Adaptive Temperature Tuning for Mellowmax in Deep Reinforcement Learning

Seungchan Kim
Department of Computer Science
Brown University
seungchan_kim@brown.edu

George Konidaris
Department of Computer Science
Brown University
gdk@cs.brown.edu

Abstract

Mellowmax is an alternative softmax operator that has recently been used in deep reinforcement learning in the context of action-selection, reducing overestimation bias, and removing the need for a target network. However, Mellowmax has a temperature parameter which must be tuned to obtain optimal results. Previous work has relied on grid-search to find the optimal temperature parameter, which is computationally expensive, requires a domain simulator, and fixes a single parameter setting for the entire learning process. We propose a novel online method that adaptively tunes the Mellowmax temperature parameter using a meta-gradient reinforcement learning approach. We test the approach in a simple OpenAI gym domain, Acrobot, and present preliminary results that our adaptive algorithm performs better than the previous method with a fixed temperature setting.

1 Introduction

Mellowmax [1] is a recently proposed alternative softmax operator for reinforcement learning with several interesting properties. Unlike the well-known Boltzmann softmax operator, Mellowmax has a non-expansion property that ensures convergence to a unique fixed point. It is also a differentiable and smooth approximation of the max function, and hence facilitates analysis via gradient-based optimizations. Previous work has demonstrated other useful properties of Mellowmax, such as connections with entropy-regularization [8, 9] and convergent off-policy learning for non-linear function approximation [4].

More recent work has shown that Mellowmax can be combined with deep reinforcement learning. In particular, when combined with Deep Q-Network (DQN) [7] in the context of action-value summarization, Mellowmax yields comparable performance to the softmax Boltzmann operator in Atari games [11]. Mellowmax also alleviates the overestimation bias of Q-value estimations, and reduces the need for a separate target network in DQN [5].

However, one limitation of Mellowmax is that it has a temperature hyperparameter, ω, which must be tuned to maximize performance on each task. Previous work [5, 11] has either relied on exhaustive grid-search to find optimal temperature values or reported algorithmic performance based on a small set of arbitrary temperature values. These approaches can be inefficient, especially in deep reinforcement learning settings where training takes a long time. Grid searches also have fundamental drawbacks, such as finding a single fixed temperature parameter setting, which may not perform as well as dynamically adjusting the parameter.

We propose a novel online method to adaptively tune the temperature parameter ω for deep reinforcement learning. We use meta-gradient reinforcement learning [13], a gradient-based meta learning algorithm that optimizes the return function itself by adapting its tunable meta-parameter.

Neural Information Processing Systems (NeurIPS) 2019 Deep Reinforcement Learning Workshop.

We show that we can adjust the temperature parameter due to the differentiability of Mellowmax with respect to ω. We test the performance of this algorithm on Acrobot, a simple OpenAI gym control domain [3], and present preliminary results that our adaptive algorithm performs better than a fixed parameter identified by grid search.

2 Background

2.1 The Mellowmax Operator and Deep Reinforcement Learning

The Mellowmax operator [1] is an alternative softmax operator defined as:

$$mm_\omega(\mathbf{x}) = \frac{\log(\frac{1}{n}\sum_{i=1}^{n}\exp(\omega x_i))}{\omega}, \tag{1}$$

where $\mathbf{x}$ is an input vector of n real numbers, and ω is a temperature parameter.[1] Mellowmax is a non-expansion, which ensures convergence to a unique fixed point. Another interpretation of this operator is as a smoothed version of max function : if $\omega \to \infty$, then $mm_\omega(\mathbf{x})$ becomes $\max(\mathbf{x})$, and if $\omega \to 0$, then $mm_\omega(\mathbf{x})$ becomes mean($\mathbf{x}$). Any positive ω value between 0 and ∞ offers a continuous interpolation between max and mean. Thus, ω controls the degree of smoothness of the Mellowmax operator.

