

Smoothing Advantage Learning

Yaozhong Gan^{1,2}, Zhe Zhang^{1,2}, Xiaoyang Tan^{1,2}

¹College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics

²MIIT Key Laboratory of Pattern Analysis and Machine Intelligence

{yzgancn, zhangzhe, x.tan}@nuaa.edu.cn

Abstract

Advantage learning (AL) aims to improve the robustness of value-based reinforcement learning against estimation errors with action-gap-based regularization. Unfortunately, the method tends to be unstable in the case of function approximation. In this paper, we propose a simple variant of AL, named smoothing advantage learning (SAL), to alleviate this problem. The key to our method is to replace the original Bellman Optimal operator in AL with a smooth one so as to obtain more reliable estimation of the temporal difference target. We give a detailed account of the resulting action gap and the performance bound for approximate SAL. Further theoretical analysis reveals that the proposed value smoothing technique not only helps to stabilize the training procedure of AL by controlling the trade-off between convergence rate and the upper bound of the approximation errors, but is beneficial to increase the action gap between the optimal and sub-optimal action value as well.

1 Introduction

Learning an optimal policy in a given environment is a challenging task in high-dimensional discrete or continuous state spaces. A common approach is through approximate methods, such as using deep neural networks (Mnih et al. 2015). However, an inevitable phenomenon is that this could introduce approximation/estimation errors - actually, several studies suggest that this leads to sub-optimal solutions due to incorrect reinforcement (Thrun and Schwartz 1993; van Hasselt, Guez, and Silver 2016; Lan et al. 2020).

Advantage learning operator (AL) (Bellemare et al. 2016) is a recently proposed method to alleviate the negative effect of the approximation and estimation errors. It is a simple modification to the standard Bellman operator, by adding an extra term that encourages the learning algorithm to enlarge the action gap between the optimal and sub-optimal action value at each state. Theoretically, it can be shown that the AL operator is an optimality-preserving operator, and increasing the action gap not only helps to mitigate the undesirable effects of errors on the induced greedy policies, but may be able to achieve a fast convergence rate as well (Farahmand 2011).

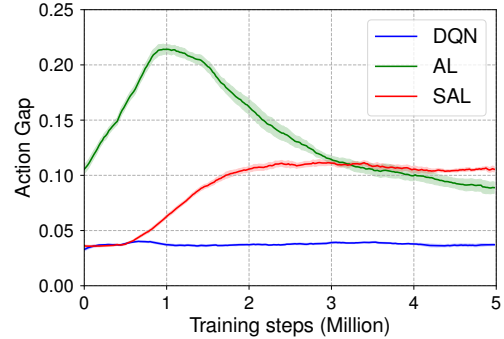


Figure 1: An illustration of trajectories of action gaps between the optimal and sub-optimal action at each training step in Asterix of MinAtar (Young and Tian 2019). It shows that compared to the baseline DQN method, the AL could suffer from the problem of over-enlarging the action gaps at the early stage of the training procedure, while such behavior is well controlled by the proposed SAL method.

Due to these advantages, the AL method has drawn increasing attention recently and several variants appear in literatures. For example, generalized value iteration (G-VI) (Kozuno, Uchibe, and Doya 2017) is proposed to alleviate the overestimation of AL, while Munchausen value iteration (M-VI) (Vieillard, Pietquin, and Geist 2020) adds a scaled log-policy to the immediate reward under the entropy regularization framework. Both methods impose extra constraints in the policy space, such as entropy regularization and Kullback-Leibler (KL) divergence, and technically, both can be seen as a soft Q operator plus a soft advantage term.

However, one problem less studied in literature about the AL method is that it could suffer from the problem of over-enlarging the action gaps at the early stage of the training procedure. From Figure 1, one can see that the AL method faces this problem. Although the action gaps are rectified at a later stage, this has a very negative impact on the performance of the algorithm. We call that this issue is an incorrect action gaps phenomenon. Naively using the AL operator tends to be both aggressive and risky. Thus, the difficulty of learning may be increased. Recall that the AL method includes two terms, i.e., the temporal-difference (TD) tar-

get estimation and an advantage learning term. From the perspective of function approximation, both terms critically rely on the performance of the underlying Q neural network, which predicts the action value for any action taken at a given state. The method becomes problematic when the optimal action induced by the approximated value function does not agree with the true optimal action. This may significantly increase the risk of incorrect action gap values.

Based on these observations, we propose a new method, named SAL (smoothing advantage learning), to alleviate this problem. Our key idea is to use the value smoothing techniques (Lillicrap et al. 2016) to improve the reliability and stability of the AL operator. For such purpose, we incorporate the knowledge about the current action value, estimated using the same target network. Our SAL works in the value space, which is consistent with the original motivation of the AL operator. Specifically, we give a detailed account of the resulting action gap and the performance bound for approximate SAL. Further theoretical analysis reveals that the proposed value smoothing technique helps to stabilize the training procedure and increase the action gap between the optimal and sub-optimal action value, by controlling the trade-off between convergence rate and the upper bound of the approximation errors. We also analyze how the parameters affect the action gap value, and show that the coefficient α of advantage term can be expanded to $(0, \frac{2}{1+\gamma})$. This expands the original theory (Bellemare et al. 2016). Finally, we verify the feasibility and effectiveness of the proposed method on several publicly available benchmarks with competitive performance.

