Cooperative Coaching in Robot Learning

Jeff G. Schneider
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Christopher M. Brown
Computer Science Dept.
University of Rochester
Rochester, NY 14627

Abstract

Many closed loop learning algorithms perform gradient descent on a cost function with respect to the parameters of a learning controller. We observe that both *local closed loop learners*, which consider only the cost of the current time step, and *optimal control based closed loop learners*, which consider the future effects of control actions, can become stuck in sub-optimal local minima in the cost function. We propose the use of "Cooperating Coaches" to deal with this problem. Each coach attempts gradient descent based on its own cost function and they work together to avoid getting stuck in local minima. When one coach has achieved the best result it can (the gradient for its cost function is zero), another coach takes over to guide the search through the parameter space. We demonstrate cooperative coaching on the problem of curve tracking with an inverted pendulum and show that it yields faster, smoother tracking of target curves by combining the best aspects of two different coaches.

1 Introduction

We use the architecture in Fig. 1 as our basic model for closed loop learning. A sequence of desired outputs for the robot is entered at the left (referred to as a desired curve or trajectory). The fixed controller and the learning controller observe the desired and actual output from the robot. The fixed controller computes a control signal based on a pre-specified method (for example, PID control). The learning controller contains a function approximator mapping actual and desired outputs onto control values. Its output is added to the output of the fixed controller and the result is sent to the robot. The function approximator is modified upon presentation of training data with a learning rule. The fixed controller provides reasonable control signals in all parts of the robot's state space, while the learning controller attempts to find offsets from the fixed controller output that will improve performance. The performance of a controller can be measured with any metric, but common ones are the sum of squared errors between actual and desired states, the sum of the magnitude of all control signals sent out, and the time it takes to pass through a sequence of desired states. These metrics favor accuracy, smoothness, and speed respectively.

The topic of this paper is algorithms for training learning controllers. We will refer to specific instances of learning controller methods as "coaches." A coach guides the execution of the system in order to obtain training data for the learning controller. Training data consists of a set of desired and actual states along with a control signal that the coach suggests the controller should output when it sees those states. In this paper we show how several coaches can work together to produce an improved learning controller. The idea is conceptually similar to the way human athletes train. Coaches give them a series of drills to perform that do not directly relate to their sport, but are expected to help some aspect of their performance.

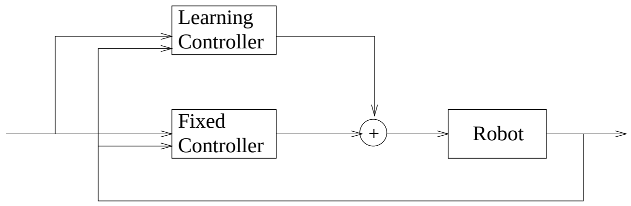

Figure 1: Basic closed loop learner

Learning controllers can be classified according to what aspects of the system's execution are used to compute training data for learning. Many methods, called local closed loop learners, compute training data at each time step, by considering the state, desired state, and error at the current time only. The training data is then used to modify the parameters of the learning controller. Adaptive control, feedback error learning [5], and the use of modified Kohonen maps for robot control [9] are all examples of local closed loop learners. In any task where changing the controller parameters to reduce tracking error at the current time step is always the right thing to do, local learning performs well. However, many difficult dynamic tasks have cost functions that can be optimized only by controllers that sacrifice accuracy during some parts of a task in order to perform better at others. The use of local learning on these tasks can result in the learning getting stuck at a local minimum that is far from the global optimum.

An optimal control [7] formulation for learning can be used to address the shortcomings of local closed loop learners. In optimal control, a cost function is specified for the performance of an entire task. The controller chooses controls that minimize the cost incurred at the current and all future time steps. A controller constructed under this scheme can trade off accuracy for improved overall performance. One way to do this is through the use of the neural net backpropagation algorithm in a scheme called *backpropagation-in-time* [8]. It uses the learning rule from backprop to propagate performance error back in time over an entire task execution. The result is derivatives relating control signals to their effects on the entire task execution. These derivatives are used to update the weights in a neural network controller. A closed loop learner that computes gradients of total cost as a function of controller parameters is an improvement over local closed loop learning, but may still suffer from convergence to sub-optimal local minima.

