UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING
Department of Electrical and Computer Engineering

MONASH University

The Effect of Multi-step Methods on Overestimation in Deep Reinforcement Learning

Lingheng Meng, Rob Gorbet, Dana Kulić

Abstract

Multi-step (also called *n*-step) methods in Reinforcement Learning (RL) have been shown to be more efficient than the 1-step method due to faster propagation of the reward signal, both theoretically and empirically, in tasks exploiting tabular representation of the value-function. Recently, research in Deep Reinforcement Learning (DRL) also shows that multi-step methods improve learning speed and final performance in applications where the value-function and policy are represented with deep neural networks. However, there is a lack of understanding about what is contributing to the boost of performance. In this work, we analyze the effect of multi-step methods on alleviating the overestimation problem in DRL, where multi-step experiences are sampled from a replay buffer. Specifically building on top of Deep Deterministic Policy Gradient (DDPG), we propose Multi-step DDPG (MDDPG), where different step sizes are manually set, and a variant called Mixed Multi-step DDPG (MMDDPG) where an average over different multi-step backups is used as an update target for the Q-value function.

Motivation

> Multi-step (also called n-step) methods in Reinforcement Learning (RL), with tabular representation of the value-function, have been shown to be more efficient than the 1-step method due to faster propagation of the reward signal.

> Research in Deep Reinforcement Learning (DRL), with value-function and policy approximated by deep neural networks, shows that multi-step methods improve learning speed and final performance.

> However, there is a lack of understanding about what is contributing to the boost of performance of multi-step methods in DRL.

Background

Overestimation Problem [1]

Assume Q^{true} is represented by a function approximator Q^{approx} with noise $E(s', a')$:

$$Q^{approx}(s', a') = Q^{true}(s', a') + E(s', a')$$

Then, for Q-Learning technique

$$Q^{approx}(s, a) \leftarrow r(s, a) + \max_{a'} Q^{approx}(s', a')$$

zero-mean noise may easily result in overestimation problem because

$$\max_{a'} Q^{approx}(s', a') > \max_{a'} Q^{true}(s', a')$$

E.g., if

$$Q^{true}(s', a') = 0 \quad \text{and} \quad \mathbb{E}\left[E(s', a')\right] = 0$$

then

$$\begin{aligned}\max_{a'} Q^{approx}(s', a') \\ = \max_{a'} \left[Q^{true}(s', a') + E(s', a')\right] \\ = \max_{a'}(0 + E(s', a')) > 0\end{aligned}$$

while

$$\max_{a'} Q^{true}(s', a') = 0$$

Deep Deterministic Policy Gradient (DDPG) [2]

Critic, i.e. Q-value, is optimized by minimizing

$$L_{\theta^Q} = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim U(D)} \left[\left(\hat{Q}_t - Q_{\theta^Q}(s_t, a_t)\right)^2\right]$$

where $\hat{Q}_t = r_t + \gamma \max_{a \leftarrow \mu_{\theta^{\mu-}}(s_{t+n})} Q_{\theta^{Q-}}(s_{t+1}, a)$, $Q_{\theta^{Q-}}$ is target critic, and $\mu_{\theta^{\mu-}}$ is target actor representing the optimal policy.

Actor, i.e. policy, is optimized by maximizing

$$J_{\theta^\mu} = \mathbb{E}_{s_t \sim U(D)}\left[Q_{\theta^Q}(s_t, \mu_{\theta^\mu}(s_t))\right]$$

where Q_{θ^Q} and μ_{θ^μ} are online critic and actor.

Proposed Methods

Multi-step DDPG (MDDPG)

Bootstrapped target Q is based on multi-step immediate rewards

$$\hat{Q}_t^{(n)} = \begin{cases} \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \max_a Q_{\theta^{Q-}}(s_{t+n}, a), \\ \quad if\ \forall\ k \in [1, \cdots, n]\ \ and\ d_{t+k} \neq 1; \\ \sum_{i=0}^{k-1} \gamma^i r_{t+i}, \\ \quad if\ \exists\ k \in [1, \cdots, n]\ \ and\ d_{t+k} = 1. \end{cases}$$

where n indicates n immediate rewards are used.

