Regularized Anderson Acceleration for Off-Policy Deep Reinforcement Learning

Wenjie Shi, Shiji Song, Hui Wu, Ya-Chu Hsu, Cheng Wu, Gao Huang*
Department of Automation, Tsinghua University, Beijing, China
Beijing National Research Center for Information Science and Technology (BNRist)
{shiwj16, wuhui14, xuyz17}@mails.tsinghua.edu.cn
{shijis, wuc, gaohuang}@tsinghua.edu.cn

Abstract

Model-free deep reinforcement learning (RL) algorithms have been widely used for a range of complex control tasks. However, slow convergence and sample inefficiency remain challenging problems in RL, especially when handling continuous and high-dimensional state spaces. To tackle this problem, we propose a general acceleration method for model-free, off-policy deep RL algorithms by drawing the idea underlying regularized Anderson acceleration (RAA), which is an effective approach to accelerating the solving of fixed point problems with perturbations. Specifically, we first explain how policy iteration can be applied directly with Anderson acceleration. Then we extend RAA to the case of deep RL by introducing a regularization term to control the impact of perturbation induced by function approximation errors. We further propose two strategies, i.e., progressive update and adaptive restart, to enhance the performance. The effectiveness of our method is evaluated on a variety of benchmark tasks, including Atari 2600 and MuJoCo. Experimental results show that our approach substantially improves both the learning speed and final performance of state-of-the-art deep RL algorithms. The code and models are available at: https://github.com/shiwj16/raa-drl.

1 Introduction

Reinforcement learning (RL) is a principled mathematical framework for experience-based autonomous learning of policies. In recent years, model-free deep RL algorithms have been applied in a variety of challenging domains, from game playing [1, 2] to robot navigation [3, 4]. However, sample inefficiency, i.e., the required number of interactions with the environment is impractically high, remains a major limitation of current RL algorithms for problems with continuous and high-dimensional state spaces. For example, many RL approaches on tasks with low-dimensional state spaces and fairly benign dynamics may even require thousands of trials to learn. Sample inefficiency makes learning in real physical systems impractical and severely prohibits the applicability of RL approaches in more challenging scenarios.

A promising way to improve the sample efficiency of RL is to learn models of the underlying system dynamics. However, learning models of the underlying transition dynamics is difficult and inevitably leads to modelling errors. Alternatively, off-policy algorithms such as deep Q-learning (DQN) [1] and its variants [5, 6], deep deterministic policy gradient (DDPG) [7], soft actor-critic (SAC) [8] and off-policy hierarchical RL [9], which instead aim to reuse past experience, are commonly used to alleviate the sample inefficiency problem. Unfortunately, off-policy algorithms are typically based on policy iteration or value iteration, which repeatedly apply the Bellman operator of interest and generally require an infinite number of iterations to converge exactly to the optima. Moreover, the Bellman iteration constructs a contraction mapping which converges asymptotically to the optimal

*Corresponding auther.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

value function [10]. Iterating this mapping essentially results in a fixed-point problem [11] and thus may be unacceptably slow to converge. These issues are further exacerbated when nonlinear function approximator such as neural network is utilized or the tasks have continuous state and action spaces.

This paper explores how to accelerate the convergence or improve the sample efficiency for model-free, off-policy deep RL. We make the observation that RL is closely linked to fixed-point iteration: the optimal policy can be found by solving a fixed-point problem of associated Bellman operator. Therefore, we attempt to embrace the idea underlying Anderson acceleration (also known as Anderson mixing, Pulay mixing) [12, 13], which is a method capable of speeding up the computation of fixed-point iterations. While the classic fixed-point iteration repeatedly applies the operator to the last estimate, Anderson acceleration searches for the optimal point that has minimal residual within the subspace spanned by several previous estimates, and then applies the operator to this optimal estimate. Prior work [14] has successfully applied Anderson acceleration to value iteration and preliminary experiments show a significant speed up of convergence. However, existing application is only feasible on simple tasks with low-dimensional, discrete state and action spaces. Besides, as far as we know, Anderson acceleration has never been applied to deep RL due to some long-standing issues including biases induced by sampling a minibatch and function approximation errors.

In this paper, Anderson acceleration is first applied to policy iteration under a tabular setting. Then, we propose a practical acceleration method for model-free, off-policy deep RL algorithms based on regularized Anderson acceleration (RAA) [15], which is a general paradigm with a Tikhonov regularization term to control the impact of perturbations. The structure of perturbations could be the noise injected from the outside and high-order error terms induced by a nonlinear fixed-point iteration function. In the context of deep RL, function approximation errors are major perturbation source for RAA. We present two bounds to characterize how the regularization term controls the impact of function approximation errors. Two strategies, i.e., progressive update and adaptive restart, are further proposed to enhance the performance. Moreover, our acceleration method can be implemented readily to deep RL algorithms including Dueling-DQN [5] and twin delayed DDPG (TD3) [16] to solve very complex, high-dimensional tasks, such as Atari 2600 and MuJoCo [17] benchmarks. Finally, the empirical results show that our approach exhibits a substantial improvement in both learning speed and final performance over vanilla deep RL algorithms.

