SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

December 2021, Vol. 64 222105:1–222105:15
https://doi.org/10.1007/s11432-019-2889-x

Accelerated value iteration via Anderson mixing

Yujun LI

Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240*, China*

Received 18 August 2019/Revised 9 November 2019/Accepted 21 December 2019/Published online 19 November 2021

Abstract In this paper, we introduce the Anderson acceleration technique developed to be applied to reinforcement learning tasks. We develop an accelerated value iteration algorithm referred as Anderson accelerated value iteration (A2VI) and an accelerated deep Q-learning algorithm denoted as deep Anderson accelerated Q-learning (DA2Q) algorithm. The proposed approach allows improving the performance of value iteration by interpolating historical data. We perform a theoretical analysis on linear convergence and conduct performance evaluation of the proposed algorithms, including synthetic experiments and classical control tasks. We conclude that both theoretical and empirical results confirm the effectiveness of the proposed algorithm.

Keywords reinforcement learning, Q-learning, value iteration, Anderson acceleration, deep neural networks

Citation Li Y J. Accelerated value iteration via Anderson mixing. Sci China Inf Sci, 2021, 64(12): 222105, https://doi.org/10.1007/s11432-019-2889-x

1 Introduction

Reinforcement learning (RL) [1] is used to identify an optimal policy regarding a sequential decision-making problem. Several related algorithms have been proposed over time, including Q-learning [2], SARSA [1, 3], HQ-learning [4, 5], and policy gradient methods [6]. To handle complex state spaces and achieve better generalization performance, researchers have proposed the concept of function approximators [1, 6, 7]. Inspired by the success of deep learning, researchers have applied deep neural networks to the reinforcement learning algorithms [8–12] and achieved impressive results in a wide range of fields such as Atari 2600 [12], non-zero-sum games [13], missile aerodynamic design [14], and music generation [15].

Value iteration (VI) [16] and policy iteration (PI) [17] are the two widely used approaches employed in RL. The main difference between them is that PI seeks to evaluate a current policy accurately during training iterations, which requires a significantly smaller number of policy improvement steps to converge to an optimal value, compared with VI. Although PI has a faster convergence rate than that of VI, most of the existing methods employ VI-based procedures as PI is much more computational costly or even intractable under complex environments.

To retain the fast convergence property of PI, while reducing its computational costs, researchers have proposed several modifications to the original PI approach [18, 19]. The modified PI method [19] is intended to deal with this problem by approximating the solution. It is used to evaluate a policy using the truncated Neumann expansion of an inverse matrix. However, this approximation requires additional iterative steps, which still makes it computationally inefficient in the case of complex decision problems in which sampling is costly.

In recent years, acceleration techniques have attracted increasing attention with respect to both VI and PI. Classical methods for accelerating VI include Gauss-Seidel value iteration [19] and Jacobi value iteration (JAC) [19]. More recently, Alla et al. [18] proposed an acceleration method that switches between a coarse-mesh VI and fine-mesh PI at different stages. Laurini et al. [20] implemented a Jacobi-like acceleration method and applied it to dynamic programming problems. In the recent study [21], the VI procedure was accelerated by only updating a part of the values. However, acceleration techniques in reinforcement learning have not been fully exploited.

Email: liyujun145@gmail.com

In turn, interpolation methods have been widely used in the first-order optimization algorithms [22–25]. These methods allow extracting information from the historical data and are proven to converge faster than the vanilla gradient methods. In the literature, few attempts have been made to apply interpolation techniques to RL. Averaged-DQN [26] was used to calculate the average Q-value based on the historical data and demonstrate that such operation could be successfully applied to variance reduction, which has been widely studied in a variety of studies [27, 28]. Zhang et al. [29] proposed a type-I Anderson acceleration (AA) method for non-smooth fixed-point iterations using convergence analysis; however, rate analysis was not considered. Present study is focused on the type-II AA method, and we conduct the convergence rate analysis in local and global cases.

In this paper, to solve the policy evaluation problem more efficiently, we propose an algorithm based on multi-step interpolation. More specifically, the solution to the policy evaluation problem is approximately represented using a weighted combination of historical values. The weights are adaptively updated by solving the constrained optimization problem comprising historical values. To reduce the computational complexity, we rely on the Anderson mixing method [30–32] to perform the approximation with only a limited amount of historical data. The proposed approach can be used to fill the gap between VI and PI as it allows updating a policy without adding much extra computational complexity to the original VI procedure. Further, we extend this approach to a deep reinforcement learning algorithm.

2 Preliminaries

In this study, we consider a finite-state and finite-action scenario in reinforcement learning. A Markov decision process (MDP) system is defined by a 5-tuple $(\mathcal{S}, \mathcal{A}, P, \boldsymbol{r}, \gamma)$, where $\mathcal{S}$ is a finite state space, $\mathcal{A}$ is a finite action space, $P \in \mathbb{R}^{(|\mathcal{S}|\times|\mathcal{A}|)\times|\mathcal{S}|}$ denotes a collection of state-to-state transition probabilities, $\boldsymbol{r} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ denotes a reward matrix, γ denotes a discount factor. A policy π is a mapping from $\mathcal{S}$ to $\mathcal{A}$, which produces an action for each state. The transition matrix $P_\pi \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ and reward vector $\boldsymbol{r}_\pi \in \mathbb{R}^{|\mathcal{S}|}$ under policy π are defined as $P_\pi(i,j) = P((i,\pi(i)),j)$, $\boldsymbol{r}_\pi(i) = \boldsymbol{r}(i,\pi(i))$, respectively. Given a policy π, the value $\boldsymbol{v}^\pi \in \mathbb{R}^{|\mathcal{S}|}$ and the Q-value $\boldsymbol{Q}^\pi \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ are defined as

$$\boldsymbol{v}^\pi(s) \triangleq \mathbb{E}_{s_0=s, s_{t+1}\sim P_\pi(s_t,\cdot)} \sum_{t=0}^{\infty} \gamma^t \boldsymbol{r}_\pi(s_t),$$

$$\boldsymbol{Q}^\pi(s,a) \triangleq \boldsymbol{r}(s,a) + \mathbb{E}_{s_1\sim P_a(s,\cdot), s_{t+1}\sim P_\pi(s_t,\cdot)} \sum_{t=1}^{\infty} \gamma^t \boldsymbol{r}_\pi(s_t),$$

where $P_a(s,\cdot)$ means the probability distribution of states transited from state s by action a, $P_\pi(s,\cdot)$ means the probability distribution of states transited from state s by policy π. $\boldsymbol{v}^\pi$ satisfies the following Bellman equation:

$$\boldsymbol{v}^\pi = \Gamma_\pi(\boldsymbol{v}^\pi) \triangleq \boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}^\pi.$$

A policy $\pi^* = \operatorname{argmax}_\pi \boldsymbol{v}^\pi$ is called the optimal policy if $\boldsymbol{v}^{\pi^*}(s) \geqslant \boldsymbol{v}^\pi(s)$ for any policy π and state s. Its value or Q-value is defined as $\boldsymbol{v}^*$ or $\boldsymbol{Q}^*$. Note that $\boldsymbol{v}^*$ satisfies the Bellman optimality equation:

$$\boldsymbol{v}^* = \Gamma(\boldsymbol{v}^*) \triangleq \max_\pi(\boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}^*).$$

We use $\boldsymbol{v} \geqslant \boldsymbol{v}'$ to represent $\boldsymbol{v}(s) \geqslant \boldsymbol{v}'(s)$ for any s. Finding an optimal policy is equivalent to finding a fixed point of the operator $\Gamma(\cdot)$. We say $\boldsymbol{v}$ is monotonic improving if $\Gamma(\boldsymbol{v}) \geqslant \boldsymbol{v}$ and define the set of such values by V_B.

2.1 Fixed-point iteration methods

Value iteration is the most widely used and best-understood algorithm for solving Markov decision problems. It solves the fixed-point problem by repeating the following step:

$$\boldsymbol{v}^{(t+1)} = \Gamma(\boldsymbol{v}^{(t)}) = \max_\pi(\boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}^{(t)}).$$

An alternative solution is policy iteration, which updates both the value $\boldsymbol{v}^{(t)}$ and the policy $\pi^{(t)}$ during each iteration. Policy iteration comprises the following two steps.

- (Policy evaluation) Find a $\boldsymbol{v}^{(t)}$ such that

$$\boldsymbol{v}^{(t)} = \Gamma_{\pi^{(t)}}(\boldsymbol{v}^{(t)}) = \boldsymbol{r}_{\pi^{(t)}} + \gamma P_{\pi^{(t)}}\boldsymbol{v}^{(t)}, \tag{1}$$

which can be directly computed by

$$\boldsymbol{v}^{(t)} = (I - \gamma P_{\pi^{(t)}})^{-1}\boldsymbol{r}_{\pi^{(t)}}, \tag{2}$$

or repeating to update $\boldsymbol{v}^{(t)}$ to $\Gamma_{\pi^{(t)}}(\boldsymbol{v}^{(t)})$ [1].

- (Policy improvement) Improve the current policy by

$$\pi^{(t+1)} = \underset{\pi}{\operatorname{argmax}}(\boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}^{(t)}).$$

Theoretical analysis has shown that VI enjoys a γ-linear convergence rate (i.e., $\|\boldsymbol{v}^{(t)} - \boldsymbol{v}^*\|_\infty \leqslant \gamma\|\boldsymbol{v}^{(t-1)} - \boldsymbol{v}^*\|_\infty$), while PI converges much faster with $\|\boldsymbol{v}^{(t)} - \boldsymbol{v}^*\|_\infty \leqslant K\|\boldsymbol{v}^{(t-1)} - \boldsymbol{v}^*\|_\infty^2$, where K is a constant related with γ and the given MDP [19]. Both VI and PI are model-based algorithms because the greedy policy cannot be determined when $\boldsymbol{r}$ and P are unknown. VI can be reformulated in the formulation of $\boldsymbol{Q}(s,a)$, and it is referred to as Q-learning [2]. We will analyze the proposed method under $\boldsymbol{v}$-notation, but the analysis also works under the corresponding $\boldsymbol{Q}$-notation.

