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 with in two environments from the Gymnasium, an open-source fork of the OpenAI Gym. [1] During the experiment, the environment's states are discretized through tile-coding into a binary vector.

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, the values of α are written post-division by the number of tilings.

we observe that linear approximation in general learns CartPool-v1 poorly, and stops learning early on, achieving rewards of sub-30. This is the case for both Algorithms and a wide range of hyperparameter combinations. 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 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 takeaways are limited because this is not based on

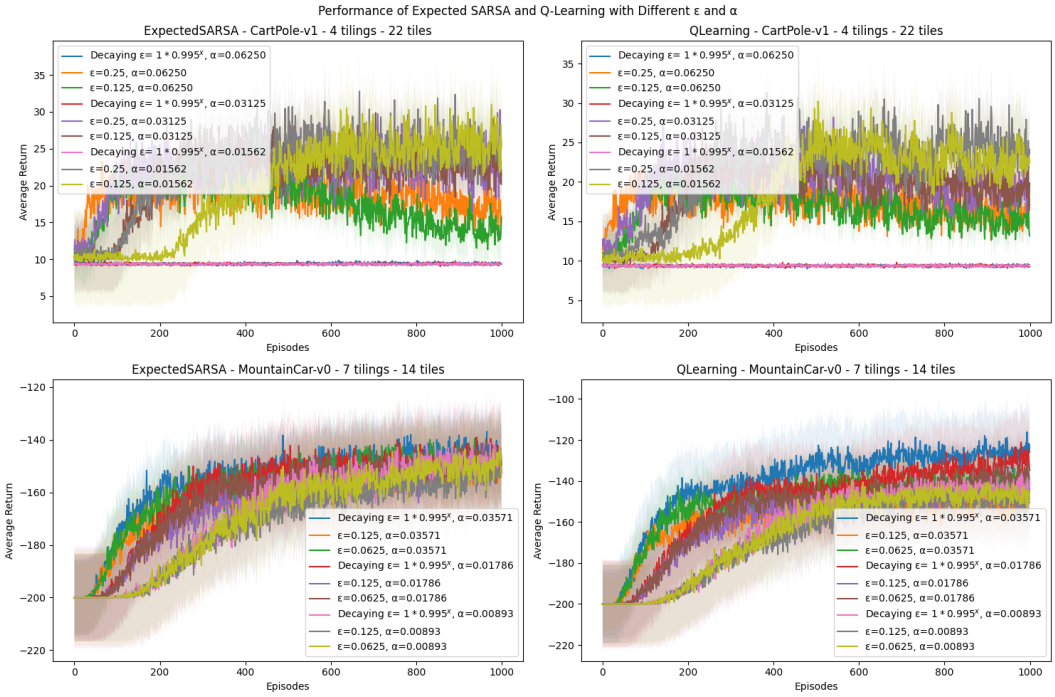

Figure 1: ExpectedSARSA and Q-Learning's Performances on CartPole-v1 and MountainCar-v0

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 towards that opmtimal 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 as well. 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 abrubtly 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

The gradient policy theorem states that

$$\nabla_\theta J(\theta) = E_\pi \left[\nabla_\theta \log \pi_\theta(s, a) \cdot G_t\right]$$

3 Policy-based methods with linear function approximation

3.1 Results

3.2 Discussion

3.2.1 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. At best, we have been able to do some learning in the early episodes using a few algorithms, but all stabilized sub-100 rewards. This is a likely indication that the environment is too complex for a linear approximation, particularly on states at a later stage of the game.

With these in mind, we conclude that linear approximation is not sub-optimal for the CartPole-v1 environment. Future experiments with non-linear function approximation may be suitable to attempt to achieve better results.

3.2.2 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 state 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 provide little to no information about a good path to reach the terminal state. 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, we would need to experiment with larger compute, notably with higher episodes and higher steps before truncation to successfully capture a good amount of successful trajectories in the early stages.

References

[1] 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.