Machine Learning, 22, 33–57 (1996)

Linear Least–Squares Algorithms for Temporal Difference Learning

STEVEN J. BRADTKE bradtke@gte.com
GTE Data Services, One E Telecom Pkwy, DC B2H, Temple Terrace, FL 33637

ANDREW G. BARTO barto@cs.umass.edu
Dept. of Computer Science, University of Massachusetts, Amherst, MA 01003-4610

Editor: Leslie Pack Kaelbling

Abstract. We introduce two new temporal difference (TD) algorithms based on the theory of linear least–squares function approximation. We define an algorithm we call Least–Squares TD (LS TD) for which we prove probability–one convergence when it is used with a function approximator linear in the adjustable parameters. We then define a recursive version of this algorithm, Recursive Least–Squares TD (RLS TD). Although these new TD algorithms require more computation per time–step than do Sutton's TD(λ) algorithms, they are more efficient in a statistical sense because they extract more information from training experiences. We describe a simulation experiment showing the substantial improvement in learning rate achieved by RLS TD in an example Markov prediction problem. To quantify this improvement, we introduce the *TD error variance* of a Markov chain, σ_{TD}, and experimentally conclude that the convergence rate of a TD algorithm depends linearly on σ_{TD}. In addition to converging more rapidly, LS TD and RLS TD do not have control parameters, such as a learning rate parameter, thus eliminating the possibility of achieving poor performance by an unlucky choice of parameters.

Keywords: Reinforcement learning, Markov Decision Problems, Temporal Difference Methods, Least–Squares

1. Introduction

The class of temporal difference (TD) algorithms (Sutton, 1988) was developed to provide reinforcement learning systems with an efficient means for learning when the consequences of actions unfold over extended time periods. They allow a system to learn to predict the total amount of reward expected over time, and they can be used for other prediction problems as well (Anderson, 1988, Barto, et al., 1983, Sutton, 1984, Tesauro, 1992). We introduce two new TD algorithms based on the theory of linear least–squares function approximation. The recursive least–squares function approximation algorithm is commonly used in adaptive control (Goodwin & Sin, 1984) because it can converge many times more rapidly than simpler algorithms. Unfortunately, extending this algorithm to the case of TD learning is not straightforward.

We define an algorithm we call Least–Squares TD (LS TD) for which we prove probability–one convergence when it is used with a function approximator linear in the adjustable parameters. To obtain this result, we use the instrumental variable approach (Ljung & Soderström, 1983, Söderström & Stoica, 1983, Young, 1984) which provides a way to handle least–squares estimation with training data that is noisy on both the input and output observations. We then define a recursive version of this algorithm, Re-

cursive Least–Squares TD (RLS TD). Although these new TD algorithms require more computation per time step than do Sutton's TD(λ) algorithms, they are more efficient in a statistical sense because they extract more information from training experiences. We describe a simulation experiment showing the substantial improvement in learning rate achieved by RLS TD in an example Markov prediction problem. To quantify this improvement, we introduce the *TD error variance* of a Markov chain, σ_{TD}, and experimentally conclude that the convergence rate of a TD algorithm depends linearly on σ_{TD}. In addition to converging more rapidly, LS TD and RLS TD do not have control parameters, such as a learning rate parameter, thus eliminating the possibility of achieving poor performance by an unlucky choice of parameters.

We begin in Section 2 with a brief overview of the policy evaluation problem for Markov decision processes, the class of problems to which TD algorithms apply. After describing the TD(λ) class of algorithms and the existing convergence results in Sections 3 and 4, we present the least–squares approach in Section 5. Section 6 presents issues relevant to selecting an algorithm, and Sections 7 and 8 introduce the TD error variance and use it to quantify the results of a simulation experiment.

2. Markov Decision Processes

TD(λ) algorithms address the policy evaluation problem associated with discrete–time stochastic optimal control problems referred to as Markov decision processes (MDPs). An MDP consists of a discrete–time stochastic dynamic system (a controlled Markov chain), an immediate reward function, R, and a measure of long–term system performance. Restricting attention to finite–state, finite–action MDP's, we let X and A respectively denote finite sets of states and actions, and P denote the state transition probability function. At time step t, the controller observes the current state, x_t, and executes an action, a_t, resulting in a transition to state x_{t+1} with probability $P(x_t, x_{t+1}, a_t)$ and the receipt of an immediate reward $r_t = R(x_t, x_{t+1}, a_t)$. A (stationary) *policy* is a function $\mu : X \rightarrow A$ giving the controller's action choice for each state.

For each policy μ there is a *value function*, V^μ, that assigns to each state a measure of long–term performance given that the system starts in the given state and the controller always uses μ to select actions. Confining attention to the *infinite–horizon discounted* definition of long–term performance, the value function for μ is defined as follows:

$$V^\mu(x) = E_\mu[\sum_{k=0}^{\infty} \gamma^k r_k | x_0 = x],$$

where γ, $0 \leq \gamma < 1$, is the discount factor and E_μ is the expectation given that actions are selected via μ. (In problems in which one can guarantee that there will exist some finite time τ such that $r_k = 0$ for $k \geq \tau$, then one can set $\gamma = 1$.) The objective of the MDP is to find a policy, μ^*, that is optimal in the sense that $V^{\mu^*}(x) \geq V^\mu(x)$ for all $x \in X$ and for all policies μ.

Computing the evaluation function for a given policy is called *policy evaluation*. This computation is a component of the policy iteration method for finding an optimal policy.

and it is sometimes of interest in its own right to solve prediction problems, the perspective taken by Sutton (Sutton, 1988). The evaluation function of a policy must satisfy the following consistency condition: for all $x \in X$:

$$V^{\mu}(x) = \sum_{y \in X} P(x, y, \mu(x))[R(x, y, \mu(x)) + \gamma V^{\mu}(y)].$$

This is a set of $|X|$ linear equations which can be solved for V^{μ} using any of a number of standard direct or iterative methods when the functions R and P are known. The TD(λ) family of algorithms apply to this problem when these functions are not known. Since our concern in this paper is solely in the problem of evaluating a fixed policy μ, we can omit reference to the policy throughout. We therefore denote the value function V^{μ} simply as V, and we omit the action argument in the functions R and P. Furthermore, throughout this paper, by a Markov chain we always mean a finite–state Markov chain.

3. The TD(λ) Algorithm

Although any TD(λ) algorithm can be used with a lookup table function representation, it is most often described in terms of a parameterized function approximator. In this case, V_t, the approximation of V at time step t, is defined by $V_t(x) = f(\theta_t, \phi_x)$, for all $x \in X$, where θ_t is a parameter vector at time step t, ϕ_x is a feature vector representing state x, and f is a given real–valued function differentiable with respect to θ_t for all ϕ_x. We use the notation $\nabla_{\theta_t} V_t(x)$ to denote the gradient vector at state x of V_t as a function of θ_t.

Table 1. Notation used in the discussion of the TD(λ) learning rule.

x, y, z	states of the Markov chain
x_t	the state at time step t
r_t	the immediate reward associated with the transition from state x_t to x_{t+1}; $r_t = R(x_t, x_{t+1})$.
$\bar{r}$	the vector of expected immediate rewards; $\bar{r}_x = \sum_{y \in X} P(x, y) R(x, y)$; $r_t = \sum_{y \in X} P(x_t, y) R(x_t, y)$
S	the vector of starting probabilities
X'	the transpose of the vector or matrix X
V	the true value function
ϕ_x	the feature vector representing state x
ϕ_t	the feature vector representing state x_t. $\phi_t = \phi_{x_t}$.
Φ	the matrix whose x-th row is ϕ_x
π_x	the proportion of time that the Markov chain is expected to spend in state x
Π	the diagonal matrix $\mathrm{diag}(\pi)$
θ^*	the true value function parameter vector
θ_t	the estimate of θ^* at time t
$V_t(x)$	the estimated value of state x using parameter vector θ_t
$\alpha_{\eta(x_t)}$	the step-size parameter used to update the value of θ_t
$\eta(x_t)$	the number of transitions from state x_t up to time step t.