2 Background

2.1 Notation and Basic Definition

The reinforcement learning problem can be modeled as a Markov Decision Processes (MDP) which described by a tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$. \mathcal{S} and \mathcal{A} are the finite state space and finite action space, respectively. The function $P(s'|s, a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ outputs the transition probability from state s to state s' under action a . The reward on each transition is given by the function $R(s, a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, whose maximum absolute value is R_{\max} . $\gamma \in [0, 1)$ is the discount factor for long-horizon returns. The goal is to learn a policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ (satisfies $\sum_a \pi(a|s) = 1$) for interacting with the environment. Under a given policy π , the action-value function is defined as

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}:\infty, a_{t+1}:\infty} [G_t | s_t, a_t]$$

where $G_t = \sum_{i=0}^{\infty} \gamma^i R_{t+i}$ is the discount return. The state value function and advantage function are denoted by $V^\pi(s) = \mathbb{E}_\pi[Q^\pi(s, a)]$ and $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$, respectively. An optimal action state value function is $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ and optimal state value function is $V^*(s) = \max_a Q^*(s, a)$. The optimal advantage function is defined as $A^*(s, a) = Q^*(s, a) - V^*(s)$. Note that $Q^*(s, a)$ and $V^*(s)$ are bounded by $V_{\max} := \frac{1}{1-\gamma} R_{\max}$. The corresponding optimal policy is defined as $\pi^*(a|s) = \arg \max_a Q^*(s, a)$, which selects the highest-value action in each state. For brevity, let $\|f\| = \max_x |f(x)|$ denote

the L_∞ norm. And for functions f and g with a domain \mathcal{X} , $f \geq (>)g$ mean $f(x) \geq (>)g(x)$ for any $x \in \mathcal{X}$.

2.2 The Bellman operator

The Bellman operator \mathcal{T}^π is defined as,

$$\mathcal{T}^\pi Q(s, a) = R(s, a) + \gamma \sum_{s', a'} P(s'|s, a) \pi(a'|s') Q(s', a') \quad (1)$$

It is known that \mathcal{T}^π is a contraction (Sutton and Barto 1998) - that is, $Q^\pi(s, a) := \lim_{k \rightarrow \infty} (\mathcal{T}^\pi)^k Q_0(s, a)$ is a unique fixed point. And, the optimal Bellman operator \mathcal{T} is defined as,

$$\begin{aligned} \mathcal{T}Q(s, a) &= \max_\pi \mathcal{T}^\pi Q(s, a) \\ &= R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q(s', a') \end{aligned} \quad (2)$$

It is well known (Feinberg 1996; Bellman and Dreyfus 1959; Bellman 1958) that the optimal Q function $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ is the unique solution Q^* to the optimal Bellman equation and satisfies

$$\begin{aligned} Q^*(s, a) &= \mathcal{T}Q^*(s, a) \\ &= R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q^*(s', a') \end{aligned}$$

Q -learning is a method proposed by Watkins (1989) that estimates the optimal value function Q^* through value iteration. Due to the simplicity of the algorithm, it has become one of the most popular reinforcement learning algorithms.

2.3 The Advantage Learning operator and its Modifications

The advantage learning (AL) operator was proposed by Baird (1999) and further studied by Bellemare et al. (2016). Its operator can be defined as:

$$\mathcal{T}_{AL}Q(s, a) = \mathcal{T}Q(s, a) + \alpha(Q(s, a) - V(s)) \quad (3)$$

where $V(s) = \max_a Q(s, a)$, \mathcal{T} is the optimal Bellman operator Eq.(2) and $\alpha \in [0, 1)$. And its operator has very nice properties.

Definition 2.1. (Optimality-preserving) (Bellemare et al. 2016) An operator \mathcal{T}' is Optimality-preserving, if for arbitrary initial function $Q_0(s, a)$, $s \in \mathcal{S}$, $a \in \mathcal{A}$, letting $Q_{k+1}(s, a) := \mathcal{T}'Q_k(s, a)$,

$$\hat{V}(s) := \lim_{k \rightarrow \infty} \max_a Q_k(s, a)$$

exists, is unique, $\hat{V}(s) = V^*(s)$, and for all $a \in \mathcal{A}$,

$$Q^*(s, a) < V^*(s) \implies \limsup_{k \rightarrow \infty} Q_k(s, a) < V^*(s). \quad (4)$$

Definition 2.2. (Gap-increasing) (Bellemare et al. 2016) An operator \mathcal{T}' is gap-increasing, if for arbitrary initial function $Q_0(s, a)$, $s \in \mathcal{S}$, $a \in \mathcal{A}$, letting $Q_{k+1}(s, a) = \mathcal{T}'Q_k(s, a)$, and $V_k(s) = \max_a Q_k(s, a)$, satisfy

$$\liminf_{k \rightarrow \infty} [V_k(s) - Q_k(s, a)] \geq V^*(s) - Q^*(s, a). \quad (5)$$

They proved that when there is no approximation error, the state value function $V_k(s)$, $V_k(s) = \max_a Q_k(s, a)$ during k -th Q -iteration with \mathcal{T}_{AL} , is convergent and equal to the optimal state value function $V^*(s)$ - this operator is optimal-preserving operator. And this AL operator can enhance the difference between Q -values for an optimal action and sub-optimal actions - this operator is gap-increasing operator, to make estimated Q -value functions less susceptible to function approximation/estimation error.

Recently, Kozuno et al. (Kozuno, Uchibe, and Doya 2017) proposed generalized value iteration (G-VI), a modification to the AL:

$$\mathcal{T}_{G,\tau}Q(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) m_\tau Q(s', \cdot) + \alpha(Q(s, a) - m_\tau Q(s, \cdot))$$

where $m_\tau Q(s, \cdot) := \tau \ln \langle \frac{1}{|\mathcal{A}|}, \exp \frac{Q}{\tau} \rangle = \tau \ln \sum_a \frac{1}{|\mathcal{A}|} \exp \frac{Q(s, a)}{\tau}$ is the mellowmax operator, $\alpha \in [0, 1)$.

