Gradient Temporal-Difference Learning with Regularized Corrections

Sina Ghiassian [*1] **Andrew Patterson** [*1] **Shivam Garg** [1] **Dhawal Gupta** [1] **Adam White** [1 2] **Martha White** [1]

Abstract

It is still common to use Q-learning and temporal difference (TD) learning—even though they have divergence issues and sound Gradient TD alternatives exist—because divergence seems rare and they typically perform well. However, recent work with large neural network learning systems reveals that instability is more common than previously thought. Practitioners face a difficult dilemma: choose an easy to use and performant TD method, or a more complex algorithm that is more sound but harder to tune and all but unexplored with non-linear function approximation or control. In this paper, we introduce a new method called TD with Regularized Corrections (TDRC), that attempts to balance ease of use, soundness, and performance. It behaves as well as TD, when TD performs well, but is sound in cases where TD diverges. We empirically investigate TDRC across a range of problems, for both prediction and control, and for both linear and non-linear function approximation, and show, potentially for the first time, that Gradient TD methods could be a better alternative to TD and Q-learning.

1. Introduction

Off-policy learning—the ability to learn the policy or value function for one policy while following another—underlies many practical implementations of reinforcement learning. Many systems use experience replay, where the value function is updated using previous experiences under many different policies. A similar strategy is employed in asynchronous learning systems that use experience from several different policies to update multiple distributed learners (Espeholt et al., 2018). Off-policy updates can also be used to learn a policy from human demonstrations. In general, many algorithms attempt to estimate the optimal policy from samples generated from a different exploration policy. One of the most widely-used algorithms, Q-learning—a temporal difference (TD) algorithm—is off-policy by design: simply updating toward the maximum value action in the current state, regardless of which action the agent selected.

Both TD and Q-learning, however, have well documented convergence issues, as highlighted in the seminal counterexample by Baird (1995). The fundamental issue is the combination of function approximation, off-policy updates, and bootstrapping: an algorithmic strategy common to sample-based TD learning and Dynamic Programming algorithms (Precup, Sutton & Dasgupta, 2001). This combination can cause the value estimates to grow without bound (Sutton & Barto, 2018). Baird's result motivated over a decade of research and several new off-policy algorithms. The most well-known of these approaches, the Gradient TD methods (Sutton et al., 2009), make use of a second set of weights and importance sampling.

Although sound under function approximation, these Gradient TD methods are not commonly used in practice, likely due to the additional complexity of tuning two learning rate parameters. Many practitioners continue to use unsound approaches such as TD and Q-learning for good reasons. The evidence of divergence is based on highly contrived toy counter-examples. Often, many large scale off-policy learning systems are designed to ensure that the target and behaviour policies are similar—and therefore less off-policy—by ensuring prioritization is mixed with random sampling (Schaul et al., 2016), or frequently syncing the actor policies in asynchronous architectures (Mnih et al., 2016). However, if agents could learn from a larger variety of data streams, our systems could be more flexible and potentially more data efficient. Unfortunately, it appears that current architectures are not as robust under these more aggressive off-policy settings (van Hasselt et al., 2018). This results in a dilemma: the easy-to-use and typically effective TD algorithm can sometimes fail, but the sound Gradient TD algorithms can be difficult to use.

There are algorithms that come close to achieving convergence and lower variance updates without the need to tune multiple stepsize parameters. Retrace (Munos et al., 2016)

[*]Equal contribution [1]Amii, Department of Computing Science, University of Alberta. [2]DeepMind, Alberta. Correspondence to: Sina Ghiassian <ghiassia@ualberta.ca>, Andrew Patterson <ap3@ualberta.ca>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

and its prediction variant Vtrace (Espeholt et al., 2018) reduce the variance of off-policy updating, by clipping importance sampling ratios. These methods, however, are built on off-policy TD and so still have divergence issues (Touati et al., 2018). The sound variants of these algorithms (Touati et al., 2018), and the related work on an algorithm called ABQ (Mahmood, Yu & Sutton, 2017), maintain some of the variance reduction, but rely on Gradient TD to obtain soundness and so inherit the issues therein—the need to tune multiple stepsize parameters. Linear off-policy prediction can be reformulated as a saddlepoint problem, resulting in one time-scale, true gradient descent variant of the GTD2 algorithm (Mahadevan et al., 2014; Liu et al., 2015; Liu et al., 2016). The Emphatic TD algorithm achieves convergence with linear function approximation and off-policy updates using only a single set of weights and thus one stepsize parameter (Sutton et al., 2016). Unfortunately, high variance updates reduce the practicality of the method (White & White, 2016). Finally, Hybrid TD algorithms (Hackman, 2012, White & White, 2016) were introduced to automatically switch between TD updates when the data is on-policy, and gradient-style updates otherwise, thus ensuring convergence. In practice these hybrid methods are more complicated to implement and can have stability issues (White & White, 2016).

In this paper we introduce a new Gradient TD method, called TD with Regularized Corrections (TDRC). With more regularization, the algorithm acts like TD, and with no regularization, it reduces to TD with gradient Corrections (TDC). We find that for an interim level of regularization, TDRC obtains the best of both algorithms, and is not sensitive to this parameter: a regularization parameter of 1.0 was effective across all experiments. We show that our method (1) outperforms other Gradient TD methods overall across a variety of problems, and (2) matches TD when TD performs well while maintaining convergence guarantees. We demonstrate that TDC frequently outperforms the saddlepoint variant of Gradient TD, motivating why we build on TDC and the utility of being able to shift between TD and TDC by setting the regularization parameter. We then highlight why TDRC improves so significantly on TDC, by examining TDC's sensitivity to its second stepsize. We conclude with a demonstration in control, with non-linear function approximation, showing that (1) TDC can perform very well in some settings and very poorly in others, and (2) TDRC is always comparable to Q-learning, and in some cases, is much better.

2. Background

In this paper we tackle the policy evaluation problem in Reinforcement Learning. We model the agent's interactions with its environment as a Markov Decision Process (MDP). The agent and environment interact continually. On each time step $t = 0, 1, 2, \ldots$, the agent selects an action $A_t \in \mathcal{A}$ in state $S_t \in \mathcal{S}$. Based on the agent's action A_t and the transition dynamics, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, the environment transitions into a new state, S_{t+1}, and emits a scalar reward R_{t+1}. The agent selects actions according to its policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. The main objective in policy evaluation is to estimate the value of a state s, defined as the expected discounted sum of future rewards under π:

$$\begin{aligned} v_\pi(s) &\stackrel{\text{def}}{=} \mathbb{E}_\pi\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s\big] \\ &= \mathbb{E}_\pi[G_t | S_t = s]\,, \end{aligned} \tag{1}$$

where $\gamma \in [0, 1]$, $G_t \in \mathbb{R}$ is called the *return*, and $\mathbb{E}_\pi$ is the expectation taken with respect to future states, actions, and rewards generated by π and P.

