When the Best Move Isn't Optimal: Q-learning with Exploration

George H. John
Computer Science Department
Stanford University
Stanford, CA 94305-4110
gjohn@cs.stanford.edu
http://robotics.stanford.edu/~gjohn

Abstract

In delayed reinforcement learning, an agent is concerned with the problem of discovering an optimal *policy*, a function mapping states to actions. The most popular delayed reinforcement learning technique, Q-learning, has been proven to produce an optimal policy under certain conditions. However, often the agent does not follow the optimal policy faithfully—the agent must also *explore* the world. The optimal policy produced by Q-learning is no longer optimal if its prescriptions are only followed occasionally. In many situations (*e.g.*, dynamic environments), the agent never stops exploring. In such domains Q-learning converges to policies that are suboptimal when combined with the exploration policy, and we give several examples of this problem. We present $\overline{\text{Q}}$-learning, a slight modification of Q-learning that provides a policy resulting in higher reward when combined with a particular exploration strategy.

1 Introduction

Delayed reinforcement learning attempts to solve a class of important problems called *Markov Decision Processes* (MDP). In such problems, an agent finds the world in some state and must choose some action to perform. Executing an action results in some new state becoming the current state, and the execution also might result in some immediate reward or punishment r_t. Here we only discuss *finite* MDP's, where the number of states and actions is finite. We further assume that all states are fully observable; that is, the agent's perceptual capabilities are sufficient to distinguish between all possible states. The goal is to maximize future cumulative discounted reward $\sum_{t=0}^{\infty} \gamma^t r_t$. The γ term, referred to as the *discount rate*, insures the finiteness of the cumulative reward. (Schwartz (1993) discusses the undiscounted case, but the question of whether to discount is peripheral to our concerns.)

We may completely describe the behavior of the agent with a policy function π mapping states to actions that specifies the action to be executed by the agent in each state. The job of the learning algorithm is to discover that policy π^* which maximizes cumulative discounted reward. Q-learning, developed in Watkins (1989), is an algorithm that attempts to discover π^* by building and refining estimates $\widehat{Q}(s, a)$ of the expected future cumulative discounted reward for each state-action pair. The "Q-value" of an action in a given state is the estimated cumulative discounted reward received by taking that action and following the optimal policy thereafter. Thus, once Q-values have converged an agent may perform optimally by greedily choosing from among the available actions the one with highest Q-value. Q-learning is a very powerful algorithm, because theorems regarding its convergence to the optimal policy have set it on sound theoretical ground (Watkins & Dayan 1992) and it also seems to work well in practice (Mahadevan & Connell 1992).

1.1 When is Q-learning the Right Thing to Do?

The theoretical objective of the Q-learning algorithm seems to be often misunderstood, and we wish to address this concern here in detail. Our chief objection is that although Q-learning is guaranteed to converge to an optimal policy, when actually used in practice it is almost always combined with some exploration algorithm. Exploration refers to the general idea of taking actions that do not maximize expected reward, but that we hope might give new knowledge about the world and thus lead to an improved policy. The dilemma of whether to *exploit* (take an action that the current policy identifies as the best) or to *explore* is a problem common to all reinforcement learning algorithms. Our goal in this paper is to argue that while in some cases we do wish to discover the optimal policy π^*, in most cases we are in fact searching for some policy $\pi^{(*|e)}$; that is, some policy that is optimal when *combined* with a given exploration strategy e.

In many reinforcement learning problems, it is important to have *online learning*, in which the agent (perhaps an autonomous robot or situated software agent) must learn as it goes along. If reinforcement learning were used to control a situated agent in a dynamic environment, we would want the agent to be able to take advantage of serendipitous changes and attempt to work around any harmful changes in the environment (*e.g.*, Sutton's (1990) "unblocking" and "blocking" problems). We might say that we need an *adaptive*, rather than an *adapted* agent. In such cases where exploration is always required, online learning is the only option, and the modified version of Q-learning presented here should be used.

