Learning to Walk with Foresight

Yosef Berezovskiy

Abstract

This paper investigates the effectiveness of Twin Delayed Deep Deterministic Policy Gradient with Forward-looking mechanism (TD3-FORK) to control bipedal locomotion in the BipedalWalker-v3 environments. TD3-FORK extends the TD3 algorithm by incorporating predictions of future states and rewards, enabling more informed decision-making. The performance of two TD3-FORK variants is analysed: TD3-FORK-S, employing single-step future state and reward predictions, and TD3-FORK-DQ, using a two-step future state prediction. Results show that TD3-FORK-S achieves stable convergence and solves the standard environment, while TD3-FORK-DQ solves the harder version of the environment.

1 Methodology

Two versions of the BipedalWalker-v3 environment exist. The standard version presents a scenario with slightly uneven terrain, whereas the hardcore version includes complex obstacles (such as ladders, stumps, and pitfalls). In both variants, the action space consists of motor speed values ranging from -1 to 1 for each of the four joints at the hips and knees to control the robots locomotion. The observation space includes sensory inputs; hull angle speed, angular velocity, horizontal and vertical speeds, the position and angular speed of joints, leg contact with the ground, and measurements from 10 lidar rangefinders. To successfully complete the standard version, an algorithm must achieve a cumulative score exceeding 300 reward points across 1600 time steps. The hardcore version requires achieving the same score within 2000 time steps [1]. Solving the environment is quantitatively defined as accumulating over 300 reward points for 100 consecutive episodes; to demonstrate consistent robustness [2].

Physical control systems analogous to BipedalWalker-v3, involve continuous action spaces. The Deep Deterministic Policy Gradient (DDPG) algorithm adapts the Q-learning technique, conventionally applied only to discrete action space problems, to continuous domain. DDPG employs deep neural networks (NNs) to represent both the actor (policy) and critic (value function) and employs a deterministic policy for action selection with some added noise sampled from either Gaussian or Uhlenbeck-Ornstein (OU) processes to promote better exploration [4].

However, DDPG suffers from several limitations. Overestimation of Q-values by the critic network, leads to the generation of suboptimal policies and unstable learning dynamics. Another issue is overfitting of the value function; since DDPG employs deterministic policies, these can overfit the value estimate, resulting in a significant function approximation error, affecting the critic's update process where the learning target is highly susceptible to inaccuracies [3].

The Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm improves upon DDPG. TD3 employs Clipped Double Q-learning by using two critic networks instead of one; during updates, the target value is calculated as the minimum of the two Q-values predicted by the target critic networks, which partially mitigates the overestimation bias. To improve exploration and to help overcome overfitting, TD3 employs Target Policy Smoothing by adding Gaussian noise to the target action during the training of the critic, smoothing the value function. Lastly, TD3 employs Delayed Policy Updates, updating the policy and target networks less frequently than the critic networks, ensuring that the value estimates from the critic are more accurate before they are used to update the policy, thus promoting more stable learning outcomes [3]. The policy loss function for TD3 is shown below:

$$L(\phi) = -\mathbb{E}\left[Q_\psi(s_t, A_\phi(s_t))\right] \tag{1}$$

where:

- $L(\phi)$ represents the loss of the actor network with parameters ϕ.
- $Q_\psi(s_t, A_\phi(s_t)))$ is the output of one of the critic networks, estimating the value of taking action $A_\phi(s_t)$ in state s_t. ψ are the parameters of this critic network.
- $A_\phi(s_t)$ is the action proposed by the actor network given state s_t.

While TD3 offers significant improvements over DDPG it makes its decisions solely based on the current state and action. TD3-FORK, a training algorithm that introduces a forward-looking (FORK) mechanism by incorporating predictions of future states and future rewards. This approach is motivated by the observation that considering potential future outcomes can lead to more informed and efficient decision-making [6].

TD3-FORK extends the TD3 architecture with a system network F_θ, responsible for predicting the next state $\tilde{s}_{t+1} = F_\theta(s_t, a_t)$ given the current state s_t and action a_t. As well as with a reward network, R_η, which predicts the reward given the current state and action s.t. $\tilde{r}_t = R_\eta(s_t, a_t)$. These networks allow the agent to anticipate the consequences of its actions and factor them into the policy update [6].

Multiple variants of the TD-FORK were proposed in [6] and this paper focuses on TD3-FORK-S and TD3-FORK-DQ variants of the algoritm.

TD3-FORK-S has shown itself more sample efficient and generally best performing variant solving the normal environment of the BidepdalWalker-v3, as per the proposed results in the literature [6]. TD3-FORK-S employs a single-step forward-looking mechanism, modifying the policy loss function to incorporate both $\tilde{s}_{t+1} = F_\theta(s_t, a_t)$ and $\tilde{r}_t = R_\eta(s_t, a_t)$:

$$L(\phi) = \mathbb{E}\left[-Q_\psi(s_t, A_\phi(s_t)) - wR_n(s_t, A_\phi(s_t)) - w\gamma Q_\psi(\tilde{s}_{t+1}, A_\phi(\tilde{s}_{t+1}))\right], \tag{2}$$

where:

- $Q_\psi(s_i, A_\phi(s_i))$ is the output of one of the critic networks at state s_i for action $A_\phi(s_i)$ calculated with parameters ψ for t $\leq i \leq t+1$.
- w and γ are weight factors for the reward and the discount factor for the future Q-value, respectively.
- $A_\phi(\tilde{s}_{i+1})$ is the action chosen by the actor network at the next state $\tilde{s}_{i+1}$ for $t \leq i \leq t+1$.

