Neural Networks and Deep Learning - Homework 3

Deep Reinforcement Learning

Davide Garbelotto

January 2020

1 Introduction

The goal of this experience is to practice the basics of the Deep Reinforcement Learning framework and explore some advanced methods. All the tasks are developed using environments of the Gym library by OpenAI. The first goal is to modify the Deep Reinforcement algorithm seen at the Laboratory, trying to solve the *Cart Pole* environment more efficiently. Then a more complex way to face the same environment is implemented: instead of using as the observation space the 4-dimensional space return by the environment, the pixels of the screen are directly used. With these settings, the hardest part is to extract the information on the state directly from the image seen while playing the game. Eventually, another environment, the *Mountain Car*, has been solved, using similar settings as the first task, but with a different strategy.

2 Cart Pole - Version 1

2.1 Introduction

The *Cart Pole* game is a simple environment where the objective is to keep the pole up, moving the cart to the right or to the left. It's simple because it has only two possible actions, and the observation space returned by the environment is of dimension 4. It indeed contains the position and the velocity of the cart and the angular position and velocity of the pole. From this information the agent should be able to learn which states are safe and which others are dangerous. This is done through reward feedback, which in this case assume only the value of 1 or 0: it's 1 if the pole is still up, whereas it's 0 in the loss case.

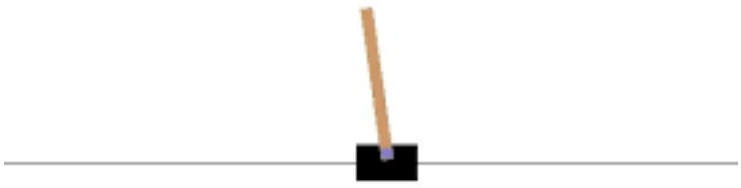


Figure 1: Cart Pole Environment

The deep neural network is employed to estimate the q-values, which are the starting point for the agent to decide which action to take. Indeed, the agent has also to follow an exploration profile, which is based on the trade-off between exploration and exploitation and is described in detail in the Methods section.

The game is won when the agent is able to keep the pole balanced for 500 consecutive steps. On the contrary, if the pole angle reaches a certain point the game is lost because whatever action the agent will take, it won't be able to recover the pole balance.

2.2 Methods

2.2.1 Network structure and settings

The network responsible for estimating the q-values is composed of three linear layers, the first two made of 128 nodes and the last one giving in output the values for the two possible actions. The activation function is the hyperbolic tangent *Tanh*. The optimizer is a standard Stochastic Gradient Descent.

2.2.2 Exploration Profile

The most interesting feature to tune is the exploration profile. Two different algorithms have been implemented: the epsilon greedy and the softmax. The first one is based on the parameter epsilon, which defines the probability for the agent to choose a random action. In the other case, with probability $1 - epsilon$, the agent to chooses the best action, i.e. the one with the highest q-value. The softmax algorithm, instead, takes in input the q-value of all the possible actions and estimates the probability to choose each of them through a softmax operation. Both of the algorithms depend on a parameter: in one case the epsilon, in the other the softmax temperature. In general, the training aims to let the agent explore the environment at the beginning, and once he has gained some experience, to let him exploit the best actions. To this purpose, the strategy adopted is to use a high temperature (or epsilon) value at the beginning of the training and to incrementally decrease it as the training goes on. In the case of epsilon, clearly, the parameter has to be in the probability range, whereas the temperature can exceed the value of 1. One of the ways to model this exploration profile is by exponentially decreasing the parameters. Other exploration profiles have been tested, such as the one represented in Figure 2a, made of two exponential decreasing with different intensities and starting time. Eventually, the simple exponential one, represented in Figure 2b, outperformed the others. Within the exponential approach, different shapes have been tried, tuning the parameter such as the initial value, the decreasing rate and the number of steps.

