Reinforcement Learning with Dynamic Boltzmann Softmax Updates

Ling Pan[1] , **Qingpeng Cai**[2] , **Qi Meng**[3] , **Wei Chen**[3] , **Longbo Huang**[1]
[1]IIIS, Tsinghua University
[2]Alibaba Group
[3]Microsoft Research
pl17@mails.tsinghua.edu.cn, qingpeng.cqp@alibaba-inc.com, {meq,wche}@microsoft.com,
longbohuang@tsinghua.edu.cn

Abstract

Value function estimation, i.e., prediction, is an important task in reinforcement learning. The Boltzmann softmax operator is a natural value estimator and can provide several benefits. However, it does not satisfy the non-expansion property, and its direct use may fail to converge even in value iteration. In this paper, we propose to update the value function with dynamic Boltzmann softmax (DBS) operator, which has good convergence property in the setting of planning and learning. Experimental results on GridWorld show that the DBS operator enables better estimation of the value function, which rectifies the convergence issue of the softmax operator. Finally, we propose the DBS-DQN algorithm by applying the DBS operator, which outperforms DQN substantially in 40 out of 49 Atari games.

1 Introduction

Reinforcement learning has achieved groundbreaking success for many decision making problems [Kober *et al.*, 2013; Mnih *et al.*, 2015]. Without full information of the environment, the agent learns an optimal policy by interacting with the environment from experience.

Value function estimation, i.e., prediction, is an important task in reinforcement learning [Xu *et al.*, 2018; Pan *et al.*, 2019]. In the prediction task, it requires the agent to have a good estimate of the value function in order to update towards the true value function. A key factor to prediction is the action-value summary operator. The action-value summary operator for a popular off-policy method, Q-learning [Watkins, 1989], is the hard max operator, which always commits to the maximum action-value function according to current estimation for updating the value estimator. The "hard max" updating scheme may lead to misbehavior due to noise in stochastic environments [Hasselt, 2010; van Hasselt, 2013; Fox *et al.*, 2015]. Even in deterministic environments, this may not be accurate as the value estimator is not correct in the early stage of the learning process. Consequently, it is important to choose an appropriate action-value summary operator.

The Boltzmann softmax operator is a natural value estimator based on the Boltzmann softmax distribution, which is a widely-used scheme to address the exploration-exploitation dilemma in reinforcement learning [Azar *et al.*, 2012; Cesa-Bianchi *et al.*, 2017]. In addition, the Boltzmann softmax operator provides benefits for reducing overestimation and gradient noise in deep Q-networks [Song *et al.*, 2019]. However, despite the advantages, it is challenging to apply the Boltzmann softmax operator in value function estimation. As shown in [Littman and Szepesvári, 1996; Asadi and Littman, 2017], it is not a non-expansion, which may lead to multiple fixed-points and thus the optimal value function of this policy is not well-defined. Non-expansion is a vital and widely-used sufficient property to guarantee the convergence of the planning and learning algorithm. Without such property, the algorithm may misbehave or even diverge.

We propose to update the value function using the dynamic Boltzmann softmax (DBS) operator with good convergence guarantee. Our idea is to make the parameter β time-varying while being state-independent. We prove that having β_t approach ∞ suffices to guarantee the convergence of value iteration with the DBS operator. Therefore, the DBS operator rectifies the convergence issue of the Boltzmann softmax operator with fixed parameters. Note that we also achieve a tighter error bound for the fixed-parameter softmax operator in general cases compared with [Song *et al.*, 2019]. In addition, we show that the DBS operator achieves good convergence rate. Based on this theoretical guarantee, we apply the DBS operator to estimate value functions in the setting of model-free reinforcement learning without known model. We prove that the corresponding DBS Q-learning algorithm also guarantees convergence. Finally, we propose the DBS-DQN algorithm, which generalizes our proposed DBS operator from tabular Q-learning to deep Q-networks using function approximators in high-dimensional state spaces.

It is crucial to note the DBS operator is the only one that meets all desired properties proposed in [Song *et al.*, 2019] up to now, as it ensures Bellman optimality, enables overestimation reduction, directly represents a policy, can be applicable to double Q-learning [Hasselt, 2010], and requires no tuning.

To examine the effectiveness of the DBS operator, we conduct extensive experiments to evaluate the effectiveness and efficiency. We first evaluate DBS value iteration and DBS Q-learning on a tabular game, the GridWorld. Our results show that the DBS operator leads to smaller error and better performance than vanilla Q-learning and soft Q-learning [Haarnoja *et al.*, 2017]. We then evaluate DBS-DQN on large

scale Atari2600 games, and we show that DBS-DQN outperforms DQN in 40 out of 49 Atari games.

The main contributions can be summarized as follows:

- We analyze the error bound of the Boltzmann softmax operator with arbitrary parameters, including static and dynamic.
- We propose the DBS operator, which has good convergence property in the setting of planning and learning.
- We conduct extensive experiments to verify the effectiveness of the DBS operator in 49 Atari games. Experimental results verify our theoretical analysis and demonstrate the effectiveness of the DBS operator.

