Stabilizing Q Learning Via Soft Mellowmax Operator

Yaozhong Gan, Zhe Zhang, Xiaoyang Tan

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics
MIIT Key Laboratory of Pattern Analysis and Machine Intelligence
{yzgancn, zhangzhe, x.tan}@nuaa.edu.cn

Abstract

Learning complicated value functions in high dimensional state space by function approximation is a challenging task, partially due to that the max-operator used in temporal difference updates can theoretically cause instability for most linear or non-linear approximation schemes. Mellowmax is a recently proposed differentiable and non-expansion softmax operator that allows a convergent behavior in learning and planning. Unfortunately, the performance bound for the fixed point it converges to remains unclear, and in practice, its parameter is sensitive to various domains and has to be tuned case by case. Finally, the Mellowmax operator may suffer from oversmoothing as it ignores the probability being taken for each action when aggregating them. In this paper we address all the above issues with an enhanced Mellowmax operator, named SM2 (Soft Mellowmax). Particularly, the proposed operator is reliable, easy to implement, and has provable performance guarantee, while preserving all the advantages of Mellowmax. Furthermore, we show that our SM2 operator can be applied to the challenging multi-agent reinforcement learning scenarios, leading to stable value function approximation and state of the art performance.

Introduction

Reinforcement Learning (RL) aims to learn an optimal mapping from states to actions, through experience obtained by interacting with the underlying environment, in terms of some accumulated performance criteria. One major challenge of RL is that in many real-world scenarios (e.g., Atari games (Mnih et al. 2015), continuous control task (Dhariwal et al. 2017), and robot control (Lillicrap et al. 2016)), one has to explore the optimal control in high dimensional state and/or action space, leading to the so-called 'Curse of Dimensionality' problem. Among others, one popular and successful approach to address this tractability issue is to rely on suitable approximate representations. For example, in value function-based RL, it is required that a target value function can be sufficiently precisely captured by some general-purpose function approximators, such as a deep neural network (Mnih et al. 2015). However, training such a value function approximator is in general difficult and can diverge even in some simple cases (Sutton 1996). It is found that in practice the deep RL is sensitive to both the initial condition and the underlying training dynamics (Henderson et al. 2017).

Theoretically the max-operator in Bellman equation and temporal difference (TD) can keep most linear or non-linear approximation schemes from converging (Tsitsiklis and Van Roy 1997). One possible reason behind this is that the non-differentiable property of max-operator has a discretization effect during the TD target estimation - that is, small changes in value function may cause large policy variation, making a suboptimal policy being incorrectly reinforced. This issue is termed *overestimation phenomenon* in literatures (Thrun and Schwartz 1993), which turns out to be a major source of instability of the original deep Q-network (DQN) (Mnih et al. 2015). To address this issue, double DQN (DDQN) adopts a heuristic approach by decoupling the action selection and the TD target estimation (van Hasselt, Guez, and Silver 2016).

Recently there has been increasing interest in stablizing the training procedure of non-linear function approximator by replacing the max function with softmax operators with better theoretical properties. In (Anschel, Baram, and Shimkin 2017), a method named Averaged-DQN is proposed by basing their max-operator on a running average function over previously learned Q-values estimates. They show that this is beneficial to alleviate the overestimation issue by controlling its upper bound. To encourage exploration and improve robustness, several authors investigate using the technique of entropy regularization to smooth policy (Fox, Pakman, and Tishby 2016; Mei et al. 2019; Dai et al. 2018; Lee, Choi, and Oh 2017), this leads to softmax operators. (Song, Parr, and Carin 2019) revisit the softmax Bellman operator and show that it can consistently outperform its max counterpart on several Atari games.

Along this line of research, an alternative mellowmax operator is proposed in (Asadi and Littman 2017). This is a differentiable and non-expansion softmax operator that allows a convergent behavior in learning and planning. Unfortunately, the performance bound for the fixed point it converges to remains unclear, and in practice, its parameter is sensitive to various domains and has to be tuned case by case (Kim et al. 2019). Finally, the Mellowmax operator may suffer from oversmoothing as it ignores the probability being taken for each action when aggregating them.

In this paper we address all the above issues of Mellow-max with an enhanced operator, named SM2 (Soft Mellow-max). Particularly, the SM2 operator can be thought of as a weight quasi-arithmetic mean (Beliakov, Sola, and Calvo 2016), taking into account all action values in each state and the corresponding policy. We show that the proposed operator not only has provable performance guarantee, but is reliable, easy to implement, and capable of stablizing the TD training by alleviating the overestimation error.

As another contribution of this work, we extend our study of soft operator into the field of multi-agent reinforcement learning (MARL) (Kraemer and Banerjee 2016; Oliehoek, Spaan, and Vlassis 2011), showing that *overestimation phenomenon* could become even worse with the increasing number of agents. This is an important issue but unfortunately often ignored by the community, and our result highlights the need to pay more attention on this. To this end, we prove that our SM2 operator is helpful in tackling the overestimation problem in MARL, and provide the lower and upper bound of the overestimation reduction that could be achieved. Extensive experimental results on the StarCraft Multi-agent Challenge (SMAC) (Vinyals et al. 2017) benchmark show that our proposed method significantly improves the performance and sample efficiency compared to several state of the art MARL algorithms.