Using additional notation, summarized in Table 1, the TD(λ) learning rule for a differentiable parameterized function approximator (Sutton, 1988) updates the parameter

vector θ_t as follows:

$$\begin{aligned}\theta_{t+1} &= \theta_t + \alpha_{\eta(x_t)} \left[R_t + \gamma V_t(x_{t+1}) - V_t(x_t) \right] \sum_{k=1}^{t} \lambda^{t-k} \nabla_{\theta_t} V_t(x_k) \\ &= \theta_t + \alpha_{\eta(x_t)} \Delta\theta_t,\end{aligned}$$

where

$$\begin{aligned}\Delta\theta_t &= \left[R_t + \gamma V_t(x_{t+1}) \quad V_t(x_t) \right] \sum_{k=1}^{t} \lambda^{t-k} \nabla_{\theta_t} V_t(x_k) \\ &- \left[R_t + \gamma V_t(x_{t+1}) - V_t(x_t) \right] \Sigma_t\end{aligned}$$

and

$$\Sigma_t = \sum_{k=1}^{t} \lambda^{t-k} \nabla_{\theta_t} V_t(x_k). \tag{1}$$

Notice that $\Delta\theta_t$ depends only on estimates, $V_t(x_k)$, made using the latest parameter values, θ_t. This is an attempt to separate the effects of changing the parameters from the effects of moving through the state space. However, when V_t is not linear in the parameter vector θ_t and $\lambda \neq 0$, the sum Σ_t cannot be formed in an efficient, recursive manner. Instead, it is necessary to remember the x_k and to explicitly compute $\nabla_{\theta_t} V_t(x_k)$ for all $k < t$. This is necessary because if V_t is nonlinear in θ_t, $\nabla_{\theta_t} V_t(x_k)$ depends on θ_t. Thus, Σ_t cannot be defined recursively in terms of Σ_{t-1}. Because recomputing Σ_t in this manner at every time step can be expensive, an approximation is usually used. Assuming that the step–size parameters are small, the difference between θ_t and θ_{t-1} is also small. Then an approximation to Σ_k can be defined recursively as

$$\Sigma_t = \lambda\Sigma_{t-1} + \nabla_{\theta_t} V_t(x_t). \tag{2}$$

If V *is* linear in θ, then (2) can be used to compute Σ_t exactly. No assumptions about the step–size parameters are required, and no approximations are made.

We will be concerned in this paper with function approximators that are linear in the parameters, that is, functions that can be expressed as follows: $V_t(x) = \phi_x' \theta_t$, where ϕ_x' denotes the transpose of ϕ_x so that $\phi_x' \theta_t$ is the inner product of ϕ_x and θ_t. In this case, (2) becomes

$$\Sigma_t = \lambda\Sigma_{t-1} + \phi_t,$$

so that (1) simplifies to

$$\Sigma_t = \sum_{k=1}^{t} \lambda^{t-k} \phi_k.$$

```
1   Select θ_0.
2   Set t ← 0.
3   for n ← 0 to ∞ {
4       Choose a start state x_t according to the start-state probabilities given by S.
5       Set Δ_n ← 0.
6       while x_t is not an absorbing state {
7           Let the state change from x_t to x_{t+1} according to the Markov chain transition
            probabilities.
8           Set Δ_n = Δ_n + Δθ_t, where Δθ_t is given by (3).
9           t ← t + 1.
10      }
11      Update the parameters at the end of trial number n: θ_{n+1} = θ_n + α_n Δ_n.
12  }
```

Figure 1. Trial-based TD(λ) for absorbing Markov chains. A trial is a sequence of states generated by the Markov chain, starting with some initial state and ending in an absorbing state. The variable n counts the number of trials. The variable k counts the number of steps within a trial. The parameter vector θ is updated only at the end of a trial.

4. Previous Convergence Results for TD(λ)

Convergence of a TD(λ) learning rule depends on the state representation, $\{\phi_x\}_{x \in X}$, and the form of the function approximator. Although TD(λ) rules have been used successfully with function approximators that are nonlinear in the parameter vector θ, most notably the use of a multi-layer artificial neural network in Tesauro's backgammon programs (Tesauro, 1992), convergence has only been proven for cases in which the value function is represented as a lookup table or as a linear function of θ when the feature vectors are linearly independent.[1] Sutton (Sutton, 1988) and Dayan (Dayan, 1992) proved parameter convergence in the mean under these conditions, and Dayan and Sejnowski (Dayan & Sejnowski, 1994) proved parameter convergence with probability 1 under these conditions for TD(λ) applied to absorbing Markov chains in a *trial based* manner, i.e., with parameter updates only at the end of every trial. A trial is a sequence of states generated by the Markov chain, starting with some initial state and ending in an absorbing state. The start state for each trial is chosen according to a probability distribution S. Figure 1 describes this algorithm. Since parameter updates take place only at the end of each trial, $\Delta\theta_t$ must be defined somewhat differently from above:

$$\Delta\theta_t = \left[R_t + \gamma\theta_n'\phi_{t+1} - \theta_n'\phi_t \right] \sum_{k=1}^{t} \lambda^{t-k}\phi_k, \tag{3}$$

where n is the trial number and t is the time step. The parameter vector θ_n is held constant throughout trial n, and is updated only at the end of each trial.

Less restrictive theorems have been obtained for the TD(0) algorithm by considering it as a special case of Watkins' (Watkins, 1989) Q-learning algorithm. Watkins and Dayan (Watkins & Dayan, 1992), Jaakkola, Jordan, and Singh, (Jaakkola, et al., 1994), and Tsitsiklis (Tsitsiklis, 1993) note that since the TD(0) learning rule is a special case of Q-learning, their probability–one convergence proofs for Q-learning can be used to show that *on–line* use of the TD(0) learning rule (i.e., not trial–based) with a lookup–table function representation converges to V with probability 1. Bradtke (Bradtke, 1994) extended Tsitsiklis' proof to show that on–line use of TD(0) with a function approximator that is linear in the parameters and in which the feature vectors are linearly independent also converges to V with probability 1.

Bradtke also proved probability–one convergence under the same conditions for a normalized version of TD(0) that he called NTD(0) (Bradtke, 1994). Bradtke also defined the NTD(λ) family of learning algorithms, which are normalized versions of TD(λ). As with similar learning algorithms, the size of the input vectors ϕ_x can cause instabilities in TD(λ) learning until the step–size parameter, α, is reduced to a small enough value. But this can make the convergence rate unacceptably slow. The NTD(λ) family of algorithms addresses this problem. Since we use NTD(λ) in the comparative simulations presented below, we define it here.

The NTD(λ) learning rule for a function approximator that is linear in the parameters is

$$\theta_{t+1} = \theta_t + \alpha_t(x_t) \left[R_t + \gamma\theta_t'\phi_{t+1} - \theta_t'\phi_t \right] \sum_{k=1}^{t} \lambda^{t-k} \frac{\phi_k}{\epsilon + \phi_k'\phi_k}, \tag{4}$$

where ϵ is some small, positive number. If we know that all of the ϕ_t are non–zero, then we can set ϵ to zero. The normalization does not change the directions of the updates; it merely bounds their size, reducing the chance for unstable behavior.

5. A Least–Squares Approach to TD Learning

The algorithms described above require relatively little computation per time step, but they use information rather inefficiently compared to algorithms based on the least squares approach. Although least–squares algorithms require more computation per time step, they typically require many fewer time steps to achieve a given accuracy than do the algorithms described above. This section describes a derivation of a TD learning rule based on least–squares techniques. Table 2 summarizes the notation we use in this section.

5.1. *Linear Least–Squares Function approximation*

This section reviews the basics of linear least–squares function approximation, including instrumental variable methods. This background material leads in the next section to a least–squares TD algorithm. The goal of linear least–squares function approximation is

Table 2. Notation used throughout this section in the discussion of Least–Squares algorithms.

Ψ	$\Psi : \Re^n \longrightarrow \Re$, the linear function to be approximated
ω_t	$\omega_t \in \Re^n$, the observed input at time step t
ψ_t	$\psi_t \in \Re$, the observed output at time step t
η_t	$\eta_t \in \Re$, the observed output noise at time step t
$\hat{\omega}_t$	$\hat{\omega}_t = \omega_t + \zeta_t$, the noisy input observed at time t
ζ_t	$\zeta_t \in \Re^n$, the input noise at time step t
$\mathrm{Cor}(x, y)$	$\mathrm{Cor}(x, y) = E\{xy'\}$, the correlation matrix for random variables x and y
ρ_t	$\rho_t \in \Re^n$, the instrumental variable observed at time step t

to linearly approximate some function $\Psi : \Re^n \longrightarrow \Re$ given samples of observed inputs $\omega_t \in \Re^n$ and the corresponding observed outputs $\psi_t \in \Re$. If the input observations are not corrupted by noise, then we have the following situation:

$$\begin{aligned} \psi_t &= \Psi(\omega_t) + \eta_t \\ &= \omega_t'\theta^* + \eta_t, \end{aligned} \tag{5}$$

where θ^* is the vector of true (but unknown) parameters and η_t is the output observation noise.

