Gradient Descent Temporal Difference-difference Learning

Anonymous authors
Paper under double-blind review

Abstract

Off-policy algorithms, in which a behavior policy differs from the target policy and is used to gain experience for learning, have proven to be of great practical value in reinforcement learning. However, even for simple convex problems such as linear value function approximation, these algorithms are not guaranteed to be stable. To address this, alternative algorithms that are provably convergent in such cases have been introduced, the most well known being gradient descent temporal difference (GTD) learning. This algorithm and others like it, however, tend to converge much more slowly than conventional temporal difference learning. In this paper we propose *gradient descent temporal difference-difference (Gradient-DD) learning* in order to improve GTD2, a GTD algorithm (Sutton et al., 2009b), by introducing second-order differences in successive parameter updates. We investigate this algorithm in the framework of linear value function approximation, theoretically proving its convergence by applying the theory of stochastic approximation. Studying the model empirically on the random walk task, the Boyan-chain task, and the Baird's off-policy counterexample, we find substantial improvement over GTD2 and, in several cases, better performance even than conventional TD learning.

1 Introduction

Off-policy algorithms for value function learning enable an agent to use a behavior policy that differs from the target policy in order to gain experience for learning. However, because off-policy methods learn a value function for a target policy given data due to a different behavior policy, they are slower to converge than on-policy methods and may even diverge when applied to problems involving function approximation (Baird, 1995; Sutton & Barto, 2018).

Two general approaches have been investigated to address the challenge of developing stable and effective off-policy temporal-difference algorithms. One approach is to use importance sampling methods to warp the update distribution so that the expected value is corrected (Precup et al., 2000; Mahmood et al., 2014). This approach is useful for the convergence guarantee, but it does not address stability issues. The second main approach to addressing the challenge of off-policy learning is to develop true gradient descent-based methods that are guaranteed to be stable regardless of the update distribution. (Sutton et al., 2009a;b) proposed the first off-policy gradient-descent-based temporal difference algorithms (GTD and GTD2, respectively). These algorithms are guaranteed to be stable, with computational complexity scaling linearly with the size of the function approximator. Empirically, however, their convergence is much slower than conventional temporal difference (TD) learning, limiting their practical utility (Ghiassian et al., 2020; White & White, 2016). Building on this work, extensions to the GTD family of algorithms (see (Ghiassian et al., 2018) for a review) have allowed for incorporating eligibility traces (Maei & Sutton, 2010; Geist & Scherrer, 2014), non-linear function approximation such as with a neural network (Maei, 2011), and reformulation of the optimization as a saddle point problem (Liu et al., 2015; Du et al., 2017). However, due to their slow convergence, none of these stable off-policy methods are commonly used in practice.

In this work, we introduce a new gradient descent algorithm for temporal difference learning with linear value function approximation. This algorithm, which we call *gradient descent temporal difference-difference* (Gradient-DD) learning, is an acceleration technique that employs second-order differences in successive parameter updates. The basic idea of Gradient-DD is to modify the error objective function by additionally

considering the prediction error obtained in the last time step, then to derive a gradient-descent algorithm based on this modified objective function. In addition to exploiting the Bellman equation to get the solution, this modified error objective function avoids drastic changes in the value function estimate by encouraging local search around the current estimate. Algorithmically, the Gradient-DD approach only adds an additional term to the update rule of the GTD2 method, and the extra computational cost is negligible. We prove its convergence by applying the theory of stochastic approximation. This result is supported by numerical experiments, which also show that Gradient-DD obtains better convergence in many cases than conventional TD learning.

1.1 Related Work

In related approaches to ours, some previous studies have attempted to improve Gradient-TD algorithms by adding regularization terms to the objective function. These approaches have used have used l_1 regularization on weights to learn sparse representations of value functions Liu et al. (2012), or l_2 regularization on weights Ghiassian et al. (2020). Our work is different from these approaches in two ways. First, whereas these previous studies investigated a variant of TD learning with gradient corrections, we take the GTD2 algorithm as our starting point. Second, unlike these previous approaches, our approach modifies the error objective function by using a distance constraint rather than a penalty on weights. The distance constraint works by restricting the search to some region around the evaluation obtained in the most recent time step. With this modification, our method provides a learning rule that contains second-order differences in successive parameter updates.

Our approach is similar to trust region policy optimization (Schulman et al., 2015) or relative entropy policy search (Peters et al., 2010), which penalize large changes being learned in policy learning. In these methods, constrained optimization is used to update the policy by considering the constraint on some measure between the new policy and the old policy. Here, however, our aim is to find the optimal value function, and the regularization term uses the previous value function estimate to avoid drastic changes in the updating process.

Our approach bears similarity to the natural gradient approach widely used in reinforcement learning (Amari, 1998; Bhatnagar et al., 2009; Degris et al., 2012; Dabney & Thomas, 2014; Thomas et al., 2016), which also features a constrained optimization form. However, Gradient-DD is distinct from the natural gradient. The essential difference is that, unlike the natural gradient, Gradient-DD is a trust region method, which defines the trust region according to the difference between the current value and the value obtained from the previous step. From the computational cost viewpoint, unlike natural TD (Dabney & Thomas, 2014), which needs to update an estimate of the metric tensor, the computational cost of Gradient-DD is essentially the same as that of GTD2.

2 Gradient descent method for off-policy temporal difference learning

2.1 Problem definition and background

In this section, we formalize the problem of learning the value function for a given policy under the Markov decision process (MDP) framework. In this framework, the agent interacts with the environment over a sequence of discrete time steps, $t = 1, 2, \ldots$. At each time step the agent observes a state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$. In response, the environment emits a reward $r_t \in \mathbb{R}$ and transitions the agent to its next state $s_{t+1} \in \mathcal{S}$. The state and action sets are finite. State transitions are stochastic and dependent on the immediately preceding state and action. Rewards are stochastic and dependent on the preceding state and action, as well as on the next state. The process generating the agent's actions is termed the behavior policy. In off-policy learning, this behavior policy is in general different from the target policy $\pi : \mathcal{S} \to \mathcal{A}$. The objective is to learn an approximation to the state-value function under the target policy in a particular environment:

$$V(s) = \mathrm{E}_\pi \left[\sum\nolimits_{t=1}^{\infty} \gamma^{t-1} r_t | s_1 = s \right], \tag{1}$$

where $\gamma \in [0, 1)$ is the discount rate.

In problems for which the state space is large, it is practical to approximate the value function. In this paper we consider linear function approximation, where states are mapped to feature vectors with fewer components than the number of states. Specifically, for each state $s \in \mathcal{S}$ there is a corresponding feature vector $\mathbf{x}(s) \in \mathbb{R}^p$, with $p \leq |\mathcal{S}|$, such that the approximate value function is given by

$$V_{\mathbf{w}}(s) := \mathbf{w}^\top \mathbf{x}(s). \tag{2}$$

The goal is then to learn the parameters $\mathbf{w}$ such that $V_{\mathbf{w}}(s) \approx V(s)$.

2.2 Gradient temporal difference learning

A major breakthrough for the study of the convergence properties of MDP systems came with the introduction of the GTD and GTD2 learning algorithms (Sutton et al., 2009a;b). We begin by briefly recapitulating the GTD algorithms, which we will then extend in the following sections. To begin, we introduce the Bellman operator $\mathbf{B}$ such that the true value function $\mathbf{V} \in \mathbb{R}^{|\mathcal{S}|}$ satisfies the Bellman equation:

$$\mathbf{V} = \mathbf{R} + \gamma \mathbf{P}\mathbf{V} =: \mathbf{B}\mathbf{V},$$

where $\mathbf{R}$ is the reward vector with components $\mathrm{E}(r_n|s_n = s)$, and $\mathbf{P}$ is a matrix of the state transition probabilities under the behavior policy. In temporal difference methods, an appropriate objective function should minimize the difference between the approximate value function and the solution to the Bellman equation.

Having defined the Bellman operator, we next introduce the projection operator $\mathbf{\Pi}$, which takes any value function $\mathbf{V}$ and projects it to the nearest value function within the space of approximate value functions of the form Eqn. (2). Letting $\mathbf{X}$ be the matrix whose rows are $\mathbf{x}(s)$, the approximate value function can be expressed as $\mathbf{V}_{\mathbf{w}} = \mathbf{X}\mathbf{w}$. The projection operator is then given by

$$\mathbf{\Pi} = \mathbf{X}(\mathbf{X}^\top \mathbf{D}\mathbf{X})^{-1}\mathbf{X}^\top \mathbf{D},$$

where the matrix $\mathbf{D}$ is diagonal, with each diagonal element d_s corresponding to the probability of visiting state s.

