INVESTIGATING THE BEHAVIOUR OF $Q(\lambda)$

J. Wyatt, G. Hayes and J. Hallam

1 Motivation

There is a spectrum of methods for learning robot control. At one end there are *model-free* methods (eg. Q-learning, AHC, bucket brigade), and at the other there are *model-based* methods, (eg. dynamic programming by value or policy iteration). The advantage of one technique is the weakness of the other. Model-based methods use experience effectively, but are computationally expensive; model-free methods are cheap computationally, but require an order of magnitude more experience. In the middle ground there are now methods like Q-DYNA [4] and Prioritised Sweeping [1] which use a learned model to speed temporal credit assignment. The optimal trade-off is dependent on the relative cost of experience and computation for particular tasks. Unfortunately we frequently do not know the cost balance for a particular task. Hence the ultimate goal of this work is to understand more about the sorts of methods that might work well on a wide variety of cost ratios, and in particular how model free methods might be extended.

In this paper we examine the behaviour of one such model-free algorithm, $Q(\lambda)$ [2]. This algorithm shows promise because it combines the best features of Sutton's TD(λ) algorithm [5] with those of Watkins Q-learning [6]. Despite being a model-free algorithm, it has been reported to outperform Prioritised Sweeping, the current best method for learning a policy and a model at the same time. Here we look at the effect on its performance of using replacing or accumulating traces, and at the problem of exploration sensitivity.

1 $t = 0$; $\hat{Q}(x,a) = 0$ and $Tr(x,a) = 0, \forall x, a$
2 $a_t = \arg\max_a(\hat{Q}(x_t, a))$
3 $e'_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{Q}_t(x_t, a_t)$
$e_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{V}_t(x_t)$
4 $\forall x, a$ do
$Tr(x,a) = \gamma\lambda Tr(x,a)$
$\hat{Q}_{t+1}(x,a) = \hat{Q}_t(x,a) + \alpha Tr(x,a) e_t$
5 $\hat{Q}_{t+1}(x_t, a_t) = \hat{Q}_t(x_t, a_t) + \alpha e'_t$
6 $Tr(x_t, a_t) = Tr(x_t, a_t) + 1$
7 step 2

x_t is the state at time t. a_t is the action chosen at time t. $\hat{Q}(x,a)$ is the estimate of the value of the state-action pair (x,a). $\hat{V}(x)$ is the estimated value of state x.
$\hat{V}(x) = \max_a \hat{Q}(x,a)$. r_t is the reinforcement immediately following time t.
$0 \le \gamma \le 1$ is the discount rate for reward. $0 \le \lambda \le 1$ is the decay rate for the eligibility trace.
Tr is a matrix of eligibility values, indexed by state and action. e'_t and e_t are error signals.

Figure 1: The $Q(\lambda)$ algorithm.

2 Background

Figure 1 shows the $Q(\lambda)$ algorithm as published. We can modify the algorithm to employ replacing traces rather than the accumulating trace used previously. The decay of the replacing trace is

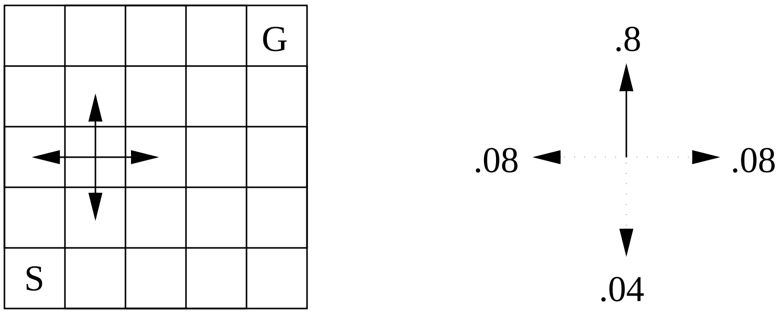

Figure 2: The solid arrows in the grid represent the most likely directions of travel for each of the four actions. On the right the solid arrow represents the most likely direction of travel for one action and the dotted arrows represent the other probabilities. The probability distributions for other actions may be derived by a series of 90° rotations.

determined by

$$Tr_{t+1}(x,a) = \begin{cases} 1 & \text{if } x = x_t \text{ and } a = a_t \\ \gamma\lambda Tr_t(x,a) & \text{otherwise} \end{cases} \tag{1}$$