2 Preliminaries

A Markov decision process (MDP) is defined by a 5-tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the set of states and actions, $p(s'|s,a)$ the transition probability from state s to state s' under action a, and $r(s,a)$ the corresponding immediate reward. The discount factor is denoted by $\gamma \in [0,1)$.

The agent interacts with the environment with its policy π, a mapping from state to action. The objective is to find an optimal policy that maximizes the expected discounted long-term reward $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t|\pi]$, which can be solved by estimating value functions. The state value of s and state-action value of s and a under policy π are defined as $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t|s_0 = s]$ and $Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t|s_0 = s, a_0 = a]$. The optimal value functions are defined as $V^*(s) = \max_\pi V^\pi(s)$ and $Q^*(s,a) = \max_\pi Q^\pi(s,a)$.

The optimal value function V^* and Q^* satisfy the Bellman equation, which is defined recursively as in Eq. (1):

$$\begin{aligned} V^*(s) &= \max_{a \in A} Q^*(s,a), \\ Q^*(s,a) &= r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V^*(s'). \end{aligned} \tag{1}$$

Starting from an arbitrary initial value function V_0, the optimal value function V^* can be computed by value iteration [Bellman, 1957] according to an iterative update $V_{k+1} = \mathcal{T}V_k$, where $\mathcal{T}$ is the Bellman operator defined by $(\mathcal{T}V)(s) = \max_{a \in A} \left[r(s,a) + \sum_{s' \in S} p(s'|s,a)\gamma V(s') \right]$.

When the model is unknown, Q-learning [Watkins and Dayan, 1992] is an effective algorithm to learn by exploring the environment. Value estimation and update for a given trajectory (s,a,r,s') for Q-learning is defined as:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') \right), \tag{2}$$

where α denotes the learning rate. Note that Q-learning employs the hard max operator for value function updates, i.e., $\max(\mathbf{X}) = \max_i x_i$. Another common operator is the log-sum-exp operator (Eq. (5) in [Haarnoja *et al.*, 2017]): $L_\beta(\mathbf{X}) = \frac{1}{\beta} \log(\sum_{i=1}^n e^{\beta x_i})$. The Boltzmann softmax operator is defined as: $\mathrm{boltz}_\beta(\mathbf{X}) = \frac{\sum_{i=1}^n e^{\beta x_i} x_i}{\sum_{i=1}^n e^{\beta x_i}}$.

3 Dynamic Boltzmann Softmax Updates

In this section, we propose the dynamic Boltzmann softmax operator (DBS) for value function updates.

The DBS operator is defined as: $\forall s \in \mathcal{S}$,

$$\mathrm{boltz}_{\beta_t}(Q(s,\cdot)) = \frac{\sum_{a \in \mathcal{A}} e^{\beta_t Q(s,a)} Q(s,a)}{\sum_{a \in \mathcal{A}} e^{\beta_t Q(s,a)}}, \tag{3}$$

where β_t is non-negative. Our core idea of the DBS operator is to dynamically adjust the value of β_t during the iteration.

We now give theoretical analysis of the proposed DBS operator and show that it has good convergence guarantee.

3.1 Value Iteration with DBS Updates

DBS value iteration admits a time-varying, state-independent sequence $\{\beta_t\}$ and updates the value function according to boltz_{β_t} by iterating the following steps:

$$\begin{aligned} &Q_{t+1}(s,a) \leftarrow \sum_{s'} p(s'|s,a) \left[r(s,a) + \gamma V_t(s') \right], \forall s, a \\ &V_{t+1}(s) \leftarrow \mathrm{boltz}_{\beta_t}(Q_{t+1}(s,\cdot)), \forall s \end{aligned} \tag{4}$$

For the ease of notations, we denote $\mathcal{T}_{\beta_t}$ the function that iterates any value function by Eq. (4).

Therefore, the way to update the value function is according to the exponential weighting scheme which is related to both the current estimator and the parameter β_t.

Theoretical Analysis

It has been shown that the Boltzmann softmax operator is not a non-expansion [Littman and Szepesvári, 1996], as it does not satisfy Ineq. (5).

$$\begin{aligned} &|\mathrm{boltz}_\beta(Q_1(s,\cdot)) - \mathrm{boltz}_\beta(Q_2(s,\cdot))| \\ \le &\max_a |Q_1(s,a) - Q_2(s,a)|, \ \forall s \in \mathcal{S}. \end{aligned} \tag{5}$$

Indeed, the non-expansion property is a vital and widely-used sufficient condition for achieving convergence of learning algorithms. If the operator is not a non-expansion, the uniqueness of the fixed point may not be guaranteed, which can lead to misbehaviors in value iteration.

In Theorem 1, we provide a novel analysis which demonstrates that the DBS operator enables the convergence of DBS value iteration to the optimal value function. We give the proof sketch due to limited space.

