Toward Efficient Gradient-Based Value Estimation

Arsalan Sharifnassab [1] **Richard Sutton** [1]

Abstract

Gradient-based methods for value estimation in reinforcement learning have favorable stability properties, but they are typically much slower than Temporal Difference (TD) learning methods. We study the root causes of this slowness and show that Mean Square Bellman Error (MSBE) is an ill-conditioned loss function in the sense that its Hessian has large condition-number. To resolve the adverse effect of poor conditioning of MSBE on gradient based methods, we propose a low complexity batch-free proximal method that approximately follows the Gauss-Newton direction and is asymptotically robust to parameterization. Our main algorithm, called RANS, is efficient in the sense that it is significantly faster than the residual gradient methods while having almost the same computational complexity, and is competitive with TD on the classic problems that we tested.

1. Introduction

Value estimation is a core problem in reinforcement learning (Sutton & Barto, 2018), and is a key ingredient in several policy optimization methods, e.g., (Bhatnagar et al., 2009; Minh et al., 2015; Lillicrap et al., 2015) . The popular class of value estimation algorithms based on temporal difference learning via forward bootstrapping; including TD(λ) (Sutton, 1988), Expected Sarsa (van Seijen et al., 2009), and Q-learning (Watkins & Dayan, 1992) have found substantial empirical success when combined with proper policy optimization (Minh et al., 2015; Minh et al., 2016; Lillicrap et al., 2015). Nevertheless, these algorithms are not gradient-based optimization methods (Barnard, 1993) and their convergence cannot be guaranteed for general function approximation setting (Baird, 1995; Tsitsiklis & Van Roy, 1997; Brandfonbrener & Bruna, 2019). The stability problem of TD learning has inspired other classes of value estimation algorithms that involve optimizing a loss function through gradient updates. This includes Residual Gradient (RG) algorithm for minimizing Mean Squared Bellman Error (MSBE) (Baird, 1995), Gradient-TD algorithms for minimizing projected Bellman error (Sutton et al., 2009; Maei et al., 2010; Maei, 2011; Hackman, 2012), and their extensions for optimizing a dual formulation of BE2 (Liu et al., 2015; Macua et al., 2014; Dai et al., 2017). These algorithms enjoy the general robustness and convergence properties of Stochastic Gradient Descent (SGD), but are known to be slower than TD in tabular and linear function approximation settings (Baird, 1995; Schoknecht & Merke, 2003; Gordon, 1999; Ghiassian & Sutton, 2021) .

In this paper, we investigate the root causes of the slowness problem of gradient-based value estimation by taking a deeper look into the landscape of MSBE, and propose linear complexity methods to alleviate these problems. We provide theoretical results showing that MSBE is an ill-conditioned loss function in the senses that the condition-number of its Hessian matrix is typically very large. This explains slowness of gradient-based value estimation methods, because gradient descent in general is slow in minimizing ill-conditioned loss functions. In contrast, algorithms like Newton and Gauss-Newton methods are invariant to conditioning of the loss. Unfortunately a direct implementation of these methods requires matrix inversion, which is computationally costly even if computed incrementally. We propose a linear complexity incremental algorithm, called Residual Approximate Gauss-Newton (RAN), that incorporates a trace to approximate the Gauss-Newton direction and then updates the weights along that trace. We show that RAN can be equivalently formulated as a batch-free proximal algorithm. A weakness of RAN is that it requires double sampling (Baird, 1995), which limits its use in stochastic environments. We propose a double-sampling-free extension of RAN by following similar ideas that underlie GTD-type methods. The resulting algorithms significantly outperform RG and GTD2, being orders of magnitudes faster on the simple classic environments that we tested, while having almost similar computational complexity to RG and GTD2.

We then turn our focus to a second cause of slowness of gradient-based value estimation: under function approximation, sample gradients of MSBE involve large outliers

[1]Authors are with the Department of Computing Science, University of Alberta, Canada. Correspondence to: Arsalan Sharifnassab <sharifna@ualberta.ca>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

that carry important information, resulting in large variance of stochastic updates. Outliers of this type often appear in every episode (usually at pre-terminal transitions), and are specific to gradient-based value estimation methods (i.e. such outliers do not appear in TD learning). We propose a general technique called outlier-splitting, which results in no information loss as opposed to the standard clipping methods. Our main value estimation algorithm, called RAN with outlier-Splitting (RANS), has linear computational complexity and has only one effective hyper-parameter (and some other hyper-parameters that can be set to their default values), thanks to its adaptive step-size mechanism. Our empirical results on a few classic control environments with neural network function approximation show significant improvement over RG, and achieving competitive performance to TD.

2. Background

We consider a discounted Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{R}$ is a set of rewards, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{R} \to [0, 1]$ is the environment dynamics determining the probability of the next state and immediate reward given a current state and action pair, and $\gamma \in [0, 1]$ is a discount factor. We fix a stationary policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, and let $p_\pi(s', a', r|s, a) = p(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a)\pi(A_{t+1} = a'|S_{t+1} = s')$. We consider an episodic and online setting where a data stream $(S_1, A_1, R_1), (S_2, A_2, R_2), \ldots$ is generated according to the policy π. The action-value function $q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, at each state s and action a, is the expected discounted sum of rewards obtained by starting from state s and action a and following policy π. We then define the value function $v_\pi : \mathcal{S} \to \mathbb{R}$ as $v_\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[q_\pi(s, a)]$.

In value estimation, we aim to obtain an estimate of the true action-values q_π, usually through a function $q_{\mathbf{w}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ parameterized by a d-dimensional weight vector $\mathbf{w}$. Corresponding to $q_{\mathbf{w}}$ is a Bellman residual at each state and action pair (s, a), defined as

$$\delta_{\mathbf{w}}(s, a) \stackrel{\text{def}}{=} \mathbb{E}_{s',a',r\sim p_\pi(\cdot,\cdot,\cdot|s,a)}\big[r+\gamma q_{\mathbf{w}}(s', a')-q_{\mathbf{w}}(s, a)\big].$$

According to Bellman equations (Sutton & Barto, 2018; Bertsekas & Tsitsiklis, 1996), $q_{\mathbf{w}} = q_\pi$ if and only if $\delta_{\mathbf{w}}(s, a) = 0$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$. In this view, $MSBE(\mathbf{w})$, defined below, serves as a proxy for the quality of estimates $\mathbf{w}$:

$$MSBE_{\mathcal{D}}(\mathbf{w}) \stackrel{\text{def}}{=} \mathbb{E}_{(s,a)\sim\mathcal{D}}\big[\delta_{\mathbf{w}}(s, a)^2\big], \tag{1}$$

where $\mathcal{D}$ is some distribution over states and action pairs. When the distribution is online, we drop the subscript $\mathcal{D}$ and write $MSBE(\cdot)$. For simplicity of notation, we also write $\mathbb{E}_{s,a}[\cdot]$ to denote the expectation with respect to state and action pairs sampled from the online distribution. In the same vein, we consider a parameterized estimate $v_{\mathbf{w}} : \mathcal{S} \to \mathbb{R}$ of value function v_π, and let

$$MSBE^V_{\mathcal{D}}(\mathbf{w}) \stackrel{\text{def}}{=} \mathbb{E}_{s\sim\mathcal{D}}\big[\delta_{\mathbf{w}}(s)^2\big], \tag{2}$$

where $\delta_{\mathbf{w}}(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[\delta_{\mathbf{w}}(s, a)]$, and $\mathcal{D}$ is some distribution over states.

Gradient-based value estimation methods use gradient-based optimization algorithms to minimize MSBE or other related objectives such as MSPBE (Sutton et al., 2009). The first and simplest method in this category is the RG algorithm (Baird, 1995). In this algorithm, to obtain an unbiased sample estimate of $\nabla_{\mathbf{w}}(\delta_{\mathbf{w}}(S_t, A_t)^2)$, we require independent samples (S_{t+1}, A_{t+1}, R_t) and $(S'_{t+1}, A'_{t+1}, R'_t)$ from $p_\pi(\cdot, \cdot, \cdot|S_t, A_t)$. For simplicity of notation, at time t, we let

$$\delta_t \stackrel{\text{def}}{=} R_t + \gamma q_{\mathbf{w}}(S_{t+1}, A_{t+1}) - q_{\mathbf{w}}(S_t, A_t), \tag{3}$$

$$\delta'_t \stackrel{\text{def}}{=} R'_t + \gamma q_{\mathbf{w}}(S'_{t+1}, A'_{t+1}) - q_{\mathbf{w}}(S_t, A_t), \tag{4}$$

where $\mathbf{w}_t = \mathbf{w}$. The RG update is then

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha\delta'_t\nabla_{\mathbf{w}}\delta_t. \tag{5}$$

The requirement for two independent sample transitions at time t is called double sampling (Sutton & Barto, 2018).

In stochastic environments, double sampling is possible only if we have a correct model of the world. In other words, MSBE minimizer is *not learnable* if an exact model of the underlying stochastic environment is not available, which is the case in real-world applications (Sutton & Barto, 2018). A general technique to circumvent double sampling is using Fenchel duality to obtain an equivalent saddle point formulation of MSBE (Dai et al., 2017, Du et al., 2017) as

$$\min_{\mathbf{w}} \max_{\hat{\delta}(\cdot,\cdot)} \mathbb{E}_{s,a}\left[\delta_{\mathbf{w}}(s, a)\,\hat{\delta}(s, a) - \frac{1}{2}\hat{\delta}(s, a)^2\right], \tag{6}$$

where $\hat{\delta}(s, a)$ is an auxiliary variable that serves as a proxy of $\delta_{\mathbf{w}}(s, a)$. In practice, one can consider a parametric approximation $\hat{\delta}_{\boldsymbol{\theta}}(\cdot, \cdot)$ of $\hat{\delta}(\cdot, \cdot)$, and perform gradient updates on the resulting minimax problem $\min_{\mathbf{w}} \max_{\boldsymbol{\theta}} \mathbb{E}_{s,a}\left[\delta_{\mathbf{w}}(s, a)\,\hat{\delta}_{\boldsymbol{\theta}}(s, a) - \frac{1}{2}\hat{\delta}_{\boldsymbol{\theta}}(s, a)^2\right]$:

$$\begin{aligned} \mathbf{w} &\leftarrow \mathbf{w} - \alpha\hat{\delta}_{\boldsymbol{\theta}}(S_t, A_t)\nabla_{\mathbf{w}}\delta_t, \\ \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \eta\big(\delta_t - \hat{\delta}_{\boldsymbol{\theta}}(S_t, A_t)\big)\nabla_{\boldsymbol{\theta}}\hat{\delta}_{\boldsymbol{\theta}}(S_t, A_t), \end{aligned} \tag{7}$$

(Sutton et al., 2009; Liu et al., 2020; Dai et al., 2017). Intuitively, this is similar to the RG algorithm in (5) except for using the parametric approximation $\hat{\delta}_{\boldsymbol{\theta}}(S_t, A_t)$ instead of δ'_t, and updating $\hat{\delta}_{\boldsymbol{\theta}}(s, a)$ by SGD on $\mathbb{E}_{s,a}\big[(\hat{\delta}_{\boldsymbol{\theta}}(s, a) - \delta_{\mathbf{w}}(s, a))^2\big]$. The GTD2 algorithm (Sutton et al., 2009) is a special case of (7) in which $q_{\mathbf{w}}$ and $\hat{\delta}_{\boldsymbol{\theta}}$ are linear approximations of the form $q_{\mathbf{w}}(s, a) = \boldsymbol{\phi}_{s,a}^T\mathbf{w}$ and $\hat{\delta}_{\boldsymbol{\theta}}(s, a) = \boldsymbol{\phi}_{s,a}^T\boldsymbol{\theta}$, for feature vectors $\boldsymbol{\phi}_{s,a}$ (Liu et al., 2020; Dai et al., 2017).

3. MSBE loss is ill-conditioned

The condition-number of a symmetric square matrix, H, is defined as the ratio of its largest to smallest singular values, $\max_{\mathbf{x}:\|\mathbf{x}\|=1} |\mathbf{x}^T H \mathbf{x}| / \min_{\mathbf{y}:\|\mathbf{y}\|=1} |\mathbf{y}^T H \mathbf{y}|$. For a quadratic function $f(\mathbf{x}) = \mathbf{x}^T H \mathbf{x}$, we define the condition-number, $\mathcal{C}(f)$, of f as the condition-number of its Hessian matrix H. Intuitively, level sets (or contours) of a convex quadratic function have an elliptical shape, and the condition-number $\mathcal{C}(f)$ equals the squared ratio between the largest and the smallest diameters of each of these ellipsoids (see Fig. 1). We say that f is ill-conditioned if $\mathcal{C}(f)$ is very large. Then, the level sets of an ill-conditioned quadratic function have the shape of ellipsoids that are thin. It is known that the convergence rate of the gradient descent on a quadratic loss f scales with $\mathcal{C}(f)$ (Polyak, 1964), which can be very slow for ill-conditioned loss functions.

In this section, we consider linear function approximation. In this case, $MSBE^V_{\mathcal{D}}(\cdot)$ defined in (2) is a convex quadratic function. We denote the condition-number of $MSBE^V_{\mathcal{D}}(\cdot)$ under uniform distribution $\mathcal{D}$ by $\mathcal{C}$. We let l be the average episode length, defined as the expected time until termination when starting from a state, uniformly averaged over all states. We also let $h \stackrel{\text{def}}{=} \mathbb{E}_{s\sim\text{unif}}\big[\mathrm{P}(S_{t+1} = s | S_t = s)\big]$ be the the average self-loop probability. Note that h is typically much smaller than 1.

Theorem 3.1. *In the tabular case, the following statements hold for any discount factor $\gamma \in [0, 1]$:*

a) For any MDP and under any policy, we have

$$\mathcal{C} \geq \frac{1-\gamma h}{4} \min\left(\frac{1}{(1-\gamma)^2}, l^2\right) \tag{8}$$

where l is the average episode length and h is the average self-loop probability.

b) For any $n \geq 1$, there exists an n-state MDP and a policy for which $\mathcal{C} \geq \gamma^4 n^2/(1-\gamma)^2$.

The proof is given in Appendix A.1. A similar result also holds for the condition-number of $MSBE$ defined in (1) (see Proposition A.1 in Appendix A.2). Theorem 3.1 shows that MSBE is typically ill-conditioned in the tabular case. This explains the slow convergence of gradient-based methods for minimizing MSBE. As an example, the bound in (8) implies that for $\gamma = .99$ and for any MDP and policy pair with average episode length at least 100 and average self-loop probability no larger than 0.1, we have $\mathcal{C} > 2000$. Moreover, Theorem 3.1 (b) implies that for $\gamma = .99$, there is a 100-state Markov chain for which $\mathcal{C} > 96,000,000$.

