

APPROXIMATING Q-VALUES WITH BASIS FUNCTION REPRESENTATIONS

Philip Sabes

Department of Brain and Cognitive Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139
`sabes@psyche.mit.edu`

The consequences of approximating Q-Values with function approximators are investigated. Two criteria of optimality are introduced, a global and local criterion, and the viability of each is investigated for the case of a linear combination of prechosen basis functions. It is found that optimizing the global cost function is unlikely to result in nearly optimal policies when the set of bases is not complete. The local cost function is found to be plagued by local minima.

INTRODUCTION

Recently, the subject of learning from delayed rewards has attracted a good deal of interest. This interest is due in part to the development of algorithms such as Q-Learning [6], which are stochastic extensions of Dynamic Programming techniques. While Q-Learning is capable of learning to interact optimally with its environment without any *a priori* knowledge, it has some limitations. Q-Learning requires the storage of a Q-Value for every state-action pair, a daunting memory requirement for large dimensional real-world problems. For example, a 3-D Pole Balancing problem has a 10-D state space and a 3-D action space, requiring 10^{13} Q-Values for a crude 10 bins/dimension discretization. Additionally, the standard look-up table version of Q-Learning allows for no generalization over the state-action space, which seems unreasonably conservative for most real-world problems. Finally, it is usually the case that only a small number of the total state-action pairs are ever really visited. Thus it could be possible, using an alternative representation for the Q-Values, to concentrate computational power in the part of the domain that is relevant for the agent, thereby greatly reducing the number of free parameters needed.

For these reasons, a Q-Learner that employs an approximating representation for the estimated Q-Value function could extend the range of applications for which the algorithm is suited. While Q-Learning with lookup table Q-Value estimates provably converges to an optimal solution given some reasonable conditions [3, 7], these proofs do not extend in a straightforward manner to the case where general function approximators are used for the estimated Q-Values. Although there have been a number of successes combining reinforcement learning with function approximation, e.g. [4], there is evidence that approximating the value function can, in general, lead to problems when that value function is used for control. Thrun and Schwartz show, in this volume, that approximation can lead to systematic overestimation of the Q-Values during training [5]. Here, the effects on policy of estimating Q-Values with function approximators is investigated.

The next section provides a brief introduction to Q-Learning. Then two different error functions for Q-Learning with approximated Q-value estimates are presented, and, in the two sections which follow, each is evaluated.

Q-LEARNING

Q-Learning is an on-line, interactive, algorithm for learning an optimal policy in a markov state environment [6]. At each time step the agent is in a state, $x \in S$, and must take an action, $a \in A$. The agent then incurs a random cost, $C(x, a)$, with expected value $\bar{C}(x, a)$, and moves to a new state with transition probabilities $P(y|x, a), \forall y \in S$.

The goal of the learner is to find the policy which minimizes the total discounted cost-to-go, $\sum_{t=0}^{\infty} \gamma^t C(x_t, a_t)$, where $\gamma \leq 1$ is the discount rate. Q-Learning achieves this goal by learning, for each $(x, a) \in S \times A$, an estimate of the expected optimal discounted cost-to-go after executing action a in state x . In other words, it estimates the expected cost-to-go given that the current state is x , the current action is a , and for all future time steps it will execute the (unknown) optimal policy. These estimates are called Q-Values. Formally, the true Q-Values are define as,

$$Q^*(x_t, a_t) \equiv \bar{C}(x_t, a_t) + E_{\{x_t, x_{t+1}, \dots\}} \left[\sum_{s=t+1}^{\infty} \gamma^{(s-t)} \bar{C}(x_s, a_s^*) \right], \quad (1)$$

where a_s^* is the optimal action for state x_s . It can easily be shown that the Q-Values satisfy a version of the Bellman Equations:

$$Q^*(x, a) = \bar{C}(x, a) + \gamma \left[\sum_{y \in S} P(y|x, a) \min_{b \in A} Q^*(y, b) \right], \quad \forall (x, a) \in S \times A. \quad (2)$$

