On the Worst-case Analysis of Temporal-difference Learning Algorithms*

Robert E. Schapire†
Manfred K. Warmuth‡

UCSC-CRL-94-42
October 27, 1994

Baskin Center for
Computer Engineering & Information Sciences
University of California, Santa Cruz
Santa Cruz, CA 95064 USA

ABSTRACT

We study the worst-case behavior of a family of learning algorithms based on Sutton's method of temporal differences. In our on-line learning framework, learning takes place in a sequence of trials, and the goal of the learning algorithm is to estimate a discounted sum of all the reinforcements that will be received in the future. In this setting, we are able to prove general upper bounds on the performance of a slightly modified version of Sutton's so-called TD(λ) algorithm. These bounds are stated in terms of the performance of the best linear predictor on the given training sequence, and are proved without making any statistical assumptions of any kind about the process producing the learner's observed training sequence. We also prove lower bounds on the performance of any algorithm for this learning problem, and give a similar analysis of the closely related problem of learning to predict in a model in which the learner must produce predictions for a whole batch of observations before receiving reinforcement.

---

*A preliminary extended abstract of this paper appeared in *Machine Learning: Proceedings of the Eleventh International Conference*, 1994.

†Address: AT&T Bell Laboratories, 600 Mountain Avenue, Room 2A-424, Murray Hill, NJ 07974. Email address: schapire@research.att.com

‡Address: Computer and Information Sciences, University of California, Santa Cruz, CA 95064. Email address: manfred@cse.ucsc.edu

1 Introduction

We study the following prediction problem: At each point in time $t = 1, 2, \ldots$, a learning agent makes an observation about the current state of its environment, which is summarized by a real vector $\mathbf{x}_t \in \mathbb{R}^N$. After having made this observation, the learning agent receives some kind of feedback from its environment, which is summarized by a real number r_t. The goal of the learning agent is to learn to predict the future feedback that is likely to follow given the current observation vector $\mathbf{x}_t$.

For instance, r_t might be the price of a stock, and $\mathbf{x}_t$ the current market conditions. The learner's goal is then to predict the future price of the stock. In such a situation, we would likely be interested in the price of the stock not only for tomorrow, but for some time into the future. For instance, we might be interested in the average price of the stock over the next year. Or, rather than cutting off the average after a fixed amount of time, we might ask that the learner predict a weighted (or discounted) sum of the stock prices on each day in the future, giving lesser weight to days farther in the future.

In this paper, following Sutton [6], we take up the study of this latter type of prediction problem. More precisely, we study the problem of learning to predict the discounted sum

$$y_t := \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{1}$$

where $\gamma \in [0, 1)$ is some fixed constant called the *discount rate parameter*.

At each time step t, after receiving the instance vector $\mathbf{x}_t$ and prior to receiving the reinforcement signal r_t, we ask that the learning algorithm make a prediction $\hat{y}_t$ of the value of y_t. We measure the performance of the learning algorithm in terms of the discrepancy between $\hat{y}_t$ and y_t. There are many ways of measuring this discrepancy; in this paper, we use the quadratic loss function. That is, we define the *loss* of the learning algorithm at time t to be $(\hat{y}_t - y_t)^2$, and the loss for an entire sequence of predictions is just the sum of the losses at each trial. Thus, the goal of the learning algorithm is to minimize its loss over a sequence of observation/feedback trials.

We study the worst-case behavior of a family of learning algorithms based on Sutton's *method of temporal differences* [6]. Specifically, we analyze (a slightly modified version of) Sutton's so-called TD(λ) algorithm in a worst-case framework that makes no statistical assumptions of any kind. All previous analyses [2, 3, 4, 6, 7] of TD(λ) have relied heavily on stochastic assumptions about the nature of the environment that is generating the data observed by the learner; for instance, the learner's environment is often modeled by a Markov process.

The primary contribution of our paper is to introduce a method of worst-case analysis to the area of temporal-difference learning. We present upper bounds on the loss incurred by our temporal-difference learning algorithm (denoted by TD$^*(\lambda)$) which hold even when the sequence of observations $\mathbf{x}_t$ and reinforcement signals r_t is arbitrary.

To make our bounds meaningful in such an adversarial setting, we compare the performance of the learning algorithm to the loss that would be incurred by the best prediction function among a family of prediction functions; in this paper, this class will always be the set of linear prediction functions. More precisely, for any vector $\mathbf{u} \in \mathbb{R}^N$, let

$$L^{\ell}(\mathbf{u}, S) := \sum_{t=1}^{\ell} (\mathbf{u} \cdot \mathbf{x}_t - y_t)^2$$

denote the loss of vector $\mathbf{u}$ on the first ℓ trials of training sequence S. That is, $L^\ell(\mathbf{u}, S)$ is the loss that would be incurred by a prediction function that predicts $\mathbf{u} \cdot \mathbf{x}_t$ on each observation vector $\mathbf{x}_t$.

We compare the performance of our learning algorithms to the performance of the best vector $\mathbf{u}$ (of bounded norm) that minimizes the loss on the given sequence. For example, we prove below that, for any training sequence S, the loss on the first ℓ trials of TD*(1) is at most[1]

$$\min_{\substack{||\mathbf{u}|| \le U \\ L^\ell(\mathbf{u},S) \le K}} \left(L^\ell(\mathbf{u}, S) + 2\sqrt{K} U X c_\gamma + ||\mathbf{u}||^2 X^2 c_\gamma^2 \right) \tag{2}$$

where $c_\gamma = (1+\gamma)/(1-\gamma)$. (Here, U, X and K are parameters that are used to "tune" the algorithm's "learning rate:" specifically, it is assumed that $||\mathbf{x}_t|| \le X$, and that $\min\{L^\ell(\mathbf{u}, S) : ||\mathbf{u}|| \le U\} \le K$. Various methods are known for guessing these parameters when they are unknown; see, for instance, Cesa-Bianchi, Long and Warmuth's paper [1].) Thus, TD*(1) will perform reasonably well, provided that there exists some linear predictor $\mathbf{u}$ that gives a good fit to the training sequence.

To better understand bounds such as those given in equation (2), it is often helpful to consider the average per-trial loss that is guaranteed by the bound. Suppose for the moment, as is likely to be the case in practice, that U, X and γ are fixed, and that K grows linearly with the number of trials ℓ, so that $K = O(\ell)$. Then equation (2) implies that the average per-trial loss of TD*(1) (i.e., the total cumulative loss of TD*(1) divided by the number of trials ℓ) is at most

$$\min_{\substack{||\mathbf{u}|| \le U \\ L^\ell(\mathbf{u},S) \le K}} \left(\frac{L^\ell(\mathbf{u}, S)}{\ell} + O\left(\frac{1}{\sqrt{\ell}}\right) \right).$$

In other words, as the number of trials ℓ becomes large, the average per-trial loss of TD*(1) rapidly approaches the average loss of the best vector $\mathbf{u}$. Furthermore, the rate of convergence is given explicitly as $O(1/\sqrt{\ell})$.

Note that the above result, like all the others presented in this paper, provides a characterization of the learner's performance after only a *finite* number of time steps. In contrast, most previous work on TD(λ) has focused on its asymptotic performance. Moreover, previous researchers have focused on the convergence of the learner's hypothesis to a "true" or "optimal" model of the world. We, on the other hand, take the view that the learner's one and only goal is to make good predictions, and we therefore measure the learner's performance entirely by the quality of its predictions.

The upper bound given in equation (2) on the performance of TD*(1) is derived from a more general result we prove on the worst-case performance of TD*(λ) for general λ. Our bounds for the special case when $\lambda = 0$ or $\lambda = 1$ can be stated in closed form. The proof techniques used in this paper are similar but more general than those used by Cesa-Bianchi, Long and Warmuth [1] in their analysis of the Widrow-Hoff algorithm (corresponding to the case that $\gamma = 0$).

[1] In this paper we only use one vector norm, the L_2-norm: $||\mathbf{u}|| = \sqrt{\sum_{i=1}^N u_i^2}$.

Note that $\min\{L^\ell(\mathbf{u}, S) : \mathbf{u} \in \mathbb{R}^N\}$ is the best an arbitrary linear model can do that knows all $y_1 \cdots y_\ell$ ahead of time. If the on-line learner were given y_t at the end of trial t (i.e., if $\gamma = 0$), then the Widrow-Hoff algorithm would achieve a worst case bound of

$$\min_{\substack{||\mathbf{u}|| \leq U \\ L^\ell(\mathbf{u},S) \leq K}} \left(L^\ell(\mathbf{u}, S) + 2\sqrt{K} U X + ||\mathbf{u}||^2 X^2 \right)$$

(matching the bound in equation (2) with γ set to 0). However, in our model, the learner is given only the reinforcements r_t, even though its goal is to accurately estimate the infinite sum y_t given in equation (1). Intuitively, as γ gets larger, this task becomes more difficult since the learner must make predictions about events farther and farther into the future. All of our worst-case loss bounds depend explicitly on γ and, not surprisingly, these bounds typically tend to infinity or become vacuous as γ approaches 1. Thus, our bounds quantify the price one has to pay for giving the learner successively less information.