Lower bounds similar to Theorem 3.1 are not possible for non-tabular linear function approximation. This is because different feature representations can improve or worsen the condition-number. To see why, note that in the linear function approximation case and under uniform state distribution, $MSBE^V_{\text{unif}}(\mathbf{w}) = \mathbf{w}^T \Phi^T (I - \gamma P)^T (I - \gamma P) \Phi \mathbf{w}$, where Φ is an $n \times d$ matrix, each row of which is a feature vector of a state; and P is the transition matrix. For the specific choice $\Phi = (I - \gamma P)^{-1}$ we obtain $\mathcal{C} = 1$, while for the case that Φ is not full-rank, we have $\mathcal{C} = \infty$.

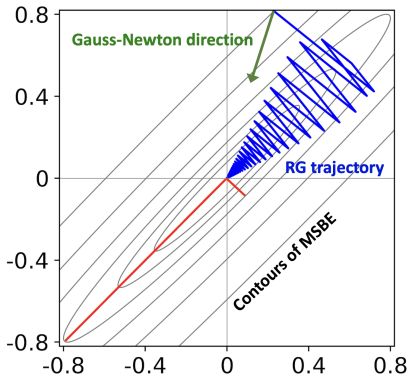

Figure 1. Level sets of MSBE (gray curves) in a 2-state loop environment with $p(s_0 \to s_1) = p(s_1 \to s_0) = 1$ for $\gamma = 0.8$. Here, condition number of MSBE is 81, and is equal to the squared ratio between the diameters (red) of each ellipsoid. The solution trajectory of RG (blue) for $\alpha = .9$ and the Gauss-Newton direction (green) are also depicted. In this environment, $\mathcal{C} = O(1/(1-\gamma)^2)$ (see Theorem 3.1), which rapidly grows for larger γ.

In general, since underparameterized function approximation reduces parameters dimension, it usually improves condition-number. Fig. 2 illustrates dependence of $\mathcal{C}$ on the number of features, d, in an extended Boyan chain environment (Boyan, 2002) with 200 states and with random binary features (see Appendix G.1 for details). We observe that smaller d results in better condition-number; but this comes at the cost of larger value-error at MSBE minimum (the red curve in Fig. 2), where by value-error we mean $\mathbb{E}_{s\sim\text{unif}}\big[\mathbb{E}_{a\sim\pi(\cdot|s)}[\|q_{\mathbf{w}}(s,a) - q_\pi(s,a)\|^2]\big]$. See Appendix B for more experiments on condition number under linear function approximation.

4. A review of the Gauss-Newton method

Consider an expected loss function of the form $F(\mathbf{w}) = \mathbb{E}_f[f^2(\mathbf{w})]$, and the associated Hessian matrix $H_F = \mathbb{E}[\nabla f \nabla f^T] + \mathbb{E}[f\, H_f]$, where H_f denotes Hessian of sample function f. The first term on the right hand side, $\mathbb{E}[\nabla f \nabla f^T]$, is called the Gauss-Newton matrix and is denoted by G. The Gauss-Newton algorithm then updates $\mathbf{w}$ as $\mathbf{w} \leftarrow \mathbf{w} - \alpha G^{-1} \nabla F(\mathbf{w})$. In the special case that functions f are linear, we have $H_f = 0$ and thereby $H_F = G$. In this case, Gauss-Newton and Newton methods become equivalent. However, the Gauss-Newton algorithm has two advantages in the non-linear case. Firstly, $G^{-1}\nabla F(\mathbf{w})$ is

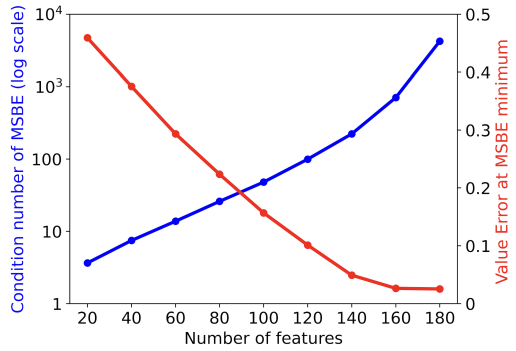

Figure 2. Condition-number of MSBE (blue) and value-error at MSBE minimizer (red) versus number of features, in a 200-state extended Boyan chain with random binary features.

always a descent direction, as opposed to the Newton updates that may climb uphill and converge to local maxima or saddle points (Nesterov & Polyak, 2006). Secondly, G can be computed in terms of gradients while H entails second order derivatives which are not as easily accessible in certain settings (Nocedal & Wright, 1999).

As opposed to gradient descent which is prohibitively slow in ill-conditioned problems, Newton and Gauss-Newton methods are invariant to conditioning of the loss function. Some recent works proposed using Gauss-Newton method for value estimation (Gottwald et al., 2021). However, these algorithms require matrix inversion, which is computationally costly even if computed incrementally via Sherman-Morrison formula with quadratic complexity (Sherman & Morrison, 1950). In the next section, we propose an incremental low complexity approach that approximately follows the Gauss-Newton direction.

5. Our first algorithm: RAN

The Gauss-Newton direction for minimizing MSBE is $\mathbf{m}_{GN}(\mathbf{w}) = G_{\mathbf{w}}^{-1}\nabla MSBE(\mathbf{w})/2$, where $G_{\mathbf{w}} = \mathbb{E}_{s,a}\big[\nabla\delta_{\mathbf{w}}(s,a)\,\nabla\delta_{\mathbf{w}}(s,a)^T\big]$ is the Gauss-Newton matrix and $\nabla MSBE(\mathbf{w}) = 2\mathbb{E}_{s,a}\big[\delta_{\mathbf{w}}(s,a)\,\nabla\delta_{\mathbf{w}}(s,a)\big]$. Then, $\mathbf{m}_{GN}$ is the minimizer of the following quadratic function:

$$L(\mathbf{m}) \stackrel{\text{def}}{=} \frac{1}{2}\,\mathbb{E}_{s,a}\left[\big(\delta_{\mathbf{w}}(s,a) - \nabla\delta_{\mathbf{w}}(s,a)^T\mathbf{m}\big)^2\right]. \quad (9)$$

This is because for any $\mathbf{m}$,

$$\begin{aligned}\nabla_{\mathbf{m}}L(\mathbf{m}) &= \mathbb{E}_{s,a}\left[\big(\nabla\delta_{\mathbf{w}}(s,a)^T\mathbf{m} - \delta_{\mathbf{w}}(s,a)\big)\nabla\delta_{\mathbf{w}}(s,a)\right]\\ &= G_{\mathbf{w}}\mathbf{m} - \nabla MSBE(\mathbf{w})/2,\end{aligned}$$

and therefore $\nabla_{\mathbf{m}}L(\mathbf{m}_{GN}) = \mathbf{0}$. We follow a two time scale approach (Yao et al., 2009, Bhatnagar et al., 2009; Dabney & Thomas, 2014) to incrementally find an approximate minimizer $\mathbf{m}$ of L and update $\mathbf{w}$ along that direction. More concretely, given a $\beta > 0$ and $\lambda \in [0,1]$, at time t, we update $\mathbf{m}$ along an unbiased sample gradient of $\beta L(\mathbf{m}) + (1-\lambda)\|\mathbf{m}\|^2$,

$$\mathbf{m} \leftarrow \lambda\mathbf{m} + \beta\big(\delta'_t - \mathbf{m}^T\nabla\delta'_t\big)\nabla\delta_t, \quad (10)$$

where δ_t and δ'_t are defined in (3) and (4), and $(1-\lambda)\|\mathbf{m}\|^2$ is a Levenberg–Marquardt regularizer (Marquardt, 1963)[1]. We then update $\mathbf{w}$ along $\mathbf{m}$, i.e, $\mathbf{w} \leftarrow \mathbf{w} - \alpha\mathbf{m}$.

Algorithm 1 RAN

Parameters: step-sizes α, β, and decay parameter λ
Initialize: $\mathbf{m} = \mathbf{0}$ and $\mathbf{w}$
for $t = 1, 2, \ldots$ **do**
  consider δ_t and δ'_t defined in (3) and (4), respectively
  $\mathbf{m} \leftarrow \lambda\mathbf{m} + \beta\,\delta'_t\nabla\delta_t$
  $\mathbf{m} \leftarrow \mathbf{m} - \beta(\mathbf{m}^T\nabla\delta_t)\nabla\delta_t$
  $\mathbf{w} \leftarrow \mathbf{w} - \alpha\mathbf{m}$
end for

Algorithm 1 gives the pseudo code of the RAN algorithm. For better stability and faster convergence, the update of $\mathbf{m}$ in RAN is of the form

$$\mathbf{m} \leftarrow \lambda\mathbf{m} + \beta\big(\delta'_t - \mathbf{m}^T\nabla\delta_t\big)\nabla\delta_t. \quad (11)$$

which is the same as (10) except for using $\nabla\delta_t$ instead of $\nabla\delta'_t$. The updates in (11) have lower variance compared to (10), and additionally $\nabla\delta_t\nabla\delta_t^T$ in (11) is positive semi-definite, as opposed to the finite sample estimate of the Gauss-Newton matrix $\frac{1}{\tau}\sum_{t=1}^{\tau}\nabla\delta_t\nabla\delta_t'^T$ in (10). For fixed $\mathbf{w}$, the update in (11) is in expectation along

$$-\big(\mathbb{E}_t[\nabla\delta_t\nabla\delta_t^T] + (1-\lambda)I\big)^{-1}\nabla MSBE(\mathbf{w}). \quad (12)$$

In Appendix C, we provide further intuition for RAN, by presenting a derivation of Algorithm 1 as a proximal method with momentum for minimizing MSBE. In this view, $\mathbf{m}$ serves as a momentum of MSBE gradients, to which we add a correction term equal to the gradient of a penalty function that aims to regularize the change in $\delta_{\mathbf{w}}(s,a)$ for all state action pairs (s,a).

Convergence of the RAN algorithm can be shown in two-time-scale regime where $\alpha_t, \beta_t \to 0$, with α diminishing faster than β (i.e., $\alpha_t/\beta_t \to 0$)[2]. Convergence of such two-time-scale algorithms is well-studied (Kushner & Yin, 2003; Konda & Tsitsiklis, 1999; Bhatnagar et al., 2009),

[1] We have empirically observed that $\lambda = 1$ often leads to slow convergence, because it causes large inertia in $\mathbf{m}$, and therefore large oscillations in $\mathbf{w}$. The best performance is achieved for $\lambda \in (0.99, 0.9999)$.

[2] Note that the two-time-scale view is only for the purpose of convergence analysis, and in practice we consider fixed or adaptive step-sizes whose ratio needs not go to zero.

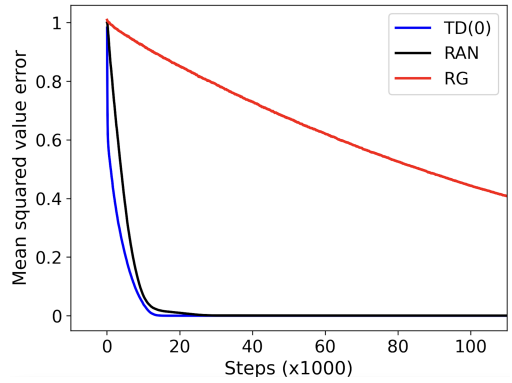

Figure 3. The Hallway experiment discussed in Section 5.

under some smoothness and irreducibly conditions. In Appendix D, we discuss different conditions for convergence of Algorithm 1 in the two-time-scale regime. Moreover, in this regime, RAN is robust to reparameterization:

Proposition 5.1 (Informal)**.** *For $\lambda = 1$ and asymptotically small step-sizes $\alpha \to 0$ and $\alpha/\beta \to 0$, the trajectory of* $\mathbf{w}$ *in the RAN algorithm is invariant to any differentiable and bijective non-linear transformation on parameterization.*

The formal version of Proposition 5.1 and its proof are given in Appendix E.

We evaluated the performance of RAN in a simple benchmark environment. Consider an environment with n states and one action, in which each state $i = 1, 2 \ldots, n$ transits to state $\min(i + 1, n)$ with probability $1 - \epsilon$, and transits to a terminal state with probability ϵ, for some $\epsilon \in [0, 1)$. This is a generalization of the Hallway environment (Baird, 1995), and is known to be a challenging task for the RG algorithm (Baird, 1995). We tested Algorithm 1 in this environment with $n = 50$, $\epsilon = 0.01$, and $\gamma = 0.99$ in the tabular setting (see Appendix G.2 for the details of this experiment). The learning curves are depicted in Fig. 3. We observe that, in this experiment, Algorithm 1 is about 30 times faster than RG, and reaches a convergence rate close to TD(0) (Sutton, 1988; Sutton & Barto, 2018).

6. Double-sampling-free RAN algorithm

In Algorithm 1, we require double sampling to compute δ'_t. In this section, we propose a Double-Sampling-Free version of RAN, called DSF-RAN. Double sampling is easily doable in deterministic environments (Saleh & Jiang, 2019; Zhang et al., 2020), in which case δ'_t can be computed using an independent sample A'_{t+1} from the policy. However, for double sampling in stochastic environments, we require a model to get an independent sample S'_{t+1} of the next state, which is typically possible only in simulated environments.

To resolve the double sampling issue of RAN in stochastic environments, we use the technique discussed in Section 2, which was also used in the GTD2 algorithm. More specifically, instead of δ'_t in Algorithm 1, we use a parametric approximation $\hat{\delta}_{\boldsymbol{\theta}}(S_t, A_t)$ of $\delta_{\mathbf{w}}(S_t, A_t)$, parameterized by $\boldsymbol{\theta}$. Similar to GTD2 (see (7)), we then learn $\boldsymbol{\theta}$ through SGD on $\mathbb{E}_{s,a}\big[(\hat{\delta}_{\boldsymbol{\theta}}(s,a) - \delta_{\mathbf{w}}(s,a))^2\big]$. Pseudo code of DSF-RAN is given in Algorithm 2.