Then, Q is optimized by minimizing

$$L_{\theta^Q} = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim U(D)} \left[\left(\hat{Q}_t - Q_{\theta^Q}(s_t, a_t)\right)^2\right]$$

Mixed Multi-step DDPG (MMDDPG)

- An average over target Q-values with different step sizes from 1 to n

$$\hat{Q}_t^{(n_{avg})} = \frac{1}{n} \sum_{i=1}^{n} \hat{Q}_t^{(i)}$$

- The minimum of a set of target Q-values

$$\hat{Q}_t^{(n_{min})} = \min_{i \sim [1,n]} \hat{Q}_t^{(i)}$$

- An average over target Q-values with step size from 2 to n, considering n= 1 is the most prone to overestimation:

$$\hat{Q}_t^{(n_{avg-1})} = \frac{1}{n-1} \sum_{i=2}^{n} \hat{Q}_t^{(i)}$$

Experiment Results

Experimental Evidence of Multi-step Methods' Effect on Alleviating Overestimation

From Fig. 1:

- Almost all MDDPG(n) with n >1 outperform DDPG
- Bad performance of DDPG corresponds to an extremely large Q-value

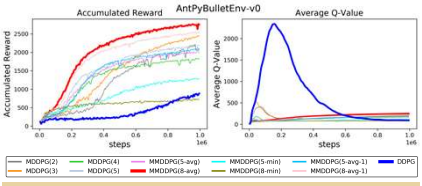

Fig. 1 Comparison among MDDPG, MMDDGP and DDPG on AntPyBulletEnv-v0, here for each task accumulated reward and average Q-value are shown side-by-side correspondingly to demonstrate the relationship between the overestimation of Q-value and performance.

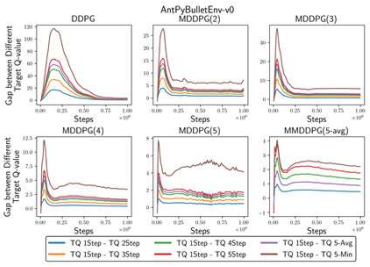

Fig. 2 The Difference in Estimated Target Q-values Between 1-step and Multi-step Methods, where the larger the value, the bigger the difference.

Four key characteristics can be observed in Fig. 2:

- All positive gaps means multi-step methods provide smaller estimated target Q-values than that of the 1-step method.
- The larger the step, the smaller the corresponding estimated target Q-value.
- The difference becomes smaller with increased interactions.
- The magnitude of the estimated Q-value decreases as the step size n increases.

Performance Comparison

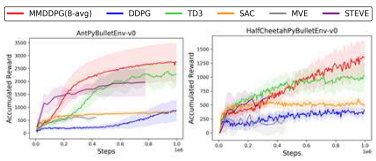

Fig. 3. Learning curves for PyBulletGym tasks.

Discussion

3 ways to calculate $\hat{Q}$ depending on how the experiences are acquired: (1) offline, (2) online, (3) model-based expansion.

$$\hat{Q}_t^{(n)}(s_t, a_t) = r_t + \overbrace{\underbrace{\sum_{k=1}^{n-1} \gamma^k r_{t+k}}_{underestimation\ prone}}^{offline,\ online,\ model} + \gamma^n \underbrace{\max_a Q_{\theta^{Q-}}(s_{t+n}, a)}_{overestimation\ prone}$$

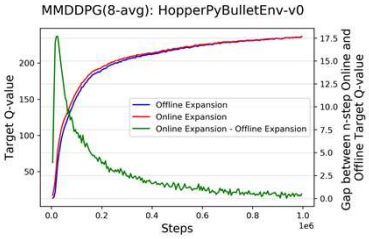

Fig. 4 Comparison between Online and Offline Multi-step Expansion, where the blue and the red line correspond to average of offline and online multi-step expansion over a mini-batch sampled from replay buffer, and the green line is the gap between them.

References

[1] S. Thrun, and A. Schwartz. "Issues in using function approximation for reinforcement learning." 1993.

[2] T. P. Lillicrap, J. J. Hunt, A. Pritzel, et al. "Continuous control with deep reinforcement learning." 2015.