COMP579: Assignment 3

Jacqueline Wu, Sam Zhang

March 2024

1 Value-based method with linear function approximation

We implement Q-Learning and Expected SARSA and experiment them within two environments from the Gymnasium, an open-source fork of the OpenAI Gym.[2] During the experiment, the environment's states are discretized through tile-coding into a binary vector, with a randomized offset for each tile.

1.1 Methodologies

We experiment with a fixed set of 3 different learning rates $\alpha \in \{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}\}$. For each learning rate, we also experiment with 3 different ϵ, where one 1 is an exponentially decaying ϵ and the other 2 are fixed. We use decaying ϵ to gauge the effect of encouraging exploration in the early stages of learning and eliminating it in the later stages.

For each learning rate, we divide them by the number of tilings to avoid integer overflow during the experiment.

1.2 Results

The results of the experiments are shown in Figure 1, each with their tile coding parameters and their values of α are post-division by the number of tilings.

We observe that both experiments learned CartPool-v1 poorly, and stopped learning early on, achieving rewards of sub-30. Out of all combinations, the best-performing combination for both algorithms is $\alpha = \frac{1/16}{\text{number of tilings}}, \epsilon = 0.25$, where learning happened the fastest and remained the most stable.

Both algorithms performed better on MountainCar-v0, with the best performing Expected-SARSA achieving consistent rewards of reward ≈ -150, and Q-Learning achieving reward ≈ -130. The best performing hyperparameters combination for both algorithms is $\alpha = \frac{1/4}{\text{number of tilings}}, \epsilon = 1 \times 0.995^{\text{episode}}$.

1.3 Discussion

We begin by highlighting the poor performance of value-based linear approximation on CartPole-v1. This is the case for both algorithms and a wide range of hyperparameter combinations; no algorithm stood out as being particularly better than the other, potentially due to the 4-dimensional state space being too complex for a linear approximation to capture.

The results do show a few characteristics with different hyperparameters, such as better stability with a lower learning rate, and decaying ϵ preventing the model from learning at all, but the

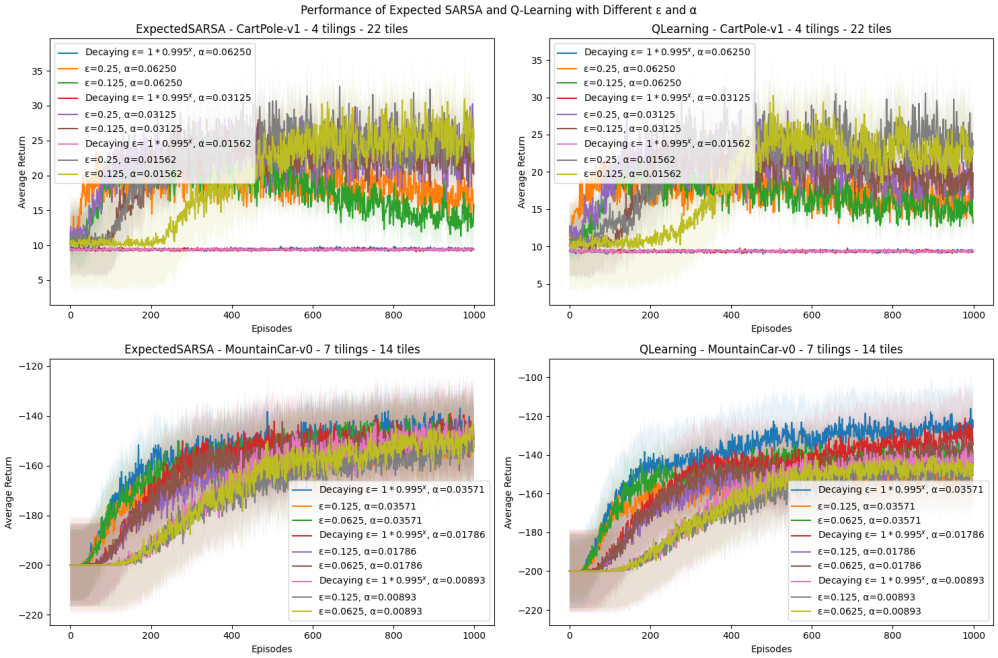

Figure 1: Mean reward per episode of ExpectedSARSA and Q-Learning on CartPole-v1 and MountainCar-v0

takeaways are limited because this is not based on a great performance, future experiments with more episodes, a different RL algorithm, or non-linear function approximation may be suitable to attempt to achieve better results.

For MountainCar-v0, both algorithms performed better, with obvious patterns of learning and stabilization, suggesting that linear function approximation is suitable. Out of the two algorithms, the Q-Learning algorithm performed better than ExpectedSARSA in average reward. This may imply that the environment is better suited for off-policy learning, potentially because the risk of negative outcomes in the environment is not high.