Dynamic Programming uses an iterative application of the Bellman Equations, with the current estimate plugged into the right hand side, to find the true value function. Here, since the learner does not know the transition probabilities or the $\bar{C}(x, a)$'s, it must use a Monte-Carlo sample of the new D.P. style estimate, giving the Q-Learning update rule:

$$Q_{t+1}(x_t, a_t) = (1 - \alpha_t) Q_t(x_t, a_t) + \alpha_t [C(x_t, a_t) + \gamma \min_b Q(x_{t+1}, b)], \quad (3)$$

where $\alpha_t < 1$ is the learning rate at time t .

BASIS FUNCTION REPRESENTATIONS OF THE Q-VALUES

The estimated Q-Value function will be approximated by a linear combination of a prechosen set of basis functions,

$$\hat{Q}(x, a) = \sum_i w_i g_i(x, a) = \mathbf{w}^T \mathbf{g}(x, a). \quad (4)$$

The bases could, for example, be gaussians randomly placed in the $S \times A$ space. The goal of the learner is to find the optimal set of weight vectors, \mathbf{w}^* . However, it is not clear what the criterion of optimality should be in this case. There are two immediate candidates for an error function, a global criterion, based on the distance from the estimate to the true Q-Values, and a local criterion, which measures the discrepancy between the left and right hand sides of Equation (2).

The global error function is defined as the L_2 distance from the true Q-Values,

$$E_1(\mathbf{w}) \equiv \frac{1}{2} \sum_{x, a} [\hat{Q}(x, a) - Q^*(x, a)]^2 = \frac{1}{2} \sum_{x, a} [\mathbf{w}^T \mathbf{g}(x, a) - Q^*(x, a)]^2$$

which has the gradient,

$$\partial E_1(\mathbf{w}) / \partial \mathbf{w} = \sum_{x, a} [\mathbf{w}^T \mathbf{g}(x, a) - Q^*(x, a)] \mathbf{g}(x, a). \quad (5)$$

To develop a learning rule from this gradient, one might proceed in a manner similar to stochastic gradient descent, with $\Delta \mathbf{w}$ at time t proportional to the (x_t, a_t) term in the above sum, except that the $Q^*(x, a)$'s

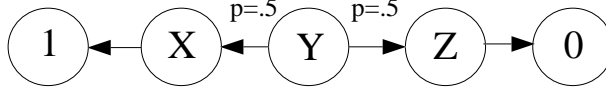


Figure 1: An example where the sampled gradient of the local error function points to the wrong minimum.

are unknown. However, at time t , the learner has a new Monte-Carlo estimate of the right hand side of Equation (2) (with the Q^* 's replaced by the \hat{Q}_t 's) available, which it can use as a new (biased) estimate of $Q^*(x_t, a_t)$. This gives an update rule for the w 's:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha_t [\mathbf{w}_t^T \mathbf{g}(x_t, a_t) - C(x_t, a_t) - \gamma \min_b \mathbf{w}_t^T \mathbf{g}(x_{t+1}, b)] \mathbf{g}(x_t, a_t). \quad (6)$$

Note that this rule is exactly Q-Learning for the case where there is basis function for each state-action pair which is unity for that pair and zero elsewhere. The weights then become the Q-Value estimates, and there is clearly a unique global minimum of $E_1(\mathbf{w})$, and a stationary point of Equation (6), when the \mathbf{w} are the true Q-Values. However, in the general case it is not true that the update rule has a stationary point at the minimum of $E_1(\mathbf{w})$. This is due both to the fact that the current Q-Value estimate is biased with respect to the true Q-Values and to the fact that the probability of visiting a given state-action pair depends on the current Q-Values estimates. In the sequel, the issue of convergence of Equation (6) will be set aside, and the the Q-Values which would result if the optimum of the global criterion could be achieved will be examined.