Given (5), the least–squares approximation to θ^*at time t is the vector θ_t that minimizes the quadratic objective function

$$J_t = \frac{1}{t}\sum_{k=1}^{t} [\psi_k - \omega_k'\theta_t]^2.$$

Taking the partial derivative of J_t with respect to θ_t, setting this equal to zero and solving for the minimizing θ_t gives us the t^{th} estimate for θ^*,

$$\theta_t = \left[\frac{1}{t}\sum_{k=1}^{t} \omega_k\omega_k'\right]^{-1}\left[\frac{1}{t}\sum_{k=1}^{t} \omega_k\psi_k\right]. \tag{6}$$

The following lemma, proved in ref. (Young, 1984), gives a set of conditions under which θ_t as defined by (6) converges in probability to θ^*:

LEMMA 1 *If the correlation matrix $Cor(\omega, \omega)$ is nonsingular and finite, and the output observation noise η_k is uncorrelated with the input observations ω_k, then θ_t as defined by (6) converges in probability to θ^*.*

Equation (5) models the situation in which observation errors occur only on the output. In the more general case, the input observations are also noisy. Instead of being able to directly observe ω_t, we can only observe $\hat{\omega}_t = \omega_t + \zeta_t$, where ζ_t is the input observation noise vector at time t. This is known as an *errors–in–variables* situation (Young, 1984). The following equation models the errors–in–variables situation:

$$\begin{aligned} \psi_t &= \Psi(\omega_t) + \eta_t \\ &= \Psi(\hat{\omega}_t - \zeta_t) + \eta_t \\ &= \hat{\omega}_t'\theta^* - \zeta_t'\theta^* + \eta_t. \end{aligned} \tag{7}$$

The problem with the errors–in–variables situation is that we cannot use $\hat{\omega}_t$ instead of ω_t in (6) without violating the conditions of Lemma (1). Substituting $\hat{\omega}_t$ directly for ω_t in (6) has the effect of introducing noise that is dependent upon the current state. This introduces a bias, and θ_t no longer converges to θ^*. One way around this problem is to introduce *instrumental variables* (Ljung & Söderström, 1983, Söderström & Stoica, 1983, Young, 1984). An instrumental variable, ρ_t, is a vector that is correlated with the true input vectors, ω_t, but that is uncorrelated with the observation noise, ζ_t. The following equation is a modification of (6) that uses the instrumental variables and the noisy inputs.

$$\theta_t = \left[\frac{1}{t}\sum_{k=1}^{t} \rho_k \hat{\omega}_k'\right]^{-1} \left[\frac{1}{t}\sum_{k=1}^{t} \rho_k \psi_k\right]. \tag{8}$$

The following lemma, proved in ref. (Young, 1984), gives a set of conditions under which the introduction of instrumental variables solves the errors–in–variables problem.

LEMMA 2 *If the correlation matrix $Cor(\rho, \omega)$ is nonsingular and finite, the correlation matrix $Cor(\rho, \zeta) = 0$, and the output observation noise η_t is uncorrelated with the instrumental variables ρ_t, then θ_t as defined by (8) converges in probability to θ^*.*

5.2. *Algorithm LS TD*

Here we show how to use the instrumental variables method to derive an algorithm we call Least–Squares TD (LS TD), a least–squares version of the TD algorithm. The TD algorithm used with a linear–in–the–parameters function approximator addresses the problem of finding a parameter vector, θ^*, that allows us to compute the value of a state x as $V(x) = \phi_x'\theta^*$. Recall that the value function satisfies the following consistency condition:

$$\begin{aligned} V(x) &= \sum_{y \in X} P(x,y)[R(x,y) + \gamma V(y)] \\ &= \sum_{y \in X} P(x,y)R(x,y) + \gamma \sum_{y \in X} P(x,y)V(y) \\ &= \bar{r}_x + \gamma \sum_{y \in X} P(x,y)V(y), \end{aligned}$$

where $\bar{r}_x$ is the expected immediate reward for any state transition from state x. We can rewrite this in the form used in Section 5.1 as

$$\begin{aligned} \bar{r}_x &= V(x) - \gamma \sum_{y \in X} P(x,y)V(y) \\ &= \phi_x'\theta^* - \gamma \sum_{y \in X} P(x,y)\phi_y'\theta^* \end{aligned}$$

$$= (\phi_x - \gamma \sum_{y \in X} P(x,y)\phi_y)'\theta^*, \tag{9}$$

for every state $x \in X$. Now we have the same kind of problem that we considered in Section 5.1. The scalar output, $\bar{r}_x$, is the inner product of an input vector, $\phi_x - \gamma \sum_{y \in X} P(x,y)\phi_y$, and the true parameter vector, θ^*.

For each time step t, we therefore have the following equation that has the same form as (5):

$$r_t = (\phi_t - \gamma \sum_{y \in X} P(x_t,y)\phi_y)'\theta^* + (r_t - \bar{r}_t), \tag{10}$$

where r_t is the reward received on the transition from x_t to x_{t+1}. $(r_t - \bar{r}_t)$ corresponds to the noise term η_t in (5). The following lemma, proved in Appendix A, establishes that this noise term has zero mean and is uncorrelated with the input vector $\omega_t = \phi_t - \gamma \sum_{y \in X} P(x_t,y)\phi_y$:

LEMMA 3 *For any Markov chain, if x and y are states such that $P(x,y) > 0$, with $\eta_{xy} = R(x,y) - \bar{r}_x$ and $\omega_x = (\phi_x - \gamma \sum_{y \in X} P(x,y)\phi_y)$, then $E\{\eta\} = 0$, and $Cor(\omega,\eta) = 0$.*

Therefore, if we know the state transition probability function, P, the feature vector ϕ_x, for all $x \in X$, if we can observe the state of the Markov chain at each time step, and if $\mathrm{Cor}(\omega,\omega)$ is nonsingular and finite, then by Lemma (1) the algorithm given by (6) converges in probability to θ^*.

In general, however, we do not know P, $\{\phi_x\}_{x \in X}$, or the state of the Markov chain at each time step. We assume that all that is available to define θ_t are ϕ_t, ϕ_{t+1} and r_t. Instrumental variable methods allow us to solve the problem under these conditions. Let $\hat{\omega}_t = \phi_t - \gamma\phi_{t+1}$, and $\zeta_t = \gamma \sum_{y \in X} P(x_t,y)\phi_y - \gamma\phi_{t+1}$. Then we can observe

$$\begin{aligned} \hat{\omega}_t &= \phi_t - \gamma\phi_{t+1} \\ &= (\phi_t - \gamma \sum_{y \in X} P(x_t,y)\phi_y) - (\gamma \sum_{y \in X} P(x_t,y)\phi_y - \gamma\phi_{t+1}) \\ &= \omega_t + \zeta_t, \end{aligned}$$

with $\omega_t = \phi_t - \gamma \sum_{y \in X} P(x_t,y)\phi_y$ and $\zeta_t = \gamma \sum_{y \in X} P(x_t,y)\phi_y - \gamma\phi_{t+1}$. We see, then, that the problem fits the errors-in-variables situation. Specifically, the following equation in the form of (7) is equivalent to the consistency condition (9):

$$r_t = (\phi_t - \gamma\phi_{t+1})'\theta^* - (\gamma \sum_{y \in X} P(x_t,y)\phi_y - \gamma\phi_{t+1})'\theta^* + (r_t - \bar{r}_t).$$

Following Section 5.1, we introduce an instrumental variable, ρ_t, to avoid the asymptotic bias introduced by errors-in-variables problems. The following lemma, which is proved in Appendix A, shows that $\rho_t = \phi_t$ is an instrumental variable because it is uncorrelated with the input observation noise, ζ_t, defined above:

```
1  Set t = 0.
2  repeat forever {
3      Set x_t to be a start state selected according to the probabilities given by S.
4      while x_t is not an absorbing state {
5          Let the state change from x_t to x_{t+1} according to the Markov chain
           transition probabilities.
6          Use (11) to define θ_t.
7          t = t + 1.
8      }
9  }
```

Figure 2. Trial–based LS TD for absorbing Markov chains. A trial is a sequence of states that starts at some start state, follows the Markov chain as it makes transitions, and ends at an absorbing state.

LEMMA 4 *For any Markov chain, if (1) x and y are states such that $P(x,y) > 0$; (2) $\zeta_{xy} = \gamma \sum_{z \in X} P(x,z)\phi_z - \gamma\phi_y$; (3) $\eta_{xy} = R(x,y) - r_x$, and (4) $\rho_x = \phi_x$, then (1) $Cor(\rho,\eta) = 0$; and (2) $Cor(\rho,\zeta) = 0$.*

Using ϕ_x as the instrumental variable, we rewrite (8) to obtain the LS TD algorithm:

$$\theta_t = \left[\frac{1}{t} \sum_{k=1}^{t} \phi_k(\phi_k - \gamma\phi_{k+1})' \right]^{-1} \left[\frac{1}{t} \sum_{k=1}^{t} \phi_k r_k \right]. \tag{11}$$

Figure 2 shows how (8) can be used as part of a trial–based algorithm to find the value function for an absorbing Markov chain. Figure 3 shows how (8) can be used as part of an algorithm to find the value function for an ergodic Markov chain. Learning takes place on–line in both algorithms, with parameter updates after every state transition. The parameter vector θ_t is not well defined when t is small since the matrix $\left[\frac{1}{t} \sum_{k=1}^{t} \phi_k(\phi_k - \gamma\phi_{k+1})' \right]$ is not invertible.