Algorithm 2 DSF-RAN

Parameters: step-sizes α, β, η, and decay parameter λ
Initialize: $\mathbf{m} = \mathbf{0}$, $\mathbf{w}$, $\boldsymbol{\theta}$.
for $t = 1, 2, \ldots$ **do**
$\nabla\delta_t = \gamma \nabla_{\mathbf{w}} q_{\mathbf{w}}(S_{t+1}, A_{t+1}) - \nabla_{\mathbf{w}} q_{\mathbf{w}}(S_t, A_t)$
$\mathbf{m} \leftarrow \lambda \mathbf{m} + \beta\, \hat{\delta}_{\boldsymbol{\theta}}(S_t, A_t)\, \nabla\delta_t$
$\mathbf{m} \leftarrow \mathbf{m} - \beta(\mathbf{m}^T \nabla\delta_t)\nabla\delta_t$
$\mathbf{w} \leftarrow \mathbf{w} - \alpha\mathbf{m}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta\big(\delta_t - \hat{\delta}_{\boldsymbol{\theta}}(S_t, A_t)\big)\nabla_{\boldsymbol{\theta}} \hat{\delta}_{\boldsymbol{\theta}}(S_t, A_t)$
end for

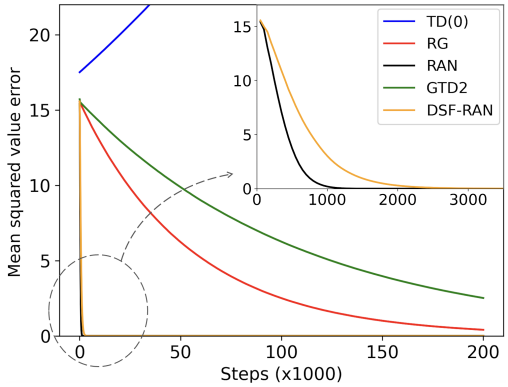

Figure 4. The Baird's star experiment discussed in Section 6

We tested RAN and DNS-RAN algorithms on Baird's Star environment (Baird, 1995), that is a Markov chain with six states, each represented by seven features (see Appendix G.3 for details of this experiment). The results are illustrated in Fig. 4. We observe that in this environment, RAN and DNS-RAN converge about 200 times faster than RG and GTD2 algorithms, respectively. It is well-known that off-policy TD(0) is unstable in this environment (Baird, 1995).

7. The problem of outliers

In this section we argue that the gradient of MSBE involves large outliers and discuss its impact on the RAN algorithm. For simplicity, temporarily suppose that the set of actions is a singleton, $\mathcal{A} = \{a\}$. In the function approximation case, successive states S_t and S_{t+1} often have similar representations. As a result, $\nabla q_{\mathbf{w}}(S_t, a)$ and $\gamma \nabla q_{\mathbf{w}}(S_{t+1}, a)$ are often similar, rendering $\nabla\delta_t = \gamma \nabla q_{\mathbf{w}}(S_{t+1}, a) - \nabla q_{\mathbf{w}}(S_t, a)$ to be small (Zhang et al., 2020). This would not have been

problematic if $\nabla\delta_t$ was small for all t, in which case we could compensate by increasing the step-size. However, $\nabla\delta_t$ can occasionally be large, for example when S_{t+1} is a terminal state in which case $\nabla\delta_t = -\gamma\nabla q_{\mathbf{w}}(S_t, a)$, or when S_{t+1} is far from S_t (e.g., in large jump transitions). Although these *outliers* occur with low probability, they carry important information. For example, the pre-terminal transitions are important because they pin down the estimated values to the terminal values. In environments with larger action sets, if the policy has small entropy, A_{t+1} and A_t would have similar representations with high probability, causing $\nabla\delta_t$ to be small.

We now discuss how these outliers affect RAN. The updates of $\mathbf{m}$ in Algorithm 1 involve a momentum (of MSBE gradient) term $\lambda\mathbf{m} + \delta'_t\nabla\delta_t$ and a correction term $-\beta(\nabla\delta_t^T\mathbf{m})\nabla\delta_t$ that aims to slowly modify $\mathbf{m}$ towards the approximate Gauss-Newton direction. However, when $\nabla\delta_t$ is an outlier, $\beta(\nabla\delta_t^T\mathbf{m})\nabla\delta_t$ can grow very large, cause an overshoot, and completely change the direction of $\mathbf{m}$. In particular, if $\beta\|\nabla\delta_t\|^2 > 1$, then magnitude of the correction term would be larger than the projection of $\mathbf{m}$ on $\nabla\delta_t$, i.e.

$$\big|\langle\beta(\nabla\delta_t^T\mathbf{m})\nabla\delta_t, \nabla\delta_t\rangle\big| > \big|\langle\nabla\delta_t^T\mathbf{m}\rangle\big|, \tag{13}$$

which results in an overshoot along $\nabla\delta_t$. Such overshoots hinder $\mathbf{m}$ from tracking the approximate Gauss-Netwon direction.

To reduce the adverse effect of outliers, one can reduce step-size β, at the cost of slowed down learning. Another popular solution is gradient clipping (Zhang et al., 2019). However, as discussed in the first paragraph of this section, the outliers in our problem carry important information, which can be lost via gradient clipping.

8. Outlier-splitting

We now propose *outlier-splitting* as a general meta-technique for stochastic optimization, appropriate for the case that data contains rare sample functions with abnormally large gradients, and these sample functions carry important information that would be lost in gradient clipping. We first explain the key idea by an example. Consider minimizing $f_1 + \cdots + f_n$ for smooth functions $f_1, \ldots, f_n$. Suppose that f_1 is an outlier in the sense that the norm of its gradient is locally k times larger than the gradient norms of other functions, for some integer $k > 1$. The idea is that instead of applying SGD on $f_1 + \cdots + f_n$, we break down f_1 into k copies of f_1/k and apply SGD on $f_1/k + \cdots + f_1/k + f_2 + \cdots + f_n$ in a random order. The latter updates are outlier-free while being equivalent to the former updates in expectation. We now proceed to a formal description.

Consider SGD on an objective function $F = \mathbb{E}[f]$. For any sample sample function f and any point $\mathbf{w}$, we consider a non-negative measure $\xi(f, \mathbf{w})$; e.g., $\xi(f, \mathbf{w}) = \|\nabla f(\mathbf{w})\|$ or $\|\nabla f(\mathbf{w})\|^2$. Let $\bar{\xi}$ be a trace of ξ, updated by $\bar{\xi} \leftarrow \lambda_\xi\bar{\xi} + (1-\lambda_\xi)\xi(f_t, \mathbf{w}_t)$, where $\lambda_\xi \in (0,1)$ is a constant close to 1. We say that f_t is an outlier if $\xi(f_t, \mathbf{w}_t) \geq \rho\bar{\xi}_t$, for some *outlier threshold* $\rho > 1$.

Algorithm 3 Outlier-splitting for online SGD, applied to loss function $F = \mathbb{E}[f]$

Parameters: step-size β, outlier threshold ρ, trace parameter λ_ξ, outlier sampling probability σ.
Initialize: $\hat{\xi} = 0$, $\mathbf{w}$.
for $t = 1, 2, \ldots$ **do**
  $\hat{\xi} \leftarrow \lambda_\xi\hat{\xi} + (1-\lambda_\xi)\xi(f_t, \mathbf{w})$
  $\bar{\xi} = \hat{\xi}/(1-\lambda_\xi^t)$ ▷ `bias-corrected trace estimate`
  $k = \lfloor \xi(f_t, \mathbf{w})/(\rho\bar{\xi}) \rfloor + 1$
  $\mathbf{w} \leftarrow \mathbf{w} - (\beta/k)\nabla f_t(\mathbf{w})$
  if $k > 1$ **then**
    Store $(f, k, k-1)$ in the outlier buffer
  end if
  With probability $\min(1, \sigma * \text{length of outlier bufffer})$**:**
    Sample (f, k', j) uniformly form outlier buffer
    $k'' = \max\big(k', \lfloor \xi(f, \mathbf{w})/(\rho\bar{\xi}) \rfloor + 1\big)$
    $\mathbf{w} \leftarrow \mathbf{w} - (\beta/k'')\nabla f(\mathbf{w})$
    if $j > 1$ **then**
      Replace (f, k', j) with $(f, k', j-1)$ in the buffer
    else if $j = 1$ **then**
      Remove (f, k', j) from the outlier buffer
    end if
end for

The pseudo code of the outlier-splitting method for online SGD is given in Algorithm 3. At time t of this algorithm, we let

$$k = \left\lfloor \frac{\xi(f_t, \mathbf{w}_t)}{\rho\bar{\xi}_t} \right\rfloor + 1. \tag{14}$$

If f_t is an outlier (equivalently $k > 1$), instead of f_t we pretend to have k copies of f_t/k. We use one of these copies to do a gradient update at time t, and store the remaining $k-1$ copies in a buffer to use them for future updates. These copies are stored in one cell of an *outlier-buffer* as a tuple $(f_t, k, k-1)$, where $k-1$ indicates the number of remaining copies to be used for future updates. In each iteration we perform one update based on the online sample, and perform at most one update based on a sample from the buffer. More concretely, in each iteration t, after applying a gradient update $\mathbf{w} \leftarrow \mathbf{w} - (\beta/k)\nabla f_t(\mathbf{w})$, we take a sample (f, k_f, j) from the outlier buffer with some positive probability, and perform a gradient update $\mathbf{w} \leftarrow \mathbf{w} - (\beta/k_f)\nabla f(\mathbf{w})$.

We now show that the outlier buffer is stable. The expected

number of copies, $k-1$, added to the buffer at time t satisfies

$$\mathbb{E}[k-1] \leq \mathbb{E}_t\left[\frac{\xi(f_t,\mathbf{w}_t)}{\rho\bar{\xi}_t}\right] \simeq \frac{\mathbb{E}_t\left[\xi(f_t,\mathbf{w}_t)\right]}{\rho\mathbb{E}[\bar{\xi}_t]} = \frac{1}{\rho} < 1,$$

where the inequality is due to (14) and the approximate equality is because $\bar{\xi}_t$ is a long-time average. On the other hand, as the length of the outlier buffer increases, the probability of performing a sample update from the buffer goes to 1. In this case, arrival rate to the buffer, $1/\rho$, is smaller than its departure rate, 1; implying stability of the outlier buffer.

9. Our main algorithm: RANS

Our final algorithm, RAN with outlier Splitting (RANS), is a combination of RAN, outlier-splitting, and adaptive step-size ideas. In order to improve updates of $\mathbf{m}$, we employ an adaptive vector step-size $\boldsymbol{\beta}$ that evolves according to a mechanism quite similar to RMSProp (Kochenderfer & Wheeler, 2019), as we discuss next.

Consider a trace vector $\boldsymbol{\nu}_t$ of $(\nabla\delta_t)^2$ updated according to

$$\boldsymbol{\nu}_t \leftarrow \lambda'\boldsymbol{\nu}_{t-1} + (1-\lambda')(\nabla\delta_t)^2,$$

where $(\nabla\delta_t)^2$ is the entrywise square vector of $\nabla\delta_t$, and $\lambda' \in [0,1)$ is a constant. We consider an outlier-measure

$$\xi_t = \langle \frac{1}{\sqrt{\boldsymbol{\nu}_t}} \odot \nabla\delta_t, \nabla\delta_t\rangle \tag{15}$$

where $1/\sqrt{\boldsymbol{\nu}_t}$ is entrywise square root, and $\odot$ and $\langle\cdot,\cdot\rangle$ denote entrywise product and inner product of two vectors, respectively. We then compute the trace $\bar{\xi}$ and k as in Section 8: $\bar{\xi}_t \leftarrow \lambda'\bar{\xi}_t + (1-\lambda')\xi_t$ and $k = \lfloor \xi_t/(\rho\bar{\xi}_t)\rfloor + 1$. We finally fix an $\eta \in (0,1)$ and choose the step-size

$$\boldsymbol{\beta}_t = \frac{\eta}{\rho\bar{\xi}_t}\frac{1}{\sqrt{\boldsymbol{\nu}_t}}. \tag{16}$$

The pseudo code of RANS is given in Algorithm 4 in Appendix F. The algorithm involves applying the outlier-splitting method on the updates of $\mathbf{m}$ in RAN, and using the adaptive step-size in (16).

We now shows that the outlier-splitting mechanism in RANS effectively prevents overshoots of type (13) in the updates of $\mathbf{m}$. Given the above choice of $\boldsymbol{\beta}_t$, we have

$$\begin{aligned}\frac{1}{k}\langle\boldsymbol{\beta}_t\odot\nabla\delta_t,\nabla\delta_t\rangle &= \frac{1}{k}\frac{\eta}{\rho\bar{\xi}_t}\langle\frac{1}{\sqrt{\boldsymbol{\nu}_t}}\odot\nabla\delta_t,\nabla\delta_t\rangle \\ &\leq \frac{\rho\bar{\xi}_t}{\xi_t}\frac{\eta}{\rho\bar{\xi}_t}\langle\frac{1}{\sqrt{\boldsymbol{\nu}_t}}\odot\nabla\delta_t,\nabla\delta_t\rangle = \eta,\end{aligned}$$

where the first equality is from the definition of $\boldsymbol{\beta}_t$ in (16), the inequality is due to the definition of k, and the last equality follows from the definition of ξ_t in (15). This implies that

$$\big|\langle\frac{1}{k}\boldsymbol{\beta}(\nabla\delta_t^T\mathbf{m})\nabla\delta_t,\nabla\delta_t\rangle\big| \leq \eta\big|\nabla\delta_t^T\mathbf{m}\big|. \tag{17}$$

Therefore overshoots of type (13) do not occur in RANS.

The RANS algorithm has hyperparameters α, η, ρ, λ, λ', and σ (the outlier sampling probability). Setting $\eta = 0.2$ and $\rho = 1.2$ are always good choices. Furthermore, our experiments show that the parameters λ, λ', and σ can be set to the default values $\lambda = 0.999$, $\lambda' = 0.9999$, and $\sigma = 0.02$ without much performance degradation. In this case, the RANS algorithm would have essentially one hyper-parameter α, just like RG and TD algorithms with Adam optimizer (Kingma & Ba, 2014). The per-iteration computational complexity of RANS is at most twice the RG algorithm with Adam optimizer.

10. Experiments

Similar to TD, the RANS algorithm can be utilized within any control loop. More specifically, one can used RAN instead of TD to estimate Q-values and plug these estimates into the policy update of interest, including actor-critic algorithms like A3C (Minh et al., 2016), deterministic policy gradient algorithms (Silver et al., 2014) like DDPG (Lillicrap et al., 2015), and greedy/soft-max policy updates like DQN (Minh et al., 2015). In this section, we assess the performance of softmax policy updates and deterministic policy gradient methods when the Q-functions in these algorithms are calculated using RANS.