The natural measure of how closely the approximation $\mathbf{V}_{\mathbf{w}}$ satisfies the Bellman equation is the mean-squared Bellman error:

$$\mathrm{MSBE}(\mathbf{w}) = \|\mathbf{V}_{\mathbf{w}} - \mathbf{B}\mathbf{V}_{\mathbf{w}}\|^2_{\mathbf{D}}, \tag{3}$$

where the norm is weighted by $\mathbf{D}$, such that $\|\mathbf{V}\|^2_{\mathbf{D}} = \mathbf{V}^\top \mathbf{D}\mathbf{V}$. However, because the Bellman operator follows the underlying state dynamics of the Markov chain, irrespective of the structure of the linear function approximator, $\mathbf{B}\mathbf{V}_{\mathbf{w}}$ will typically not be representable as $\mathbf{V}_{\mathbf{w}}$ for any $\mathbf{w}$. An alternative objective function, therefore, is the mean squared *projected* Bellman error (MSPBE), which we define as

$$J(\mathbf{w}) = \|\mathbf{V}_{\mathbf{w}} - \mathbf{\Pi}\mathbf{B}\mathbf{V}_{\mathbf{w}}\|^2_{\mathbf{D}}. \tag{4}$$

Following (Sutton et al., 2009b), our objective is to minimize this error measure. As usual in stochastic gradient descent, the weights at each time step are then updated by $\Delta\mathbf{w} = -\alpha\nabla J(\mathbf{w})$, where $\alpha > 0$, and

$$-\frac{1}{2}\nabla J(\mathbf{w}) = -\mathrm{E}[(\gamma\mathbf{x}_{n+1} - \mathbf{x}_n)\mathbf{x}_n^\top][\mathrm{E}(\mathbf{x}_n\mathbf{x}_n^\top)]^{-1}\mathrm{E}(\delta_n\mathbf{x}_n). \tag{5}$$

For notational simplicity, we have denoted the feature vector associated with s_n as $\mathbf{x}_n = \mathbf{x}(s_n)$. We have also introduced the temporal difference error $\delta_n = r_n + (\gamma\mathbf{x}_{n+1} - \mathbf{x}_n)^\top \mathbf{w}_n$. Let $\boldsymbol{\eta}_n$ denote the estimate of $[\mathrm{E}(\mathbf{x}_n\mathbf{x}_n^\top)]^{-1}\mathrm{E}(\delta_n\mathbf{x}_n)$ at the time step n. Because the factors in Eqn. (5) can be directly sampled, the resulting updates in each step are

$$\begin{aligned}
\delta_n =& r_n + (\gamma\mathbf{x}_{n+1} - \mathbf{x}_n)^\top \mathbf{w}_n \\
\boldsymbol{\eta}_{n+1} =& \boldsymbol{\eta}_n + \beta_n(\delta_n - \mathbf{x}_n^\top \boldsymbol{\eta}_n)\mathbf{x}_n \\
\mathbf{w}_{n+1} =& \mathbf{w}_n - \alpha_n(\gamma\mathbf{x}_{n+1} - \mathbf{x}_n)(\mathbf{x}_n^\top \boldsymbol{\eta}_n).
\end{aligned} \tag{6}$$

These updates define the GTD2 learning algorithm, which we will build upon in the following section.

3 Gradient descent temporal difference-difference learning

In this section we modify the objective function by additionally considering the difference between $\mathbf{V}_{\mathbf{w}}$ and $\mathbf{V}_{\mathbf{w}_{n-1}}$, which denotes the value function estimate at step $n-1$ of the optimization. We propose a new objective $J_{\mathrm{GDD}}(\mathbf{w}|\mathbf{w}_{n-1})$, where the notation "$\mathbf{w}|\mathbf{w}_{n-1}$" in the parentheses means that the objective is defined given $\mathbf{V}_{\mathbf{w}_{n-1}}$ of the previous time step $n-1$. Specifically, we modify Eqn. (4) as follows:

$$J_{\mathrm{GDD}}(\mathbf{w}|\mathbf{w}_{n-1}) = J(\mathbf{w}) + \kappa \|\mathbf{V}_{\mathbf{w}} - \mathbf{V}_{\mathbf{w}_{n-1}}\|_{\mathbf{D}}^2, \tag{7}$$

where $\kappa \geq 0$ is a parameter of the regularization, and we assume $\kappa \sim O(1)$. We show in Section A.1 of the appendix that minimizing Eqn. (7) is equivalent to the following optimization

$$\arg\min_{\mathbf{w}} J(\mathbf{w}) \text{ s.t. } \|\mathbf{V}_{\mathbf{w}} - \mathbf{V}_{\mathbf{w}_{n-1}}\|_{\mathbf{D}}^2 \leq \mu \tag{8}$$

where $\mu > 0$ is a parameter which becomes large when κ is small, so that the MSPBE objective is recovered as $\mu \to \infty$, equivalent to $\kappa \to 0$ in Eqn. (7).

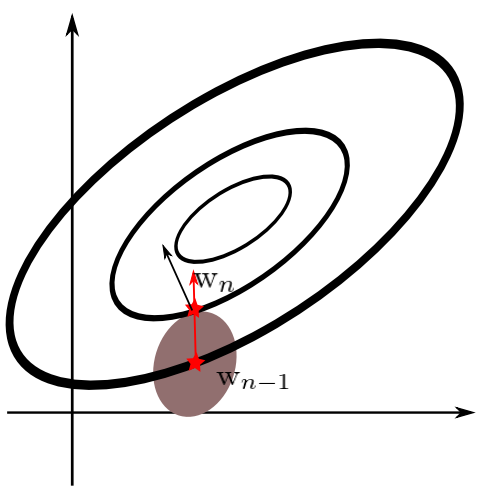


Figure 1: Schematic diagram of Gradient-DD learning with $\mathbf{w} \in \mathbb{R}^2$. Rather than updating $\mathbf{w}$ directly along the gradient of the MSPBE (black arrow), the update rule selects $\mathbf{w}_n$ (red arrow) that minimizes the MSPBE while satisfying the constraint $\|\mathbf{V}_{\mathbf{w}} - \mathbf{V}_{\mathbf{w}_{n-1}}\|_{\mathbf{D}}^2 \leq \mu$ (shaded ellipse).

Rather than simply minimizing the optimal prediction from the projected Bellman equation, the agent makes use of the most recent update to look for the solution, choosing a $\mathbf{w}$ that minimizes the MSPBE while following the constraint that the estimated value function should not change too greatly, as illustrated in Fig. 1. In effect, the regularization term encourages searching around the estimate at previous time step, especially when the state space is large.

Eqn. (8) shows that the regularized objective is a *trust region* approach, which seeks a direction that attains the best improvement possible subject to the distance constraint. The trust region is defined by the value distance rather than the weight distance, meaning that Gradient-DD also makes use of the natural gradient of the objective around $\mathbf{w}_{n-1}$ rather than around $\mathbf{w}_n$ (see Section A.2 of the appendix for details). In this sense, our approach can be explained as a trust region method that makes use of natural gradient information to prevent the estimated value function from changing too drastically.

For comparison with related approaches using natural gradients, in Fig. 8 of the appendix we compare the empirical performance of our algorithm with natural GTD2 and natural TDC Dabney & Thomas (2014) using the random walk task introduced below in Section 5.

With these considerations in mind, the negative gradient of $J_{\mathrm{GDD}}(\mathbf{w}|\mathbf{w}_{n-1})$ is

$$\begin{aligned} &-\frac{1}{2}\nabla J_{\mathrm{GDD}}(\mathbf{w}|\mathbf{w}_{n-1}) \\ =&-\mathrm{E}[(\gamma\mathbf{x}_{n+1} - \mathbf{x}_n)\mathbf{x}_n^\top][\mathrm{E}(\mathbf{x}_n\mathbf{x}_n^\top)]^{-1}\mathrm{E}(\delta_n\mathbf{x}_n) - \kappa\mathrm{E}[(\mathbf{x}_n^\top\mathbf{w}_n - \mathbf{x}_n^\top\mathbf{w}_{n-1})\mathbf{x}_n]. \end{aligned} \tag{9}$$

Because the terms in Eqn. (9) can be directly sampled, the stochastic gradient descent updates are given by

$$\begin{aligned}\delta_n =& r_n + (\gamma \mathbf{x}_{n+1} - \mathbf{x}_n)^\top \mathbf{w}_n \\ \boldsymbol{\eta}_{n+1} =& \boldsymbol{\eta}_n + \alpha_n(\delta_n - \mathbf{x}_n^\top \boldsymbol{\eta}_n)\mathbf{x}_n \\ \mathbf{w}_{n+1} =& \mathbf{w}_n - \kappa\alpha_n(\mathbf{x}_n^\top \mathbf{w}_n - \mathbf{x}_n^\top \mathbf{w}_{n-1})\mathbf{x}_n - \alpha_n(\gamma \mathbf{x}_{n+1} - \mathbf{x}_n)(\mathbf{x}_n^\top \boldsymbol{\eta}_n). \end{aligned} \tag{10}$$

These update equations define the Gradient-DD method, in which the GTD2 update equations (6) are generalized by including a second-order update term in the third update equation, where this term originates from the squared bias term in the objective (7). Since Gradient-DD is not sensitive to the step size of updating $\boldsymbol{\eta}$ (see Fig. 7 in the appendix), the updates of Gradient-DD only have a single shared step size α_n rather than two step sizes α_n, β_n as GTD2 and TDC used. It is worth noting that the computational cost of our algorithm is essentially the same as that of GTD2. In the following sections, we shall analytically and numerically investigate the convergence and performance of Gradient-DD learning.

4 Convergence Analysis

In this section we establish the convergence of Gradient-DD learning. Denote $\mathbf{G}_n = \begin{bmatrix} \mathbf{x}_n\mathbf{x}_n^\top & \mathbf{x}_n(\mathbf{x}_n - \gamma\mathbf{x}_{n+1})^\top \\ -(\mathbf{x}_n - \gamma\mathbf{x}_{n+1})\mathbf{x}_n^\top & \mathbf{0} \end{bmatrix}$, and $\mathbf{H}_n = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{x}_n\mathbf{x}_n^\top \end{bmatrix}$. We rewrite the update rules in Eqn. (10) as a single iteration in a combined parameter vector with $2n$ components, $\boldsymbol{\rho}_n = (\boldsymbol{\eta}_n^\top, \mathbf{w}_n^\top)^\top$, and a new reward-related vector with $2n$ components, $\mathbf{g}_{n+1} = (r_n\mathbf{x}_n^\top, \mathbf{0}^\top)^\top$, as follows:

$$\boldsymbol{\rho}_{n+1} = \boldsymbol{\rho}_n - \kappa\alpha_n\mathbf{H}_n(\boldsymbol{\rho}_n - \boldsymbol{\rho}_{n-1}) + \alpha_n(\mathbf{G}_n\boldsymbol{\rho}_n + \mathbf{g}_{n+1}), \tag{11}$$

