CS 5033 - RL Project Checkpoint

Airi Shimamura Khoi Trinh

1. Introduction

For our RL project this semester, we want to build an RL agent that can easily beat the CartPole game.

In this checkpoint document, we will provide details and updates on our current experiments, as well as mention any difficulties we encountered, and list any future work to be done. A slight change from the proposal, Airi is implementing Q-Learning, while Khoi will be implementing SARSA learning instead of TD-Learning.

2. Hypothesis

Currently, we are testing one hypothesis for this project: We expect both the Q-Learning and SARSA algorithms to be able to finish training after 1000 episodes and that the SARSA algorithm will have the higher average rewards over 100 episodes.

Both Airi and Khoi's experiments will be performed in accordance to this hypothesis.

3. Experiments Done

3.1. Airi's Progress - Q-Learning

Q-learning is a reinforcement learning algorithm that enables an agent to learn an optimal policy by observing and updating the estimated value of state-action pairs. The agent selects an action and observes the next state and reward. Q-Learning is a very similar algorithm to SARSA learning. However, Q-learning uses an off-policy learning approach, updating the Q-value function using the maximum expected future reward

For Q-Learning, she assumed that when the number of episodes gets close to 1000, the pole is balanced upright by the agent choosing to move the cart left or right, while making sure the cart's center not disappear from the screen. The environment is set up as mentioned in the proposal, and in this case, $\gamma = 0.9$, $\alpha = 0.5$, and $\epsilon = 0.5$ for the parameters. After running the algorithm several times, the plot of average reward and total steps during episodes shows that it reached over 50 for the average reward and over 200 for total steps a few times, but both of them are not close enough to the goal. Also, the pole is being balanced better when the number of episodes increase, but the cart still shifts right to left and moves away from the center position.

For one run of 1000 episodes, here are her current results. Note that due to the random nature of taking actions, each run will produce a different graph.

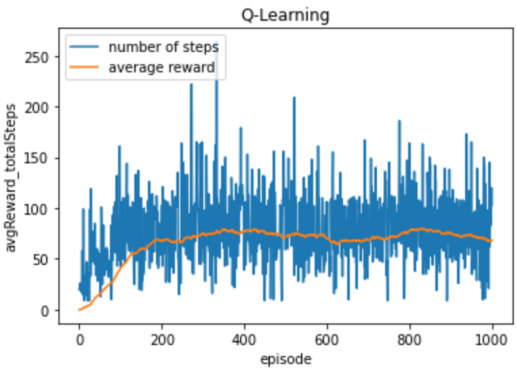

Figure 1. Q-Learning Results Over 1000 Episodes

Looking at the graph, we can see that the average rewards reached a plateau of approximately 55 after about 200 episodes, this shows that the agent is indeed being trained to take more optimal actions as the number of episodes increased.

3.2. Khoi's Progress - SARSA Learning

The SARSA algorithm is a popular reinforcement learning algorithm. At each time step, the agent selects an action according to its policy, observes the next state and reward, and updates the estimated value of the current state-action pair using a learning rate and discount factor. The algorithm maintains a Q-value function that estimates the expected future rewards for each state-action pair. The agent selects an action using an exploration-exploitation strategy and takes the selected action in the environment.

To this end, Khoi is trying to implement an ϵ greedy method for this SARSA algorithm. He also assumed that the agent will successfuly be trained in 1000 episodes. For his setup, the hyperparameters are: $\epsilon = 0.5$; $\gamma = 0.9$; and $\alpha = 0.5$

For the criteria related to the CartPole environment, the

agent will play the game for a set number of episodes, and in each episodes, the last 100 steps will have their rewards recorded (if the number of steps is less than 100, then the average will be over however many steps was taken for that episode) in order to generate an average. If any of the failing conditions is met, a penalty of -10 is given, otherwise, a reward of +1 is given. In the plot below, notice that a few episodes have taken around 300 steps, and in general, the average reward stays around the 50 mark.

For one run of 1000 episodes, here are his current results. Note that due to the random nature of taking actions, each run will produce a different graph.

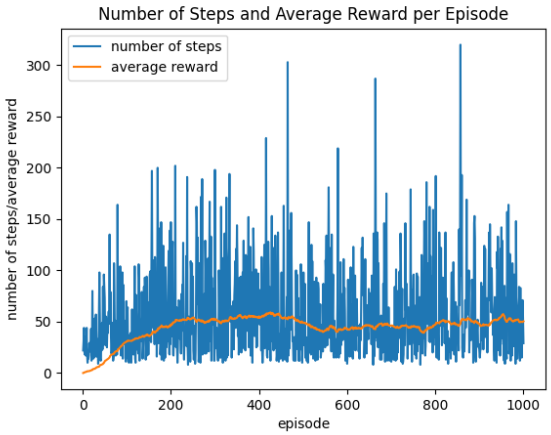

Figure 2. SARSA Results Over 1000 Episodes

Looking at the graph, we can see that the average rewards reached a plateau of approximately 50 after about 210 to 220 episodes, this shows that the agent is indeed being trained to take more optimal actions as the number of episodes increased. However, for this run, the reward seems to diverge after 410 episodes, and the average drops down to roughly 48, and remains around that mark for the rest of the episodes. This could be attributed to the fact that SARSA is using the Q value of the next state, action pair; while Q-Learning uses the maximum expected Q value instead. If the number of episodes were to be increased, this divergence could be amended.

4. General Analysis

The hypothesis was that the algorithms should be able to finish training after 1000 episodes and that SARSA would perform better, but this was not the case. We think that the reason the training didn't finish is that the number of episodes as well as the hyperparameters weren't optimal. So in order to rectify this, we plan on increasing the number of total episodes. In addtion, we are planning to do sensitivity analysis with the hyperparameters and will compare SARSA and Q-Learning further. In particular, we think the performance might increase if the value of epsilon is higher that allows the agent to take more random actions and explore the action space more effectively.

5. Difficulties Encountered

One of the main difficulties we have with this project is figuring out a good way to discretize the state space of the environment. As such, this hindered our abilities to perform more experiements in a timely manner. Fortunately, we came across the **digitize()** function from the **numpy** package that does the job quite well.

Another issue that we are facing, due to the random nature of picking an action, no two runs are the same. Therefore, conducting some sort of sensitivity analysis for our hyperparameters has proven to be difficult. We have tried to solve this by setting a seed number at the beginning of our script, but that does not seem to be an effective solution yet.

Addiontally, for both algorithms, the result is unstable. Sometimes the plot showed that the average reward converged after 200 episodes, and sometimes the result got better then worsen again as the episode number increased. This can be seen in the graph of Khoi's SARSA run. Therefore, it was hard to see if the agent was trained well or not.

6. Future Work

Our main goal in the next week or so is to finish the training for both algorithms, and then find a way to save it to memory, making it easily recallable in a fresh envinronment.

Secondly, as mentioned above, we will be doing sensitivity analysis of the hyperparameters; namely α, γ, ϵ and see how it would affect our algorithms.

Third, for Khoi's SARSA algorithms, he wants to explore using SARSA(Λ) and compare it to the current SARSA run.

Fourth, for both algorithms, we will explore different methods for ϵ decay. Currently, ϵ is kept at a constant 0.1 for both algorithm. Exploring ϵ decay (such as exponential decay) will provide some useful insights for the comparison of the algorithms.

Finally, we want to refine our training to fit our hypothesis of being able to finish training in 1000 episodes and make SARSA outperform Q-Learning. Addtionally, after we have successfuly train the agent, we want to be able to render a video of the game being played; and save the video for presentation purposes.