The main difference between VI and PI is whether the current policy is fully evaluated. Though PI converges faster than VI, this advantage diminishes in the case where the number of states is large. In this case, the calculation of (2) is quite time-consuming. In the implementation, an alternative approach to (2) is to repeat updating $\boldsymbol{v}^{(t)}$ to $\Gamma_{\pi^{(t)}}(\boldsymbol{v}^{(t)})$ until the error $\|\boldsymbol{v}^{(t)} - \Gamma_{\pi^{(t)}}(\boldsymbol{v}^{(t)})\|$ is tolerated [1]. To provide sufficient accuracy, this procedure still requires a quite number of inner repeating iterations. Therefore, most of deep reinforcement learning algorithms employ a value iteration procedure [9, 11, 12].

2.2 Anderson acceleration

Anderson acceleration is an iterative method for solving fixed-point problems:

$$g(x) = x,$$

where $g : \mathbb{R}^n \to \mathbb{R}^n$ and $x \in \mathbb{R}^n$ [31]. When g is a contraction, a conventional method is to iteratively update $x^{(t)} = g(x^{(t-1)})$ from an arbitrary value $x^{(0)}$. The iterative value $x^{(t)}$ converges to the optimal value as t tends to infinity.

Anderson acceleration method maintains a history of k residuals $f(x^{(i)}) = g(x^{(i)}) - x^{(i)}$, $i \in \{t-1, t-2, \ldots, t-k\}$. To update the value $x^{(t)}$ $(t > k)$, we compute a weighted parameter $\alpha \in \mathbb{R}^k$ by solving the following optimization problem:

$$\alpha = \arg\min_\alpha \left\| \sum_{i=1}^k \alpha_i f(x^{(t-i)}) \right\| \quad \text{s.t.} \quad \sum_i \alpha_i = 1.$$

One could use any norm in the minimization step. Then the updated value $x^{(t)}$ is a weighted sum $x^{(t)} = \sum_{i=1}^k \alpha_i g(x^{(t-i)})$. Inspired by this technique, we utilize a history of values in value iteration and propose an improved value iteration algorithm.

3 Anderson accelerated value iteration

In this section, we demonstrate the Anderson accelerated value iteration (A2VI) algorithm. The algorithm aims to approximately solve the policy evaluation problem, circumventing the matrix inversion and iterative procedures mentioned above.

We introduce the motivation of the proposed algorithm as follows. We utilize the linear property of (1), define $B_\pi(\boldsymbol{v}) = \Gamma_\pi(\boldsymbol{v}) - \boldsymbol{v}$ and convert the problem into an equivalent form of solving the equation $B_\pi(\boldsymbol{v}) = \boldsymbol{0}$. Directly solving the equation is computationally complex and can be implemented by the following process instead. Suppose we have obtained a set of values $B_{\pi^{(t)}}(\boldsymbol{v}^{(1)}), B_{\pi^{(t)}}(\boldsymbol{v}^{(2)}), \ldots, B_{\pi^{(t)}}(\boldsymbol{v}^{(t)})$

with respect to $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(t)}$ and a set of weights $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_t)^{\mathrm{T}}$, subject to $\sum_{i=1}^{t} \alpha_i = 1$ and $\alpha_i \in \mathbb{R}$, which satisfies that

$$\sum_{i=1}^{t} \alpha_i B_{\pi^{(t)}}(\boldsymbol{v}^{(i)}) = \boldsymbol{0}.$$

Consequently, the combination $\tilde{\boldsymbol{v}} = \sum_{i=1}^{t} \alpha_i \boldsymbol{v}^{(i)}$ will satisfy the following relationship:

$$\begin{aligned} B_{\pi}(\tilde{\boldsymbol{v}}) &= \boldsymbol{r}_{\pi^{(t)}} + \gamma P_{\pi^{(t)}} \tilde{\boldsymbol{v}} - \tilde{\boldsymbol{v}} = \sum_{i=1}^{t} \alpha_i (\Gamma_{\pi^{(t)}}(\boldsymbol{v}^{(i)}) - \boldsymbol{v}^{(i)}) \\ &= \sum_{i=1}^{t} \alpha_i B_{\pi^{(t)}}(\boldsymbol{v}^{(i)}) = \boldsymbol{0}. \end{aligned} \tag{3}$$

This relation implies $\tilde{\boldsymbol{v}} = \sum_{i=1}^{t} \alpha_i \boldsymbol{v}^{(i)}$ can be viewed as a solution to (1). However, this procedure needs to keep track of the previous values and recompute $\Gamma_{\pi^{(t)}}$ for each $\boldsymbol{v}^{(i)}$. To reduce the huge memory usage and computing requirements, we keep the most recent values, i.e., $\{\boldsymbol{v}^{(t-i)} \mid i = 1, 2, \ldots, k\}$, and for each $\boldsymbol{v}^{(t-i)}$ we replace $B_{\pi^{(t)}}(\boldsymbol{v}^{(t-i)})$ with the previously computed values $B_{\pi^{(t-i)}}(\boldsymbol{v}^{(t-i)})$. This modification is based on the fact that the recent successive policies do not change dramatically and therefore $B_{\pi^{(t-i)}}(\boldsymbol{v}^{(t-i)}) \approx B_{\pi^{(t)}}(\boldsymbol{v}^{(t-i)})$. In conclusion, this approach is used to approximately solve the policy evaluation problem.

Another critical issue is that there is no guarantee of the existence of $\boldsymbol{\alpha}$ given that k is small, since the dimension of $B_{\pi}(\boldsymbol{v})$ is usually much higher than k, and thereby it is not guaranteed that there is an $\boldsymbol{\alpha}$ to make the weighted sum to zero. Inspired by the Anderson acceleration technique [30, 31, 33], we instead look for a weighted combination of $\{B_{\pi^{(t-i)}}(\boldsymbol{v}^{(t-i)})\}_{i=1}^{k}$ with the weight $\boldsymbol{\alpha}^{(t)}$, where

$$\boldsymbol{\alpha}^{(t)} = \operatorname*{argmin}_{\boldsymbol{\alpha} \in \Omega \cap \Lambda} \| B^{(t)} \boldsymbol{\alpha} \| \tag{4}$$

and $B^{(t)} = (B_{\pi^{(t-1)}}(\boldsymbol{v}^{(t-1)}), B_{\pi^{(t-2)}}(\boldsymbol{v}^{(t-2)}), \ldots, B_{\pi^{(t-k)}}(\boldsymbol{v}^{(t-k)}))$. $\boldsymbol{\alpha}^{(t)}$ is a vector that takes values in the space of $\Omega \cap \Lambda$ where $\Omega = \{\boldsymbol{\alpha} \mid \boldsymbol{1}^{\mathrm{T}} \boldsymbol{\alpha} = 1\}$, and Λ is an extra constraint on the values attainable by $\boldsymbol{\alpha}$. Typically, Λ can be chosen from the following forms:

- Total space, $\Lambda_{\mathrm{tot}} = \mathbb{R}^k$;
- Boxing constraint, $\Lambda_{\mathrm{box}} = \{\boldsymbol{\alpha} \mid -m\boldsymbol{1} \leqslant \boldsymbol{\alpha} \leqslant m\boldsymbol{1}\}$;
- Convex combination constraint, $\Lambda_{\mathrm{cvx}} = \{\boldsymbol{\alpha} \mid \boldsymbol{0} \leqslant \boldsymbol{\alpha} \leqslant \boldsymbol{1}\}$;
- Extrapolation constraint, $\Lambda_{\mathrm{exp}} = \{\boldsymbol{\alpha} \mid \alpha_1 \geqslant 1, \alpha_i \leqslant 0, i = 2, 3, \ldots, k\}$.

An ad-hoc solution is that $\boldsymbol{v}^{(t)} = \sum_{i=1}^{k} \alpha_i^{(t)} \boldsymbol{v}^{(t-i)}$ where $\alpha_i^{(t)}$ is the i-th element of $\boldsymbol{\alpha}^{(t)}$. It will result in that the weighted combination always locates in the subspace expanded by historical values $\boldsymbol{v}^{(t-1)}, \boldsymbol{v}^{(t-2)}, \ldots, \boldsymbol{v}^{(t-k)}$. However, the solution of (1) may be not in such a subspace. To address this problem, we perform an additional value iteration step to the combination. Then we will get the updated value,

$$\boldsymbol{v}^{(t)} = \max_{\pi} \left(\boldsymbol{r}_{\pi} + \gamma P_{\pi} \left[\sum_{i=1}^{k} \alpha_i^{(t)} \boldsymbol{v}^{(t-i)} \right] \right).$$

In the case of L_2-norm and total space, i.e., $\Lambda = \Lambda_{\mathrm{tot}}$, Eq. (4) has a closed-form solution. With an application of Lagrange multiplier method, we have the following equivalent minimax problem:

$$\min_{\boldsymbol{\alpha}} \max_{\lambda \geqslant 0} L(\boldsymbol{\alpha}, \lambda) = \boldsymbol{\alpha}^{\mathrm{T}} (B^{(t)})^{\mathrm{T}} B^{(t)} \boldsymbol{\alpha} + \lambda (\boldsymbol{1}^{\mathrm{T}} \boldsymbol{\alpha} - 1),$$

where λ is the Lagrange multiplier. Using the KKT conditions, the stationarity of $L(\boldsymbol{\alpha}, \lambda)$ suggests that $\partial L / \partial \boldsymbol{\alpha} = 0$, which leads to $\boldsymbol{\alpha} = -\lambda [(B^{(t)})^{\mathrm{T}} B^{(t)}]^{-1} \boldsymbol{1}/2$. Then by the primal feasibility, we can get that $\lambda = -2/(\boldsymbol{1}^{\mathrm{T}} [(B^{(t)})^{\mathrm{T}} B^{(t)}]^{-1} \boldsymbol{1})$. Therefore, the solution of $\boldsymbol{\alpha}^{(t)}$ is $[(B^{(t)})^{\mathrm{T}} B^{(t)}]^{-1} \boldsymbol{1} / \boldsymbol{1}^{\mathrm{T}} [(B^{(t)})^{\mathrm{T}} B^{(t)}]^{-1} \boldsymbol{1}$.

