A Teaching Method for Reinforcement Learning

Jeffery A. Clouse
Paul E. Utgoff
Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts 01003 U.S.A.

COINS Technical Report 92-7
January 22, 1992

Abstract

This paper presents a method for accelerating the learning rates of existing techniques for learning to solve multiple-step decision tasks. Reinforcement learning algorithms are known for their slow learning rates, and researchers have focused recently on decreasing those rates. In this paper, a method that allows a human expert to interact with a reinforcement learning algorithm is shown to accelerate the learning process. The results of the experiment, in which a human teaches a system to balance a pole, show that the teaching method significantly decreases the learning time with little input from the human expert.

Contents

1 Introduction

This paper presents a method for accelerating the learning rates of existing techniques for learning to solve multiple-step decision tasks. When learning to solve these tasks, the most difficult issue is that of credit assignment. The learning component does not receive feedback until the end of the problem solving episode. Which of the many actions that led to the success or failure should be credited or blamed for that outcome? Reinforcement learning algorithms, e.g. (Barto, Sutton & Anderson, 1983), are well-suited to this task. These algorithms allow a system to develop decision policies with sparse feedback from the environment, and do not require a model of the problem to be solved. One disadvantage of these algorithms is their slow learning rates. Whitehead (1991) proved that a particular type of reinforcement learning, unbiased Q-learning (Watkins, 1989), requires time exponential in the length of the optimal solution path to find an optimal solution and suggested that in general reinforcement learning algorithms have the same time complexity.

Recently, several researchers have presented methods for accelerating reinforcement learning (Lin, 1991; Utgoff & Clouse, 1991; Whitehead, 1991). Lin's (1991) approach provides the reinforcement learner with "lessons". The lessons are sequences of (state, action, resulting state, feedback) quadruples that begin with an initial state and proceed to a goal state. To further speed learning, the quadruples are presented to the system in reverse chronological order. Lin's method is able to learn tasks that are not learnable via a straight reinforcement learning algorithm. The learning component of Utgoff and Clouse (1991) can query an outside agent as to the correct action to take when its confidence in its own decisions is low. This learning component solved the domain task much more quickly with the help of the agent than without it. Whitehead (1991) suggests two methods for accelerating learning. One is called Learning with an External Critic, where an external agent provides immediate feedback to the learner for every one of its actions; and the other is Learning by Watching, where the learner gains experience by observing other agents performing similar tasks. Whitehead shows that under certain restrictive conditions these methods reduce the complexity of the learning problem to linear in the length of the optimal solution path. In summary, all of these methods provide the learning component with richer sources of experience, and thus accelerate learning.

The method presented in this paper is an on-line mechanism for allowing the learning component to receive advice from a human expert. The new method permits the expert to teach the learning component by guiding it while it performs the multiple-step task. This teaching method is described in the Section 2. The reinforcement learning algorithm employed in this research is the ACE/ASE algorithm (Barto, et al. 1983). Specifications of that algorithm and a modification necessary to realize the new method are presented in Section 3. The multiple-step problem we study is the classic dynamic system of the cart and pole (Section 4). Section 5 describes the experiment, which illustrates that the teaching method significantly decreases the learning time with little input from the human expert.

2 Teaching Method

The teaching method allows a human expert to guide a reinforcement learning system. The expert assumes the role of a teacher, and the learning system that of a student. The objective of the teacher is to impart expert knowledge to the student with as little advice as possible. While the student is learning a multiple-step decision task via a reinforcement learning algorithm, the teacher monitors the student's progress. If the teacher determines that the student is not progressing well, then the teacher offers advice in the form of an action to be applied to the task. The student accepts that decision as the correct one to make. By offering advice, the teacher is conveying to the student a preference for the advised action over all other actions. The student learns from the guidance by reinforcing the tendency to choose that action in that situation. This is implemented by sending a positive reinforcement signal to the ACE/ASE algorithm.

The teaching method does not assume that the expert is perfect. The ACE/ASE algorithm maintains histories of states and actions in the form of feature and eligibility traces. These traces allow recent states and actions to receive some of the credit or blame for the current status of the task, without requiring every past state and action be stored individually. When the expert advises the learning component, that decision becomes part of the traces and thus can receive further credit or blame for the status of the problem. For instance, when the expert advises that an action be taken, the action is performed and the learning component is updated accordingly. Soon after, if the system receives negative feedback from the environment, the expert's action will receive partial blame for that feedback.