Theorem 1. *Consider the update rules (11) with step-size sequences α_n. Let the TD fixed point be $\mathbf{w}^*$, such that $\mathbf{V}_{\mathbf{w}^*} = \mathbf{\Pi}\mathbf{B}\mathbf{V}_{\mathbf{w}^*}$. Suppose that (A0) $\alpha_n \in (0,1)$, $\sum_{n=1}^\infty \alpha_n = \infty$, $\sum_{n=1}^\infty \alpha_n^2 < \infty$, (A1) $(\mathbf{x}_n, r_n, \mathbf{x}_{n+1})$ is an i.i.d. sequence with uniformly bounded second moments, (A2) $E[(\mathbf{x}_n - \gamma\mathbf{x}_{n+1})\mathbf{x}_n^\top]$ and $E(\mathbf{x}_n\mathbf{x}_n^\top)$ are non-singular, (A3) $\sup_n \|\boldsymbol{\rho}_{n+1} - \boldsymbol{\rho}_n\|$ is bounded in probability, (A4) κ is a constant such that $0 \leq \kappa < \infty$. Then as $n \to \infty$, $\mathbf{w}_n \to \mathbf{w}^*$ with probability 1.*

Proof sketch. Due to the second-order difference term in Eqn. (11), the analysis framework in (Borkar & Meyn, 2000) does not directly apply to the Gradient-DD algorithm when (A0) holdes, i.e., step size is tapered. Likewise, the two-timescale convergence analysis (Bhatnagar et al., 2009) is also not directly applicable. Defining $\mathbf{u}_{n+1} = \boldsymbol{\rho}_{n+1} - \boldsymbol{\rho}_n$, we rewrite the iterative process in Eqn. (11) into two parallel processes which are given by

$$\boldsymbol{\rho}_{n+1} = \boldsymbol{\rho}_n - \kappa\alpha_n\mathbf{H}_n\mathbf{u}_n + \alpha_n(\mathbf{G}_n\boldsymbol{\rho}_n + \mathbf{g}_{n+1}), \tag{12}$$

$$\mathbf{u}_{n+1} = -\kappa\alpha_n\mathbf{H}_n\mathbf{u}_n + \alpha_n(\mathbf{G}_n\boldsymbol{\rho}_n + \mathbf{g}_{n+1}). \tag{13}$$

We analyze the parallel processes Eqns. (12) & Eqn. (13) instead of directly analyzing Eqn. (11). Our proofs have three steps. First we show $\sup_n \|\boldsymbol{\rho}_n\|$ is bounded by applying the stability of the stochastic approximation (Borkar & Meyn, 2000) into the recursion Eqn. (12). Second, based on this result, we shall show that $\mathbf{u}_n$ goes to 0 in probability by analyzing the recursion Eqn. (13). At last, along with the result that $\mathbf{u}_n$ goes to 0 in probability, by applying the convergence of the stochastic approximation (Borkar & Meyn, 2000) into the recursion Eqn. (12), we show that $\boldsymbol{\rho}_n$ goes to the TD fixed point which is given by the solution of $\mathbf{G}\boldsymbol{\rho} + \mathbf{g} = 0$. The details are provided in Section A.3 of the Appendix. □

Theorem 1 shows that Gradient-DD maintains convergence as GTD2 under some mild conditions. The assumptions (A0), (A1), and (A2) are standard conditions in the convergence analysis of Gradient TD learning algorithms (Sutton et al., 2009a;b; Maei, 2011). The assumption (A3) is weak since it means only that the incremental update in each step is bounded in probability. The assumption (A4) requires that κ is a constant, meaning $\kappa = O(1)$. Given this assumption, the contribution of the term $\kappa\mathbf{H}_n\mathbf{u}_n$ is controlled by α_n as $n \to \infty$.

5 Empirical Study

In this section, we assess the practical utility of the Gradient-DD method in numerical experiments. To validate performance of Gradient-DD learning, we compare Gradient-DD learning with GTD2 learning, TDC learning (TD with gradient correction (Sutton et al., 2009b)), TDRC learning (TDC with regularized correction (Ghiassian et al., 2020)) and TD learning in both tabular representation and linear representation. We conducted three tasks: a simple random walk task, the Boyan-chain task, and Baird's off-policy counterexample. In each task, we evaluate the performance of a learning algorithm by empirical root mean-squared (RMS) error: $\sqrt{\sum_{s\in\mathcal{S}} d_s(V_{\mathbf{w}_n}(s) - V(s))^2}$. The reason we choose the empirical RMS error rather than root projected mean-squared error or other measures as Ghiassian et al. (2018; 2020) used is because it is a direct measure of concern in practical performance.

5.1 Random walk task

As a first test of Gradient-DD learning, we conducted a simple random walk task (Sutton & Barto, 2018). The random walk task has a linear arrangement of m states plus an absorbing terminal state at each end. Thus there are $m+2$ sequential states, $S_0, S_1, \cdots, S_m, S_{m+1}$, where $m = 10$, 20, or 40. Every walk begins in the center state. At each step, the walk moves to a neighboring state, either to the right or to the left with equal probability. If either edge state (S_0 or S_{m+1}) is entered, the walk terminates. A walk's outcome is defined to be $r = 0$ at S_0 and $r = 1$ at S_{m+1}. Our aim is to learn the value of each state $V(s)$, where the true values are $(1, \cdots, m)/(m+1)$. In all cases the approximate value function is initialized to the intermediate value $V_0(s) = 0.5$. We first consider tabular representation of the value function. We also consider linear approximation representation and obtain the similar results, which are reported in Fig. 10 of the appendix. We consider that the learning rate α_n is tapered according to the schedule $\alpha_n = \alpha(10^3+1)/(10^3+n)$. We tune $\alpha \in \{10^{-12/4}, 10^{-11/4}, \cdots, 10^{-1/4}, 1\}$. We also consider the constant step sizes and obtain the similar results, which are reported in Fig. 9 of the appendix. For GTD2 and TDC, we set $\beta_n = \zeta\alpha_n$ with $\zeta \in \{1/64, 1/16, 1/4, 1, 4\}$. We set the $\kappa = 1$. We also investigate the sensitivity to κ in Fig. 6 of the appendix, where we show that $\kappa = 1$ is a good and natural choice in the empirical study.

We first compare the methods by plotting the empirical RMS error from averaging the final 100 steps as a function of step size α in Fig. 2, where 20,000 episodes are used. We also plot the average error of all episodes during training and report these results in Fig. 5 of the Appendix. From these figures, we can make several observations. (1) Gradient-DD clearly performs better than the GTD2 and TDC methods. This advantage is consistent in various settings, and gets bigger as the state space becomes large. (2) Gradient-DD performs similarly to TDRC and conventional TD learning, with a similar dependence on α, although Gradient-DD exhibits greater sensitivity to the value of α in the log domain than these other algorithms. In summary, Gradient-DD exhibits clear advantages over the GTD2 algorithm, and its performance is also as good as TDRC and conventional TD learning.

Next we look closely at the performance during training in Fig. 2. For each method, we tuned $\alpha \in \{10^{-12/4}, \cdots, 10^{-1/4}, 1\}$ by minimizing the average error of the last 100 episodes. We also compare the performance when α is tuned by minimizing the average error of the last 100 episodes and report in Fig. 5 of the appendix. From these results, we draw several observations. (1) For all conditions tested, Gradient-DD converges much more rapidly than GTD2 and TDC. The results also indicate that Gradient-DD even converges as fast as TDRC and conventional TD learning. (2) The advantage of Gradient-DD learning over GTD2 grows as the state space increases in size. (3) Gradient-DD has consistent and good performance under both the constant step size setting and under the tapered step size setting. In summary, the Gradient-DD learning curves in this task show improvements over other gradient-based methods and performance that matches conventional TD learning.

Like TDRC, the updates of Gradient-DD only have a single shared step size α_n, i.e., $\beta_n = \alpha_n$, rather than two independent step sizes α_n and β_n as in the GTD2 and TDC algorithms. A possible concern is that the performance gains in our second-order algorithm could just as easily be obtained with existing methods by adopting this two-timescale approach, where the value function weights are updated with a smaller step size than the second set of weights. Hence, in addition to investigating the effects of the learning rate, size of the state space, and magnitude of the regularization parameter, we also investigate the effect of using

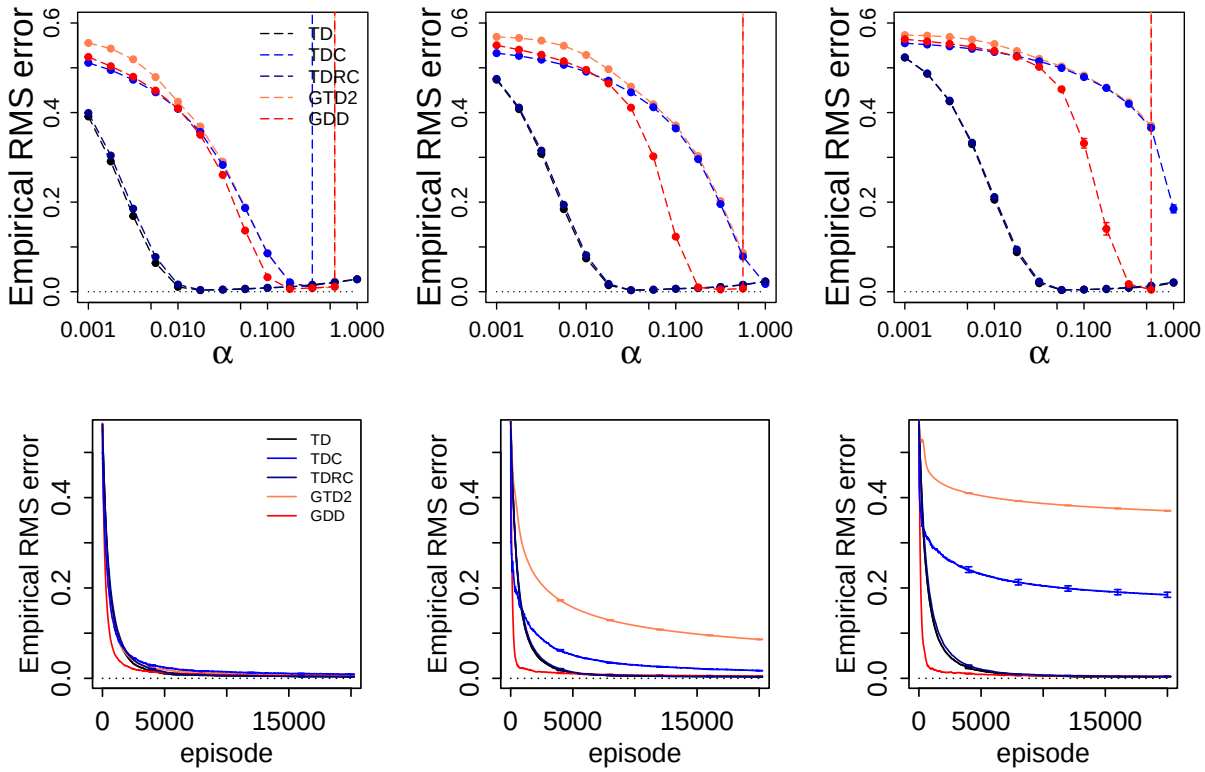


Figure 2: The random walk task with tabular representation and tapering step size $\alpha_n = \alpha(10^3+1)/(10^3+n)$. Upper: Performance as a function of α; Lower: performance over episodes. In each row, state space size 10 (left), 20 (middle), or 40 (right). The curves are averaged over 50 runs, with error bars denoting the standard error of the mean, though most are vanishingly small.