Furthermore, Mellowmax is differentiable with respect to ω, contrary to the non-differentiable max function.

$$\frac{\partial mm_\omega(\mathbf{x})}{\partial \omega} = \frac{1}{\omega^2}\{\frac{\omega\sum_{i=1}^{n} x_i\exp(\omega x_i)}{\sum_{i=1}^{n}\exp(\omega x_i)} - \log(\frac{1}{n}\sum_{i=1}^{n}\exp(\omega x_i))\}. \tag{2}$$

In reinforcement learning, Mellowmax operator can be used as an action-values summarizer:

$$mm_\omega(\mathbf{Q}) = \frac{\log(\frac{1}{n}\sum_{i=1}^{n}\exp(\omega Q(s,a_i)))}{\omega}, \tag{3}$$

where $\mathbf{Q} = [Q(s,a_1), Q(s,a_2), ..., Q(s,a_n)]$ is an input vector of action-values (or Q-values) in Q-learning. Thus, Mellowmax can replace the max function in the Bellman update equation and be incorporated into Q-learning as follows:

$$Q_{new}(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma mm_{\omega,a'}Q(s',a') - Q(s,a)]. \tag{4}$$

Using Equation 4, we can combine Deep Q-Network (DQN) [7] and Mellowmax in the context of action-values summarizations. Since the Q-function is approximated with neural network, the Bellman equation of the combination of DQN and Mellowmax (or Mellowmax-DQN) becomes:

$$\theta \leftarrow \theta + \alpha[r + \gamma mm_\omega\widehat{Q}(s',a';\theta^-) - Q(s,a;\theta)]\nabla_\theta Q(s,a;\theta), \tag{5}$$

where θ is the weight vector of Q network, and θ^- is the weight vector of the target network $\widehat{Q}$. This procedure is detailed in Algorithm 1.

Previous research [11] investigated the performance of DQN with different types of Bellman operators—the weighted softmax Boltzmann operator and Mellowmax operator—and compared their performance with DQN. Mellowmax-DQN achieves performance competitive with the DQN and Softmax-DQN in playing Atari games. This work also proposed an equivalence relationship between Mellowmax and the weighted softmax Bellman operator in value function estimations in DQN.

Another line of research [5] focused on the role of Mellowmax in reducing the need for a separate target network $\widehat{Q}$. The performance of DQN is known to degrade in the absence of the separable target network, but Mellowmax can obviate the need for a target network by making DQN stable without it. This algorithm, named DeepMellow [5], can be viewed as a special case of Algorithm 1: if we set the target network update frequency C to 1, the separate target network $\widehat{Q}$ is copied to be the value network Q every step, so the presence of *separate* target network is eliminated. Additionally,

Algorithm 1 Mellowmax Deep Q-Network

Fix the temperature parameter ω
Initialize experience replay memory D, set learning rate α
Initialize action-value network Q with random weights θ and target network $\widehat{Q}$ with $\theta^- = \theta$
Set the target network update frequency C
for episode = 1 to M **do**
  Initialize the starting state s_1
  for t= 1 to T **do**
    Select a random action a_t with probability ϵ. Otherwise, $a_t = \arg\max_a Q(s_t, a; \theta)$
    Execute action a_t, observe reward r_t, and obtain the next states_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in D
    Sample random batch $\tau = (s_j, a_j, r_j, s_{j+1})$ from D
    Compute the objective function $J = \frac{1}{2}[r_j + \gamma mm_{\omega}\widehat{Q}(s_{j+1}, a'; \theta^-) - Q(s_j, a_j; \theta)]^2$
    Update parameters: $\theta \leftarrow \theta - \alpha\frac{\partial J(\theta)}{\partial \theta}$
    Every C steps, copy $\widehat{Q} = Q$
  end for
end for

this work suggested that the overestimation bias in value function estimations caused by the max operator can be reduced using Mellowmax.

One limitation of using Mellowmax is the presence of an additional temperature parameter ω. This parameter must be tuned, because the performance of the algorithm largely depends on the choice of the temperature value. Both of the previous works have performed grid-search to find the optimal temperature parameter from a set of possible values. However, this approach is problematic for three reasons: 1) the grid-search finds a single temperature value for the entirety of learning, but fixing the temperature may not be the best strategy, when the dynamic adjustment of the parameter is needed, 2) it requires a domain simulator for each parameter search, making it unusable for online interactions with a real environment and 3) it is computationally expensive, especially when using deep neural networks that take several hours to train. Therefore, the need for an adaptive, online method for tuning ω is acute.

