Learning to walk in Bipedal Environments

Anonymous author

Abstract

This paper explores the possible algorithms which could be used to solve the BipedalWalker-v3 environments [9]. From our study, it is found that Truncated Quantile Critic [7] is successful at solving the easy environment with consistent mean rewards greater than 300 over the last 100 episodes. Although, we find that despite trying different parameters, we are not able to solve the Bipedal Hardcore environment according to Open AI's definition [10].However, we are able to consistently get mean rewards greater than 270 over the span of 100 episodes.

1 Methodology

As Bipedal environments don't provide an accurate model as part of the problem definition, we decided that 'model-free' algorithms would be best suited to solve our environment. Initially, we wanted to explore a solution using a deterministic policy. Therefore, we start with Twin Delayed DDPG(TD3) [11], which learns a deterministic policy π by maximising the Q-function. We noticed that the rewards stayed consistently around -80 initially. They started increasing around episode 200, suggesting that due to the deterministic nature of the actor's policy the trade-off balance between exploration and exploitation was not correct. Furthermore, the variance of rewards was very high, suggesting that the agent was overestimating its performance and exploring erroneous solutions. To mitigate this, we thought of using Soft actor critic (SAC) [5]. This is because SAC uses entropy regularisation to maximise the trade-off between expected return and entropy. SAC performed better, and had a better sampling efficiency. Although, there was still a high variance in the rewards. Literature suggests this to be from the overestimation of the Q-function [4]. Therefore to mitigate overestimation bias we implemented TQC based on Samsung labs implementation [8]. We experimented with the parameters of TQC. We experimented with number of approximations as [25, 30, 35]. This was coupled with the number of atoms dropped for each approximation from [2,3,4]. Also we varied the number of critics as [5,6,7]. We find that the optimal parameters were 25 number of approximations, 2 atoms dropped and 5 critics. Additionally, we increased the initial entropy temperature coefficient from e^0 in the original implementation to e^1, to promote greater exploration at the start. This proved useful in the hard environment. The rest of our best parameters were consistent to the parameters of the original implementation [8].

1.1 Soft Actor Critic (SAC)

SAC [5] optimises a stochastic policy(π_θ) in an off-policy manner. It utilises entropy regularisation and clipped double-Q trick [5]. Entropy is a gauge of the policy's randomness, which when learnt maximises the trade-off between exploration and exploitation [15]. It also prevents a convergence to a local minima. Also, with clipped double-Q trick [5] SAC learns two Q-functions $Q_{\phi 1}$ $Q_{\phi 2}$ simultaneously, and uses the smaller of the two Q-values to form the targets shown in equation 1.

$$y(r, s', d) = r + \gamma(1-d)(\min_{j=1,2} Q_{\phi_{targ,j}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')), \tilde{a}' \sim \pi_\theta(.|s') \tag{1}$$

Here, α refers to the entropy temperature coefficient. These targets are updated, and used to learn both Q-functions in the Mean-squared bellman error(MSBE) [15], shown in equation 2.

$$L(\phi_i, D) = \mathbb{E}_{(s,a,r,s',d)\sim D}[(Q_{\phi_i}(s,a) - y(r,s',d))^2] \tag{2}$$

1.2 Truncated Quantile Critic (TQC)

Off policy algorithms, such as SAC and TD3 [11] suffer from overestimation bias displayed in Figure 1. This can be explained using the Jensen inequality in equation 3.

$$\mathbb{E}_U[\max_a(Q(a) + U(a))] \geq \max_a \mathbb{E}_u[Q(a) + U(a)] = \max_a Q(a) \tag{3}$$

Equation 3 displays that the expected value of our estimated Q-function, over the action-dependent random noise $U(a)$ will always be greater than or equal to the expected value of $U(a)$. As a result, the agent overestimates it performance, and pursues erroneous estimates thus degrading performance.

To mitigate overestimation bias, TQC employs an ensemble of distributional critics, and

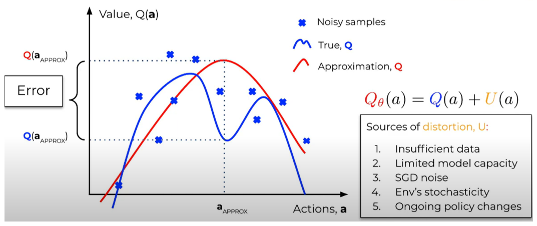

Figure 1: Diagram from [1], displaying the intuition behind the overestimation of the Q-function

truncates the right tail of these mixtures. This is an improvement because clipped double-Q learning parameters resulted in coarse bias control. For example, approximating and aggregation of 2 Q-functions might result in overestimation bias, whereas 3 Q-functions might result in underestimation bias.

