Softmax Deep Double Deterministic Policy Gradients

Ling Pan[1], **Qingpeng Cai**[2], **Longbo Huang**[1]
[1]Institute for Interdisciplinary Information Sciences, Tsinghua University
pl17@mails.tsinghua.edu.cn, longbohuang@tsinghua.edu.cn
[2]Alibaba Group
qingpeng.cqp@alibaba-inc.com

Abstract

A widely-used actor-critic reinforcement learning algorithm for continuous control, Deep Deterministic Policy Gradients (DDPG), suffers from the overestimation problem, which can negatively affect the performance. Although the state-of-the-art Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm mitigates the overestimation issue, it can lead to a large underestimation bias. In this paper, we propose to use the Boltzmann softmax operator for value function estimation in continuous control. We first theoretically analyze the softmax operator in continuous action space. Then, we uncover an important property of the softmax operator in actor-critic algorithms, i.e., it helps to smooth the optimization landscape, which sheds new light on the benefits of the operator. We also design two new algorithms, Softmax Deep Deterministic Policy Gradients (SD2) and Softmax Deep Double Deterministic Policy Gradients (SD3), by building the softmax operator upon single and double estimators, which can effectively improve the overestimation and underestimation bias. We conduct extensive experiments on challenging continuous control tasks, and results show that SD3 outperforms state-of-the-art methods.

1 Introduction

Deep Deterministic Policy Gradients (DDPG) [25] is a widely-used reinforcement learning [27, 31, 25, 30] algorithm for continuous control, which learns a deterministic policy using the actor-critic method. In DDPG, the parameterized actor network learns to determine the best action with highest value estimates according to the critic network by policy gradient descent. However, as shown recently in [15], one of the dominant concerns for DDPG is that it suffers from the overestimation problem as in the value-based Q-learning [38] method, which can negatively affect the performance with function approximation [35]. Therefore, it is of vital importance to have good value estimates, as a better estimation of the value function for the critic can drive the actor to learn a better policy.

To address the problem of overestimation in actor-critic, Fujimoto et al. propose the Twin Delayed Deep Deterministic Policy Gradient (TD3) method [15] leveraging double estimators [20] for the critic. However, directly applying the Double Q-learning [20] algorithm, though being a promising method for avoiding overestimation in value-based approaches, cannot fully alleviate the problem in actor-critic methods. A key component in TD3 [15] is the Clipped Double Q-learning algorithm, which takes the minimum of two Q-networks for value estimation. In this way, TD3 significantly improves the performance of DDPG by reducing the overestimation. Nevertheless, TD3 can lead to a large underestimation bias, which also impacts performance [10].

The Boltzmann softmax distribution has been widely adopted in reinforcement learning. The softmax function can be used as a simple but effective action selection strategy, i.e., Boltzmann exploration [34, 9], to trade-off exploration and exploitation. In fact, the optimal policy in entropy-regularized reinforcement learning [18, 19] is also in the form of softmax. Although it has been long believed that the softmax operator is not a non-expansion and can be problematic when used to update value

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

functions [26, 4], a recent work [33] shows that the difference between the value function induced by the softmax operator and the optimal one can be controlled in discrete action space. In [33], Song et al. also successfully apply the operator for value estimation in deep Q-networks [27], and show the promise of the operator in reducing overestimation. However, the proof technique in [33] does not always hold in the continuous action setting and is limited to discrete action space. Therefore, there still remains a theoretical challenge of whether the error bound can be controlled in the continuous action case. In addition, we find that the softmax operator can also be beneficial when there is no overestimation bias and with enough exploration noise, while previous works fail to understand the effectiveness of the operator in such cases.

In this paper, we investigate the use of the softmax operator in updating value functions in actor-critic methods for continuous control, and show that it has several advantages that makes it appealing. Firstly, we theoretically analyze the properties of the softmax operator in continuous action space. We provide a new analysis showing that the error between the value function under the softmax operator and the optimal can be bounded. The result paves the way for the use of the softmax operator in deep reinforcement learning with continuous action space, despite that previous works have shown theoretical disadvantage of the operator [26, 4]. Then, we propose to incorporate the softmax operator into actor-critic for continuous control. We uncover a fundamental impact of the softmax operator, i.e., it can smooth the optimization landscape and thus helps learning empirically. Our finding sheds new light on the benefits of the operator, and properly justifies its use in continuous control.

We first build the softmax operator upon single estimator, and develop the Softmax Deep Deterministic Policy Gradient (SD2) algorithm. We demonstrate that SD2 can effectively reduce overestimation and outperforms DDPG. Next, we investigate the benefits of the softmax operator in the case where there is underestimation bias on top of double estimators. It is worth noting that a direct combination of the operator with TD3 is ineffective and can only worsen the underestimation bias. Based on a novel use of the softmax operator, we propose the Softmax Deep Double Deterministic Policy Gradient (SD3) algorithm. We show that SD3 leads to a better value estimation than the state-of-the-art TD3 algorithm, where it can improve the underestimation bias, and results in better performance and higher sample efficiency.

We conduct extensive experiments in standard continuous control tasks from OpenAI Gym [6] to evaluate the SD3 algorithm. Results show that SD3 outperforms state-of-the-art methods including TD3 and Soft Actor-Critic (SAC) [19] with minimal additional computation cost.

2 Preliminaries

The reinforcement learning problem can be formulated by a Markov decision process (MDP) defined as a 5-tuple $(\mathcal{S}, \mathcal{A}, r, p, \gamma)$, with $\mathcal{S}$ and $\mathcal{A}$ denoting the set of states and actions, r the reward function, p the transition probability, and γ the discount factor. We consider a continuous action space, and assume it is bounded. We also assume the reward function r is continuous and bounded, where the assumption is also required in [32]. In continuous action space, taking the $\max$ operator over $\mathcal{A}$ as in Q-learning [38] can be expensive. DDPG [25] extends Q-learning to continuous control based on the Deterministic Policy Gradient [32] algorithm, which learns a deterministic policy $\pi(s;\phi)$ parameterized by ϕ to maximize the Q-function to approximate the $\max$ operator. The objective is to maximize the expected long-term rewards $J(\pi(\cdot;\phi)) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r(s_k, a_k) | s_0, a_0, \pi(\cdot;\phi)]$. Specifically, DDPG updates the policy by the deterministic policy gradient, i.e.,

$$\nabla_\phi J(\pi(\cdot;\phi)) = \mathbb{E}_s \left[\nabla_\phi (\pi(s;\phi)) \nabla_a Q(s, a; \theta)|_{a=\pi(s;\phi)} \right], \tag{1}$$

where $Q(s, a; \theta)$ is the Q-function parameterized by θ which approximates the true parameter θ^{true}. We let $\mathcal{T}(s')$ denote the value estimation function, which is used to estimate the target Q-value $r + \gamma \mathcal{T}(s')$ for state s'. Then, we see that DDPG updates its critic according to $\theta' = \theta + \alpha \mathbb{E}_{s,a\sim\rho} \left(r + \gamma \mathcal{T}_{\text{DDPG}}(s') - Q(s, a; \theta) \right) \nabla_\theta Q(s, a; \theta)$, where $\mathcal{T}_{\text{DDPG}} = Q(s', \pi(s'; \phi^-); \theta^-)$, ρ denotes the sample distribution from the replay buffer, α is the learning rate, and ϕ^-, θ^- denote parameters of the target networks for the actor and critic respectively.

3 Analysis of the Softmax Operator in Continuous Action Space

In this section, we theoretically analyze the softmax operator in continuous action space by studying the performance bound of value iteration under the operator.

The softmax operator in continuous action space is defined by $\text{softmax}_\beta(Q(s,\cdot)) = \int_{a\in\mathcal{A}} \frac{\exp(\beta Q(s,a))}{\int_{a'\in\mathcal{A}}\exp(\beta Q(s,a'))da'} Q(s,a)da$, where β is the parameter of the softmax operator.

In Theorem 1, we provide an $O(1/\beta)$ upper bound for the difference between the max and softmax operators. The result is helpful for deriving the error bound in value iteration with the softmax operator in continuous action space. The proof of Theorem 1 is in Appendix A.1.

Theorem 1 *Let* $\mathcal{C}(Q,s,\epsilon) = \{a|a\in\mathcal{A}, Q(s,a)\geq \max_a Q(s,a)-\epsilon\}$ *and* $F(Q,s,\epsilon) = \int_{a\in\mathcal{C}(Q,s,\epsilon)} 1da$ *for any* $\epsilon > 0$ *and any state* s*. The difference between the max operator and the softmax operator is* $0 \leq \max_a Q(s,a) - \text{softmax}_\beta(Q(s,\cdot)) \leq \frac{\int_{a\in\mathcal{A}} 1da - 1 - \ln F(Q,s,\epsilon)}{\beta} + \epsilon$.

