Learning to Control an Inverted Pendulum Using Neural Networks

Charles W. Anderson

ABSTRACT: An inverted pendulum is simulated as a control task with the goal of learning to balance the pendulum with no a priori knowledge of the dynamics. In contrast to other applications of neural networks to the inverted pendulum task, performance feedback is assumed to be unavailable on each step, appearing only as a failure signal when the pendulum falls or reaches the bounds of a horizontal track. To solve this task, the controller must deal with issues of delayed performance evaluation, learning under uncertainty, and the learning of nonlinear functions. *Reinforcement* and *temporal-difference* learning methods are presented that deal with these issues in order to avoid unstable conditions and balance the pendulum.

Introduction

The inverted pendulum is a classic example of an inherently unstable system. Its dynamics are basic to tasks involving the maintenance of balance, such as walking and the control of rocket thrusters. A number of control design techniques have been investigated using the inverted pendulum [1]–[4]. The successful application of these design techniques requires considerable knowledge of the system to be controlled, including an accurate model of the dynamics of the system and an expression of the system's desired behavior, usually in the form of an objective function.

How can control be accomplished when such knowledge is not available? This question is addressed here by considering the inverted pendulum control problem when the dynamics are not known a priori and an analytical objective function is not given. All that is known are the values and ranges of the state variables of the inverted pendulum system and that a negative *failure signal* is to be maximized over time. A function that selects control actions given the current state of the pendulum must be learned through experience by trying various actions and noting the results, starting with no hints as to which actions are correct.

Without an objective function to evaluate states and actions, modifications to the controller can be based only on the occurrence of failure signals. A long sequence of actions can develop before a failure signal is encountered, resulting in the difficult *assignment-of-credit* problem, where it is necessary to decide which actions in the sequence contributed to the failure.

In this paper, neural network learning methods are described that learn to generate successful action sequences by acquiring two functions: an *action function*, which maps the current state into control actions, and an *evaluation function*, which maps the current state into an evaluation of that state. The evaluation function is used to assign credit to individual actions. Two networks having a similar structure are used to learn the action and evaluation functions. They will be referred to as the *action network* and the *evaluation network*.

As shown in later sections, the desired evaluation function for the inverted pendulum task is nonlinear; a single-layer neural network cannot form this map. One solution to this problem is to transform the original state variables into a new representation with which a single-layer network can form the evaluation function. Barto et al. [5] demonstrated a quantization of the state space of the inverted pendulum with which single-layer networks could learn to balance the pendulum. A second solution is to add a second adaptive layer that learns such a representation. Anderson [6] extended the work of Barto et al. by applying a form of the popular error back-propagation method to two-layered networks that learn to balance the pendulum given the actual state variables of the inverted pendulum as input.

In this paper, the work of Barto et al. and Anderson is summarized by discussing the neural network structures and learning methods from a functional viewpoint and by presenting the experimental results. First, the inverted pendulum task and previous applications of neural networks to this task are described.

Presented at the 1988 American Control Conference, Atlanta, Georgia, June 15–17, 1988. Charles W. Anderson is with the Self-Improving Systems Department of GTE Laboratories, Inc., Waltham, MA 02254.

Inverted Pendulum

The inverted pendulum task involves a pendulum hinged to the top of a wheeled cart that travels along a track, as shown in Fig. 1. The cart and pendulum are constrained to move within the vertical plane. The state at time t is specified by four real-valued variables: the angle between the pendulum and vertical and the angular velocity (θ_t and $\dot{\theta}_t$) and the horizontal position and velocity of the cart (h_t and $\dot{h}_t$). The inverted pendulum system was simulated using the following equations of motion, where the units of θ, h, and time t are radians, meters, and seconds, respectively, and where g is the acceleration due to gravity (9.8 m/sec^2), F_t the output of the action network (± 10 N), m_c the mass of the cart (1.0 kg), m the mass of the pendulum plus the cart (1.1 kg), and l the distance from the pivot to the pendulum's center of mass (0.5 m).

$$\ddot{\theta}_t = \frac{mg \sin \theta_t - \cos \theta_t [F_t + m_p l \dot{\theta}_t^2 \sin \theta_t]}{(4/3)ml - m_p l \cos^2 \theta_t}$$

$$\ddot{h}_t = \{F_t + m_p l [\dot{\theta}_t^2 \sin \theta_t - \ddot{\theta}_t \cos \theta_t]\}/m$$

This system was simulated by numerically approximating the equations of motion using Euler's method with a time step of $\tau = 0.02$ sec and discrete-time state equations of the form $\theta[t+1] = \theta[t] + \tau\dot{\theta}[t]$. The sampling rate of the inverted pendulum's state and the rate at which control forces are applied are the same as the basic simulation rate, i.e., 50 Hz.