Theorem 1 (Convergence of value iteration with the DBS operator)**.** *For any dynamic Boltzmann softmax operator* boltz_{β_t}*, if* β_t *approaches* ∞*, the value function after* t *iterations* V_t *converges to the optimal value function* V^**.*

Proof Sketch. Let $\mathcal{T}_m$ be the function that iterates any value function by the max operator. Then, we have

$$\begin{aligned} ||(\mathcal{T}_{\beta_t} V_1) - (\mathcal{T}_m V_2)||_\infty \le &\underbrace{||(\mathcal{T}_{\beta_t} V_1) - (\mathcal{T}_m V_1)||_\infty}_{(I)} \\ &+ \underbrace{||(\mathcal{T}_m V_1) - (\mathcal{T}_m V_2)||_\infty}_{(II)} \end{aligned} \tag{6}$$

For the term (I), we have

$$||(\mathcal{T}_{\beta_t} V_1) - (\mathcal{T}_m V_1)||_\infty \le \frac{\log(|A|)}{\beta_t} \tag{7}$$

For the term (II), we have

$$||(\mathcal{T}_m V_1) - (\mathcal{T}_m V_2)||_\infty \leq \gamma ||V_1 - V_2|| \tag{8}$$

Combining (6), (7), and (8), we have

$$||(\mathcal{T}_{\beta_t} V_1) - (\mathcal{T}_m V_2)||_\infty \leq \gamma ||V_1 - V_2||_\infty + \frac{\log(|A|)}{\beta_t} \tag{9}$$

As $\max$ is a contraction mapping, then from Banach fixed-point theorem [Banach, 1922] we have $\mathcal{T}_m V^* = V^*$.

By the definition of DBS value iteration in Eq. (4),

$$\begin{aligned} ||V_t - V^*||_\infty =& ||(\mathcal{T}_{\beta_t} ... \mathcal{T}_{\beta_1})V_0 - (\mathcal{T}_m ... \mathcal{T}_m)V^*||_\infty \\ \leq& \gamma ||(\mathcal{T}_{\beta_{t-1}} ... \mathcal{T}_{\beta_1})V_0 - (\mathcal{T}_m ... \mathcal{T}_m)V^*||_\infty + \frac{\log(|A|)}{\beta_t} \\ \leq& \gamma^t ||V_0 - V^*||_\infty + \log(|A|) \sum_{k=1}^{t} \frac{\gamma^{t-k}}{\beta_k} \end{aligned} \tag{10}$$

If $\beta_t \to \infty$, then $\lim_{t\to\infty} \sum_{k=1}^{t} \frac{\gamma^{t-k}}{\beta_k} = 0$.

Taking the limit of the right hand side of Eq. (10), we obtain $\lim_{t\to\infty} ||V_{t+1} - V^*||_\infty = 0$. □

During the process of dynamically adjusting β_t, although the non-expansion property may be violated for some certain values of β, we only need the state-independent parameter β_t to approach infinity to guarantee the convergence.

Next, we justify that the DBS operator has good convergence rate guarantee. We omit the proof due to lack of space.

Theorem 2 (Convergence rate of value iteration with the DBS operator)**.** *For any power series $\beta_t = t^p (p > 0)$, let V_0 be an arbitrary initial value function such that $||V_0||_\infty \leq \frac{R}{1-\gamma}$, where $R = \max_{s,a} |r(s,a)|$, we have that for any non-negative $\epsilon < 1/4$, after $\max\{O\big(\frac{\log(\frac{1}{\epsilon}) + \log(\frac{1}{1-\gamma}) + \log(R)}{\log(\frac{1}{\gamma})}\big), O\big((\frac{1}{(1-\gamma)\epsilon})^{\frac{1}{p}}\big)\}$ steps, the error $||V_t - V^*||_\infty \leq \epsilon$.*

For the larger value of p, the convergence rate is faster. Note that when p approaches ∞, the convergence rate is dominated by the first term, which has the same order as that of the standard Bellman operator, implying that the DBS operator is competitive with the standard Bellman operator in terms of the convergence rate in known environment.

From the proof techniques in Theorem 1, we derive the error bound of value iteration with the Boltzmann softmax operator with fixed parameter β in Corollary 1.

Corollary 1 (Error bound of value iteration with the Boltzmann softmax operator)**.** *For any Boltzmann softmax operator with fixed parameter β, we have $\limsup_{t\to\infty} ||V_t - V^*||_\infty \leq \min\left\{\frac{\log(|A|)}{\beta(1-\gamma)}, \frac{2R}{(1-\gamma)^2}\right\}$.*

Here, we show that after an infinite number of iterations, the error between the value function V_t computed by the Boltzmann softmax operator with the fixed parameter β at the t-th iteration and the optimal value function V^* can be upper bounded. However, although the error can be controlled, the direct use of the Boltzmann softmax operator with fixed parameter may introduce performance drop in practice, due to the fact that it violates the non-expansion property.

Thus, we conclude that the DBS operator performs better than the traditional Boltzmann softmax operator with fixed parameter in terms of convergence.

