IN: PROCEEDINGS OF THE SEVENTH ANNUAL CONF. OF THE COGNITIVE SCIENCE SOC. 1985

THE LEARNING OF WORLD MODELS BY CONNECTIONIST NETWORKS*

Richard S. Sutton
GTE Labs, Waltham, MA 02254

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

Abstract—Connectionist learning schemes have hitherto seemed far removed from cognitive skills such as reasoning, planning, and the formation of internal models. In this article we investigate what sort of world models a connectionist system might learn and how it might do so. A learning scheme is presented that forms such models based on observed stimulus-stimulus relationships. The basis of the scheme is a recurrently connected network of simple, neuron-like processing elements. The net produces a set of predictions of future stimuli based on the current stimuli, where these predictions are based on a model and involve multiple-step chains of predictions. Results are presented from computer simulations of the scheme connected to a simple world consisting of a stochastic maze (Markov process). By wandering around the maze the network learns its construction. When reinforcement is subsequently introduced, the solution to the maze is learned much more quickly than it is without the "exploration" period. The form and logic of the experiment is the same as that of the *latent learning* experiments of animal learning research.

World Models and Connectionism

Many of the predominant high-level theories of cognition give a central role to internal models (e.g., see Gelertner and Gerstenhaber, 1956; Craik, 1943; Dennett, 1978; Simon, 1969; or Sutton and Barto, 1981, for a review). Their central idea is that much of what we mean by "thought" can be understood as the formation, modification, and use of internal models of the world. Here, by internal models we mean cognitive structures that allow us to predict the future and to anticipate the results of possible actions we might take. In particular, we mean models that allow chains of anticipation and inference, e.g., "if I do A then B will occur, and if B, then C will occur, etc."

In connectionist networks, all concepts (e.g., perceptions of the environment, stimuli and actions) are represented as activity in the elements of the network. In other words, the activity of the network acts as a static model of the current situation. The natural place to model the dynamics of the environment is in the connections between elements: if stimulus or action A is always follows by stimulus B, then a strong connection from A to B models

* 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 authors also wish to thank Andy Barto, Chuck Anderson, John Moore, and Martha Steenstrup for discussing and contributing to these ideas.

this fact in the sense that it causes the evolution of activity in the network to mimic the evolution of states in the external environment. Of course the network's evolution will be much quicker than the environment's, so its predictions will be anticipatory and thereby valuable for directing behavior.

That the behavior of activity within a network might be used to model the behavior of the world is a simply stated idea with many ramifications. The advantages are apparent: besides inheriting the parallelism and mechanistic advantages of connectionism, the theory is also extremely parsimonious, since world modeling is achieved using the same apparatus—the connections of a network—that is used to implement simple stimulus-response mappings. Connectionist internal models could conceivably form a sort of "missing link" between cognitive processes such as reasoning and planning on the one hand, and stimulus-response learning and neuroscience on the other.

Problems with the idea are also apparent. If a network's connections are used to implement the dynamics of an internal model, how can they also be used to implement stimulus-response mappings? Or how can they also be used to implement the structures necessary for recognizing complex concepts and executing complex actions? In addition, models can easily involve self-referential chains, e.g., "A is followed by B which is followed by A again." If a network forms cycles of excitatory connections in order to model such an environment, it courts the danger of positive feedback and instability. The results presented here deal with the self-reference issue, but not with the issue of how to integrate the different functions of a network. We assume that the other functions are completely separate from the network or sub-network that learns and embodies the internal model.

From the point of view of artificial intelligence, where the use of internal models is commonplace, the primary advantage of connectionist models is that they are well-suited for learning. Standard AI techniques include sophisticated methods for searching a model efficiently when the model is completely defined and known. In practice, the model is often only partially known, and search methods for such cases are less well understood. What is needed is the ability both to learn a model and to search it as it is being learned. Connectionist structures seem to be well suited to learning methods, particularly in uncertain and noisy domains. As such they suggest an attractive direction in which to look for ways of learning searchable models.