Background

Commonly, 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 state space and action space respectively. The function $P(s'|s, a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longmapsto [0, 1]$ outputs the transition probability from state s to state s' under action a. The reward on each transition is given by the function $R(s, a) : \mathcal{S} \times \mathcal{A} \longmapsto \mathbb{R}$, whose maximum absolute value is R_{max}, $R_{max} = \max |R(s, a)|$. $\gamma \in [0, 1)$ is the discount factor for long-horizon returns. The goal is to learn a policy $\pi : \mathcal{S} \times \mathcal{A} \longmapsto [0, 1]$ for interacting with the environment.

The Bellman and Mellowmax Operator

Based on the MDP model, 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 standard Bellman operator $\mathcal{T}$ is defined as,

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

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 standard Bellman equation and satisfies:

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

The corresponding optimal policy is defined as $\pi^* = \arg\max_a Q^*(s, a)$.

In DQN (Mnih et al. 2015), the action value function $Q_\theta(s, a)$ is implemented with a deep neural network parametrized by θ, which takes the state s as input and outputs the Q value of every action a. In order to learn the parameters θ, batches of transitions are sampled from the *replay buffer*, which stores the transition tuples $\langle s, a, R, s' \rangle$. By minimizing the squared TD error, the training object can be represented as:

$$\min_\theta \frac{1}{2} \|Q_\theta(s, a) - y_T^{DQN}\|^2 \quad (2)$$

where $y_T^{DQN} = R(s, a) + \gamma \max_{a'} Q_{\theta^-}(s', a')$ denotes the target value, θ^- represents the parameters of a target network that are periodically copied from θ.

To alleviate the overestimation problem and make the training process more stable, double-DQN (van Hasselt, Guez, and Silver 2016) uses two separate neural networks for action selection and evaluation and replaces the target y_T^{DQN} in Eq.(2) with

$$y_T^{DDQN} = R(s, a) + \gamma Q_{\theta^-}(s', \arg\max_{a'} Q_\theta(s', a'))$$

That is, it selects the action based on the current network θ and evaluates the Q value using the target network θ^-.

Mellowmax operator (Asadi and Littman 2017; Kim et al. 2019) is an alternative way to reduce the overestimation bias, and is defined as:

$$mm_\omega Q(s', \cdot) = \frac{1}{\omega} \log[\sum_{i=1}^{n} \frac{1}{n} \exp(\omega Q(s', a_i'))] \quad (3)$$

where $\omega > 0$, and by adjusting it, the $mm_\omega Q(s, \cdot)$ value can change between the mean and maximum of all action values $Q(s, \cdot)$ under state s. Analogous to the Bellman operator Eq.(1), we can also define the Mellomax Bellman operator as:

$$\mathcal{T}_m Q(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) mm_\omega Q(s', \cdot) \quad (4)$$

Similarly, mellow-DQN algorithm modifies the target y_T^{DQN} in Eq.(2) with $y_T^{mm_\omega} = R(s, a) + \gamma mm_\omega Q_{\theta^-}(s', \cdot)$.

It can be shown that mm_ω is non-expansion under the infinity norm. Therefore, $Q(s, a)$ is guaranteed to converge to a unique fixed point by recursively iterating $\mathcal{T}_m$. However, the relationship between the stationary Q value and the optimal Q value is still unknown. And in practice, the parameter ω is sensitive to different tasks and has to be tuned case by case (Kim et al. 2019). According to Eq.(3), Mellowmax can be thought of as a controlled smoothing operator that aggregates Q values over the whole action space to give an estimation of future return. Under some complicated environments with sparse rewards or large action space, the estimated action values calculated by Mellowmax may tend to be much smaller than the ones by max operator with an improper ω value. We call this oversmoothing issue.

Soft Mellowmax

In this section, we first introduce a new operator called soft mellowmax (SM2) operator, which aims to avoid the shortcomings of mellow operator mentioned before, and then give its theoretic properties.

The Soft Mellowmax Operator

The soft mellowmax operator is defined as

$$sm_\omega Q(s, \cdot) = \frac{1}{\omega} \log[\sum_{i=1}^{n} soft_\alpha(Q(s', a_i')) \exp(\omega Q(s', a_i'))] \quad (5)$$

where $soft_\alpha(Q(s', a_i')) = \frac{\exp(\alpha Q(s', a_i'))}{\sum_{j=1}^{n} \exp(\alpha Q(s', a_j'))}$, $\omega > 0$ and $\alpha \in \mathbb{R}$, which can be viewed as a particular instantiation of the weighted quasi-arithmetic mean (Beliakov, Sola, and Calvo 2016). The $soft_\alpha Q$ term can be thought of as a policy probability, and can be justified from the perspective of entropy regularization framework with KL divergence (Fox, Pakman, and Tishby 2016; Mei et al. 2019).

Analogous to the Bellman operator Eq.(1), we define the Soft Mellowmax Bellman operator as:

$$\mathcal{T}_{sm}Q(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a)sm_\omega Q(s', \cdot) \quad (6)$$

Note that if $\alpha = 0$, the $\mathcal{T}_{sm}$ operator is reduced to the Mellomax Bellman operator $\mathcal{T}_m$. As will shown below, $\mathcal{T}_{sm}$ extends the capability of the $\mathcal{T}_m$ in several aspects, in terms of provable performance bound, overestimation bias reduction and sensitiveness to parameters settings.

Contraction

The following theorem shows that that $\mathcal{T}_{sm}$ is a contraction. That is, it is guaranteed to converge to a unique fixed point.