2.2 Meta-Gradient Reinforcement Learning

Meta-gradient reinforcement learning [13] aims to maximize the rewards for agent by optimizing the form of return function. This work proposes that the agent can learn the return itself by treating it as a function parameterized by tunable meta-parameters.

In deep reinforcement learning, the value function is parameterized by θ, and the main goal of an agent is to update θ as below:

$$\theta' \leftarrow \theta + f(\tau, \theta, \eta) = \theta - \alpha\frac{\partial J(\tau, \theta, \eta)}{\partial \theta}, \tag{6}$$

where τ is the batch samples of transition (or trajectories), θ is the parameter vector for value function, and η is the meta-parameter that should be adapted. The idea of meta-gradient reinforcement learning is to first update θ by computing the objective J using a batch sample τ, resulting in new parameters θ'; then, using an independent batch sample τ', updated parameter θ', and reference meta parameter $\overline{\eta}$, the agent obtains the derivative of meta-objective J' with respect to η:

$$\frac{\partial J'(\tau', \theta', \overline{\eta})}{\partial \eta} = \frac{\partial J'(\tau', \theta', \overline{\eta})}{\partial \theta'}\frac{d\theta'}{d\eta}. \tag{7}$$

The term on the left side, $\frac{\partial J'(\tau', \theta', \overline{\eta})}{\partial \eta}$, is hard to compute directly, but $\frac{\partial J'(\tau', \theta', \overline{\eta})}{\partial \theta'}$ and $\frac{d\theta'}{d\eta}$ can be computed via modern deep learning frameworks with some approximations [13].

[1]It might be more appropriately named an inverse temperature to be consistent with the Boltzmann softmax operator. We use the term temperature for ω here to be consistent with prior work.

The meta-parameter η is finally updated with the meta-learning rate β by the following equation:

$$\eta' \leftarrow \eta - \beta \frac{\partial J'(\tau', \theta', \overline{\eta})}{\partial \theta'} \frac{d\theta'}{d\eta}. \tag{8}$$

In order for this approach to work, the return function must be differentiable with respect to the meta-parameter η. For example, discount factor γ or bootstrapping parameter λ are good examples of meta-parameters, since the return function is differentiable with respect to γ and λ. Fortunately, both Mellowmax and the return function of Mellowmax-DQN are also differentiable with respect to ω. Therefore, we propose to use meta-gradient reinforcement learning to adapt ω.

3 Meta-Gradient Reinforcement Learning and Mellowmax

3.1 The Meta-Gradient Update for Mellowmax-DQN

In the case of Mellowmax-DQN, we set the loss objective function J as below:

$$J(\tau, \theta, \omega) = \frac{1}{2}[r_t + \gamma mm_\omega \widehat{Q}(s_{t+1}, a; \theta^-) - Q(s_t, a_t; \theta)]^2. \tag{9}$$

Here τ represents the minibatch samples for the update (s_t, a_t, r_t, s_{t+1}) tuples, θ are the Q-function parameters, α is a learning rate, and ω is the meta-parameter that we want to adapt. At every iteration, the Q-learning algorithm will update θ in a direction that minimizes $J(\tau, \theta, \omega)$.

Next, we define another meta-objective function J', using the same form of loss function as before:

$$J(\tau', \theta', \overline{\omega}) = \frac{1}{2}[r'_t + \gamma mm_{\overline{\omega}} \widehat{Q}(s'_{t+1}, a'; \theta^-) - Q(s'_t, a'_t; \theta')]^2. \tag{10}$$

Note that when we compute the meta-objective J', we use another independent batch sample τ' (denoted as $(s'_t, a'_t, r'_t, s'_{t+1})$ tuples). After each iteration of Equation 9, θ is updated to θ', and we use the updated θ' when we compute the meta-objective J'. We also use the reference meta-parameter value $\overline{\omega}$. Our goal is to adapt the value of ω such that it optimizes the meta-objective function J', using the equation below:

$$\frac{\partial J'(\tau', \theta', \overline{\omega})}{\partial \omega} = \frac{\partial J'(\tau', \theta', \overline{\omega})}{\partial \theta'} \frac{d\theta'}{d\omega}. \tag{11}$$