Any learning method using gradient descent on a cost function may have problems with local minima, regardless of whether it is local or optimal (Jordan and Rumelhart discuss both methods in their paper on forward modeling [4]). Usually, control problems and cost metrics are designed so that there are no spurious local minima or at least they do not interfere with learning. Sanger [10] shows how bad solutions can be avoided by starting with a simple task that has only one local minimum (thus it is also the global minimum) and slowly increasing its difficulty during learning. Unfortunately, many difficult learning problems can not be formulated in a way that removes problematic local minima. In this paper we propose "co-operative coaching" as a way to learn difficult tasks. Section 2 describes a restricted and a generalized algorithm for cooperative coaching. Section 3 presents the results of simulation experiments on curve tracking with an inverted pendulum, and section 4 concludes.

2 Cooperative Coaching

The models in this section are based on the following assumptions:

1. The use of local learning methods fails to produce acceptable performance.
2. While the use of optimal control based learning may produce improved performance, it is still insufficient due to problems with local minima.
3. There is more than one coach available (they may be local or optimal based) and each performs optimization on different cost functions.
4. There is a "true" cost function we want to minimize. Minimizing the cost function employed by each coach is expected to change the controller parameters in a way that will ultimately help minimize the true cost function.

2.1 Cooperating coaches

Fig. 2 is a block diagram of the architecture for cooperating coaches that work together to learn a particular task. During each trial, only one coach is active. It provides the training signal to the learning controller. At the beginning of each trial a decision is made about which coach will be active for the trial.

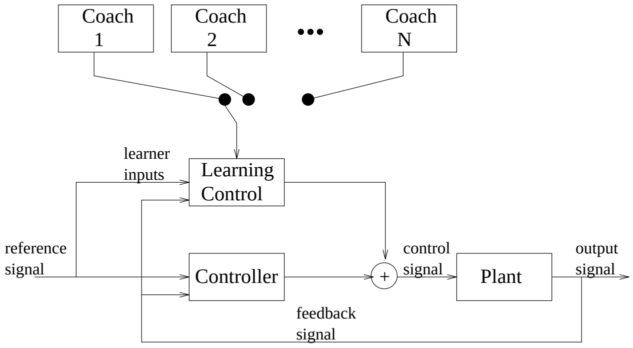

Figure 2: Cooperating coaches

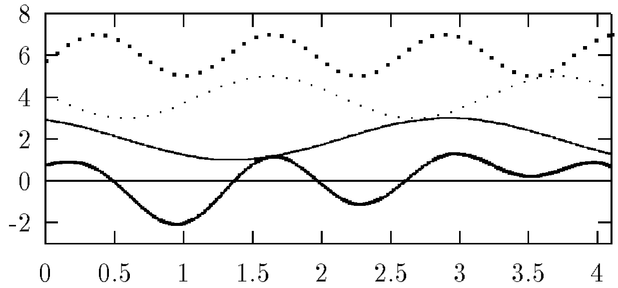

Figure 3: A function minimization example. The bottom curve is the sum of the top three.

Usually, this is best done by considering the performance of the previous few trials.

"Cooperating coaches" is an idea intended to overcome the local minima problems that may occur with gradient descent learning control solutions. We illustrate the idea with a simple example of function minimization. Consider the function at the bottom of Fig. 3. It is the sum of the three sine waves plotted above it. Suppose we use gradient descent and begin at 4.0 on the x axis. The process will quickly stop at a local minimum. Now suppose we start with gradient information from the function at the top. From there we reach a local minimum around 3.5. Then suppose we switch to gradient information from the second function to get to 2.7 and to the third function to get to 1.4. From there we can use the gradient of the original function to get to 0.9. Of course this is not a particularly useful way to do function minimization in general. For robot skills, however, each of the component functions represents one of the coaches that makes use of some heuristic information to guide the search in the right direction. Each of the coaches can suffer from local minima or even instability of the learning process when operated alone, but by switching between them a better solution may be found.

