., 0.01). Although this makes the rule a better model of animal learning (and brings it into better agreement with asymmetric versions of the Law of Effect), our reasons for introducing this asymmetry are purely functional and will be explained later. The use of the term $E\{y(t)|s(t)\}$ makes the learning process converge properly. It causes the weights to change according to the discrepancy between an estimate of what the element should have done in the presence of stimulus pattern $x(t)$ (i.e., $r(t)$ $y(t)$) and what it usually does in the presence of $x(t)$. The magnitude of weight modification decreases as the usual response approaches the estimated desired response, and it causes a relatively larger weight change when the desired response is not a very likely output of the element. Additional explanation of these features of the algorithm is presented in the Section "Theoretical analysis" where the A_{R-P} element is discussed more theoretically.

According to Eq. (1), the A_{R-P} element uses a fixed threshold of zero. Alternatively, if one adopts the view in which the threshold varies randomly, then the threshold has a fixed mean value. There is a standard way to allow the threshold to vary, or to let the mean of the random threshold vary, as a result of a learning process. Let the input signal on one of the normal input pathways always equal a positive constant; for example, let $x_1(t) = 1$ for all t. Since the contribution to the weighted sum, $s(t)$, from this input is $w_1(t)$, we can regard the unit's threshold on trial t to be $-w_1(t)$. Alternatively, if we regard the element as having a random threshold, its mean value is $-\mu - w_1(t)$, where μ is the mean of the random variable $\eta(t)$. By altering w_1, known as the threshold weight, according to the rule used to alter the other weights, the threshold is effectively varied in such a way that a wider class of transfer functions can be learned.[8] All the adaptive elements in the networks we simulate have a constant input value of 1 on pathway x_1. Of course one need not regard this input pathway as being literally present, and we do not show it in the figures.

[8] A linear threshold unit with n weights and a fixed threshold of zero divides the space of all possible input patterns, $\mathcal{R}^n$, into two regions by means of an $(n-1)$-dimensional hyperplane that passes through the origin. If the threshold varies, the hyperplane need not pass through the origin. See Nilsson (1965) or Duda and Hart (1973)

[9] In these tasks the actions of the learning system only affect the probability of reward; they do not have any influence on the choice of input patterns x. If this latter type of contingency were present, the tasks would be more complex control problems and would present difficulties beyond the scope of the present article. See Barto et al. (1983) or Sutton (1984)

Simulation of a single A_{R-P} element

In order to illustrate the learning behavior of an A_{R-P} element, we present simulation results showing how a single A_{R-P} element is able to learn when the environment provides stimulus patterns and reinforcement feedback according to the following probabilistic scheme. Let X be the set of all stimulus patterns to be used in training. Each pattern $x \in X$ is an n dimensional vector of real numbers. Suppose that for each trial t the environment selects a stimulus pattern $x(t) = x \in X$ with probability ξ^x and presents it to the A_{R-P} element via its n normal input pathways. For each pattern x in X and each action y, the environment returns reward ($r(t) = +1$) with probability $d(x, y)$ when the A_{R-P} element emits action y in the presence of input pattern x. Penalty ($r(t) = -1$) is delivered 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 that corresponds to the largest reward probability. More precisely, for each pattern x, let y^x be the action such that $d(x, y^x) = \max\{d(x, +1), d(x, -1)\}$; reward probability is maximized when for all x in X the element produces output $y(t) = y^x$ in the presence of pattern $x(t) = x$ with probability one. Learning tasks like this one are related to instrumental, or cued operant, tasks used by animal learning theorists, and the stimulus patterns in the set X correspond to discriminative stimuli. However, since we have not yet thoroughly explored the A_{R-P} algorithm as a model of animal behavior in instrumental conditioning experiments, we prefer to use the terminology of Barto and Anandan (1985) and call these tasks *associative reinforcement learning tasks*.[9]

In order to show the progress of the learning process we use as a measure of performance the probability that the element will receive reward on the average trial given its current set of weights. This value, which we denote M_t when computed based on the weight values at trial t, additionally depends on the probability that each input pattern will occur on a trial and the reward probabilities that characterize the environmental contingencies. In particular, let $p_t^{+1|x} = Pr\{y(t) = +1 | x(t) = x\}$ and $p_t^{-1|x} = Pr\{y(t) = -1 | x(t) = x\} = 1 - p_k^{+1|x}$. These are the action probabilities for trial t conditional on the presence of input pattern x. Although it is not explicit in this notation, these probabilities are functions of the weight values on trial t and the distribution function of the random number used in generating the action [Eq. (1)]. Given all this, we can define our performance measure:

$$M_t = \sum_{x \in X} \xi^x \left[Pr\{r(t) = +1 | x(t) = x\} \right]$$

$$= \sum_{x \in X} \xi^x \left[d(x, +1) p_k^{+1|x} + d(x, -1) p_k^{-1|x} \right].$$

This measure is maximized when the optimal action for each input pattern occurs with probability 1, in which case it is

$$M_{\max} = \sum_{x \in X} \xi^x \max\{d(x, +1), d(x, -1)\}.$$

The distribution function of the threshold noise used in all the simulations described in this article is the logistic distribution, which we denote Ψ, given by

$$\Psi(s) = \frac{1}{1 + e^{-s/T}}, \qquad (3)$$

where T is a parameter. This is a sigmoidal function that is

similar to a normal distribution function but is easier to evaluate. It is also used in the studies of statistical cooperativity of Ackley et al. (1985), Hinton and Sejnowski (1983), and Smolensky (1983), where T is the "computational temperature" of the system. As T approaches zero, the function given by Eq. (3) approaches the step function with a discontinuity at $s = 0$, which means that the element becomes more deterministic.

Given this distribution function, the term $E\{y(t)|s(t)\}$ in Eq. (2) becomes a specific function of $s(t)$. In particular, since $\Psi(s) = Pr\{\eta(t) \leq s\}$, we have that $p_t^{-|x} = Pr\{s(t) + \eta(t) \leq 0\} = \Psi(-s(t))$, and $p_t^{+|x} = 1 - \Psi(-s(t))$. Therefore we have that

$$E\{y(t)|s(t)\} = -1 \cdot p_t^{-|x} + 1 \cdot p_t^{+|x}$$
$$= 1 - 2\Psi(-s(t))$$
$$= \frac{e^{s(t)/T} - 1}{e^{s(t)/T} + 1}.$$

This is plotted as a function of $s(t)$ for $T = 1, 0.5$, and 0.25 in Figure 2. In all simulations presented in this article, we set $T = 0.5$, so that this function has a slope of 1 at the origin.

The task simulated here is an analog of a conditioned inhibition procedure for instrumental conditioning. There are two stimulus components: the presence of one component alone signals the contingency in which action +1 is optimal, and the presence of both components together signals the contingency in which action −1 is optimal. One would expect that the first stimulus component would become excitatory and that the second would become sufficiently inhibitory to counteract the excitation of the first. For this task, then, the stimulus patterns are the vectors $x^{(1)} = (1,0)$ and $x^{(2)} = (1,1)$, corresponding respectively to the cases in which only the first stimulus component is presented and both stimulus components are presented. These vectors are equally likely to occur at each trial ($\xi^1 = \xi^2 = 0.5$). Since the first stimulus component is 1 for both patterns, we could regard the weight associated with it as a threshold weight as described above. Both weights are set to zero at the start of each sequence of trials, which makes the actions initially equiprobable for both stimulus patterns.

The reward probabilities implemented by the element's environment are given by the following table:

x	$d(x,-1)$	$d(x,+1)$
(1,0)	0.6	0.9
(1,1)	0.4	0.2

Table entry $d(x, y)$ is the reward probability given that the element produces action y when receiving x as input. Thus it is optimal for the learning system to respond to (1,0) with action +1 to obtain reward with probability 0.9, and to respond to (1,1) with action −1 to obtain reward with proability 0.4. Therefore, in this task $M_{max} = (0.9 + 0.4)/2 = 0.65$, and the initial overall reward probability is $(0.6 + 0.9 + 0.4 + 0.2)/4 = 0.525$.

Figure 3a shows results of simulating an A_{R-P} element in this task with $\rho = 0.5$ and three different values of λ: 0.01, 0.05, and 0.25. Plotted for each trial t is the average of M_t over 100 runs, where a run is a sequence of 5000 trials, each starting with weight values set to zero and using different random numbers. The dashed lines show theoretical asymptotic performance levels for the three values of λ. Exactly how performance varies with the parameters will be treated in the Section "Theoretical analysis". Here just note that this asymptote approaches the optimal performance level 0.65 as λ decreases and that the learning rate decreases as λ decreases. The average final weight vectors for $\lambda = 0.01$, 0.05, and 0.25 are respectively (2.99, −4.05), (2.73, −3.08), (1.91, −1.71), meaning that in each case the first stimulus component becomes excitatory and the second becomes inhibitory. Figure 3b shows the performance of the A_{R-P} element with $\lambda = 0.05$ in a single run. All the individual runs that we have observed show similar behavior.

From this simulation it is clear that it takes the A_{R-P} element a considerable number of trials to approach asymptotic performance levels. However, tasks like this involve several subleties that can make them quite difficult. The first type of sublety would be present even if the element were not required to discriminate between input patterns. Consequently, let us focus only on the learning problem for a given fixed input vector x. Note that the reward probabilities for producing the two actions in the presence of x need not sum to one: that is, it need not be true that $d(x, +1) + d(x, -1) = 1$. This means that for a given input x, it might be true that no matter what action the unit produces, it usually receives reward (i.e., $d(x, +1) > 0.5$ *and* $d(x, -1) > 0.5$, as for pattern $x^{(1)}$ in the simulation); or it might be true that no

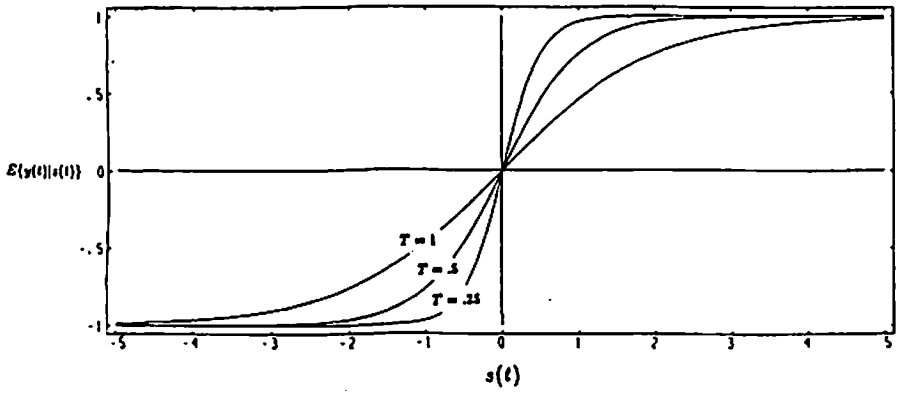

Fig. 2. Plots of $E\{y(t)|s(t)\}$. When the noise distribution function Ψ is the logistic distribution function, $E\{y(t)|s(t)\}$ as a function of $s(t)$ is sigmoidal. Here it is plotted for $T = 1$, 0.5, and 0.25. The curve approaches the discontinuous step function as T approaches zero. We use $T = 0.5$ in all of the simulations reported here (except where otherwise noted)

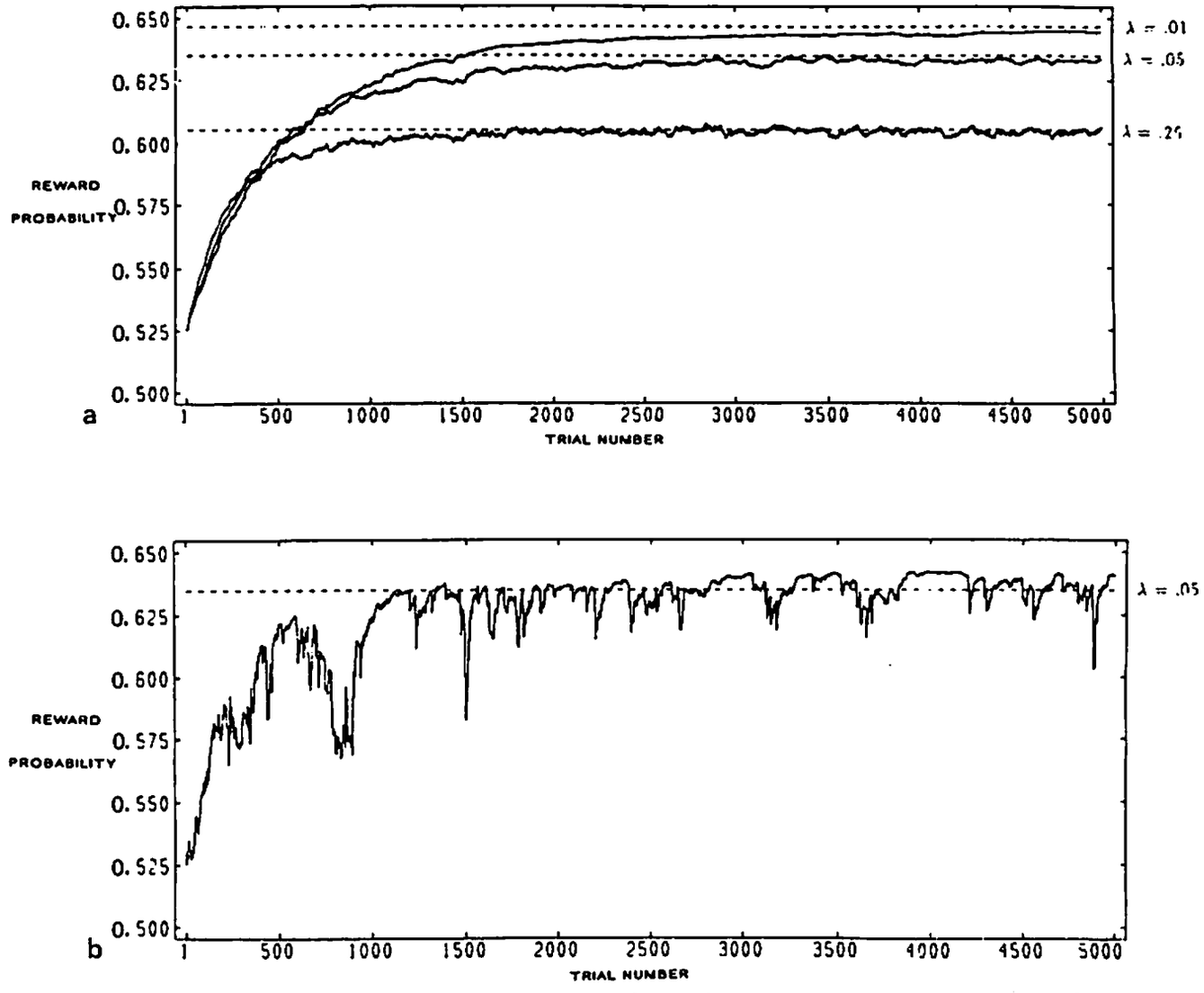

Fig. 3a and b. Simulation results for a single A_{R-P} element. a Average of the performance measure M_t over 100 sequences of 5000 trials with $\rho = 0.5$ and $\lambda = 0.01$, 0.05, and 0.25. The dashed lines show the theoretical asymptotic values given the three values of λ. As λ decreases the asymptote approaches perfect performance (0.65) and the learning rate decreases. b Performance of an A_{R-P} element with $\rho = 0.5$ and $\lambda = 0.05$ for a single sequence of 5000 trials