In many problems of interest, the agent cannot directly observe the state. Instead, on each step the agent observes a featurized representation of the state $\mathbf{x}_t \stackrel{\text{def}}{=} \mathbf{x}(S_t) \in \mathbb{R}^n$, where $n \ll |\mathcal{S}|$. In this setting, the agent cannot estimate the value of each state individually, but must approximate the value with a parametric function. In this paper, we focus on the case of linear function approximation, where the value estimate $\hat{v} : \mathcal{S} \times \mathbb{R}^n \to \mathbb{R}$ is simply formed as an inner product between $\mathbf{x}(s)$ and a learned set of weights $\mathbf{w} \in \mathbb{R}^n$ given by $\hat{v}(s, \mathbf{w}) \stackrel{\text{def}}{=} \mathbf{w}^\top \mathbf{x}(s)$.

Our objective is to adjust $\mathbf{w}_t$ on each time step to construct a good approximation of the true value: $\hat{v} \approx v_\pi$. Perhaps the most well known and successful algorithm for doing so is temporal difference (TD) learning :

$$\begin{aligned} \delta_t &\stackrel{\text{def}}{=} R_{t+1} + \gamma \mathbf{w}_t^\top \mathbf{x}_{t+1} - \mathbf{w}_t^\top \mathbf{x}_t \\ \mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t + \alpha_t \delta_t \mathbf{x}_t \end{aligned} \tag{2}$$

for stepsize $\alpha_t > 0$. TD is guaranteed to be convergent under linear function approximation and on-policy sampling.

The classical TD algorithm was designed for on-policy learning; however, it can be easily extended to the off-policy setting. In *on-policy* learning, the policy used to select actions is the same as the policy used to condition the expectation in the definition of the value function (Eq. 1). Alternatively, we might want to make *off-policy* updates, where the actions are chosen according to some *behavior policy* b, different from the *target policy* π used in Eq. 1. If we view value estimation as estimating the expected return, this off-policy setting corresponds to estimating an expectation conditioned on one distribution with samples collected under another. TD can be extended to make off-policy updates by using importance sampling ratios $\rho_t \stackrel{\text{def}}{=} \frac{\pi(A_t|S_t)}{b(A_t|S_t)} \geq 0$. The resulting algorithm is a minor modification of TD, $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t \rho_t \delta_t \mathbf{x}_t$, where δ_t is defined in Eq. 2.

Off-policy TD can diverge with function approximation, but fortunately there are several TD-based algorithms that

are convergent. When TD learning converges, it converges to the TD fixed point: the weight vector where $\mathbb{E}[\delta_t \mathbf{x}_t] = 0$. Interestingly, TD does not perform gradient descent on any objective to reach the TD fixed point. So, one way to achieve convergence is to perform gradient descent on an objective whose minimum corresponds to the TD-fixed point. Gradient TD methods do exactly this on the Mean Squared Projected Bellman Error (MSPBE) (see Eq. 7).

There are several ways to approximate and simplify the gradient of MSPBE, each resulting in a different algorithm. The two most well-known approaches are TD with Corrections (TDC) and Gradient TD (GTD2). Both these require double the computation and storage of TD, and employ a second set of learned weights $\mathbf{h} \in \mathbb{R}^n$ with a different stepsize parameter $\eta\alpha_t$, where η is a tunable constant. The updates for the TDC algorithm otherwise are similar to TD:

$$\begin{aligned}\mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t + \alpha_t \rho_t \delta_t \mathbf{x}_t - \alpha_t \rho_t \gamma (\mathbf{h}_t^\top \mathbf{x}_t)\mathbf{x}_{t+1} \\ \mathbf{h}_{t+1} &\leftarrow \mathbf{h}_t + \eta\alpha_t \big[\rho_t \delta_t - (\mathbf{h}_t^\top \mathbf{x}_t)\big]\mathbf{x}_t. \end{aligned} \tag{3}$$

The GTD2 algorithm uses the same update for $\mathbf{h}_t$, but the update to the primary weights is different:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t \rho_t (\mathbf{x}_t - \gamma \mathbf{x}_{t+1})(\mathbf{h}_t^\top \mathbf{x}_t). \tag{4}$$

The Gradient TD algorithms are not widely used in practice and are considered difficult to use. In particular, for TDC, the second stepsize has a big impact on performance (White & White, 2016), and the theory suggests that $\eta > 1$ is necessary to guarantee convergence (Sutton et al., 2009).

Attempts to improve Gradient TD methods has largely come from rederiving GTD2 using a saddlepoint formulation of the MSPBE (Mahadevan et al., 2014). This formulation enables us to view GTD2 as a one-time scale algorithm with a single set of weights $[\mathbf{w}, \mathbf{h}]$ using a single global stepsize parameter. In addition, saddlepoint GTD2 can be combined with acceleration techniques like Mirror Prox (Mahadevan et al., 2014) and stochastic variance reduction methods such as SAGA and SVRG (Du et al., 2017). Unfortunately, Mirror Prox has never been shown to improve performance over vanilla GTD2 (White & White, 2016; Ghiassian et al., 2018). Current variance reduction methods like SAGA are only applicable in the offline setting, and extension to the online setting would require new methods (Du et al., 2017). In Appendix B we include comparisons of off-policy prediction algorithms in the batch setting, including recent Kernel Residual Gradient methods (Feng et al., 2019). These experiments suggest that accelerations do not change the relative ranking of the algorithms in the batch setting.

TD is widely considered more sample efficient than all the methods discussed above. A less well-known family of algorithms, called Hybrid methods (Maei, 2011; Hackman, 2012; White & White, 2016), were designed to exploit the sample efficiency of TD when data is generated on-policy—they reduce to TD in the on-policy setting—and use gradient corrections, like TDC, when the data is off-policy. These methods provide some of the ease-of-use benefits of TD, but unfortunately do not enjoy the same level of stability as the Gradient TD methods: for instance, HTD can diverge on Baird's counterexample (White & White, 2016).

3. TD with Regularized Corrections

In this section we develop a new algorithm, called TD with Regularized Corrections (TDRC). The idea is very simple: to regularize the update to the secondary parameters $\mathbf{h}$. The inspiration for the algorithm comes from behavior observed in experiments (see Section 4). Consistently, we find that TDC outperforms—or is comparable to—GTD2 in terms of optimizing the MSPBE; as we reaffirm in our experiments. These results match previous experiments comparing these two algorithms (White & White, 2016; Ghiassian et al., 2018). Previous results suggested that TDC could match TD (White & White, 2016); but, as we highlight in Section 4, this is only when the second stepsize is set so small that TDC is effectively behaving like TD. This behavior is unsatisfactory because to have guaranteed convergence—e.g. on Baird's Counterexample—the second stepsize needs to be large. Further, it is somewhat surprising that attempting to obtain an estimate of the gradient of the MSPBE, as done by TDC, can perform so much more poorly than TD.

Notice that the $\mathbf{h}$ update is simply a linear regression update for estimating the (changing) target δ_t conditioned on $\mathbf{x}_t$, for both GTD2 and TDC. As $\mathbf{w}$ converges, δ_t approaches zero, and consequently $\mathbf{h}$ goes to $\mathbf{0}$ as well. But, a linear regression estimate of $\mathbb{E}[\delta_t | S_t = s]$ is not necessarily the best choice. In fact, using ridge regression—ℓ_2 regularization—can provide a better bias-variance trade-off: it can significantly reduce variance without incurring too much bias. This is in particular true for $\mathbf{h}$, where asymptotically $\mathbf{h} = 0$ and so the bias disappears.