2 Related Work

Prior works have made a number of efforts to improve the sample efficiency and speed up the convergence of deep RL from different respects, such as variance reduction [18, 19], model-based RL [20, 21, 22], guided exploration [23, 24], etc. One of the most widely used techniques is off-policy RL, which combines temporal difference [25] and experience replay [26, 27] so as to make use of all the previous samples before each update to the policy parameters. Though introducing biases by using previous samples, off-policy RL alleviates the high variance in estimation of Q-value and policy gradient [28]. Consequently, fast convergence is rendered when under fine parameter-tuning.

As one kernel technique of off-policy RL, temporal difference is derived from the Bellman iteration which can be regarded as a fixed-point problem [11]. Our work focuses on speeding up the convergence of off-policy RL via speeding up the convergence of the eseential fixed-point problem, and replying on a technique namely Anderson acceleration. This method is exploited by prior work [12, 29] to accelerate the fixed-point iteration by computing the new iteration as linear combination of previous evaluations. In the linear case, the convergence rate of Anderson acceleration has been elaborately analyzed and proved to be equal to or better than fixed-point iteration in [13]. For nonlinear fixed-point iteration, regularized Anderson acceleration is proposed by [15] to constrain the norm of coefficient vector and reduce the impact of perturbations. Recent works [14, 30] have applied the Anderson acceleration to value iteration and deep neural network, and preliminary experiments show that a significant speedup of convergence is achieved. However, there is still no research showing its acceleration effect on deep RL for complex high-dimensional problems, as far as we know.

3 Preliminaries

Under RL paradigm, the interaction between an agent and the environment is described as a Markov Decision Process (MDP). Specifically, at a discrete timestamp t, the agent takes an action a_t in a state s_t and transits to a subsequent state s_{t+1} while obtaining a reward $r_t = r(s_t, a_t)$ from the environment. The transition between states satisfies the Markov property, i.e., $P(s_{t+1}|s_t, a_t, \ldots, s_0, a_0) = P(s_{s+t}|s_t, a_t)$. Usually, the RL algorithm aims to search a policy $\pi(a|s)$ that maximizes the expected

sum of discounted future rewards. Q-value function describes the expected return starting from a state-action pair (s, a): $Q^\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, a_0 = a\right]$, where the policy $\pi(a|s)$ is a function or conditional distribution mapping the state space $\mathcal{S}$ to the action space $\mathcal{A}$.

3.1 Off-policy reinforcement learning

Most off-policy RL algorithms are derived from policy iteration, which alternates between policy evaluation and policy improvement to monotonically improve the policy and the value function until convergence. For complex environments with unknown dynamics and continuous spaces, policy iteration is generally combined with function approximation, and parameterized Q-value function (or critic) and policy function are learned from sampled interactions with environment. Since critic is represented as parameterized function instead of look-up table, the policy evaluation is replaced with an optimization problem which minimizes the squared temporal difference error, the discrepancy between the outputs of critics after and before applying the Bellman operator

$$L(\theta) = \mathbb{E}\left[\left((\mathcal{T}Q_{\theta'})(s, a) - Q_\theta(s, a)\right)^2\right], \tag{1}$$

where typically the Bellman operator is applied to a separate target value network $Q_{\theta'}$ whose parameter is periodically replaced or softly updated with copy of current Q-network weight.

In off-policy RL field, prior works have proposed a number of modifications on the Bellman operator to alleviate the overestimation or function approximation error problems and thus achieved significant improvement. Similar to policy improvement, DQN replaces the current policy with a greedy policy for the next state in the Bellman operator

$$(\mathcal{T}Q_{\theta'})(s_t, a_t) = \mathbb{E}_{s_{t+1}, r_t}\left[r(s_t, a_t) + \gamma \max_a Q_{\theta'}(s_{t+1}, a)\right]. \tag{2}$$

As the state-of-the-art actor-critic algorithm for continuous control, TD3 [16] proposes a clipped double Q-learning variant and a target policy smoothing regularization to modify the Bellman operator, which alleviates overestimation and overfitting problems,

$$(\mathcal{T}Q_{\theta'})(s_t, a_t) = \mathbb{E}_{s_{t+1}, r_t}\left[r(s_t, a_t) + \gamma \min_{j=1,2} Q_{\theta'_j}(s_{t+1}, \pi_{\phi'}(s_{t+1}) + \epsilon)\right], \tag{3}$$

where $Q_{\theta_j}(s, a)(j = 1, 2)$ denote two critics with decoupled parameters θ_j. The added noise $\epsilon \sim \mathrm{clip}(\mathcal{N}(0, \sigma), -c, c)$ is clipped by the positive constant c.

