A COMPARISON OF SUPERVISED AND REINFORCEMENT LEARNING METHODS ON A REINFORCEMENT LEARNING TASK

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

Abstract

The "forward modeling" approach of Jordan and Rumelhart has been shown to be applicable when supervised learning methods are to be used for solving reinforcement learning tasks. Because such tasks are natural candidates for the application of reinforcement learning methods, there is a need to evaluate the relative merits of these two learning methods on reinforcement learning tasks. We present one such comparison here on a task involving learning to control an unstable, non-minimum phase, dynamic system. The comparison shows that the reinforcement learning method used performs better than the supervised learning method. An examination of the learning behavior of the two methods indicates that the differences in performance can be attributed to the underlying mechanics of the two learning methods, which provides grounds for believing that similar performance differences can be expected on other reinforcement learning tasks as well. This suggests that there is a set of tasks for which reinforcement learning methods are naturally applicable and more appropriate to use.

1 Introduction

The process of learning can often be viewed as improvement of some measure of performance by appropriate modification of the learning system's behavior. It is usually assumed that there is a teacher in the environment that supplies the learning system information pertinent to this performance measure.[1] Several researchers have suggested that learning tasks can be classified based on the quality of the training information provided by the teacher (e.g., [3, 9]). Two major classes of tasks can be distinguished based on this criterion: *supervised learning tasks* and *reinforcement learning tasks*. In supervised learning tasks, the teacher provides target outputs or an error vector that specifies how the learning system should modify its actions in order to improve performance. Thus the teacher can be seen as instructing the learning system, perhaps indirectly, about the actions to perform in order to improve performance. In contrast, the role of the teacher in reinforcement learning tasks is more evaluative than instructional, and the teacher is sometimes called a *critic* because of this role. The critic provides evaluations of the learning system's actions as training information, leaving it to the learning system to determine how to modify its actions so as to obtain better evaluations in the future. In other words, the critic in reinforcement learning does not tell the learning system *what to do* to improve performance, while the teacher in supervised learning does. Supervised and reinforcement learning *methods* reflect this difference in the content of the training information. While supervised learning methods are primarily devised to store the training instances in a manner that enables good generalization, reinforcement learning methods have to incorporate an additional exploration process that can be used to discover the appropriate actions to store.

As discussed in [3, 9], there is a distinction between learning *tasks* and learning *methods* that is akin to Marr's [10] distinction between computational and algorithmic levels. This distinction is of importance when attempting to evaluate the utility of a learning method or family of methods. Often, any of several different methods can be used for solving a class of tasks. Likewise, a single method may be applicable to several classes of tasks. This results in considerable confusion when one attempts to evaluate the relative merits of specific methods employed for solving specific problems. As pointed out by Jordan and Rumelhart [9], while it is always possible to use supervised learning methods to solve reinforcement learning tasks, and vice versa, it might be inefficient to do so in practice. Thus, for example, reinforcement learning methods are widely regarded as being inefficient when compared to supervised learning methods. Although this belief is based primarily on performance comparisons on supervised learning tasks, there is a tendency to overlook the task dependency and to assume that reinforcement learning methods are always less efficient. But is this assumption justified? In order to find out, one needs to ask the question: how do supervised and reinforcement learning methods compare on reinforcement learning

[1] Except in the case of *unsupervised learning* tasks, wherein the necessary information can be considered to be built into the learning system (see [2, 3]).

tasks? This paper presents one such comparison and represents a step towards answering this question.

The learning control task we have selected for the comparison involves learning to control an unstable dynamic system: the much-studied one-dimensional cart-pole system [18, 11, 4, 13, 5, 1, 8, 19]. We compare the performance of two learning methods on this task. The first is a supervised learning method that involves using an internal critic to obtain supervised training information for the controller. This approach has been used by Jordan and Jacobs [8] on the cart-pole task. The second learning method is based on reinforcement learning and has been implemented using the Jordan-Jacobs work as a reference, thereby facilitating a direct comparison.