In addition to these upper bounds, we prove a general lower bound on the loss of *any* algorithm for this prediction problem. Such a lower bound may be helpful in determining what kind of worst-case bounds can feasibly be proved. None of our upper bounds match the lower bound; it is an open question whether this remaining gap can be closed (this is possible in certain special cases, such as when $\gamma = 0$).

Finally, we consider a slightly different, but closely related learning model in which the learner is given a whole batch of instances at once and the task is to give a prediction for all instances before an outcome is received for each instance in the batch. The loss in a trial t is $||\hat{\mathbf{y}}_t - \mathbf{y}_t||^2$, where $\hat{\mathbf{y}}_t$ is the vector of predictions and $\mathbf{y}_t$ the vector of outcomes. Again, the goal is to minimize the additional total loss summed over all trials in excess of the total loss of the best linear predictor (of bounded norm).

In this batch model all instances count equally and the exact outcome for each instance is received at the end of each batch. A special case of this model is when the algorithm has to make predictions on a whole batch of instances before receiving the *same* outcome for all of them (a case studied by Sutton [6]).

We again prove worst-case bounds for this model (extending Cesa-Bianchi, Long and Warmuth's previous analysis [1] for the noise-free case). We also prove matching lower bounds for this very general model, thus proving that our upper bounds are the optimal worst-case bounds.

The paper is outlined as follows. Section 2 describes the on-line model for temporal difference learning. Section 3 gives Sutton's original temporal difference learning algorithm TD(λ) and introduces our new algorithm TD*(λ). Section 4 contains the worst-case loss bounds for the new algorithm, followed by Section 5 containing a lower bound for the on-line model. We present contains our results for the batch model in Section 6. Finally, we discuss the merits of the method of worst-case analysis in Section 7.

2 The prediction model

In this section, we describe our on-line learning model. Throughout the paper, N denotes the dimension of the learning problem. Each *trial* t ($t = 1, 2, \ldots$) proceeds as follows:

1. The learner receives instance vector $\mathbf{x}_t \in \mathbb{R}^N$.
2. The learner is required to compute a prediction $\hat{y}_t \in \mathbb{R}$.

3. The learner receives a *reinforcement signal* $r_t \in \mathbb{R}$.

The goal of the learner is to predict not merely the next reinforcement signal, but rather a discounted sum of all of the reinforcements that will be received in the future. Specifically, the learner is trying to make its prediction $\hat{y}_t$ as close as possible to

$$y_t := \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

where $\gamma \in [0, 1)$ is a fixed parameter of the problem. (We will always assume that this infinite sum converges absolutely for all t.)

Note that if we multiply y_t by the constant $1 - \gamma$, we obtain a weighted average of all the future r_t's. Thus it might be more natural to use the variables $y'_t = y_t(1 - \gamma)$. (For instance, if all r_t equal r, then the modified variables y'_t all equal r as well.) However, for the sake of notational simplicity, we use the variables y_t instead (as was done by Sutton [6] and others).

The infinite sequence of pairs of instances $\mathbf{x}_t$ and reinforcement signals r_t is called a *training sequence* (usually denoted by S). The loss of the learner at trial t is $(y_t - \hat{y}_t)^2$, and the total loss of an algorithm A on the first ℓ trials is

$$L^\ell(A, S) := \sum_{t=1}^{\ell} (y_t - \hat{y}_t)^2.$$

Similarly, the total loss of a weight vector $\mathbf{u} \in \mathbb{R}^N$ on the first ℓ trials is defined to be

$$L^\ell(\mathbf{u}, S) := \sum_{t=1}^{\ell} (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2.$$

The purpose of this paper is to exhibit algorithms whose loss is guaranteed to be "not too much worse" than the loss of the *best* weight vector for the entire sequence. Thus, we would like to show that if there exists a weight vector $\mathbf{u}$ that fits the training sequence well, then the learner's predictions will also be reasonably good.

3 Temporal-difference algorithms

We focus now on a family of learning algorithms that are only a slight modification of those considered by Sutton [6]. Each of these algorithms is parameterized by a real number $\lambda \in [0, 1]$. For any sequence S and $t = 1, 2, \cdots$, let

$$\mathbf{X}_t^\lambda := \sum_{k=1}^{t} (\gamma\lambda)^{t-k} \mathbf{x}_k. \tag{3}$$

The learning algorithm $TD(\lambda)$ works by maintaining a weight vector $\mathbf{w}_t \in \mathbb{R}^N$. The initial weight vector $\mathbf{w}_1$ may be arbitrary, although in the simplest case $\mathbf{w}_1 = \mathbf{0}$. The weight vector $\mathbf{w}_t$ is then updated to the new weight vector $\mathbf{w}_{t+1}$ using the following update rule:

$$\mathbf{w}_{t+1} := \mathbf{w}_t + \eta_t(r_t + \gamma\hat{y}_{t+1} - \hat{y}_t)\mathbf{X}_t^\lambda. \tag{4}$$

Algorithm TD$^*(\lambda)$

Parameters: discount rate $\gamma \in [0,1)$
$\lambda \in [0,1]$
start vector $\mathbf{w}_1 \in \mathbb{R}^N$
method of computing learning rate η_t
Given: training sequence $\mathbf{x}_1, r_1, \mathbf{x}_2, r_2, \ldots$
Predict: $\hat{y}_1, \hat{y}_2, \ldots$
Procedure:

get $\mathbf{x}_1$
$\mathbf{X}_1^\lambda \leftarrow \mathbf{x}_1$
$\hat{y}_1 \leftarrow \mathbf{w}_1 \cdot \mathbf{X}_1^\lambda$
for $t = 1, 2, \ldots$
predict $\hat{y}_t$ $\quad (* \; \hat{y}_t = \mathbf{w}_t \cdot \mathbf{X}_t^\lambda - \sum_{k=1}^{t-1} (\gamma\lambda)^{t-k} \hat{y}_k \; *)$
get r_t
get $\mathbf{x}_{t+1}$
$\mathbf{X}_{t+1}^\lambda \leftarrow \mathbf{x}_{t+1} + (\gamma\lambda)\mathbf{X}_t^\lambda$
compute η_t
$\hat{y}_{t+1} \leftarrow \dfrac{\mathbf{w}_t \cdot \mathbf{x}_{t+1} + \eta_t (r_t - \hat{y}_t) \mathbf{X}_t^\lambda \cdot \mathbf{X}_{t+1}^\lambda}{1 - \eta_t \gamma \mathbf{X}_t^\lambda \cdot \mathbf{X}_{t+1}^\lambda}$
$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t (r_t + \gamma \hat{y}_{t+1} - \hat{y}_t) \mathbf{X}_t^\lambda$
end

Figure 1: Pseudocode for TD$^*(\lambda)$.

The constant η_t is called the *learning rate* on trial t. We will discuss later how to set the learning rates using prior knowledge about the training sequence.

In Sutton's original presentation of $TD(\lambda)$, and in most of the subsequent work on the algorithm, the prediction at each step is simply $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$. We, however, have found that a variant on this prediction rule leads to a simpler analysis, and, moreover, we were unable to obtain worst-case loss bounds for the original algorithm TD(λ) as strong as the bounds we prove for the new algorithm.

Our variant of TD(λ) uses the same update (4) for the weight vector as the original algorithm, but predicts as follows:

$$\begin{aligned} \hat{y}_t &:= \mathbf{w}_t \cdot \mathbf{x}_t + \sum_{k=1}^{t-1} (\gamma\lambda)^{t-k} (\mathbf{w}_t \cdot \mathbf{x}_k - \hat{y}_k) \\ &= \mathbf{w}_t \cdot \mathbf{X}_t^\lambda - \sum_{k=1}^{t-1} (\gamma\lambda)^{t-k} \hat{y}_k . \end{aligned} \tag{5}$$

This new algorithm, which we call TD$^*(\lambda)$, is summarized in Figure 1.