The LS TD algorithm has some similarity to an algorithm Werbos (Werbos, 1990) proposed as a linear version of his Heuristic Dynamic Programming (Lukes, et al., 1990, Werbos, 1987, Werbos, 1988, Werbos, 1992). However, Werbos' algorithm is not amenable to a recursive formulation, as is LS TD, and does not converge for arbitrary initial parameter vectors, as does LS TD. See ref. (Bradtke, 1994).

It remains to establish conditions under which LS TD converges to θ^*. According to Lemma 2, we must establish that $Cor(\rho,\omega)$ is finite and nonsingular. We take this up in the next section.

5.3. *Convergence of Algorithm LS TD*

In this section we consider the asymptotic performance of algorithm LS TD when used on–line to approximate the value functions of absorbing and ergodic Markov chains. The following lemma, proved in Appendix A, starts the analysis by expressing $\theta_{LSTD} \stackrel{\text{def}}{=} \lim_{t \to \infty} \theta_t$, the limiting estimate found by algorithm LS TD for θ^*, in a convenient form.

```
1  Set t = 0.
2  Select an arbitrary initial state, x_0.
3  repeat forever {
4      Let the state change from x_t to x_{t+1} according to the Markov chain transi-
       tion probabilities.
5      Use (11) to define θ_t.
6      t = t + 1.
7  }
```

Figure 3. LS TD for ergodic Markov chains.

LEMMA 5 *For any Markov chain, when (1) θ_t is found using algorithm LS TD; (2) each state $x \in X$ is visited infinitely often; (3) each state $x \in X$ is visited in the long run with probability 1 in proportion π_x; and (4) $[\Phi'\Pi(I - \gamma P)\Phi]$ is invertible, where Φ is the matrix of whose x–th row is ϕ_x, and Π is the diagonal matrix* $\mathrm{diag}(\pi)$, *then*

$$\theta_{\mathrm{LSTD}} = [\Phi'\Pi(I - \gamma P)\Phi]^{-1}[\Phi'\Pi\bar{r}]$$

with probability 1.

The key to using Lemma 5 lies in the definition of π_x: the proportion of time that the Markov chain is expected to spend over the long run in state x. Equivalently, π_x is the expected proportion of state transitions that take the Markov chain out of state x. For an ergodic Markov chain, π_x is the invariant, or steady–state, distribution associated with the stochastic matrix P (Kemeny & Snell, 1976). For an absorbing Markov chain, π_x is the expected number of visits out of state x during one transition sequence from a start state to a goal state (Kemeny & Snell, 1976). Since there are no transitions out of a goal state, $\pi_x = 0$ for all goal states. These definitions prepare the way for the following two theorems. Theorem 1 gives conditions under which LS TD as used in Figure 2 will cause θ_{LSTD} to converge with probability 1 to θ^* when applied to an absorbing Markov chain. Theorem 2 gives conditions under which LS TD as used in Figure 3 will cause θ_{LSTD} to converge with probability 1 to θ^* when applied to an ergodic Markov chain.

THEOREM 1 (CONVERGENCE OF LS TD FOR ABSORBING MARKOV CHAINS)
When using LS TD as described in Figure 2 to estimate the value function for an absorbing Markov chain, if (1) S is such that there are no inaccessible states; (2) $R(x, y) = 0$ whenever both $x, y \in \mathcal{T}$, the set of absorbing states; (3) the set of feature vectors representing the non-absorbing states, $\{\phi_x \mid x \in \mathcal{N}\}$, is linearly independent; (4) $\phi_x = 0$ for all $x \in \mathcal{T}$; (5) each ϕ_x is of dimension $m = |\mathcal{N}|$; and (6) $0 \leq \gamma \leq 1$; then θ^* is finite and the asymptotic parameter estimate found by algorithm LS TD, θ_{LSTD}, converges with probability 1 to θ^* as the number of trials (and state transitions) approaches infinity.

Different conditions are required in the absorbing and ergodic chain cases in order to meet the conditions of Lemma 5. The conditions required in Theorem 1 are generaliza-

tions of the conditions required for probability 1 convergence of TD(λ) for absorbing Markov chains. The conditions required in Theorem 2 are much less restrictive, though the discount factor γ must be less than 1 to ensure that the value function is finite.

THEOREM 2 (CONVERGENCE OF LS TD FOR ERGODIC MARKOV CHAINS)
When using LS TD as described in Figure 3 to estimate the value function for an ergodic Markov chain, if (1) the set of feature vectors representing the states, $\{\phi_x \mid x \in X\}$, is linearly independent; (2) each ϕ_x is of dimension $N = |X|$; (3) $0 < \gamma < 1$; then θ^* is finite and the asymptotic parameter estimate found by algorithm LS TD, θ_{LSTD}, converges with probability 1 to θ^* as the number of state transitions approaches infinity.

Theorems 1 and 2 provide convergence assurances for LS TD similar to those provided by Tsitsiklis (Tsitsiklis, 1993) and Watkins and Dayan (Watkins & Dayan, 1992) for the convergence of TD(0) using a lookup–table function approximator.

Proof of Theorem 1: Condition (1) implies that, with probability 1, as the total number of state transitions approaches infinity, the number of times each state $x \in X$ is visited approaches infinity. Since this is an absorbing chain, we have with probability 1 that the states are visited in proportion π as the number of trials approaches infinity. Therefore, by Lemma 5, we know that with probability 1

$$\theta_{\mathrm{LSTD}} = [\Phi'\Pi(I - \gamma P)\Phi]^{-1}[\Phi'\Pi r],$$

assuming that the inverse exists.

Conditions (3), (4), and (5) imply that Φ has rank m, with row x of Φ consisting of all zeros for all $x \in \mathcal{T}$. Condition (1) implies that Π has rank m. Row x of Π consists of all zeros, for all $x \in \mathcal{T}$. Φ has the property that if all rows corresponding to absorbing states are removed, the resulting submatrix is of dimensions $(m \times m)$ and has rank m. Call this submatrix A. Π has the property that if all rows and columns corresponding to absorbing states are removed, the resulting submatrix is of dimensions $(m \times m)$ and has rank m. Call this submatrix B. $(I - \gamma P)$ has the property that if all rows and columns corresponding to absorbing states are removed, the resulting submatrix is of dimensions $(m \times m)$ and has rank m (Kemeny & Snell, 1976). Call this submatrix C. It can be verified directly by performing the multiplications that $[\Phi'\Pi(I - \gamma P)\Phi] = [A'BCA]$. Therefore, $[\Phi'\Pi(I - \gamma P)\Phi]$ is of dimensions $(m \times m)$ and has rank m. Thus, it is invertible.

Now, (9) can be rewritten using matrix notation as

$$\bar{r} = (I - \gamma P)\Phi\theta^*. \tag{12}$$

This, together with conditions (2) and (6), implies that θ^* is finite. Finally, substituting (12) into the expression for θ_{LSTD} gives us

$$\begin{aligned}\theta_{\mathrm{LSTD}} &= [\Phi'\Pi(I - \gamma P)\Phi]^{-1}[\Phi'\Pi(I - \gamma P)\Phi]\theta^* \\ &= \theta^*.\end{aligned}$$

Thus, θ_{LSTD} converges to θ^* with probability 1. ■

Proof of Theorem 2: Since this is an ergodic chain, as t approaches infinity we have with probability 1 that the number of times each state $x \in X$ is visited approaches infinity. We also have with probability 1 that the states are visited in the long run in proportion π. Ergodicity implies that $\pi_x > 0$ for all $x \in X$. Therefore, Π is invertible. Condition (3) implies that $(I - \gamma P)$ is invertible. Conditions (1) and (2) imply that Φ is invertible. Therefore, by Lemma 5, we know that with probability 1

$$\theta_{\text{LSTD}} = [\Phi'\Pi(I - \gamma P)\Phi]^{-1} [\Phi'\Pi\bar{r}].$$

Condition (3) together with Equation (12) imply that θ^* is finite. And, as above, substituting (12) into the expression for θ_{LSTD} gives

$$\begin{aligned}\theta_{\text{LSTD}} &= [\Phi'\Pi(I - \gamma P)\Phi]^{-1} [\Phi'\Pi(I - \gamma P)\Phi]\,\theta^* \\ &= \theta^*.\end{aligned}$$

Thus, θ_{LSTD} converges to θ^* with probability 1. ■

5.4. *Algorithm RLS TD*

Algorithm LS TD requires the computation of a matrix inverse at each time step. This means that LS TD has a computational complexity of $O(m^3)$, assuming that the state representations are of length m. We can use Recursive Least–Squares (RLS) techniques (Goodwin & Sin, 1984, Ljung & Söderström, 1983, Young, 1984) to derive a modified algorithm, Recursive Least–Squares TD (RLS TD), with computational complexity of $O(m^2)$. The following equation set specifies algorithm RLS TD:

$$d_t = R_t - (\phi_t - \gamma\phi_{t+1})'\theta_{t-1} \tag{13}$$

$$C_t = C_{t-1} - \frac{C_{t-1}\phi_t(\phi_t - \gamma\phi_{t+1})'C_{t-1}}{1 + (\phi_t - \gamma\phi_{t+1})'C_{t-1}\phi_t} \tag{14}$$

$$\theta_t = \theta_{t-1} + \frac{C_{t-1}}{1 + (\phi_t - \gamma\phi_{t+1})'C_{t-1}\phi_t} d_t\phi_t. \tag{15}$$

Notice that (15) is the TD(0) learning rule for function approximators that are linear in the parameters, except that the scalar step–size parameter has been replaced by a gain matrix. The user of an RLS algorithm must specify θ_0 and C_0. C_t is the t^{th} sample estimate of $\frac{1}{t}\text{Cor}(\rho, \hat{\omega})^{-1}$, where ρ and $\hat{\omega}$ are defined as in Section 5.2. C_0^{-1} is typically chosen to be a diagonal matrix of the form βI, where β is some large positive constant. This ensures that C_0, the initial guess at the correlation matrix, is approximately 0, but is invertible and symmetric positive definite.

