Fast Gradient-Descent Methods for Temporal-Difference Learning with Linear Function Approximation

Richard S. Sutton,[1] **Hamid Reza Maei,**[1] **Doina Precup,**[2] **Shalabh Bhatnagar,**[3] **David Silver,**[1] **Csaba Szepesvári,**[1] **Eric Wiewiora**[1]

[1]Reinforcement Learning and Artificial Intelligence Laboratory, University of Alberta, Edmonton, Canada
[2]School of Computer Science, McGill University, Montreal, Canada
[3]Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India

Abstract

Sutton, Szepesvári and Maei (2009) recently introduced the first temporal-difference learning algorithm compatible with both linear function approximation and off-policy training, and whose complexity scales only linearly in the size of the function approximator. Although their *gradient temporal difference* (GTD) algorithm converges reliably, it can be very slow compared to conventional linear TD (on on-policy problems where TD is convergent), calling into question its practical utility. In this paper we introduce two new related algorithms with better convergence rates. The first algorithm, *GTD2*, is derived and proved convergent just as GTD was, but uses a different objective function and converges significantly faster (but still not as fast as conventional TD). The second new algorithm, *linear TD with gradient correction*, or *TDC*, uses the same update rule as conventional TD except for an additional term which is initially zero. In our experiments on small test problems and in a Computer Go application with a million features, the learning rate of this algorithm was comparable to that of conventional TD. This algorithm appears to extend linear TD to off-policy learning with no penalty in performance while only doubling computational requirements.

1. Motivation

Temporal-difference methods based on gradient descent and linear function approximation form a core part of the modern field of reinforcement learning and are essential to many of its large-scale applications. However, the simplest and most popular methods, including TD(λ), Q-learning, and Sarsa, are not true gradient-descent methods (Barnard 1993) and, as a result, the conditions under which they converge are narrower and less robust than can usually be guaranteed for gradient-descent methods. In particular, convergence cannot be guaranteed for these methods when they are used with off-policy training (Baird 1995). Off-policy training—training on data from one policy in order to learn the value of another—is useful in dealing with the exploration-exploitation tradeoff and for intra-option learning (Sutton, Precup & Singh 1999). Finding an off-policy temporal-difference algorithm that is effective on large applications with linear function approximation has been one of the most important open problems in reinforcement learning for more than a decade.

Several non-gradient-descent approaches to this problem have been developed, but none has been completely satisfactory. Second-order methods, such as LSTD (Bradtke & Barto 1996; Boyan 1999), can be guaranteed stable under general conditions, but their computational complexity is $O(n^2)$, where n is the number of features used in the linear approximator, as compared to $O(n)$ for TD(λ) and the other simple methods. In applications, n is often too large (e.g., $n = 10^6$ in Computer Go, Silver et al. 2007) for second-order methods to be feasible. Incremental methods such as iLSTD (Geramifard et al. 2006) reduce per-time-step computation to $O(n)$, but still require $O(n^2)$ memory. Precup and colleagues (2001; 2006) have explored $O(n)$ solutions based on importance sampling, but these methods still have the potential for instability, converge slowly, or apply only to special cases.

In this paper we explore the development of true stochastic gradient-descent algorithms for temporal-difference learning with linear function approximation, building on the work of Baird (1995; 1999) and of Sutton, Szepesvári and Maei (2009).

Appearing in *Proceedings of the 26^{th} International Conference on Machine Learning*, Montreal, Canada, 2009.

2. Linear value-function approximation

We consider a prototypical case of temporal-difference learning, that of learning a linear approximation to the state-value function for a given policy and Markov decision process (MDP) from sample transitions. We take both the MDP and the policy to be stationary, so their combination determines the stochastic dynamics of a Markov chain. The state of the chain at each time t is a random variable, denoted $s_t \in \{1, 2, ..., N\}$, and the state-transition probabilities are given by a matrix P. On each transition from s_t to s_{t+1}, there is also a reward, r_{t+1}, whose distribution depends on both states. We seek to learn the parameter $\theta \in \Re^n$ of an approximate value function $V_\theta : \mathcal{S} \to \Re$ such that

$$V_\theta(s) = \theta^\top \phi_s \approx V(s) = E\left\{\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s\right\}, \quad (1)$$

where $\phi_s \in \Re^n$ is a feature vector characterizing state s, and $\gamma \in [0, 1)$ is a constant called the *discount rate*.

In this paper we consider *one-step* temporal-difference learning (corresponding to $\lambda = 0$ in TD(λ)), in which there is one independent update to θ for each state transition and associated reward. There are several settings corresponding to how the state transitions are generated. In the on-policy setting, for example, the state transitions come directly from the continuing evolution of the Markov chain. We assume that the Markov chain is ergodic and uni-chain, so there exists a limiting distribution d such that $d_s = \lim_{t\to\infty} \mathbb{P}(s_t = s)$.[1] In the on-policy case, d is linked to the transition probabilities (in particular, we know that $P^\top d = d$) and this linkage is critical to the convergence of algorithms such as conventional TD (Tsitsiklis & Van Roy 1997). In this paper, we consider a general setting (introduced in Sutton, Szepesvári & Maei 2009) in which the first state of each transition is chosen i.i.d. according to an arbitrary distribution d that may be unrelated to P (this corresponds to off-policy learning). This setting defines a probability over independent triples of state, next state, and reward random variables, denoted (s_k, s'_k, r_k), with associated feature-vector random variables $\phi_k = \phi_{s_k}$ and $\phi'_k = \phi_{s'_k}$. From these we can define, for example, the temporal-difference error,

$$\delta_k = r_k + \gamma\theta_k^\top \phi'_k - \theta_k^\top \phi_k,$$

used in the conventional linear TD algorithm (Sutton 1988):

$$\theta_{k+1} \leftarrow \theta_k + \alpha_k \delta_k \phi_k, \quad (2)$$

where α_k is a sequence of positive step-size parameters.