matter what action the unit produces, it usually receives penalty (i.e., $d(x, +1) < 0.5$ *and* $d(x, -1) < 0.5$, as for pattern $x^{(2)}$ in the simulation). Given these possibilities, the feedback received from producing one action provides *no information about the suitability of the other action.* This implies that it is not possible to solve this type of problem by any procedure that resembles the following: pick any one action and perform it sufficiently often in the presence of x to obtain a good estimate of the probability of reward associated with it; if the estimated reward probability is greater than 0.5, then that action is the right one, otherwise the other action is the right one. No matter how good (or how bad) an action appears, the other action might be better (or worse). Consequently, in the presence of each input pattern, an algorithm must produce *both* actions sufficiently often in order to figure out which one is better.

But what does it mean to perform each action *sufficiently often*? Since one can never obtain a perfectly accurate estimate of a probability by means of any finite number of observations, each action, even the inferior one, must be performed infinitely often. The following algorithm, for example, always leaves a chance of deciding on the wrong action: for a given input pattern, perform each action N times, where N is as large as desired but finite, then choose the action that yielded the largest proportion of rewards. Obviously, no matter how large N is, the observed reward frequency may be different from the actual reward probability. But on the other hand one need not demand complete certainty in making the decision, and hence one can simply take N as large as is required to make the decision with a desired confidence level. This strategy, however, raises the question of how the trials are to be allocated among the actions. Does one perform one action N times in succession before trying the other action; or would it be better to continually intermix the trials of the actions so that there is little chance of performing the worst action for long periods of time? What is the best way of doing this?

This last dilemma is fundamental to adaptation and

learning. There is a tradeoff between using knowledge already acquired in order to perform well and the necessity to acquire more knowledge. In the decision problems being considered, in order to improve the accuracy of the reward estimate for the action that *appears* to be worst, the system must temporarily abandon performing the action that *appears* to be best, thereby probably sacrificing performance in order to keep open the possibility of attaining optimal performance in the limit. Systems need to maintain a balance between these requirements, and what kind of balance is appropriate depends on how stationary the environment is. If the environment is changing rapidly, for example, then a bias toward using current knowledge is appropriate since careful knowledge-gathering may take too long to be useful.[10] Cover and Hellman (1970) and Robbins (1952) discuss this tradeoff, and we were first made aware of it through the work of Holland (1975).

These questions and others have made decision problems under this kind of uncertainty a topic of considerable mathematical research. Several distinct theoretical traditions have developed around these questions, and there are several approaches to designing algorithms. The theory on which the A_{R-P} algorithm is based is that of learning automata which originated with the Russian research (e.g., Tsetlin 1973; Varshavsky 1968, 1972) and has been extensively developed since then (Narendra and Thathachar 1974). An independent but similar line of research was conducted by mathematical psychologists in the 1950s and 1960s (e.g., Atkinson et al. 1965; Bush and Estes 1959; Bush and Mosteller 1955). Another tradition concerns sequential decision problems and the "two-armed bandit problem" (e.g., Cover and Hellman 1970; Cover 1968; Robbins 1952). These problems have been studied under a variety of assumptions such as finite memory and finite time. Despite this well-developed theory, we are not aware of algorithms that combine these ideas with the discrimination abilities of linear threshold elements in the way the A_{R-P} algorithm does. Neuron-like adaptive elements studied in the past are not able to learn effectively if the environment imposes contingencies like those in the simulation.

In addition to these problems arising from the nondeterministic nature of the environmental feedback, the task in the simulation also illustrates problems involving the discriminability of stimulus patterns. At one extreme, all the stimulus patterns are identical and no discrimination is possible. Here the task becomes one in which the system must adapt to a changing environment with no clue as to when the reward probabilities change. Performance is likely to be poor since the system cannot respond selectively to different situations. At the other extreme, all the stimulus patterns are totally dissimilar so that any action can be associated with any stimulus pattern. In this case, any necessary discriminations can be made but no transfer of training can occur from one stimulus to another. Consequently, learning has to occur separately for each stimulus pattern and is therefore likely to be relatively slow.

The more interesting case in which similar stimulus patterns must be discriminated falls between these extremes. Suppose two input patterns are similar by virtue of their sharing a subset of active stimulus components, but that the optimal actions for each are different. This is the case illustrated in the simulated task. How can generalization between the patterns due to their similarity be overcome? This question was of great concern to psychologists studying "stimulus sampling theory" (Atkinson and Estes 1963), and several methods were proposed to address it (Bush and Mosteller 1951b; Restle 1955). The A_{R-P} element, however, uses a method developed independently by engineers and computer scientists who were studying pattern classification. The A_{R-P} element forms a *linear discriminant function*, determined by the weight values, that can discriminate between any two different but arbitrarily similar patterns (where pattern similarity is determined by the vector dot product). However, being forced to discriminate between two patterns obviously has implications on the decisions that are possible with respect to other patterns. Pattern classification theory therefore focusses on how decision rules partition the space of all possible stimulus patterns. The relation between stimulus sampling theory and pattern classification is discussed further by Barto (1984) and Sutton (1984), and good introductions to the theory of pattern classification can be found in Duda and Hart (1973) and Nilsson (1965).

This method of discriminating stimulus patterns can also be related to models of classical conditioning, such as that of Rescorla and Wagner (1972), in which the total associative strength of a stimulus is represented as linear combination of the associative strengths of its component stimuli. Sutton and Barto (1981) point out, for example, that the Rescorla/Wagner model of classical conditioning is essentially identical to a pattern-classification method developed earlier by Widrow and Hoff (1960). The A_{R-P} element is very closely related to both of these systems as is detailed in the Section "Relationship of the A_{R-P} element to other adaptive elements". This implies that the A_{R-P} element is capable of producing analogs of all of the stimulus context effects that are accounted for by the Rescorla/Wagner model – but the paradigm resembles the instrumental, or cued operant, paradigm rather than classical conditioning. An analog of conditioned inhibition is illustrated by the simulation just described, and analogs of overshadowing and blocking are also produced by the A_{R-P} element. Behavior not included in the Rescorla/Wagner model's repertoire, such as latent inhibition, is also not produced by the A_{R-P} element.

The neuron-like elements that are usually studied by theorists are not designed to learn in tasks like the associative reinforcement learning tasks illustrated here. In some cases, adaptive elements can be modified to learn under reward/penalty feedback, but the resulting elements are not able to learn effectively for arbitrary reward contingencies. In the next section, we discuss learning in networks and argue that it is just this ability that allows the A_{R-P} element to reliably learn to cooperate with other elements in a network.

Networks of A_{R-P} elements

The most commonly studied neuron-like adaptive elements

[10] These differing requirements for differing degrees of environmental stability are related to ecologists' concepts of K-selection and r-selection. K-selection favors characteristics suitable for slowly-varying and predictable environments, such as large size, long life, and few carefully nurtured offspring; r-selection favors characteristics suitable for unstable environments, such as the ability to reproduce rapidly. See Dawkins (1982)

are capable of what engineers and computer scientists call "learning with a teacher" or "supervised learning" (e.g., Duda and Hart 1973). In this paradigm, the element is provided with a signal that directly specifies what its response should be for each stimulus pattern in a training set. The element adjusts weights so that its output signals match these training signals. This paradigm's major interest to theorists lies in the fact that since the resulting response rule applies to patterns not presented during training, it produces a form of generalization. This paradigm, which we discuss in more detail in the Section "Relationship of the A_{R-P} element to other adaptive elements", is closely related to the classical conditioning paradigm and does not involve all of the subleties of the associative reinforcement learning task.

Adaptive elements designed for learning in this paradigm reveal critical limitations when they are used as components of adaptive networks. In order to train such a network, each component element must be provided with its own individualized training signal. This means that there must be a "teacher" that knows enough about what every component must do that it can furnish each component with desired responses for a sufficiently varied training sequence. Consequently, although networks of such elements can be trained to implement *any* associative mapping, the details of the implementation must be worked out beforehand. The generalization abilities of the elements can still make this worthwhile (e.g., Hinton 1981), but the process seems better described as a form of programming rather than as a form of learning – some agency needs to know from the start "how" the desired mapping is to be implemented by the network. However, the type of network learning that is of interest here, and which would be much more useful, occurs when some agency just knows "what" constitutes the desired behavior of the network, and the network, as it were, figures out for itself how to accomplish it.

It is important to contrast this objective with what has been called "unsupervised learning", "learning without a teacher", or "self-learning" (e.g., Duda and Hart 1973). Here, the learning process extracts structure that is inherent in the input stream rather than forming associations in a manner directed or constrained by an outside agency. It constructs "clusters" of patterns, where patterns in the same cluster are more similar to one another than to patterns in other clusters according to a built-in similarity measure. Unfortunately, the label "unsupervised" incorrectly suggests that unsupervised learning is more difficult or more powerful than is supervised learning. It suggests that the learning system is able to solve, without supervision, the same class of problems a supervised learning system requires supervision to solve. However, the kind of clustering an unsupervised learning system does is not directed toward the satisfaction of any constraint except that imposed by the built-in measure of similarity and perhaps a built-in specification of the number of clusters. A supervised system is in fact *more* adaptive than is an unsupervised system because it forms clusters in order to solve problems posed to it by environmental contingencies rather than to solve a problem of its own.[11] Unsupervised learning is more accurately regarded as supervised learning with a fixed, built-in teacher.

These observations are relevant to the present discussion because unsupervised learning can be readily extended to multilayered networks as illustrated by the "neocognitron" of Fukushima (1980). Successive layers form clusters of the clusters formed by preceding layers. Although hierarchical clustering by a layered network is an extremely interesting process, it does not address the problem in which we are interested, that is, the problem of learning "how" to do something on the basis of information only specifying "what."

Consider an adaptive network operating in an environment that can evaluate the behavior of the network, that is, the *collective* behavior of the network's elements, but cannot specify the desired behavior of each individual component.[12] This can occur in several ways. For example, the environment may be capable of evaluating the consequences of the network's behavior in controlling some aspect of the overt behavior of the entire learning system. Here, not only is the proper behavior of each network component unknown, but the proper behavior of the overall network may be recognizable only through its effects, which may quite indirect, on overt behavior. In other paradigms, the network's environment may know the desired behavior of a subset of the network's components, where the remaining elements must interact with these components in some unknown way in order for them to perform as desired. In all of these cases, structural learning must take place, and a difficult structural credit-assignment problems exists (Section "Introduction").

A learning task of this kind occurs when a network of some arbitrary number of elements faces a problem similar to the one to which we subjected a single A_{R-P} element in the preceding section. The network's environment presents stimulus patterns to the network by making the patterns' components available as input to some subset of the network's elements. We call the elements that receive this external stimulation the input elements. The output signals of another subset of elements are received by the environment, and patterns of these signals constitute the "overt" actions of the network. These are the output elements, or to use the term of Hinton and Sejnowski (1983), "visible elements." The elements that are not output elements (including any input elements that are not output elements) we call the "hidden elements" after Hinton and Sejnowski (1983). Suppose that the environment evaluates the activity of the visible elements and broadcasts a reinforcement signal to all the elements of the network. Since all elements receive the same reinforcement, they have no conflicts of interest and constitute a "team" according to game-theoretic terminology (Marshak and Radner 1972).

If we view the problem from the perspective of an individual adaptive 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.

[11] A qualification is in order here. If coupled to a component that adaptively adjusts the built-in similarity measure based on system performance, an unsupervised learning system can contribute to more powerful forms of directed learning

[12] By an agency in a network's environment, we do not necessarily mean an agency outside of the organism or 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. It would be underestimating the degree of abstraction inherent in our approach to identify the boundary between a network and its environment with the boundary between an organism and its environment

Consequently, let us focus on the learning task faced by one of the hidden elements. Even if the environment deterministically evaluates the network's actions, the relationship between this element's actions and the evaluation signal will not appear to be deterministic. This is true because the evaluation depends on the behavior of other elements, and relationships between them, in addition to the behavior of the element in question. Since it lacks knowledge about what the other elements are doing and how their activity influences the global behavior of the network, the given element will perceive that the evaluation is randomly related to its actions. 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 probabilistic contingencies. The element must be able to detect correlations between its actions and the reinforcement it receives that are buried in noise generated by the rest of the network. This is why the A_{R-P} element's ability to improve performance in arbitrary stochastic contingencies is essential to its performance as an adaptive network component.

