In *Proceedings of the Twenty-Second Conference on Artificial Intelligence* (AAAI-07), Vancouver, Canada, July 2007.

Temporal Difference and Policy Search Methods for Reinforcement Learning: An Empirical Comparison

Matthew E. Taylor, Shimon Whiteson, and Peter Stone
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188
{mtaylor, shimon, pstone}@cs.utexas.edu

Abstract

Reinforcement learning (RL) methods have become popular in recent years because of their ability to solve complex tasks with minimal feedback. Both genetic algorithms (GAs) and temporal difference (TD) methods have proven effective at solving difficult RL problems, but few rigorous comparisons have been conducted. Thus, no general guidelines describing the methods' relative strengths and weaknesses are available. This paper summarizes a detailed empirical comparison between a GA and a TD method in Keepaway, a standard RL benchmark domain based on robot soccer. The results from this study help isolate the factors critical to the performance of each learning method and yield insights into their general strengths and weaknesses.

Introduction

Reinforcement learning (RL) problems are characterized by agents making sequential decisions with the goal of maximizing total reward, which may be time delayed. RL problems contrast with classical planning problems in that agents do not know *a priori* how their actions will affect the world and with supervised learning because the agent is never given training examples with the correct action labeled.

Temporal difference (TD) methods are one popular way to solve RL problems. TD methods learn a *value function* that estimates the expected long-term reward for taking a particular action in a state. *Genetic algorithms* (GAs) can also address RL problems by searching the space of policies for one that receives maximal reward. Although these two approaches have both had success in difficult RL tasks, only a few studies (Gomez & Miikkulainen 2002; Moriarty & Miikkulainen 1996; Pollack, Blair, & Land 1997) have directly compared them. As a result, there are currently no general guidelines describing the methods' relative strengths and weaknesses.

One possible reason for the lack of such comparisons is that the two communities are largely disjoint. While TD practitioners typically publish papers in AAAI or ICML, the GA community more often sends papers to GECCO or CEC. As an example of this separation, one of the canonical books on RL (Sutton & Barto 1998) explicitly does not discuss evolutionary methods because they "do not consider evolutionary methods by themselves to be especially well suited to reinforcement learning problems." We view such a divide as unproductive: both classes of algorithms can be used to effectively solve the same type of problems and therefore rigorous comparisons between them are important.

This paper takes a step towards bridging the divide by presenting results from a previous study (Taylor, Whiteson, & Stone 2006) that compares two representative TD and GA methods. This study uses *3 vs. 2 Keepaway* (Stone *et al.* 2006), a standard RL benchmark domain based on robot soccer in which agents have noise in both their sensors and actuators. In addition to the benchmark task, we also experiment in fully observable and deterministic variants of Keepaway that are designed to isolate factors critical to the performance of each method.

The results demonstrate that the relative performance of the two methods depends on the given task's particular characteristics. Based on these results, we suggest some useful guidelines for objectively choosing between the two approaches. We intend for these guidelines to be useful when a novel RL task is encountered and either a TD or GA method could be used. Furthermore, we hope that this study will encourage researchers to conduct more empirical comparisons across the two communities and to explore a variety of algorithms and tasks, thereby testing how broadly the conclusions we have drawn can generalize.

The Benchmark Keepaway Task

RoboCup simulated soccer is a multiagent domain that has noisy sensors and actuators as well as hidden state. In 3 vs. 2 Keepaway, a subproblem of the full 11 vs. 11 simulated soccer game, a team of 3 *keepers* attempts to maintain possession of the ball on a 20m x 20m field while 2 *takers* attempt to gain possession of the ball or force the ball out of bounds, ending an *episode*. All our experiments are run on a code base derived from version 0.6 of the benchmark Keepaway implementation[1] and we set parameters to be consistent with past research (Stone, Sutton, & Kuhlmann 2005; Stone *et al.* 2006).

Keepaway is an appealing platform for empirical comparisons because previous studies (Kostiadis & Hu 2001; Stone, Sutton, & Kuhlmann 2005) have already established the performance of TD methods. While GAs have been applied to variations of Keepaway (Hsu & Gustafson 2002; Whiteson *et al.* 2005), they have never been applied to the benchmark version of the task.