The literature [6] recommends TD3-FORK-DQ for solving BipedalWalkerHardcore but lacks justification for this choice. This paper argues that while incorporating predictions of future rewards seems intuitive, accurately forecasting rewards in complex, sparse-reward environments like BipedalWalkerHardcore is highly challenging. Therefore, a variant of TD3-FORK that predicts only future states is typically used. Additionally, the multi-step dynamics of BipedalWalkerHardcore necessitate a model that can anticipate the delayed effects of actions. Hence, TD3-FORK-DQ, which incorporates a two-step forward-looking mechanism, is selected to better manage these dynamics. The policy loss function for TD3-FORK-DQ hence incorporates $\tilde{s}_{t+1} = F_\theta(s_t, a_t)$ and $\tilde{s}_{t+2} = F_\theta(\tilde{s}_{t+1}, A_\phi(\tilde{s}_{t+1}))$:

$$L(\phi) = \mathbb{E}\left[-Q_\psi(s_t, A_\phi(s_t)) - w\left(Q_\psi(\tilde{s}_{t+1}, A_\phi(\tilde{s}_{t+1})) - w'Q_\psi(\tilde{s}_{t+2}, A_\phi(\tilde{s}_{t+2}))\right)\right] \tag{3}$$

where w' is a weight factor multiplied by a constant, e.g. 0.5 as in the current implementation.

Additionally, although forecasting proposed above enhances the performance of the algorithms during training, the benefits of such method diminish due to the inaccuracies of predictions made by system and reward networks, introducing unnecessary variance in training as the model begins to approach its objective - hence why, as proposed by [6], adaptive weighting is used to leverage benefits of FORK at the beginning of training process and to reduce the weight of predictions as model converges, s.t weight $w = (\overline{r}/r_0)_0^1 w_0$, where $\overline{r}$ is the moving average of accumulated reward per episode, and r_0 is a predefined goal (reward of 300 in the context of BipedalWalker-v3), and w_0 is the initial weight, $0 \leq w \leq 1$.

To ensure consistency with established research, the hyperparameters used in this work are adopted directly from the original TD3-FORK paper [6]. This allowed to focus more on the analysis of the algorithm's behaviour leveraging the findings of the original work, isolating the effects of specific modifications or enviromental changes.

Furthermore, to address complexities of the difficult version of the environment, additional modifications have been implemented, or experimented with. In the original BipedalWalkerHardcore environment, a reward of -100 is given when the walker falls down, which heavily penalises failures. This potentially discourages exploration by the agent - reducing penalty to -5 as in [6] addresses this problem. Furthermore, [6] suggests to increase the importance of positive actions relative to the penalty for failing to encourage the agent to focus more on maximising the reward received rather than just avoiding the penalty - hence why, other rewards a scaled by a factor of 5. Other modification as in [6] involved balancing the training data stored in the replay buffer by adding failed episodes more frequently than successful episodes to ensure agent does not overfit the successful trajectories and learns to recover from failures - hence replay buffer is implemented with a 5:1 ratio for failed and successful episodes.

Modification inspired by [5] involved skipping frames to increase sample efficiency, however as later elaborated, it conflicted with strengths of implementing FORK agent.

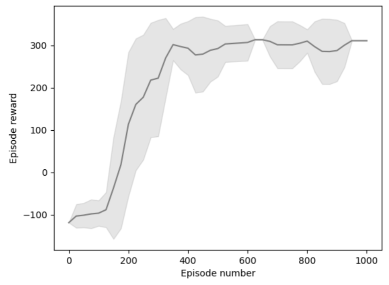
2 Convergence Results

Multiple experiments have been conducted to achieve better, faster, convergence, concerning processes to sample noise from for action selection, their parameters, and training time frequency.

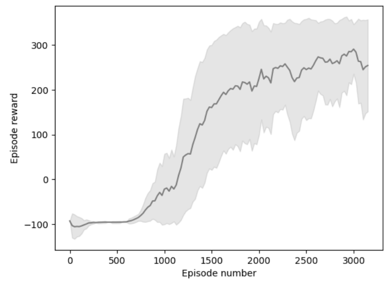
For BipedalWalker-v3 the best convergence has been shown by TD3-FORK-S (Figure 1 (a)) applying samples of OU and clipped Gaussian processes to agent's action selection and target actions, respectively. OU noise is a type of temporally correlated noise, meaning its current value is influenced by its past values. It may have improved the convergence rate as temporal correlation introduces a "momentum" effect, making the noise smoother and more conducive to exploring actions that require sustained effort in a particular direction. The environment has relatively smooth terrain, allowing the walker to maintain momentum and make gradual adjustments to its locomotion. Initially, the agent experiences negative rewards but rapidly improves, indicating effective learning. By around 200 episodes, the rewards stabilise at a higher level, with reduced variance, suggesting the development of a reasonably effective policy. The convergence is stable with sharp positive gradient until agent reaches close to optimal results. Past 300 episodes, the reward levels fluctuate but generally maintain a high average. Algorithm learns consistently, without getting stuck at local stationary points; solves the environment per aforementioned definition.