But how can a hidden element inprove its reward probability when its output cannot directly effect the environment? The only possibility is for its actions to assist visible elements increase their reward probabilities; and this might be possible only by these actions assisting intermediate elements. For example, a hidden element might adjust its weights in order to produce a signal A that another hidden element combines with other information to produce a signal B, where signal B, in turn, allows a visible element to make a required discrimination. This does not require "altruism" on the part of the hidden element since its reward probability increases along with that of any other element.[13] The problem is not to provide the elements with sufficient incentive to cooperate in this way; it is rather to endow them with sufficient ability to discover how they can contribute to the common goal.

In the preceding discussion, we assumed that the reinforcement signal generated by the environment is broadcast to all the elements of the network, but this is only one of many possibilities. It represents a "worst case" in which no knowledge exists within the learning system that can ameliorate the structural credit-assignment problem. If reinforcement centers exist in the network's environment that can send appropriate *individualized* reinforcement signals to different subregions of the network, then the task faced by an interior element can be made easier. Such centers may exist in animal brains, and would certainly be useful in man-made learning networks, but we must ask where their knowledge comes from. If it is innate, then the learning system already knows something about how to implement the process being required by the environment, and structural credit assignment is easier. However, except in the unlikely case that the learning system already knows how to solve any task it is likely to face, the plight of an interior element that we have described will always exist within the subregions of the network over which reinforcement is uniform. On the other hand, if such knowledgeable reinforcement centers acquire their knowledge form experience, then it is likely that elements within these centers face learning problems similar to the one we have described. In either case, then, the problem we are considering occurs within the network at some level and to some degree. We now turn to some examples.

[13] The situation is analogous to kin selection in evolutionary processes in which genes are selected because they cause individuals to help close kin, i.e., other organisms that are likely to share those genes. This can account for the evolution of certain forms of apparent altruistic behavior. Here, since all the elements in a network receive the same reinforcement signals, the elements are related to one another in a way analogous to genetic relatedness

A minimal case of cooperative learning

Figure 4 shows a network of two A_{R-P} elements, e_1 and e_2. Only e_1 receives stimulus 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 this network faces an associative reinforcement learning problem like the one described above for a single A_{R-P} element. That is, the network's output, the output of e_2, affects the reward probability in a manner that depends on the stimulus pattern presented to e_1. Both elements receive the same reinforcement signal. 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 of both elements, but in the absence of a communication link, e_2 remains blind to the discriminative stimulus and therefore cannot learn to respond selectively in a discrimination task. On the other hand, in the absence of a communication link, e_1 can sense the discriminative stimulus but cannot influence the reinforcement received. The complementary specialties of the two elements have to be combined in order for each to attain optimal performance. In simulating this situation, we arranged for the action of e_1 to potentially influence e_2 by providing an interconnecting pathway with an initial weight of zero. If this weight 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 its input. Conversely, the correct behavior of e_1 depends on the value of the interconnecting weight, that is, on how e_2 has learned to respond to its input signals. Thus the two elements must adapt simultaneously in a tightly-coupled cooperative fashion in order to maximize reward frequency.

To be more specific, we set up the simulation in the following way. Each element is provided with a constant input to allow its threshold to vary (as described above) and one other input pathway. We regard only this second stimulus component as the stimulus pattern x, treating the constant input as part of an element's internal mechanism. Although according to Eq. (1) A_{R-P} elements produce actions +1 and −1, we find it convenient to recode these to be 1 and 0, respectively, when we are considering networks. In other words, the A_{R-P} mechanism works exactly as specified by Eq. (1) and (2) (with $y(t) = +1$ or -1), but an output signal of −1 is changed to 0 when transmitted to other elements or to the network's environment. We think of the output values 1 and 0 as "responding" and "not responding" respectively. With this recoding, each element of the network in Figure 4 can therefore receive the input "pattern" 0 or 1, where for e_1 it is generated by the network's environment, and for e_2 it is the (recoded) output of e_1.

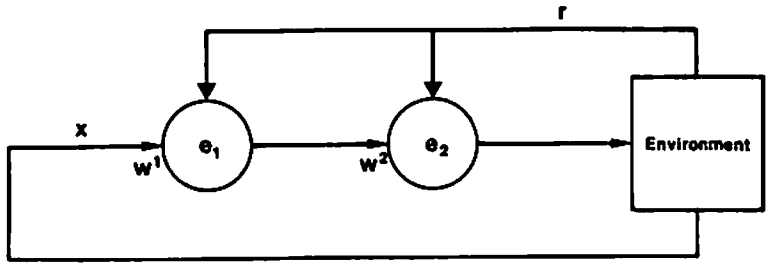

Fig. 4. A simple network of two A_{R-P} elements. Only e_1 receives the discriminative stimulus, x, and only e_2 can affect reinforcement, r, which is delivered to both elements

The reward probabilites implemented by the network's environment are given by the following table:

x	$d(x, 0)$	$d(x, 1)$
0	0.9	0.1
1	0.1	0.9

Table entry $d(x, y)$ is the network reward probability given that e_1 receives x as input and e_2 responds with y as output, that is, given that the network as a whole responds to x with y. Thus it is optimal for the network to respond to $x = 0$ with action 0 to obtain reward with probability 0.9, and to respond to $x = 1$ with action 1 to obtain reward with probability 0.9. In this task $M_{max} = (0.9 + 0.9)/2 = 0.9$, and the initial overall reward probability (with all weights zero) is $(0.9 + 0.1 + 0.1 + 0.9)/4 = 0.5$. Note that if the network fails to discriminate by responding identically to all input patterns, the overall reward probability is $(0.9 + 0.1)/2 = 0.5$.

There are two ways the network can solve this problem. Let us denote the weight associated with e_i's (nonconstant) input pathway w^i, $i = 1, 2$. In the first solution, e_1 learns to fire only when stimulus $x = 1$ is present by setting its threshold high (i.e., setting its threshold weight negative) and setting w^1 positive. Element e_2 does the same thing – sets its threshold high and w^2 positive – so that it fires only when stimulated by e_1's firing. Consequently, the network as a whole fires only when $x = 1$. In the second solution, e_1 learns to fire at all times *except* when stimulus $x = 1$ is present, and e_2 learns to fire at all times *except* when e_1 fires. Then when e_1 is silent in response to $x = 1$, e_2 is disinhibited and so fires.

In simulating a trial with this network, and with all the networks to be considered, the environment first presents a stimulus pattern to the network, and then proceeding from the input side of the network, we sequentially compute the output of the successive elements so that their actions are available as input to "downstream" elements. This is possible because the networks described here do not have recurrent connections. When the network's overt action is generated, the environment produces the reinforcement signal, and all the elements update their weights. We view the weight modifications as occurring simultaneously for all elements, although this is actually done sequentially by the computer program.

Some features of this tightly regimented procedure for computing a trial can be relaxed without presenting major difficulties. The environment can generate stimulus patterns, monitor network actions, and generate reinforcement simultaneously and continuously. This would require that we pay more attention to the real-time aspects of the problem than we do here, but we do not think it would be difficult. The addition of current connections within the network, however, presents deeper issues that we have not yet considered.[14]

Figure 5a and b show the behavior of the network for a typical sequence of 500 trials with $\lambda = 0.04$ and $\rho = 1.5$. Figure 5a 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 (nonconstant) input is 0, and the second shows the same thing for input 1. Both of these probabilities start at 0.5 since the weights are initially zero, and they change in approximately the same way for about the first 50 trials. This means that during these trials the element is experimenting with firing and not in the presence of both input signals. At this point the two conditional probabilities show the beginning of differentiation between the two cases, which becomes unequivocal by about trial 80. From then on, with a few brief exceptions, e_1 has a high probability of firing in response to an input of 1 and a low probability of firing in response to an input of 0. Figure 5b 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 (*not* for the values of e_2's local input). Since the network learns to respond correctly, e_2 learns to remain silent unless excited by e_1's activity; that is, the first solution is formed in which both w^1 and w^2 become positive and both units set high thresholds. Figure 5c shows the evolution of the overall performance measure M_t. Figure 5d is a histogram of the number of trials required to reach a criterion of 98% of M_{max} for each of 100 sequences of trials. In all sequences the network reached this criterion before 1500 trials. In 45% of the sequences, the network produced the first solution; in the remainder it produced the second.

A series of two elements in a discrimination task provides one of the simplest examples we could devise to demonstrate statistical cooperativity of self-interested elements. It is clear that the A_{R-P} elements effectively form a link that permits them to obtain higher reward rates than they could attain if they were to act independently. Moreover, an element contributes to the formation of this link only because doing so furthers its interests. We interpret this as a form of cooperativity in the literal game-theoretic sense. One may

[14] The research by Ackley et al. (1985), Hinton and Sejnowski (1983), Hopfield (1982), and Smolensky (1983) dealing with stochastic networks containing recurrent connections in probably relevant, but we wish to avoid the restriction to symmetric connections that is essential for their results. We do not see major difficulties with using A_{R-P} elements in networks with recurrent connections, but we have not yet studied this case in detail

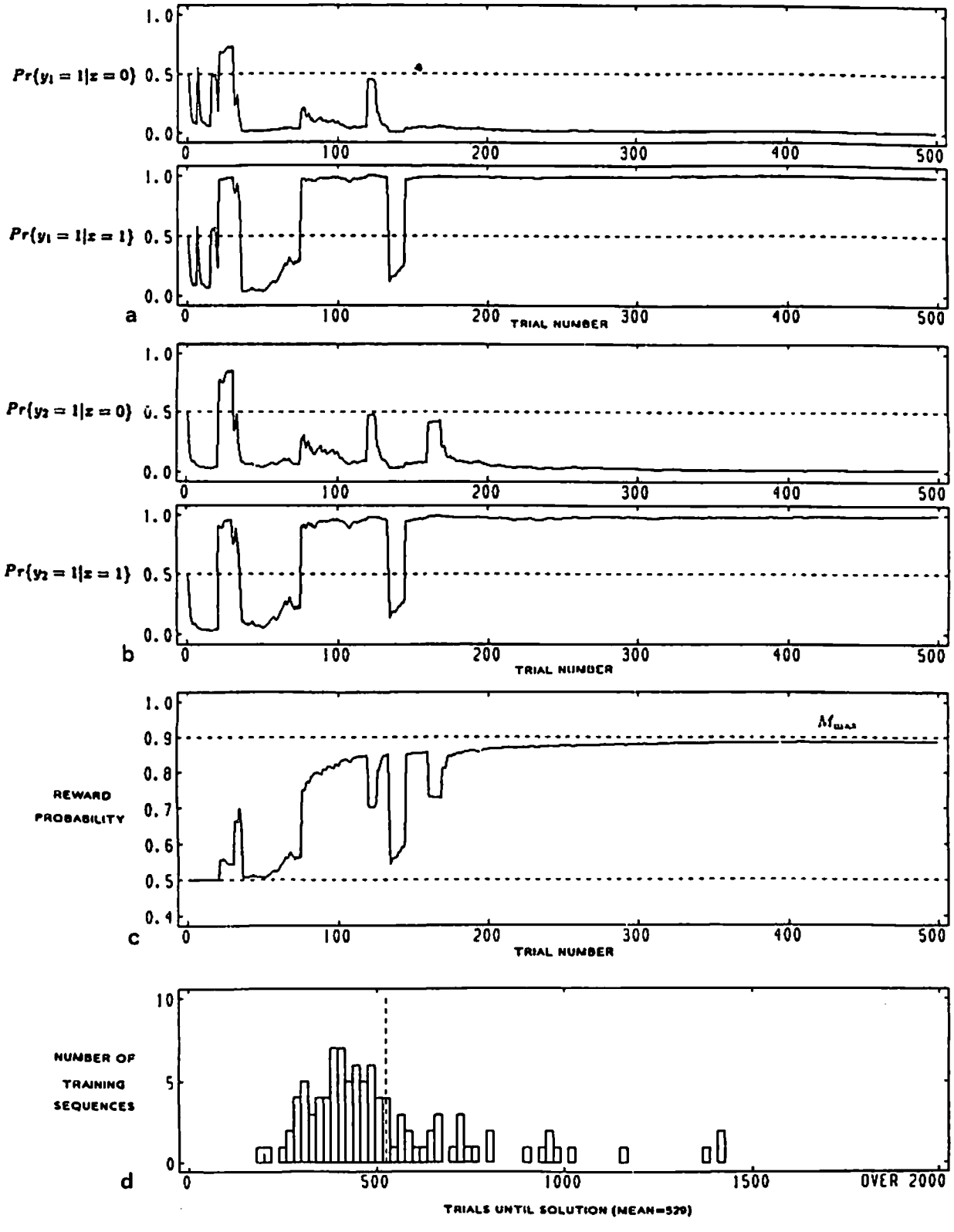

Fig. 5. Simulation results for the two-element network. See text for explanation

regard the link as a kind of "agreement" by which the elements form a coalition for mutual benefit. We have simulated series of 3, 4, and 5 elements with appropriate connections being made in all cases, although learning slows considerably as the depth of the network increases. Although the discrimination required in these tasks is not difficult, the necessity to construct a long chain of elements that faithfully transmits the discriminative stimulus is quite difficult. The correct behavior for any element depends on the behavior implemented by all the other elements so that the solution cannot be constructed from stable solutions to subtasks. This has implications, which we discuss below, about what initial network architectures might be expected to support faster learning than others.

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 illustrate the simplest example of a task that cannot be solved by a single linear threshold element, or any single-layer network of them. In this problem the hidden element is not needed simply to transmit a discriminative stimulus to the visible element; the hidden element must also learn to respond to particular configurations of its stimulus components in order to create a signal that the visible element needs to behave properly. In our simulation, a network of two A_{R-P} elements is placed in a task requiring it to form the two-component exclusive-or mapping. The network has a single hidden element, e_1, and a single visible element, e_2, which are connected as shown in Figure 6. The stimulus patterns are all the two-component binary vectors: $x^{(0)} = (0,0)$, $x^{(1)} = (0,1)$, $x^{(2)} = (1,0)$, $x^{(3)} = (1,1)$. These patterns are equally likely to occur on any trial. Each element also has a constant input and a threshold weight.