Remark. A recent work [33] studies the distance between the two operators in the discrete setting. However, the proof technique in [33] is limited to discrete action space. This is because applying the technique requires that for any state s, the set of the maximum actions at s with respect to the Q-function $Q(s,a)$ covers a continuous set, which often only holds in special cases in the setting with continuous action. In this case, ϵ can be 0 and the bound still holds, which turns into $\frac{\int_{a\in\mathcal{A}} 1da - 1 - \ln F(Q,s,0)}{\beta}$. Note that when $Q(s,a)$ is a constant function with respect to a, the upper bound in Theorem 1 will be 0, where the detailed discussion is in Appendix A.1.

Now, we formally define value iteration with the softmax operator by $Q_{t+1}(s,a) = r_t(s,a) + \gamma\mathbb{E}_{s'\sim p(\cdot|s,a)}[V_t(s')]$, $V_{t+1}(s) = \text{softmax}_\beta(Q_{t+1}(s,\cdot))$, which updates the value function using the softmax operator iteratively. In Theorem 2, we provide an error bound between the value function under the softmax operator and the optimal in continuous action space, where the proof can be found in Appendix A.2.

Theorem 2 *For any iteration* t*, the difference between the optimal value function* V^* *and the value function induced by softmax value iteration at the* t*-th iteration* V_t *satisfies:*

$$||V_t - V^*||_\infty \leq \gamma^t ||V_0(s) - V^*(s)||_\infty + \frac{1}{1-\gamma}\frac{\beta\epsilon + \int_{a\in\mathcal{A}} 1da - 1}{\beta} - \sum_{k=1}^{t}\gamma^{t-k}\frac{\min_s \ln F(Q_k,s,\epsilon)}{\beta}.$$

Therefore, for any $\epsilon > 0$, the error between the value function induced by the softmax operator and the optimal can be bounded, which converges to $\epsilon/(1-\gamma)$, and can be arbitrarily close to 0 as β approaches to infinity. Theorem 2 paves the way for the use of the softmax operator for value function updates in continuous action space, as the error can be controlled in a reasonable scale.

4 The Softmax Operator in Actor-Critic

In this section, we propose to employ the softmax operator for value function estimation in standard actor-critic algorithms with single estimator and double estimators. We first show that the softmax operator can smooth the optimization landscape and help learning empirically. Then, we show that it enables a better estimation of the value function, which effectively improves the overestimation and underestimation bias when built upon single and double estimators respectively.

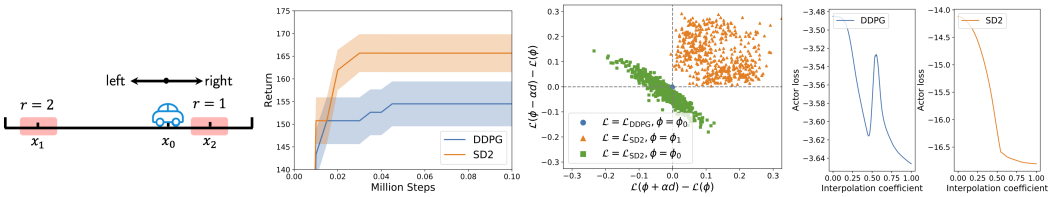
4.1 The Softmax Operator Helps to Smooth the Optimization Landscape

We first show that the softmax operator can help to smooth the optimization landscape. For simplicity, we showcase the smoothing effect based on a comparative study of DDPG and our new SD2 algorithm (to be introduced in Section 4.2). SD2 is a variant of DDPG that leverages the softmax operator to update the value function, which is the only difference between the two algorithms. We emphasize that the smoothing effect is attributed to the softmax operator, and also holds for our proposed SD3 algorithm (to be introduced in Section 4.3), which uses the operator to estimate value functions.[1]

We design a toy 1-dimensional, continuous state and action environment MoveCar (Figure 1(a)) to illustrate the effect. The car always starts at position $x_0 = 8$, and can take actions ranging in $[-1.0, 1.0]$ to move left or right, where the left and right boundaries are 0 and 10. The rewards are

[1] See Appendix B.1 for the comparative study on the smoothing effect of TD3 and SD3.

$+2$, $+1$ in neighboring regions centered at $x_1 = 1$ and $x_2 = 9$, respectively, with the length of the neighboring region to be 1. In other positions, the reward is 0. The episode length is 100 steps. We run DDPG and SD2 on MoveCar for 100 independent runs. To exclude the effect of the exploration, we add a gaussian noise with the standard deviation to be high enough to the action during training, and both two algorithms collects diverse samples in the warm-up phase where actions are sampled from a uniform distribution. More details about the experimental setup are in Appendix B.2. The performance result is shown in Figure 1(b), where the shaded area denotes half a standard deviation for readability. As shown, SD2 outperforms DDPG in final performance and sample efficiency.

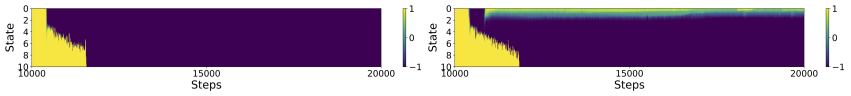


(a) MoveCar. (b) Performance. (c) Scatter plot. (d) Linear interpolation.

Figure 1: Analysis of smoothing effect in the MoveCar environment.

We investigate the optimization landscape based on the visualization technique proposed in [2]. According to Eq. (1), we take the loss function of the actor $\mathcal{L}(\phi) = \mathbb{E}_{s\sim\rho}\left[-Q(s, \pi(s;\phi);\theta)\right]$ as the objective function in our analysis. To understand the local geometry of the actor losses $\mathcal{L}_{\text{DDPG}}$ and $\mathcal{L}_{\text{SD2}}$, we randomly perturb the corresponding policy parameters ϕ_0 and ϕ_1 learned by DDPG and SD2 during training from a same random initialization. Specifically, the key difference between the two parameters is that ϕ_0 takes an action to move left in locations $[0, 0.5]$, while ϕ_1 determines to move right. Thus, ϕ_1 are better parameters than ϕ_0. The random perturbation is obtained by randomly sampling a batch of directions d from a unit ball, and then perturbing the policy parameters in positive and negative directions by $\phi \pm \alpha d$ for some value α. Then, we evaluate the difference between the perturbed loss functions and the original loss function, i.e., $\mathcal{L}(\phi \pm \alpha d) - \mathcal{L}(\phi)$.

Figure 1(c) shows the scatter plot of random perturbation. For DDPG, the perturbations for its policy parameters ϕ_0 are close to zero (blue circles around the origin). This implies that there is a flat region in $\mathcal{L}_{\text{DDPG}}$, which can be difficult for gradient-based methods to escape from [12]. Figure 2(a) shows that the policy of DDPG converges to always take action -1 at each location. On the other hand, as all perturbations around the policy parameters ϕ_1 of SD2 with respect to its corresponding loss function $\mathcal{L}_{\text{SD2}}$ are positive (orange triangles), the point ϕ_1 is a local minimum. Figure 2(b) confirms that SD2 succeeds to learn an optimal policy to move the agent to high-reward region $[0.5, 1.5]$. To illustrate the critical effect of the softmax operator on the objective, we also evaluate the change of the loss function $\mathcal{L}_{\text{SD2}}(\phi_0)$ of SD2 with respect to parameters ϕ_0 from DDPG. Figure 1(c) shows that ϕ_0 is in an almost linear region under $\mathcal{L}_{\text{SD2}}$ (green squares), and the loss can be reduced following several directions, which demonstrates the benefits of optimizing the softmax version of the objective.



(a) Learned policy of DDPG. (b) Learned policy of SD2.

Figure 2: Policies of DDPG and SD2 during learning for each state (y-axis) at each step (x-axis).

To further analyze the difficulty of the optimization of $\mathcal{L}_{\text{DDPG}}$ on a more global view, we linearly interpolate between the parameters of the policies from DDPG and SD2, i.e., $\alpha\phi_0 + (1-\alpha)\phi_1$ $(0 \leq \alpha \leq 1)$ as in [16, 2]. Figure 1(d) illustrates the result with varying values of α. As shown, there exists at least a monotonically decreasing path in the actor loss of SD2 to a good solution. As a result, the smoothing effect of the softmax operator on the optimization landscape can help learning and reduce the number of local optima, and makes it less sensitive to different initialization.