2.2.3 Advanced Methods

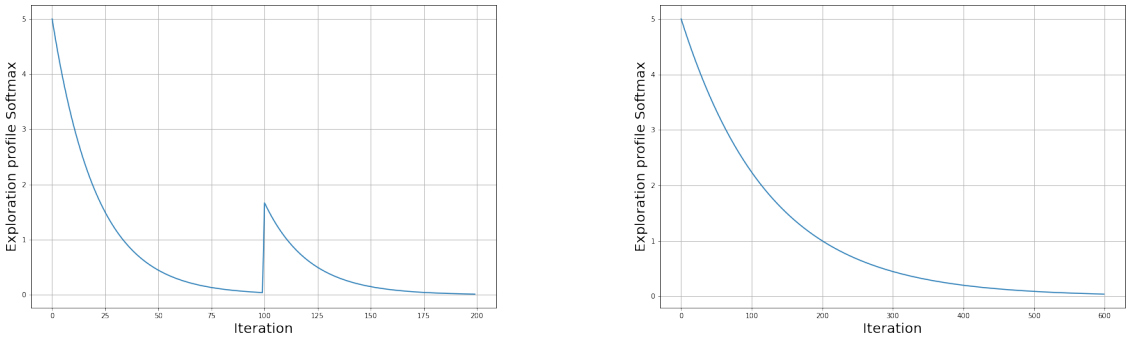
One important method used to reach good results in a limited amount of epochs has been to modify the reward function of the system. Indeed, in the beginning, the cart learnt to keep the pole up, but he experienced the problem of going out of the screen, which led to the loss of the game anyways. To solve this issue a penalization has been added to the reward function. The modification consists of subtracting a position term, extracted from the observation space, multiplied by a position parameter, which encodes the strength of the regularization. In this way, the agent learns that if he goes towards the two extremities of the screen he will get a lower reward, and therefore it's better for him to stay in the centre. A similar regularization has been applied to the pole angle because in an intermediate result, the cart was able to keep the pole for some time, but the latter was oscillating a lot, risking of overstepping the maximum angle and losing the game. The angle term in the reward function has brought to greater stability of the pole. Hence, the final reward function was defined as:

$$R = R_{old} - \lambda_{position}|position| - \lambda_{angle}|angle|$$

The two parameters $\lambda_{position}$ and λ_{angle} have been manually tuned to maximize the performance.

2.3 Results

To check the effectiveness of the training, the agent is tested on 10 episodes, but this time the temperature (or the epsilon) parameter is set to zero, such that the best action is always chosen. The results showed

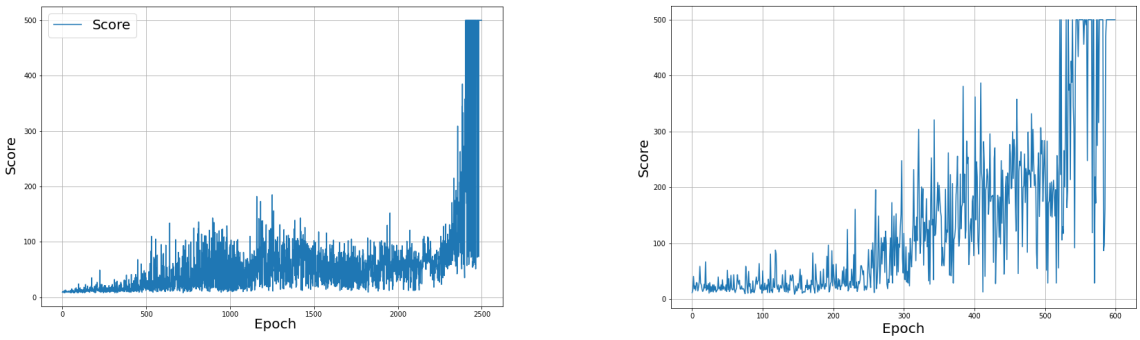


(a) Mixed

(b) Exponential

Figure 2: Exploration Profiles Softmax

that the softmax approach outperformed the epsilon greedy one, being able to learn how to win the game regularly, in a significantly smaller number of steps. The learning plots of the two cases are shown in Figure 3. It can be easily observed that in the softmax case, the agent is able to win the game in slightly more than 500 epochs, whereas in the epsilon greedy case, the training needs around 2400 epochs. Nonetheless, in both cases the agent is able to learn how to solve the game in a robust way, i.e. he always wins the game in the test episodes.



(a) Epsilon Greedy

(b) Softmax

Figure 3: Training plots

3 Cart Pole - Version 2

3.1 Introduction

This section studies the same *Cart Pole* environment, but the approach to the problem is different. Indeed, this time, the observation space is no more the 4-dimensional space, encoding velocity and position of the cart and the pole, but this information have to be extracted directly from the pixels of the screen. The input of the deep q-learning network is the screen image, made of the pixel values. To extract meaningful information from this kind of input, the most appropriate network is a CNN. Through the convolution operations performed by the filters, the network has to understand what is the current state of the cart pole, such that it can evaluate the best action to take.