The reward probabilities are given by the following table:

x	$d(x, 0)$	$d(x, 1)$
(0,0)	0.9	0.1
(0,1)	0.1	0.9
(1,0)	0.1	0.9
(1,1)	0.9	0.1

Table entry $d(x, y)$ is the reward probability given that the network receives x as input and responds with action y. The optimal reward probability is $M_{max} = 0.9$, which is obtained when the action of the visible element is the exclusive-or of the pattern components, that is, when e_2 fires when one or the other, but not both, stimulus components are present. It must also not fire when both components are absent. A single A_{R-P} element can be correct for at most three of the four cases, yielding a reward probability of 0.7, since weights do not exist that allow a single linear threshold element to respond correctly to all four stimuli (see Duda and Hart 1973; or Minsky and Papert 1969). However, the performance of the network of Figure 6 can approach M_{max} if the hidden element learns to respond only to the fourth case and the visible element takes advantage of this signal to "debug" its responding. This can happen in several ways depending on whether the hidden element learns to turn on or off for the fourth case.

Figure 7 shows performance of the two-element network for a typical sequence of 5000 trials with $\rho = 1.5$ and $\lambda = 0.08$. In Figure 7a are graphs showing how the output probabilities of the visible element develop for each input pattern; Figure 7b shows the analogous information for the hidden element; and Figure 7c shows the overall performance of the network as a function of the trial number. The visible element quickly learns to respond correctly to all patterns except $x^{(1)} = (0,1)$ (Fig. 7a), causing the network performance to level off near 0.7 (Fig. 7c). Eventually ($t \approx 1400$) the hidden element comes to respond reliably to $x^{(1)}$ and to reliably not respond to any other pattern (Fig. 7b). At the same time, the visible element begins to be excited by the hidden element's signal so that its output tends to be correct more frequently for all four patterns (Fig. 7a). Once this mutually beneficial relationship between e_1 and e_2 begins, it quickly develops until amost perfect performance is achieved (the theoretical asymptote is 0.892 for this value of λ). It is clear that this is a cooperative process. Figure 8 shows a histogram of the number of trials until a criterion of 95% of M_{max} is attained for each of 100 sequences of trials. The average number of trials until criterion is 3501, or about 875 trials for each stimulus pattern. In all of the sequences the network reached this criterion before 15,000 trials.

This example illustrates how a hidden unit can learn to respond to particular constellations of stimulus components, and to influence other elements, in a useful way without being provided with explicit instructional information. Consequently, this two-element network can learn to implement any of the 16 two-valued functions of two binary stimulus components. This raises the following question: Why not *begin* with elemental components that can individually implement all of these functions? Or, more generally, why

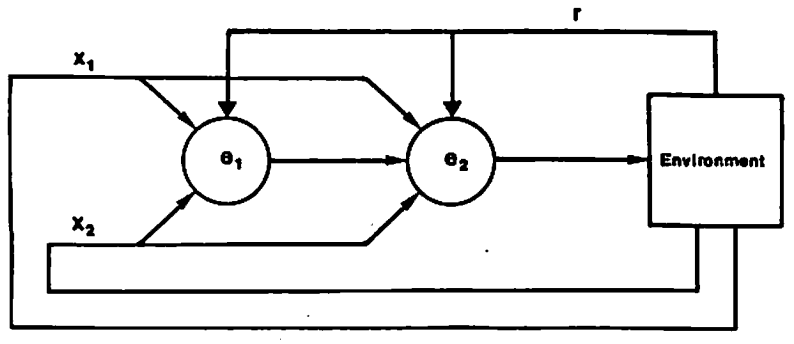

Fig. 6. Network for the exclusive-or task. The elements must cooperate in order for the network to learn to implement the exlusive-or mapping

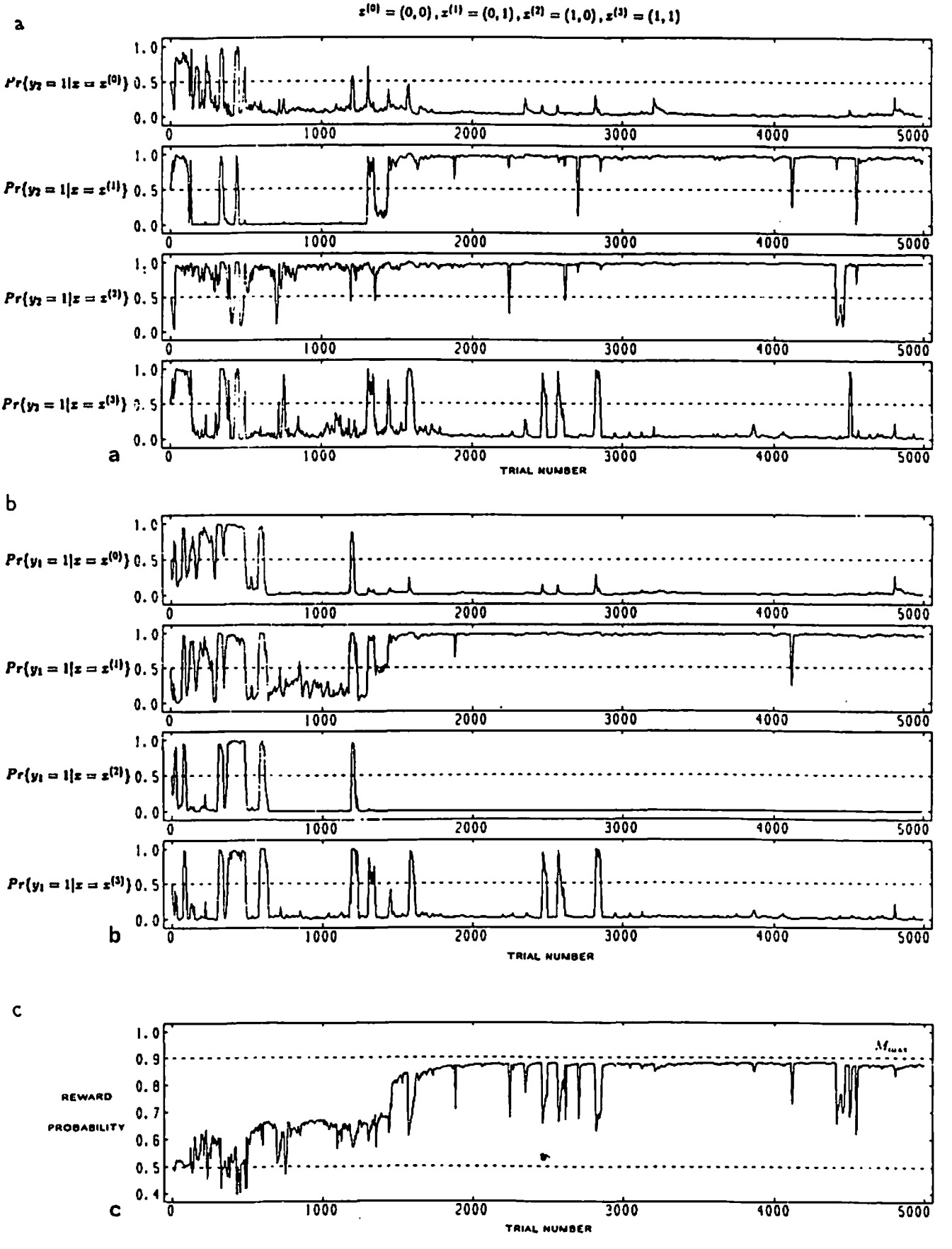

Fig. 7a–c. Behavior of network elements in a typical sequence of 5000 trials in the exclusive-or task. See text for explanation

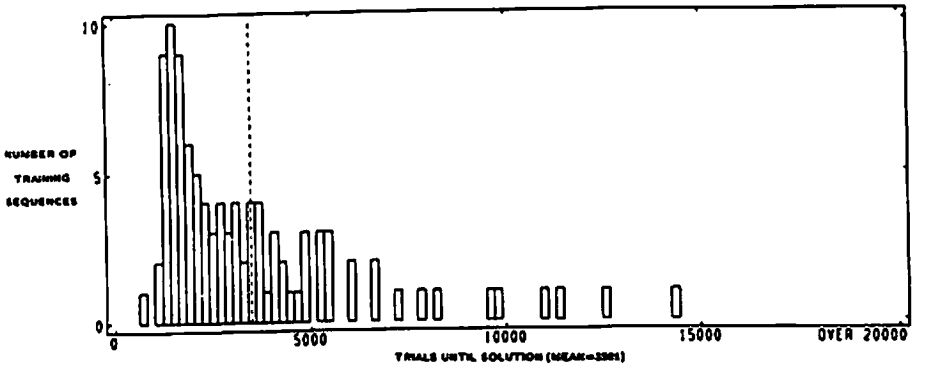

Fig. 8. Simulation results for 100 sequences of trials in the exclusive-or task. The histogram shows the number of trials to a criterion of 95% of optimal performance for each of 100 sequences of trials

not use elements that can implement transfer functions more complicated then memoryless linear threshold functions? It is certainly the case, for example, that neurons are not simple linear threshold elements. We have no objections to this except that such a theoretical approach would not by itself solve the problems we are addressing. It would still be necessary to consider ways of obtaining cooperative interactions among these more complex elements, and the same problems we have been discussing would appear again. By focussing on elements with relatively simple transfer functions, we can study the problems of cooperative learning in a relatively simple framework. Moreover, although it is not illustrated here, we believe that there are no major obstacles to extending what we have learned using linear threshold elements to networks of more complicated primitives.

A more difficult nonlinear task

The network shown in Figure 9 has six input components and a single principal output pathway (from element 5). There are 39 weights to adust: one associated with each of the pathway intersections and one threshold weight for each element. There is also a reinforcement pathway which is not shown in the Figure. 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 k address input pathways and 2^k data input pathways (here $k = 2$) each of which is associated with a distinct k-bit address. 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 pathway 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 (element 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 $T = 0.5$ except for the visible element (element 5) which uses $T = 0$ (and therefore essentially uses the perceptron algorithm; see the Section "Relationship of the A_{R-P} element to other adaptive elements"). Figure 10 is a histogram of the number of trials required for the network to respond 99% correctly for 1000 consecutive trials in each of 30 sequences of trials with $\rho = 1$ and $\lambda = 0.01$. The average number of trials required is 133,149, or about 2080 presentations of each stimulus pattern. In every sequence the network reached the criterion before 350,000 trials.

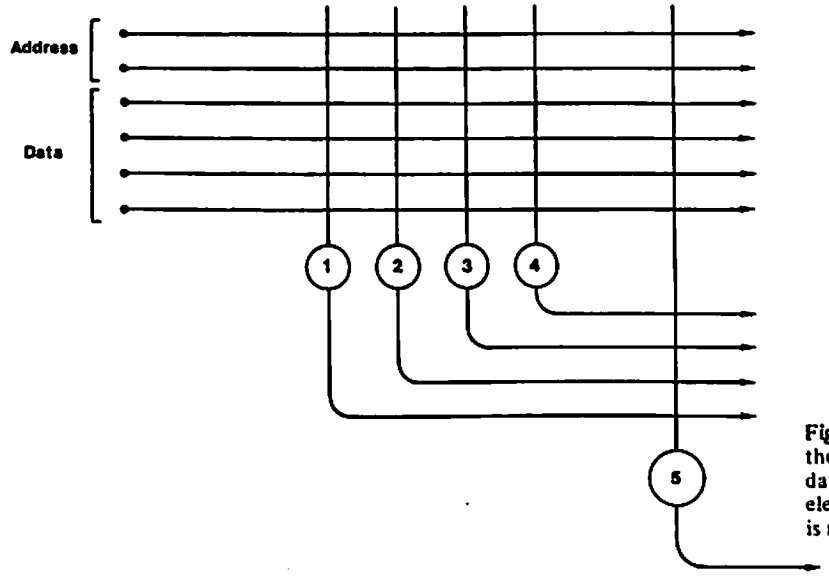

Fig. 9. Network for the multiplexer task. Depending on the activity pattern over the address pathways, one of the data signals must be transmitted as the output of e_5. All elements receive the same reinforcement signal, which is not shown

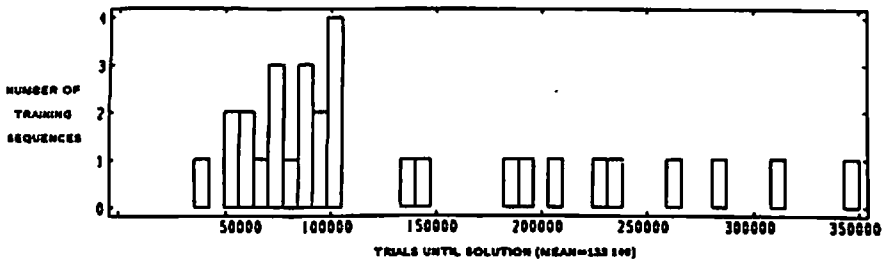

Fig. 10. Simulation results for the multiplexer task. The histogram shows the number of trials required for the network to reach criterion performance in each of 30 sequences of trials

This is a difficult task since the natural generalizations produced as a result of similarities among the stimulus patterns tend to be irrelevant or misleading with respect to the required actions of the network. Consequently, this task represents a rather stringent test of the learning method. The hidden elements (elements 1–4) must tune to certain constellations of the stimulus components in order to disrupt misleading generalizations. There are several ways that this can be done, and the network comes up with different solutions in different sequences of trials. This task also illustrates some of the computational sophistication that can arise with the formation of highly nonlinear functions. Linear threshold functions can exhibit only a very restricted form of context sensitivity: contextual information can bias activation one way or the other, effectively raising or lowering the threshold. Nonlinear context sensitivity, on the other hand, can result in the complete alteration of behavior as a function of contextual information. The exclusive-or task of the preceding example illustrates this in the simplest form, where one stimulus component can be regarded as switching the processing of the second stimulus component between the identity and inversion functions. The multiplexer illustrates a more extreme form by which the contextual information provided over the address pathways completely alters the set of signals to which the principal element is sensitive. A similar phenomenon appears to occur with the so-called "place cells" in hippocampus that seem to represent different places depending on the context (Kubie and Rank 1983).

