To Discount or not to Discount in Reinforcement Learning: A Case Study Comparing R Learning and Q Learning

Sridhar Mahadevan
Department of Computer Science and Engineering
University of South Florida
Tampa, Florida 33620
mahadeva@csee.usf.edu

Abstract

Most work in reinforcement learning (RL) is based on discounted techniques, such as Q learning, where long-term rewards are geometrically attenuated based on the delay in their occurence. Schwartz recently proposed an *undiscounted* RL technique called R learning that optimizes *average reward*, and argued that it was a better metric than the discounted one optimized by Q learning. In this paper we compare R learning with Q learning on a simulated robot box-pushing task. We compare these two techniques across three different exploration strategies: two of them undirected, Boltzmann and semi-uniform, and one recency-based directed strategy. Our results show that Q learning performs better than R learning, even when both are evaluated using the same undiscounted performance measure. Furthermore, R learning appears to be very sensitive to choice of exploration strategy. In particular, a surprising result is that R learning's performance noticeably deteriorates under Boltzmann exploration. We identify precisely a limit cycle situation that causes R learning's performance to deteriorate when combined with Boltzmann exploration, and show where such limit cycles arise in our robot task. However, R learning performs much better (although not as well as Q learning) when combined with semi-uniform and recency-based exploration. In this paper, we also argue for using medians over means as a better distribution-free estimator of average performance, and describe a simple non-parametric significance test for comparing learning data from two RL techniques.

1 Introduction and Motivation

Recently, several different reinforcement learning (RL) techniques have been developed, such as Q learning [20], TD(λ) [14], and R learning [12]. Although the former two techniques have been applied with some success to diverse domains, ranging from backgammon [18] to robotics[5, 9, 10], very little is known about the empirical effectiveness of R learning. In his paper proposing R learning [12], Schwartz argued convincingly that optimizing average reward is better than optimizing discounted reward. He constructed simple Markov decision problems where high short term reward can cause Q learning (for fixed values of the discount factor γ) into converging to a sub-optimal policy, whereas R learning is able to avoid this local maxima and converge to the best long-term optimal policy. But how well does R learning really work on a task domain that has not been explicitly designed to discriminate long-term optimal behavior from short-term optimal behavior. In other words, if we applied R learning to a task domain that had previously been studied with Q learning, how well would it do? That is the goal of our paper.

Empirical testing of R learning assumes even more importance because unlike the case with Q learning, there is no proof that R learning will converge to the optimal policy. In comparing two RL techniques, the fundamental problem is that in contrast to inductive learning techniques, where there exist accepted evaluation methods such as cross-validation [4], there are no corresponding schemes for RL. One unique characteristic of RL is that the learner has to select actions stochastically to tradeoff exploration of the environment vs. exploitation of the currently learned policy. Stochastic action selection means that two successive runs of an RL technique may produce very different performance improvement curves. To ensure reliable results, the techniques must be run a number of times on the same domain with the same parameter settings. An *average* performance improvement curve must then be extracted from these multiple runs to compare two RL techniques. Another problem is that the performance of an RL technique may vary with type of exploration strategy used.

Accordingly, in this paper we compare the average performance of R learning and Q learning over multiple runs using two types of undirected exploration strategies [19], semi-uniform exploration and Boltzmann exploration, and one recency-based directed exploration method [15]. Computing averaged performance improvement plots does not reveal the actual variation across runs. For this reason, we will show the variation across runs in this paper. Also, we deviate from the usual tendency of estimating averages using the mean [9]. The *median* is a better distribution-free estimator of population average [11], and we introduce a simple non-parametric significance test for comparing two median-based performance curves.

We show there are significant differences between R learning and Q learning in a robot box pushing task. We used this domain previously in a detailed study of Q learning [10]. Accordingly, we were curious to see how an undiscounted RL method like R learning would work on this domain. We find somewhat surprisingly that R learning performs worse than Q learning, even when both are evaluated using an undiscounted performance measure. Furthermore, Q learning also is more robust than R learning. The latter's performance is very sensitive to choice of exploration method. For example, R learning deteriorates when used with the popular Boltzmann exploration method [9, 14] because of a limit cycle condition, which we precisely identify. However, R learning performs much better with semi-uniform undirected and recency-based directed exploration methods.