The goal of the inverted pendulum task is to apply a sequence of right and left forces of fixed magnitude to the cart such that the pendulum is balanced and the cart does not hit the edge of the track. A zero-magnitude

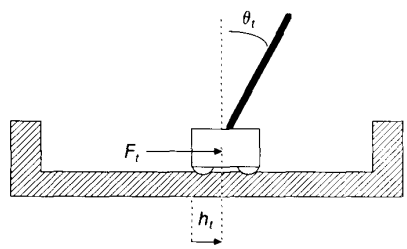

Fig. 1. The inverted pendulum.

force is not permitted. Bounds on the angle and on the cart's horizontal position specify the states for which a failure signal occurs. There is no unique solution—any trajectory through the state space that does not result in a failure signal is acceptable. The only information regarding the goal of the task is provided by a failure signal, which signals either the pendulum falling past ± 12 deg or the cart hitting the bounds of the track at ± 2.4 m. These two kinds of failure are not distinguishable in the case considered herein.

The goal as just stated makes this task very difficult; the failure signal is a delayed and rare performance measure. Before describing a solution to this formulation of the inverted pendulum task, we briefly discuss other approaches that assume the existence of additional task-specific knowledge.

A traditional control approach when the dynamic equations of motion are known is to assume that the control force F_t is a linear function of the four state variables (θ_t, $\dot{\theta}_t$, h_t, $\dot{h}_t$) with constant coefficients $b_1, \ldots, b_4$:

$$F_t = b_1\theta_t + b_2\dot{\theta}_t + b_3h_t + b_4\dot{h}_t$$

The coefficients b_i are chosen to stabilize the linearized version of the system differential equations for θ_t and h_t small in magnitude. This is the approach followed by Cheok and Loh [1] in a similar problem; they use linear feedback in three (of the four) variables to obtain stable control for a ball-balancing experiment. The success of this approach depends heavily on the match between the actual system dynamics and the linearized approximation.

The earliest application of neural networks to the inverted pendulum task is that of Widrow and Smith [7] and Widrow [8]. They approached the problem as described earlier, using traditional control methods to derive a control law to stabilize the linearized system for small θ_t and h_t. Then they trained a network to mimic the output of the control law by observing the input-output behavior of the control law as it balanced the pendulum. Guez and Selinsky [9] extended this approach to include multilayer networks trained by observing a nonlinear control law.

When the dynamics are not known, it is necessary to use some adaptive or learning approach to obtain a stable control, which is some unknown function U of the four state variables:

$$F_t = U(\theta_t, \dot{\theta}_t, h_t, \dot{h}_t)$$

With unknown dynamics, a controller cannot be designed to provide examples of desired behavior. A human controller who is able to stabilize an inverted pendulum is an alternative source of desired behavior, as demonstrated by Widrow and Smith [7], [8] and Guez and Selinsky [9]. Tolat and Widrow [10] provide a third demonstration of training with a human controller, with the novel use of feedback based on pixel values derived from a visual image of the pendulum location rather than the pendulum's state variables.

If neither a designed controller nor a human expert is available, learning must be guided by some measure of actual performance. For example, Connell and Utgoff [11] measured performance by the distance from the current state to the (0, 0, 0, 0) state, taking the action on each step that most reduced this difference. As is the case for traditional control methods, this amounts to adding the knowledge that the system must be stabilized about a particular state. Rosen et al. [12] assumed a different type of knowledge, the knowledge that the inverted pendulum task is a failure-avoidance task. For avoidance tasks, longer trajectories through state space between failures are more desirable, so Rosen et al. identified actions that resulted in cycles in state space as the preferred actions.

The only performance measure present in our formulation of the inverted pendulum task is the failure signal. The approach to this task, as summarized in the next two sections, is an example of how successful control can be learned when limited task-specific information is available.

Solution Using Two Single-Layer Networks

The architecture of a network and the computations performed by each unit specify a function from input to output vectors. The function is parameterized by the numerical connection weights between units and on the inputs to the network. The function is altered by a learning method that adjusts the values of the weights.

Learning can be based on several forms of evaluative feedback (see Hinton [13] for a review). *Supervised* learning methods, the most commonly used in neural networks, require a training set of data consisting of input vectors and corresponding desired output vectors. Such methods cannot be applied to tasks for which the desired output of the network is not known. The inverted pendulum, as we have defined it, is such a task: the correct action for most states is not even well-defined, since many trajectories are possible that indefinitely avoid failure.

If the desired output is not available, the performance of the network must be evaluated indirectly by considering the effect of its output on the environment with which the network interacts. *Reinforcement* learning methods can be applied when this effect is measured by changes in an evaluation signal, or reinforcement—a term borrowed from theories of animal learning from which reinforcement learning methods originated [14].

Next we describe the experiments by Barto et al. [5] involving two single-layer networks. See Sutton [15] for a more thorough, extended treatment of this approach to the inverted pendulum task and of reinforcement learning methods in general.

We distinguish the two networks by calling one the action network and the other the evaluation network. The *action network* learns to select actions as a function of states. It consists of a single unit having two possible outputs, one for each of the two allowable control actions of pushing left or right on the cart with a fixed-magnitude force. The output of the unit is probabilistic—the probability of generating each action depends on the weighted sum of the unit's inputs, i.e., the inner product of the input vector and the unit's weight vector.