To limit the scope of this paper, we pay little attention to the issues involved in how a connectionist net should best use a model, and instead concentrate on how it maybe acquired. We also concentrate on capturing inferences that are not dependent on the actions selected, that is, we focus on stimulus-stimulus rather than response-stimulus inferences. A network and learning procedure are presented which can learn to mimic environments, including multiple-step causal chains, as outlined above. The learning procedure, though deterministic, is similar to the stochastic learning procedure used in the Boltzmann machine (Ackley, Hinton, and Sejnowski, 1985) in that it involves letting a recurrent network run to equilibrium in response to two different stimulus patterns.

There are numerous ties between this work and animal learning theory. The learning involved in building an internal model is analogous to the S-S learning effectively advocated by Tolman and others in response to the rise of behaviorism (e.g., see Hilgard and Bower, 1975). The final simulation experiment reported here is directly analogous to their "latent learning" experiments, in which it was shown that animals learn extensively about their environment even if never rewarded or punished, the learning remaining latent and unexpressed in behavior until reinforcement is introduced. The ties between this work and modern animal learning theory are even stronger. Contemporary animal learning theory, particularly that part concerning classical or Pavlovian conditioning, has a very cognitive orientation (e.g., see Dickinson, 1980). Many of its results bear directly on the issues of how a declarative internal model is learned by animals. The learning scheme used here is a descendant of the model of classical conditioning developed by Sutton and Barto (1981), pursuing the network approach illustrated by Moore and Stickney (1980).

Recurrent Networks and Markov Worlds

The network used here is of the simplest of connectionist designs. Each element receives as input the activity of every element and an external input from outside the net. These external stimuli are the variables to be predicted. They correspond to the stimuli A, B, etc., mentioned earlier, while the connections within the net are meant to model the world's dynamics. Accordingly, each connection between elements has a modifiable connection weight, while each external input has a direct and unmodifiable effect on the activity of its corresponding element. The activity of the network is updated according to

$$y' = x + Vy \tag{1}$$

where y is the old activity vector, y' is the new activity vector, x is the external stimulus vector, and V is the matrix of modifiable connection weights. The process given by (1) may converge (reach equilibrium), diverge, or oscillate, depending on the weights. For the weights found by the learning scheme in our simulations, it has always converged.

A perfect stimulus-stimulus model would produce complete information about future stimuli. Given a stimulus x_t at time t, it would be able to reply with $E\{x_{t+k} \mid x_t\}$, for any $k > 0$. Consider the special case of the stimuli x_t being generated by a Markov process with state transition matrix P *, where each x_t is a vector with one component for each state, that component being 1 when the Markov process is in that state and 0 otherwise. Then $E\{x_{t+1} \mid x_t\} = Px_t$ and, in fact, $E\{x_{t+k} \mid x_t\} = P^k x_t$. Recurrent nets are good at computing such powers of matrices: If an autonomous (inputless) recurrent network with connection matrix $V = P$ is initialized to $y = x_t$, then its activity will naturally compute, in sequence, the desired $P^k x_t$.

* This is the matrix $[p_{ij}]$, where each p_{ij} is the probability of a transition from state j to state i, given that the process is in state j.

The above is a particularly clear special case, but is of little value in most practical cases. The principal problem is that this approach focuses on a single-step model of the world. It can know that A precedes B only if A *immediately* precedes B, or if distinct stimuli C, D, E, etc., occur for every step intervening between A and B. This approach forces the model to be an extremely local, myopic one, while the major observable regularities inevitably appear at a variety of time scales. Alternatively, we can imagine learning separately the 1-step, 2-step, 3-step, etc., transition probabilities for the environment, and then using each to answer specific questions. However, this approach violates the whole idea of making multiple-step predictions by combining separate predictive steps. In addition, it is not clear that all this information is needed. Do we need to know what is going to happen at each future time step separately, or can some of this information by merged without significant loss? To answer this question we must turn to that of how the predictions are going to be used.