4.2 Softmax Deep Deterministic Policy Gradients (SD2)

Here we present the design of SD2, which is short for Softmax Deep Deterministic Policy Gradients, where we build the softmax operator upon DDPG [32] with a single critic estimator.

Specifically, SD2 estimates the value function using the softmax operator, and the update of the critic of SD2 is defined by Eq. (2), where the the actor aims to optimize a soft estimation of the return.

$$\theta' = \theta + \alpha \mathbb{E}_{s,a\sim\rho} \left(r + \gamma \mathcal{T}_{\mathrm{SD2}}(s') - Q\left(s, a; \theta\right)\right) \nabla_\theta Q(s, a; \theta). \tag{2}$$

In Eq. (2), $\mathcal{T}_{\mathrm{SD2}}(s') = \mathrm{softmax}_\beta(Q(s', \cdot; \theta^-))$. However, the softmax operator involves the integral, and is intractable in continuous action space. We express the Q-function induced by the softmax operator in expectation by importance sampling [18], and obtain an unbiased estimation by

$$\mathbb{E}_{a'\sim p}\left[\frac{\exp(\beta Q(s', a'; \theta^-))Q(s', a'; \theta^-)}{p(a')}\right] / \mathbb{E}_{a'\sim p}\left[\frac{\exp(\beta Q(s', a'; \theta^-))}{p(a')}\right], \tag{3}$$

where $p(a')$ denotes the probability density function of a Gaussian distribution. In practice, we sample actions obtained by adding noises which are sampled from a Gaussian distribution $\epsilon \sim \mathcal{N}(0, \sigma)$ to the target action $\pi(s'; \phi^-)$, i.e., $a' = \pi(s'; \phi^-) + \epsilon$. Here, each sampled noise is clipped to $[-c, c]$ to ensure the sampled action is in $\mathcal{A}_c = [-c + \pi(s'; \phi^-), c + \pi(s'; \phi^-)]$. This is because directly estimating $\mathcal{T}_{\mathrm{SD2}}(s')$ can incur large variance as $1/p(a')$ can be very large. Therefore, we limit the range of the action set to guarantee that actions are close to the original action, and that we obtain a robust estimate of the softmax Q-value. Due to space limitation, we put the full SD2 algorithm in Appendix C.1.

4.2.1 SD2 Reduces the Overestimation Bias

Besides the smoothing effect on the optimization landscape, we show in Theorem 3 that SD2 enables a better value estimation by reducing the overestimation bias in DDPG, for which it is known that the critic estimate can cause significant overestimation [15], where the proof is in Appendix C.2.

Theorem 3 *Denote the bias of the value estimate and the true value induced by* $\mathcal{T}$ *as* $\mathrm{bias}(\mathcal{T}) = \mathbb{E}\left[\mathcal{T}(\mathrm{s}')\right] - \mathbb{E}\left[\mathrm{Q}(\mathrm{s}', \pi(\mathrm{s}'; \phi^-); \theta^{\mathrm{true}})\right]$. *Assume that the actor is a local maximizer with respect to the critic, then there exists noise clipping parameter* $c > 0$ *such that* $\mathrm{bias}(\mathcal{T}_{\mathrm{SD2}}) \leq \mathrm{bias}(\mathcal{T}_{\mathrm{DDPG}})$.

We validate the reduction effect in two MuJoCo [36] environments, Hopper-v2 and Walker2d-v2, where the experimental setting is the same as in Section 5. Figure 3 shows the performance comparison between DDPG and SD2, where the shaded area corresponds to standard deviation. The red horizontal line denotes the maximum return obtained by DDPG in evaluation during training, while the blue vertical lines show the number of steps for DDPG and SD2 to reach that score. As shown in Figure 3, SD2 significantly outperforms DDPG in sample efficiency and final performance. Estimation of value functions is shown in Figure 4(a), where value estimates are averaged over 1000 states sampled from the replay buffer at each timestep, and true values are estimated by averaging the discounted long-term rewards obtained by rolling out the current policy starting from the sampled states at each timestep. The bias of corresponding value estimates and true values is shown in Figure 4(b), where it can be observed that SD2 reduces overestimation and achieves a better estimation of value functions.

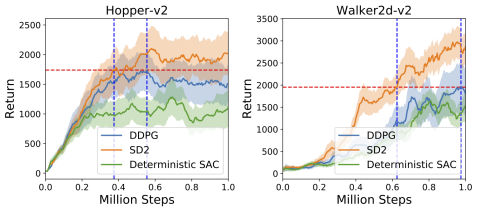
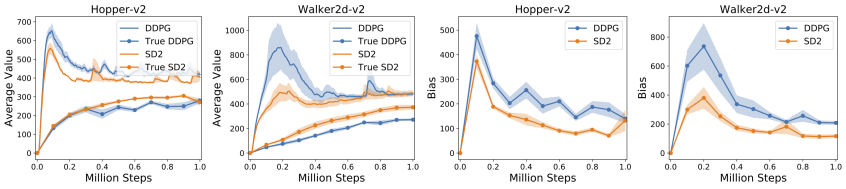


Figure 3: Performance comparison of DDPG and SD2, and Deterministic SAC.

Regarding the softmax operator in SD2, one may be interested in comparing it with the log-sum-exp operator applied in SAC [19]. To study the effect of different operators, we compare SD2 with a variant of SAC with deterministic policy and single critic for fair comparison. The performance of Deterministic SAC (with fine-tuned parameter of log-sum-exp) is shown in Figure 3, which underperforms DDPG and SD2, where we also observe that its absolute bias is larger than that of DDPG, and worsens the overestimation problem. Its value estimates can be found in Appendix C.3.



(a) Value estimates and true values. (b) Bias of value estimations.

Figure 4: Comparison of estimation of value functions of DDPG and SD2.

4.3 Softmax Deep Double Deterministic Policy Gradients (SD3)

In Section 4.2.1, we have analyzed the effect of the softmax operator in the aspect of value estimation based on DDPG which suffers from overestimation. We now investigate whether the softmax operator is still beneficial when there is underestimation bias. We propose a novel method to leverage the softmax operator with double estimators, called Softmax Deep Double Deterministic Policy Gradients (SD3). We show that SD3 enables a better value estimation in comparison with the state-of-the-art TD3 algorithm, which can suffer from a large underestimation bias.

TD3 [15] maintains a pair of critics as in Double Q-learning [20], which is a promising approach to alleviate overestimation in the value-based Q-learning [38] method. However, directly applying Double Q-learning still leads to overestimation in the actor-critic setting. To avoid the problem, Clipped Double Q-learning is proposed in TD3 [15], which clips the Q-value from the double estimator of the critic by the original Q-value itself. Specifically, TD3 estimates the value function by taking the minimum of value estimates from the two critics according to $y_1, y_2 = r + \gamma \min_{i=1,2} Q_i(s', \pi(s'; \phi^-); \theta_i^-)$. Nevertheless, it may incur large underestimation bias, and can affect performance [24, 10].

We propose to use the softmax operator based on double estimators to address the problem. It is worth noting that a direct way to combine the softmax operator with TD3, i.e., apply the softmax operator to the Q-value from the double critic estimator and then clip it by the original Q-value, as in Eq. (4) is ineffective.

$$y_i = r + \gamma \min\left(\mathcal{T}_{\mathrm{SD2}}^{-i}(s'), Q_i(s', \pi(s'; \phi^-); \theta_i^-)\right), \quad \mathcal{T}_{\mathrm{SD2}}^{-i}(s') = \mathrm{softmax}_\beta(Q_{-i}(s', \cdot; \theta_{-i}^-)). \tag{4}$$

This is because according to Theorem 3, we have $\mathcal{T}_{\mathrm{SD2}}^{-i}(s') \leq Q_{-i}(s', \pi(s'; \phi^-); \theta_{-i}^-)$, then the value estimates result in even larger underestimation bias compared with TD3. To tackle the problem, we propose to estimate the target value for critic Q_i by $y_i = r + \gamma \mathcal{T}_{\mathrm{SD3}}(s')$, where

$$\mathcal{T}_{\mathrm{SD3}}(s') = \mathrm{softmax}_\beta\left(\hat{Q}_i(s', \cdot)\right), \quad \hat{Q}_i(s', a') = \min\left(Q_i(s', a'; \theta_i^-), Q_{-i}(s', a'; \theta_{-i}^-)\right). \tag{5}$$

Here, target actions for computing the softmax Q-value are obtained by the same way as in the SD2 algorithm in Section 4.2. The full SD3 algorithm is shown in Algorithm 1.