The rule (4) for updating $\mathbf{w}_{t+1}$ has $\mathbf{w}_{t+1}$ implicit in $\hat{y}_{t+1}$, so at first it seems impossible to do this update rule. However, by multiplying equation (4) by $\mathbf{X}_{t+1}^\lambda$, one can first solve for $\hat{y}_{t+1}$ and then compute $\mathbf{w}_{t+1}$. Specifically, this gives a solution for $\hat{y}_{t+1}$ of

$$
\begin{aligned}
&\frac{(\mathbf{w}_t + \eta_t(r_t - \hat{y}_t)\mathbf{X}_t^\lambda) \cdot \mathbf{X}_{t+1}^\lambda - \sum_{k=1}^{t} (\gamma\lambda)^{t+1-k} \hat{y}_k}{1 - \eta_t \gamma \mathbf{X}_t^\lambda \cdot \mathbf{X}_{t+1}^\lambda} \\
= \quad &\frac{(\mathbf{w}_t + \eta_t(r_t - \hat{y}_t)\mathbf{X}_t^\lambda) \cdot \mathbf{X}_{t+1}^\lambda - (\gamma\lambda)\mathbf{w}_t \cdot \mathbf{X}_t^\lambda}{1 - \eta_t \gamma \mathbf{X}_t^\lambda \cdot \mathbf{X}_{t+1}^\lambda} \\
= \quad &\frac{\mathbf{w}_t \cdot \mathbf{x}_{t+1} + \eta_t(r_t - \hat{y}_t)\mathbf{X}_t^\lambda \cdot \mathbf{X}_{t+1}^\lambda}{1 - \eta_t \gamma \mathbf{X}_t^\lambda \cdot \mathbf{X}_{t+1}^\lambda}
\end{aligned}
$$

where, in the first equality, we assume inductively that equation (5) holds at trial t. Thus, we can solve successfully for $\hat{y}_{t+1}$ provided that $\gamma\eta_t \mathbf{X}_t^\lambda \cdot \mathbf{X}_{t+1}^\lambda \neq 1$, as will be the case for all the values of η_t we consider. Also, note that $\hat{y}_{t+1}$ is computed after the instance $\mathbf{x}_{t+1}$ is received but before the reinforcement r_{t+1} is available (see Figure 1 for details).

Note that for the prediction $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$ of TD(λ),

$$
\begin{aligned}
\nabla_{\mathbf{w}_t}(y_t - \hat{y}_t)^2 &= -2\eta_t(y_t - \hat{y}_t)\mathbf{x}_t \\
&\approx -2\eta_t(r_t + \gamma\hat{y}_{t+1} - \hat{y}_t)\mathbf{x}_t.
\end{aligned}
$$

(Since $y_t = r_t + \gamma y_{t+1}$ is not available to the learner, it is approximated by $r_t + \gamma\hat{y}_{t+1}$.) Thus with the prediction rule of TD(λ) the update rule (4) *is not* gradient descent for all choices of λ. Curiously, with the new prediction rule (5) of TD$^*(\lambda)$ the update rule (4) used by both algorithms *is* gradient descent,[2] since if $\hat{y}_t$ is set according to the new prediction rule then

$$
\begin{aligned}
\nabla_{\mathbf{w}_t}(y_t - \hat{y}_t)^2 &= -2\eta_t(y_t - \hat{y}_t)\mathbf{X}_t^\lambda \\
&\approx -2\eta_t(r_t + \gamma\hat{y}_{t+1} - \hat{y}_t)\mathbf{X}_t^\lambda.
\end{aligned}
$$

We can also motivate the term $-\sum_{k=1}^{t-1}(\gamma\lambda)^{t-k}\hat{y}_k$ in the prediction rule of TD$^*(\lambda)$ given in equation (5): In this paper, we are comparing the total loss of the algorithm with the total loss of the best linear predictor, so both algorithms TD(λ) and TD$^*(\lambda)$ try to match the y_t's with an on-line linear model. In particular, if $y_t = \mathbf{w} \cdot \mathbf{x}_t$ (that is, the y_t's are a linear function of the $\mathbf{x}_t$'s) and the initial weight vector is the "correct" weight vector $\mathbf{w}$, then the algorithms should always predict correctly (so that $\hat{y}_t = y_t$) and the weight vector $\mathbf{w}_t$ of the algorithms should remain unchanged. Using the fact that $r_t = \mathbf{w} \cdot \mathbf{x}_t - \gamma\ \mathbf{w} \cdot \mathbf{x}_{t+1}$, it is easy to prove by induction that both algorithms have this property.

Thus, in sum, the prediction rule $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{X}_t^\lambda + c_t$ is motivated by gradient descent, where c_t is any term that does not depend on the weight vector $\mathbf{w}_t$. The exact value for c_t is derived using the fact that, in the case described above, we want $\hat{y}_t = y_t$ for all t.

4 Upper bounds for TD$^*(\lambda)$

In proving our upper bounds, be begin with a very general lemma concerning the performance of TD$^*(\lambda)$. We then apply the lemma to derive an analysis of some special cases of interest.

Lemma 1: *Let $\gamma \in [0,1)$, $\lambda \in [0,1]$, and let S be an arbitrary training sequence such that $||\mathbf{X}_t^\lambda|| \leq X$ for all trials t. Let $\mathbf{u}$ be any weight vector, and let $\ell > 0$.*

[2] The factor of two in front of η_t can be absorbed into η_t.

If we execute TD$^*(\lambda)$ *on* S *with initial vector* $\mathbf{w}_1$ *and learning rates* $\eta_t = \eta$ *where* $0 < \eta X^2 \gamma < 1$, *then*

$$L^\ell(\mathrm{TD}^*(\lambda), S) \leq \inf \left\{ \frac{bL^\ell(\mathbf{u}, S) + ||\mathbf{u} - \mathbf{w}_1||^2}{C_b} : b > 0, C_b > 0 \right\}$$

where C_b *equals*

$$2\eta - \eta^2 X^2(1+\gamma^2) - \frac{\eta^2}{b}\left(1 + \left(\frac{\gamma - \gamma\lambda}{1-\gamma\lambda}\right)^2\right)$$
$$-2\left|\eta - \frac{\eta^2}{b}\right|\left(\frac{\gamma-\gamma\lambda}{1-\gamma\lambda}\right)\gamma\lambda - 2\left|\left(\eta - \frac{\eta^2}{b}\right)(\gamma - \gamma\lambda) - \eta^2 X^2 \gamma\right|.$$

Proof For $1 \leq t \leq \ell$, we let $e_t = y_t - \hat{y}_t$, and $e_{\mathbf{u},t} = y_t - \mathbf{u} \cdot \mathbf{x}_t$. We further define $e_{\ell+1} = 0$. Note that the loss of the algorithm at trial t is ${e_t}^2$ and the loss of $\mathbf{u}$ is ${e_{\mathbf{u},t}}^2$. Since, for $t < \ell$,

$$\begin{aligned} r_t + \gamma\hat{y}_{t+1} - \hat{y}_t &= r_t + \gamma\hat{y}_{t+1} - (r_t + \gamma y_{t+1}) + y_t - \hat{y}_t \\ &= e_t - \gamma e_{t+1}, \end{aligned}$$

we can write equation (4), the update of our algorithm, conveniently as

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta(e_t - \gamma e_{t+1})\mathbf{X}_t^\lambda. \tag{6}$$

To simplify the proof, we also define

$$\mathbf{w}_{\ell+1} = \mathbf{w}_\ell + \eta e_\ell \mathbf{X}_\ell^\lambda$$

so that equation (6) holds for all $t \leq \ell$. (In fact, this definition of $\mathbf{w}_{\ell+1}$ differs from the vector that would actually be computed by TD$^*(\lambda)$. This is not a problem, however, since we are here only interested in the behavior of the algorithm on the first ℓ trials which are unaffected by this change.)

We use the function progr_t to measure how much "closer" the algorithm gets to $\mathbf{u}$ during trial t as measured by the distance function $||\cdot||^2$:

$$\mathrm{progr}_t = ||\mathbf{u} - \mathbf{w}_t||^2 - ||\mathbf{u} - \mathbf{w}_{t+1}||^2.$$

Let $\Delta\mathbf{w}_t = \mathbf{w}_{t+1} - \mathbf{w}_t$ for $t \leq \ell$. We have that

$$\begin{aligned} ||\Delta\mathbf{w}_t||^2 &= \eta^2(e_t - \gamma e_{t+1})^2 ||\mathbf{X}_t^\lambda||^2 \\ &\leq \eta^2 X^2 (e_t - \gamma e_{t+1})^2 \end{aligned}$$

and that

$$\begin{aligned} \Delta\mathbf{w}_t \cdot (\mathbf{w}_t - \mathbf{u}) &= \eta(e_t - \gamma e_{t+1})(\mathbf{w}_t \cdot \mathbf{X}_t^\lambda - \mathbf{u} \cdot \mathbf{X}_t^\lambda) \\ &= \eta(e_t - \gamma e_{t+1}) \sum_{k=1}^{t} (\gamma\lambda)^{t-k}(\hat{y}_k - \mathbf{u} \cdot \mathbf{x}_k) \\ &= \eta(e_t - \gamma e_{t+1}) \sum_{k=1}^{t} (\gamma\lambda)^{t-k}(e_{\mathbf{u},k} - e_k) \end{aligned}$$

where the second equality follows from equations (3) and (5), and the last equality from the fact that

$$\begin{aligned}\hat{y}_k - \mathbf{u}\cdot\mathbf{x}_k &= \hat{y}_k - y_k + y_k - \mathbf{u}\cdot\mathbf{x}_k \\ &= e_{\mathbf{u},k} - e_k\,.\end{aligned}$$