When the teacher offers guidance there is an implicit assumption that the most recent actions of the student are not acceptable. The learning component should not credit its own recent actions in this case; the traces should not be used as part of the student's update. Doing so would violate that assumption. The action suggested by the teacher will be the only one taken into consideration. To implement this, the histories are discarded when the expert offers advice.

The teacher can offer advice at any time. This is different from the algorithm of Utgoff and Clouse (1991) because the learner does not stop its exploration to query the expert as to the appropriate action. The teacher does not need to produce an entire sequence of actions, from initial state to goal state, as in Lin (1991); the teacher only provides guidance occasionally. This teaching method is also different from Learning with an External Critic (Whitehead, 1991) because the expert does not reinforce any of the student's actions. The expert causes one of its own actions to be reinforced. The Learning By Watching (Whitehead, 1991) mechanism does not apply because the student does not observe another system in operation, it changes its own system based on advice from the teacher.

3 Reinforcement Learning Algorithm

The reinforcement learning algorithm employed in this research is the ACE/ASE algorithm (Barto, et al. 1983). The ACE (Adaptive Critic Element) implements an evaluation function, and the ASE (Associative Search Element) implements a decision policy. Both of these elements are linear threshold units. The assumptions of this algorithm are: the learning component knows the possible set of actions for any given state, it can manipulate the multiple-step process by performing actions, it does not have an internal model of the problem, and it receives feedback from the environment, however infrequently, as to the status of the problem.

3.1 Details of the ACE/ASE algorithm

The ASE is a mapping from states to actions. The output of the ASE is given by:

$$y(t) = f\left[\sum_{i=1}^{n} w_i(t)x_i(t) + \text{noise}(t)\right]$$

where $y(t)$ is the action chosen, the x_i's are the input features to the ASE, the w_i's are the weights of the unit, "noise" is a real random variable, and the threshold function f is

$$f(x) = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases}$$

This particular threshold function causes the ASE to map to one of two actions. Other threshold functions, such as the sigmoid function, are possible and map to different sets of actions. The "noise" random variable was uniformly distributed on $(-0.005, 0.005)$.

The ACE is a mapping from states to predicted feedback. It performs as a generator of internal feedback signals for the ASE. The output of the ACE is given by:

$$p(t) = \sum_{i=1}^{m} v_i(t)z_i(t)$$

where the z_i's are the ACE input features and the v_i's are the weights. We assume the value, $p(t)$, of a failed state is zero (0).

The evaluation function is trained with a temporal difference method (Sutton, 1988). The change in its weights is based on its previous value, $p(t-1)$, its discounted current value, $\gamma p(t)$, and the reinforcement signal from the environment, $r(t)$. The error value used to change the weights of the evaluation function is given by:

$$\hat{r}(t) = r(t) + \gamma p(t) - p(t-1)$$

Thus, the update rule for the evaluation function weights is:

$$v_i(t+1) = v_i(t) + \beta\hat{r}(t)\bar{z}_i(t) \tag{1}$$

where $v_i(t+1)$ and $v_i(t)$ are the new and old values of the ith weight, β is a learning rate parameter, $\hat{r}(t)$ is the error value, and $\bar{z}_i(t)$ is the trace of the ith ACE feature.

The feature trace provides a mechanism for allowing the features of old states to contribute an exponentially decreasing amount to the weight change. The feature trace for the evaluation function is updated by:

$$\bar{z}_i(t) = \lambda\bar{z}_i(t-1) + (1-\lambda)z_i(t)$$

where $\bar{z}_i(t)$ and $\bar{z}_i(t-1)$ are the new and old values of the trace for the ith ACE feature, λ is a trace decay rate, $0 \leq \lambda < 1$, and $z_i(t)$ is the value of the ith ACE feature. $\bar{z}$ compiles the past feature vectors into one vector.

The update rule for the decision policy is:

$$w_i(t+1) = w_i(t) + \alpha\hat{r}(t)e_i(t) \tag{2}$$

where $w_i(t+1)$ and $w_i(t)$ are the new and old weights on the ith feature, α is a learning rate parameter, $\hat{r}(t)$ is the reinforcement signal, and $e_i(t)$ is the eligibility trace of the ith ASE feature.