2.2 Cooperating learning controllers

Fig. 4 is a block diagram for a generalized version of cooperating coaches called cooperating learning controllers. This architecture removes many restrictions present in the cooperating coaches model. Cooperat-

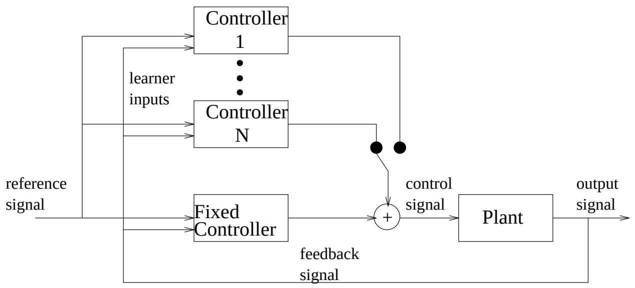

Figure 4: Cooperating learning controllers

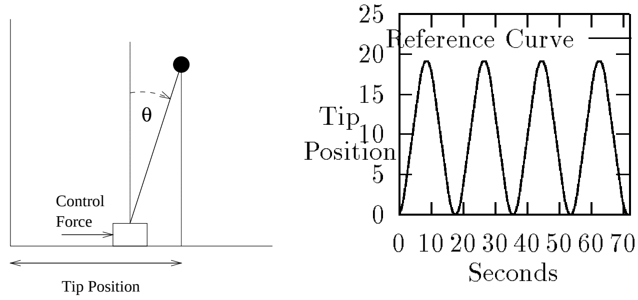

Figure 5: Inverted pendulum tracking task

ing coaches uses a single function approximator operated on by several different coaches. This means that the system is always at only one point in the learning controller parameter space. With cooperating learning controllers, each coach has its own function approximator. This allows parallel consideration of several points in the learning controller parameter space since the function approximator with each coach may represent different points in the space.

Cooperation occurs implicitly in the cooperating coaches model as each coach takes its turn manipulating the learning controller, but it is explicit in the cooperating learning controllers model. Only one coach is active during each trial, but any other learning controller may use its output as a training signal. In this scheme each coach may attempt to track a new curve in its own way when its turn comes up without doing damage to the whole system. If it fails, none of the other learning controllers will train on its output and its errors will not be incorporated into their function approximators. However, when one controller succeeds, any of the others may learn from its success. The selection algorithm must decide which coach to activate *and* which others will watch and learn from it.

This generalization allows additional flexibility. There is only one set of learning controller inputs for cooperating coaches since there is only one function approximator. In cooperating learning controllers, however, each one can have a different set of inputs. One may use only the reference signal and run open-loop, while another makes full use of the robot state. Some parts of a task may be easier to learn with different inputs than others.

When learning is complete, cooperating coaches need only turn off the training from the coaches. With cooperating learning controllers there is an issue about which should execute during normal operation. One possibility is to select the one with the best overall performance. Another is to select different controllers dynamically during execution for different reference curves. This is similar to the "mixture of experts" idea in classification [2] or the combination of control laws in fuzzy control.

3 Simulation Experiments on Curve Tracing with an Inverted Pendulum

This section presents a learning problem, a fixed controller and two simple coaches that satisfy the assumptions for cooperative coaching listed in section 2. We report the results of learning with each of the simple coaches alone for comparison. Finally, we report the results of using both coaches together with the two cooperative models presented above.