[1] Our results apply also to the episodic case if d_s is taken to be the proportion of time steps in state s. In this case, the sum in (1) is only over a finite number of time steps, the rows of P may sum to less than 1, and γ may be equal to 1 (as long as $(\gamma P)^\infty = 0$).

3. Objective functions

An objective function is some function of the modifiable parameter θ that we seek to minimize by updating θ. In gradient descent, the updates to θ are proportional to the gradient or sample gradient of the objective function with respect to θ. The first question then, is what to use for the objective function? For example, one natural choice might be the mean squared error (MSE) between the approximate value function V_θ and the true value function V, averaged over the state space according to how often each state occurs. The MSE objective function is

$$\begin{aligned} \text{MSE}(\theta) &= \sum_s d_s \left(V_\theta(s) - V(s)\right)^2 \\ &\stackrel{\text{def}}{=} \| V_\theta - V \|_D^2 . \end{aligned}$$

In the second equation, V_θ and V are viewed as vectors with one element for each state, and the norm $\| v \|_D^2 = v^\top D v$ is weighted by the matrix D that has the d_s on its diagonal.

In temporal-difference methods, the idea is instead to use an objective function representing how closely the approximate value function satisfies the Bellman equation. The true value function V satisfies the Bellman equation exactly:

$$\begin{aligned} V &= R + \gamma P V \\ &\stackrel{\text{def}}{=} TV, \end{aligned}$$

where R is the vector with components $E\{r_{t+1} \mid s_t = s\}$ and T is known as the *Bellman operator*. A seemingly natural measure of how closely the approximation V_θ satisfies the Bellman equation is the *mean-square Bellman error*:

$$\text{MSBE}(\theta) = \| V_\theta - TV_\theta \|_D^2 . \quad (3)$$

This is the objective function used by the most important prior effort to develop gradient-descent algorithms, that by Baird (1995, 1999). However, most temporal-difference algorithms, including TD, LSTD, and GTD, do not converge to the minimum of the MSBE. To understand this, note that the Bellman operator follows the underlying state dynamics of the Markov chain, irrespective of the structure of the function approximator. As a result, TV_θ will typically not be representable as V_θ for any θ. Consider the projection operator Π which takes any value function v and projects it to the nearest value function representable by the function approximator:

$$\Pi v = V_\theta \text{ where } \theta = \arg\min_\theta \| V_\theta - v \|_D^2 .$$

In a linear architecture, in which $V_\theta = \Phi\theta$ (where Φ is the matrix whose rows are the ϕ_s), the projection operator is linear and independent of θ:

$$\Pi = \Phi(\Phi^\top D \Phi)^{-1} \Phi^\top D$$

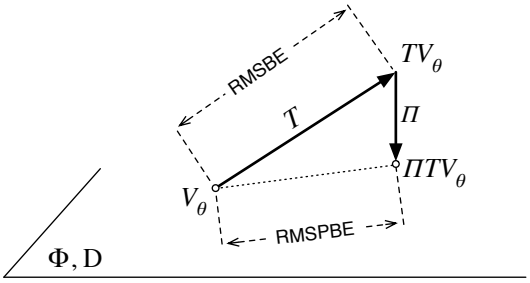

Figure 1. Geometric relationships between the square roots of the two Bellman-error objective functions.

All the algorithms mentioned above find or converge to a fixpoint of the composed projection and Bellman operators, that is to a value of θ such that

$$V_\theta = \Pi T V_\theta. \tag{4}$$

We call this value of θ the TD fixpoint. In the current work, we take as our objective the deviation from this fixpoint. That is, we use as our objective function the mean-square *projected* Bellman error:

$$\text{MSPBE}(\theta) \quad = \quad \| V_\theta - \Pi T V_\theta \|^2_D \,. \tag{5}$$

Figure 1 shows the relationship between this and the MSBE objective function geometrically. Although many previous works have highlighted the goal of achieving the TD fixpoint (4), the present work seems to be the first to focus on the MSPBE as an objective function to be minimized (but see Antos, Szepesvári and Munos 2008, p. 100). Further insight into the difference between the two Bellman error objective functions can be gained by considering the episodic example in Figure 2.

Finally, we close this discussion of objective functions by giving the function used to derive the original GTD algorithm. This objective function does not seem to have a ready geometric interpretation. Here we call it the *norm of the expected TD update*:

$$\text{NEU}(\theta) = \mathbb{E}[\delta\phi]^\top \mathbb{E}[\delta\phi]\,. \tag{6}$$

4. Derivation of the new algorithms

In this section we derive two new algorithms as stochastic gradient descent in the projected Bellman error objective (5). We first establish some relationships between the relevant expectations and vector-matrix quantities:

$$\mathbb{E}\big[\phi\phi^\top\big] = \sum_s d_s \phi_s \phi_s^\top = \Phi^\top D \Phi,$$

$$\begin{aligned}\mathbb{E}[\delta\phi] \quad &= \quad \sum_s d_s \phi_s \left(R_s + \gamma \sum_{s'} P_{ss'} V_\theta(s') - V_\theta(s) \right) \\ &= \quad \Phi^\top D (T V_\theta - V_\theta),\end{aligned}$$