Initial values of the weights are zero, making the two actions equally probable. The action unit learns via a reinforcement learning method. It tries actions at random and makes incremental adjustments to its weights, and, thus, to its action probabilities, after receiving nonzero reinforcements. The only nonzero reinforcement present in the inverted pendulum task is a failure signal. Learning good actions is extremely slow when based on this rare and delayed signal.

A second mechanism is needed to apportion the blame for the failure among the actions in the sequence leading to the failure. This mechanism is provided by the *evaluation network*, which also consists of a single unit. The evaluation unit learns the expected value of a discounted sum of future failure signals by means of a *temporal-difference*, or TD, method of prediction, developed by Sutton [16]. TD methods learn associations among signals separated in time, such as the inverted pendulum state vectors and failure signals. Through learning, the output of the evaluation network comes to predict the failure signal, with the strength of the prediction indicating how soon failure can be expected to occur. The predictions are adjusted after each step by an amount proportional to the network's input and the difference between the new prediction, based on the current state of the inverted pendulum, and the previous prediction, based on the previous state, i.e., the *temporal difference* or change in predic-

tion of failure. This sequence of prediction changes ends with the occurrence of failure, and the final prediction change is dependent on the difference between the failure signal and the previous prediction. Convergence theorems for the TD class of algorithms have been proven by Sutton [16]. Ongoing work by Sutton includes the investigation of relationships between TD methods and dynamic programming.

The output of the evaluation unit is the inner product of the input vector and the unit's weight vector. Assuming the input vector is a representation of the inverted pendulum's state (discussed subsequently) and that the weights are developed by the TD method, the scalar output of the evaluation unit provides a ranking of the states. The difference in the unit's output on the transition from one state to another is used to judge the effectiveness of the previous action. An increase in the evaluation signifies a transition to a state having a weaker prediction of failure and that the probability of the preceding action should be increased. Similarly, the probability of repeating an action that precedes an evaluation decrease should be lowered. In this way, the change in the evaluation network's output serves as a reinforcement during the possibly long periods between failures. However, the learned evaluation function is not always helpful, particularly before much experience has been gained. The learning methods for updating both the evaluation and action networks must deal with this uncertainty.

The performance of any learning system is highly dependent on its input representation. The four real-valued state variables are an adequate representation for the action unit, since the optimal control law for a similar inverted pendulum task is linear and can be approximated by the stochastic action unit. However, using this representation would prevent the evaluation unit from being able to form a good prediction of failure for the following reason.

Consider evaluations as a function of just θ and $\dot{\theta}$, the pendulum's angle and angular velocity. The failure signal is defined to have the value -1 on failure and 0 for all other states. A failure occurs when the value of θ is less than -12 deg or greater than 12 deg. States that occur just prior to failure typically have either high θ and high $\dot{\theta}$, or low θ and low $\dot{\theta}$, i.e., the pendulum is falling in the same direction in which it is leaning. These states should produce an evaluation near -1, a strong prediction of failure. Other states, such as those for which the pendulum is moving toward the balanced position, should have an evaluation closer to 0, a weak prediction of failure. Thus, the shape of this evaluation function as the state moves from $-\theta$, $-\dot{\theta}$ to $+\theta$, $+\dot{\theta}$ is nonmonotonic, first rising from -1 toward 0, then falling back toward -1.

Since the output of the evaluation unit is a linear function of its input, this evaluation function cannot be formed. A different representation of the state must be used for which this function is linear. One alternative is to adopt a table lookup strategy and divide the state space into discrete, nonoverlapping regions, associating a unique input component and weight with each region. The input components then could be binary-valued, with only one being nonzero at a time, signaling which region the current state of the pendulum is in. A unique evaluation can be assigned to each region by adjusting the corresponding weight, approximating any function to an accuracy determined by the coarseness of the state-space partitioning.

The experiments described here were motivated by the work of Michie and Chambers [17], who devised a learning system called BOXES which learned to control an inverted pendulum using a state representation of discrete regions as described earlier. To compare with the performance of the BOXES learning system, the same representation was used. The regions of the state space were formed by the intersections of six intervals along the θ dimension and three intervals along the $\dot{\theta}$, h, and $\dot{h}$ dimensions, making a total of 162 regions. The resulting networks are shown in Fig. 2. Each unit receives the 162 binary input components, and the evaluation unit's output directs the learning process for both units.

The primary purpose of these experiments was to compare the learning performances of the TD prediction method (called the *Adaptive Critic* in [5]) and the method used by Michie and Chambers to assign credit to actions. The key difference between the methods is that Michie and Chambers used counters in each region to remember past states and times between state occurrences and failure, and that learning occurred only on failure. The TD method allows learning to occur continuously using the learned evaluation function and differences in its output as reinforcement, rather than waiting for further failures.