Nonlinear, unstable systems are among the most difficult to control. As an example, consider tracing a curve with the tip of an inverted pendulum (also known as cart/pole) as shown in Fig. 5. The input control to this system is a horizontal force applied at the base of the inverted pendulum (the cart). Its output is the horizontal position of the tip of the pole. The goal is to trace the curve as quickly (first term in eq. 1) and accurately (second term) as possible with minimal control effort (third term). There are two additional constraints on the problem: 1) There is a maximum magnitude on the control force that can be applied to the system. 2) If the pendulum ever falls to the horizontal position, the trial ends in failure. The problem is stated more formally as minimize the cost functional:

$$J = \alpha_1(t_f - t_0) + \int_{t_0}^{t_f} \alpha_2(y_d - y)^2 + \alpha_3 u^2 dt \qquad (1)$$

subject to the constraints:

$$\begin{array}{rclrcl} \dot{\vec{x}} & = & f(\vec{x}, u) & u_{min} \leq & u & \leq u_{max} \\ y & = & g(\vec{x}, u) & \vec{x}_{min} \leq & \vec{x} & \leq \vec{x}_{max} \end{array} \qquad (2)$$

where the αs are relative weights, t_0 and t_f are the beginning and ending times, y_d is the desired output, y is the actual output, u is the control force applied, and $\vec{x}$ is the robot state. In practice the goal is to minimize the cost functional over a class of target curves. Here we consider a class consisting of a single target curve at different speeds.

Note that this problem is considerably more difficult than the problem of just stabilizing an inverted pendulum which has been done with a variety of learning algorithms ([1, 6, 3] for example). The size of the state space to be learned is much larger and the desire to track curves as quickly as possible pushes the system to operate at the boundary of its instability. We use the following state equations for our simulations:

x_0	- pole angle	x_1	- pole angular velocity
x_2	- cart position	x_3	- cart velocity
y	- output		

$$\begin{aligned}
\dot{x_0} &= x_1 \\
\dot{x_1} &= c_p(g\sin(x_0) - u\cos(x_0) - c_j x_1 - c_a(x_1)^2) \\
\dot{x_2} &= x_3 \\
\dot{x_3} &= c_c u \\
y &= x_2 + c_l \sin(x_0)
\end{aligned}$$

The constants are based on pole length, coefficients of joint and air friction, and mass of the cart. For this simulation, the length of the pendulum is 15 m, the mass of the cart is 10 kg, and the friction coefficients are 0.01 and 0.02. The friction coefficients were chosen such that they contribute roughly 10% of the torque on average over the reference curves we use. Note that these equations assume a system in which the mass of the pole is insignificant compared to the mass of the cart.

3.1 Instantaneous inverse dynamics

In this section we look at a fixed controller and a local learner that satisfies assumption 4 for cooperative coaching on the problem of curve tracking with an inverted pendulum. They are based on the instantaneous inverse dynamics of the system. At each control cycle, the value of interest is the force that will make the system be exactly on the reference curve at the next control cycle. However, use of the instantaneous inverse dynamics alone is insufficient and thus assumption 1 holds for the inverted pendulum [11].

3.1.1 A fixed controller

Our closed loop learning algorithms start with a fixed controller that is capable of some minimal level of performance. We use the following nonlinear controller (its derivation can be found in [11]):

$$u = \frac{\ddot{y}_d - 14.7\cos(x_0)\sin(x_0)}{1.0 - 1.5\cos^2(x_0)} \tag{3}$$

$$\ddot{y}_d = \frac{2((y_d - y) - \dot{y}\tau)}{\tau^2} \tag{4}$$

where τ is the length of the control cycle. The controller is derived from the simulation state equations, but without the terms due to friction in the state equation for x_1. They are left out in order to test the learning algorithms starting with a controller that does not know the correct inverse dynamics.