It has to be taken into account that this observation space is significantly bigger than the one in the first task, and therefore it is harder for the network to extract the important information and to reach optimal performance.

3.2 Methods

3.2.1 Network structure and settings

The convolutional neural network is composed of three convolutional layers, made of respectively 16, 32 and 32 5x5 sized filters and a stride of 2. These are followed by a flattening and a linear layer which outputs the q-values for the two possible actions. To improve the stability Batch Normalization layers are interposed after each convolutional layer. Relu activation functions are used as non-linearities.

The optimizer, in this case, is a RMSprop, which guarantees a faster learning.

3.2.2 Advanced methods

The strategy implemented is to feed to the network the difference between two consecutive frames, the current one and the next one. In this way the information load is much lighter because between two adjacent time-steps, the pixel values are not likely to change a lot. Moreover, before computing this difference, the frames are resized such that only a small part of the screen, the one containing the cart pole, is represented in the pixels. In particular, the resize operation extracts a 40x90 pixel image, and it is processed through a cubic interpolation.

3.3 Results

The results are not as good as in the first simple case, indeed the agent is not able to reach the maximum points and win the game. Nevertheless, it shows relevant improvements in the performance during the training. As it is shown in the training plot of Figure 4, the agent is able to reach an average value of 171.95 among the 20 test episodes. The episode scores are reported in the bar plot of Figure 5.

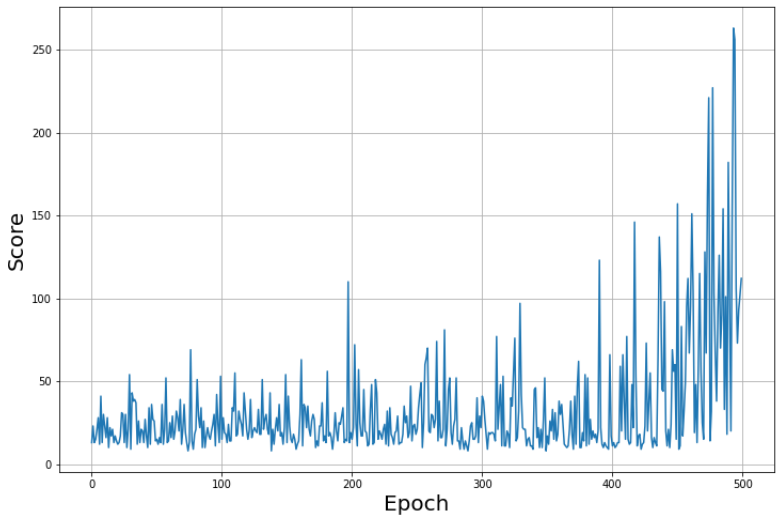


Figure 4: Training plot

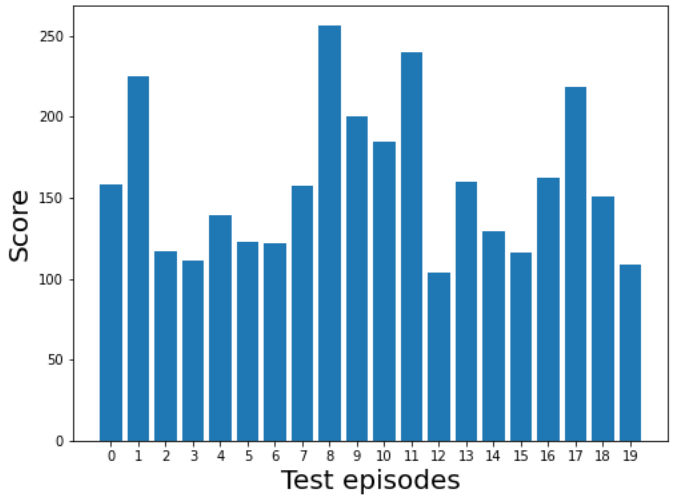


Figure 5: Test scores

4 Mountain Car

4.1 Introduction

In this last section the *Mountain Car* environment has been analysed and solved. As shown in Figure 6, the goal of the game is to let the car reach the top of the mountain on the right, where there is the yellow flag. The agent can only perform three different actions: going to the right, going to the left or staying still. The problem is calibrated such that the car has not enough power to reach directly the top of the mountain by itself, but it has to exploit the momentum gained by the steepness of the curves.

