Learning to Walk using Adaptively Calibrated Critics

nfqn37

Abstract

This paper demonstrates the application of ACC, an actor-critic method designed to dynamically adjust bias in TD targets to mitigate overestimation bias issues. The ACC approach was deployed to address the challenges of the BipedalWalker-v3 environment in both its easy and hardcore modes. The method successfully resolved the easy mode within approximately 310 episodes and achieved a high score of 320 in hardcore mode.

1 Environment Description

We aim to solve the BipedalWalker-v3 environment which includes both basic and hardcore versions, the latter featuring obstacles. The action space is continuous and consists of motor speed values ranging from -1 to 1 for each of the four joints at the hips and knees. The observation space encompasses hull angle speed, angular velocity, horizontal and vertical speeds, joint positions and their speeds, leg-ground contacts, and 10 LIDAR rangefinder measurements, without including coordinates. Rewards are earned for moving forward, with over 300 points possible by reaching the end, while falling incurs a penalty of -100 points. Motor torque application slightly reduces the score. Successful completion requires reaching the end within 1600 steps (2000 for hardcore), with both environments featuring dynamically varying terrains to challenge the agent's ability to generalize across settings [3].

2 Methodology

In continuous action spaces, exact maximization over actions is infeasible, prompting reliance on actor-critic architectures where the actor is trained to maximize the Q-value. While these methods effectively learn policies to maximize the Q-function, they are prone to overestimation bias, particularly in off-policy settings [2].

To combat overestimation bias, we explored architectures such as TD3 [1] and SAC [4], but superior outcomes were achieved using Truncated Quantile Critics (TQC). TQC employs an ensemble of distributional critics and enhances bias adjustment precision by selectively truncating targets from pooled estimates [5]. This selective truncation facilitates more nuanced control between over- and underestimation, significantly improving the robustness of the policy learning process. However, the parameter d, which determines the number of targets to drop, must be individually configured for each environment. Adaptively Calibrated Critics (ACC) enhance the TQC framework by introducing a dynamic mechanism to adjust the quantile targets exclusion parameter d, enabling more precise bias regulation [2]. Empirical evaluations were conducted on both models using standard hyperparameters in the Bipedal Walker environment. ACC results in superior sample efficiency compared to TQC, leading to its selection for further implementation. Subsequent investigations focused on the impact of key hyperparameters on performance optimization. Notably, an increase in the frequency of critic updates significantly enhanced sample efficiency.

2.1 TQC Algorithm Description

The TQC algorithm learns a distribution over future augmented rewards, diverging from traditional approaches that estimate a point-based Q-function It utilises distributional representation of a critic and truncation of critics prediction to allow for arbitrary granular

overestimation control and an ensemble of multiple critics for further performance improvements.

An ensemble of N networks $\{\theta_1, \ldots, \theta_N\}$ each predicts a distribution $Z_{\theta_n}(s_t, a_t)$:

$$Z_\theta(s_t, a_t) = \frac{1}{M} \sum_{m=1}^{M} \delta(\theta_m(s_t, a_t)) \tag{1}$$

where the Dirac deltas are positioned at quantile locations defined by:

$$\tau_m = \frac{2m-1}{2M}, \quad m \in \{1, \ldots, M\}$$

The network is trained to identify these quantile locations $\theta_m(s, a)$ by regressing the predictions $\theta_m(s_t, a_t)$ onto Bellman targets computed as:

$$y_m(s_t, a_t) = r_t + \gamma(\theta_m(s_{t+1}, a_{t+1}) - \alpha \log \pi_\phi(a_{t+1}|s_{t+1}))$$

A unified Bellman target distribution is computed by pooling targets from all networks, sorting them, and selecting the kN smallest targets to define the target distribution:

$$Y(s_t, a_t) = \frac{1}{kN} \sum_{i=1}^{kN} \delta(y_i(s_t, a_t))$$

The truncation of quantiles from the target distribution aims to prevent overestimation bias, with the number of dropped targets per network $d = M - k$ being a hyperparameter tuned per environment.

The networks are then trained by minimizing the quantile Huber loss:

$$L(s_t, a_t; \theta_n) = \frac{1}{kNM} \sum_{m=1}^{M} \sum_{i=1}^{kN} \rho_H^{\tau_m}(y_i(s_t, a_t) - \theta_{m,n}(s_t, a_t))$$