Discussion of the simulation results

These simulation experiments suggest that layered networks of A_{R-P} elements can reliably learn nonlinear associative mappings without being explicitly instructed how to implement them. However, these results also suggest that 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 have not yet systematically done this. Preliminary comparison with unsophisticated random search through the space of all combinations of weight values indicates that A_{R-P} networks are vastly faster, but much more work needs to be done before we can knowledgeably comment on the efficiency of A_{R-P} networks. It must be remembered, however, that the dimensionality of this search space is relatively large even for the small networks simulated here; for example, the weight space for the multiplexer network has 39 dimensions. Still, learning times may become exceedingly long for large networks.

There are several factors that bear on the issue of learning rate. First, in the simulations described here, each sequence of trials begins with all the weights set to zero so that the required mapping has to be learned from scratch. This is merely a methodological convenience and does not imply that we are philosphically inclined toward a *tabula rasa* view of learning and intelligence. On the contrary, it is likely that large, deep networks are only capable of learning sufficiently quickly if they begin with initial pathways and weight values that place the network's behavior "in the ballpark." Moreover, it should not be overlooked that the learning methods described here also permit networks to recover from damage. In this case, the starting point for adaptive reorganization will depend on the extent of the damage.

A second factor concerns the tradeoff between speed and accuracy that is inherent in any type of stochastic search. Although the networks are able to improve performance with any admissible parameter values ($0 < \lambda \leq 1$, $\rho > 0$), these values effect the speed of learning and the degree of performance eventually achieved. The simulation in the Section "A self-interested adaptive element" (Fig. 3) shows the tradeoff for various values of λ for a single A_{R-P} element, and similar results appear when networks are simulated. The parameter ρ has a more complex effect on performance but also participates in this tradeoff.[15] In the next section, theoretical results for a single A_{R-P} element are described which we have not rigorously extended to networks of A_{R-P} elements; nevertheless, our network simulations *suggest* the following. If one is willing to accept the network getting stuck at suboptimal performance levels, then learning can be made much more rapid. On the other hand, if one demands eventual near-optimal performance, then one has to be willing to wait for it. The important point, however, is that unlike the situation for networks with a single adaptive layer, near-optimal performance will always be achieved (we must emphasize again that this has not been proven). A compromise between these extremes might be obtainable by systematically varying the parameters as learning proceeds – for example, by starting with λ near one and decreasing it as performance improves, but we have not yet experimented with this.

A related issue concerns the manner in which a network's performance approaches a desired level. If a large network facing a difficult learning problem can maintain a high level of performance while the learning process is underway, then a long wait for an optimal solution may not be such a problem. Networks of A_{R-P} elements appear to have this property if their architectures are appropriate. For example, in solving the exclusive-or task described above, the network quickly solved the easy part of the problem by learning to

[15] For A_{R-P} elements that use the logistic distribution [Eq. (3)], the same effect can be achieved by either varying ρ or T: increasing ρ has the same effect as decreasing T

respond correctly to three of the four cases. Considerably more trials were required to obtain the complete solution, but during this period the network's performance remained relatively high (Fig. 7c). Behavior like this is characteristic of networks which do not require long chains of elements to be formed to allow environmentally supplied stimuli to influence the visible elements. This suggests that effective network architectures might be like that of Figures 6 and 9, with hidden elements forming auxiliary side networks rather than being strictly interposed between layers. This architecture is also plausible from an evolutionary perspective if we imagine that additional network structure is added around existing functional structure rather than being inserted into it.

Theoretical analysis

In this section we provide a theoretical justification for the reliable performance of the A_{R-P} element as an adaptive network component. By first looking carefully at the nonassociative aspects of the task faced by a hidden network element and then at the learning capabilities of a single A_{R-P} element, one can gain some understanding of the learning process. In the next section, we relate the A_{R-P} element to adaptive elements developed in the past and gain some understanding why networks of those elements are not able to perform reliably in similar tasks. The notation used in this section is that developed in the Section "A self-interested adaptive element".

Contingency space

To help understand the nonassociative aspects of the task faced by a hidden element we borrow the notion of *contingency space* from animal learning theorists (see Staddon 1984). Recall that an associative reinforcement learning task for a learning system with two actions +1 and −1, as described in "A self-interested adaptive element", is characterized by two reward probabilities, $d(x, +1)$ and $d(x, -1)$, for each input pattern x in X. We can therefore represent the task by a set of points, one of each x, plotted in a contingency space whose coordinates are respectively the reward probability given action +1 in the presence of x and the reward probability given action −1 in the presence of x (Fig. 11). A single point corresponds to one of the component nonassociative reinforcement learning tasks that comprise the associative task. Various regions of the contingency space correspond to nonassociative reinforcement learning tasks that pose different kinds of problems for learning algorithms.

Letting the abscissa be the reward probability for action +1, the diagonal line of slope 1 divides the space into two triangular regions, the lower-right one corresponding to (nonassociative) reinforcement learning tasks in which action +1 is the optimal action, and the upper-left one corresponding to tasks in which action −1 is the optimal action. Without loss of generality, let us assume that action +1 is optimal and focus only on the lower-right triangle, i.e., the set of

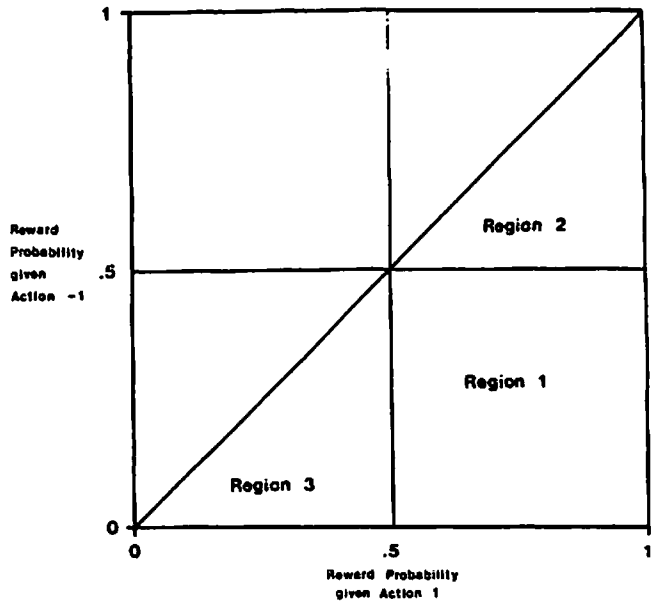

Fig. 11. Contingency space. Each point in contingency space corresponds to a nonassociative reinforcement learning task. Tasks falling in the different labelled regions present different types of difficulties for a learning algorithm; see text for details

points with $d(x, +1) > d(x, -1)$. All of the regions to be delineated have obvious counterparts in the upper-right triangle. Figure 11 shows three numbered regions. Region 1 consists of all those points with $d(x, +1) > 0.5$ and $d(x, -1) < 0.5$. We consider this to be the "easy" region since the environment usually rewards the optimal action and usually penalizes the non-optimal one. Notice that the set of tasks in which the reward probabilities for the two actions sum to one lies completely within the easy region. Region 2 consists of those points with $d(x, +1) > 0.5$ and $d(x, -1) > 0.5$. These tasks are much more difficult because the environment usually rewards both actions, and the learning algorithm must figure out, so to speak, which of two good actions is better – "the greater of two goods." Region 3 consists of points corresponding to tasks that are difficult for the opposite reason: the learning system must figure out which of two bad actions is better – "the lesser of two evils."

Tasks that fall in regions 2 and 3 present special difficulties because the predominant estimate for the desired action, i.e., the more frequent value of $r(t)\,y(t)$, depends on the action probabilities of the learning system. For tasks in region 2, for example, an action can more frequently appear to be the desired action just because it is being performed more frequently than the other action; for tasks in region 3, an action can more frequently appear to be the desired action just because it is being performed less frequently than the other action. For tasks in the easy region, on the other hand, the predominant estimate for the desired action is independent of the learning system's action probabilities, and learning algorithms can be simpler. These observations, which are shown mathematically by Barto and Anandan (1985), provide one way of understanding the role of the term $E\{y(t)|s(t)\}$ used in the A_{R-P} algorithm [Eq. (2)]. This term adjusts the magnitude of weight changes to counteract any advantage an action may appear to have that is

actually due to its being performed more (or less) frequently than the other action.[16]

The point of this analysis is that the task faced by a hidden element in a network will not generally fall in the easy region of contingeny space for all of its input patterns. For example, in the exclusive-or task described above, before trial 1400, the hidden element, e_1, is usually penalized no matter what it does when input pattern $x^{(1)}$ is present. This occurs because the visible element is reliably acting incorrectly in this case, having incorrectly generalized from its experiences with the other three patterns. Thus, for this input pattern, the task of e_1 falls in region 3 of contingency space, a hard region. In fact, the task faced by the visible element, e_2, also falls in region 3 for this input pattern. As all of the weights of the network change, the point representing the task of e_1 given $x^{(1)}$ moves about contingency space but generally remains in region 3. At the same time, the task of e_2 moves about contingency space but generally also remains in region 3. Both elements need to be able to make progress under these circumstances. As we see next, the A_{R-P} element is able to solve associative reinforcement learning problems whose component problems fall anywhere within contingency space.

The A_{R-P} convergence theorem

The learning capabilities of the A_{R-P} element when faced with an associative reinforcement learning task are summarized by a theorem proved by Barto and Anandan (1985). Here we informally describe this result. Several conditions on both the task and the A_{R-P} element are sufficient to ensure the convergence result. The set X of stimulus patterns used for training must be a linearly independent set of vectors (i.e., no vector in the set can be a linear combination of any of the others). Further, each stimulus pattern in X must have a nonzero probability of being presented on any trial, which implies that in any infinite sequence of trials, each pattern in X will (almost surely) occur infinitely often. Each random number $\eta(t)$ used by the A_{R-P} element must be selected according to a (cumulative) distribution function that is continuous and strictly monotonically increasing. The major implication of this condition is that any increase (decrease) in the weighted sum $s(t)$ always increases (decreases) the probability that the element emits action +1; there is no ceiling effect. This excludes the case in which the $\eta(t)$ are selected according to any uniform distribution, including the deterministic case in which all the $\eta(t)$ equal the same constant. It is satisfied by the logistic distribution we used in the simulations. Finally, the parameter ρ in Eq. (2) must decrease at a certain rate with successive trials. This is a standard condition for the convergence of certain types of pattern classification algorithms, but it is often ignored in practice since learning is greatly slowed by decreasing ρ and the algorithms tend to behave well with it held constant. Although decreasing ρ is required for the convergence result to be stated, we held it constant in all the simulations presented. See Barto and Anandan (1985) for the details of this condition. No restriction of any kind is placed on the reward probabilities $d(x, +1)$ and $d(x, -1)$; the component tasks can therefore fall in any region of contingency space.

Under these conditions, the following convergence result holds for an A_{R-P} element in an associative reinforcement learning task. For any initial weight values and any value of λ, $0 < \lambda < 1$, every infinite sequence of learning trials causes the A_{R-P} element to converge[17] to a weight vector that causes the correct action to be produced in response to each stimulus pattern in X with probability greater than 0.5, i.e., the correct action will be more likely than the incorrect one. Further – and this is the important part – the smaller λ is, the larger will be the probability of producing the correct action in response to each stimulus pattern in X when the process converges; in fact, as λ approaches zero, the probability of each correct action approaches one. By the correct action for an input pattern x we mean the action y^x such that $d(x, y^x) = \max\{d(x, +1), d(x, -1)\}$. Recall that λ determines the relative effectiveness of reward and penalty in the learning process. As λ approaches zero, the process becomes more asymmetrical, with the effect of penalty approaching zero. Interestingly, the proof by Barto and Anandan (1985) does not go through, and the result appears not to hold, when λ actually equals zero.

One of the most important implications of this result is that the A_{R-P} element can improve its performance in a wide range of associative reinforcement learning tasks without the necessity of using different values of the parameters ρ and λ for tasks involving contingencies in different regions of contingency space. Therefore learning is possible when there is no a priori-knowledge about the contingencies – a property critical to the performance of A_{R-P} elements that are embedded in networks. Unfortunately, however, the situation is not quite so straightforward. For given parameter values, the asymptotic performance level is only guaranteed to be better than chance. How much better than chance depends not only on λ, as described above, but also on the environmental contingencies. The asymptotic performance level with a fixed value of λ decreases as contingencies move deeper into the hard regions of contingency space (i.e., closer to the lower-left or upper-right corners). This suggests that one should set λ very small so that adequate performance will be attained in all foreseeable circumstances. Unfortunately, however, learning rate decreases as λ decreases. So we are faced with a tradeoff unless methods are devised for varying λ as learning proceeds.

The most serious limitation of this result is the requirement that the stimulus vectors form a linearly independent set. Although this is a much weaker condition than orthogonality of the stimulus vectors, it implies that there can be at most n different patterns in the training set, where n is the number of input pathways of the adaptive element (excluding the reinforcement pathway). Does this mean that we have to make sure that no linear dependencies ever exist among the input vectors? Fortunately, the answer seems to be no. Linear independence of the training vectors ensures that

[16] Another method for coping with this difficulty is to use another set of weights to construct an estimate of the reward probability for each input pattern. Weight updates are then based on the discrepancy between this estimate and the observed reward. The effect of this "reinforcement comparison mechanism" is to adjust the effective reinforcement so that the contingency appears to be in the easy region of contingency space. Sutton (1984) extensively discusses this approach

[17] "*Every* infinite sequence" is not quite accurate. The random process almost surely converges, meaning that the probability of seeing an infinite sequence of trials for which the result does not hold is zero

any associative mapping from X to action probabilities can be implemented by the adaptive element.[18] In particular, there exist weight values that yield performance as close to optimal as desired. However, although it has not yet been proven, when X is not a linearly independent set, it is likely that the A_{R-P} element is able to find the weight values that yield some sort of best approximation to the optimal performance that can be attained by adjusting the weights. Algorithms capable of doing this for problems restricted to the "easy" region of contingency space are well-known (such as the Widrow/Hoff rule described in the Section "Relationship of the A_{R-P} element to other adaptive elements"), where best approximation means one yielding the least mean-square error. We are currently working toward extending the A_{R-P} convergence theorem in this direction.