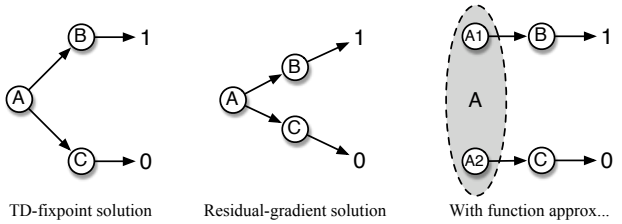

Figure 2. Backward-bootstrapping example. In the left and middle panels, episodes begin in state A then transition either to B or to C with equal probability before proceeding to termination with a reward of 1 or 0 (all other transitions have zero reward). The vertical positions of the states represent their values according to the TD-fixpoint solution (left panel) and according to the residual-gradient (RG) solution (middle panel; Baird 1995, 1999). State A, for example, has height midway between 0 and 1 in both solutions, corresponding to its correct value of $\frac{1}{2}$ (because episodes starting in A end half the time with a total reward of 1 and half the time with a total reward of 0, and $\gamma = 1$). In the TD solution, states B and C are given values of 1 and 0 respectively, whereas in the RG solution they are given the values $\frac{3}{4}$ and $\frac{1}{4}$. The 1,0 values are correct in that these states are always followed by these rewards, but they result in large TD errors, of $\delta = \pm\frac{1}{2}$, on transitions out of A. The RG solution has smaller TD errors, of $\delta = \pm\frac{1}{4}$, on all of its transitions, resulting in a smaller mean-square TD error *per episode* of $\frac{1}{4}^2 \times 2 = \frac{1}{8}$ as compared to $\frac{1}{2}^2 = \frac{1}{4}$ for the TD solution. That is, the RG solution splits the TD error over two transitions to minimize squared TD error overall. The RG solution is also sometimes described as backwards bootstrapping—making the value of a state look like the value of the state that preceded it as well as the state that followed it. It has long been recognized that backwards bootstrapping is to be avoided (Sutton 1988; Dayan 1992) but the RG algorithm has remained of interest because it is a gradient-descent method and thus guaranteed to converge (whereas TD(λ) converges only on-policy) and because it has a "two sample version" that minimizes the MSBE rather than the squared TD error. The key difference here is that, from A, the squared TD error tends to be large but the expected TD error (the Bellman error) tends to be zero (as long as the B and C values are distributed symmetrically around $\frac{1}{2}$). The TD solution 1,0 is in fact the minimum MSBE solution on this problem, and this has led to the widespread belief that the MSBE solves the problem of backwards bootstrapping and the appearance of the $\frac{3}{4}$,$\frac{1}{4}$ solution. However, this is not the case in general; once function approximation is introduced, the MSBE and MSPBE solutions differ, and the $\frac{3}{4}$,$\frac{1}{4}$ solution may reappear. An example of this is shown in the right panel, where the previous state A is split into two states, A1 and A2, that share the same feature representation; they look the same and must be given the same approximate value. Trajectories start in one of the two A states each with 50% probability, then proceed deterministically either to B and 1, or to C and 0. From the observable data, this example looks just like the previous, except here taking multiple samples is no help because the system is deterministic, and they will all be the same. Here, the $\frac{3}{4}$,$\frac{1}{4}$ solution minimizes not just the squared TD error, but the MSBE as well; only the MSPBE criterion puts the minimum at the $1, 0$ solution. The MSBE objective causes function approximation resources to be expended trying to reduce the Bellman error associated with A1 and A2, whereas the MSPBE objective takes into account that their approximated values will ultimately be projected onto the same value function.

and note that

$$\begin{aligned}\Pi^\top D\Pi &= (\Phi(\Phi^\top D\Phi)^{-1}\Phi^\top D)^\top D(\Phi(\Phi^\top D\Phi)^{-1}\Phi^\top D)\\ &= D^\top\Phi(\Phi^\top D\Phi)^{-1}\Phi^\top D\Phi(\Phi^\top D\Phi)^{-1}\Phi^\top D\\ &= D^\top\Phi(\Phi^\top D\Phi)^{-1}\Phi^\top D.\end{aligned}$$

Using these relationships, the projected objective can be written in terms of expectations as

$$\begin{aligned}&\text{MSPBE}(\theta)\\ &= \| V_\theta - \Pi TV_\theta \|_D^2\\ &= \| \Pi(V_\theta - TV_\theta) \|_D^2\\ &= (\Pi(V_\theta - TV_\theta))^\top D(\Pi(V_\theta - TV_\theta))\\ &= (V_\theta - TV_\theta)^\top\Pi^\top D\Pi(V_\theta - TV_\theta)\\ &= (V_\theta - TV_\theta)^\top D^\top\Phi(\Phi^\top D\Phi)^{-1}\Phi^\top D(V_\theta - TV_\theta)\\ &= (\Phi^\top D(TV_\theta - V_\theta))^\top(\Phi^\top D\Phi)^{-1}\Phi^\top D(TV_\theta - V_\theta)\\ &= \mathbb{E}[\delta\phi]^\top\,\mathbb{E}\big[\phi\phi^\top\big]^{-1}\,\mathbb{E}[\delta\phi]\,.\end{aligned}$$

From this form, it is clear that MSPBE differs from NEU (6) only by the inclusion of the inverse of the feature-covariance matrix. As in prior work (Sutton, Szepesvari & Maei 2009) we use a second modifiable parameter $w \in \Re^n$ to form a quasi-stationary estimate of all but one of the expectations in the gradient of the objective function, thereby avoiding the need for two independent samples. Here we use a conventional linear predictor which causes w to approximate

$$w \approx \mathbb{E}\big[\phi\phi^\top\big]^{-1}\,\mathbb{E}[\delta\phi]\,. \tag{7}$$

Using this, we can write the negative gradient of the MSPBE objective function as

$$\begin{aligned}-\frac{1}{2}\nabla\text{MSPBE}(\theta) &= \mathbb{E}\big[(\phi-\gamma\phi')\phi^\top\big]\,\mathbb{E}\big[\phi\phi^\top\big]^{-1}\,\mathbb{E}[\delta\phi]\\ &\approx \mathbb{E}\big[(\phi-\gamma\phi')\phi^\top\big]\,w,\end{aligned}$$

which can be directly sampled. The resultant $O(n)$ algorithm, which we call GTD2, is

$$\theta_{k+1} = \theta_k + \alpha_k(\phi_k - \gamma\phi'_k)(\phi_k^\top w_k), \tag{8}$$

where w_k is updated by

$$w_{k+1} = w_k + \beta_k(\delta_k - \phi_k^\top w_k)\phi_k. \tag{9}$$

The derivation of our second new algorithm starts from the same expression for the gradient and then takes a slightly different route:

$$\begin{aligned}&-\frac{1}{2}\nabla\text{MSPBE}(\theta)\\ &= \mathbb{E}\big[(\phi-\gamma\phi')\phi^\top\big]\,\mathbb{E}\big[\phi\phi^\top\big]^{-1}\,\mathbb{E}[\delta\phi]\\ &= \big(\mathbb{E}\big[\phi\phi^\top\big] - \gamma\mathbb{E}\big[\phi'\phi^\top\big]\big)\,\mathbb{E}\big[\phi\phi^\top\big]^{-1}\,\mathbb{E}[\delta\phi]\\ &= \mathbb{E}[\delta\phi] - \gamma\mathbb{E}\big[\phi'\phi^\top\big]\,\mathbb{E}\big[\phi\phi^\top\big]^{-1}\,\mathbb{E}[\delta\phi]\\ &\approx \mathbb{E}[\delta\phi] - \gamma\mathbb{E}\big[\phi'\phi^\top\big]\,w,\end{aligned}$$

which is then sampled, resulting in the following $O(n)$ algorithm, which we call *TD with gradient correction*, or TDC for short:

$$\theta_{k+1} = \theta_k + \alpha_k\delta_k\phi_k - \alpha\gamma\phi'_k(\phi_k^\top w_k), \tag{10}$$

where w_k is generated by (9) as in GTD2. Note that the update to θ_k is the sum of two terms, and that the first term is exactly the same as the update (2) of conventional linear TD. The second term is essentially an adjustment or correction of the TD update so that it follows the gradient of the MSPBE objective function. If the second parameter vector is initialized to $w_0 = 0$, and β_k is small, then this algorithm will start out making almost the same updates as conventional linear TD. Note also that after the convergence of θ_k, w_k will converge to zero again.

5. Proof of convergence of GTD2

The purpose of this section is to establish that the GTD2 algorithm converges with probability one to the TD fixpoint (4) under standard assumptions.

Theorem 1 (Convergence of GTD2)**.** *Consider the GTD2 iterations (8) and (9) with step-size sequences α_k and β_k satisfying $\beta_k = \eta\alpha_k$, $\eta > 0$, $\alpha_k, \beta_k \in (0,1]$, $\sum_{k=0}^\infty \alpha_k = \infty$, $\sum_{k=0}^\infty \alpha_k^2 < \infty$. Further assume that (ϕ_k, r_k, ϕ'_k) is an i.i.d. sequence with uniformly bounded second moments. Let $A = \mathbb{E}\big[\phi_k(\phi_k - \gamma\phi'_k)^\top\big]$, $b = \mathbb{E}[r_k\phi_k]$, and $C = \mathbb{E}\big[\phi_k\phi_k^\top\big]$. Assume that A and C are non-singular. Then the parameter vector θ_k converges with probability one to the TD fixpoint (4).*

Proof. The proof is very similar to that given by Sutton, Szepesvári and Maei (2009) for GTD, and we refer the reader to that reference for further details. It is shown there that the TD fixpoint can be written as the condition

$$-A\theta + b = 0. \tag{11}$$

First, we rewrite the algorithm's two iterations as a single iteration in a combined parameter vector with $2n$ components, $\rho_k^\top = (d_k^\top, \theta_k^\top)$, where $d_k = w_k/\sqrt{\eta}$, and a new reward-related vector with $2n$ components, $g_{k+1}^\top = (r_k\phi_k^\top, 0^\top)$, as follows:

$$\rho_{k+1} = \rho_k + \alpha_k\sqrt{\eta}\,(G_{k+1}\rho_k + g_{k+1})\,,$$

where

$$G_{k+1} = \begin{pmatrix} -\sqrt{\eta}\phi_k\phi_k^\top & \phi_k(\gamma\phi'_k - \phi_k)^\top \\ (\phi_k - \gamma\phi'_k)\phi_k^\top & 0 \end{pmatrix}.$$

Let $G = \mathbb{E}[G_k]$ and $g = \mathbb{E}[g_k]$. Note that G and g are well-defined as by the assumption the process $\{\phi_k, r_k, \phi'_k\}_k$ is i.i.d. In particular,

$$G = \begin{pmatrix} -\sqrt{\eta}\,C & -A \\ A^\top & 0 \end{pmatrix}, \quad g = \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

Further, note that (11) follows from $G\rho + g = 0$, where $\rho^\top = (d^\top, \theta^\top)$.

Now we apply the ordinary differential equation (ODE) approach and Theorem 2.2 of Borkar & Meyn (2000). For this purpose we write $\rho_{k+1} = \rho_k + \alpha_k\sqrt{\eta}(G\rho_k + g + (G_{k+1} - G)\rho_k + (g_{k+1} - g)) = \rho_k + \alpha'_k(h(\rho_k) + M_{k+1})$, where $\alpha'_k = \alpha_k\sqrt{\eta}$, $h(\rho) = g + G\rho$ and $M_{k+1} = (G_{k+1} - G)\rho_k + g_{k+1} - g$. Let $\mathcal{F}_k = \sigma(\rho_1, M_1, \ldots, \rho_{k-1}, M_k)$ be sigma fields generated by the quantities ρ_i, M_i, $i \leq k$, $k \geq 1$. Theorem 2.2 requires the verification of the following conditions: (i) The function h is Lipschitz and $h_\infty(\rho) = \lim_{r\to\infty} h(r\rho)/r$ is well-defined for every $\rho \in \Re^{2n}$; (ii-a) The sequence $(M_k, \mathcal{F}_k)$ is a martingale difference sequence, and (ii-b) for some $c_0 > 0$, $\mathbb{E}\big[\|M_{k+1}\|^2 \,|\, \mathcal{F}_k\big] \leq c_0(1 + \|\rho_k\|^2)$ holds for any initial parameter vector ρ_1; (iii) The sequence α'_k satisfies $0 < \alpha'_k \leq 1$, $\sum_{k=1}^\infty \alpha'_k = \infty$, $\sum_{k=1}^\infty (\alpha'_k)^2 < +\infty$; (iv) The ODE $\dot\rho = h_\infty(\rho)$ has the origin as a globally asymptotically stable equilibrium and (v) The ODE $\dot\rho = h(\rho)$ has a unique globally asymptotically stable equilibrium. Clearly, $h(\rho)$ is Lipschitz with coefficient $\|G\|$ and $h_\infty(\rho) = G\rho$. By construction, $(M_k, \mathcal{F}_k)$ satisfies $\mathbb{E}[M_{k+1}|\mathcal{F}_k] = 0$ and $M_k \in \mathcal{F}_k$, i.e., it is a martingale difference sequence. Condition (ii-b) can be shown to hold by a simple application of the triangle inequality and the boundedness of the second moments of (ϕ_k, r_k, ϕ'_k). Condition (iii) is satisfied by our conditions on the step-size sequences α_k, β_k.