2 The learning control task

The cart-pole learning control task, also known as the pole-balancing or inverted pendulum problem, involves learning to balance a rigid pole mounted on a mobile cart by applying appropriate forces to the cart. As depicted in Figure 1, the cart is restricted to move along a one dimensional track of bounded length. The position and velocity of the cart (x, $\dot{x}$) and the pole (θ, $\dot{\theta}$) represent the state of the cart-pole system, and the dynamics of the system are given by the equation

$$\begin{bmatrix} M+m & ml\cos\theta \\ ml\cos\theta & I \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} - \begin{bmatrix} ml\sin\theta \\ mgl\sin\theta \end{bmatrix} \dot{\theta}^2 = \begin{bmatrix} F \\ 0 \end{bmatrix},$$

where M and m are the masses of the cart and the pole respectively, l is *half* the length of the pole, I is the inertia of the pole around the base, g is the acceleration due to gravity, and F is the force applied to the cart. Values used in our simulations are $M = 1.0$kg, $m = 0.1$kg, and $l = 0.5$m.

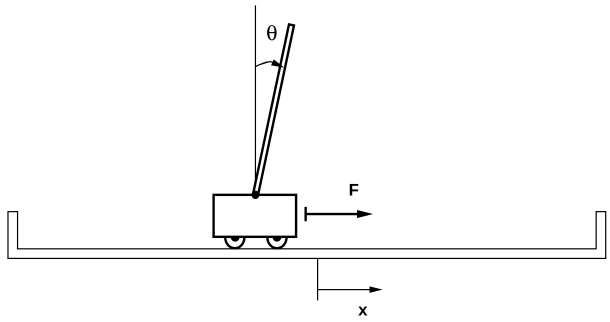

Figure 1: *The cart-pole system.*

Training information to the learning system is in the form of a failure signal f_{ext} that occurs when the pole falls beyond a critical angle or the cart hits either end of the track. Since this information is an evaluation rather than instructions on how to balance the pole, the learning task is best viewed as a reinforcement learning task. Furthermore, the training signal occurs very infrequently, creating a difficult *temporal credit assignment problem* [14]. Solutions to this credit assignment problem usually involve training an internal critic network to generate predictions of future evaluations. In the absence of external evaluations, these predictions are used by the learning system as if they were actual evaluations received from the environment. Examples of algorithms used for learning to predict future evaluations include the TD(λ) (Temporal Difference) family of algorithms developed by Sutton [15] and the related Q-learning algorithm developed by Watkins [16].

For the pole-balancing task, Jordan and Jacobs [8] used the temporal difference algorithm outlined below to train the internal critic. We used the same algorithm in the simulations reported in this paper. The algorithm has been designed to predict z_t, the reciprocal of the time until failure, given the current time step t. Hence z_t should equal 1 on the time step immediately preceding failure, and should also satisfy the relationship

$$\frac{1}{z_t} = \frac{1}{z_{t+1}} + 1.$$

Therefore the error used to train the internal critic network to estimate the reciprocal of time until failure is:

$$e_t = \begin{cases} 1 - \hat{z}_{t-1} & \text{if there is a failure at time } t, \\ \frac{1}{1+1/\hat{z}_t} - \hat{z}_{t-1} & \text{otherwise,} \end{cases} \tag{1}$$

where $\hat{z}_t$ is the estimate produced by the internal critic. Moreover, the evaluation (failure signal) seen by the learning system at each time step becomes

$$f_t = \begin{cases} f_{ext} & \text{if there is a failure at time } t, \\ \hat{z}_t & \text{otherwise.} \end{cases} \tag{2}$$

3 The "forward modeling" approach of Jordan and Jacobs (1990)

The network architecture used by Jordan and Jacobs [8] is shown in Figure 2. While very similar to architectures used previously (e.g., [4, 1]), there is an important difference, namely, the inclusion of connections (shown as bold lines in Figure 2) through which the output of the controller is fed as input to the internal critic.[2] It is through these connections that errors are propagated back to serve as supervised training signals for the controller. This kind of architecture has been described in detail by Jordan and Rumelhart [9], who present the "forward modeling" approach as a method of transforming supervised training information in a *distal* coordinate space into a *proximal*

[2]Jordan and Jacobs refer to the internal critic as the *forward model*.

coordinate space. A similar architecture has been suggested by Werbos [17], and Munro [12] has used such an architecture on a reinforcement learning task.