Additionally, aggregating by using the minimum results in consideration of only one possible solution, which is wasteful. Thus, TQC takes a distributional approach (inspired from QR-DQN [2]). This is done by approximating the return random variable $Z^\pi(s,a) := \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$, as opposed to approximating expectation of the return (Q-function) $Q^\pi(s,a) = \mathbb{E}[Z^\pi(s,a)]$. This approach encourages the learning of intrinsic randomness ('aleatoric uncertainty' [7]) of the policy and the environment. Furthermore, we can utilise the granularity of distributional representation to gain more precise overestimation control.

We train N approximations $Z_{\psi 1}(s,a), ..., Z_{\psi N}(s,a)$, each with a mixture of M atoms. In equation 4, each Z_{ψ_n} maps the tuple (s,a) to a probability distribution on atoms $\theta^m_{\psi_n} \forall m \in M$.

$$Z_{\psi n}(s,a) := \frac{1}{M}\sum_{m=1}^{M} \delta(\theta^m_{\psi_n}(s,a)) \tag{4}$$

Where δ is the Dirac delta function is used to get points of non-zero distribution. The approximations in equation 4 are trained on the temporal difference target distribution $Y(s,a)$. To construct this, we pool the atoms of distributions $Z_{\phi i}(s',a') \forall i \in N$ into a set $Z(s',a')$.

$$Z(s',a') := \{\theta^m_{\phi n}(s',a') \mid n \in [1,N], m \in [1,M]\} \tag{5}$$

This set is then ordered in ascending order with respect to the atoms locations. The right tail of the set is then truncated, removing the atoms with largest locations. During this from the total $N \times M$ atoms, the largest d atoms from each approximation are removed (Figure 2). Thus leaving us with the smallest $K \times N$ atoms, where $k = M - d$. This provides fine-grained level of overestimation control, by varying parameters d and N. Using the smallest KN atoms, we define atoms on the target distribution, where α is the entropy temperature coefficient.

$$y_i(s,a) := r(s,a) + \gamma[z_i(s',a') - \alpha \log \pi_\phi(a'|s')] \mid z_i(s',a') \in Z(s',a') \tag{6}$$

Equation 6 is used to get the target distribution:

$$Y(s,a) := \frac{1}{KN}\sum_{i=1}^{KN} \delta(y_i(s,a)) \tag{7}$$

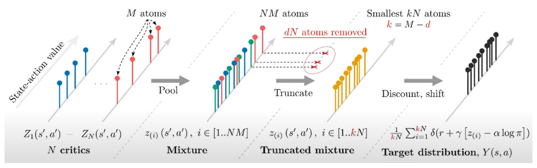
State-action value
M atoms
NM atoms
dN atoms removed
Smallest kN atoms
$k = M - d$
Pool
Truncate
Discount, shift
$Z_1(s', a') \cdots Z_N(s', a')$
N critics
$z_{(i)}(s', a'),\ i \in [1..NM]$
Mixture
$z_{(i)}(s', a'),\ i \in [1..kN]$
Truncated mixture
$\frac{1}{kN}\sum_{i=1}^{kN}\delta(r + \gamma[z_{(i)} - \alpha \log \pi])$
Target distribution, $Y(s, a)$

Figure 2: Diagram from [1], outlining the architecture of TQC

From this target distribution we estimate the Q-value by averaging the locations of the remaining kN atoms. with this truncated estimate, we also account for inflated overestimation due to a high variance in return. If the variance is large, the estimate of the Q-value would be lower. Although, in practice we populate $Z(s', a')$ with atoms predicted by the target network $Z_{\bar{\phi}i}(s', a')$ $s.t$ $i \in N$ to provide stability.
To train we minimise the 1-Wasserstein distance between each approximation $Z_{\psi n}(s, a)$, and the temporal difference (TD) target distribution $Y(s, a)$. To minimise this distance, we approximate the quantiles of the target distribution. Thus, learning the locations for the quantile fractions $t_m = \frac{2m-1}{2M}, \forall m \in M$. To approximate t_m, with $\theta^m_{\psi n}$, $m \in M$ by minimising the loss function over the parameters ψ_n, where L^k is shown in equation 9

$$J_z(\psi_n) = \mathbb{E}_{D,\pi}[L^k(s_t, a_t; \psi_n)] \tag{8}$$

$$L^K(s, a; \psi_n) = \frac{1}{KNM} \sum_{m=1}^{M} \sum_{i=1}^{KN} \rho^H_{t_m}(y_i(s, a) - \theta^m_{\psi_n}(s, a)) \tag{9}$$