2 Two Reinforcement Learning Methods

We begin by briefly reviewing Q learning [20] and R learning [12]. In general, reinforcement learning is based on the framework of Markov decision problems (MDP) [8, 16]. These consist of a set of states S, a set of actions A, and a reward function mapping states (and possibly actions) to real numbers. The goal of learning is to determine a policy $\Pi : S \rightarrow A$ that maximizes a performance measure. In discounted reinforcement learning, the performance measure being optimized is [3]:

$$J^{\Pi}(x_0) = \lim_{n \rightarrow \infty} E(\sum_{t=0}^{n-1} \gamma^t R^{\Pi}_{x_t,a})$$

where $\gamma < 1$ is the discount factor, and $R^{\Pi}_{x_t,a}$ is the reward received for doing action a in state x_t where actions are chosen using policy Π. E denotes expected value. Undiscounted reinforcement learning, on the other hand, optimizes a different performance measure based on average reward, namely [3]:

$$J^{\Pi}(x_0) = \lim_{n \rightarrow \infty} \frac{1}{n} E(\sum_{t=0}^{n-1} R^{\Pi}_{x_t,a})$$

Note that here the rewards are not discounted, and instead the cumulative rewards are being averaged.

2.1 Q Learning

Q learning [20] is based on inferring a utility function $Q(x,a)$ over states x and actions a. Initially, all the $Q(x,a)$ values are set to 0. At each step, the learner chooses the action a with the maximum $Q(x,a)$ value. However, occasionally the learner will need to explore suboptimal actions to ensure that it does not get stuck in a local maximum. After every action, the $Q(x,a)$ values are updated using the following rule

$$Q_{new}(x,a) \leftarrow Q_{old}(x,a) + \beta(r + \gamma e_Q(y) - Q_{old}(x,a))$$

where r is the reward received for doing action a. $e_Q(y)$ is the utility of state y (defined as the maximum $Q(y,a)$ value over all actions a), γ is a discount factor (less than 1) that is used to ensure finiteness of the Q values, and β is the learning rate.

2.2 R Learning

Since Q learning discounts future rewards, it prefers actions that result in short-term perhaps mediocre reward to those that result in long-term sustained rewards. Recently, Schwartz [12] proposed a technique called R learning which maximizes the *average* reward per step. Schwartz argued that most previous experimental work on Q learning, including ours [10], measure the increase in average reward versus learning time to determine the effectiveness of learning. However, Q learning does not maximize average reward, but discounted cumulative reward. Thus, R learning should in fact produce better results than Q learning.

R learning is similar to Q learning, in that it is based on inferring $R(x,a)$ utility values for choosing an action a in state x. Initially all the R values are set to 0. At each situation, the learner chooses the action that has the highest R value, except that sometimes it chooses an exploratory non-optimal action. R values are adjusted after each action, based on the following learning rule:

$$R_{new}(x,a) \leftarrow R_{old}(x,a) + \beta(r - \rho + e_R(y) - R_{old}(x,a))$$

which differs from the rule for Q learning given earlier, by subtracting the average reward ρ from the immediate reward r, and by not discounting the next state utility $e_R(y)$ (which is defined as the maximum $R(y,a)$

value over all actions a). The average reward ρ is estimated as

$$\rho_{new} \leftarrow \rho_{old} + \alpha(r + e_R(y) - e_R(x) - \rho_{old})$$

A key point here is that ρ is updated only when a non-random action a is performed, that is when $e_R(x) = R(x, a)$. One attractive aspect about the average reward ρ is that it does not depend on any particular state, and is constant over the entire state space [3].

3 A Simulated Robot Box-Pushing Task

To compare R learning and Q learning, we need to choose some task domain that we can run both techniques on to collect experimental data. The domain we use is a simulated robot box-pushing domain which we used extensively in our previous work on Q learning [10]. This simulator is shown in Figure 1.

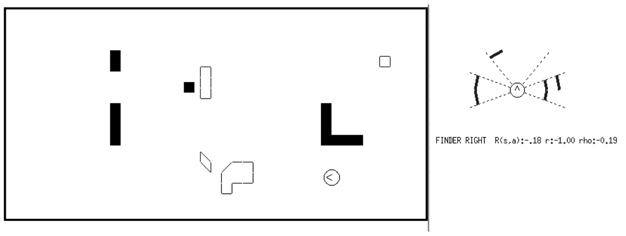


Figure 1: A simulated robot environment

The simulator depicts a room environment modeled after the actual environment in which a robot moves about. The robot is represented by the circular figure; the "nose" indicates the orientation of the robot. Dark shaded objects represent obstacles. Outline figures represent boxes. The robot uses simulated "sonar" sensors that are patterned after real sonar sensors. These indicate the presence or absence of objects at near and far distances from the robot.

Figure 1 shows the sonar pattern seen by the robot at the location depicted in the simulator. There are totally 16 sonar bits, one "near" bit and one "far" bit in each of 8 radial directions. In this paper we reduced the sonar data to 10 bits. The reason for compressing sonar data is to avoid the issue of *function approximation* from complicating this study (we addressed the function approximation issue in our previous work [10]). The 16 sonar bits were reduced to 10 bits by disjoining adjacent near and far bits. There are also 2 bits of non-sonar information, BUMP and STUCK, which indicate whether the robot is touching something and whether it is wedged. Thus, there are totally 12 state bits, and 4096 possible states.