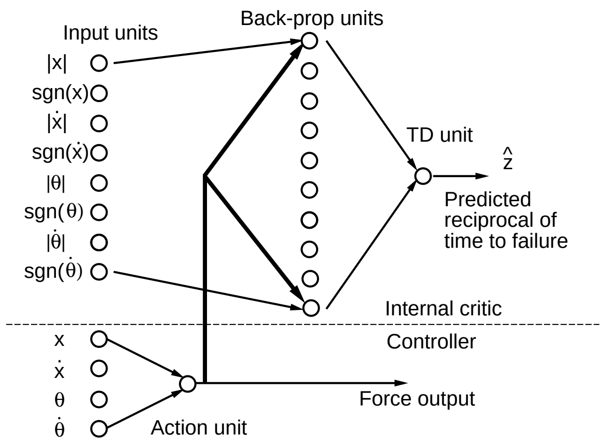

Figure 2: *The network architecture used in all the simulations presented in this paper. The internal critic network has 8 input units, a hidden layer of 10 back-propagation units, and a single temporal difference output unit. The controller has 4 input units and a single action unit. In simulations of the supervised learning method, a "noisy" linear unit was used as the action unit, while in simulations of the reinforcement learning method, a stochastic real-valued (SRV) unit [7] was used.*

For the cart-pole learning task, the environment does not provide supervised training information for the controller even in the distal coordinate space, which in this case is just the unidimensional space of failure signals. However, since the goal of the controller is to avoid failure, an appropriate "desired" failure signal is zero, and the appropriate distal error for the controller is zero minus the failure signal f_t.[3] This distal error for the controller is back-propagated through the internal critic network and through the connections to the controller to provide a proximal error for the controller. The weights of the controller are adjusted so as to minimize the proximal error, which in turn results in minimizing the distal error and hence driving the failure signal to the desired value of zero.

The internal critic network is also trained simultaneously with the controller. In its case, however, back-propagation is used to adjust the weights so as to minimize the prediction error defined in Equation (1).

[3] Recall that the output of the internal critic (i.e., the predicted reciprocal of time until failure) is substituted for the failure signal when there is no failure signal from the environment (see Equation (2)).

4 The reinforcement learning approach

The architecture of the network used in the reinforcement learning approach was identical to the one used in the supervised learning approach except for one difference (see Figure 2). The controller's linear action unit was replaced by a reinforcement learning unit. Specifically, we used the stochastic real-valued (SRV) unit described in [7]. The SRV unit has been designed to generate real-valued outputs stochastically and to use the resultant reinforcement values to adjust its weights so as to maximize the expected reinforcement over time [6]. The unit does this by estimating the local gradient of the reinforcement with respect to its weights and using this estimate to perform gradient ascent (see [6] for details).

In this approach, training information for the controller is a reinforcement signal r_t that is computed directly from the failure signal f_t as

$$r_t = 1.0 - f_t.$$

Maximizing the expected reinforcement therefore results in the desired effect of minimizing the failure signal. The internal critic network is also trained simultaneously, using back-propagation to adjust the weights so as to minimize the prediction error (Equation 1), as in the supervised learning approach.

5 Results of the comparison

We ran 100 training runs each of the supervised and reinforcement learning methods. Each training run represents an attempt to train a network with small initial random weights to balance the pole. A training run lasted until either the pole was balanced for 1000 consecutive time steps (as in ref. [8]) or 30000 failures occurred during the course of the run. Following ref. [8], in all our simulations, the controller produced real-valued forces, and we restarted the cart-pole system in a random state after each failure. We also used identical input representations (see Figure 2), which take advantage of the symmetries in the cart-pole system dynamics. Furthermore, the learning rate parameters used for the supervised learning network were identical to those used in ref. [8], and instead of optimizing these parameters for the reinforcement learning network, we used the *same* parameters as were used in the supervised learning network. These learning rate parameters were 0.05 for the controller and 0.3 for the internal critic network except on the connections from the controller, where the learning rate was 0.9.