The ultimate use of the model will be to answer questions of the form "how desirable are the consequences of this possible action?" and "now that I've made an action, how well does it appear to have turned out?" One consequence of this is that we will not be so concerned with making predictions of what will happen at specific future times as much as we we will be concerned with the sum of future predictions. Rather than predicting that "A will probably happen at $t+k$," we will want to predict "A will probably happen n times in the near future." Rather than predicting x_{t+k} we wish to predict something more like $\sum_{n=1}^{k} x_{t+n}$. A similar approach, but which avoids an abrupt cutoff at an arbitrary time limit $t+k$, is to predict $\sum_{n=1}^{\infty} \gamma^n x_{t+n}$, where $0 \leq \gamma < 1$. This measure adds up all future stimuli x_{t+k}, weighting each according to γ^k, i.e., the farther in the future a stimulus will occur, the less it contributes to the sum. The value of γ determines how rapidly the weighting of future stimuli drops off.

An estimate of $\sum_{n=1}^{\infty} \gamma^n x_{t+n}$ is also readily computed by a recurrent network if its connection matrix incorporates the environment's probability transition matrix. The equilibrium value for the process given by (1) is $\sum_{n=0}^{\infty} V^n x_t$ (taking $x = x_t$). If V is a fraction γP of the probability transition matrix P of a Markov process, then this sum is $E\left\{\sum_{n=0}^{\infty} \gamma^n x_{t+n}\right\}$, from which the desired quantity, $E\left\{\sum_{n=1}^{\infty} \gamma^n x_{t+n}\right\}$ can be immediately obtained by subtracting x_t. In other words, if V is a fraction of the one-step predictive relationships of the environment, then the network will converge to predictions of how often each stimulus will occur in the near future.

A major advantage of being concerned only with the sum of all future predictions is that one can easily incorporate predictions that span several steps and that cannot be made as chains of one-step predictions. Recall that these are needed when A precedes B with a "gap" of one or more time steps with no distinct stimuli filling the gap. Let us call a prediction "irreducible" if it cannot be reduced to a chain of shorter predictions. All single-step predictions are irreducible predictions, and, as we discussed above, for the special case of Markov environments that reveal their state completely, these are the only irreducible predictions. In almost any real problem, however, multi-step irreducible predictions are common. .

Irreducible predictions have a nice relationship to sums of future predictions. As a generalization of the one-step prediction matrix P, we propose the irreducible prediction matrix $\hat{P}$, whose elements $\hat{p}_{ij}$ represent the irreducible prediction of the ith input from the presence of the jth input. We conjecture that as long as one is interested only in the sum of future predictions one can replace P with $\hat{P}$, i.e., that $\sum_{n=0}^{\infty} P^n x = \sum_{n=0}^{\infty} \hat{P}^n x$, $\forall x$.* If one is discounting, things are slightly more complicated. In order for $\sum_{n=0}^{\infty} \hat{P}^n x$ to match $\sum_{n=0}^{\infty} \gamma^n P^n x$, the irreducible predictions in $\hat{P}$ should be discounted according to their length, i.e., one-step predictions should be discounted by γ, two-step irreducible predictions by γ^2, etc. An example is given below as part of the simulation results.

