To appear in *Machine Learning: Proceedings of the Tenth International Conference*,
Morgan Kaufmann, San Mateo, CA, 1993.

A Reinforcement Learning Method for Maximizing Undiscounted Rewards

Anton Schwartz
Computer Science Dept.
Stanford University
Stanford, CA 94305
schwartz@cs.stanford.edu

Abstract

While most Reinforcement Learning work utilizes temporal discounting to evaluate performance, the reasons for this are unclear. Is it out of desire or necessity? We argue that it is not out of desire, and seek to dispel the notion that temporal discounting is necessary by proposing a framework for undiscounted optimization. We present a metric of undiscounted performance and an algorithm for finding action policies that maximize that measure. The technique, which we call R-learning, is modelled after the popular Q-learning algorithm [17]. Initial experimental results are presented which attest to a great improvement over Q-learning in some simple cases.

1 Introduction

In the paradigm of Reinforcement Learning (RL), an agent finds itself in an environment and must learn by trial and error to take actions so as to maximize, over the long run, rewards which it receives in return. The techniques for accomplishing this estimate the rewards that performing an action may reap over the course of time, so as to choose actions maximizing that measure. This requires summarizing a possibly infinite sequence of rewards in a finite measure. To do this, researchers have relied on *temporal discounting*, a practice of giving exponentially diminishing importance to rewards far in the future, so as to summarize the rewards in a single finite number.

While temporal discounting is convenient, it comes at a price. For one, it may make behaviors with quick but mediocre results look more attractive than efficient behaviors reaping long-term benefits. Moreover, even when it favors behaviors with adequate farsightedness, it can greatly impede the process by which Q-learning arrives at its solutions.

In this paper we outline some of the problems caused by temporal discounting, and set out to provide a workable alternative. We put forth an undiscounted measure of policy performance, and present an algorithm designed to arrive at policies which maximize that measure. The method, which we call R-learning, resembles Q-learning inasmuch as it measures the value—for a different notion of value—of state-action pairs, relative to its current policy, and in any state recommends that action which maximizes value. The key element introduced is a measure of average reward, which serves as a standard of comparison. When examined according to a measure of utility that is depends on this average, improvements to policies become more salient. This boosts performance in a number of ways.

While all of the principle ideas and techniques presented in this paper (including the formal notions of average and average-adjusted value, Theorem 3, and the R-learning algorithm) are the result of our own work, we became aware while writing that the ideas were first created and explored in the Dynamic Programming literature, often with much greater generality and elegance. As a result, this paper may serve as an introduction to the concepts of undiscounted Dynamic Programming, and as the first attempt to bring those concepts to bear on the problems and practices of Reinforcement Learning. The R-learning algorithm remains, to our knowledge, a novel technical contribution.

A more qualitative account of the ideas presented in this paper is given in [13].

2 Background

In RL, the learner's environment is modelled as a Markov Decision Process (MDP). An MDP specifies a set S of states and a set A of actions. At each step in time the process is in some state, and an action must be chosen; this action has the effect of changing the current state and producing a scalar reinforcement value. The reinforcement value, or reward, represents the extent to which we can consider the action to have had immediately desirable or undesirable consequences. Formally, an MDP is described by a 4-tuple $\langle S, A, P, r\rangle$ where $P : S \times S \times A \mapsto [0, 1]$ gives the probability (written $P_{ss'}(a)$) of moving into state s' when action a is performed in state s, and $r : S \times A \mapsto \mathbb{R}$ gives the corresponding expected reward. In this paper we will deal exclusively with finite MDP's, *i.e.*, ones with finite sets S and A. A *policy* is a mapping from S to A, suggesting which action to perform in each state.[1] The goal of RL methods is to arrive, by performing actions and observing their outcomes, at a policy which maximizes the rewards accumulated over time.

Q-learning [17] is the most widely used and studied method for RL and, like most, it uses discounted value as its criterion of optimality. The *discounted value*, or *discounted return*, of a policy π in a state s is defined as the expected value

$$V_\gamma^\pi(s) \stackrel{def}{=} E\left[\sum_{t=0}^{\infty} \gamma^t r_{s,t}^\pi\right] \tag{1}$$

where $r_{s,t}^\pi$, a random variable, represents the reward that will be received t time steps after the learner begins executing policy π in state s, and γ is a temporal discounting constant, $0 \le \gamma < 1$.