The $\hat{r}$ value serves as an internal reinforcement signal to the ASE. If the sum of the current reinforcement and the current discounted prediction, $r(t)+\gamma p(t)$, is greater than the previous predicted reinforcement, $p(t-1)$, then the most recent action has increased the predicted reinforcement of the system. The actions in the eligibility trace are rewarded; the weight vector is moved in a direction that will make those actions more probable. If the difference is negative, which means that the current state should have a lower predicted reinforcement, $\hat{r}$ will be negative, and the previous actions will receive negative reinforcement.

The eligibility trace (e) apportions credit or blame to previous states and actions in a manner similar to the feature trace, $\bar{z}$. The eligibility trace is updated by

$$e_i(t) = \delta e_i(t-1) + (1-\delta)y(t)x_i(t)$$

where $e_i(t)$ and $e_i(t-1)$ are the new and old eligibility trace values for the ith ASE feature value, and δ is an eligibility trace decay rate, $0 \le \delta < 1$. The $y(t)$ term allows the weight vector to move in such a way as to allow that same action to become more likely to occur if the action is rewarded, and less likely if it is punished.

3.2 Addition to Allow Teaching

Whenever the teacher takes an action within the multiple-step process, the two histories (the feature trace and the eligibility trace) are not used to update the weights of either the ACE or the ASE (as discussed in Section 2). The update rule for the ACE becomes:

$$v_i(t+1) = v_i(t) + \beta\hat{r}(t)z_i(t)$$

where $v_i(t+1)$ and $v_i(t)$ are the new and old values of the ith weight, β is a learning rate, $\hat{r}(t)$ is the error value, and $z_i(t)$ is the ith input feature. The change from the other evaluation function weight update rule (Equation 1) is the use of the ACE features instead of the feature trace. Similarly, the weight update rule for the ASE becomes:

$$w_i(t+1) = w_i(t) + \alpha\hat{r}(t)y(t)x_i(t)$$

where $w_i(t+1)$ and $w_i(t)$ are the new and old weights on the ith ASE feature, α is a learning rate, $y(t)$ is the teacher's action, and $x_i(t)$ is the ith feature. The difference between this update rule and Equation 2 is the use of $y(t)x_i(t)$ instead of $e_i(t)$.

The two traces are reset to zeroes when the expert offers advice so that the recent actions of the student will not effect the weight updates.

3.3 Parameters

There are several parameters in the ACE/ASE reinforcement learning algorithm. In this research those values were set at $\alpha = 0.15$, $\beta = 0.25$, $\gamma = 0.95$, $\delta = 0.9$, and $\lambda = 0.8$. These parameter settings, except those for the two learning rates, were taken directly from (Barto, et al. 1983). The original learning rates ($\alpha = 1000.0$ and $\beta = 0.5$) were found to be too large. Presumably, the different parameter values are needed because the input representations we use for the ACE and ASE differ greatly from that in Barto, et. al. (1983) (see Section 4.4).

4 Pole Balancing

The multiple-step problem employed in this research is the classic dynamic system of the cart and pole. In this section we describe the system, outline previous work on it within the reinforcement learning paradigm, and describe the simulation.

4.1 The Cart-Pole Task

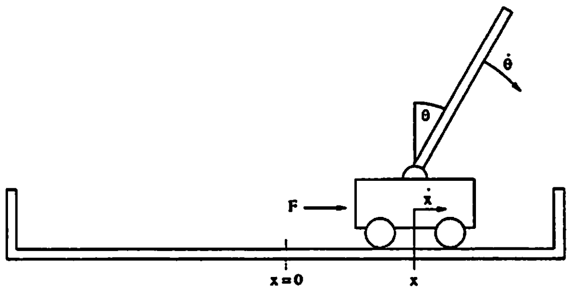

Figure 1. The Cart-Pole System

For the cart-pole system, shown in Figure 1, the wheeled cart can move freely on the one-dimensional, bounded track. The pole is hinged to the top of the cart and can swing left or right. The objective of the task is to keep the pole balanced while keeping the cart within the bounds of the track. We refer to this two-part task simply as "balancing the pole." To achieve that end, a fixed magnitude force F can be applied at unit time intervals to the left or right of the cart. The state of the cart-pole system is given by four state variables, the position and velocity of the cart (x, and $\dot{x}$) and of the pole (θ and $\dot{\theta}$).