A second error function, more locally defined, is the squared difference between the two sides of Equation (2), again with the Q^* 's replaced by the \hat{Q} 's.

$$\begin{aligned} E_2(\mathbf{w}) &\equiv \frac{1}{2} \sum_{x,a} [\hat{Q}(x, a) - \bar{C}(x, a) - \gamma \sum_{y \in S} P(y|x, a) \min_b \hat{Q}(y, b)]^2 \\ &= \frac{1}{2} \sum_{x,a} [\mathbf{w}^T \mathbf{g}(x, a) - \bar{C}(x, a) - \gamma \sum_{y \in S} P(y|x, a) \min_b \mathbf{w}^T \mathbf{g}(y, b)]^2 \end{aligned} \quad (7)$$

which has the gradient,

$$\begin{aligned} \partial E_2(\mathbf{w}) / \partial \mathbf{w} &= \sum_{x,a} [\mathbf{w}_t^T \mathbf{g}(x, a) - \bar{C}(x, a) - \gamma \sum_{y \in S} P(y|x, a) \mathbf{w}_t^T \mathbf{g}(y, b^*(y, \mathbf{w}))] \\ &\quad [\mathbf{g}(x, a) - \gamma \sum_{y \in S} P(y|x, a) \mathbf{g}(y, b^*(y, \mathbf{w}))], \end{aligned} \quad (8)$$

where $b^*(y, \mathbf{w}) \equiv \arg \min_b \mathbf{w}^T \mathbf{g}(y, b)$. In this case, we can derive the weight update rule,

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha_t [\mathbf{w}_t^T \mathbf{g}(x_t, a_t) - C(x_t, a_t) - \gamma \mathbf{w}_t^T \mathbf{g}(x_{t+1}, b^*(x_{t+1}, \mathbf{w})) - \gamma \mathbf{g}(x_t, a_t) + \gamma \mathbf{g}(x_{t+1}, b^*(x_{t+1}, \mathbf{w}))]. \quad (9)$$

Consider again the case where there is basis function for each state-action pair and the weights become the Q-Value estimates. One sees that the difference between the two update rules is that the second learning rule, Equation (9), has the learning proceeding in both a forward and backward direction, temporally. It would seem that in many cases this is not desirable. For example, when there is an absorbing state where the true Q-Values are known after the first visit, knowledge of cost-to-go until reaching that state should percolate backwards only, as it does with the global rule. The local rule will, however, allow the incorrect estimates from states that immediately proceed the absorbing state to affect the correct absorbing state Q-Value estimate.

To be more concrete, consider the example in Figure 1, where each state has only one action, and the end states are absorbing with costs 1 and 0, respectively. The true Q-Values are $[Q_x^*, Q_y^*, Q_z^*] = [1, .5, 0]$. With a lookup table representation of the Q-Values, these values also minimize E_2 , and hence are a zero point of its gradient. However, if the learner is started in state Y on every trial, and moves down the gradient by stochastic sampling with the rule Equation (9), the surface that the learner is really moving down is

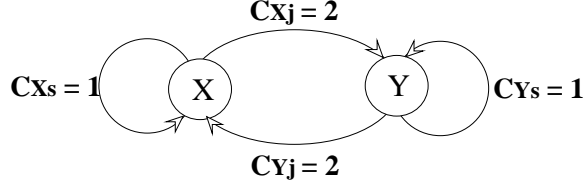


Figure 2: A simple environment.