Note that as motivation for Q-learning, Watkins (1989) discusses evolutionary adaptation of behavior, viewing the behavior of a given species as a result of "exploration" by evolution, but now fixed in the genes. Thus the adaptation and exploration in the past yields a fixed behavior in the present, but often we want the

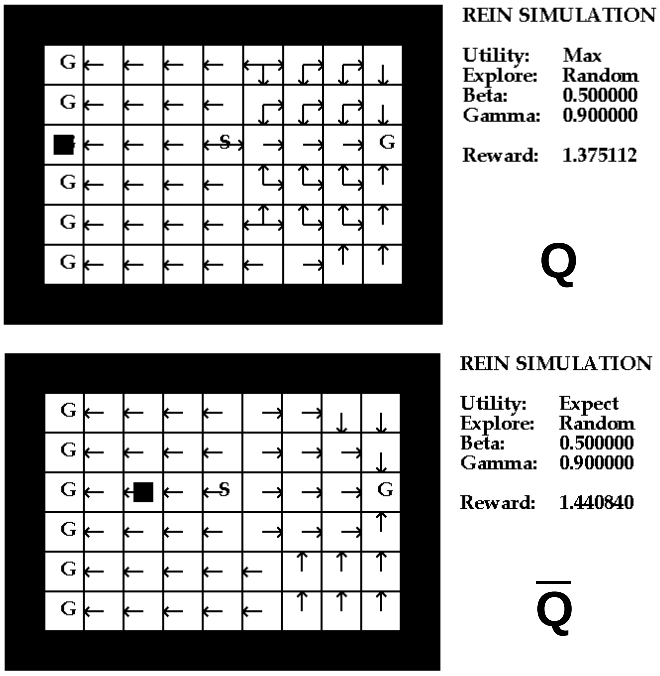

Figure 1: The "wall of pleasure" world, showing policies found using Q-learning and $\overline{\text{Q}}$-learning. The starting state is marked S, the goal states G, and the policy found is denoted by arrows. *Reward* shows the total reward received by the agent divided by the total number of timesteps (10^6) for the simulation (The small rectangle appearing in the pictures is actually our "robot" getting caught in the snapshot.)

present behavior to be adaptive as well. Watkins also notes our key point of contention: "Note that this is learning of the *ability* to follow a maximally efficient strategy: an animal with this ability need not always actually follow the efficient strategy, but it can do so if it chooses to." It is precisely the point of this paper that *if* the agent does not always follow the optimal policy but rather departs from the optimal policy in some *systematic* way, then there exists a policy that yields *greater* reward than the optimal policy when it is also disobeyed in a systematic way.

1.2 An Example: The "Wall of Pleasure"

As motivation, we present evidence from a grid-world domain showing that Q-learning can be improved upon when we intend for our agent to explore continually. The example is a toy world concocted as an obvious case where a naive combination of exploration with the optimal policy π^* leads to trouble. Figure 1 shows two policies generated by our simulator using Q-learning and $\overline{\text{Q}}$-learning, our modification to Q-learning which searches for $\pi^{(*|e)}$. We call this world the "Wall of Pleasure" because a transition into each state adjacent to the left wall generates a reward for the agent. (After entering a goal state, the agent is repositioned at the starting

Table 1: Standard notation used in reinforcement learning, with additions.

MDP	Markov Decision Process. The current state is perfectly known but actions may be stochastic (*i.e.*, not predictable with certainty).
β,γ	learning rate, discounting rate
x, a, y, r	Some current state, action, resulting state, and reward received.
in state x_t.	
$P^a(x, y)$	Probability that executing action a causes transition from x to y.
$\pi(x)$	The action assigned to state x by a stationary policy π
π^*	The optimal policy, one that assigns actions to states such that cumulative reward is maximized by faithfully following pi^*.
$\pi^{(*\|e)}$	The optimal policy *given* a particular exploration strategy
$V^\pi(x)$	The analytical expected future cumulative reward received when starting in state x and executing policy π thereafter.
$V(x)$	Defined to be $V^{\pi^*}(x)$
$\widehat{V}(x)$	The current approximation of $V(x)$.
$Q^\pi(x, a)$	The analytical expected future cumulative reward received when starting in state x, taking action a now and following π thereafter.
$Q(x, a)$	Defined as $Q^{\pi^*}(x, a)$
$\widehat{Q}(x, a)$	The current approximation of $Q(x, a)$.

state.) For exploration, the agent executed a random action 30% of the time.