We know of no other algorithm that is provably capable of this combination of decision-under-uncertainty and associative learning. Other algorithms, such as those using the "reinforcement comparison" approach developed by Sutton (1984), may possess the required degree of robustness, and we are in the process of systematically comparing their performance in networks with that of the A_{R-P} element.

Relationship of the A_{R-P} element to other adaptive elements

The A_{R-P} algorithm was devised by extending several well-known algorithms and is therefore closely related to them. By comparing adaptive elements implementing these algorithms with the A_{R-P} element, we can place the A_{R-P} element in its proper historical and theoretical perspective and gain some understanding about why the other elements are not able to learn as hidden elements of networks. In addition to the algorithms discussed here, many others have been proposed in the literature on adaptive pattern classification, and it is impossible to treat them all. We have chosen examples that are the most well-known, that have been presented as neuron-like adaptive elements, and that represent major classes of algorithms. Some of the adaptive elements described here are also described by Sutton and Barto (1981), and only what is necessary to make the presentation self-contained will be repeated. All of the elements to be described use the notation introduced in Figure 1.

Correlational elements

A simple rule for updating the connection weights is

$$\Delta w_i(t) = \rho\, y(t)\, x_i(t), \qquad (4)$$

for each i, $1 \leq i \leq n$, where $\rho > 0$ is a constant determining the rate of change of w_i. This rule adjusts a weight according to the correlation between the presynaptic signal, $x_i(t)$, and the postsynaptic signal, $y(t)$. It is perhaps the most literal mathematical interpretation of Hebb's postulated learning rule (Hebb 1949), and has been used extensively, with a variety of modifications, in theoretical adaptive network studies. For example, units employing this rule have been used in associative memory networks that have a number of interesting properties (Anderson et al. 1977; Hinton and Anderson 1981; Kohonen 1977).

Adaptive elements using variants of this rule are suited to a type of learning that resembles classical conditioning. The unconditioned stimulus (US) arrives at the unit via a pathway that has a large positive weight. The US therefore causes the unit to fire (thereby contributing to the unconditioned response, or UR). Any input pathway that is active when the US-UR occurs therefore has its weight increased, making a signal arriving on that path (a conditioned stimulus, or CS) more efficacious in firing the unit (and thereby contributing to the production of the conditioned response, or CR). Although this pure-contiguity view of classical conditioning is not accurate, the rule can be modified in a number of ways to make it a better model of classical conditioning as discussed extensively by Sutton and Barto (1981), who also discuss the necessity to add additional mechanisms to obtain a stable learning procedure. For our purposes, it is important to note that although there is no specialized reinforcement pathway, a given pathway, or a set of pathways, is initially provided with a weight of large magnitude, often unaffected by the learning rule, that acts not only as a built-in reflex pathway but also as a training pathway. A signal arriving via this pathway forces the unit's activity to be high or low. Equation (4) changes the weights of the simultaneously active input pathways so that the unit's output will tend to be driven correspondingly high or low when those active input pathways are active again in the future, even if the training pathways are not active.

This type of learning, which is performed better in many respects by more sophisticated learning rules, is a form of supervised learning in which the element's environment specifies desired responses. Although the training signals are often regarded as a form of reinforcement (just as the US in classical conditioning is regarded as a reinforcer), this type of element is in no sense capable of maximizing reward frequency, or of controlling any aspect of its input, and we do not regard it as a self-interested element.

Widrow/Hoff element

Widrow and Hoff (1960) described an adaptive element that they called an "adaline", for *ada*ptive *lin*ear *e*lement (see also Widrow 1962). Its response is determined by comparing the weighted sum of the inputs to a fixed threshold:

$$y(t) = \begin{cases} +1, \text{ if } s(t) > 0; \\ -1, \text{ if } s(t) \leq 0; \end{cases} \qquad (5)$$

and the weights are updated as follows:

$$\Delta w_i(t) = \rho\,[z(t) - s(t)]\, x_i(t), \qquad (6)$$

for each i, $1 \leq i \leq n$. Here, $z(t)$ is the value of a specialized training signal giving the desired response of the unit. This rule is suitable for supervised learning tasks as described for correlational rules except that it is designed to adjust the weights in order to match each $z(t)$ and $s(t)$ as closely as possible, that is, to reduce the error, or discrepancy, $e(t) = z(t) - s(t)$. When $s(t)$ is too low, $e(t)$ is positive, and Eq. (6) increases (decreases) the weights of pathways carrying positive (negative) signals, $x_i(t)$. This causes s to be larger when a similar input pattern appears in the future. When $s(t)$ is too high, $e(t)$ is negative, and the same thing happens *mutatis mutandis*. This process is called "error-correction."

[18] Of course, this does not mean that any associative mapping from $\mathcal{R}^n$ to action probabilities can be implemented by a single element

There is a well-developed theory about this learning rule, some of which is discussed in Sutton and Barto (1981), where it is also pointed out that it is essentially the same as Rescorla and Wagner's (1972) model of classical conditioning. It is capable of forming desired associative mappings under a broader set of conditions then are simpler correlational rules. However, we still do not consider this adaptive element to be self-interested since it is not able to learn to control any aspect of its input. Its goal, as it were, is simply to produce a match between its actions and the training signal.

Perceptron element

Although Rosenblatt (1962) studied many learning rules for his "perceptron", the rule that has come to be called the perceptron learning rule is an error-correction procedure very similar to the Widrow/Hoff rule. Element responses are produced according to Eq. (5), and weights are updated by the following rule:

$$\Delta w_i(t) = \rho\,[z(t) - y(t)]\,x_i(t). \tag{7}$$

for each i, $1 \leq i \leq n$, where $z(t)$ is +1 or −1.[19] Here the error is the difference between the desired output and the actual output, $y(t)$, rather than the weighted sum, $s(t)$, as in Eq. (6). No weight changes are made when the response is correct. Despite their similarities, the Widrow/Hoff and perceptron rules have significant differences in their convergence properties (see Duda and Hart 1973; Minsky and Papert 1969) but these need not concern us here.

Although there is considerable controversy about the relationship between classical and instrumental conditioning (see, for example, Mackintosh 1983), there seems to be no disagreement that error-correction learning rules, such as the Widrow/Hoff and perceptron rules, are not designed for learning in paradigms involving response contingencies. Nevertheless, we have seen descriptions of these rules, and the paradigms in which they are intended to operate, that are very misleading in this regard. Sometimes the discrepancy, or error, $e(t) = z(t) - s(t)$ for the Widrow/Hoff rule and $e(t) = z(t) - y(t)$ for the perceptron rule, is regarded as being computed by the learning system's environment rather than by the system itself. In this case, the training signal is this error signal, which *is* response-contingent feedback. Consequently, the term "trial-and-error learning", generally treated as roughly synonymous with instrumental learning, has been applied to this process. This is a very misleading view because the error signal provides a different kind of information than does a reward/penalty signal. Considering the perceptron rule, for example, $e(t) = +1$ tells the element that it should have fired when it did not, and $e(t) = -1$ tells it that it should not have fired when it did. On the other hand, a reward/penalty signal, such as $r(t)$ used by the A_{R-P} element, evaluates the action performed without directly specifying what action would have been correct. For example, $r(t) = +1$ tells the A_{R-P} element that the action just performed, *whatever it was*, should be performed more frequently in the presence of the current stimulus. It does not tell it that the action should have been +1. Now in the case of just two possible actions, the element can combine a reward/penalty signal with knowledge of its action to deduce an error signal (e.g., if it just performed action −1 and received a penalty $r(t) = -1$, then it should have performed action +1, so the error is +1). This is the basis of the selective bootstrap element and the reinforcement learning perceptron described below.

Associative search element

The author and colleagues described an adaptive element in some previous publications that is mentioned here in order to make the connection to that earlier work (Anderson 1982; Barto et al. 1981; Barto and Sutton 1981b). In addition, this element, which we call the "associative search element", is perhaps the simplest extension of the correlational rule [Eq. (4)] which makes it applicable to associative reinforcement learning. An explanation of this element serves to relate the basic idea behind all of the reinforcement learning rules to the Hebbian postulate. The output of the associative search element is computed in the same way that it is computed by an A_{R-P} element: the weighted sum of the input signals is compared with a random threshold according to Eq. (1). The weights are updated according to the following rule:

$$\Delta w_i(t) = \rho\, r(t)\, y(t)\, x_i(t), \tag{8}$$

for each i, $1 \leq i \leq n$, where $r(t)$ is +1 (reward) or −1 (penalty). This is just the correlational rule [Eq. (4)] with an extra factor that modulates the process according to reinforcement. In previous studies we were careful to point out that there is a necessary time delay between an action and the contingent reinforcement. In the simplest case in which the delay is always a single time step, the rule appears as follows:

$$\Delta w_i(t) = \rho\, r(t)\, y(t-1)\, x_i(t-1).$$

Since in the present discussion we are not addressing real-time issues, we ignore any delay occurring within a trial, and Eq. (8) is adequate if it is understood that $r(t)$ is the reinforcement contingent upon action $y(t)$.

This rule makes clear the three factors minimally required for associative reinforcement learning: the stimulus signal, x; the action produced in its presence, y; and the consequent evaluation, r. One can view the learning process as one in which basic yx correlations are formed *but held in abeyance* until the relevant reinforcement occurs,[20] at which time they are "fixed" in a manner that depends on the type of reinforcement received. The adaptive element proposed by Klopf (1972), which he called the "heterostat", first introduced this idea to us. The A_{R-P} element incorporates this same principle but has as a basis a rule like the Windrow/Hoff or perceptron rule instead ot the simpler correlational rule. The associative search element is not capable of performing well when facing tasks with contingencies falling in the hard regions of contingency space, and it is not able to discriminate among similar stimulus patterns in the way that the A_{R-P} element can. Its performance improves in both of

[19] Usually this rule is defined with a threshold function that yields the values 0 and 1 instead of −1 and +1, and $z(t) = 0$ or 1. This version is the same as ours if its ρ is twice ours

[20] Again, we are purposefully ignoring the problem of determining when the relevant reinforcement occurs (the "temporal credit-assignment problem"). See Barto et al. (1983) and Sutton (1984)

these respects when the reinforcement it receives is preprocessed by a mechanism that compares the actual reinforcement with that "expected" when acting in the presence of similar stimulus patterns. This approach is developed extensively by Sutton (1984).

Selective bootstrap element

Widrow et al. (1973) described an extension of the Widrow/Hoff algorithm that is, to the best of my knowledge, the algorithm in the literature most closely related to the A_{R-P} algorithm. Whereas the Widrow/Hoff and perceptron elements receive a training signal, $z(t)$, that directly specifies the desired response at trial t, the selective bootstrap element receives a reward/penalty signal, $r(t)$, as does the A_{R-P} element. The selective bootstrap element uses the deterministic thresholding given by Eq. (5) to determine its output. It uses the following equation to update its weights:

$$\Delta w_i(t) = \rho\,[r(t)\,y(t) - s(t)]\,x_i(t), \qquad (9)$$

for each i, $1 \leq i \leq n$, where $r(t)$ is +1 (reward) or −1 (penalty). This element therefore differs from the A_{R-P} element in that its output is a deterministic function of $s(t)$ and $s(t)$ is used instead of $E\{y(t)|s(t)\}$ in the weight update equation. Additionally, it is a symmetric rule that changes weights by equal magnitudes upon reward and penalty, although Widrow et al. (1973) did discuss an asymmetric version. Loosely speaking, this element is a deterministic and symmetric version of the A_{R-P} element, and its behavior can be understood in similar terms.[21]

It is instructive to discuss the reason that Widrow et al. (1973) chose the term "selective bootstrap adaptation" to describe this learning process. Their starting point was the supervised learning paradigm in which the training signal, $z(t)$, specified desired responses, but they supposed that this training signal was not available. They called learning by means of the Widrow/Hoff rule with $z(t) = y(t)$ "positive bootstrap adaptation". It updates weights as if the output actually produced was in fact the desired response – bootstrapping, as it were, based on its own actions. On the other hand, they called learning by means of the Widrow/Hoff rule with $z(t) = -y(t)$ "negative bootstrap adaptation". In this case weights are updated as if the out *not* produced was the desired response. Finally, "selective bootstrap adaptation" means switching from positive to negative bootstrap adaptation, or vice versa, depending on a signal from the environment, $r(t)$, indicating reward or penalty. Equation (9) can then be seen as switching the Widrow/Hoff rule [Eq. (6)] between its positive and negative bootstrapping modes. The A_{R-P} element can be given a similar interpretation.

Although the selective bootstrap element and the A_{R-P} element are similar, they have very different learning capabilities. Whereas the A_{R-P} element can learn effectively for response contingencies that fall anywhere within contingency space (Fig. 11), the selective bootstrap element is only able to learn reliably when facing response contingencies that fall within the easy region of contingency space (region 1). Barto and Anandan (1985) describe simulations that compare the performance of these elements for various types of contingencies. The selective bootstrap element can learn much faster than the A_{R-P} element for easy tasks, but either oscillates or sometimes converges to the wrong action in hard tasks. This severely limits the utility of the selective bootstrap element as an adaptive network component.

Reinforcement learning perceptron

Although this adaptive element has never been singled out as being different in any significant way from the perceptron element described above, it is mentioned here because it is a special case of the A_{R-P} element. Note that the response mapping rule of the A_{R-P} element [Eq. (1)] reduces to that of the perceptron element [Eq. (5)] if each random variable $\eta(t)$ in Eq. (1) is always zero. In this case, the expected output given $s(t)$ is just the actual output $y(t)$, so that if we let $\lambda = 1$ the A_{R-P} learning rule given by Eq. (2) becomes

$$\Delta w_i(t) = \rho\,[r(t)\,y(t) - y(t)]\,x_i(t),$$

for each i, $1 \leq i \leq n$, where $r(t)$ is +1 (reward) or −1 (penalty). This is just the perceptron rule [Eq. (7)] modified to accept reward/penalty signals rather than desired responses.[22]

The distinction between this version of the perceptron rule and that given above in the Subsection "Perceptron element" is so slight that several authors have described the reinforcement learning version without noting its difference from Rosenblatt's original form (e.g., Minsky and Papert 1969). Unfortunately, this special case of the A_{R-P} algorithm does not satisfy the conditions required for the A_{R-P} convergence theorem since the noise distribution function is not continuous and strictly monotonically increasing (it is a step function). It performs very poorly in tasks that fall anywhere in contingency space except the upper left and lower right corners that correspond to deterministic tasks.