We will compare future learning experiments against the performance of the fixed controller by itself. The reference curve is constructed with a trapezoidal velocity curve. It cycles through equal time segments of constant acceleration, constant positive velocity, constant deceleration, and constant negative velocity. It is shown on the righthand side of Fig. 5. We have three goals for tracking these curves and each is handled differently. First, we would like good positional tracking accuracy. In practice, the fixed controller always ensures that accuracy is excellent, therefore we do not discuss accuracy further. Second, we would like to track the curves with the lowest possible control effort. This is measured as the sum of the squared forces as in the discrete form of the third term in eq. 1. This goal is addressed implicitly by the learning algorithms. Finally, we want to track the reference curves as quickly as possible. We accomplish this explicitly by presenting the system with faster reference curves. Slow ones are presented first and successively faster ones are given as the system learns.

The controller is discrete and chooses constant forces to apply over 200 ms intervals. The maximum force is 50 N. The curve marked "Simple Controller" in fig. 6 shows the performance of the fixed controller in terms of control effort and tracking speed. The vertical axis lists control effort and the horizontal axis gives the average speed the curve is traced. One goal is to lower the plot on the vertical axis and the other is to extend it farther right along the horizontal axis. The reason it stops where it does is that the fixed controller drops the pole if asked to trace the curve any faster than 2.5 m/s. Of course, the plot could go on forever to the left since the controller is able to track all slower curves easily. Note that the control effort does not monotonically decrease or increase as a function of curve speed. The interaction of the frequency of the curve being attempted with the natural frequency of the pendulum makes some curves significantly easier or harder than others of similar speed.

3.1.2 Feedback error learning

Feedback error learning is one example of a closed loop learning control scheme designed to learn the instantaneous inverse dynamics of a robot. It was proposed by Kawato and his colleagues [5]. In our experiments with it we use a function approximator having four input variables. They are the current and previous desired tip position and tip velocity. We start with the imperfect fixed controller that does not contain the friction terms and rely on the mechanisms of the learning scheme to adapt to this deficiency. Fig. 6 shows the results of using feedback error learning and compares it to the fixed controller by itself. The results are typical of what has been shown by other authors using feedback error learning. As the system learns a better approximation of the instantaneous inverse dynamics the total control effort is reduced and assumption 4 for cooperative coaching is satisfied. Unfortunately, feedback error learning has failed to do what we ultimately want: to trace the reference curve faster. It never succeeded in learning to do something that the fixed controller by itself could not do.

3.2 The smoothing coach

There are problems with using a controller based only on the instantaneous effects of its controls. In order to improve the controller, it should consider the future effects of its actions as in optimal control. In [11] we describe such a coach for the inverted pendulum. It is called a smoothing coach and we show that

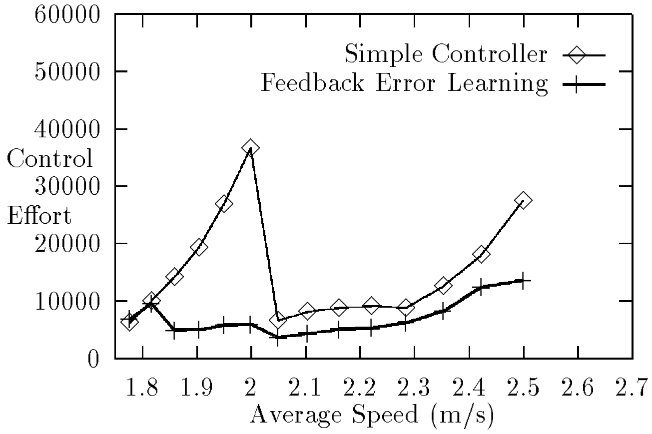

Figure 6: Performance of feedback error learning

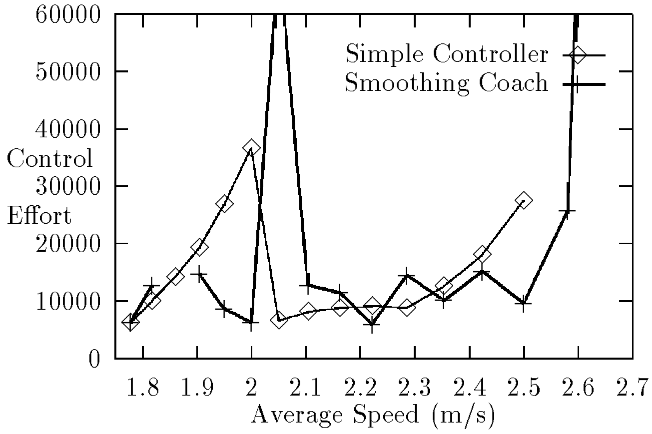

Figure 7: Performance of smoothing coach

it satisfies assumption 4 for cooperative coaching in this section.

