Removing the Target Network from Deep Q-Networks with the Mellowmax Operator

Extended Abstract

Seungchan Kim, Kavosh Asadi, Michael Littman, George Konidaris
Brown University
Providence, Rhode Island
seungchan_kim@brown.edu,k8@brown.edu,mlittman@cs.brown.edu,gdk@cs.brown.edu

ABSTRACT

Deep Q-Network (DQN) is a learning algorithm that achieves human-level performance in high-dimensional domains like Atari games. We propose that using an softmax operator, Mellowmax, in DQN reduces its need for a separate target network, which is otherwise necessary to stabilize learning. We empirically show that, in the absence of a target network, the combination of Mellowmax and DQN outperforms DQN alone.

ACM Reference Format:
Seungchan Kim, Kavosh Asadi, Michael Littman, George Konidaris. 2019. Removing the Target Network from Deep Q-Networks with the Mellowmax Operator. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 3 pages.

1 INTRODUCTION

Reinforcement learning [11] is a standard framework for studying agents that learn to make sequential decisions to maximize a utility function in potentially stochastic environments. Recent breakthroughs have shown that the simple Q-learning algorithm can yield human-level performance in complex domains (e.g., Atari games [2]) when combined with deep neural networks. The first algorithm to successfully instantiate this combination was the Deep Q-Network (or simply DQN) [7].

An important element of DQN is a target network, a technique introduced to stabilize learning. A target network is a copy of the action-value function (or Q-function) that is held constant to serve as a stable target for learning for some fixed number of timesteps. However, the use of a target network moves us farther from online learning, a desired property in reinforcement learning [10–12]. Our goal is to reduce the need for a target network in DQN while ensuring stable learning and good performance.

Our results show that the recently proposed action-selection operator, Mellowmax [1], reduces the need for a target network. We incorporate the Mellowmax operator into DQN, and propose the Mellowmax-DQN (MMDQN) algorithm. We test the performance of MMDQN in Acrobot, Lunar Lander, and Seaquest, and empirically show that MMDQN performs better than DQN in the absence of a target network.

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

2 MELLOWMAX-DQN

2.1 Mellowmax Operator

In reinforcement learning, softmax operators have been used to trade off exploration (trying new actions) and exploitation (trying good actions). We use softmax in the context of value-function optimization, where we focus on the Mellowmax operator [1] in particular:

$$mm_\omega(x) := \frac{\log(\frac{1}{n}\sum_{i=1}^{n}\exp(\omega x_i))}{\omega}. \quad (1)$$

This recently proposed operator yields convergent behavior in learning and planning due to its non-expansion property, has an entropy-regularization interpretation [4, 8, 9, 13], and facilitates convergent off-policy learning even with non-linear function approximation [3]. Additionally, the temperature parameter ω offers interpolation between max ($\omega \to \infty$) and mean ($\omega \to 0$).

2.2 Deep Q-Network

Deep Q-Network (DQN) [7] is a variation of simple Q-learning with three modifications: (1) it uses deep neural networks to approximate the Q-function, (2) it samples a random minibatch of transitions from experience replay buffer as training data, and (3) it employs a target network that delays the update of target values to increase learning stability. The update equation of DQN is

$$\theta \leftarrow \theta + \alpha\big(r + \gamma \max_{a'} \widehat{Q}(s', a'; \theta^-) - Q(s, a; \theta)\big)\nabla_\theta Q(s, a; \theta)\,, \quad (2)$$

where the real action-value Q is parameterized by θ while the separate target network $\widehat{Q}$ is parameterized by θ^-. The separate weights θ^- is synchronized with θ after periodic updates. This process prevents divergence, because it fixes $\widehat{Q}$ which can serve as a stable target during updates. However, note that using a target network adds a delay between the time that Q is updated and the time $\widehat{Q}$ is updated. We aim to show that it is possible to remove target network from DQN, while still ensuring stable learning and good performances.

2.3 MMDQN

We combine the Mellowmax operator and DQN, and propose the Mellowmax-DQN (MMDQN) algorithm. MMDQN replaces the max operator in the update equation above with Mellowmax, as in the framework of generalized MDPs [6]. MMDQN further differs from DQN as it does not use a separate target network $\widehat{Q}$ and its weights θ^-. Thus, the update equation of MMDQN is

$$\theta \leftarrow \theta + \alpha\big(r + \gamma\, mm_\omega \underset{a'}{Q}(s', a'; \theta) - Q(s, a; \theta)\big)\nabla_\theta Q(s, a; \theta)\,. \quad (3)$$

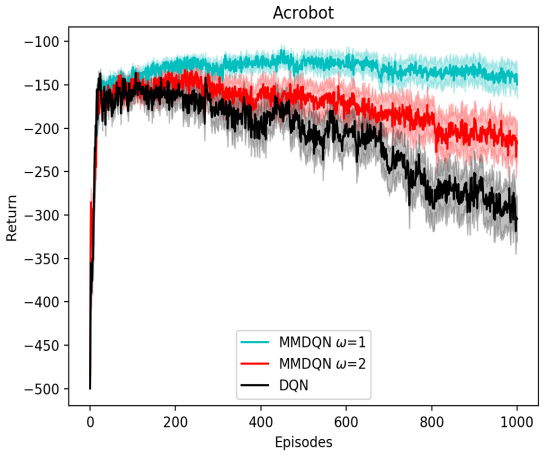

Figure 1: MMDQN ($\omega = 1, \omega = 2$) and DQN with no target network in Acrobot