Additionally, in the simulations using the supervised learning method, we incorporated Jordan and Jacobs' assumption that disturbances in the form of additive noise having a uniform distribution on $[-0.001, 0.001]$ act on the output of the controller. The role of these disturbances is discussed in Section 6. In order to evaluate the effect of such disturbances on the reinforcement learning method, we also ran 100 runs of the reinforcement learning method with identical noise added to the output of the controller.

When the supervised learning method was used, 72 out of 100 training runs resulted in success, with the remaining 28 failing to balance the pole on any of 30000 attempts. The average number of failures before success over all the successful training runs was 11164.5 failures, and the median was 9244 failures. In the case of the reinforcement learning method, 93 training runs were successful, with an average number of failures before success of 2385.89 failures, and a median of 1148 failures. When disturbances were added to the output of the reinforcement learning controller, the number of successful runs dropped to 88. The average number of failures before success over all the successful runs was 2689.3 failures and the median was 1383.5 failures. Histograms of the distribution of the successful training runs in terms of the number of failures before success for each learning method are shown in Figure 3.

These results indicate a substantial difference between the two learning methods in terms of learning speed and overall learning behavior. We will discuss the reasons for these differences in the next section.

6 Discussion and conclusions

One of the reasons that comparison on the cart-pole task is particularly meaningful is the necessity of an internal critic for predicting future evaluations in order to tackle the temporal credit assignment problem. The internal critic is necessary *regardless* of the learning method employed, and has to be built through training experience. The supervised learning method needs the internal critic network to translate (via back-propagation) distal errors in evaluations into proximal errors in actions of the controller, while the reinforcement learning method needs the internal critic to obtain evaluations in the absence of environmental evaluations. Therefore the difference in learning performance of the two methods essentially stems from the accuracy of the internal critic that is built during training and the manner in which it is used, and *not* from any bias in the task setup or the training information received.[4]

Let us first consider the problem of building an accurate internal critic. If the internal critic is to produce accurate predictions over all of its input space, it is vital that it "sees" inputs that are well distributed over its entire input space. In other words, exploration of the (state, action) input space is imperative. The two methods under consideration rely on different mechanisms to effect this exploration. In the supervised learning method, external disturbances in the form of additive white noise are assumed to affect the control output *at all times*, perturbing the actions of the controller and hence introducing variations in the inputs to the internal critic. A potential problem with this approach is the lack of direct control over the noise and hence over the exploration. While higher magnitudes of noise will enable better exploration—leading to a more accurate internal critic and thus to faster acquisition of control—they will be detrimental to the long-term performance of the controller. Conversely, lower noise levels will ensure stable control after learning but will slow the learning of a useful internal critic and hence the acquisition of control.

In contrast, active exploration of the action space is an integral part of reinforcement learning methods. In order to discover the best output to produce, reinforcement learning methods generate outputs stochastically and use the resultant evaluations to guide the search for the best output. On the cart-pole task, using reinforcement learning therefore ensures the desired exploration of the input space of the internal critic. Moreover, reinforcement learning methods have to directly control the stochasticity of the output in order to exhibit stable learning behavior. This means that as learning proceeds and performance on the task improves, the exploratory variations in the controller's output decrease, leading to a more stable controller.