Learning automata

Although learning automata are not typically cast as neuron-like adaptive elements, a number of interesting connections can be pointed out by relating the A_{R-P} element, as well as some of the other elements described above, to various classes of learning automata (Narendra and Thathachar 1974). Learning automata are designed for nonassociative versions of the associative reinforcement learning task that we described in the Section "A self-interested adaptive element". Consequently their theory addresses the problem of decision making under uncertainty but not the problem of forming optimal associative mappings. Specifically, the associative reinforcement learning task reduces to the task focussed upon by learning automaton theorists if the set X of possible stimulus patterns contains just a single pattern. This means that the learning system always senses the same input pattern and is not required to form any discriminations. This kind of task corresponds to a single point in contingency space.

The class of learning automata relevant here consists of

[21] It is not literally the deterministic specialization of the A_{R-P} element since if the random numbers in Eq. (1) are all equal to zero, then $E\{y(t)|s(t)\} = y(t)$ rather than $s(t)$ as in Eq. (9)

[22] For the case in which the A_{R-P} element uses the logistic distribution [Eq. (3)], the A_{R-P} element becomes the perceptron element when T is zero

"variable-structure stochastic" learning automata, which can be described as methods for updating action probabilities. Suppose that on each trial the automaton can perform one action from the set $y^{(1)}, \ldots, y^{(m)}$. At each trial t, the automaton selects an action $y(t)$ according to a probability vector $(p_t^{(1)}, \ldots, p_t^{(m)})$, where $p_t^{(i)} = Pr\{y(t) = y^{(i)}\}$, $1 \leq i \leq m$. These automata implement a common-sense notion of reinforcement learning: if action $y^{(i)}$ is chosen and the environment's feedback indicates reward, then $p^{(i)}$ is increased and the probabilites of the other actions are decreased; whereas if the feedback indicates penalty, then $p^{(i)}$ is decreased and the probabilities of the other actions are adjusted. Many methods that have been studied are similar to the following *linear reward-penalty* (L_{R-P}) method, which was proposed for the case of two actions by Bush and Mosteller (1951a):

If $y(t) = y^{(i)}$ and the resultant evaluation is reward (i.e., $r(t) = +1$), then

$$p_{t+1}^{(i)} = p_t^{(i)} + \alpha(1 - p_t^{(i)})$$

$$p_{t+1}^{(j)} = (1 - \alpha) p_t^{(j)}, j \neq i.$$

If $y(t) = y^{(i)}$ and $r(t) = -1$ (penalty), then

$$p_{t+1}^{(i)} = (1 - \beta) p_t^{(i)}$$

$$p_{t+1}^{(j)} = \frac{\beta}{m-1} + (1 - \beta) p_t^{(j)}, j \neq i,$$

where $0 < \alpha, \beta < 1$. When $\alpha = \beta$, the algorithm is the symmetric L_{R-P} algorithm, and when $\beta = 0$, it is called the *linear reward-inaction* (L_{R-I}) algorithm.

Barto and Anandan (1985) show that the A_{R-P} algorithm reduces to the two-action version of this algorithm ($m = 2$) if the distribution function for the noisy threshold is a uniform distribution (i.e., each random value $\eta(t)$ in Eq. (1) is equally likely to fall anywhere between, say, -1 and $+1$), and the input pattern is held constant (and nonzero) over trials. If the distribution function is not uniform, the A_{R-P} algorithm similarly reduces to a nonlinear learning automaton algorithm. This means that an A_{R-P} element that receives, in addition to reinforcement input, just a constant threshold input, and adjusts just its threshold weight, is an example of a stochastic learning automaton. This is in fact how the A_{R-P} algorithm was designed, and the convergence theorem described in the Section "Theoretical analysis" is an extension of an existing theorem due to Lakshmivarahan (1981) for a class of stochastic learning automaton algorithms.

The selective bootstrap element and the reinforcement learning perceptron also reduce to learning automata when their input patterns are held constant. Since the resulting learning automata are deterministic, they have well-known difficulties when facing contingencies in the hard parts of contingency space. The reinforcement learning perceptron in fact reduces to the "win-stay/lose-shift" strategy (also known as the two-state Tsetlin automaton) which performs better than chance, but far from optimally, under all nondeterministic contingencies.

A final interesting connection is that the symmetric A_{R-P} algorithm ($\lambda = 1$) will "probability match" when facing contingencies in which the reward probabilities for its two actions sum to one for each stimulus pattern. This means, for example, that if the reward probabilities for performing actions $+1$ and -1 are respectively 0.7 and 0.3, then the A_{R-P} element will eventually perform action $+1$ and -1 with respective probabilities 0.7 and 0.3. This yields an overall reward probability that is better than chance but far from optimal. Probability matching was extensively studied by mathematical psychologists (see the review by Meyers 1976).

Boltzmann machines

Although we cannot point out direct relationships between learning by networks of A_{R-P} elements and learning by the "Boltzmann machine" of Hinton et al. (Ackley et al. 1985; Hinton and Sejnowski 1983), this approach is of sufficient interest in the context of stochastic cooperativity that we briefly discuss it. A similar approach has been independently developed by Smolensky (1983). A Boltzmann machine is a network of stochastic linear threshold units, each having an input/output function identical to that of an A_{R-P} element using the logistic distribution [Eq. (1) and (3)]. Unlike the networks considered here, these networks are symmetrically connected, meaning that if unit A influences unit B, then unit B influences unit A in exactly the same way. This assumption permits the application of mathematical results from statistical thermodynamics to determine the relative probability of each pattern of activation at equilibrium. If one regards the interconnection weights as specifying constraints that are to hold between the activities of pairs of units, then at equilibrium the probability of an activity pattern is higher to the extent that it simultaneously satisfies all of these constraints. There is an analogy between the degree the constraints are satisfied and the energy of a physical system – the system evolves so as to spend a higher proportion of time in low energy states. The computational temperature of the system, T, can be manipulated to affect both the time it takes the system to reach equilibrium and the equilibrium probabilities of activity patterns. This connection between statistical physics and networks of neuron-like elements is due to Hopfield (1982), with an earlier but less specific connection being made by Cragg and Temperley (1954). The computational process of obtaining equilibrium probabilities is described by Kirkpatrick et al. (1983) and by Geman and Geman (1984).

The learning algorithm proposed by Hinton and Sejnowski (1983) is of interest here because it addresses the problem of assigning credit to the hidden elements. The training paradigm is similar to the supervised-learning paradim discussed above for the Widrow/Hoff and perceptron elements. Here, however, only the visible elements of the network are directly told what they should be doing. It turns out that the gradient of overall network performance with respect to any weight, even an interior weight, can be determined using only the behavior of the two elements the weight connects *provided* this behavior is measured when the network is operating according to the equilibrium probability distribution. The learning procedure then changes weights according to this gradient by a stochastic hill-climbing method. Although it can be proved that the gradient of the global performance index can be determined locally in this manner, it is not necessarily true that the stochastic hill-climbing procedure is

always capable of avoiding false peaks. Nevertheless, simulation experiments show that the process does tend to work, but as in the case of networks of A_{R-P} elements, there are many unanswered questions about the amount of time needed for learning and how it increases as problems become harder. See Ackley et al. (1985) for the most complete discussion of this learning process.

Despite some superficial similarities, networks of A_{R-P} elements and Boltzmann machines are quite different. First, the units of Boltzmann machines are not self-interested components that learn to cooperate in the sense of game theory. Second, the networks of A_{R-P} elements we have simulated so far have all been layered networks without recurrent connections, whereas the Boltzmann learning procedure is restricted to symmetrically connectioned, hence totally recurrent, networks. In a layered network the entire stage corresponding to the running of a Boltzmann network or Harmony system (Smolenski 1983) to equilibrium 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 subtle constraint satisfaction problems. On the other hand, once a layered network has learned, its performance in computing this function is essentially instantaneous. 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.

As the preceding comparisons show, the A_{R-P} element is closely related to a number of existing adaptive elements and learning algorithms. Under one set of restrictions, it specializes to more conventional deterministic adaptive elements, such as those embodying the perceptron algorithm, that are designed to form linear associative mappings when explicit and reliable training information is available. Through this direction of specialization, the A_{R-P} element makes contact with models of animal behavior in classical conditioning such as the Rescorla/Wagner model. Under another set of restrictions, the A_{R-P} element specializes to stochastic learning algorithms that are capable of improving their performance under response contingencies that are disguised by high degrees of noise. Through this connection, the A_{R-P} element makes contact with both the stochastic automaton algorithms developed by engineers and the stochastic learning models of mathematical psychology. Consequently, the A_{R-P} element lies in the intersection of important classes of algorithms developed within theoretical traditions that have remained largely separate. Although we think that these theoretical connections are interesting for their own sake, our major interest in A_{R-P} elements, and similar elements, is a result of their ability to learn to cooperate with one another as components of multilayered adaptive networks. The various theoretical threads brought together in the A_{R-P} element complement one another in ways critical to this capability.

These properties of the A_{R-P} element suggest that it may be a good candidate for careful empirical investigation as a model of associative learning at the cellular level. From a broad perspective, the A_{R-P} mechanism is Hebbian since it bases synaptic change on both pre- and postsynaptic signals. Consequently, the empirical support, or lack of it, for Hebbian synapses is relevant to the status of the A_{R-P} element as a neuronal model. Viana di Prisco (1984) provides a good review of the current state of the evidence for Hebbian synapses, and the overall picture is far from compelling. However, the A_{R-P} model would suggest that the conditions for synaptic change in associative learning are considerably more stringent than those required by the Hebbian postulate. That the conditions for synaptic change in associative learning are more stringent than those suggested by the Hebbian postulate may provide an explanation for the relative difficulty in obtaining synaptic changes with the usual experimental manipulations designed to test the Hebbian postulate.

On the other hand, it may not be correct to associate an A_{R-P} element with a neuron in the manner suggested by Figure 1. For example, the influence of postsynaptic activity on synaptic modification required by the A_{R-P} algorithm could be mediated by pathways external to the neuron that carry information about post-synaptic activity to the presynaptic terminals. This implementation of the A_{R-P} algorithm would be similar to one suggested by Sutton and Barto (1981) for a different algorithm. Alternatively, the processes required by the A_{R-P} algorithm could occur at an organizational grain *finer* than entire neurons, for example, at the level of groups of adjacent synapses. These possibilities suggest that although the A_{R-P} element is "neuron-like" in the tradition of the adaptive elements described in this section, its specific features may not map in a literal way onto actual neurons. We would hope, however, that the principles of learning embodied in the A_{R-P} algorithm would survive various interpretations.

Discussion

Cooperativity in neural networks undoubtedly takes many forms, some of which are surely represented in the mathematical models and computer simulations to which the label *cooperative computation* is usually applied. The research described in this article is an attempt to add another level of meaning to computational cooperativity by starting with elemental units having preferred inputs and means for learning how to obtain them in a variety of environments. Whereas a set of computational units whose activity is mutually supporting through excitatory interactions may be usefully regarded as a coalition, a set of self-interested adaptive units in a similar arrangement may be understood to have entered into this configuration because it furthered their individual interests to do so. We therefore gain some sense both of how the coalition formed and why it is maintained.

A major factor affecting the ability of A_{R-P} elements to learn as hidden elements of networks is their use of randomness. Spontaneous random activity gives A_{R-P} elements the ability to improve performance under response contingencies falling in arbitrary regions of contingency space. We have argued that this ability is essential for network self-organization because each component's environment implements time-varying contingencies that cannot be guaranteed to remain within the "easy" regions of contingency space. Although we have not demonstrated it here, deterministic elements such as those reviewed in the Section "Relationship of the A_{R-P} element . . ." are not able to do this and fail as adaptive network components except in

simple cases. Intuitively, the random component of an A_{R-P} element's activity provides variety that causes the activity of a network to explore more thoroughly the space of activity patterns. Deterministic elements do not test enough possibilities before they stop learning. As an A_{R-P} element's weights increase in magnitude, its behavior becomes more deterministic in a way carefully controlled to prevent the inferior actions from dominating.

The simulations presented here show that 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. It is commonplace to postulate the existence in nervous systems of command hierarchies in which high-level commands are directed to lower-level processes that are able to carry them out. The form of learning with which we have been concerned provides a means for a subordinate process to acquire the knowledge required to carry out such commands. The "what" knowledge of the superordinate center is sufficient to generate evaluative feedback for its subordinate but is not sufficient to provide detailed instruction.

Network learning algorithms capable of learning under these conditions in an efficient and reliable way have remained elusive despite considerable effort over the last thirty years. We can by no means claim that the approach described here solves all of the problems. We have not extended the A_{R-P} convergence theorem to networks of A_{R-P} elements, and we have not yet answered the critical questions concerning the rate of learning and how it changes as the networks become larger. It is relatively easy to devise algorithms that are guaranteed to find optimal solutions by "brute-force" search of the space of all possible weight values. The point is to do it much more quickly than these, or more quickly on the average, while probably settling for solutions that are sufficiently good but not necessarily optimal. We are currently in the process of performing comparative simulation studies that will allow us to suggest answers to these questions. At present we only know that for relatively small networks the simplest brute-force search generally yields almost no improvement in performance by the time the A_{R-P} networks are performing near optimally.