This is very close to munchausen value iteration¹ (M-VI) (Vieillard, Pietquin, and Geist 2020). Both algorithms can be derived from using entropy regularization in addition to Kullback-Leibler (KL) divergence. In fact, they can be seen as soft Q operator plus a soft advantage term. But the hyperparameters τ controls the asymptotic performance - that is, the performance gap between the optimal action-value function Q^* and Q^{π_k} , the control policy π_k at iteration k , is related to τ , so both algorithms may be sensitive to the hyperparameters τ (Kim et al. 2019; Gan, Zhang, and Tan 2021). When $\tau \rightarrow 0$, it retrieves advantage learning. And taking $\alpha = 1$, it retrieves dynamic policy programming (DPP) (Azar, Gómez, and Kappen 2012). In addition, a variant of AL for improved exploration was studied in (Ferret, Pietquin, and Geist 2021). In Appendix, we show that G-VI and M-VI are equivalent. Meanwhile, we also give the relationship between their convergence point and AL.

3 Methods

We discussed that an incorrect action gap maybe affect the performance of the algorithm from the Figure 1. The possible reason is that the estimated Q function obtained by the approximated value function is greatly affected by errors. In order to alleviate it, we propose a new method, inspired by the value smoothing technique (Lillicrap et al. 2016). Next, we first introduce our new operator, named the smoothing advantage learning operator.

3.1 The Smoothing Advantage Learning operator

We first provide the following optimal smooth Bellman operator (Fu et al. 2019; Smirnova and Dohmatob 2020) which is used to analyze the smoothing technique:

$$\mathcal{T}_\omega Q(s, a) = (1 - \omega)Q(s, a) + \omega \mathcal{T}Q(s, a) \quad (6)$$

where $\omega \in (0, 1)$, and \mathcal{T} is the optimal Bellman operator. They proved that the unique fixed point of the optimal smooth Bellman operator is same as the optimal Bell-

man operator. In fact, from the operator aspect, the coefficient ω can be expanded to $(0, \frac{2}{1+\gamma})$ (See Appendix for detailed derivation). Note that if $0 < \omega < 1$, the uncertainty caused by maximum operator is reduced, due to the introduction of the action-value function approximation $Q(s, a)$. And the \mathcal{T}_ω can be thought of as a conservative operator. While $1 < \omega < \frac{2}{1+\gamma}$, the uncertainty increases, and overemphasizes the bootstrapping is both aggressive and risky in estimating TD learning.

We propose a smoothing advantage learning operator (SAL), defined as:

$$\mathcal{T}_{SAL}Q(s, a) = \mathcal{T}_\omega Q(s, a) + \alpha[Q(s, a) - V(s)] \quad (7)$$

where $V(s) = \max_a Q(s, a)$, \mathcal{T}_ω is the smooth Bellman operator Eq.(6) and $\alpha \in \mathbb{R}$.

Note that if $\omega = 1$, the \mathcal{T}_{SAL} operator is reduced to the advantage learning operator \mathcal{T}_{AL} Eq.(3). And when $\alpha = 0$, the \mathcal{T}_{SAL} operator is reduced to the optimal smooth Bellman operator \mathcal{T}_ω Eq.(6). In practice, the TD target of the SAL operator is calculated by a estimation - the current state estimation, the bootstrapping estimate of the next state for future return and an advantage term. Considering the fragility of the prediction of neural network-based value function approximators especially when the agent steps in an unfamiliar region of the state space, the accuracy of the estimated reinforcement signal may directly affect the performance of the algorithm. By doing this, we show that it helps to enhance the network's ability to resist environmental noise or estimation errors by slow update/iteration. It can make the training more stable (Remark 3.3) and increase the action gap Eq.(11).

3.2 Convergence of SAL

In this section, let's prove that the Q_k , during k -th Q -iterations with \mathcal{T}_{SAL} , converges to a unique fixed point, and our algorithm can increase the larger action gap value Eq.(11) compared with AL algorithm, when there is no approximation error.

Definition 3.1. (All-preserving) An operator T' is all-preserving, if for arbitrary initial function $Q^0(s, a)$, letting $Q^{k+1}(s, a) := T'Q^k(s, a)$,

$$\hat{Q}^*(s, a) \triangleq \lim_{k \rightarrow \infty} Q^k(s, a)$$

exists, for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, satisfy

$$\max_a \hat{Q}^*(s, a) = \max_a Q^*(s, a) \quad (8)$$

and

$$Q^*(s, a_{[1]}) \geq Q^*(s, a_{[2]}) \geq \dots \geq Q^*(s, a_{[n]}) \Rightarrow \quad (9)$$

$$\hat{Q}^*(s, a_{[1]}) \geq \hat{Q}^*(s, a_{[2]}) \geq \dots \geq \hat{Q}^*(s, a_{[n]})$$

where $Q^*(s, a)$ is a stable point with the optimal Bellman operator \mathcal{T} , and $[i]$ donate i -th largest index of action a among $Q^*(s, a)$.

¹In M-VI, $\langle \frac{1}{|\mathcal{A}|}, \exp \frac{Q}{\tau} \rangle$ is replaced by $\langle 1, \exp \frac{Q}{\tau} \rangle$

Thus under an all-preserving operator, all optimal actions remain optimal, and the order of good and bad actions of \hat{Q}^* remains the same as Q^* . Compared with the definition of optimality-preserving operator 2.1, the definition 3.1 is more restrictive. Because the operator \mathcal{T}' is an all-preserving operator, the \mathcal{T}' must be an optimality-preserving operator.