For BipedalWalkerHardcore-v3 the best convergence has been shown by TD3-FORK-DQ (Figure 1 (b)) using Gaussian process to sample noise for both, agent's action selection and target actions. The hardcore version requires quick changes in actions and momentum. The temporal correlation of OU noise has shown to hinder exploration in this environment. Initially, the agent struggles, completing the environment for the first time in as much as 900 episodes, but gradually adapts, showing steady improvement in performance up to 1500 episodes, starting to achieve rewards above 300 points with high consistency. The phase between 1500 to 2500 episodes is marked by significant fluctuations, suggesting ongoing exploration and strategy refinement in response to the challenging terrain. From 2500 to 3000 episodes, rewards continue to fluctuate but stabilize close to higher levels, indicating near-convergence to a robust policy. Algorithm learns consistently, without getting stuck at local stationary points and the model finds optimal policy and solves the environment per aforementioned definition.

Frame skipping was attempted to increase algorithm's sample efficiency [5] for both BipedalWalker-v3 and BipedalWalkerHardcore-v3, however altering the environment dynamics disrupts the predictive capability of reward and system networks, hindering the convergence.



(a) Reward trajectory for the BipedalWalker-v3 normal environment



(b) Reward trajectory for the BipedalWalkerHardcore-v3 environment

Figure 1: Reward trajectories of agents in the BipedalWalker-v3 normal and hardcore

3 Limitations

A key limitation of TD3-FORK lies in its sample efficiency. While the forward-looking mechanism can lead to faster policy learning, it introduces additional training overhead. Both the system and reward networks require substantial data and training time to achieve accurate predictions. However, as shown by the performance of the algorithm in the BipedalWalkerHardcore-v3 environment, lower sample efficiency is fairly compensated by stable convergence to optimal policy in complex environments.

Furthermore inaccurate predictions of future reward and state networks can have a cascading effect. The agent might take actions based on faulty predictions, leading to states that were not accurately modeled during training. This was observed to hinder the learning process.

As experiments with different backbones have been performed inspired by [5], in particular using an LSTM to encode state information including past state information. The aim was to provide more data to train reward and system networks to accelerate their learning process, however due to the complexity of the system the algorithm overfits. Hence, despite TD3-FORK being an effective method, its complexity could limit its extension to more complicated tasks, or to tasks which require additional components, like an LSTM.

Future Work

Future work shall focus on mitigating the limitations of TD3-FORK, particularly its sample efficiency and dependence on accurate predictions, while exploring its potential for further enhancements. To address the training overhead associated with the system and reward networks, curriculum learning can be implemented. This involves gradually increasing the complexity of the environments, starting with simplified scenarios to facilitate more accurate initial predictions and progressively introducing more challenging situations as the networks learn. This depends on whether the difficulty of the environment can be gradually adjusted. In conjunction with curriculum learning, data augmentation techniques could be explored to enhance the diversity and richness of the training data, leading to improved prediction accuracy and generalization capabilities. Ensemble methods, where multiple system and reward networks are trained independently and their predictions are combined, can be explored to reduce the impact of individual network biases and improve overall prediction accuracy. Furthermore, alternative exploration strategies, such as parameter space noise, can introduce more abrupt and diverse changes in actions.

References

[1] *Bipedal Walker - Gym Documentation.* `https://www.gymlibrary.dev/environments/box2d/bipedal_walker/`. Accessed: May 13, 2024.

[2] J. Dibachi and J. Azoulay. "Teaching a Robot to Walk Using Reinforcement Learning". In: *arXiv* (Dec. 2021). Accessed: May 13, 2024. eprint: `2112.07031`. URL: `http://arxiv.org/abs/2112.07031`.

[3] S. Fujimoto, H. van Hoof, and D. Meger. "Addressing Function Approximation Error in Actor-Critic Methods". In: *arXiv* (Oct. 2018). DOI: `10.48550/arXiv.1802.09477`. eprint: `1802.09477`.

[4] T. P. Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv* (July 2019). Accessed: May 10, 2024. eprint: `1509.02971`. URL: `http://arxiv.org/abs/1509.02971`.

[5] U. Özalp. "Bipedal Robot Walking by Reinforcement Learning in Partially Observed Environment". Accessed: May 01, 2024. MA thesis. Middle East Technical University, 2021. URL: `https://open.metu.edu.tr/handle/11511/92170`.

[6] H. Wei and L. Ying. "FORK: A Forward-Looking Actor For Model-Free Reinforcement Learning". In: *arXiv* (Sept. 2021). Accessed: May 06, 2024. eprint: `2010.01652`. URL: `http://arxiv.org/abs/2010.01652`.