Since $-\mathrm{progr}_t = 2\Delta\mathbf{w}_t\cdot(\mathbf{w}_t - \mathbf{u}) + ||\Delta\mathbf{w}_t||^2$, we have that

$$\begin{aligned}-||\mathbf{w}_1 - \mathbf{u}||^2 &\leq ||\mathbf{w}_{\ell+1} - \mathbf{u}||^2 - ||\mathbf{w}_1 - \mathbf{u}||^2 \\ &= -\sum_{t=1}^{\ell}\mathrm{progr}_t \\ &\leq 2\eta\sum_{t=1}^{\ell}\left[(e_t - \gamma e_{t+1})\sum_{k=1}^{t}(\gamma\lambda)^{t-k}(e_{\mathbf{u},k} - e_k)\right] + \eta^2X^2\sum_{t=1}^{\ell}(e_t - \gamma e_{t+1})^2 \quad (7)\end{aligned}$$

This can be written more concisely using matrix notation as follows: Let $\mathbf{Z}_k$ be the $\ell\times\ell$ matrix whose entry (i,j) is defined to be 1 if $j = i + k$ and 0 otherwise. (For instance, $\mathbf{Z}_0$ is the identity matrix.) Let $\mathbf{D} = \mathbf{Z}_0 - \gamma\mathbf{Z}_1$, and let

$$\mathbf{V} = \sum_{t=0}^{\ell-1}(\gamma\lambda)^t\mathbf{Z}_t.$$

Finally, let $\mathbf{e}$ (respectively, $\mathbf{e_u}$) be the length ℓ vector whose tth element is e_t (respectively, $e_{\mathbf{u},t}$). Then the last expression in equation (7) is equal to

$$\eta^2X^2\mathbf{e}^T\mathbf{D}^T\mathbf{D}\mathbf{e} + 2\eta\mathbf{e}^T\mathbf{D}^T\mathbf{V}^T(\mathbf{e_u} - \mathbf{e}). \quad (8)$$

This can be seen by noting that, by a straightforward computation, the tth element of $\mathbf{De}$ is $e_t - \gamma e_{t+1}$, and the tth element of $\mathbf{V}^T(\mathbf{e_u} - \mathbf{e})$ is

$$\sum_{k=1}^{t}(\gamma\lambda)^{t-k}(e_{\mathbf{u},k} - e_k).$$

Using the fact that $2\mathbf{p}^T\mathbf{q} \leq ||\mathbf{p}||^2 + ||\mathbf{q}||^2$ for any pair of vectors $\mathbf{p},\mathbf{q}$, we can upper bound equation (8), for any $b > 0$, by

$$\eta^2X^2\mathbf{e}^T\mathbf{D}^T\mathbf{D}\mathbf{e} - 2\eta\mathbf{e}^T\mathbf{D}^T\mathbf{V}^T\mathbf{e} + \frac{\eta^2}{b}\mathbf{e}^T\mathbf{D}^T\mathbf{V}^T\mathbf{V}\mathbf{D}\mathbf{e} + b\mathbf{e_u}^T\mathbf{e_u} \quad (9)$$

(where we use $\mathbf{p} = (\eta/\sqrt{b})\,\mathbf{VDe}$ and $\mathbf{q} = \sqrt{b}\,\mathbf{e_u}$). Defining

$$\mathbf{M} = \eta^2X^2\mathbf{D}^T\mathbf{D} - \eta(\mathbf{VD} + \mathbf{D}^T\mathbf{V}^T) + \frac{\eta^2}{b}\mathbf{D}^T\mathbf{V}^T\mathbf{VD},$$

and noting that $\mathbf{e}^T\mathbf{D}^T\mathbf{V}^T\mathbf{e} = \mathbf{e}^T\mathbf{VDe}$, we can write equation (9) simply as

$$\mathbf{e}^T\mathbf{M}\mathbf{e} + b\mathbf{e_u}^T\mathbf{e_u}.$$

Note that $\mathbf{M}$ is symmetric. It is known that in this case

$$\max_{\mathbf{e}\neq\mathbf{0}}\frac{\mathbf{e}^T\mathbf{M}\mathbf{e}}{\mathbf{e}^T\mathbf{e}} = \rho(\mathbf{M}) \quad (10)$$

where $\rho(\mathbf{M})$ is the largest eigenvalue of $\mathbf{M}$. Thus, for all vectors $\mathbf{e}$, $\mathbf{e}^T\mathbf{M}\mathbf{e} \leq \rho(\mathbf{M})\mathbf{e}^T\mathbf{e}$. It follows from equation (7) that

$$-||\mathbf{w}_1 - \mathbf{u}||^2 \leq b\sum_{t=1}^{\ell} {e_{\mathbf{u},t}}^2 + \rho(\mathbf{M})\sum_{t=1}^{\ell} {e_t}^2$$

so

$$-\rho(\mathbf{M})\sum_{t=1}^{\ell} {e_t}^2 \leq ||\mathbf{w}_1 - \mathbf{u}||^2 + b\sum_{t=1}^{\ell} {e_{\mathbf{u},t}}^2.$$

In the appendix, we complete the proof by arguing that $\rho(\mathbf{M}) \leq -C_b$. ■

Having proved Lemma 1 in gory generality, we are now ready to apply it to some special cases to obtain bounds that are far more palatable. We begin by considering the case that $\lambda = 0$. Note that TD(0) and TD*(0) are identical.

Theorem 2: *Let $0 \leq \gamma < 1$, and let S be any sequence of instances/reinforcements. Assume that we know a bound X for which $||\mathbf{x}_t|| \leq X$.*

If TD*(0) *uses any start vector $\mathbf{w}_1$ and learning rates $\eta_t = \eta = 1/(X^2+1)$, we have for all $\ell > 0$ and for all $\mathbf{u} \in \mathbb{R}^N$:*

$$L^\ell(\text{TD}^*(0), S) \leq \frac{(1+X^2)(L^\ell(\mathbf{u}, S) + ||\mathbf{w}_1 - \mathbf{u}||^2)}{1-\gamma^2}. \tag{11}$$

Assume further that we know bounds K and U such that for some $\mathbf{u}$ we have $L^\ell(\mathbf{u}, S) \leq K$ and $||\mathbf{w}_1 - \mathbf{u}|| \leq U$. Then for the learning rate

$$\eta_t = \eta = \frac{U}{X\sqrt{K} + X^2 U}$$

we have that

$$L^\ell(\text{TD}^*(0), S) \leq \frac{L^\ell(\mathbf{u}, S) + 2UX\sqrt{K} + X^2||\mathbf{w}_1 - \mathbf{u}||^2}{1-\gamma^2}. \tag{12}$$

Proof When $\lambda = 0$, C_b simplifies to

$$2\eta - \eta^2\left(X^2 + \frac{1}{b}\right)(1+\gamma^2) - 2\gamma\left|\eta - \eta^2\left(X^2 + \frac{1}{b}\right)\right|.$$

To minimize the loss bound given in Lemma 1, we need to maximize C_b with respect to η. It can be shown that, in this case, C_b is maximized, for fixed b, when

$$\eta = \frac{1}{X^2 + 1/b}. \tag{13}$$

The first bound (equation (11)) is then obtained by choosing $b = 1$.

If bounds K and U are known as stated in the theorem, an optimal choice for b can be derived by plugging the choice for η given in equation (13) into the bound in Lemma 1, and replacing $L^\ell(\mathbf{u}, S)$ by K and $||\mathbf{u} - \mathbf{w}_1||^2$ by U. This gives

$$\frac{(bK + U)(X^2 + 1/b)}{1-\gamma^2}$$

which is minimized when $b = U/(X\sqrt{K})$. Plugging this choice of b into the bound of Lemma 1 (and setting η as in equation (13)) gives the bound

$$\begin{aligned}
&\frac{\left(UL^\ell(\mathbf{u},S)/X\sqrt{K} + ||\mathbf{u}-\mathbf{w}_1||^2\right)\left(X^2 + X\sqrt{K}/U\right)}{1-\gamma^2} \\
&= \frac{L^\ell(\mathbf{u},S) + L^\ell(\mathbf{u},S)XU/\sqrt{K} + ||\mathbf{u}-\mathbf{w}_1||^2 X\sqrt{K}/U + ||\mathbf{u}-\mathbf{w}_1||^2X^2}{1-\gamma^2} \\
&\leq \frac{L^\ell(\mathbf{u},S) + 2XU\sqrt{K} + ||\mathbf{u}-\mathbf{w}_1||^2X^2}{1-\gamma^2}.
\end{aligned}$$

■

Next, we consider the case that $\lambda = 1$.