Relation to Existing Results

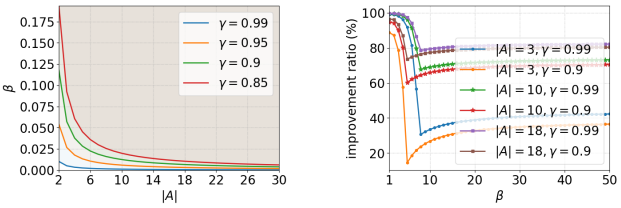
In this section, we compare the error bound in Corollary 1 with that in [Song *et al.*, 2019] (Theorem 3), which studies the error bound of the softmax operator with a fixed parameter β.

Theorem 3 (Song et al., 2019)**.** *Let $\mathcal{T}_{soft}$ be the function that iterates any value function by the Boltzmann softmax operator, then $\forall (s,a), \liminf_{t\to\infty} \mathcal{T}^t_{soft} Q_0(s,a) \geq Q^*(s,a) - \frac{\gamma(|A|-1)}{(1-\gamma)} \max\left\{\frac{1}{\beta+2}, \frac{2Q_{max}}{1+e^\beta}\right\}$.*

Different from [Song *et al.*, 2019], we provide a more general convergence analysis of the softmax operator covering both static and dynamic parameters. We also achieve a tighter error bound when

$$\beta \geq \frac{2}{\max\{\frac{\gamma(|A|-1)}{\log(|A|)}, \frac{2\gamma(|A|-1)R}{1-\gamma}\} - 1}, \tag{11}$$

where R can be normalized to 1 and $|A|$ denotes the number of actions. The term on the RHS of Eq. (11) is quite small as shown in Figure 1(a), where we set γ to be some commonly used values in $\{0.85, 0.9, 0.95, 0.99\}$. The shaded area corresponds to the range of β within our bound is tighter, which is a general case. Note that the case where β is extremely small, i.e., approaches 0, is usually not considered in practice. Figure 1(b) shows the improvement of the error bound that is defined as $\frac{\text{their bound} - \text{our bound}}{\text{their bound}} \times 100\%$, where $|A|$ ranges from 3 to 18 which is common in the Arcade Learning Environment [Bellemare *et al.*, 2013]. Moreover, we also give an analysis of the convergence rate of the DBS operator.



(a) Range of β within which our bound is tighter.

(b) Improvement ratio.

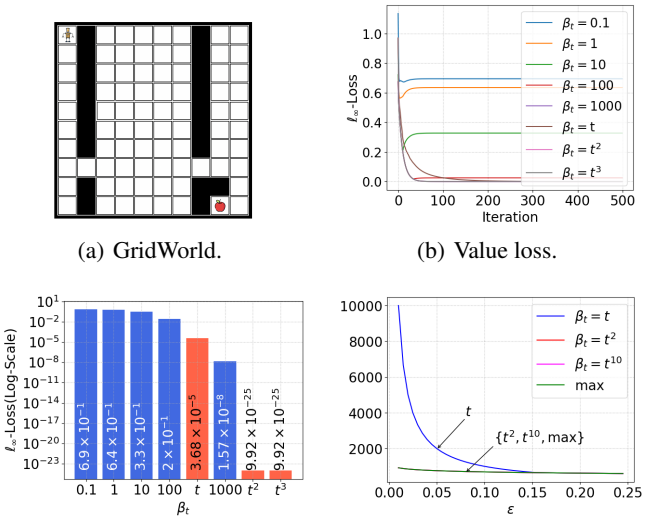
Figure 1: Error bound comparison.

Please note that our comparison with [Song *et al.*, 2019] is fair. In particular, [Song *et al.*, 2019] claim that $\mathcal{T}_{soft}$ converges to $\mathcal{T}$ with exponential rate in terms of β. However, we find that the analysis is incorrect (refer to the proof of Theorem 3 in Appendix A.1). [1]

[1] Eq. (A8) in the proof [Song *et al.*, 2019] relies on the assumption that the the gap function $\delta_i(s) = Q(s, a_{[1]}) - Q(s, a_{[i]})$ is constant. However, the gap function is varying during value iteration. Thus, Eq. (A7) only leads to the following result: $\mathcal{T}^k Q_0(s,a) - \mathcal{T}^k_{soft} Q_0(s,a) \leq \frac{\gamma(1-\gamma^k)}{1-\gamma} \max_{\delta\in\Delta} \sum_{i=2}^{m} \frac{\delta_i(s)}{1+e^{\beta\delta_i(s)}}$, where Δ denotes all possible gap functions during value iteration, instead of a fixed gap function.

Empirical Results

We first evaluate DBS value iteration to verify our convergence results in a 10×10 GridWorld (Figure 2(a)), which is a larger variant of the environment of [O'Donoghue *et al.*, 2016], with the dark grids representing walls. The agent starts at the upper left corner and aims to eat the apple at the bottom right corner upon receiving a reward of $+1$. Otherwise, the reward is 0. The terminal state is that the agent successfully eats the apple or a maximum number of steps 300 is reached. For this experiment, we consider the discount factor $\gamma = 0.9$.