Theorem 1. *(Contraction) Let Q_1, Q_2 are two different Q-value functions. If $\omega > 0$ and $-\frac{\omega}{1 - e^{-c\omega}} \le \alpha \le \frac{\omega}{e^{c\omega} - 1}$ where $c = \frac{2R_{max}}{1-\gamma}$. Then*

$$\|\mathcal{T}_{sm}Q_1 - \mathcal{T}_{sm}Q_2\| \le \gamma\|Q_1 - Q_2\|$$

The proof of the theorem is given in Appendix.

Here we give an example, showing that if there is no restriction on α, the $\mathcal{T}_{sm}$ operator will not be a contraction.
Example 1.
Let $\mathcal{S} = \{0\}$, $\mathcal{A} = \{1, 2\}$, $\alpha = 1$, $\omega = 1$, $Q_1(0, 1) = 50$, $Q_1(0, 2) = 1$, $Q_2(0, 1) = 5$, and $Q_2(0, 2) = 1$, then

$$|sm_\omega Q_1 - sm_\omega Q_2| \approx 45.018 \ge \max_a |Q_1 - Q_2| = 45$$

Therefore, we have shown that $\mathcal{T}_{sm}$ is a contraction under certain conditions, but how about the quality of the converged fixed point then? we answer this next.

Performance Bound

In this section, we derive the bound of the performance gap between the optimal action-value function Q^* and Q_k during k-th Q-iterations with $\mathcal{T}_{sm}$ Eq.(6). And we only discuss the case of $\alpha \ge 0$ in the following sections.

Theorem 2. *Let the optimal Q-value function Q^* be a fixed point during Q-iteration with $\mathcal{T}$, i.e. $Q^*(s, a) = \mathcal{T}Q^*(s, a)$, and $Q^k(s, a) \triangleq \mathcal{T}_{sm}^k Q^0(s, a)$ for arbitrary initial function $Q^0(s, a)$ during k-th Q-iteration. $\forall (s, a)$*

(I) If $\alpha \ge \omega$, then

$$\limsup_{k \to \infty} \|Q^*(s, a) - Q^k(s, a)\| \le \frac{\gamma}{\omega(1 - \gamma)} \log(\frac{1 + n}{2})$$

(II) If $\alpha < \omega$, then

$$\limsup_{k \to \infty} \|Q^*(s, a) - Q^k(s, a)\| \le \frac{\gamma}{\omega(1 - \gamma)} \log(n - \frac{\alpha(n-1)}{\alpha + \omega})$$

Corollary 1. *For the Mellowmax Bellman operator $\mathcal{T}_m$ Eq.(4), defined $Q^k(s, a) \triangleq \mathcal{T}_m^k Q^0(s, a)$, we have*

$$\lim_{k \to \infty} \|Q^\star(s, a) - Q^k(s, a)\| \le \frac{\gamma}{\omega(1 - \gamma)} \log(n)$$

The proof of the theorem is given in Appendix.

This result shows that the optimal action-value function Q^* and Q^k can be limited to a reasonable range in either cases, and the bound is independent of the dimension of state space, but depends on the size n of action space.

Furthermore, it shows that $\mathcal{T}_{sm}$ converges to $\mathcal{T}$ with a linear rate with respect to ω when $\alpha \ge \omega$. If $\alpha < \omega$, it converges to $\mathcal{T}$ with respect to ω and α. Compared with Mellowmax Bellman operator, our method can converge to the optimal Q^* more quickly. Note that the softmax operator (Song, Parr, and Carin 2019) has a better convergence rate, that is, exponential with respect to the parameters instead of linear, and its scalar term is $(n - 1)$. In contrast, the scalar term of our method and some other methods within entropy-regularized MDP framework (Dai et al. 2018; Lee, Choi, and Oh 2017) are $\log n$, as shown in theorem 2.

Alleviating Overestimation

The overestimation issue of Q-learning due to the max operator may lead to the suboptimal policy and cause the failure of Q-learning (Thrun and Schwartz 1993), and it is may be one of the main reasons of the poor performance of DQN in many real-world scenarios (van Hasselt, Guez, and Silver 2016). Recently this issue has received increasing attentions (van Hasselt, Guez, and Silver 2016; Anschel, Baram, and Shimkin 2017; Song, Parr, and Carin 2019). In this section, following the same assumptions (van Hasselt, Guez, and Silver 2016), we show that our SM2 operator is able to reduce the overestimation in Q-learning.

Theorem 3. *Assume all the true optimal action value are equal at $Q^*(s, a) = V^*(s)$ and the estimation errors $Z^{s,a} = Q(s, a) - Q^*(s, a)$ are independently distributed uniformly random in $[-\epsilon, \epsilon]$, defined the overestimation error $\Theta \triangleq \mathbb{E}[\max_a Q(s, a) - Q^*(s, a)]$, and $\Theta_{sm} \triangleq \mathbb{E}[sm_\omega Q(s, a) - Q^*(s, a)]$,*

(I) If $\alpha \ge \omega$, then

$$\Theta - \Theta_{sm} \in (0, \frac{1}{\omega} \log(\frac{1 + n}{2})]$$

(II) If $\alpha < \omega$, then

$$\Theta - \Theta_{sm} \in (0, \frac{1}{\omega}\log(n - \frac{\alpha(n-1)}{\alpha + \omega})]$$

(III) The overestimation error for $\mathcal{T}_{sm}$ is increasing monotonically w.r.t. $\alpha \in [0, \infty)$ and $\omega \in (0, \infty)$.