[1] Available at http://www.cs.utexas.edu/~AustinVilla/sim/Keepaway/

Each keeper learns to maximize the average episode length by learning to maximize the team's reward in this task. Each keeper, when in possession of the ball, must use its policy to map the current 13-dimensional state of the world (see Figure 1) to one of three higher-level macro-actions: $A = \{$hold, passToTeammate1, passToTeammate2$\}$. Any keeper not in possession of the ball follows a static policy and either attempts to go to the ball or try to get open for a pass. Takers follow a static policy and do not learn. The initial state is different in each episode because the same keeper does not always start with the ball, the three keepers are placed near three corners, and the takers are placed near the fourth corner (rather than in exact locations).

Learning Keepaway

This section provides an overview of the GA and TD methods used in our experiments. There are a wide variety of both GAs and TD methods in use today but to empirically compare these different approaches we must focus on specific instantiations. *NeuroEvolution of Augmenting Topologies* (NEAT) (Stanley & Miikkulainen 2002), a GA that performs a policy search by evolving neural networks, has had substantial success in RL domains like double pole balancing (Stanley & Miikkulainen 2002) and robot control (Stanley & Miikkulainen 2004a). *Sarsa* (Rummery & Niranjan 1994; Singh & Sutton 1996) is a popular TD method that has also had multiple empirical successes (Sutton 1996; Sutton & Barto 1998).

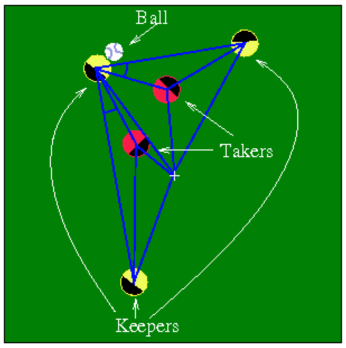

Figure 1: 13 state variables are used for learning with 3 keepers and 2 takers. The state is ego-centric and rotationally invariant for the keeper with the ball; there are 11 distances, indicated with blue lines, between players and the center of the field as well as 2 angles along passing lanes.

We use Sarsa and NEAT as representative methods because of their empirical success in the benchmark Keepaway task (Stone, Sutton, & Kuhlmann 2005) or variations thereof (Whiteson *et al.* 2005), and because they are the methods that the authors are most familiar with. In addition to the practical advantages, this familiarity enables us to set both algorithms' parameters with confidence. We attempted to give equal effort to optimizing the parameters of each algorithm, though such factors are admittedly difficult to control for. Wherever possible, the original study (Taylor, Whiteson, & Stone 2006) quantifies the amount of tuning involved.

NEAT

NEAT represents policies with neural networks and searches "policy space" to find an appropriate topology and set of weights for the networks. NEAT begins with a uniform population of 100 simple networks with no hidden nodes and inputs connected directly to outputs. Over time new structure is introduced to networks in the population incrementally via two special evolutionary mutation operators that add new hidden nodes and links to the network. Only the structural mutations that yield performance advantages tend to survive evolution's selective pressure. NEAT also learns appropriate weights for the networks through crossover and mutation.

NEAT is a general purpose optimization technique and can be applied to a wide variety of problems. We use NEAT to perform policy search RL and represent policies via a population of neural network *action selectors*. Inputs to each network describe the agent's current state. There is one output for each available action and the agent takes whichever action has the highest activation.

Each candidate network is evaluated by allowing it to control the keepers' behavior and observing how much reward it receives. The policy's fitness is the sum of the rewards accrued while under the network's control. In deterministic domains, each policy in the population can be evaluated in a single episode. However, in stochastic domains it is necessary to evaluate each member of the population for many episodes to get accurate fitness estimates, and we used preliminary experiments to determine that NEAT learns well when each policy in Keepaway is evaluated for an average of 60 episodes.

Learning with Sarsa

Sarsa learns to estimate the action-value function, $Q(s, a)$, which predicts the long-term expected return of taking an action, a, in some state, s. Learning a value function allows an agent to estimate the efficacy of each action in a given state, in contrast to GAs which evaluate entire policies holistically without regard to individual actions' values.