(a) GridWorld. (b) Value loss.

(c) Value loss of the last iteration in log scale. (d) Convergence rate.

Figure 2: DBS value iteration in GridWorld.

The value loss of value iteration is shown in Figure 2(b). As expected, for fixed β, a larger value leads to a smaller loss. We then zoom in on Figure 2(b) to further illustrate the difference between fixed β and dynamic β_t in Figure 2(c), which shows the value loss for the last episode in log scale. For any fixed β, value iteration suffers from some loss which decreases as β increases. For dynamic β_t, the performance of t^2 and t^3 are the same and achieve the smallest loss in the domain game. Results for the convergence rate is shown in Figure 2(d). For higher order p of $\beta_t = t^p$, the convergence rate is faster. We also observe that the convergence rate of t^2 and t^{10} is very close and matches the performance of the standard Bellman operator as discussed before.

From the above results, we find a convergent variant of the Boltzmann softmax operator with good convergence rate, which paves the path for its use in reinforcement learning algorithms with little knowledge about the environment.

3.2 Q-learning with DBS Updates

In this section, we show that the DBS operator can be applied in a model-free Q-learning algorithm.

According to the DBS operator, we propose the DBS Q-learning algorithm, where the value function is updated by

$$Q(s,a) \leftarrow Q(s,a) + \alpha_t \left[r + \gamma \text{boltz}_{\beta_t}\left(Q(s',\cdot)\right) - Q(s,a) \right] \tag{12}$$

Please note that the action selection policy is different from the Boltzmann distribution (e.g., ϵ-greedy policy).

As seen in Theorem 2, a larger p results in faster convergence rate in value iteration. However, this is not the case in Q-learning, which differs from value iteration in that it knows little about the environment, and the agent has to learn from experience. If p is too large, it quickly approximates the max operator that favors commitment to current action-value function estimations. This is because the max operator always greedily selects the maximum action-value function according to current estimation, which may not be accurate in the early stage of learning or in noisy environments.

Theoretical Analysis

We get that DBS Q-learning converges to the optimal policy as in Theorem 4, which is under the same additional condition as in DBS value iteration.

Theorem 4 (Convergence of DBS Q-learning)**.** *DBS Q-learning converges to optimal $Q^*(s,a)$ values if (1) the state and action spaces are finite, and all state-action pairs are visited infinitely often; (2) $\sum_t \alpha_t(s,a) = \infty$ and $\sum_t \alpha_t^2(s,a) < \infty$; (3) $\lim_{t\to\infty} \beta_t = \infty$; (4) Var$(r(s,a))$ is bounded.*

Besides the convergence guarantee, we show that the DBS operator can mitigate the overestimation phenomenon of the max operator in Q-learning [Watkins, 1989].

Let $X = \{X_1, ..., X_M\}$ be a set of random variables, where the mean of variable X_i is denoted by μ_i. Note that in value function estimation, X_i denotes random values of action i for a fixed state. The true value of the maximum expected value is $\mu^*(X) = \max_i \mu_i$. Let $\hat{\mu}_i$ denote the sample mean of μ_i and $\hat{F}$ denote the joint distribution of $\hat{\mu}$, where $\hat{\mu} = (\hat{\mu}_1, ..., \hat{\mu}_M)$. The bias of any action-value summary operator $\bigotimes$ is defined as $\text{Bias}(\hat{\mu}^*_{\bigotimes}) = \mathbb{E}_{\hat{\mu}\sim\hat{F}}[\bigotimes \hat{\mu}] - \mu^*(\text{X})$, i.e., the difference between the expected estimated value output by the operator and the maximum expected value.

We now compare the bias for different common operators and we derive the following theorem.

Theorem 5. *Let $\hat{\mu}^*_{B_{\beta_t}}, \hat{\mu}^*_{\max}, \hat{\mu}^*_{L_\beta}$ denote the estimator with the DBS operator, the max operator, and the log-sum-exp operator. For any given set of M random variables, we have $\forall t,\ \forall \beta$, $\text{Bias}(\hat{\mu}^*_{\text{B}_{\beta_t}}) \le \text{Bias}(\hat{\mu}^*_{\max}) \le \text{Bias}(\hat{\mu}^*_{\text{L}_\beta})$.*

In Theorem 5, we show that although the log-sum-exp operator [Haarnoja *et al.*, 2017] is able to encourage exploration because its objective is an entropy-regularized form of the original objective, it may worsen the overestimation phenomenon. In addition, the optimal value function induced by the log-sum-exp operator is biased from the optimal value function of the original MDP [Dai *et al.*, 2018]. In contrast, the DBS operator ensures convergence to the optimal value function as well as reduction of overestimation.

Empirical Results

We now evaluate DBS Q-learning in the same GridWorld environment. Figure 3 shows the number of steps the agent spent until eating the apple in each episode, and a fewer number of steps corresponds to a better performance.