For the last two conditions, we begin by showing that the real parts of all the eigenvalues of G are negative. First, we show that G is non-singular. Using the determinant rule for partitioned matrices, we get $\det(G) = \det(A^\top C^{-1} A) = (\det C)^{-1}(\det A)^2 \neq 0$. This indicates that all the eigenvalues of G are non-zero. Now, let $\lambda \in \mathbb{C}$, $\lambda \neq 0$ be an eigenvalue of G with corresponding normalized eigenvector $x \in \mathbb{C}^{2n}$; that is, $\|x\|^2 = x^*x = 1$, where x^* is the complex conjugate of x. Hence $x^*Gx = \lambda$. Let $x^\top = (x_1^\top, x_2^\top)$, where $x_1, x_2 \in \mathbb{C}^n$. Using the definition of G, $\lambda = x^*Gx = -\sqrt{\eta}\|x_1\|_C^2 - x_1^*Ax_2 + x_2^*A^\top x_1$, where $\|x_1\|_C^2 = x_1^*Cx_1$. Because A is real, $A^* = A^\top$, and it follows that $(x_1^*Ax_2)^* = x_2^*A^\top x_1$. Thus, $\mathrm{Re}(\lambda) = \mathrm{Re}(x^*Gx) = -\sqrt{\eta}\|x_1\|_C^2 \leq 0$. We are now done if we show that x_1 cannot be zero. If $x_1 = 0$, then from $\lambda = x^*Gx$ we get that $\lambda = 0$, which contradicts with $\lambda \neq 0$. Thus, (iv) is satisfied. Finally, for the ODE $\dot\rho = h(\rho)$, note that $\rho^* = -G^{-1}g$ is the unique asymptotically stable equilibrium with $\bar V(\rho) = (G\rho + g)^T(G\rho + g)/2$ as its associated strict Liapunov function. The claim now follows. □

6. Proof of Convergence of TDC

Theorem 2 (Convergence of TD with Gradient Correction)**.** *Consider the iterations (10) and (9) of the TD with gradient corrections algorithm. Let the step-size sequences α_k and β_k, $k \geq 0$ satisfy in this case $\alpha_k, \beta_k > 0$, for all k, $\sum_{k=0}^\infty \alpha_k = \sum_{k=0}^\infty \beta_k = \infty$, $\sum_{k=0}^\infty \alpha_k^2$, $\sum_{k=0}^\infty \beta_k^2 < \infty$ and that $\dfrac{\alpha_k}{\beta_k} \to 0$ as $k \to \infty$. Further assume that (ϕ_k, r_k, ϕ'_k) is an i.i.d. sequence with uniformly bounded second moments. Let $A = \mathbb{E}\big[\phi_k(\phi_k - \gamma\phi'_k)^\top\big]$, $b = \mathbb{E}[r_k\phi_k]$, and $C = \mathbb{E}\big[\phi_k\phi_k^\top\big]$. Assume that A and C are non-singular matrices. Then the parameter vector θ_k converges with probability one to the TD fixpoint (4).*

Proof. The proof of this theorem relies on a two-timescale difference in the step-size schedules $\{\alpha_k\}$ and $\{\beta_k\}$; see Borkar (1997) for a convergence analysis of general two-timescale stochastic approximation recursions. The recursions (9) and (10) correspond to the faster and slower recursions respectively. This is because beyond some integer $N_0 > 0$ (i.e., $\forall k \geq N_0$), the increments in (9) are uniformly larger than those in (10) and hence converge faster. Along the faster timescale, i.e., the one corresponding to $\{\beta_k\}$ (see Borkar (1997) for a detailed description of faster and slower timescales), the associated system of ODEs corresponds to

$$\dot\theta\,(t) = 0, \tag{12}$$

$$\dot w(t) = \mathbb{E}[\delta_t\phi_t \mid \theta(t)] - Cw(t). \tag{13}$$

The ODE (12) suggests that $\theta(t) \equiv \theta$ (i.e., a time invariant parameter) when viewed from the faster timescale. Indeed, recursion (10) can be rewritten as

$$\theta_{k+1} = \theta_k + \beta_k\xi(k),$$

where, from (10), $\xi(k) = \left(\dfrac{\alpha_k}{\beta_k}\left(\delta_k\phi_k - \gamma\phi'_k\phi_k^\top w_k\right)\right) \to 0$ almost surely as $k \to \infty$, because $\dfrac{a_k}{\beta_k} \to 0$ as $k \to \infty$. By the Hirsch lemma (Theorem 1, pp. 339 of Hirsch 1989), it follows that $\| \theta_k - \theta \| \to 0$ almost surely as $k \to \infty$ for some θ that depends on the initial condition θ_0 of recursion (10).

