Fener Ahmed]

Learning to Walk Using TATD3

Anonymous author

Abstract

This paper proposes to learn to walk using the TATD3 algorithm, which is based on the TD3 algorithm, on the Bipedal Walker V3 and Bipedal Walker V3 Hardcore environment.

1 Methodology

To tackle the complexities of efficiently controlling batch processes which experience varying dynamics and conditions, we have developed the Twin Actor Twin Delayed Deep Deterministic Policy Gradient (TATD3). This method enhances the existing Twin Delayed Deep Deterministic Policy Gradient (TD3) by integrating a dual actor network setup. TATD3 aims to improve the exploration and exploitation dynamics within continuous control tasks, encouraging robust policy development and quicker convergence [3].

1.1 Actor-Critic Framework

TATD3 employs an actor-critic framework, wherein two separate networks called actors and critics work together to determine the optimal policy. Actors are tasked with choosing actions according to the current policy, while critics evaluate these actions by estimating value functions. The dual actor configuration in TATD3 promotes a varied exploration of action space, reducing the likelihood of premature convergence to suboptimal policies.

1.2 Functions

The policy function, denoted by π(s, θ), maps states to actions, optimising the policy parameters θ through gradient ascent methods. The value function, or Q-function, Q(s,a,ω), with ω being the critic parameters, assesses the action quality taken by the actor. TATD3 uses twin critics to address the overestimation bias often seen in single-critic setups. The actor's policy is updated using the lower Q-value from these critics, which helps to stabilise the learning updates [2].

1.3 Math Formulation

The twin actors in TATD3 operate under a deterministic policy gradient framework. The objective function for policy optimization is given by:

$$J(\theta) = \mathbb{E}_{s\sim\rho^\beta}\left[Q^\pi(s, \pi(s|\theta))\right] \tag{1}$$

where ρ^β represents the state distribution under a behavior policy β. The gradient of J with respect to the actor parameters θ is estimated by:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\rho^\beta}\left[\nabla_\theta \pi(s|\theta) \nabla_a Q^\pi(s, a)|_{a=\pi(s|\theta)}\right] \tag{2}$$

This gradient dictates how the policy should be updated to maximize expected returns. For the critic, the temporal difference error used to update the Q-function is:

$$\delta = r + \gamma Q(s', \pi(s'|\theta')) - Q(s, a) \tag{3}$$

where s' is the next state, r is the reward, and γ is the discount factor. The critic parameters ω are updated by minimizing the squared temporal difference error, reinforcing the policy's effectiveness.

1.4 Reward Function

We've modified reward functions to make the learning process more suited to batch processes, encouraging efficient progression through operational phases. These adjustments aim to increase the controller's adaptability to dynamic changes, which is essential for batch process applications. The reward structures emphasise both immediate operational efficiency and long-term process stability.

Since TATD3 allows for more exploratory behaviour, having a smaller learning rate allowed for the model to converge faster, compared to having a larger learning rate which would cause it to be unstable.

1.5 Reward Buffer

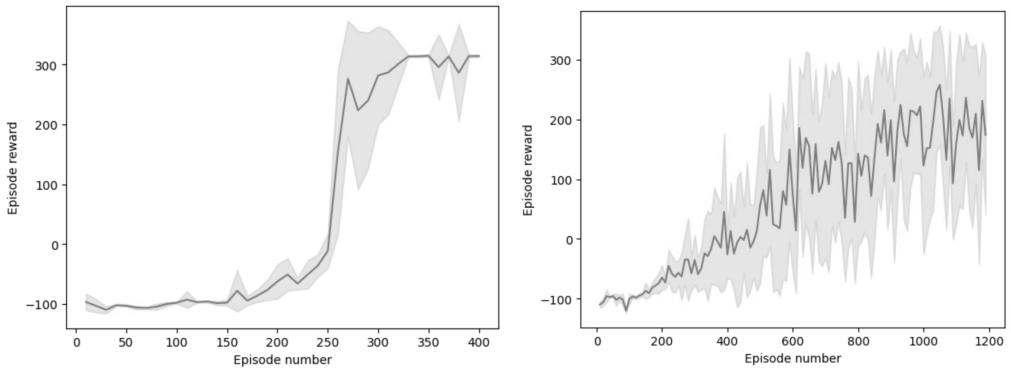
The replay buffer, implemented in our TATD3 algorithm, is a crucial component designed to store and manage the experiences gathered by the agent during its interaction with the environment. This implementation is used from the replay buffer used in Deep Q-Learning networks (DQN) as outlined by the OpenAI Baselines repository [1]. The primary purpose of the replay buffer is to decouple the experience gathering from the learning process, thereby improving the stability and efficiency of the learning algorithm.

In reinforcement learning, the agent interacts with the environment by taking actions based on its current policy, receiving rewards, and transitioning to new states. These interactions are termed as experiences and are represented as tuples of (state, action, reward, next state, done). Storing these experiences in a replay buffer allows the agent to break the temporal correlations between consecutive experiences, which is essential for stable and efficient learning.

The replay buffer enables the agent to:

- Sample Experiences Randomly: By sampling random mini-batches of experiences from the replay buffer, the learning process can be more stable and less correlated, mitigating the risk of divergence in the learning process.
- Re-use Past Experiences: Experiences stored in the buffer can be re-used multiple times, enhancing the data efficiency of the learning algorithm. This is particularly important in environments where gathering new experiences is costly or time-consuming.
- Smooth the Learning Process: The random sampling process from the buffer helps in smoothing out the learning updates, leading to more stable convergence of the policy and value function approximations.