For DBS Q-learning, we apply the power function $\beta_t = t^p$ with p denoting the order. As shown, DBS Q-learning with

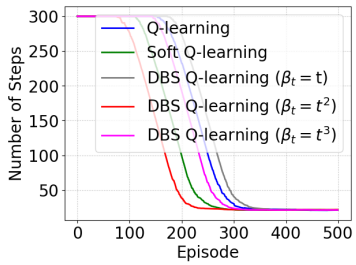


Figure 3: Performance comparison of DBS Q-learning, Soft Q-learning, and Q-learning in GridWorld.

the quadratic function achieves the best performance. Note that when $p = 1$, it performs worse than Q-learning in this simple game, which corresponds to our results in value iteration (Figure 2) as $p = 1$ leads to an nonnegligible value loss. When the power p of $\beta_t = t^p$ increases further, it performs closer to Q-learning. Soft Q-learning [Haarnoja *et al.*, 2017] uses the log-sum-exp operator, where the parameter is chosen with the best performance for comparison. In Figure 3, soft Q-learning performs better than Q-learning as it encourages exploration by entropy regularization. However, it underperforms DBS Q-learning ($\beta_t = t^2$) as DBS Q-learning can guarantee convergence to the optimal value function and can eliminate the overestimation phenomenon. Thus, we choose $p = 2$ in the following Atari experiments.

4 The DBS-DQN Algorithm

Now we show that the DBS operator can further be applied to problems with high dimensional state space and action space.

The DBS-DQN algorithm is shown in Algorithm 1. We compute the parameter of the DBS operator by applying the power function $\beta_t(c) = c \cdot t^2$ as the quadratic function performs the best in our previous analysis. Here, c denote the coefficient, and contributes to controlling the speed of the increase of $\beta_t(c)$. In many problems, it is critical to choose the hyper-parameter c. In order to make the algorithm more practical in problems with high-dimensional state spaces, we propose to learn to adjust c in DBS-DQN by the meta gradient-based optimization technique based on [Xu *et al.*, 2018].

The gradient-based optimization method follows the online cross-validation principle [Sutton, 1992]. Given current experience $\tau = (s, a, r, s_{next})$, the parameter θ of the function approximator is updated according to

$$\theta' = \theta - \alpha \frac{\partial J(\tau, \theta, c)}{\partial \theta}, \tag{13}$$

where α denotes the learning rate, and the loss of the neural network is

$$\begin{aligned} J(\tau, \theta, c) &= \frac{1}{2}\left[V(\tau, c; \theta^-) - Q(s, a; \theta)\right]^2, \\ V(\tau, c; \theta^-) &= r + \gamma \mathrm{boltz}_{\beta_t(c)}\left(Q(s_{next}, \cdot; \theta^-)\right). \end{aligned} \tag{14}$$

The corresponding gradient of $J(\tau, \theta, c)$ over θ is

$$\frac{\partial J(\tau, \theta, c)}{\partial \theta} = -\left[V(\tau, c; \theta^-) - Q(s, a; \theta)\right] \frac{\partial Q(s, a; \theta)}{\partial \theta}. \tag{15}$$

Algorithm 1 DBS Deep Q-Network

initialize experience replay buffer $\mathcal{B}$
initialize Q-function and target Q-function with random weights θ and θ^-
initialize the coefficient c of β_t of the DBS operator
for episode t = 1, ..., M **do**
  initialize state s
  calculate $\beta_t(c) = c \cdot t^2$
  for step = 1, ..., T **do**
    choose a from s using ϵ-greedy policy
    execute a, observe reward r, next state s_{next}, and done flag d
    store experience (s, a, r, s_{next}, d) in $\mathcal{B}$
    sample random minibatch of experiences (s, a, r, s_{next}, d) from $\mathcal{B}$
    set $y = r + \gamma(1 - d)\mathrm{boltz}_{\beta_t}\left(\hat{Q}(s_{next}, \cdot; \theta^-)\right)$
    perform a gradient descent step on $(y - Q(s, a; \theta))^2$ w.r.t. θ
    update c according to Eq.(18)
    reset $\hat{Q} = Q$ every C steps

Then, the coefficient c is updated based on the subsequent experience $\tau' = (s', a, r', s'_{next})$ according to the gradient of the squared error $J(\tau', \theta', \bar{c})$ between the value function approximator $Q(s'_{next}, a'; \theta')$ and the target value function $V(\tau', \bar{c}; \theta^-)$, where $\bar{c}$ is the reference value. The gradient is computed according to the chain rule in Eq. (16).

$$\frac{\partial J'(\tau', \theta', \bar{c})}{\partial c} = \underbrace{\frac{\partial J'(\tau', \theta', \bar{c})}{\partial \theta'}}_{A} \underbrace{\frac{\mathrm{d}\theta'}{\mathrm{d}c}}_{B}. \tag{16}$$