Corollary 2. *Under the above assumption, defined $\Theta_m \triangleq \mathbb{E}[mm_\omega Q(s,a) - Q^*(s,a)]$, we have*

$$\Theta - \Theta_m \in (0, \frac{1}{\omega}\log(n)]$$

The proof of the theorem is given in Appendix.

This theorem implies that our SM2 algorithm can reduce the overestimation of max-operator. Particularly, when ω is fixed, we can adjust the scalar of α to control the degree of overestimation in part (II). Similarly, we also analyze the overestimation of Mellowmax operator. If the ω of SM2 and Mellowmax are same, the overestimation bias of Mellowmax is smaller, which may also cause the oversmoothing issue. SM2 considers the weights of difference Q values, which take the importance of each Q value into account. Compared with (Song, Parr, and Carin 2019), the scalar term of our method is at most $\log(n)$, but independent of γ and R_{max}.

Soft Mellowmax for Mulit-Agent Reinforcement Learning

In this section, we investigate how to apply the proposed SM2 operator for multi-agent reinforcement learning (MARL). Particularly, we consider a fully cooperative multi-agent scenario under the paradigm of centralized training with decentralized execution (CTDE). The goal is to find a separate policy for each agent i, such that when following the joint policy $\boldsymbol{\pi}$ in an predefined environment, the expected centralized value Q_{tot} is maximized, under certain coordinative constraints between centralized value function Q_{tot} and decentralized value functions Q_i. Formally, we consider the following optimization problem.

$$\underset{\boldsymbol{\pi}}{\text{maximize}} \underset{s_0 \sim \rho_0(s_0), \boldsymbol{a} \sim \boldsymbol{\pi}}{\mathbb{E}}[Q_{tot}(s_0, \boldsymbol{a})]$$
$$\text{subject to } Q_{tot}(s, \boldsymbol{a}) = f(\boldsymbol{Q})(s, \boldsymbol{a})$$

where $\rho_0 : \mathcal{S} \longmapsto [0,1]$ is the distribution of the initial state s_0. The value functions Q_i of each agent i are combined to give the centralized value function Q_{tot}, through a state-dependent continuous function f, where $f(\boldsymbol{Q})(s, \boldsymbol{a}) = f(Q_1(s, a_1), \cdots, Q_N(s, a_N))$, with $\boldsymbol{Q} = (Q_1, Q_2, \cdots, Q_N)$ being the joint value vector and $\mathbf{a} = (a_1, a_2, \cdots, a_N)$ being the joint action vector of the N agents. In what follows, for simplicity, we ignore $(s, \mathbf{a})$.

Assumption 1. *For any $\boldsymbol{Q} = [Q_i]_{i=1}^N$, and exits $l \geq 0$, $L > 0$, the following holds*

$$l \leq \frac{\partial Q_{tot}}{\partial Q_i} \leq L, \ i = 1, 2, \cdots, N.$$

Assumption 1 suggests that Q_{tot} is a monotonic function and a Lipschitz continuous function about $\mathbf{Q}$ (every function that has bounded first derivatives is Lipschitz continuous). Among others, both VDN (Sunehag et al. 2018) and QMIX (Rashid et al. 2018) satisfy this assumption in practice. We take this assumption as a prior to capture the coordination information required to solve cooperative tasks. That is, each agent has a chance to influence the team reward positively instead of canceling out with each other in fully cooperative scenarios.

Following the same assumptions of theorem 3, we first quantify the overestimation problem in the situation of MARL, as follows,

Theorem 4. *Under the assumptions of Theorem 3 Defined the overestimation error $\Theta^1 \triangleq \mathbb{E}[Q_{tot}(\max_{\boldsymbol{a}} \boldsymbol{Q})(s, \boldsymbol{a}) - Q_{tot}(\boldsymbol{Q})^*(s, \boldsymbol{a})]$, then*

$$\Theta^1 \in [\epsilon l N \frac{n-1}{n+1}, \epsilon L N \frac{n-1}{n+1}]$$

The proof of the theorem is given in Appendix.

This result implies that the amount of overestimation is proportional to the number of agents, which means that in MARL, the overestimation problem is more serious than that in the single agent case as more than one agents are involved in learning. As shown in the experimental section (c.f., Figure 3), the QMIX does suffer from this issue.

Next, we show that our SM2 operator is beneficial to address this, and gives both lower and upper bounds of the range of the overestimation reduction.

Theorem 5. *Under the assumptions of Theorem 3. Defined the overestimation error $\Theta^1_{sm} \triangleq \mathbb{E}[Q_{tot}(sm_\omega \boldsymbol{Q})(s, \boldsymbol{a}) - Q_{tot}(\boldsymbol{Q})^*(s, \boldsymbol{a})]$,*

(I) If $\alpha \geq \omega$, then

$$\Theta^1 - \Theta^1_{sm} \in (0, \frac{1}{\omega}LN\log(\frac{1+n}{2})]$$

(II) If $\alpha < \omega$, then

$$\Theta^1 - \Theta^1_{sm} \in (0, \frac{1}{\omega}LN\log(n - \frac{\alpha(n-1)}{\alpha + \omega})]$$

The proof of the theorem is given in Appendix.

This theorem implies that our SM2 algorithm can reduce overestimation in MARL. The scale of the overestimation error is related to the selection of hyperparameters and the number of agents.

Experiment

In this section, we present our experimental results to verify the effectiveness of the proposed SM2 method.

