RL Project Checkpoint

Airi Shimamura Khoi Trinh

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

CartPole v1 is a game environment provided in the Gym package from OpenAI. In this environment, there is a pole, attached to a cart (hence the name CartPole). The cart moves along a frictionless track. Force is applied in the left and right direction of the cart. The goal of the game is to keep the pole upright for as long as possible. For each step taken, a +1 reward is given, including the termination step. The maximum points achievable in the game is 475. There are a few conditions that, if met, will end the game.

In this checkpoint document, we will provide details and update on our current experiments. A slight change from the proposal, Airi are implmeneting Q-Learning, while Khoi will be implementing SARSA learning instead.

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 get close to 1000, the pole is balanced upright by wiggling the cart and not disappear from the screen. The environment is set up as mentioned in the proposal, and let gamma = 0.9, alpha = 0.5, and epsilon = 0.5 for the parameters then run the Q-learning algorithm several times and analyze performances of the agent by comparing plots and checking behavior of the agent. After running the algorithm several times, the plot of average reward and total steps during episodes shows that it reached over 150 for the average reward and over 400 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.

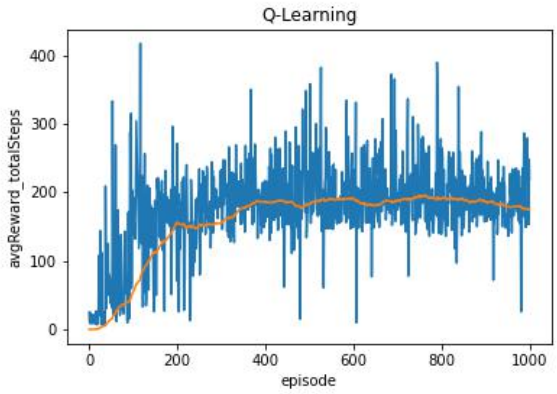

Figure 1. Q Learning Results Over 1000 Episodes

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: $\epsilon = 0.1$; $\gamma = 0.9$; 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, 100 steps will have their rewards recorded, in order to generate an average. If the failing conditions is met, a penalty of -10 is given, otherwise, a reward of +1 is given.

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.

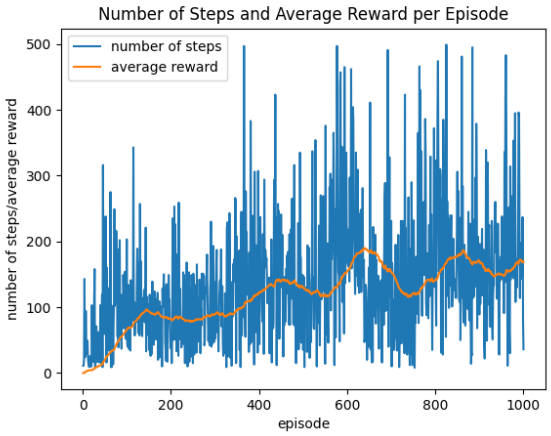

Figure 2. SARSA Results Over 1000 Episodes

General Analysis

The hypothesis was that the algorithms should be able to finished training after 1000 episodes, but this was not the case. The reason that it failed to complete the training is that the agent was not trained enough, so in order to rectify this, we plan on increasing the number of total episodes until the training is completed. Also, the learning rate and the value of epsilon should be tested with different numbers since they are other elements being affect performance of the agent. Especially, the performance might be better if the value of epsilon is increased since a larger number of epsilon allows the agent to act more randomly.

Difficulties Encounterd

One of the main difficulties we have with this project is figuring out a good way to discretize the state space of the environment. Fortunately, we came across the **digitize()** function from the **numpy** package that seems to do the job quite well.

Another issue that we are facing, is due to the random nature of picking an action, no two runs are the same, so conducting some sort of sensitivity analysis for our hyperparameters have 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 the 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.

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, to be able to recall it easily in a fresh environment.