distinct values for the two learning rates, α_n and β_n, although we set $\beta_n = \zeta\alpha_n$ with $\zeta = 1$. To do this, we set $\beta_n = \zeta\alpha_n$ for GDD, with $\zeta \in \{1/64, 1/16, 1/4, 1, 4\}$, and report the results in Fig. 7 of the appendix. The results show that comparably good performance of Gradient-DD is obtained under these various ζ, providing evidence that the second-order difference term in our approach provides an improvement beyond what can be obtained with previous gradient-based methods using the two time scale approach.

5.2 Boyan-chain task

We next investigate Gradient-DD learning on the Boyan-chain problem, which is a standard task for testing linear value-function approximation (Boyan, 2002). In this task we allow for $4p-3$ states, each of which is represented by a p-dimensional feature vector, with $p = 20, 50$, or 100. The p-dimensional representation for every fourth state from the start is $[1, 0, \cdots, 0]$ for state s_1, $[0, 1, 0, \cdots, 0]$ for s_5, $\cdots$, and $[0, 0, \cdots, 0, 1]$ for the terminal state s_{4p-3}. The representations for the remaining states are obtained by linearly interpolating between these. The optimal coefficients of the feature vector are $(-4(p-1), -4(p-2), \cdots, 0)/5$. In each state, except for the last one before the end, there are two possible actions: move forward one step or move forward two steps, where each action occurs with probability 0.5. Both actions lead to reward -0.3. The last state before the end just has one action of moving forward to the terminal with reward -0.2. We tune $\alpha \in \{10^{-2}, 10^{-1.5}, 10^{-1}, 10^{-3/4}, 10^{-1/2}, 10^{-1/4}, 10^{-1/8}, 1, 10^{1/8}, 10^{1/4}\}$ for each method by minimizing the average error of the final 100 episodes. The step size is tapered according to the schedule $\alpha_n = \alpha(2\times 10^3+1)/(2\times 10^3+n)$. For GTD2 and TDC, we set $\beta_n = \zeta\alpha_n$ with $\zeta \in \{1/64, 1/16, 1/4, 1, 4\}$. In this task, we set $\gamma = 1$. As in the random-walk task, we set $\kappa = 1$.

We report the performance as a function of α and the performance over episodes in Fig. 3, where we tune α by the performance based on the average error of the last 100 episodes. We also compare the performance based on the average error of all episodes during training and report the results in Fig. 11 of the appendix. These figures lead to conclusions similar to those already drawn in the random walk task. (1) Gradient-DD has much faster convergence than GTD2 and TDC, and generally converges to better values. (2) Gradient-DD is competitive with TDRC and conventional TD learning despite being somewhat slower at the beginning episodes. (3) The improvement over GTD2 or TDC grows as the state space becomes larger. (4) Comparing the performance with constant step size versus that with tapered step size, the Gradient-DD method performs better with tapered step size than it does with constant step size.

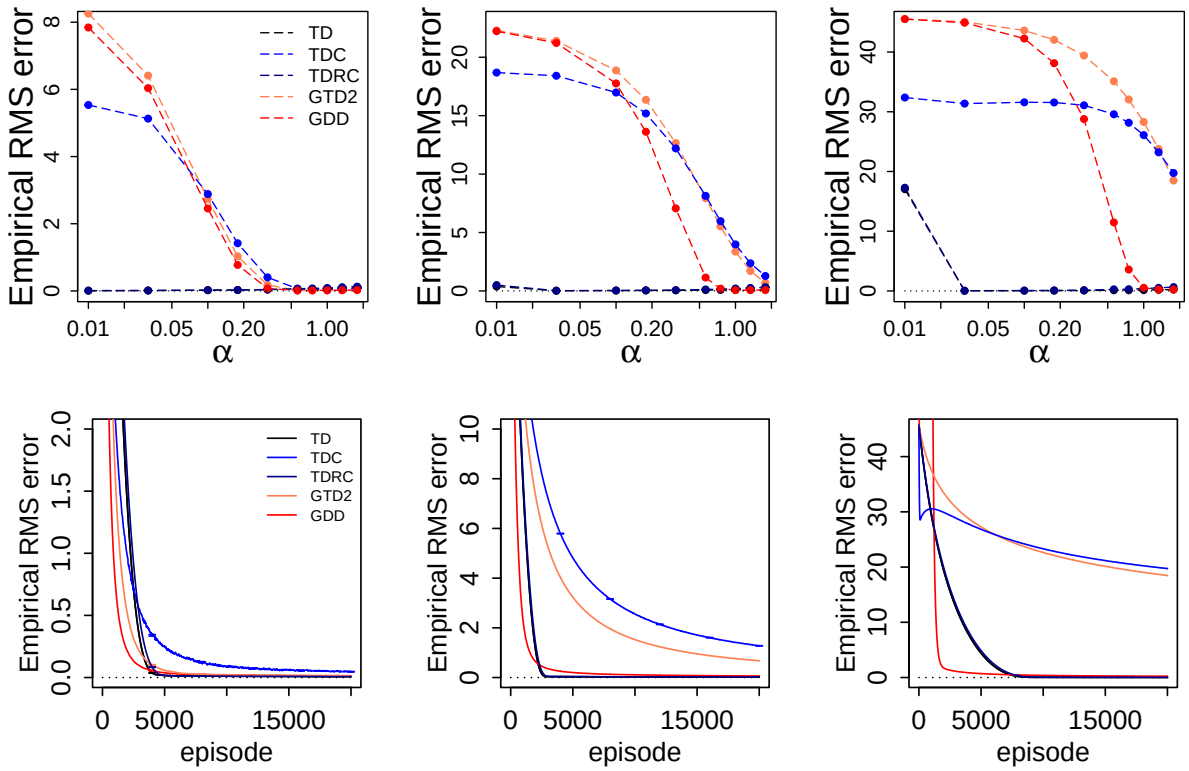
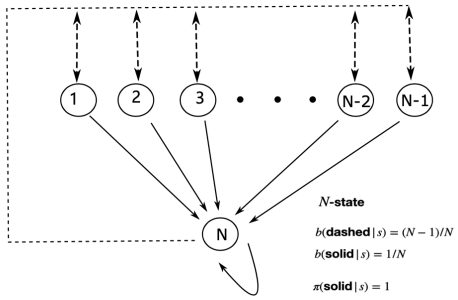


Figure 3: The Boyan Chain task with linear approximation and tapering step size $\alpha_n = \alpha(2 \times 10^3 + 1)/(2 \times 10^3 + n)$. Upper: Performance as a function of α; Lower: performance over episodes. In each row, the feature size is 20 (left), 50 (middle), or 100 (right). The curves are averaged over 50 runs, with error bars denoting the standard error of the mean, though most are vanishingly small across runs.

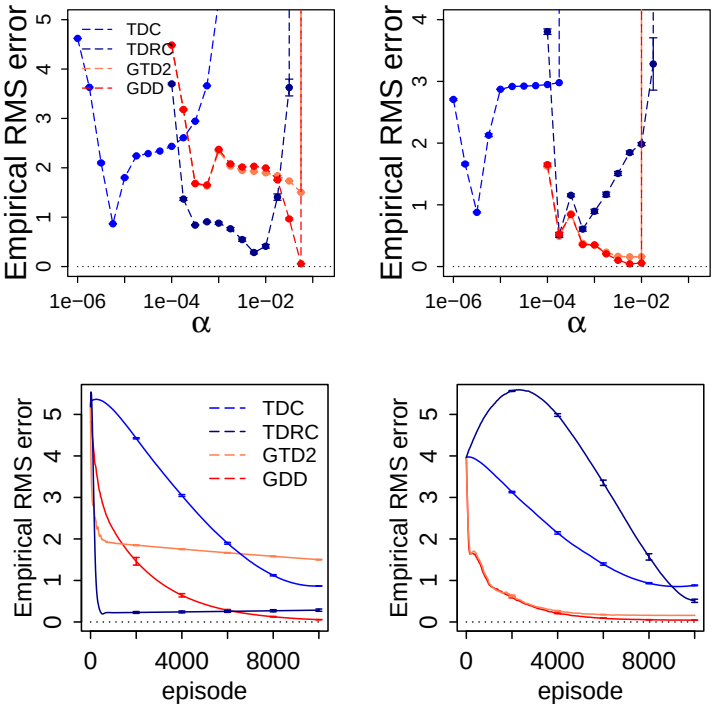
5.3 Baird's off-policy counterexample

We also verify the performance of Gradient-DD on Baird's off-policy counterexample (Baird, 1995; Sutton & Barto, 2018), illustrated schematically in Fig. 4, for which TD learning famously diverges. We show results from Baird's counterexample with $N = 7, 20$ states. The reward is always zero, and the agent learns a linear value approximation with $N + 1$ weights $w_1, \cdots, w_{N+1}$: the estimated value of the j-th state is $2w_j + w_{N+1}$ for $j \leq N - 1$ and that of the N-th state is $w_N + 2w_{N+1}$. In the task, the importance sampling ratio for the dashed action is $(N - 1)/N$, while the ratio for the solid action is N. Thus, comparing different state sizes illustrates the impact of importance sampling ratios in these algorithms. The initial weight values are $(1, \cdots, 1, 10, 1)^\top$. Constant α is used in this task. We set $\gamma = 0.99$. For TDC and GTD2, thus we tune $\zeta \in \{4^{-2}, 4^{-1}, 1, 4^2, 4^3\}$. Meanwhile we tune α for TDC in a wider region. For Gradient-DD, we tune $\kappa \in \{4^{-1}, 1, 4\}$. We tune α separately for each algorithm by minimizing the average error from the final 100 episodes.