Sarsa is an acronym for State Action Reward State Action, describing the 5-tuple needed to perform the update: (s, a, r, s', a'), where s and a are the the agent's current state and action, r is the immediate reward the agent receives from the environment, and s' and a' are the agent's subsequent state and chosen action. In the simple case, the action-value function is represented in a table, with one entry per state-action pair. After each action, the table is updated via: $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha(r + \gamma Q(s', a'))$, where α is the learning rate and γ is a discount factor used to weight immediate rewards more heavily than future rewards.

In continuous domains like Keepaway, the value function cannot be represented in a table. In such cases, TD methods rely on *function approximators*, which map state-action pairs to values via more concise, parameterized functions and use supervised learning methods to set these parameters. Many function approximators have been used, including neural networks, CMACs, and radial basis functions (Sutton & Barto 1998). In this study we use a radial basis function approximator (RBF), a method with previous empirical successes (Stone *et al.* 2006). Sarsa is able to modify the weights in the RBF function approximator over time, using standard ϵ-greedy action selection, so that the keepers are able to learn an approximation for the true action-value function and thus increase the average episode length.

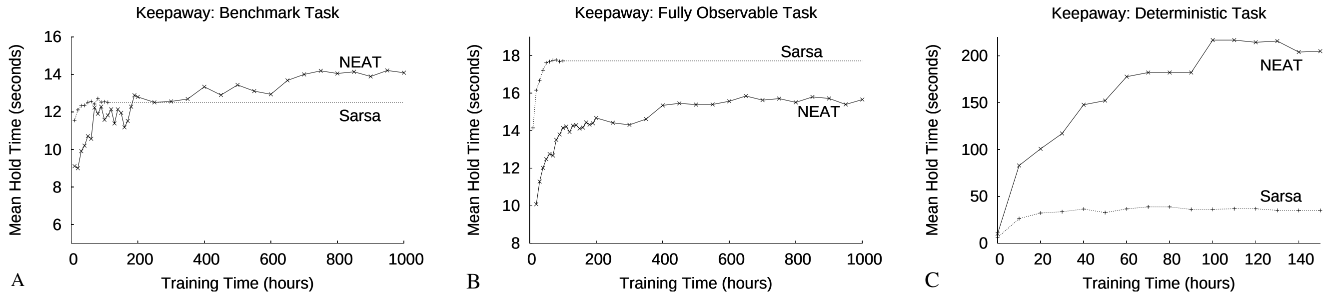

Figure 2: A comparison of the mean hold times of the policies discovered by NEAT and Sarsa in: (A) the benchmark task, (B) the fully observable version of the task, and (C) the deterministic version of the Keepaway task.

Results

In order to compare the two learning methods, we first ran a series of independent trials using each learning method until each trial plateaued. Afterwards, we evaluated the best learned policy in each trial after various amounts of time spent learning. All times reported in this paper refer to simulator time. Thus we report only the time the agents spend interacting with the environment and ignore the time they are performing calculations; the running time for our learning methods is negligible compared to that of the Soccer Server.

Benchmark Keepaway Results

After running trials in the benchmark version of Keepaway, we computed the mean hold time of the best policy found so far by each method at regular intervals. The mean hold times were then averaged across all trials of each of the methods to obtain the plots shown in Figure 2A. Sarsa learners plateau before the NEAT learners, but the performance of the Sarsa learners is extended on the graph even after learning has finished, denoted by a horizontal performance line without plotted data points. For presentation purposes we plot the average performance every 10 hours for the first 200 hours and then every 50 hours after that (increasing the sampling resolution does not reveal any interesting detail).

The results demonstrate that NEAT can learn better policies in the benchmark Keepaway task than Sarsa with RBFs, the best performing TD method to date. Student's t-tests confirm the difference in performance between NEAT and Sarsa is statistically significant for times greater than or equal to 650 hours. While NEAT ultimately learns better policies, it requires many more evaluations to do so. Sarsa learns much more rapidly: in the early part of learning its average policy is much better than NEAT's.