Overview

We performed two types of experiments: one is about single agent learning and the other is about multi-agent learning. For single agent RL, we used several games from PyGame Learning Environment (PLE) (Tasfi 2016) and MinAtar (Young and Tian 2019); while for MARL, we adopted

(a) Catcher-PLE   (b) Pixelcopter-PLE
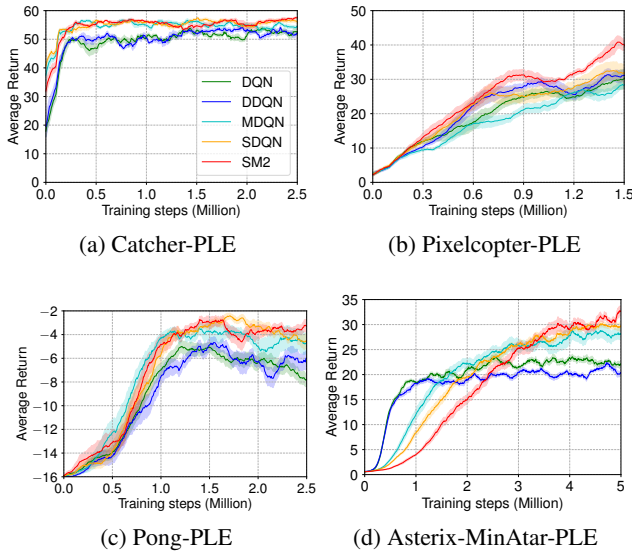
(c) Pong-PLE   (d) Asterix-MinAtar-PLE

Figure 1: Learning curves on the PLE and MinAtar environments. The depicted return is averaged over 10 test episodes every 5000 steps across 5 independent runs.

Algorithm	DQN	DDQN	SDQN	MDQN	SM2
Catcher	52.47 (3.48)	52.06 (1.30)	56.36 (2.21)	55.53 (1.32)	**57.49** (0.84)
Pixelcopter	25.95 (11.02)	28.37 (10.61)	31.79 (8.69)	28.27 (3.11)	**40.11** (5.57)
Pong	-8.67 (2.63)	-6.15 (2.26)	-4.59 (2.06)	-4.70 (2.88)	**-3.29** (1.81)
Asterix	22.16 (2.97)	20.29 (2.65)	29.77 (3.90)	28.15 (3.24)	**32.66** (3.08)

Table 1: Mean of average return for different methods in Catcher (Catcher-PLE), Pixelcopter (Pixelcopter-PLE), Pong (Pong-PLE) and Asterix (Asterix-MinAtar) games (standard deviation in parenthesis).

StarCraft Multi-agent Challenge (SMAC) (Samvelyan et al. 2019). For each type of experiments, we used as baselines several related soft operators or approaches aiming to reduce overestimation for Q-learning, including DDQN (van Hasselt, Guez, and Silver 2016), SoftMax (Song, Parr, and Carin 2019) and MellowMax (Asadi and Littman 2017). We choose the candidates α and ω from the set of $\{1, 2, 5, 10, 15\}$ for SM2, and choose ω among $\{1, 5, 10, 15, 20\}$ for Softmax (Song, Parr, and Carin 2019). As for Mellowmax (Asadi and Littman 2017), we choose ω among $\{5, 10, 30, 50, 100, 200\}$. We select the best hyperparameters for these methods on each environment separately.

PLE and MinAtar

Experimental Setup. For PLE game environments, the neural network was a multi-layer perceptron with hidden

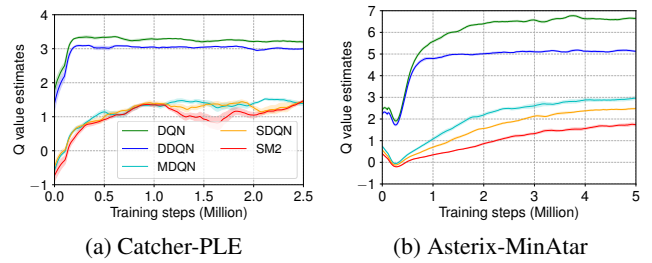(a) Catcher-PLE   (b) Asterix-MinAtar

Figure 2: The estimated Q values for our methods SM2, and comparison methods. The mean and a standard deviation are shown across 5 independent runs.

layer fixed to [64, 64]. The discount factor was 0.99. The size of the replay buffer was 10000. The weights of neural networks were optimized by RMSprop with gradient clip 5. The batch size was 32. The target network was updated every 200 frames. ϵ-greedy was applied as the exploration policy with ϵ decreasing linearly from 1.0 to 0.01 in 1, 000 steps. After 1, 000 steps, ϵ was fixed to 0.01. The rest of the experimental setting are detailed in the appendix.

Performance Comparison. As mentioned before, we used the classical DQN algorithm (Mnih et al. 2015) and its several variants: DDQN (van Hasselt, Guez, and Silver 2016), Soft DQN (SDQN) (Song, Parr, and Carin 2019) and Mellow DQN (MDQN) (Asadi and Littman 2017) as baselines for performance comparison. We evaluated the average return of 10 test episodes every 5000 training steps.

Figure 1 shows all the results of each algorithm. From the figure, one can see that the proposed SM2 performs as well as or better than other algorithms in all the domains. In addition, except the early stages in Asterix, SM2 maintains a faster learning in most of the time. Table 1 gives the mean and standard deviation of average returns for all algorithms across the four games at the end of training. According to these quantitative results, one can see that SM2 achieves the best final performance with the lower variance.