3.1 The algorithm

Based on the previous discussion, we demonstrate the k-step Anderson accelerated value iteration in Algorithm 1. In the first k steps, the value is updated according to the original VI. After k steps, we

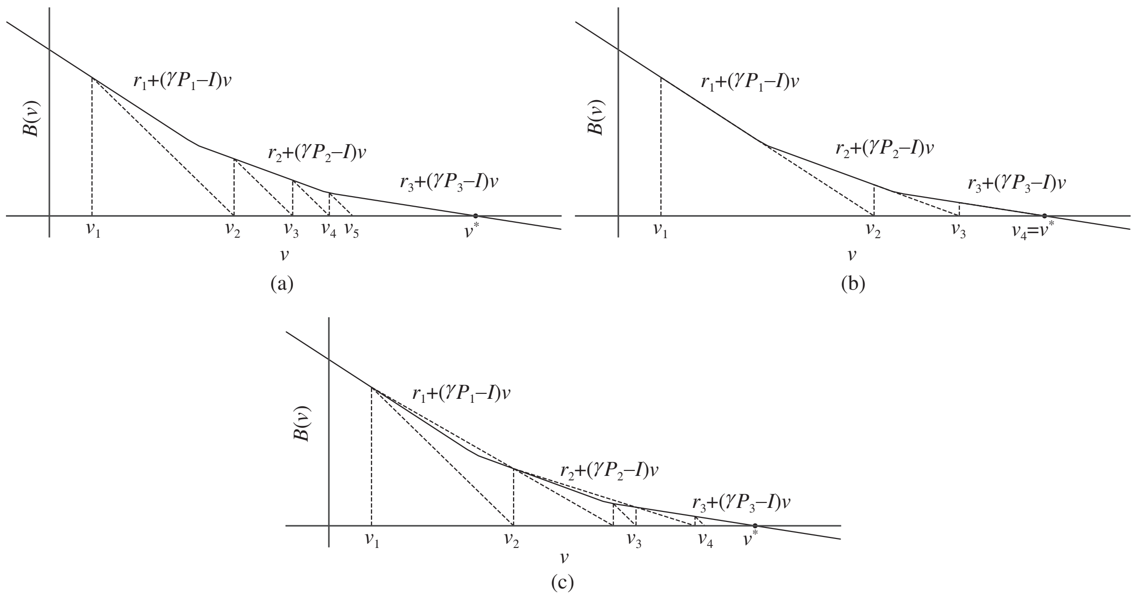

Figure 1 Geometric interpolation of (a) value iteration (VI), (b) policy iteration (PI) and (c) Anderson accelerated value iteration (A2VI).

perform an interpolation procedure, where the weights are attained from solving the problem (4). The original value iteration algorithm is considered as a special case of the proposed algorithm with $k = 1$.

Algorithm 1 Anderson accelerated value iteration (A2VI)

```
1: Input: $\boldsymbol{v}^{(0)}, P, \boldsymbol{r}, \gamma, k, T$.
2: for $t = 1, 2, \dots, T$ do
3:     $B_{\pi^{(t-1)}}(\boldsymbol{v}^{(t-1)}) = \max_{\pi} (\boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}^{(t-1)}) - \boldsymbol{v}^{(t-1)}$;
4:     if $t < k$ then
5:         $\boldsymbol{v}^{(t)} = \max_{\pi}(\boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}^{(t-1)})$;
6:     else
7:         Calculate $(\alpha_1^{(t)}, \alpha_2^{(t)}, \dots, \alpha_k^{(t)})$ by solving the optimization problem (4);
8:         $\boldsymbol{v}^{(t)} = \max_{\pi}(\boldsymbol{r}_\pi + \gamma P_\pi [\sum_{i=1}^{k} \alpha_i^{(t)} \boldsymbol{v}^{(t-i)}])$;
9:     end if
10: end for
11: $\pi^{(T)} = \mathrm{argmax}_\pi (\boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}^{(T)})$;
12: Return: $\boldsymbol{v}^{(T)}, \pi^{(T)}$.
```

Both AA and A2VI have the same spirit of interpolating on historical data. However, A2VI is not a trivial generalization of AA on value iteration. Note that AA has the updating rule $\boldsymbol{v}^{(t)} = \sum_{i=1}^{k} \alpha_i^{(t)} \Gamma(\boldsymbol{v}^{(t-i)})$, while A2VI exchanges the order of the operator `sum` and $\Gamma(\cdot)$, i.e., $\boldsymbol{v}^{(t)} = \Gamma(\sum_{i=1}^{k} \alpha_i^{(t)} \boldsymbol{v}^{(t-i)})$. This exchange puts the nonsmooth operator max in $\Gamma(\cdot)$ out of the affine combination and thereby facilitates the theoretical analysis.

3.2 Compare A2VI with VI and PI

A2VI is a compromise between VI and PI, and we present a geometric explanation on the iterative steps of VI, PI and A2VI in Figure 1. In what follows, we demonstrate an explanation similar to the study of Puterman [19]. In each subfigure, the horizontal axis indicates vector space V where the value $\boldsymbol{v}$ is located. The vertical axis indicates the Bellman residual $B(\boldsymbol{v})$ which is defined as $B(\boldsymbol{v}) = \Gamma(\boldsymbol{v}) - \boldsymbol{v}$ for each value $\boldsymbol{v}$. The three lines $\boldsymbol{r}_i + (\gamma P_i - I)\boldsymbol{v}$ for $i \in \{1, 2, 3\}$ represent the linear relationship between the bellman residual and the value under three policies. Note that the solid polygonal lines indicate the maximum of the three lines for $i \in \{1, 2, 3\}$.

Figure 1 demonstrates the three approaches as follows. In VI, $v_2 = \Gamma(v_1) = \Gamma(v_1) - v_1 + v_1$. Therefore, v_2 is attained by making a vertical line at $(v_1, 0)$, finding its intersection with the solid polygonal line at $(v_1, B(v_1))$, then drawing a line with slope -1 through $(v_1, B(v_1))$ and finding its intersection with the horizon axis at $(v_2, 0)$. Repeat this step in the following values. In PI, we start from the policy improvement and find the optimal policy corresponding to the current value. Then we compute the value corresponding to the optimal policy by solving (2) and get $v_2 = (I - \gamma P_1)^{-1} r_1$. Therefore, v_2 is attained by first getting $(v_1, B(v_1))$ in the same way as value iteration, then calculating the tangent line through $(v_1, B(v_1))$ and finding its intersection with the horizon axis. In A2VI with $k = 2$, we first get v_i for $i \leqslant 2$ through value iteration. When $i = 3$, we use A2VI to get v_3. First, we compute the weight $\boldsymbol{\alpha}$ by solving (4). Assume that the minimum value of (4) is 0. Thus, the weighted value $\sum_{i=1}^{2} \alpha_i v_{3-i}$ is the intersection of the straight line passing through $(v_1, B(v_1))$ and $(v_2, B(v_2))$ with the horizontal axis. Then, we perform one step of value iteration. Therefore, v_3 is attained by first performing in a similar style to policy iteration except that the tangent line is replaced with a secant line. Then a value iteration step is performed to get v_3. The successive values are obtained by repeating this step.

Figure 1 demonstrates that VI only uses the current value of the Bellman residual, while PI is similar to Newton's method [34], using the gradient information to achieve a faster convergence rate. The proposed method serves as an intermediate between them, each step of which is composed of an ordinary value iteration step and a secant step. In specific, the replacement of the tangent line to a secant line can be viewed as a quasi-Newton's method, which is shown computationally more efficient while keeping a fast convergence rate in several particular settings. Both PI and A2VI converge to the fixed point in a smaller number of steps than VI. Compared with PI, A2VI is more practical because it approximates the tangent line by a secant line, which circumvents the costly policy evaluation step.

4 Theoretical analysis

In this section, we conduct theoretical analysis for the proposed algorithm. Before the analysis, a lemma shows the local linearity of the Bellman operator in MDPs with a unique optimal policy.

Lemma 1. For any MDP whose optimal policy is unique, there exists a $\delta > 0$, an optimal value $\boldsymbol{v}^*$ and an optimal policy π^* such that for any $\boldsymbol{v} \in U_\delta(\boldsymbol{v}^*) = \{\boldsymbol{v} | \|\boldsymbol{v} - \boldsymbol{v}^*\|_\infty \leqslant \delta\}$,

$$\Gamma(\boldsymbol{v}) = \boldsymbol{r}_{\pi^*} + \gamma P_{\pi^*} \boldsymbol{v}.$$

Proof. Because the optimal policy is unique, for any nonoptimal policy π, for any state s such that $\pi(s) \neq \pi^*(s)$ we have that $[\Gamma_{\pi^*}(\boldsymbol{v}^*)]_s > [\Gamma_\pi(\boldsymbol{v}^*)]_s$, where $[\cdot]_s$ means executing operations on state s. Let $A(\pi) = \{s | \pi(s) \neq \pi^*(s)\}$.

Suppose the optimal policy is π^*, and then there exists ε such that

$$\min_{\pi \neq \pi^*} \min_{s \in A(\pi)} [\Gamma(\boldsymbol{v}^*) - \Gamma_\pi(\boldsymbol{v}^*)]_s > \varepsilon > 0,$$

since the optimal policy is unique and the state space and the action space are finite.

Consider a value vector $\boldsymbol{v} \in U_\delta(\boldsymbol{v}^*)$, where we choose $\delta = \frac{\varepsilon}{3\gamma}$. Then for any policy π, we have that

$$\|\Gamma_\pi(\boldsymbol{v}^*) - \Gamma_\pi(\boldsymbol{v})\|_\infty = \|\gamma P_\pi(\boldsymbol{v}^* - \boldsymbol{v})\|_\infty \leqslant \gamma \|\boldsymbol{v}^* - \boldsymbol{v}\|_\infty \leqslant \frac{\varepsilon}{3}.$$

For any policy π and any state $s \in A(\pi)$, we have that

$$\begin{aligned}
[\Gamma_{\pi^*}(\boldsymbol{v}) - \Gamma_\pi(\boldsymbol{v})]_s &= [(\Gamma_{\pi^*}(\boldsymbol{v}) - \Gamma_{\pi^*}(\boldsymbol{v}^*)) + (\Gamma_{\pi^*}(\boldsymbol{v}^*) - \Gamma_\pi(\boldsymbol{v}^*)) + (\Gamma_\pi(\boldsymbol{v}^*) - \Gamma_\pi(\boldsymbol{v}))]_s \\
&\geqslant \varepsilon - \|\Gamma_{\pi^*}(\boldsymbol{v}) - \Gamma_{\pi^*}(\boldsymbol{v}^*)\|_\infty - \|\Gamma_\pi(\boldsymbol{v}^*) - \Gamma_\pi(\boldsymbol{v})\|_\infty \\
&\geqslant \varepsilon - \frac{\varepsilon}{3} - \frac{\varepsilon}{3} \\
&= \frac{\varepsilon}{3},
\end{aligned}$$

which means π does not choose the optimal action in state s. Therefore, if π selects the optimal action in every state $s \in \mathcal{S}$, then we must have $\pi_s = \pi_s^*, \forall s \in \mathcal{S}$, which implies $\pi^* = \arg\max_\pi \Gamma_\pi(\boldsymbol{v})$, i.e., $\Gamma(\boldsymbol{v}) = \boldsymbol{r}_{\pi^*} + \gamma P_{\pi^*} \boldsymbol{v}$.