Fully Observable Keepaway Results

In the benchmark Keepaway task, the agents' sensors are noisy. Since the agents can only partially observe the true state of the world, the task is non-Markovian: the probability distribution over next states is not independent of the agents' state and action histories. Sarsa's TD update assumes that the environment is Markovian, though in practice TD methods can still perform well when it is not (Sutton & Barto 1998). By contrast, NEAT can evolve recurrent networks that cope with non-Markovian tasks (Gomez & Schmidhuber 2005).

We hypothesized that Sarsa's relative performance would improve if sensor noise was removed, rendering the Keepaway task fully observable and effectively Markovian. To test this hypothesis, we conducted a second experiment in a version of the Keepaway task which had all noise from the simulated agents' sensors removed. Figure 2B shows the results of these experiments, with mean hold times computed as before and averaged across all trials of a single method. Sarsa again learns much more rapidly than NEAT, but now also learns substantially better policies. The difference in performance between NEAT and Sarsa is statistically significant for all points graphed.

Deterministic Keepaway Results

Even in the fully observable version of the task, which has no sensor noise, the task remains highly stochastic due to noise in the agents' actuators and randomness in the start state. In both the benchmark task and the fully observable variation, this stochasticity greatly slows NEAT's learning rate: NEAT is able to learn well only when the fitness estimate for each candidate policy is averaged over many episodes. Therefore, we tested a second variation of Keepaway in which noise was removed from the sensors and actuators, and the agent's initial states were fixed which created a completely deterministic task. We hypothesized that NEAT would learn much faster in this deterministic variation as it could perfectly evaluate each organism in a single episode.

These experiments used the same parameters as before, except NEAT required only 100 episodes per generation (1 episode per policy in the population). Figure 2C shows the experimental results with mean hold times computed as before and averaged across all trials of a single method. In the deterministic version of the task, NEAT learns more rapidly than Sarsa, in addition to discovering dramatically superior policies, and the difference in performance is statistically significant for all points graphed.

Discussion

Investigating the tradeoffs between different RL methods is important for three reasons. First, cases where one class of method outperforms the other may suggest ways to enhance particular methods, e.g. by finding ways to play to each methods' strengths. For instance, hybrid methods (Whiteson & Stone 2006) may be able to combine benefits from each method.

Second, RL is beginning to play an increasingly important role in complex systems like those used to solve difficult *challenge problems*. Consequently, developing accurate recommendations about which methods to use when is becoming more and more crucial. The experiments presented in this paper provide a preliminary basis for such recommendations. In particular, the results suggest that the choice between using a GA and a TD method should be made based on some of the given task's characteristics. In deterministic domains, the fitness of an organism can be quickly evaluated and GAs are likely to excel. If the task is fully observable but nondeterministic, TD methods may have an advantage. If the task is partially observable and nondeterministic, each method may have different advantages: TD methods in speed and GAs in asymptotic performance.

Third, such experiments test claims about the strengths and weaknesses of the different approaches. For example, the results in the benchmark Keepaway task show that GAs can excel at RL tasks, challenging Sutton and Barto's claim that such methods are not well suited to them. Additionally, results in the benchmark and fully observable tasks confirm the common conception that GAs are slow even when they learn well, through results in the deterministic task highlight how quickly GAs can learn when fitness evaluations are accurate. Furthermore, results in the fully observable task show that the performance of TD methods is sensitive to the presence of the Markov property, buttressing claims by the GA community that their methods are better equipped to tackle non-Markovian tasks.

No empirical comparison between two parameterized learning methods can be completely conclusive. Most learning methods have some number of parameters to set, and the amount of time devoted to "tweaking" them can have a dramatic impact on the success of learning. Furthermore, no single study is able to compare all algorithmic variants or consider every relevant domain. Though both the algorithms and the domain used in this paper were carefully chosen to be representative, further data is clearly needed to make conclusive statements about the relative strengths and weaknesses of GAs and TD methods. We hope that in the future there will be many more studies that address these types of issues.