The Learning Procedure

At each new time step $t+1$ the environment generates a new stimulus vector x_{t+1}. How can this information be used to update the predictions made by the net at time t using x_t? Within each time step the network is run completely to equilibrium (by repeated application of (1)) for both x_t and x_{t+1}; let us denote the equilibrium value for the net's activity in response to x_t as z, and the equilibrium value in response to x_{t+1} as z'. What is the desired relationship between z and z'? Recall that z is supposed to estimate the discounted sum of future x's:

$$z \approx \sum_{n=0}^{\infty} \gamma^n x_{t+n}.$$

Similarly, $z' \approx \sum_{n=0}^{\infty} \gamma^n x_{t+n+1}$, so z's desired value can be rewritten in terms of z' as

$$z \approx x_t + \gamma z'.$$

The difference between z and $x_t + \gamma z'$ can be taken as an error in z and used to update the connection matrix V. According to the learning scheme, each connection weight is incremented proportionally to the product of this difference $(x_t + \gamma z' - z)$ at the post-connection element and the recent external input to the pre-connection element. This change tends to cause z to change so as to reduce the difference. Symbolically, the learning scheme is defined by:

$$\Delta V_t = \beta(x_t + \gamma z' - z)\bar{x}_t^T, \qquad V_0 = 0,$$

where β is a positive learning constant and $\bar{x}_t^T$ denotes the transpose of a vector $\bar{x}_t$ of averages of recent values of x (taking the transpose makes the vector product here an

* This should probably be taken as a formal statement of the definition of an irreducible prediction, with the conjecture being that such a matrix $\hat{P}$ exists.

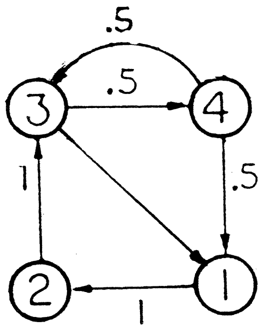
outer product). $\bar{x}_t$ is iteratively computed as follows:

$$\bar{x}_t = \lambda \bar{x}_{t-1} + (1-\lambda)x_t, \qquad \bar{x}_0 = 0$$

where the parameter λ, $0 \le \lambda < 1$, determines how strongly previous values of x influence current changes to V.

Simulation Results

Figure 1 shows the state transition structure of a simple autonomous Markov process. The numbers on the arcs indicate the probability that the indicated transition occurs; this information is also given in the state transition matrix P. Using the methods described above, a network successfully learned to model this environment. The stimuli presented to the net were unit basis vectors distinctly identifying the current state: if the environment was in state i at time t, then the ith component of x_t was 1 and all other components were 0. The parameters used were $\gamma = .5$, $\beta = .1$, and $\lambda = .5$. The activity of the network after 200 steps was used as an approximation to its asymptotic behavior. Figure 2 shows a comparison of γP and the V matrix found by the learning algorithm after 1000 steps: The match is quite good; by choosing β smaller and taking more steps we were able to make V approach γP as closely as we desired.



$$P = \begin{pmatrix} 0 & 0 & .5 & .5 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & .5 \\ 0 & 0 & .5 & 0 \end{pmatrix}$$

Figure 1. State transition structure and matrix of a simple autonomous Markov process. Each circle represents a state of the process and each arc a possible state transition. The numbers of the arcs indicate the probability of the state transition.

Next we introduced a "gap" in the environment. The stimulus vector for State 2 was changed to 0, so that nothing could be associated with it. As a result, the two-step predictive relationship between State 1 and State 3 became irreducible. Figure 3 shows the new matrix found. Since stimulus component 2 no longer occurs, nothing has become associated with it. The associations from 1 to 2 and from 2 to 3, which were formerly each of strength γ, have been replaced by a single association from 1 to 3 of strength γ^2. This two-step, direct association from 1 to 3 compensates for the network's inability to form

$$\gamma P = \begin{pmatrix} 0 & 0 & .25 & .25 \\ .5 & 0 & 0 & 0 \\ 0 & .5 & 0 & .25 \\ 0 & 0 & .25 & 0 \end{pmatrix} \qquad V = \begin{pmatrix} .00 & .00 & .31 & .27 \\ .50 & -.01 & .01 & -.01 \\ .00 & .51 & .01 & .25 \\ .00 & -.01 & .17 & -.02 \end{pmatrix}$$