The results in Fig. 3 (from [5]) show steps between failures versus failures for the single-layer network and for Michie and Chambers' BOXES system. The curves in the figure are averaged over 10 experiments, each starting with the learning system in a completely naive state and terminated either after 100 failures or 500,000 action steps (a simulated time of almost 3 hr). Balancing time increases with experience for both learning systems, but the networks attain a much longer balancing time, demonstrating the superiority of the TD method of learning between failures for this task. The final flattening of the networks' curve is a ceiling effect due to the termination of runs longer than 500,000 steps; the length of the final balancing period for each run was assigned to the remaining failures. If run longer, this curve would continue to increase. After 500,000 steps, the probability that the network will generate actions leading to failure becomes very small and continues to decrease with additional experience.

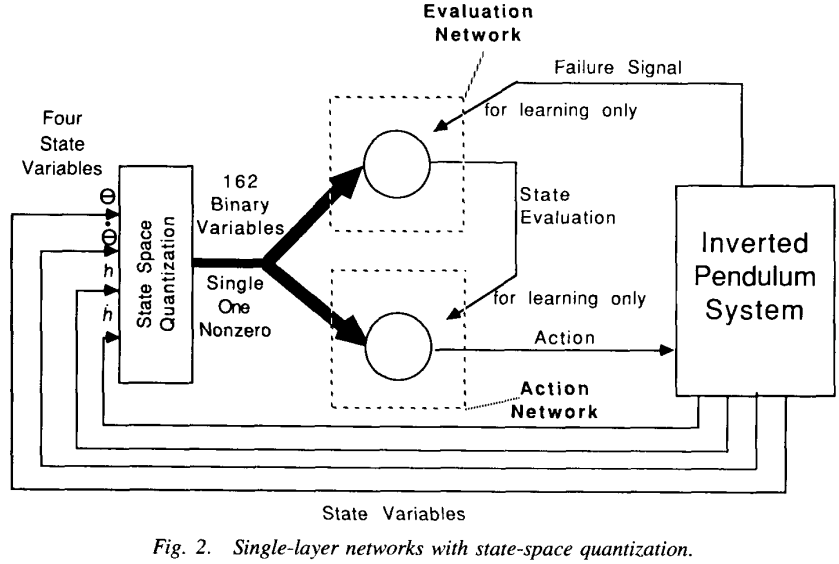

Fig. 2. Single-layer networks with state-space quantization.

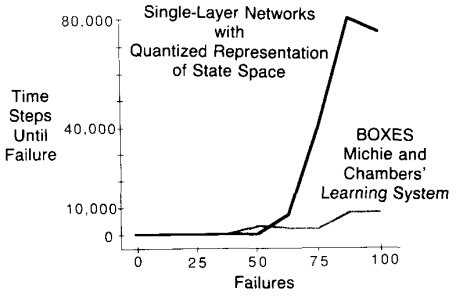

Fig. 3. Learning curves for single-layer networks and BOXES using quantized representation of state space.

Solution Using Two-Layer Networks

In deciding how to divide the state space, one must strike a balance between generality and learning speed. A very fine quantization with many regions permits accurate approximation of complex functions, but learning the correct output for each of the many regions requires much experience. Learning can be faster with a coarse quantization because learning for one state in a region is transferred to all states in the region, but only functions whose output remains relatively constant over regions can be represented. Clearly, an adaptive representation that learns a quantization or other form of feature set based on experience is needed. It should learn to make fine discriminations among some states and coarse generalizations among others, as appropriate for a given task.

The experiments with two-layer networks by Anderson [6], described in this section, are a step in this direction. A second layer of adaptive units is added to the single units described in the previous section. The real-valued state variables are given as input to every unit in both layers, and the outputs from the new units become additional inputs to the original units. This structure is shown in Fig. 4.

The original units are called the output units of the networks. The new units are called *hidden units* because their outputs do not have a direct effect on the network's environment. Whereas output-unit learning can be based on evaluation differences, there is no analogous signal on which to base learning in the hidden units. After assigning credit to an individual action, there remains the problem in a multilayer network of distributing this credit among the hidden units that influenced the selection of that action by the output unit.

This is one of the major problems that slowed developments in adaptive networks after the original work in the 1950s and 1960s. However, recent work has suggested that gradient descent techniques, even with the well-known problems of plateaus and local minima, may be feasible solutions to learning in hidden units for some problems. A gradient descent technique for learning in hidden units having particular nonlinear, differentiable, output functions has been studied and given the name *error back-propagation* [18]. Anderson [6] used variants of error back-propagation to learn in the hidden units of the evaluation and action networks. The errors propagated to hidden evaluation units were based on the differences in the evaluation network's output, whereas errors for hidden action units were based on this difference and on which action was taken.

The results of Anderson's two-layer experiments are shown in Fig. 5. The curves are averaged results of 10 experiments, each starting with a naive network and terminated after 10,000 failures or 500,000 steps. The two-layer networks learned to balance the pendulum for an average of about 24 min of simulated real time, before runs were terminated at the 500,000th step. Single-layer networks, when given the state variables as input, were unable to learn to balance the pendulum, actually doing only slightly better than the nonlearning strategy of choosing actions randomly with equal probability. Even though a good control law can be represented by the single-layer action network, the fact that the linear evaluation network cannot form useful evaluations prevented the control law from being learned.