Replacing traces have been devised and investigated recently [3]. It has been shown that the estimates produced by replacing and accumulating traces are closely related to two different Monte Carlo estimates of expected return. Replacing traces are interesting because they are closely linked to the maximum likelihood estimate of expected return. This is the same as the estimate of return we build through dynamic programming. The implication is that by using replacing traces we can produce estimates that are closer to the maximum likelihood estimate than by using accumulating traces. Replacing traces have also been shown to outperform accumulating traces in a control task. The main question concerning eligibility traces for Q-learning is *how* sensitive they make the algorithm to deviation from the optimal policy. As implemented by Peng and Williams Q(λ) is exploration sensitive, but they claim this sensitivity should be less than for the AHC algorithm. We decided to investigate the exploration sensitivity of Q(λ) more closely by measuring its performance when extreme deviations from the optimal policy occur. As an experimental control we used an exploration method where the number of deviations from the optimal policy was relatively low.

3 Experiment 1

3.1 Method

We tested $Q(\lambda)$ with both accumulating and replacing traces on a Markov Decision Process (MDP) represented in the form of 5-by-5 grid (Figure 2). In each state there were four possible actions, each with stochastic transitions; the start and goal states being in diagonal corners. The goal state is absorbing and on entry generates reinforcement of 1. All other states carry reinforcement of 0. The $Q(\lambda)$ algorithm was run with both accumulating and replacing traces on this task. For the agents which deviate from the optimal policy a counter-based exploration method was used. At each step this method chooses the action expected to take the agent to the least visited neighbouring state. This is determined by maximising $f(a,x)$.

$$f(a,x) = \beta\hat{Q}(x,a) + \frac{n(x)}{\sum_y n(y)\hat{P}_{xy}(a)} \tag{2}$$

where $n(s)$ is the number of times that state has been visited, and $\hat{P}_{xy}(a)$ is the estimated probability of the transition $x \rightarrow y$ given a. β controls the relative weighting of exploration and exploitation. Here $\beta = 0$ to give pure exploration. Counter-based exploration requires a model which was here constructed

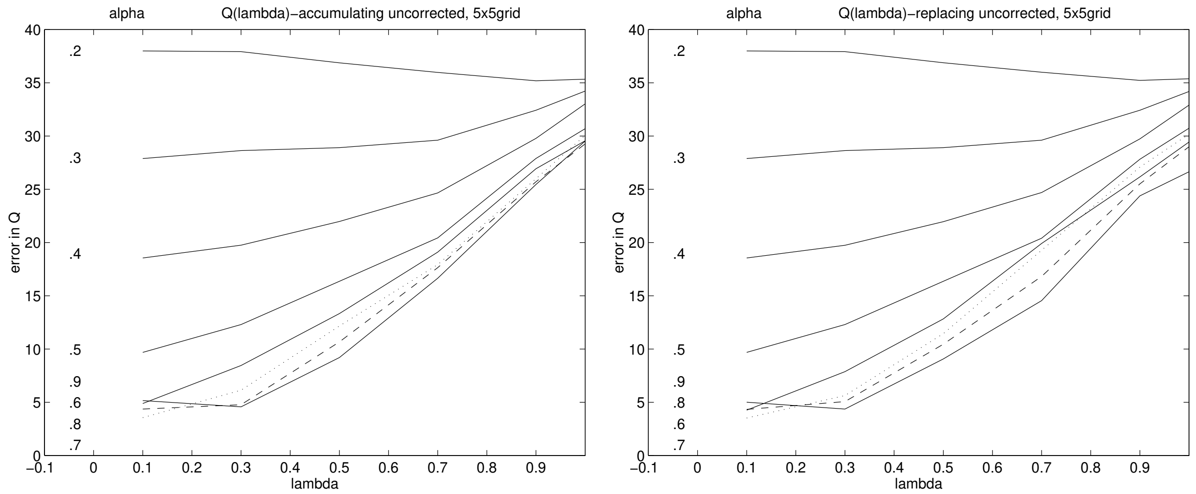

Figure 3: Performance of $Q(\lambda)$ when deviating heavily from the optimal policy (pure counterbased exploration) on a 5x5grid.

from observations of the actual state transitions. The agent will start by having no observations and randomly chooses actions. As the estimates used in the model improve it will try to drive the agent to the least visited neighbouring state. The use of a model here does not make $Q(\lambda)$ a model-based method, because the updates of the Q-values are still model-free. We are merely using this model-based exploration method to achieve deviation from the optimal policy. The agent was also run on the same task using a semi-uniform exploration method[1] with the probability of the optimal action being .95, and a uniform distribution across all actions being taken otherwise. We used each of these learners with both accumulating and replacing traces. For the counterbased exploration method it was found to be sufficient to take 10 runs of 20 trials each, and for the semi-uniform based exploration method we took 200 runs of 50 trials each[2].