We approximate $d\theta'/d\omega$ by choosing $\mu = 0$ as done in previous work [13],

$$\frac{d\theta'}{d\omega} = \frac{d\{\theta + f(\tau, \theta, \omega)\}}{d\omega} = \frac{d\theta}{d\omega} + \frac{\partial f(\tau, \theta, \omega)}{\partial \omega} + \frac{\partial f(\tau, \theta, \omega)}{\partial \theta} \frac{d\theta}{d\omega}.$$

$$= \{I + \frac{\partial f(\tau, \theta, \omega)}{\partial \theta}\} \frac{d\theta}{d\omega} + \frac{\partial f(\tau, \theta, \omega)}{\partial \omega} = \mu \frac{d\theta}{d\omega} + \frac{\partial f(\tau, \theta, \omega)}{\partial \omega} \approx \frac{\partial f(\tau, \theta, \omega)}{\partial \omega}. \tag{12}$$

This leads us to

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

and the update equation of temperature parameter becomes:

$$\omega \leftarrow \omega + \Delta\omega = \omega - \beta \frac{\partial J'(\tau', \theta', \overline{\omega})}{\partial \omega}$$

$$= \omega - \beta \frac{\partial J'(\tau', \theta', \overline{\omega})}{\partial \theta'} \frac{\partial}{\partial \omega} \{-\alpha \frac{\partial J(\tau, \theta, \omega)}{\partial \theta}\}. \tag{14}$$

3.2 Adaptive Temperature Parameter Tuning of Mellowmax DQN

We now present the adaptive online algorithm that tunes the temperature parameter of Mellowmax operator in DQN. The procedure is detailed in Algorithm 2.

Algorithm 2 Adaptive Mellowmax Deep Q-Network with Meta-Gradient

Initialize temperature parameter $\omega = \omega_0$, and set a reference temperature parameter $\overline{\omega}$
Initialize experience replay memory D, set learning rate α, set meta-learning rate β
Initialize action-value network Q with random weights θ and target network $\widehat{Q}$ with $\theta^- = \theta$
for episode = 1 to M **do**
  Initialize the starting state s_1
  for t= 1 to T **do**
    Select a random action a_t with probability ϵ. Otherwise, $a_t = \arg\max_a Q(s_t, a; \theta)$
    Execute action a_t, observe reward r_t, and obtain the next state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in D
    Sample random batch $\tau = (s_j, a_j, r_j, s_{j+1})$ from D for value function update
    Sample another random batch $\tau' = (s_k, a_k, r_k, s_{k+1})$ from D for meta-parameter update
    Compute the objective function $J = \frac{1}{2}[r_j + \gamma mm_{\omega}_{a'} \widehat{Q}(s_{j+1}, a'; \theta^-) - Q(s_j, a_j; \theta)]^2$
    Update parameters: $\theta' \leftarrow \theta - \alpha \frac{\partial J(\theta)}{\partial \theta}$
    Compute the meta-objective
    $J'(\tau', \theta', \overline{\omega}) = \frac{1}{2}[r_k + \gamma mm_{\overline{\omega}}_{a'} \widehat{Q}(s_{k+1}, a'; \theta^-) - Q(s_k, a_k; \theta')]^2$
    Update meta-parameter $\omega' \leftarrow \omega - \beta \frac{\partial J'(\tau', \theta', \overline{\omega})}{\partial \omega}$
    Every C steps, copy $\widehat{Q} = Q$
  end for
end for

Implementation Since we are adapting the meta-parameter temperature ω, the return function becomes non-stationary and often results in instability during training. As in previous work [13], we borrow the idea of universal value function approximation [10] to alleviate the potential instability that can be caused by non-stationary return function. We use a one-dimensional embedding vector E, whose size is n_e (where $n_e < |S|$). We output $e_\omega = E(\omega)$, and concatenate this vector to the state vector s. The input of Q-function now becomes $[s, e_\omega]$, and we update the embedding vector E every iteration along with θ.