Consider now the recursion (9). Let $M_{k+1} = (\delta_k\phi_k - \phi_k\phi_k^T w_k) - \mathbb{E}\big[(\delta_k\phi_k - \phi_k\phi_k^T w_k) \mid \mathcal{F}(k)\big]$, where $\mathcal{F}(k) = \sigma(w_l, \theta_l, l \leq k; \phi_s, \phi'_s, r_s, s < k)$, $k \geq 1$ are the sigma fields generated by w_0, θ_0, $w_{l+1}, \theta_{l+1}, \phi_l, \phi'_l$, $0 \leq l < k$. It is easy to verify that M_{k+1}, $k \geq 0$ are integrable random variables that satisfy $\mathbb{E}[M_{k+1} \mid \mathcal{F}(k)] = 0$, $\forall k \geq 0$. Also, because r_k, ϕ_k and ϕ'_k have uniformly bounded second moments, it can be seen that

$$\mathbb{E}\big[\| M_{k+1} \|^2 \big| \mathcal{F}(k)\big] \leq c_1(1 + \| w_k \|^2 + \| \theta_k \|^2),\ k \geq 0,$$

for some constant $c_1 > 0$. Now consider the ODE pair (12)-(13). Because $\theta(t) \equiv \theta$ from (12), the ODE (13) can be written as

$$\dot w(t) = \mathbb{E}[\delta_t\phi_t \mid \theta] - Cw(t). \tag{14}$$

Now consider the function $h(w) = \mathbb{E}[\delta\phi \mid \theta] - Cw$, i.e., the driving vector field of the ODE (14). For (14), $w^* = C^{-1}\mathbb{E}[\delta\phi \mid \theta]$ is the unique globally asymptotically stable equilibrium. Let $h_\infty(\cdot)$ be the function defined by $h_\infty(w) = \lim_{r\to\infty} \frac{h(rw)}{r}$. Then $h_\infty(w) = -Cw$. For the ODE

$$\dot{w}(t) = h_\infty(w(t)) = -Cw(t),$$

the origin is a globally asymptotically stable equilibrium because C is a positive definite matrix (because it is non-negative definite and nonsingular). Assumptions (A1) and (A2) of Borkar & Meyn 2000 are now verified and by their Theorem 2.2 we obtain that $\| w_k - w^* \| \to 0$ almost surely as $k \to \infty$.

Consider now the slower timescale recursion (10). In the light of the above, (10) can be rewritten as

$$\theta_{k+1} = \theta_k + \alpha_k\delta_k\phi_k - \alpha_k\gamma\phi'_k\phi_k^\top C^{-1}\mathbb{E}[\delta_k\phi_k \mid \theta_k]\,. \quad (15)$$

Let $\mathcal{G}(k) = \sigma(\theta_l, l \le k; \phi_s, \phi'_s, r_s, s < k)$ be sigma fields generated by the quantities $\theta_0, \theta_{l+1}, \phi_l, \phi'_l, 0 \le l < k$. Let

$$\begin{aligned} Z_{k+1} &= (\delta_k\phi_k - \gamma\phi'_k\phi_k^\top C^{-1}\mathbb{E}[\delta_k\phi_k \mid \theta_k]) \\ &\quad - \mathbb{E}\big[(\delta_k\phi_k - \gamma\phi'_k\phi_k^\top C^{-1}\mathbb{E}[\delta_k\phi_k \mid \theta_k]) \mid \mathcal{G}(k)\big] \\ &= (\delta_k\phi_k - \gamma\phi'_k\phi_k^\top C^{-1}\mathbb{E}[\delta_k\phi_k \mid \theta_k]) \\ &\quad - \mathbb{E}[\delta_k\phi_k \mid \theta_k] - \gamma\mathbb{E}\big[\phi'_k\phi_k^\top\big]\, C^{-1}\mathbb{E}[\delta_k\phi_k \mid \theta_k]\,. \end{aligned}$$

It is easy to see that $Z_k, k \ge 0$ are integrable random variables and $\mathbb{E}[Z_{k+1} \mid \mathcal{G}(k)] = 0, \forall k \ge 0$. Further,

$$\mathbb{E}\big[\| Z_{k+1} \|^2 \big| \mathcal{G}(k)\big] \le c_2(1 + \| \theta_k \|^2),\ k \ge 0,$$

for some constant $c_2 > 0$, again because r_k, ϕ_k and ϕ'_k have uniformly bounded second moments.

Consider now the following ODE associated with (10):

$$\dot{\theta}(t) = (I - \mathbb{E}\big[\gamma\phi'\phi^T\big]\, C^{-1})\mathbb{E}[\delta\phi \mid \theta(t)]\,. \quad (16)$$

Let $\bar{h}(\theta(t))$ be the driving vector field of the ODE (16). Note that

$$\begin{aligned} \bar{h}(\theta(t)) &= (I - \mathbb{E}\big[\gamma\phi'\phi^T\big]\, C^{-1})\mathbb{E}[\delta\phi \mid \theta(t)] \\ &= (C - \mathbb{E}\big[\gamma\phi'\phi^T\big])C^{-1}\mathbb{E}[\delta\phi \mid \theta(t)] \\ &= (\mathbb{E}\big[\phi\phi^T\big] - \mathbb{E}\big[\gamma\phi'\phi^T\big])C^{-1}\mathbb{E}[\delta\phi \mid \theta(t)] \\ &= A^TC^{-1}(-A\theta(t) + b), \end{aligned}$$

because $\mathbb{E}[\delta\phi \mid \theta(t)] = -A\theta(t) + b$.

Now $\theta^* = A^{-1}b$ can be seen to be the unique globally asymptotically stable equilibrium for (16). Let $\bar{h}_\infty(\theta) = \lim_{r\to\infty} \frac{\bar{h}(r\theta)}{r}$. Then $\bar{h}_\infty(\theta) = -A^TC^{-1}A\theta$. Consider now the ODE

$$\dot{\theta}(t) = -A^TC^{-1}A\theta(t). \quad (17)$$

Because C^{-1} is positive definite and A has full rank (as it is nonsingular by assumption), the matrix $A^TC^{-1}A$ is also positive definite. The ODE (17) has the origin as its unique globally asymptotically stable equilibrium. The assumptions (A1)-(A2) of Borkar & Meyn 2000 are once again verified and the claim follows. □

7. Empirical Results

To begin to assess the practical utility of the new algorithms, we compared their empirical learning rate to that of GTD and conventional TD on four small problems—three random-walk problems and a Boyan-chain problem—and a larger Computer Go problem. All of these problems were episodic, undiscounted, and involved only on-policy training with a fixed policy.