3.2 Anderson acceleration for value iteration

Most RL algorithms are derived from a fundamental framework named policy iteration which consists of two phases, i.e. policy evaluation and policy improvement. The policy evaluation estimates the Q-value function induced by current policy by iterating a Bellman operator from an initial estimate. Following the policy evaluation, the policy improvement acquires a better policy from a greedy strategy, The policy iteration alternates two phases to update the Q-value and the policy respectively until convergence. As a special variant of policy iteration, value iteration merges policy evaluation and policy improvement into one iteration

$$V_{k+1}(s) \leftarrow (\mathcal{T}V_k)(s) = \max_a \mathbb{E}_{s',r}\left[r + \gamma V_k(s')\right], \forall s \in \mathcal{S}, \tag{4}$$

and iterates it until convergence from a initial V_0, where the Bellman operation is only repeatedly applied to the last estimate. Anderson acceleration is a widely used technique to speed up the convergence of fixed-point iterations and has been successfully applied to speed up value iteration [14] by linearly combining previous m $(m > 1)$ value estimates,

$$V_{k+1} \leftarrow \sum_{i=1}^{m} \alpha_i^k \mathcal{T}V_{k-m+i}, \tag{5}$$

where the coefficient vector $\alpha^k \in \mathbb{R}^m$ is determined by minimizing the norm of total Bellman residuals of these estimates,

$$\alpha^k = \underset{\alpha \in \mathbb{R}^m}{\mathrm{argmin}} \left\| \sum_{i=1}^{m} \alpha_i (\mathcal{T}V_{k-m+i} - V_{k-m+i}) \right\|, \text{ s.t. } \sum_{i=1}^{m} \alpha_i = 1. \tag{6}$$

For the ℓ_2-norm, the minimum can be analytically solved by using the Karush-Kuhn-Tucker conditions. Corresponding coefficient vector is given by

$$\alpha^k = \frac{(\Delta_k^T \Delta_k)^{-1}\mathbf{1}}{\mathbf{1}^T(\Delta_k^T \Delta_k)^{-1}\mathbf{1}}, \tag{7}$$

where $\Delta_k = [\delta_{k-m+1}, \dots \delta_k] \in \mathbb{R}^{|\mathcal{S}| \times m}$ is a Bellman residuals matrix with $\delta_i = \mathcal{T}V_i - V_i \in \mathbb{R}^{|\mathcal{S}|}$, and $\mathbf{1} \in \mathbb{R}^m$ denotes the vector with all components equal to one [14].

4 Regularized Anderson Acceleration for Deep Reinforcement Learning

Our regularized Anderson acceleration (RAA) method for deep RL can be derived starting from a direct implementation of Anderson acceleration to the classic policy iteration algorithm. We will first present this derivation to show that the resulting algorithm converges faster to the optimal policy than the vanilla form. Then, a regularized variant is proposed for a more general case with function approximation. Based on this theory, a progressive and practical acceleration method with adaptive restart is presented for off-policy deep RL algorithms.

4.1 Anderson acceleration for policy iteration

As described above, Anderson acceleration can be directly applied to value iteration. However, policy iteration is more fundamental and suitable to scale to deep RL, compared to value iteration. Unfortunately, the implementation of Anderson acceleration is complicated when considering policy iteration, because there is no explicit fixed-point mapping between the policies in any two consecutive steps, which make it impossible to straightforwardly apply Anderson acceleration to the policy π.

Due to the one-to-one mapping between policies and Q-value functions, policy iteration can be accelerated by applying Anderson acceleration to the policy improvement, which establishes a mapping from the current Q-value estimate to the next policy. In this section, our derivation is based on a tabular setting, to enable theoretical analysis. Specifically, for the prototype policy iteration, suppose that estimates have been computed up to iteration k, and that in addition to the current estimate Q^{π_k}, the $m-1$ previous estimates $Q^{\pi_{k-1}}, ..., Q^{\pi_{k-m+1}}$ are also known. Then, a linear combination of estimates Q^{π_i} with coefficients α_i [2] reads

$$Q_\alpha^k = \sum_{i=1}^{m} \alpha_i Q^{\pi_{k-m+i}} \text{ with } \sum_{i=1}^{m} \alpha_i = 1. \tag{8}$$

Due to this equality constraint, we define *combined Bellman operator* $\mathcal{T}_c$ as follows

$$\mathcal{T}_c Q_\alpha^k = \sum_{i=1}^{m} \alpha_i \mathcal{T} Q^{\pi_{k-m+i}}. \tag{9}$$

Then, one searches a coefficient vector α^k that minimizes the following objective function J defined as the combined Bellman residuals among the entire state-action space $\mathcal{S} \times \mathcal{A}$,

$$\alpha^k = \underset{\alpha \in \mathbb{R}^m}{\operatorname{argmin}} J(\alpha) = \underset{\alpha \in \mathbb{R}^m}{\operatorname{argmin}} \left\| \sum_{i=1}^{m} \alpha_i (\mathcal{T} Q^{\pi_{k-m+i}} - Q^{\pi_{k-m+i}}) \right\|, \text{ s.t. } \sum_{i=1}^{m} \alpha_i = 1. \tag{10}$$

In this paper, we will consider the ℓ_2-norm, although a different norm may also be feasible (for example ℓ_1 and ℓ_∞, in which case the optimization problem becomes a linear program). The solution to this optimization problem is identical to (7) except that $\Delta_k = [\delta_{k-m+1}, ..., \delta_k] \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|\times m}$ with $\delta_i = \mathcal{T} Q^{\pi_i} - Q^{\pi_i} \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$. Detailed derivation can be found in Appendix A.1 of the supplementary material. Then, the new policy improvement steps are given by

$$\pi_{k+1}(s) = \underset{a}{\operatorname{argmax}}\, Q_\alpha^k(s,a) = \underset{a}{\operatorname{argmax}} \sum_{i=1}^{m} \alpha_i^k Q^{\pi_{k-m+1}}(s,a), \forall s \in \mathcal{S}. \tag{11}$$

Meanwhile, Q-value estimate $Q^{\pi_{k+1}}$ can be obtained by iteratively applying the following policy evaluation operator by starting from some initial function Q_0,

$$Q_i(s,a) \leftarrow \mathbb{E}_{s\prime,r}\left[r + \gamma \mathbb{E}_{a\prime\sim\pi_{k+1}}[Q_{i-1}(s',a')]\right], \forall (s,a) \in (\mathcal{S},\mathcal{A}). \tag{12}$$

In fact, the effect of acceleration can be explained intuitively. The linear combination Q_α^k is a better estimate of Q-value than the last one Q^{π_k} in terms of combined Bellman residuals. Accordingly, the policy is improved from a better policy baseline corresponding to the better estimate of Q-value.