Theorem 3: *Let $0 \leq \gamma < 1$, $\ell > 0$ and let S be any sequence of instances/reinforcements. Assume that we know a bound X for which $||\mathbf{X}_t^1|| \leq X$, and that we know bounds K and U such that for some $\mathbf{u}$ we have $L^\ell(\mathbf{u},S) \leq K$ and $||\mathbf{w}_1 - \mathbf{u}|| \leq U$. Then if* TD*(1) *uses any start vector $\mathbf{w}_1$ and learning rates*

$$\eta_t = \eta = \frac{U}{UX^2(1+\gamma)^2 + X(1+\gamma)\sqrt{K}},$$

then

$$L^\ell(\text{TD}^*(1),S) \leq L^\ell(\mathbf{u},S) + 2\sqrt{K}(1+\gamma)UX + (1+\gamma)^2||\mathbf{w}_1-\mathbf{u}||^2X^2.$$

Proof When $\lambda = 1$,

$$C_b = 2\eta - \eta^2X^2(1+\gamma^2) - \frac{\eta^2}{b}.$$

This is maximized, with respect to η, when

$$\eta = \frac{1}{X^2(1+\gamma^2) + 1/b}.$$

Proceeding as in Theorem 2, we see that the best choice for b is

$$b = \frac{U}{(1+\gamma)X\sqrt{K}}.$$

Plugging into the bound in Lemma 1 completes the theorem. ■

The bound in equation (2) is obtained from Theorem 3 by setting $\mathbf{w}_1 = \mathbf{0}$, and noting that

$$||\mathbf{X}_t^1|| \leq \sum_{k=1}^{t} \gamma^{t-k}||\mathbf{x}_k|| \leq \frac{\max\{||\mathbf{x}_k|| : 1 \leq k \leq t\}}{1-\gamma}$$

by the triangle inequality. Note that the bounds in equations (2) and (12) are incomparable in the sense that, depending on the values of the quantities involved, either bound can be better than the other. This suggests that TD*(0) may or may not be better than TD*(1) depending on the particular problem at hand; the bounds we have derived quantify those situations in which each will perform better than the other.

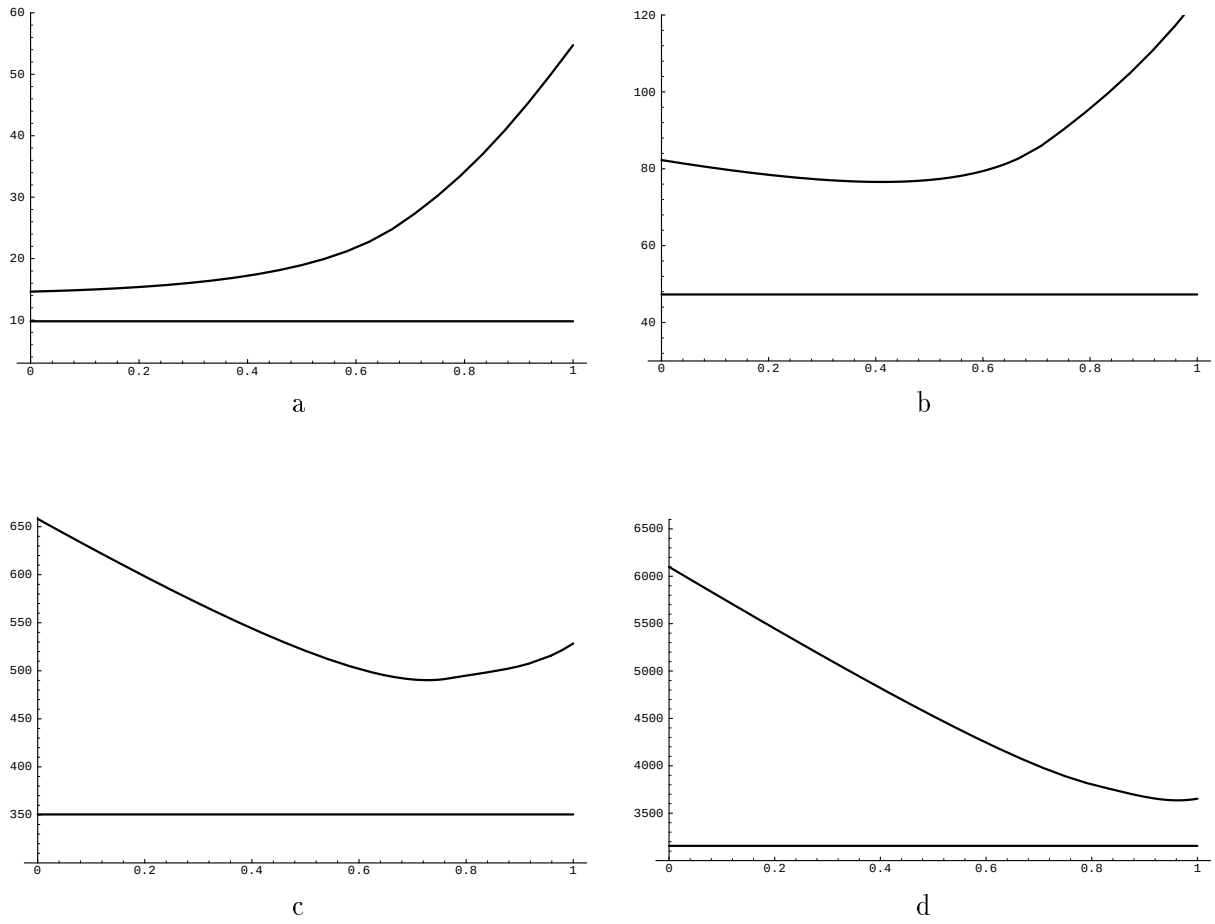

Figure 2: The loss bound given in Lemma 1 as a function of λ when η is chosen so as to minimize the bound.

Ultimately, we hope to extend our analysis to facilitate the optimal choice of $\eta > 0$ and $\lambda \in [0, 1]$. In the meantime, we can numerically find the choices of η and λ that minimize the worst-case bound given in Lemma 1. Figure 2 shows graphs of the worst-case bound given in Lemma 1 as a function of λ when η is chosen so as to minimize our worst-case bound and for fixed settings of the other parameters. More specifically, in all the graphs we have assumed $||\mathbf{w}_1 - \mathbf{u}|| = 1$, and $||\mathbf{x}_t|| \leq 1$ (which implies that $||\mathbf{X}_t^\lambda|| \leq 1/(1 - \gamma\lambda)$). We have also fixed $\gamma = 0.7$. Figures 2a, b, c and d assume that $L^\ell(\mathbf{u}, S)$ equals 3, 30, 300 and 3000, respectively, and each curve shows the upper bound on $L^\ell(\mathrm{TD}^*(\lambda), S)$ given in Lemma 1. The solid line in each figure shows the lower bound obtained in Section 5. In each figure the x-axis crosses the y-axis at the value of $L^\ell(\mathbf{u}, S)$. Note that the gap between the lower bound and $L^\ell(\mathbf{u}, S)$ grows as $\theta(\sqrt{L^\ell(\mathbf{u}, S)})$ when all other variables are kept constant. (This is not visible from the figures because the scaling of the figures varies.) The figures were produced using Mathematica.

As the figures clearly indicate, the higher the loss $L^\ell(\mathbf{u}, S)$, the higher should be our choice for λ. It is interesting that in some intermediate cases, an intermediate value for λ in $(0, 1)$ is the best choice.

5 A lower bound

We next prove a lower bound on the performance of *any* learning algorithm in the model that we have been considering.

Theorem 4: *Let $\gamma \in [0,1]$, $X > 0$, $K \geq 0$, $U \geq 0$ and ℓ a positive integer. For every algorithm A, there exists a sequence S such that the following hold:*

1. $||\mathbf{x}_t|| \leq X$,
2. $K = \min\{L^\ell(\mathbf{u}, S) : ||\mathbf{u}|| \leq U\}$, *and*
3. $L^\ell(A, S) \geq (\sqrt{K} + UX\sqrt{\sigma_\ell})^2$

where $\sigma_\ell := \sum_{k=0}^{\ell-1} \gamma^{2k}$.

Proof The main idea of the proof is to construct a training sequence in which the learning algorithm A receives essentially no information until trial ℓ, at which time the adversary can force the learner to incur significant loss relative to the best linear predictor.