4.3.1 SD3 Improves the Underestimation Bias

In Theorem D.1, we present the relationship between the value estimation of SD3 and that of TD3, where the proof is in Appendix D.1.

Theorem 4 *Denote $\mathcal{T}_{\mathrm{TD3}}$, $\mathcal{T}_{\mathrm{SD3}}$ the value estimation functions of TD3 and SD3 respectively, then we have* $\mathrm{bias}(\mathcal{T}_{\mathrm{SD3}}) \geq \mathrm{bias}(\mathcal{T}_{\mathrm{TD3}})$.

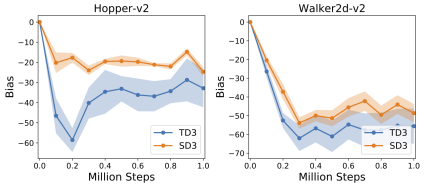


Figure 5: Comparison of the bias of value estimations of TD3 and SD3.

As illustrated in Theorem D.1, the value estimation of SD3 is larger than that of TD3. As TD3 leads to an underestimation value estimate [15], we get that SD3 helps to improve the underestimation bias of TD3. Therefore, according to our SD2 and SD3 algorithms, we conclude that the softmax operator can not only reduce the overestimation bias when built upon DDPG, but also improve the underestimation bias when built upon TD3. We empirically validate the theorem using the same two MuJoCo environments and estimation of value

Algorithm 1 SD3

Initialize critic networks Q_1, Q_2, and actor networks π_1, π_2 with random parameters $\theta_1, \theta_2, \phi_1, \phi_2$
Initialize target networks $\theta_1^- \leftarrow \theta_1$, $\theta_2^- \leftarrow \theta_2$, $\phi_1^- \leftarrow \phi_1$, $\phi_1^- \leftarrow \phi_1$
Initialzie replay buffer $\mathcal{B}$
for $t = 1$ to T **do**
  Select action a with exploration noise $\epsilon \sim \mathcal{N}(0, \sigma)$ based on π_1 and π_2
  Execute action a, observe reward r, new state s' and done d
  Store transition tuple (s, a, r, s', d) in $\mathcal{B}$ // d is the done flag
  for $i = 1, 2$ **do**
    Sample a mini-batch of N transitions $\{(s, a, r, s', d)\}$ from $\mathcal{B}$
    Sample K noises $\epsilon \sim \mathcal{N}(0, \bar{\sigma})$
    $\hat{a}' \leftarrow \pi_i(s'; \phi_i^-) + \text{clip}(\epsilon, -c, c)$
    $\hat{Q}(s', \hat{a}') \leftarrow \min_{j=1,2} \left(Q_j(s', \hat{a}'; \theta_j^-)\right)$
    $\text{softmax}_\beta \left(\hat{Q}(s', \cdot)\right) \leftarrow \mathbb{E}_{\hat{a}' \sim p} \left[\frac{\exp(\beta \hat{Q}(s', \hat{a}')) \hat{Q}(s', \hat{a}')}{p(\hat{a}')}\right] / \mathbb{E}_{\hat{a}' \sim p} \left[\frac{\exp(\beta \hat{Q}(s', \hat{a}'))}{p(\hat{a}')}\right]$
    $y_i \leftarrow r + \gamma(1 - d)\text{softmax}_\beta \left(\hat{Q}(s', \cdot)\right)$
    Update the critic θ_i according to Bellman loss: $\frac{1}{N} \sum_s (Q_i(s, a; \theta_i) - y_i)^2$
    Update actor ϕ_i by policy gradient: $\frac{1}{N} \sum_s \left[\nabla_{\phi_i}(\pi(s; \phi_i)) \nabla_a Q_i(s, a; \theta_i)|_{a=\pi(s;\phi_i)}\right]$
    Update target networks: $\theta_i^- \leftarrow \tau\theta_i + (1 - \tau)\theta_i^-$, $\phi_i^- \leftarrow \tau\phi_i + (1 - \tau)\phi_i^-$
  end for
end for

functions and true values as in Section 4.2.1. Comparison of the bias of value estimates and true values is shown in Figure 5, where the performance comparison is in Figure 8. As shown, SD3 enables better value estimations as it achieves smaller absolute bias than TD3, while TD3 suffers from a large underestimation bias. We also observe that the variance of value estimates of SD3 is smaller than that of TD3.

5 Experiments

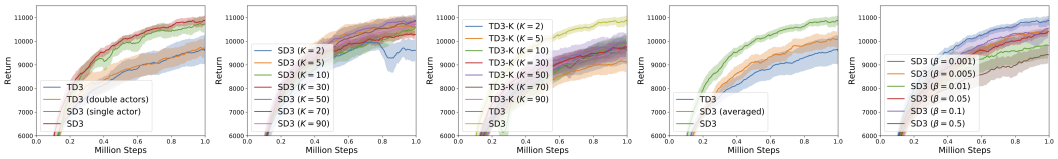
In this section, we first conduct an ablation study on SD3, from which we aim to obtain a better understanding of the effect of each component, and to further analyze the main driver of the performance improvement of SD3. Then, we extensively evaluate the SD3 algorithm on continuous control benchmarks and compare with state-of-the-art methods.

We conduct experiments on continuous control tasks from OpenAI Gym [6] simulated by MuJoCo [36] and Box2d [8]. We compare SD3 with DDPG [25] and TD3 [15] using authors' open-sourced implementation [14]. We also compare SD3 against Soft Actor-Critic (SAC) [19], a state-of-the-art method that also uses double critics. Each algorithm is run with 5 seeds, where the performance is evaluated for 10 times every 5000 timesteps. SD3 uses double actors and double critics based on the structure of Double Q-learning [20], with the same network configuration as the default TD3 and DDPG baselines. For the softmax operator in SD3, the number of noises to sample K is 50, and the parameter β is mainly chosen from $\{10^{-3}, 5 \times 10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}, 5 \times 10^{-1}\}$ using grid search. All other hyperparameters of SD3 are set to be the same as the default setting for TD3 on all tasks except for Humanoid-v2, as TD3 with the default hyperparameters almost fails in Humanoid-v2. To better demonstrate the effectiveness of SD3, we therefore employ the fine-tuned hyperparameters provided by authors of TD3 [14] for Humanoid-v2 for DDPG, TD3 and SD3. Details for hyperparameters are in Appendix E.1, and the implementation details are publicly available at `https://github.com/ling-pan/SD3`.

5.1 Ablation Study

We first conduct an ablative study of SD3 in an MuJoCo environment HalfCheetah-v2 to study the effect of structure and important hyperparameters.

Structure. From Figure 6(a), we find that for SD3 and TD3, using double actors outperforms its counterpart with a single actor. This is because using a single actor as in TD3 leads to a same training target for both critics, which can be close during training and may not fully utilize the double



(a) (b) (c) (d) (e)

Figure 6: Ablation study on HalfCheetah-v2 (mean $\pm$ standard deviation). (a) Structure. (b) Number of noises K. (c) Comparison with TD3-K. (d) Comparison with SD3 (averaged). (e) Parameter β.

estimators. However, TD3 with double actors still largely underperforms SD3 (either with single or double actors).

The number of noises K. Figure 6(b) shows the performance of SD3 with varying number of noise samples K. The performance of all K values is competitive except for $K = 2$, where it fails to behave stable and also underperforms other values of K in sample efficiency. As SD3 is not sensitive to this parameter, we fix K to be 50 in all environments as it performs best. Note that doing so does not incur much computation cost as setting K to be 50 only takes 3.28% more runtime on average compared with $K = 1$ (in this case the latter can be viewed as a variant of TD3 with double actors).

The effect of the softmax operator. It is also worth studying the performance of a variant of TD3 using K samples of actions to evaluate the Q-function (TD3-K). Specifically, TD3-K samples K actions by the same way as in SD3 to compute Q-values before taking the $\min$ operation (details are in Appendix E.2). As shown in Figure 6(c), TD3-K outperforms TD3 for some large values of K, but only by a small margin and still underperforms SD3. We also compare SD3 with its variant SD3 (averaged) that directly averages the K samples to compute the Q-function, which underperforms SD3 by a large margin as shown in Figure 6(d). Results confirm that the softmax operator is the key factor for the performance improvement for SD3 instead of other changes (multiple samples).