With the above lemma, we present a local convergence analysis of the proposed algorithm under the boxing constraint.

Theorem 1. For any MDP with a unique optimal policy, there exists a $\delta > 0$ and an optimal value $\boldsymbol{v}^*$, such that for any initial value $\boldsymbol{v}^{(0)} \in U_\delta(\boldsymbol{v}^*) = \{\boldsymbol{v} | \|\boldsymbol{v} - \boldsymbol{v}^*\|_\infty \leqslant \delta\}$, the A2VI algorithm under the boxing constraint $(-m\mathbf{1} \leqslant \boldsymbol{\alpha} \leqslant m\mathbf{1})$ maintains the following properties:

(i) $\|\Gamma(\boldsymbol{v}^{(t)}) - \boldsymbol{v}^{(t)}\|_\infty \leqslant \gamma \|\Gamma(\boldsymbol{v}^{(t-1)}) - \boldsymbol{v}^{(t-1)}\|_\infty, \forall t = 1, 2, \ldots$;

(ii) γ-linear convergence rate, $\|\boldsymbol{v}^{(t)} - \boldsymbol{v}^*\|_\infty \leqslant \frac{\gamma^t}{1-\gamma} \|\Gamma(\boldsymbol{v}^{(0)}) - \boldsymbol{v}^{(0)}\|_\infty, \forall t = 1, 2, \ldots$.

Proof. From Lemma 1, we know there exists an optimal value $\boldsymbol{v}^*$ and a $\tilde{\delta} > 0$ such that the optimal Bellman operator is a linear function on $U_{\tilde{\delta}}(\boldsymbol{v}^*)$. We now set δ sufficiently small such that

$$\frac{km(1+\gamma)}{1-\gamma} \|\boldsymbol{v}^{(0)} - \boldsymbol{v}^*\|_\infty < \frac{km(1+\gamma)}{1-\gamma} \delta < \tilde{\delta}.$$

The property (i) is trivial for the first $k-1$ steps, which are performed exactly by standard value iteration. To see the property (ii) for the first $k-1$ steps, note that for any value $\boldsymbol{v}$,

$$\begin{aligned} \|\boldsymbol{v} - \boldsymbol{v}^*\|_\infty &= \|\boldsymbol{v} - \Gamma(\boldsymbol{v}) + \Gamma(\boldsymbol{v}) - \Gamma(\boldsymbol{v}^*)\|_\infty \\ &\leqslant \|\boldsymbol{v} - \Gamma(\boldsymbol{v})\|_\infty + \|\Gamma(\boldsymbol{v}) - \Gamma(\boldsymbol{v}^*)\|_\infty \\ &\leqslant \|\Gamma(\boldsymbol{v}) - \boldsymbol{v}\|_\infty + \gamma \|\boldsymbol{v} - \boldsymbol{v}^*\|_\infty. \end{aligned} \tag{5}$$

Therefore, we have that $\|\boldsymbol{v}^{(t)} - \boldsymbol{v}^*\|_\infty \leqslant \frac{1}{1-\gamma} \|\Gamma(\boldsymbol{v}^{(t)}) - \boldsymbol{v}^{(t)}\|_\infty$. Combining the property (i) in the first $k-1$ steps, it is easy to see that the property (ii) also holds in the first $k-1$ steps.

When $t \geqslant k$, we prove the result by induction. Given t, suppose the conclusion is correct for previous steps $\{0, 1, \ldots, t-1\}$, and then we have that

$$\begin{aligned} \left\| \sum_{i=1}^k \alpha_i^{(t)} \boldsymbol{v}^{(t-i)} - \boldsymbol{v}^* \right\|_\infty &\leqslant \sum_{i=1}^k |\alpha_i^{(t)}| \|\boldsymbol{v}^{(t-i)} - \boldsymbol{v}^*\|_\infty \leqslant \sum_{i=1}^k |\alpha_i^{(t)}| \frac{1}{1-\gamma} \|\Gamma(\boldsymbol{v}^{(t-i)}) - \boldsymbol{v}^{(t-i)}\|_\infty \\ &\leqslant \sum_{i=1}^k |\alpha_i^{(t)}| \frac{1}{1-\gamma} \|\Gamma(\boldsymbol{v}^{(0)}) - \boldsymbol{v}^{(0)}\|_\infty \leqslant \frac{km}{1-\gamma} \|\Gamma(\boldsymbol{v}^{(0)}) - \boldsymbol{v}^{(0)}\|_\infty. \end{aligned} \tag{6}$$

Note that for any value $\boldsymbol{v}$, we have that

$$\begin{aligned} \|\boldsymbol{v} - \boldsymbol{v}^*\|_\infty &= \|\boldsymbol{v} - \Gamma(\boldsymbol{v}) + \Gamma(\boldsymbol{v}) - \Gamma(\boldsymbol{v}^*)\|_\infty \\ &\geqslant \|\boldsymbol{v} - \Gamma(\boldsymbol{v})\|_\infty - \|\Gamma(\boldsymbol{v}) - \Gamma(\boldsymbol{v}^*)\|_\infty \\ &\geqslant \|\Gamma(\boldsymbol{v}) - \boldsymbol{v}\|_\infty - \gamma \|\boldsymbol{v} - \boldsymbol{v}^*\|_\infty. \end{aligned}$$

Therefore, we have that $\|\boldsymbol{v}^{(t)} - \boldsymbol{v}^*\|_\infty \geqslant \frac{1}{1+\gamma} \|\Gamma(\boldsymbol{v}^{(t)}) - \boldsymbol{v}^{(t)}\|_\infty$. Eq. (6) implies that

$$\left\| \sum_{i=1}^k \alpha_i^{(t)} \boldsymbol{v}^{(t-i)} - \boldsymbol{v}^* \right\|_\infty \leqslant \frac{km(1+\gamma)}{1-\gamma} \|\boldsymbol{v}^{(0)} - \boldsymbol{v}^*\|_\infty < \frac{km(1+\gamma)}{1-\gamma} \delta < \tilde{\delta}.$$

It follows that

$$\begin{aligned} \|\boldsymbol{v}^{(t)} - \boldsymbol{v}^*\|_\infty = \left\| \Gamma\left(\sum_{i=1}^k \alpha_i^{(t)} \boldsymbol{v}^{(t-i)} \right) - \Gamma(\boldsymbol{v}^*) \right\|_\infty &\leqslant \gamma \left\| \sum_{i=1}^k \alpha_i^{(t)} \boldsymbol{v}^{(t-i)} - \boldsymbol{v}^* \right\|_\infty \\ \leqslant \left\| \sum_{i=1}^k \alpha_i^{(t)} \boldsymbol{v}^{(t-i)} - \boldsymbol{v}^* \right\|_\infty < \tilde{\delta}. \end{aligned}$$

Therefore, we have that $\sum_{i=1}^k \alpha_i^{(t)} \boldsymbol{v}^{(t-i)} \in U_{\tilde{\delta}}(\boldsymbol{v}^*), \boldsymbol{v}^{(t)} \in U_{\tilde{\delta}}(\boldsymbol{v}^*)$, which implies that

$$\Gamma\left(\sum_{i=1}^k \alpha_i^{(t)} \boldsymbol{v}^{(t-i)} \right) = \boldsymbol{r}_{\pi^*} + \gamma P_{\pi^*} \sum_{i=1}^k \alpha_i^{(t)} \boldsymbol{v}^{(t-i)}, \quad \Gamma(\boldsymbol{v}^{(t)}) = \boldsymbol{r}_{\pi^*} + \gamma P_{\pi^*} \boldsymbol{v}^{(t)}.$$

Then we can get that

$$\Gamma(\boldsymbol{v}^{(t)}) - \boldsymbol{v}^{(t)} = \boldsymbol{r}_{\pi^*} + (\gamma P_{\pi^*} - I) \boldsymbol{v}^{(t)}$$

$$
\begin{aligned}
&= \boldsymbol{r}_{\pi^*} + (\gamma P_{\pi^*} - I)\left(\boldsymbol{r}_{\pi^*} + \gamma P_{\pi^*}\sum_{i=1}^{k}\alpha_i^{(t)}\boldsymbol{v}^{(t-i)}\right)\\
&= \sum_{i=1}^{k}\alpha_i^{(t)}(\boldsymbol{r}_{\pi^*} + (\gamma P_{\pi^*} - I)(\boldsymbol{r}_{\pi^*} + \gamma P_{\pi^*}\boldsymbol{v}^{(t-i)}))\\
&= \sum_{i=1}^{k}\alpha_i^{(t)}\gamma P_{\pi^*}(\boldsymbol{r}_{\pi^*} + \gamma P_{\pi^*}\boldsymbol{v}^{(t-i)} - \boldsymbol{v}^{(t-i)})\\
&= \gamma P_{\pi^*}\sum_{i=1}^{k}\alpha_i^{(t)}(\Gamma(\boldsymbol{v}^{(t-i)}) - \boldsymbol{v}^{(t-i)}).
\end{aligned}
$$

Taking the infinity norm on both sides of the equation and utilizing the definition of $\alpha_i^{(t)}, i = 1, 2, \ldots, k$, which minimizes (4), we get that

$$
\begin{aligned}
\|\Gamma(\boldsymbol{v}^{(t)}) - \boldsymbol{v}^{(t)}\|_\infty &\leqslant \gamma\|P_{\pi^*}\|_\infty\left\|\sum_{i=1}^{k}\alpha_i^{(t)}(\Gamma(\boldsymbol{v}^{(t-i)}) - \boldsymbol{v}^{(t-i)})\right\|_\infty\\
&\leqslant \gamma\|\Gamma(\boldsymbol{v}^{(t-1)}) - \boldsymbol{v}^{(t-1)}\|_\infty = \gamma\|\Gamma(\boldsymbol{v}^{(t-1)}) - \boldsymbol{v}^{(t-1)}\|_\infty.
\end{aligned}
$$

Therefore, we justify property (i). Next, combining (5) and property (i), we have that

$$
\|\boldsymbol{v}^{(t)} - \boldsymbol{v}^*\|_\infty \leqslant \frac{1}{1-\gamma}\|\Gamma(\boldsymbol{v}^{(t)}) - \boldsymbol{v}^{(t)}\|_\infty \leqslant \frac{\gamma^t}{1-\gamma}\|\Gamma(\boldsymbol{v}^{(0)}) - \boldsymbol{v}^{(0)}\|_\infty,
$$

which obtains property (ii) which indicates a γ-linear convergence rate [35].