Q-learning seeks to find a policy π^* which maximizes $V_\gamma^\pi(s)$ in all states $s \in S$. (We call such policies γ-*optimal*.) To do so, it makes particular use of the *action-value* form of (1),

$$Q_\gamma^\pi(s,a) \stackrel{def}{=} V_\gamma^{a;\pi}(s), \tag{2}$$

in which $a;\pi$ denotes the nonstationary policy which performs action a once and thereafter follows policy π. The algorithm for Q-learning is based on the relation

$$Q_\gamma^\pi(s,a) = r(s,a) + \gamma \sum_{s' \in S} P_{ss'}(a) V_\gamma^\pi(s'), \tag{3}$$

and on the fact that for γ-optimal policies π^* (and no others),

$$V_\gamma^{\pi^*}(s) = \max_{a \in A} Q_\gamma^{\pi^*}(s,a) \quad \text{for all } s \in S. \tag{4}$$

Q-learning operates by maintaining a function $\hat{Q} : S \times A \mapsto \mathbb{R}$, which initially maps all values to zero, and is updated every time an action is executed as follows. Let $x \stackrel{\beta}{\leftarrow} y$, for $0 \le \beta \le 1$, denote the operation of assigning to variable x the value $x + \beta \cdot (y - x)$. If action a is performed in state s, resulting in an immediate reward r_{imm} and a transition into state s', then the value of $\hat{Q}$ on input $\langle s, a \rangle$ is modified by performing

$$\hat{Q}(s,a) \stackrel{\beta}{\leftarrow} r_{imm} + \gamma \max_{a' \in A} \hat{Q}(s', a') \tag{5}$$

for an appropriate learning rate β. At any time, the Q-values induce a policy $\pi^{\hat{Q}}$ which maps each state s to the action a maximizing $\hat{Q}(s,a)$. Given certain assumptions, this procedure is guaranteed to arrive at a policy which maximizes discounted value from every state [18].

In cases where the value $V_1^{\pi^*}(s)$ exists and is finite (*cf.* Equation 1), V_1 can be used as an undiscounted measure of performance, and Q-learning with $\gamma = 1$ can be shown to converge to V_1-optimal policies [18]. But the MDP's of which the above stipulation holds are *goal tasks* in which one seeks to reach an absorbing set of "goal" states via a minimal path; and in those domains researchers tend to use discounted Q-learning for other reasons[2] (*e.g.*, [8, 16]).

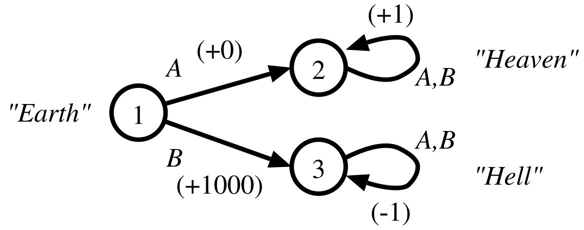

Figure 1: A trivial MDP on which discounted and undiscounted measures may disagree. States are indicated by circles, actions are given in italics, their associated state transitions are given by arrows, and their immediate rewards are given in parentheses.

3 Measures of Performance

If we look at the graphs which papers in RL use to report the performance of their systems, we find that they are almost universally graphs of undiscounted measures: either total cumulative reward or average reward per time step (*e.g.*, [6, 8, 9, 16]). But Q-Learning maximizes a future-discounted measure of reward instead. The problem is that that these criteria, in general, need not coincide.

3.1 Discounted Performance

For instance, consider the MDP of Figure 1. Here, attention to (undiscounted) future rewards will clearly mandate that a policy choose action A in state 1. But for any $\gamma < \frac{500}{501} \approx 0.998$, $Q_\gamma^\pi(1,B) > Q_\gamma^\pi(1,A)$ regardless of π—which makes methods such as Q-learning prefer action B. In fact, given any γ, there is some value we can set for $r(1,B)$ which makes the γ-discounted criterion favor action B over action A.

It is true that for any finite MDP there is some sufficiently large γ for which the discounted and undiscounted measures agree. However, proper choice of such a γ requires detailed knowledge of the domain—knowledge that we do not want to presuppose. Even with such knowledge, a parameter such as γ that needs to be tailored to suit individual domains is clearly undesirable.

In light of these observations, one may wonder why researchers use discounting at all. We address several possible answers to this question:

Discounting to compensate for rate of interest. Discounted value is the primary criterion for performance in the field

[1] Except where otherwise noted, we use *policy* to refer to a stationary policy, *i.e.*, one which does not vary over time [12].

[2] These reasons, too lengthy to be presented in this paper, will be explained in detail elsewhere. They do not apply to the R-learning method.

of Operations Research, where dynamic programming has been used and studied extensively. There the motivation is economic: one assumes that interest is available on earned rewards at the rate of $(100\varphi)\%$ per unit of time. In order to compensate for the interest one can earn on rewards achieved in the present, a reward of r units t time steps into the future must be evaluated by its *present value*, $(1+\varphi)^{-t}r$. This gives motivation for using $\gamma = (1+\varphi)^{-1}$ in the above framework. However, this explanation can be dismissed out of hand by observing that RL researchers do not let interest accrue on rewards in their reportings of cumulative reward, nor do they evaluate the rewards in terms of their present value at any moment in time.

Discounting to express the finiteness of an agent's lifetime. Another possible reason to value present rewards more than future rewards is the assumption that the agent might die sometime before it is able to reap future rewards. In this case, the γ represents an assumption that the agent has a probability $(1-\gamma)$ of dying at any step in time, in which case all future rewards will be zero. A different intuition with the same mathematical interpretation is this: *Discounting to express uncertainty about the future.* Some have argued that future rewards may be uncertain because of a changing environment [1], and use discounting to reflect that any reward expected at future time t may turn out to be zero instead with a probability of $(1-\gamma^t)$. But both of these interpretations of discounting beg the following question: If researchers assume that agents may die or that rewards may turn to naught, why do they measure performance in domains where neither of these eventualities come to pass?