The robot can move about the simulator environment by taking five possible actions: going forward or turning left and right by two varying degrees. Instead of learning the whole task in a monolithic fashion, following our previous work the robot learns the task by decomposing it into three parts: learning to find a box, push a box, and unwedge from a stalled situation. This decomposition is given to the robot and is not learned, although techniques have been developed for automatically producing sequential decompositions [13]. Three separate reward functions are used, one for each behavior. For finding, the robot is rewarded for approaching objects, that is it is rewarded by $+3$ when it goes forward and turns on the front near sonar bits. It is punished by -1 when these bits are off. For pushing, the robot is rewarded by $+1$ if it went forward and remained in contact with an object. It is punished by -3 for losing contact. Finally, for unwedging, the robot is rewarded by $+3$ if it went forward and did not get stuck. It is punished by -1 for being stuck. In all three reward functions, the default reward is 0.

The simulator is a simplification of the real world situation (which we studied previously [10]) in several important respects. First, boxes in the simulator can only move translationally. Thus a box will move without rotation even if a robot pushes the box with a force which is not aligned with the axis of symmetry. Second, the sonars on the real robot are prone to various types of noise, whereas the sensors on a simulator robot are "clean". Finally, actions are deterministic (in the sense that taking the same action twice from the same state will produce identical results) on the simulator, unlike on the real robot. Even though this simulator is a fairly inaccurate model of reality, it is sufficiently complex to illustrate the issues raised in this paper.

4 Experimental Design

Similar to most previous work in RL [7, 9, 13], we used the following experimental design format. The overall learning run is broken down into trials. For our task domain, we will set a trial as consisting of 100 steps. After 100 steps, the environment and the robot are reset, except that at every trial the robot is placed at a random unoccupied location in the environment facing a random orientation. The location of boxes and obstacles does not vary across trials. A learning run lasts for 300 trials. We carried out 10 runs to get average performance estimates.

For each training run, we measured the percentage of steps (computed cumulatively from the beginning of each run) that the robot was pushing a box at the end of each trial. This metric gives an overall estimate of task performance, and corresponds to a reward function which only gives payoffs (of $+1$) when the robot actually pushed a box. We also measured how well the robot learned each separate behavior, which turned out to be very useful in understanding differences between the two RL techniques.

4.1 Different Types of Exploration Strategies

Following Thrun's terminology [19], we divide exploration methods into *undirected* and *directed* methods. Undirected exploration methods do not use the results of learning to guide exploration; they merely select a random action some of the time. Two popular undirected exploration methods are Boltzmann exploration, which we study in Section 5, and semi-uniform exploration, which we study in Section 6. Directed exploration methods use the results of learning to decide where to concentrate the exploration efforts. A well known directed method is *interval estimation* [7], which estimates confidence intervals on the probability of getting high reward using an action. We will use a simpler recency-based exploration method described in Sutton's DYNA paper [15]. We compare Q learning and R learning using recency-based exploration in Section 7.

5 Experiment 1: Boltzmann Undirected Exploration

We begin by comparing R learning and Q learning using the Boltzmann undirected exploration method. We set the parameter $\gamma = 0.9$, $\beta = 0.2$, and $\alpha = 0.01$. The Boltzmann exploration function [9, 15] assigns the probability of doing an action a in state x as

$$p(x,a) = \frac{e^{\frac{U(x,a)}{T}}}{\sum_a e^{\frac{U(x,a)}{T}}}$$

where T is a "temperature" parameter that controls the degree of randomness. $U(x,a)$ is the utility of doing action a in state x and can be either $Q(x,a)$ or $R(x,a)$, respectively. In this experiment, the temperature T was initialized at 30, and gradually decayed to 0.5 using a decaying scheme similar to that described in [1].

Instead of simply showing the average curve, it is instructive to also view the variation over different runs. This data is shown in Figure 2, where an "errorbar" is used to display the maximum and minimum values along with the median value at the end of each trial over the 10 runs. We deviate from the typical practice in RL in *not* using the mean performance [9] since the mean is a poor estimate of the average if the distribution is not normal, or if the number of runs is small. We have found that median averaged curves often look substantially different from mean averaged curves. Mean-averaged curves look smoother, but as both Q and R learning exhibit considerable variation over runs, isolated low or high extremum values disproportionately distort the mean.