Generally, it is difficult to obtain the global convergence rate analysis of A2VI, since the operator max is nonsmooth. To guarantee the convergence, we introduce a rejection step to the original algorithm. We propose the A2VI algorithm with the rejection step, which only differs with Algorithm 1 at lines 9–14. After calculating $\boldsymbol{\alpha}^{(t)}$, we test whether the affine combination $\sum_{i=1}^{k}\alpha_i^{(t)}\boldsymbol{v}^{(t-i)}$ lies in $V_B = \{\boldsymbol{v} \mid \Gamma(\boldsymbol{v}) \geqslant \boldsymbol{v}\}$. If the answer is negative, the interpolation step will be replaced with an ordinary value iteration step. We put the pseudocode of A2VI with the rejection step in Algorithm 3 in the experiment. With this modification, we can have the following convergence properties.

Theorem 2. For the A2VI algorithm with the rejection step and the convex combination constraint $\Lambda = \Lambda_{\text{cvx}} = \{\boldsymbol{\alpha} \mid \boldsymbol{0} \leqslant \boldsymbol{\alpha} \leqslant \boldsymbol{1}\}$, if $\boldsymbol{v}^{(0)} \in V_B = \{\boldsymbol{v} \mid \Gamma(\boldsymbol{v}) \geqslant \boldsymbol{v}\}$, then we have that

(i) $\boldsymbol{v}^{(t)} \in V_B,\ \|\Gamma(\boldsymbol{v}^{(t)}) - \boldsymbol{v}^{(t)}\|_\infty \leqslant \gamma\|\Gamma(\boldsymbol{v}^{(t-1)}) - \boldsymbol{v}^{(t-1)}\|_\infty, \forall t = 1, 2, \ldots$;

(ii) γ-linear convergence rate, $\|\boldsymbol{v}^{(t)} - \boldsymbol{v}^*\|_\infty \leqslant \frac{\gamma^t}{1-\gamma}\|\Gamma(\boldsymbol{v}^{(0)}) - \boldsymbol{v}^{(0)}\|_\infty$, $\forall t = 1, 2, \ldots$.

Proof. First, we show that if $\boldsymbol{u} \geqslant \boldsymbol{v}$, then $\Gamma(\boldsymbol{u}) \geqslant \Gamma(\boldsymbol{v})$. As stated in Section 2, $\boldsymbol{u} \geqslant \boldsymbol{v}$ means that $\boldsymbol{u}(s) \geqslant \boldsymbol{v}(s)$ for any state s. Suppose $\tilde{\pi} = \arg\max_\pi \boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}$, therefore for any s we have that

$$
\Gamma(\boldsymbol{u})(s) \geqslant \Gamma_{\tilde{\pi}}(\boldsymbol{u})(s) \geqslant \Gamma_{\tilde{\pi}}(\boldsymbol{v})(s) = \Gamma(v)(s), \tag{7}
$$

which obtains $\Gamma(\boldsymbol{u}) \geqslant \Gamma(\boldsymbol{v})$.

Now, consider the difference between $\Gamma(\boldsymbol{v}^{(t)})$ and $\boldsymbol{v}^{(t)}$,

$$
\begin{aligned}
\Gamma(\boldsymbol{v}^{(t)}) - \boldsymbol{v}^{(t)} &= \max_\pi(\boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}^{(t)}) - \max_\pi\left(\boldsymbol{r}_\pi + \gamma P_\pi\sum_{i=1}^{k}\alpha_i^{(t)}\boldsymbol{v}^{(t-i)}\right)\\
&\leqslant \boldsymbol{r}_{\pi^{(t)}} + \gamma P_{\pi^{(t)}}\boldsymbol{v}^{(t)} - \boldsymbol{r}_{\pi^{(t)}} - \gamma P_{\pi^{(t)}}\sum_{i=1}^{k}\alpha_i^{(t)}\boldsymbol{v}^{(t-i)}\\
&= \gamma P_{\pi^{(t)}}\left(\max_\pi\left(\boldsymbol{r}_\pi + \gamma P_\pi\sum_{i=1}^{k}\alpha_i^{(t)}\boldsymbol{v}^{(t-i)}\right) - \sum_{i=1}^{k}\alpha_i^{(t)}\boldsymbol{v}^{(t-i)}\right)\\
&\leqslant \gamma P_{\pi^{(t)}}\sum_{i=1}^{k}\alpha_i^{(t)}\left(\max_\pi(\boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}^{(t-i)}) - \boldsymbol{v}^{(t-i)}\right)
\end{aligned}
$$

$$= \gamma P_{\pi^{(t)}} \sum_{i=1}^{k} \alpha_i^{(t)} (\Gamma(\boldsymbol{v}^{(t-i)}) - \boldsymbol{v}^{(t-i)}).$$

Combining with the rejection step, we will get that $\Gamma(\boldsymbol{v}^{(t)}) - \boldsymbol{v}^{(t)} \geqslant 0$. First, we show that if $\boldsymbol{v} \in V_B$, then $\Gamma(\boldsymbol{v}) \in V_B$. If $\boldsymbol{v} \in V_B$, then $\Gamma(\boldsymbol{v}) \geqslant \boldsymbol{v}$. Thus, Eq. (7) suggests that $\Gamma(\Gamma(\boldsymbol{v})) \geqslant \Gamma(\boldsymbol{v})$, i.e., $\Gamma(\boldsymbol{v}) \in V_B$.

Then, we show that if $\boldsymbol{v}^{(i)} \in V_B$ for $i < t$, then $\boldsymbol{v}^{(t)} \in V_B$. According to the rejection algorithm, we will get a set of $\alpha_i^{(t)}$ by solving (4). If $\Gamma(\sum_{i=1}^{k} \alpha_i^{(t)} \boldsymbol{v}^{(t-i)}) \geqslant \sum_{i=1}^{k} \alpha_i^{(t)} \boldsymbol{v}^{(t-i)}$, then $\sum_{i=1}^{k} \alpha_i^{(t)} \boldsymbol{v}^{(t-i)} \in V_B$ and $\boldsymbol{v}^{(t)} = \Gamma(\sum_{i=1}^{k} \alpha_i^{(t)} \boldsymbol{v}^{(t-i)}) \in V_B$. If $\Gamma(\sum_{i=1}^{k} \alpha_i^{(t)} \boldsymbol{v}^{(t-i)}) < \sum_{i=1}^{k} \alpha_i^{(t)} \boldsymbol{v}^{(t-i)}$, then $\boldsymbol{v}^{(t)} = \Gamma(\boldsymbol{v}^{(t-1)})$ owing to the rejection step. Then $\boldsymbol{v}^{(t-1)} \in V_B$ suggests that $\boldsymbol{v}^{(t)} \in V_B$. Therefore, we have that $\Gamma(\boldsymbol{v}^{(t)}) - \boldsymbol{v}^{(t)} \geqslant 0$. Further, we can get that

$$\begin{aligned}
\|\Gamma(\boldsymbol{v}^{(t)}) - \boldsymbol{v}^{(t)}\|_\infty &\leqslant \gamma \|_\infty P_{\pi^{(t)}}\|_\infty \left\| \sum_{i=1}^{k} \alpha_i^{(t)} (\Gamma(\boldsymbol{v}^{(t-i)}) - \boldsymbol{v}^{(t-i)}) \right\|_\infty \\
&\leqslant \gamma \|\Gamma(\boldsymbol{v}^{(t-1)}) - \boldsymbol{v}^{(t-1)}\|_\infty.
\end{aligned}$$

The second inequality is owing to the definition of $\alpha_i^{(t)}$, $i = 1, 2, \ldots, k$.

Next, we demonstrate the property (ii) in the following. According to the triangle inequality of norm and $\Gamma(\pi^*) = \pi^*$, we have that

$$\begin{aligned}
\|\boldsymbol{v}^{(t)} - \boldsymbol{v}^*\|_\infty &= \|\boldsymbol{v}^{(t)} - \Gamma(\boldsymbol{v}^{(t)}) + \Gamma(\boldsymbol{v}^{(t)}) - \boldsymbol{v}^*\|_\infty \\
&\leqslant \|\boldsymbol{v}^{(t)} - \Gamma(\boldsymbol{v}^{(t)})\|_\infty + \|\Gamma(\boldsymbol{v}^{(t)}) - \Gamma(\boldsymbol{v}^*)\|_\infty \\
&\leqslant \gamma \|\boldsymbol{v}^{(t-1)} - \Gamma(\boldsymbol{v}^{(t-1)})\|_\infty + \gamma \|\boldsymbol{v}^{(t)} - \boldsymbol{v}^*\|_\infty.
\end{aligned}$$

The last inequality is established owing to the property (i) and the contraction of the operator Γ. Thus, we can get that $(1-\gamma)\|\boldsymbol{v}^{(t)} - \boldsymbol{v}^*\|_\infty \leqslant \gamma \|\Gamma(\boldsymbol{v}^{(t-1)}) - \boldsymbol{v}^{(t-1)}\|_\infty$. Substituting the property (i) into this inequation, we can obtain the property (ii) that indicates a γ-linear convergence rate [35].

