Robot Learning A.Y. 24/25 - Lab 3

Mattia Sabato
DAUIN, Dipartimento di Automatica e Informatica
Politecnico di Torino, Italy
mattia.sabato@studenti.polito.it

Abstract: This paper presents an analysis and comparison of different policy gradient methods applied on a continuous version of the Cart-Pole environment. Starting from the most basic REINFORCE, for which we show its convergence instabilities and some variants able to achieve superior performances, we then move to Actor-Critic methods, which are instead able to efficiently solve the environment through the combination of policy optimization with value function approximation.

1 Introduction

Policy gradient methods represent a class of techniques designed to directly learn a policy, i.e., a decision-making strategy, rather than approximating state or action-value functions as evaluators. These methods are particularly well-suited for environments with high-dimensional or continuous action spaces, where value-based methods often encounter significant challenges. One of the key advantages of policy gradient methods is their intrinsic stochasticity, which can be especially beneficial in handling aliased states – situations where similar states require varied actions. This stochastic behavior allows agents to incorporate randomness into their actions, which proves to be advantageous in many cases. Despite their strengths, policy gradient methods are prone to high variance and training instabilities. These challenges often arise from sudden changes in the policy, which can result also from small adjustments to the parameters during training. To address these issues, techniques such as introducing baselines – which help distinguish above-average rewards from below-average ones – and employing a *critic* to estimate value functions are commonly used. To evaluate the convergence properties of various policy gradient algorithms, we conducted the experiments in a modified Cart-Pole environment characterized by a continuous scalar action space, where the action's value directly determines the magnitude of the force applied to the cart.

2 REINFORCE

The first policy gradient method we implemented and tested is REINFORCE, an episodic algorithm that updates the parameters θ of a stochastic policy $\pi_\theta(a|s)$ at the end of every episode, each characterized by a variable number of timesteps T, using the following rule:

$$\theta \leftarrow \theta + \alpha \frac{1}{T} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(s_t, a_t) v_t \tag{1}$$

where α is the step size and v_t represents the discounted return calculated using a discount factor $\gamma = 0.98$. Whereas the use of the logarithm in (1) is primarily motivated by mathematical convenience, the role of v_t is not easily inferrable. For example, it can be considered as a weight capable of providing that supervision lacking in Reinforcement Learning (RL), thus creating a parallel with maximum likelihood estimation. However, its presence is actually needed to extend learning policies using simple likelihood ratios over One-Step Markov Decision Processes (MDPs) to the Multi-Step scenario.

7th Conference on Robot Learning (CoRL 2023), Atlanta, USA.

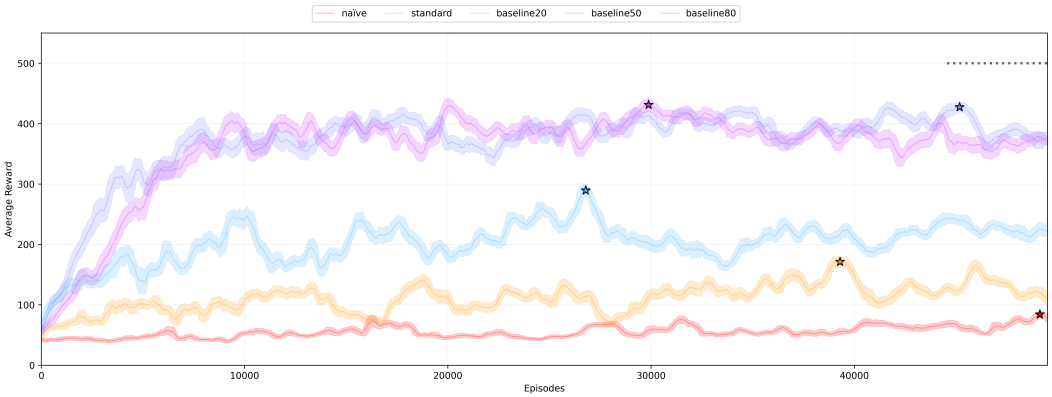

Figure 1: Average total reward computed using a sliding window of length 500 over the mean across 30 independent runs of 50,000 episodes. Different implementations yield very different behaviors, the worst being the naïve version and the best being the one using a constant baseline. Standardizing the rewards is not sufficient to improve the performance, whereas increasing the baseline has a positive outcome until the reaching of a plateau. The markers represent when the average highest reward is reached and its value, the dotted line represents the highest achievable reward given the fixed limit of 500 timesteps for each episode.