This highlights a potential reason that TD frequently outperforms TDC and GTD2 in experiments: the variance of $\mathbf{h}$. If TD already performs well, it is better to simply use the zero variance but biased estimate $\mathbf{h}_t = \mathbf{0}$. Adding ℓ_2 regularization with parameter β, i.e. $\beta\|\mathbf{h}\|_2^2$, provides a way to move between TD and TDC. For a very large β, $\mathbf{h}$ will be pushed close to zero and the update to $\mathbf{w}$ will be lower variance and more similar to the TD update. On the other hand, for $\beta = 0$, the update reduces to TDC and the estimator $\mathbf{h}$ will be an unbiased estimator with higher variance.

The resulting update equations for TDRC are

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha \rho_t \delta_t \mathbf{x}_t - \alpha \rho_t \gamma (\mathbf{h}_t^\top \mathbf{x})\mathbf{x}_{t+1} \tag{5}$$

$$\mathbf{h}_{t+1} \leftarrow \mathbf{h}_t + \alpha \big[\rho_t \delta_t - (\mathbf{h}_t^\top \mathbf{x}_t)\big]\mathbf{x}_t - \alpha\beta \mathbf{h}_t. \tag{6}$$

The update to $\mathbf{w}$ is the same as TDC, but the update to $\mathbf{h}$ now has the additional term $\alpha\beta\mathbf{h}_t$ which corresponds to the gradient of the ℓ_2 regularizer. The updates only have a single shared stepsize, α, rather than a separate stepsize for the secondary weights $\mathbf{h}$. We make this choice precisely for our motivated reason upfront: for ease-of-use. Further, we find empirically that this choice is effective, and that the reasons for TDC's sensitivity to the second stepsize are mainly due to the fact that a small second stepsize enables TDC to behave like TD (see Section 4.2). Because TDRC has this behavior by design, a shared stepsize is more effective.

While there are many approaches to reduce the variance of the estimator, $\mathbf{h}$, we use an ℓ_2 regularizer because (1) using the ℓ_2 regularizer ensures the set of solutions for TDRC match TD; (2) the resulting update is asymptotically unbiased, because it biases towards the known asymptotic solution of $\mathbf{h}$; and (3) the strongly convex ℓ_2 regularizer improves the convergence rate. TDC convergence proofs impose conditions on the size of the stepsize for $\mathbf{h}$ to ensure that it converges more quickly than the "slow-learner" $\mathbf{w}$, and so increasing convergence rate for $\mathbf{h}$ should make it easier to satisfy this condition. Additionally, the ℓ_2 regularizer biases the estimator $\mathbf{h}$ towards $\mathbf{h} = \mathbf{0}$, the known optimum of the learning system as $\mathbf{w}$ converges. This means that the bias imposed on $\mathbf{h}$ disappears asymptotically, changing only the transient trajectory (we prove this in Theorem 3.1).

As a final remark, we motivate that TDRC should not require a second stepsize, but have introduced a new parameter (β) to obtain this property. The idea, however, is that TDRC should be relatively insensitive to β. The choice of β sweeps between two reasonable algorithms: TD and TDC. If we are already comfortable using TD, then it should be acceptable to use TDRC with a larger β. A smaller β will still result in a sound algorithm, though its performance may suffer due to the variance of the updates in $\mathbf{h}$. In our experiments, we in fact find that TDRC performs well for a wide range of β, and that our default choice of $\beta = 1.0$ works reasonably across all the problems that we tested.

3.1. Theoretically Characterizing the TDRC Update

The MSPBE (Sutton et al., 2009) is defined as

$$\begin{aligned}\text{MSPBE}(\mathbf{w}_t) &\stackrel{\text{def}}{=} \mathbb{E}[\delta_t \mathbf{x}_t]^\top \mathbb{E}\left[\mathbf{x}_t \mathbf{x}_t^\top\right]^{-1} \mathbb{E}[\delta_t \mathbf{x}_t] \qquad (7)\\ &= (-\mathbf{A}\mathbf{w} + \mathbf{b})^\top \mathbf{C}^{-1} (-\mathbf{A}\mathbf{w} + \mathbf{b})\end{aligned}$$

where $\mathbb{E}[\delta_t \mathbf{x}_t] = \mathbf{b} - \mathbf{A}\mathbf{w}_t$ for

$$\mathbf{C} \stackrel{\text{def}}{=} \mathbb{E}\left[\mathbf{x}\mathbf{x}^\top\right], \quad \mathbf{A} \stackrel{\text{def}}{=} \mathbb{E}\left[\mathbf{x}(\mathbf{x} - \gamma\mathbf{x}')^\top\right], \quad \mathbf{b} \stackrel{\text{def}}{=} \mathbb{E}[R\mathbf{x}].$$

The TD fixed point corresponds to $\mathbb{E}[\delta_t \mathbf{x}_t] = \mathbf{0}$ and so to the solution to the system $\mathbf{A}\mathbf{w}_t = \mathbf{b}$. The expectation is taken with respect to the target policy π, unless stated otherwise.

The expected update for TD corresponds to $\mathbb{E}[\delta_t \mathbf{x}_t] = \mathbf{b} - \mathbf{A}\mathbf{w}_t$. The expected update for $\mathbf{w}$ in TDC corresponds to the gradient of the MSPBE,

$$-\frac{1}{2}\nabla \text{MSPBE}(\mathbf{w}_t) = \mathbf{A}^\top \mathbf{C}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{w}_t).$$

Both TDC and GTD2 estimate $\mathbf{h} \stackrel{\text{def}}{=} \mathbf{C}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{w}_t) = \mathbb{E}\left[\mathbf{x}_t \mathbf{x}_t^\top\right]^{-1} \mathbb{E}[\delta_t \mathbf{x}_t]$, to get the least squares estimate $\mathbf{h}^\top \mathbf{x}_t \approx \mathbb{E}[\delta_t | \mathbf{x}_t]$ for targets δ_t. TDC rearranges terms, to sample this gradient differently than GTD2; for a given $\mathbf{h}$, both have the same expected update for $\mathbf{w}$: $\mathbf{A}^\top \mathbf{h}$.