As we mentioned in the introduction, our research is an attempt to study networks of adaptive elements possessing *enough but not more* behavioral sophistication to allow them to learn to enter into cooperative relationships with other elements like themselves. It is therefore driven by computational, rather than biological, issues. But an assumption common to nearly all theoretical studies of networks of neuron-like computing elements is that insight can be gained into biological mechanisms by studying the computational problems they apparently solve and by proposing mechanisms that are compatible with presumed biological constraints (Marr and Poggio 1977). We believe that the problem of learning by the "hidden" elements of a network is fundamental whether the network is of natural origin or is man-made. It is hard to imagine where explicit instructional information might come from for all of the neural units involved in long-term adaptation to unforeseen changes in the demands made on an organism's behavior. On purely logical grounds, the credit-assignment problem in its various forms seems inescapable. The strategy illustrated by our research, in which credit assignment is accomplished through the cooperative interaction of stochastic self-interested adaptive components, appears to have promise both as a technique in man-made networks and as a hypothesis about neural networks. Provided the components are sufficiently robust, reliable learning is a more or less natural consequence of the interaction of self-interested adaptive components. However, as we have attempted to make clear, the required degree of robustness is not easy to achieve, and has not been achieved by neuron-like computational elements studied in the past.

It is fitting to close by paraphrasing a remark made by the evolutionary biologist G. C. Williams (while discussing populations of organisms, 1966): We urge the reader to maintain a conceptual distinction between a network of adapted components and an adapted network of components. The behavioral success of the networks we have presented is an incidental consequence of the adaptations by which each component attempts to improve its own performance – they are networks of adapted components. It is not at all paradoxical that the behavioral capabilities of such a network can far outstrip the behavioral capabilities of any of its constituent elements.

Acknowledgements. 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. The author thanks C. W. Anderson, whose research with networks has been indispensible and who performed some of the simulations reported here; R. S. Sutton, whose theoretical and experimental skills have contributed in many ways; and P. Anandan, who is largely responsible for the convergence theorem stated here. Additional thanks are due to J. S. Judd and B. Pinette for their participation in discussions of this work; A. H. Klopf, whose hypotheses have helped shape this research; and M. A. Arbib and J. W. Moore for their helpful discussions and expert guidance through the literature.

References

Ackley DH, Hinton GE, Sejnowski TJ (1985) A learning algorithm for Boltzmann machines. Cogn Sci 9:147–169

Allanson JT (1956) Some properties of randomly connected neural networks. In: Cherry C (ed) Information theory. Butterworth, London

Amari S (1974) A method of statistical neurodynamics. Kybernetik 14:201–215

Amari S, Arbib MA (1977) Competition and cooperation in neural nets. In: Metzler J (ed) Systems neuroscience. Academic Press, New York

Amari S, Arbib MA (eds) (1982) Competition and cooperation in neural nets. Springer, Berlin Heidelberg New York

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

Anderson JA, Silverstein JW, Ritz SA, Jones RS (1977) Distinctive features, categorical perception, and probability learning: some applications of a neural model. Psychol Rev 85:413–451

Atkinson RC, Bower GH, Crothers EJ (1965) An introduction to mathematical learning theory. Wiley, New York

Atkinson RC, Estes WK (1963) Stimulus sampling theory. In: Luce RD, Bush RR, Galanter E (eds) Handbook of mathematical psychology, vol II. Wiley, New York

Barto AG (ed) (1984) Simulation experiments with goal-seeking adaptive elements. Air Force Wright Aeronautical Laboratories/ Avionics Laboratory Technical Report AFWAL-TR-84-1022, Wright-Patterson AFB, Ohio

Barto AG, Anandan P (1985) Pattern recognizing stochastic learning automata. IEEE Transactions on Systems, Man, and Cybernetics 15:360–375

Barto AG, Sutton RS (1981a) Goal seeking components for adaptive intelligence: an initial assessment. Air Force Wright Aeronautical Laboratories/Avionics Laboratory Technical Report AFWAL-TR-81-1070, Wright-Patterson AFB, Ohio

Barto AG, Sutton RS (1981b) Landmark learning: an illustration of associative search. Biol Cybern 42:1–8

Barto AG, Sutton RS, Anderson CW (1983) Neruonlike elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, and Cybernetics 13:835–846

Barto AG, Sutton RS, Brouwer PS (1981) Associative search network: a reinforcement learning associative memory. Biol Cybern 40:201–211

Beurle RL (1956) Properties of a mass of cells capable of regenerating pulses. Phil Trans Roy Soc (Lond) B240:231–277

Brandon RN, Burian RM (eds) (1984) Genes, organisms, populations: controversies over the units of selection. MIT Press, Cambridge, MA

Brooks R (1983) Experiments in distributed problem solving with interative refinement. Univ of Massachusetts Ph D Dissertation, Dept of Computer and Information Science

Bush RR, Estes WK (eds) (1959) Studies in mathematical learning theory. Stanford University Press, Stanford

Bush RR, Mosteller F (1951a) A mathematical model for simple learning. Psychological Review 58:313–323 (Reprinted in Luce RD, Bush RR, Galanter E (eds) (1963) Readings in mathematical psychology, vol 1. Wiley, New York, pp 278–288)

Bush RR, Mosteller F (1951b) A model for stimulus generalization and discrimination. Psychol Rev 58:413–423 (Reprinted in Luce RD, Bush RR, Galanter E (eds) (1963) Readings in mathematical psychology, Vol 1. Wiley, New York, pp 289–299)

Bush RR, Mosteller F (1955) Stochastic models for learning. Wiley, New York

Cover TM (1968) A note on the two-armed bandit problem with finite memory. Inf Control 12:371–377

Cover TM, Hellman ME (1970) The two-armed bandit problem with time-invariant finite memory. IEEE Transactions on Information Theory 16:185–195

Cragg BG, Temperley HNV (1954) The organization of neurones: a cooperative analogy. EEG Clin Neurophysiol 6:85–92

Crane HD (1978) Beyond the seventh synapse: the neural marketplace of the mind. SRI Research Memorandum, Stanford Research Institute, Menlo Park, CA

Dawkins R (1976) The selfish gene. Oxford University Press, Oxford

Dawkins R (1982) The extended phenotype. Freeman, Oxford

Dev P (1974) Segmentation process in visual perception. International Journal of Man-Machine Studies 7

Didday RL (1976) A model of visuomotor mechanisms in the frog optic tectum. Math Biosci 30:169–180

Duda RO, Hart PE (1973) Pattern classification and scene analysis. Wiley, New York

Estes WK (1950) Toward a statistical theory of learning. Psychol Rev 57:94–107

Farley BG, Clark WA (1954) Simulation of self-organizing systems by digital computer. IRE Transactions on Information Theory 4:76–84

Feldman JA (1982) Dynamic connections in neural networks. Biol Cybern 46:27–39

Feldman JA (ed) (1985) Special issue on connectionist models and their applications. Cogn Sci 9

Fukushima K (1980) Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern 36:193–202

Geman S, Geman D (1983) Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 6

Grossberg S (1978) Competition, decision, concensus. Journal of Mathematical Analysis and Applications 66:470–493

Grossberg S (1980) Biological competition: decision rules, pattern formation, and oscillations. Proceedings of the National Academy of Science 77:2338–2342

Harth E, Csermerly TJ, Beek B, Lindsay RD (1970) Brain functions and neural dynamics. J Theor Biol 26:93–120

Harth E (1976) Visual perception: a dynamic theory. Biol Cybern 22:169–180

Hebb DO (1949) The organization of behavior. Wiley, New York

Hinton GE (1981) Implementing semantic networks in parallel hardware. In: Hinton G, Anderson J (eds) Parallel models of associative memory. Erlbaum, Hilsdale, NJ, pp 161–187

Hinton GE, Anderson J (1981) Parallel models of associative memory. Erlbaum, Hilsdale, NJ

Hinton GE, Sejnowski TJ (1983) Analyzing cooperative computation. Proceedings of the Fifth Annual Conference of the Cognitive Science Society, Rochester NY

Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, Michigan

Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Science 79:2554–2558

Julesz B (1971) Foundations of cyclopean perception. University of Chicago Press, Chicago

Kilmer WL, McCulloch WS, Blum J (1969) A model of the vertebrate central command system. Int Man-Mach Stud 1:279–309

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680

Klopf AH (1974) 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, Dallas, Texas

Klopf AH (1982) The hedonistic neuron: a theory of memory, learning, and intelligence. Hemisphere, Washington, DC

Kohonen T (1977) Associative memory: a system theoretic approach. Springer, Berlin Heidelberg New York

Kurose JF, Schwartz M, Yemini Y (1985) A microeconomic approach to distributed resource sharing in computer communication networks. 5th International Conference on Distributed Computing Systems. IEEE Computer Society, Denver

Kubie JL, Rank JB Jr (1983) Sensory behavioral correlates in individual hippocampal neurons in three situations: space and context. In: Seifert W (ed) Neurobiology of the hippocampus. Academic Press, New York, pp 433–447

Lakshmivarahan S (1981) Learning algorithms and applications. Springer, Berlin Heidelberg New York

Lesser V, Corkill D (1981) Functionally accurate cooperative distributed systems. IEEE Transactions on Systems, Man, and Cybernetics, 11

Luce RD, Raiffa H (1957) Games and decisions, Wiley, New York

MacGregor RJ, Lewis ER (1977) Neural modelling. Plenum Press, New York

Mackintosh NJ (1983) Conditioning and associative learning. Oxford University Press, New York

Marr D, Poggio T (1976) Cooperative computation of stereo disparity. Science 194:283–287

Marr D, Poggio T (1977) From understanding computation to understanding neural circuitry. Neurosci Res Progr Bul 15:470–488

Marshak J, Radner R (1972) Economic theory of teams. Yale University Press, New Haven

Minsky ML (1954) Theory of neural-analog reinforcement systems and its application to the brain-model problem. Princeton Univ Ph D Dissertation

Minsky ML (1961) Steps toward artificial intelligence. Proceedings of the Institute of Radio Engineers 49:8–30. (Reprinted in Feigenbaum EA, Feldman J (eds) (1963) Computers and thought. McGraw-Hill, New York, pp 406–450)

Minsky ML, Papert S (1969) Perceptrons: an introduction to computational geometry. MIT Press, Cambridge, MA

Minsky ML, Selfridge OG (1961) Learning in random nets. Information Theory, Fourth London Symposium. Butterworths, London

Moore GP, Perkel DH, Segundo JP (1966) Statistical analysis and functional interpretation of neuronal spike data. Ann Rev Physiol 28:493–522

Meyers JL (1976) Probability learning and sequence learning. In: Estes WK (ed) Handbook of learning and cognitive processes, Vol 3. Erlbaum, Hillsdale, NJ, pp 171–205

Narendra KS, Thathachar MAL (1974) Learning automata – a survey. IEEE Transactions on Systems, Man, and Cybernetics 4:323–334

Neumann J von (1956) Probabilistic logics and the synthesis of reliable organisms from unreliable components. In: Shonnon CE, McCarthy J (eds) Automata studies. Princeton University Press, Princeton, NJ, pp 43–98

Neumann J von, Morgenstern O (1953) Theory of games and economic behavior. Princeton University Press, Princeton, NJ
Nilsson NJ (1965) Learning machines. McGraw-Hill, New York
Rescorla RA, Wagner AR (1972) A theory of Pavlovian conditioning: variations in the effectiveness of reinforcement and non-reinforcement. In: Black AH, Prokasy WF (eds) Classical conditioning II: current research and theory. Appleton-Century-Crofts, New York
Restle F (1955) A theory of discrimination learning. Psychol Rev 62:11–19
Robbins H (1952) Some aspects of the sequential design of experiments. Bull Am Math Soc 58:527–532
Rosenblatt F (1962) Principles of neurodynamics. Spartan Books, New York
Smolenski P (1983) Harmony theory: a mathematical framework for stochastic parallel processing. Proceedings of the National Conference on Artificial Intelligence AAAI-83, Washington, DC
Staddon JER (1983) Adaptive behavior and learning. Cambridge University Press, Cambridge
Sutton RS (1984) Temporal aspects of credit assignment in reinforcement learning. University of Massachusetts Ph D Dissertation
Sutton RS, Barto AG (1981) Toward a modern theory of adaptive networks: expectation and prediction. Psychol Rev 88:135–171
Szentágothai J, Arbib MA (1974) Conceptual models of neural organization. Neurosci Res Progr Bul 12:3
Thorndike EL (1911) Animal intelligence. Hafner, Darien, Conn
Tsetlin ML (1973) Automation theory and modelling of biological systems. Academic Press, New York
Uttley AM (1965) The probability of neural connexions. Proc Roy Soc (Lond) B144:229–240
Varshavsky VI (1968) Collective behavior and control problems. In: Michie D (ed) Machine intelligence, Vol 3. American Elsevier, New York
Varshavsky VI (1972) Some effects in the collective behavior of automata. In: Meltzer B, Michie D (eds) Machine intelligence, Vol 7. Wiley, New York
Viana Di Prisco G (1984) Hebb synaptic plasticity. Prog Neurobiol 22:89–102
Widrow B (1962) Generalization and information storage in networks of adaline "neurons". In: Yovits M, Jacobi G, Goldstein G (eds) Self-organizing systems. Spartan Books, New York
Widrow B, Gupta NK, Maitra S (1973) Punish/reward: learning with a critic in adaptive threshold systems. IEEE Transactions on Systems, Man, and Cybernetics 5:455–465
Widrow B, Hoff ME (1960) Adaptive switching circuits. 1960 WESCON Convention Record Part IV, pp 96–104
Williams GC (1966) Adaptation and natural selection. Princeton University Press. Princeton. (Reprinted in part in Brandon RN, Burian RM (eds) (1984) Genes, organisms, populations: controversies over the units of selection. MIT Press, Cambridge, MA, pp 52–68)
Wilson HR, Cowan JD (1972) Excitatory and inhibitory interactions in localized populations of neurons. Biophys J 12:1–24
Yemini Y (1982) Selfish optimization in computer networks. Proceedings of the 20th IEEE Conference on Decision and Control. IEEE Press, Piscataway, NJ, pp 281–285
Yemini Y, Kleinrock L (1979) On a general rule for access control or, silence is golden . . . Proceedings of the International Symposium on Flow Control in Computer Networks. North Holland Press, Amsterdam, pp 335–347