The convergence of RLS TD requires the same conditions as algorithm LS TD, plus one more. This is Condition A.1, or equivalently, Condition A.2:

Condition A.1: $\left[C_0^{-1} + \sum_{k=1}^{t} \rho_k\hat{\omega}_k'\right]$ must be non–singular for all times t.

Condition A.2: $[1 + \hat{\omega}_t'C_{t-1}\rho_t] \neq 0$ for all times t.

Under the assumption that the conditions A.1 and A.2 are maintained, we have that

$$C_t = \left[C_0^{-1} + \sum_{k=1}^{t} \rho_k \hat{\omega}_k' \right]^{-1}$$

and that

$$\begin{aligned} \theta_t &= \left[C_0^{-1} + \sum_{k=1}^{t} \rho_k \hat{\omega}_k' \right]^{-1} \left[C_0^{-1}\theta_0 + \sum_{k=1}^{t} \rho_k \psi_k \right] \\ &= \left[\frac{1}{t} C_0^{-1} + \frac{1}{t} \sum_{k=1}^{t} \rho_k \hat{\omega}_k' \right]^{-1} \left[\frac{1}{t} C_0^{-1}\theta_0 + \frac{1}{t} \sum_{k=1}^{t} \rho_k \psi_k \right]. \end{aligned}$$

If the conditions A.1 and A.2 are not met at some time t_0, then all computations made thereafter will be polluted by the indeterminate or infinite values produced at time t_0. The non–recursive algorithm LS TD does not have this problem because the computations made at any time step do not depend directly on the results of computations made at earlier time steps.

5.5. *Dependent or Extraneous Features*

The value function for a Markov chain satisfies the equation

$$V = \left[I - \gamma P \right]^{-1} \bar{r}.$$

When using a function approximator linear in the parameters, this means that the parameter vector θ must satisfy the linear equation

$$\Phi\theta = \left[I - \gamma P \right]^{-1} \bar{r}. \tag{16}$$

In this section, the rows of Φ consist only of the feature vectors representing the non–absorbing states, and V only includes the values for the non–absorbing states. This is not essential, but it makes the discussion much simpler. Let $n = |\mathcal{N}|$ be the number of non–absorbing states in the Markov chain. Matrix Φ has dimension $n \times m$, where m is the length of the feature vectors representing the states.

Now, suppose that $\text{rank}(\Phi) = m < n$. Dayan (Dayan, 1992) shows that in this case trial–based TD(λ) (Figure 1) converges to $\left[\Phi'\Pi(I - \gamma P)\Phi\right]^{-1}\left[\Phi'\Pi\bar{R}\right]$ for $\lambda = 0$. This is the same result we achieved in Lemma 5, since $m = \text{rank}(\Phi)$ if and only if $[\Phi'\Pi(I - \gamma P)\Phi]$ is invertible[2]. The proofs of Theorems 1 and 2 show convergence of θ_{LSTD} to $\left[\Phi'\Pi(I - \gamma P)\Phi\right]^{-1}\left[\Phi'\Pi\bar{R}\right]$ as a preliminary result. Thus, θ_{LSTD} converges for both absorbing and ergodic chains as long the assumptions of Lemma (5) are satisfied.

Suppose, on the other hand, that $\text{rank}(\Phi) = n < m$. This means that the state representations are linearly independent but contain extraneous features. Therefore, there are more adjustable parameters than there are constraints, and an infinite number of parameter vectors θ satisfy (16). The stochastic approximation algorithms TD(λ) and NTD(λ)

converge to some θ that satisfies (16). Which one they find depends on the order in which the states are visited. LS TD does not converge, since $\left[\frac{1}{t}\sum_{k=1}^{t}\phi_k(\phi_k - \gamma\phi_{k+1})'\right]$ is not invertible in this case. However, RLS TD converges to some θ that satisfies (16). In this case, too, the θ to which the algorithm converges depends on the order in which the states are visited.

6. Choosing an Algorithm

When TD(λ) and NTD(λ) algorithms are used with function approximators that are linear in the parameters, they involve $O(m)$ costs at each time step when measured either in terms of the number of basic computer operations, or in terms of memory requirements, where m is the length of the feature vectors representing the states. Algorithm LS TD's costs are $O(m^3)$ in time and $O(m^2)$ in space at each time step, while RLS TD's are $O(m^2)$ in both time and space. TD(λ) and NTD(λ) are clearly superior in terms of cost per time step. However, LS TD and RLS TD are more efficient estimators in the statistical sense. They extract more information from each additional observation. Therefore, we would expect LS TD and RLS TD to converge more rapidly than do TD(λ) and NTD(λ). The use of LS TD and RLS TD is justified, then, if the increased costs per time step are offset by increased convergence rate.

The performance of TD(λ) is sensitive to a number of interrelated factors that do not affect the performance of either LS TD or RLS TD. Convergence of TD(λ) can be dramatically slowed by a poor choice of the step–size (α) and trace (λ) parameters. The algorithm can become unstable if α is too large, causing θ_t to diverge. TD(λ) is also sensitive to the norms of the feature vectors representing the states. Judicious choice of α and λ can prevent instability, but at the price of decreased learning rate. The performance of TD(λ) is also sensitive to $\|\theta_0 - \theta^*\|$, the distance between θ^* and the initial estimate for θ^*. NTD(λ) is sensitive to these same factors, but normalization reduces the sensitivity. In contrast, algorithms LS TD and RLS TD are insensitive to all of these factors. Use of LS TD and RLS TD eliminates the possibility of poor performance due to unlucky choice of parameters.

The transient behavior of a learning algorithm is also important. TD(λ) and NTD(λ) remain stable (assuming that the step–size parameter is small enough) no matter what sequence of states is visited. This is not true for LS TD and RLS TD. If $C_t^{-1} = \left[C_0^{-1} + \sum_{k=1}^{t}\rho_k\bar{\omega}_k'\right]$ is ill–conditioned or singular for some time t, then the estimate θ_t can very far from θ^*. LS TD will recover from this transient event, and is assured of converging eventually to θ^*. The version of RLS TD described in Section 5.4 will not recover if C_t^{-1} is singular. It may or may not recover from an ill–conditioned C_t^{-1}, depending on the machine arithmetic. However, there are well–known techniques for protecting RLS algorithms from transient instability (Goodwin & Sin, 1984).

TD(λ), NTD(λ), and RLS TD have an advantage over LS TD in the case of extraneous features, as discussed in Section 5.5. TD(λ), NTD(λ), and RLS TD converge to the correct value function in this situation, while LS TD does not.

None of the factors discussed above makes a definitive case for one algorithm over another in all situations. The choice depends finally on the computational cost structure imposed on the user of these algorithms.

7. The TD Error Variance

One of the interesting characteristics of the TD error term,

$$e_{\mathrm{TD}}(\theta_{t-1}) = R_t + \gamma\phi'_{t+1}\theta_{t-1} - \phi'_t\theta_{t-1},$$

is that it does not go to zero as θ_t converges to θ^*, except in the trivial case of a deterministic Markov chain. This is readily verified by inspection of (10). We define the *TD error variance*, σ_{TD}, of a Markov chain as follows:

$$\begin{aligned}\sigma_{\mathrm{TD}} &= E\left\{e_{\mathrm{TD}}(\theta^*)^2\right\} \\ &= E\left\{\left[R_t + \gamma\phi'_{t+1}\theta^* - \phi'_t\theta^*\right]^2\right\} \\ &= \sum_{x\in X}\pi_x \sum_{y\in X} P(x,y)\left[R(x,y) + \gamma\phi'_y\theta^* - \phi'_x\theta^*\right]^2.\end{aligned}$$

σ_{TD} is the variance of the TD error term under the assumptions that θ_t has converged to θ^*, and that the states (and the corresponding TD errors) are sampled on–line by following a sample path of the Markov chain. σ_{TD} is a measure of the noise that cannot be removed from any of the TD learning rules (TD(λ), NTD(λ), LS TD, or RLS TD), even after parameter convergence. It seems reasonable to expect that experimental convergence rates depend on σ_{TD}.