The manner in which the internal critic is used to train the controller also has a critical impact on the learning behavior. As discussed above, the supervised learning method is based on the idea of propagating distal errors back along the connections from the controller to the internal critic to serve as supervised training information for the controller. Jordan and Rumelhart [9] show how the back-propagation process in effect computes the gradient of the output of the internal critic with respect to the controller's output. If the controller's training information is to be meaningful, the internal critic has to be reasonably accurate at the current point in (state, action) space, *and* the gradient computed using the internal critic has to have a positive inner product with the true gradient of the evaluation function with respect to the controller's outputs (see ref. [9], pp. 11-13). We have already discussed the factors governing the first condition above. The gradient, however, is defined mainly by the weights on the connections from the controller to the internal critic. Therefore the supervised learning method is critically dependent on the weights on these connections.

But because of the redundancy of the information pro-

[4] If, for example, the environment provided appropriate evaluations at all times, then the reinforcement learning method would not be required to build an internal critic, while the supervised learning method would still have to do so. Such a situation would therefore be biased in favor of the reinforcement learning method.

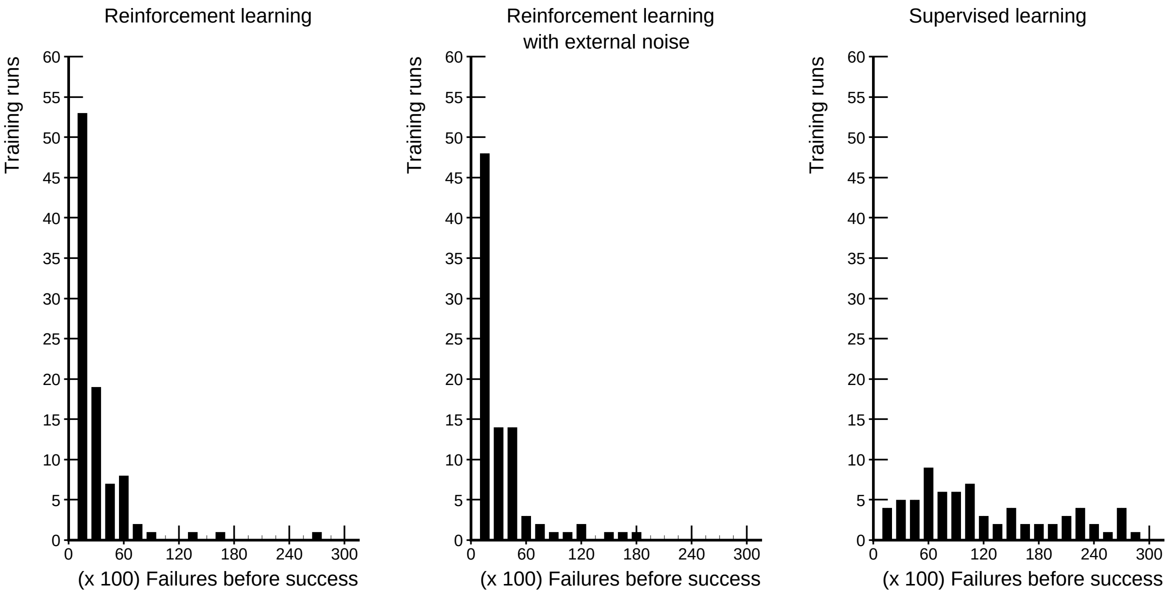

Figure 3: *The distributions of the successful training runs in terms of the number of failures before success using each learning method.*

vided to the internal critic via these connections,[5] the internal critic can make accurate predictions even with arbitrary weights on these connections. While it is true that, in practice, back-propagation tends to drive these weights towards reasonable values, there is reason to believe that the predictions of the internal critic become accurate long before these weights take on meaningful values. This is because accurate predictions are possible independent of these weight values, and once the predictions are accurate, there is no output error to drive the adaptation of these weights. Fully aware of this problem, Jordan and Jacobs [8] sought to mitigate its effect by increasing the learning rate on the connections from the controller to the internal critic. Nonetheless, due to this problem, the provision of meaningful gradient information to the controller is delayed, thereby slowing learning.