Theorem 3. For the A2VI algorithm with the rejection step and the extrapolation constraint $\Lambda = \Lambda_{\exp} = \{\boldsymbol{\alpha} \mid \alpha_1 \geqslant 1, \alpha_i \leqslant 0, i = 2, 3, \ldots, k\}$, if $\boldsymbol{v}^{(0)} \geqslant \boldsymbol{0}$ and $\boldsymbol{v}^{(0)} \in V_B$, then we have

(i) monotone improving values, $\boldsymbol{v}^{(t-1)} \leqslant \boldsymbol{v}^{(t)} \leqslant \boldsymbol{v}^*, \boldsymbol{v}^{(t)} \in V_B, \forall t = 1, 2, \ldots$;

(ii) γ-linear convergence rate, $\|\boldsymbol{v}^* - \boldsymbol{v}^{(t)}\|_\infty \leqslant \gamma \|\boldsymbol{v}^* - \boldsymbol{v}^{(t-1)}\|_\infty$.

Proof. First, we consider $t \in \{1, 2, \ldots, k-1\}$. A2VI performs the value iteration step $\boldsymbol{v}^{(t)} = \Gamma(\boldsymbol{v}^{(t-1)})$. Eq. (7) suggests that $\boldsymbol{v}^{(t-1)} \leqslant \boldsymbol{v}^{(t)}$. The property (ii) is evident when taking value iteration step.

Then, we consider the case when $t \geqslant k$. We give a proof by induction. Suppose the conclusion holds for the first $t-1$ steps, and then

$$\begin{aligned}
\boldsymbol{v}^{(t)} &= \max_\pi \left(\boldsymbol{r}_\pi + \gamma P_\pi \sum_{i=1}^{k} \alpha_i^{(t)} \boldsymbol{v}^{(t-i)} \right) \\
&\geqslant \boldsymbol{r}_{\tilde{\pi}} + \gamma P_{\tilde{\pi}} \left(\sum_{i=1}^{k} \alpha_i^{(t)} \boldsymbol{v}^{(t-i)} \right) \\
&\geqslant \boldsymbol{r}_{\tilde{\pi}} + \gamma P_{\tilde{\pi}} \boldsymbol{v}^{(t-1)} \\
&\geqslant \boldsymbol{v}^{(t-1)},
\end{aligned}$$

where $\tilde{\pi} = \arg\max_\pi \boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}^{(t-1)}$. The second inequality comes from the extrapolation restriction. The third inequality is owing to that $\boldsymbol{v}^{(t-1)} \in V_B$, which is shown in Theorem 2. Note that $\boldsymbol{v}^{(t)}$ monotonously converges to $\boldsymbol{v}^*$, thus $\boldsymbol{v}^{(t)} \leqslant \boldsymbol{v}^*$. Consider the difference between $\boldsymbol{v}^*$ and $\boldsymbol{v}^{(t)}$, we have that

$$\begin{aligned}
\boldsymbol{v}^* - \boldsymbol{v}^{(t)} &= \boldsymbol{v}^* - \max_\pi \left(\boldsymbol{r}_\pi + \gamma P_\pi \sum_{i=1}^{k} \alpha_i^{(t)} \boldsymbol{v}^{(t-i)} \right) \\
&\leqslant \boldsymbol{v}^* - \left(\boldsymbol{r}_{\pi^*} + \gamma P_{\pi^*} \sum_{i=1}^{k} \alpha_i^{(t)} \boldsymbol{v}^{(t-i)} \right)
\end{aligned}$$

$$= \gamma P_{\pi^*} \sum_{i=1}^{k} \alpha_i^{(t)} (\boldsymbol{v}^* - \boldsymbol{v}^{(t-i)})$$
$$\leqslant \gamma P_{\pi^*} (\boldsymbol{v}^* - \boldsymbol{v}^{(t-1)}).$$

Taking the infinite norm on both sides, we get that

$$\|\boldsymbol{v}^* - \boldsymbol{v}^{(t)}\|_\infty \leqslant \gamma \|P_{\pi^*}\|_\infty \|\boldsymbol{v}^* - \boldsymbol{v}^{(t-1)}\|_\infty \leqslant \gamma \|\boldsymbol{v}^* - \boldsymbol{v}^{(t-1)}\|_\infty.$$

This is exactly property (ii). Thus, we can get that $\|\boldsymbol{v}^{(t)} - \boldsymbol{v}^*\|_\infty \leqslant \gamma^t \|\boldsymbol{v}^{(0)} - \boldsymbol{v}^*\|_\infty$, which indicates a linear convergence rate [35].

5 Deep Anderson accelerated Q-learning

Combined with the technique of deep learning, the proposed method can be applied to the DQN algorithm [12], resulting in the deep Anderson accelerated Q-learning (DA2Q) algorithm (Algorithm 2).

Algorithm 2 Deep Anderson accelerated Q-learning (DA2Q)

```
1: **Input:** $M, N, T, \gamma, b, B, \varepsilon, \eta, K, C$.
2: Initialize replay memory $\mathcal{D}$ to capacity $N$, initialize Q-value function $Q$ with random weights $\theta$;
3: $\theta_{-k} = \theta$, $\alpha_1 = 1, \alpha_k = 0$ for $k = 2, \dots, K$;
4: $s = 0$;
5: for episode $= 1, 2, \dots, M$ do
6:     Get an initial state $s_1$;
7:     for $t = 1, 2, \dots, T$ do
8:         With probability $\varepsilon$ select a random action $a_t$, otherwise select $a_t = \arg\max_a Q(s_t, a; \theta)$;
9:         Execute action $a_t$, observe reward $r_t$ and state $s_{t+1}$;
10:        Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$;
11:        Sample a random minibatch of transitions $\{(s_j, a_j, r_j, s'_j)\}_{j=1}^{b}$ from $\mathcal{D}$;
12:        for $j = 1, 2, \dots, b$ do
13:            $y_j = \begin{cases} r_j, & \text{for terminal state,} \\ r_j + \gamma \max_a (\sum_{k=1}^{K} \alpha_k Q(s'_j, a; \theta_{-k})), & \text{for non-terminal state,} \end{cases}$
14:        end for
15:        $L(\theta) = \frac{1}{b} \sum_{j=1}^{b} (y_j - Q(s_j, a_j; \theta))^2$;
16:        $\theta = \theta - \eta \frac{\partial L}{\partial \theta}$;
17:        $s = s + 1$;
18:        if $s \bmod C = 0$ then
19:            Assign $\theta_{-k} = \theta_{-(k-1)}$ for $k = K, K-1, \dots, 2$. Assign $\theta_{-1} = \theta$.
20:            if $s \geqslant K(C-1)$ then
21:                Sample a random minibatch of transitions $\{(s_j, a_j, r_j, s'_j)\}_{j=1}^{B}$ from $\mathcal{D}$;
22:                for $j = 1, 2, \dots, B$ do
23:                    for $k = 1, 2, \dots, K$ do
24:                        $d_j^k = \begin{cases} r_j - Q(s_j, a_j; \theta_{-k}), & \text{for terminal state,} \\ r_j + \gamma \max_a Q(s'_j, a; \theta_{-k}) - Q(s_j, a_j; \theta_{-k}), & \text{for non-terminal state,} \end{cases}$
25:                    end for
26:                end for
27:            end if
28:            Get parameters $\alpha_1, \alpha_2, \dots, \alpha_K$ by solving (4).
29:        end if
30:    end for
31: end for
```

Considering a state-action value matrix $\boldsymbol{Q} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$, we can get a fixed-point equation:

$$\boldsymbol{Q} = \Gamma(\boldsymbol{Q}),$$

where $\boldsymbol{Q}(s,a)$ is given by the Bellman equation $\boldsymbol{Q}(s,a) = \boldsymbol{r}(s,a) + \gamma \max_{a'} \boldsymbol{Q}(s,a')$. Here we use the same notation $\Gamma(\cdot)$ in Section 2 for consistency. Let us define the residual value of $\boldsymbol{Q}$ as

$$F(\boldsymbol{Q}) = \Gamma(\boldsymbol{Q}) - \boldsymbol{Q},$$

which must vanish since the operator $\Gamma(\cdot)$ is a contraction. For a state-action pair (s,a), we have that $F(\boldsymbol{Q}(s,a)) = \Gamma(\boldsymbol{Q}(s,a)) - \boldsymbol{Q}(s,a)$. Given the latest iterative value $\boldsymbol{Q}^{(t-1)}$ and the previous $k-1$ ones

$\boldsymbol{Q}^{(t-2)}, \boldsymbol{Q}^{(t-3)}, \ldots, \boldsymbol{Q}^{(t-k)}$, the new one $\boldsymbol{Q}^{(t)}$ is calculated by solving the following problem:

$$\min_{\alpha} \left\| \sum_{i=1}^{k} \alpha_i F(\boldsymbol{Q}^{(t-i)}) \right\|_F, \quad \text{s.t.} \quad \sum_i \alpha_i = 1,$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ is its affine coordinates, and $\| \cdot \|_F$ is the Frobenius norm. Then we can get $\boldsymbol{Q}^{(t)} = \Gamma(\sum_{i=1}^{k} \alpha_i \boldsymbol{Q}^{(t-i)})$ by the following method in Section 3.

In practice, we cannot access the whole state space and action space. We use the technique of replay memory and collect previous state-action pairs [12]. We randomly sample state-action pairs to update iterative values. Let B denote a minibatch of sampled pairs, Q_B denote a vector by stacking $\{Q(s,a)\}_{(s,a)\in B}$. Then we can get the following optimization problem:

$$\min_{\alpha} \left\| \sum_{i=1}^{k} \alpha_i F(Q_B^{(t-i)}) \right\|_2, \quad \text{s.t.} \quad \sum_i \alpha_i = 1.$$

The state-action value $\boldsymbol{Q}(s,a)$ is parameterized by a neural network $\boldsymbol{Q}(s,a) = \boldsymbol{Q}(s,a;\theta)$ where θ is the weights of the neural network. We update the affine coordinates every C time steps and therefore get the deep Anderson accelerated Q-learning algorithm. Please see Algorithm 2 for the details.

6 Experiments

To validate the applicability of the proposed method, we conduct several experiments.

6.1 Synthetic experiments

We consider decision problems on random MDP. Suppose that there are M states and N actions. The transition probabilities of MDP are generated from a uniform distribution within $[0,1]$ and are normalized to guarantee that $\sum_{j=1}^{M} P(a, s_i, s_j) = 1$ for all a and s_i. Rewards are generated from a standard normal distribution. The discount factor γ is set equal to 0.9. We compare the proposed algorithm with the several alternative approaches, as presented in Figure 2.