2 Convergence Results



After approximately 300 episodes, the TATD3 algorithm achieves an average score of around 300 on the normal BipedalWalker-v3 environment. This consistent performance indicates

that the agent has effectively learned the task of walking efficiently. Achieving such a high average score in a relatively short number of episodes is significant. It reflects the efficiency and effectiveness of the TATD3 algorithm in training the bipedal robot to navigate the environment successfully [4].

The reason the agent occasionally scores below 300, yet maintains an average around this mark, can be attributed to the inherent variability in the environment and the exploratory nature of the reinforcement learning process. In reinforcement learning, agents continuously explore and exploit their learned policies. This balance ensures they do not get stuck in suboptimal behaviours. While exploration can lead to lower scores in some episodes, it ultimately helps the agent refine its policy and achieve better long-term performance.

This performance is considered very good because it demonstrates both rapid convergence and high final performance. Rapid convergence means the agent quickly learns effective policies, reducing the computational resources and time needed for training. High final performance indicates the agent can handle the task well, achieving scores close to the maximum possible.

However, there is still room for improvement. One potential enhancement could involve fine-tuning the hyperparameters, such as the learning rate or the noise added during exploration. Adjusting these parameters could help the agent achieve even more stable and higher scores. Another area for improvement could be the reward function. By making the reward function more adaptive or incorporating additional factors that reflect the complexities of the walking task, the agent might learn more nuanced policies that yield better performance.

The figure on the right, above shows the learning progress of the TATD3 agent in the BipedalWalkerHardcore-v3 environment over 1200 episodes. This environment is much more difficult than the standard BipedalWalker-v3, with more obstacles and rougher terrain. Despite these challenges, the TATD3 agent shows a clear upward trend in performance.

At first, the agent finds it hard to get positive rewards, which reflects the difficulty of the environment. The rewards vary a lot, showing the agent's ongoing exploration and attempts to find an effective walking strategy. However, after about 300 episodes, the agent's performance begins to improve noticeably. The average episode reward starts to stabilise and gradually increases, showing the agent's learning progress.

By around 800 episodes, the agent consistently achieves positive rewards, with the average reward continuing to rise. This improvement indicates that the agent has learned a more effective policy for navigating the hardcore environment. The ongoing fluctuations in reward values highlight the variability and unpredictability of the terrain, but the overall trend is positive.

The convergence of the agent's performance in this challenging environment is significant. It demonstrates the robustness of the TATD3 algorithm, particularly its ability to handle complex control tasks with difficult dynamics. The dual actor framework helps the agent explore a wider range of actions, which is crucial for mastering such a challenging task.

3 Limitations

In the normal environment, it reaches 300 score at around 300 episodes, but after that it still fluctuates and is still unstable sometimes going below the 300 score.

While the TATD3 agent has shown impressive results, there are several potential ways to improve further:

- Enhanced Exploration Strategies: Introducing better exploration strategies, such as changing exploration rates over time or using curiosity-driven exploration, could help the agent find better policies more quickly. These techniques could be particularly helpful in the diverse terrain of BipedalWalkerHardcore-v3.
- Refined Reward Function: Improving the reward function to better capture the details of efficient walking and obstacle navigation could lead to faster learning

and higher final performance. Adding factors like energy efficiency or smoother movement to the reward function might encourage more desirable behaviours.

- Dynamic Hyperparameter Tuning: Using dynamic adjustment of hyperparameters, such as learning rates and noise levels, could improve the training process. Adaptive learning rates could help the agent converge more quickly, while adjusting exploration noise could balance the trade-off between exploration and exploitation more effectively.

Future Work

In the future, the hyper parameters need to be tuned more, starting with the batch size, which could significantly impact the score. Possibly implement an even better state of the art model which could bring more stability to the score and also reach score of 300 faster and allow for it to be stable.

References

[1] *baselines/baselines/deepq/replay_buffer.py at master · openai/baselines*. en. URL: https://github.com/openai/baselines/blob/master/baselines/deepq/replay_buffer.py (visited on 05/14/2024).

[2] Scott Fujimoto, Herke van Hoof, and David Meger. *Addressing Function Approximation Error in Actor-Critic Methods*. arXiv:1802.09477 [cs, stat]. Oct. 2018. DOI: 10.48550/arXiv.1802.09477. URL: http://arxiv.org/abs/1802.09477 (visited on 05/14/2024).

[3] Tanuja Joshi et al. "Twin actor twin delayed deep deterministic policy gradient (TATD3) learning for batch process control". In: *Computers & Chemical Engineering* 155 (Dec. 2021). arXiv:2102.13012 [cs, eess], p. 107527. ISSN: 00981354. DOI: 10.1016/j.compchemeng.2021.107527. URL: http://arxiv.org/abs/2102.13012 (visited on 05/14/2024).

[4] Timothy P. Lillicrap et al. *Continuous control with deep reinforcement learning*. arXiv:1509.02971 [cs, stat]. July 2019. DOI: 10.48550/arXiv.1509.02971. URL: http://arxiv.org/abs/1509.02971 (visited on 05/14/2024).