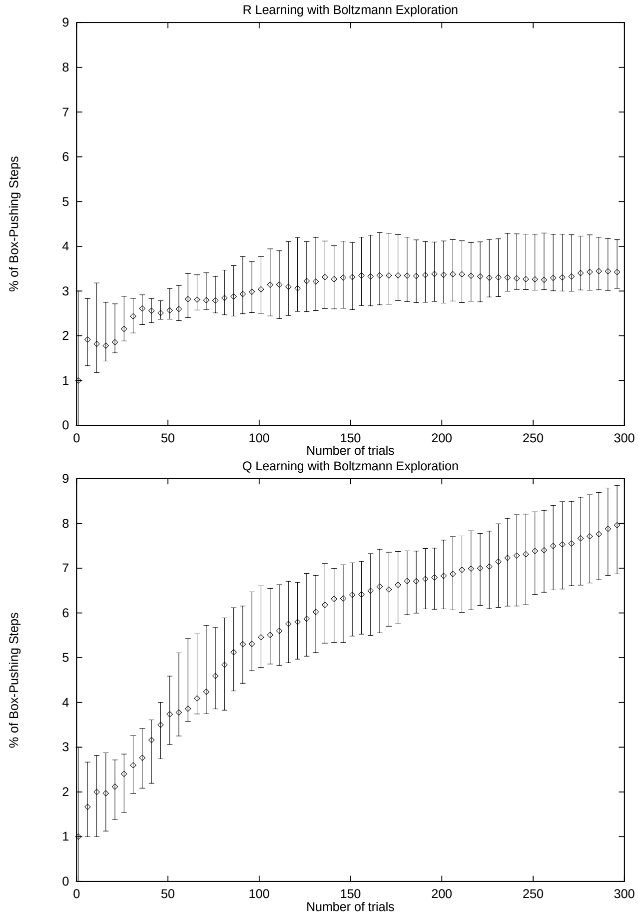


Figure 2: Median-averaged performance over ten independent runs of R and Q learning with Boltzmann exploration. The "errorbars" show the median values along with the highest and lowest value.

We notice first from Figure 2 that Q learning outperforms R learning by more than a factor of two in this experiment. Furthermore, Q learning's performance shows no signs of declining, whereas R learning is leveling off. To judge the robustness of this result, we decided to vary some of the initial parameters. We tried a higher value for the learning rate parameter $\alpha = 0.5$. As Figure 3 shows, there is some slight improvement, but it is not dramatic. We also tried a much lower initial temperature $T = 5$. Figure 3 shows the results in this case, and R learning's performance has actually worsened.

One clue to understanding why Q learning is performing so much better than R learning is look at how well the three individual behaviors were learned by both of them. As Figure 4 shows, Q learning is able to learn the unwedging behavior far more successfully than R learning. This discrepancy explains why the overall performance of R learning is so mediocre: the robot is not able to unwedge itself from stalled states.

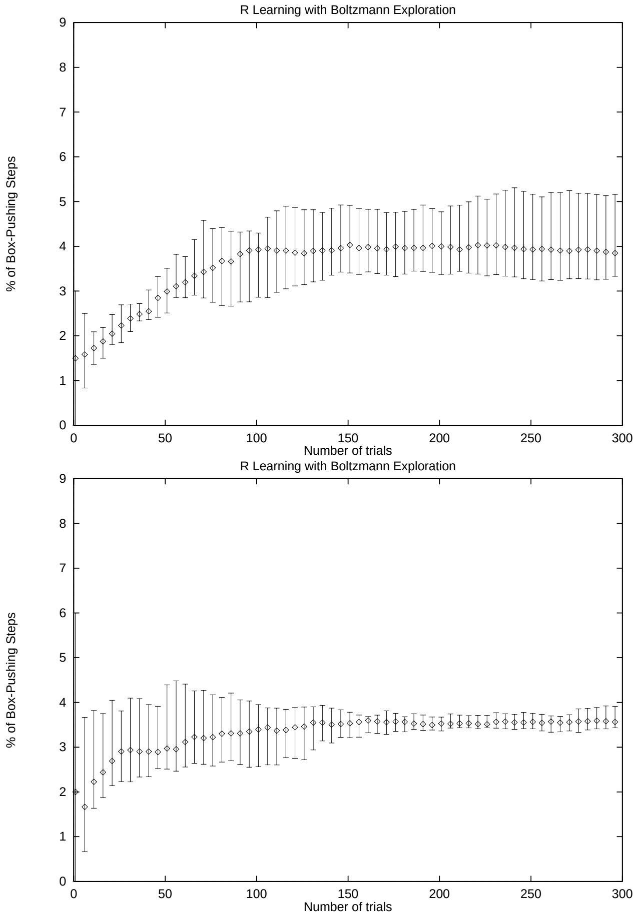


Figure 3: The graph on the top shows the performance of R learning with a high learning rate $\alpha = 0.5$. The graph on the bottom shows the performance of R learning with low initial temperature $T = 5$.