We must also specify the loss function and trainable variables for the meta-update process, as we update θ for neural network Q. We set the loss function as

$$K = \underbrace{\frac{\partial J'(\tau', \theta', \overline{\omega})}{\partial \theta'}}_{A} \underbrace{\{-\alpha \frac{\partial J(\tau, \theta, \omega)}{\partial \theta}\}}_{B}, \tag{15}$$

then the gradient of K with respect to ω becomes:

$$\frac{\partial K}{\partial \omega} = A\frac{\partial B}{\partial \omega} + B\frac{\partial A}{\partial \omega} = A\frac{\partial B}{\partial \omega}, \tag{16}$$

since A is independent of ω, and $\frac{\partial A}{\partial \omega}$ is zero. Thus, we get

$$\frac{\partial K}{\partial \omega} = A\frac{\partial B}{\partial \omega} = \frac{\partial J'(\tau', \theta', \overline{\omega})}{\partial \theta'} \frac{\partial}{\partial \omega}\{-\alpha \frac{\partial J(\tau, \theta, \omega)}{\partial \theta}\}, \tag{17}$$

and

$$\omega' \leftarrow \omega - \beta\frac{\partial K}{\partial \omega} = \omega - \beta\frac{\partial J'(\tau', \theta', \overline{\omega})}{\partial \theta'} \frac{\partial}{\partial \omega}\{-\alpha \frac{\partial J(\tau, \theta, \omega)}{\partial \theta}\}, \tag{18}$$

which is now the same form as Equation 14.

4 Preliminary Empirical Results

We now present preliminary empirical results on Acrobot, a simple openAI gym control domain. We tested performance in two different versions of Mellowmax-DQN: a) the ordinary version where the max function is replaced by Mellowmax as mentioned above, and b) DeepMellow [5], where the target network update frequency C is set to 1.

Experimental Details We used three layers of dense neural networks to approximate the value functions Q and the target network $\widehat{Q}$. Each neural network is composed of 500 neurons of 3 hidden layers, and after each hidden layer, ReLU activation function is applied. The learning rate (α) for Q-function update is 10^{-4} and the meta-learning rate (β) for temperature update is 10^{-5}. The batch-size for Q-function update is 64, while the batch-size for meta-update is 8. We used ADAM [6] for both network-optimizer and meta-optimizer. Meta-update occurs along with every agent update, and the target network frequency C is set to 100 for the test (a), and 1 for the test (b), as mentioned above. For each graph plot, we averaged 20 independent runs with random seeds. The length of one-dimensional embedding vector E (embedding size) is 3, and the trace decay (μ) is 0.

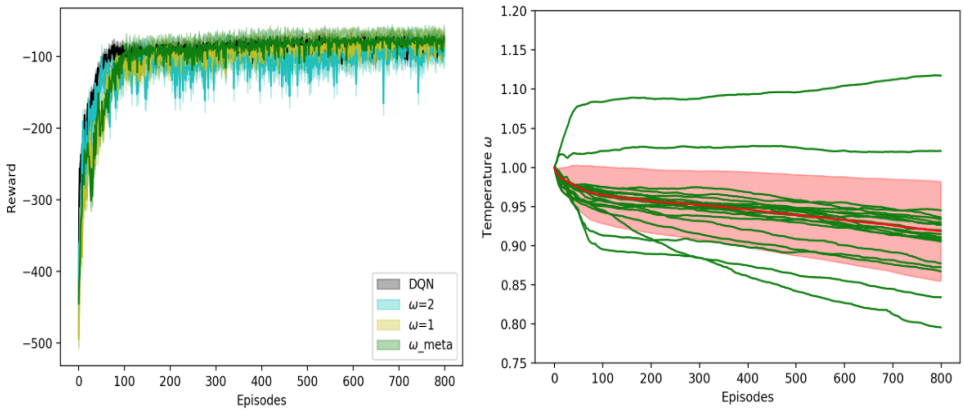

Figure 1: [Left] Mellowmax-DQN with meta-gradient that adaptively tunes temperature parameter (green), compared with Mellowmax-DQN with fixed temperature values (yellow, cyan) and DQN (black). [Right] The evolution curves (green) of temperature ω parameter values during training (20 independent runs) using meta-gradients. The red curve represents the average over 20 random seeds, and red shaded region covers standard deviation.

