Machine Learning, 22, 283–290 (1996)

Technical Note

Incremental Multi-Step Q-Learning

JING PENG jp@vislab.ucr.edu
College of Engineering, University of California, Riverside, CA 92521

RONALD J. WILLIAMS rjw@ccs.neu.edu
College of Computer Science, Northeastern University, Boston, MA 02115

Editor: Leslie Pack Kaelbling

Abstract. This paper presents a novel incremental algorithm that combines Q-learning, a well-known dynamic-programming based reinforcement learning method, with the TD(λ) return estimation process, which is typically used in actor-critic learning, another well-known dynamic-programming based reinforcement learning method. The parameter λ is used to distribute credit throughout sequences of actions, leading to faster learning and also helping to alleviate the non-Markovian effect of coarse state-space quantization. The resulting algorithm, *Q(λ)-learning*, thus combines some of the best features of the Q-learning and actor-critic learning paradigms. The behavior of this algorithm has been demonstrated through computer simulations.

Keywords: reinforcement learning, temporal difference learning

1. Introduction

The incremental multi-step Q-learning (Q(λ)-learning) method is a new direct (or model-free) algorithm that extends the one-step Q-learning algorithm (Watkins, 1989) by combining it with TD(λ) returns for general λ (Sutton, 1988) in a natural way for delayed reinforcement learning. By allowing corrections to be made incrementally to the predictions of observations occurring in the past, the Q(λ)-learning method propagates information rapidly to where it is important. The Q(λ)-learning algorithm works significantly better than the one-step Q-learning algorithm on a number of tasks and its basis in the integration of one-step Q-learning and TD(λ) returns makes it possible to take advantage of some of the best features of the Q-learning and actor-critic learning paradigms and to be a potential bridge between them. It can also serve as a basis for developing various multiple time-scale learning mechanisms that are essential for applications of reinforcement learning to real world problems.

2. TD(λ) Returns

Direct dynamic-programming based reinforcement learning algorithms are based on updating state values or state-action values according to state transitions as they are experienced. Each such update is in turn based on the use of a particular choice of estimator for the value being updated, which spells out differences among various learning methods.

This section describes an important and computationally useful class of such estimators – the TD(λ) estimators (Sutton, 1988; Watkins, 1989).

Let the world state at time step t be x_t, and assume that the learning system then chooses action a_t. The immediate result is that a reward r_t is received by the learner and the world undergoes a transition to the next state, x_{t+1}. The objective of the learner is to choose actions maximizing discounted cumulative rewards over time. More precisely, let γ be a specified discount factor in $[0, 1)$. The *total discounted return* (or simply *return*) received by the learner starting at time t is given by

$$\mathbf{r}_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^n r_{t+n} + \cdots.$$

The objective is to find a policy π, or rule for selecting actions, so that the expected value of the return is maximized. It is sufficient to restrict attention to policies that select actions based only on the current state (called *stationary* policies). For any such policy π and for any state x we define

$$V^\pi(x) = E[\mathbf{r}_0 | x_0 = x, a_i = \pi(x_i) \text{ for all } i \geq 0],$$

the expected total discounted return received when starting in state x and following policy π thereafter. If π is an optimal policy we also use the notation V^* for V^π. Many dynamic-programming-based reinforcement learning methods involve trying to estimate the state values $V^*(x)$ or $V^\pi(x)$ for a fixed policy π.

An important class of methods for estimating V^π for a given policy π is the TD(λ) estimators, which have been investigated by Sutton (1988) and later by Watkins (1989). Following Watkins' notation, let $\mathbf{r}_t^{(n)}$ denote the *corrected n-step truncated return* for time t, given by

$$\mathbf{r}_t^{(n)} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n \hat{V}_{t+n}^\pi(x_{t+n}) \tag{1}$$

where $\hat{V}_t^\pi$ is the estimate of V^π at time t. If $\hat{V}^\pi$ were equal to V^π, then the corrected truncated returns would be unbiased estimators of V^π. Watkins (Watkins, 1989) shows that corrected truncated returns have the *error-reduction property* in that the expected value of the corrected truncated return is closer to V^π than $\hat{V}^\pi$ is.