$1/2[(Q_x^* - 1)^2 + (Q_y^* - Q_x^*)^2 + (Q_y^* - Q_z^*)^2 + (Q_z^* - 0)^2]$, which has a minimum at $[\cdot75, \cdot5, 25]$. Stochastic gradient descent of E_2 does not find the true minimum. In [8], Werbos finds the same problem with a similar error function defined for Dynamic Programming value function estimates. He evaluates the case where the dynamic equations are linear with additive zero-mean noise, and the cost function and value function estimates are linear functions of the current state. For his version of the local error function, he finds that the sampled gradient has an extra term which depends on the variance of the state transition noise, a term which is averaged over in the case of the global error function. In both his case and the one considered here, the source of the error is due to averaging over sampled transitions in a noisy environment. The global error function, which implements what Werbos called Heuristic Dynamic Programming, is able to converge despite this noise, due to the contraction properties of the D.P. Iteration (see, for example, [2]).

THE GLOBAL ERROR FUNCTION

The global criterion will not, in general, have an optimal Q-Value function which leads to a nearly optimal policy in the case where the bases do not span the space of functions from $S \times A$ to \mathbb{R} . To see why this should be the case, consider the simple environment shown in Figure 2, with $S = \{X, Y\}$ and $A = \{stay, jump\}$. If the true Q-Value function is represented as a four dimensional vector, and $\gamma = \cdot5$, then the true Q-Value vector is $\mathbf{Q}^* \equiv [Q_{X_s}^*, Q_{X_j}^*, Q_{Y_s}^*, Q_{Y_j}^*]^T = [2, 3, 2, 3]^T$. The optimal policy is to always *stay*.

Assume that there is only one basis function, $\mathbf{g} = [1, 0, 1, 0]^T$, say. Then the estimate of the Q-Values is just a single parameter multiplying the basis, $\hat{\mathbf{Q}} = w\mathbf{g} = [w, 0, w, 0]^T$. It can easily be seen that the weight which minimizes $E_1(w)$ is $w^* = 2$. However, this results in an estimate $\hat{\mathbf{Q}}$ which leads to exactly the wrong policy, always *jump*.

In fact, considering the four basis functions below, which span \mathbb{R}^4 ,

$$\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad (10)$$

it can be seen that any combination of the first three bases will lead to the wrong policy. Only the fourth of the bases above captures the component of the true Q-Value vector needed to produce the optimal policy. In order to be able to learn the correct policy with one basis vector, that vector must lie sufficiently close to the fourth basis above. If the bases are chosen randomly from a spherically symmetrical distribution, the probability of being able to learn the correct policy is approximately .29, .36, or .53, for 1, 2 or 3 bases, respectively.

In general, the effect of minimizing $E_1(\mathbf{w})$ is to project the true Q-Values onto the space spanned by the available bases. Upon such projection, the relative ordering of two elements in the vector can switch. For example, in the case above, $Q_{X_j}^* > Q_{X_s}^*$, but after projecting \mathbf{Q}^* onto $\mathbf{g} = [1, 0, 1, 0]^T$, the inequality is reversed, $\hat{Q}_{X_j} < \hat{Q}_{X_s}$. If the two elements in question represent Q-Values for two different actions starting from the same state, and if the larger of the two, in \mathbf{Q}^* , represents the optimal action from that state, then such a flipping of order upon projection would result in the wrong policy for that state.

In order to determine the probability of order flipping due to projection, a Monte Carlo simulation was performed with random Q-Vectors and bases. N -dimensional Q-vectors were chosen with uniform probability in $[0, 1]^N$, and basis vectors were chosen uniformly from a zero mean spherical gaussian with unit variance.

