AFU study

Abstract

Introduction

Related Work

Off-policy reinforcement learning has become essential for efficient data usage, especially in domains such as robotics where interactions are costly or risky. Traditional off-policy algorithms, such as Deep Deterministic Policy Gradient (DDPG) [Biblio] and its improved variant (TD3) [Biblio], rely on deterministic actor-critic architectures. These approaches, while sample-efficient, are prone to instability due to overestimation biases and convergence to local optima. Soft Actor-Critic (SAC), a more recent off-policy algorithm, addresses some of these issues by incorporating entropy regularization, but it still faces challenges in offline settings where the distribution of data may differ significantly from the target policy.

Soft Actor-Critic (SAC) [Biblio] is a widely used off-policy deep reinforcement learning algorithm based on the maximum entropy framework. SAC jointly learns a stochastic policy (actor) and two Q-value functions (critics), with a value function used for stability. The actor is updated to maximize expected reward while also maximizing entropy, encouraging exploration and robustness. SAC is highly sample-efficient and performs well in continuous control tasks, but it relies heavily on accurate Q-value estimation and often struggles in purely offline settings due to value overestimation and distributional shift between the behavior and target policies.

Several algorithms have been developed to address the challenges of reinforcement learning in both online and offline settings. Among these, Implicit Q-Learning (IQL), and Calibrated Q-Learning (Cal-QL) stand out for their performance and conceptual innovations.

To address some of the challenges in offline reinforcement learning, Implicit Q-Learning (IQL) [Biblio] introduces a conservative update strategy that avoids explicit policy learning. Instead of using the actor-critic paradigm, IQL trains a Q-function and a value function from a fixed dataset, using quantile regression to mitigate overestimation. The policy is then implicitly defined via advantage-weighted regression, without requiring interaction with the environment. This separation makes IQL robust to distribution shift and improves performance in the offline RL setting. However, IQL is sensitive to the choice of expectile parameter, which hinders its extension to online learning.

Building on the idea of conservative Q-learning, Calibrated Q-Learning (Cal-QL) [Biblio] further refines Q-value estimation by incorporating calibration techniques. The motivation behind Cal-QL is that many offline RL failures stem from poor calibration of the learned Q-function, which can result in incorrect value targets and unstable learning. Cal-QL introduces a calibration loss that penalizes Q-values inconsistent with observed returns, thereby aligning the learned Q-function more closely with the empirical data distribution. This leads to more reliable policy evaluation and selection, particularly in high-stakes or high-variance environments. Nevertheless, such methods remain sensitive

to distributional shifts during the transition to online learning. It was also proven that the performance of Cal-QL plumets after 200k steps of training, which is a significant drawback for online learning.

In contrast, Actor-Free Updates (AFU) [Biblio] proposes a fully decoupled critic learning mechanism, where Q-value targets are estimated via regression and conditional gradient scaling. This design enables the critic to learn from arbitrary data, including random or suboptimal transitions, opening the door to truly off-policy and robust RL algorithms in both online and offline settings.

Background

We consider a standard Markov Decision Process (MDP), defined as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, R, \gamma)$, where $\mathcal{S}$ is a continuous state space, $\mathcal{A}$ a continuous action space, T the transition probability, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, and $\gamma \in [0, 1)$ the discount factor. The goal is to learn a policy π that maximizes the expected cumulative reward:

$$\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right].$$

The optimal Q-function $Q^*(s, a)$ is defined as:

$$Q^*(s, a) = \mathbb{E}_{\pi^*} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a \right],$$

where π^* denotes the optimal policy.

In continuous action spaces, computing $\max_a Q(s, a)$ is intractable, which motivates the use of parametric actors or regression-based methods like AFU to approximate this maximization.

Methods

Environment

In order to test our algorithms, we wrapped the gymnasium environments in personalized wrapper that contains the following methods: _set_state, get_obs, get_observation_space and get_action_space.

Algorithms

We reimplemented DDPG, SAC, AFU, IQL and Cal-QL with the BBRL library.

Overview

AFU (Actor-Free Updates) is an off-policy reinforcement learning algorithm that decouples the critic update from the actor. The core innovation lies in how AFU solves the *max-Q problem*: instead of relying on an actor network to approximate $\arg\max_a Q(s, a)$, it trains a value network $V_\phi(s)$ via regression to estimate $\max_a Q(s, a)$ directly.

AFU employs three types of networks:

– A Q-function $Q_\psi(s, a)$, trained using temporal-difference learning;

– A value network $V_{\phi_i}(s)$, trained to approximate $\max_a Q(s, a)$;

– An advantage network $A_{\xi_i}(s,a)$, enforcing $Q(s,a) \approx V(s) + A(s,a)$, with $A(s,a) \leq 0$.