Fig. 4 demonstrates that Gradient-DD works better on this counterexample than GTD2, TDC, and TDRC. It is worthwhile to observe that when the state size is 20, TDRC become unstable, meaning serious unbalance of importance sampling ratios may cause TDRC unstable. We also note that, because the linear approximation leaves a residual error in the value estimation due to the projection error, the RMS errors of GTD2, TDC, and TDRC in this task do not go to zero. In contrast to other algorithms, the errors from our Gradient-DD converge to zero.



(a) Illustration of the extended Baird's off-policy counterexample. The "solid" action usually goes to the N-th state, and the "dashed" action usually goes to one of the other $N-1$ states, each with equal probability.



(b) The performance of various algorithms.

Figure 4: Baird's off-policy counterexample. Upper in (b): Performance as a function of α; Lower in (b): performance over episodes. From left to right in (b): 7-state and 20-state.

6 Conclusion and discussion

In this work, we have proposed Gradient-DD learning, a new gradient descent-based TD learning algorithm. The algorithm is based on a modification of the projected Bellman error objective function for value function approximation by introducing a second-order difference term. The algorithm significantly improves upon existing methods for gradient-based TD learning, obtaining better convergence performance than conventional linear TD learning.

Since GTD learning was originally proposed, the Gradient-TD family of algorithms has been extended to incorporate eligibility traces and learning optimal policies (Maei & Sutton, 2010; Geist & Scherrer, 2014), as well as for application to neural networks (Maei, 2011). Additionally, many variants of the vanilla Gradient-TD methods have been proposed, including HTD (Hackman, 2012) and Proximal Gradient-TD (Liu et al., 2016). Because Gradient-DD just modifies the objective error of GTD2 by considering an additional squared-bias term, it may be extended and combined with these other methods, potentially broadening its utility for more complicated tasks.

One potential limitation of our method is that it introduces an additional hyperparameter relative to similar gradient-based algorithms, which increases the computational requirements for hyperparameter optimization. This is somewhat mitigated by our finding that the algorithm's performance is not particularly sensitive to

values of κ, and that $\kappa \sim 1$ was found to be a good choice for the range of environments that we considered. Another potential limitation is that we have focused on value function prediction in the two simple cases of tabular representations and linear approximation, which has enabled us to provide convergence guarantees, but not yet for the case of nonlinear function approximation. An especially interesting direction for future study will be the application of Gradient-DD learning to tasks requiring more complex representations, including neural network implementations. Such approaches are especially useful in cases where state spaces are large, and indeed we have found in our results that Gradient-DD seems to confer the greatest advantage over other methods in such cases. Intuitively, we expect that this is because the difference between the optimal update direction and that chosen by gradient descent becomes greater in higher-dimensional spaces (cf. Fig. 1). This performance benefit in large state spaces suggests that Gradient-DD may be of practical use for these more challenging cases.

References

S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

L. C. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the 12 th International Conference on Machine Learning*, pp. 30–37, 1995.

S. Bhatnagar, R.S. Sutton, M. Ghavamzadeh, and M. Lee. Natural actor-critic algorithm. *Automatic*, 73: 2471–2482, 2009.

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

Justin A. Boyan. Technical update: least-squares temporal difference learning. *Machine Learning*, 49:233–246, 2002.

W. Dabney and P. Thomas. Natural temporal difference learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2014.

T. Degris, P.M. Pilarski, and R.S. Sutton. Model-free reinforcement learning with continuous action in practice. In *Proceedings of the 2012 American Control Conference*, 2012.

S. S. Du, J. Chen, L. Li, L. Xiao, and D. Zhou. Stochastic variance reduction methods for policy evaluation. In *Proceedings of the 34 th International Conference on Machine Learning*, 2017.

M. Geist and B. Scherrer. Off-policy learning with eligibility traces: A survey. *Journal of Machine Learning Research*, 15:289–333, 2014.

S. Ghiassian, A. Patterson, M. White, R.S. Sutton, and A. White. Online off-policy prediction. arXiv:1811.02597, 2018.

S. Ghiassian, A. Patterson, S. Garg, D. Gupta, A. White, and M. White. Gradient temporal-difference learning with regularized corrections. In *International Conference on Machine Learning*, 2020.

L. Hackman. Faster gradient-TD algorithms. Master's thesis, University of Alberta, Edmonton, 2012.

B. Liu, S. Mahadevan, and J. Liu. Regularized off-policy TD-learning. In *Advances in Neural Information Processing Systems*, 2012.

B. Liu, J. Liu, M. Ghavamzadeh, S. Mahadevan, and M. Petrik. Finite-sample analysis of proximal gradient TD algorithms. In *Proceedings of the 31st International Conference on Uncertainty in Artificial Intelligence*, pp. 504–513, 2015.

B. Liu, J. Liu, M. Ghavamzadeh, S. Mahadevan, and M. Petrik. Proximal gradient temporal difference learning algorithms. In *The 25th International Conference on Artificial Intelligence (IJCAI-16),*, 2016.

H.R. Maei. *Gradient temporal-difference learning algorithms*. PhD thesis, University of Alberta, Edmonton, 2011.

H.R. Maei and R.S. Sutton. GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *Proceedings of the 3rd Conference on Artificial General Intelligence*, pp. 91–96, 2010.

A. R. Mahmood, H. van Hasselt, and R. S. Sutton. Weighted importance sampling for off-policy learning with linear function approximation. In *Advances in Neural Information Processing Systems 27*, 2014.

J. Peters, K. Mülling, and Y. Altün. Relative entropy policy search. In *AAAI Conference on Artificial Intelligence*, 2010.

D. Precup, R. S. Sutton, and S. Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the 17 th International Conference on Machine Learning*, 2000.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

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

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition edition, 2018.

R.S. Sutton, H.R. Maei, D. Precup, S. Bhatnagar, D. Silver, Cs. Szepesvári, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th International Conference on Machine Learning*, 2009b.

P. Thomas. GeNGA: A generalization of natural gradient ascent with positive and negative convergence results. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

P. Thomas, B.C. Silva, C. Dann, and E. Brunskill. Energetic natural gradient descent. In *Proceedings of the 33th International Conference on Machine Learning*, 2016.

A. White and M. White. Investigating practical linear temporal difference learning. In *International Conference on Autonomous Agents and Multi-Agent Systems*, 2016.

A Appendix

A.1 On the equivalence of Eqns. (7) & (8)

The Karush-Kuhn-Tucker conditions of Eqn. (8) are the following system of equations

$$\begin{cases} \frac{d}{d\mathbf{w}}J(\mathbf{w}) + \kappa\frac{d}{d\mathbf{w}}(\|\mathbf{V}_{\mathbf{w}} - \mathbf{V}_{\mathbf{w}_{n-1}}\|_{\mathbf{D}}^2 - \mu) = 0; \\ \kappa(\|\mathbf{V}_{\mathbf{w}} - \mathbf{V}_{\mathbf{w}_{n-1}}\|_{\mathbf{D}}^2 - \mu) = 0; \\ \|\mathbf{V}_{\mathbf{w}} - \mathbf{V}_{\mathbf{w}_{n-1}}\|_{\mathbf{D}}^2 \leq \mu; \\ \kappa \geq 0. \end{cases}$$

These equations are equivalent to

$$\begin{cases} \frac{d}{d\mathbf{w}}J(\mathbf{w}) + \kappa\frac{d}{d\mathbf{w}}\|\mathbf{V}_{\mathbf{w}} - \mathbf{V}_{\mathbf{w}_{n-1}}\|_{\mathbf{D}}^2 = 0 \text{ and } \kappa > 0, \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } \|\mathbf{V}_{\mathbf{w}} - \mathbf{V}_{\mathbf{w}_{n-1}}\|_{\mathbf{D}}^2 = \mu; \\ \frac{d}{d\mathbf{w}}J(\mathbf{w}) = 0 \text{ and } \kappa = 0, \text{ if } \|\mathbf{V}_{\mathbf{w}} - \mathbf{V}_{\mathbf{w}_{n-1}}\|_{\mathbf{D}}^2 < \mu. \end{cases}$$

Thus, for any $\mu > 0$, there exists a $\kappa \geq 0$ such that $\frac{d}{d\mathbf{w}}J(\mathbf{w}) + \mu\frac{d}{d\mathbf{w}}\|\mathbf{V}_{\mathbf{w}} - \mathbf{V}_{\mathbf{w}_{n-1}}\|_{\mathbf{D}}^2 = 0$.

A.2 The relation to natural gradients

In this section, we shall show that Gradient-DD is related to, but distinct from, the natural gradient. We thank a reviewer for pointing out the connection between Gradient-DD and the natural gradient.

Following Amari (1998) or Thomas (2014), the natural gradient of $J(\mathbf{w})$ is the direction obtained by solving the following optimization:

$$\lim_{\epsilon\to 0} \arg\min_{\Delta} J(\mathbf{w} + \epsilon\Delta) \text{ s.t. } \epsilon^2\Delta^\top\mathbf{X}^\top\mathbf{D}\mathbf{X}\Delta \leq \mu. \tag{A.1}$$

We can note that this corresponds to the ordinary gradient in the case where the metric tensor $\mathbf{X}^\top\mathbf{D}\mathbf{X}$ is proportional to the identity matrix.