The random-walk problems were all based on the standard Markov chain (Sutton 1988; Sutton & Barto 1998) with a linear arrangement of five states plus two absorbing terminal states at each end. Episodes began in the center state of the five, then transitioned randomly with equal probability to a neighboring state until a terminal state was reached. The rewards were zero everywhere except on transition into the right terminal state, upon which the reward was $+1$. We used three versions of this problem, differing only in their feature representations. The first representation, which we call *tabular features*, was the familiar table-lookup case in which, for example, the second state was represented by the vector $\phi_2 = (0, 1, 0, 0, 0)^\top$. The second representation, which we call *inverted features*, was chosen to cause extensive inappropriate generalization between states; it represented the second state by $\phi_2 = (\frac{1}{2}, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})^\top$ (the value $\frac{1}{2}$ was chosen to give the feature vectors unit norm). The third representation, which we called *dependent features*, used only $n = 3$ features and was not sufficient to solve the problem exactly. The feature vectors for the five states, left to right, were $\phi_1 = (1, 0, 0)^\top$, $\phi_2 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0)^\top$, $\phi_3 = (\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})^\top$, $\phi_4 = (0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})^\top$, and $\phi_5 = (0, 0, 1)^\top$. The Boyan-chain problem is a standard episodic task for comparing TD-style algorithms with linear function approximation (see Boyan 2002 for details). We used the version with 14 states and $n = 4$ features.

We applied GTD, GTD2, TDC, and TD to these problems with a range of constant values for their step-size parameters. The parameter α was varied over a wide range of values, in powers of 2. For the GTD, GTD2, and TDC algorithms, the ratio $\eta = \beta/\alpha$ took values from the set $\{\frac{1}{4}, \frac{1}{2}, 1, 2\}$ for the random-walk problems; one lower power of two was added for the Boyan-chain problem. The initial parameter vectors, θ_0 and w_0, were set to 0 for all algorithms.

Each algorithm and parameter setting was run for 100-500

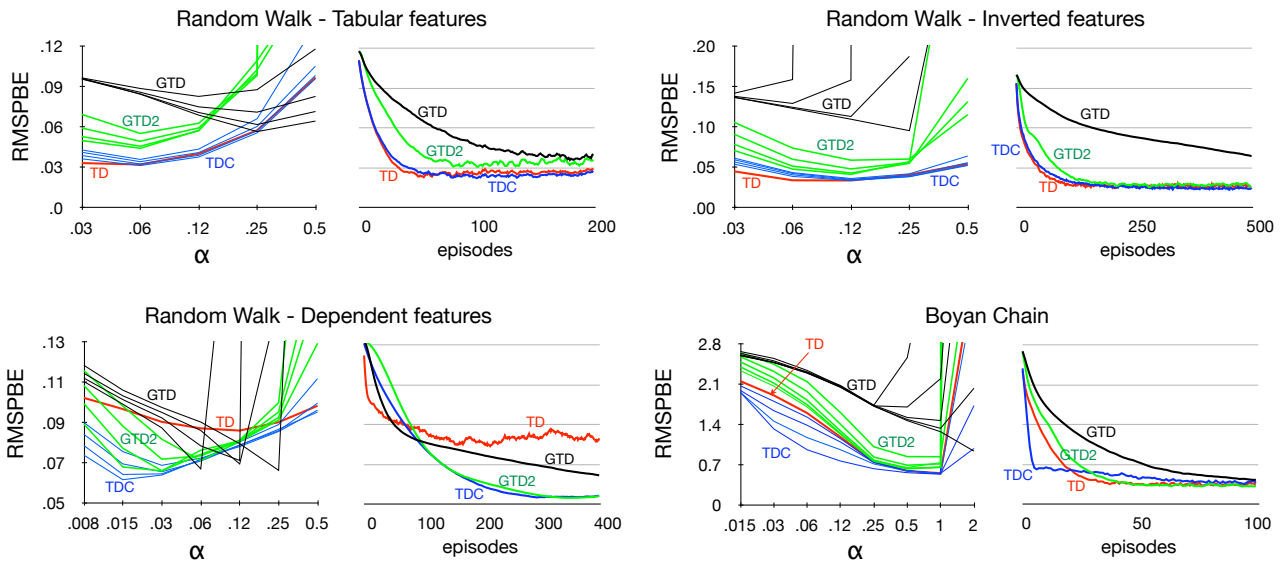

Figure 3. Empirical results on the four small problems—three versions of the 5-state random walk plus the 14-state Boyan chain. In each of the four panels, the right subpanel shows a learning curve at best parameter values, and the left subpanel shows a parameter study plotting the average height of the learning curve for each algorithm, for various $\eta = \beta/\alpha$, as a function of α.

episodes depending on the problem, with the square root of the MSPBE, MSBE, NEU, and MSE (see Section 4) computed after each episode, then averaged over 100 independent runs. Figure 3 summarizes all the results on the small problems using the MSPBE as the performance measure. The results for the other objective functions were similar in all cases and produced the same rankings. The standard errors are all on the order of 10^{-4} or less, so are not shown. All the algorithms were similar in terms of their dependence and sensitivity to the step sizes. Overall, GTD learned the slowest, followed after a significant margin by GTD2, followed by TDC and TD. TDC and TD performed similarly, with the extra flexibility provided by β sometimes allowing TDC to perform slightly better.

To get a measure of how well the new algorithms perform on a larger problem, we applied them to learning an evaluation function for 9x9 Computer Go. We used a version of RLGO (Silver et al. 2007) modified to use a purely linear evaluation function. This system used 969,894 binary features corresponding to all possible shapes in every 3x3, 2x2, and 1x1 region of the board. Using weight sharing to take advantage of location-independent and location-dependent symmetries, the million features are reduced to a parameter vector of $n = 63{,}303$ components. With this large of a parameter vector, $O(n^2)$ methods are not feasible. To make the problem as straightforward as possible, we sought to learn the value function for a fixed policy, in this case for the policy that chose randomly among the legal moves. Experience was generated by self-play, with all rewards zero except upon winning the game, when the reward was 1.