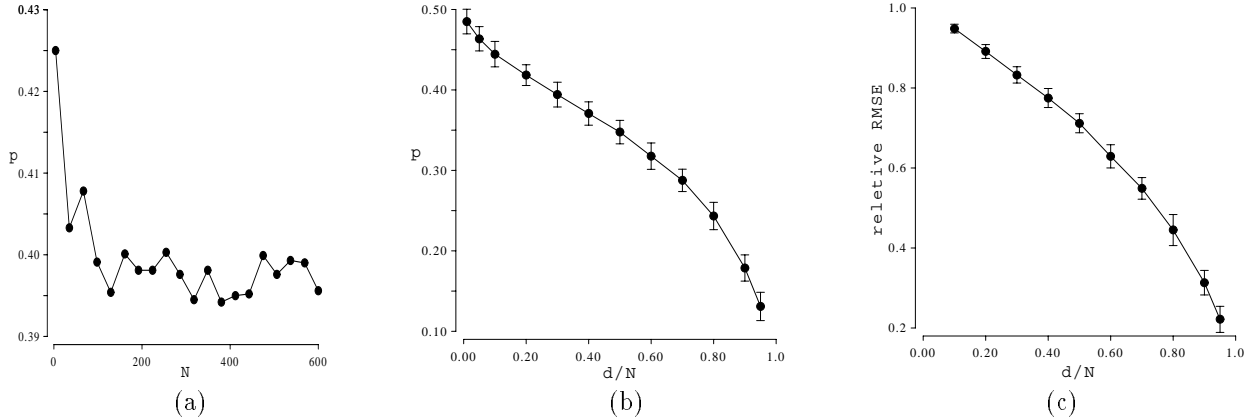


Figure 3: Percentage of N -dimensional vector element pairs that flipped their relative order upon projection onto a random d -dimensional subspace: (a) as a function of N , for $d/N = .30$; and (b) as a function of d/N . (c) Relative RMSE as a function of d/N .

A run consisted of selecting one Q -Vector and d bases, projecting Q^* onto the d -dimensional space spanned by the bases, and comparing the $.5N(N-1)$ pairs of elements in the original and projected Q -vectors to see if the relative order of the pair switched.

When the ratio of d/N was fixed, and the dimension of the space, N , was varied, a surprising result was observed. The percentage, p , of element pairs that flipped order started out high and decreased with increasing N until about $N = 100$, at which point it leveled off. A typical plot of p vs. N , for $d/N = .30$, is shown in Figure 3(a).

Next, the dependence of the large N value of p on the number of bases was investigated. For a given value of d/N , 50 runs were performed for each of 6 equally spaced values of $300 \leq N \leq 600$, and p was calculated for each run. The overall means and standard deviations of p as a function of d/N are shown in Figure 3(b). Note that the flipping probabilities are significant, even in the case when the number of independent bases is a large fraction of the dimension of the space of functions from $S \times A$ to \mathfrak{R} . For comparison, Figure 3(c) also shows the relative root mean square error, $\|\hat{Q} - Q^*\|^2 / \|Q^*\|^2$, as a function of d/N . Not surprisingly, the curve of p and relative RMSE have a qualitatively similar shape.

For a given state, it might be the case that one action has a much larger Q -Value than any other action. If the flipping probabilities depend heavily on the magnitude of the ratio of the two Q -Values, then in such a case, the probability of finding the correct policy might still be high. Thus, the dependence of $p = P(\hat{Q}_i > \hat{Q}_j | Q_i^* \leq Q_j^*)$ on the ratio Q_i^*/Q_j^* was investigated. Simulations were conducted as above, with $N = 400$ and $d/N = .3$ and $.8$. For each d/N , 75 runs were performed and flipping percentage was tallied in a histogram of bin-width .001. The results are shown in Figure 4, with each point representing 10 bins. Note that although p does depend on the ratio of the Q -Values, the flipping percentage is still significant even for very small ratios. Even in the case when one Q -Value is more than 1000 times the magnitude of another, the average flipping probability, $p_{.001}$, is approximately .30, for $d/N = .3$, or .070, for $d/N = .8$.

These results have strong implications for the likelihood of finding a nearly optimal policy. Consider the case of an environment where n actions are available at each state and the largest ratio between Q -Values for a given state is 100. Then, assuming independence of the behavior of different pairs, the probability of the optimal action, $\arg \min_b Q^*(x, b)$, still being optimal with respect to \hat{Q} is bounded above by $(1 - p_{.01})^{(n-1)}$. Then, even if the learner is using $.80 * N$ bases (in which case $p_{.01} = .09$), the expected fraction of states with the correct policy will only be about .68 if there are 5 actions per state, and .43 if there are 10 actions per state. Since a major motivation of using function approximators was to reduce memory requirements, it is likely that values of d/N used in practice will be much smaller, and as the d/N gets smaller and the number of actions per state grows, these probabilities fall off quickly.