This paper will proceed on the premise that the reason why people use discounting is none of these but a more practical one: *By discounting future rewards, one makes their infinite sum finite.* Which is to say, researchers use a discounted value measure because there exists no workable alternative. We proceed to provide such an alternative.

3.2 Undiscounted Performance

We wish to compare total undiscounted rewards reaped by a policy from a state; but since policies are likely to accrue rewards steadily over time, it is generally impossible to merely compare the infinite sums of rewards. Naturally, we turn to some comparison of the finite cumulative sums. Define the *n-period value* of a policy as

$$V^{\pi}_{(n)}(s) \overset{def}{=} \sum_{t=0}^{n-1} E\left[r^{\pi}_{s,t}\right].$$

One way to obtain a finite measure is to look at the average reward per unit of time incurred by a policy over the long run. We define the *average reward* of a policy π started in state s as

$$\rho^{\pi}(s) \overset{def}{=} \lim_{n\to\infty} \frac{V^{\pi}_{(n)}(s)}{n}.$$

While a policy needs to maximize average reward in order for us to consider it optimal in an undiscounted sense, the converse may not be true. For example, in some state s, two policies π_1 and π_2 may yield the same average reward even though π_1 will constantly outperform π_2 in that $V^{\pi_1}_{(n)}(s) = V^{\pi_2}_{(n)}(s) + c \ \ (c>0)$ for large n. In such cases we would like to have an additional measure which is sensitive to such constants c. One possibility is

$$\lim_{n\to\infty} V^{\pi}_{(n)}(s) - n\cdot\rho^{\pi}(s),$$

the limiting difference between cumulative performance and the line through the origin with slope $\rho^{\pi}(s)$. (We may think of this line as the cumulative performance of the hypothetical reference case in which π were to reap its average reward $\rho^{\pi}(s)$ at every step.) This measure, while intuitive, may not be well defined when the policy reaches periodic limit cycles. But in such cases we can generalize it to a measure which is guaranteed to exist. We define the *average-adjusted value* [5] of policy π in state s as the Cesàro or "limit in the average" [4] version of the simpler expression above:

$$\sigma^{\pi}(s) \overset{def}{=} \lim_{n\to\infty} \frac{\sum_{m=0}^{n-1} V^{\pi}_{(m)}(s) - m\cdot\rho^{\pi}(s)}{n}. \quad (6)$$

If we summarize the performance of a policy π in state s by a linear function in n approximating $V^{\pi}_{(n)}(s)$, then $\rho^{\pi}(s)$ is that line's slope and $\sigma^{\pi}(s)$ the y-intercept.

These two values, then, provide us with an undiscounted standard of performance. We wish foremost to maximize $\rho^{\pi}(s)$ and secondarily (ρ's being equal) to maximize $\sigma^{\pi}(s)$. So let us define the *total value* of a policy π from state s as the ordered pair

$$V^{\pi}_{T}(s) \overset{def}{=} \langle \rho^{\pi}(s), \sigma^{\pi}(s)\rangle$$

and use lexicographical order as the ordering on V_T. That is, we say that $\langle\rho,\sigma\rangle > \langle\rho',\sigma'\rangle$ if and only if $(\rho>\rho')$ or $(\rho=\rho'$ and $\sigma>\sigma')$.

We define undiscounted action-values analogously to the discounted case:

$$Q^{\pi}_{T}(s,a) \overset{def}{=} V^{a;\pi}_{T}(s).$$

To make parallels to the discounted case clear, we will use $V^{\pi}_{\sigma}(s)$ and $Q^{\pi}_{\sigma}(s,a)$ to refer to the σ components of $V^{\pi}_{T}(s)$ and $Q^{\pi}_{T}(s,a)$, respectively. Let us write $\pi_1 \succeq_T \pi_2$ to mean that $V^{\pi_1}_{T}(s) \geq V^{\pi_2}_{T}(s)$ for all states s. We say that π is *T-optimal* whenever $\pi \succeq_T \pi'$ for all policies π'. Like γ-optimal policies, stationary T-optimal policies are guaranteed to exist for finite MDP's [5]. We will henceforth use π^* to denote T-optimal policies.

The notion of T-optimality exists under a variety of names in the literature [11]. It is a stronger condition than average-reward optimality, but weaker than other forms including Blackwell optimality [3]. Blackwell optimality, the most selective one in the literature according to which all finite MDP's have optimal stationary policies, may be viewed as lexicographically maximizing an infinite series of which ρ and σ are the first two terms. So far we have not seen the need to utilize the higher order terms.