From Eq. (7), we have

$$\begin{aligned} Q_{k+1}(s, a) &:= \mathcal{T}_{SAL} Q_k(s, a) \\ &= \omega Q_k(s, a) + \alpha [Q_k(s, a) - V_k(s)] \\ &= \omega A_k \mathcal{T} \hat{B}_k(s, a) - \alpha A_k B_k(s) + \lambda^k Q_0(s, a) \end{aligned}$$

where $A_k \hat{B}_k(s, a) = Q_{k-1}(s, a) + \lambda V_{k-2}(s) + \lambda^2 V_{k-3}(s) + \dots + \lambda^{k-1} V_0(s)$, $A_k = \frac{1-\lambda^k}{1-\lambda} = 1 + \lambda + \lambda^2 + \dots + \lambda^{k-1}$ is the weighted regular term, $\lambda = 1 - \omega + \alpha$, $B_k(s) = \max_{a_{k-1}} \hat{B}_k(s, a)$, and $V_i(s) = \max_a Q_i(s, a)$ is i -th Q -iterations, $i = 1, 2, \dots, k-1$. For simplicity, we assume $Q_0(s, a) = 0$. See Appendix for detailed derivation.

Now let's prove that the smoothing advantage learning operator \mathcal{T}_{SAL} is both all-preserving and gap-increasing.

Theorem 3.1. *Let \mathcal{T} and \mathcal{T}_{SAL} respectively be the optimal Bellman operator and the smoothing advantage learning operator defined by Eq.(2) and Eq.(7). Letting $Q^{k+1}(s, a) = \mathcal{T}_{SAL} Q^k(s, a)$, and Q^* is a stable point during Q -iteration with \mathcal{T} , and $V^*(s) = \max_a Q^*(s, a)$. If $0 \leq \alpha < \omega < \frac{2}{1+\gamma}$, then $\lim_{k \rightarrow \infty} B_k(s) = V^*(s)$ and*

$$\begin{aligned} \hat{Q}^*(s, a) &\triangleq \lim_{k \rightarrow \infty} Q^k(s, a) \\ &= \frac{1}{\omega - \alpha} [\omega Q^*(s, a) - \alpha V^*(s)] \end{aligned}$$

Furthermore, we also have the set of $\arg \max_a Q^*(s, a)$ is equal to set $\arg \max_a \hat{Q}^*(s, a)$, and the operator \mathcal{T}_{SAL} is all-preserving.

The proof of the theorem is given in Appendix.

This result shows the relationship between the stable point $\hat{Q}^*(s, a)$ and the optimal point $Q^*(s, a)$. The $\hat{Q}^*(s, a)$ and $Q^*(s, a)$ not only have the same maximum values, but also have the same order of good and bad actions. If $\omega = 1$, the \mathcal{T}_{SAL} operator is reduced to the advantage learning operator \mathcal{T}_{AL} . Hence the above analysis applies to the \mathcal{T}_{AL} operator as well - that is, the advantage learning operator \mathcal{T}_{AL} is all-preserving. More importantly, it's possible that α is greater than 1. This not only generalizes the properties of the advantage learning operator \mathcal{T}_{AL} (Bellemare et al. 2016) (where it is concluded that $\hat{V}^*(s) \triangleq \max_a \hat{Q}^*(s, a) = V^*(s)$ and satisfy $0 \leq \alpha < 1$), but also implies that all optimal actions of $Q^*(s, a)$ remain optimal through the operator \mathcal{T}_{AL} instead of at least ones (Bellemare et al. 2016). This later advantage may be the main reason for the good performance of \mathcal{T}_{AL} (shown in the experimental section).

Next, we will give the relationship between the operator \mathcal{T}_{SAL} and \mathcal{T} in term of action-gap value.

Theorem 3.2. *Let \mathcal{T} and \mathcal{T}_{SAL} respectively be the optimal Bellman operator and the smoothing advantage learning operator defined by Eq.(2) and Eq.(7). If $0 \leq \alpha < \omega < \frac{2}{1+\gamma}$,*

letting $\hat{Q}^(s, a)$ is a stable point during Q -iteration with \mathcal{T}_{SAL} , and Q^* is a stable point with \mathcal{T} , and $V^*(s) = \max_a Q^*(s, a)$. For $\forall s \in \mathcal{S}, a \in \mathcal{A}$, then we have*

$$Gap(\mathcal{T}_{SAL}; s, a) = \frac{\omega}{\omega - \alpha} Gap(\mathcal{T}; s, a) \quad (10)$$

where $Gap(\mathcal{T}_{SAL}; s, a) = V^*(s) - \hat{Q}^*(s, a)$ denote the \mathcal{T}_{SAL} operator's action gap, and $Gap(\mathcal{T}; s, a) = V^*(s) - Q^*(s, a)$ denote the \mathcal{T} operator's action gap. Furthermore, we have

- 1) the operator \mathcal{T}_{SAL} is gap-increasing;
- 2) if α is fixed, the action gap $Gap(\mathcal{T}_{SAL}; s, a)$ monotonically decreases w.r.t $\omega \in (\alpha, \frac{2}{1+\gamma})$;
- 3) if ω is fixed, the action gap $Gap(\mathcal{T}_{SAL}; s, a)$ monotonically increases w.r.t $\alpha \in [0, \omega)$.

The proof of the theorem is given in Appendix.

This result shows that the action gap $Gap(\mathcal{T}_{SAL}; s, a)$ is a multiple of the $Gap(\mathcal{T}; s, a)$. It also reveals how hyper-parameters of ω and α affect the action gap. Furthermore, in the case of $\omega = 1$, the action gap $Gap(\mathcal{T}_{SAL}; s, a)$ is equal to the $Gap(\mathcal{T}_{AL}; s, a)$. From Theorem 3.2, if $0 \leq \alpha < \omega \leq 1$, we can conclude that

$$Gap(\mathcal{T}_{SAL}; s, a) \geq Gap(\mathcal{T}_{AL}; s, a) \geq Gap(\mathcal{T}; s, a). \quad (11)$$

In other words, the operator SAL helps to enlarge the gap between the optimal action-value and the sub-optimal action values. This is beneficial to improve the robustness of the estimated Q value against environmental noise or estimation errors.