We can now consider the expected update for TDRC. Solving for the ℓ_2 regularized problem with target δ_t, we get $(\mathbb{E}\left[\mathbf{x}_t \mathbf{x}_t^\top\right] + \beta\mathbf{I})\mathbf{h} = \mathbb{E}[\delta_t \mathbf{x}_t]$ which implies $\mathbf{h}_\beta = \mathbf{C}_\beta^{-1}(\mathbf{b} - \mathbf{A}\mathbf{w}_t)$ for $\mathbf{C}_\beta \stackrel{\text{def}}{=} \mathbf{C} + \beta\mathbf{I}$. To get a similar form to TDC, we consider the modified expected update $\mathbf{A}_\beta^\top \mathbf{h}_\beta$ for $\mathbf{A}_\beta \stackrel{\text{def}}{=} \mathbf{A} + \beta\mathbf{I}$. We can get the TDRC update by rearranging this expected update, similarly to how TDC is derived

$$\begin{aligned}&\mathbf{A}_\beta^\top \mathbf{h}_\beta = (\mathbb{E}\left[(\mathbf{x} - \gamma\mathbf{x}')\mathbf{x}^\top\right] + \beta\mathbf{I})\mathbf{h}_\beta\\ &= \left(\mathbb{E}\left[\mathbf{x}\mathbf{x}^\top\right] + \beta\mathbf{I} - \gamma\mathbb{E}\left[\mathbf{x}'\mathbf{x}^\top\right]\right) \mathbf{C}_\beta^{-1} \mathbb{E}[\delta_t \mathbf{x}_t]\\ &= \left(\mathbb{E}\left[\mathbf{x}\mathbf{x}^\top\right] + \beta\mathbf{I}\right) \mathbf{C}_\beta^{-1} \mathbb{E}[\delta_t \mathbf{x}_t] - \gamma\mathbb{E}\left[\mathbf{x}'\mathbf{x}^\top\right] \mathbf{C}_\beta^{-1} \mathbb{E}[\delta_t \mathbf{x}_t]\\ &= \mathbb{E}[\delta_t \mathbf{x}_t] - \gamma\mathbb{E}\left[\mathbf{x}'\mathbf{x}^\top\right] \mathbf{h}_\beta\end{aligned}$$

This update equation for the primary weights looks precisely like the update in TDC, except that our $\mathbf{h}$ is estimated differently. Despite this difference, we show in Theorem I.1 (in Appendix I) that the set of TDRC solutions $\mathbf{w}$ to $\mathbf{A}_\beta^\top \mathbf{h}_\beta = \mathbf{0}$ includes the TD fixed point, and this set is exactly equivalent if $\mathbf{A}_\beta$ is full rank.

In the following theorem (proof in Appendix H) we directly compare convergence of TDRC to TDC. Though the TDRC updates are no longer gradients, we maintain the convergence properties of TDC. This theorem extends the TDC convergence result to allow for $\beta > 0$, where TDC corresponds to TDRC with $\beta = 0$.

Theorem 3.1 (Convergence of TDRC) *Consider the TDRC update, with a TDC like stepsize multiplier $\eta \geq 0$:*

$$\mathbf{h}_{t+1} = \mathbf{h}_t + \eta\alpha_t \left[\rho_t \delta_t - \mathbf{h}_t^\top \mathbf{x}_t\right] \mathbf{x}_t - \eta\alpha_t\beta\, \mathbf{h}_t, \qquad (8)$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \rho_t \delta_t\, \mathbf{x}_t - \alpha_t \rho_t \gamma (\mathbf{h}_t^\top \mathbf{x}_t)\, \mathbf{x}_{t+1}, \qquad (9)$$

*with stepsizes $\alpha_t \in (0, 1]$, satisfying $\sum_{t=0}^\infty \alpha_t = \infty$ and $\sum_{t=0}^\infty \alpha_t^2 < \infty$. Assume that $(\mathbf{x}_t, R_t, \mathbf{x}_{t+1}, \rho_t)$ is an i.i.d. sequence with uniformly bounded second moments for states and rewards, $\mathbf{A} + \beta\,\mathbf{I}$ and $\mathbf{C}$ are non-singular, and that the standard coverage assumption (Sutton & Barto, 2018) holds, i.e. $b(A|S) > 0 \;\; \forall S, A$ where $\pi(A|S) > 0$. Then $\mathbf{w}_t$ converges with probability one to the TD fixed point if **either** of the following are satisfied:*

(i) $\mathbf{A}$ *is positive definite,* ***or***

(ii) $\beta < -\lambda_{max}(\mathbf{H}^{-1}\,\mathbf{A}\,\mathbf{A}^\top)$ *and* $\eta > -\lambda_{min}(\mathbf{C}^{-1}\,\mathbf{H})$*, with* $\mathbf{H} \stackrel{def}{=} \frac{\mathbf{A}+\mathbf{A}^\top}{2}$*. Note that when* $\mathbf{A}$ *is not positive definite,* $-\lambda_{max}(\mathbf{H}^{-1}\,\mathbf{A}\,\mathbf{A}^\top)$ *and* $-\lambda_{min}(\mathbf{C}^{-1}\,\mathbf{H})$ *are guaranteed to be positive real numbers.*

We can extend this result to allow for singular $\mathbf{C}$, which was not possible for TDC. The set of conditions on η and β, however, are more complex. We include this result in Appendix H.4, with conditions given in Eq. 22.

Theorem 3.1 shows that TDRC maintains convergence when TD is convergent: the case when $\mathbf{A}$ is positive definite. Otherwise, TDRC converges under more general settings than TDC, because it has the same conditions on η as given by Maei (2011) but allows for $\beta > 0$. The upper bound on β makes sense, since as $\beta \to \infty$, TDRC approaches TD. Examining the proof, it is likely that the conditions on η could actually be relaxed (see Eq. C3).

One advantage of TDRC is that the matrix $\mathbf{C}_\beta = \mathbf{C} + \beta\,\mathbf{I}$ is non-singular by construction. This raises the question: could we have simply changed the MSPBE objective to use $\mathbf{C}_\beta$ and derived the corresponding TDC-like algorithm? This is easier than TDRC, as the proof of convergence for the resulting algorithm trivially extends the proof from Maei (2011), as the change to the objective function is minimal. We derive corresponding TDC-like update and demonstrate that it performs notably worse than TDRC in Appendix A.

4. Experiments in the Prediction Setting

We first establish the performance of TDRC across several small linear prediction tasks where we carefully sweep hyper-parameters, analyze sensitivity, and average over many runs. The goal is to understand if TDRC has similar performance to TD, with similar parameter sensitivity, but avoids divergence. Before running TDRC, we set $\beta = 1.0$ across all the experiments to refrain from tuning this additional parameter.

4.1. Prediction Problems

In the prediction setting, we investigate three different problems with variations in feature representations, target and behavior policies. We choose problems that have been used in prior work empirically investigating TD methods. The first problem, Boyan's chain (Boyan, 2002), is a 13 state Markov chain where each state is represented by a compact feature representation. This encoding causes inappropriate generalization during learning, but v_π can be represented perfectly with the given features.

Code for all experiments is available at:
https://github.com/rlai-lab/Regularized-GradientTD

The second problem is Baird's (1995) well-known star counterexample. In this MDP, the target and behavior policy are very different resulting in large importance sampling corrections. Baird's Counterexample has been used extensively to demonstrate the soundness of Gradient TD algorithms, so provides a useful testbed to demonstrate that TDRC does not sacrifice soundness for ease-of-use.

Finally, we include a five state random walk MDP. We use three different feature representations: tabular (unit basis vectors), inverted, and dependent features. This last problem was chosen so that we could exactly mirror the experiments used in prior work benchmarking TDC, GTD2, and TD (Sutton et al., 2009). Like Hackman (2012), we used an off-policy variant of the problem. The behavior policy chooses the left and right action with equal probability, and the target policy chooses the right action 60% of the time. Figure 18 in the appendix summarizes all three problems.