Conclusion

This paper presents the results of a detailed empirical comparison between NEAT and Sarsa in Keepaway, a standard RL benchmark domain based on robot soccer. The results demonstrate that NEAT can learn better policies in this domain than Sarsa with RBF function approximation, the previous best known method, though it requires more evaluations to do so. Additional experiments in two variations of Keepaway demonstrate that Sarsa learns better policies when the domain is fully observable and NEAT learns faster when the domain has a deterministic fitness function. Together, these results help isolate factors critical to the performance of each method and yield insights into how they may perform on additional tasks. Additional studies using more domains and different algorithms are necessary to draw definitive guidelines about when to use a GA or TD method. This study provides a model of how comparisons may be done fairly and provides an initial data point for establishing such guidelines. Furthermore, we hope that this study helps demonstrate the need for further collaboration between practitioners of the different approaches to RL so that such methods can be more accurately evaluated, compared, and made useful in practice.

Acknowledgments

We would like to thank Ken Stanley for help setting up NEAT in Keepaway, as well as Nate Kohl, David Pardoe, Joseph Reisinger, Jefferson Provost, and the anonymous reviewers for helpful comments and suggestions. This research was supported in part by DARPA grant HR0011-04-1-0035, NSF CAREER award IIS-0237699, and NSF award EIA-0303609.

References

Gomez, F., and Miikkulainen, R. 2002. Robust nonlinear control through neuroevolution. Technical Report AI02-292.

Gomez, F., and Schmidhuber, J. 2005. Co-evolving recurrent neurons learn deep memory POMDPs. In *GECCO-05: Proceedings of the Genetic and Evolutionary Computation Conference*, 491–498.

Hsu, W. H., and Gustafson, S. M. 2002. Genetic programming and multi-agent layered learning by reinforcements. In *Genetic and Evolutionary Computation Conference*, 764–771. New York,NY: Morgan Kaufmann.

Kostiadis, K., and Hu, H. 2001. KaBaGe-RL: Kanerva-based generalization and reinforcement learning for possession football. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*.

Moriarty, D. E., and Miikkulainen, R. 1996. Efficient reinforcement learning through symbiotic evolution. *Machine Learning* 22:11–32.

Pollack, J. B.; Blair, A. D.; and Land, M. 1997. Coevolution of a backgammon player. In Langton, C. G., and Shimohara, K., eds., *Artificial Life V: Proc. of the Fifth Int. Workshop on the Synthesis and Simulation of Living Systems*, 92–98. Cambridge, MA: The MIT Press.

Rummery, G. A., and Niranjan, M. 1994. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG-RT 116, Engineering Department, Cambridge University.

Singh, S. P., and Sutton, R. S. 1996. Reinforcement learning with replacing eligibility traces. *Machine Learning* 22:123–158.

Stanley, K. O., and Miikkulainen, R. 2002. Evolving neural networks through augmenting topologies. *Evolutionary Computation* 10(2):99–127.

Stanley, K. O., and Miikkulainen, R. 2004a. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research* 21:63–100.

Stone, P.; Kuhlmann, G.; Taylor, M. E.; and Liu, Y. 2006. Keepaway soccer: From machine learning testbed to benchmark. In Noda, I.; Jacoff, A.; Bredenfeld, A.; and Takahashi, Y., eds., *RoboCup-2005: Robot Soccer World Cup IX*, volume 4020. Berlin: Springer Verlag. 93–105.

Stone, P.; Sutton, R. S.; and Kuhlmann, G. 2005. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior* 13(3):165–188.

Sutton, R. S., and Barto, A. G. 1998. *Introduction to Reinforcement Learning*. MIT Press.

Sutton, R. S. 1996. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, 1038–1044.

Taylor, M. E.; Whiteson, S.; and Stone, P. 2006. Comparing evolutionary and temporal difference methods in a reinforcement learning domain. In *Proc. of the Genetic and Evolutionary Computation Conference*, 1321–28.

Whiteson, S., and Stone, P. 2006. Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*.

Whiteson, S.; Kohl, N.; Miikkulainen, R.; and Stone, P. 2005. Evolving keepaway soccer players through task decomposition. *Machine Learning* 59(1):5–30.