Learning to Walk Randomly

Anonymous author

Abstract

This paper proposes using a TD3 algorithm to learn to walk. This is a simple actor-critic policy alongside a replay buffer to provide the algorithm with plenty of opportunities to learn. We evaluate it on OpenAI's Gym in the BipedalWalker environment, then evaluate its effectiveness on a 'hardcore' version of the environment, which sets many obstacles in the environment that the algorithm should learn to navigate.

1 Methodology

This paper proposes training a bipedal robot to learn to walk using a policy by Fujimoto et al. [2] in their 2018 paper, known as the Twin Delayed Deep Deterministic policy gradient algorithm. This algorithm is unique in that, as opposed to one actor and one critic, TD3 uses two critics. In order to reduce overestimation bias, the algorithm uses a clipped version of Double-Q learning as proposed by Hasselt in their 2010 paper [4], using the original formulation with one actor ($\pi_{\phi 1}$) and a pair of critics ($Q_{\theta 1}, Q_{\theta 2}$), reducing computational costs as the critics can be compared to check whether overestimation is occurring.

TD3 also builds on top of the Deep Deterministic Policy Gradient algorithm, proposed by Lillicrap et al. in their 2015 paper [5], applying various optimisation such as target policy smoothing regularisation, delayed policy updates and the aforementioned clipped Double-Q learning for improved stability and performance. The critic pair is updated towards the minimum target value of the actions selected by the target policy:

$$y = r + \gamma \underset{i=1,2}{min} Q_{\theta_i'}(s', \pi_{\phi'}(s') + \epsilon)$$

$$\epsilon \sim clip(\mathcal{N}(0, \sigma), -c, c)$$

The algorithm also features a replay buffer, which is used to learn from, as each action-reward pair is stored to inform the algorithm with regards to which actions are the most beneficial.

We originally attempted to use TD3-FORK from Wei and Ying's 2021 paper [6]. While the paper originally stated it could solve BipedalWalker-v3 quickly, claiming to train in as little as 4 hours on a single GPU, after performing our own experiments, we found out it converged slower than the original TD3 algorithm at around 450 steps (see Figure 1), as well as running considerably slower. After running it for 12 hours on the university supercomputer, TD3-FORK only managed to run 800 episodes, whereas the "base" TD3 algorithm would be able to finish the full 1000 episodes in considerably less time. These results are shown in Figure 1, below.

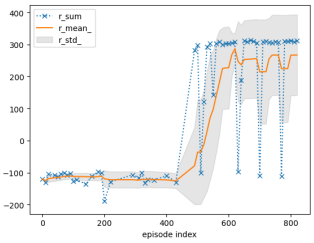

Figure 1: TD3-FORK algorithm on BipedalWalker-v3 after 12 hours

2 Convergence Results

The model does converge after around 350 episodes, after doing a lot of exploration earlier on within the training loop in the standard environment, quickly learning to run, opting for a forward balancing method for running through the environment. From 400 and onwards, the algorithm can quickly climb to 300 consistently, but does have some runs in which the initial randomness of the algorithm occasionally reduces its score, shown in Figure 2.

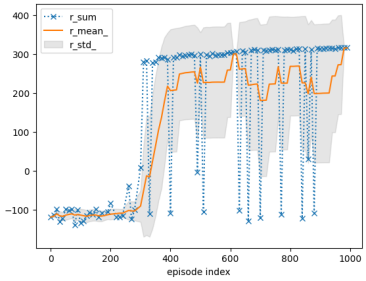

Figure 2: TD3 results on BipedalWalker-v3

Within the hardcore environment, due to the slow startup, the algorithm takes considerably longer to learn how to navigate each obstacle as, at first, the algorithm only learns how to get as close to the obstacle as possible without passing it in order to maximise its score without minimising its risk. This meant that the policy noise needed to be increased while the discount rate was decreased in order to prioritise exploitation over exploration within the more difficult to navigate environment. Even with this increase in noise and various hyperparameter changes, the model was still slow to learn, often getting stuck in local maxima and not being able to navigate in certain environment instances. After 2400 steps of running, we found very few "good" runs that illustrated its talent at navigating unknown areas, as shown in Figure 3.

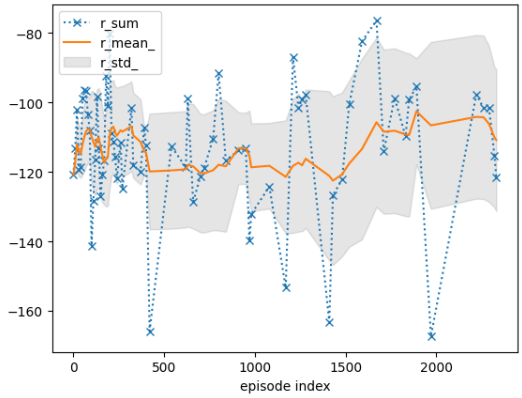

Figure 3: TD3 results on BipedalWalker-v3 Hardcore

The log from this 2400 episode run was unfortunately corrupted, so we have submitted a 2000-episode log that shows a similar trend, where the model seems to converge to a local maxima, getting as close as possible to a given obstacle without navigating over it.

3 Limitations

The model itself is not very limited as it can learn fast, but does spend a lot of time at the start exploring the possibilities within the environment, slowing down the learning process significantly.

Due to the random nature of the algorithm, it would struggle to learn in the hardcore environment as it would change every time, meaning the algorithm had to adapt to many different scenarios, so the algorithm would need to run for considerably longer than for the easy environment. Since the code ran slowly over 3000 steps and would not complete within 24 hours, we ran it for 2400 steps, then dialled back to 2000 steps to ensure it could finish running within the maximum of 24 hours allotted to us on the supercomputer, meaning it had less time to learn.

Future Work

Due to the lack of convergence within the hardcore environment, a different approach to the actor-critic off-policy method may be useful. As covered by Aydogmus and Yilmaz in their 2023 paper [1], various algorithms are good at navigating the environment, such as a soft actor-critic (SAC) method, which adopts the actor-critic methodolgy while using a neural network to select actions in the space. As first proposed by Haarnoja et al. in 2018 [3], by combining off-policy updates with a stable stochastic actor-critic formulation, alongside a strong emphasis on entropy regularisation, SAC can learn in environments with continuous action spaces, such as BipedalWalker and MuJoCo, to a great degree of success. Aydogmus and Yilmaz found SAC to reach an average reward of 320.3 in the BipedalWalker-v3 environment in their research, which would make it a brilliant option for furthering this work.

References

[1] Omur Aydogmus and Musa Yilmaz. "Comparative analysis of reinforcement learning algorithms for bipedal robot locomotion". In: *IEEE Access* 12 (2023), pp. 7490–7499.

[2] Scott Fujimoto, Herke van Hoof, and David Meger. *Addressing Function Approximation Error in Actor-Critic Methods*. 2018. arXiv: `1802.09477 [cs.AI]`. URL: `https://arxiv.org/abs/1802.09477`.

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

[4] Hado Hasselt. "Double Q-learning". In: *Advances in neural information processing systems* 23 (2010).

[5] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[6] Honghao Wei and Lei Ying. "Fork: A forward-looking actor for model-free reinforcement learning". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE. 2021, pp. 1554–1559.