For the term (B), according to Eq. (13), we have

$$\frac{\mathrm{d}\theta'}{\mathrm{d}c} = \alpha\gamma \frac{\partial \mathrm{boltz}_{\beta_t(\bar{c})}(Q(s'_{next}, \cdot; \theta^-))}{\partial c} \frac{\partial Q(s, a; \theta)}{\partial \theta}. \tag{17}$$

Then, the update of c is as in Eq. (18), with η denoting the learning rate. [2]

$$c' = c - \eta \frac{\partial J'(\tau', \theta', \bar{c})}{\partial c}. \tag{18}$$

Note that it can be hard to choose an appropriate static value of sensitive parameter β. Therefore, it requires rigorous tuning of the task-specific fixed parameter β in different games in [Song *et al.*, 2019], which may limit its efficiency and applicability [Haarnoja *et al.*, 2018]. In contrast, the DBS operator is effective and efficient as it does not require tuning.

4.1 Experimental Setup

We evaluate the DBS-DQN algorithm on 49 Atari video games from the Arcade Learning Environment [Bellemare *et al.*, 2013], a standard challenging benchmark for deep reinforcement learning algorithms, by comparing it with DQN. For fair comparison, we use the same setup of network architectures and hyper-parameters as in [Mnih *et al.*, 2015] for

[2] In implementation, c is clipped to $\max\{0.01, c\}$.

both DQN and DBS-DQN. Our evaluation procedure is 30 no-op evaluation which is identical to [Mnih *et al.*, 2015], where the agent performs a random number (up to 30) of "do nothing" actions in the beginning of an episode. For each game, we train each algorithm for 50M steps for 3 independent runs to evaluate the performance.

4.2 Performance Comparison

To characterize the effectiveness of DBS-DQN, its improvement over DQN is shown in Figure 4, where the improvement is defined as the relative human normalized score: $\frac{\text{score}_{\text{agent}} - \text{score}_{\text{baseline}}}{\max\{\text{score}_{\text{human}}, \text{score}_{\text{baseline}}\} - \text{score}_{\text{random}}} \times 100\%$, with DQN serving as the baseline. In all, DBS-DQN exceeds the performance of DQN in 40 out of 49 Atari games.

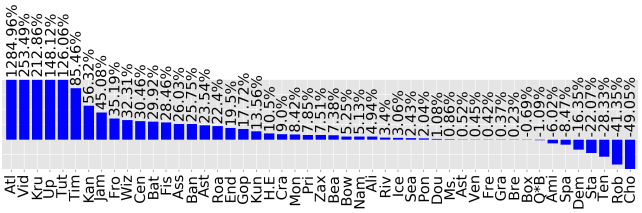


Figure 4: Relative human normalized score on Atari games.

We also compare the median in human normalized score [Van Hasselt *et al.*, 2016] defined as: $\frac{\text{score}_{\text{agent}} - \text{score}_{\text{random}}}{\text{score}_{\text{human}} - \text{score}_{\text{random}}} \times 100\%$, where human score and random score are taken from [Wang *et al.*, 2015]. The medians for DQN and DBS-DQN are 84.72% and 104.49%, respectively. As shown, DBS-DQN significantly outperforms DQN in terms the median of the human normalized score, and surpasses human level.

In addition, we compare DBS-DQN with its variant using fine-tuned fixed coefficient c in $\beta_t(c)$, i.e., without gradient-based optimization, in each game, to further demonstrate the effectiveness and efficiency of DBS-DQN. The median of the human normalized score for DBS-DQN (fine-tuned c) is 103.95%, which underperforms DBS-DQN but outperforms DQN. Note that DBS-DQN (fine-tuned c) achieves fairly good performance in term of the median and beats DQN in 33 out of 49 Atari games, which further illustrate the strength of our proposed DBS updates without gradient-based optimization of c.

Figure 5 shows the learning curves (moving averaged) of different algorithms, where S-DQN [Song *et al.*, 2019] is with the best parameter β for each game from grid search in $\{1, 5, 10\}$ (which is the same experimental setting in [Song *et al.*, 2019]). As shown, DBS-DQN outperforms other algorithms. It is also worth noting that S-DQN can fail in some games even with best parameter β.

5 Related Work

The Boltzmann softmax distribution is widely used in reinforcement learning [Littman *et al.*, 1996; Azar *et al.*, 2012; Song *et al.*, 2019]. [Singh *et al.*, 2000] studied convergence of on-policy algorithm Sarsa, where they considered a dynamic scheduling of the parameter in softmax action selection strategy. However, the state-dependent parameter is impractical in

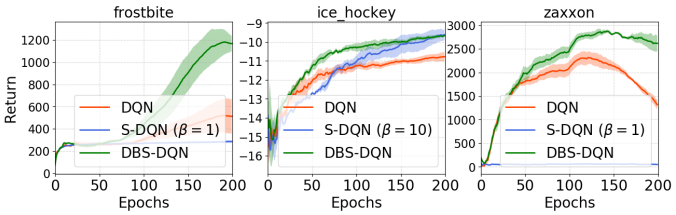


Figure 5: Learning curves in Atari games (shaded area represent mean ± standard deviation).