To see how these results held up when the Q -Values were taken from a real problem and more typical basis functions were chosen, the global criterion was applied to the race track problem with gaussian basis functions. Details of the race-track follow [1]. The state space was discrete, with a 2-D position and a 2-D

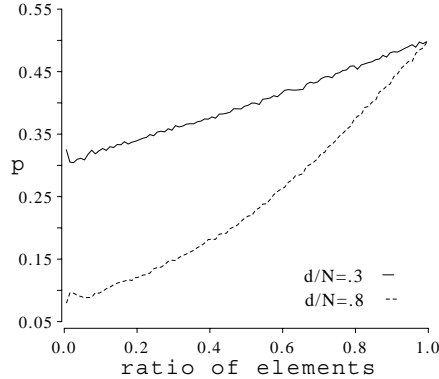


Figure 4: Percentage of N -dimensional vector element pairs that flipped their relative order upon projection onto a random d -dimensional subspace as a function of the ratio of the smaller element to the larger.

velocity. The actions consisted of forces (accelerations) applied to the car: $+1, 0, \text{ or } -1$ in each direction, giving 9 possible actions. The dynamics were stochastic, in that 10 percent of the time the action chosen was ignored and the car continued on with the same velocity as before, i.e.

$$\begin{array}{ll} \text{with probability .9:} & \mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_t + \mathbf{a}_t \\ & \mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{a}_t \end{array} \quad \begin{array}{ll} \text{with probability .1:} & \mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_t \\ & \mathbf{v}_{t+1} = \mathbf{v}_t \end{array}$$

A trial consisted of the car starting at one of the randomly chosen starting positions with zero velocity and being charged a cost of one for each time step until it crossed over the finish line. There was no explicit cost for running off the track, but the car was then sent back to (randomly) one of the start states and the velocity was reset to zero. Finally, there was no discount factor used, since in any case the optimal solution is simply the shortest path to cross the finish line.

Lookup table Q-Learning can be used to find nearly optimal solutions to these problems. Given these Q-Values, the weight vector, \mathbf{w}^* , which minimizes the $E_1(\mathbf{w})$ can be computed directly by setting the gradient in Equation (5) to zero:

$$\mathbf{w}^* = \left[\sum_{x,a} \mathbf{g}(x,a) \mathbf{g}^T(x,a) \right]^{-1} \left[\sum_{x,a} \mathbf{g}(x,a) Q^*(x,a) \right]. \quad (11)$$

Weights were actually computed with a weighted least-squared version of Equation (11), where each (x,a) was weighted by the number of times the lookup table Q-Learner visited the state, x . The weights found by this method were optimal in the $E_1(\mathbf{w})$ sense (modulo the weighting factors), but they often lead to policies which never made it to the finish line.

In the case of a very simple race-track with 40 positions and over 29,000 state-action pairs, lookup table Q-Learning converged to within 10 percent of the optimal average trial length (approximately 5 steps/trial) after 15,000 trials, and performed optimally by 30,000 trials. After convergence, only about 15 state-action pairs were visited with probability greater than .01 (about 1 visit per 20 runs), meaning that the effective number of dimensions in the weighted problem was on the order of 100 – 150. 100 gaussians with preset covariance matrices (a variety of diagonal covariances were tried) and centers chosen uniformly over the $S \times A$ space were used, and the optimal weights were found for 10 different bases sets. The average weighted relative RMSE was .29, a value comparable to the values found in the previous section for similar d/N . For all 10 cases, the Q-Values led to policies that never reached the finish line.