We briefly explore an important property of average reward:

Fact 1 *If two states s and s' are such that executing policy π from either state leads to the same ergodic set*[3] *of states in S, then $\rho^{\pi}(s) = \rho^{\pi}(s')$.*

This has the following consequence:

Corollary 2 *For any finite MDP, either*
1. $\rho^{\pi^*}(s)$ is independent of s, or
2. There are states between which no policy can ever guarantee passage.

The corollary tells us that in all cases of interest to RL, the optimal policy has a single average reward, since MDP's of which the second clause holds would normally violate the frequency of visit assumptions required for the convergence of stochastic approximation methods such as Q-learning [18].

While for many MDP's there exist policies that induce multiple ergodic sets, a simplification commonly made in the Dynamic Programming literature is that all policies are *unichain*, *i.e.*, give rise to a single ergodic set [11]. In light of Fact 1, this lets one assume that all policies, not only π^*, achieve a single state-independent average reward. At times we will make this assumption so as to facilitate the presentation of R-learning. A discussion of the behavior of R-learning in the general multichain case is forthcoming.

Attention to σ may seem frivolous, as σ represents merely a constant offset to cumulative value, likely to be quickly dominated by the repeated contributions of ρ. But for terminal goal tasks, average reward is merely a function of the goal state that is reached, invariant of the behavior that leads to the goal.[4] Thus, σ is of great importance; it is the entire measure of efficiency in reaching the goal.

Moreover, even if we are only interested in maximizing average reward, σ turns out to be a crucial instrument in that maximization. For any policy π, states which lead to the same ergodic set—and all states do, given the unichain assumption—share a common value of $\rho^{\pi}(s)$, so their values $V_T^{\pi}(s)$ differ only with regard to their second component, $\sigma^{\pi}(s)$. Likewise, for all actions a that lead to the same ergodic set, the action-values $Q_T^{\pi}(s,a)$ share a constant ρ component. Clearly, then, σ is the key to policy improvement. When an action a is found for which $Q_{\sigma}^{\pi}(s,a) > V_{\sigma}^{\pi}(s)$, then changing π to choose a in s results in an improvement of V_T^{π}, increasing $\rho(s)$ if s is a recurrent state under π.

[3] Any policy, applied to an MDP, gives rise to a Markov chain. An ergodic set of a Markov chain is a minimal set of states which, once entered, will never be left [7].

[4] Any policy which reaches the goal, strictly speaking, remains there forever. The average reward of a policy reflects this fact, even though a researcher generally stops the simulation at that point and begins a new trial from a different state. In practice, learners are rarely if ever allowed to execute a fixed stationary policy, so $\rho^{\pi}(s)$ need not express the average reward observed during experimental trials.

4 The Connection Between Discounted and Undiscounted Value

Before proposing a method for learning T-optimal policies, we note a connection between discounted and undiscounted value. The main result of interest is the following [11]:

Theorem 3 *For any policy π and state s,*

$$V_{\gamma}^{\pi}(s) = \frac{\rho^{\pi}(s)}{1-\gamma} + \sigma^{\pi}(s) + \varepsilon(s,\gamma), \quad \textit{where } \lim_{\gamma \to 1} \varepsilon(s,\gamma) = 0.$$

We may read this statement as saying that for values of γ approaching 1, the discounted value $V_{\gamma}^{\pi}(s)$ is composed of two nonvanishing terms: one that is a large constant multiple of the average reward expected from starting policy π in state s, and one that is the average-adjusted value of s. We will refer to these as the ρ-term and σ-term, respectively.

So for γ close to 1, V_{γ} may be seen as approximating the lexicographical preference of V_T by giving much more weight to ρ than to σ. But the approximation is imperfect, as is manifest in the fact that V_{γ} dictates opting for a quick constant bonus in reward over a long-term improvement in average reward when $\Delta\sigma$, the gain in finite short-term reward, is greater than $\frac{\Delta\rho}{1-\gamma}$, the scaled difference in long-term average reward. This explains the faulty choice in the example of Figure 1.

Conversely, a simple corollary of Theorem 3 lets us understand average-adjusted value in terms of discounted value:

Corollary 4 *For any policy π and state s,*

$$V_{\sigma}^{\pi}(s) = \lim_{\gamma \to 1} E\left[\sum_{t=0}^{\infty} \gamma^t \left(r_{s,t}^{\pi} - \rho^{\pi}(s)\right)\right]. \qquad (7)$$

If we think of the quantities $r_{s,t}^{\pi} - \rho^{\pi}(s)$ as *average-adjusted rewards* [13], then this corollary lets us view $V_{\sigma}^{\pi}(s)$ as the expected discounted sum of average-adjusted rewards, for vanishingly little discounting (*cf.* Equation 1).

5 Learning T-Optimal Policies

When we express average-adjusted values in the form of Equation 7, it is easy to verify that they obey the following recurrence relation:

$$\sigma^{\pi}(s) = r(s,\pi(s)) - \rho^{\pi}(s) + \sum_{s' \in S} P_{ss'}(\pi(s))\, \sigma^{\pi}(s'). \quad (8)$$

The recurrence, like the expression for value in Corollary 4, is analogous to the discounted case, but with the following two modifications:

1. Rewards are average-adjusted by subtracting out ρ^{π}.
2. The effect of γ is eliminated by bringing γ arbitrarily close to 1.