For softmax policy updates, we conducted experiments on Acrobot and Cartpole environments. We used a single-layer neural network with 64 hidden units with ReLU activation to learn the action-values via three algorithms, TD(0), RG, and RANS. The actions are chosen according to a softmax distribution on the action-values. Fig. 5 illustrates expected returns versus number of step. We trained TD(0) and RG using Adam optimizer. Refer to Appendix G.4 for complementary experimental results and details of the experiments. The results show that the RANS algorithm outperforms RG and TD on these environments.

For the deterministic policy gradient actor updates, we conducted experiments on MuJoCo environments: Hopper and HalfCheetah. We employed a two-layer feedforward neural network with 400 and 300 hidden ReLU units respectively for both the actor and critic, and a final tanh unit following the output of the actor. The actor was trained using deterministic policy gradient policy updates (Silver et al., 2014; Lillicrap et al., 2015), while the critics were trained by three algorithms: RANS, Adam TD with delayed target network update, and Adam RG. We considered an on-policy setting where samples are drawn from the current policy

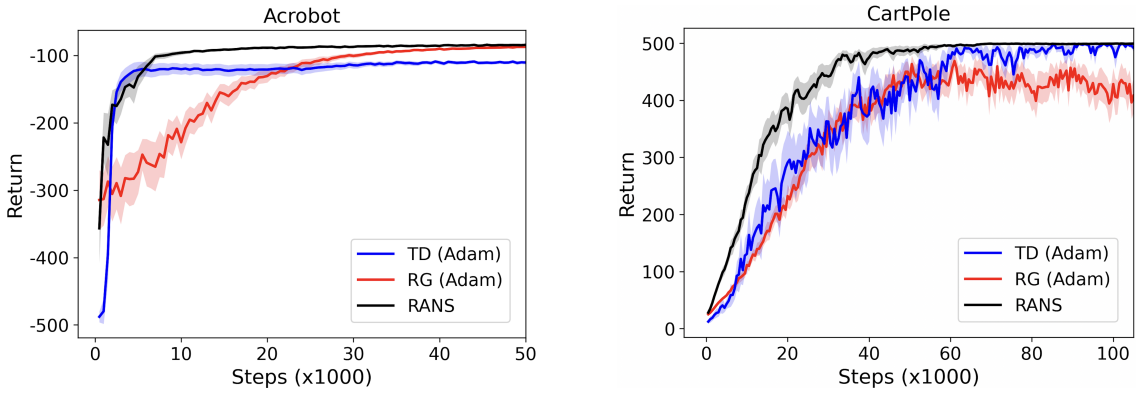

Figure 5. Performance of RANS, TD(0), and RG on classic control tasks. A single-layer neural network with 64 hidden ReLU units was used to learn the Q-values, and a softmax distribution on the Q-values was used as the policy.

in an online manner and are directly fed into the actor and critic training algorithms. We did not use replay buffers or batch updates. Fig. 6 depicts the learning curves of these algorithms. Refer to Appendix G.4 for additional experimental results and details of the experiments. The results indicate that the RANS algorithm surpasses TD and RG in these environments. It is important to note that the results cannot be fairly compared to the state-of-the-art because our setting is on-policy and does not take advantage of replay buffers and batch updates. We leave the integration of these techniques into the RANS ideas for future work.

11. Related works

Poor conditioning of MSBE was previously observed in (Wang & Ueda, 2021) through study of an example Markov chains. More specifically, Wang and Ueda (2021) analyzed a particular n-state Markov chain and showed that the condition-number of MSBE in this Markov chain scales with n^2. They also showed that the condition-number scales with $1/(1-\gamma)^2$ in another example Markov chain. In comparison, our lower bound in Theorem 3.1 (a) holds for every Markov chain, and the lower bound in Theorem 3.1 (b) scales with $n^2/(1-\gamma)^2$.

A prevalent explanation for slowness of gradient-based value estimation methods is the so called information flow in the wrong direction (Baird, 1995). More concretely, each update in RG can be decomposed into a forward bootstrapping component (or a TD update) and a backward bootstrapping component (the so called wrong direction of information flow). A common approach for accelerating the gradient updates is by suppressing the second component (e.g., via some sort of combination with TD updates), especially in early stages of training. The acceleration gained in the residual algorithm (Baird, 1995), TDC (Sutton et al., 2009), TDRC, and QRC (Ghiassian et al., 2020) can be understood from this perspective. In contrast, acceleration gained in our algorithms does not rely on combinations with TD updates.

Use of Gauss-Newton method for value estimation was explicitly proposed in (Gottwald et al., 2021; Gottwald & Shen, 2022), recently. Value estimation algorithms based on Kalman filter (Choi & Van Roy, 2006; Geist & Pietquin, 2010) are also known to have an equivalent form to online Gauss-Newton updates (Geist & Pietquin, 2010). Sun and Bagnell (2015) studied MSBE minimization with Newton method. However, all of the above methods involve approximating a variant of the Hessian or Gauss-Newton matrices and solving a system of linear equations in each iteration, which is computationally costly.

Yao et al. (2009) proposed a low complexity two time-scale method, called LMS-2, for stochastic linear regression. Our RAN algorithm can be perceived as a generalization of the LMS-2 algorithm to MSBE minimization under non-linear function approximation. Several other algorithms including least squares TD (Sutton & Barto, 2018) and (Devraj & Meyn, 2017) also leverage matrix gain for improved convergence, under linear function approximation.

In the same spirit, natural gradient methods (Amari, 1998; Kakade, 2001; Martens, 2020) also enjoy robustness to parameterization. Dabney and Thomas (2014), Knight and Lerner (2018), and Achiam et al. (2019) proposed natural gradients algorithms for value estimation. Dabney and Thomas (2014) also proposed a low complexity two time scale implementation that has high-level algorithmic similarities to the RAN algorithm.

In Section 5 and Appendix C we showed that the RAN algorithm can be perceived as a proximal method. A proximal method for value estimation, called GTD2-MP, was proposed in (Liu et al., 2020; Mahadevan et al., 2014). However, these works consider a Bregman divergence that does not depend on the value estimates. In fact,

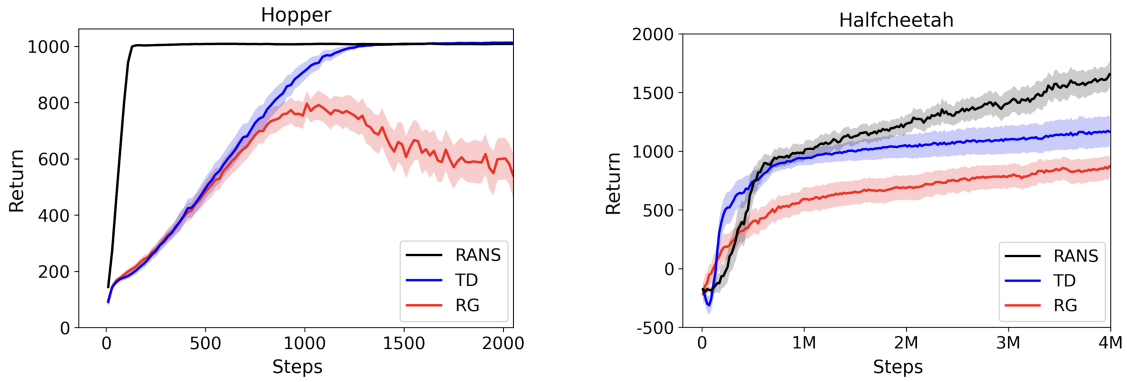

Figure 6. Performance of RANS, TD with target networks, and RG algorithms on simple MuJoCo environments. The Q-values, trained via these algorithms, were integrated into a standard deterministic policy gradient control loop for policy training.

as the step-size goes to zero, update direction of GTD2-MP tends to the expected GTD2 update direction. Schulman et al. (2015), Sun and Bagnell (2015), and Zhu and Murray (2022) considered proximal methods with value dependent penalties of the form $E[(v_{\mathbf{w}_{t+1}}(S_t) - v_{\mathbf{w}_t}(S_t))^2]$. Although the resulting expected update direction $\mathbb{E}_s[\nabla v_{\mathbf{w}}(s)\nabla v_{\mathbf{w}}(s)^T]^{-1}\nabla MSBE(\mathbf{w})$ is robust to parameterization, it is not robust against poor conditioning. For example, in the tabular case, this expected update direction simplifies to $\nabla MSBE(\mathbf{w})$, which is the same as RG. In contrast, in the proximal view of the RAN algorithm, we used penalties of type $E[(\delta_{\mathbf{w}_{t+1}}(S_t, A_t) - \delta_{\mathbf{w}_t}(S_t, A_t))^2]$, which provides robustness to the conditioning of MSBE, as discussed in Section 5 and Appendix C.

The control algorithm SBEED (Dai et al., 2018) involves mirror descent RG along with some other ideas including entropy regularization, akin to SAC (Haarnoja et al., 2018), and blending RG with *naive residual gradient* (Sutton and Barto, 2018, Chapter 11). The entropy regularization technique, in particular, is known to produce significant performance gains. It is noteworthy that the entropy regularization technique can be used in conjunction with RANS and DFS-RAN algorithms, as well.

Karampatziakis and Langford (2010) and Tian and Sutton (2019) proposed a method, called *sliding-step*, to reduce the adverse effect of outliers in certain problems. This method is pretty similar to the outlier-splitting algorithm, with the only difference that in the sliding-step method, all k updates $\mathbf{w} \leftarrow \mathbf{w} - \nabla f_t(\mathbf{w})/k$ are applied sequentially and before time $t+1$, while the outlier-splitting method spreads these updates over a long time. Another simple approach is using momentum to reduce the variance of updates. However, smoothing large outliers requires large momentum parameters, in which case the delayed effect of gradients propagate far into future and become out-dated, pushing $\mathbf{w}$ along outdated outlier gradient even if the outlier gradient at current $\mathbf{w}$ is reversed.

12. Future works and discussion

In this paper, we highlighted causes that underlie slowness of gradient-based value estimation methods, and proposed low complexity techniques to resolve them. Our focus was on the on-policy case, however the proposed algorithms are easily applicable for off-policy learning when combined with standard importance sampling techniques. We provided evidence for the potential of the proposed algorithms via experiments on a few classic environments.

Other than applying standard techniques (such as batch updates, replay buffers, different forms of step-size adaptation, etc.) and testing the algorithms on more complex environments, there are several directions for future research. This includes adopting the unbiased gradient estimate of (9) in (10) instead of the biased estimate in (11), and comparing these methods with other means of solving (9), including conjugate gradient and low rank approximation of the Gauss-Newton matrix. Another important direction is further exploration of the proposed double-sampling-free methods in stochastic environments with neural network function approximation. On the theory side, it would be interesting to study condition-number of MSBE, and in general the shape of MSBE landscape, under linear and non-linear function approximation under common feature representations in asymptotically large environments. Moreover, non-asymptotic behavior and finite sample complexity analysis of the proposed methods would be very helpful for understanding the effectiveness of these algorithms in reducing sensitivity to condition-number.

13. Acknowledgments

The authors want to thank Yi wan, Sina Ghiassian, John N. Tsitsiklis, and Saber Salehkaleybar for their valuable feedback in various stages of development of this work.

References

Achiam, J., Knight, E., and Abbeel, P. Towards characterizing divergence in deep Q-learning. *arXiv preprint arXiv:1903.08894*, 2019.

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

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

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

Bertsekas, D. and Tsitsiklis, J. N. *Neuro-dynamic programming*. Athena Scientific, 1996.

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., and Lee, M. Natural actor-critic algorithms. *Automatica*, 45, 2009.

Boyan, J. A. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49(2):233–246, 2002.

Brandfonbrener, D. and Bruna, J. Geometric insights into the convergence of nonlinear TD learning. *arXiv preprint arXiv:1905.12185*, 2019.

Choi, D. and Van Roy, B. A generalized Kalman filter for fixed point approximation and efficient temporal-difference learning. *Discrete Event Dynamic Systems*, 16(2):207–239, 2006.

Dabney, W. and Thomas, P. Natural temporal difference learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

Dai, B., He, N., Pan, Y., Boots, B., and Song, L. Learning from conditional distributions via dual embeddings. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 1458–1467, 2017.

Deb, R. and Bhatnagar, S. Gradient temporal difference with momentum: Stability and convergence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6488–6496, 2022.

Devraj, A. M. and Meyn, S. Zap Q-learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Du, S., Chen, J., Li, L., Xiao, L., and Zhou, D. Stochastic variance reduction methods for policy evaluation. In *Proceedings of the International Conference on Machine Learning*, pp. 1049-1058, 2017.

Geist, M. and Pietquin, O. Kalman temporal differences. *Journal of artificial intelligence research*, 39:483–532, 2010.

Ghiassian, S. and Sutton, R. S. An empirical comparison of off-policy prediction learning algorithms in the four rooms environment. *arXiv preprint arXiv:2109.05110*, 2021.

Ghiassian, S., Patterson, A., Garg, S., Gupta, D., White, A., and White, M. Gradient temporal-difference learning with regularized corrections. In *Proceedings of the International Conference on Machine Learning*, pp. 3524–3534, 2020.

Gordon, G. J. *Approximate solutions to Markov decision processes*. Ph.D. thesis, Carnegie Mellon University, 1999.

Gottwald, M. and Shen, H. On the compatibility of multistep lookahead and hessian approximation for neural residual gradient. In *Proceedings of the Multi-disciplinary Conference on Reinforcement Learning and Decision Making*, 2022.

Gottwald, M., Gronauer, S., Shen, H., and Diepold, K. Analysis and optimization of Bellman residual errors with neural function approximation. *arXiv preprint arXiv:2106.08774*, 2021.

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

Juditsky, A. and Nemirovski, A. *Optimization for Machine Learning*. MIT Press, 2011.

Kakade, S. M. A natural policy gradient. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

Karampatziakis, N. and Langford, J. Online importance weight aware updates. *arXiv preprint arXiv:1011.1576*, 2010.

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

Knight, E. and Lerner, O. Natural gradient deep Q-learning. *arXiv preprint arXiv:1803.07482*, 2018.

Kochenderfer, M. J. and Wheeler, T. A. *Algorithms for optimization*. MIT Press, 2019.

Konda, V. and Tsitsiklis, J. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, volume 12, 1999.