Now we rewrite Eqn. (8) as

$$\|\mathbf{V}_{\mathbf{w}} - \mathbf{V}_{\mathbf{w}_{n-1}}\|_{\mathbf{D}}^2 = (\mathbf{w} - \mathbf{w}_{n-1})^\top\mathbf{X}^\top\mathbf{D}\mathbf{X}(\mathbf{w} - \mathbf{w}_{n-1}).$$

Denote $\epsilon\Delta = \mathbf{w} - \mathbf{w}_{n-1}$, where ϵ is the radius of the circle of $\mathbf{w}$ around $\mathbf{w}_{n-1}$ and Δ is a unit vector. Thus, we have

$$\|\mathbf{V}_{\mathbf{w}} - \mathbf{V}_{\mathbf{w}_{n-1}}\|_{\mathbf{D}}^2 = \epsilon^2\Delta^\top\mathbf{X}^\top\mathbf{D}\mathbf{X}\Delta.$$

For the MSPBE objective, we have

$$J(\mathbf{w}) = J(\mathbf{w}_{n-1} + \mathbf{w} - \mathbf{w}_{n-1}) = J(\mathbf{w}_{n-1} + \epsilon\Delta).$$

Minimizing Eqn. (8) is equivalent to the following optimization

$$\arg\min_{\Delta} J(\mathbf{w}_{n-1} + \epsilon\Delta) \text{ s.t. } \epsilon^2\Delta^\top\mathbf{X}^\top\mathbf{D}\mathbf{X}\Delta \leq \mu. \tag{A.2}$$

In the limit as $\epsilon \to 0$, the above optimization is equivalent to

$$\arg\min_{\Delta} \Delta^\top\nabla J(\mathbf{w}_{n-1}) \text{ s.t. } \epsilon^2\Delta^\top\mathbf{X}^\top\mathbf{D}\mathbf{X}\Delta \leq \mu.$$

Thus, given the metric tensor $\mathbf{G} = \mathbf{X}^\top\mathbf{D}\mathbf{X}$, $-\mathbf{G}^{-1}\nabla J(\mathbf{w}_{n-1})$ is the direction of steepest descent, *i.e.* the natural gradient, of $J(\mathbf{w}_{n-1})$. The natural gradient of $J(\mathbf{w})$, on the other hand, is the direction of steepest descent at $\mathbf{w}$, rather than at $\mathbf{w}_{n-1}$.

Therefore, our Gradient-DD approach makes use of the natural gradient of the objective around $\mathbf{w}_{n-1}$ rather than around $\mathbf{w}_n$ in Eqn. (A.1). This explains the distinction of the updates of our Gradient-DD approach from the updates of directly applying the natural gradient of the objective $\mathbf{w}$.

A.3 Proof of Theorem 1

We introduce an ODE result on stochastic approximation in the following lemma, then show the convergence of our GDD approach in Theorem 1 by applying this result.

Lemma 1. *(Theorems 2.1 & 2.2 of (Borkar & Meyn, 2000)) Consider the stochastic approximation algorithm described by the d-dimensional recursion*

$$\mathbf{y}_{n+1} = \mathbf{y}_n + a_n[f(\mathbf{y}_n) + \mathbf{M}_{n+1}].$$

Suppose the following conditions hold: (c1) The sequence $\{\alpha_n\}$ *satisfies* $0 < \alpha_n < 1$, $\sum_{n=1}^{n} \alpha_n = \infty$, $\sum_{n=1}^{n} \alpha_n^2 < \infty$. *(c2) The function* f *is Lipschitz, and there exists a function* f_∞ *such that* $\lim_{r\to\infty} f_r(\mathbf{y}) = f_\infty(\mathbf{y})$, *where the scaled function* $f_r : \mathbb{R}^d \to \mathbb{R}^d$ *is given by* $f_r(\mathbf{y}) = f(r\mathbf{y})/r$. *(c3) The sequence* $\{\mathbf{M}_n, \mathcal{F}_n\}$, *with* $\mathcal{F}_n = \sigma(\mathbf{y}_i, \mathbf{M}_i, i \le n)$, *is a martingale difference sequence. (c4) For some* $c_0 < \infty$ *and any initial condition* y_0, $E(\|\mathbf{M}_{n+1}\|^2|\mathcal{F}_n) \le c_0(1 + \|\mathbf{y}_n\|^2)$. *(c5) The ODE* $\dot{\mathbf{y}} = f_\infty(\mathbf{y})$ *has the origin as a globally asymptotically stable equilibrium. (c6) The ODE* $\dot{\mathbf{y}}(t) = f(\mathbf{y}(t))$ *has a unique globally asymptotically stable equilibrium* $\mathbf{y}^*$. *Then (1) under the assumptions (c1-c5),* $\sup_n \mathbf{y}_n < \infty$ *in probability. (2) under the assumptions (c1-c6), as* $n \to \infty$, $\mathbf{y}_n$ *converges to* $\mathbf{y}^*$ *with probability 1 .*

Now we investigate the stochastic gradient descent updates in Eqn. (11), which is recalled as follows:

$$\boldsymbol{\rho}_{n+1} = \boldsymbol{\rho}_n - \kappa\alpha_n\mathbf{H}_n(\boldsymbol{\rho}_n - \boldsymbol{\rho}_{n-1}) + \alpha_n(\mathbf{G}_n\boldsymbol{\rho}_n + \mathbf{g}_{n+1}). \tag{A.3}$$

The iterative process in Eqn. (A.3) can be rewritten as

$$(\boldsymbol{\rho}_{n+1} - \boldsymbol{\rho}_n) = -\kappa\alpha_n\mathbf{H}_n(\boldsymbol{\rho}_n - \boldsymbol{\rho}_{n-1}) + \alpha_n(\mathbf{G}_n\boldsymbol{\rho}_n + \mathbf{g}_{n+1}). \tag{A.4}$$

Defining

$$\mathbf{u}_{n+1} = \boldsymbol{\rho}_{n+1} - \boldsymbol{\rho}_n.$$

Eqn. (A.4) becomes

$$\mathbf{u}_{n+1} = -\kappa\alpha_n\mathbf{H}_n\mathbf{u}_n + \alpha_n(\mathbf{G}_n\boldsymbol{\rho}_n + \mathbf{g}_{n+1}).$$

Thus, the iterative process in Eqn. (A.3) is rewritten as two parallel processes that are given by

$$\boldsymbol{\rho}_{n+1} = \boldsymbol{\rho}_n - \kappa\alpha_n\mathbf{H}_n\mathbf{u}_n + \alpha_n(\mathbf{G}_n\boldsymbol{\rho}_n + \mathbf{g}_{n+1}), \tag{A.5}$$

$$\mathbf{u}_{n+1} = -\kappa\alpha_n\mathbf{H}_n\mathbf{u}_n + \alpha_n(\mathbf{G}_n\boldsymbol{\rho}_n + \mathbf{g}_{n+1}). \tag{A.6}$$

Our proofs have three steps. First we shall show $\sup_n \|\boldsymbol{\rho}_n\|$ is bounded by applying the ordinary differential equation approach of the stochastic approximation (the 1st result of Lemma 1) into the recursion Eqn. (A.5). Second, based on this result, we shall show that $\mathbf{u}_n$ goes to 0 in probability by analyzing the recursion Eqn. (A.6). At last, along with the result that $\mathbf{u}_n$ goes to 0 in probability, by applying the 2nd result of Lemma 1 into the recursion Eqn. (A.5), we show that $\boldsymbol{\rho}_n$ goes to the TD fixed point, which is given by the solution of $\mathbf{G}\boldsymbol{\rho} + \mathbf{g} = 0$.

First, we shall show $\sup_n \|\boldsymbol{\rho}_n\|$ is bounded. Eqn. (A.5) is rewritten as

$$\boldsymbol{\rho}_{n+1} = \boldsymbol{\rho}_n + \alpha_n(f(\boldsymbol{\rho}_n) + \mathbf{M}_{n+1}), \tag{A.7}$$

where $f(\boldsymbol{\rho}_n) = (\mathbf{G}\boldsymbol{\rho}_n + \mathbf{g}) - \kappa\mathbf{H}\mathbf{u}_n$ and $\mathbf{M}_{n+1} = ((\mathbf{G}_n - \mathbf{G})\boldsymbol{\rho}_n + \mathbf{g}_{n+1} - \mathbf{g}) - \kappa(\mathbf{H}_n - \mathbf{H})\mathbf{u}_n$. Let $\mathcal{F}_n = \sigma(\mathbf{u}_0, \boldsymbol{\rho}_0, \mathbf{M}_0, \mathbf{u}_1, \boldsymbol{\rho}_1, \mathbf{M}_1, \cdots, \mathbf{u}_n, \boldsymbol{\rho}_n, \mathbf{M}_n)$ be σ-fields generated by the quantities $\mathbf{u}_i, \boldsymbol{\rho}_i, \mathbf{M}_i$, $i \le n$.

Now we verify the conditions (c1-c5) of Lemma 1. Condition (c1) is satisfied under the assumption of the step sizes. Clearly, $f(\mathbf{u})$ is Lipschitz and $f_\infty(\boldsymbol{\rho}) = \mathbf{G}\boldsymbol{\rho}$, meaning Condition (c2) is satisfied. Condition (c3) is satisfied by noting that $(M_n, \mathcal{F}_n)$ is a martingale difference sequence, i.e., $\mathrm{E}(\mathbf{M}_{n+1}|\mathcal{F}_n) = 0$.

We next investigate $\mathrm{E}(\|\mathbf{M}_{n+1}\|^2|\mathcal{F}_n)$. From the triangle inequality, we have that

$$\|\mathbf{M}_{n+1}\|^2 \le 2\|(\mathbf{G}_n - \mathbf{G})\|^2\|\boldsymbol{\rho}_n\|^2 + 2\|\kappa(\mathbf{H}_n - \mathbf{H})\|^2\|\mathbf{u}_n\|^2. \tag{A.8}$$

From Assumption A3 in Theorem 1 that $\|\mathbf{u}_n\|$ is bounded and the Assumption A1 that $(\mathbf{x}_n, r_n, \mathbf{x}_{n+1})$ is an i.i.d. sequence with uniformly bounded second moments, there exists some constant c_0 such that

$$\|\mathbf{G}_n - \mathbf{G}\|^2 \leq c_0/2, \text{ and } \|\kappa(\mathbf{H}_n - \mathbf{H})\|^2 \|\mathbf{u}_n\|^2 \leq c_0/2.$$

Thus, Condition (c4) is satisfied.

