An-Najah National University

Faculty of Engineering and Information Technology

Computer Science Apprenticeship Program
Artificial Intelligence Track

Introduction to Artificial Intelligence

Reinforcement Learning

Instructor Name: Dr. Bahaa Thabet

Student Name: Yousef Salah Khodroj

Registration Number: 12113351

Reinforcement Learning (gym) package

In this project, we solved the Taxi game problem via using gym package, which is a python package that helps developers to build models and train these models using Reinforcement Learning Algorithms, by providing ready made environments such as Taxi environment, or letting them build their own environments using python programming language.

In this document, I am discussing training results for the Taxi environment, while I am training a new model to find the optimal policy for paths, to take passengers off from different places, and drop them off in their desired destination, all below experiments having the following hyperparameters:

Learning rate = 0.1

Experiment 1:

In this experiment, I am training model 1000 times (1000 episodes), and I write down some results of This experiment, logically in the first episode where we trying to train new model result might be bad, because agent (the Taxi) doesn't know anything about the environment, so it performs random actions a lot, makes mistakes a lot, lose rewards a lot, until it achieve his goal (terminal state), which is to drop pickled passengers off in their desired destination.

Epochs 1000	Rewards	Steps
Epoch 0	-1646	704
Epoch 100	-225	174
Epoch 200	-180	156
Epoch 300	-202	178
Epoch 400	0	21
Epoch 500	-90	84
Epoch 600	-92	77
Epoch 700	14	7
Epoch 800	-119	95
Epoch 900	-17	29

Epoch 1000	-15	27

Experiment 2:

In this experiment, we trained a model little bit more than the previous experiment, we trained the model 5000 times (5000 epochs), there is a significantly enhancement in model rewards and taken steps, you can see observations in the below table, which takes one episode results for each 500 epochs, the last episode wasn't a good one, because our agent is still learning from environment, and the exploration rate = 0.1, which might be quite small to let agent explore new paths, because 90% of times it will follow what he learned before.

Epochs 5000	Rewards	Steps
Epoch 0	-1391	593
Epoch 500	-102	114
Epoch 1000	-13	34
Epoch 1500	8	13
Epoch 2000	7	14
Epoch 2500	6	15
Epoch 3000	5	16
Epoch 3500	-14	17
Epoch 4000	-5	26
Epoch 4500	7	14
Epoch 5000	-10	22

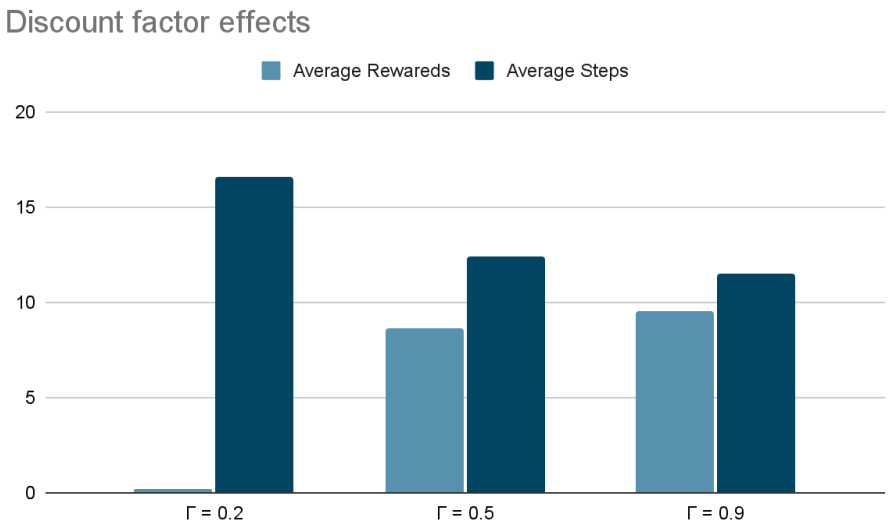
Experiment 3:

In this experiment, we trained our model on 10000 episodes, and the model now becomes much better, which is clear in the below table, where observed rewards become good and positive ones, and steps count is quite a bit smaller than before.

Epochs 10000	Rewards	Steps
Epoch 0	-1625	674
Epoch 1k	13	8
Epoch 2k	8	13
Epoch 3k	11	10
Epoch 4k	-5	17
Epoch 5k	-3	15
Epoch 6k	8	13
Epoch 7k	12	9
Epoch 8k	10	11
Epoch 9k	14	7
Epoch 10k	3	19

Experiment 4:

In this experiment, we trained our model 10000 epochs three times, where each time we change the discounted factor value as following (0.3, 0.5, 0.9) respectively, after training phase, we let our agent to play 10 times tha Taxi game, the below chart summarizes the average reward and average steps corresponding to its discount factor value.



Discount factor has a high effect on the rewards and steps counts, because higher discounting factors leads to bigger rewards count,

I want to prove it as a formula.

Assume we need three steps to go directly to the goal, and discount factors values as follows 0.1 and 0.9 respectively for games A and B, all states have no score reward except the last one (terminal one) have 100 score reward.

I want to use the Markov decision process and ignore other factors for simplicity.

Game A discount factor = 0.1:

Reward = 0 + (0 * 0.1) + (100 * 0.01) = 10 scores

Game A discount factor = 0.9:

Reward = 0 + (0 * 0.9) + (100 * 0.81) = 81 scores

Thus, higher discount factor value tends to higher rewards values for the same path with the same steps.

My opinion: to enhance this model to produce the optimal policy or almost optimal policy, you can initialize exploration value to be high in the initial episodes, and as trained your model as you decrease the exploration value, to let agent learn and explore a lot of states in the beginning, and starts using what he learned in the begging.