We may see that first modification enables the second as follows: In the representation of V_γ given by Theorem 3, the first term is the one which blows up for γ close to 1 when $\rho^\pi \neq 0$; this is what prevents us from simply using the undiscounted sum V_1 in the first place. But average-adjusting the incoming rewards reduces the problem to one where $\rho^\pi = 0$, eliminating the first term altogether. This done, we are free to let γ approach 1 in order to eliminate the contribution of the high order term $\varepsilon(s, \gamma)$. What is left is a measure of the σ-term alone which, as we have mentioned, is precisely that upon which we wish to base action selection.

By applying these two modifications to the standard Q-learning algorithm, what results is a technique for approximating $Q_\sigma^{\pi^*}(s, a)$ instead of $Q_\gamma^{\pi^*}(s, a)$, and hence maximizing total rather than γ-discounted value. The only additional machinery needed is a mechanism for approximating the average reward of the successive policies suggested by the algorithm. The method, R-learning, is now presented:

The R-Learning Algorithm

1. Begin with a $|S| \times |A|$ table of real numbers $R(s, a)$, all initialized to zero, and a real-valued variable ρ, also initialized to zero.
2. Repeat:
   - 2a. From the current state s, perform an action a, chosen by some exploration/action-selection mechanism (this is an orthogonal component of the system, just as it is for Q-learning). Observe the immediate reward r_{imm} received and the subsequent state s'.
   - 2b. Update R according to:
   $$R(s, a) \xleftarrow{\beta} r_{imm} - \rho + U_R(s') \tag{9}$$
   where $U_R(s') = \max_{a'} R(s', a')$ and β is a learning rate parameter.
   - 2c. If $R(s, a) = U_R(s)$ (*i.e.*, if a agrees with the policy π^R), then update ρ according to:
   $$\rho \xleftarrow{\alpha} r_{imm} + U_R(s') - U_R(s) \tag{10}$$
   where α is a learning rate parameter.

One might wonder why we do not simply approximate ρ via exponential averaging of immediate rewards (*i.e.*, $\rho \xleftarrow{\alpha} r_{imm}$), performed on *every* tick. This is so as to restrict our attention to the policy π^R, uninfluenced by the conflicting behavior of the exploration mechanism that is ultimately responsible for action choice. Exploratory actions, which generally incur subaverage rewards, would skew the approximation of ρ if included in the calculation. The appearance of the term $U_R(s') - U_R(s)$ in Equation 9 plays the role of compensating for known variations in the rewards received in different states; it serves to adjust for periods of low reward when the agent is using its optimal policy to *recover* from suboptimal exploratory actions, and more generally to minimize the variance of the values used to estimate ρ.

One may easily show that whenever R-learning converges it must arrive at a T-optimal policy. But unlike the discounted methods, whose mathematics have been extensively explored (*e.g.*, [18]), a proof of the convergence of R-learning has not been established. Nonetheless, related techniques such as undiscounted Policy Improvement [5] are well understood, and work toward convergence results is currently under way.

6 Advantages of R-Learning

R-learning is designed to arrive at T-optimal policies, and we have already argued for the advantages of undiscounted performance criteria. But in addition to maximizing undiscounted performance, R-learning displays several computational advantages over existing techniques. As a result, even when one can be sure of choosing γ so as to allow Q-learning to arrive at T-optimal policies, it is often preferable to use R-learning (bearing in mind that the algorithm has not yet been *proven* to converge).

For discounted methods such as Q-learning to arrive at policies that do not overlook temporally distant rewards, they must use large values of γ. But it is well known that for successive approximation techniques the geometric rate of decay in the approximation error is proportional to γ, so that high values of γ can slow the convergence dramatically [2]. As a result, one might expect an undiscounted method such as R-learning to converge extremely slowly. In fact the opposite is generally true, for the following reasons:

6.1 Better Initial Estimates

In cases where the optimal policy π^* has nonzero average reward and γ is near 1, Q-learning spends much of its time converging to the large ρ-term of $V_\gamma^{\pi^*}$, which is invariant of s, whereas the σ-term is the one needed for action choice. During this time, the true values of σ necessary for action selection may be entirely obscured by approximation error, causing poor performance. R-values, in contrast, have no contribution from ρ. As a result, the values, which begin at zero, already reflect their ρ-term, and need only converge on σ.

6.2 Faster Propagation of Rewards

Many researchers have pointed out that information about rewards can be propagated very slowly across states by Q-learning [10, 16]. In particular, we note that the ρ-terms of the Q-values, though constant over a state space, may be updated only locally, and one state at a time. By contrast, R-learning uses ρ to effectively store a common ρ-term for all its action-values. This term is updated on every iteration, and lets information be propagated instantly throughout the state space. Section 7.2 presents experimental data attesting to the resulting speedup.