4.2 Regularized variant with function approximation

For RL control tasks with continuous state and action spaces, or high-dimensional state space, we generally consider the case in which Q-value function is approximated by a parameterized function approximator. If the approximation is sufficiently good, it might be appropriate to use it in place of Q^π in (8)-(12). However, there are several key challenges when implementing Anderson acceleration with function approximation.

[2] Notice that we don't impose a positivity condition on the coefficients.

First, notice that the Bellman residuals in (10) are calculated among the entire state-action space. Unfortunately, sweeping entire state-action space is intractable for continuous RL, and a fine grained discretization will lead to the curse of dimensionality. A feasible alternative to avoid this issue is to use a sampled Bellman residuals matrix $\widetilde{\Delta}_k$ instead. To alleviate the bias induced by sampling a minibatch, we adopt a large sample size N_A specifically for Anderson acceleration.

Second, function approximation errors are unavoidable and lead to biased solution of Anderson acceleration. The intricacies of this issue will be exacerbated by deep models. Therefore, function approximation errors will induce severe perturbation when implementing Anderson acceleration to policy iteration with function approximation. In addition to the perturbation, the solution (7) contains the inverse of a squared Bellman residuals matrix, which may suffer from ill-conditioning when the squared Bellman residuals matrix is rank-deficient, and this is a major source of numerical instability in vanilla Anderson acceleration. In other words, even if the perturbation is small, its impact on the solution can be arbitrarily large.

Under the above observations, we scale the idea underlying RAA to the policy iteration with function approximation in this section. Then, the coefficient vector (10) is now adjusted to $\widetilde{\alpha}^k$ that minimizes the perturbed objective function added with a Tikhonov regularization term,

$$\widetilde{\alpha}^k = \underset{\alpha\in\mathbb{R}^m}{\operatorname{argmin}} \left\| \sum_{i=1}^{m} \alpha_i (\mathcal{T}Q^{\pi_{k-m+i}} - Q^{\pi_{k-m+i}} + e_{k-m+i}) \right\| + \lambda \left\|\alpha\right\|^2, \ \text{s.t.} \sum_{i=1}^{m} \alpha_i = 1, \tag{13}$$

where e_{k-m+i} represents the perturbation induced by function approximation errors. The solution to this regularized optimization problem can be obtained analytically similar to (10),

$$\widetilde{\alpha}^k = \frac{(\widetilde{\Delta}_k^T \widetilde{\Delta}_k + \lambda I)^{-1}\mathbf{1}}{\mathbf{1}^T(\widetilde{\Delta}_k^T \widetilde{\Delta}_k + \lambda I)^{-1}\mathbf{1}}, \tag{14}$$

where λ is a positive scalar representing the scale of regularization. $\widetilde{\Delta}_k = [\widetilde{\delta}_{k-m+1}, ..., \widetilde{\delta}_k] \in \mathbb{R}^{N_A \times m}$ is the sampled Bellman residuals matrix with $\widetilde{\delta}_i = \mathcal{T}Q^{\pi_i} - Q^{\pi_i} + e_i \in \mathbb{R}^{N_A}$.

In fact, the regularization term controls the norm of coefficient vector produced by RAA and reduces the impact of perturbation induced by function approximation errors, as shown analytically by the following proposition.

Proposition 1. *Consider two identical policy iterations $\mathcal{I}_1$ and $\mathcal{I}_2$ with function approximation. $\mathcal{I}_2$ is implemented with regularized Anderson acceleration and takes into account approximation errors, whereas $\mathcal{I}_1$ is only implemented with vanilla Anderson acceleration. Let α^k and $\widetilde{\alpha}^k$ be the coefficient vectors of $\mathcal{I}_1$ and $\mathcal{I}_2$ respectively. Then, we have the following bounds*

$$\|\widetilde{\alpha}^k\| \leq \sqrt{\frac{\lambda + \|\widetilde{\Delta}_k\|^2}{m\lambda}}, \quad \|\widetilde{\alpha}^k - \alpha^k\| \leq \frac{\|\widetilde{\Delta}_k^T\widetilde{\Delta}_k - \Delta_k^T\Delta_k\| + \lambda}{\lambda}\|\alpha^k\|. \tag{15}$$

Proof. See Appendix A.2 of the supplementary material. □

From the above bounds, we can observe that regularization allows a better control of the impact of function approximation errors, but also causes an inevitable gap between $\widetilde{\alpha}^k$ and α^k. Qualitatively, large regularization scale λ means less impact of function approximation errors. On the other hand, overlarge λ leads to very small norm of coefficient vector $\widetilde{\alpha}^k$, which means the coefficients for previous estimates is nearly identical. However, according to (10), equal coefficients are probably far away from the optima α^k and thus result in great performance loss of Anderson acceleration.