Note that $\mathbf{G}$ is defined here with opposite sign relative to $\mathbf{G}$ in (Maei, 2011). From (Sutton et al., 2009a) and (Maei, 2011), the eigenvalues of the matrix $-\mathbf{G}$ are strictly negative under the Assumption A2. Therefore, Condition (c5) is satisfied. Thus, applying the 1st part of Lemma 1 shows that $\sup_n \|\boldsymbol{\rho}_n\|$ is bounded in probability.

Second, we investigate the recursion Eqn. (A.6). Let $\mathbf{y}_{n+1} = (\mathbf{G}_n \boldsymbol{\rho}_n + \mathbf{g}_{n+1})$. Then

$$\begin{aligned}
\mathbf{u}_{n+1} &= \alpha_n[-\kappa \mathbf{H}_n \mathbf{u}_n + \mathbf{y}_{n+1}] \\
&= \alpha_n \mathbf{y}_{n+1} + \alpha_n \alpha_{n-1}(-\kappa \mathbf{H}_n)\mathbf{y}_n + \alpha_n \alpha_{n-1}\alpha_{n-2}(-\kappa \mathbf{H}_n)(-\kappa \mathbf{H}_{n-1})\mathbf{y}_{n-1} \\
&\quad + \cdots + \alpha_n \prod_{k=0}^{n-1} \alpha_k (-\kappa \mathbf{H}_{k+1})\mathbf{y}_1 + \prod_{k=0}^{n} \alpha_k (-\kappa \mathbf{H}_k)\mathbf{u}_0.
\end{aligned} \tag{A.9}$$

Note that $\|\mathbf{H}_n\| \leq 1/\kappa$ due to $\|\mathbf{x}_n\| \leq 1/\kappa$ and that there exists a constant c such that $\|\boldsymbol{\rho}_n\| \leq c$ due to the above result that $\sup_n \|\boldsymbol{\rho}_n\| < \infty$ in probability. Without loss of generality, we assume that $\|\mathbf{x}_n\| \leq 1/\kappa$. Eqn. (A.9) implies that

$$\|\mathbf{u}_{n+1}\| \leq c\left(\alpha_n + \alpha_n\alpha_{n-1} + \alpha_n\alpha_{n-1}\alpha_{n-2} + \cdots + \prod_{k=0}^{n} \alpha_k\right) + \prod_{k=0}^{n} \alpha_k \|\mathbf{u}_0\|. \tag{A.10}$$

Under Assumption A0, $\alpha_n \to 0$ as $n \to 0$. Based on this, Lemma 2 (given in the following section) tells us that $\alpha_n + \alpha_n\alpha_{n-1} + \alpha_n\alpha_{n-1}\alpha_{n-2} + \cdots + \prod_{k=0}^{n} \alpha_k \to 0$ as $n \to 0$. Thus, Eqn. (A.10) implies that $\mathbf{u}_n \to 0$ in probability.

Finally, for applying the 2nd part of Lemma 1, we just need to verify Condition (c6). Because $\mathbf{u}_n$ goes to 0 with probability 1, Eqn. (A.7) tells us that the associated ODE corresponds to

$$\mathbf{G}\boldsymbol{\rho} + \mathbf{g} = 0.$$

Thus, Condition (c6) is satisfied. The theorem is proved.

A.4 A Lemma

Lemma 2. *Denote $\epsilon_n = \alpha_n + \alpha_n\alpha_{n-1} + \cdots + \alpha_n\alpha_{n-1}\cdots\alpha_0$. If $\alpha_n \to 0$ as $n \to \infty$, then $\epsilon_n \to 0$ as $n \to \infty$.*

Proof. Because $\alpha_n \to 0$ as $n \to \infty$, there exists $\alpha \in (0, 1)$ and some integer N such that $\alpha_n \leq \alpha < 1$ when $n \geq N$. Define a sequence ε_n such that

$$\begin{aligned}
\varepsilon_n &= 1 + \alpha\varepsilon_{n-1} \text{ for } n \geq N+1; \\
\varepsilon_N &= \epsilon_N.
\end{aligned}$$

Obviously,

$$\epsilon_n \leq \varepsilon_n, \forall n \geq N. \tag{A.11}$$

Now we investigate the sequence ε_n.

$$\begin{aligned}
\varepsilon_n &= 1 + \alpha\varepsilon_{n-1} = 1 + \alpha(1 + \alpha\varepsilon_{n-2}) = \cdots = 1 + \alpha + \cdots + \alpha^{n-N-1} + \alpha^{n-N}\varepsilon_N \\
&\leq \sum\nolimits_{k=0}^{\infty} \alpha^k + \alpha^{n-N}\varepsilon_N = 1/(1-\alpha) + \alpha^{n-N}\varepsilon_N.
\end{aligned}$$

Thus, we have that

$$\sup_{n \geq N} \varepsilon_n < \infty. \tag{A.12}$$

From Eqns. (A.11) & (A.12), we have

$$\sup_{n\geq 0} \epsilon_n < \infty. \tag{A.13}$$

From the definition of ϵ_n, we have that $\epsilon_n = \alpha_n + \alpha_n \epsilon_{n-1}$. It follows that

$$\alpha_n = \frac{\epsilon_n}{1+\epsilon_{n-1}} \geq \frac{\epsilon_n}{1+\sup_{k\geq 0}\epsilon_k}.$$

From the assumption $\alpha_n \to 0$ as $n \to \infty$ and Eqn. (A.13), we have $\epsilon_n \to 0$ as $n \to \infty$. □

A.5 Additional empirical results

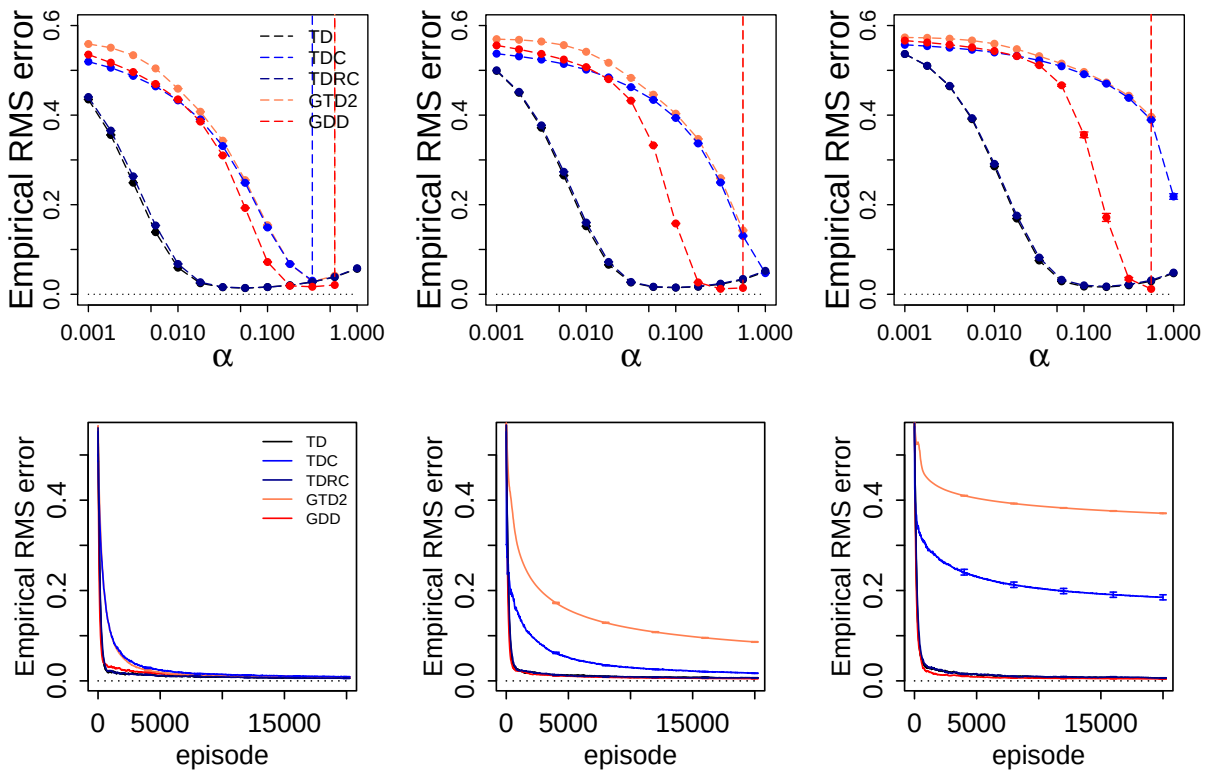


Figure 5: The random walk task with tabular representation. The setting is similar to Fig. 2, but the performance is evaluated by the average error of all episodes, and α is tuned by minimizing the average error of all episodes. Upper: Performance as a function of α; Lower: performance over episodes. From left to right: state space size 10 (left), 20 (middle), or 40 (right).

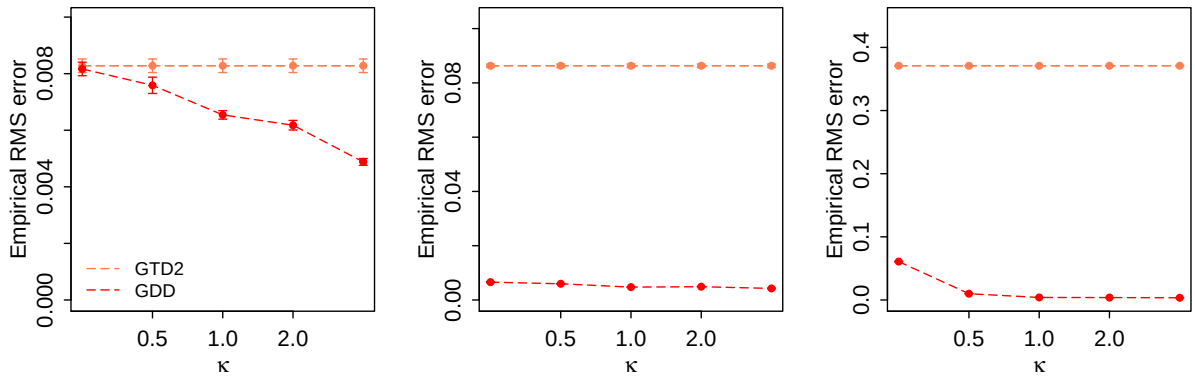


Figure 6: Performance of Gradient-DD in the random walk task in the tabular representation with $\kappa \in \{0.25, 0.5, 1, 2, 4\}$. From left to right: state space size 10 (left), 20 (middle), or 40 (right). In each figure, α is tuned for each algorithm by minimizing the average error of the last 100 episodes. Results are averaged over 50 runs, with error bars denoting standard error of the mean.