6.3 Value Disambiguation

Even for MDP's where γ-optimal policies are T-optimal for small γ—in which case one would hope for Q-learning to converge rapidly despite the above concerns—there turn out to be compelling reasons to prefer R-learning. Let us examine any case where temporal credit assignment is necessary—where some T-suboptimal action gives immediate reward as large as that of the T-optimal action. That is, for some state s of an MDP with T-optimal policy π, $Q_T^{\pi^*}(s,a) > Q_T^{\pi^*}(s,b)$, but $r(s,b) \geq r(s,a)$. Then one may show that there is a value γ_0 such that $Q_{\gamma_0}^{\pi^*}(s,a) = V_{\gamma_0}^{\pi^*}(s,b)$, and for which $Q_\gamma^{\pi^*}(s,a) > Q_\gamma^{\pi^*}(s,b)$ holds of all $\gamma_0 \leq \gamma < 1$.[5] To be sure to prefer the T-optimal action, one must choose a γ within this interval. This granted, one would like to choose a γ at the very bottom of the range so as to speed convergence. But choosing γ arbitrarily close to γ_0 will result in an arbitrarily small difference between $Q_\gamma^{\pi^*}(s,a)$ and $Q_\gamma^{\pi^*}(s,b)$.[6] This is problematic for at least the following two reasons:

First, for stochastic MDP's, approximation errors of Q-values are due not only to the process of value iteration, but also to the Monte Carlo sampling it employs. When differences in true action-values for competing actions are small they are more likely to be obscured by such errors, which may cause suboptimal actions to be chosen. An example of this phenomenon is presented in Section 7.1.

Secondly, many popular exploration methods select actions stochastically according to their relative action-values [6, 16, 17]. In such cases, an action whose Q-value differs from the optimal one by a small amount will be chosen almost as frequently as the optimal action. This may result in a substantial loss of cumulative reward.

One may see that R-learning minimizes these problems, because using average-adjusted value maximizes the differences in value in the following sense:

$$Q_\sigma^{\pi^*}(s,a) - Q_\sigma^{\pi^*}(s,b) = \sup_{\gamma_0 \leq \gamma < 1} \left\{ Q_\gamma^{\pi^*}(s,a) - Q_\gamma^{\pi^*}(s,b) \right\}$$

In summary, choosing γ to be large for Q-learning results in poor initial estimates and slow convergence, while choosing γ small, when it does not result in T-suboptimal solutions, may reduce performance dramatically for other reasons. R-learning eliminates the hazards that accompany small values of γ, while using other means to speed convergence.

[5] This follows from the proof of existence of stationary Blackwell optimal policies ([3], Theorem 5).

[6] In goal tasks where nonzero reward is given only upon achieving the goal, this is the familiar problem wherein temporal differencing causes all actions in states distant from the goal to have approximately zero value.

6.4 Linearity of Undiscounted Values

In the case of a deterministic MDP, we may read the recurrence for σ (Equation 8) as telling us that

$$\sigma^\pi(s') - \sigma^\pi(s) = \rho(s) - r(s, \pi(s))$$

where s' is the state reached by performing $\pi(s)$ in s. Since unichain policies have $\rho(s)$ constant, it follows that for regions of S where following π incurs a constant reward, average-adjusted value of successive states encountered will change linearly. By contrast, the γ present in Q-values makes increases and decreases in those values exponential, growing in magnitude with temporal proximity to rewards.

Linearity is very desirable in the case where S is a metric space, and the policy action causes uniform movement along some dimension of S. In this case the R-values for the policy action vary linearly over that dimension, potentially permitting the use of simple and accurate linear interpolation to speed up learning and even allowing learning over continuous state spaces.

Finally, the linearity of σ values can be useful in situations where we want to combine action-values for multiple individual tasks to determine values for a composite task. Singh [14] has explored this possibility for sets of goal tasks.

6.5 Q-Learning Seen as a Special Case of R-learning

Occasions might arise where temporal discounting is a useful feature for any of the reasons given in Section 3.1. As we saw, the discounted value V_γ measures total value in the case where at each point in time there is a probability $1-\gamma$ of an exogenous incident after which all rewards are zero. Given any MDP, we may introduce this assumption explicitly by adding one absorbing state s_{ab} with autotransitions yielding reward zero, to which all actions from all other states lead with probability $1-\gamma$. (Such actions receive their normal reward, since it is only the state they result in that changes.) Accordingly, we multiply all preexisting transition probabilities by γ. Now let quantities marked by a dot denote values pertaining to the modified MDP. We first observe that for all policies and all states s of the new decision problem, $\dot\rho^\pi(s) = 0$, and build this fact into the algorithm by eliminating step 2c. Now by Equation 8 we have, for all $s \in S$,

$$\begin{aligned}
\dot Q_\sigma^\pi(s,a) &= \dot r(s,a) - \dot\rho^\pi(s) + \sum_{s' \in \dot S} \dot P_{ss'}(a) \dot V_\sigma^\pi(s') \\
&= \dot r(s,a) - \dot\rho^\pi(s) + (1-\gamma)\dot V_\sigma^\pi(s_{ab}) + \\
&\quad \sum_{s' \neq s_{ab}} \left(\gamma P_{ss'}(a)\right) \dot V_\sigma^\pi(s') \\
&= r(s,a) + \gamma \sum_{s' \in S} P_{ss'}(a) \dot V_\sigma^\pi(s').
\end{aligned}$$

In other words, $\dot Q_\sigma$ obeys the same recurrence relation as Q_γ. By modifying the R-learning update of step 2b to reflect the recurrence of $\dot\sigma$ instead of σ, we are left with precisely the Q-learning algorithm.

