Learning from an Automated Training Agent

Jeffery A. Clouse
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
clouse@cs.umass.edu

Abstract

A learning agent employing reinforcement learning is hindered because it only receives the critic's sparse and weakly informative training information. We present an approach in which an automated training agent may also provide occasional instruction to the learner in the form of actions for the learner to perform. The learner has access to both the critic's feedback and the trainer's instruction. In the experiments, we vary the level of the trainer's interaction with the learner, from allowing the trainer to instruct the learner at almost every time step, to not allowing the trainer to respond at all. We also vary a parameter that controls how the learner incorporates the trainer's actions. The results show significant reductions in the average number of training trials necessary to learn to perform the task.

1 INTRODUCTION

In reinforcement learning, an automated agent attempts to develop a policy that indicates the actions to choose in performing a multiple-step task. The automated agent performs on-line, trial-and-error experiments in the task, receiving occasional feedback from an environmental critic in the form of simple, scalar signals that indicate the value of the state in which the feedback is received. Using methods such as actor/critic architectures (Sutton, 1984) and Q-learning (Watkins, 1989), the learner adapts its policy to ensure that it acquires high levels of scalar feedback in the future.

Many researchers have demonstrated success with reinforcement learning methods (Gullapalli, 1992; Lin, 1992; Whitehead, 1992; Barto, Bradtke & Singh, 1993). However, the feedback provided by the environmental critic is only weakly informative. Suppose, after performing a series of actions, the learner receives a negative scalar signal from the critic, signifying that the current state is not desirable. The critic's infrequent feedback does not indicate the correct action to choose, the actions that were responsible for the feedback, nor a sequence of actions that might have been better. The learner is hindered because it only receives this weakly informative and sparse training information.

To alleviate this deficiency, we add another source of training information to the learning scenario—a training agent. In Figure 1, the unshaded portion represents the standard reinforcement learning scenario, with a *task*, a *learning agent*, and a *critic*. The shaded portion represents the addition of the *training agent*, which observes the task state and provides instruction to the learning agent.

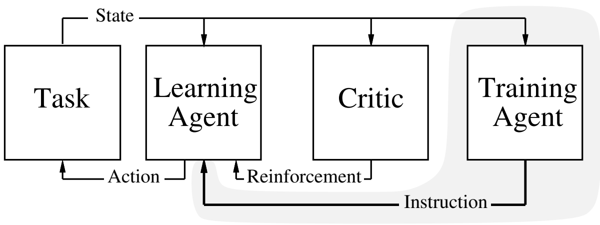

Figure 1: Trainer Providing Instruction

In the next section, we present our approach, and discuss how our work relates to other work on instructing learning agents. In Section 3 we describe an empirical evaluation of our approach in which we achieved significant improvements in learning rate and in learned performance versus learning without a trainer. We then discuss areas for further research, and present conclusions.

2 OUR APPROACH

Placing a training agent in the reinforcement learning scenario raises several questions. Each is a dimension along which research in this area can be described.

Firstly, what form of instruction does the trainer supply to the learner? Secondly, when should the trainer provide the learning agent with instruction? Finally, how does the learner incorporate the trainer's instruction into its developing policy?

Although many choices exist for responding to these questions, our approach is based on specific answers. Below, we take the questions in turn, presenting how our approach handles each, and describing the approaches of related research.

• *What* form of instruction does the training agent provide?

Related work has examined training agents that provide the learner with: actions that the learner should perform (Clouse & Utgoff, 1992; Lin, 1992), the exact state values (Utgoff & Clouse, 1991), and IF-THEN rules (Gordon & Subramanian, 1994; Maclin & Shavlik, 1994).

In our approach, the automated trainer instructs the learning agent by giving actions that the learner should perform. We choose this form of advice for two main reasons. Firstly, individual actions are an inherent form of instruction for an automated training agent to provide: by building an agent that can perform the task, one has built a trainer. Secondly, as will be discussed later, incorporating the trainer's actions into the learner's developing policy requires few changes to the underlying reinforcement learning methods.

During training, the trainer in our approach evaluates the current task state and makes a decision about the action it would perform, which it then suggests to the learner. We assume that the trainer can perform the task at a moderate level of expertise, so that the instruction it provides to the learner is informative.

• *When* does the trainer provide instruction?