The Q-Values for each state were compared pairwise (i.e. only Q-Values for different actions in the same state were compared), and the average fraction (weighted as above) that flipped their relative order after projection onto the gaussian bases was .43. This value is somewhat larger than what was seen above for similar d/N and relative RMSE, but a difference is not unexpected given the radically different distribution of the bases. The average, weighted, percentage of the states for which the optimal action, according to the lookup table Q-Values, was no longer optimal with respect to the approximated values was .81. This number is smaller than the value we would expect if the flipping probabilities were independent, but it is still quite

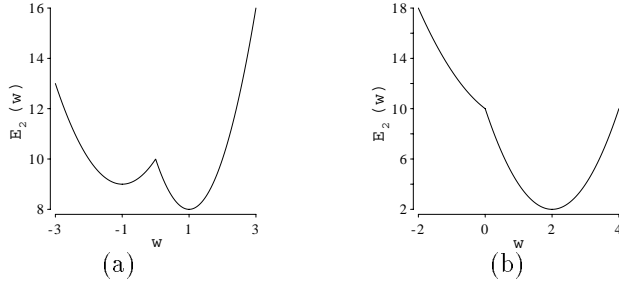


Figure 5: $E_2(w)$ vs. w . (a) for the basis $[1, 0, 1, 0]^T$, (b) for the basis $[0, 1, 0, 1]^T$.

large. It must be noted, however, that in cases when the true Q-Values are almost equal, relative flipping of their order upon projection does not matter. In this example, most states had more than one optimal action, and the Q-Values for those actions were often nearly equal. Thus, one should consider the average, weighted, fraction of states for which the action chosen had a lookup table Q-Value more than .30 below the optimal lookup table Q-Value. (True Q-Values were all integers, as they represented steps-to-go until the finish line. Thus a difference of .3 meant that the action really was non-optimal.) This fraction was computed to be .45. Again, this is lower than would be expected if the flipping probabilities were independent, but it is still high enough to prevent the policy from reaching the finish line.

In summary, when the Q-vector is projected onto a subspace of reduced dimension, the probability of the relative order of two Q-values flipping is high, resulting in an even higher probability of choosing a suboptimal action in any given state. The basic problem is that function approximation, which minimizes $E_1(\mathbf{w})$, is an indirect route to the ultimate goal of minimizing the expected cost-to-go. The relatively small errors introduced in the intermediate step can, as seen in the race track example, cause very large deviations from optimal performance.

THE LOCAL ERROR FUNCTION

One reason for being suspicious of the local error function has already been seen: the zero of the sampled gradient does not lie at the minimum of the error function itself. Here a second reason will be discussed. Even if unbiased estimates of the gradient were available, it is unlikely that a good policy would result, due to a proliferation of local minima on the the surface of the local error function.

Consider again the simple example from the previous section, illustrated in Figure 2. Again, assume the single basis vector, $g = [1, 0, 1, 0]^T$, so that our estimated Q-Values depend only on the scalar parameter, w , i.e. $\hat{\mathbf{Q}} = [w, 0, w, 0]^T$. Note that the error function $E_2(\mathbf{w})$ (see Equation (7)) depends on terms which include $b^*(y, \mathbf{w})$, and is otherwise quadratic in w . Thus, $E_2(\mathbf{w})$ must be evaluated separately for the case where $w < 0$, so the $b^*(y, w) = stay, \forall y$, and the case where $w > 0$, so the $b^*(y, w) = jump, \forall y$.

$$E_2(w) = \begin{cases} (w + 1)^2 + 9 & \text{if } w < 0 \\ 2(w - 1)^2 + 8 & \text{if } w > 0 \end{cases} \quad (12)$$

As can be seen in Figure 5(a), $E_2(w)$ has two local minima, one on each side of the origin. The deeper minimum, at $w = 1$, leads to the incorrect policy, always *jump*. However, the minimum at $w = -1$ gives the desired policy, always *stay*. The existence of these two local minima means that convergence to the optimal policy using the local learning rule will depend heavily on the initial conditions. In general, the minimum of the parabola corresponding to one side of the origin may or may not lie on that side. If, for example, the analysis is redone for the basis $[0, 1, 0, 1]^T$, one sees that there is only one local minimum, at the value $w = 2$, which will lead to the optimal policy (see Figure 5(b)). Although, in these two examples there was a local minimum in the optimal policy's half-plane, this is not always true; it is a simple exercise to construct a cost schedule that results in no local minima or a local minimum only on the side of the origin corresponding to the incorrect policy.