The input space of the smoothing coach's function approximator is three-dimensional. The dimensions are the current angular position of the pole, the current angular velocity of the pole, and the desired tip acceleration as given in eq. 4. Observation of the inverted pendulum in operation shows that it usually falls over as the result of wild oscillations in the control signal. These oscillations also use high accelerations and thus increase the control effort. Therefore, we chose to use a smoothing coach that modifies the control signal by passing a low-pass filter over it (each control action is replaced by the average of itself with its temporal neighbors). After each trial this smoothing coach is used to transform the sequence of controls actually sent into a set of desired control signals for the function approximator to train on. Fig. 7 shows the results of using this smoothing coach. The control effort for some speeds is higher than with the fixed controller, but the system is now able to trace the trajectory up to a speed of 2.7 m/s. However, the coached controller causes the system to drop the pendulum at a speed of 1.9 m/s even though the fixed controller alone is capable of performing at that speed. Two things can be concluded from observing these experiments. First, the smoothing coach worked to increase stability and allow the system to track the reference curve faster. Second, a large acceleration is still needed to reverse the direction of motion during tracking and the coach acted to inhibit that acceleration. That inhibition resulted in higher control effort and instability at some speeds.

Despite the success of the smoothing coach, it would be difficult for an optimal control formulation of the inverted pendulum problem to lead us to the best solution. Further details of how such formulations of this problem can encounter sub-optimal local minima, and a demonstration that assumption 2 for cooperative coaching holds can be found in [11]. Dynamic programming often solves problems of this type. However, a straightforward application of it to this problem fails because the optimal trajectories often run near the boundary at which the maximum force can no longer stand the pole up. Errors due to quantization of the state space tend to push the system over the boundary.

3.3 Experimental results

In our experiments, feedback error learning does better against some performance metrics (lower control effort) while the smoothing coach controller does better against others (faster tracking speed). In the following experiments these two systems are combined as cooperating coaches and as cooperating learning controllers. Each coach considers the same data as it had previously. We use the following heuristic algorithm to control the selection of coaches.

1. Increase the desired tracking speed.
2. Run trials with the smoothing coach until the trajectory can be tracked stably.
3. *Cooperating learning controllers only:* Train the feedback error learning controller on the stable execution by the smoothing coach.
4. Switch over to the feedback error learning coach.
5. Run trials until no further improvement in control effort is observed
6. *Cooperating learning controllers only:* Train the smoothing coach controller on any feedback error learning execution that lowers control effort.
7. Repeat steps 1-7 until step 2 fails to track a faster trajectory stably.
8. *Cooperating learning controllers only:* Executions are done with the feedback error learning controller.

Experimental results using cooperating coaches are shown in Fig. 8. The combined system tracks trajectories as fast as 2.7 m/s, just as the smoothing coach