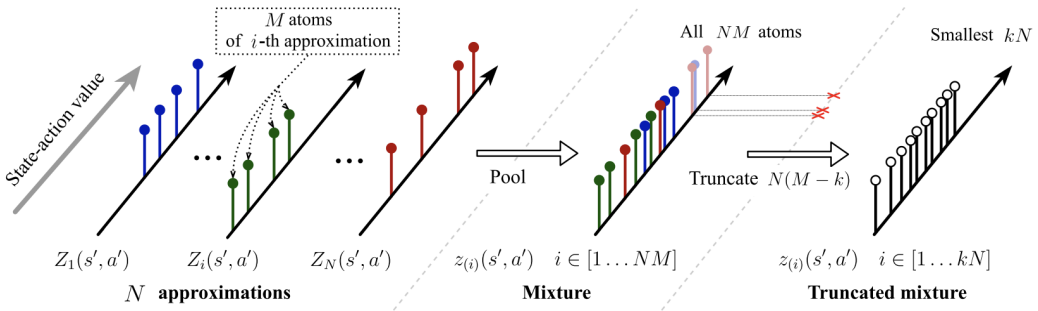
where

$$\rho_H^\tau(u) = |\tau - \mathbb{1}(u < 0)| L_1^H(u)$$

and $L_1^H(u)$ denotes the Huber loss with a parameter of 1.

The policy is trained similar to SAC, aiming to maximize an entropy-penalized estimate of the Q-value:

$$J(\phi) = \mathbb{E}_{s \sim D, a \sim \pi_\phi} \left[\frac{1}{NM} \sum_{m=1}^{M} \sum_{n=1}^{N} \theta_{m,n}(s, a) - \alpha \log \pi_\phi(a|s) \right]$$



2.2 ACC algorithm description

ACC is designed to dynamically adjust the bias in TD targets across a range of environments. It involves a bias-controlling mechanism that adaptively adjusts based on observed discrepancies between the estimated Q-values and actual returns. This mechanism is particularly crucial in continuous action spaces where direct maximization is computationally infeasible, necessitating reliance on actor-critic frameworks where the actor maximizes the expected Q-value, denoted as $Q^\pi(s,a) = \mathbb{E}[R^\pi(s,a)]$.

ACC introduces a set of estimators $\{Q^\beta(s,a)\}$ for each state-action pair (s,a), defined over a parameter β ranging within $[\beta_{\min}, \beta_{\max}]$. These estimators are designed to ensure that the true Q-value $Q(s,a)$ is bounded by $Q^{\beta_{\min}}(s,a)$ and $Q^{\beta_{\max}}(s,a)$, with $Q^\beta(s,a)$ being a continuous and monotone increasing function in β.

The unbiased estimator $Q^{\beta^*}(s,a)$ for $Q(s,a)$ is obtained by averaging samples of discounted returns $R(s,a)$, and β^* is dynamically adjusted to minimize the difference between this estimator and the averaged returns. This dynamic adjustment is performed periodically, at the end of episodes, and after a predefined number of environment steps (T_β), as described in the algorithm:

$$\beta_{\text{new}} = \beta_{\text{old}} + \alpha \sum_{t=1}^{T_\beta} (R(s_t, a_t) - \hat{Q}(s_t, a_t)), \tag{2}$$

where α is a step-size parameter and t indexes over the most recent T_β state-action pairs. This update reduces β in cases of overestimation and increases it when underestimation is detected.

ACC's general form can be defined specifically within the framework of TQC, allowing for the selective dropping of quantile targets from a pooled set. The number of targets dropped, denoted by d, is derived from β as $d = d_{\max} - \beta$, where $d_{\max}$ is the maximum allowable drop, and $\beta_{\min} = 0$, $\beta_{\max} = d_{\max}$.

2.3 Hyper-parameter Tuning

Increasing the frequency of critic updates per environment step is shown to accelerate learning [2]; however the rapid modification of targets increase the likelihood of divergence leading to instability. Our experiment tested critic update rates of 1, 2, and 4 times per iteration. The results, as depicted in Figure 1, indicate that increasing the update frequency to four times per iteration significantly enhances sample efficiency, with convergence occurring at 60 episodes, compared to 160 episodes at the baseline once per iteration. However, it is important to note that quadrupling the update frequency resulted with computation time also increasing fourfold. We also investigate the impact of number of Z (critic) networks on sample efficiency. The recommended parameter is 5 critic networks [5], which is shown in Figure 1 to significantly outperform configurations with two. Testing further, we found that increasing to ten critic networks yields marginally better results than five but comes with considerable increases in computational demand.

2.4 Other Experiments

We also explored alternative strategies such as Sample Multiple Reuse (SMR) [6], where we repeatedly update the model using a fixed batch of samples within a single optimization loop. While the REDQ-SMR variant demonstrated early signs of convergence within just 30 episodes, the extensive training times rendered a full-scale implementation impractical. Conversely, increasing the number of critic updates in the ACC framework yielded a comparable improvement in sample efficiency but was computationally more efficient.

3 Convergence Results

As depicted in Figure 3, ACC demonstrated successful convergence in an easier setting, exhibiting initial signs of convergence around the 60th episode. The performance consistently improved, reaching a 100 moving average score of 300 after approximately 310 episodes.