Then Sutton's TD(λ) return from time t is

$$\begin{aligned}
\mathbf{r}_t^\lambda &= (1-\lambda)[\mathbf{r}_t^{(1)} + \lambda \mathbf{r}_t^{(2)} + \lambda^2 \mathbf{r}_t^{(3)} + \cdots] \\
&= r_t + \gamma(1-\lambda)\hat{V}_t^\pi(x_{t+1}) + \gamma\lambda[r_{t+1} + \gamma(1-\lambda)\hat{V}_{t+1}^\pi(x_{t+2}) + \cdots] \\
&= r_t + \gamma(1-\lambda)\hat{V}_t^\pi(x_{t+1}) + \gamma\lambda \mathbf{r}_{t+1}^\lambda.
\end{aligned} \tag{2}$$

The TD(0) return is just $\mathbf{r}_t^0 = r_t + \gamma \hat{V}_t^\pi(x_{t+1})$ and the TD(1) return is

$$\mathbf{r}_t^1 = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

which is the exact actual return. Watkins (1989) argues that, in a Markov decision problem, the choice of λ is a trade-off between bias and variance. Sutton's empirical demonstration (Sutton, 1988) favors intermediate values of λ that are closer to 0. More

recent analysis (Sutton & Singh, 1994) suggests that in certain prediction tasks near optimal performance can be achieved by setting λ at each time step to the transition probability of the immediately preceding transitions. For further details, see (Sutton, 1988; Sutton & Singh, 1994; Watkins, 1989).

3. One-Step Q-Learning

One-step Q-learning of Watkins (1989), or simply Q-learning, is a simple incremental algorithm developed from the theory of dynamic programming (Ross, 1983) for delayed reinforcement learning. In Q-learning, policies and the value function are represented by a two-dimensional lookup table indexed by state-action pairs. Formally, using notation consistent with that of the previous section, for each state x and action a let

$$Q^*(x,a) = R(x,a) + \gamma \sum_y P_{xy}(a) V^*(y) \tag{3}$$

where $R(x,a) = E\{r_0 | x_0 = x, a_0 = a\}$, and $P_{xy}(a)$ is the probability of reaching state y as a result of taking action a in state x. It follows that $V^*(x) = \max_a Q^*(x,a)$.

Intuitively, Equation (3) says that the state-action value, $Q^*(x,a)$, is the expected total discounted return resulting from taking action a in state x and continuing with the optimal policy thereafter. More generally, the Q function can be defined with respect to an arbitrary policy π as $Q^\pi(x,a) = R(x,a) + \gamma \sum_y P_{xy}(a) V^\pi(y)$ and Q^* is just Q^π for an optimal policy π.

The Q-learning algorithm works by maintaining an estimate of the Q^* function, which we denote by $\hat{Q}^*$, and adjusting $\hat{Q}^*$ values (often just called *Q-values*) based on actions taken and reward received. This is done using Sutton's prediction difference, or TD error (Sutton, 1988)–the difference between the immediate reward received plus the discounted value of the next state and the Q-value of the current state-action pair:

$$r + \gamma \hat{V}^*(y) - \hat{Q}^*(x,a)$$

where r is the immediate reward, y is the next state resulting from taking action a in state x, and $\hat{V}^*(x) = \max_a \hat{Q}^*(x,a)$. Then the values of $\hat{Q}^*$ are adjusted according to

$$\hat{Q}^*(x,a) = (1-\alpha)\hat{Q}^*(x,a) + \alpha(r + \gamma \hat{V}^*(y)) \tag{4}$$

where $\alpha \in (0,1]$ is a learning rate parameter. In terms of the notation described in the previous section, Equation (4) may be rewritten as

$$\hat{Q}^*(x,a) = (1-\alpha)\hat{Q}^*(x,a) + \alpha \mathbf{r}^0 \tag{5}$$

That is, the Q-learning method uses TD(0) as its estimator of expected returns. Note that the current estimate of the Q^* function implicitly defines a greedy policy by $\pi(x) = \arg\max_a \hat{Q}^*(x,a)$. That is, the greedy policy is to select actions with the largest estimated Q-value.

It is important to note that the one-step Q-learning method does not specify what actions the agent should take at each state as it updates its estimates. In fact, the agent may take whatever actions it pleases. This means that Q-learning allows arbitrary experimentation while at the same time preserving the current best estimate of states' values. Furthermore, since this function is updated according to the ostensibly optimal choice of action at the following state, it does not matter what action is actually followed at that state. For this reason, Q-learning is not *experimentation-sensitive*. On the other hand, because actor-critic learning updates the state value at any state based on the actual action selected, not on what would have been the optimal choice of action, it is experimentation-sensitive.