5.1 A Limit Cycle Situation in R Learning

However, this explanation raises an even more puzzling question: why does R learning not learn the unwedging behavior? From our past work, we have found that it is usually the easiest behavior to learn! Upon further analysis, we found that under a certain condition, R learning when combined with Boltzmann exploration can exhibit cyclical behavior (which we can think of as a limit cycle). Figure 5 illustrates this problem in its simplest form (actually this very situation occurs in our robot domain).

Consider the situation shown in the figure. In state 1, the best action is to go right (marked r), and in situation 2, the best action is to left (marked l). Furthermore, the utilities of these actions happen to be equal. That is,

$$R(1, r) = R(2, l) = e_R(1) = e_R(2)$$

Finally, the immediate reward received by the robot

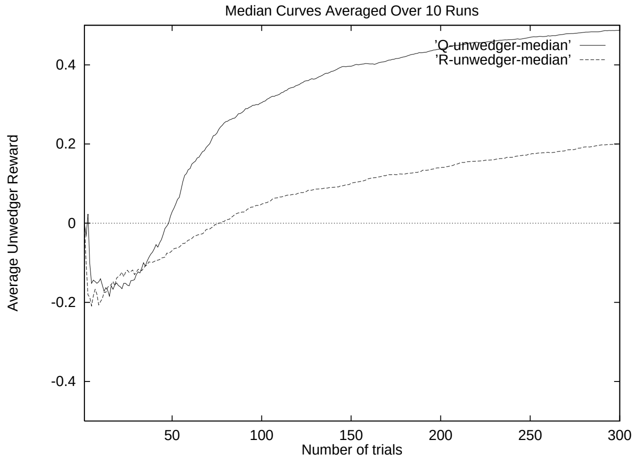


Figure 4: This graph compares the performance of Q and R learning using the unwedging behavior (the Q curve is above the R curve).

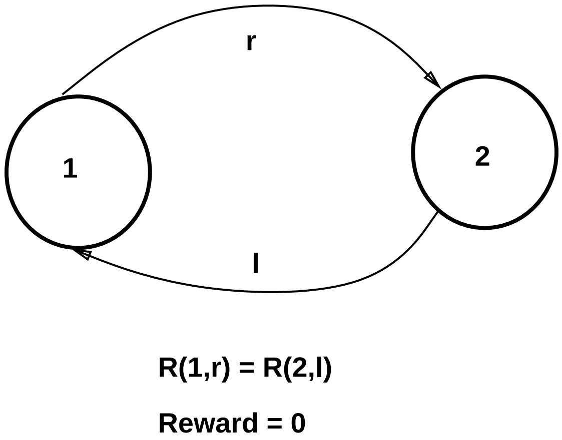


Figure 5: A situation where R learning gets into a limit cycle.

for going left or right is 0. Under these conditions, the update equations for R learning can be simplified to yield

$$R(1, r) \leftarrow R(1, r) - \beta\rho$$

$$R(2, l) \leftarrow R(2, l) - \beta\rho$$

To simplify the argument, we are assuming synchronous updating, but the results hold even for asynchronous updating. Thus, $R(1, r)$ and $R(2, l)$ will decay over time. However, the average reward ρ is itself decaying since under these conditions, the average reward update equation turns out to be

$$\rho \leftarrow (1 - \alpha)\rho$$

When ρ decays to zero (or below the floating point precision), the utilities $R(1,r)$ and $R(2,l)$ will stop decaying. The relative decay rate will of course depend on β and α, but one can easily show cases where the R values will decay much slower than ρ. For example, given the values $R(1,l) = 3.0$, $\rho = 0.5$, $\alpha = 0.1$, $\beta = 0.2$, after 1000 iterations, $R(1,l) = 2$, but $\rho < 10^{-46}$! Since the R values are not changing when ρ has decayed to zero, and the robot is selecting actions using the Boltzmann exploration, when the temperature is sufficiently low (so that the R values are primarily used to select actions), the robot will simply oscillate between going left and going right. We witnessed this situation occurring frequently in our simulations when the robot tried to learn unwedging. Eventually, it does get out of the situation, but much learning effort has been wasted that even after 30,000 steps (300 trials of 100 steps), it has still not learned unwedging.

Interestingly, Q learning will not get into the same limit cycle problem illustrated above because the Q values will always decay by a fixed amount. Under the same conditions as above, the update equation for Q learning can be written as