Kushner, H. and Yin, G. G. *Stochastic approximation and recursive algorithms and applications*. Springer Science & Business Media, 2003.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez,

T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Liu, B., Liu, J., Ghavamzadeh, M., Mahadevan, S., and Petrik, M. Finite-sample analysis of proximal gradient td algorithms. *arXiv preprint arXiv:2006.14364*, 2020.

Macua, S. V., Chen, J., Zazo, S., and Sayed, A. H. Distributed policy evaluation under multiple behavior strategies. *IEEE Transactions on Automatic Control*, 60(5): 1260–1274, 2014.

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

Maei, H. R., Szepesvári, C., Bhatnagar, S., and Sutton, R. S. Toward off-policy learning control with function approximation. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 719–726, 2010.

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

Marquardt, D. W. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

Martens, J. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(1):5776–5851, 2020.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pp. 1928–1937, 2016.

Nesterov, Y. and Polyak, B. T. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

Nocedal, J. and Wright, S. J. *Numerical optimization*. Springer, 1999.

Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Saleh, E. and Jiang, N. Deterministic bellman residual minimization. In *Proceedings of Optimization Foundations for Reinforcement Learning Workshop at NeurIPS*, 2019.

Schoknecht, R. and Merke, A. TD(0) converges provably faster than the residual gradient algorithm. In *Proceedings of International Conference on Machine Learning*, pp. 680–687, 2003.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Sherman, J. and Morrison, W. J. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Annals of Mathematical Statistics*, 21(1): 124–127, 1950.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. *International Conference on Machine Learning*, pp. 387-395, 2014.

Strang, G. *Linear algebra and its applications*. Thomson, Brooks/Cole, Belmont, CA, 2006.

Sun, W. and Bagnell, J. A. Online bellman residual algorithms with predictive error guarantees. 2015.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT Press, 2018.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 993–1000, 2009.

Tian, T. and Sutton, R. S. Extending sliding-step importance weighting from supervised learning to reinforcement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 67–82. Springer, 2019.

Tsitsiklis, J. and Van Roy, B. Analysis of temporal-difference learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 9, 1996.

Van Seijen, H., Van Hasselt, H., Whiteson, S., and Wiering, M. A theoretical and empirical analysis of expected Sarsa. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pp. 177–184, 2009.

Wang, Z. T. and Ueda, M. A convergent and efficient deep Q network algorithm. *arXiv preprint arXiv:2106.15419*, 2021.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3):279–292, 1992.

Yao, H., Bhatnagar, S., and Szepesvári, C. LMS-2: Towards an algorithm that is as cheap as LMS and almost as efficient as RLS. In *Proceedings of the 48h Conference on Decision and Control (CDC)*, pp. 1181–1188. IEEE, 2009.

Zhang, J., He, T., Sra, S., and Jadbabaie, A. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881*, 2019.

Zhang, S., Boehmer, W., and Whiteson, S. Deep residual reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, pp. 1611–1619, 2020.

Zhu, R. J. and Murray, J. M. Gradient descent temporal difference-difference learning. *arXiv preprint arXiv:2209.04624*, 2022.

Appendices

A. Proof of condition-number bounds

In this appendix, we first present the proof of Theorem 3.1 that involves bounds on the condition-number of $MSBE^V(\cdot)$. We then establish similar bounds for $MSBE(\cdot)$ defined in (1).

A.1. Proof of Theorem 3.1

Note that an MDP with a fixed policy boils down to a Markov chain with termination. Consider a Markov chain with termination that has n non-terminal states, and let P be its associated $n \times n$ transition matrix. Note that if transitions from a state can terminate with positive probability, sum over the corresponding row of P will be less than one. Let

$$A \stackrel{\text{def}}{=} (I - \gamma P)^T (I - \gamma P). \tag{18}$$

In the tabular setting and under uniform state distribution, we have $MSBE^V(\mathbf{w}) = \mathbf{w}^T A \mathbf{w}/n$. Therefore, the condition-number $\mathcal{C}$ of $MSBE^V(\cdot)$ is equal to the condition-number of A. Let λ_{max} and λ_{min} denote the largest and smallest eigenvalues of A, respectively. It follows that

$$\mathcal{C} = \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}}. \tag{19}$$

Proof of Part (a). We first propose an upper bound for λ_{min} and then a lower bound for λ_{max}. For states $i = 1, \dots, n$, let l_i be the expected number of steps until termination when we start from state i and follow the Markov chain's transitions. Then, for any state $i = 1, \dots, n$, we have

$$l_i = 1 + \sum_{j=1}^{n} P_{ij} l_j. \tag{20}$$

Let $\boldsymbol{l} \stackrel{\text{def}}{=} [l_1, \dots, l_n]^T$ be the vector representation of $l_1, \dots, l_n$. Then, (20) can be written in the vector form as

$$\boldsymbol{l} = \mathbf{1} + P\boldsymbol{l}. \tag{21}$$

where $\mathbf{1}$ is the vector of all ones. It then follows that

$$(I - \gamma P)\boldsymbol{l} = \boldsymbol{l} - \gamma P \boldsymbol{l} = \boldsymbol{l} - \gamma(\boldsymbol{l} - \mathbf{1}) = (1 - \gamma)\boldsymbol{l} + \gamma \mathbf{1}, \tag{22}$$

where the second equality follows from (21). Let $l \stackrel{\text{def}}{=} (l_1 + \cdots, l_n)/n$ be the mean of $l_1, \dots, l_n$. Cauchy-Schwarz inequality implies that

$$\frac{\|\boldsymbol{l}\|^2}{n} = \frac{1}{n} \sum_{i=1}^{n} l_i^2 \geq \left(\frac{1}{n} \sum_{i=1}^{n} l_i \right)^2 = l^2. \tag{23}$$

For the smallest eigenvalue of A, we have

$$\begin{aligned}
\lambda_{\text{min}} &\leq \frac{\boldsymbol{l}^T A \boldsymbol{l}}{\|\boldsymbol{l}\|^2} \\
&= \frac{\|(I - \gamma P)\boldsymbol{l}\|^2}{\|\boldsymbol{l}\|^2} \\
&= \frac{\|(1 - \gamma)\boldsymbol{l} + \gamma \mathbf{1}\|^2}{\|\boldsymbol{l}\|^2} \\
&= \frac{(I - \gamma)^2 \|\boldsymbol{l}\|^2 + 2\gamma(1 - \gamma)\mathbf{1}^T \boldsymbol{l} + \gamma^2 n}{\|\boldsymbol{l}\|^2} \\
&= (I - \gamma)^2 + \frac{2\gamma(1 - \gamma) n l + \gamma^2 n}{\|\boldsymbol{l}\|^2} \\
&\leq (I - \gamma)^2 + \frac{2\gamma(1 - \gamma) l + \gamma^2}{l^2} \\
&= \left(I - \gamma + \frac{\gamma}{l} \right)^2,
\end{aligned} \tag{24}$$

where the first inequality follows from the definition of the smallest eigenvalue of a symmetric matrix, the first equality is due to the definition of A in (18), the second equality results from (22), the fourth equality is from the definition of l, and the second inequality follows from (23).

Let trace(A) be the trace of A defined as the sum of diagonal entries of A. It is well-known that the trace of any matrix is equal to the sum of eigenvalues of that matrix (Strang, 2006). Therefore, for the largest eigenvalue of A, we have

$$\begin{aligned}
\lambda_{\max} &\geq \frac{1}{n}\text{trace}(A) \\
&= \frac{1}{n}\sum_{i=1}^{n} A_{ii} \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}(I_{ji} - \gamma P_{ji})^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\left((1-\gamma P_{ii})^2 + \sum_{j\neq i}\gamma^2 P_{ji}^2\right) \\
&\geq \frac{1}{n}\sum_{i=1}^{n}(1-\gamma P_{ii})^2 \\
&\geq \left(\frac{1}{n}\sum_{i=1}^{n}(1-\gamma P_{ii})\right)^2 \\
&= \left(1 - \frac{\gamma}{n}\sum_{i=1}^{n} P_{ii}\right)^2 \\
&= (1-\gamma h)^2,
\end{aligned} \tag{25}$$

where the first inequality is because trace(A) equals the sum of eigenvalues of A, the first equality is from the definition of trace, the second equality is due to the definition of A in (18), the third inequality follows from the Cauchy-Schwarz inequality, and the last equality is from the definition $h = \sum_{i=1}^{n} P_{ii}/n$ in the theorem statement. Plugging (24) and (25) into (19), we obtain

$$\mathcal{C} = \frac{\lambda_{\max}}{\lambda_{\min}} \geq \frac{(1-\gamma h)^2}{\lambda_{\min}} \geq \frac{(1-\gamma h)^2}{(1-\gamma+\gamma/l)^2} \geq \frac{(1-\gamma h)^2}{2(1-\gamma)^2 + 2\gamma^2/l^2} \geq \frac{(1-\gamma h)^2}{4}\min\left(\frac{1}{(1-\gamma)^2}, \frac{l^2}{\gamma^2}\right),$$

where the first and second inequalities are due to (25) and (24), respectively. This implies (8) and completes the proof of Part (a) of Theorem 3.1.

Proof of Part (b). Consider an n-state Markov chain with transition matrix

$$P = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix}. \tag{26}$$

In what follows, we derive bounds on the largest and smallest eigenvalues of A defined in (18). Let $\epsilon = \gamma/(n-1)$ and $\mathbf{v} = [-\epsilon, \ldots, -\epsilon, 1]^T$. Then, $P\mathbf{v} = \mathbf{1}$, and as a result,

$$\mathbf{v}^T A \mathbf{v} = \left\|(I-\gamma P)\mathbf{v}\right\|^2 = \left\|\mathbf{v} - \gamma\mathbf{1}\right\|^2 = (n-1)(\gamma+\epsilon)^2 + (1-\gamma)^2 \geq (n-1)(\gamma+\epsilon)^2 = \frac{n^2\gamma^2}{n-1}, \tag{27}$$

where the first equality is from the definition of A in (18), and the last equality is due to the definition of ϵ. On the other hand,

$$\|\mathbf{v}\|^2 = (n-1)\epsilon^2 + 1 = \frac{\gamma^2}{n-1} + 1 = \frac{n-1+\gamma^2}{n-1} \leq \frac{n}{n-1}. \tag{28}$$

It follows that

$$\lambda_{\max} \geq \frac{\mathbf{v}^T A \mathbf{v}}{\|\mathbf{v}\|} \geq \frac{n^2\gamma^2/(n-1)}{n/(n-1)} = n\gamma^2, \tag{29}$$

where the second inequality is due to (27) and (28).

In order to bound the smallest eigenvalue of A, let $\mathbf{x} = [\gamma, \ldots, \gamma, 1]^T$. Therefore $P\mathbf{x} = \mathbf{1}$ and

$$(I - \gamma P)\mathbf{x} = \mathbf{x} - \gamma P\mathbf{x} = \mathbf{x} - \gamma\mathbf{1} = [0, \ldots, 0, 1-\gamma]^T. \tag{30}$$

It follows that

$$\lambda_{\min} \leq \frac{\mathbf{x}A\mathbf{x}}{\|\mathbf{x}\|^2} = \frac{\|(I-\gamma P)\mathbf{x}\|^2}{\|\mathbf{x}\|^2} = \frac{(1-\gamma)^2}{\|\mathbf{x}\|^2} = \frac{(1-\gamma)^2}{(n-1)\gamma^2 + 1} \leq \frac{(1-\gamma)^2}{n\gamma^2} \tag{31}$$

where the first equality is from the definition of A in (18), the second equality is due to (30), and the third equality follows from the definition of $\mathbf{x}$. Plugging (29) and (31) into (19), we obtain

$$\mathcal{C} = \frac{\lambda_{\max}}{\lambda_{\min}} \geq \frac{n\gamma^2}{(1-\gamma)^2/(n\gamma^2)} = \frac{\gamma^4 n^2}{(1-\gamma)^2}.$$

This completes the proof of Part (b) of Theorem 3.1.

A.2. Condition-number of the MSBE defined in terms of action-values

Theorem 3.1 involves bounds on the condition-number of $MSBE^V(\cdot)$ defined in (2). In this appendix, we establish similar bounds for $MSBE(\cdot)$ defined in (1).

Given an MDP and a policy π, we consider an *induced augmented Markov chain* that is a Markov chain whose states are the state-action pairs of the MDP and its transition probabilities are as follows. For any $s, s' \in \mathcal{S}$ and $a, a' \in \mathcal{A}$, the probability of transition from (s, a) to (s', a') in the induced augmented Markov chain is

$$p'_\pi(s', a'|s, a) \stackrel{\text{def}}{=} \int_r p_\pi(s', a', r|s, a)\, dr \tag{32}$$

where p_π is defined in Section 2. We consider a tabular setting, and denote the condition-number of $MSBE_{\mathcal{D}}(\cdot)$ under uniform distribution $\mathcal{D}$ on state-action pairs by $\mathcal{C}'$. We let

$$h' \stackrel{\text{def}}{=} \frac{1}{nm} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} p'_\pi(s, a|s, a) \tag{33}$$

be the self-loop probability in the induced augmented Markov chain, where n is the number of states and $m = |\mathcal{A}|$ is the number of actions. Also let l' be the expected number of steps until termination when starting from a uniformly random state-action pair. The following proposition is the counterpart of Theorem 3.1 for $\mathcal{C}'$.

Proposition A.1. *In the tabular case, the following statements hold for any discount factor $\gamma \in [0, 1]$:*

a) For any MDP and any policy π,

$$\mathcal{C}' \geq \frac{1-\gamma h'}{4} \min\left(\frac{1}{(1-\gamma)^2}, l'^2\right). \tag{34}$$

b) For any $n, m > 0$, there exists an MDP with n states and m actions, and a policy π for which, $\mathcal{C}' \geq \gamma^4(nm)^2/(1-\gamma)^2$.

Proof. We can perceive the dynamics under any given MDP and policy as an induced augmented Markov chain defined in (32). Applying the proof of Theorem 3.1 on this induced augmented Markov chain implies Proposition A.1. □

B. Experiment on condition number under linear function approximation

We ran an experiment to investigate the growth of condition number, $\mathcal{C}$, in an extended Boyan chain under linear function approximation. Fig. 7 shows the dependence of $\mathcal{C}$ on the size of extended Boyan chain, under standard Boyan feature vectors (see Appendix G.1 for details). The number of standard Boyan chain features d in this experiments, satisfies $n = 4d - 3$, where n is the number of states. We observe that the condition number can grow very large under linear function approximation even when $d/n < 1$ (in this case $d/n \simeq 1/4$).

