IE7860: Intelligent Analytics

Assignment 9 : Reinforcement Learning

# Taxi Problem

This task was introduced in [Dietterich2000] to illustrate some issues in hierarchical reinforcement learning. There are 4 locations (labeled by different letters) and the job is to pick up the passenger at one location and drop them off at another. The agent receives +20 points for a successful dropoff, and loses 1 point for every timestep it takes. There is also a 10 point penalty for illegal pick-up and drop-off actions.

A screenshot of a cell phone

Description automatically generated

# Q-Learning

Qlearning is an off policy learning python implementation.This is a python implementation of the qlearning algorithm in the Sutton and Barto's book on RL. It's called SARSA because - (state, action, reward, state,action). The only difference between SARSA and Qlearning is that SARSA takes the next action based on the current policy while qlearning takes the action with maximum utility of next state.

A screenshot of a cell phone

Description automatically generated

# Methodology

Q learning is rather easy to implement due to the availability of open AI. Where we have the ability to create any sort of environment and assign conditions and task for the agents to execute. .There are also a certain amount of predefined environments which can be used to implement agents of various methodology. I choose the taxi V3 environment built into open AI which is simply a problem of transport where there are four points defined on the grid and the objective of the agent is to take a passenger off from one location and then drop them off at another location. There is a point system in regards to this which involves a positive reinforcement for moving in the right direction by either picking up or dropping off a passenger and there is a negative reinforcement for every time steps it takes for the agent to reach the destination. We begin by defining all the parameters required for Q learning which include the alpha, gamma, epsilon and the number of episodes. Each of these constants are used in the formula given above. Then we go ahead and define the function for it and move one by one to each episode. .While the total reward is calculated for each of these episodes and stored in the list along with the actions taken by the agent. After which we define the test case for testing the learning by the model. Here we use the reward and episode list to guide or agent over each step, and the steps which result in a positive reinforcement are outputted.

# Results

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A close up of a logo

Description automatically generatedA screenshot of a cell phone

Description automatically generated

The above list of outputs moves from left to right and downwards in progression. Where we can see it takes 133 steps to make the first correct move but once it has converged we can see that the number of steps it takes the agent to move through each correct action is reduced and even the return journey is rather optimized hence the reward is higher for this test case. The starting point was at the first slot and the first pick up point was at B while the drop-off was at G. the pickup is completed at state 477 and it takes 97 less states for it to drop off at G. the benefit with such a test case and environment set up is that with more and more test cases being run for the same environment the agent becomes more and more intelligent and always improve its performance.

A screenshot of a social media post

Description automatically generated

From the image above we can see that the reward is plotted with each .episode of training where the agent is left free to traverse the environment so as to increase its reward by gaining more positive reinforcement and reducing the number of times steps it takes. The training converges relatedly fast at around 2000 episodes without further improvement due to the simplicity of the reward criteria. If the positive reinforcement and the negative reinforcement criteria was to be changed by adding a few real-world constraints such as the distance along with limited movement space from each cell this would add to the reward scheme and might develop a better or faster approach. It can also lead to a negative consequence, since the supervision from the user is only based on the reward criteria and the constraints placed on it, there is no actual control over the learning of the agent.

# Conclusion

In conclusion I would say that this methodology of reinforcement learning is a relatively fast one and the whole objective of continuous learning is what is driving the AI in the current scenario. There are a lot of practical applications to it as well including scheduling problems and a lot of optimization problems along with simulation problems where decision-making is critical.