The parameter β. The parameter β of the softmax operator directly affects the estimation of value functions, and controls the bias of value estimations, which is a critical parameter for the performance. A smaller β leads to lower variance while a larger β results in smaller bias. Indeed, there is an intermediate value that performs best that can best provide the trade-off as in Figure 6(e).

5.2 Performance Comparison

The performance comparison is shown in Figure 8, where we report the averaged performance as the solid line, and the shaded region denotes the standard deviation. As demonstrated, SD3 significantly outperforms TD3, where it achieves a higher final performance and is more stable due to the smoothing effect of the softmax operator on the optimization landscape and a better value estimation. Figure 7 shows the number of steps for TD3 and SD3 to reach the highest score of TD3 during training. We observe that SD3 learns much faster than TD3. It is worth noting that SD3 outperforms SAC in most environments except for Humanoid-v2, where both algorithms are competitive.

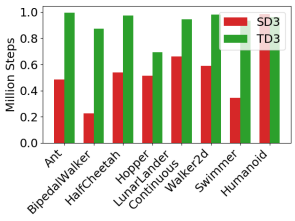


Figure 7: Sample efficiency comparison.

6 Related Work

How to obtain good value estimation is an important problem in reinforcement learning, and has been extensively investigated in discrete system control for deep Q-network (DQN) [20, 37, 3, 33, 24, 29]. Ensemble-DQN [3] leverages an ensemble of Q-networks which can reduce variance while Averaged-DQN [3] uses previously learned Q-value estimates by averaging them to lower value estimations. Lan et al. [24] propose to use an ensemble scheme to control the bias of value estimates for DQN [27]. In [33], Song et al. apply the softmax operator for discrete control in DQNs, and validate the performance gain by showing that softmax can reduce overestimation and gradient noise in DQN. In [29], Pan et al. propose a convergent variant of the softmax operator for discrete control. In this paper, our focus is to investigate the properties and benefits of the softmax operator in continuous control,

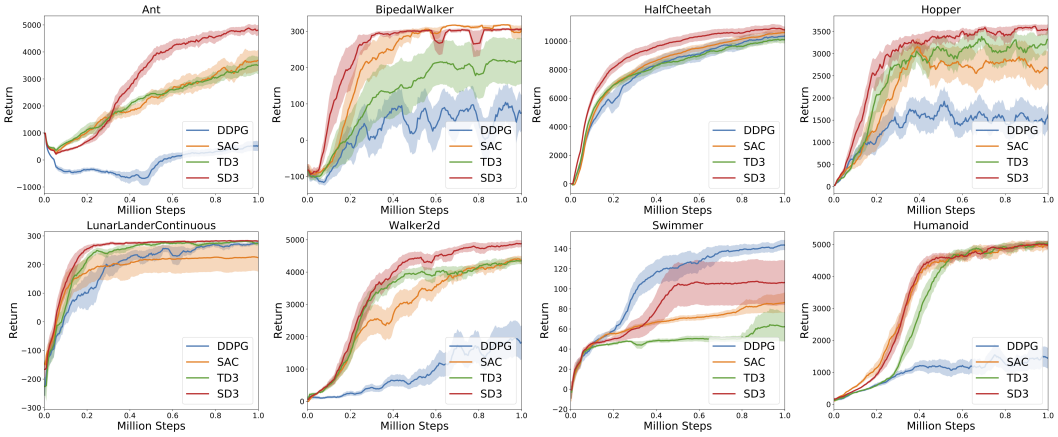


Figure 8: Performance comparison in MuJoCo environments.

where we provide new analysis and insights. TD3 [15] is proposed to tackle the overestimation problem in continuous action space. However, it can suffer from a large underestimation problem, which is a focus of this work. There are several works that build on and improve DDPG including prioritized experience replay [21], distributional [5], model-based [17, 7, 13], evolution methods [23], etc. Prior works [11, 28] generalize DDPG for learning a stochastic Gaussian policy, while we uncover and study benefits of softmax operator with deterministic policy. Achiam et al. [1] study the divergence problem of deep Q-learning, and propose PreQN to ensure that the value function update is non-expansive. However, PreQN can be computationally expensive, while SD3 is efficient.

7 Conclusion

In this paper, we show that it is promising to use the softmax operator in continuous control. We first provide a new analysis for the error bound between the value function induced by the softmax operator and the optimal in continuous control. We then show that the softmax operator (i) helps to smooth the optimization landscape, (ii) can reduce the overestimation bias and improve performance of DDPG when combined with single estimator (SD2), and (iii) can also improve the underestimation bias of TD3 when built upon double estimators (SD3). Extensive experimental results on standard continuous control benchmarks validate the effectiveness of the SD3 algorithm, which significantly outperforms state-of-the-art algorithms. For future work, it is interesting to study an adaptive scheduling of the parameter β in SD2 and SD3. In addition, it also worths to quantify the bias reduction for overestimation and underestimation. It will also be an interesting direction to unify SD2 and SD3 into a same framework to study the effect on value estimations.

Acknowledgments

We thank the anonymous reviewers for their valuable feedbacks and suggestions. The work of Ling Pan and Longbo Huang was supported in part by the National Natural Science Foundation of China Grant 61672316, the Zhongguancun Haihua Institute for Frontier Information Technology and the Turing AI Institute of Nanjing.

Broader Impact

Recent years have witnessed unprecedented advances of deep reinforcement learning in real-world tasks involving high-dimensional state and action spaces that leverages the power of deep neural networks including robotics, transportation, recommender systems, etc. Our work investigates the Boltzmann softmax operator in updating value functions in reinforcement learning for continuous control, and provides new insights and further understanding of the operator. We show that the error bound of the value function under the softmax operator and the optimal can be bounded and it is promising to use the softmax operator in continuous control. We demonstrate the smoothing

effect of the softmax operator on the optimization landscape, and shows that it can provide better value estimations. Experimental results show the potential of our proposed algorithm to improve final performance and sample efficiency. It will be interesting to apply our algorithm in practical applications.

A Proofs in Section 4

A.1 Proof of Theorem 1

Theorem 1 *Let* $\mathcal{C}(Q,s,\epsilon) = \{a|a\in\mathcal{A}, Q(s,a)\geq \max_a Q(s,a)-\epsilon\}$ *and* $F(Q,s,\epsilon) = \int_{a\in\mathcal{C}(Q,s,\epsilon)} 1da$ *for any* $\epsilon > 0$ *and any state* s*. The difference between the max operator and the softmax operator satisfies*

$$0 \leq \max_a Q(s,a) - \mathrm{softmax}_\beta\left(Q(s,\cdot)\right) \leq \frac{\int_{a\in\mathcal{A}} 1da - 1 - \ln F(Q,s,\epsilon)}{\beta} + \epsilon. \tag{6}$$

Proof. For the left-hand-side, we have by definition that

$$\mathrm{softmax}_\beta\left(Q(s,\cdot)\right) \leq \int_{a\in\mathcal{A}} \frac{\exp(\beta Q(s,a))}{\int_{a'\in\mathcal{A}}\exp(\beta Q(s,a'))da'} \max_{a'} Q(s,a')da \leq \max_a Q(s,a). \tag{7}$$

For the right-hand-side, we first provide a relationship between the softmax operator and the log-sum-exp operator $\mathrm{lse}_\beta\left(Q(s,\cdot)\right)$ (Eq. (9) in [18]) in continuous action spaces, i.e., $\mathrm{lse}_\beta\left(Q(s,\cdot)\right) = \frac{1}{\beta}\ln\int_{a\in\mathcal{A}}\exp(\beta Q(s,a))da$.

Denote the probability density function of the softmax distribution by $p_\beta(s,a) = \frac{\exp(\beta Q(s,a))}{\int_{a'\in\mathcal{A}}\exp(\beta Q(s,a'))da'}$. We have

$$\begin{aligned}
&\mathrm{lse}_\beta\left(Q(s,\cdot)\right) - \mathrm{softmax}_\beta\left(Q(s,\cdot)\right)\\
=&\frac{\ln\int_{a\in\mathcal{A}}\exp(\beta Q(s,a))da}{\beta} - \int_{a\in\mathcal{A}} p_\beta(s,a)Q(s,a)da\\
=&\frac{\int_{a\in\mathcal{A}} p_\beta(s,a)(\ln\int_{a'\in\mathcal{A}}\exp(\beta Q(s,a'))da')da}{\beta} - \frac{\int_{a\in\mathcal{A}} p_\beta(s,a)\beta Q(s,a)da}{\beta}\\
=&\frac{\int_{a\in\mathcal{A}} -p_\beta(s,a)\ln p_\beta(s,a)da}{\beta}.
\end{aligned} \tag{8}$$

As $p_\beta(s,a)$ is non-negative, we have that $\forall a, -p_\beta(s,a)\ln p_\beta(s,a) \leq 1 - p_\beta(s,a)$.