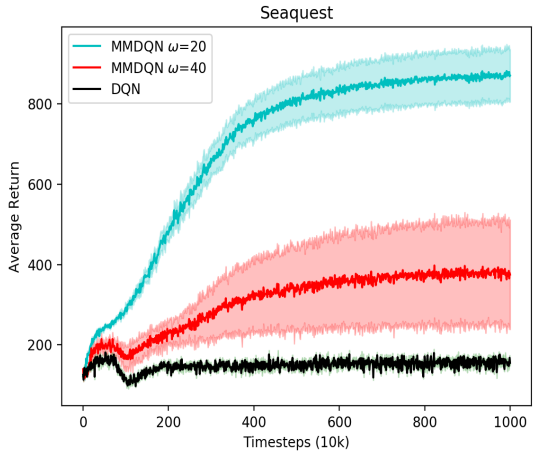

Figure 3: MMDQN ($\omega = 20, \omega = 40$) and DQN with no target network in Seaquest

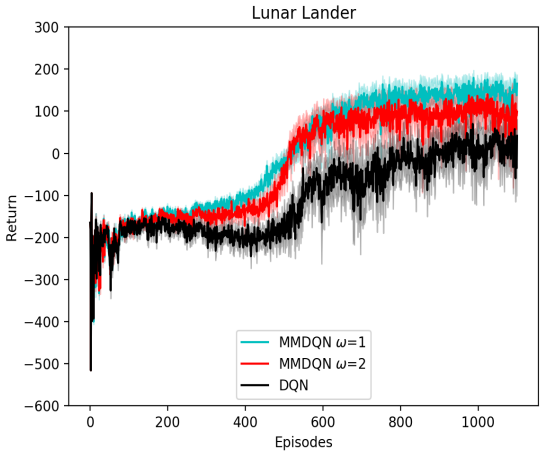

Figure 2: MMDQN ($\omega = 1, \omega = 2$) and DQN with no target network in Lunar Lander

3 EXPERIMENTS

We chose two control domains (Acrobot, Lunar Lander) and one Atari game domain (Seaquest) to test the performances of MMDQN. For the control domains, we searched for the optimal hyperparameters and architectures by testing different sizes of deep neural networks (number of hidden layers $\in \{1, 2, 3\}$, number of neurons per layer $\in \{100, 200, 300, 500\}$) to approximate the Q-function and testing different learning-rate values $(0.001, 0.0005, 0.0001)$. We found that the best parameters were 3 layers, 300 neurons per layer, and a learning rate=0.0005 for Acrobot, and 3 hidden layers, 500 neurons per layer, and a learning rate=0.0001 for Lunar Lander. For Seaquest, we used an open source DQN implementation as a baseline [5], and modified the code to tune the hyperparameters as published in the original DQN paper [7]. We ran 100 trials for Acrobot, 50 trials for Lunar Lander, and 5 trials for Seaquest, respectively, averaging their returns to obtain final mean scores.

4 RESULTS

We compared the performances of MMDQN and DQN in the absence of a target network. Figure 1 shows that the MMDQN achieves more stability than DQN in Acrobot. The learning curve of DQN goes upward fast, but soon starts fluctuating and fails to improve towards the end. By contrast, MMDQN (ω=1) succeeds in the absence of a target network. We quantified the performance of MMDQN and DQN by computing the sum of areas under their learning curves. Setting the areas under the curves of MMDQN ($\omega = 1$) as 100, the areas under the curves of MMDQN ($\omega = 2$) and DQN were 88.9% and 78.7%, respectively. Similar results were observed in Lunar Lander (Figure 2): MMDQN ($\omega = 1, 2$) achieves more stable learning and higher average returns than DQN. The areas under the curves of MMDQN ($\omega = 2$) and DQN were 93.4% and 79.2% of that of MMDQN ($\omega = 1$).

Finally, in Seaquest, the performance gaps between MMDQN and DQN widened, as shown in Figure 3. DQN without a target network was unable to learn, but MMDQN (ω=20) learned stably. The areas under the curves of MMDQN ($\omega = 40$) and DQN were 47.5% and 22.8% of that of MMDQN ($\omega = 20$).

5 CONCLUSION

Mellowmax-DQN (MMDQN) combines the Mellowmax operator and Deep Q-Network. MMDQN also does not use a separate target network; it eliminates the delays of action-value updates caused by a target network, and moves us closer to online reinforcement learning. In multiple domains, our experimental results showed that MMDQN achieves stable learning and outperforms DQN in the absence of a target network.

REFERENCES

[1] Kavosh Asadi and Michael L. Littman. 2017. An Alternative Softmax Operator for Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 243–252.

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

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

[4] Roy Fox, Ari Pakman, and Naftali Tishby. 2016. Taming the Noise in Reinforcement Learning via Soft Updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, UAI 2016, June 25-29, 2016, New York City, NY, USA*.

[5] Taehoon Kim. 2018. Deep Reinforcement Learning in Tensorflow. https://github.com/carpedm20/deep-rl-tensorflow. (2018).

[6] Michael L. Littman and Csaba Szepesvári. 1996. A Generalized Reinforcement-Learning Model: Convergence and Applications. In *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*. 310–318.

[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. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[8] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. 2017. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*. 2775–2785.

[9] Gergely Neu, Anders Jonsson, and Vicenç Gómez. 2017. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798* (2017).

[10] Harm Seijen and Rich Sutton. 2014. True online TD (lambda). In *International Conference on Machine Learning*. 692–700.

[11] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction*. MIT Press.

[12] Hado van Hasselt and Richard S Sutton. 2015. Learning to predict independent of span. *arXiv preprint arXiv:1508.04582* (2015).

[13] Brian D Ziebart. 2010. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. (2010).