Many trainer-augmented systems allow instruction to be provided at the whim of a human trainer (Lin, 1991; Clouse & Utgoff, 1992; Maclin & Shavlik, 1994). Although these systems exhibit good performance, they do not address the issue of when the instruction should be provided, because they allow a human to guide the system in an unprincipled manner.

In order to evaluate policies for deciding when to instruct the learner, one needs to examine automated trainers that employ particular policies: One has more control over an automated trainer than a human.

In an early system that employs an automated strategy for supplying instruction, the learner improves its performance by asking for help when it has low confidence in its own action choices, based on the size of the difference of previous state evaluations (Utgoff & Clouse, 1991).

Deciding when the trainer should instruct is still an open question. Example policies for deciding when to provide the learner with instruction include: functions of time, which change with respect to the number of actions executed; functions of the learner's performance, which may provide more instruction when the learner is doing poorly; and functions of the current task state, which allow the trainer to instruct the learner in difficult situations.

In our approach, the training agent employs a straightforward strategy based on an instruction rate parameter that establishes the percentage of time steps on which the trainer instructs the learner. The rate parameter specifies the probability that the trainer will give the learner an action on any given time step. When the trainer does not provide an action, the learner chooses an action based on its developing policy. One can also view this approach as one where the trainer always provides actions to the learner, but the learner only accepts them a certain percentage of the time.

• *How* does the learner incorporate the trainer's instruction into its policy?

The problem that must be addressed is how to take non-scalar feedback, such as actions or IF-THEN rules, and incorporate it into the reinforcement learning mechanisms that were designed to learn from scalar feedback.

Two related systems address the question by circumventing the reinforcement learning methods. Maclin and Shavlik (1994) apply the the trainer's compiled instruction *directly* to the learner's policy using techniques from knowledge-based neural networks (Towell, Shavlik & Noordewier, 1990); thus, converting high-level advice into weight changes applied to the network that represents the learner's policy. Gordon and Subramanian (1994) rely on genetic algorithms, instead of reinforcement learning, in order to learn from environmental feedback. In their system, after the IF-THEN rules are converted into operational rules with the help of domain knowledge, they become part of a population of rules that are continually refined by the genetic operators as the learner performs the task.

In our approach, the learning agent incorporates the trainer's actions into its policy by first performing the action, just as if the learner itself had chosen it. Then, because the action was picked by the trainer, the learner receives a predetermined amount of feedback, which it treats as if it was received from the critic. The control of the reinforcement learning mechanism must change to allow actions and reinforcement to come from the trainer, but the update formulas will remain the same.

Clouse and Utgoff (1992) and Lin (1992) demonstrate success via similar approaches. In addition to performing the steps presented above, Clouse and Utgoff (1992) discard state history information so that the learner's previous actions do not get credited for the feedback received when performing the trainer's action. This change to the reinforcement methods was based on the assumption that the trainer would only provide instruction in situations where the learner's most recent actions were inappropriate. However, this assumption will not always hold: the trainer may give the learner an action in order to emphasize that action *and* to reward the actions that lead up to it. Because our trainer provides actions at random times, we leave the history information intact.

In Lin's (1992) system, the trainer provides the learning agent with an entire sequence of actions that the learner then executes, receiving whatever feedback the trainer received while performing them originally.

3 EMPIRICAL STUDY

To determine whether our approach improves the performance of the learner, we compared a learner that does not receive instruction to a series of learners that receive varying amounts of instruction from the trainer. We present the details of the experiments and a discussion of the results in the following sections.

3.1 PROBLEM DOMAIN

The problem domain we use is a car navigation task based on the Race Track game (Gardner, 1973). Except for minor differences in the simulation, this is the same domain presented in Clouse & Utgoff (1992). The objective of the task is to drive the car from the start line to the finish line without driving off the track, which can be of any length and shape and is described by a left boundary, a right boundary, a start line with designated start points, and a finish line. Figure 2 depicts one of the tracks employed in the experiments.

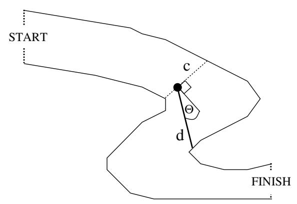

Figure 2: Example Track

At each time step, the car can accelerate by one speed unit, decelerate by one speed unit, turn left, or turn right. Alternatively, the car may maintain its current velocity by performing the NONE action. In the simulation, the amount the car turns depends on the current speed of the car: low speeds allow a large turn angle, whereas high speeds restrict the turn angle. The car is also constrained by upper and lower speed limits.