Note that $\int_{a\in\mathcal{A}} p_\beta(s,a)da = 1$. We have

$$\mathrm{lse}_\beta\left(Q(s,\cdot)\right) - \mathrm{softmax}_\beta\left(Q(s,\cdot)\right) \leq \frac{\int_{a\in\mathcal{A}} 1da - 1}{\beta}. \tag{9}$$

Secondly, by the definition of the log-sum-exp operator and the fact that $\mathcal{C}(Q,s,\epsilon)$ is a subset of $\mathcal{A}$, we have

$$\begin{aligned}
\mathrm{lse}_\beta\left(Q(s,\cdot)\right) &= \frac{\ln\int_{a\in\mathcal{A}}\exp(\beta Q(s,a))da}{\beta}\\
&\geq \frac{\ln\int_{a\in\mathcal{C}(Q,s,\epsilon)}\exp(\beta Q(s,a))da}{\beta}\\
&\geq \frac{\ln\int_{a\in\mathcal{C}(Q,s,\epsilon)}\exp\left(\beta(\max_a Q(s,a)-\epsilon)\right)da}{\beta}\\
&\geq \frac{\ln F(Q,s,\epsilon) + \beta\left(\max_a Q(s,a)-\epsilon\right)}{\beta}.
\end{aligned} \tag{10}$$

As a result, we get the inequality of the max operator and the log-sum-exp operator as in Eq. (11).

$$\mathrm{lse}_\beta\left(Q(s,\cdot)\right) \geq \max_a Q(s,a) + \frac{\ln F(Q,s,\epsilon) - \beta\epsilon}{\beta}. \tag{11}$$

Finally, combining Eq. (9) and Eq. (11), we obtain Eq. (6). □

Remark. In a special case where for any state s, the set of the maximum actions at s with respect to the Q-function $Q(s,a)$ covers a continuous set, ϵ can be 0 and the upper bound in Eq. (6) still holds as $F(Q,s,0) > 0$, which turns into $\frac{\int_{a\in\mathcal{A}} 1da - 1 - \ln F(Q,s,0)}{\beta}$. Please also note that when $Q(s,a)$ is a constant function w.r.t. a, the upper bound will be 0 in this case as from Eq. (8), we get that $\mathrm{lse}_\beta\left(Q(s,\cdot)\right) - \mathrm{softmax}_\beta\left(Q(s,\cdot)\right) = \frac{\ln \int_{a\in\mathcal{A}} 1da}{\beta}$ and $\mathrm{lse}_\beta\left(Q(s,\cdot)\right) \geq \max_a Q(s,a) + \frac{\ln \int_{a\in\mathcal{A}} 1da}{\beta}$.

A.1.1 Discussion of Theorem 1 and Results in [33].

A recent work [33] studies the distance between the max and the softmax operators in the discrete setting. However, the proof technique in [33] is limited to discrete action space, and cannot be naturally extended to continuous action space. Specifically, following the line of analysis in [33], the gap between the max operator and the softmax operator is given by

$$\frac{\int_{a\in\mathcal{A}} \frac{1}{D}\exp(-\beta\delta(s,a))\delta(s,a)da}{\int_{a\in\mathcal{A}} \frac{1}{D}\exp(-\beta\delta(s,a))da} = \frac{\int_{a\in\mathcal{A}-\mathcal{A}_m} \frac{1}{D}\exp(-\beta\delta(s,a))\delta(s,a)da}{c + \int_{a\in\mathcal{A}-\mathcal{A}_m} \frac{1}{D}\exp(-\beta\delta(s,a))da}, \tag{12}$$

where $\mathcal{A}_m$ is the set of actions where the Q-function $Q(s,\cdot)$ attains the maximum value, $\delta(s,a) = \max_{a'} Q(s,a') - Q(s,a)$, $D = \int_{a\in\mathcal{A}} 1da$, and $c = \int_{a\in\mathcal{A}} \frac{I_{a\in\mathcal{A}_m}}{D} da$, where $I_{a\in\mathcal{A}_m}$ is the indicator function of event $\{a \in \mathcal{A}_m\}$. Please note that the analysis in [33] requires that $c > 0$, which does not always hold in the continuous case. As a result, the proof technique in [33] cannot be naturally extended to the continuous action setting, and we provide a new and different analysis in Theorem 1.

A.2 Proof of Theorem 2

Theorem 2 *For any iteration t, the difference between the optimal value function V^* and the value function induced by softmax value iteration at the t-th iteration V_t satisfies:*

$$||V_t - V^*||_\infty \leq \gamma^t ||V_0(s) - V^*(s)||_\infty + \frac{1}{1-\gamma}\frac{\beta\epsilon + \int_{a\in\mathcal{A}} 1da - 1}{\beta} - \sum_{k=1}^{t} \gamma^{t-k} \frac{\min_s \ln F(Q_k,s,\epsilon)}{\beta}. \tag{13}$$

Proof. By the definition of softmax value iteration, we get

$$\begin{aligned} &|V_{t+1}(s) - V^*(s)| \\ =&|\mathrm{softmax}_\beta\left(Q_{t+1}(s,\cdot)\right) - \max_a Q^*(s,a)| \\ \leq&|\mathrm{softmax}_\beta\left(Q_{t+1}(s,\cdot)\right) - \max_a Q_{t+1}(s,a)| + |\max_a Q_{t+1}(s,a) - \max_a Q^*(s,a)|. \end{aligned} \tag{14}$$

According to Theorem 1 and the fact that the max operator is non-expansive [22], we have

$$|V_{t+1}(s) - V^*(s)| \leq \frac{\beta\epsilon + \int_{a\in\mathcal{A}} 1da - 1 - \ln F(Q_{t+1},s,\epsilon)}{\beta} + \max_a |Q_{t+1}(s,a) - Q^*(s,a)|. \tag{15}$$

We also have the following inequality

$$|Q_{t+1}(s,a') - Q^*(s,a')| \leq \gamma \max_{s'} |V_t(s') - V^*(s')|. \tag{16}$$

Combining (15) and (16), we obtain

$$||V_{t+1}(s) - V^*(s)||_\infty \leq \frac{\beta\epsilon + \int_{a\in\mathcal{A}} 1da - 1 - \min_s \ln F(Q_{t+1},s,\epsilon)}{\beta} + \gamma ||V_t(s) - V^*(s)||_\infty. \tag{17}$$

Therefore, we have

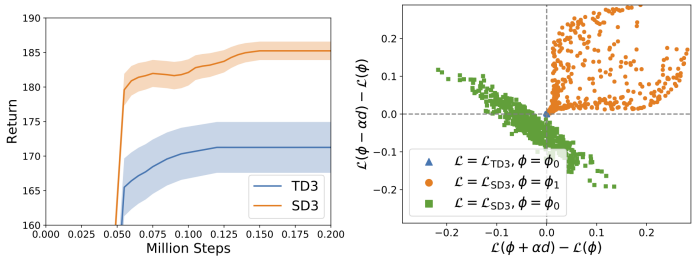
$$
\begin{aligned}
&||V_t(s) - V^*(s)||_\infty \\
\leq &\gamma^t ||V_0(s) - V^*(s)||_\infty + \sum_{k=1}^{t} \gamma^{t-k} \frac{\beta\epsilon + \int_{a\in\mathcal{A}} 1da - 1 - \min_s \ln F(Q_k, s, \epsilon)}{\beta} \\
\leq &\gamma^t ||V_0(s) - V^*(s)||_\infty + \frac{1}{1-\gamma} \frac{\beta\epsilon + \int_{a\in\mathcal{A}} 1da - 1}{\beta} - \sum_{k=1}^{t} \gamma^{t-k} \frac{\min_s \ln F(Q_k, s, \epsilon)}{\beta}.
\end{aligned} \tag{18}
$$

□

B Softmax Helps to Smoooth the Optimization Landscape

B.1 Comparative Study on the Smoothing Effect of TD3 and SD3

We demonstrate the smoothing effect of SD3 on the optimization landscape in this section, where experimental setup is the same as in Section 4.1 in the text for the comparative study of SD2 and DDPG. Experimental details can be found in Section B.2.