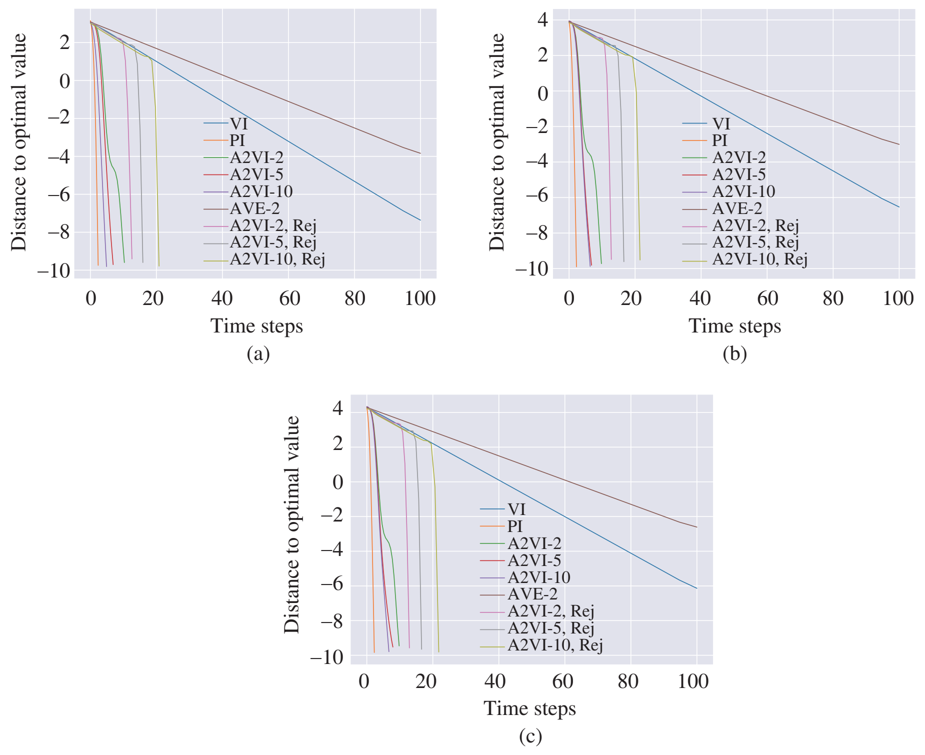

Figure 2 (Color online) Experimental results on synthetic Markov decision processes. The distance to optimal value is expressed in logarithmic scale. The number after the algorithm name represents the number of historical values.

Table 1 Convergence rates of different algorithms in the problem of random MDP

	VI	PI	A2VI-2	A2VI-5	A2VI-10	AVE-2	A2VI-2, Rej	A2VI-5, Rej	A2VI-10, Rej
10 states, 3 actions	0.7857	0.0001	0.0314	0.0033	0.0013	0.8527	0.0008	0.0007	0.0005
20 states, 5 actions	0.7861	0.0001	0.0266	0.0041	0.0017	0.8530	0.0008	0.0005	0.0006
20 states, 10 actions	0.7862	0.0001	0.0268	0.0074	0.0021	0.8532	0.0009	0.0006	0.0006

- VI: value iteration,
- PI: policy iteration,
- AVE: averaged value iteration,
- A2VI: Anderson accelerated value iteration, and
- A2VI, Rej: Anderson accelerated value iteration with the rejection step (see Algorithm 3 for details).

With regard to the above methods, we compare the distance from the values provided by them to the optimal value during the iterative training process. The optimal value is obtained by successive iterations using PI 1000 times. AVE is implemented by setting equal weights in A2VI. Figure 2 demonstrates the three experimental results with different numbers of states and actions, from left to right: (1) 10 states and three actions, (2) 20 states and five actions, and (3) 20 states and 10 actions, respectively. Each curve represents an average of the distances in the training processes with 10 random initializations. Figure 2 demonstrates the logarithmic value of $\|\boldsymbol{v}^{(t)} - \boldsymbol{v}^*\|_2$ corresponding to time step t.

Algorithm 3 Anderson accelerated value iteration with the rejection step

```
1: Input: $\boldsymbol{v}^{(0)}, P, \boldsymbol{r}, \gamma, k, T$.
2: for $t = 1, 2, \ldots, T$ do
3:     $B_{\pi^{(t-1)}}(\boldsymbol{v}^{(t-1)}) = \max_\pi (\boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}^{(t-1)}) - \boldsymbol{v}^{(t-1)}$;
4:     if $t < k$ then
5:         $\boldsymbol{v}^{(t)} = \max_\pi(\boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}^{(t-1)})$;
6:     else
7:         Calculate $(\alpha_1^{(t)}, \alpha_2^{(t)}, \ldots, \alpha_k^{(t)})$ by solving the optimization problem (4);
8:         $\tilde{\boldsymbol{v}} = \sum_{i=1}^k \alpha_i^{(t)} \boldsymbol{v}^{(t-i)}$;
9:         if $\max_\pi (\boldsymbol{r}_\pi + \gamma P_\pi \tilde{\boldsymbol{v}}) \geqslant \tilde{\boldsymbol{v}}$ then
10:            $\boldsymbol{v}^{(t)} = \max_\pi (\boldsymbol{r}_\pi + \gamma P_\pi \tilde{\boldsymbol{v}})$;
11:        else
12:            $\alpha_1^{(t)} = 1$, $\alpha_i^{(t)} = 0$ for $i \neq 1$;
13:            $\boldsymbol{v}^{(t)} = \max_\pi(\boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}^{(t-1)})$;
14:        end if
15:    end if
16: end for
17: $\pi^{(T)} = \mathrm{argmax}_\pi(\boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}^{(T)})$;
18: Return: $\boldsymbol{v}^{(T)}, \pi^{(T)}$.
```

From the results presented in Figure 2, it can be seen that PI has the fastest convergence among all other methods. However, each step of PI includes 100 extra inner iterations to evaluate the policy. Among the considered VI methods, A2VI is observed to be much faster than others. At the early training stage, A2VI with the rejection step shows the same convergence speed as VI and demonstrates the faster convergence speed at subsequent stages. This is owing to the fact that the condition in line 9 of Algorithm 3 is not satisfied at the early training stage, but is satisfied later. Table 1 demonstrates the convergence rates of the considered algorithms in practice. It can be seen that PI has the fastest convergence rate. We can observe that the convergence rate of A2VI in practice is consistent with that in theoretical analysis by not exceeding the uppper bound.

Next, we investigate the influence of the historical data length in A2VI; specifically, we consider the length of k of 2, 5, and 10. We can see that increasing k leads to a decrease in the number of iteration steps. However, we do not observe this phenomenon in the case of A2VI with the rejection step. This is because a larger k requires more time steps to satisfy the condition in line 9 of Algorithm 3.

6.2 Experiments with deep learning based techniques

To evaluate the performance of the proposed method in complex environments, we apply it to the classical control tasks in Gym [36]. Herein, we compare DA2Q with DQN [12] and Averaged-DQN (AVE) [26]. As the transition probability is unknown in these model-free tasks, we do not test the performance of DA2Q with the rejection step.

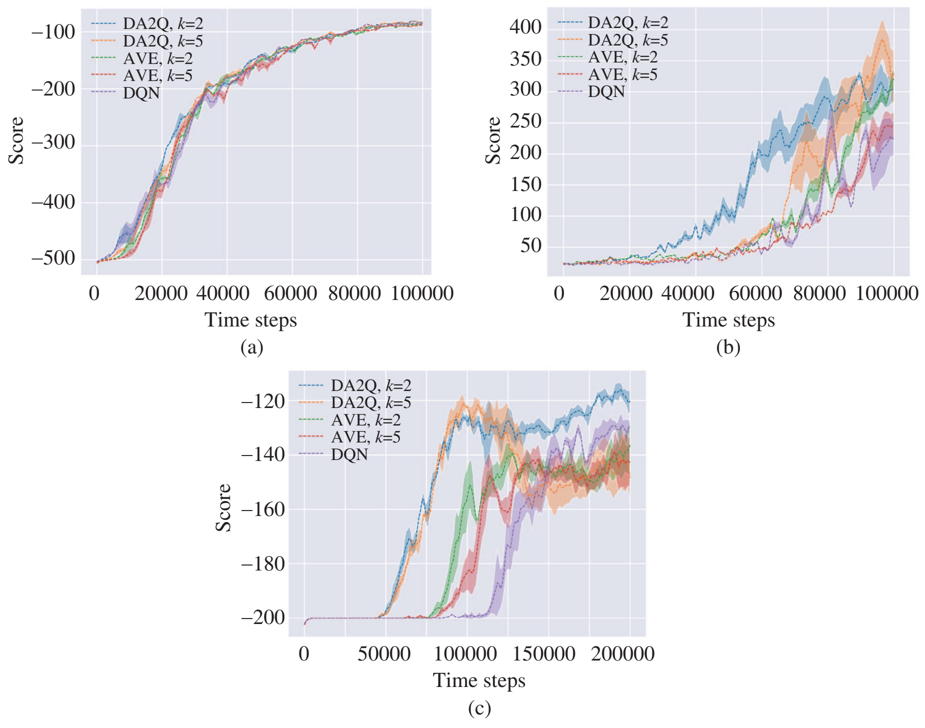

Figure 3 (Color online) Training performance on control tasks. The dashed line is the mean score. The shaded area is the 0.25 standard deviation. DA2Q is the proposed algorithm. AVE is averaged-DQN [26]. DQN is deep Q-learning [12]. (a) Acrobot; (b) CartPole; (c) MountainCar.

In the experimental setups of control tasks, we consider the neural architecture in DQN [12] in which the Q-value network is composed of the three fully connected hidden layers and one fully connected output one. Each layer except the final one is followed by a rectified linear unit (ReLU). The first fully connected layer has 128 units; the second has 128 units; the third has 64 units; and the final layer outputs the state-action values. We use the Adam optimizer [37] with the learning rate of 0.0001. The discount factor is set equal to $\gamma = 0.99$. The exploration policy is an ε-greedy policy with a linearly decayed rate ε.