Results First, we observed the performance of adaptive Mellowmax-DQN with meta-gradients, setting the initial and reference temperature value as $\overline{\omega} = 1$. We compared it against the performance of Mellowmax-DQN with fixed temperature $\omega = 1, 2$ and DQN. (We did grid-search on the choice of temperature, trying $\omega \in \{0.25, 0.5, 0.75, 1, 2, 3, 5, 10\}$ and we use the best two values: $\omega \in \{1, 2\}$ for comparison.) The target network update frequency C was set to 100. Figure 1 (left) shows the rewards of each algorithm over 800 episodes. Adaptive Mellowmax-DQN, which starts with $\overline{\omega} = 1$ as initial value and tunes its parameter along the training, yields higher cumulative rewards than the Mellowmax-DQN with fixed temperature settings ($\omega = 1, 2$). It also showed comparable performance with DQN. In order to compare the scores of each algorithm, we computed the sum of areas under each curve (y-axis lower bound is -500). Setting the areas under the average curve of DQN graph as 100, the Mellowmax-DQN with fixed parameter $\omega = 1, 2$ achieves 95.2% and 93.7%, respectively. The adaptive Mellowmax-DQN achieves 98.7%, almost as good as DQN.

Figure 1 (right) shows the evolution curves of the temperature parameter ω of 20 independent runs (green) and their average (red). The initial temperature parameter is $\overline{\omega} = 1$, and the agent learns to adapt the temperature parameter to optimize its results. Although there are several outliers, the final end result of adapted ω were between 0.90 and 0.95 (after 800 episodes).

The next case shows more promising results of our approach: we observed the performance of this algorithm when the target network update frequency C is set to 1. Previous work [5] showed that Mellowmax, rather than the max operator, makes DQN perform better in the absence of a separate target network (or equivalently, when target network update frequency C=1). In addition to the previous work, we present the performance of Mellowmax-DQN with adaptively tuned temperature parameter by comparing it against DQN (C=1) and Mellowmax-DQN with fixed temperature values. We did grid-search on the choice of ω, testing $\omega \in \{0.25, 0.5, 0.75, 1, 2, 5, 8, 10, 20\}$ and used best values $\omega = 0.75, 1, 2, 5$ for comparison. The initial, reference value for temperature parameter is $\overline{\omega} = 1$ as in the previous case.

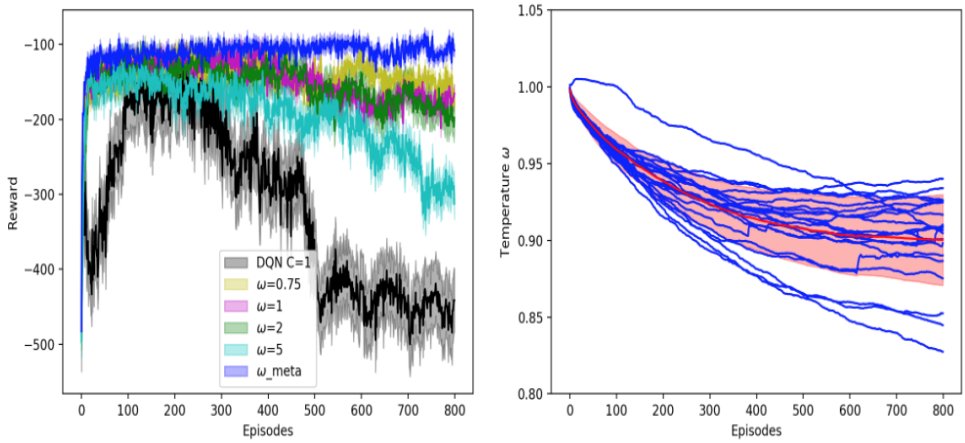

Figure 2: [Left] Mellowmax-DQN and DQN in the absence of a separate target network. Adaptive Mellowmax-DQN with meta-gradients (blue) achieves best performance when compared against Mellowmax-DQN with fixed temperature values (yellow, purple, green, cyan) and DQN (black). [Right] The evolution curves (blue) of ω during training (20 independent runs) using meta-gradients. The red curve is the average over 20 random seeds, and the shaded region covers standard deviation.