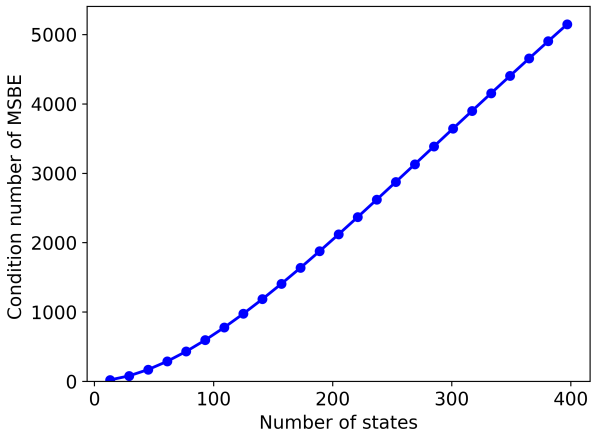

Figure 7. Condition-number of MSBE versus number of states in an extended Boyan chain under linear function approximation with Boyan chain's standard features ($n = 4d - 3$).

C. RAN as a proximal algorithm

In this section, we provide further intuition for the RAN algorithm, by showing that Algorithm 1 can be equivalently derived as a proximal algorithm with momentum for minimizing MSBE. Given an objective function f, a proximal algorithm in its general form aims to find an approximate solution for a proximal operator of the following form in each iteration

$$\mathbf{w}_{t+1} \leftarrow \operatorname*{argmin}_{\mathbf{w}} \Big(f(\mathbf{w}) + D_t(\mathbf{w}, \mathbf{w}_{ref}) \Big), \tag{35}$$

where $\mathbf{w}_{ref}$ is a reference point, usually equal to $\mathbf{w}_t$ or a trace of past $\mathbf{w}$'s, and D_t is a penalty function (also called divergence) that encourages $\mathbf{w}_{t+1}$ to stay close to $\mathbf{w}_{ref}$. In the special case that D_t is a fixed Bregman divergence, (35) boils down to the mirror-descent algorithm (Juditsky & Nemirovski, 2011). However, in general, D_t can be a time varying function and can depend on the local shape of the objective f.

Back to the value estimation problem, for any consecutive state-action pairs (s, a) and (s', a'), and any pair $\mathbf{w}$ and $\mathbf{w}'$ of weights, we let

$$\delta_{\mathbf{w}}(s, a, s', a') \stackrel{\text{def}}{=} \gamma q_{\mathbf{w}}(s', a') - q_{\mathbf{w}}(s, a), \tag{36}$$

and

$$\Delta\delta_{\mathbf{w},\mathbf{w}'}(s, a, s', a') \stackrel{\text{def}}{=} \delta_{\mathbf{w}}(s, a, s', a') - \delta_{\mathbf{w}'}(s, a, s', a'). \tag{37}$$

Consider a divergence measure of the form $D_t(\mathbf{w}, \mathbf{w}_t) = c\,\mathbb{E}_{s,a}\Big[\mathbb{E}_{s',a'|s,a}\left[\Delta\delta_{\mathbf{w},\mathbf{w}_t}(s, a, s', a')^2\right]\Big]$, for some constant $c > 0$. Then, the proximal operator in (35) turns into

$$\operatorname*{argmin}_{\mathbf{w}} MSBE(\mathbf{w}) + c\,\mathbb{E}_{s,a}\Big[\mathbb{E}_{s',a'|s,a}\left[\Delta\delta_{\mathbf{w},\mathbf{w}_t}(s, a, s', a')^2\right]\Big]. \tag{38}$$

To obtain a low complexity incremental version of the above proximal updates, we consider doing sample updates along the gradient of (38) at time t. For this purpose[3], we work with $\mathbf{w}_{ref} = \mathbf{w}_{t-1}$ instead of $\mathbf{w}_{ref} = \mathbf{w}_t$, i.e. we consider the proximal objective $MSBE(\mathbf{w}) + c\,\mathbb{E}_{s,a}\Big[\mathbb{E}_{s',a'|s,a}\left[\Delta\delta_{\mathbf{w},\mathbf{w}_{t-1}}(s, a, s', a')^2\right]$ and its unbiased sample gradient

$$\begin{aligned} \mathbf{g}_t &\stackrel{\text{def}}{=} \big(\delta'_t - c\,\Delta\delta_{\mathbf{w}_t,\mathbf{w}_{t-1}}(S_t, A_t, S_{t+1}, A_{t+1})\big)\nabla\delta_t \\ &\simeq \big(\delta'_t - c\,(\mathbf{w}_t - \mathbf{w}_{t-1})^T\nabla\delta_t\big)\nabla\delta_t, \end{aligned} \tag{39}$$

[3]Here we avoid using $\mathbf{w}_{ref} = \mathbf{w}_t$ because in this case, for any (s, a, s', a'), $\Delta\delta_{\mathbf{w}_t,\mathbf{w}_t}(s, a, s', a') = 0$. This implies that $\nabla_{\mathbf{w}}\Delta\delta_{\mathbf{w},\mathbf{w}_t}(s, a, s', a')^2\big|_{\mathbf{w}=\mathbf{w}_t} = 0$. As such, the penalty would not affect gradient of the proximal objective at $\mathbf{w} = \mathbf{w}_t$.

where the approximate equality is because $(\mathbf{w}_t - \mathbf{w}_{t-1})^T \nabla \delta_t$ is a first order approximation of $\Delta \delta_{\mathbf{w}_t, \mathbf{w}_{t-1}}$. Let $\hat{\mathbf{g}}_t = \big(\delta'_t - c(\mathbf{w}_t - \mathbf{w}_{t-1})^T \nabla \delta_t\big) \nabla \delta_t$ and consider the approximate gradient update $\mathbf{w} \leftarrow \mathbf{w} - \beta \hat{\mathbf{g}}_t$. We further employ a momentum to reduce the variance of these updates. The resulting momentum based algorithm is of the form

$$\begin{aligned} \mathbf{m}_t &= \lambda \mathbf{m}_{t-1} + \beta \hat{\mathbf{g}}_t, \\ \mathbf{w}_{t+1} &= \mathbf{w}_t - \alpha \mathbf{m}_t. \end{aligned} \tag{40}$$

Let $\eta = 1/\alpha$. Since $\mathbf{w}_t - \mathbf{w}_{t-1} = \alpha \mathbf{m}_{t-1}$, $\hat{\mathbf{g}}_t$ simplifies to

$$\hat{\mathbf{g}}_t = \big(\delta'_t - \mathbf{m}_{t-1}^T \nabla \delta_t\big) \nabla \delta_t.$$

Plugging this into (40), we obtain updates that are identical to (11). This establishes our earlier claim that the RAN algorithm can be equivalently formulated as a proximal algorithm with momentum for minimizing MSBE.

As another intuitive perspective for Algorithm 1, we can perceive $\mathbf{m}$ as a momentum of MSBE gradients, to which we add a correction term equal to a sample gradient of the penalty $\mathbb{E}_{s,a}\big[\mathbb{E}_{s',a'|s,a}\big[\Delta \delta_{\mathbf{w},\mathbf{w}_t}(s,a,s',a')^2\big]\big]$, in each iteration. This momentum spreads the effect of each MSBE gradient over an $O(1/(1-\lambda))$-long horizon, which provides enough time for the correction updates to trim the direction of those MSBE gradients. The correction terms are small along directions that $\nabla \delta$ is small, allowing MSBE gradients to accumulate along those directions. Conversely, the correction terms are large along directions that $\nabla \delta$ is large, preventing $\mathbf{m}$ from growing large along those directions. This leads to accelerated convergence along the directions where $\nabla \delta$ is small, while preventing instability along directions where $\nabla \delta$ is large.

D. Convergence of the RAN algorithm

In this appendix, we study conditions for convergence of Algorithm (11) under different choices of λ. Convergence proofs of two-time-scale approaches are well-studied, and generally involve tedious yet pretty standard statistical arguments. As such, similar to several other papers (Konda & Tsitsiklis, 1999; Dabney & Thomas, 2014), here we keep our arguments in a high level, and only discuss the steps of the proof without going into the proof details. Throughout this Appendix, we consider irreducible and aperiodic Markov chains with finite number of states. We assume that the function approximation $Q_{\mathbf{w}}(s,a)$ is a differentiable function of $\mathbf{w}$ with bounded and Lipschitz constant derivatives. By boundedness of derivatives we mean that there exists a $C > 0$ such that for consecutive state-action pairs (s,a,s',a') and any $\mathbf{w}$,

$$\big\| \nabla \delta_{\mathbf{w}}(s,a,s',a') \big\| < C, \tag{41}$$

where $\delta_{\mathbf{w}}(s,a,s',a')$ is defined in (36). Given a distribution $\mathcal{D}$ over state and action pairs, for any $\mathbf{w} \in \mathbb{R}^d$, we let

$$\hat{G}_{\mathbf{w}} \stackrel{\text{def}}{=} \mathbb{E}_{s,a \sim \mathcal{D}}\Big[\mathbb{E}_{s',a'|s,a}\left[\nabla \delta_{\mathbf{w}}(s,a,s',a')\, \nabla \delta_{\mathbf{w}}(s,a,s',a')^T\right]\Big]. \tag{42}$$

We study convergence in three different regimes on λ, namely $\lambda = 1$, $\lambda = 1 - c\beta_t$ for some constant $c > 0$, and for a constant $\lambda < 1$.

Case 1: ($\lambda = 1$). We assume that α_t and β_t are decreasing positive sequneces, satisfying

$$\sum_{t=0}^{\infty} \alpha_t = \sum_{t=0}^{\infty} \beta_t = \infty, \qquad \sum_{t=0}^{\infty} \alpha_t^2 < \infty \sum_{t=0}^{\infty} \beta_t^2 < \infty, \qquad \frac{\alpha_t}{\beta_t} \to 0. \tag{43}$$

We further assume that $\hat{G}_{\mathbf{w}}$ defined in (42) is uniformly positive definite, in the sense that there is an $\epsilon > 0$ such that for any $\mathbf{w}$ and any $\mathbf{x} \in \mathbb{R}^d$, we have $\mathbf{x}^T \hat{G}_{\mathbf{w}} \mathbf{x} \geq \epsilon \|\mathbf{x}\|^2$. In this case, for fixed $\mathbf{w}$, the updates in (11) converge to $\hat{G}_{\mathbf{w}}^{-1} \nabla MSBE_{\mathcal{D}}(\mathbf{w})$. Since $\alpha_t/\beta_t \to 0$, $\mathbf{w}$ is updated much slower than $\mathbf{m}$. As such, $\mathbf{m}$ is updated with much larger step-sizes, perceiving $\mathbf{w}$ as almost stationary, and therefore $\mathbf{m}$ converges to an asymptotically small neighborhood of $\hat{G}_{\mathbf{w}}^{-1} \nabla MSBE_{\mathcal{D}}(\mathbf{w})$. Since $\hat{G}_{\mathbf{w}}$ is uniformly positive definite, this $\mathbf{m}$ is an absolutely decreasing direction for $MSBE_{\mathcal{D}}(\mathbf{w})$. Then, standard proof techniques for stochastic approximation algorithms can be used to establish convergence of $\mathbf{w}$ to a stationary point of MSBE.

Case 2: ($\lambda = 1 - c\beta_t$, for some constant $c > 0$). In this case, the update rule (11) boils down to:

$$\begin{aligned} \mathbf{m} &\leftarrow \lambda \mathbf{m} + \beta_t (\delta'_t - \mathbf{m}^T \nabla \delta_t) \nabla \delta_t \\ &= (1 - c\beta_t)\mathbf{m} + \beta_t \big(\delta'_t - \mathbf{m}^T \nabla \delta_t\big) \nabla \delta_t \\ &= \mathbf{m} - \beta_t \big((cI + \nabla \delta_t \nabla \delta_t^T)\mathbf{m} - \delta'_t \nabla \delta_t\big) \\ &= \mathbf{m} - \beta_t \nabla_{\mathbf{m}} \left(\frac{1}{2} \mathbf{m}^T \big(cI + \nabla_{\mathbf{w}} \delta_t \nabla_{\mathbf{w}} \delta_t^T\big)\mathbf{m} - \delta'_t \nabla_{\mathbf{w}} \delta_t^T \mathbf{m}\right), \end{aligned} \tag{44}$$

where I is the identity matrix. Therefore, in each iteration, $\mathbf{m}$ is updated along a sample gradient of the loss function $\hat{L}(\mathbf{m}) = \mathbf{m}^T\big(cI+\hat{G}_{\mathbf{w}}\big)\mathbf{m}-\nabla_{\mathbf{w}}MSBE(\mathbf{w})^T\mathbf{m}$, where $\hat{G}_{\mathbf{w}}$ is defined in (42). Thus, assuming (43), $\mathbf{m}$ will asymptotically converge to the minimizer $\big(cI+\hat{G}_{\mathbf{w}}\big)^{-1}\nabla MSBE(\mathbf{w})$ of $\hat{L}$. Since $cI+\hat{G}_{\mathbf{w}}$ is uniformly positive definite, this is an absolutely descent direction for $MSBE(\mathbf{w})$. Then, standard proof techniques for stochastic approximation algorithms can be used to establish convergence of $\mathbf{w}$ to a stationary point of MSBE.