A set of sensors gives the learner information about the location of the car on the track: the distance straight ahead to the track's boundary, the relative placement of the car between the left and right track boundaries, the angle of the car's motion with respect to the track, and whether the finish line is directly ahead of the car. The first three of these sensors are represented in Figure 2 as d, c, and θ, respectively. If the car in the figure were to maintain its current heading, it would follow the segment labeled d.

The environmental critic gives one of three reinforcement signals at each step: 1.0 when the car crosses the finish line, -1.0 when the car crosses a track boundary, and -0.01 otherwise, to simulate fuel consumption.

3.2 REINFORCEMENT LEARNING METHOD

The learner employs a modified Q-learning reinforcement learning method (Watkins, 1989) to develop its policy. The two modifications allows the learner to incorporate the trainer's proffered actions. Each action that the learner can choose has a separate Q-function, each of which is a linear neuron represented as a weight vector. The input feature vector for the Q-functions represents the current state, $x(t)$, and is composed of scaled sensor values and the second-order terms for those values (each of the values multiplied by itself and the other values). Each action's Q-function value, $Q_a(x(t))$, is given by the inner product of the weight vector and the feature vector.

The policy defined by the Q-functions is based on the Boltzmann distribution of the values of the neurons: the next action, a, is randomly chosen based on the probabilities given by

$$\forall a \in \text{actions},\ p(a) = \frac{e^{Q_a(x(t))/T}}{\sum_{b \in \text{actions}} e^{Q_b(x(t))/T}}$$

where $x(t)$ is the current state, T is the temperature parameter that is reduced over time to diminish the randomness of the policy, and "actions" is the set of actions the learner can perform. When the trainer does not give an action to the learner, the learner chooses its action based on the probabilities defined by the equation above. When the trainer provides an action, the learner does not check its policy, but performs the trainer's action—this is the first modification.

After the learner performs the chosen action, changing the task state from $x(t)$ to $x(t+1)$, it receives reinforcement r. When the action was provided by the trainer, the learner receives reinforcement for performing the action, regardless of any reinforcement from the environmental critic—this is the second modification.

Before changing the Q-function weights, the learner updates the eligibility traces that allow previous actions to receive partial credit or blame for the current feedback. The eligibility trace of the action chosen is updated by

$$\bar{x}_{a,i}(t) = \lambda \bar{x}_{a,i}(t-1) + (1-\lambda)x_i(t)$$

where $\bar{x}_{a,i}(t)$ and $\bar{x}_{a,i}(t-1)$ are the new and old values of the trace of the ith feature for the chosen action, λ is a trace decay rate, $0 \leq \lambda < 1$, and $x_i(t)$ is the ith feature of state $x(t)$. The eligibility traces of the actions not chosen are decayed:

$$\bar{x}_{b,i}(t) = \lambda \bar{x}_{b,i}(t-1)$$

Finally, the learner modifies the neuron weights, $w_{b,i}$, of every Q-function based on the previous and current states, $x(t)$ and $x(t+1)$, the reinforcement received, r, the eligibility traces, $\bar{x}_b$, the reinforcement discount factor, γ, and the learning rate, α:

$$w_{b,i}(t+1) = w_{b,i}(t) + \alpha(r + \gamma V_{x(t+1)} - Q_a(x(t)))\bar{x}_{b,i}(t)$$

$$V_{x(t+1)} = \max_{b \in \text{actions}} [Q_b(x(t+1))]$$

The parameter settings in the experiments were: $\alpha = 0.35$ (learning rate), $\gamma = 0.75$ (reinforcement discount factor), $\lambda = 0.75$ (trace decay rate). The temperature decays slowly from $T = 5.0$ down to its lowest value, $T = 0.2$, via a linear factor of 0.9995.

3.3 TRAINING AGENT

The trainer that instructs the learning agent is automated and capable of performing the task at a moderate level of expertise. Although the trainer does not always perform the optimal actions, the trainer can drive the car from each start position of the track to the finish line.

Building the trainer involved a cyclic design, code, and test approach. First, a moderate amount of knowledge engineering, of examining different race track situations and ascertaining the action that a good driver should take, determined the initial code. Then the trainer was allowed to drive the car on the track. When the trainer crossed a track boundary, the code was redesigned and re-tested. This process continued until the trainer could drive on both the tracks used in the experiments, not only from every start location, but also from other, randomly selected points on the track.