In the first case, the policy gradient is simply $\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s, a)r]$, with r representing the immediate reward after choosing the action a in state s. In the second case, where we generalize to arbitrary MDPs, we replace r with $Q^{\pi_\theta}(s, a)$, which represents the long-term value of choosing that action. Finally, the use of the return v_t boils down to the fact that it is an unbiased sample of the true action-value function, thus allowing us to consider any Multi-Step MDP. Having a continuous action space, we adopted a Gaussian policy $a \sim \mathcal{N}(\mu(s), \sigma^2)$, with a fixed variance of 5 and a state-dependent mean learned during training. A naïve approach as the one presented in (1) is extremely inefficient and is unable to learn effectively when used in the continuous Cart-Pole environment, as shown in Figure 1. This is likely due to the fact that the cumulative reward of each episode can greatly vary from one rollout to the other, eventually affecting the training stability. Moreover, having a non-negative reward at each timestep, with no reference to an average one signaling how good the agent has been performing so far, can further increase the variance. All these situations may induce a never ending degradation of the policy's quality that, not being able to recover from it, breaks the training procedure preventing the agent to reach convergence. To mitigate the impact of these problems, we tested two different approaches and analyzed the performance change obtained. The first variant implies standardizing the rewards of each rollout as follows:

$$z_t = \frac{v_t - \hat{\mu}_v}{\hat{\sigma}_v}, \quad z \sim \mathcal{N}(0, 1^2) \tag{2}$$

where $\hat{\mu}_v$ is the estimated mean of the rewards of the last rollout, $\hat{\sigma}_v$ is their estimated standard deviation, and z_t represents the standardized return that will be used at training time. As expected, thanks to the substitution of the absolute magnitude with a relative one, this modification allows to reach slightly better performances, which are still not enough to solve the environment. A further option is to use a constant baseline B, whose role is to promote only returns higher than its value rather than all positive ones, implemented as follows:

$$\theta \leftarrow \theta + \alpha \frac{1}{T} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(s_t, a_t)(v_t - B) \tag{3}$$

where we considered $B \in \{20, 50, 80\}$ for all our experiments. The results demonstrate a positive correlation between the value of the baseline and the agent's performance. However, this improvement quickly saturates, reaching a plateau for a baseline value of approximately 80.

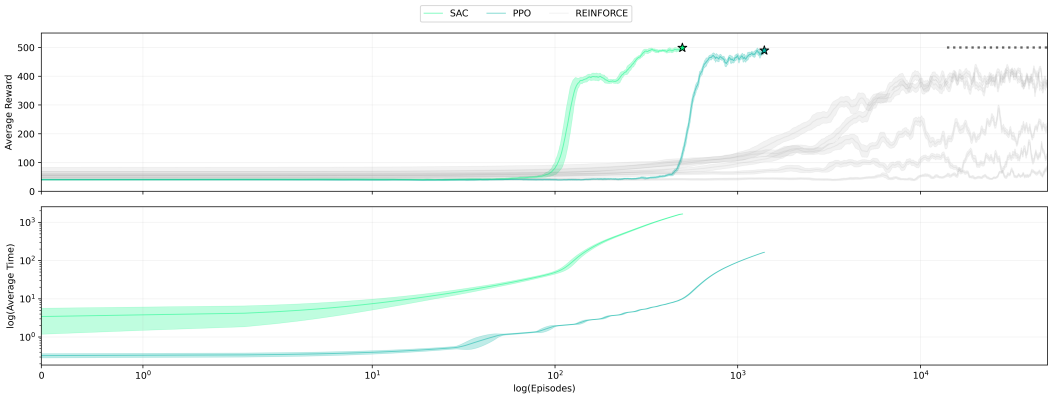

Figure 2: Comparison between the average reward and the average time needed for PPO and SAC with respect to REINFORCE variants, computed over 30 independent runs with a fixed timesteps limit of 500. Actor-Critic methods strictly outperform standard ones, reaching higher rewards in a fraction of time. The wall-clock time needed drastically changes for each algorithm, making PPO the one reaching the best compromise between performance and efficiency.