A direct comparison between these results and the single-layer results of the previous section cannot be made. In the single-layer experiments, the state variables of the pendulum were reset to zero after every failure. When this was done for the two-layer experiments, the networks learned to balance the pendulum but did not learn to keep the cart in the center of the track. Generalization of what was learned in the center of the track prevented the learning of different, more appropriate, actions at the ends of the track. To ensure richer experience early in a training run, the state variables were reset to random values after failure. With this change, the networks learned actions for centering the cart *and* for balancing the pendulum. Another possible solution to this difficulty is to increase the importance of centering the cart by using a larger, more negative, failure signal on bumping the track bounds relative to the failure signal for the pendulum exceeding its angular bounds. This was not tested.

Analysis of what the networks have learned is aided by plotting output values of units as surfaces with respect to two of the four state variables. Plotting the output of the evaluation network with respect to θ and $\dot{\theta}$ for fixed values of h and $\dot{h}$, as shown in Fig. 6, produces a surface with a ridge running from $-\theta$, $+\dot{\theta}$ to $+\theta$, $-\dot{\theta}$ with lowest values for $-\theta$, $-\dot{\theta}$ and $+\theta$, $+\dot{\theta}$. This function ranks states for which the pendulum is moving toward vertical more favorable than other

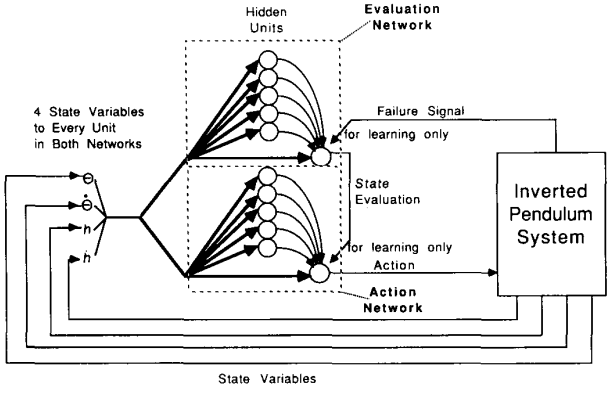

Fig. 4. Two-layer networks receiving unquantized state variables.

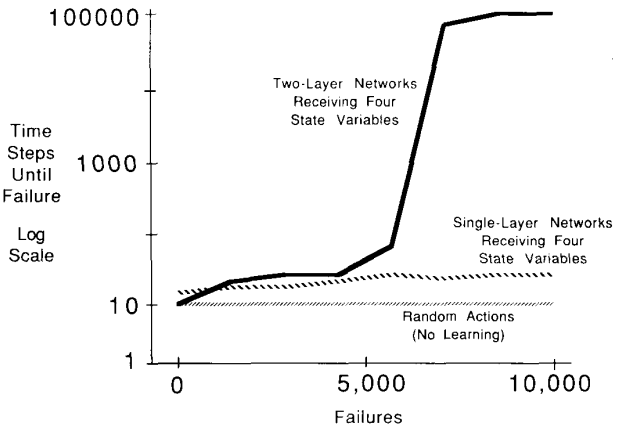

Fig. 5. Learning curves for two-layer and single-layer networks receiving unquantized state variables.

states. The ridge is shifted for different values of h and $\dot{h}$: when the cart is approaching the right end of the track ($+h$ and $+\dot{h}$), states for which the pendulum is to the left of vertical are evaluated more favorably than the states with the pendulum straight up. This is exactly what is required; the pendulum must be "balanced" a bit left of vertical to permit a larger proportion of pushes to the left to bring the pendulum back to the center of the track. The reverse situation exists on the left side of the track.

To understand what is learned by the hidden units, their outputs can be similarly plotted. Figure 7 shows the output of one hidden unit from the evaluation network. This unit has learned a function that is simply a positively sloped ramp from $-\theta$, $-\dot{\theta}$ toward $+\theta$, $+\dot{\theta}$, with the slope decreasing to zero near the midrange of θ and $\dot{\theta}$. The contribution that this unit makes to the evaluation network's output depends on the value of the weight with which its output is connected to the output unit and on the values of the other output unit's weights. Recall that, in addition to the hidden units' outputs, the output unit receives the four state variables as input. The output unit simply computes a weighted sum of its inputs, so can represent any linear function of the state variables and hidden unit outputs. For the learning run from which these figures are generated, we find that the output unit has used the four state-variable inputs to form one side of the ridge in Fig. 6 as a positive linear ramp from $+\theta$, $+\dot{\theta}$ toward $-\theta$, $-\dot{\theta}$. The addition of the hidden unit's output (Fig. 7) pulls down the $-\theta$, $-\dot{\theta}$ corner of the evaluation network surface. This hidden unit function, which we can call a *feature*, is sufficient to form the ridge. In fact, all five hidden units tend to redundantly develop the same function. If more features were required, the hidden units would probably develop different functions.