8. Experiments

This section describes two experiments designed to demonstrate the advantage in convergence speed that can be gained through using least–squares techniques. Both experiments compare the performance of NTD(λ) with that of RLS TD in the on–line estimation of the value function of a randomly generated ergodic Markov chain, the first with five states and the second with fifty states (see Appendix B for the specification of the smaller of the Markov chains). The conditions of Theorem 2 are satisfied in these experiments, so that the lengths of the state representation vectors equal five and fifty respectively in the two experiments. In a preliminary series of experiments, not reported here, NTD(λ) always performed at least as well as TD(λ), while showing less sensitivity to the choice of parameters, such as initial step size. Appendix C describes the algorithm we used to set the step size parameters for NTD(λ). Figures 4, 5, and 6 show the experimental results.

The x-axis of Figure 4 measures the TD error variance of the test Markov chain, which was varied over five distinct values from $\sigma_{\mathrm{TD}} = 10^{-1}$ through $\sigma_{\mathrm{TD}} = 10^{3}$ by scaling the cost function R. The state transition probability function, P, and the state representations,

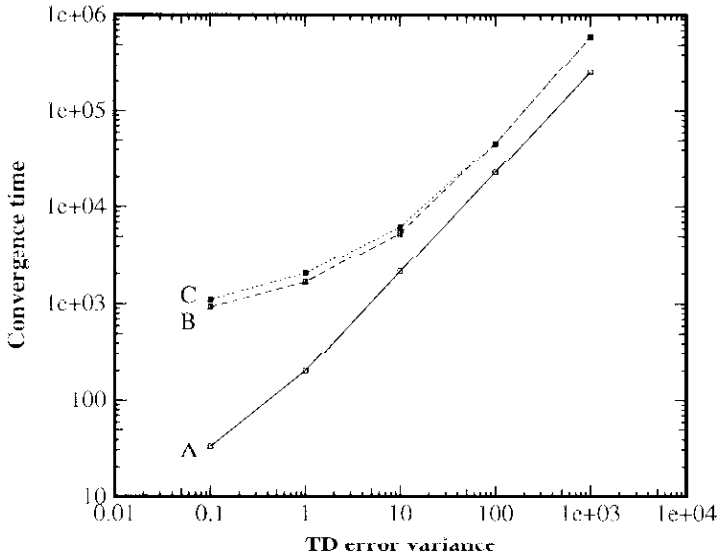

Figure 4. Comparison of RLS TD and NTD(λ) on a randomly generated 5 state ergodic Markov chain. The x axis measures the TD error variance of the test Markov chain, which was varied over five distinct values from $\sigma_{\mathrm{TD}} = 10^{-1}$ through $\sigma_{\mathrm{TD}} = 10^{3}$ by scaling the cost function R. The y-axis measures the average convergence time over 100 training runs of on–line learning. There was one time step counted for each interaction with the environment. The parameter vector was considered to have converged when the average of the error $\|\theta_t - \theta^*\|_\infty$ fell below 10^{-2} and stayed below this value thereafter. Graph A shows the performance of RLS TD. Graph B shows the performance of NTD(λ) where $\|\theta_0 - \theta^*\|_2 = 1$. Graph C shows the performance of NTD(λ) where $\|\theta_0 - \theta^*\|_2 = 2$.

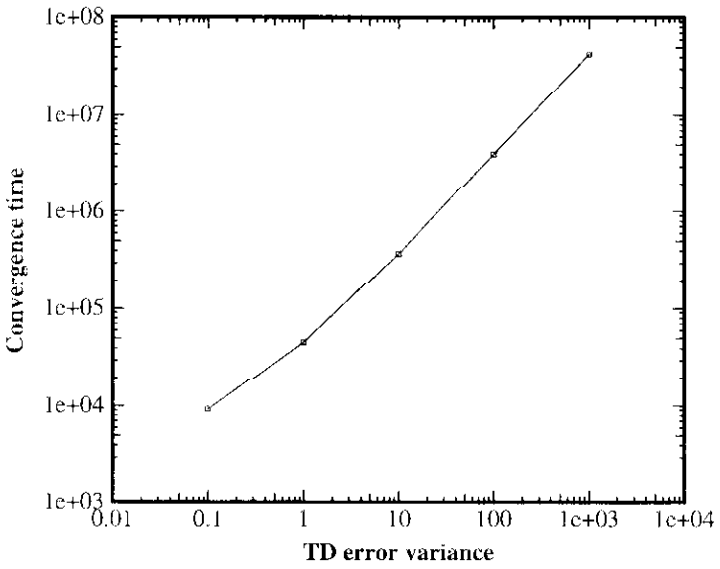

Figure 5. Performance of RLS TD on a randomly generated 50 state ergodic Markov chain. The x-axis measures the TD error variance of the test Markov chain, which was varied over five distinct values from $\sigma_{\text{TD}} = 10^{-1}$ through $\sigma_{\text{TD}} = 10^{3}$ by scaling the cost function R. The y-axis measures the average convergence time over 100 training runs of on-line learning. The parameter vector was considered to have converged when the average of the error $\|\theta_t - \theta^*\|_\infty$ fell below 10^{-2} and stayed below this value thereafter.

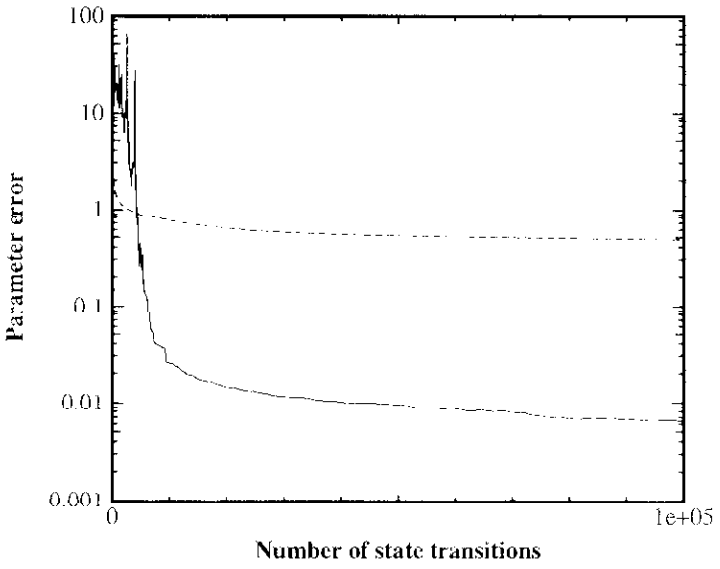

Figure 6. Learning curves for RLS TD (solid) and NTD(λ) (dashed) on a randomly generated 50 state ergodic Markov chain, for $\sigma_{\text{TD}} = 1$. The x-axis measures the number of state transitions, or the number of time steps. The y axis measures the average parameter error over 100 training runs of on line learning. The parameter error was defined as $\|\theta_t - \theta^*\|_\infty$.

Φ, were left unchanged. The y-axis of Figure 4 measures the average convergence time over 100 training runs of on-line learning. This was computed as follows. For each of the 100 training runs, $\|\theta_t - \theta^*\|_\infty$ was recorded at each time step (where $\|\cdot\|_\infty$ denotes the l_∞, or max, norm). These 100 error curves were averaged to produce the mean error curve. Finally, the mean error curve was inspected to find the time step t at which the average error fell below 10^{-2} and stayed below 10^{-2} for all the simulated times steps thereafter.

Graph A of Figure 4 shows the performance of RLS TD. The other two graphs show the performance of NTD(λ) given different initial values for θ_0. Graph B shows the performance of NTD(λ) when θ_0 was chosen so that $\|\theta_0 - \theta^*\|_2 = 1$ (where $\|\cdot\|_2$ denotes the l_2, or Euclidean, norm). Graph C shows the performance of NTD(λ) when θ_0 was chosen so that $\|\theta_0 - \theta^*\|_2 = 2$. One can see that the performance of NTD(λ) is sensitive to the distance of the initial parameter vector from θ^*. In contrast, the performance of RLS TD is not sensitive to this distance (θ_0 for Graph A was the same as that for Graph B). The performance of NTD(λ) is also sensitive to the settings of four control parameters: λ, α_0, c, and τ. The parameters c and τ govern the evolution of the sequence of step-size parameters (see Appendix C). A search for the best set of control parameters for NTD(λ) was performed for each experiment in an attempt to present NTD(λ) in the best light.[3] The control parameter ϵ (see Equation 4) was held constant at 1.0 for all experiments.