3.2 Results

Figures 3 and 4 show the performance of each variant across a range of values of α and λ. Each line shows the performance of the algorithm at a particular α value. Performance was measured by calculating the total absolute error in the Q-values at the end of 20 trials. This error measure is given by Equation 3, where $\hat{Q}$ and Q^* are matrices indexed by state and action. Q^* is the matrix of actual Q-values[3].

$$error = \sum_{x,a} |Q^*(x,a) - \hat{Q}(x,a)| \tag{3}$$

It can be seen that the difference in the performance of agents with accumulating and replacing traces is negligible for most values of λ, whether or not we deviate from the optimal policy. This is largely because for this task the rate of revisits to each state-action pair is relatively low, and consequently accumulating traces do not build up to a point where the estimated Q-values become unstable. In short the Monte Carlo every-visit estimate is close to the first-visit estimate. The significant difference is not between trace types, but between exploration methods. When the agent deviates to a great extent from the optimal policy the performance of both kinds of trace deteriorates rapidly as $\lambda \rightarrow 1$ (Figure 3). But,

[1] We must deviate from the optimal policy with a finite probability each step. Otherwise the Q-values will not converge. This is the exploration method which gives us long term convergence while staying reasonably close to the optimal policy.

[2] For the counter-based learner the performance of each parameter set stayed the same after this point. Because the agents with uncorrected traces were the only agents not to converge in the long run to zero error for at least some parameter values it was found better to run other agents for up to 50 trials. Also, in order to ensure the significance of the differences in performance other agents were run for 200 runs for each parameter set.

[3] These were obtained by running value iteration on the transition function and the function defining one-step return.

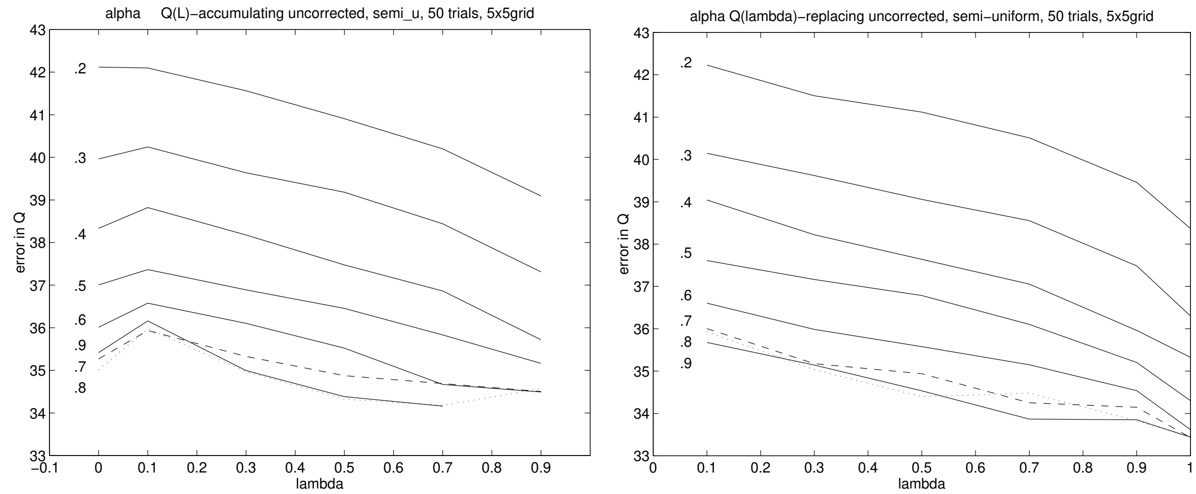

Figure 4: Performance of $Q(\lambda)$ when deviating to a small degree from the optimal policy (using semi-uniform exploration with $P_{best} = .95$) on a 5x5grid.

when the deviations from the optimal policy are small (the semi-uniform case: Figure 4) the accuracy continues to improve until λ reaches quite high values. Performance for replacing traces continued to improve all the way to $\lambda = 1$, whereas accumulating traces did produce unstable behaviour at $\lambda = 1$, as shown in Figure 5. Thus some difference was observed in the relative performance of accumulating and replacing traces.