The main objective of the trainer, as it drives, is to remain on the track. The secondary objective is to cross the finish line as quickly as possible. Thus, whenever the trainer is in a situation in which it can accelerate, it will do so. Otherwise, the trainer either turns to avoid crossing a track boundary and to stay close to the center of the track, or slows down. The trainer does not produce an action when the car is driving the wrong way on the track.

3.4 EXPERIMENTS

In the first experiment, the learning agent does not have access to the trainer: it relies solely on the environmental critic for training information. In the remaining experiments, the learner receives varying amounts of instruction from the trainer, controlled by an instruction rate parameter, which we specify as a percentage. The instruction rate indicates the probability that the trainer will provide the learner with an action on any given time step. We varied the instruction rate from 1%— the trainer only provides actions on 1% of the time steps—up to 100%— the trainer provides an action every time.

In order to determine whether the policy of giving the learner positive reward for performing the trainer's actions is effective, we also experimented with two levels for the reward. In one set of experiments, the learner receives a reinforcement signal of -0.01 for performing the trainer's action, in effect receiving no reward. In another set of experiments, the learner received a reinforcement signal of 1.0, thus rewarding the learner for performing the trainer's action.

Every individual experiment consisted of twenty *runs*, each of which began with the Q-function weights and the eligibility trace values initialized to zeroes. A run ended when the learner could satisfy the stopping criterion of navigating the car across the finish line from each of the designated starting positions on the track. For the tracks we employed, there are eighteen (18) such positions.

Each run consisted of several *trials*, which all began with the car on one of the designated start positions. A trial ended in failure when the car drove off the track, after which a new trial began. A trial ended in success when the car crossed the finish line, at which time the stopping criterion for a run was tested. If the stopping criterion was satisfied, the entire run was ended. Otherwise, the run continued with the beginning of a new trial.

For each run, we recorded the number of trials necessary for the learner to achieve the stopping criterion and the total number of actions performed by the learner throughout the run. After the run completed, we also measured the fitness of the learned policy by counting the number of actions executed while navigating from each start position to the finish line. Because the fitness measure represents the number of actions performed, it reflects the accumulated reinforcement.

3.5 RESULTS AND DISCUSSION

In the following sections we present the results of the experiments, discussing each of the three statistics measured: the number of trials necessary to achieve the stopping criterion, the total number of actions per-

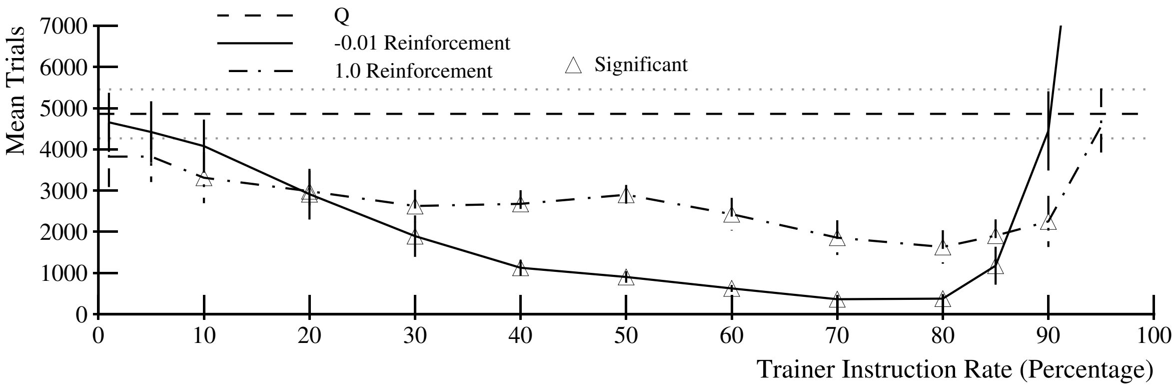

Figure 3: Mean of Trials

formed, and the number of actions performed while evaluating the fitness of the learned policies.

We performed the various experiments with two different tracks. The results described below are based on the track shown in Figure 2. The second track was extended to be about twice as long as this track. The basic results for the longer track are the same as reported below for the shorter track, although the values of all the statistics were higher because the track was longer.