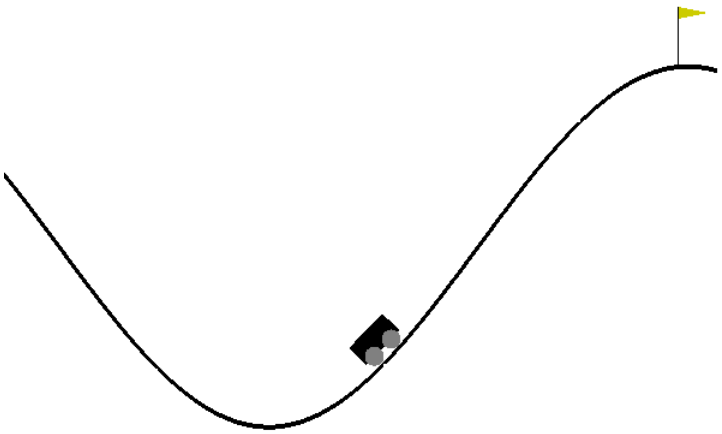


Figure 6: Mountain Car Environment

The strategy will be therefore to oscillate up and down until a certain height is reached, thus being able to go for the final sprint to the flag on the right hill. The game finishes if either the car reaches the flag or if the time limit of 200 expires. This means that, in order to win, the agent has to learn how to reach the top of the mountain in less than 200 time steps.

4.2 Methods

The network structure is quite similar to the one used in the first task. It is made of 3 fully connected layers, with respectively 400, 300 and 3 neurons, the last one being the output of the q-values for the three different actions. The activation functions are Tanh and the optimizer is the Stochastic Gradient Descent. The exploration profile is, as in the first task, an exponentially decreasing of the softmax temperature, whose parameters have been tuned on this specific problem.

The most challenging part of this environment is caused by the different reward system With respect to the previous one. Indeed, the reward is always equal to -1, except in the case the car reaches the flag, when it is equal to 0. The main problem is that the agent doesn't receive any positive feedback until he reaches the top of the hill and therefore struggles a lot to learn the correct strategy to win the game. To fix this issue the rewarding system has been slightly changed. The best way to keep track of the progress of the agent is to register the maximum position coordinate he's able to reach during each episode. Indeed, in order to win the game, he has to arrive at the maximum position coordinate, i.e. he has to reach the right extreme of the screen. To give to the agent continuous feedback, the reward has been re-defined as the maximum position reached during that episode. In this way the more the agent is able to push himself to the right of the screen, the higher is the reward he gets. He, therefore, learns quickly that he has to gain a moment to reach his goal.

Another alternative which has been tried was to maximize the velocity of the car, adding a velocity term to the reward function. Indeed, the cart learnt to gain velocity and reached more easily the top of the hill. Yet, the maximum position strategy produced better results.

4.2.1 Results

Thanks to the reward system implemented, the results are positive since the car is able to always reach the flag in the test episodes. From the training plot shown in Figure 7, it can be observed how the maximum position reached is oscillating around -0.4, i.e. the valley position, until epochs 1250, when the agent suddenly learns how to reach the flag position at 0.4. This is also due to the incremental decrease of the softmax temperature which regulates the exploration profile of the system.

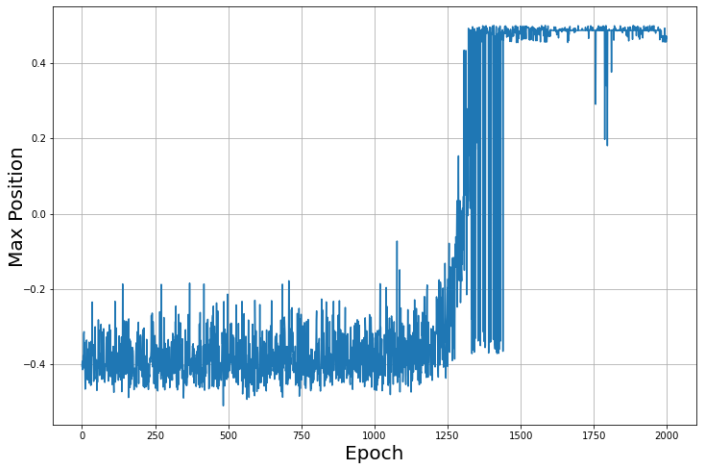


Figure 7: Maximum position monitoring during training