4 Experiment 2

4.1 Method

From Experiment 1 it can clearly be seen that with fixed λ the $Q(\lambda)$ algorithm is exploration sensitive. This is unfortunate as one of the main advantages of one-step Q-learning is its insensitivity to exploration. There is, however, a simple solution. Watkins points out that if λ is varied according to whether the action chosen is a policy action or not[4], then exploration sensitivity can be avoided. To test this empirically we repeated the previous experiment, using Equation 4 to determine the value of λ. We refer to these algorithms as the corrected versions of Q(λ). The important point is that when a non-policy action is taken $\lambda = 0$, so that no previous Q-values are contaminated by the effects of experimentation.

$$\lambda_t = \begin{cases} 1 & \text{if } a_t = \arg\max_a(\hat{Q}(x_t, a)) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

4.2 Results

The results are shown in Figures 6 and 7. It can be seen that again the difference in performance between accumulating and replacing traces is negligible except at very high λ. By using corrected traces, however, we see marked improvement in performance over uncorrected traces. Specifically, when we deviate from the optimal policy, performance improves as $\lambda \to 1$ rather than the other way round. The only exception to this is when we use accumulating traces with semi-uniform exploration and $\lambda = 1$. In this instance behaviour becomes unstable, although performance still appears to be an improvement over that of uncorrected traces[5] (see Figure **??**).The semi-uniform exploration method produced very similar performance to when it was used with the uncorrected algorithms, and this

[4]A policy action is any action which appears to be optimal.

[5]Unfortunately it was not possible to obtain significant results at this point, as can be seen in Figure 5.

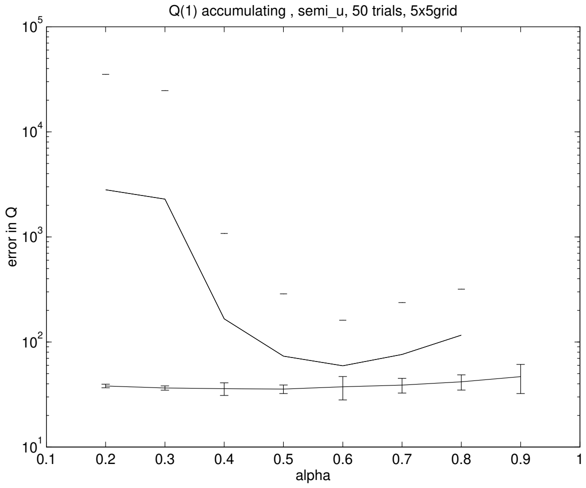

Figure 5: Performance of $Q(1)$ using both corrected (lower line) and uncorrected (upper line) accumulating traces with semi-uniform exploration on a 5x5grid. Because the y-axis is logarithmic it can clearly be seen that the variance across 200 runs is still so large that the differences are not significant.