Figure 3 shows the result of the experiments on the classical control tasks. In the Acrobot task, all considered methods demonstrate the similar performance. In the CartPole and MountainCar tasks, DA2Q shows the distinct performance of acceleration compared with the other algorithms. Thereafter, we compare the performance of DA2Q with $k = 2$ and $k = 5$. Unlike A2VI, DA2Q does not show that a larger k leads to better performance. This is owing to the fact that the neural network is employed for value function approximation in the model-free environment, and the policy evaluation is estimated from a batch of samples, which cannot provide the sufficient accuracy. In the CartPole task, DA2Q with $k = 5$ attains a higher final score, while DA2Q with $k = 2$ demonstrates faster acceleration at the early stage of the training process. In the MountainCar task, DA2Q with $k = 2$ and with $k = 5$ have the similar performance in terms of acceleration at the early stage. However, the performance of DA2Q with $k = 5$ deteriorates after it attains a high score. In total, DA2Q with $k = 2$ achieves the most stable performance in terms of the control tasks.

Compared with DQN, additional computational cost of DA2Q is relatively low, as $\boldsymbol{\alpha}$ is updated every C steps, which only involves an inversion on a small matrix of the size $k \times k$. The k target values are computed in parallel in TensorFlow [38], which costs the same time as in DQN. Moreover, additional runtime can be ignored when compared with the costly backpropagation and interaction with the environment.

7 Related work

The most relevant related work is related to averaged Q-learning [26], which is used to average the most recent values. In fact, it is a special case of the proposed algorithm with regularization. Specifically, we add a regularization term to (4) and obtain the optimization problem $\boldsymbol{\alpha}^{(t)} = \arg\min_{\boldsymbol{\alpha}} \|B^{(t)}\boldsymbol{\alpha}\|_2^2 + \beta\|\boldsymbol{\alpha}\|_2^2$

where $\boldsymbol{\alpha} \in \{\boldsymbol{\alpha} \mid \boldsymbol{\alpha}^{\mathrm{T}}\mathbf{1} = 1\}$ and β is a coefficient. Similarly, using the Lagrange multiplier method, we can derive the solution $\boldsymbol{\alpha}^{(t)} = [(B^{(t)})^{\mathrm{T}}B^{(t)} + \beta\mathbf{I}]^{-1}\mathbf{1}/\mathbf{1}^{\mathrm{T}}[(B^{(t)})^{\mathrm{T}}B^{(t)} + \beta\mathbf{I}]^{-1}\mathbf{1}$. Moreover, when β tends to infinity, the value of $\boldsymbol{\alpha}^{(t)}$ goes to $\mathbf{1}/k$. This implies exactly the same update method of averaged Q-learning [26]. Therefore, the averaged Q-learning is considered as a special case of the proposed method. We compare it with the proposed method in the conducted experiments.

The present research work is focused on the reinforcement learning field. Moreover, we notice that there are several studies in the field of optimization related to the present study. Scieur et al. [23] proposed RMPE to extrapolate the parameters produced by an iterative gradient algorithm to achieve a faster convergence rate. Then, Scieur et al. [24] proposed RNA by extending RMPE to the stochastic case. It may deem that their updating rules are close to those of the proposed algorithm; however, A2VI has an entirely different spirit with regard to them. Both RMPE and RNA do not change the updating rules of the original algorithms; while A2VI utilizes the last k parameters to update the new one, RMPE and RNA extrapolate the whole parameter history and thereby require time-consuming computation on a large matrix. The nonlinear term in RMPE and RNA is induced by high-order noise, which can be neglected in the convergence analysis. However, nonlinearity in A2VI is induced by the non-smooth operator max on a set of non-negligible terms, which means that we cannot rely to their analysis in the linear case. RMPE and RNA correspond to the general optimization problems, while the proposed method is focused on the reinforcement learning problems.

Zhang et al. [29] also conducted the convergence analysis on the AA method. They focused on the convergence analysis on the type-I AA, while the proposed approach is more related to the type-II AA. Moreover, the proposed algorithm is not a straightforward application of AA to VI. AA employs the updating rule $v^{(t)} = \sum_i \alpha_i \Gamma(v^{(t-i)})$, while A2VI implies including the weighted sum into the fixed-point iteration $v^{(t)} = \Gamma(\sum_i \alpha_i v^{(t-i)})$. In this way, the proposed approach has the two advantages: (1) obtaining geometric interpretation of the algorithm; (2) making the convergence analysis tractable. Furthermore, Zhang et al. [29] applied the type-I AA to the general non-smooth optimization problem. They provided the convergence analysis as the iteration step n goes to infinity, but without the convergence rate analysis. In the present study, we conduct the local linear convergence rate analysis without the rejection step in Theorem 1, as well as the global one with the rejection step in Theorems 2 and 3.

8 Conclusion

In the present paper, we have proposed the Anderson accelerated value iteration algorithm, which is a novel acceleration approach for reinforcement learning. We have demonstrated the convergence property of the proposed method under certain conditions. The proposed algorithm empirically achieves the superior performance with regard to the conducted synthetic experiments and several control tasks. Despite the success of the proposed algorithm, the convergence analysis for the general case is absent in the present study, and thereby is left for the future work.

Acknowledgements We would like to thank Chengzhuo NI, Guangzeng XIE, Wenhao YANG, Luo LUO, Zhihua ZHANG for their discussion in this work.

References

1 Sutton R S, Barto A G. Reinforcement Learning: An Introduction. Cambridge: MIT Press, 1998
2 Watkins C J, Dayan P. Q-learning. Mach Learn, 1992, 8: 279–292
3 Rummery G A, Niranjan M. On-line Q-learning using connectionist systems. Cambridge: University of Cambridge, Department of Engineering, 1994, 37: 20
4 Wiering M, Schmidhuber J. HQ-learning. Adaptive Behav, 1997, 6: 219–246
5 Chen C L, Dong D Y, Li H-X, et al. Hybrid MDP based integrated hierarchical Q-learning. Sci China Inf Sci, 2011, 54: 2279–2294
6 Sutton R S, McAllester D, Singh S, et al. Policy gradient methods for reinforcement learning with function approximation. In: Proceedings of Conference on Neural Information Processing Systems, 2000
7 Kaelbling L P, Littman M L, Moore A W. Reinforcement learning: a survey. J Artif Intell Res, 1996, 4: 237–285
8 Bellemare M G, Dabney W, Munos R. A distributional perspective on reinforcement learning. In: Proceedings of International Conference on Machine Learning, 2017
9 Schaul T, Quan J, Antonoglou I, et al. Prioritized experience replay. In: Proceedings of International Conference on Learning Representations, 2016
10 Van H H, Guez A, Silver D. Deep reinforcement learning with double q-learning. In: Proceedings of AAAI Conference on Artificial Intelligence, 2016
11 Wang Z Y, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning. In: Proceedings of International Conference on Machine Learning, 2015
12 Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with deep reinforcement learning. 2013. ArXiv:1312.5602

13 Li X X, Peng Z H, Liang L, et al. Policy iteration based Q-learning for linear nonzero-sum quadratic differential games. Sci China Inf Sci, 2019, 62: 052204
14 Yan X H, Zhu J H, Kuang M C, et al. Missile aerodynamic design using reinforcement learning and transfer learning. Sci China Inf Sci, 2018, 61: 119204
15 Dieleman S, Aaron V D O, Karen S. The challenge of realistic music generation: modelling raw audio at scale. In: Proceedings of Conference on Neural Information Processing Systems, 2018
16 Bellman R. A Markovian decision process. J Math Mech, 1957, 6: 679–684
17 Howard R A. Dynamic Programming and Markov Processes. Hoboken: John Wiley & Sons, 1964
18 Alla A, Falcone M, Kalise D. An efficient policy iteration algorithm for dynamic programming equations. SIAM J Sci Comput, 2015, 37: 181–200
19 Puterman M L. Markov Decision Processes: Discrete Stochastic Dynamic Programming. Hoboken: John Wiley & Sons, 2014
20 Laurini M, Micelli P, Consolini L, et al. A Jacobi-like acceleration for dynamic programming. In: Proceedings of Conference on Decision and Control, 2016. 7371–7376
21 Laurini M, Consolini L, Locatelli M. A consensus approach to dynamic programming. IFAC-PapersOnLine, 2017, 50: 8435–8440
22 Bubeck S. Convex optimization: algorithms and complexity. FNT Mach Learn, 2015, 8: 231–357
23 Scieur D, D'Aspremont A, Bach F. Regularized nonlinear acceleration. In: Proceedings of Conference on Neural Information Processing Systems, 2016
24 Scieur D, Bach F, D'Aspremont A. Nonlinear acceleration of stochastic algorithms. In: Proceedings of Conference on Neural Information Processing Systems, 2017
25 Xie G Z, Wang Y T, Zhou S C, et al. Interpolatron: interpolation or extrapolation schemes to accelerate optimization for deep neural networks. 2018. ArXiv: 1805.06753
26 Anschel O, Baram N, Shimkin N. Averaged-DQN: variance reduction and stabilization for deep reinforcement learning. In: Proceedings of International Conference on Machine Learning, 2017
27 Johnson R, Zhang T. Accelerating stochastic gradient descent using predictive variance reduction. In: Proceedings of Conference on Neural Information Processing Systems, 2013
28 Chen C Y, Wang W L, Zhang Y Z, et al. A convergence analysis for a class of practical variance-reduction stochastic gradient MCMC. Sci China Inf Sci, 2019, 62: 012101
29 Zhang J, O'Donoghue B, Boyd S. Globally convergent type-I anderson acceleration for non-smooth fixed-point iterations. 2018. ArXiv: 1808.03971
30 Anderson D G. Iterative procedures for nonlinear integral equations. J ACM, 1965, 12: 547–560
31 Walker H F, Ni P. Anderson acceleration for fixed-point iterations. SIAM J Numer Anal, 2011, 49: 1715–1735
32 Toth A, Kelley C T. Convergence analysis for anderson acceleration. SIAM J Numer Anal, 2015, 53: 805–819
33 Ortega J M, Rheinboldt W C. Iterative solution of nonlinear equations in several variables. 1970
34 Puterman M L, Brumelle S L. The analytic theory of policy iteration. In: Proceedings of Conference on Dynamic Programming and Its Applications, 1978. 91–113
35 Nesterov Y. Introductory lectures on convex programming volume I: Basic course. 1998
36 Brockman G, Cheung V, Pettersson L, et al. OpenAI Gym. 2016. ArXiv: 1606.01540
37 Kingma D P, Ba L J. Adam: a method for stochastic optimization. 2014. ArXiv: 1412.6980
38 Abadi M, Barham P, Chen J M, et al. Tensorflow: a system for large-scale machine learning. In: Proceedings of Conference on Operating Systems Design and Implementation, 2016. 265–283