The policy found by Q-learning is frequently indifferent as to which action should be taken. At the starting state the estimated rewards from going left or right are the same, so Q-learning does not care which way to go. This is perfectly sensible if no exploration is performed – whether the agent goes left or right, the best thing to do is to head straight for the nearest goal, which is only four steps away in either direction. However, given that exploration *is* going to occur, it is better to move left from the starting state. On the left side of the world, exploration can hardly hurt the agent – if it moves up or down, it has wasted one timestep but is still the same distance from the nearest goal. However, on the right side of the world, any exploration is likely to move the agent further from the nearest goal. Thus, our $\overline{\text{Q}}$-learning algorithm (which takes into account the exploration strategy used) is able to discover that greater reward is received by always moving left from the starting state. While the agent using Q-learning achieves an average reward of 1.38, our agent using $\overline{\text{Q}}$-learning achieves an average reward of 1.44.

2 The Algorithms: Q-learning and $\overline{\text{Q}}$-learning

As mentioned above, the goal of Q-learning is to find a policy π^* that maximizes the reward received by the agent over time. To define this policy and the algorithms used to discover it, we will need a bit of notation. Fortunately, Chrisman (1993) has done the reinforcement community the favor of collecting various suggestions and coming up with standard notation for Q-learning, a slightly modified subset of which is described in Table 1.

The most popular reinforcement technique, Q-learning (Watkins 1989), approximates $V(s)$ stochastically without requiring a model using these update rules:

$$\widehat{Q}(x,a) \leftarrow \beta(r+\gamma\widehat{V}(y)) + (1-\beta)\widehat{Q}(x,a) \tag{1}$$

$$\widehat{V}(x) \leftarrow \max_a \widehat{Q}(x,a) \tag{2}$$

Equation 2 tells us how to update $\widehat{V}(s)$ after updating $\widehat{Q}(x,a)$: we replace $\widehat{V}(x)$ with the greatest (over all actions a) $\widehat{Q}(x,a)$. This is not always correct. $V(x)$ is defined to be the *expected* cumulative future reward received by following the optimal policy after starting in state x. In mathematics,

$$V(x) \stackrel{\text{def}}{=} \sum_a \Pr(a|x)Q(x,a) \ , \tag{3}$$

where $\Pr(a|x)$ denotes the probability that the agent will choose a in state x. Equations 3 and 2 only agree if the agent really does follow a policy π^* that chooses the action with the best Q-value at each step. This is not true of an agent that must also perform exploration online. By ignoring the exploration component, the agents finally arrive at policies that are suboptimal when combined with exploration.

The solution to the problem is straightforward and already given in the arguments above. We replace Equation 2 with

$$\widehat{V}(x) \leftarrow \sum_a Pr(a|x)\widehat{Q}(x,a) \ . \tag{4}$$

We call the resulting algorithm $\overline{\mathrm{Q}}$ since in statistics $\overline{X}$ denotes a sample mean of a random variable, which connotes the expected value calculation above.

Note that the effect of Equation 4 may be achieved using Equation 2 by pretending that the agent always selects the action $a_i = \arg\max_a Q(s,a)$ and modeling exploration (sometimes randomly choosing a_j instead) as a stochastic component to a_i. The approaches are equivalent mathematically, which we believe demonstrates the correctness of our approach. However, the stochastic approach presents the problem that we are requiring the agent to relearn what it already knows. Given the exploration strategy, if we can solve for $Pr(a|x)$ exactly, then using Equation 4 will allow faster learning than estimating the state transition and reward probability distributions for a.

3 Experiments

Below we discuss specific solutions to the expected-utility problem in the context of the two most popular exploration strategies: random walk exploration and Boltzmann exploration. We show that $\overline{\mathrm{Q}}$-learning learns policies with higher reward than Q-learning. We use simple grid worlds as in Sutton (1990).

3.1 Random Walk Exploration

A common exploration strategy is to simply follow the optimal policy with probability k, and select actions uniformly at random the rest of the time. Thus instead of always choosing $\arg\max_a Q(s,a)$, the actual policy is probabilistic.