Case 3: (Constant $\lambda < 1$). As opposed to the Cases 1 and 2, here we do not need two-time-scale step-sizes. More specifically, we assume that $\alpha_t > 0$ is constant and β_t is a decreasing positive sequence satisfying $\sum_{t=0}^{\infty}\beta_t = \infty$ and $\sum_{t=0}^{\infty}\beta_t^2 < \infty$. We also assume that $\mathbf{w}$ remains bounded which can be enforced either by projection of $\mathbf{w}$ on a compact set or by adequate normalization of α_t (see (Konda & Tsitsiklis, 1999) and (Kushner & Yin, 2003)). It follows from (41) that for any time t,

$$|\delta_t| = |\delta_{\mathbf{w}_t}(S_t, A_t, S_{t+1}, A_{t+1})| \leq C' + C\|\mathbf{w}_t\|. \tag{45}$$

for some constant C'. Then, from (11) we obtain

$$\begin{aligned}
\|\mathbf{m}_t\| &= \big\|\lambda\mathbf{m}_{t-1} + \beta_t\big(\delta'_t - \mathbf{m}_{t-1}^T\nabla\delta_t\big)\nabla\delta_t\big\| \\
&\leq \lambda\|\mathbf{m}_{t-1}\| + \beta_t\big(|\delta'_t| + \|\mathbf{m}_{t-1}\|\cdot\|\nabla\delta_t\|\big)\|\nabla\delta_t\| \\
&\leq \lambda\|\mathbf{m}_{t-1}\| + \beta_t\big(|\delta'_t| + C\|\mathbf{m}_{t-1}\|\big)C \\
&\leq \lambda\|\mathbf{m}_{t-1}\| + \beta_t C\big(C' + C\|\mathbf{w}_t\| + C\|\mathbf{m}_{t-1}\|\big) \\
&= (\lambda + \beta_t C^2)\|\mathbf{m}_{t-1}\| + \beta_t(C'C + C^2\|\mathbf{w}_t\|),
\end{aligned} \tag{46}$$

where the second and third inequalities follow from (41) nad (45), respectively. When β_t is small enough such that $\lambda + \beta_t C^2 < 1$, it follows from the boundedness assumption of $\mathbf{w}$ that $\|\mathbf{m}_t\| = O(\beta_t/(1-\lambda))$. Therefore, (11) can be expressed as

$$\begin{aligned}
\mathbf{m}_t &= \lambda\mathbf{m}_{t-1} + \beta_t\delta'_t\nabla\delta_t - \beta_t(\mathbf{m}_{t-1}^T\nabla\delta_t)\nabla\delta_t \\
&= \lambda\mathbf{m}_{t-1} + \beta_t\delta'_t\nabla\delta_t + O\big(\beta_t^2/(1-\lambda)\big) \\
&= \sum_{\tau=0}^{t}\big(\lambda^\tau\beta_{t-\tau}\delta'_{t-\tau}\nabla\delta_{t-\tau}\big) + O\big(\beta_t^2/(1-\lambda)^2\big).
\end{aligned}$$

Therefore, $\mathbf{m}$ is essentially a momentum of sample gradient of MSBE. Then, convergence of $\mathbf{w}$ to a stationary point of MSBE follows from standard techniques for analysing SGD algorithms with momentum.

E. Asymptotic robustness of RAN to parameterization

In this appendix, we present a formal version of Proposition 5.1 and its proof. Here, we assume that $\lambda = 1$ and consider an asymptotically small step-sizes regime with $\alpha \to 0$ and $\alpha/\beta \to 0$. As discussed in Section 5 and Appendix D, when $\alpha/\beta \to 0$, $\mathbf{m}$ in the RAN algorithm converges to the approximate Gauss-Newton direction

$$\mathbf{m}_{GN}(\mathbf{w}, q) = \hat{G}_{\mathbf{w},q}^{-1}\, g_{\mathbf{w},q}, \tag{47}$$

where

$$\hat{G}_{\mathbf{w},q} = \mathbb{E}_{s,a\sim\mathcal{D}}\Big[\mathbb{E}_{s',a'|s,a}\Big[\big(\gamma\nabla_{\mathbf{w}}q_{\mathbf{w}}(s',a') - \nabla_{\mathbf{w}}q_{\mathbf{w}}(s,a)\big)\big(\gamma\nabla_{\mathbf{w}}q_{\mathbf{w}}(s',a') - \nabla_{\mathbf{w}}q_{\mathbf{w}}(s,a)\big)^T\Big]\Big], \tag{48}$$

$$g_{\mathbf{w},q} = \mathbb{E}_{s,a\sim\mathcal{D}}\Big[\mathbb{E}_{s',a',r|s,a}\left[r + \gamma q_{\mathbf{w}}(s',a') - q_{\mathbf{w}}(s,a)\right]\mathbb{E}_{s',a'|s,a}\big[\gamma\nabla_{\mathbf{w}}q_{\mathbf{w}}(s',a') - \nabla_{\mathbf{w}}q_{\mathbf{w}}(s,a)\big]\Big]. \tag{49}$$

where $\mathcal{D}$ is a distribution over state and action pairs. In this case, when $\alpha \to 0$ the trajectory of $\mathbf{w}$ approaches the trajectory of the following Ordinary Differential Equations (ODE):

$$\dot{\mathbf{w}} = -\mathbf{m}_{GN}(\mathbf{w}, q), \tag{50}$$

where $\dot{\mathbf{w}}$ is the derivative of $\mathbf{w}$ with respect to time. We refer to (50) as the ODE formulation of two time-scale RAN for the q function.

Let $u : \mathbb{R}^d \to \mathbb{R}^d$ be a (possibly non-linear) bijective and differentiable mapping. Suppose that the Jacobian of $u(\cdot)$ is invertible everywhere. We define $\tilde{q}$ as a reparameterization of the q function by u. More specifically, for any state-action pair (s, a) and any $\mathbf{v} \in \mathbb{R}^d$, we let

$$\tilde{q}_{\mathbf{v}}(s, a) = q_{u(\mathbf{v})}(s, a). \tag{51}$$

Consider the ODE formulation of two time-scale RAN for the $\tilde{q}$ function:

$$\dot{\mathbf{v}} = -\mathbf{m}_{GN}(\mathbf{v}, \tilde{q}), \tag{52}$$

for $\mathbf{m}_{GN}(\mathbf{w}, \tilde{q})$ defined in (47).

The following proposition is a formal version of Proposition 5.1. It draws a connection between solution trajectories of (50) and (52).

Proposition 6.1 (Formal)**.** *Let $u : \mathbb{R}^d \to \mathbb{R}^d$ be a (possibly non-linear) bijective and differentiable mapping with invertible Jacobian, and consider the corresponding reparameterization $\tilde{q}$ of the q function as in (51). Let $\mathbf{w}_t$ and $\mathbf{v}_t$ for $t \geq 0$ be solution trajectories of the ODE formulations of two time-scale RAN in (50) and (52), respectively. If $\mathbf{w}_0 = u(\mathbf{v}_0)$, then $\mathbf{w}_t = u(\mathbf{v}_t)$, for all $t \geq 0$. Moreover, $q_{\mathbf{w}_t}(s, a) = \tilde{q}_{\mathbf{v}_t}(s, a)$, for all times $t \geq 0$ and all state-action pairs (s, a).*

The proposition suggests that the two-time scale RAN algorithm with asymptotically small step-sizes is invariant to any non-linear bijective reparameterization of the q function.

Proof of Proposition 6.1. For any $\mathbf{v} \in \mathbb{R}^d$, let $J_{\mathbf{v}}$ be the Jacobian matrix of $u(\mathbf{v})$. Then, for any state-action pair (s, a) and any $\mathbf{v} \in \mathbb{R}^d$,

$$\nabla_{\mathbf{v}} \tilde{q}_{\mathbf{v}}(s, a) = \nabla_{\mathbf{v}} q_{u(\mathbf{v})}(s, a) = J_{\mathbf{v}}^T \, \nabla_{\mathbf{w}} q_{\mathbf{w}}(s, a)\big|_{\mathbf{w}=u(\mathbf{v})}, \tag{53}$$

where the first equality is from the definition of $\tilde{q}$ in (51), and the second equality follows from the chain rule for differentiation. Then,

$$\begin{aligned}
\hat{G}_{\mathbf{v},\tilde{q}} &= \mathbb{E}_{s,a\sim\mathcal{D}}\Big[\mathbb{E}_{s',a'|s,a}\left[\left(\gamma\nabla_{\mathbf{v}}\tilde{q}_{\mathbf{v}}(s',a') - \nabla_{\mathbf{v}}\tilde{q}_{\mathbf{v}}(s,a)\right)\left(\gamma\nabla_{\mathbf{v}}\tilde{q}_{\mathbf{v}}(s',a') - \nabla_{\mathbf{v}}\tilde{q}_{\mathbf{v}}(s,a)\right)^T\right]\Big] \\
&= \mathbb{E}_{s,a\sim\mathcal{D}}\Big[\mathbb{E}_{s',a'|s,a}\left[J_{\mathbf{v}}^T\left(\gamma\nabla_{\mathbf{w}}q_{\mathbf{w}}(s',a') - \nabla_{\mathbf{w}}q_{\mathbf{w}}(s,a)\right)\left(\gamma\nabla_{\mathbf{w}}q_{\mathbf{w}}(s',a') - \nabla_{\mathbf{w}}q_{\mathbf{w}}(s,a)\right)^T J_{\mathbf{v}}\,\big|_{\mathbf{w}=u(\mathbf{v})}\right]\Big] \\
&= J_{\mathbf{v}}^T \hat{G}_{\mathbf{w},q}\, J_{\mathbf{v}}\,\big|_{\mathbf{w}=u(\mathbf{v})},
\end{aligned} \tag{54}$$

where the first and the last equalities are due to (48) second equality follows from (53) In the same vein,

$$\begin{aligned}
g_{\mathbf{v},\tilde{q}} &= \mathbb{E}_{s,a\sim\mathcal{D}}\Big[\mathbb{E}_{s',a',r|s,a}\left[r + \gamma\tilde{q}_{\mathbf{v}}(s',a') - \tilde{q}_{\mathbf{v}}(s,a)\right]\mathbb{E}_{s',a'|s,a}\big[\gamma\nabla_{\mathbf{v}}\tilde{q}_{\mathbf{v}}(s',a') - \nabla_{\mathbf{v}}\tilde{q}_{\mathbf{v}}(s,a)\big]\Big] \\
&= \mathbb{E}_{s,a\sim\mathcal{D}}\Big[\mathbb{E}_{s',a',r|s,a}\left[r + \gamma\tilde{q}_{\mathbf{v}}(s',a') - \tilde{q}_{\mathbf{v}}(s,a)\right]\mathbb{E}_{s',a'|s,a}\big[\gamma J_{\mathbf{v}}^T\nabla_{\mathbf{w}}q_{\mathbf{w}}(s',a') - J_{\mathbf{v}}^T\nabla_{\mathbf{w}}q_{\mathbf{w}}(s,a)\big]\,\big|_{\mathbf{w}=u(\mathbf{v})}\Big] \\
&= \mathbb{E}_{s,a\sim\mathcal{D}}\Big[\mathbb{E}_{s',a',r|s,a}\left[r + \gamma q_{\mathbf{w}}(s',a') - q_{\mathbf{w}}(s,a)\right]\mathbb{E}_{s',a'|s,a}\big[\gamma J_{\mathbf{v}}^T\nabla_{\mathbf{w}}q_{\mathbf{w}}(s',a') - J_{\mathbf{v}}^T\nabla_{\mathbf{w}}q_{\mathbf{w}}(s,a)\big]\,\big|_{\mathbf{w}=u(\mathbf{v})}\Big] \\
&= J_{\mathbf{v}}^T g_{\mathbf{w},q}\,\big|_{\mathbf{w}=u(\mathbf{v})},
\end{aligned} \tag{55}$$

where the first and the last equalities are due to (49), the second equality follows from (53), and the third equality is from the definition of $\tilde{q}$ in (51). Plugging (54) and (55) into (50) and (52), we obtain

$$\begin{aligned}
\dot{\mathbf{v}} &= -\mathbf{m}_{GN}(\mathbf{v}, \tilde{q}) \\
&= -\hat{G}_{\mathbf{v},\tilde{q}}^{-1}\, g_{\mathbf{v},\tilde{q}} \\
&= -\left(J_{\mathbf{v}}^T\,\hat{G}_{\mathbf{w},q}\,J\right)^{-1}\, g_{\mathbf{v},\tilde{q}}\,\big|_{\mathbf{w}=u(\mathbf{v})} \\
&= -J_{\mathbf{v}}^{-1}\,\hat{G}_{\mathbf{w},q}^{-1} J_{\mathbf{v}}^{-T}\, g_{\mathbf{v},\tilde{q}}\,\big|_{\mathbf{w}=u(\mathbf{v})} \\
&= -J_{\mathbf{v}}^{-1}\,\hat{G}_{\mathbf{w},q}^{-1}\, g_{\mathbf{w},q}\,\big|_{\mathbf{w}=u(\mathbf{v})} \\
&= -J_{\mathbf{v}}^{-1}\,\mathbf{m}_{GN}(\mathbf{w}, q)\,\big|_{\mathbf{w}=u(\mathbf{v})} \\
&= J_{\mathbf{v}}^{-1}\,\dot{\mathbf{w}}\,\big|_{\mathbf{w}=u(\mathbf{v})},
\end{aligned} \tag{56}$$

Algorithm 4 RANS

Hyper parameters: stepsize α, $\eta \in (0,1)$ outlier threshold $\rho > 1$, momentum and trace parameters $\lambda, \lambda' \in [0,1)$, outlier sampling probability σ. Good default values for η, ρ, λ, λ', and p based on our experiments are $\eta = 0.2$, $\rho = 1.2$, $\lambda = 0.999$, $\lambda' = 0.9999$, and $\sigma = 0.02$.
Initialize: $\mathbf{m} = \mathbf{0}$, $\hat{\boldsymbol{\nu}} = \mathbf{0}$, $\hat{\xi} = 0$, and $\mathbf{w}$.
for $t = 1, 2, \ldots$ **do**
  Take two independent samples (S_{t+1}, A_{t+1}) and (S'_{t+1}, A'_{t+1}), and consider δ_t and δ'_t as in (3) and (4).
  $\hat{\boldsymbol{\nu}}_t \leftarrow \lambda' \hat{\boldsymbol{\nu}}_{t-1} + (1-\lambda')(\nabla \delta_t)^2$ ▷ $(\nabla\delta_t)^2$ is the entrywise square vector of $\nabla\delta_t$
  $\boldsymbol{\nu}_t = \hat{\boldsymbol{\nu}}_t / (1-\lambda'^t)$ ▷ bias-corrected trace of $(\nabla\delta_t)^2$
  $\xi_t = \langle (1/\sqrt{\boldsymbol{\nu}_t}) \odot \nabla\delta_t, \nabla\delta_t \rangle$ ▷ Outlier measure
  $\hat{\xi} \leftarrow \lambda' \hat{\xi} + (1-\lambda')\xi_t$
  $\bar{\xi} = \hat{\xi}/(1-\lambda'^t_\xi)$ ▷ bias-corrected trace of ξ
  $k = \lfloor \xi_t / (\rho \bar{\xi}) \rfloor + 1$ ▷ outlier-splitting factor
  $\boldsymbol{\beta} = \eta / (\rho \bar{\xi}_t \sqrt{\boldsymbol{\nu}_t})$
  $\mathbf{m} \leftarrow \lambda \mathbf{m} + \big(\delta'_t - \mathbf{m}^T \nabla\delta_t\big)\boldsymbol{\beta} \odot \nabla\delta_t / k$
  $\mathbf{w} \leftarrow \mathbf{w} - \alpha \mathbf{m}$
  if $k > 1$ **then**
    Store $(S_t, A_t, S_{t+1}, A_{t+1}, r_t, , S'_{t+1}, A'_{t+1}, r'_t, k, k-1)$ in the outlier buffer ▷ the last entry in the tuple indicates the remaining number of future updates based on this sample
  end if
  With probability $\min(1, \sigma *$ length of outlier bufffer$)$**:** ▷ do an update using outlier buffer samples
    Sample $(S_\tau, A_\tau, S_{\tau+1}, A_{\tau+1}, r_\tau, S'_{\tau+1}, A'_{\tau+1}, r'_\tau, k', j)$ uniformly form the outlier buffer
    Let δ_t and δ'_t be as in (3) and (4), respectively
    $\xi = \langle (1/\sqrt{\boldsymbol{\nu}_t}) \odot \nabla\delta_\tau, \nabla\delta_\tau \rangle$
    $k'' = \max\big(k', \lfloor \xi / (\rho\bar{\xi}) \rfloor + 1\big)$
    $\mathbf{m} \leftarrow \lambda \mathbf{m} + \big(\delta'_\tau - \mathbf{m}^T \nabla\delta'_\tau\big)\boldsymbol{\beta} \odot \nabla\delta_\tau / k''$
    $\mathbf{w} \leftarrow \mathbf{w} - \alpha \mathbf{m}$
    if $j > 1$ **then**
      Replace $(S_\tau, \ldots, k', j)$ with $(S_\tau, \ldots, k', j-1)$ in the outlier buffer
    else if $j = 1$ **then**
      Remove $(S_\tau, \ldots, k', j)$ from the outlier buffer
    end if
end for