4.3 Implementation on off-policy deep reinforcement learning

As discussed in last section, it is impossible to directly use policy iteration in very large continuous domains. To that end, most off-policy deep RL algorithms apply the mechanism underlying policy iteration to learn approximations to both the Q-value function and the policy. Instead of iterating policy evaluation and policy improvement to convergence, these off-policy algorithms alternate between optimizing two networks with stochastic gradient descent. For example, actor-critic method is a well-known implementation of this mechanism. In this section, we show that RAA for policy iteration can be readily extended to existing off-policy deep RL algorithms for both discrete and continuous control tasks, with only a few modifications to the update of critic.

Algorithm 1: RAA-Dueling-DQN Algorithm

Initialize a critic network Q_θ with random parameters θ;
Initialize m target networks $\theta^i \leftarrow \theta$ $(i = 1, ..., m)$ and replay buffer $\mathcal{D}$;
Initialize restart checking period T_r and maximum training steps K;
Set $k = 0$, $c_1 = 1$, $\Delta_{min} = \inf$, $\Delta_{T_r} = 0$;
while $k < K$ **do**
  Receive initial observation state s_0;
  for $t = 1$ *to* T **do**
    Set $k = k + 1$, and $m_k = \min(c_k, m)$;
    With probability ε select a random action a_t, otherwise select $a_t = \mathrm{argmax}_a \, Q_\theta(s_t, a)$;
    Execute a_t, receive r_t and s_{t+1}, store transition (s_t, a_t, r_t, s_{t+1}) into $\mathcal{D}$;
    Sample minibatch of transitions (s, a, r, s') from $\mathcal{D}$;
    Perform Anderson acceleration steps (13)-(14) and obtain $\widetilde{\alpha}^k$, $\Delta_{T_r} = \Delta_{T_r} + \|\widetilde{\delta}_k\|_2^2$;
    Update the critic by minimizing the loss function (18) with y_t equal to the RHS of (17);
    Update target networks every M steps: $\theta^i \leftarrow \theta^{i+1}$ $(i = 1, ..., m-1)$ and $\theta^m \leftarrow \theta$;
    $c_{k+1} = c_k + 1$;
    if $k \mod T_r = 0$ **then**
      $\Delta_{min} = \min(\Delta_{min}, \Delta_{T_r})$;
      if $\Delta_{T_r} > \Delta_{min}$ **then**
        $\Delta_{min} = \inf$, and $c_{k+1} = 1$;

4.3.1 Regularized Anderson acceleration for actor-critic

Consider a parameterized Q-value function $Q_\theta(s_t, a_t)$ and a tractable policy $\pi_\phi(a_t|s_t)$, the parameters of these networks are θ and ϕ. In the following, we first give the main results of RAA for actor-critic. Then, RAA is combined with Dueling-DQN and TD3 respectively.

Under the paradigm of off-policy deep RL (actor-critic), RAA variant of policy iteration (11)-(12) degrades into the following Bellman equation

$$Q_\theta(s_t, a_t) = \mathbb{E}_{s_{t+1}, r_t}\left[r_t + \gamma \sum_{i=1}^{m} \widetilde{\alpha}_i \max_{a_{t+1}} Q_{\theta^i}(s_{t+1}, a_{t+1})\right], \tag{16}$$

where θ^i is the parameters of target network before i update steps. Furthermore, to mitigate the instability resulting from drastic update step of Anderson acceleration, the following progressive Bellman equation (or progressive update) with RAA is used practically,

$$Q_\theta(s_t, a_t) = \beta \sum_{i=1}^{m} \widetilde{\alpha}_i Q_{\theta^i}(s_t, a_t) + (1-\beta)\mathbb{E}_{s_{t+1}, r_t}\left[r_t + \gamma \sum_{i=1}^{m} \widetilde{\alpha}_i \max_{a_{t+1}} Q_{\theta^i}(s_{t+1}, a_{t+1})\right], \tag{17}$$

where β is a small positive coefficient.

Generally, the loss function of critic is then formulated as the following squared consistency error of Bellman equation,

$$L_Q(\theta) = \mathbb{E}_{(s_t, a_t)\in\mathcal{D}}\left[(Q_\theta(s_t, a_t) - y_t)^2\right], \tag{18}$$