3.3 Performance Bound for Approximate SAL

In the previous section, we discussed the convergence of the SAL in the absence of approximation error. In this section, we prove that our algorithm can achieve more stable training compared with AL algorithm by error propagation (Remark 3.3). Now, we derive the performance bound for approximate SAL, defined by

$$\begin{aligned} Q_{k+1}(s, a) &:= (1 - \omega) Q_k(s, a) + \omega [\mathcal{T} Q_k(s, a) + \epsilon_k] \\ &\quad + \alpha [Q_k(s, a) - V_k(s)] \end{aligned} \quad (12)$$

where $V_k(s) = \max_a Q_k(s, a)$, \mathcal{T} is the optimal Bellman operator Eq.(2), $0 \leq \alpha < \omega < \frac{2}{1+\gamma}$, ϵ_k is an approximation error at k -iteration. In general, when calculating the $\mathcal{T} Q_k$, error ϵ_k is inevitable, because (1) we do not have direct access to the optimal Bellman operator, but only some samples from it, and (2) the function space in which Q belongs is not representative enough. Thus there would be an approximation error $\mathcal{T} Q_k(s, a) + \epsilon_k$ between the result of the exact value iteration (VI) and approximate VI (AVI) (Munos 2007; Azar, Gómez, and Kappen 2012; Kozuno, Uchibe, and Doya 2017; Vieillard, Pietquin, and Geist 2020; Smirnova and Dohmatob 2020). Just for simplicity, we assume that $Q_0(s, a) = 0$ throughout this section.

Theorem 3.3. (Error propagation) *Consider the approximate SAL algorithm defined by Eq.(12), π_k is a policy greedy*

w.r.t. $Q_k(s, a)$, and Q^{π_k} is the unique fixed point of the Bellman operator \mathcal{T}^{π_k} . If $0 \leq \alpha < \omega < \frac{2}{1+\gamma}$, then, we have

$$\begin{aligned} \|Q^* - Q^{\pi_k}\| &\leq \frac{2\gamma}{A_{k+1}(1-\gamma)} \sum_{i=0}^k \xi^i \lambda^{k-i} V_{\max} \\ &\quad + \frac{2\gamma\omega}{A_{k+1}(1-\gamma)} \sum_{i=0}^{k-1} \xi^i \left\| \sum_{j=0}^{k-1-i} \lambda^{k-1-i-j} \epsilon_j \right\| \end{aligned}$$

where $V_{\max} = \frac{1}{1-\gamma} R_{\max}$, $A_k = \frac{1-\lambda^k}{1-\lambda}$ is the weighted regular term, $\lambda = 1 - \omega + \alpha$, $\xi = |1 - \omega| + \omega\gamma$.

The proof of the theorem is given in Appendix.

Corollary 3.1. (Kozuno, Uchibe, and Doya 2017) For approximate AL, when $\omega = 1$, if $0 \leq \alpha < 1$, define $V_{\max} = \frac{1}{1-\gamma} R_{\max}$ and $A_k = \frac{1-\alpha^k}{1-\alpha}$, we have

$$\begin{aligned} \|Q^* - Q^{\pi_k}\| &\leq \frac{2\gamma}{A_{k+1}(1-\gamma)} \sum_{i=0}^k \gamma^i \alpha^{k-i} V_{\max} \\ &\quad + \frac{2\gamma}{A_{k+1}(1-\gamma)} \sum_{i=0}^{k-1} \gamma^i \left\| \sum_{j=0}^{k-1-i} \alpha^{k-1-i-j} \epsilon_j \right\| \end{aligned}$$

Remark 3.1. When $\alpha = 0$ and $\omega = 1$, we have

$$\|Q^* - Q^{\pi_k}\| \leq \frac{2\gamma^{k+1}}{1-\gamma} V_{\max} + \frac{2\gamma}{1-\gamma} \sum_{i=0}^{k-1} \gamma^i \|\epsilon_{k-1-i}\|$$

The conclusion is consistent with approximate modified policy iteration (AMPI) (Scherrer et al. 2015).

Remark 3.2. From the first term on the right of theorem 3.3, by the mean value theorem, there exist θ between ξ and λ , it has $\sum_{i=0}^k \xi^i \lambda^{k-i} = (k+1)\theta^k$, and exist $\hat{\theta}$ between α and γ , it has $\sum_{i=0}^k \gamma^i \alpha^{k-i} = (k+1)\hat{\theta}^k$. Since

$$\sum_{i=0}^k \xi^i \lambda^{k-i} \geq \sum_{i=0}^k \gamma^i \alpha^{k-i} \geq \gamma^k > 0,$$

we have $1 > (k+1)\theta^k \geq (k+1)\hat{\theta}^k \geq \gamma^k > 0$. That is, the convergence rate of approximate SAL is much slower than the approximate AL and approximate VI. From formula Eq.(11), we know that the slower convergence rate can increase more action-gap compared with AL.

Remark 3.3. From the second term on the right of theorem 3.3, assume error terms ϵ_k satisfy for all k , $\|\epsilon_k\| \leq \epsilon$ for some $\epsilon \geq 0$, if $0 \leq \alpha < \omega < 1$, we ignore $\frac{2\gamma}{1-\gamma}$, defined

$$\begin{aligned} SAL(\epsilon) &= \omega \frac{1-\lambda}{1-\lambda^{k+1}} \sum_{i=0}^k \xi^i \sum_{j=0}^{k-1-i} \lambda^{k-1-i-j} \epsilon_j, \\ AL(\epsilon) &= \frac{1-\alpha}{1-\alpha^{k+1}} \sum_{i=0}^k \gamma^i \sum_{j=0}^{k-1-i} \alpha^{k-1-i-j} \epsilon_j, \end{aligned}$$

we have $SAL(\epsilon) \leq AL(\epsilon)$. In other words, it effectively reduces the supremum of approximate error compared with approximate AL. In a sense, it may make the training procedure more stable, if there exist approximate error.