Figure 2. Comparison of γP and the found connection matrix V for the environment shown in Figure 1.

an indirect association from 1 to 3 by way of the now undetectable State 2. As a result, the predictions from all states other than State 2 are exactly the same as they were when State 2 was observable. Figure 4 compares the ideal total prediction matrix $\sum_{n=1}^{\infty} \gamma^n P^n$ with the total prediction matrix $\sum_{n=1}^{\infty} V^n$ found by the network.

$$V = \begin{pmatrix} .00 & 0 & .26 & .26 \\ 0 & 0 & 0 & 0 \\ .25 & 0 & -.00 & .30 \\ .01 & 0 & .25 & -.01 \end{pmatrix}$$

Figure 3. The connection matrix found for the environment shown in Figure 1 when State 2 was a "gap" and could not be the basis for predictions.

Finally, we simulated the larger Markov environment shown in Figure 5, and we introduced a reinforcement learning mechanism to illustrate how a model learned by this method can be used to improve performance on reinforcement learning tasks. At each time step one of two actions, LEFT or RIGHT, was selected by the reinforcement learning system. This selection determines which of the arcs from each state shown in Figure 5 was followed; where a state has only one arc leaving it, the arc was followed independent of the action selected. The experiment had two phases, 1) a pretraining phase, during which no reinforcement was delivered and the environment was randomly explored, and 2) a test phase, during which reinforcement was delivered upon reaching state F. During the test phase, optimal performance is achieved by selecting the RIGHT action from states A, B, C, and D, thereby moving around the maze as rapidly as possible and visiting state F as

$$\sum_{n=1}^{\infty} \gamma^n P^n = \begin{pmatrix} .09 & .18 & .36 & .36 \\ .55 & .09 & .18 & .18 \\ .29 & .58 & .16 & .36 \\ .07 & .15 & .29 & .09 \end{pmatrix} \qquad \sum_{n=1}^{\infty} V^n = \begin{pmatrix} .09 & 0 & .38 & .37 \\ 0 & 0 & 0 & 0 \\ .29 & 0 & .15 & .34 \\ .08 & 0 & .28 & .08 \end{pmatrix}$$

Figure 4. Comparison of ideal and found total prediction matrices for the environment shown in Figure 1 with State 2 a "gap".

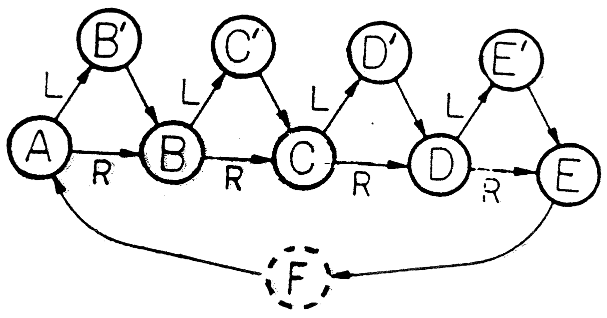


Figure 5. State transition structure of a larger Markov environment. State F is observable and reinforcing during the test phase of the experiment.

frequently as possible.

During the test phase, all states were represented in the usual way: There was one component of the stimulus vector for each state, and if the environment was in that state, then the component was 1, otherwise it was 0. A non-zero 10th component, corresponding to state F, was viewed as reinforcement by the reinforcement learning algorithm. To prevent reinforcement during the pretraining phase, the 10th component of the stimulus vector was held at 0, making state F a "gap" state.

