Homework 3 - Deep Reinforcement Learning

Michele Puppin ID 1227474 michele.puppin@studenti.unipd.it

(Dated: August 26, 2021)

I. INTRODUCTION

In this homework we will implement and test neural network models for solving reinforcement learning problems, where intelligent agents have to take actions in an environment in order to maximize the notion of cumulative reward. In particular we consider *Deep Q-learning* where a *Q-value function* determines how good a certain action is, given a state, for an agent following a policy. Since the computation of *Q-values* directly through value iterations is computationally demanding, we can use deep neural networks to estimate the optimal *Q-function*.

We will train the models in different virtual environments provided by *gym* library. First of all, we train a *Deep Q-learning* algorithm on the Cart-Pole environment, where the agent scope is to balance a pole standing on a cart by moving the cart itself. Secondly, we train an similar algorithm on the Cart-Pole environment but using screen pixels as input for the agent. Finally, we test the algorithm on a different environment, the Mountain-Car, where car has to climb a mountain.

These models are implemented in Python using the PyTorch framework.

II. RL FOR CART-POLE ENVIRONMENT

1. Methods

In Cart-Pole environment, a pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled with two possible actions: by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center. The state space of the system is described by the cart position, the cart velocity, the pole angle and the pole angular velocity.

As said, we implement a simple deep feed-forward neural network to approximate the *Q-function*. The structure is the following:

- an input layer with 4 neurons, one neuron for each dimension of the state space;
- two hidden layers with 128 neurons;
- an output layer with 2 neurons, one for each possible action.

We use hyperbolic tangent as activation function and we use Stochastic Gradient Descent (SGD) optimizer with no momentum. We use Smooth L1 loss function.

Moreover, we implement *replay memory*, where we store all of the agent's experiences at each time step over a certain number of episodes played by the agent. With a technique called *experience replay* we will randomly sample from this dataset to train the network.

We use *softmax policy* to probabilistically choose the action, by sampling it from a softmax distribution of the *Q-values* with a certain temperature τ. We use a exponentially decreasing exploration profile for the value of the temperature at the i-th episode as in:

$$\tau_i = \tau_0 \exp\left(-\frac{i\beta \log(\tau_0)}{N}\right)$$

where τ_0 is the initial value of the temperature and β can be tuned for optimal performances. Note that to avoid the cart exiting the screen we add a penalty proportional to how far the cart is from the center.

We tune the following hyperparameters in order to optimize the model performances:

- the parameter for temperature $\beta \in \{5, 10, 15\}$;
- the parameter for the long term reward $\gamma \in \{0.97, 0.98, 0.99\}$;
- the optimizer learning rate $\eta \in \{10^{-1}, 10^{-2}, 10^{-3}\}$.

A stopping procedure stops the training when the maximum score is reached for 10 consecutive steps. We implement a random grid search to find best hyperparameters.

2. Results

The score profile obtained for the parameters used in Lab notebook are shown inf Fig. 1a, the algorithm is stopped after around 850 iterations. Best hyperparameters are found to be:

- the parameter for temperature $\beta = 15$;
- the parameter for the long term reward $\gamma = 0.98$;
- the optimizer learning rate $\eta = 10^{-1}$.

The results obtained with the best hyperparameters are shown in Fig. 1b, the algorithm is stopped after around 485 iterations. We see that the tuning of the hyperparameters, in particular of the temperature of the exploration profile, can heavily affect the performance of the algorithm. A deeper search could provide better performing hyperparameters.

Finally, we test the algorithm by setting the temperature to zero and we observe that the model is able to reach the best score at every trial.

III. RL FOR CART-POLE ENVIRONMENT USING SCREEN PIXELS

1. Methods

Now we train the same algorithm presented before on the same environment. In the previous case we observe the system by analysing the state space of the system (cart position, cart velocity, pole angle and the pole angular velocity) automatically provided by a function of the environment (`step`). Now we neglect this information and we consider as state the image provided by the `render` function of the environment. We consider only one colour channel. The image obtained is properly cut and reshaped to reduce the number of pixel and favour computational speed. Since we are dealing with images, we now substitute the fully-connected feed-forward neural network used in the previous section with a convolutional neural network.

