Bellman Momentum on Deep Reinforcement Learning

Extended Abstract

Huihui Zhang
Dongsheng Intelligent Technolody Co., Ltd.
Suzhou, China
huihuizhang2014@gmail.com

ABSTRACT

The sable point may pretend to be optimal and will further degrade the asymptotical performance of the whole training task. We try to solve this problem by seeking more aspects to prepare effective policy regularization, which will provide better policy exploration when faced with suboptimal stable points. As we know, the action is a multidimensional vector with each element as a random variable, so their probabilities compose a vector that can indicate some direction, which is exactly the information we can utilize.

KEYWORDS

Off-policy, Reinforcement learning, Policy exploration

ACM Reference Format:
Huihui Zhang. 2024. Bellman Momentum on Deep Reinforcement Learning: Extended Abstract. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 3 pages.

1 INTRODUCTION

The function approximation is usually adopted to deal with continuous control problems . Under this setting, the actor-critic approach is unavoidably employed to get the flexibility of a two-scale function approximation, which further induce the saddle point problem due to the iterative update of two parameters. Since saddle points share the property of zero gradient with the optimal point, some of them can be stable and disguise to be optimal, which will also restrict the asymptotical performance of training. Besides, the intrinsic noise accompanying the function approximation will make the value estimates even more inaccurate and exaggerate the harm of saddle points. Just like the countless suboptimal points that can be hardly detected, the saddle point problem is difficult to solve, since RL can be seen as a "black box" that can only be partially observed by inputting some actions.

The purpose of this work is to find ways to tackle the suboptimal and saddle points to improve the asymptotical performance as well as not losing the accuracy of value estimates.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

2 METHODS

2.1 Momentum Value Iteration

We model the value penalty as the cosine similarity between the target momentum and the direction of target policy that is a vector composed of the probabilities of all action parts. Given the MDP denoted by $(\mathcal{S}, \mathcal{A}, p, r)$, a modified Bellman backup operator $\mathcal{T}^\pi$ can be given by

$$\mathcal{T}^\pi Q(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}} \left[Q^t(s_{t+1}, a_{t+1}) \right], \quad (1)$$

where $a_{t+1} \sim \pi(\cdot|s_{t+1})$, and $Q^t(s_{t+1}, a_{t+1})$ is the sum of $Q(s_{t+1}, a_{t+1})$ and $\beta \cos(\vec{\pi}(a_{t+1}|s_{t+1}), \vec{M}(s_{t+1}, a_{t+1}))$. β is the hyperparameter that is used to weight the MMTC term, $\cos(\cdot, \cdot)$ represents the cosine similarity operator, $\vec{\pi}$ represents the vector whose elements are the probabilities of actions following the target policy, and $\vec{M}$ shows the momentum.

2.2 Momentum Regularization

We optimize the policy to maximize the combined value of the expected return and the cosine similarity between the policy direction and the momentum as

$$\max_\theta \mathbb{E}_s \left[Q(s, \pi_\theta(s)) + \beta \cos(\vec{\pi}_\theta(\pi_\theta(s)|s), \vec{M}(s, \pi_\theta(s))) \right], \quad (2)$$

where π_θ is the policy approximation parameterized by θ. The cosine similarity can be computed as

$$\cos = \frac{< \vec{\pi}_\theta(\pi_\theta(s)|s), \vec{M}(s, \pi_\theta(s))) >}{|\vec{\pi}_\theta(\pi_\theta(s)|s)| \left|\vec{M}(s, \pi_\theta(s)))\right|}, \quad (3)$$

where cos is short for $\cos(\vec{\pi}_\theta(\pi_\theta(s)|s), \vec{M}(s, \pi_\theta(s)))$, the computation is just the inner product of two vectors divided by their norms.

2.3 Momentum Constrained Algorithm

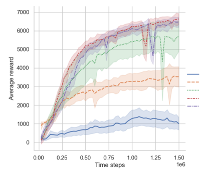
Given the current and target networks of both the Q-value and the policy, the loss function of updating critic parameters can be estimated at the policy evaluation step as

$$L(\omega) = \mathbb{E}_{(s,a,r,s')} \left[\frac{1}{2} (r + \gamma Q^t_{\omega'}(s', a') - Q_\omega(s, a))^2 \right], \quad (4)$$

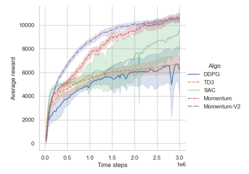
$$Q^t_{\omega'}(s', a') = Q_{\omega'}(s', a') + \beta \cos(\vec{\pi}_{\theta'}(a'|s'), \vec{M}(s', a')), \quad (5)$$

where $a' = \pi_{\theta'}(s')$ is the action following the target policy parameterized by θ', (s, a, r, s') is sampled from the replay buffer, ω and ω' parameterize the critic network and its target estimate, respectively, and $\pi_{\theta'}(\cdot|s')$ is the target policy pdf conditioned on the next state s'.