We report the total RMSPBE over 3000 steps, measured on each time step, averaged over 200 independent runs. The learning algorithms under study have tunable meta-parameters that can dramatically impact the efficiency of learning. We extensively sweep the values of these meta-parameters (as described in Appendix G), and report both summary performance and the sensitivity of each method to its meta-parameters. For all results reported in the prediction setting, we use the Adagrad (Duchi, Hazan & Singer, 2011) algorithm to adapt a vector of stepsizes for each algorithm. Additional results for constant scalar stepsizes and ADAM vector stepsizes can be found in Appendix B and Appendix E; the conclusions are similar.

4.2. Overall Performance

We first report performance for both the best stepsize as well as provide the parameter sensitivity plots in Figure 1. In the bar plot, we compactly summarize relative performance to TDRC. TDRC performs well across problems, while every other method has at least one setting where it does noticeably worse than TDRC. GTD2 generally learns more slowly than other methods. This result is unsurprising, as it relies so heavily on $\mathbf{h}$ for learning $\mathbf{w}$: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha(\mathbf{x}_t - \gamma\mathbf{x}_{t+1})\mathbf{h}_t^\top\mathbf{x}_t$. In the beginning, when $\mathbf{h}$ is inaccurate, the updates for $\mathbf{w}$ are poor. TDC generally learns much faster. In Boyan's chain, however, TDC seems to suffer from variance in $\mathbf{h}$. The features in this environment cause bigger changes in $\mathbf{h}$ than in the other environments. TDRC, on the other hand, which regularizes $\mathbf{h}$, significantly improves learning in Boyan's chain. TD and HTD perform very well across all problems except for Baird's. Finally, Vtrace—which uses a TD update with importance sampling ratios clipped at 1—performs slightly worse than TD due to the introduced bias, but does not mitigate divergence issues due to off-policy learning in Baird's.

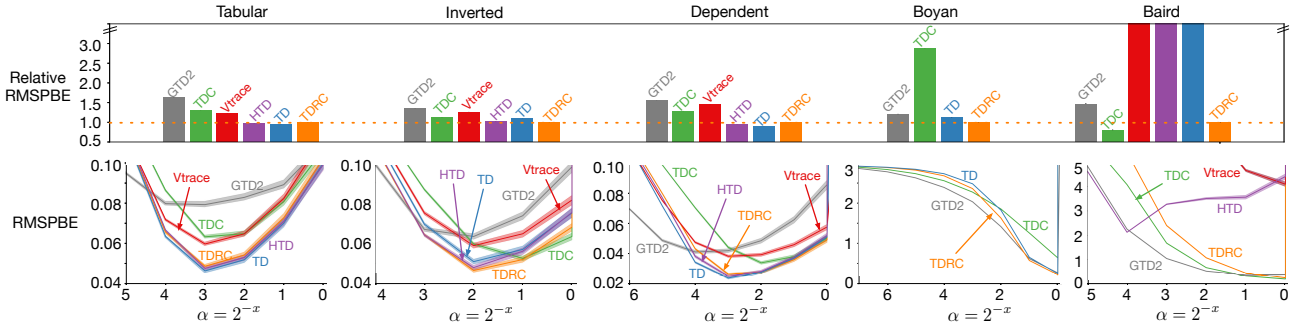

Figure 1. **Top:** The normalized average area under the RMSPBE learning curve for each method on each problem. Each bar is normalized by TDRC's performance so that each problem can be shown in the same range. All results are averaged over 200 independent runs with standard error bars shown at the top of each rectangle, though most are vanishingly small. TD and VTrace both diverge on Baird's Counterexample, which is represented by the bars going off the top of the plot. HTD's bar is also off the plot due to its oscillating behavior. **Bottom:** Stepsize sensitivity measured using average area under the RMSPBE learning curve for each method on each problem. HTD and VTrace are not shown in Boyan's Chain because they reduce to TD for on-policy problems. Values for bar graphs are given in Table 1.

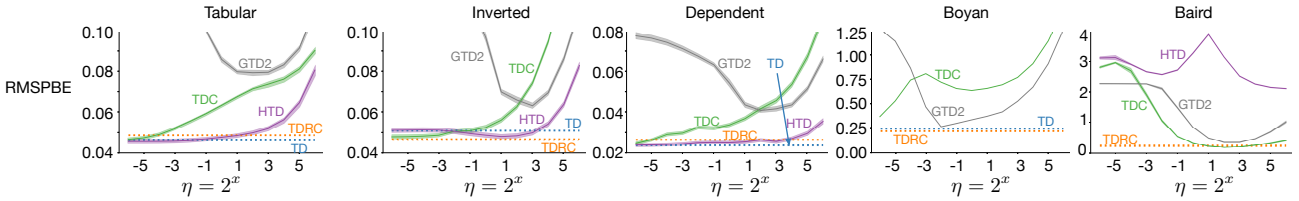

Figure 2. Sensitivity to the second stepsize, for changing parameter η. All methods use Adagrad. All methods are free to choose any value of α for each η. Methods that do not have a second stepsize are shown as a flat line. Values swept are $\eta \in \{2^{-6}, 2^{-5}, \ldots, 2^{5}, 2^{6}\}$.

The results reported here for TDC do not match previous results which indicate performance generally as good as TD (White & White, 2016). The reason for this discrepancy is that previous results carefully tuned the second stepsize $\eta\alpha$ for TDC. The need to tune η is part of the difficulty in using TDC. To better understand the role it is playing here, we include an additional result where we sweep η as well as α for TDC; for completeness, we also include this sweep for GTD2 and HTD. We sweep $\eta \in \{2^{-6}, 2^{-5}, \ldots, 2^{5}, 2^{6}\}$. This allows for $\eta\alpha$ that is very near zero as well as $\eta\alpha$ much larger than α. The theory for TDC suggests η should be larger than 1. The results in Figure 2, however, demonstrate that TDC almost always prefers the smallest η; but for very small η TDC is effectively a TD update. By picking a small η, TDC essentially keeps $\mathbf{h}$ near zero—its initialization—and so removes the gradient correction term. TDC was therefore able to match TD by simply tuning a parameter so that it effectively *was* TD. Unfortunately, this is not a general strategy, for instance in Baird's, TDC picks $\eta \geq 1$ and small η perform poorly.

4.3. Sensitivity to β

So far we have only used TDRC with a regularization parameter $\beta = 1$. This choice was both to avoid over-tuning our method, as well as to show that an intuitive default value could be effective across settings. Intuitively, TDRC should not be sensitive to β, as both TDC ($\beta = 0$) and TD (large β) generally perform reasonably. Picking a $\beta > 0$ should enable TDRC to learn faster like TD—by providing a lower variance correction—as long as it's not too large, to ensure we avoid the divergence issues of TD.