The learning problem is to develop a controller that can balance the pole. The cart-pole system has been studied by many researchers (Widrow & Smith, 1964; Michie & Chambers, 1968; Barto, Sutton & Anderson, 1983; Selfridge, Sutton & Barto, 1985; Connell & Utgoff, 1987; Anderson, 1989; Jordan & Jacobs, 1990; Gullapalli, 1991). In the recent learning research, the only information given to the controller as it learns are the state description of the cart-pole system at unit time intervals and a failure signal when either the pole falls beyond a predefined angle or the cart hits an end of the track. The controller knows nothing about the dynamics of the cart-pole system. This model of the learning problem was taken by the systems described in the next section as well as our own research.

4.2 Related Cart-Pole Work

The BOXES program (Michie & Chambers, 1968) learned to balance the pole given only failure signals when the pole was no longer balanced. Michie and Chambers partitioned the state space into many discrete regions, assigning a controller to each region. Each controller learned a control action for its particular region of the state space. In a reimplementation of the BOXES system (Barto, et al. 1983), only one of ten runs was able to balance the pole for 500,000 time steps in under 100 attempted trials.

In their own system, Barto, et. al. (1983) also partitioned the state space into distinct regions. They employed two neuron-like adaptive elements, the Adaptive Search Element (ASE) and the Adaptive Critic Element (ACE). (These elements were discussed in Section

3). The input features for both elements was a vector of 162 binary values. Only one of those values was "on" for any given state, representing the region into which the state fell. Their system was able to balance the pole for 500,000 time steps after an average of 75 failed trials.

Both of the above systems divided the state space into separate regions. The particular quantization chosen was based on domain knowledge and experimentation with the cart-pole system. The CART program (Connell & Utgoff, 1987) did not segment the state space. The decision policy learned was based on the distribution of states in the state space that had been identified as either desirable or undesirable. Undesirable states were those that occurred directly before a failure. Desirable states were identified from the middle of long balancing episodes. CART learned to balance the pole for 70,000 time steps in only 16 trials. Although this seems to be a great improvement over the previous sytems, its mechanism for assigning desirability to states is less general than the ACE/ASE approach.

All of the aforementioned systems were solving a subset of the cart-pole task; keeping the pole balanced when the cart started motionless near the center of the track and the pole was nearly vertical and motionless as well. In contrast, Anderson (1989) allowed the cart and pole to begin in random states and used the cart-pole state variables as input to his learning component. The learning component consisted of two multi-layer networks. One implemented an ACE-like evaluation function, and the other an ASE-like decision policy. Anderson's system was able to balance the pole for 100,000 time steps after an average of more than 10,000 failed trials.

In our simulation of the system, the cart and pole begin in random states (see Section 4.3), and the input representations for the ACE and ASE are based on the cart-pole state variables (see Section 4.4).

4.3 Cart-Pole Simulation

The cart-pole system was simulated numerically by approximating two nonlinear differential equations by the Euler method with a time step of 0.02 seconds. The equations are:

$$\ddot{\theta}_t = \frac{g \sin\theta_t + \cos\theta_t \left[\frac{-F_t - m_p l \dot{\theta}_t^2 \sin\theta_t + \mu_c \mathrm{sgn}(\dot{x}_t)}{m_c + m_p}\right] - \frac{\mu_p \dot{\theta}_t}{m_p l}}{l\left[\frac{4}{3} - \frac{m_p \cos^2\theta_t}{m_c + m_p}\right]}$$

$$\ddot{x}_t = \frac{F_t + m_p l \left[\dot{\theta}_t^2 \sin\theta_t - \ddot{\theta}_t \cos\theta_t\right] - \mu_c \mathrm{sgn}(\dot{x}_t)}{m_c + m_p}$$

where $\dot{x}_t$ is the velocity of the cart, $\ddot{x}_t$ is the acceleration of the cart, θ_t is the angle of the pole, $\dot{\theta}_t$ is the velocity of the pole, $\ddot{\theta}_t$ is the acceleration of the pole, g is the acceleration due to gravity (9.8 m/s^2), m_c is the mass of the cart (1.0 kg), m_p is the mass of the pole (0.1 kg), l is half of the pole length (0.5 m), F_t is the force applied to the cart ($\pm$10.0 newtons), μ_c is the coefficient of friction of the cart on the track (0.0005), and μ_p is the coefficient of friction of the pole on the cart (0.000002).