Overestimation Reduction. To verify the capability of the proposed SM2 method to make the training process of DQN more stable, we compared the estimated Q-values (and the corresponding mean and one standard deviation) from DQN, DDQN, MDQN, SDQN and SM2 in Catcher and Asterix tasks. Figure 2 gives the results. It shows that SM2 can reduce the overestimation bias, which is consistent with the analysis made in Section 3.4. To some extent, this overestimation reduction also helps to explain the higher average return and more stable learning for SM2 in Figure 1.

The StarCraft Multi-agent Challenge

Comparison with Other Overestimation Reduction Methods. In the first set of experiments, we used the QMIX (Rashid et al. 2018) as our basic test bed and adopted several improved algorithms based on it for the overestimation reduction: DQMIX which used double network (van Hasselt, Guez, and Silver 2016) in QMIX, WDQMIX which
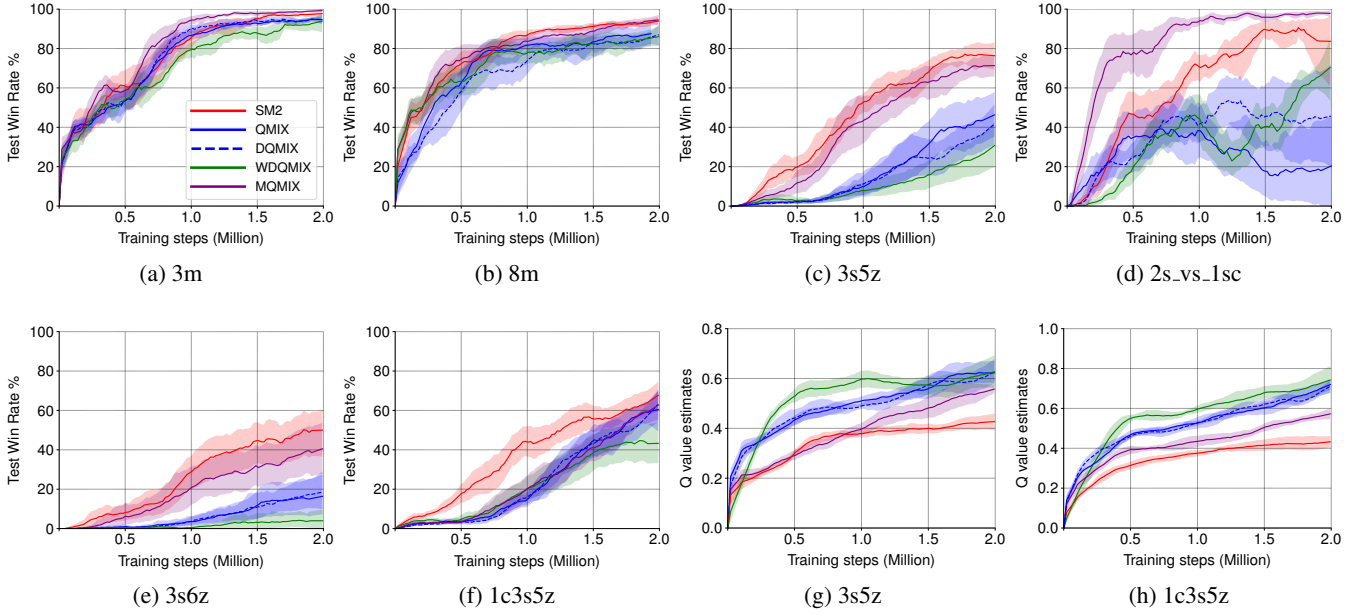
Figure 3: Test win rates for our methods SM2 with $\alpha = 10$ and $\omega = 5$, and comparison methods QMIX, double-QMIX (DQMIX) and Mellowmax-QMIX (MQMIX) in six different scenarios (a)-(f). The estimated Q-values for our methods SM2 and other algorithms in (g)-(h). The mean and 95% confidence interval are shown across 10 independent runs.
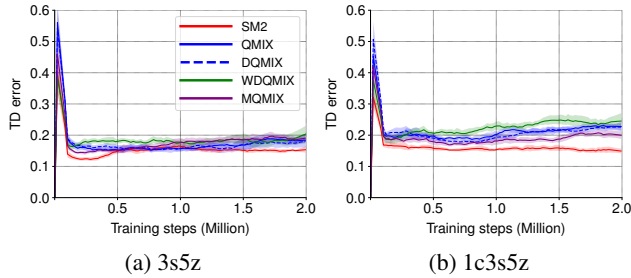

Figure 4: The TD error of different algorithms are evaluated. The mean and 95% confidence interval are shown across 10 independent runs.

Figure 5: The estimated Q values of SM2 under condition $\alpha < \omega$. The mean and 95% confidence interval are shown across 10 independent runs.

used weighted double network (Zheng et al. 2018) in QMIX, MQMIX which used mellowmax in QMIX.