It is also worth noting that a decaying ϵ performed noticeably better than fixed ϵ for the specific environment + Q-Learning combination, suggesting that encouraging exploration in the early stages of learning may have led the model to establish a good understanding of the optimal actions and their rewards, and then exploit this knowledge through learning toward that optimal action via Q-Learning.

1.3.1 Differences in Environments

Apart from the algorithms and the hyperparameters, the two environments have shown some contrasting differences. Decaying ϵ did not work at all for CartPole, whereas it worked best in MountainCar. Cartpole performed best with a smaller learning rate, and MountainCar performed best on its largest learning rate. These differences may stem from the difference in complexity of the two environments; MountainCar is only a two-dimensional environment that might see advantages in learning more abruptly with large learning rates, Q-Learning, and large epsilons early, whereas CartPole is a four-dimensional environment that may require more careful and refined learning that

we were unable to achieve.

2 Policy Gradient Theorem

From the gradient policy theorem (with $\gamma = 1$), we have

$$\begin{aligned}\nabla_z J(\pi) &= \mathbb{E}_\pi \left[\nabla_z \log \pi_z(s,a) \cdot A^\pi(s,a)\right] \\ &= \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \nabla_z \log \pi_z(s_t,a_t) \cdot A^\pi(s_t,a_t)\right]\end{aligned}$$

Since $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s,a)$, and we choose $V^\pi(s,a)$ to be the baseline.
We begin with the gradient of the log policy function; consider some arbitrary $s' \in \mathcal{S}, a' \in \mathcal{A}$:

$$\begin{aligned}\frac{\partial \log \pi(s,a)}{\partial z(s',a')} &= \nabla_{z(s',a')} \left[(z(s,a) - \log \sum_{a' \in A} \exp(z(s,a')))\right] \\ &= \nabla_{z(s',a')}(z(s,a) - \nabla_z \log \sum_{a' \in A} \exp(z(s,a'))) \\ &= \nabla_{z(s',a')}(z(s,a) - \nabla_z \log \sum_{a' \in A} \exp(z(s,a'))) \\ &= \mathbb{I}(s'=s)\left[\mathbb{I}(a'=a) - \frac{\exp z(s,a)}{\sum_{a' \in \mathcal{A}} \exp z(s,a')}\right] \\ &= \mathbb{I}(s'=s)\left[\mathbb{I}(a'=a) - \pi(s,a)\right]\end{aligned}$$

We then use this in the policy gradient theorem:

$$\begin{aligned}\nabla_z J(\pi) &= \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \mathbb{I}[s_t = s]\left(\mathbb{I}[a_t = a]A^\pi(s,a) - \pi_\theta(a|s)A^\pi(s_t,a_t)\right)\right] \\ &= \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \mathbb{I}[s_t, a_t = s, a]A^\pi(s,a)\right] - \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \mathbb{I}[s_t = s]A^\pi(s_t,a_t)\right] \\ &= \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \mathbb{I}[s_t, a_t = s, a]A^\pi(s,a)\right] - \pi_\theta(a|s)\sum_{t=0}^{\infty} \mathbb{E}_\pi[\mathbb{I}[s_t = s]A^\pi(s_t,a_t)] \\ &\quad \text{because the expected advantage is 0} \\ &= \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \mathbb{I}[s_t, a_t = s, a]A^\pi(s,a)\right] - 0 \\ &= \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \mathbb{I}[s_t, a_t = s, a]A^\pi(s,a)\right] \\ &= \mathbb{E}_{s' \sim d^\pi}\mathbb{E}_{a' \sim \pi(\cdot|s)} \left[\sum_{t=0}^{\infty} \mathbb{I}[s_t, a_t = s, a]A^\pi(s,a)\right] \\ &= d^\pi(s)\pi(a|s)A^\pi(s,a) \qquad \square\end{aligned}$$

3 Policy-based methods with linear function approximation

3.1 Methodologies

Experimental methodologies were similar to Question 1, but we selected a single learning rate per algorithm and environment. For each algorithm (REINFORCE with baseline and ActorCritic) and environment, we experiment with a static temperature and an exponentially decreasing temperature.

Learning rates for this question are not divided by the number of tilings, since we were not required to use a set of fixed learning rates.