In the simulation, the pole was balanced if its angle was within twelve degrees of vertical ($-12^\circ \leq \theta \leq 12^\circ$). The length of the track was 4.8 meters (-2.4 m $\leq x \leq 2.4$ m). These simulation equations and values were the same ones used in (Barto, Sutton & Anderson, 1983; Selfridge, Sutton & Barto, 1985; Connell & Utgoff, 1987).

At the beginning of each trial the cart-pole state variables were set to random values uniformly distributed within the following ranges:

$$\begin{array}{rcccl} -1.25\text{m} & < & x & < & 1.25\text{m} \\ -1.25\text{m/s} & < & \dot{x} & < & 1.25\text{m/s} \\ -2.5^\circ & < & \theta & < & 2.5^\circ \\ -15.0^\circ/\text{s} & < & \dot{\theta} & < & 15.0^\circ/\text{s} \end{array}$$

These ranges were chosen to give the learning component a rich set of training problems. A problem initialized near the extremes of the above ranges is difficult to solve: the cart is near an edge, moving towards that edge and the pole is leaning in the direction the cart is headed as well as moving in that direction. It is solvable, therefore, all states whose variables are within these ranges are solvable.

4.4 Representation

The ASE features were the cart-pole state variables. This representation for the decision policy is sufficient because the optimal control law for a similar control task is nearly linear (Anderson, 1989). The ACE features were the first- and second-order terms of the cart-pole state variables $(x, \dot{x}, \theta, \dot{\theta}, x^2, x\dot{x}, x\theta, x\dot{\theta}, \dot{x}^2, \dot{x}\theta, \dot{x}\dot{\theta}, \theta^2, \theta\dot{\theta}, \text{and } \dot{\theta}^2)$. Anderson's discussion of the evaluation function space suggests that this representation is also sufficient.

The state variables were scaled so that their values were in $[-1.0, 1.0]$. The symmetric scaling allows quicker learning for the linear threshold units (Hampson & Volper, 1986). For the locations of the cart and pole, the scale factors were 2.4m and 12.0°, respectively. Because the bounds on the velocity variables were not know a priori, several runs of the cart-pole system were executed to determine those bounds. The scaling could have been done automatically, but was not. The maximum absolute value for the cart's velocity was found to be about 3.9 m/s, so the scale value was set to 4.0m/s. For the pole's velocity, the maximum absolute value seen was about 147°. The scale factor was set at 157.56° (2.75 radians).

5 Experiment

To determine whether the teaching method improves the performance of the learning component on the cart-pole task, forty (40) runs of the system were performed. Each run consisted of several trials, and began with the weight vectors of the ACE and ASE set to zero. A run ended with the successful completion of a trial. A trial began with the cart-pole system in a random state, as described in Section 4.3. A trial ran until either a failure signal was received, in which case a new trial was begun, or the pole had been balanced successfully for 10,000 time steps, a successful completion.

In the first twenty runs the human expert did not interact with the system; the only feedback received was a failure signal when the pole fell outside of the required range or the cart had hit an end of the track. Table 1 presents the averages and standard deviations of the results of the experiment. The results of these runs are in the left columns of the table.

The next twenty runs were performed under the same stopping condition. In these runs, however, a human agent taught the learning component. The human interacted with the

	Without Teaching		With Teaching	
	$\bar{x}$	s	$\bar{x}$	s
Failed Trials	992.00	723.47	8.25	6.10
Time Steps	181,529.20	11,3420.71	10,767.05	866.52
Guidance	0.00	0.00	7.25	4.31

Table 1. Results

learning component via an interactive graphics program. The graphics program allows its user to watch the cart-pole system as the learning component manipulates it. Furthermore, the user can advise the learner by pressing single keys on the keyboard, one to advise pushing the cart to the left and another for pushing to the right. The graphics program simulates the cart-pole system in real-time, drawing the cart-pole configuration on the screen every 0.02 seconds, the time increment of the simulation. Because the system was simulated in real-time, the user can only advise the system to execute single actions. Each user keystroke is expert guidance, without which the system can learn by itself. The results of these runs are in the right columns of Table 1.