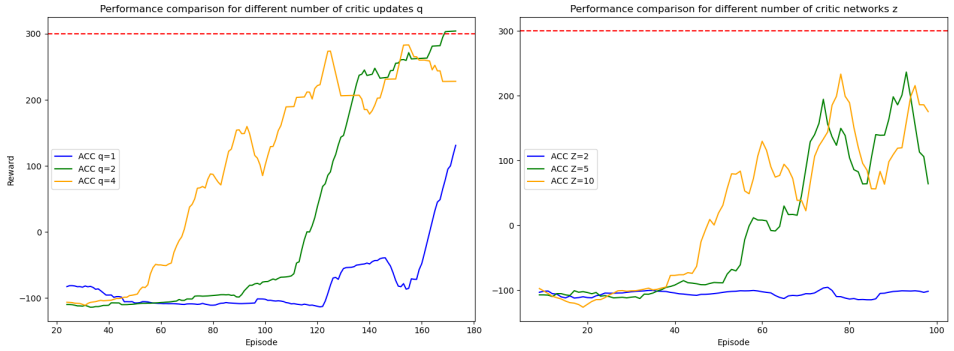


Figure 1: Results by Critic Network Count Z and Update Frequency q for first few episodes

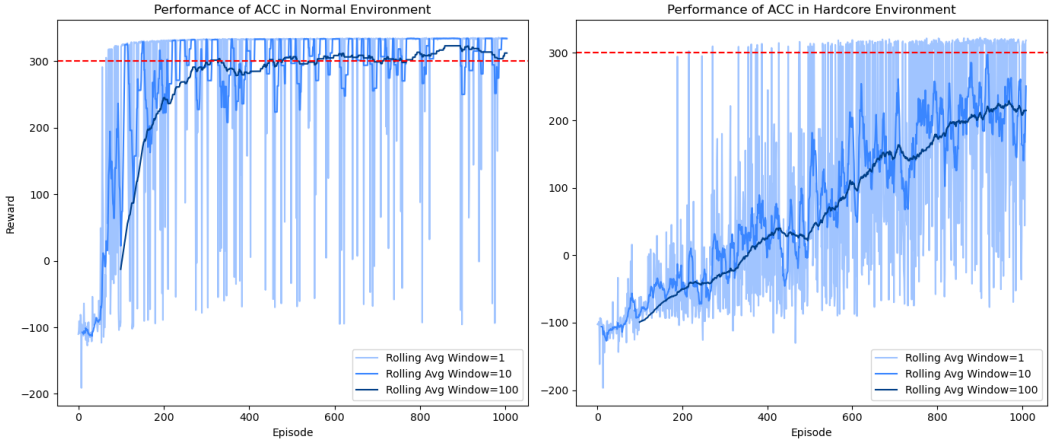


Figure 2: ACC performance on both Environments over 1000 episodes

Conversely, in the more challenging hardcore environment, the algorithm did not achieve a consistent 100 moving average score of 300. While episodes scoring above 300 were first recorded post the 200th episode, these instances became more frequent after the 500th episode. The highest score within the first 1000 episodes was recorded at 320. Despite these challenges, the agent exhibited gradual improvement, suggesting that continued training could potentially yield closer approach towards finding an optimal policy.

4 Limitations

Increasing the number of critic updates enhances the learning process but introduces greater instability into the training, necessitating occasional restarts. Furthermore, training with four critic updates quadruples the duration compared to a single update, thereby complicating hyperparameter tuning due to extended training times.

Future Work

The application of the SMR technique within the REDQ-SMR architecture demonstrated promising results. However, time constraints limited the feasibility of using this method. Future work should focus on a comparative analysis of REDQ against the results currently achieved. Additionally, further refinement of ACC parameters could yield strategies more precisely adapted to the specific environment.

References

[1] S. Dankwa and W. Zheng. "Twin-Delayed DDPG: A Deep Reinforcement Learning Technique to Model a Continuous Movement of an Intelligent Robot Agent". In: *Proc. 3rd International Conference on Vision, Image and Signal Processing (ICVISP)*. Vancouver, BC, Canada: Association for Computing Machinery, 2020, pp. 66–70. ISBN: 9781450376259. DOI: 10.1145/3387168.3387199.

[2] Nicolai Dorka et al. *Adaptively Calibrated Critic Estimates for Deep Reinforcement Learning*. 2022. arXiv: 2111.12673 [cs.LG].

[3] GymLibrary. *Bipedal Walker*. https://www.gymlibrary.dev/environments/box2d/bipedal_walker/. Accessed: day-month-year.

[4] Tuomas Haarnoja et al. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. 2018. arXiv: 1801.01290 [cs.LG].

[5] Arsenii Kuznetsov et al. *Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics*. 2020. arXiv: 2005.04269 [cs.LG].

[6] Jiafei Lyu et al. *Off-Policy RL Algorithms Can be Sample-Efficient for Continuous Control via Sample Multiple Reuse*. 2023. arXiv: 2305.18443 [cs.LG].