The piecewise quadratic nature of $E_2(\mathbf{w})$ is completely general. The quadratic pieces lie in conical regions, "policy cones", which are separated by hyperplanes with O^{th} or 1^{st} order discontinuities, corresponding to

the values of \mathbf{w} where $\mathbf{w}^T \mathbf{g}(x, a) = \mathbf{w}^T \mathbf{g}(x, b)$, for some $x \in S$, and some $a, b \in A, a \neq b$. A general rule has yet to be found for predicting whether a policy cone's quadratic surface will have a local minimum inside the cone, generating a policy stable with respect to the local learning rule. In any case, it is likely that there will be more than one stable solution, and thus even if unbiased gradient information were available, the final policy would depend heavily on the initial weights of the bases.

CONCLUSION

The implications of using function approximators to represent Q-Values has been investigated. Two criteria of optimality were considered. The main result concerning the global criterion is that optimality in the sense of least-squares distance of the Q-Value estimates to their true values does not translate into nearly optimal policies. When the global criterion is used in conjunction with a linear combination of bases functions, it will, with high probability, lead to extremely suboptimal policies in the case where the bases do not span the state-action space.

The local learning criterion, on the other hand, may have local minima in the conical regions of weight space corresponding to any of the possible policies. Thus, while there may be minima near the optimal policy, the policy the learner finds will depend heavily on the initial conditions of the weight vector.

These results suggest that a linear combination of basis functions will often not be a feasible representation for estimating the Q-Values. Although a crafty choice of bases could result in a good policy with a relatively small number of bases, this could require knowing the Q-Values in advance. Although that assumption was made here, the goal of this work is to point out potential difficulties with using function approximators in more general reinforcement learning settings. In any case, success of combining function approximators with Q-Learning will require careful selection of the approximators being used. More complicated or more specialized representation schemes will have to be employed, making analysis of convergence and performance even more difficult.

Acknowledgments

This research was supported by an ONR Graduate Fellowship. I would like to thank my advisor, Michael Jordan, for his guidance, and Peter Dayan for his insightful comments and helpful criticism.

References

- [1] Barto, A. G., Bradtke, S. J., and Singh, S. P. (1993). Learning to Act Using Real-Time Dynamic Programming. (COINS technical report 91-57). Amherst: University of Massachusetts.
- [2] Bertsekas, D.P. (1987). *Dynamic Programming* Englewood Cliffs, NJ: Prentice Hall, Inc.
- [3] Jaakkola, T., Jordan, M. I., and Singh, S. P. (1993). On the Convergence of Stochastic Iterative Dynamic Programming Algorithms. Submitted for Publication.
- [4] Tesauro, G. J. (1992). Practical Issues in Temporal Difference Learning *Machine Learning*, **8**, pp.257–277.
- [5] Thrun, S. B. and Schwartz, A. (1993). Issues in Using Function Approximation for Reinforcement Learning. Same volume.
- [6] Watkins, C. J. C. H. (1989). Learning from Delayed Rewards. PhD Thesis, University of Cambridge, England.
- [7] Watkins, C. J. C. H. and Dayan, P. (1992). Q-Learning. *Machine Learning*, **8**, pp.279–292.
- [8] Werbos, P. J. (1990). Consistency of HDP Applied to a Simple Reinforcement Learning Problem. *Neural Networks*, **3**, pp.179–189.