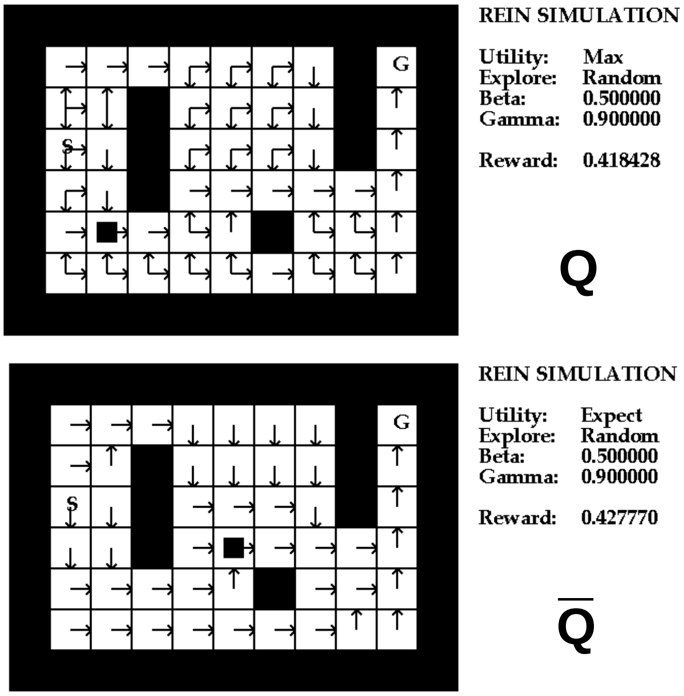

Figure 2: The world from Sutton (1990). The goal gives a reward of +9.

$$\Pr(a|x) = \begin{cases} k + \frac{1-k}{|A|} & \text{if } a = \arg\max_i Q(x, a_i) \\ \frac{1-k}{|A|} & \text{otherwise} \end{cases} \tag{5}$$

As a testament to the benefit of occasionally performing random actions, an errant keystroke one day caused an experiment to be run on the world from Sutton (1990), shown in Figure 2, instead of the more contrived world in Figure 1. Perhaps surprisingly, $\overline{\text{Q}}$-learning also outperforms Q-learning in Sutton's world. Notice that although Q-learning is often undecided as to which of two or three actions to perform, $\overline{\text{Q}}$ always prefers a single action, and its preference is always justified by the exploration strategy. For example, along the bottom row, $\overline{\text{Q}}$ always moves right, exactly the right thing to do. Since it stays against the bottom wall, a DOWN exploration step can't move it further from the goal – it's already as far down as it can get. But since Q-learning may move up when it's on the bottom row, an exploration step can send it back down again, moving it further from the goal.

A further experiment (the "walls of pain") was run using a world similar to Figure 1, but the agent was always started from the bottom left and the single goal was at the top right. Running into a wall causes the agent "pain," a reward of -1. Again $\overline{\text{Q}}$-learning produces a more sensible policy, one that moves the agent away from the wall quickly and keeps it away so that exploration steps will not be harmful. Q-learning is indifferent towards following walls or staying away from walls. Q-learning

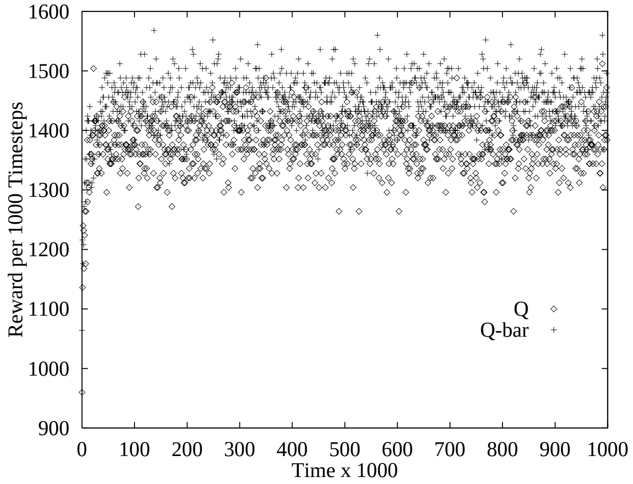

Figure 3: Reward per 1,000 timesteps for the Q and $\overline{\mathrm{Q}}$ algorithms using Boltzmann exploration on the "Wall of Pleasure" domain.

converted to an average reward of .440 while $\overline{\mathrm{Q}}$ converged to .468.