In comparison, the reinforcement learning method relies only on the *output* of the internal critic for its training information. Because the reliability of the predictions of the internal critic is probably higher than that of the gradient estimates obtained using the internal critic's weights, the reinforcement learning method gets meaningful training information more frequently than the supervised learning method. The drawback of using reinforcement learning, however, is that the gradient of the output of the internal critic with respect to the controller's output has to be estimated stochastically. Nevertheless, the results presented in the previous section indicate that estimation of the gradient by the reinforcement learning method can be faster than estimation of the gradient through the weights of the internal critic.

The relative performances of supervised and reinforcement learning methods on the cart-pole task, together with the above analysis of the reasons that lead to the observed difference in learning behavior, suggest that on some of the tasks studied by connectionist researchers, it might be more efficacious to use reinforcement learning methods. Of course, one cannot draw any definite conclusions based on a single comparison. It is hoped, however, that the results presented here serve to indicate the necessity and importance of making additional comparisons.

Acknowledgements

I would like to thank Michael Jordan for providing the simulation package using which these simulations were performed, and more importantly, for his advice and many helpful discussions. Thanks are also due to Robert Jacobs for his help in replicating the results described in [8], and to Andrew Barto, Richard Yee, and Roderic Grupen for many helpful comments on drafts of this paper. This material is based upon work supported by the Air Force Office of Scientific Research, Bolling AFB, under Grant AFOSR-89-0526 and by the National Science Foundation under Grant ECS-8912623.

[5] Providing the controller's output to the internal critic is redundant because the controller's output is a function of the state of the cart-pole system, which is already provided as input to the internal critic.

References

[1] C. W. Anderson. Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, 9(3):31–37, 1989.

[2] H. B. Barlow. Unsupervised learning. *Neural Computation*, 1:295–311, 1989.

[3] A. G. Barto. Some learning tasks from a control perspective. COINS Technical Report 91-122, University of Massachusetts, Amherst, MA, 1991.

[4] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:835–846, 1983.

[5] M. E. Connell and P. E. Utgoff. Learning to control a dynamic physical system. *Comput. Intell.*, 3:330–337, 1987.

[6] V. Gullapalli. Associative reinforcement learning of real-valued functions. Technical Report 90-129, University of Massachusetts, Amherst, MA 01003, 1990.

[7] V. Gullapalli. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, 3:671–692, 1990.

[8] M. I. Jordan and R. A. Jacobs. Learning to control an unstable system with forward modeling. In D. S. Touretzky, editor, *Advances in neural information processing systems II*. Morgan Kaufman, San Mateo, CA, 1990.

[9] M. I. Jordan and D. E. Rumelhart. Forward models: Supervised learning with a distal teacher. Center for Cognitive Science Occasional Paper #40, Massachusetts Institute of Technology, Cambridge, MA, 1990.

[10] D. Marr, editor. *Vision*. W. H. Freeman, San Francisco, 1982.

[11] D. Michie and R. A. Chambers. BOXES: An experiment in adaptive control. In E. Dale and D. Michie, editors, *Machine Intelligence 2*, pages 137–152. Oliver and Boyd, 1968.

[12] P. Munro. A dual back-propagation scheme for scalar reward learning. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, pages 165–176, Seattle, WA, 1987.

[13] O. Selfridge, R. S. Sutton, and A. G. Barto. Training and tracking in robotics. In *Proceedings of the Ninth International Joint Conference of Artificial Intelligence*, Los Angeles, CA, August 1985.

[14] R. S. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts, Amherst, MA, 1984.

[15] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[16] C. J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, Cambridge University, Cambridge, England, 1989.

[17] P. J. Werbos. Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(1):7–20, 1987.

[18] B. Widrow and F. W. Smith. Pattern-recognizing control systems. In *Computer and Information Sciences (COINS) Proceedings*, Washington, D.C., 1964. Spartan.

[19] A. P. Wieland. Evolving controls for unstable systems. In D. S. Touretsky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton, editors, *Connectionist Models: Proceedings of the 1990 Summer School*. Morgan Kaufmann Publishers, 2929 Campus Drive, Suite 260, San Mateo, CA 94403, 1991.