We ran two learning systems on this experiment, called MODEL and NOMODEL. MODEL constructed an internal model during the pretraining phase and then used the model during the test phase by reinforcing its action selections according to changes in the models *predictions* of reinforcement, i.e., according to the 10th component of $x_t + \gamma z' - z$. NOMODEL did not construct a model and reinforced its action selections according to external reinforcement directly, i.e., according to the 10th component of x. Both systems select their actions by maintaining a weight w_i for each state i. When state i is entered, RIGHT is selected with probability $1/(1+e^{-w_i/T})$, T a positive parameter; otherwise

LEFT is selected. If w_i is zero (as it is initially), then this probability is 1/2. If w_i is positive then RIGHT will be favored in state i, and if w_i is negative, LEFT will be favored. Whatever action is selected in a state, the probability of repeating that action will be incremented or decremented proportionally to the reinforcement (as defined above for MODEL and NOMODEL) that follows the action selection. The size of the increment or decrement also depends on how closely the reinforcement follows the action selection; it is proportional to λ_w^k, where k is the (possibly zero) number of steps intervening between action and reinforcement. For a more complete discussion of the type of reinforcement learning algorithm employed here see (Sutton, 1984; Barto, Sutton, and Anderson, 1983).

In theory, the use of a model can be a great advantage on this task. The "wrong" actions from states A, B, C, and D all ultimately lead to the goal state, just more slowly than the correct actions. Without a model, a learning system can easily become stuck making a wrong action selection without realizing that it could do better. To avoid this, learning must be very slow. With a model, on the other hand, this problem can be completely avoided. Once reinforcement has been received in state F, a pre-existing model can make good predictions of reinforcement from every state. For example, the model would immediately predict that state C is better (closer to reinforcement) than state C', that state B is better than state B', etc. Armed with this information about the relative desirability of states, in theory a much better job can be done assigning credit and blame to action selections.

Both the pretraining and test phases lasted until the learning system had completed 100 circuits of the environment from state A back to state A. The first reinforcement occurred on the 101st circuit. For the NOMODEL learning system we used the parameters that gave the best performance on this problem. These were $T=0.5$ and $\lambda_w=0.3$. For the MODEL learning system, we chose "reasonable" parameter values without making any attempt to find the best. For the reinforcement-learning part of the system, the parameters used were $T=10^{-11}$ and $\lambda_w=0$; for the model-learning part of the system, the parameters used were $\lambda=0$, $\beta=0.01$, and $\gamma=0.8$. The activity of the network after 10 applications of (1) was used as an approximation to its asymptotic (equilibrium) activity.

We found that the MODEL system did in fact perform much better than the NOMODEL learning system on this task. Figure 6 compares the performance of the MODEL and NOMODEL learning systems on this experiment. Average performance over 10 runs is plotted for each learning system for each circuit of the test phase. The performance measure used for a circuit is the number of steps actually taken in completing the circuit. Note that lower values are better for this performance measure. The learning system without a model clearly had a difficult time with this task. Its performance improved over the duration of the test phase, but remained erratic and far from optimal. The learning system with a model, on the other hand, acheived optimal or near optimal performance after just 3 circuits on all ten runs. The use of a model results in a much higher performance level after fewer experiences with reinforcement on this task.