Solving the Max-Q Problem

The value and advantage networks are trained using the following soft-constrained loss:

$$\Lambda'_{V,A}(\phi,\xi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\big[Z\big(\Upsilon^a(s) - Q_\psi(s,a), A_\xi(s,a)\big)\big],$$

where

$$Z(x,y) = \begin{cases}(x+y)^2, \text{if } \ x \geq 0\\ x^2+y^2, \ \text{otherwise}\end{cases}$$

, and $\Upsilon^a(s)$ modulates the gradient update to down-weight increases in V_ϕ when they would otherwise overestimate Q. This conditional gradient rescaling avoids the instability issues observed in regularized approaches like IQL or SQL.

Actor Training in AFU-alpha

In AFU-alpha, a stochastic actor π_θ is used for data collection and trained using a SAC-style loss:

$$L_\pi(\theta) = \mathbb{E}_{s,a\sim\pi_\theta}\big[\alpha \log \pi_\theta(a \mid s) - Q_\psi(s,a)\big],$$

$$L_{\text{temp}}(\alpha) = \mathbb{E}_{s,a\sim\pi_\theta}\big[-\alpha \log \pi_\theta(a \mid s) - \alpha \bar{H}\big],$$

where $\bar{H}$ is a target entropy and α is a temperature parameter optimized during training.

AFU-beta: Actor Guidance via Gradient Projection

To improve upon AFU-alpha and address SAC-like failure modes, AFU-beta introduces a regressed guidance network $\mu_\zeta(s)$, trained to match actions with high Q-values. Actor gradients are then projected to avoid directions that deviate from $\mu_\zeta(s)$ in low-value regions:

$$\mathbf{G}_{s,a}\big(\nabla_a Q_\psi(s,a)\big) = \begin{cases}\text{proj}_{\perp(\mu_\zeta(s)-a)}\big(\nabla_a Q_\psi(s,a)\big), \text{if misaligned and } \ Q(s,a) < V(s),\\ \nabla_a Q_\psi(s,a), \qquad\qquad\qquad\qquad \text{otherwise .}\end{cases}$$

This results in a modified policy gradient $\nabla_\theta^{\text{MODIF}} L_\pi(\theta)$, which helps prevent the actor from becoming trapped in local optima.

Connection to Research Objectives

AFU's design directly addresses our two research hypotheses:

1. **Learning from Random Data:** Since AFU's critic is actor-independent, it can learn effectively from random or off-policy transitions, unlike SAC or DDPG.
2. **Offline-to-Online Stability:** AFU's critic does not suffer from over- or underestimation when exposed to unseen actions during the transition from offline to online learning, offering more stable performance.

In this study we will try to validate these hypotheses by creating various experimental settings. We will also compare AFU with other algorithms such as SAC, IQL and Cal-QL.

Experimental study

Off and On-Policy Training

First Results.

We evaluated AFU on the CartPoleContinuous environment and obtained results that closely match those observed using Mr. Perrin's original implementation. [These results are included in the figures attached to this report] – Mettre les figures. AFU significantly outperforms SAC on this task. Notably, AFU achieves these results in only 50k iterations, compared to the 200k required by SAC. As shown in the histograms, AFU also demonstrates greater consistency, suggesting improved stability and reliability during training.

However in the Pendulum environment, despite multiple runs, AFU performs poorly, failing to converge even after 1 million steps. In contrast, SAC reliably converges in under 50k steps. Given these results, we did not proceed with Off-Policy training for AFU on this environment.

Finally, we conducted experiments on LunarLander. This time, AFU performed well and successfully converged. We runned 1 million steps, but analysis of the learning curve indicates that convergence was already achieved by around 200k steps. We provide histograms showing performance distributions, which (albeit potentially exaggerated) suggest that AFU substantially outperforms SAC on this task.

Encouraged by these results, we tested AFU in an Off-Policy setting on LunarLander, using 200k training steps. Unfortunately, the results were similar to those of SAC: neither algorithm achieved meaningful convergence. To verify whether this was due to insufficient data, we extended the training to 1 million steps, but AFU still failed to converge under Off-Policy training.

These results suggest that while AFU performs competitively–and at times better than SAC–in On-Policy settings, its Off-Policy capabilities remain limited, particularly in more complex environments like LunarLander.

Make AFU more intersting than SAC in an Off-Policy setting.

Our preliminary results suggest that the initial hypothesis – that AFU can effectively learn from uniformly generated state-action data – is likely incorrect. This observation has led us to explore potential reasons behind this behavior and to design alternative experimental settings that could both validate AFU's utility and demonstrate its Off-Policy potential compared to algorithms such as SAC. We outline two such experiments below.

1. Varying the Degree of Policy-Driven Behavior via an Epsilon Parameter

A straightforward experimental modification involves introducing a tunable parameter $\varepsilon \in [0, 1]$. At each step of data generation, a uniform random variable is sampled. If the sample is less than ε, a state-action pair is drawn uniformly at random, mimicking our prior Off-Policy setting. Otherwise, the action taken is the one prescribed by the current policy, corresponding to the On-Policy setting.