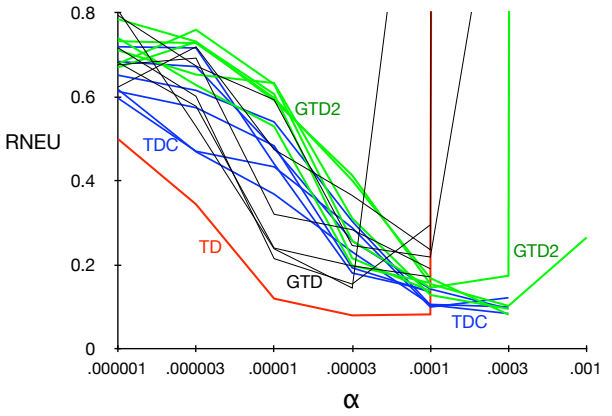

Figure 4. Residual error in learning an evaluation function for 9x9 Computer Go, as a function of algorithm and step-size, in terms of the square root of the norm of the expected TD update (6). The η values used were $\frac{1}{16}$, $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, 1, and 2.

We applied all four algorithms to this problem with a range of step sizes. In each run, θ was initialized to random values uniformly distributed in $[-0.1, 0.1]$. The secondary parameter, w, was initialized to 0. Training then proceeded for 1000 complete games, after which θ was frozen and another 1000 games run to compute an estimate of an objective function. The objective functions cannot be exactly computed here because of the size of the problem. The NEU objective is the most straightforward to estimate sim-

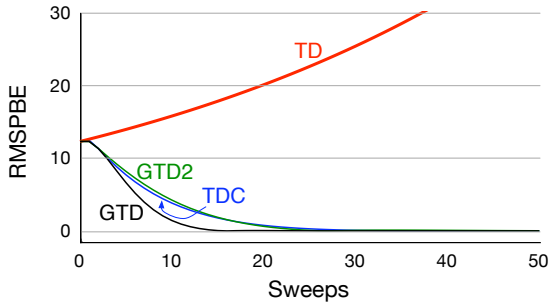

Figure 5. Learning curves on Baird's off-policy counterexample: TD diverges, whereas the gradient methods converge. This is the 7-state version of the "star" counterexample (Baird 1995), for which divergence is monotonic. Updating was done synchronously in dynamic-programming-like sweeps through the state space. For TD, $\alpha = 0.1$. For the gradient algorithms, $\alpha = 0.05$ and $\eta = 10$. The initial parameter value was $\theta_0 = (1, 1, 1, 1, 1, 1, 10, 1)^\top$, and $\gamma = 0.99$.

ply by averaging the value of $\delta\phi$ over the 1000 test games and then taking the norm of the resultant vector. It is this performance measure that we recorded and averaged over 40 runs to produce the data shown in Figure 4. The results are remarkably consistent with what we saw in the small problems. The GTD algorithm was the slowest, followed by GTD2, TDC, and TD, though the differences between the last three are probably not significant given the coarseness of the parameter sampling and the standard errors, which were about 0.05 in this experiment (they are omitted from the graph to reduce clutter).

Finally, Figure 5 shows results for an off-policy learning problem, demonstrating that the gradient methods converge on a well known counterexample (Baird 1995) for which TD diverges.

8. Conclusion

We have introduced two new gradient-based temporal-difference learning algorithms that minimize a natural performance measure, the projected Bellman error, and proved them convergent with linear function approximation in a general setting that includes both on-policy and off-policy learning. Both algorithms have time and memory complexity that is linear in the number of features used in the function approximation, and both are significantly faster than GTD, the only previously proposed algorithm with these properties. Moreover, the TD with gradient corrections algorithm appears to be comparable in speed to conventional linear TD on on-policy problems. This is the first time that all these desirable features—linear complexity, speed, and convergence with off-policy learning and function approximation—have been achieved in one algorithm.

References

Antos, A., Szepesvári, Cs., Munos, R. (2008). Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning 71*:89–129.

Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 30–37.

Baird, L. C. (1999). Reinforcement Learning Through Gradient Descent. PhD thesis, Carnegie-Mellon University Technical Report CMU-CS-99-132.

Barnard, E. (1993). Temporal-difference methods and Markov models. *IEEE Transactions on Systems, Man, and Cybernetics 23*(2):357–365.

Borkar, V. S. (1997). Stochastic approximation with two timescales. *Systems and Control Letters 29*:291-294.

Borkar, V. S. and Meyn, S. P. (2000). The ODE method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control And Optimization 38*(2):447–469.

Boyan, J. (2002). Technical update: Least-squares temporal difference learning. *Machine Learning 49*:233–246.

Bradtke, S., Barto, A. G. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning 22*:33–57.

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

Geramifard, A., Bowling, M., Sutton, R. S. (2006). Incremental least-square temporal difference learning. *Proceedings of the National Conference on Artificial Intelligence*, pp. 356–361.

Hirsch, M. W. (1989). Convergent activation dynamics in continuous time networks. *Neural Networks 2*:331–349.

Precup, D., Sutton, R. S. and Dasgupta, S. (2001). Off-policy temporal-difference learning with function approximation. *Proceedings of the 18th International Conference on Machine Learning*, pp. 417–424.

Precup, D., Sutton, R. S., Paduraru, C., Koop, A., Singh, S. (2006). Off-policy learning with recognizers. *Advances in Neural Information Processing Systems 18*.

Silver, D., Sutton, R. S., Müller, M. (2007). Reinforcement learning of local shape in the game of Go. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 1053–1058.

Sturtevant, N. R., White, A. M. (2006). Feature construction for reinforcement learning in hearts. In *Proceedings of the 5th International Conference on Computers and Games*.

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

Sutton, R. S., Precup D., and Singh, S (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence 112*:181–211.

Sutton, R. S., Szepesvári, Cs., Maei, H. R. (2009). A convergent $O(n)$ algorithm for off-policy temporal-difference learning with linear function approximation. In *Advances in Neural Information Processing Systems 21*. MIT Press.

Tsitsiklis, J. N., and Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control 42*:674–690.