With this we can also create a dependency between each learnable location $\theta^m_{\psi n}(s, a)$, and all atoms of the truncated mixture of target distribution. Thus reducing errors for all approximations.We optimise the policy parameters ϕ, by minimising the loss

$$J_\pi(\phi) = \mathbb{E}_{D,\pi}[\alpha \log \pi_\phi(a|s) - \frac{1}{NM} \sum_{m,n=1}^{M,N} \theta_{\psi n^m}(s, a)] \tag{10}$$

Note, we use non-truncated critics during policy optimisation, and truncated during value learning. This is because it prevents double truncation, whilst preventing propagation of errors through TD learning updates.

2 Convergence Results

Open AI defines the environment to be 'solved', when mean rewards over the previous 100 episodes is greater than 300 [10]. Our agent solves the BipedalWalker easy environment within 500 episodes (figure 4). Additionally, at later episodes the walker is able to utilise the full range of motion for both legs. Furthermore, the robot starts to learn to run after 600 episodes and the length of the videos start to decrease. The hardcore is a very challenging environment with obstacles. Hence, we initialise greater entropy temperature coefficient α for the walker, this is so that the walker explores more initially. Overtime , to overcome protruding obstacles the walker learns to lean back and raise its front leg. Similarly, for troughs in the landscapes, the robot learns to widen its legs, and lower its centre of mass to make itself more stable. However, despite these improvements we are not able to get mean rewards of 300, over the last 100 episodes in the hard environment, but there are several episodes where the walker is able to garner more than 300 in rewards. Finally, the variance of rewards is much greater in the hardcore environment, in comparison to the easy environment. The complexity and variability of the environment, which changes with each reset, can make it challenging to mitigate overestimation bias. Overall, TQC still shows a significant improvement over TD3 and SAC in terms of sampling efficiency and convergence. It also shows very promising signs in the hardcore environment.

3 Limitations

One of the limitations is the long training times for TQC. Agent in the easy environment requires a few hours of training to converge, and in the hardcore environment it requires

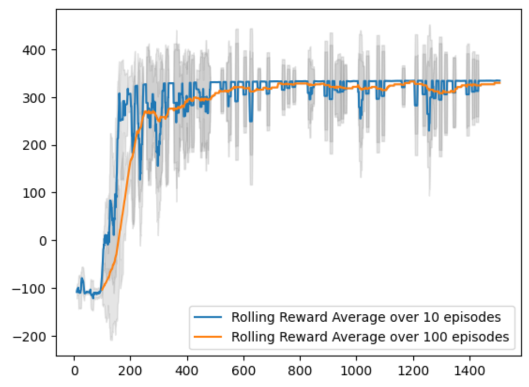

Figure 3: Rewards on the Easy environment

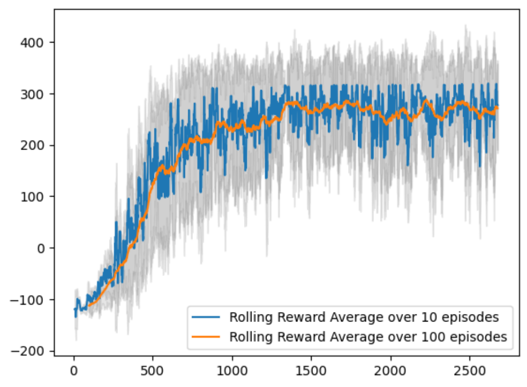

Figure 4: Rewards on the Hard environment

a day to train. Thus, we could only perform limited parameter tuning. TQC assumes that our agent is operating in a fully observable environment, although in the hardcore environment we are modelling a Partially Observable Markov Decision Process(POMDP) [6] as a MDP. This is because of the lack of knowledge about our observation design being complete or not. This means that our model does not account for all possible variance of the environment. This introduces additional room for error in our predictions. Finally, due to limited computational power we were not able to train our model for a large number of critics.