This setup allows us to generate a range of datasets interpolating between the On-Policy regime ($\varepsilon = 0$) and fully random Off-Policy regime ($\varepsilon = 1$). We propose to plot the performance across varying ε values, using statistical summaries such as whisker plots (e.g., Q1, IQM, Q3) to visualize trends. If AFU is truly more Off-Policy capable than SAC, its performance degradation should be less pronounced as ε increases. This experiment offers a simple yet effective framework for assessing the Off-Policy robustness of learning algorithms. [Résultats à inclure dans le rapport]

2. State-Space Constraints and the Role of Accessible States

To better understand the mechanisms behind AFU's behavior, we return to the tabular case. In this setting, it is possible to explicitly construct a lookup table that maps each state to the optimal action over an infinite horizon. When the state space is sufficiently small, exhaustive exploration via random state-action pairs can, in principle, lead to convergence—even for algorithms that are not inherently Off-Policy.

For example, in a simple 4x5 maze with 4 actions per state, the total number of state-action pairs is only 80. After sufficient repetitions (e.g., 80k samples), learning the optimal action mapping becomes feasible via brute force. In deep reinforcement learning, such a lookup table is no longer feasible due to the continuous and high-dimensional nature of the state and action spaces. However, the underlying goal — learning a function that approximates such a mapping — remains intact.

This perspective sheds light on our results in CartPoleContinuous. Despite the continuous nature of the problem, its effective state-space dimensionality is low. The first state variable (cart position) exhibits a high degree of symmetry and repetition, and we further constrained velocities to lie between -8 and $+8$. These simplifications reduce the practical complexity of the environment. Thus, brute-force learning may still succeed, even when relying on random data.

Importantly, much of the theoretical state-space is not reachable during normal episodes. For example, while velocity can range from $-\infty$ to $+\infty$, in practice, only a narrow subspace of values is observed. We refer to this subspace as the accessible state space. Uniformly sampling across the full space may therefore introduce states that are highly unrealistic and ultimately irrelevant to learning a useful policy.

This observation generalizes. Consider a tabular environment with 100B states, where the optimal trajectory only spans 10 states. Random sampling would rarely encounter meaningful states, and learning would stagnate. In LunarLander, we observed behaviors in which the agent exits the visible screen or exhibits unusual dynamics such as rotating mid-air — scenarios that are rarely encountered under a standard policy. Learning accurate Q-values for such states may come at the expense of learning for states within the accessible region.

Based on this, we propose a second experiment. First, define bounds on each dimension of the state space to better approximate the accessible subspace. Training the algorithm solely within these constraints may yield better convergence. We can then extend this by performing random walks through the environment: sample an initial state within the constrained bounds, and then take random actions for each step. This would naturally constrain the set of states encountered during training.

Such an experimental setup simulates learning from behavior that is entirely unrelated to the agent's own policy. A critic capable of learning accurate Q-values in this setting – as AFU purports to do – should outperform critics like SAC, which rely on the actor for accurate Q-value estimation. This setup thus offers a clearer test of the Off-Policy learning capacity of AFU.

The same ε-based framework from the first experiment could be employed here, offering another way to measure degradation in performance as the data distribution diverges from the policy. [Résultats à inclure dans le rapport]

Offline to Online Training

First Results.

We also evaluated AFU in an Offline-to-Online setting, where the agent first learns from a fixed dataset and then transitions to online learning. We used the same datasets as in the previous experiments, but this time we trained AFU for 1 million steps in an offline setting before switching to online training.

Conclusion

Appendix A

Experimental Details

Environment Wrappers.

The custom environment wrappers used in this study include the following methods:
– `_set_state`: Allows resetting the environment to a specific state.
– `get_obs`: Retrieves the current observation.
– `get_observation_space`: Returns the observation space of the environment.
– `get_action_space`: Returns the action space of the environment.

These wrappers were implemented to ensure compatibility with the algorithms and to facilitate reproducibility of the experiments.

Hyperparameters.

The following hyperparameters were used for the experiments:
– Learning rate:
– Discount factor (γ):
– Batch size:
– Replay buffer size:
– Target network update rate (τ):
– Temperature parameter (α):

Reproducibility

The code for all algorithms and experiments is available at [GitHub Repository]. The repository includes detailed instructions for setting up the environment and running the experiments.

Additional Figures

Figures referenced in the report, including learning curves and performance histograms, are provided in the supplementary materials. These figures illustrate the comparative performance of the algorithms across different environments and settings.

Appendix B

Cal-QL errors

The Cal-QL algorithm was observed to exhibit significant performance degradation after 200k steps of training. This behavior was consistent across multiple runs and environments, indicating a potential issue with the algorithm's stability or convergence properties in the offline setting.