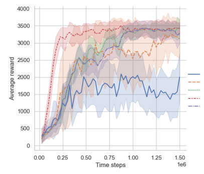
The surrogate objective function to update the current actor parameter θ at the expected policy improvement step can be given



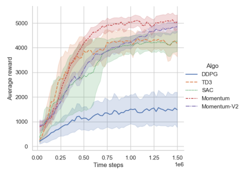
(a)



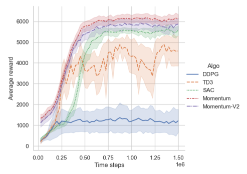
(b)



(c)



(d)



(e)

Figure 1: Average reward versus time step in (a) Ant; (b) Halfcheetah; (c) Hopper; (d) Walker2d; (e) Humanoid

by

$$J(\theta) = \mathbb{E}_s \left[Q_\omega(s, \pi_\theta(s)) + \beta \cos(\vec{\pi}_\theta(\pi_\theta(s)|s), \vec{M}(s, \pi_\theta(s))) \right], \quad (6)$$

where $a = \pi_\theta(s)$ is the reparameterized action with parameter θ based on s sampled from the tuple of experience dataset. We organize the above procedures as the MMTC algorithm, whose pseudocode is described by Algorithm 1.

Algorithm 1 MMTC Algorithm

```
1:  Initialize parameters $\omega \leftarrow \omega_0$, $\theta \leftarrow \theta_0$
2:  Initialize target parameters $\omega' \leftarrow \omega'_0$, $\theta' \leftarrow \theta'_0$
3:  Initialize the learning rates $l_c$, $l_a$ for the critic and the actor,
    the time step $t \leftarrow 0$, the soft update hyperparameter $\tau$, the
    maximum time step $T$, the batch size $B$ and the replay buffer
    $\mathcal{D} \leftarrow \emptyset$.
4:  while $t < T$ do
5:      Select action $a_t \sim \pi_{\theta_t}(a_t|s_t)$
6:      Observe the reward and next state $s_{t+1}, r_t \sim T(s_{t+1}|s_t, a_t)$
7:      Store transition $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
8:      Sample a batch of transitions $\mathcal{B} = (s, a, r, s')_{i=1}^{B}$ from $\mathcal{D}$
9:      for each time step do
10:         $\omega_{t+1} \leftarrow \omega_t - l_c \nabla_{\omega_t} L(\omega_t)$ following (4)
11:         $\theta_{t+1} \leftarrow \theta_t + l_a \nabla_{\theta_t} J(\theta_t)$ following (6)
12:         Update ${\omega'}_{t+1}$ and $\theta'_{t+1}$ following soft update [4]
13:     end for
14:     $s_{t+1} \leftarrow s_t$
15:     $t \leftarrow t + 1$
16: end while
```

2.4 Momentum Design

One option to model the momentum is

$$\vec{M}(s, a) = \nabla_a Q_\omega(s, a). \quad (7)$$

For better stability, we prefer to adopt a discounted cumulative momentum as

$$\vec{M}(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_{a_t} Q_\omega(s_t, a_t) | s_0 = s, a_0 = a \right]. \quad (8)$$

We can get the Bellman property of momentum as

$$\vec{M}(s, a) = \nabla_a R(s, a) + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}} \left[\vec{M}(s_{t+1}, a_{t+1}) \right], \quad (9)$$

where $R(s, a)$ represents the reward function.

3 EXPERIMENTS

3.1 Benchmarks and Baselines

To evaluate the performance of our proposed algorithm, we adopt several baselines including DDPG [4], TD3 [1] and SAC [2, 3] to compare the sample efficiency and stability.

3.2 Results

Figures from Figs. 1(a)-1(e) show the evaluation of MTC algorithm in comparison with the baselines, where the label ”Momentum” represents the proposed MTC algorithm using the discounted cumulative partial gradient of the Q-value function with respect to the action in (8). According to the observation of Fig. 1(c), we can see that the MTC algorithm is more stable than other selected baselines, which is reflected on the fluctuations of algorithm curves. Confidence interval (CI) is a criterion to measure the variance of average reward. We can see from Figs. 1(a)-1(e) that CI of our MTC algorithm is less than that of other baselines. Another observation from Figs. 1(a)-1(e) is noticeably shown by the converged value of MTC algorithm, which is much higher than the selected baselines, especially for the Walker2d task. This observation is understandable because the MTC algorithm is able to skip subtpimal points and arrive much closer to the optimal point, which is contributed by the momentum constrained regularization formulated by (2). By these empirical evaluation, guiding the policy direction toward the momentum defined in (8) is beneficial to avoiding suboptimal points as well as improving the stability, and it can be realized by maximizing the cosine similarity between the policy direction and the defined momentum.

We also give the empirical results of momentum algorithm under the definition of (7), which is labeled as ”Momentum-V2” in Figs. 1(a)-1(e). We can see that except for the Halfcheetah task, ”Momentum-V2” has a bit inferior performance than MTC algorithm. According to empirical attempts, Halfcheetah is much easier for algorithms to converge than other benchmarks, which may be the reason for this observation.

ACKNOWLEDGMENTS

This work is supported by Dongsheng Intelligent Technolody Co., Ltd., a rising company that is located in Suzhou and aims to develop full potential of every employee. I would like to thank my boss Xu Han, who initiated the company, and my colleagues Yanlong Cheng and Cong Yan.

REFERENCES

[1] Scott Fujimoto, Herke Van Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. 80 (2018), 1587–1596.

[2] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. (2018), 1861–1870.

[3] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905* (2018).

[4] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).