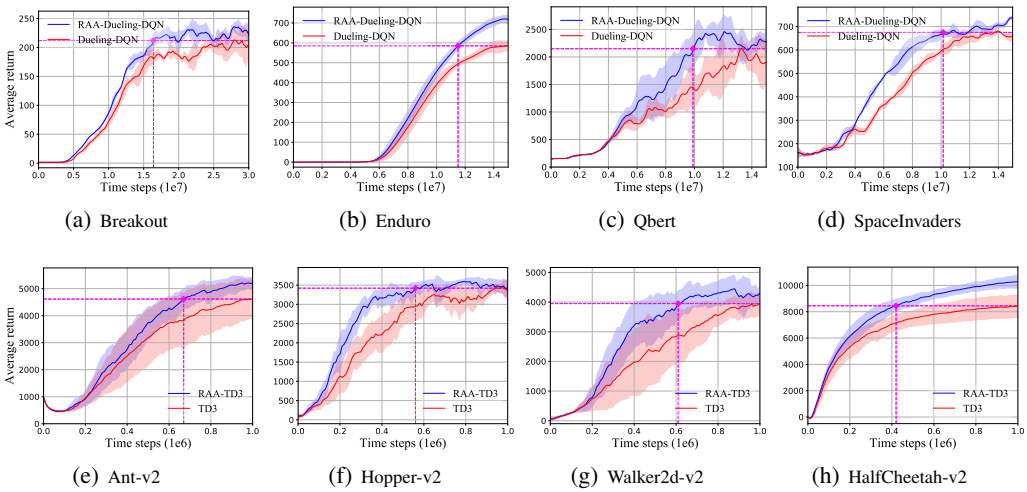
where $\mathcal{D}$ is the distribution of previously sampled transitions, or a replay buffer. The target value of Q-value function or critic is represented by y_t.

RAA-Dueling-DQN. Different from vanilla Dueling-DQN algorithm using general Bellman equation, we instead use progressive Bellman equation with RAA (17) to update the critic. That is, y_t is the RHS of (17) for RAA-Dueling-DQN.

RAA-TD3. For the case of TD3 where an actor and two critics are learned for deterministic policy and Q-value function respectively, the implementation of RAA is more complicated. Specifically, two critics $Q_{\theta_j}(j = 1, 2)$ are simultaneously trained with clipped double Q-learning. Then, the target values $y_{j,t}(j = 1, 2)$ for RAA-TD3 are given by

$$y_{j,t} = \beta \sum_{i=1}^{m} \widetilde{\alpha}_i \widehat{Q}_{\theta^i}(s_t, a_t) + (1-\beta)\mathbb{E}_{s_{t+1}, r_t}\left[r_t + \gamma \sum_{i=1}^{m} \widetilde{\alpha}_i \widehat{Q}_{\theta^i}(s_{t+1}, \pi_{\phi'}(s_{t+1}) + \epsilon)\right], \tag{19}$$

where $\widehat{Q}_{\theta^i}(s_t, a_t) = \min_{j=1,2} Q_{\theta_j^i}(s_t, a_t)$.



(a) Breakout (b) Enduro (c) Qbert (d) SpaceInvaders

(e) Ant-v2 (f) Hopper-v2 (g) Walker2d-v2 (h) HalfCheetah-v2

Figure 1: Learning Curves of Dueling-DQN, TD3 and their RAA variants on discrete and continuous control tasks. The solid curves correspond to the mean and the shaded region to the standard deviation over several trials. Curves are smoothed uniformly for visual clarity.

4.3.2 Adaptive restart

The idea of restarting an algorithm is well known in the numerical analysis literature. Vanilla Anderson acceleration has shown substantial improvements by incorporating with periodic restarts [29], where one periodically starts the acceleration scheme anew by only using information from the most recent iteration. In this section, to alleviate the problem that deep RL is notoriously prone to be trapped in local optimum, we propose an adaptive restart strategy for our RAA method.

Among the training steps of actor-critic with RAA, periodic restart checking steps are enforced to clear the memory immediately before the iteration completely crashes. More explicitly, the iteration is restarted whenever the average squared residual of current period exceeds the average squared residual of last period. Complete description of RAA-Dueling-DQN is summarized in Algorithm 1. And RAA-TD3 is given in Appendix B of the supplementary material.

5 Experiments

In this section, we present our experimental results and discuss their implications. We first give a detailed description of the environments (Atari 2600 and MuJoCo) used to evaluate our methods. Then, we report results on both discrete and continuous control tasks. Finally, we provide an ablative analysis for the proposed methodology. All default hyperparameters used in these experiments are listed in Appendix C of the supplementary material.

5.1 Experimental setup

Atari 2600. For discrete control tasks, we perform experiments in the Arcade Learning Environment. We select four games (Breakout, Enduro, Qbert and SpaceInvaders) varying in their difficulty of convergence. The agent receives $84 \times 84 \times 4$ stacked grayscale images as inputs, as described in [1].

MuJoCo. For continuous control tasks, we conduct experiments in environments built on the MuJoCo physics engine. We select a number of control tasks to evaluate the performance of the proposed methodology and the baseline methods. In each task, the agent takes a vector of physical states as input, and generates an action to manipulate the robots in the environment.

5.2 Comparative evaluation

To evaluate our RAA variant method, we select Dueling-DQN and TD3 as the baselines for discrete and continuous control tasks, respectively. Please note that we do not select DDPG as the baseline for continuous control tasks, as DDPG shows bad performance in difficult control tasks such as robotic manipulation. Figure 1 shows the total average return of evaluation rollouts during training for Dueling-DQN, TD3 and their RAA variants. We train five and seven different instances of each algorithm for Atari 2600 and MuJoCo, respectively. Besides, each baseline and corresponding RAA

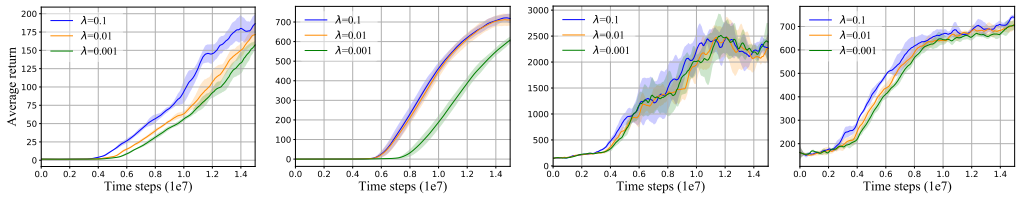
variant are trained with same random seeds set and evaluated every 10000 environment steps, where each evaluation reports the average return over ten different rollouts.

