Solving the BipedalWalker Environments

Anonymous author

Abstract

This paper surveyed four different approaches and distinguished Truncated Quantile Critic as the most advantageous. It proceeded to perform a parameter sweep to find the hyperparameters most suitable for the Bipedal Walker environment. It then augmented the neural architecture by incorporating findings from D2RL. The resultant agent is able to solve both the BipedalWalker and BipedalWalkerHardcore environments.

1 Methodology

After surveying literature, we finalized four algorithms that could potentially solve this environment: Twin Deep Deterministic Policy (TD3) [1], Soft Actor Critic (SAC) [2], Augmented Random Search (ARS) [3] and Truncated Quantile Critic (TQC) [4]. To identify the best candidate, agents were evaluated across 5 different seeds (using the hyperparameters suggested in the agent's paper). They were evaluated on sample efficiency, whereby if the average score across 100 evaluation episodes (where the agent was set to evaluation mode) was greater than 300 the environment was considered solved. TQC had the lowest average across seeds and solved the environment in the smallest number of episodes (110 episodes). Further trials were then commenced on the NCC, with a fine granularity, to conduct a parameter sweep as the first trials indicated that TQC was not entirely robust to the choice of parameters. This was done using the Python library Optuna [5]. Optuna is an automatic hyperparameter optimising framework that uses a Bayesian optimization algorithm called Tree-structured Parzen Estimator Agents to identify the most promising hyperparamters. The five most promising results of these trials are seen in table:

Gamma	Learning Rate	Batch Size	Buffer Size	Learning Starts	Tau	Number of Critics	Number of Atoms	Episodes	Timesteps
0.98	0.000302	64	1000000	100	0.02	5	25	110	95000
0.98	0.0003	128	1000000	1000	0.01	3	25	111	95000
0.98	0.000465	100	1000000	1000	0.02	5	20	125	100000
0.98	0.000273	64	1000000	1000	0.005	5	30	146	105000
0.98	0.000308	64	1000000	20000	0.005	5	25	149	145000

Table 1: The most promising hyperparameters. Learning Starts refer to how many steps the agent takes under a random policy at the beginning to help exploration. The final hyperparameters were taken as the mode (average for learning rate) of these trials.

1.1 Truncated Quantile Critic

TQC mitigates the overestimation bias often present in off-policy learning by utilising an ensemble of distributional critics and truncating this mixture.

The overestimation of the Q-function, which gives rise to the overestimation bias seen in off-policy learning, can be explained by employing Jensen's inequality [Thrun Schwartz (1993)]:

$$\mathbb{E}\ [\max\{Q(a) + U(a)\}] \geq \max \mathbb{E}[Q(a) + U(a)]\ = \max Q(a)$$

Overestimation causes the agent to pursue erroneous value estimates and this error propagates through time, degrading performance.

To address this, TQC, encouraged by the success of QR-DQN [6], takes a distributional approach and approximates the return random variable $Z^{\pi}(s, a)$ in contrast to approximating

the expectation of the return (the Q-function). This distributional approach is advantageous as it is more conducive to learning the intrinsic randomness (aleatoric uncertainty) of the environment and policy. Moreover, we can exploit the granularity of distributional representation to help control overestimation with greater precision.

In TQC, we approximate the random variable $Z^\pi(s,a) := \gamma^t R(s_t, a_t)$ by N distributions $Z_{\psi_n}(s,a)$ given by:

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

Where δ is a Dirac delta distribution with the support (where the distribution is non-zero) of the atoms $\theta^1_{\psi_n}(s,a)...\theta^m_{\psi_n}(s,a)$. Atoms are measurable sets that have a positive measure and no set of smaller positive measure is contained within it.

We train these approximators on the temporal difference target distribution, $Y(s,a)$. To construct this, atoms from the distributions $Z_{\psi_1}(s',a'),..,Z_{\psi_N}(s',a')$ are combined into the set $\mathcal{Z}(s',a')$ given by:

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

This set is then truncated at the right tail by removing atoms with the largest locations. A balance between under and overestimation can be achieved by varying the total number of atoms (N) and the number of atoms we drop using the parameter k. We use the kN smallest atoms from this set to define the atoms $y_i(s,a)$ in the target distribution:

$$y_i(s,a) := r(s,a) + \gamma\left[z_{(i)}(s',a') - \alpha \log \pi_\phi(a',s')\right]$$

Where $z_i(s',a')$ is the i^{th} smallest element of $\mathcal{Z}(s',a')$ and α is the entropy temperature coefficient. Finally, the target distribution is given by:

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

The Q-value can be estimated by the average location of the remaining atoms in this distribution. In practice, to increase the stability of training, atoms predicted by the target network distributions $(Z_{\overline{\psi_1}}(s,a)..Z_{\overline{\psi_N}}(s,a))$ are pooled instead.