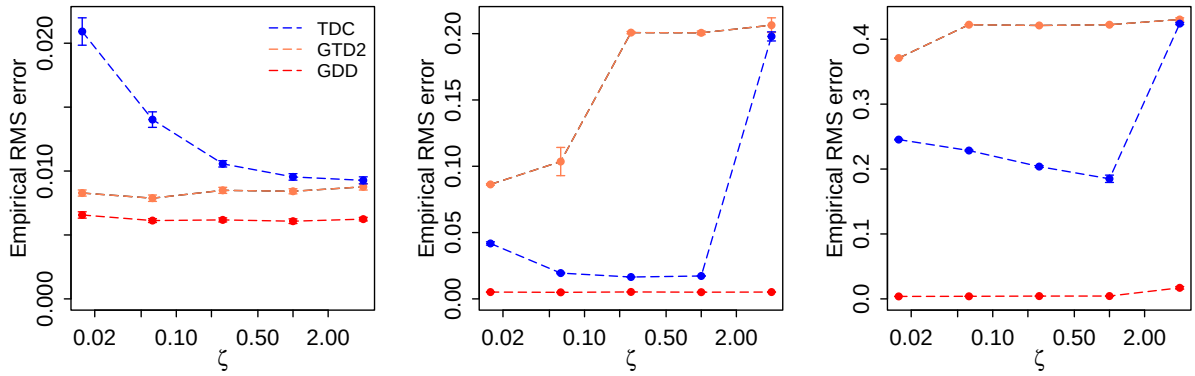


Figure 7: The random walk task in the tabular representation. Performance for various $\beta_n = \zeta\alpha_n$, with $\zeta \in \{4^{-3}, 4^{-2}, 4^{-1}, 1, 4\}$. From left to right in each row: the size of the state space is $m = 10$, $m = 20$, and $m = 40$. In each case α is tuned by by minimizing the average error of the last 100 episodes according to the their performance of corresponding algorithms. Results are averaged over 50 runs, with error bars denoting standard error of the mean.

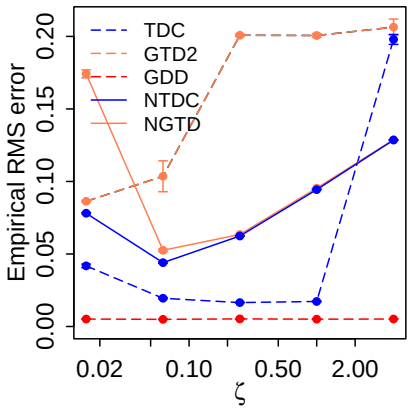


Figure 8: Performance of natural TDC and natural GTD2 in the random walk task with the tabular representation and $m = 20$. Performance for various $\beta_n = \zeta\alpha_n$, with $\zeta \in \{4^{-3}, 4^{-2}, 4^{-1}, 1, 4\}$. In each case α is tuned by by minimizing the average error of the last 100 episodes according to the their performance of corresponding algorithms. "NGTD" and "NTDC" denote the natural gradient-based algorithm of GTD2 and TDC, respectively. Results are averaged over 50 runs, with error bars denoting standard error of the mean.

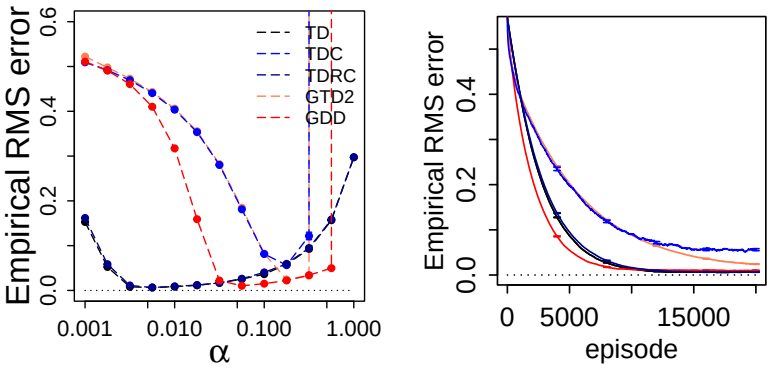


Figure 9: The random walk task with the tabular representation and constant step size. The state size is 20. The curves are averaged over 50 runs, with error bars denoting the standard error of the mean, though most are vanishingly small.

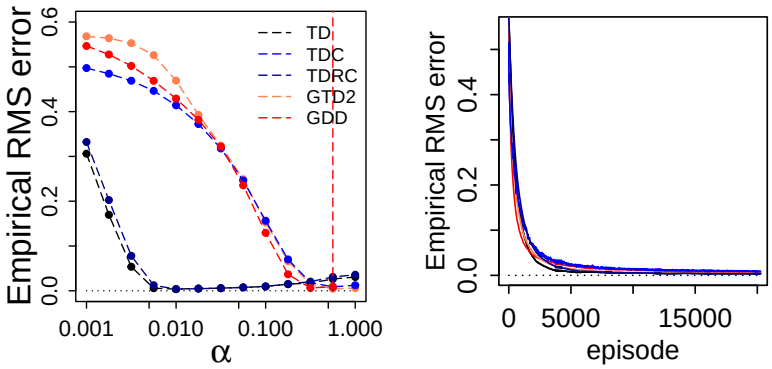


Figure 10: The random walk task with linear value-function approximation, where $\alpha_n = \alpha(10^3+1)/(10^3+n)$. The state size is $m = 20$. Each state is represented by a p-dimensional feature vector with $p = 5$, corresponding to m-state with $m = 20$. The p-dimensional representation for every fourth state from the start is $[1, 0, \cdots, 0]$ for state s_1, $[0, 1, 0, \cdots, 0]$ for s_5, $\cdots$, and $[0, 0, \cdots, 0, 1]$ for state s_{4p+1}. The representations for the intermediate states are obtained by linearly interpolating between these. The sequential states, $S_1, \cdots, S_m$ are obtained by using the first m states above. Left: Performance as α; Right: performance over episodes. The curves are averaged over 50 runs.

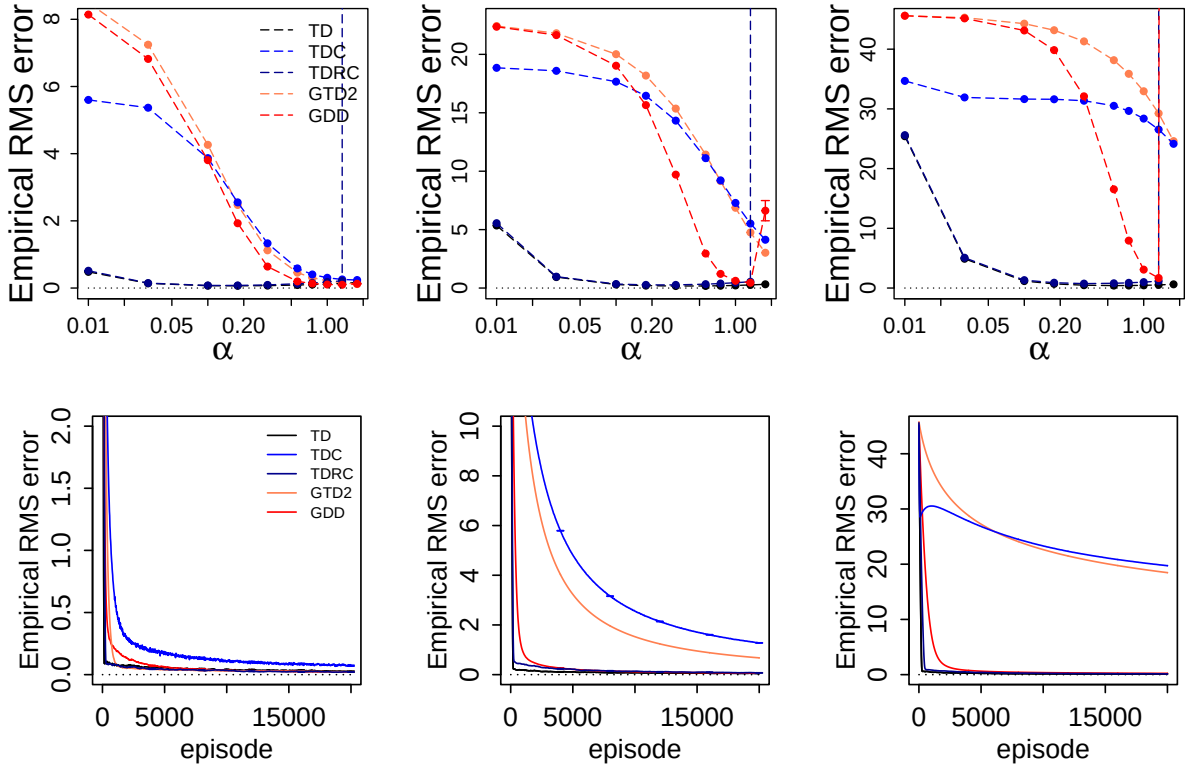


Figure 11: The Boyan Chain task with linear approximation. The setting is similar to Fig. 3, but we evaluate the performance the average error of all episodes, and α is tuned by minimizing the average error of all episodes. Upper: Performance as a function of α; Lower: performance over episodes.