This indicates that while the inclusion of a baseline is effective in reducing variance, the simplistic choice of a constant value is insufficient to fully address the challenges of the environment. The primary limitation is related to the inability to capture the relative quality of different states, as it assigns the same expected return to all states regardless of their individual characteristics. A more refined approach involves using a state-dependent baseline, $B(s)$, which reflects the expected return of being in a specific state. Ideally, it would approximate the value function $V^{\pi_\theta}(s)$, representing the long-term return expected under the current policy starting from state s. We can thus rewrite (3) as follows:

$$\theta \leftarrow \theta + \alpha \frac{1}{T} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(s_t, a_t)[Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)] \tag{4}$$

where the last term computed through the difference between the action-value and the value function is simply the advantage function $A^{\pi_\theta}(s, a)$. The problem thus reduces to the estimation of the latter. In summary, in the environment we considered in our experiments, the use of a baseline proves to be particularly useful, since the agent receives a unitary positive reward for each timestep of the episode. On these bases, we may expect a large variance in rewards if considering their absolute magnitude, and a lower one if considering their deviation from an expected value. By introducing a baseline, we can reduce the variance without changing the expectation, thus resulting in a stabler training.

2.1 Practical Issues

The above discussion has been carried on considering a simulated environment, where testing the behavior of the agent is almost immediate and virtually without consequences. However, when deployed in the real world, many issues may arise, strongly related to how we designed the training procedure. One potentially dangerous aspect, overlooked in the previous section, is the use of an *unbounded* continuous action space with no penalization. If a physical system were to be used with an unbounded policy learned in simulation, it could happen that the former ends up in a state configuration, eventually not properly learned during training, for which the underlying model outputs an extremely large value, e.g. related to the force to apply. Unlike in simulation, where an "episode" can simply be reset without lasting consequences, real-world systems cannot be restarted so easily, with eventually permanent consequences related to the actions executed by the agent. In the continuous Cart-Pole environment, for instance, the reward structure – assigning a reward of $+1$ for every timestep – focuses only on the survival of the agent, not considering any penalties for large

or extreme outputs. This myopic reward design encourages survival at all costs, at the same time disregarding the broader implications of the agent's actions, which may result in dangerous or impractical behavior. One possible solution would be to incorporate a penalty into the reward by taking into account also the output of the model, thus reducing the gain the agent has in choosing extreme values. Other solutions may include the use of transformations on the output, e.g. through the use of a logarithm in order to relax the impact of large magnitudes, considering a lower (and eventually parameterized) variance or by considering regularization terms penalizing the absolute value of the output at training time. Another important observation is that, while policy gradient methods are often applied to continuous action spaces, they can also be easily extended to discrete action spaces with few modifications. The primary one involves using a categorical distribution, rather than a normal one, as the model's output to represent the probabilities of selecting each discrete action. One challenge associated with this approach arises when dealing with a large action space. A large number of possible actions, each requiring a nonnegative probability of being sampled, can make the sampling process less efficient, making it difficult to sample the best action over a so large pool of alternatives. However, this is still more effective than value-based methods, where reliable estimations must be computed for numerous combinations of actions and states. For moderately large discrete action spaces, another potential issue can occur during policy updates. Unlike continuous outputs, where small parameter changes are likely to result in gradual adjustments, discrete actions may be affected by sudden shifts in the associated probabilities. These changes, arising from small variations in the model parameters, may lead to suboptimal behaviors during training, eventually leading the agent into a never-ending loop of performance degradation.

3 Actor-Critic Methods

Problems related to convergence stability prevent basic policy gradient algorithms from achieving a satisfactory policy. As analyzed in the previous section, in episodic tasks we can use the empirical return v_t as an estimator for $Q^{\pi_\theta}(s, a)$, thus leveraging experimental information. An alternative road involves using a parameterized function $Q_w^{\pi_\theta}(s, a)$ to approximate the true value, increasing both efficiency and stability. The set of techniques approximating both the policy and the action-value function takes the name of Actor-Critic methods and have shown remarkable improvements over standard policy gradient ones. When combined with the use of a baseline $B(s)$, which is nothing else than a parameterized function $V_u^{\pi_\theta}(s)$ mimicking the true value function, it results that we do not actually need to use several models to estimate both the action-value and the value function, being the advantage an unbiased estimate of the TD-error. The role of the Critic thus boils down to estimate the latter, requiring just one set of parameters other than the ones used by the Actor.