Figure 3 shows the mean and 95% confidence interval of the test win rate over the different algorithms in six different scenarios. The results suggest that our method SM2 achieves a higher win rate and faster learning than QMIX, DQMIX and WDQMIX. As for MQMIX, our SM2 is better or comparable in five of the six tasks. From these performance curves, we can also see that SM2 not only achieves a better performance, but also improves the sample efficiency in most scenarios. Note that DQMIX doesn't obviously improve the performance compared with QMIX except in 2s_vs_1sc map, and sometimes its performance may be worse. This suggests that the double Q networks used in MARL can not have the same effect as in the single-agent
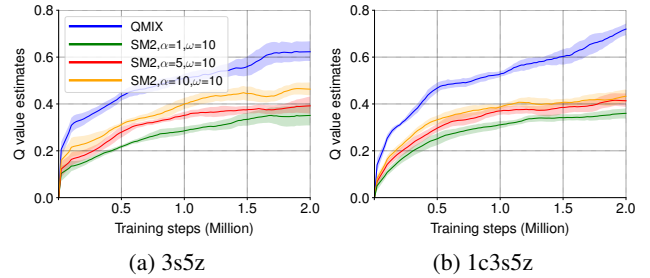
setting. In 2s_vs_1sc, it can be seen that QMIX has fallen into a suboptimal policy, and cannot jump out. MQMIX can achieve better performance - however, its performance is sensitive to the parameter ω, which has to be tuned for each task. For example, in 2s_vs_1sc, $\omega = 5$ of mellowmax can get better performance, while $\omega = 200$ in 1c3s5z map.

By contrast, SM2 can quickly learn a good policy, and doesn't fall into the suboptimal policy using the same set of hyperparameters $\alpha = 10$ and $\omega = 5$ across all the tasks.

TD error Comparison Figure 4 shows the TD error of different algorithms evaluated over different scenarios. The TD error merely at the current state is less representative, so we first sample some batches of state-action pair from the replay buffer, and calculate the absolute value of the TD error for each pair. And finally we choose the mean of them

Algorithm	QMIX	MQMIX $\omega=5$	MQMIX $\omega=10$	MQMIX $\omega=30$	MQMIX $\omega=50$	MQMIX $\omega=100$	MQMIX $\omega=200$	SM2 $\alpha=1,$ $\omega=10$	SM2 $\alpha=5,$ $\omega=5$	SM2 $\alpha=5,$ $\omega=10$	SM2 $\alpha=10,$ $\omega=5$	SM2 $\alpha=10,$ $\omega=10$
3m	94.57 (3.30)	65.08 (23.54)	97.56 (3.55)	99.23 (1.01)	98.51 (1.39)	97.36 (2.31)	96.30 (3.91)	97.03 (2.92)	98.80 (1.62)	95.51 (3.87)	97.65 (2.03)	97.30 (3.41)
8m	87.30 (9.18)	1.24 (2.26)	7.42 (7.95)	90.87 (6.10)	94.35 (3.65)	94.80 (3.45)	94.19 (3.98)	90.27 (4.12)	94.73 (3.16)	93.81 (3.76)	93.72 (3.06)	92.78 (3.65)
3s5z	45.67 (18.59)	3.49 (6.68)	3.35 (3.11)	22.97 (13.62)	51.85 (12.56)	71.29 (8.07)	63.32 (12.78)	74.16 (4.59)	73.74 (13.43)	76.85 (9.69)	76.47 (10.10)	75.42 (3.65)
2s_vs_1sc	19.82 (36.56)	97.89 (2.13)	95.44 (4.83)	83.06 (28.27)	85.55 (19.16)	66.66 (37.79)	50.19 (40.67)	95.34 (7.89)	88.86 (20.23)	92.42 (9.53)	83.69 (31.08)	92.61 (4.25)
3s6z	16.14 (16.06)	3.83 (5.73)	4.48 (6.56)	18.06 (10.70)	26.75 (15.98)	40.06 (19.99)	32.21 (21.27)	55.51 (6.03)	58.69 (15.48)	50.41 (18.77)	49.94 (18.88)	44.91 (23.29)
1c3s5z	60.15 (15.00)	8.10 (8.10)	1.42 (3.48)	16.68 (11.34)	27.07 (17.90)	55.69 (15.65)	61.83 (10.81)	54.69 (13.38)	59.02 (11.48)	61.61 (8.03)	66.80 (11.72)	63.84 (8.80)

Table 2: Mean of average return for different hyperparameters of the algorithms in six different scenarios (standard deviation in parenthesis).
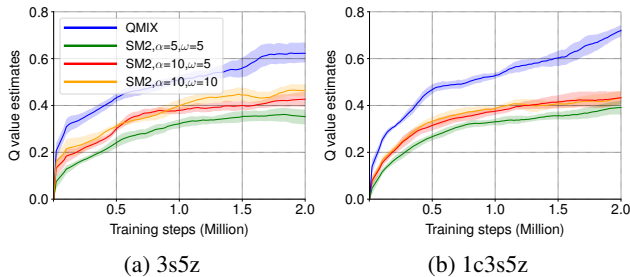

(a) 3s5z (b) 1c3s5z

Figure 6: The estimated Q values of SM2 under condition $\alpha \geq \omega$. The mean and 95% confidence interval are shown across 10 independent runs.

as the estimate of TD error. It can be seen that the TD error of SM2 is very smaller than other methods, which implies that the target estimates of SM2 are more accurate. Furthermore, the smaller TD error also makes the training of SM2 more stable.

Overestimation Reduction. Figure 3 gives the estimated Q values of different algorithms. We can see that the estimated Q values of SM2 are much smaller than others, indicating that SM2 reduces the possibility of overestimation. In Figure 3, one can see that the reduction of the overestimation makes SM2 perform better, especially on difficult scenarios. The estimated Q values of DQMIX are almost the same as QMIX, indicating that it does not reduce the overestimation of Q values significantly - actually, its performance is also similar to QMIX. Overall, our algorithm performs more conservatively in the estimation of Q value, which makes the training more stable and reduces the possibility of falling into the suboptimal policy.