We investigate this intuition by looking at performance across a range of $\beta \in 0.1 * \{2^{0}, 2^{1}, \ldots, 2^{5}, 2^{6}\}$. For $\beta = 0$, we have TDC. Ideally, performance should quickly improve for any non-negligible β, with a large flat region of good performance in the parameter sensitivity plots for a wide range of β. This is generally what we observe in Figure 3. For even very small β, TDRC noticeably improves performance over TDC, getting halfway between TDC and TD (Random Walk with Tabular or Dependent features) or in some cases immediately obtaining the good performance of TD (Random Walk with Inverted Features, Boyan's chain and Baird's). Further, in these three cases, it even performs better or comparably to both TDC and TD for all tested β. Notably, these are the settings with more complex feature representations, suggesting that the regularization parameter helps TDRC learn an $\mathbf{h}$ that is less affected by harmful aliasing in the feature representation. Finally, the results also show that $\beta = 1.0$ was in fact not optimal, and we could have obtained even better results in the previous sec-

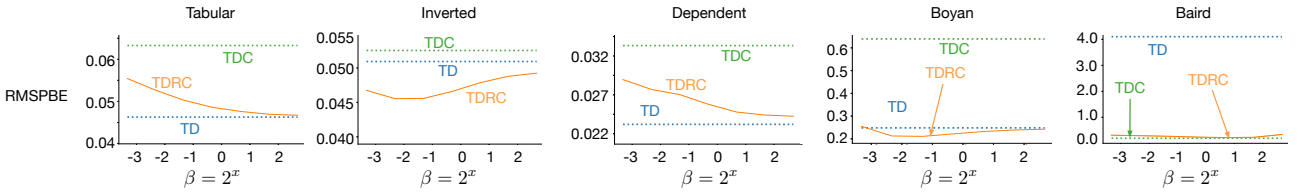

Figure 3. Sensitivity to the regularization parameter, β. TD and TDC are shown as dotted baselines, demonstrating extreme values of β; $\beta = 0$ represented by TDC and $\beta \to \infty$ represented by TD. This experiment demonstrates TDRC's notable insensitivity to β. Its similar range of values across problems, including Baird's counterexample, motivates that β can be chosen easily and is not heavily problem dependent. Values swept are: $\beta \in 0.1 * \{2^0, 2^1, \ldots, 2^5, 2^6\}$.

tion, typically with a larger β. These improvements, though, were relatively marginal over the choice of $\beta = 1.0$.

Naturally, the scale of β should be dependent on the magnitude of the rewards, because in TDRC the gradient correction term is attempting to estimate the expected TD error. One answer is to simply employ adaptive target normalization, such as Pop-Art (van Hasselt et al., 2016), and keep β equal to one. We found TDRC with $\beta = 1$ performed at least as well as TD in on-policy chain domains across a large range of reward scales (see Appendix C).

5. Experiments in the Control Setting

Like TD, TDRC was developed for prediction, under linear function approximation. Again like TD, there are natural—though in some cases heuristic—extensions to the control setting and to non-linear function approximation. In this section, we investigate if TDRC can provide similar improvements in the control setting. We first investigate TDRC in control with linear function approximation, where the extension is more straightforward. We then provide a heuristic strategy to use TDRC—and TDC—with non-linear function approximation. We demonstrate, for the first time, that Gradient TD methods can outperform Q-learning when using neural networks, in two classic control domains and two visual games.

5.1. Extending TDRC to Control

Before presenting the control experiments, we describe how to extend TDRC to control, and to non-linear function approximation. The extension to non-linear function approximation is also applicable in the prediction setting; we therefore begin there. We then discuss the extension to Q-learning which involves estimating action-values for the greedy policy.

Consider the setting where we estimate $\hat{v}(s)$ using a neural network. The secondary weights in TDRC are used to obtain an estimate of $\mathbb{E}[\delta_t | S_t = s]$. Under linear function approximation, this expected TD error is estimated using linear regression with ℓ_2 regularization: $\mathbf{h}^\top \mathbf{x}_t \approx \mathbb{E}[\delta_t | S_t = s]$. With neural networks, this expected TD error can be estimated using an additional head on the network. The target for this second head is still δ_t, with a squared error and ℓ_2 regularization. One might even expect this estimate of $\mathbb{E}[\delta_t | S_t = s]$ to improve, when using a neural network, rather than a hand-designed basis.

An important nuance is that gradients are not passed backward from the error in this second head. This choice is made for simplicity, and to avoid any issues when balancing these two losses. The correction is secondary, and we want to avoid degrading performance in the value estimates simply to improve estimates of $\mathbb{E}[\delta_t | S_t = s]$. It also makes the connection to TD more clear as β becomes larger, as the update to the network is only impacted by $\mathbf{w}$. We have not extensively tested this choice; it remains to be seen if using gradients from both heads might actually be a better choice.

The next step is to extend the algorithm to action-values. For an input state s, the network produces an estimate $\hat{q}(s, a)$ and a prediction $\hat{\delta}(s, a)$ of $\mathbb{E}[\delta_t | S_t = s, A_t = a]$ for each action. The weights $\mathbf{h}_{t+1,A_t}$ for the head corresponding to action A_t are updated using the features produced by the last layer $\mathbf{x}_t$, with $\hat{\delta}(S_t, A_t) = \mathbf{h}_{t,A_t}^\top \mathbf{x}_t$:

$$\mathbf{h}_{t+1,A_t} \leftarrow \mathbf{h}_{t,A_t} + \alpha \big[\delta_t - \mathbf{h}_{t,A_t}^\top \mathbf{x}_t\big] \mathbf{x}_t - \alpha\beta \mathbf{h}_{t,A_t} \quad (10)$$

For the other actions, the secondary weights are not updated since we did not get a target δ_t for them.

The remaining weights $\mathbf{w}_t$, which include all the weights in the network excluding $\mathbf{h}$, are updated using

$$\delta_t = R_{t+1} + \gamma q(S_{t+1}, a') - q(S_t, A_t) \quad (11)$$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha\delta_t \nabla_{\mathbf{w}} \hat{q}(S_t, A_t) - \alpha\gamma \hat{\delta}(S_t, A_t) \nabla_{\mathbf{w}} \hat{q}(S_{t+1}, a')$$

where a' is the action that the policy we are evaluating would take in state S_{t+1}. For control, we often select the greedy policy, and so $a' = \arg\max_a q(S_{t+1}, a)$ and $\delta_t = R_{t+1} + \gamma \max_a q(S_{t+1}, a) - q(S_t, A_t)$ as in Q-learning. This action a' may differ from the (exploratory) action A_{t+1} that is actually executed, and so this estimation is off-policy. There are no importance sampling ratios because we are estimating action-values.

We call this final algorithm QRC: Q-learning with Regularized Corrections. The secondary weights in QRC are initialized to $\mathbf{0}$, to maintain the similarity to TD. We can obtain, as a special case, a control algorithm based on TDC, which we call QC. If we set $\beta = 0$ in Eq. 10, we obtain QC.

We conclude this section by highlighting that there is an alternative route to use TDRC, as is, for control: by using TDRC as a critic within Actor-Critic. We provide the update equations in Appendix G.1.

5.2. Control Problems

We first test the algorithms in a well-understood setting, in which we know Q-learning is effective: Mountain Car with a tile-coding representation. We then use neural network function approximation in two classic control environments—Mountain Car and Cart Pole—and two visual environments from the MinAtar suite (Young & Tian, 2019). For all environments, we fix $\beta = 1.0$ for QRC, $\eta = 1.0$ for QC and do not use target networks (for experiments with target networks see Appendix F).