Figure 4 shows a number of things. First, RLS TD outperformed NTD(λ) at every level of σ_{TD}. RLS TD always converged at least twice as fast as NTD(λ), and did much better than that at lower levels of σ_{TD}. Next, we see that, at least for RLS TD, convergence time is a linear function of σ_{TD}: increase σ_{TD} by a factor of 10, and the convergence time can be expected to increase by a factor of 10. This relationship is less clear for NTD(λ), although the curves seem to follow the same rule for larger σ_{TD}. It appears that the effect of the initial distance from θ to θ^*, $\|\theta_0 - \theta^*\|_2$, is significant when σ_{TD} is small but becomes less important, and is finally eliminated, as σ_{TD} increases.

Figures 5 and 6 present the results of repeating the experiment described above for a randomly generated ergodic Markov chain with fifty states. Each state of this larger Markov chain has a possible transition to five other states, on average. Figure 5 shows that the convergence rates for RLS TD follow the same pattern seen in Figure 4: the convergence time rises at the same rate σ_{TD} rises. We attempted to experimentally test the convergence times of NTD(λ) on this problem as we did on the smaller problem. However, we were unable to achieve convergence to the criterion ($\|\theta_t - \theta^*\|_\infty < 10^{-2}$) for any value of σ_{TD} or any selection of the parameters λ, α_0, c, and τ. Figure 6 compares the learning curves generated by RLS TD and NTD(λ) for $\sigma_{\text{TD}} = 1$. The parameters governing the behavior of NTD(λ) were the best we could find. After some initial transients, RLS TD settles very rapidly toward convergence, while NTD(λ) settles very slowly toward convergence, making almost no progress for tens of thousands of time steps. These results indicate that the relative advantage of using the RLS TD algorithm may actually improve as the size of the problem grows, despite the order $O(m^2)$ cost required by RLS TD at each time step.

The results shown in Figures 4, 5, and 6 suggest that the use of RLS TD instead of TD(λ) or NTD(λ) is easily justified. RLS TD's costs per time step are an order of $m = |X|$ more expensive in both time and space than the costs for TD(λ) or NTD(λ). However, in the example problems, RLS TD always converged significantly faster than TD(λ) or NTD(λ), and was at least an order of m faster for smaller σ_{TD}. RLS TD has the significant additional advantage that it has no control parameters that have to be adjusted. In contrast, it required a very extensive search to select settings of the control parameters α_0, c, and τ of NTD(λ) to show this algorithm in a good light.

9. Conclusion

We presented three new TD learning algorithms, NTD(λ), LS TD, and RLS TD, and we proved probability 1 convergence for these algorithms under appropriate conditions. These algorithms have a number of advantages over previously proposed TD learning algorithms. NTD(λ) is a normalized version of TD(λ) used with a linear–in–the–parameters function approximator. The normalization serves to reduce the algorithm's sensitivity to the choice of control parameters. LS TD is a Least–Squares algorithm for finding the value function of a Markov chain. Although LS TD is more expensive per time step than the algorithms TD(λ) and NTD(λ), it converges more rapidly and has no control parameters that need to be set, reducing the chances for poor performance. RLS TD is a recursive version of LS TD.

We also defined the TD error variance of a Markov chain, σ_{TD}. σ_{TD} is a measure of the noise that is inherent in any TD learning algorithm, even after the parameters have converged to θ^*. Based on our experiments, we conjecture that the convergence rate of a TD algorithm depends linearly on σ_{TD}(Figure 4). This relationship is very clear for RLS TD, but also seems to hold for NTD(λ) for larger σ_{TD}.

The theorems concerning convergence of LS TD (and RLS TD) can be generalized in at least two ways. First, the immediate rewards can be random variables instead of constants. $R(x, y)$ would then designate the *expected* reward of a transition from state x to state y. The second change involves the way the states (and state transitions) are sampled. Throughout this chapter we have assumed that the states are visited along sample paths of the Markov chain. This need not be the case. All that is necessary is that there is some limiting distribution, π, of the states selected for update, such that $\pi_x > 0$ for all states x.

One of the goals of using a parameterized function approximator (of which the linear–in–the–parameters approximators considered in this article are the simplest examples) is to store the value function more compactly than it could be stored in a lookup table. Function approximators that are linear in the parameters do not achieve this goal if the feature vectors representing the states are linearly independent, since in this case they use the same amount of memory as a lookup table. However, we believe that the results presented here and elsewhere on the performance of TD algorithms with function approximators that are linear in the parameters are first steps toward understanding the performance of TD algorithms using more compact representations.

Appendix A

Proofs of Lemmas

In preparation for the proofs of Lemmas 3 and 4, we first examine the sum $\sum_{y \in X} P(x,y)(R(x,y) - \bar{r}_x)$ for an arbitrary state x:

$$\begin{aligned}\sum_{y \in X} P(x,y)(R(x,y) - \bar{r}_x) &= \sum_{y \in X} P(x,y)R(x,y) - \sum_{y \in X} P(x,y)\bar{r}_x \\ &= \sum_{y \in X} P(x,y)R(x,y) - \bar{r}_x \\ &= r_x - r_x \\ &= 0.\end{aligned}$$

Proof of Lemma 3: The result in the preceding paragraph leads directly to a proof that $E\{\eta\} = 0$:

$$\begin{aligned}E\{\eta\} &= E\{R(x,y) - r_x\} \\ &= \sum_{x \in X} \pi_x \sum_{y \in X} P(x,y)(R(x,y) - \bar{r}_x) \\ &= \sum_{x \in X} \pi_x \cdot 0 \\ &= 0,\end{aligned}$$

and to a proof that $\mathrm{Cor}(\omega, \eta) = 0$:

$$\begin{aligned}\mathrm{Cor}(\omega,\eta) &= E\{\omega\eta\} \\ &= \sum_{x \in X} \pi_x \sum_{y \in X} P(x,y)[\omega_x \eta_{xy}] \\ &= \sum_{x \in X} \pi_x \sum_{y \in X} P(x,y)\omega_x(R(x,y) - \bar{r}_x) \\ &= \sum_{x \in X} \pi_x \omega_x \sum_{y \in X} P(x,y)(R(x,y) - \bar{r}_x) \\ &= \sum_{x \in X} \pi_x \omega_x \cdot 0 \\ &= 0.\end{aligned}$$

■

Proof of Lemma 4: First we consider $\mathrm{Cor}(\rho, \eta)$:

$$\mathrm{Cor}(\rho,\eta) = E\{\rho\eta'\}$$

$$
\begin{aligned}
&= \sum_{x \in X} \pi_x \sum_{y \in X} P(x,y) \left[\rho_x \eta'_{xy}\right] \\
&= \sum_{x \in X} \pi_x \sum_{y \in X} P(x,y) \phi_x (R(x,y) - \bar{r}_x)' \\
&= \sum_{x \in X} \pi_x \phi_x \sum_{y \in X} P(x,y)(R(x,y) - \bar{r}_x)' \\
&= \sum_{x \in X} \pi_x \phi_x \cdot 0 \\
&= 0.
\end{aligned}
$$

Now for $\mathrm{Cor}(\rho, \zeta)$:

$$
\begin{aligned}
\mathrm{Cor}(\rho, \zeta) &= E\{\rho \zeta'\} \\
&= \sum_{x \in X} \pi_x \sum_{y \in X} P(x,y) \left[\rho_x \zeta'_{xy}\right] \\
&= \sum_{x \in X} \pi_x \sum_{y \in X} P(x,y) \phi_x (\gamma \sum_{z \in X} P(x,z) \phi_z - \gamma \phi_y)' \\
&= \sum_{x \in X} \pi_x \phi_x \sum_{y \in X} P(x,y)(\gamma \sum_{z \in X} P(x,z) \phi'_z) - \\
&\quad \sum_{x \in X} \pi_x \phi_x \sum_{y \in X} P(x,y) \gamma \phi'_y \\
&= \sum_{x \in X} \pi_x \phi_x \gamma \sum_{z \in X} P(x,z) \phi'_z - \sum_{x \in X} \pi_x \phi_x \gamma \sum_{y \in X} P(x,y) \phi'_y \\
&= 0.
\end{aligned}
$$

■

Proof of Lemma 5: Equation 11 (repeated here) gives us the t^{th} estimate found by algorithm LS TD for θ^*:

$$
\theta_t = \left[\frac{1}{t} \sum_{k=1}^{t} \phi_k (\phi_k - \gamma \phi_{k+1})'\right]^{-1} \left[\frac{1}{t} \sum_{k=1}^{t} \phi_k R_k\right].
$$