3.6 Number of Trials

Figure 3 presents the means of the number of trials necessary to achieve the stopping criterion for each experiment[1]. In order to facilitate comparison with the result of the reinforcement-only experiment (0% instruction rate), the dashed line labeled "Q" was extended from the point on the vertical axis representing the result of that experiment (an average of 4,861.6 trials). The dotted lines above and below this line enclose the 95% confidence interval for that value. The curve labeled "-0.01 Reinforcement" represents the average number of trials for each level of trainer instruction rate, with 95% confidence bars, for the -0.01 level of reinforcement. The remaining curve, labeled "1.0 Reinforcement", represents the mean number of trials for the 1.0 reinforcement level. The values that are significantly different (at the 0.01 level) from the reinforcement-only value are marked with triangles.

Figure 3 indicates that the instruction provided by the training agent does reduce the amount of training necessary to meet the stopping criterion. Both levels of reinforcement produce a similar pattern across the different levels of instruction rate. For most levels of instruction rate tested, the difference between not training and training is significant. The difference is not significant at low instruction rates: little instruction does not reduce the number of the training trials.

[1]The numerical results are presented in Table 1 at the end of the paper.

At high rates of instruction, the number of trials is the same as, or worse than, that necessary without the trainer's help. At the 100% level, which is not depicted on the graph, all the training trials exceeded the pre-specified limit of 75,000 actions per trial. At the 95% instruction level for the -0.01 reinforcement level, the average number of trials was over 15,000. These very high averages occur possibly because, as the learner relies increasingly on the trainer, it performs less exploration, which is an important aspect of reinforcement learning (Thrun & Möller, 1992). Also, at these high levels of instruction, almost all of the actions are coming from the trainer, which doesn't fail at the task. The learner may be performing poorly because it is not allowed to learn from failure.

A major difference between the two levels of reinforcement is that the learners that receive no reinforcement for performing the trainer's action (the -0.01 level) achieve the stopping criterion more quickly than the learners that are rewarded for performing the trainer's actions. It is unclear why this phenomenon occurs. It may be related to the fact that the learner that receives the higher level of reinforcement makes greater changes to its policy after executing a trainer's action, because it has received a higher reward. Because the policy is represented as a set of linear neurons, large changes affect large portions of the state space, and thus may degrade good performance in other parts of the space.

3.7 Number of Actions

Figure 4 is similar to Figure 3, but the curves do not have the exact same shape. Over most of the domain, the curves in Figure 4 do not descend as quickly as those in the previous figure. This indicates that as the instruction rate increases, the length of the trials (the ratio of the number of actions to the number of trials) also increases. That is, as the trainer helps more frequently, the trials get longer, which implies the learner fails less frequently.

The greatest difference between the two graphs occurs

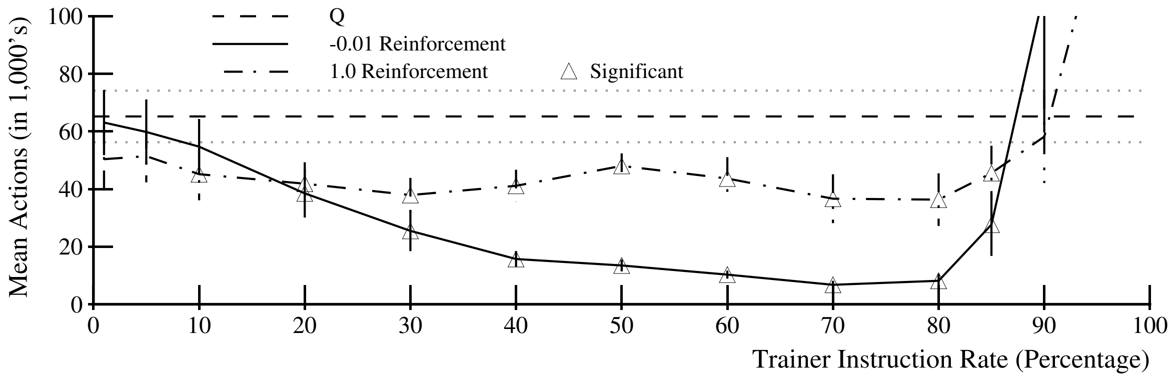

Figure 4: Mean of Total Actions

at the highest levels of instruction. The slope of the curves in Figure 4 is much steeper than that for the curves that represent the number of trials. Essentially, the trainer is performing the task at these high levels, and the learner's actions can be considered noise. So, each individual trial is long, and because there are many of them, the total number of actions is high.

3.8 Fitness Measure

Figure 5 depicts the mean of the fitness measure for the two levels of reinforcement and the various levels of instruction rate. The mean fitness of the policies learned at the -0.01 level of reinforcement are either worse than or no better than the mean fitness of the reinforcement-only policies.