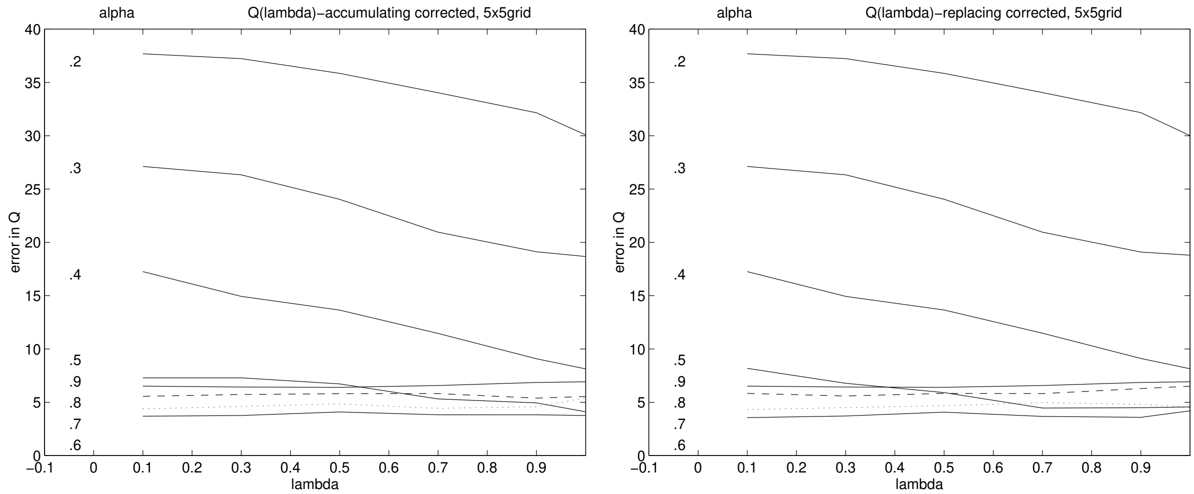

Figure 6: Performance of the corrected $Q(\lambda)$ with pure counterbased exploration on a 5x5grid.

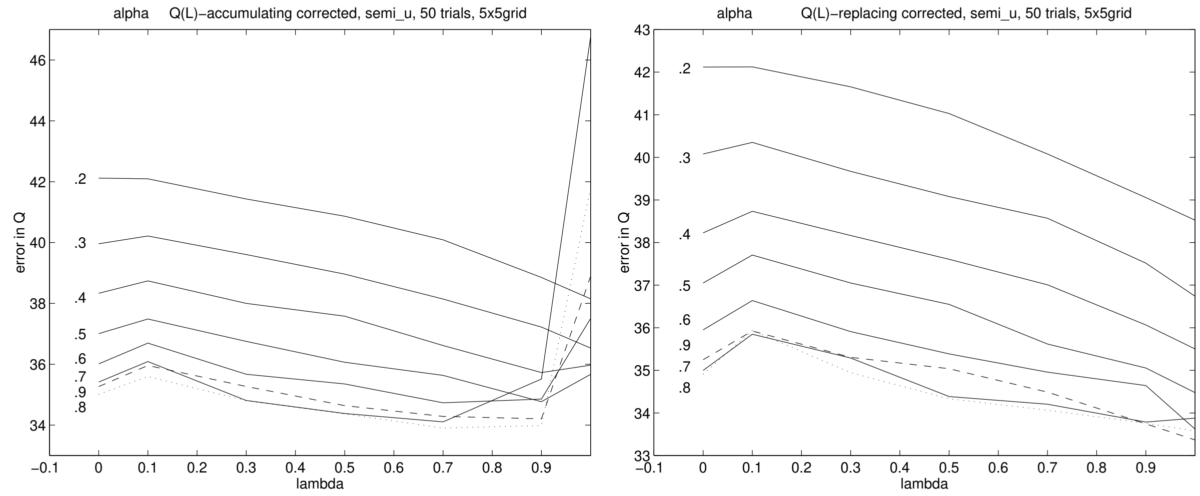

Figure 7: Performance of the corrected $Q(\lambda)$ with semi-uniform exploration on a 5x5grid. The performance of accumulating traces with $\lambda = 1$ is shown in Figure 5.

conforms with our expectation of the control. Consequently we can say that at every point corrected traces either performed as well as or better than uncorrected traces.

5 Discussion

We have showed that using corrected traces improves performance of $Q(\lambda)$ with high λ when the agent deviates from the optimal policy. It does so by removing exploration sensitivity. This is important because the ability to explore an environment efficiently (which will frequently entail deviating from the optimal policy) relies on this. Eligibility traces are essential for fast credit propagation. Combining them here produced an interesting deviation from the normal behaviour of eligibility traces with high λ values.

References

[1] Andrew W. Moore and Christopher G Atkeson. Prioritised sweeping: Reinforcement learning with less data and less real time. *accepted for publication in Machine Learing*, 1992.

[2] Jing Peng and Ronald J. Williams. Incremental multi-step q-learning. In W.W.Cohen and H.Hirsh, editors, *Machine Learning: Proceedings of the 11th International Conference*, pages 226–232, 1994.

[3] S.P. Singh and R.S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, pages 1–37, 1996.

[4] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In Bruce W. Porter and Ray J. Mooney, editors, *Machine Learning: Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224. Morgan Kaufmann, 1990.

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

[6] C.J.C.H Watkins. *Learning from delayed rewards*. Thesis, University of Cambridge, King's College, Cambridge, England, May 1989.