Machine Learning Project

Fabio Massimo Ercoli

July 2024

1 Introduction

We're presenting two prossible Q-learning implementation approaches:

- Tabular based. Implemented using a numpy dictonary.
- Neural network based. Also known as Deep Queue Network (DQN).

2 Tabular Q-Learning

2.1 Running the project

To trigger the learning process run from the *machine-learning-project* directory the following code:

```
python qlearn-app.py
```

You should see a message similar to the following:

```
observation space Discrete(500)
action space Discrete(6)
avg return before learning -92.98608555399927
avg return after learning 2.5552244875845997
```

The observation and the action space description is related to the *Gymnasium* environment we used to train and test our agent, that is the *Taxi-v3 environment*[1] that has 500 discrete possible states and 6 possible actions.

The *avg return* is the average of the score gained rolling out a series of episodes. In the case of this project 5 episodes are executed before the training and 20 after the training. The expectation is that the agent can get a better score after we properly trained it.

An episod starts from the initial state and ends in case of a termination state is reached or is trunkated since the **max episode steps** value is reached.

This is the first value we can change to tune the lerning, in general this value should be great enough to allow the agent to learn and to rollout correctly the

[1] $https://gymnasium.farama.org/environments/toy_text/taxi$

strategy it learned. We decide to increase it from 200 that is the default to 500 and this change seems to provided better performance to the learning.

How can we evaluate the goodneed of the learning? In this case we simpy compare the average score for an episode perfomed using a random strategy (before to learn) with the average score obtanied after the learn.

2.2 The score and the actions

The score is a crucial concept, since the learning activity is entirly oriented to maximaize this value.

The score is the summation of all the reward we get from the environment every time we execute an action (notice that can be negative or positive), multiplied by the **discount factor** γ.

The meaning of this value is to promote not only the rewards collected, but also the speed with which we get them. This is another value that we need to balance, since if it is too low, we can penalize the learning of long term goals.

So the score depends on the actions the agent chooses, those are randomly selected before the training and after the training they are

3 Deep Queue Network