Without loss of generality, we prove the result in the one-dimensional case[3] (i.e., $N = 1$), so we write the instance $\mathbf{x}_t$ simply as x_t. The sequence S is defined as follows: We let $x_t = \gamma^{\ell-t}X$ for $t \leq \ell$, and $x_t = 0$ for $t > \ell$ (thus satisfying part 1). The reinforcement given is $r_t = 0$ if $t \neq \ell$, and $r_\ell = sz$ where $z = UX + \sqrt{K/\sigma_\ell}$ and $s \in \{-1, +1\}$ is chosen adversarially after A has made predictions $\hat{y}_1, \ldots, \hat{y}_\ell$ on the first ℓ trials. Then

$$y_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} = \begin{cases} \gamma^{\ell-t} sz & \text{if } t \leq \ell \\ 0 & \text{otherwise.} \end{cases}$$

To see that part 2 holds, let $\mathbf{u} = u$ be any vector (scalar, really, since $N = 1$) with $|u| \leq U$. Then

$$\begin{aligned} L^\ell(u, S) &= \sum_{t=1}^{\ell} (ux_t - y_t)^2 \\ &= \sum_{t=1}^{\ell} \gamma^{2(\ell-t)} (uX - sz)^2 \\ &= (uX - sz)^2 \sigma_\ell. \end{aligned}$$

Since $|u| \leq U$, it can be seen that this is minimized when $u = sU$, in which case $L^\ell(u, S) = K$ by z's definition.

Finally, consider the loss of A on this sequence:

$$L^\ell(A, S) = \sum_{t=1}^{\ell} (\hat{y}_t - y_t)^2 = \sum_{t=1}^{\ell} (\hat{y}_t - s\gamma^{\ell-t} z)^2.$$

For any real numbers p and q, we have $(p-q)^2 + (p+q)^2 = 2(p^2 + q^2) \geq 2q^2$. Thus, if $s \in \{-1, +1\}$ is chosen uniformly at random, then A's expected loss will be

$$\frac{1}{2} \left(\sum_{t=1}^{\ell} (\hat{y}_t - \gamma^{\ell-t} z)^2 + \sum_{t=1}^{\ell} (\hat{y}_t + \gamma^{\ell-t} z)^2 \right) \geq \sum_{t=1}^{\ell} (\gamma^{\ell-t} z)^2 = z^2 \sigma_\ell = (\sqrt{K} + UX\sqrt{\sigma_\ell})^2.$$

[3] If $N > 1$, we can reduce to the one-dimensional case by zeroing all but one of the components of $\mathbf{x}_t$.

It follows that for the choice of $s \in \{-1, +1\}$ that maximizes A's loss, we will have that $L^\ell(A, S) \geq (\sqrt{K} + UX\sqrt{\sigma_\ell})^2$ as claimed. ■

When $K = 0$, Theorem 4 gives a lower bound of U^2X^2/σ_ℓ which approaches $U^2X^2/(1-\gamma^2)$ as ℓ becomes large. This lower bound matches the second bound of Theorem 2 in the corresponding case. Thus, in the "noise-free" case that there exists a vector $\mathbf{u}$ that perfectly matches the data (i.e., $\min\{L^\ell(\mathbf{u}, S) : ||\mathbf{u}|| \leq U\} = 0$), this shows that TD$^*(0)$ is "optimal" in the sense that its worst-case performance is best possible.

6 Algorithm for the batch model

In the usual supervised learning setting, the on-line learning precedes as follows: In each trial $t \geq 1$ the learner receives an instance $\mathbf{x}_t \in \mathbb{R}^N$. Then, after producing a prediction $\hat{y}_t$ it gets a reinforcement y_t and incurs loss $(\hat{y}_t - y_t)^2$.

A classical algorithm for this problem is the Widrow-Hoff algorithm. It keeps a linear hypothesis represented by the vector $\mathbf{w}_t$ and predicts with $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$. The weight vector is updated using gradient descent:

$$\mathbf{w}_{t+1} := \mathbf{w}_t - 2\eta_t(\mathbf{w}_t \cdot \mathbf{x}_t - y_t)\mathbf{x}_t.$$

Note that $2(\mathbf{w}_t \cdot \mathbf{x}_t - y_t)\mathbf{x}_t$ is the gradient of the loss $(\mathbf{w}_t \cdot \mathbf{x}_t - y_t)^2$ with respect to $\mathbf{w}_t$.

There is a straightforward generalization of the above scenario when more than one instance is received in each trial t. In this generalization, which was previously analyzed in the noise-free case by Cesa-Bianchi, Long and Warmuth [1], the learner does the following in each trial:

1. receives a real-valued matrix $\mathbf{M}_t$ with N columns;
2. predicts with $\hat{\mathbf{y}}_t = \mathbf{M}_t\mathbf{w}_t$;
3. gets reinforcement $\mathbf{y}_t$, a real column vector whose dimension is the number of rows of $\mathbf{M}_t$;
4. incurs loss $||\mathbf{M}_t\mathbf{w}_t - \mathbf{y}_t||^2$;
5. updates its linear hypothesis $\mathbf{w}_t$ using the rule

$$\mathbf{w}_{t+1} := \mathbf{w}_t - 2\eta_t\mathbf{M}_t^T(\mathbf{M}_t\mathbf{w}_t - \mathbf{y}_t).$$

The rows of the matrix $\mathbf{M}_t$ can be viewed as a batch of instances received at trial t. The algorithm has to predict on all instances received in trial t before it gets the reinforcement vector $\mathbf{y}_t$ which contains one reinforcement per row. For each instance, the algorithm is charged for the usual square loss, and the loss in trial t is summed over all instances received in that trial.

We call the algorithm described above WHM. Note that the Widrow-Hoff algorithm is a special case of WHM in which each matrix $\mathbf{M}_t$ contains exactly one row. Also, the update is standard gradient descent in that $2\mathbf{M}_t^T(\mathbf{M}_t\mathbf{w}_t - \mathbf{y}_t)$ is the gradient of the loss $||\mathbf{M}_t\mathbf{w}_t - \mathbf{y}_t||^2$ with respect to $\mathbf{w}_t$.

To model a particular reinforcement learning problem, we have the freedom to make up the matrices $\mathbf{M}_t$ and reinforcements $\mathbf{y}_t$ to suit our purpose. For example, for the case considered by Sutton [6] in which the goal is to predict a single outcome following a sequence of observations, we let $\mathbf{M}_t$ contain the instances of a run and set $\mathbf{y}_t = (z_t, \cdots, z_t)^T$, where z_t

is the reinforcement received for the tth run. (In this case, Sutton shows that the Widrow-Hoff algorithm is actually equivalent to a version of TD(1) in which updates are not made to the weight vector $\mathbf{w}_t$ until the final outcome is received.)

An *example* is a pair $(\mathbf{M}_t, \mathbf{y}_t)$, and, as before, we use S to denote a sequence of examples. We write $L^\ell(A, S)$ to denote the total loss of algorithm A on sequence S:

$$L^\ell(A, S) := \sum_{t=1}^{\ell} (\hat{\mathbf{y}}_t - \mathbf{y}_t)^2,$$

where $\hat{\mathbf{y}}_t$ is the prediction of A in the tth trial, and ℓ is the total length of the sequence. Similarly, the total loss of a weight vector $\mathbf{u} \in \mathbb{R}^N$ is defined as

$$L^\ell(\mathbf{u}, S) := \sum_{t=1}^{\ell} (\mathbf{M}_t\mathbf{u} - \mathbf{y}_t)^2.$$

The proof of the following lemma and theorem are a straightforward generalization of the worst-case analysis of the Widrow-Hoff algorithm given by Cesa-Bianchi, Long and Warmuth [1]. In the proof, we define, $||\mathbf{M}||$, the norm of any matrix $\mathbf{M}$, as

$$||\mathbf{M}|| = \max_{||\mathbf{x}||=1} ||\mathbf{M}\mathbf{x}||.$$

For comparison to the results in the first part of this paper, it is useful to note that $||\mathbf{M}|| \leq X\sqrt{m}$ where m is the number of rows of $\mathbf{M}$, and X is an upper bound on the norm of each row of $\mathbf{M}$.

For any vector $\mathbf{x}$, we write $\mathbf{x}^2$ to denote $\mathbf{x}^T\mathbf{x}$.