The influence of the human on the learning component greatly reduced the amount of time needed to solve the problem. By itself the learning component required on average 992 trials before it could balance the pole for 10,000 time steps. With the human's help, the average number of failures dropped to less than 10. To achieve this reduction in failed trials required on average 7.25 actions on the part of the human. Little input from the human produced a drastic reduction in the number of failed trials.

Also shown in the table are the number of time steps required to achieve the goal of balancing the pole. On average, the lone learning component took more than 180,000 time steps (about one hour of real-time simulation), while with human help it took less than 11,000 time steps (about 3.5 minutes). Considering that it takes almost 3.5 minutes just to balance the pole for 10,000 time steps, the human's influence produced a drastic difference by this measure as well.

6 Conclusions

The experiment shows that the teaching method accelerated the learning process greatly, and with little input from the expert. The learning system was able to incorporate the expert's actions to its benefit.

The teaching method provides a convenient mechanism by which to do automatic knowledge engineering. The knowledge is acquired through interaction with the human expert. The method does not require a priori specification of that knowledge. Furthermore, the knowledge is provided in a way that is natural for the human. The human need not specify why particular actions were chosen, nor develop hypothetical situations and the actions appropriate to them. The expert simply reacts as the situations warrant; and the learning system is able to learn from the expert's choices.

Acknowledgments

This material is based upon work supported by the National Aeronautics and Space Administration under Grant No. NCC 2-658. We thank the other members of the Machine Learning Lab at UMass, Jamie Callan, Tom Fawcett, Carla Brodley, and Rodger Zanny, for their valuable input and many stimulating conversations, and Margie Connell for her helpful comments. The cart-pole simulator was provided by Andy Barto.

References

Anderson, C. W. (1989). Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine, 9*, 31-37.

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

Connell, Margaret, E., & Utgoff, Paul E. (1987). Learning to control a dynamical physical system. *Computational Intelligence, 3*, 330-337.

Gullapalli, Vijaykumar (1991). A comparison of supervised and reinforcement learning methods on a reinforcement learning task. *Proceedings of the 1991 IEEE International Symposium on Intelligent Control* (pp. 394-399). Arlington, VA.

Hampson, S. E., & Volper, D. J. (1986). Linear function neurons: Structure and training. *Biological Cybernetics, 53*, 203-217.

Jordan, Michael I., & Jacobs, Robert A. (1990). Learning to control an unstable system with forward modeling. In Touretzky (Ed.), *Advances in Neural Information Processing Systems.* San Mateo, CA: Morgan Kaufman.

Lin, Long-Ji (1991). Programming robots using reinforcement learning and teaching. *Proceedings of the Ninth National Conference on Artificial Intelligence* (pp. 781-786). Anaheim, CA: MIT Press.

Michie, D., & Chambers, R. A. (1968). BOXES: An experiment in adaptive control. In Dale & Michie (Eds.), *Machine Intelligence 2.* Edinburgh: Oliver and Boyd.

Selfridge, O., Sutton, R. S., & Barto, A. G. (1985). Training and tracking in robotics. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 670-672). Los Angeles, CA: Morgan Kaufmann.

Sutton, R. S. (1988). Learning to predict by the method of temporal differences. *Machine Learning, 3*, 9-44.

Utgoff, P. E., & Clouse, J. A. (1991). Two kinds of training information for evaluation function learning. *Proceedings of the Ninth National Conference on Artificial Intelligence* (pp. 596-600). Anaheim, CA: MIT Press.

Watkins, C. J. C. H. (1989). *Learning with delayed rewards.* Doctoral dissertation, Psychology Department, Cambridge University.

Whitehead, Steven, D. (1991). A complexity analysis of cooperative mechanisms in reinforcement learning. *Proceedings of the Ninth National Conference on Artificial Intelligence* (pp. 607-613). Anaheim, CA: MIT Press.

Widrow, B., & Smith, F. W. (1964). Pattern-recognizing control systems. In Tou & Wilcox (Eds.), *Computer and Information Sciences Proceedings.* Washington, D.C: Spartan Books.