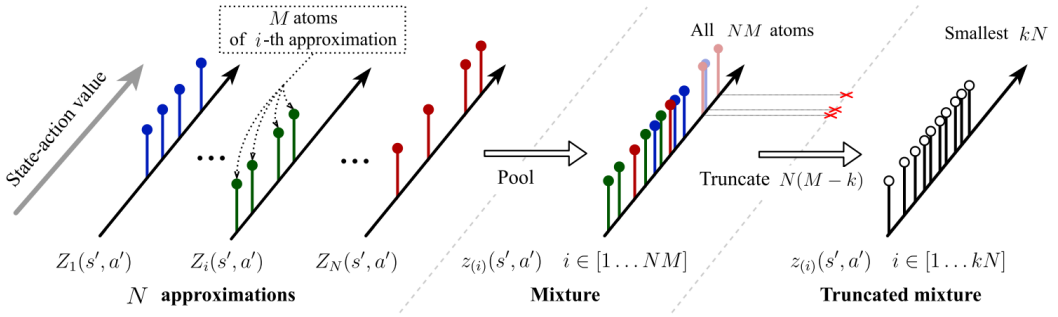


Figure 1: Diagram taken from [4] that helps elucidate the process of selecting atoms to populate the target distribution $Y(s,a)$. We evaluate N separate target critics, pool these together and then truncate the right tail.

For training, we minimise the 1–Wasserstein distance between the target distribution $Y(s,a)$ and each of the N $Z_{\psi_i}(s',a')$ distributions. . If we visualise a distribution as a pile of earth, the 1–Wasserstein distance can be understood intuitively as a the minimum cost of turning a one pile of earth into another where the cost is given by multiplying the amount of earth that needs to be moved by the mean distance is has to be moved. Formally it is given by:

$$W_P(\mu,\nu) = \inf\left(\mathbb{E}[d(X,Y)^p]\right)^{\frac{1}{p}}$$

Equivalent to minimising this distance, is learning the locations for the quantile fractions $\tau_m = \frac{2m-1}{2M}, m \in [1..M]$ in the target distribution. We approximate τ with $\theta^1_{\psi_n}(s,a)..\theta^M_{\psi_n}(s,a)$ by minimising the loss:

$$J_Z(\psi_n) = \mathbb{E}_{\mathcal{D},\pi}\left[\mathcal{L}^k(s_t, a_t; \psi_n)\right] \tag{1}$$

Where

$$\mathcal{L}^k(s,a;\psi_n) = \frac{1}{kNM}\sum_{m=1}^{M}\sum_{i=1}^{kN}\rho^{\mathcal{H}}_{\tau_m}(y_i(s,a) - \theta^m_{\psi_n}(s,a))$$

Here ρ is given by:

$$\rho^{\mathcal{H}}_{\tau}(u) = |\tau - I(u<0)|\mathcal{L}^1_H(u)$$

Where we use Huber quantile loss $\mathcal{L}^1_H(u)$ as it improves gradients for small u. The parameters of the neural network that parameterise the policy can be learnt by minimising the loss :

$$J_\pi(\phi) = \mathbb{E}_{\mathcal{D},\pi}\left[\alpha\log\pi_\phi(a|s) - \frac{1}{NM}\sum_{m,n=1}^{M,N}\theta^m_{\psi_n}(s,a)\right] \tag{2}$$

Similarly to [7], we adjust the entropy temperature coefficient α dynamically by also taking a gradient step with respect to the following loss:

$$J(\alpha) = \mathbb{E}_{\mathcal{D},\pi_\phi}\left[\log\alpha\cdot(-\log\pi_\phi(a_t|s_t) - \mathcal{H}_T)\right] \tag{3}$$

Where $\mathcal{H}_T$ is the target entropy. If the stochastic estimate of policy entropy, $(\log\pi_\phi(a_t|s_t))$ is higher than the target entropy $(\mathcal{H}_T)$, α will decrease, else if it is lower α will increase.

1.1.1 The Algorithm

The algorithm can be constructed as follows, firstly we initialise the policy and the critics $Z_{\psi_n}, Z_{\overline{\psi_n}}$. We set the replay buffer $\mathcal{D}$ to the empty set, set the target entropy $\mathcal{H}_T$ heuristically to the negative of the dimension of the action space and set $\alpha = 1, \beta = 0.005$. Next for each iteration we then collect transitions for a certain number of environment steps under a policy π_ϕ and add these to the replay buffer $\mathcal{D}$. Then, for each gradient step we sample a batch from $\mathcal{D}$ and update $\alpha, \phi, \psi_n, \overline{\psi_n}$ as follows:

$$\alpha \leftarrow \alpha - \lambda_\alpha\hat{\nabla}_\alpha J(\alpha)$$

$$\phi \leftarrow \phi - \lambda_\pi\hat{\nabla}_\phi J_\pi(\phi)$$

$$\psi_n \leftarrow \psi_n - \lambda_Z\hat{\nabla_{\psi_n}} J_Z(\psi_n)$$

$$\overline{\psi_n} \leftarrow \beta\psi_n + (1-\beta)\overline{\psi_n}$$

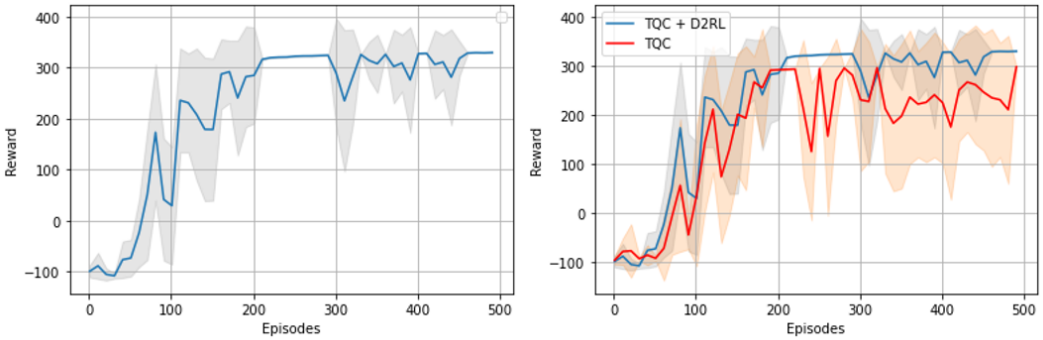
Where $\lambda_\alpha, \lambda_\pi, \lambda_Z$ are the learning rates, $\hat{\nabla}$ represents stochastic gradient descent and $J(\alpha), J_\pi(\phi), J_Z(\psi)$ are equations 3, 2, 1.

1.2 DR2L

To augment performance, a parameter sweep with different neural architectures was conducted. Increasing the depth of the neural architecture appeared to degrade performance. This can be expected because as the neural network becomes deeper, the mutual information between the input and final output likely decreases as a result of the non-linear transformations [8]. Moreover, we mirrored the findings of SAC-AE [9] as wider multi-layer perceptrons (MLP) appeared to be advantageous. This motivated the employment of the D2RL [8] architecture for the final architecture for the MLP which parameterised the critic and actor. D2RL incorporates dense connections and concatenates the state-action or state to each intermediate layer except the last layer. This overcomes the performance loss associated with parameterising the value and policy functions with deeper neural architecture and retains the benefit of wider architecture. We experimented with different layer numbers (2, 4, 8) and decided on 2.

2 Convergence Results

The agent solves both the BipedalWalker and BipedalWalkerHardcore environments. We have plotted the convergence results for the BipedalWalker. We present the augmentation in performance that D2RL adds. For these graphs, we plot the mean across 10 episodes, and the variance across 10 episodes is filled in.



The BipedalWalkerHardcore environment was solved in 651 episodes by inheriting the hyperparameters used in the BipedalWalker environment.

3 Limitations

TQC is a computationally heavy approach as a consequence of distributional representations and ensembling. We found that the introduction of D2RL architecture further augmented the computational expense of our approach. Each agent takes about a day to train.

Future Work

The agent takes a number of random steps at the start, however this exploration strategy could be improved upon by introducing intrinsic rewards [10]. These might guide the agent towards the optimal behaviour faster and help exploration at the beginning. However, intrinsic rewards are most applicable in tasks with sparse rewards so we're not sure how much additional benefit these would offer. Furthermore, an additional point of further work would be to look at the benefit of leveraging distributed training architectures with many agents in parallel on separate environment instances to hopefully mitigate the longer training times. Finally, we could tune the sampling technique used in the replay buffer to support Emphasizing Recent Experience [11] which shows a more competitive sample efficiency in comparison to Prioritised Experience Replay [12] when applied to SAC.

To make the BipedalWalkerHardcore agent more stable, a hyperparameter sweep on the environment could be conducted. We could also experiment with training on the BipedalWalker environment and then the BipedalWalkerHardcore environment.

References

[1] Scott Fujimoto, Herke Hoof, and David Meger. "Addressing function approximation error in actor-critic methods". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1587–1596.

[2] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.

[3] Horia Mania, Aurelia Guy, and Benjamin Recht. "Simple random search provides a competitive approach to reinforcement learning". In: *arXiv preprint arXiv:1803.07055* (2018).

[4] Arsenii Kuznetsov et al. “Controlling overestimation bias with truncated mixture of continuous distributional quantile critics”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5556–5566.

[5] Takuya Akiba et al. “Optuna: A next-generation hyperparameter optimization framework”. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 2623–2631.

[6] Will Dabney et al. “Distributional reinforcement learning with quantile regression”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[7] Tuomas Haarnoja et al. “Soft actor-critic algorithms and applications”. In: *arXiv preprint arXiv:1812.05905* (2018).

[8] Samarth Sinha et al. “D2rl: Deep dense architectures in reinforcement learning”. In: *arXiv preprint arXiv:2010.09163* (2020).

[9] Denis Yarats et al. “Improving sample efficiency in model-free reinforcement learning from images”. In: *arXiv preprint arXiv:1910.01741* (2019).

[10] Deepak Pathak et al. “Curiosity-driven exploration by self-supervised prediction”. In: *International conference on machine learning*. PMLR. 2017, pp. 2778–2787.

[11] Che Wang and Keith Ross. “Boosting soft actor-critic: Emphasizing recent experience without forgetting the past”. In: *arXiv preprint arXiv:1906.04009* (2019).

[12] Tom Schaul et al. “Prioritized experience replay”. In: *arXiv preprint arXiv:1511.05952* (2015).