The results in Figure 1 show that, overall, RAA variants outperform to corresponding baseline on most tasks with a large margin such as HalfCheetah-v2 and perform comparably to them on the easier tasks such as Enduro in terms of learning speed, which indicate that RAA is a feasible method to make existing off-policy RL algorithms more sample efficient. In addition to the direct benefit of acceleration mentioned above, we also observe that our RAA variants demonstrate superior or comparable final performance to the baseline methods in all tasks. In fact, RAA-Dueling-DQN can be seen as a weighted variant of Average-DQN [31], which can effectively reduce the variance of approximation error in the target values and thus shows improved performance. In summary, our approach brings an improvement in both the learning speed and final performance.

5.3 Ablation studies

The results in the previous section suggest that our RAA method can improve the sample efficiency of existing off-policy RL algorithms. In this section, we further examine how sensitive our approach is to the scaling of regularization. We also perform ablation studies to understand the contribution of each individual component: progressive update and adaptive restart. Additionally, we analyze the impact of different number of previous estimates m and compare the behavior of our proposed RAA method over different learning rates.

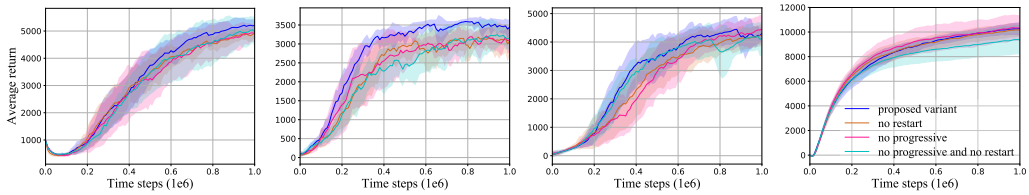
Regularization scale. Our approach is sensitive to the scaling of regularization λ, because it control the norm of the coefficient vector and reduces the impact of approximation error. According to the conclusions of Proposition 1, larger regularization magnitude implies less impact of approximation error, but overlarge regularization will make the coefficients nearly identical and thus result in substantial degradation of acceleration performance. Figure 2 shows how learning performance changes on discrete control tasks when the regularization scale is varied, and consistent conclusion as above can be drawn from Figure 2. For continuous control tasks, it is difficult to obtain same conclusion due to the dominant effect of bias induced by sampling a minibatch relative to function approximation errors. Additional learning curves on continuous control tasks can be found in Appendix D of the supplementary material.



(a) Breakout (b) Enduro (c) Qbert (d) SpaceInvaders

Figure 2: Sensitivity of RAA-Dueling-DQN to the scaling of regularization on discrete control tasks.

Progressive update and adaptive restart. This experiment compares our proposed approach with: (i) RAA without using progressive update (no progressive); (ii) RAA without adding adaptive restart (no restart); (iii) RAA without using progressive update and adding adaptive restart (no progressive and no restart). Figure 3 shows comparative learning curves on continuous control tasks. Although the significance of each component varies task to task, we see that using progressive update is essential for reducing the variance on all four tasks, consistent conclusion can also be drawn from Figure 1. Moreover, adding adaptive restart marginally improves the performance. Additional results on discrete control tasks can be found in Appendix D of the supplementary material.



(a) Ant-v2 (b) Hopper-v2 (c) Walker2d-v2 (d) HalfCheetah-v2

Figure 3: Ablation analysis of RAA-TD3 (blue) over progressive update and adaptive restart.

The number of previous estimates m. In our experiments, the number of previous estimates m is set to 5. In fact, there is a tradeoff between performance and computational cost. Fig.4 shows the results of RAA-TD3 using different $m(m = 1, 3, 5, 7, 9)$ on Walker2d task. Overall, we can conclude that larger m leads to faster convergence and better final performance, but the improvement becomes small when m exceeds a threshold. In practice, we suggest to take into account available computing resource and sample efficiency when applying our proposed RAA method to other works.

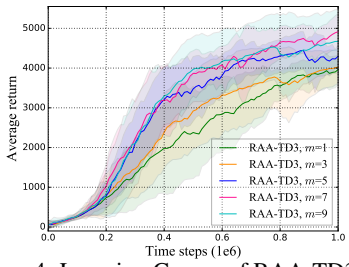


Figure 4: Learning Curves of RAA-TD3 on Walker2d-v2 with different m.

Learning rate. To compare the behavior of our proposed RAA method over different learning rates (lr), we perform additional experiments on Walker2d task, and the results of TD3 and our RAA-TD3 are shown in Fig.5. Overall, the improvement of our method is consistent across all learning rates, though the performance of both TD3 and our RAA-TD3 is bad under the setting with non-optimal learning rates, and the improvement is more significant when the learning rate is smaller. Moreover, consistent improvement of performance means that our proposed RAA method is effective and robust.