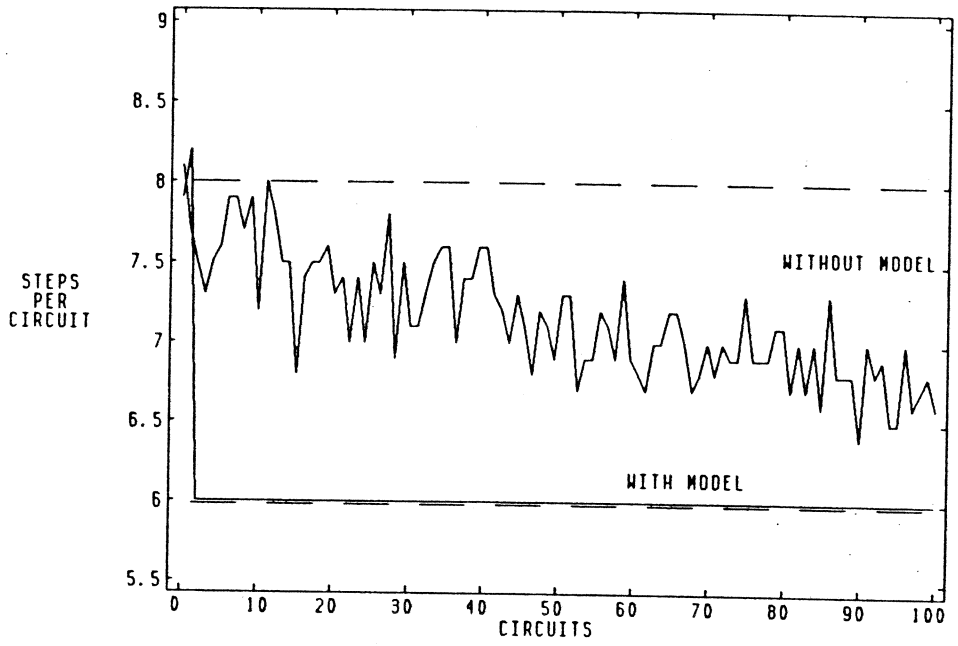


Figure 6. Average learning curves for the MODEL and NOMODEL learning systems during the test phase of the experiment whose environment is shown in Figure 5. A circuit is a complete trip from state A back to state A. The horizontal axis shows circuit number, and the vertical axis shows steps taken to complete the circuit. Note that a lower circuit time corresponds to better performance. The lower dashed line indicates the optimal (fastest) performance level, and the upper dashed line indicates the chance performance level (acheived if RIGHT and LEFT actions are chosen with equal probability from all states).

Conclusions

The modeling and internal simulation of the world is an important part of cognition, and one which the connectionist approach must eventually address. Surprisingly, in certain respects connectionism is very well suited to world modeling. A dynamically evolving network can be used as a world model by linking its activity to states of the world, as is commonly done, and by linking the evolution of that activity to the evolution of world states, as has been done here. The specific method used here to shape that the network's evolution has advantages with regard to the spanning of gaps in stimulus sequences, but is limited to forming stimulus-stimulus as opposed to response-stimulus models. We have tried to make the point that connectionist internal models are feasible, even natural, and to show an example of how they can be used to improve performance on simple learning tasks.

REFERENCES

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

Barto, A. G., Sutton R. S. & Anderson, C. W. Neuronlike elements that can solve difficult learning control problems. *IEEE Trans. on Systems, Man, and Cybernetics SMC-13*, 1983, 834–846.

Craik, K.J.W. *The Nature of Explanation.* Cambridge: Cambridge University Press, 1943.

Dennett, D.C. Why the law of effect will not go away. In: *Brainstorms.* Montgomery, Vermont: Bradford, 1978.

Dickinson, A. *Contemporary Animal Learning Theory.* Cambridge: Cambridge University Press, 1980.

Galanter, E., & Gerstenhaber, M. On thought: The extrinsic theory. *Psychological Review*, 1956, *63*, 218-227.

Hilgard, E.R., & Bower, G.H. *Theories of Learning* (fourth edition). Englewood Cliffs, New Jersey: Prentice-Hall, 1975.

Moore, J.W., & Stickney, K.J. Formation of attentional-associative networks in real time: Role of the hippocampus and implications for conditioning. *Physiological Psychology*, 1980, *8*, 207-217.

Simon, H.A. *The Sciences of the Artificial.* Cambridge, Mass.: The MIT Press, 1969.

Sutton, R. S. Temporal credit assignment in reinforcement learning. Doctoral Thesis, Dept. of Computer and Information Science, Univ. of Massachusetts, Amherst, MA, 1984.

Sutton, R.S., & Barto, A.G. An adaptive network that constructs and uses an internal model of its world. *Cognition and Brain Theory*, 1981a, *3*, 217–246.

Sutton, R.S. & Barto, A.G. Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review*, 1981b, *88*, 135–171.