This represents a positive result for moderately low levels of instruction (20% and 30%) because the learner develops a policy that is indistinguishable from the reinforcement-only learner's, but with fewer training trials (See Figure 3). However, for other levels of instruction, the learned performance is much worse than the performance without instruction. This may be due to the fact the the learner can so quickly achieve the stopping criterion: the learner doesn't have the time to refine its policy via exploration.

At the 1.0 level of reinforcement, the learner not only learns quickly, as shown in Figure 3, but also produces policies that are better than those produced without the trainer's help. Because the trainer's actions are rewarded, the learner attempts them in the future. As was described in Section 3.3, the training agent will accelerate whenever it has the opportunity. The learning agent gets rewarded for accelerating, which causes the learner to execute the SPEED action in the future. Thus, the learner develops a policy that allows it to cross the finish line quickly, improving the fitness of the learned policy.

4 FURTHER RESEARCH AND CONCLUSIONS

In order to take advantage of a training agent's knowledge of a task, one needs to answer questions about how the trainer will interact with the learning agent and how that learning agent will incorporate the trainer's instruction. We developed a particular approach to adding a training agent to a reinforcement learning scenario by examining specific answers to those questions.

We intend to perform further research with this approach. In the experiments discussed above, we only examined two levels of reinforcement received for performing a trainer's action. Because the results for the two levels were so different, we plan to study more levels of the reinforcement. We also employed a hand-coded training agent that could perform the task. We wish to examine different levels of trainer expertise and their effects on the learner's ability to develop a successful policy. Furthermore, although the race track domain is challenging, it is only one of many that we wish to explore.

In conclusion, the results suggest that providing the learner with too much instruction degrades the learner's performance, possibly because the learner does not have the opportunity to perform any exploration and is not allowed to learn from failure. We also found that the low level of reinforcement lead to quickly satisfying the stopping criterion, but the policies produced were poor. With the high level of reinforcement, the learner required more time (but still less than that needed without instruction), but produced superior policies. Our approach to providing instruction allows the learning agent to develop its policy more quickly, and to perform better on its task.

Appendix

Table 1 shows the values on which the graphs and discussion in Section 3.5 are based.

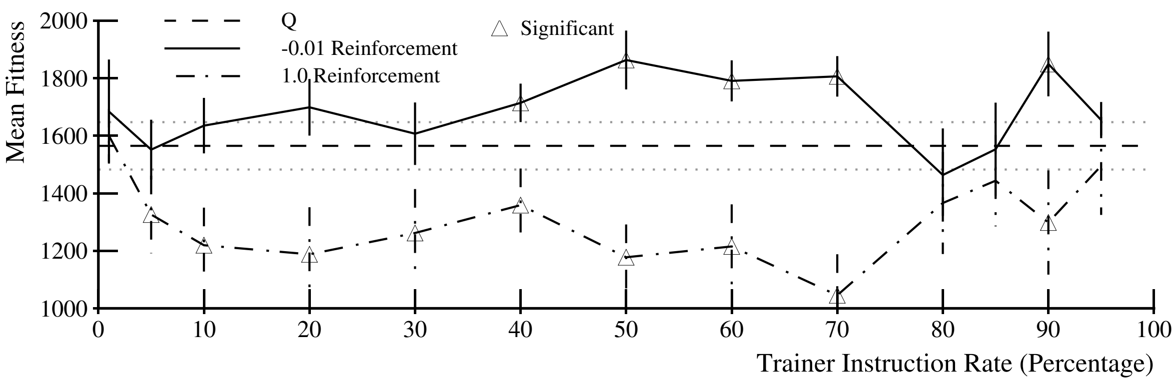

Figure 5: Mean of Fitness

Acknowledgements

I would like to thank Paul Utgoff, David Skalak, Gunnar Blix, and the anonymous reviewers who helped me improve the presentation of this material.

References

Barto, A. G., Bradtke, S. J., & Singh, S. P. (1993). *Learning to act using real-time dynamic programming*, (COINS Technical Report 93-02), Amherst, MA: University of Massachusetts, Department of Computer and Information Science.

Clouse, J. A., & Utgoff, P. E. (1992). A teaching method for reinforcement learning. *Machine Learning: Proceedings of the Ninth International Conference* (pp. 92-101). San Mateo, CA: Morgan Kaufmann.