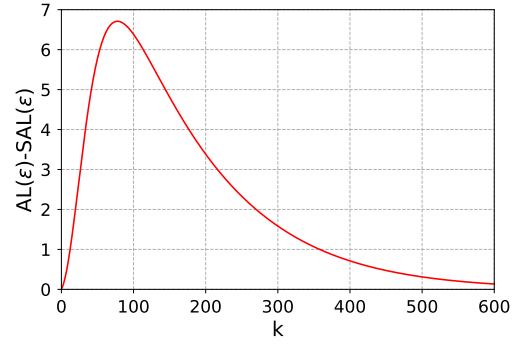


Figure 2: Numerical values of the error $AL(\epsilon) - SAL(\epsilon)$ with $\gamma = 0.99$, $\alpha = 0.9$ and $\omega = 0.95$, assuming $\epsilon = 1$.

From the above analysis, we know that the performance of the algorithm is bounded by the convergence rate term and the error term. These two terms have a direct effect on the behavior of our algorithm. Since the convergence rate of approximate SAL is much slower than the approximate AL and approximate VI, the upper error bound of the performance is very lower compared with approximate AL. This is beneficial to alleviate incorrect action gap values. From the Figure 2, we show the difference between $AL(\epsilon)$ and $SAL(\epsilon)$. It's pretty straightforward to see that $SAL(\epsilon)$ is very different from $AL(\epsilon)$ in the early stages. This result shows that this is consistent with our motivation. The estimated Q function obtained by the approximate value function may be more accurate compared with approximate AL. In the next section, we see that our method can effectively alleviate incorrect action gap values.

4 Experiment

In this section, we present our experimental results conducted over six games (Lunarlander; Asterix, Breakout, Space Invaders, Seaquest, Freeway) from Gym (Brockman et al. 2016) and MinAtar (Young and Tian 2019). In addition, we also run some experiments on Atari games in Appendix.

4.1 Implementation and Settings

To verify the effectiveness of the smoothing advantage learning (SAL), we replace the original optimal Bellman operator used in AL (Bellemare et al. 2016) with the optimal smooth Bellman operator. Algorithm 1 gives the detailed implementation pipeline in Appendix. Basically, the only difference we made over the original algorithm is that we construct a new TD target for the algorithm at each iteration, which implements an empirical version of our SAL operator with the target network, and all the remaining is kept unchanged. We conduct all the experiments mainly based on (Lan et al. 2020). The test procedures are averaged over 10 test episodes every 5000 steps across 5 independent runs. Particularly, we choose α from the set of $\{0.2, 0.3, 0.5, 0.9\}$ for AL (Bellemare et al. 2016). For SAL, we choose ω and α among $\{0.2, 0.3, 0.5, 0.9\}$, but the hyperparameters satisfy $\alpha < \omega$. For Munchausen-DQN (M-DQN) (Vieillard,

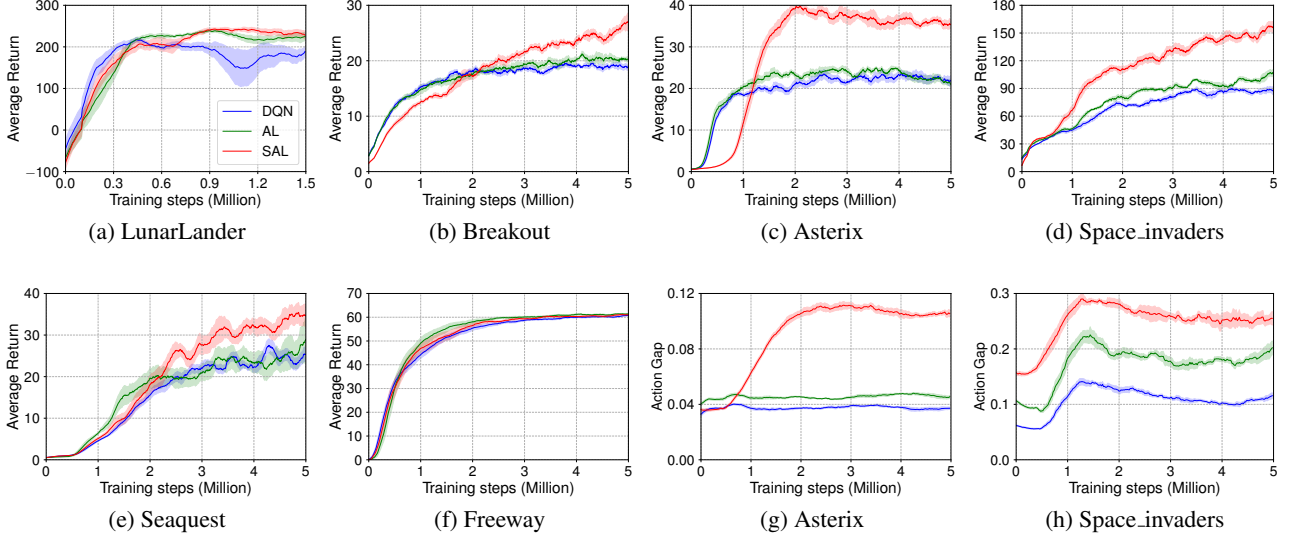


Figure 3: Learning curves on the Gym and MinAtar environments. Performance of SAL vs. AL and DQN (a-f), and the action gap of SAL and AL are evaluated (g-h).

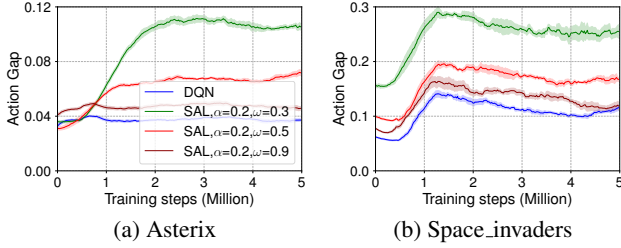


Figure 4: The action gaps of SAL are evaluated under the condition of α fixed.