3.2 Boltzmann Exploration

Another common exploration strategy, Boltzmann exploration, is also probabilistic, where the probability distribution over the actions is the Boltzmann distribution:

$$\Pr(a|x) = \frac{e^{\frac{Q(s,a)}{T}}}{\sum_i e^{\frac{Q(s,a_i)}{T}}} \quad . \tag{6}$$

Our experiments used a constant temperature $T = 1$. Usually T is initialized to some large number and then lowered according to some *annealing schedule*. At some point T becomes very small and stays small. If we had used an annealing schedule, for purposes of $\overline{\mathrm{Q}}$-learning the distribution $Pr(a|x)$ should be modeled using the *final* value of T in the annealing schedule (assuming that the agent will always explore to some extent). Figure 3 shows the superior reward achieved by $\overline{\mathrm{Q}}$-learning on the "Wall of Pleasure" example. The policies found by Q- and $\overline{\mathrm{Q}}$-learning were approximately the same as in Figure 1.

3.3 A General Solution

The solution described in the previous subsection only generalizes to stationary probabilistic exploration strategies. That is, strategies where $P(a|x)$ is only a function of the Q-values at state x. Such exploration strategies may all be called *random walk* strategies, and Whitehead (1991) has given strong arguments against such exploration strategies.

To be content with Equation 4 we need to be able to combine it with other exploration strategies, such as the counter-based methods in Sutton (1990) and Thrun

(1992). A general solution is to do away with $V(x)$ altogether, instead using

$$Q(x,a) \leftarrow \beta(r + \gamma Q(y,a')) + (1-\beta)Q(x,a) \ , \tag{7}$$

where a' is the action actually selected in state y. In this way we approximate $Q(x,a)$ by replacing it with a weighted average of itself and the Q-value of the action that was actually taken in the next state. The only problem is that a' is random since exploration is being used, and thus there can be high variance in the update equation. To fix this, we can sample a' many times and replace $Q(y,a')$ above with its average. This strategy has been tested, with no degradation in performance.

4 Conclusion

Q-learning works as advertised, learning an optimal policy. However, when agents continually explore their environment, $\overline{\text{Q}}$-learning produces policies that result in higher reward by taking this exploration into account. $\overline{\text{Q}}$-learning is equivalent to Q-learning where the effect of exploration is modeled as stochasticity in the environment, and thus we presume that the $\overline{\text{Q}}$-learning algorithm is correct. Essentially, Q-learning solves an unconstrained optimization problem while $\overline{\text{Q}}$-learning incorporates probabilistic constraints from the exploration strategy, and these constraints may complicate matters to the extent that we can no longer prove convergence to the optimal policy given the constraints. $\overline{\text{Q}}$-learning allowed our agent to find policies yielding higher average reward than the policies found by standard Q-learning.

References

Bellman, R. E. (1962), *Dynamic Programming*, Princeton University Press.

Chrisman, L. (1993), Notational conventions for the reinforcement learning workshop at ML93, Personal Communication.

Mahadevan, S. & Connell, J. (1992), "Automatic programming of behavior-based robots using reinforcement learning", *Artificial Intelligence* **55**(2-3), 311–365.

Schwartz, A. (1993), A reinforcement learning method for maximizing undiscounted rewards, *in* P. Utgoff, ed., "Proceedings of the Tenth International Conference on Machine Learning", Morgan Kaufmann.

Sutton, R. S. (1990), Integrated architectures for learning, planning, and reacting based on approximating dynamic programming, *in* B. Porter & R. Mooney, eds, "Proceedings of the Seventh International Conference on Machine Learning", Morgan Kaufmann.

Thrun, S. (1992), Efficient exploration in reinforcement learning, Technical Report CMU-CS-92-102, CMU School of Computer Science.

Watkins, C. (1989), Learning from Delayed Rewards, PhD thesis, Cambridge University. Psychology Department.

Watkins, C. & Dayan, P. (1992), "Q-learning", *Machine Learning* **8**(3/4), 279–292.

Whitehead, S. (1991), A complexity analysis of cooperative mechanisms in reinforcement learning, *in* "AAAI-91: Proceedings of the Ninth National Conference on Artificial Intelligence", AAAI Press/MIT Press.