To find the optimal Q function eventually, however, the agent must try out each action in every state many times. It has been shown (Watkins, 1989; Watkins & Dayan, 1992) that if Equation (4) is repeatedly applied to all state-action pairs in any order in which each state-action pair's Q-value is updated infinitely often, then $\hat{Q}^*$ will converge to Q^* and $\hat{V}^*$ will converge to V^* with probability 1 as long as α is reduced to 0 at a suitable rate.

Finally, Watkins (1989) has also described possible extensions to the one-step Q-learning method by using different value estimators, such as $\mathbf{r}^\lambda$ for $0 < \lambda < 1$, and he has illustrated the use of $\mathbf{r}^\lambda$ returns in Q-learning in his empirical demonstrations by memorizing past experiences and calculating these returns at the end of each learning period, where a learning period specifies the number of experiences occurring in the past the agent needs to store. The following section derives a novel algorithm that enables the value estimation process to be done incrementally.

4. Q(λ)-Learning

This section derives the Q(λ)-learning algorithm combining TD(λ) returns for general λ with Q-learning in an incremental way. Note that in terms of the notation introduced here, one-step Q-learning is simply Q(0)-learning, making it a special case.

For simplicity, in what follows we drop the superscript π in V^π and assume that the given policy π is the agent's greedy policy. Now let

$$e_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{V}_t(x_t) \tag{6}$$

and

$$e'_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{Q}_t(x_t, a_t) \tag{7}$$

where $\hat{V}(x) = \max_a \hat{Q}(x, a)$. Then, if we use Equation (7) for one step and Equation (6) thereafter, the difference between the TD(λ) return of Equation (2) and the estimated Q-value can be written as

$$\begin{aligned} \mathbf{r}_t^\lambda - \hat{Q}_t(x_t, a_t) &= e'_t + \gamma\lambda e_{t+1} + \gamma^2\lambda^2 e_{t+2} + \cdots \\ &\quad + \sum_{n=1}^{\infty} (\gamma\lambda)^n [\hat{V}_{t+n}(x_{t+n}) - \hat{V}_{t+n-1}(x_{t+n})]. \end{aligned} \tag{8}$$

If the learning rate is small, so that Q is adjusted slowly, then the second summation on the right-hand side of the above equation will be small.

The $Q(\lambda)$-learning algorithm is summarized in Figure 1, where $Tr(x, a)$ is the "activity" trace of state-action pair (x, a), corresponding to the "eligibility" trace as described in (Barto, Sutton & Anderson, 1983).

The main difficulty associated with Q(λ)-learning in a Markov decision process is that rewards received after a non-greedy action cannot be used to evaluate the agent's greedy policy since this will not be the policy that was actually followed. In other words, Q(λ)-learning is experimentation-sensitive, assuming that $\lambda > 0$ is fixed. One way around this difficulty is to zero λ on each step that a non-greedy action is taken. However, as argued in (Rummery & Niranjan, 1994), zeroing the effect of subsequent rewards on those prior to a non-greedy action is likely to be more of a hindrance than a help in converging to optimal policies since $\max_a Q(x, a)$ may not provide the best estimate of the value of the state x.

Still another difficulty is that changes in $\hat{Q}$ at each time step may affect r^{λ}, which will in turn affect $\hat{Q}$, and so on. However, these effects may not be significant for small α since they are proportional to α^2 (Peng, 1993).

