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

For Airi's progress, she was able to create a code that successfully play the game using a Q-learning algorithm.

Khoi

For Khoi's progress, he is in the process of implmeneting a SARSA training algorithm. 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. 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 850 episodes and 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.

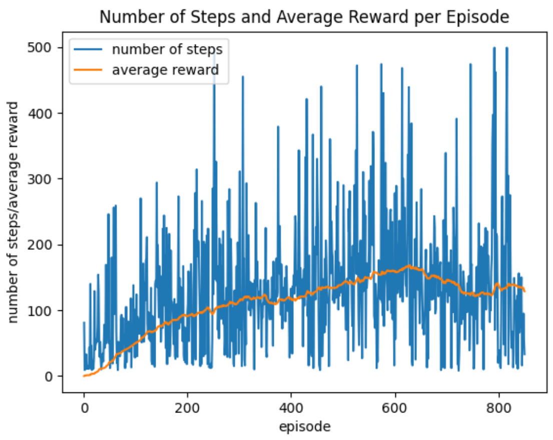

Figure 1. SARSA Results Over 850 Episodes

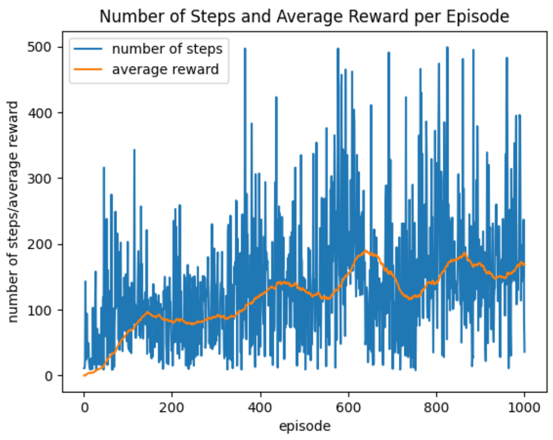

Figure 2. SARSA Results Over 1000 Episodes

Difficulties

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

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.