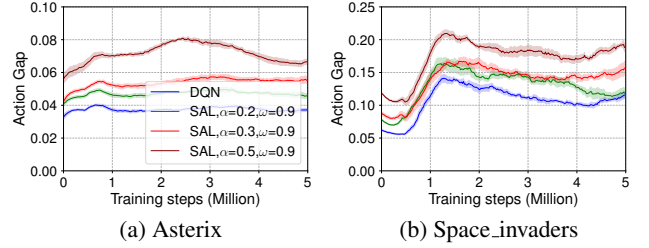


Figure 5: The action gaps of SAL are evaluated under the condition of ω fixed.

Pietquin, and Geist 2020), we fix $\tau = 0.03$, choose α from the set of $\{0.2, 0.3, 0.5, 0.9\}$. Since G-VI is equivalent to M-VI (M-DQN is a deep version of M-VI) (see the theorem 2.1 in Appendix), we don't conduct this experiment for G-VI. The number of target networks N is chosen from $\{2, 3, 5, 9\}$ for Average DQN (A-DQN) (Anschel, Baram, and Shimkin 2017). For a fair comparison, we optimize hyperparameter settings for all the compared methods on each environment before reporting the results. Please see Appendix for more details about experimental settings.

4.2 Evaluation

Firstly, we evaluate the proposed smoothing advantage learning algorithm (SAL) (see Algorithm 1 in Appendix) with a series of comparative experiments (SAL vs. AL (Bellemare et al. 2016)) on the Gym and MinAtar environments. As before, we evaluate the average of 10 test episodes every 5000 steps across 5 independent runs. And we plot the corresponding mean and 95% confidence interval in all figures.

Figure 3 gives the results of the SAL algorithm. We observed from the figure that compared to DQN, the original AL seems to not improve the performance significantly. And compared to DQN and the original AL, the SAL learns slower at the early stage, but as the learning goes on, our algorithm accelerates and finally outperforms the compared ones at most of the time. This provides us some interesting empirical evidence to support our conjecture that slow update at the early stage of the learning procedure is beneficial.

To further investigate the reason behind this, we evaluate the action gap between the optimal and sub-optimal Q values, by sampling some batches of state-action pairs from the replay buffer, and then calculate the averaged action gap values as the estimate. The figure (g) and (h) of Figure 3 gives the results. It shows that the action gap of SAL is two to three times as much as that of DQN, and is much higher than that of AL as well. From theorem 3.3, we know that the performance of the algorithm is bounded by the convergence speed term and the error term. These two terms have a

Algorithm	DQN	DDQN	A-DQN	AL	M-DQN	SAL
LunarLander	189.11 (27.18)	179.12 (32.32)	203.01 (46.21)	224.67 (26.06)	220.31 (4.98)	234.33 (18.83)
Asterix	22.16 (2.97)	20.29 (2.65)	21.70 (2.19)	21.09 (1.18)	25.78 (2.01)	35.43 (2.92)
Breakout	18.85 (0.85)	16.17 (1.02)	17.19 (0.39)	20.25 (0.95)	28.71 (2.42)	26.95 (3.76)
Space_invaders	87.41 (9.94)	83.81 (4.54)	89.54 (7.45)	106.97 (7.93)	110.25 (9.22)	155.61 (13.86)
Seaquest	25.31 (5.88)	20.50 (5.55)	25.54 (7.01)	28.29 (6.68)	41.23 (3.91)	34.70 (9.42)
Freeway	60.79 (0.65)	59.22 (0.67)	59.83 (0.76)	61.33 (0.45)	61.54 (0.18)	61.14 (0.43)

Table 1: Mean of average return for different methods in LunarLander, Asterix (Asterix-MinAtar), Breakout (Breakout-MinAtar), Space_invaders (Space_invaders-MinAtar), Seaquest (Seaquest-MinAtar) and Freeway (Freeway-MinAtar) games (standard deviation in parenthesis).

direct effect on the behavior of our algorithm. In particular, at the early stage of learning, since the accumulated error is very small, the performance of the algorithm is mainly dominated by convergence speed term. However, since the convergence rate of approximate SAL is much slower than the approximate AL and approximate VI (see remark 3.2), we can see that the mixing update slows down the learning in the beginning. After a certain number of iterations, the convergence speed terms of all three algorithms (SAL, AL, and VI) become very small (as Remark 3.2 indicates, this term decreases exponentially with the number of iterations). Consequently, the performance of the algorithm is mainly determined by the second term, i.e., the accumulated error term. This explains why at the later stage, learning stabilizes and leads to higher performance. In fact, by comparing (c) and (g) of Figure 3 or comparing (d) and (h) of Figure 3, one can see that the performance acceleration stage consistently corresponds to the increasing the action gap stage. Intuitively, increasing the action gap which increase the gap between the optimal value and the sub-optimal value help to improve the algorithm’s performance, as this helps to enhance the network’s ability to resist error/noise and uncertainty. This verifies our theoretical analysis.

In addition, Figure 1 and Figure 3 aren’t contradictory. Figure 1 shows the optimal parameter $\alpha = 0.9$ in the original paper of AL. In Figure 3, we search for the best parameter separately on each environment for all algorithms among the candidates. We also try to adjust the α value of AL separately to increase the action gap in Appendix, but we find that the performance of the AL algorithm doesn’t improve only by increasing the action gap value. We find that the action gap values are incorrectly increased at the early stage of learning. It is no benefit, as the target network tends to be unreliable in predicting the future return. And we also see that incorrect action gap value is rectified at a later stage. Thus, the difficulty of learning is increased, and the performance

of the algorithm is reduced. At the same time, our method also tries to get a larger action gap value by adjusting the parameters ω and α in Appendix. One can see that our method can adjust the parameters ω and α to steadily increase the action gap value. By comparing these two methods, we know that our proposed method can effectively alleviate the bad factors by errors, thus can learn more quickly.