Lemma 5: *Let $(\mathbf{M}, \mathbf{y})$ be an arbitrary example such that $||\mathbf{M}|| \leq M$. Let $\mathbf{s}$ and $\mathbf{u}$ be any weight vectors.*

For any $b > 0$, if we execute the matrix algorithm on this example with initial vector $\mathbf{s}$ and the learning rate

$$\eta = \frac{1}{2(||\mathbf{M}||^2 + 1/b)},$$

then

$$||\mathbf{M}\mathbf{s} - \mathbf{y}||^2 \leq (M^2 b + 1)||\mathbf{M}\mathbf{u} - \mathbf{y}||^2 + (M^2 + 1/b)(||\mathbf{u} - \mathbf{s}||^2 - ||\mathbf{u} - \mathbf{w}||^2), \qquad (14)$$

where $\mathbf{w} = \mathbf{s} - 2\eta\mathbf{M}^T(\mathbf{M}\mathbf{s} - \mathbf{y})$ denotes the weight vector of the algorithm WHM *after updating its weight vector $\mathbf{s}$.*

Proof Let $\mathbf{e} := \mathbf{y} - \mathbf{M}\mathbf{s}$ and $\mathbf{e_u} := \mathbf{y} - \mathbf{M}\mathbf{u}$. Then inequality (14) holds if

$$f := ||\mathbf{u} - \mathbf{w}||^2 - ||\mathbf{u} - \mathbf{s}||^2 + 2\eta\mathbf{e}^2 - b\mathbf{e_u}^2 \leq 0.$$

Since $\mathbf{w} = \mathbf{s} + 2\eta\mathbf{M}^T\mathbf{e}$, f can be rewritten as

$$\begin{aligned} f &= -4\eta(\mathbf{u} - \mathbf{s})^T\mathbf{M}^T\mathbf{e} + 4\eta^2||\mathbf{M}^T\mathbf{e}||^2 + 2\eta\mathbf{e}^2 - b\mathbf{e_u}^2 \\ &= -4\eta(\mathbf{e} - \mathbf{e_u})^T\mathbf{e} + 4\eta^2||\mathbf{M}^T\mathbf{e}||^2 + 2\eta\mathbf{e}^2 - b\mathbf{e_u}^2 \\ &= -2\eta\mathbf{e}^2 + 4\eta\mathbf{e_u}^T\mathbf{e} + 4\eta^2||\mathbf{M}^T\mathbf{e}||^2 - b\mathbf{e_u}^2. \end{aligned}$$

Since $2\mathbf{e_u}^T\mathbf{e} \leq \frac{b}{2\eta}\mathbf{e_u}^2 + \frac{2\eta}{b}\mathbf{e}^2$ we can upper bound f by

$$\mathbf{e}^2(-2\eta + 4\eta^2(||\mathbf{M}||^2 + 1/b)) = 0.$$

■

Theorem 6: *Let S be any sequence of examples and let M be the largest norm $||\mathbf{M}_t||$.*

If the matrix algorithm WHM *uses any start vector* $\mathbf{s}$ *and learning rates*

$$\eta_t = \frac{1}{2(||\mathbf{M}_t||^2 + M^2)},$$

then we have for any vector $\mathbf{u}$ *the bound*

$$L^\ell(\text{WHM}, S) \leq 2(L^\ell(\mathbf{u}, S) + M^2||\mathbf{s} - \mathbf{u}||^2). \tag{15}$$

Assume further that we know bounds K and U such that for some $\mathbf{u}$ we have $L^\ell(\mathbf{u}, S) \leq K$ and $||\mathbf{s} - \mathbf{u}|| \leq U$. Then for the learning rates

$$\eta_t = \frac{U}{2(||\mathbf{M}_t||^2 U + M\sqrt{K})}$$

we have

$$L^\ell(\text{WHM}, S) \leq L^\ell(\mathbf{u}, S) + 2MU\sqrt{K} + M^2||\mathbf{s} - \mathbf{u}||^2. \tag{16}$$

Proof By summing the inequality of Lemma 5 over all runs of S we get

$$L^\ell(\text{WHM}, S) \leq (bM^2 + 1)L^\ell(\mathbf{u}, S) + (M^2 + 1/b)(||\mathbf{u} - \mathbf{s}||^2 - ||\mathbf{u} - \mathbf{w}'||^2),$$

where $\mathbf{w}'$ is the weight vector to M after the last reinforcement of S is processed. Since $||\mathbf{u} - \mathbf{w}'||^2 \geq 0$, we have

$$L^\ell(\text{WHM}, S) \leq (bM^2 + 1)L^\ell(\mathbf{u}, S) + (M^2 + 1/b)||\mathbf{u} - \mathbf{s}||^2.$$

Setting $b = 1/M^2$ gives (15). Using the assumptions that $L^\ell(\mathbf{u}, S) \leq K$ and $||\mathbf{s} - \mathbf{u}|| \leq U$, we get

$$L^\ell(\text{WHM}, S) \leq L^\ell(\mathbf{u}, S) + M^2||\mathbf{s} - \mathbf{u}||^2 + bKM^2 + U^2/b. \tag{17}$$

The part of the right hand side that depends on b is $bKM^2 + U^2/b$ which is minimized when $b = U/(M\sqrt{K})$. Using this value of b in (17) gives (16). ■

In the special case that $K = 0$, setting $\eta_t = 1/||\mathbf{M}_t||^2$ gives a bound of

$$L^\ell(\text{WHM}, S) \leq L^\ell(\mathbf{u}, S) + M^2||\mathbf{s} - \mathbf{u}||^2.$$

Note that to prove this, $b = \infty$ is used. The bound for $K = 0$ was previously proved by Cesa-Bianchi, Long and Warmuth [1]. An alternate proof of the above theorem via a reduction from the corresponding theorem for the original Widrow-Hoff algorithm was recently provided by Kivinen and Warmuth [5].

The following lower bound shows that the bounds of the above theorem are best possible.

Theorem 7: *Let $N, m \geq 1$, $K, U \geq 0$ and $M > 0$. For every prediction algorithm A there exists a sequence S consisting of a single example $(\mathbf{M}, \mathbf{y})$ such that the following hold:*

1. *$\mathbf{M}$ is an $m \times N$ matrix and $||\mathbf{M}|| = M$;*
2. *$K = \min\{L^\ell(\mathbf{u}, S) : ||\mathbf{u}|| \leq U\}$; and*
3. *$L^\ell(A, S) \geq K + 2UM\sqrt{K} + U^2M^2$.*

Proof As in the proof of Theorem 4, we prove the result in the case that $N = 1$, without lose of generality. Thus, $\mathbf{M}$ is actually a column vector in $\mathbb{R}^m$.

Let each component of $\mathbf{M}$ be equal to $M/\sqrt{m}$ so that $||M|| = M$. Let each component of $\mathbf{y}$ be equal sz where $z = (MU + \sqrt{K})/\sqrt{m}$ and $s \in \{-1, +1\}$ is chosen adversarially after A has made its prediction $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_m)^T$.

To see that part 2 holds, let $\mathbf{u} = u$ be a vector (scalar, really). Then

$$L^\ell(u, S) = ||\mathbf{M} - \mathbf{y}||^2 = m(Mu/\sqrt{m} - sz)^2$$

which is minimized when $u = sU$ for $|u| \leq U$. In this case, $L^\ell(u, S) = K$.

Finally, by choosing s adversarially to maximize algorithm A's loss, we have

$$\begin{aligned} L^\ell(A, S) &= \max_{s \in \{-1,+1\}} \sum_{i=1}^{m} (\hat{y}_i - sz)^2 \\ &\geq \frac{1}{2} \sum_{i=1}^{m} \left((\hat{y}_i - z)^2 + (\hat{y}_i + z)^2 \right) \\ &\geq \sum_{i=1}^{m} z^2 = K + 2MU\sqrt{K} + M^2U^2. \end{aligned}$$

■

7 Discussion

The primary contribution of this paper is the analysis of some simple temporal-difference algorithms using a worst-case approach. This method of analysis differs dramatically from the statistical approach that has been used in the past for such problems, and our approach has some important advantages.

First, the results that are obtained using the worst-case approach are quite robust. Obviously, any analysis of any learning algorithm is valid only when the assumed conditions actually hold in the real world. By making the most minimal of assumptions — and, in particular, by making no assumptions at all about the stochastic nature of the world — we hope to be able to provide analyses that are as robust and broadly applicable as possible.

Statistical methods for analyzing on-line learning algorithms are only necessary when worst-case bounds cannot be obtained. In this paper, we demonstrated that temporal-difference learning algorithms with simple linear models are highly amenable to worst-case analysis. Although one might expect such a pessimistic approach to give rather weak results, we have found, somewhat surprisingly, that very strong bounds can often be proved even in the worst case.

Worst-case bounds for on-line linear learning algorithms can be very tight even on artificial data [5]. Good experimental performance of a particular algorithm might be seen as weak evidence for showing that the algorithm is good since every algorithm performs well on some data, particularly when the data is artificial. However, if we have a worst-case bound for a particular algorithm, then we can use experimental data to show how much worse the competitors can perform relative to the worst-case bound of the algorithm in question.

Another strength of the worst-case approach is its emphasis on the actual performance of the learning algorithm on the actually observed data. Breaking with more traditional approaches, we do not analyze how well the learning algorithm performs in expectation, or how well it performs asymptotically as the amount of training data becomes infinite, or how well the algorithm estimates the underlying parameters of some assumed stochastic model. Rather, we focus on the quality of the learner's predictions as measured against the finite sequence of data that it actually observes.