At each time step, the Q(λ)-learning algorithm loops through a set of state-action pairs which grow linearly with time. In the worst case, this set could be the entire state-action space. However, the number of state-action pairs for which actual updating is required can be kept at a manageable level by maintaining only those state-action pairs whose activity trace $(\gamma\lambda)^n$ is significant, since this quantity declines exponentially when $\gamma\lambda < 1$. For a more elaborate procedure see Cichosz & Mulawka (1995). Another approach is to

1. $\hat{Q}(x, a) = 0$ and $Tr(x, a) = 0$ for all x and a
2. Do Forever:
   - (A) $x_t \leftarrow$ the current state
   - (B) Choose an action a_t according to current exploration policy
   - (C) Carry out action a_t in the world. Let the short-term reward be r_t, and the new state be x_{t+1}
   - (D) $e'_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{Q}_t(x_t, a_t)$
   - (E) $e_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{V}_t(x_t)$
   - (F) For each state-action pair (x, a) do
     - $Tr(x, a) = \gamma\lambda Tr(x, a)$
     - $\hat{Q}_{t+1}(x, a) = \hat{Q}_t(x, a) + \alpha Tr(x, a) e_t$
   - (G) $\hat{Q}_{t+1}(x_t, a_t) = \hat{Q}_{t+1}(x_t, a_t) + \alpha e'_t$
   - (H) $Tr(x_t, a_t) = Tr(x_t, a_t) + 1$

Figure 1. The Q(λ)-Learning Algorithm.

implement a Q(λ)-learning system on a parallel machine in which each state-action pair is mapped onto a separate processor. This corresponds directly to the kind of neural network implementation first envisioned for the actor-critic approach (Barto, Sutton & Anderson, 1983).

Finally, it is interesting to note that both Q(λ)-learning and actor-critic learning use TD(λ) returns as their value estimators through a trace mechanism. Therefore, it seems reasonable to expect the Q(λ)-learning algorithm to exhibit beneficial performance characteristics attributable to the use of TD(λ) returns for $\lambda > 0$, as illustrated in (Barto, Sutton & Anderson, 1983; Sutton, 1988). At the same time, both Q(λ)-learning and one-step Q-learning construct a value function on the state-action space rather than just the state space, making them both capable of discriminating between the effects of choosing different actions in each state. Thus, while Q(λ)-learning is experimentation-sensitive, unlike one-step Q-learning, it seems reasonable to expect it to be less so than actor-critic learning. Overall, then, Q(λ)-learning appears to incorporate some of the best features of the Q-learning and actor-critic learning paradigms into a single mechanism. Furthermore, it can be viewed as a potential bridge between them.

5. Discussion

This paper has only examined the Q(λ)-learning algorithm in which the TD(λ) returns are computed by taking the maximum Q values at each state visited. There are other possibilities, however. For example, the algorithm may estimate the TD(λ) returns by using the current exploration policy. This is the algorithm, called *sarsa*, described in (Rummery & Niranjan, 1994). In this algorithm, the update rule is

$$\Delta \mathbf{w}_t = \alpha(r_t + \gamma Q_{t+1} - Q_t) \sum_{k=0}^{t} (\gamma\lambda)^{t-k} \nabla_{\mathbf{w}} Q_k \tag{9}$$

where w denotes the weights of connectionist networks, and Q_{t+1} is associated with the action selected. In terms of Q(λ) learning, the right hand side of Equations (2D) and (2E) in Figure 1 would be replaced by

$$r_t + \gamma \hat{Q}_t(x_{t+1}, a_{t+1}) - \hat{Q}_t(x_t, a_t).$$

It is demonstrated (Rummery & Niranjan, 1994) that the overall performance of Q(λ) learning, including *sarsa*, shows less sensitivity to the choice of training parameters and exhibits more robust behavior than standard Q learning. See also (Pendrith, 1994).

Experiments involving both Markovian and non-Markovian tasks, whose details we omit here, were carried out to validate the efficacy of the Q(λ)-learning algorithm. The results showed that Q(λ)-learning outperformed both actor-critic learning and one-step Q-learning on all the experiments. The significant performance improvement of the Q(λ)-learning system over the simple Q-learning system (including the case where the Q-learner was given the experiences of the Q(λ)-learner) is clearly due to the use of the TD(λ) return estimation process, which has the effect of making alterations to

past predictions throughout each trial. If this is the main benefit conferred by TD(λ), one might expect model-based, multiple-update methods like priority-Dyna (Peng, 1993; Peng & Williams, 1993; Moore & Atkeson, 1994), to perform at least as well. However, additional experiments using such techniques showed that they performed significantly worse than Q(λ)-learning. We believe the reason for this is that the coarse state-space quantization used here has the effect of making the environment non-Markovian, and increasing λ makes TD(λ) less sensitive to this non-Markovian effect. Pendrith (1994) made a similar argument on a related algorithm.