In the two classic control environments, we use 200 runs, an ϵ-greedy policy with $\epsilon = 0.1$ and a discount of $\gamma = 0.99$. In Mountain Car (Moore, 1990; Sutton, 1996), the goal is to reach the top of a hill, with an underpowered car. The state consists of the agent's position and velocity, with a reward of -1 per step until termination, with actions to accelerate forward, backward or do nothing. In Cart Pole (Barto, Sutton & Anderson, 1983), the goal is to keep a pole balanced as long as possible, by moving a cart left or right. The state consists of the position and velocity of the cart, and the angle and angular velocity of the pole. The reward is +1 per step. An episode ends when the agent fails to balance the pole or balances the pole for more than 500 consecutive steps. For non-linear control experimental details on these environments see Appendix G.3.

For the two MinAtar environments, Breakout and Space Invaders, we use 30 runs, $\gamma = 0.99$ and a decayed ϵ-greedy policy with $\epsilon = 1$ decaying linearly to $\epsilon = 0.1$ over the first 100,000 steps. In Breakout, the agent moves a paddle left and right, to hit a ball into bricks. A reward of +1 is given for every brick hit; new rows appear when all the rows are cleared. The episode ends when the agent misses the ball and it drops. In Space Invaders, the agent shoots alien ships coming towards it, and dodges their fire. A reward of +1 is given for every alien that is shot. The episode ends when the spaceship is hit by alien fire or reached by an alien ship. These environments are simplified versions from the Atari suite, designed to avoid the need for large networks and make it more feasible to complete more exhaustive comparison, including using more runs. All methods use a network with one convolutional layer, followed by a fully connected layer. All experimental settings are identical to the original MinAtar paper (see Appendix G.4 for details).

5.3. Linear Control

We compare TD, TDC and TDRC for control, both within an Actor-Critic algorithm and with their extensions to Q-learning. In Figure 4, we can see two clear outcomes from both control experiments. In both cases, the control algorithm based on TDC fails to converge to a reasonable policy. The TDRC variants, on the other hand, match the performance of TD.

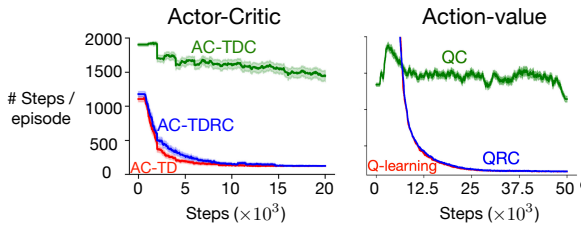

Figure 4. Numbers of steps to reach goal, averaged over runs, versus number of environment steps, in Mountain Car with tile-coded features. **Left:** Comparison of actor-critic control algorithms with various critics with ADAM optimizer. For actor critic experimental details see Appendix G.1. **Right:** Comparison of state-action value control algorithms with constant stepsizes. Stepsizes were swept over $\alpha \in \{2^{-8}, 2^{-7}, \ldots, 2^{-2}, 2^{-1}\}$ and then scaled by the number of active features. We used 16 tilings and 4×4 tiles. Results are averaged over 200 independent runs, with shaded error corresponding to standard error.

This result might be surprising, since the only difference between TDRC and TDC is regularizing $\mathbf{h}$. This small addition, though, seems to play a big role in avoiding this surprisingly bad performance of TDC, and potentially explains why gradient methods have been dismissed as hard-to-use. When we looked more closely at TDC's behavior, we found that the TDC agent improved its behavior policy quickly. But, the magnitude of the gradient corrections also grew rapidly. This high magnitude gradient correction resulted in a higher magnitude gradient for $\mathbf{w}$, and pushed down the learning rate for TDC. The constraint on this correction term provided by TDRC seems to prevent this explosive growth, allowing TDRC to attain comparable performance to the TD-based control agent.

5.4. Non-linear Control

When moving to non-linear function approximation, with neural networks, we find a more nuanced outcome: QC still suffers compared to Q-learning and QRC in the classic control environments—though less than before—yet provides substantial improvements in the two MinAtar environments.

In Figure 5, we find that QC learns more slowly than QRC and Q-learning. Again, QRC brings performance much

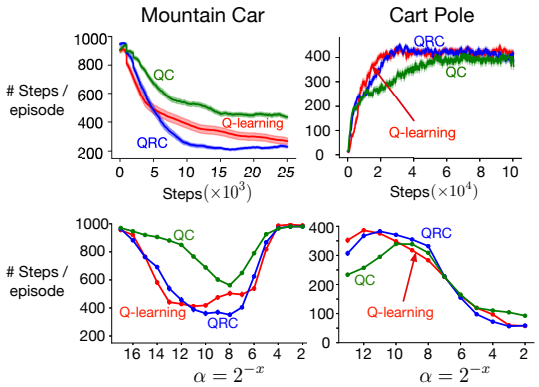

Figure 5. Performance of Q-learning, QC and QRC on two classic control environments. On top the learning curves are shown and at the bottom the parameter sensitivity for various stepsizes. Lower is better for Mountain Car (fewer steps to goal) and higher is better for Cart Pole (more steps balancing the pole). Results are averaged over 200 runs, with shaded error corresponding to standard error.

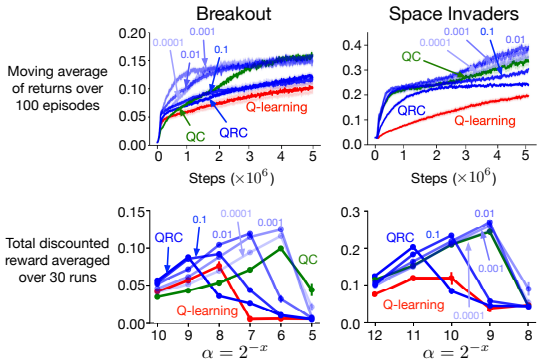

Figure 6. Performance of Q-learning, QC, and QRC in the two MinAtar environments. The learning curves in the top row depict the average return over time for the best performing stepsize for each agent. The stepsize sensitivity plots in the bottom row depict the total discounted reward achieved with several stepsize values. Higher is better. Results are averaged over 30 independent runs, with shaded error corresponding to standard error. Light blue lines show the performance of QRC with smaller regularization parameters, $\beta < 1$.

closer to Q-learning, when QC is performing notably more poorly. In Mountain Car, we tested a more highly off-policy setting: 10 replay steps. By using more replay per step, more data from older policies is used, resulting in a more off-policy data distribution. Under such an off-policy setting, we expect Q-learning to suffer, and in fact, we find that QRC actually performs better than Q-learning. We provide additional experiments on Mountain Car in Appendix D.

On the two MinAtar environments, in Figure 6, we obtain a surprising result: QC provides substantial performance improvements over Q-learning. QRC with $\beta = 1$ is not as performant as QC in this setting and instead obtains performance in-between QC and Q-learning. However, QRC with smaller values of regularization parameter (shown as lighter blue lines) results in the best performance. This outcome highlights that Gradient TD methods are not only theoretically appealing, but could actually be a better alternative to Q-learning in standard (non-adversarially chosen) problems. It further shows that, though QRC with $\beta = 1.0$ generally provides a reasonable strategy, substantial improvements could be obtained with an adaptive method for selecting β.