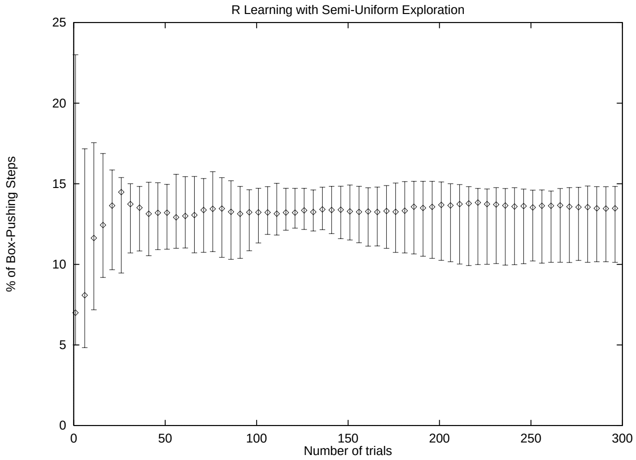
$$Q(1,l) \leftarrow Q(1,l)(1-\beta(1-\gamma))$$

$$Q(2,r) \leftarrow Q(2,r)(1-\beta(1-\gamma))$$

Now the two utilities $Q(1,r)$ and $Q(2,r)$ will continue to decay until at some point another action will be selected because its Q value will be higher. Clearly, the empirical results confirm this reasoning because Q learning does not suffer from the problem of being unable to learn unwedging. As we show below, R learning's performance greatly improves when used with the semi-uniform undirected exploration strategy, as well as with the recency based directed exploration strategy.

6 Experiment 2: Semi-Uniform Undirected Exploration

We now compare R and Q learning using a semi-uniform undirected exploration method [19]. Basically, in this method the robot executes random actions with a fixed probability. We obtained good results using this exploration strategy in our previous work [10]. In the experiment below, we set the probability of doing a random action at 10%. As Figure 6 shows, both R learning and Q learning perform much better with the semi-uniform exploration strategy. This result also confirms Thrun's work [19] where Boltzmann exploration did worse than semi-uniform exploration.



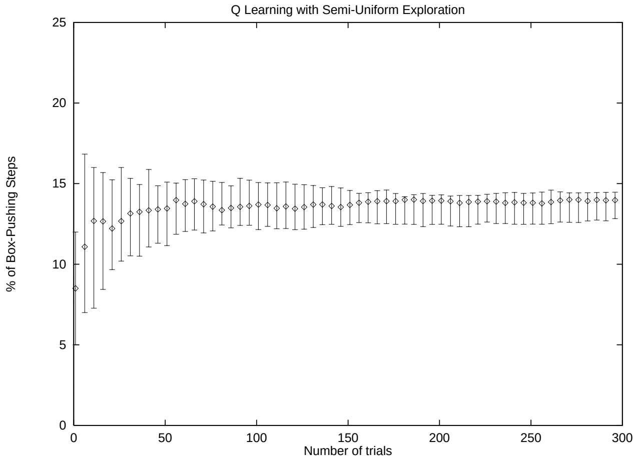


Figure 6: Median averaged performance over ten independent runs of R and Q learning using semi-uniform undirected exploration

6.1 A Non-Parametric Significance Test

Given two averaged performance improvement curves, such as in Figure 6, we need to decide if the difference between them is statistically significant. One commonly adopted test for statistical significance is the *T-test*, but it is inappropriate if the underlying distribution is not normal. A more suitable test, particularly when using medians to compute average performance curves, is the *Wilcoxon matched pair signed-rank* test [11]. Unlike the T-test, the Wilcoxon test is a distribution-free test. We give a brief description of the Wilcoxon test below (further details can be found in any statistical handbook [11]).

Given two supposedly different matched pair data samples (say, the average performance improvement produced by Q learning and R learning on identical length runs), the Wilcoxon test decides when to reject the null hypothesis that they are not different by estimating the probability that the differences between corresponding values in the two datasets are symmetrically distributed with respect to the median equal to 0. Given n data points for each sample, the pairwise differences between them (that is, $d_i^R - d_i^Q$ for each of the i data points for R and Q learning) are ranked according to their absolute values (zero differences are discarded). Identical difference values are assigned the mean rank (for example, if the first and second differences are equal, their rank is 1.5). Let the sum of the ranks produced by positive differences be S_p, and that produced by negative differences be S_n. The smaller of the two ranks S_p and S_n is used as a test statistic S. The null hypothesis (that the two samples are not statistically different) is discarded if the test statistic S is less than or equal to the critical value $S(n, \omega)$, which can be either obtained from a standard table or approximated (for $n > 25$) by the equation

$$S(n, \omega) = \frac{(n)(n+1)}{4} - z_\omega \sqrt{\frac{(n)(n+1)(2n+1)}{24}}$$

Here ω is the probability of wrongly rejecting the null hypothesis, so if we wanted 95% statistical confidence, $\omega = 0.05$. For a standard one-sided test, the value z_ω above represents the area of a normal distribution that lies to the right of it.

We can use the Wilcoxon test for testing the hypothesis that the difference between the two median curves shown in Figure 6 is statistically significant. Given an $\omega = 0.05$, and given $n = 300$ (since we are plotting the task metric at the end of each trial), we get

$$S = 4962; S(300, 0.05) = 20101.3$$

thereby showing the difference to be statistically significant. To summarize, we have argued that the median is a better estimator than the mean of the average performance across multiple runs of an RL technique. We have also illustrated the use of a non-parametric test called the Wilcoxon test for showing the statistical significance of a difference between two average performance improvement curves. We believe this approach can be applied to any comparative study of RL techniques.