It should be noted that both the fixed period learning process of Watkins (1989) for sufficiently long learning periods and the experience-replay process of Lin (1992) produce similar beneficial effects as that of Q(λ)-learning. However, both of these approaches operate in "batch" mode in that they replay, backwards, the memorized sequence of experiences that the learning agent has recently had.

From a computational standpoint, the incrementality of Q(λ)-learning makes it more attractive than Watkins' batch mode learning and Lin's experience replay process since the computation can be distributed over time more evenly, and thus under many circumstances can ease overall demands on the memory and speed. Similar arguments are made in (Sutton, 1988). Furthermore, incrementality speeds up learning. In one experiment where off-line Q(λ) learning was applied to experiences of its own and to those of on-line Q(λ) learning, it was found that the results of off-line Q(λ) learning in both cases were much worse than those obtained using on-line Q(λ) learning. One additional attractive characteristic of the Q(λ)-learning method is that it achieves greater computational efficiency without having to learn and use a model of the world (Peng, 1993; Peng & Williams, 1993; Sutton, 1990) and is well suited to parallel implementation.

Finally, although look-up table representation has been our main focus so far, it can be shown without difficulty that Q(λ) learning can be implemented on-line using connectionist networks, as is done in (Rummery & Niranjan, 1994).

6. Conclusion

The Q(λ)-learning algorithm is of interest because of its incrementality and its relationship to Q-learning (Watkins, 1989) and actor-critic learning (Barto, Sutton & Anderson, 1983). However, this algorithm, unlike the one-step Q-learning algorithm, cannot be expected to converge to the correct Q^* values under an arbitrary policy that tries every action in every state (although the obvious strategies of gradually reducing λ or gradually turning down the Boltzmann temperature as learning proceeds would probably allow such convergence). In spite of this, the Q(λ)-learning algorithm has always outperformed the one-step Q-learning algorithm on all the problems we have experimented with so far.

It is clear that in continuous-time systems, or even systems where time is discrete but very fine-grained, the use of algorithms that propagate information back one step at a time can make no sense or at least be of little value. In these cases the use of TD(λ) methods is not a luxury but a necessity. In general, λ can be viewed as a time scale parameter in such situations, and we argue that better understanding of its use in this regard is an important area for future research.

Acknowledgments

We wish to thank Rich Sutton for his many valuable suggestions and continuing encouragement. We would also like to thank the reviewers of the paper for their insightful comments and suggestions. This work was supported by Grant IRI-8921275 from the National Science Foundation.

References

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

Cichosz, P. & Mulawka, J. J. (1995). Fast and efficient reinforcement learning with truncated temporal differences. *Proceedings of the Twelfth International Conference on Machine Learning,* 99-107.

Lin, L. J. (1992). *Reinforcement learning for robots using neural networks.* Ph. D. Dissertation, Carnegie Mellon University, PA.

Moore, A. W. & Atkeson, C. G. (1994). Prioritized sweeping: reinforcement learning with less data and less time. *Machine Learning* **13**(1):103-130.

Pendrith, M. (1994). *On reinforcement learning of control actions in noisy and non-Markovian domains.* UNSW-CSE-TR-9410, University of New South Wales, Australia.

Peng, J. (1993). *Efficient Dynamic Programming-Based Learning for Control.* Ph. D. Dissertation, Northeastern University, Boston, MA 02115.

Peng, J. & Williams, R. J. (1993). Efficient learning and planning within the Dyna framework. *Adaptive Behavior* **1**(4):437-454.

Ross, S. (1983). *Introduction to Stochastic Dynamic Programming.* New York, Academic Press.

Rummery, G. A. & Niranjan, M. (1994). *On-line Q-learning using connectionist systems.* CUED/F-INFENG/TR 166, Cambridge University, UK.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning,* 216-224.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning* **3**:9-44.

Sutton, R S & Singh, S. P. (1994). On step-size and bias in temporal-difference learning. In *Eighth Yale Workshop on adaptive and Learning Systems*, pages 91-96, New Haven, CT.

Watkins, C. J. C. H. & Dayan, P. (1992). Q-learning. *Machine Learning* **8**:279-292.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards.* Ph. D. Dissertation, King's College, UK.

Received November 2, 1994
Accepted March 10, 1995
Final Manuscript October 4, 1995