The output of the action network is stochastic; therefore, its output is represented by the probability of an action being a push to the right. Figure 8 is a plot of this probability. States for which the probability is near 1 will result mostly in pushes to the right, whereas pushes to the left will result from states for which the probability is near 0. The surface shows a quick transition from left pushes to right pushes as the state shifts from $-\theta$, $-\dot{\theta}$ to $+\theta$, $+\dot{\theta}$. This transition is analogous to a switching curve for a deterministic controller. The location of the transition shifts as the cart's position and velocity change in order to maintain balance to the left of vertical when at the right side of the track and to the right of vertical when at the left side.

The output unit of the action network can form the function in Fig. 8 without the aid of hidden units. The action network's hidden units tend to evolve very little from their initial states and do not develop significant weights connecting them to the output unit.

For different initial weight values and different seeds for the random number generator, the learned evaluation and action functions will differ only slightly. The hidden unit functions and weight values, however, do differ significantly. For example, for some runs, the hidden units of the evaluation network learn functions that provide the $+\theta$, $+\dot{\theta}$ side of the evaluation hill with the output unit forming the $-\theta$, $-\dot{\theta}$ side as a function of the direct state-variable inputs, a dual solution to that shown in the preceding figures.

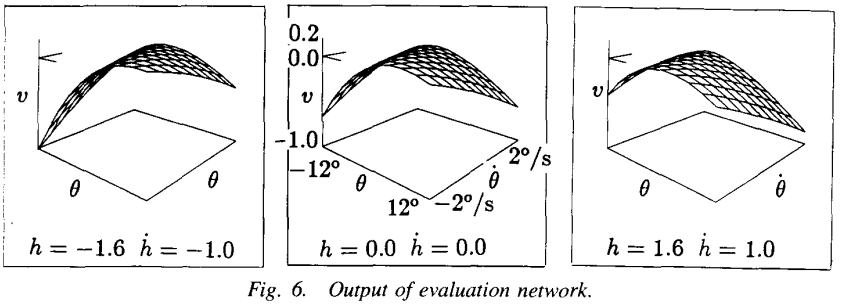

Fig. 6. Output of evaluation network.

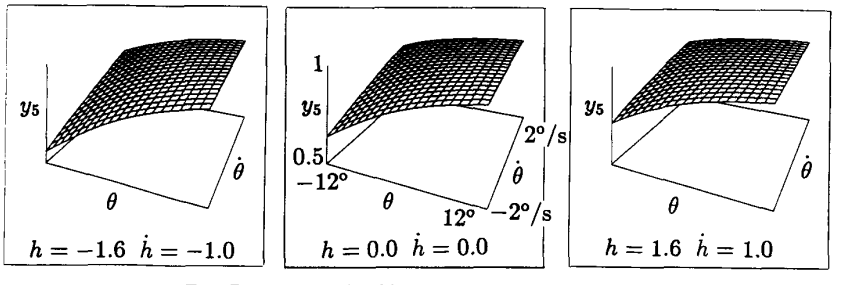

Fig. 7. Output of hidden unit in evaluation network.

Discussion

In many real-world situations, a control objective cannot be expressed as a function defined over all states, but only for a relatively small subset of states. For some control tasks, such a minimally defined objective is perhaps even desirable. Requiring a controller to bring the value of a state variable as close to zero as possible when the true objective is just to avoid extreme values

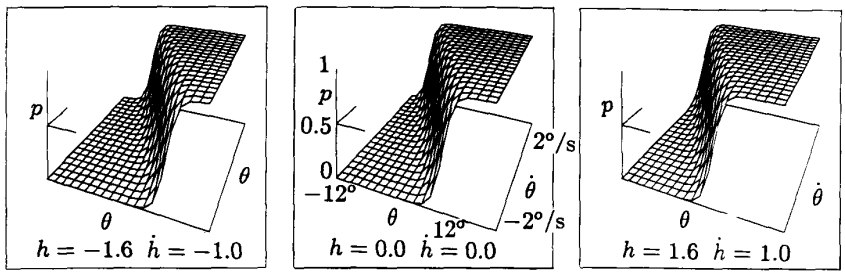

Fig. 8. Output of action network.

might interfere with the control of other state variables.

Learning from experience during periods of no performance feedback is difficult. Neural networks learning via reinforcement learning and temporal difference methods deal with this problem by simultaneously learning a probabilistic action-generating function and a state-evaluation function. This approach to the inverted pendulum task is unique; all other learning systems designed for this task assume more a priori knowledge, such as an explicit teacher providing correct actions [7], [9]. The applicability of this approach to other tasks has been demonstrated by Helferty et al. [19] for a one-legged hopping machine and by Hoskins and Himmelblau [20] in a process control situation.