Finally, our method of analysis seems to be more fine-grained than previous approaches. As a result, the worst-case approach may help to resolve a number of open issues in temporal-difference learning, such as the following:

- *Which learning rules are best for which problems?* We use the total worst-case loss as our criterion. Minimizing this criterion led us to discover the modified learning rule TD$^*(\lambda)$. Unlike the original TD(λ), this rule has a gradient descent interpretation for general λ. Our method can also be used to derive worst-case bounds for the original rule, but we were unable to obtain bounds for TD(λ) stronger than those given for TD$^*(\lambda)$. It will be curious to see how the two rules compare experimentally.

  Also, the results in Section 4 provide explicit worst-case bounds on the performance of TD$^*(0)$ and TD$^*(1)$. These bounds show that one of the two algorithms may or may not be better than the other depending on the values of the parameters X, K, etc. Thus, using a priori knowledge we may have about a particular learning problem, we can use these bounds to guide us in deciding which algorithm to use.
- *How should a learning algorithm's parameters be tuned?* For instance, we have shown how the learning rate η should be chosen for TD$^*(0)$ and TD$^*(1)$ using knowledge which may be available about a particular problem. For the choice of λ, Sutton showed experimentally that, in some cases, the learner's hypothesis got closest to the target when λ is chosen in $(0, 1)$ and that there is clearly one optimal choice. So far, our worst-case bounds for TD$^*(\lambda)$ are not in closed form when $\lambda \in (0, 1)$, but, numerically, we have found that our results are entirely consistent with Sutton's in this regard.
- *How does the performance of a learning algorithm depend on various parameters of the problem?* For instance, our bounds show explicitly how the performance of TD$^*(\lambda)$ degrades as γ approaches 1. Furthermore, the lower bounds that can sometimes be proved (such as in Section 5) help us to understand what performance is best possible as a function of these parameters.

Open problems. There remain many open research problems in this area. The first of these is to reduce the bound given in Lemma 1 to closed form to facilitate the optimal choice of $\lambda \in [0, 1]$. However, as clearly indicated by Figure 2, even when λ and η are chosen so as to minimize this bound, there remains a significant gap between the upper bounds proved in Section 4 and the lower bound proved in Section 5. This may be a weakness of our analysis, or this may be an indication that an algorithm better than either TD(λ) or TD$^*(\lambda)$ is waiting to be discovered.

As described in Section 3, TD$^*(\lambda)$ can be motivated using gradient descent. Rules of this kind can alternatively be derived within a framework described by Kivinen and Warmuth [5]. Moreover, by modifying one of the parameters of their framework, they show that rules having a qualitatively different flavor can be derived. In particular, they analyze such an algorithm, which they call EG, for the same problem that we are considering in

the special case that $\gamma = 0$. Although the bounds they obtain are generally incomparable with the bounds derived for gradient-descent algorithms, these new algorithms have great advantages in some very important cases. It is straightforward to generalize their update rule for $\gamma > 0$, but the analysis of the resulting update rule is an open problem (although we have made some preliminary progress in this direction).

Lastly, Sutton's TD(λ) algorithm can be viewed as a special case of Watkin's "Q-learning" algorithm [7]. This algorithm is meant to handle a setting in which the learner has a set of actions to choose from, and attempts to choose its actions so as to maximize its total payoff. A very interesting open problem is the extension of the worst-case approach to such a setting in which the learner has partial control over its environment and over the feedback that it receives.

Acknowledgements

We are very grateful to Rich Sutton for his continued feedback and guidance.

Manfred Warmuth acknowledges the support of NSF grant IRI-9123692 and AT&T Bell Laboratories. This research was primarily conducted while visiting AT&T Bell Laboratories.

References

[1] Nicolò Cesa-Bianchi, Philip M. Long, and Manfred K. Warmuth. Worst-case quadratic loss bounds for a generalization of the Widrow-Hoff rule. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 429–438, July 1993.

[2] Peter Dayan. The convergence of $TD(\lambda)$ for general λ. *Machine Learning*, 8(3/4):341–362, May 1992.

[3] Peter Dayan and Terrence J. Sejnowski. $TD(\lambda)$ converges with probability 1. *Machine Learning*, 14(3):295–301, 1994.

[4] Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. Technical Report 9307, MIT Computational Cognitive Science, July 1993.

[5] Jyrki Kivinen and Manfred K. Warmuth. Additive versus exponentiated gradient updates for learning linear functions. Technical Report UCSC-CRL-94-16, University of California Santa Cruz, Computer Research Laboratory, 1994.

[6] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[7] C. J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge, England, 1989.

Appendix

In this technical appendix, we complete the proof of Lemma 1 by bounding $\rho(\mathbf{M})$, the largest eigenvalue of the matrix $\mathbf{M}$.

Let $\mathbf{I}$ be the $\ell \times \ell$ identity matrix, and, for $i, j \geq 0$, define

$$\begin{aligned} \mathbf{S}_i &= \mathbf{Z}_i + \mathbf{Z}_i^T, \\ \mathbf{R}_i &= \mathbf{Z}_i^T \mathbf{Z}_i, \\ \mathbf{P}_{ij} &= \mathbf{Z}_i^T \mathbf{Z}_j + \mathbf{Z}_j^T \mathbf{Z}_i. \end{aligned}$$

Since $\mathbf{Z}_i$ is the zero matrix for $i \geq \ell$, we can rewrite $\mathbf{V}$ more conveniently as

$$\mathbf{V} = \sum_{i \geq 0} (\gamma\lambda)^i \mathbf{Z}_i.$$

By direct but tedious computations, we have that

$$\mathbf{D}^T\mathbf{D} = \mathbf{I} - \gamma\mathbf{S}_1 + \gamma^2\mathbf{R}_1,$$

and

$$\mathbf{VD} = \mathbf{I} + \left(1 - \frac{1}{\lambda}\right) \sum_{i \geq 1} (\gamma\lambda)^i \mathbf{Z}_i$$

since $\mathbf{Z}_i\mathbf{Z}_1 = \mathbf{Z}_{i+1}$ for $i \geq 0$. Also,

$$\begin{aligned} \mathbf{D}^T\mathbf{V}^T\mathbf{VD} &= \mathbf{I} + \left(1 - \frac{1}{\lambda}\right)^2 \left[\sum_{i \geq 1} (\gamma\lambda)^{2i}\mathbf{R}_i + \sum_{j > i \geq 1} (\gamma\lambda)^{i+j}\mathbf{P}_{ij}\right] \\ &\quad + \left(1 - \frac{1}{\lambda}\right) \sum_{i \geq 1} (\gamma\lambda)^i \mathbf{S}_i. \end{aligned}$$

Thus, $\mathbf{M}$ can be written as:

$$\begin{aligned} &\left(\eta^2 X^2 - 2\eta + \frac{\eta^2}{b}\right)\mathbf{I} + \left(-\eta^2 X^2 \gamma + \left(\eta - \frac{\eta^2}{b}\right)\gamma(1-\lambda)\right)\mathbf{S}_1 + \eta^2 X^2 \gamma^2 \mathbf{R}_1 \\ &\quad + \left(1 - \frac{1}{\lambda}\right)\left(\frac{\eta^2}{b} - \eta\right) \sum_{i \geq 2} (\gamma\lambda)^i \mathbf{S}_i + \frac{\eta^2}{b}\left(1 - \frac{1}{\lambda}\right)^2 \left[\sum_{i \geq 1} (\gamma\lambda)^{2i}\mathbf{R}_i + \sum_{j > i \geq 1} (\gamma\lambda)^{i+j}\mathbf{P}_{ij}\right]. \end{aligned}$$

It is known that $\rho(A + B) \leq \rho(A) + \rho(B)$ for real, symmetric matrices A and B. Further, it can be shown (for instance, using equation (10)) that

$$\begin{aligned} \rho(\mathbf{I}) &= 1; \\ \rho(\mathbf{R}_i) &\leq 1; \\ \rho(\pm\mathbf{S}_i) &\leq 2; \\ \rho(\mathbf{P}_{ij}) &\leq 2. \end{aligned}$$

Applying these bounds gives that

$$\begin{aligned} \rho(\mathbf{M}) &\leq \eta^2 X^2 - 2\eta + \frac{\eta^2}{b} + 2\left|\left(\eta - \frac{\eta^2}{b}\right)\gamma(1-\lambda) - \eta^2 X^2 \gamma\right| + \eta^2 X^2 \gamma^2 \\ &\quad -2\left(1 - \frac{1}{\lambda}\right)\left|\frac{\eta^2}{b} - \eta\right| \sum_{i \geq 2} (\gamma\lambda)^i + \frac{\eta^2}{b}\left(1 - \frac{1}{\lambda}\right)^2 \left[\sum_{i \geq 1} (\gamma\lambda)^{2i} + 2\sum_{j > i \geq 1} (\gamma\lambda)^{i+j}\right] \\ &= -C_b. \end{aligned}$$