6. Conclusions and Discussion

In this work, we introduced a simple modification of the TDC algorithm that achieves performance much closer to that of TD. Our algorithm uses a single stepsize like TD, and behaves like TD when TD performs well but also prevents divergence under off-policy sampling. TDRC is built on TDC, and, as we prove, inherits its soundness guarantees. In small linear prediction problems TDRC performs best overall and exhibits low sensitivity to its regularization parameter. In control experiments, with extensions to non-linear function approximation, we find that the resulting algorithm, QRC, performs as well as Q-learning and in some cases notably better. This constitutes the first demonstration of Gradient TD methods outperforming Q-learning, and suggests this simple modification to the standard Q-learning update—to give QRC—could provide a more general purpose algorithm.

An important next step is to better understand the conditions on the regularization parameter β and whether we can truly remove the second stepsize η. The current theorem does not remove conditions on η; in fact, it has the same conditions as TDC. We hypothesize that β should make $\mathbf{h}$ converge more quickly, and so remove the need for the stepsize for the secondary weights to be bigger. Further, the conditions on η and β both depend on domain specific quantities that are generally difficult to compute. In the small prediction problems, we were easily able to confirm that our choices of meta-parameter met the theoretical conditions, however for the larger control problems this remains an open question. In general, developing tight conditions on η and β would help facilitate comfort in using TDRC.

Another important next step is to thoroughly investigate if these empirical results hold in a broader range of environments and settings. The results in this work suggest that TDRC could potentially be a replacement for the widely used TD algorithms. It is only a small modification to an existing TD implementation, and so would not be difficult to adopt. But, to make such a bold claim, much more evidence is needed, particularly because TD has been shown to be so successful for many years.

Acknowledgments

This work was funded by NSERC and CIFAR, particularly through funding the Alberta Machine Intelligence Institute (Amii) and the CCAI Chair program. The authors also gratefully acknowledge funding from JPMorgan Chase & Co. and Google DeepMind. We would like to thank Csaba Szepesvári, Anna Harutyunyan, and the anonymous reviewers for useful feedback. We also thank Banafsheh Rafiee, Andrew Jacobsen, and Alan Chan for helpful discussions during the course of this project.

References

Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. In *International Conference on Machine Learning*, pp. 30–37. Morgan Kaufmann, San Francisco.

Barto, A. G., Sutton, R. S., Anderson, , C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics, 5*, 834-846.

Bellemare, M. G., Naddaf, Y., Veness, J., Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research, 47*, 253-279.

Borkar, V. S., Meyn, S.P. (2000). The O.D.E. Method for Convergence of Stochastic Approximation and Reinforcement Learning. *SIAM J. Control and Optimization*.

Boyan, J.A. (2002). Technical Update: Least-Squares Temporal Difference Learning. *Machine Learning*.

Dai, B., Albert, S., Lihong, L., Lin, X., Niao, H., Zhen, L., Jianshu, C., Le, S. SBEED: Convergent reinforcement learning with nonlinear function approximation. In *International Conference on Machine Learning* (pp. 1125-1134).

Du, S. S., Chen, J., Li, L., Xiao, L., and Zhou, D. (2017) Stochastic variance reduction methods for policy evaluation. In *International Conference on Machine Learning.*

Duchi, J., Hazan, E., Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research 12*:2121-2159.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I. and Legg, S. (2018) IMPALA: Scalable distributed Deep-RL with importance weighted actor-learner architectures. In *International Conference on Machine Learning.*

Feng, Y., Li, L., Liu, Q. (2019). A kernel loss for solving the bellman equation. In *Advances in Neural Information Processing Systems* (pp. 15430-15441).

Ghiassian, S., Patterson, A., White, M., Sutton, R. S., White, A. (2018). Online off-policy prediction. ArXiv:1811.02597.

Glorot, X., Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics* (pp. 249-256).

Hackman, L. (2012). *Faster Gradient TD Algorithms*. M.Sc. thesis, University of Alberta, Edmonton.

Juditsky, A., Nemirovski, A. (2011). *Optimization for Machine Learning*.

Kingma, D. P., Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Liu B, Liu J, Ghavamzadeh M, Mahadevan S, Petrik M (2015). Finite-Sample Analysis of Proximal Gradient TD Algorithms. In *International Conference on Uncertainty in Artificial Intelligence*, pp. 504-513.

Liu B, Liu J, Ghavamzadeh M, Mahadevan S, Petrik M (2016). Proximal Gradient Temporal Difference Learning Algorithms. In *International Joint Conference on Artificial Intelligence*, pp. 4195-4199.

Mahadevan, S., Liu, B., Thomas, P., Dabney, W., Giguere, S., Jacek, N., Gemp, I., Liu, J. (2014). Proximal reinforcement learning: A new theory of sequential decision making in primal-dual spaces. ArXiv:1405.6757.

Mahmood, A. R., Yu, H., Sutton, R. S. (2017). Multi-step off-policy learning without importance sampling ratios. ArXiv:1702.03006.

Maei, H. R. (2011). *Gradient temporal-difference learning algorithms*. Ph.D. thesis, University of Alberta, Edmonton.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928-1937).

Moore, A. W. (1990). *Efficient memory-based learning for robot control*. Ph.D. theis, University of Cambridge.

Munos, R., Stepleton, T., Harutyunyan, A., Bellemare, M. (2016). Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems 29*, pp. 1046–1054.

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

Reddi, S. J., Kale, S., Kumar, S. (2019). On the convergence of adam and beyond. ArXiv:1904.09237.

Schaul, T., Quan, J., Antonoglou, I., Silver, D. (2016). Prioritized experience replay. In *International Conference on Learning Representations.*

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8 (NIPS 1995),* pp. 1038–1044. MIT Press, Cambridge, MA.

Sutton, R. S., Barto, A. G. (2018). *Reinforcement Learning: An Introduction,* Second Edition. MIT Press.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, Cs., Wiewiora, E. (2009). Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *International Conference on Machine Learning*, pp. 993–1000, ACM.

Sutton, R. S., Mahmood A. R., and White M. (2016) An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*.

Touati, A., Bacon, P. L., Precup, D., Vincent, P. (2018). Convergent tree-backup and retrace with function approximation. ArXiv:1705.09322.

van Hasselt, H. P., Guez, A., Hessel, M., Mnih, V., Silver, D. (2016). Learning values across many orders of magnitude. In *Advances in Neural Information Processing Systems* (pp. 4287-4295).

van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., Modayil, J. (2018). Deep Reinforcement Learning and the Deadly Triad. ArXiv:1812.02648

White, A., White, M. (2016). Investigating Practical Linear Temporal Difference Learning. In *International Conference on Autonomous Agents & Multiagent Systems*.

Young, K., Tian, T. (2019). MinAtar: An Atari-Inspired Testbed for Thorough and Reproducible Reinforcement Learning Experiments. ArXiv:1903.03176.