The difficulties of motor control and other low-level control tasks may yield to the careful application of neural network learning methods. The ability of neural networks to handle multiple-input and -output variables, nonlinear functions, and delayed feedback as well as their potential for fast, parallel implementation warrants further investigation of neural network learning methods in control domains. It is important to realize that these methods are not special, magical, stand-alone techniques, but outgrowths of long lines of research in function approximation, optimization, signal processing, and pattern classification, and can be combined with existing control techniques in straightforward ways. For example, Miller [21] and Franklin [22] have added networks trained by supervised and reinforcement learning methods, respectively, to refine the performance of predefined controllers.

Experiments in learning control with neural networks may shed some light on how to deal with real-world uncertainties and complexities in control. However, many issues remain unresolved, such as how the performance of learning methods scales up to larger, more complex tasks than those currently being studied. For more difficult problems, a training curriculum progressing from simple to difficult parts of the problem might greatly reduce overall learning time [23].

Acknowledgments

This work was partially supported by the Air Force Office of Scientific Research and the Avionics Laboratory (Air Force Wright Aeronautical Laboratories) through Contract F33615-83-C-1078.

References

[1] K. C. Cheok and N. K. Loh, "A Ball-Balancing Demonstration of Optimal and Disturbance-Accommodating Control," *IEEE Contr. Syst. Mag.*, vol. 7, no. 1, pp. 54–57, Feb. 1987.

[2] E. Eastwood, "Control Theory and the Engineer," *Proc. IEE*, vol. 115, no. 1, pp. 203–211, Jan. 1968.

[3] R. H. Cannon, Jr., *Dynamics of Physical Systems*, McGraw-Hill, 1967.

[4] J. K. Roberge, "The Mechanical Seal," S.B. Thesis, Massachusetts Institute of Technology, Cambridge, MA, May 1960.

[5] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 834–846, Sept.–Oct. 1983.

[6] C. W. Anderson, "Strategy Learning with Multilayer Connectionist Representations," Tech. Rept. TR87-509.3, GTE Laboratories, Waltham, MA, 1987. (This is a corrected version of the report published in *Proc. Fourth International Workshop on Machine Learning*, Irvine, CA, pp. 103–114, June 1987.)

[7] B. Widrow and F. W. Smith, "Pattern-Recognizing Control Systems," *1963 Computer and Information Sciences (COINS) Symp. Proc.*, Washington, DC: Spartan, pp. 288–317, 1964.

[8] B. Widrow, "The Original Adaptive Neural Net Broom-Balancer," *Int. Symp. Circuits and Syst.*, pp. 351–357, May 1987.

[9] A. Guez and J. Selinsky, "A Trainable Neuromorphic Controller," *J. Robotic Syst.*, vol. 5, no. 4, pp. 363–388, Aug. 1988.

[10] V. V. Tolat and B. Widrow, "An Adaptive 'Broom Balancer' with Visual Inputs," *Proc. IEEE Int. Conf. on Neural Networks*, San Diego, CA, pp. II-641–II-647, July 1988.

[11] M. E. Connell and P. E. Utgoff, "Learning to Control a Dynamic Physical System," *Proc. AAAI-87*, vol. 2, pp. 456–460, American Association for Artificial Intelligence, Seattle, WA, 1987.

[12] B. E. Rosen, J. M. Goodwin, and J. J. Vidal, "State Recurrence Learning," *First Annual Int. Neural Network Society Meeting*, Boston, MA, Sept. 1988 (abstract appears in *Neural Networks*, vol. 1, Suppl. 1, p. 48, 1988).

[13] G. E. Hinton, "Connectionist Learning Procedures," Tech. Rept. CMU-CS-87-115, Carnegie-Mellon Univ., Pittsburgh, PA, 1987; to appear in *Artificial Intelligence*.

[14] R. S. Sutton and A. G. Barto, "Toward a Modern Theory of Adaptive Networks: Expectation and Prediction," *Psychol. Rev.*, vol. 88, no. 2, pp. 135–170, 1981.

[15] R. S. Sutton, "Temporal Credit Assignment in Reinforcement Learning," Doctoral Dissertation, COINS Tech. Rept. 84-02, Univ. of Massachusetts, Amherst, 1984.

[16] R. S. Sutton, "Learning to Predict by the Methods of Temporal Differences," *Machine Learning*, vol. 3, pp. 9–44, 1988.

[17] D. Michie and R. A. Chambers, "BOXES: An Experiment in Adaptive Control," *Machine Intelligence 2*, E. Dale and D. Michie, eds., Edinburgh: Oliver and Boyd, pp. 137–152, 1968.

[18] D. E. Rumelhart, G. E. Hinton, and R. W. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, D. E. Rumelhart, J. L. McClelland, and The PDP Research Group, Cambridge, MA: Bradford, 1986.

[19] J. J. Helferty, J. B. Collins, and M. Kam, "A Learning Strategy for the Control of a Mobile Robot That Hops and Runs," *Proc. IASTED-88*, Galveston, TX, pp. 7–11, International Association of Science and Technology for Development, 1988.

[20] J. C. Hoskins and D. M. Himmelblau, "Automatic Chemical Process Control Using Reinforcement Learning in Artificial Neural Networks," *First Annual Int. Neural Network Society Meeting*, Boston, MA, Sept. 1988 (abstract appears in *Neural Networks*, vol. 1, Suppl. 1, p. 446, 1988).