Future Work

In the future, we would like to model the hardcore problem as a POMDP, to do this we could add a memory component to our model. The memory would take into account the agent's past history. One such model is a Deep transformer Q-network (DTQN) [3], which leverages its transformer architecture [16] with learned positional encodings to represent our agent's history and predict the next Q-values based on the agent's history. We could potentially, use predictions from both TQC and DTQN, and aggregate them to output a more robust prediction. Additionally, to decrease training times we could implement a Loss-Adjusted Approximate Actor Prioritized Experience Replay (LA3P) [13], which improves on the shortfalls Prioritised replay buffer (PER) [14] faces on continuous environments like ours. With LA3P, we sample transitions with largest temporal difference (TD) errors. This can be thought of as 'hard negative mining' [12], as it ensures that our algorithm minimises the samples with the largest errors. This theoretically decreases training time and provides better sampling efficiency. Finally, to improve sampling efficiency we could save the weights from the easy environment and load them in when training the hard environment. By doing this the walker would not need to learn to walk and navigate the challenging terrain at the same time. Thus the walker could just focus on augmenting its walking mechanism to suit the hard environment.

References

[1] TQC authors. *Video explaination of TQC*. 2020. URL: `https://www.youtube.com/watch?v=jFyxsKBUJFQ` (visited on 02/16/2020).

[2] Will Dabney et al. "Distributional Reinforcement Learning with Quantile Regression". In: *CoRR* abs/1710.10044 (2017). arXiv: `1710.10044`. URL: `http://arxiv.org/abs/1710.10044`.

[3] Kevin Esslinger, Robert Platt, and Christopher Amato. *Deep Transformer Q-Networks for Partially Observable Reinforcement Learning*. 2022. DOI: `10.48550/ARXIV.2206.01078`. URL: `https://arxiv.org/abs/2206.01078`.

[4] Scott Fujimoto, Herke van Hoof, and David Meger. "Addressing Function Approximation Error in Actor-Critic Methods". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 1587–1596. URL: `https://proceedings.mlr.press/v80/fujimoto18a.html`.

[5] Tuomas Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *CoRR* abs/1801.01290 (2018). arXiv: `1801.01290`. URL: `http://arxiv.org/abs/1801.01290`.

[6] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. "Planning and acting in partially observable stochastic domains". In: *Artificial Intelligence* 101.1 (1998), pp. 99–134. ISSN: 0004-3702. DOI: `https://doi.org/10.1016/S0004-3702(98)00023-X`. URL: `https://www.sciencedirect.com/science/article/pii/S000437029800023X`.

[7] Arsenii Kuznetsov et al. "Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics". In: *CoRR* abs/2005.04269 (2020). arXiv: `2005.04269`. URL: `https://arxiv.org/abs/2005.04269`.

[8] Samsung Labs. *TQC implementation*. 2020. URL: `https://github.com/SamsungLabs/tqc_pytorch/tree/master/tqc` (visited on 07/14/2020).

[9] OpenAI. *Bipedal Walker*. 2021. URL: `https://www.gymlibrary.dev/environments/box2d/bipedal_walker/` (visited on 08/08/2021).

[10] OpenAI. *Bipedal Walker solving definition*. 2022. URL: `https://github.com/openai/gym/wiki/Leaderboard` (visited on 02/16/2023).

[11] Andrew Patterson et al. "A Generalized Projected Bellman Error for Off-policy Value Estimation in Reinforcement Learning". In: *CoRR* abs/2104.13844 (2021). arXiv: `2104.13844`. URL: `https://arxiv.org/abs/2104.13844`.

[12] Seita's Place. *Explaination of PER*. 2019. URL: `https://danieltakeshi.github.io/2019/07/14/per/` (visited on 07/14/2019).

[13] Baturay Saglam et al. *Actor Prioritized Experience Replay*. 2022. DOI: `10.48550/ARXIV.2209.00532`. URL: `https://arxiv.org/abs/2209.00532`.

[14] Tom Schaul et al. *Prioritized Experience Replay*. 2015. DOI: `10.48550/ARXIV.1511.05952`. URL: `https://arxiv.org/abs/1511.05952`.

[15] OpenAI Spinning up. *Bipedal Walker solving definition*. 2022. URL: `https://spinningup.openai.com/en/latest/algorithms/sac.html` (visited on 02/16/2023).

[16] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: `1706.03762`. URL: `http://arxiv.org/abs/1706.03762`.