Secondly, we analyze the influence of ω and α on the action gap values in SAL. Figure 4 and Figure 5 shows the results. It can be seen that for a fixed α value, the action gap value is decreasing monotonically w.r.t. ω , while for a fixed ω , the action gap value is increasing monotonically w.r.t. α . This is consistent with our previous theoretical analyses. In practice, we would suggest setting a higher value for α and a lower value for ω , so as to improve our robustness against the risk of choosing the wrong optimal action.

Finally, Table 1 gives the mean and standard deviation of average returns for all algorithms across the six games at the end of training. According to these quantitative results, one can see that the performance of our proposed method is competitive with other methods.

5 Conclusion

In this work, we propose a new method, called smoothing advantage learning (SAL). Theoretically, by analyzing the convergence of the SAL, we quantify the action gap value between the optimal and sub-optimal action values, and show that our method can lead to a larger action gap than the vanilla AL. By controlling the trade-off between convergence rate and the upper bound of the approximation errors, the proposed method helps to stabilize the training procedure. Finally, extensive empirical performance shows our algorithm is competitive with the current state-of-the-art algorithm, M-DQN (Vieillard, Pietquin, and Geist 2020) on several benchmark environments.

Acknowledgements

This work is partially supported by National Science Foundation of China (61976115, 61732006), AI+ Project of NUAA (NZ2020012), and research project (50912040302). We would also like to thank the anonymous reviewers, for offering thoughtful comments and helpful advice on earlier versions of this work.

References

- Anschel, O.; Baram, N.; and Shimkin, N. 2017. Averaged-DQN: Variance Reduction and Stabilization for Deep Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, volume 70, 176–185.
- Azar, M. G.; Gómez, V.; and Kappen, H. J. 2012. Dynamic policy programming. *J. Mach. Learn. Res.*, 13: 3207–3245.
- Baird, L. C. 1999. Reinforcement learning through gradient descent. Ph.D. Dissertation, Carnegie Mellon University.
- Bellemare, M. G.; Ostrovski, G.; Guez, A.; Thomas, P. S.; and Munos, R. 2016. Increasing the Action Gap: New Operators for Reinforcement Learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 1476–1483. AAAI Press.
- Bellman, R. 1958. Dynamic Programming and Stochastic Control Processes. *Inf. Control.*, 1(3): 228–239.
- Bellman, R.; and Dreyfus, S. 1959. Functional approximations and dynamic programming. *Mathematics of Computation*, 13(68): 247–247.
- Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. *arXiv preprint arXiv:1606.01540*.
- Farahmand, A. M. 2011. Action-Gap Phenomenon in Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 172–180.
- Feinberg, A. 1996. Markov Decision Processes: Discrete Stochastic Dynamic Programming (Martin L. Puterman). *SIAM*, 38(4): 689.
- Ferret, J.; Pietquin, O.; and Geist, M. 2021. Self-Imitation Advantage Learning. In *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems*, 501–509. ACM.
- Fu, J.; Kumar, A.; Soh, M.; and Levine, S. 2019. Diagnosing Bottlenecks in Deep Q-learning Algorithms. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, volume 97 of *Proceedings of Machine Learning Research*, 2021–2030.
- Gan, Y.; Zhang, Z.; and Tan, X. 2021. Stabilizing Q Learning Via Soft Mellowmax Operator. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, 7501–7509.
- Kim, S.; Asadi, K.; Littman, M. L.; and Konidaris, G. D. 2019. DeepMellow: Removing the Need for a Target Network in Deep Q-Learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, 2733–2739.
- Kozuno, T.; Uchibe, E.; and Doya, K. 2017. Unifying Value Iteration, Advantage Learning, and Dynamic Policy Programming.
- Lan, Q.; Pan, Y.; Fyshe, A.; and White, M. 2020. Maxmin Q-learning: Controlling the Estimation Bias of Q-learning. In *International Conference on Learning Representations, ICLR*.
- Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR*.
- Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. volume 518, 529–533.
- Munos, R. 2007. Performance Bounds in L_p -norm for Approximate Value Iteration. *SIAM J. Control. Optim.*, 46(2): 541–561.
- Scherrer, B.; Ghavamzadeh, M.; Gabillon, V.; Lesner, B.; and Geist, M. 2015. Approximate modified policy iteration and its application to the game of Tetris. *J. Mach. Learn. Res.*, 16: 1629–1676.
- Smirnova, E.; and Dohmatob, E. 2020. On the Convergence of Smooth Regularized Approximate Value Iteration Schemes. In *Advances in Neural Information Processing Systems*.
- Song, Z.; Parr, R.; and Carin, L. 2019. Revisiting the Softmax Bellman Operator: New Benefits and New Perspective. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, volume 97, 5916–5925.
- Sutton, R. S.; and Barto, A. G. 1998. Reinforcement learning: an introduction.
- Thrun, S.; and Schwartz, A. 1993. Issues in Using Function Approximation for Reinforcement Learning. *Proceedings of the 4th Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1–9.
- van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep Reinforcement Learning with Double Q-Learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2094–2100.
- Vieillard, N.; Pietquin, O.; and Geist, M. 2020. Munchausen Reinforcement Learning. In *Advances in Neural Information Processing Systems*.
- Watkins, C. 1989. Learning from Delayed Rewards. PhD thesis, Kings College, Cambridge.
- Young, K.; and Tian, T. 2019. MinAtar: An Atari-inspired Testbed for More Efficient Reinforcement Learning Experiments.