Report : Implementation of a domain and/or a reinforcement learning algorithm
ATCI - Advanced Topics in Computational Intelligence

Piotrowski Maciej

1 Introduction

This document is a report for the Implementation project of the "ATCI - Advanced Topics in Computational Intelligence" course. It consists of the implementation of a python program. More precisely, a Reinforcement Learning (RL) algorithm that has to learn the best behaviour for 2 different tasks from the OpenAI Gym environments. The tasks and algorithm used in this assignement were free of choice as long as it came from one of the proposed platforms and using a technique seen in class. This assignement is strongly inspired from the slides of the course, the TheComputerScientist's Youtube channel and that video which helped me a lot with the understanding of the different techniques and provided a lot of examples for the OpenAI Gym task's.

2 Selected tasks and algorithm

For this assignement, I decided to pick the Cartpole and MountainCar tasks from the OpenAI Gym. I decided to implement the Deep Q-Network with Experience Replay (using the python's Tensorflow library).

3 Results Explanation

3.1 Cart Pole

For my experiments, I decided to work with a neural network with a sequential layer with only one hidden layer containing 100 hidden units with a relu activation function and an output layer with a linear activation function. Then I've trained different agents playing with the different hyper-parameters:

- Buffer size = [2000,10000]
- Discount rate = [0.95,0.96,0.97,0.98,0.99,1]
- Batch size = [32,64]

I've trained and saved one agent with each possible parameters combination. Usually, whatever the parameters, the agent obtained the desired score of 500 after around 100 episodes. Then, I ran each of the saved agents 20 times and saved the score average for each agent. That's how I picked the best agent with the best hyper-parameters. (I'm aware that there is a bit of randomness in the training of the agents that can partially change the final results but running out of time and my algorithm being slow enough, I decided not to do it)

3.2 Mountain Car

For my experiments for Mountain Car, I've kept the same neural network as for the Cart Pole. Unfortunately, the Mountain Car game is a lot harder than the Cart Pole. That's because of the reward function. In Cart Pole, the longer we play the game, the higher the reward become and the reward increase after each step. In Mountain Car, on the other hand, the longer we play the game, the more the reward decrease and we cannot know how good the moves are until the end of the game. There is no immediate positive reward after each step. That's probably my agent struggled to reach good scores. And because that, barely every game lasted the entire number of steps so it took quite a long time. That's why I've decided to try it with only one parameters combination.

- Buffer size = 10000

- Discount rate = 0.99
- Batch size = 64

I also needed to change from 0.99 to 0.999 the multiplicative rate of the epsilon because it was decreasing way too fast. I ran it for 300 episodes and the agent managed only one time to reach the goal before reaching the maximum number of steps. Then, I tried it for 300 runs but unfortunately my agent couldn't reach the goal even 1 time. That's probably because I left not enough steps for the agent to train on. In my opinion, if I would increase the number of maximum training episodes, my agent could perform better but unfortunately it would take too much time for my pc to train or my algorithm is too slow.

4 Results Comparison

4.1 Cart Pole

memory	gamma	batch	Score avg.
2000	0.95	32	487.45
2000	0.95	64	287.85
2000	0.96	32	194.95
2000	0.96	64	388.85
2000	0.97	32	256.10
2000	0.97	64	117.05
2000	0.98	32	157.95
2000	0.98	64	180.15
2000	0.99	32	283.60
2000	0.99	64	383.10
2000	1	32	123.15
2000	1	64	257.80

memory	gamma	batch	Score avg.
10000	0.95	32	272.95
10000	0.95	64	361.30
10000	0.96	32	274.90
10000	0.96	64	347.15
10000	0.97	32	354.45
10000	0.97	64	170.35
10000	0.98	32	356.35
10000	0.98	64	323.00
10000	0.99	32	273.95
10000	0.99	64	349.20
10000	1	32	259.60
10000	1	64	451.45

In general, we can see that agents trained with bigger memory size and batch size of 64 are better than the ones with smaller memory size and batch size of 32. But there are, of course, some exceptions but it may be because of the pinch of randomness in the algorithm. Surprisingly, the agent who performed best is the one trained using a 2000 memory size, a gamma of 0.95 and a batch size of 32. It reaches an average score of 487.45 over 20 runs. For the results to be relevant we should use 3 different trained agents for each hyper-parameters combination and maybe test it on a larger number of test episodes.

5 Conclusions

This project allowed me to have a first grip on a practical assignment related to reinforcement learning. The whole project pushed me to do a lot of web search and that, in addition to the course material, allowed me a better understanding of different techniques explained during the course and to experience on my own the difficulty of designing an adequate agent training depending on the related task. In my opinion, I obtained quite satisfying result for the CartPole task but unfortunately I failed to adapt the algorithm enough to perform well on the Mountain Car task. To conclude, I'm very happy that I could discover yet another field of Machine Learning and be able to experience with it.