complex problems, e.g., Atari. Our work differs from theirs as our DBS operator is state-independent, which can be readily scaled to complex problems with high-dimensional state space. Recently, [Song *et al.*, 2019] applied the softmax operator in DQNs, and also studied the error bound of softmax. In contrast, we propose the DBS operator which rectifies the convergence issue of softmax, where we provide a more general analysis of the convergence property. A notable difference in the theoretical aspect is that we achieve a tighter error bound for softmax in general cases, and we investigate the convergence rate of the DBS operator. Besides the guarantee of Bellman optimality, the DBS operator is efficient as it does not require hyper-parameter tuning. Note that it can be hard to choose an appropriate static value of β in [Song *et al.*, 2019], which is game-specific and can result in different performance.

[Haarnoja *et al.*, 2017] utilized the log-sum-exp operator, which enables better exploration and learns deep energy-based policies. The connection between our proposed DBS operator and the log-sum-exp operator is discussed above. [Bellemare *et al.*, 2016] proposed a family of operators which are not necessarily non-expansions, but still preserve optimality while being gap-increasing. However, such conditions are still not satisfied for the Boltzmann softmax operator.

6 Conclusion

We propose the DBS operator in value function estimation with a time-varying, state-independent parameter. The DBS operator has good convergence guarantee in the setting of planning and learning, which rectifies the convergence issue of the Boltzmann softmax operator. Results validate the effectiveness of DBS-DQN in a suite of Atari games. For future work, it is worth studying the sample complexity of our proposed DBS Q-learning algorithm. It is also promising to apply the DBS operator to other state-of-the-art DQN-based algorithms, such as Rainbow [Hessel *et al.*, 2017].

Acknowledgments

We would like to thank Tie-Yan Liu for insightful discussions on this paper. 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.

References

[Asadi and Littman, 2017] Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, pages 243–252, 2017.

[Azar *et al.*, 2012] Mohammad Gheshlaghi Azar, Vicenç Gómez, and Hilbert J Kappen. Dynamic policy programming. *Journal of Machine Learning Research*, 13(Nov):3207–3245, 2012.

[Banach, 1922] Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fund. math*, 3(1):133–181, 1922.

[Bellemare *et al.*, 2013] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[Bellemare *et al.*, 2016] Marc G Bellemare, Georg Ostrovski, Arthur Guez, Philip S Thomas, and Rémi Munos. Increasing the action gap: New operators for reinforcement learning. In *AAAI*, pages 1476–1483, 2016.

[Bellman, 1957] Richard Ernest Bellman. Dynamic programming. 1957.

[Cesa-Bianchi *et al.*, 2017] Nicolò Cesa-Bianchi, Claudio Gentile, Gábor Lugosi, and Gergely Neu. Boltzmann exploration done right. In *Advances in Neural Information Processing Systems*, pages 6284–6293, 2017.

[Dai *et al.*, 2018] Bo Dai, Albert Shaw, Lihong Li, Lin Xiao, Niao He, Zhen Liu, Jianshu Chen, and Le Song. Sbeed: Convergent reinforcement learning with nonlinear function approximation. In *International Conference on Machine Learning*, pages 1133–1142, 2018.

[Fox *et al.*, 2015] Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

[Haarnoja *et al.*, 2017] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Sehoon Ha, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018.

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

[Hessel *et al.*, 2017] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*, 2017.

[Kober *et al.*, 2013] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[Littman and Szepesvári, 1996] Michael L Littman and Csaba Szepesvári. A generalized reinforcement-learning model: Convergence and applications. In *International Conference on Machine Learning*, pages 310–318, 1996.

[Littman *et al.*, 1996] Michael L Littman, Andrew W Moore, et al. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4(11, 28):237–285, 1996.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[O'Donoghue *et al.*, 2016] Brendan O'Donoghue, Remi Munos, Koray Kavukcuoglu, and Volodymyr Mnih. Combining policy gradient and q-learning. *arXiv preprint arXiv:1611.01626*, 2016.

[Pan *et al.*, 2019] Ling Pan, Qingpeng Cai, and Longbo Huang. Multi-path policy optimization. *arXiv preprint arXiv:1911.04207*, 2019.

[Singh *et al.*, 2000] Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38(3):287–308, 2000.

[Song *et al.*, 2019] Zhao Song, Ronald Parr, and Lawrence Carin. Revisiting the softmax bellman operator: New benefits and new perspective. 2019.

[Sutton, 1992] Richard S Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *AAAI*, pages 171–176, 1992.

[Van Hasselt *et al.*, 2016] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[van Hasselt, 2013] Hado van Hasselt. Estimating the maximum expected value: an analysis of (nested) cross validation and the maximum sample average. *arXiv preprint arXiv:1302.7175*, 2013.

[Wang *et al.*, 2015] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[Watkins and Dayan, 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[Watkins, 1989] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

[Xu *et al.*, 2018] Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-gradient reinforcement learning. *arXiv preprint arXiv:1805.09801*, 2018.