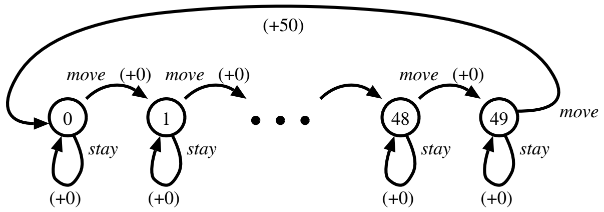

Figure 2: A simple MDP with temporally distant rewards.

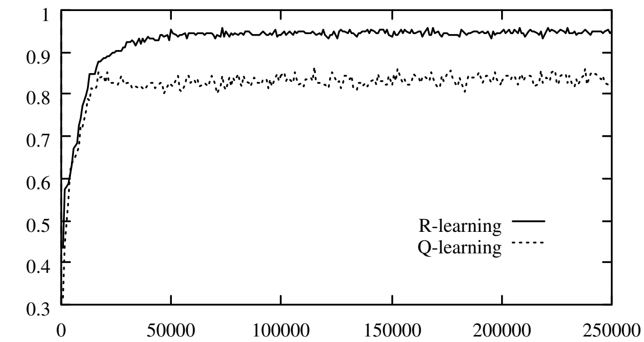

Figure 3: Performance of R-Learning versus Q-learning.

7 Experimental Results

7.1 Value Disambiguation

An initial experiment compared Q-learning and R-learning in the simple domain pictured in Figure 2, modified to yield stochastic rewards that always deviate from the values given by either +1 or -1 (with equal probability). Figure 3 shows the results. The x-axis measures number of actions performed, while the y-axis measures average reward per 5000-action interval. (The results are averaged over 50 trials.) We used random exploration with a fixed probability 0.05 of a random action at any time step; an exploration method that favors actions with near-optimal values would make the advantage of R-learning more pronounced.

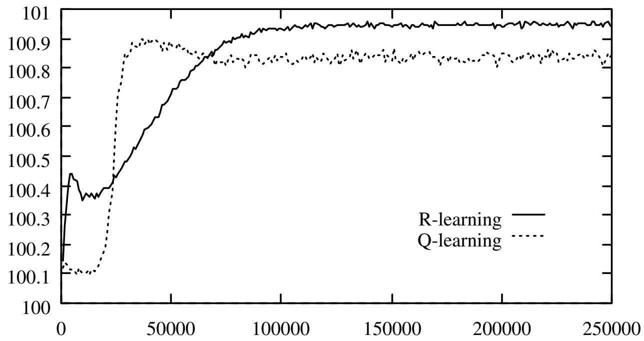

Figure 4: Performance of R-Learning versus Q-learning (all rewards increased by 100).

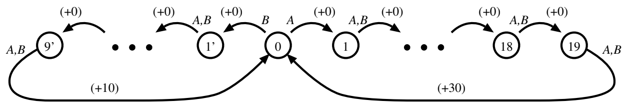

Figure 5: An MDP with two cycles. Action choice is irrelevant except in state 0.

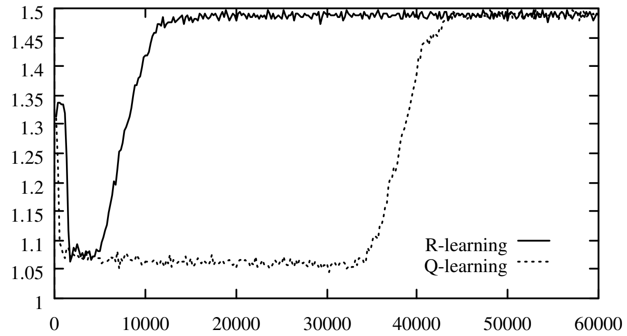

Figure 6: Comparison of learning mechanisms on a MDP with two cycles. Performance of Q-learning is poor compared to R-learning because of ramping.

Figure 4 shows a comparison of Q-learning and R-learning in the same domain with all rewards increased uniformly by 100. Notice the period in which R-learning stalls while its estimate of ρ climbs from an initial value of zero up to the final value of 101.

Both runs use $\gamma = 0.9, \beta = 0.2$. Because of the exploration strategy, an optimal policy will have an average reward of 0.95. Note that the fixed β is responsible for the fact that the Q-values never converge to the optimal value.

