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. Literature Review

2.1. Reinforcement Learning

The goal of reinforcement learning (RL) is not simply to play a game. Rather, the driving force is to create systems that can be adaptive in the real world (Arulkumaran et al., 2017).

The essence of RL is learning through interaction and reward-driven behavior. The RL agent interacts with the environment and the results of its actions; it can then adjust its behavior in response to the reward(s) received (Arulkumaran et al., 2017)

This trial-and-error behavior can be considered the root of RL.

2.2. SARSA

2.3. Q-Learning

3. Hypothesis

Hypothesis 1: We epxect the Q Learning algorithm to perform better overall than the SARSA algorithm over 10000 episodes.

Hypothesis 2: For both algorithms, we expect exponential decay of ϵ to perform the best, as in, have the highest average reward over 10000 episodes.

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

4. Experiments Done

4.1. 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.

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.

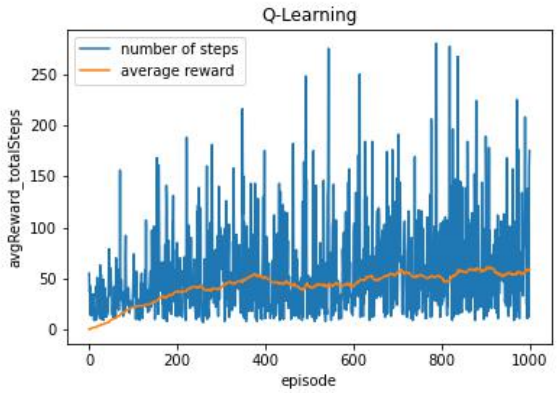

Figure 1. Q-Learning Results Over 1000 Episodes

Looking at the graph, we can see that the average rewards reached a plateau of approximately 50 after about 200 episodes, this shows that the agent is indeed being trained to take more optimal actions as the number of episodes increased.

4.2. 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 successfully 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 episode, 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.

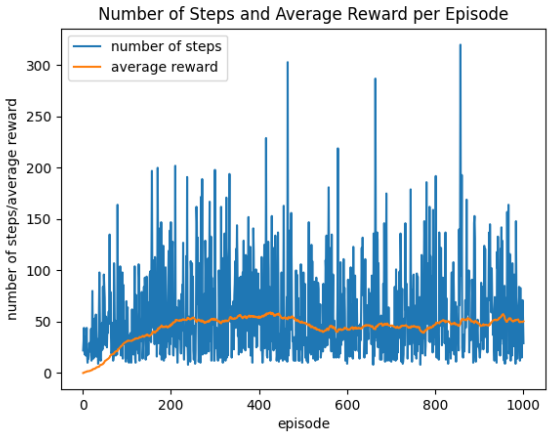

Figure 2. SARSA Results Over 1000 Episodes

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.

5. 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. Looking at the provided graph, we aren't able to conclude if one algorithm is better than the other. 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 addition, 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.

6. 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 experiments 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.

Additionally, for both algorithms, the result is unstable. For some runs, the rewards converged to an average of 50-55 after 200 episodes, and remains stable. However, divergence is encountered as well. 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.

7. Comparison to Other Methods

8. Future Work

Reference