Figure 2 (left) shows that Mellowmax-DQN with adaptively tuned temperature ω performs better than *all* of the previous results. In the absence of a separate target network, Mellowmax-DQN with fixed ω performs better than DQN ($\omega = 0.75$ performs better than $\omega = 1, 2$, which perform better than $\omega = 5$), and yet Mellowmax-DQN with meta-gradients performs even better than fixed $\omega = 0.75$, while adapting the temperature parameter ω. We plotted the evolution curves of ω of each trial as well as their averaged curve in Figure 2 (right). Starting with initial value $\overline{\omega} = 1$, the agent learns to optimize its temperature parameter ω, generally showing a slow descending trend, and maximizes its cumulative reward.

5 Discussion and Future Work

We proposed an approach to adaptively tune the temperature parameter ω of Mellowmax operator in deep reinforcement learning. Although Mellowmax can replace the max function in the Bellman update equation of DQN and serve as an alternative tool for value function estimations, choosing the additional temperature parameter ω required an expensive grid-search. Our algorithm uses the meta-gradient reinforcement learning, which learns to optimize the return function by tuning additional meta-parameters. Since the Mellowmax operator is differentiable with respect to its temperature hyperparameter ω, we can set it as a tunable meta-parameter and run the adaptive algorithm with additional meta-gradients.

We presented the empirical results of this approach in a simple control domain, Acrobot. We showed that the Mellowmax-DQN with adaptively tuned ω parameter performs better than the previous version of algorithm with fixed ω, and reduces the need for an exhaustive search for ω.

The experiments presented in this paper are preliminary. Since we have confirmed the potential of this approach on a simple control domain, we plan to test its performance on larger-scale domains. The next step is to test this algorithm on middle-scale control domains (e.g. Lunar Lander) and large-scale domains like Atari games [2] with continuous state and discrete action spaces.

Another line of possible future research direction is to investigate the adaptive overestimation bias reduction in deep Q-learning. It has been theoretically shown that Mellowmax can reduce the overestimation bias in value estimations [5]. The key element of reducing overestimation bias through Mellowmax is the choice of good temperature parameter ω. If we could make this algorithm work on large-scale Atari domains, we could compare its performance with well-known Double-DQN [12] algorithm, which addresses the overestimation bias problem in deep reinforcement learning.

References

[1] Kavosh Asadi and Michael L. Littman. An Alternative Softmax Operator for Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 243–252, 2017.

[2] 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.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *CoRR*, abs/1606.01540, 2016.

[4] 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 *Proceedings of the International Conference on Machine Learning*, pages 1133–1142, 2018.

[5] Seungchan Kim, Kavosh Asadi, Michael L. Littman, and George Konidaris. DeepMellow: Removing the Need for a Target Network in Deep Q-Learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 2733–2739, 2019.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level Control through Deep Reinforcement Learning. *Nature*, 518(7540):529–533, 2015.

[8] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the Gap Between Value and Policy Based Reinforcement Learning. In *Advances in Neural Information Processing Systems*, pages 2775–2785, 2017.

[9] Gergely Neu, Anders Jonsson, and Vicenç Gómez. A Unified View of Entropy-Regularized Markov Decision Processes. *arXiv preprint arXiv:1705.07798*, 2017.

[10] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal Value Function Approximators. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1312–1320, 2015.

[11] Zhao Song, Ronald Parr, and Lawrence Carin. Revisiting the Softmax Bellman Operator: New Benefits and New Perspective. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 5916–5925, 2019.

[12] Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2094–2100, 2016.

[13] Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-Gradient Reinforcement Learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 2402–2413, 2018.