Gardner, M. (1973). Mathematical games. *Scientific American, 228*, 108.

Gordon, D., & Subramanian, D. (1994). A multistrategy learning scheme for agent knowledge acquisition. *Informatica , 17*, 331-346.

Gullapalli, Vijaykumar (1992). *Reinforcement learning and its application to control.* Doctoral dissertation, Computer and Information Science, University of Massachusetts, Amherst, MA.

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

Lin, Long-Ji (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning, 8*, 293-321.

Maclin, R., & Shavlik, J. W. (1994). Incorporating advice into agents that learn from reinforcements. *Proceedings of the Twelfth National Conference on Artificial Intelligence* (pp. 694-699). Seattle, WA: MIT Press.

Sutton, R. S. (1984). *Temporal credit assignment in reinforcement learning.* Doctoral dissertation, Department of Computer and Information Science, University of Massachusetts, Amherst, MA.

Thrun, Sebastian B., & Möller, Knut (1992). Active exploration in dynamic environments. In Moody, Hanson & Lippman (Eds.), *Advances in Neural Information Processing Systems.* San Mateo, CA: Morgan Kaufman.

Towell, G., Shavlik, J., & Noordewier, M. (1990). Refinement of approximate domain theories by knowledge-based neural networks. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 861-866). Boston, MA: Morgan Kaufmann.

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

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

Whitehead, S. D. (1992). *Reinforcement learning for the adaptive control of perception and action.* Doctoral dissertation, Computer Science, University of Rochester, Rochester, NY.

Table 1: The information provided in this table represents the results of the experiments described in the text. The first column shows the instruction rate, as a percentage. The next three columns depict the results for the -0.01 level of reinforcement. The final three columns depict the results for the 1.0 level of reinforcement. The columns labeled "Trials" show the mean and 95% confidence interval for the number of trials necessary to achieve the stopping criterion. "Actions" show the mean and confidence interval for the number of actions performed. Finally, "Fitness" shows the values for the fitness of the learned policies.

instruction rate	-0.01 Reinforcement			1.0 Reinforcement		
	Trials[1]	Actions[1,2]	Fitness[1]	Trials[1]	Actions[1,2]	Fitness[1]
0%[3]	4861.6±594.5	65.2±9.0	1565.0±82.4	4861.6±594.5	65.2±9.0	1565.0±82.4
1	4656.2±717.6	63.1±11.3	1684.2±180.9	3824.2±817.0	50.4±11.8	1600.4±93.1
5	4421.2±746.1	59.8±11.3	1551.8±104.0	3825.7±619.3	51.4±9.1	1326.2±135.0†
10	4078.9±645.0	54.7±9.6	1635.3±96.5	3311.5±617.3†	45.2±9.1†	1219.5±130.7†
20	2912.2±614.7†	38.5±8.4†	1699.0±98.4	2984.5±499.5†	41.9±7.4†	1187.9±163.6†
30	1894.4±503.9†	25.5±7.0†	1607.2±108.7	2626.0±392.0†	37.9±6.0†	1262.0±152.8†
40	1126.3±195.5†	15.7±2.7†	1714.5±66.8†	2677.7±331.0†	41.1±5.6†	1358.9±127.3†
50	902.8±140.4†	13.5±2.1†	1863.4±102.3†	2898.7±237.7†	48.0±4.3†	1177.7±114.0†
60	625.0±84.2†	10.3±1.4†	1790.9±71.4†	2426.4±396.3†	43.7±7.4†	1215.2±146.3†
70	362.8±75.9†	6.8±1.4†	1806.6±70.3†	1856.4±423.1†	36.7±8.5†	1045.5±142.5†
80	377.0±112.0†	8.2±2.4†	1463.6±162.0	1631.3±406.6†	36.3±9.1†	1366.2±177.4
85	1176.6±460.2†	27.7±10.9†	1553.1±162.1	1905.7±394.8†	45.6±9.5†	1443.8±158.8
90	4448.1±961.1	113.7±24.7†	1809.3±83.2†	2251.3±620.6†	58.2±16.1	1298.4±181.1†
95[4]	15255.6±17046.1	428.5±486.0	1654.6±62.9	4561.3±918.2	127.7±25.7	1495.3±170.5

1 mean ± 95% confidence interval.

2 in 1,000's.

3 learning solely from the critic.

4 values for the −0.01 reinforcement rate are based on only 7 runs

† statistical significance at the 0.01 level.