Algorithm	`Hyperparameter`	Value
PPO	`lr`	0.0003
	`n_steps`	2,048
	`batch_size`	64
	`gamma`	0.99
	`gae_lambda`	0.95
SAC	`lr`	0.0003
	`buffer_size`	1,000,000
	`learning_stars`	100
	`batch_size`	512
	`tau`	0.005
	`gamma`	0.995
	`train_freq`	1
	`gradient_steps`	1
	`target_update_interval`	1

Table 1: Hyperparameter configurations for PPO and SAC, with most values aligning with their respective default settings and having made only minor adjustments to a subset. This highlights the robustness of these algorithms, demonstrating their ability to perform effectively with minimal hyperparameter tuning.

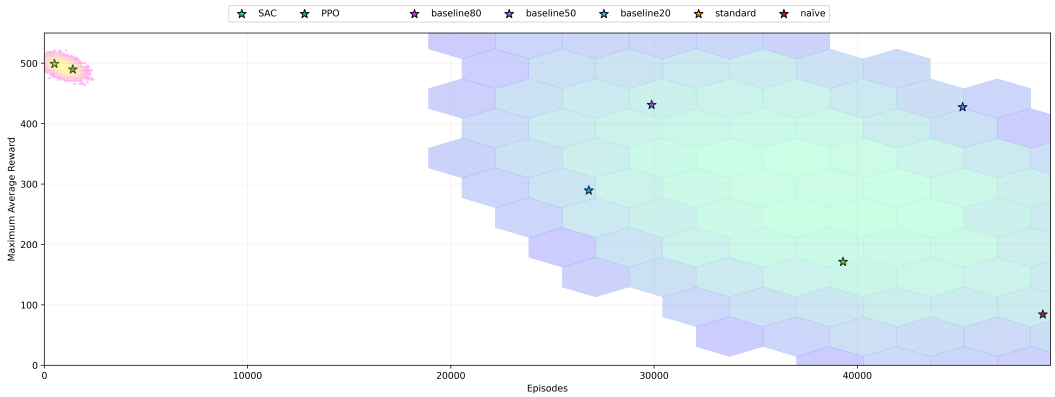

Figure 3: Scatter plot representing how the average maximum rewards are distributed with respect to the algorithm used and the number of episodes needed. The shaded areas represent confidence maps into which we can expect to find other points when repeating the experiments, the last being stochastic and not easily predictable. The large gap between the two distributions highlight how Actor-Critic methods are far superior than REINFORCE and its variant, showcasing better performances and lower variance.

In order to test the effectiveness of these methods and compare them against the results obtained in the previous section, we conducted several experiments considering the Proximal Policy Optimization (PPO) and the Soft Actor-Critic (SAC) algorithms. An initial search aiming to find the best hyperparameters for each has led to the combinations shown in Table 1, which yield the best performances and thus employed for all subsequent experiments. The use of a critic greatly improves the performances, as shown in Figure 2. Unlike basic approaches, convergence is achieved not only faster, but also to the maximum achievable cumulative reward. Moreover, the wall-clock time needed for each implementation is just a fraction of the one required by standard REINFORCE, which in general may need an enormous amount of episodes due to its low sample efficiency and high variance. Furthermore, PPO is an on-policy algorithm, but it prevents destructively large policy updates through the use of clipping and trust regions, whereas SAC, being an off-policy algorithm, can efficiently reuse previous experience. These features are crucial and particularly effective in making these two algorithms a preferred choice with respect to standard REINFORCE.

A comparison between SAC and PPO shows also how the first is less stable but more sample efficient, whereas the second has a smooth convergence but also needs more data. This is a direct consequence of the advantages and disadvantages of using, respectively, off-policy and on-policy methods. For the environment considered here, where each trajectory is not expensive to get, PPO could be considered the best alternative, as also confirmed in the experiments, being able to reach convergence in more episodes than SAC, but in much less time. Having the latter also to deal with a replay buffer and compute gradients eventually at every step, the time needed can quickly blow up. Finally, the standard REINFORCE, together with its variants, suffers from extreme instabilities at training time, and is not able to find a satisfactory policy even when provided with a larger number of training episodes with respect to the Actor-Critic-based counterparts. In Figure 3 is shown how the best rewards are distributed with respect to the episodes in which they are obtained and the algorithm used. We can derive two regions characterized by very different expected probability distributions of finding other maximum rewards by changing some hyperparameters or repeating the training. Whereas the one corresponding to algorithms relying on REINFORCE has a much larger covariance and a mean skewed towards low rewards and high number of episodes, the one associated with Actor-Critic methods has opposite features, making it far more appealing than the former.