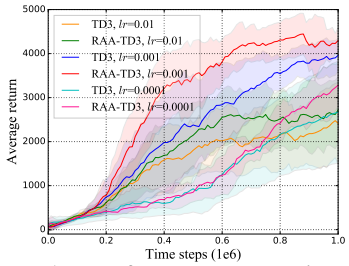


Figure 5: Performance comparison on Walker2d-v2 with different learning rates.

6 Conclusion

In this paper, we presented a general acceleration method for existing deep reinforcement learning (RL) algorithms. The main idea is drawn from regularized Anderson acceleration (RAA), which is an effective approach to speeding up the solving of fixed point problems with perturbations. Our theoretical results explain that vanilla Anderson acceleration can be directly applied to policy iteration under a tabular setting. Furthermore, RAA is extended to model-free deep RL by introducing an additional regularization term. Two rigorous bounds about coefficient vector demonstrate that the regularization term controls the norm of the coefficient vector produced by RAA and reduces the impact of perturbation induced by function approximation errors. Moreover, we verified that the proposed method can significantly accelerate off-policy deep RL algorithms such as Dueling-DQN and TD3. The ablation studies show that progressive update and adaptive restart strategies can enhance the performance. For future work, how to combine Anderson acceleration or its variants with on-policy deep RL is an exciting avenue.

Acknowledgments

Gao Huang is supported in part by Beijing Academy of Artificial Intelligence (BAAI) under grant BAAI2019QN0106 and Tencent AI Lab Rhino-Bird Focused Research Program under grant JR201914. This research is supported by the National Science Foundation of China (NSFC) under grant 41427806.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[3] M. Zhang, Z. McCarthy, C. Finn, S. Levine, and P. Abbeel, "Learning deep neural network policies with continuous memory states," in *2016 IEEE International Conference on Robotics and Automation*, 2016, pp. 520–527.

[4] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu *et al.*, "Learning to navigate in complex environments," in *Proceedings of the International Conference on Learning Representations*, 2017.

[5] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33rd International Conference on Machine Learning*, vol. 48. PMLR, 2016, pp. 1995–2003.

[6] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proceedings of the International Conference on Learning Representations*, 2016.

[8] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. PMLR, 2018, pp. 1861–1870.

[9] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 3303–3313.

[10] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*. Athena Scientific Belmont, MA, 1996, vol. 5.

[11] A. Granas and J. Dugundji, *Fixed point theory*. Springer Science & Business Media, 2013.

[12] H. F. Walker and P. Ni, "Anderson acceleration for fixed-point iterations," *SIAM Journal on Numerical Analysis*, vol. 49, no. 4, pp. 1715–1735, 2011.

[13] A. Toth and C. Kelley, "Convergence analysis for anderson acceleration," *SIAM Journal on Numerical Analysis*, vol. 53, no. 2, pp. 805–819, 2015.

[14] M. Geist and B. Scherrer, "Anderson acceleration for reinforcement learning," *arXiv preprint arXiv:1809.09501*, 2018.

[15] D. Scieur, A. d'Aspremont, and F. Bach, "Regularized nonlinear acceleration," in *Advances In Neural Information Processing Systems*, 2016, pp. 712–720.

[16] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. PMLR, 2018, pp. 1587–1596.

[17] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[18] E. Greensmith, P. L. Bartlett, and J. Baxter, "Variance reduction techniques for gradient estimates in reinforcement learning," *Journal of Machine Learning Research*, vol. 5, no. Nov, pp. 1471–1530, 2004.

[19] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proceedings of the International Conference on Learning Representations*, 2016.

[20] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on Machine Learning*, 2011, pp. 465–472.

[21] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation*, 2017, pp. 1714–1721.

[22] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee, "Sample-efficient reinforcement learning with stochastic ensemble value expansion," in *Advances in Neural Information Processing Systems*, 2018, pp. 8224–8234.

[23] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Advances in Neural Information Processing Systems*, 2014, pp. 1071–1079.

[24] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine, "Path integral guided policy search," in *2017 IEEE International Conference on Robotics and Automation*, 2017, pp. 3381–3388.

[25] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[26] L.-J. Lin, "Reinforcement learning for robots using neural networks," Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, Tech. Rep., 1993.

[27] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," in *Proceedings of the International Conference on Learning Representations*, 2017.

[28] S. Gu, T. Lillicrap, Z. Ghahramani, R. E. Turner, and S. Levine, "Q-prop: Sample-efficient policy gradient with an off-policy critic," in *Proceedings of the International Conference on Learning Representations*, 2017.

[29] N. C. Henderson and R. Varadhan, "Damped anderson acceleration with restarts and monotonicity control for accelerating em and em-like algorithms," *Journal of Computational and Graphical Statistics*, pp. 1–42, 2019.

[30] G. Xie, Y. Wang, S. Zhou, and Z. Zhang, "Interpolatron: Interpolation or extrapolation schemes to accelerate optimization for deep neural networks," *arXiv preprint arXiv:1805.06753*, 2018.

[31] O. Anschel, N. Baram, and N. Shimkin, "Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, 2017, pp. 176–185.