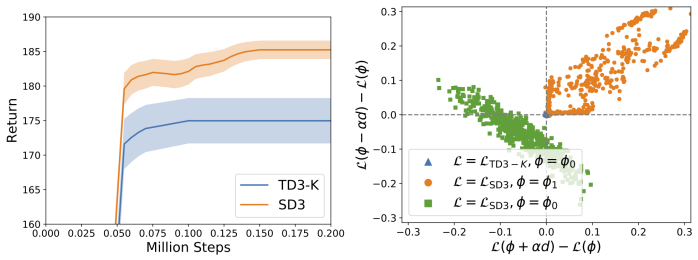


(a) Performance comparison. (b) Scatter plot.

Figure 9: Analysis of smoothing effect of TD3 and SD3 in the MoveCar environment.

The performance comparison of SD3 and TD3 is shown in Figure 9(a), where SD3 significantly outperforms TD3. Next, we analyze the smoothing effect of SD3 on the optimization landscape by the same way as in Section 4.1 in the text. We obtain the scatter plot of random perturbation in Figure 9(b). Specifically, ϕ_0 and ϕ_1 are policy parameters learned by TD3 and SD3 during training from a same random initialization, where ϕ_0 corresponds to a sub-optimal policy that determines to move to the right at the initial position x_0 while ϕ_1 corresponds to an optimal policy that determines to move to the left and is able to stay in the high-reward region. As demonstrated in Figure 9(b), for TD3, we observe that blue triangles are around the origin, so the perturbations for its policy parameters ϕ_0 are close to zero. This implies that there is a flat region in $\mathcal{L}_{\text{TD3}}$ and can be difficult for gradient-based methods to escape from [12]. For SD3, as the perturbations for its policy parameters ϕ_1 with respect to $\mathcal{L}_{\text{SD3}}$ are all positive (orange circles), the point ϕ_1 is likely a local optimum. To demonstrate the critical effect of the softmax operator on the objective, we also evaluate the change of the loss function $\mathcal{L}_{\text{SD3}}$ of SD3 with respect to parameters ϕ_0 from TD3. As shown in Figure 9(b), the green squares indicate that the loss $\mathcal{L}_{\text{SD3}}$ can be reduced following many directions, which shows the advantage of optimizing the softmax version of the objective.

So far, we have demonstrated the smoothing effect of SD3 over TD3. We further compare SD3 and TD3-K (which is introduced in Section 5.1 in the text) to demonstrate the critical effect of the softmax operator on the objective. The performance comparison is shown in Figure 10(a), where K is the same as in SD3. As shown, although TD3-K performs better than TD3, SD3 still outperforms TD3-K by a large margin in final performance. With the same way of random perturbation as in the previous part, we demonstrate the scatter plot of the random perturbation in Figure 10(b). Similarly, the evaluation of the change of the loss function $\mathcal{L}_{\text{SD3}}$ of SD3 with respect to parameters ϕ_0 from TD3-K (green squares) shows the critical effect of the softmax operator on the objective, as the loss can be reduced following a number of directions.



(a) Performance comparison. (b) Scatter plot.

Figure 10: Analysis of smoothing effect of TD3-K and SD3 in the MoveCar environment.

B.2 Experimental Setup in the MoverCar Environment

Hyperparameters of DDPG and SD2 are summarized in Table 1. For TD3 and SD3, hyperparameters are the same as in Table 2, except that we use $\mathcal{N}(0, 0.5)$ as in Table 1 for exploration to ensure that the exploration noise is high enough during training to exclude the effect of lack of exploration. For the TD3-K algorithm in Figure 10, K is the same as in SD3. We run all algorithms are 100 times (with different random seeds 0-99).

Table 1: Hyperparameters of DDPG and SD2.

Hyperparameter	**Value**
Shared hyperparameters (From [15])	
Batch size	100
Critic network	$(400, 300)$
Actor network	$(400, 300)$
Learning rate	10^{-3}
Optimizer	Adam
Replay buffer size	10^6
Warmup steps	10^4
Exploration policy	$\mathcal{N}(0, 0.5)$
Discount factor	0.99
Target update rate	5×10^{-3}
Hyperparameters for SD2	
Number of samples K	50
Action sampling noise $\bar{\sigma}$	0.2
Noise clipping coefficient c	0.5

C Softmax Deep Deterministic Policy Gradients

C.1 The SD2 Algorithm

The full SD2 algorithm is shown in Algorithm 2.

C.2 Proof of Theorem 3

Theorem 3 *Denote the bias of the value estimate and the true value induced by $\mathcal{T}$ as* $\text{bias}(\mathcal{T}) = \mathbb{E}\left[\mathcal{T}(\mathrm{s}')\right] - \mathbb{E}\left[\mathrm{Q}(\mathrm{s}', \pi(\mathrm{s}'; \phi^-); \theta^{\text{true}})\right]$. *Assume that the actor is a local maximizer with respect to the critic, then there exists noise clipping parameter $c > 0$ such that* $\text{bias}(\mathcal{T}_{\text{SD2}}) \leq \text{bias}(\mathcal{T}_{\text{DDPG}})$.

Proof. By definition, we have

$$\mathcal{T}_{\text{DDPG}}(s') = Q(s', \pi(s'; \phi^-); \theta^-), \quad \mathcal{T}_{\text{SD2}}(s') = \text{softmax}_\beta(Q(s', \cdot; \theta^-)) \tag{19}$$

Assume that the actor is a local maximizer with respect to the critic. There exists $c > 0$ such that for any state s', the action selected by the policy $\pi(s'; \phi^-)$ at state s' is a local maximum of the

Algorithm 2 SD2

Initialize the critic network Q and the actor network π with random parameters θ, ϕ
Initialize target networks $\theta^- \leftarrow \theta$, $\phi^- \leftarrow \phi$
Initialzie replay buffer $\mathcal{B}$
for $t = 1$ to T **do**
  Select action a with exploration noise $\epsilon \sim \mathcal{N}(0, \sigma)$ based on π
  Execute action a, observe reward r, new state s' and done d
  Store transition tuple (s, a, r, s', d) in $\mathcal{B}$
  Sample a mini-batch of N transitions $\{(s, a, r, s', d)\}$ from $\mathcal{B}$
  Sample K noises $\epsilon \sim \mathcal{N}(0, \bar{\sigma})$
  $\hat{a}' \leftarrow \pi(s'; \phi^-) + \text{clip}(\epsilon, -c, c)$
  $\text{softmax}_\beta\left(Q(s', \cdot; \theta^-)\right) \leftarrow \mathbb{E}_{\hat{a}' \sim p}\left[\frac{\exp(\beta Q(s', \hat{a}'; \theta^-)) Q(s', \hat{a}'; \theta^-)}{p(\hat{a}')}\right] / \mathbb{E}_{\hat{a}' \sim p}\left[\frac{\exp(\beta Q(s', \hat{a}'; \theta^-))}{p(\hat{a}')}\right]$
  $y_i \leftarrow r + \gamma(1 - d)\text{softmax}_\beta\left(Q(s', \cdot; \theta^-)\right)$
  Update the parameter θ of the critic according to Bellman loss: $\frac{1}{N}\sum_s (Q(s, a; \theta) - y)^2$
  Update the parameter ϕ of the actor by policy gradient: $\frac{1}{N}\sum_s \left[\nabla_\phi(\pi(s; \phi)) \nabla_a Q(s, a; \theta)|_{a=\pi(s;\phi)}\right]$
  Update target networks: $\theta^- \leftarrow \tau\theta + (1 - \tau)\theta^-$, $\phi^- \leftarrow \tau\phi + (1 - \tau)\phi^-$
end for

Q-function $Q(s', \cdot; \theta^-)$, i.e.,

$$Q(s', \pi(s'; \phi^-); \theta^-) = \max_{a \in \mathcal{A}_c} Q(s', a; \theta^-). \tag{20}$$

From Theorem 1, we have that

$$\text{softmax}_\beta(Q(s', \cdot; \theta^-)) \leq \max_{a \in \mathcal{A}_c} Q(s', a; \theta^-). \tag{21}$$

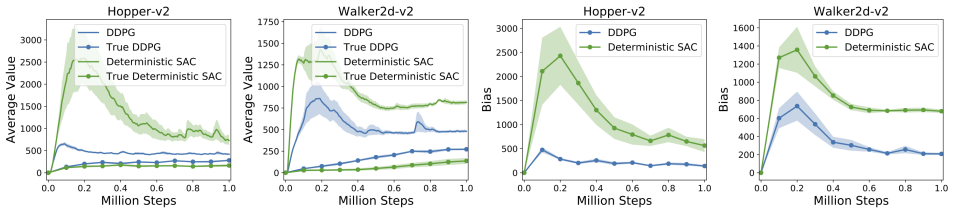
Thus,

$$\text{softmax}_\beta(Q(s', \cdot; \theta^-)) \leq Q(s', \pi(s'; \phi^-); \theta^-), \tag{22}$$

and we obtain $\mathcal{T}_{\text{SD2}}(s') \leq \mathcal{T}_{\text{DDPG}}(s')$. Therefore, $\text{bias}(\mathcal{T}_{\text{SD2}}) \leq \text{bias}(\mathcal{T}_{\text{DDPG}})$. □

C.3 Comparison of Estimation of Value Functions of DDPG and Deterministic SAC

Figure 11(a) shows the value estimates and true values of DDPG and Deterministic SAC, and Figure 11(b) demonstrates the corresponding bias of value estimations, from which we observe that Deterministic SAC incurs larger overestimation bias in comparison with DDPG.