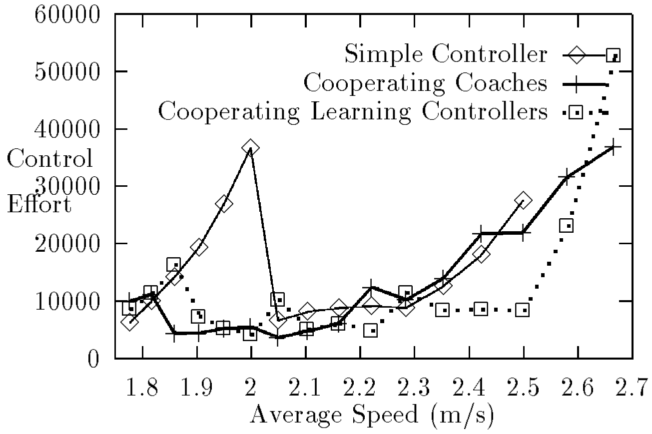

Figure 8: Performance of cooperating coaches and learning controllers

does when operated alone, while correcting the problem the smoothing coach had with the 1.9 m/s trajectory. It also uses 55.1% less control effort than the smoothing coach does when operated alone. It does, however, use 40.0% more control effort than feedback error learning at the speeds feedback error learning is capable of doing by itself. The resulting system successfully combines the strengths of the two controllers to produce a superior result.

Results of experiments using cooperating learning controllers are shown in Fig. 8. This architecture is also capable of tracking trajectories up to 2.7 m/s. It uses 60.7% less control effort than the smoothing coach alone and 19.0% more than feedback error learning on the trajectories it can track. These numbers are better than the cooperating coaches model, but not enough to make a strong statement about which model is preferable. The cooperating learning controllers model is at least as powerful as cooperating coaches, because cooperating coaches is just a special case where each controller uses the same parameters in its function approximation and always learns from the active coach. However, the cooperating coaches model may be preferable based on its simplicity.

4 Conclusions and Open Problems

We proposed a method to combine learning controllers with different properties to obtain a new one with the best properties of each. This method addresses the problem of local minima in the learning process by switching between coaches. One coach actively trains the system at each execution, but control may be switched between several coaches during the training. In our experiments, two controllers were combined to produce one that accomplishes faster trajectory following and reduced control effort. Although our experiments used only two controllers, it is possible to use any number. The controller's coaches might differ in the type of heuristic, the parameters of a particular filter, or function approximation techniques. Each learning controller could "get its shot" at reference curves from the training set and the first one to succeed would try to pass "the answer" on to the others.

An important open problem is to find a suitable selection algorithm that chooses the active controller during learning. We use simple algorithms in our experiments, but it is likely that the selection algorithm is crucial for good performance. One possibility is to label each coach as good for improving some aspect of the performance metric. When a reference curve is presented, the selection algorithm could decide which aspect of the metric needs to be improved most for that particular curve and choose the coach most likely to do so. A more sophisticated algorithm would learn which coaches are best at making which improvements based on their performance on previous curves.

There is a danger that training by a series of different cooperating controllers might cause interference and result in a non-convergent process. At the present we have not investigated the conditions when this might occur because we have explicitly chosen to develop algorithms for maximizing performance in cases where typical assumptions about plants and reference signals do not hold. However, we view the further mathematical analysis of these algorithms as an interesting and important problem.

References

[1] A. Barto, R. Sutton, and C. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 1983.

[2] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Compuation*, 1991.

[3] M. Jordan and R. Jacobs. Learning to control an unstable system with forward modeling. In *Advances in Neural Information Processing Systems*, 1990.

[4] M. Jordan and D. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16, 1992.

[5] M. Kawato, Y. Uno, M. Isobe, and R. Suzuki. A hierarchical neural network model for voluntary movement with application to robotics. *IEEE Control Systems Magazine*, 8, 1988.

[6] J. Koza and M. Keane. Cart centering and broom balancing by genetically breeding populations of control strategy programs. In *International Joint Conference on Neural Networks*, 1990.

[7] F. Lewis. *Optimal Control*. John Wiley and Sons, 1986.

[8] B. Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, pages 263–269, 1989.

[9] H. Ritter, T. Martinetz, and K. Schulten. *Neural Computation and Self-Organizing Maps*. Addison-Wesley, 1992.

[10] T. Sanger. Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE Transactions on Robotics and Automation*, 10:323–333, 1994.

[11] J. Schneider. *Robot Skill Learning Through Intelligent Experimentation*. PhD thesis, University of Rochester, December 1994.