7.2 Convergence Rate

A second experiment used the double-loop domain shown in Figure 5. Average rewards of the two methods are plotted in Figure 6, compiled over 100 runs. The reason for the poor performance by Q-learning is that the shorter cycle, though it gives less per-step payoff than the longer one, allows rewards to be propagated more quickly. As a result, it's Q-values converge faster than the longer one's, making it look more favorable during the long process of convergence.

These experiments use the same parameters as the previous ones, except that here γ is increased to 0.99 to allow Q-learning to learn the T-optimal policy at all. For larger γ, the effect is even more pronounced: When $\gamma = 0.999$ instead of 0.99, the speedup of R-learning over Q-learning is more than forty-fold.

8 Related Work

In Section 2 we remarked that Q-learning may be used to find V_1-optimal policies in cases where that undiscounted measure is finite. Now we may further observe that in such cases, $V_T^\pi(s) = \langle 0, V_1^\pi(s)\rangle$ and the recurrence for V_σ reduces to that of V_1. In this sense, the theory of total value V_T subsumes that of V_1, and R-learning is a generalization of undiscounted Q-learning.

Whereas Q-learning is an asynchronous, stochastic approximation version of the well-understood dynamic programming method of value iteration, R-learning has no such precise analog in the literature. Though several successive approximation algorithms exist for computing σ, none of them approximate ρ explicitly; this added complexity present in R-learning, while making it more robust in the multichain case, necessitates new methods of analysis, which we are working to develop. We know of no successive approximation methods in the Dynamic Programming literature for handling the multichain case.

We have recently learned of the related work of Westerdale [19], who presents a bucket brigade technique based on the notions of average and average-adjusted value. Since he does not draw any connections to the Dynamic Programming or Reinforcement Learning literature, the precise contribution of this work is difficult to assess.

The general notion that performance should be viewed relative to a standard of reference, which underlies average-adjusted value and the workings of R-learning, has appeared in many places in the psychological literature, as well as in Sutton's Reinforcement Comparison algorithm [15]. A presentation of R-learning from a more psychological standpoint is offered in [13].

9 Conclusion

Until now, most of the work in RL has maintained one standard of performance while using algorithms that maximize another. The discounted algorithms have been well understood, but the performance standards have not: In goal tasks, maximal undiscounted value V_1 has been sought, while, in recurrent domains, high average reward has been the aim. The concept of total reward presented here subsumes these two criteria, and is the basis for R-learning, a method we have proposed to maximize them both. We have used total reward as a tool to gain a better understanding of some of the computational shortcomings of the discounted techniques, and have shown that R-learning may ameliorate many of those problems.

While informal tests have shown R-learning to be applicable to a variety of domains, more empirical and theoretical work is necessary to establish its viability as a robust algorithm for reinforcement learning. Some of this work is currently under way.

Acknowledgements

I wish to thank Nils Nilsson, Rich Sutton, George John, and Moshe Tennenholtz for their helpful comments. This work was supported by an IBM Graduate Fellowship.

References

[1] A. G. Barto, R. S. Sutton, and C. J. C. H. Watkins. Learning and sequential decision making. Technical Report COINS 89-95, Dept. of Computer and Information Science, University of Massachusetts, Amherst, 1989.

[2] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, Englewood Cliffs, NJ, 1989.

[3] D. Blackwell. Discrete dynamic programming. *Ann. Math. Statist.*, 33:719–726, 1962.

[4] G. H. Hardy. *Divergent Series*. Clarendon Press, Oxford, 1949.

[5] R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

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

[7] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Van Nostrand, Princeton, NJ, 1960.

[8] L.-J. Lin. Programming robots using reinforcement learning and teaching. In *Proceedings AAAI-91*, pages 781–786. MIT Press, Cambridge, MA, 1991.

[9] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. In *Proceedings AAAI-91*, pages 768–773. MIT Press, Cambridge, MA, 1991.

[10] R. A. McCallum. Using transitional proximity for faster reinforcement learning. In *Proceedings of the Ninth International Workshop on Machine Learning*, pages 316–321. Morgan Kaufmann, San Mateo, CA, 1992.

[11] M. L. Puterman. Markov decision processes. In D. P. Heyman and M. J. Sobel, editors, *Handbooks in OR & MS, Vol. 2*, pages 331–434. Elsevier, North-Holland, 1990.

[12] S. M. Ross. *Introduction to Stochastic Dynamic Programming*. Academic Press, New York, 1983.

[13] A. Schwartz. Thinking locally to act globally: A novel approach to reinforcement learning. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*. Lawrence Erlbaum, Hillsdale, NJ, 1993.

[14] S. P. Singh. Transfer of learning by composing solutions for elemental sequential tasks. *Machine Learning*, 8(3/4):323–339, May 1992.

[15] R. S. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, Department of Computer and Information Sciences, University of Massachusetts, 1984.

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

[17] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, 1989.

[18] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[19] T. H. Westerdale. Quasimorphisms or queasymorphisms? Modeling finite automaton environments. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 128–147. Morgan Kaufmann, San Mateo, CA, 1991.