7 Experiment 3: Recency-based Directed Exploration

The final experiment is to compare R with Q learning using a directed exploration method. Thrun [19] found that directed exploration methods were markedly superior to undirected methods using the AHC technique [2] on a simulated robot navigation task. In this section we test if this result also holds for Q learning and R learning in the robot box-pushing domain.

We implemented the recency-based exploration method described in Sutton's DYNA paper [15]. In this approach, a count is maintained of the number of actions elapsed since each state action pair was experienced. When selecting an action, a fraction of this count is added to the utility of a state action pair. In particular, the utility of doing an action is defined as

$$U(x, a) \leftarrow Q(x, a) + c\sqrt{(count(x, a))}$$

where c is a small positive number.

The results are shown in Figure 7. Q learning is once again performing better than R learning, although the latter's performance is much better than with Boltzmann exploration and a little better than its performance with semi-uniform undirected exploration. Using the Wilcoxon test once again, we can verify that the differences between the two median curves are statistically significant at the 95% level. The precise numbers are

$$S = 1294; S(300, 0.05) = 20101.3$$

which shows the difference is statistically significant.

8 Summary of Results and Discussion

The results of our experiments are summarized in Table 1. The table compares the average number of box pushing steps (in percentages) for R learning and Q learning across three different exploration strategies. The numbers are median averaged over 10 runs of 300 trials, each trial lasting 100 steps.

Method	Boltzmann	Semi-Uniform	Recency
R	3.4%	13.5%	16.7%
Q	8%	14.0%	17.3%

Table 1: Average performance over 10 runs measured in terms of median box-pushing percentages.

We have carried out several additional experiments for different values of the various parameters, such as the learning rate α, with similar outcomes. Although futher experimentation is required (which we discuss below), the data so far is quite striking. What our results show is Q learning is consistently better than R learning in our box pushing task. Unlike the simple illustrative examples in Schwartz's paper, the box-pushing domain is a fairly complex task. It is surprising that Q learning turns out to be superior to R learning when it is evaluated using an undiscounted performance metric, which should favor R learning over Q learning. It also casts some doubt on Schwartz's conjecture [12] that R learning is superior to Q learning even when discounting is the right thing to do. Furthermore, R learning seems to be quite sensitive to the choice of exploration strategy. Its lackluster performance with Boltzmann exploration is perhaps the

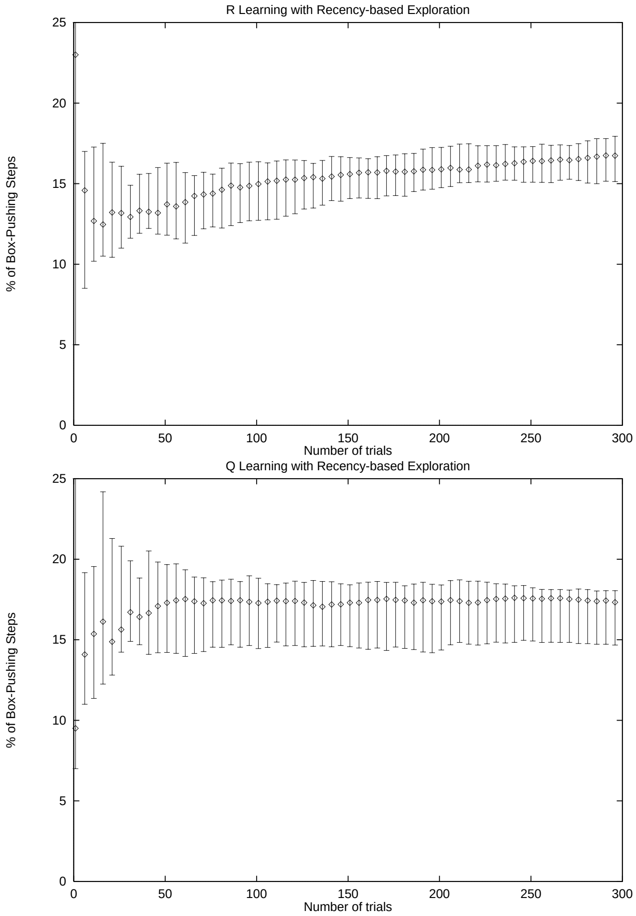


Figure 7: Median-averaged performance of R and Q learning over ten independent runs using recency-based directed exploration.

most surprising result in this paper. We attribute this poor performance to limit cycles as explained above. Finally, it is clear from our experiments that the performance of an RL technique is very dependent on the type of exploration method used. Both techniques performed their best with directed exploration methods, and more poorly with undirected methods with Boltzmann being much worse than semi-uniform.