(a) Value estimates and true values.

(b) Bias of value estimations.

Figure 11: Comparison of estimations of value functions of DDPG and Deterministic SAC.

D Softmax Deep Double Deterministic Policy Gradients

D.1 Proof of Theorem 4

Theorem 4 *Denote $\mathcal{T}_{\text{TD3}}$, $\mathcal{T}_{\text{SD3}}$ the value estimation functions of TD3 and SD3 respectively, then we have* $\text{bias}(\mathcal{T}_{\text{SD3}}) \geq \text{bias}(\mathcal{T}_{\text{TD3}})$.

Proof. By definition, we have

$$\mathcal{T}_{\mathrm{TD3}}(s') = \hat{Q}_i(s', \hat{a}'), \quad \mathcal{T}_{\mathrm{SD3}}(s') = \mathrm{softmax}_\beta(\hat{Q}_i(s', \hat{a}')). \tag{23}$$

Since

$$\mathbb{E}\left[\hat{Q}_i(s', \hat{a}')\right] = \mathbb{E}\left[\mathrm{softmax}_0(\hat{Q}_i(s', \hat{a}'))\right], \tag{24}$$

it suffices to prove that $\forall \beta \geq 0$,

$$\mathrm{softmax}_\beta\left(\hat{Q}_i(s', \hat{a}')\right) \geq \mathrm{softmax}_0\left(\hat{Q}_i(s', \hat{a}')\right). \tag{25}$$

Now we show that the softmax operator is increasing with β. By definition,

$$\begin{aligned}
&\nabla_\beta \mathrm{softmax}_\beta\left(Q(s,\cdot)\right) \\
=&\nabla_\beta \frac{\int_{a\in\mathcal{A}} e^{\beta Q(s,a)} Q(s,a) da}{\int_{a'\in\mathcal{A}} e^{\beta Q(s,a')} da'} \\
=&\frac{\int_{a\in\mathcal{A}} e^{\beta Q(s,a)} Q^2(s,a) da \times \int_{a'\in\mathcal{A}} e^{\beta Q(s,a')} da'}{\left(\int_{a'\in\mathcal{A}} e^{\beta Q(s,a')} da'\right)^2} - \frac{\left(\int_{a\in\mathcal{A}} e^{\beta Q(s,a)} Q(s,a) da\right)^2}{\left(\int_{a'\in\mathcal{A}} e^{\beta Q(s,a')} da'\right)^2}.
\end{aligned} \tag{26}$$

From the Cauchy-Schwarz inequality, we have $\forall \beta, \nabla_\beta \mathrm{softmax}_\beta\left(Q(s,\cdot)\right) \geq 0$. Thus, $\mathrm{softmax}_\beta$ attains its minimum at $\beta = 0$. □

E Experimental Setup

E.1 Hyperparameters

Hyperparameters of DDPG, TD3, and SD3 are shown in Table 2, where DDPG refers to 'OurDDPG' in [14]. Note that all hyperparameters are the same for all environments except for Humanoid-v2, as TD3 with default hyperparameters in this environment almost fails. For Humanoid-v2, the hyperparameters is a tuned set as provided in author's open-source implementation [14] to make TD3 work in this environment. For SD3, the parameter β is 10^{-3} for Ant-v2, 5×10^{-3} for HalfCheetah-v2, 5×10^{-2} for BipedalWalker-v2, Hopper-v2, and Humanoid-v2, 10^{-1} for Walker2d-v2, 5×10^{-1} for LunarLanderContinuous-v2, and a relatively large $\beta = 5 \times 10^2$ for Swimmer-v2.

Table 2: Hyperparameters of DDPG, TD3, and SD3.

Hyperparameter	**All environments except for Humanoid-v2**	**Humanoid-v2**
Shared hyperparameters (From [15, 14])		
Batch size	100	256
Critic network	$(400, 300)$	$(256, 256)$
Actor network	$(400, 300)$	$(256, 256)$
Learning rate	10^{-3}	3×10^{-4}
Optimizer	Adam	
Replay buffer size	10^6	
Warmup steps	10^4	
Exploration policy	$\mathcal{N}(0, 0.1)$	
Discount factor	0.99	
Target update rate	5×10^{-3}	
Noise clip	0.5	
Hyperparameters for TD3 (From [15])		
Target update interval	2	
Target noise	0.2	
Hyperparameters for SD3		
Number of samples K	50	
Action sampling noise $\bar{\sigma}$	0.2	

E.2 The TD3-K Algorithm

We compare SD3 with TD3-K, a variant of TD3 that uses K samples of actions to evaluate the Q-function, to demonstrate that using multiple samples is not the main factor for the performance improvement of SD3. Specifically, TD3-K samples K actions a' by the same way as in SD3 to compute Q-values, and take the $\min$ operation over the averaged Q-values, i.e., $y_{1,2} = r + \gamma \min_{i=1,2} \left(\frac{1}{K} \sum_{j=1}^{K} Q_i(s', a'; \theta_i^-) \right)$.

References

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

[2] Z. Ahmed, N. Le Roux, M. Norouzi, and D. Schuurmans. Understanding the impact of entropy on policy optimization. In *International Conference on Machine Learning*, pages 151–160, 2019.

[3] O. Anschel, N. Baram, and N. Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 176–185. JMLR. org, 2017.

[4] K. Asadi and M. L. Littman. An alternative softmax operator for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 243–252. JMLR. org, 2017.

[5] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. Tb, A. Muldal, N. Heess, and T. Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

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

[8] E. Catto. Box2d: A 2d physics engine for games. 2011.

[9] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu. Boltzmann exploration done right. In *Advances in neural information processing systems*, pages 6284–6293, 2017.

[10] K. Ciosek, Q. Vuong, R. Loftin, and K. Hofmann. Better exploration with optimistic actor critic. In *Advances in Neural Information Processing Systems*, pages 1785–1796, 2019.

[11] K. Ciosek and S. Whiteson. Expected policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[12] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.

[13] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

[14] S. Fujimoto. Open-source implementation for TD3. `https://github.com/sfujim/TD3`.

[15] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. 2018.

[16] I. J. Goodfellow, O. Vinyals, and A. M. Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

[17] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pages 2829–2838, 2016.

[18] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361, 2017.

[19] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870, 2018.

[20] H. V. Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

[21] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

[22] T. Jaakkola, M. I. Jordan, and S. P. Singh. Convergence of stochastic iterative dynamic programming algorithms. In *Advances in neural information processing systems*, pages 703–710, 1994.

[23] S. Khadka and K. Tumer. Evolution-guided policy gradient in reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1188–1200, 2018.

[24] Q. Lan, Y. Pan, A. Fyshe, and M. White. Maxmin q-learning: Controlling the estimation bias of q-learning. In *International Conference on Learning Representations*, 2020.

[25] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[26] M. L. Littman and C. Szepesvári. A generalized reinforcement-learning model: Convergence and applications. In *ICML*, volume 96, pages 310–318, 1996.

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

[28] O. Nachum, M. Norouzi, G. Tucker, and D. Schuurmans. Smoothed action value functions for learning gaussian policies. In *International Conference on Machine Learning*, pages 3692–3700, 2018.

[29] L. Pan, Q. Cai, Q. Meng, W. Chen, and L. Huang. Reinforcement learning with dynamic boltzmann softmax updates. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 1992–1998, 2020.

[30] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[31] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[32] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. 2014.

[33] Z. Song, R. E. Parr, and L. Carin. Revisiting the softmax bellman operator: New benefits and new perspective. *arXiv preprint arXiv:1812.00456*, 2018.

[34] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. 2011.

[35] S. Thrun and A. Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

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

[37] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[38] C. J. C. H. Watkins. Learning from delayed rewards. 1989.