Sensitiveness to Hyperparameters Setting. In this section, we analyze the influence of α and ω on the estimated Q value in SM2. From Figure 5 and 6, two sets of com-

parative experiments are implemented under two conditions about $\alpha \geq \omega$ and $\alpha < \omega$. We find that the gap between the Q values of SM2 and QMIX becomes smaller as the scales of α and ω increase. In other words, if α and ω increase, the estimated Q values will also increase. This result is consistent with previous theoretical analyses. Table 2 shows the effect of different hyperparameters of Mellowmax and SM2. This suggests that Mellowmax would have the bad performance with an improper ω. In contrast, SM2 doesn't require a lot of hyperparameter adjustments across different tasks. When $\omega = 10$, we find that a small α can greatly improve the algorithm performance. And different choices of α and ω have little difference on performance, which shows the insensitivity of SM2.

Conclusion and Future Work

In our work, we propose a new method SM2, which is an enhanced version of the Mellowmax operator. Our theoretical analysis and extensive experimental results show that SM2 can effectively reduce the overestimation bias, and own the capability to converge to the optimal Q function in a stable way. We also show that the overestimation is proportional to the number of agents under some mild conditions in MARL, and our proposed method is beneficial to address this issue and achieves better multi-agent cooperations as well.

In this paper, we only discussed the case where the hyperparameters of SM2 are constant. It's an interesting direction to investigate how to tune them automatically in the future, just like (Pan et al. 2020; Kim and Konidaris 2019).

Acknowledgements

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.

Asadi, K.; and Littman, M. L. 2017. An Alternative Softmax Operator for Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, volume 70, 243–252.

Beliakov, G.; Sola, H. B.; and Calvo, T. 2016. *A Practical Guide to Averaging Functions*, volume 329. Springer.

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.

Dai, B.; Shaw, A.; Li, L.; Xiao, L.; He, N.; Liu, Z.; Chen, J.; and Song, L. 2018. SBEED: Convergent Reinforcement Learning with Nonlinear Function Approximation. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, 1133–1142. PMLR.

Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; Wu, Y.; and Zhokhov, P. 2017. OpenAI Baselines. https://github.com/openai/baselines.

Feinberg, A. 1996. Markov Decision Processes: Discrete Stochastic Dynamic Programming (Martin L. Puterman). *SIAM* 38(4): 689.

Fox, R.; Pakman, A.; and Tishby, N. 2016. Taming the Noise in Reinforcement Learning via Soft Updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, USA*.

Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; and Meger, D. 2017. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560* .

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.

Kim, S.; and Konidaris, G. 2019. Adaptive Temperature Tuning for Mellowmax in Deep Reinforcement Learning .

Kraemer, L.; and Banerjee, B. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* 190: 82–94.

Lee, K.; Choi, S.; and Oh, S. 2017. Sparse Markov Decision Processes with Causal Sparse Tsallis Entropy Regularization for Reinforcement Learning. *CoRR* abs/1709.06293.

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*.

Mei, J.; Xiao, C.; Huang, R.; Schuurmans, D.; and Müller, M. 2019. On Principled Entropy Exploration in Policy Optimization. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, 3130–3136.

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. *Nature* 518(7540): 529–533.

Oliehoek, F. A.; Spaan, M. T. J.; and Vlassis, N. A. 2011. Optimal and Approximate Q-value Functions for Decentralized POMDPs URL http://arxiv.org/abs/1111.0062.

Pan, L.; Cai, Q.; Meng, Q.; Chen, W.; and Huang, L. 2020. Reinforcement Learning with Dynamic Boltzmann Softmax Updates 1992–1998.

Rashid, T.; Samvelyan, M.; de Witt, C. S.; Farquhar, G.; Foerster, J. N.; and Whiteson, S. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, volume 80, 4292–4301.

Samvelyan, M.; Rashid, T.; de Witt, C. S.; Farquhar, G.; Nardelli, N.; Rudner, T. G. J.; Hung, C.; Torr, P. H. S.; Foerster, J. N.; and Whiteson, S. 2019. The StarCraft Multi-Agent Challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS*, 2186–2188.

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.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V. F.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; and Graepel, T. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS*, 2085–2087.

Sutton, R. S. 1996. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*, 1038–1044.

Tasfi, N. 2016. PyGame Learning Environment. https://github.com/ntasfi/PyGame-Learning-Environment.

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.

Tsitsiklis, J. N.; and Van Roy, B. 1997. Analysis of temporal-diffference learning with function approximation. In *Advances in neural information processing systems*, 1075–1081.

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.

Vinyals, O.; Ewalds, T.; Bartunov, S.; Georgiev, P.; Vezhnevets, A. S.; Yeo, M.; Makhzani, A.; Küttler, H.; Agapiou, J. P.; Schrittwieser, J.; Quan, J.; Gaffney, S.; Petersen, S.; Simonyan, K.; Schaul, T.; van Hasselt, H.; Silver, D.; Lillicrap, T. P.; Calderone, K.; Keet, P.; Brunasso, A.; Lawrence, D.; Ekermo, A.; Repp, J.; and Tsing, R. 2017. StarCraft II: A New Challenge for Reinforcement Learning. *CoRR* abs/1708.04782.

Young, K.; and Tian, T. 2019. MinAtar: An Atari-inspired Testbed for More Efficient Reinforcement Learning Experiments URL http://arxiv.org/abs/1903.03176.

Zheng, Y.; Meng, Z.; Hao, J.; and Zhang, Z. 2018. Weighted Double Deep Multiagent Reinforcement Learning in Stochastic Cooperative Environments. 421–429. Springer.