3.2 Results

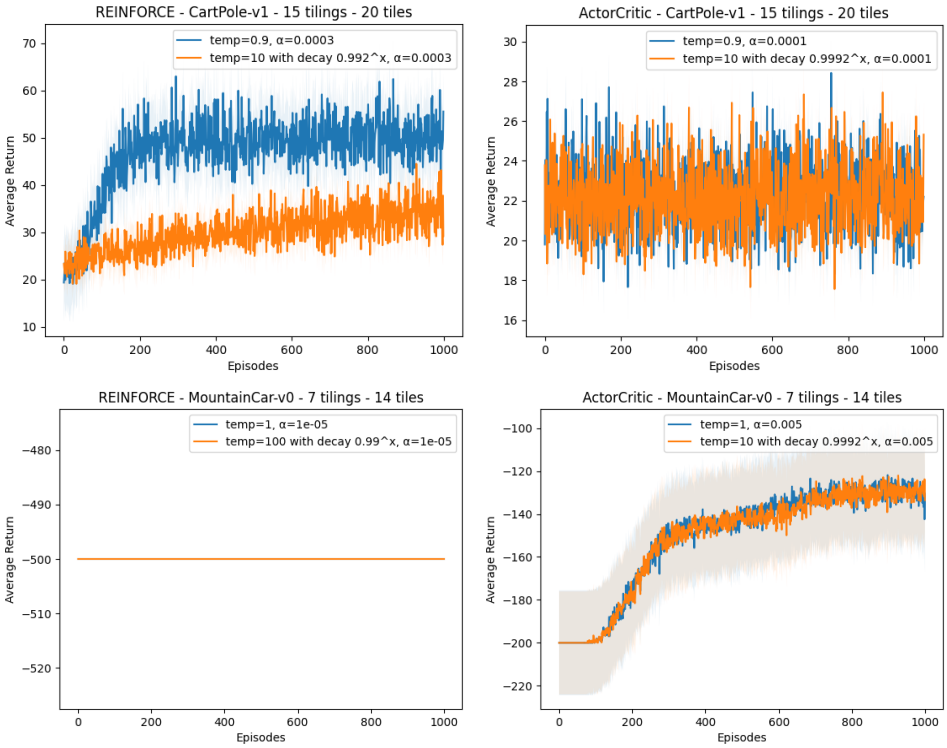

Figure 2: Mean reward per episode of REINFORCE and ActorCritic on CartPole-v1 and MountainCar-v0

Figure 2 showcases the results of the experiments with policy-based methods.

REINFORCE performed better than ActorCritic on CartPole-v1, achieving a mean reward of ≈ 50, surpassing value-based methods from Q1 as well. ActorCritic struggled to learn, achieving a mean reward of ≈ 25, similar to the value-based methods.

REINFORCE was entirely unable to learn MountainCar-v0, even with an adjusted truncation step limit, achieving a mean reward of ≈ -500. ActorCritic performed better, achieving similar results as the best-performing value-based method, with a mean reward of ≈ -130.

3.3 Discussion

Overall, policy-based methods did not perform significantly better than value-based methods, with a minor exception on CartPole-v1 and REINFORCE showing a slightly higher mean reward than all other methods.

We highlight a few points of the environments observed in the results: notably, the non-linearity of Cartpole-v1 and the difficulty in learning the Mountaincar-v0 environment with Monte-Carlo variants.

3.3.1 Non-Linearity of CartPole-v1

Through the results in this section and Question 1, we observe that linear approximation for the CartPole-v1 environment has performed poorly all across the board, whether the algorithm is value-based or policy-based. At best, we have been able to achieve some learning in the early episodes using a few algorithms, but all algorithms stabilized at sub-100 rewards, which is considered low given that the problem is considered solved at reward=190[1]. This is likely an indication that the environment is too complex for a linear approximation, particularly on states indicating a later part of the game.

One useful observation is that the REINFORCE algorithm performed marginally better than Actor-Critic, and methods from Q1 in the CartPole-v1 environment. This performance edge could likely be attributed to REINFORCE's strategy of learning from complete trajectories, allowing it to accumulate comprehensive experience from each episode. This method of learning enables a more nuanced understanding of the cause-and-effect relationship between actions and their long-term outcomes.

With these in mind, we conclude that linear approximation is sub-optimal for the CartPole-v1 environment, but algorithms of Monte-Carlo variants may be useful for general learning. Future experiments should explore non-linear function approximation, perhaps with a focus on reinforcement.

3.3.2 Monte-Carlo Methods and MountainCar-v0

We notice a severely poor performance of the Monte-Carlo method on MountainCar-v0, despite achievements of decent rewards in the other methods of similar linear approximation. The method required significantly more tuning and fine adjustments based on the nature of the environment, such as a specific decreasing temperature algorithm.

This is unsurprising, notice that the environment's reward system states that for each step that the car is not at the flag, the environment returns a reward of -1. Given that Monte Carlo methods learned through an entire trajectory, a truncated episode would result in a reward that provides little to no information about a good path to reach the terminal state, as the algorithm learns from the **cumulative reward of completed episodes**. Intuitively, a decreasing temperature would be helpful for the model to reach the terminal state, and then learn from early success runs. But in practice, this has been extremely difficult to achieve, and the model has failed to learn a policy to reach the terminal state.

Overall, with the positive results on other algorithms and the extra attention needed to tune the REINFORCE algorithm, we conclude that Monte-Carlo variations are not desired for the MountainCar-v0 environment. If we were to continue with the REINFORCE algorithm, experimenting with larger computations, notably with higher episodes and higher steps before truncation in order to successfully capture a good amount of successful trajectories in the early stages would be helpful.

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[2] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023.