where the equalities are respectively due to (52), (47), (54), the assumption that the Jacobian J of u is invertible, (55), (47), and (50). Therefore, if at time t, $\mathbf{w}_t = u(\mathbf{v}_t)$, then $\dot{\mathbf{w}}_t = J_{\mathbf{v}} \dot{\mathbf{v}}_t = \frac{d}{dt} u(\mathbf{v}_t)$. This together with the assumption $\mathbf{w}_0 = u(\mathbf{v}_0)$ implies that $\mathbf{w}_t = u\big(\mathbf{v}_t\big)$, for all $t \geq 0$. It then follows from the definition of $\tilde{q}$ in (51) that $q_{\mathbf{w}_t}(s,a) = \tilde{q}_{\mathbf{v}_t}(s,a)$, for all times $t \geq 0$ and all state-action pairs (s,a). This completes the proof of Proposition 1. □

F. Pseudo code of RANS

The pseudo code of the RANS algorithm is given in Algorithm 4. Note that in the two time scale regime, where $\alpha, \eta \to 0$ and $\alpha/\eta \to 0$, outlier-splitting would have no effect on the expected direction of $\mathbf{m}$ updates. In this case, convergence of the RANS algorithm follows from similar arguments to Appendix D.

G. Details of experiments

G.1. Experiments of Section 3 and Appendix B

An n-state extended Boyan chain with termination is a Markov chain with termination with states $0, 1, \ldots, n-1$ and transition probabilities: $P(1 \to 0) = 1$ and $P(i \to i-1) = P(i \to i-2) = 0.5$ for $i = 2, 3, \ldots, n-1$. Furthermore, state 0 goes to a terminal state with probability 1. By standard features, we mean feature representations similar to (Boyan,

2002). More specifically, given a $d > 1$ and $n = 4d - 3$, we consider d standard features for the n-state extended Boyan-chain as follows. In the ith standard feature vector for $i = 0, 1, \ldots, d-1$, the jth entry is equal to $1 - |j - 4i|/4$ for $j = \max(0, 4i - 3), \ldots, \min(d - 1, 4i + 3)$, and all other entries equal zero. For the special case of $n = 13$ and $d = 4$, the standard features would be the same as the features considered in (Boyan, 2002).

By random binary features we mean an $n \times d$ feature matrix Φ with i.i.d. entries that take 0 and 1 values with equal probability. For evaluating value-errors in Fig. 2 we consider reward 1 for the transition at state 0 (that leads to the terminal state), and reward 0 for all other transitions. Note that the reward function does not affect condition-number. Each point in this first experiment is the median of 100 independent runs with different random feature matrices. We use median instead of mean to eliminate the adverse effect of unbounded values in degenerate cases (e.g., infinite condition-number in the case of low rank feature matrices).

For both experiments in Fig. 2 and 7 we use discount factor $\gamma = 0.995$. Condition-numbers and value-errors in these experiments are computed with respect to uniform state distribution.

G.2. Experiment of Section 5

We considered an extension of the Hallway environment with 50 states and one action, in which each state $s = 1, 2 \ldots, 50$ transits to state $\min(s + 1, 50)$ with probability 0.99, and transits to a terminal state with probability 0.01. The experiment was tabular with $\gamma = 1$. All rewards were set equal to zero, in which case the correct q-values are $q_\pi(s, \cdot) = 0$, for all sates s. All algorithms were initialized with $q(s, \cdot) = 1$, for $s = 1, \ldots, 50$. Fig. 3 illustrates learning curves of value-error $\sum_{s=1}^{50} \big(q(s, 1) - q_\pi(s, 1)\big)^2/50$ for RAN (Algorithm 1), RG, and TD(0) algorithms. Each point is an average over 100 independent runs. We optimized the parameters for RG and Algorithm 1. The parameters used in the experiments are as follows. For RAN, we set $\alpha = 0.025$, $\beta = 0.4$, and $\lambda = 0.9998$. For RG and TD(0) we used $\alpha = 0.5$.

G.3. Experiment of Section 6

We ran an experiment on Baird's Star environment (Baird, 1995). We performed off-policy learning with uniform state distribution. All rewards were set to zero, in which case the correct q-values are $q_\pi(s, \cdot) = 0$, for all sates s. We used the initial point $\mathbf{w} = [2, 1, 1, 1, 1, 1, 1]$ for all algorithms. Fig. 4 illustrates learning curves of value-error $\sum_{s=1}^{6} \big(q_{\mathbf{w}}(s, 1) - q_\pi(s, 1)\big)^2/6$ for the RAN (Algorithm 1), DSF-RAN (Algorithm 2), RG, GTD2, and TD(0) algorithms. Each point is an average over 10 independent runs. TD(0) was unstable on this environment, and we chose a very small step-size $\alpha = 10^{-5}$ for TD(0). For all other algorithms, we used optimized parameters as follows. For RAN, we set $\alpha = 2$, $\beta = 0.15$, and $\lambda = 0.995$. For DSF-RAN, we set $\alpha = 1$, $\beta = 0.15$, $\eta = 0.3$, and $\lambda = 0.995$. For RG we used $\alpha = 0.3$. For GTD2 we set $\alpha = 0.15$ and $\beta = 0.3$.

G.4. Experiment of Section 10

We ran an experiment on classic control tasks –Acrobot and Cartpole– to test the performance of the RANS algorithm. We used a single-layer neural network with 64 hidden ReLU units to learn the action-values, while choosing actions according to a softmax distribution on the action-values, $a \sim \text{Softmax}\big(q_{\mathbf{w}}(s, \cdot)\big)$. The network for $q_{\mathbf{w}}$ was trained with three algorithms: TD(0), RG, and RANS (Algorithm 4). Since RANS incorporates adaptive step-sizes, for fair comparison we trained TD(0) and RG using Adam optimizer.

For each algorithm, we performed training for 100 randomly generated random seeds. For each training seed, in order to obtain an estimate of expected returns, we took an average over 400 independent environment simulations once every 500 steps. In Fig. 8 we plot the average of these estimated expected returns over the 100 training seeds.

In the Cartpole environment, once an algorithm reaches score 500, it will not see any failure for many episodes in a row. In this case, the agent starts to forget the actions that prevented failure and led it to success, causing the performance to drop before it can rise again. This phenomenon is known as catastrophic forgetting, and leads to large oscillations in learning curves. In order to hide the effect of catastrophic forgetting on the learning curves, in Fig. 5 we eliminated the 50 worst return estimates (corresponding to the 50 worst training seeds) at each point and plotted the mean of top 50 return estimates. The shades in Fig. 5 and Fig. 8 show the 99 percent confidence intervals over the averaged data.

We did not use replay buffer and batch updates. We used a small quadratic regularizer with coefficient 10^{-5} on the weights of the neural network. For all algorithms, we used optimized parameters that maximize area under the curves, as follows. In

Acrobot experiment:

- For TD(0), we used softmax coefficient 1 and Adam optimizer with step-size 0.005.
- For RG, we used softmax coefficient 16 and Adam optimizer with step-size 0.001.
- For RANS, we used softmax coefficient 16 and $\alpha = 0.005$ and set all other parameters were set to their default values described in Algorithm 4.

In Cartpole experiment:

- For TD(0), we used softmax coefficient 0.005 and Adam optimizer with step-size 0.3.
- For RG, we used softmax coefficient 0.002 and Adam optimizer with step-size 0.3.
- For RANS, we used softmax coefficient 8 and $\alpha = 0.001$ and set all other parameters were set to their default values described in Algorithm 4.

In another experiment, we evaluated the performance of RANS on simple MuJoCo environments–Hopper and HalfCheetah. We used the standard (300×400) two layer DDPG neural network architecture (Lillicrap et al., 2015) for the actor and the Q-network. The network for $q_{\mathbf{w}}$ was trained with three algorithms: TD with delayed targets, RG, and RANS (Algorithm 4). For TD and RG we used Adam optimizer. The actor was trained using a deterministic policy gradient, in principle similar to (Lillicrap et al., 2015) but with Adam optimizer and without delayed target actor.

For each algorithm, we performed training for 100 randomly generated random seeds. In Fig. 9 we plot average returns over the 100 training seeds.

We used no replay buffer and no batch updates in our experiments. We employed a small quadratic regularizer with coefficient 10^{-5} on the weights of the neural network for $q_{\mathbf{w}}$. For all algorithms, we optimized all parameters to maximize area under the return curves. Parameters used for the Hopper environment are as follows.

- For TD, we used delayed target with Polyak averaging factor 0.001 and Adam optimizer with step-size 0.005 for Q-network updates. We set $\gamma = 0.99$ and used actor learning rate 10^{-8}.
- For RG, we used Adam optimizer with step-size 0.001 for Q-network updates, actor learning rate 10^{-8}, and $\gamma = 0.999$.
- For RANS, we set $\alpha = 0.001$ and all other parameters were set to their default values (see Algorithm 4). In this experiment, we sued actor learning rate 10^{-7} and $\gamma = 0.999$.

Parameters used for the HalfCheetah environment are as follows:

- For TD, we used delayed target with Polyak averaging factor 4×10^{-5} and Adam optimizer with step-size 3×10^{-5} for Q-network updates. We set $\gamma = 0.99$ and used actor learning rate 10^{-5}.
- For RG, we used Adam optimizer with step-size 10^{-5} for Q-network updates, actor learning rate 10^{-5}, and $\gamma = 0.999$.
- For RANS, we set $\alpha = 0.003$ and all other parameters were set to their default values (see Algorithm 4). In this experiment, we sued actor learning rate 10^{-6} and $\gamma = 0.999$.

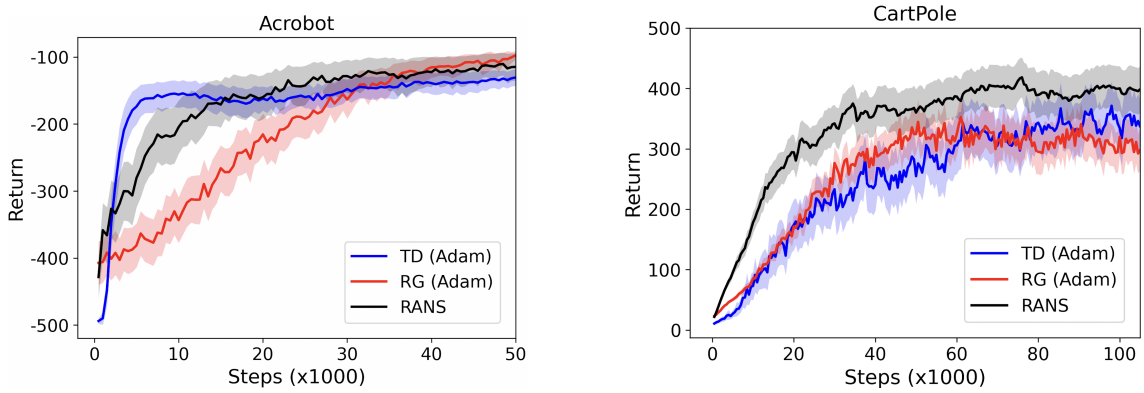

Figure 8. Performance of RANS, TD(0), and RG on classic control tasks. The only difference with Fig.5 is that here, each point is an average of estimate returns over 100 independent training. See Appendix G.4 for details.

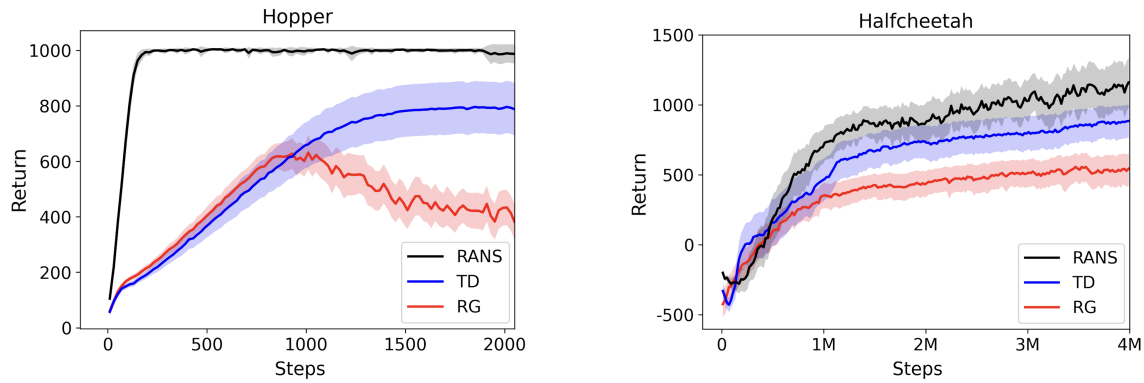

Figure 9. Performance of RANS, TD with target networks, and RG algorithms on simple MuJoCo environments. The only difference with Fig.6 is that here, each point is an average of estimate returns over 100 independent training, whereas Fig.6 depicts the average of top 50 percent of estimated returns.