9 Limitations of Our Study

The experimental results reported in this paper suffer from some limitations. The box-pushing domain, being deterministic, may not be an ideal domain to study R learning. We plan to study stochastic variants of this domain in the near future. The overall task metric we used, the percentage of box-pushing steps, is related only indirectly to the individual reward functions for the various behaviors. However, we have found that direct comparisons between the reward functions themselves (such as in Figure 4) typically show Q learning to be superior to R learning. The limit cycle analysis we made is not sufficiently rigorous, and we plan to study this in more depth. We also need to conduct longer trials and higher number of runs. Finally, our experiments do not directly reveal how to improve R learning, which would be a logical next step to explore.

10 Future Work

Obviously, further experimentation will be necessary to shore up some of the conclusions of this study. Briefly, some of the directions for future work are:

- *Different domains:* We plan to test additional task domains to see if similar results arise in them. Recently, Tadepalli and Ok [17] have found that R learning performs better than Q learning in an automated guided vehicle (AGV) domain using semi-uniform exploration, if the task parameters are fixed such that long term optimal behavior does not coincide with short term optimal behavior. One immediate problem would be to see how well R learning performs in the AGV domain with Boltzmann and directed exploration. Another direction to pursue is to compare R and Q learning for some variation of the box-pushing task where optimal long-term behavior demonstrably differs from optimal short-term behavior.
- *Other Average Reward Methods:* Another direction to purse is to study other average reward learning algorithms from the engineering literature. For example, Jalali and Ferguson [6] describe a model-based average reward learning method. The aforementioned paper by Tadepalli and Ok contains some initial results comparing this method with R learning in the AGV domain. We plan to study this method in our box pushing domain also.
- *Function approximation:* We have avoided the issue of function approximation in this paper altogether, and instead assumed that the policy can be implemented by a state action table. We chose to ignore function approximation deliberately to avoid confusion, but we hope to undertake further comparative tests of R learning and Q learning using neural-net based directed exploration strategies such as developed by Thrun [19].

11 Acknowledgements

I thank Alka Indurkhya for advising me on the use of appropriate statistical tests. Prasad Tadepalli illustrated the need for using a matched pair test, and provided useful comments on earlier drafts of this paper. Rich Sutton's incisive feedback on an earlier draft of the paper helped clarify the interpretation of the empirical results. Lastly, I thank Kevin Bowyer and Larry Hall for allowing me generous use of the Sparc 10's in their labs for running long simulations.

References

[1] A. Barto, S. Bradtke, and S. Singh. Learning to act using real-time dynamic programming. Technical Report CMPSCI TR 93-02, Univ. of Massachusetts, Amherst, 1993.

[2] A. Barto, R. Sutton, and C. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5):834–846, 1983.

[3] D. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, 1987.

[4] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Chapman and Hall, 1984.

[5] J. Connell and S. Mahadevan. *Robot Learning*. Kluwer Academic Publishers, 1993.

[6] A. Jalali and M. Ferguson. Computationally efficient adaptive control algorithms for Markov chains. In *Proceedings of the 28th Conference on Decision and Control*, pages 1283–1288, 1989.

[7] L. Kaelbling. *Learning in Embedded Systems*. PhD thesis, Stanford University., 1990.

[8] J. Kemeny and J. Snell. *Finite Markov Chains*. Van Nostrand, 1960.

[9] L. Lin. *Reinforcement Learning for Robots using Neural Networks*. PhD thesis, Carnegie-Mellon Univ, 1993.

[10] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55:311–365, 1992. Appeared originally as IBM TR RC16359, Dec 1990.

[11] L. Sachs. *Applied Statistics: A Handbook of Techniques*. Springer-Verlag, 1982.

[12] A. Schwartz. A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 298–305. Morgan Kaufmann, 1993.

[13] S. Singh. *Learning to Solve Markovian Decision Processes*. PhD thesis, Univ of Massachusetts, Amherst, 1994.

[14] R. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.

[15] R. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224. Morgan Kaufmann, 1990.

[16] R. Sutton, editor. *Reinforcement Learning*. Kluwer Academic Press, 1992. Special Issue of Machine Learning Journal Vol 8, Nos 3-4, May 1992.

[17] P. Tadepalli and D. Ok. H learning: A reinforcement learning method to optimize undiscounted average reward. Technical Report 94-30-01, Oregon State Univ., 1994.

[18] G. Tesauro. Practical issues in temporal difference learning. In R. Sutton, editor, *Reinforcement Learning*. Kluwer Academic Publishers, 1992.

[19] S. Thrun. The role of exploration in learning control. In D. A. White and D. A. Sofge, editors, *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*. Van Nostrand Reinhold. To Appear.

[20] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, 1989.