[21] W. T. Miller, "Sensor-Based Control of Robotic Manipulators Using a General Learning Algorithm," *IEEE J. Robotics Automat.*, vol. RA-3, no. 2, pp. 157–165, Apr. 1987.

[22] J. A. Franklin, "Learning Control in a Robotic System," *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Alexandria, VA, pp. 466–470, Oct. 1987.

[23] O. G. Selfridge, R. S. Sutton, and A. G. Barto "Training and Tracking in Robotics," *Proc. IJCAI-85*, pp. 670–672.

Charles W. Anderson received the B.S. degree in computer science from the University of Nebraska in 1978 and the M.S. and Ph.D. degrees in computer science from the University of Massachusetts, Amherst, in 1982 and 1986, respectively. He is currently a Senior Member of the Technical Staff in the Self-Improving Systems Department of GTE Laboratories, Waltham, Massachusetts, where he is studying learning methods for multilayer connectionist networks with a focus on learning methods for control domains, including problems in process and robotic control. In addition to connectionist learning methods and control, his interests include optimization, pattern classification, computer graphics, and simulation.

Doctoral Dissertations

The information about doctoral dissertations should be typed double-spaced using the following format and sent to:

Prof. Bruce H. Krogh
Dept. of Electrical and Computer Engrg.
Carnegie-Mellon University
Pittsburgh, PA 15213

Ohio State University
Iftar, Altug, "Robust Controller Design for Large Scale Systems."
Date: August 1988.
Supervisor: Ümit Özgüner.
Current Address: Department of Electrical Engineering, University of Toronto, Toronto, Ontario M5S 1A4, Canada.

Ohio State University
Barbieri, Enrique, "Modelling and Control of Planar Flexible Structures with Application to an Optical Tracking System."
Date: September 1988.
Supervisor: Ümit Özgüner.
Current Address: Department of Electrical Engineering, Tulane University, 204 Stanley Thomas Hall, New Orleans, LA 70118-5674.

University of Texas at Austin
Cho, Hangju, "Supremal and Maximal Sublanguages Arising in Supervisor Synthesis Problems with Partial Observations."
Date: August 1988.
Supervisor: Steven I. Marcus.
Current Address: Agency for Defense Development (2-2-2), Daejeon, P.O. Box 35, Daejeon, South Korea.

Swiss Federal Institute of Technology, Zurich, Switzerland

Constantinescu, I., "On the Asymptotic Eigenstructure of Multivariable Systems with High Feedback Gain."

Date: March 1988.
Supervisor: Hans P. Geering.
Current Address: Measurement and Control Laboratory, Swiss Federal Institute of Technology, ETH-Zentrum, 8092 Zurich, Switzerland.

University of New Mexico
Park, Hong Bae, "Nominal H^2 Feedback System Optimization with Simultaneous Stabilization Constraints."
Date: May 1988.
Supervisor: Peter Dorato.
Current Address: Department of Electronics, Kyungpook National University, Taegu, Korea 702-701.

University of New Mexico
Park, Hong Bae, "Nominal H^2 Feedback Optimization with Simultaneous Stabilization Constraints."
Date: August 1988.
Supervisor: Peter Dorato.
Current Address: Department of Electronics, Kyungpook National University, Taegu 635, Korea.

Purdue University
King, Andrew, "Discretization and Model Reduction for a Class of Nonlinear Systems."
Date: August 1988.
Supervisor: R. E. Skelton.
Current Address: Hughes Aircraft Company, P.O. Box 92919, Los Angeles, CA 90009.

Purdue University
Hu, Anren, "Modal Cost Analysis of Flexible Structures for Control Design."

Date: January 1988.
Supervisor: R. E. Skelton.
Current Address: Dynacs Engineering Company, 2280 US 19 North, Clearwater, FL 33575.

Purdue University
Collins, Emmanuel, "State Covariance Assignment of Discrete Systems."
Date: May 1987.
Supervisor: R. E. Skelton.
Current Address: Harris Corporation, P.O. Box 137, Melbourne, FL 32901.

University of Manchester Institute of Science & Technology
Muha, P.A., "Incorporation of an Expert System into an Existing Computer-Aided Control Systems Design Package."
Date: June 1988.
Supervisor: Dr. P. A. Cook.
Current Address: Engineering Faculty, Universiti Kebangsaan Malaysia, 43600 UKM, Bangi, Malaysia.

University of Cambridge
Lam, James, "Model Reduction of Delay Systems."
Date: March 1988.
Supervisor: Keith Glover.
Current Address: Department of Applied Mathematics, City Polytechnic of Hong Kong, Nathan Road, Hong Kong.

University of Tennessee
McCullough, Claire L., "Error Considerations in Distributed Estimation."
Date: June 1988.
Supervisor: Dr. J. Douglas Birdwell.
Current Address: Department of Electrical & Computer Engineering, University of Alabama in Huntsville, Huntsville, AL 35899.