As t grows we have by condition (2) that the *sampled* transition probabilities between each pair of states approaches the *true* transition probabilities, P, with probability 1. We also have by condition (3) that each state $x \in X$ is visited in the proportion π_x with probability 1. Therefore, given condition (4) we can express the limiting estimate found by algorithm LS TD, θ_{LSTD}, as

$$
\theta_{\mathrm{LSTD}} = \lim_{t \to \infty} \theta_t
$$

$$
\begin{aligned}
&= \lim_{t\to\infty}\left[\frac{1}{t}\sum_{k=1}^{t}\phi_k(\phi_k-\gamma\phi_{k+1})'\right]^{-1}\left[\frac{1}{t}\sum_{k=1}^{t}\phi_k R_k\right] \\
&= \left[\lim_{t\to\infty}\frac{1}{t}\sum_{k=1}^{t}\phi_k(\phi_k-\gamma\phi_{k+1})'\right]^{-1}\left[\lim_{t\to\infty}\frac{1}{t}\sum_{k=1}^{t}\phi_k R_k\right] \\
&= \left[\sum_x \pi_x \sum_y P(x,y)\phi_x(\phi_x-\gamma\phi_y)'\right]^{-1}\left[\sum_x \pi_x\phi_x\sum_y P(x,y)R(x,y)\right] \\
&= \left[\sum_x \pi_x \sum_y P(x,y)\phi_x(\phi_x-\gamma\phi_y)'\right]^{-1}\left[\sum_x \pi_x \bar{R}_x\phi_x\right] \\
&= \left[\Phi'\Pi(I-\gamma P)\Phi\right]^{-1}\left[\Phi'\Pi\bar{R}\right].
\end{aligned}
$$

■

Appendix B

The Five–State Markov Chain Example

The transition probability function of the five–state Markov chain used in the experiments appears in matrix form as follows, where the entry in row i, column j is the probability of a transition from state i to state j (rounded to two decimal places):

$$
\begin{bmatrix}
0.42 & 0.13 & 0.14 & 0.03 & 0.28 \\
0.25 & 0.08 & 0.16 & 0.35 & 0.15 \\
0.08 & 0.20 & 0.33 & 0.17 & 0.22 \\
0.36 & 0.05 & 0.00 & 0.51 & 0.07 \\
0.17 & 0.24 & 0.19 & 0.18 & 0.22
\end{bmatrix}
$$

The feature vectors representing the states (rounded to two decimal places) are listed as the rows of the following matrix Φ:

$$
\begin{bmatrix}
74.29 & 34.61 & 73.48 & 53.29 & 7.79 \\
61.60 & 48.07 & 34.68 & 36.19 & 82.02 \\
97.00 & 4.88 & 8.51 & 87.89 & 5.17 \\
41.10 & 40.13 & 64.63 & 92.67 & 31.09 \\
7.76 & 79.82 & 43.78 & 8.56 & 61.11
\end{bmatrix}
$$

The immediate rewards were specified by the following matrix (which has been rounded to two decimal places), where the entry in row i, column j determines the immediate reward for a transition from state i to state j. The matrix was scaled to produce the different TD error variance values used in the experiments. The relative sizes of the immediate rewards remained the same.

$$R = \begin{bmatrix} 104.66 & 29.69 & 82.36 & 37.49 & 68.82 \\ 75.86 & 29.24 & 100.37 & 0.31 & 35.99 \\ 57.68 & 65.66 & 56.95 & 100.44 & 47.63 \\ 96.23 & 14.01 & 0.88 & 89.77 & 66.77 \\ 70.35 & 23.69 & 73.41 & 70.70 & 85.41 \end{bmatrix}$$

Appendix C

Selecting Step–Size Parameters

The convergence theorem for NTD(0) (Bradtke, 1994) requires a separate step–size parameter, $\alpha(x)$, for each state x, that satisfies the Robbins and Monro (Robbins & Monro, 1951) criteria

$$\sum_{k=1}^{\infty} \alpha_k(x) = \infty \qquad \text{and} \qquad \sum_{k=1}^{\infty} \alpha_k(x)^2 < \infty$$

with probability 1, where $\alpha_k(x)$ is the step–size parameter for the k–th visitation of state x. Instead of a separate step–size parameter for each state, we used a single parameter α_t, which we decreased at every time step. For each state x there is a corresponding subsequence $\{\alpha_t\}_x$ that is used to update the value function when x is visited. We conjecture that if the original sequence $\{\alpha_t\}$ satisfies the Robbins and Monro criteria, then these subsequences also satisfy the criteria, with probability 1. The overall convergence rate may be decreased by use of a single step–size parameter since each subsequence will contain fewer large step sizes.

The step–size parameter sequence $\{\alpha_t\}$ was generated using the "search then converge" algorithm described by Darken, Chang, and Moody (Darken, et al., 1992):

$$\alpha_t = \alpha_0 \frac{1 + \frac{c}{\alpha_0}\frac{t}{\tau}}{1 + \frac{c}{\alpha_0}\frac{t}{\tau} + \tau\frac{t^2}{\tau^2}}.$$

The choice of parameters α_0, c, and τ determines the transition of learning from "search mode" to "converge mode". Search mode describes the time during which $t \ll \tau$. Converge mode describes the time during which $t \gg \tau$. α_t is nearly constant in search mode, while $\alpha_t \simeq \frac{c}{t}$ in converge mode. The ideal choice of step–size parameters moves θ_t as quickly as possible into the vicinity of θ^* during search mode, and then settles into converge mode.

Notes

1. If the set of feature vectors is linearly independent, then there exist parameter values such that *any* real valued function of X can be approximated with zero error by a function approximator linear in the parameters. Using terminology from adaptive control (Goodwin & Sin, 1984), this situation is said to satisfy the *exact matching* condition for arbitrary real–valued functions of X.

2. m can not be less than rank(Φ). If $m >$ rank(Φ), then $[\Phi' \Pi (I - \gamma P)\Phi]$ is an $(m \times m)$ matrix with rank less than m. It is therefore not invertible.
3. The search for the best settings for λ, α_0, c, and τ was the limiting factor on the size of the state space for this experiment.

References

Anderson, C. W., (1988). Strategy learning with multilayer connectionist representations. Technical Report 87-509.3, GTE Laboratories Incorporated, Computer and Intelligent Systems Laboratory, 40 Sylvan Road, Waltham, MA 02254.

Barto, A. G., Sutton, R. S. & Anderson, C. W. (1983). Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:835–846.

Bradtke, S. J., (1994). *Incremental Dynamic Programming for On-Line Adaptive Optimal Control*. PhD thesis, University of Massachusetts, Computer Science Dept. Technical Report 94-62.

Darken, C. Chang, J. & Moody, J., (1992) Learning rate schedules for faster stochastic gradient search. In *Neural Networks for Signal Processing 2 — Proceedings of the 1992 IEEE Workshop*. IEEE Press.

Dayan, P., (1992). The convergence of TD(λ) for general λ. *Machine Learning*, 8:341–362.

Dayan, P. & Sejnowski, T.J., (1994). TD(λ): Convergence with probability 1. *Machine Learning*.

Goodwin, G.C. & Sin, K.S., (1984). *Adaptive Filtering Prediction and Control*. Prentice-Hall, Englewood Cliffs, N.J.

Jaakkola, T, Jordan, M.I. & Singh, S.P., (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6).

Kemeny, J.G. & Snell, J.L., (1976). *Finite Markov Chains*. Springer-Verlag, New York.

Ljung, L. & Söderström, T., (1983). *Theory and Practice of Recursive Identification*. MIT Press, Cambridge, MA.

Lukes, G., Thompson, B. & Werbos, P., (1990). Expectation driven learning with an associative memory. In *Proceedings of the International Joint Conference on Neural Networks*, pages I:521–524.

Robbins, H & Monro, S., (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407.

Söderström, T. & Stoica, P.G., (1983). *Instrumental Variable Methods for System Identification*. Springer-Verlag, Berlin.

Sutton, A.S., (1984). *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, Department of Computer and Information Science, University of Massachusetts at Amherst, Amherst, MA 01003.

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

Tesauro, G.J., (1992). Practical issues in temporal difference learning. *Machine Learning*, 8(3/4):257–277.

Tsitsiklis, J.N., (1993). Asynchronous stochastic approximation and Q-learning. Technical Report LIDS-P-2172, Laboratory for Information and Decision Systems, MIT, Cambridge, MA.

Watkins, C. J. C. H., (1989). *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England.

Watkins, C. J. C. H. & Dayan, P., (1992). Q-learning. *Machine Learning*, 8(3/4):257–277, May 1992.

Werbos, P.J., (1987). Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(1):7–20.

Werbos, P.J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356, 1988.

Werbos, P.J., (1990). Consistency of HDP applied to a simple reinforcement learning problem. *Neural Networks*, 3(2):179–190.

Werbos, P.J., (1992). Approximate dynamic programming for real time control and neural modeling. In D. A. White and D. A. Sofge, editors, *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, pages 493–525. Van Nostrand Reinhold, New York.

Young, P., (1984). *Recursive Estimation and Time-series Analysis*. Springer–Verlag.

Received November 10, 1994
Accepted March 10, 1995
Final Manuscript October 4, 1995