The architecture is the following:

- Convolutional part:
  - a first convolutional layer with one input channel (the image is in gray scale) and 64 output channels, we use a 4×4 filter that slides over the image with a stride of 2 and a padding of 0;
  - a second convolutional layer with 64 input channels and 64 output channels, we use a 4×4 kernel with a stride of 2 and a padding of 0;
  - a third convolutional layer with 64 input channels and 32 output channels, we use a 3×3 kernel with a stride of 1 and a padding of 0;

- Linear part:
  - a first linear layer with an input size of depending on image dimension and an output size of 128;
  - a second linear layer with an input size of 128 and an output size which is equal to the action space dimension.

We use ReLU activation function for the convolutional part, while we use the hyperbolic tangent for the linear part.
The model is trained with the best hyperparameters found in the previous section.

2. Results

The results obtained are shown in Fig. 2a. The model is no able to reach the maximum score, best performances are achieved with a score around 70. We see that considering a static image does not provide sufficient information to the learner. If cart position and pole angle can be easily inferred by the network, cart velocity and pole angular velocity information are totally missing.

Therefore, we try to improve the model by passing to the convolutional neural network a sequence of 4 images. The CNN in his case takes as input 4 different channels. Now the time dimension can be considered by the network which will try to infer the dynamics from the sequence. This implementation require just a little adjustment of the code in order to stack together and then handle the four images.

In Fig. 2b we see observe the profile score for 300 epochs, which was the maximum number of epochs available for training given the limited amount of computational resources. It is not possible to determine if some learning is happening, a further analysis should be performed.

IV. RL FOR MOUNTAIN-CAR ENVIRONMENT

1. Methods

In the Mountain-Car environment, a car is on a one-dimensional track, positioned between two mountains. The goal is to drive up the mountain on the right; however, the car's engine is not strong enough to scale the mountain in a single pass. Therefore, the only way to succeed is to drive back and forth to build up momentum. Three actions are possible: accelerate to the right, accelerate to the right or apply a null acceleration. The state space of the system is described by the car position and the car velocity.

The goal is to keep the car moving, so we reward the agent with +1 whenever it is accelerating in the same direction of the velocity, otherwise we give a -1 reword.

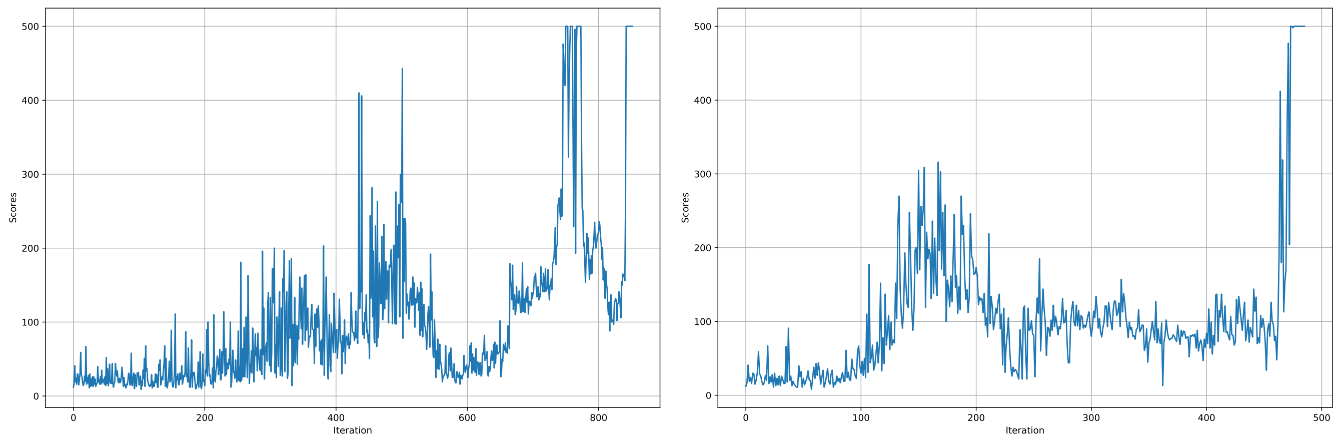
We use the same model used for the Cart-Pole environment with the same tuned hyperparameters and exploration profile. The number of input and output neurons of the neural network are set accordingly to the new environment.

2. Results

The results obtained are shown in Fig. 3, the algorithm is stopped after around 105 iterations.

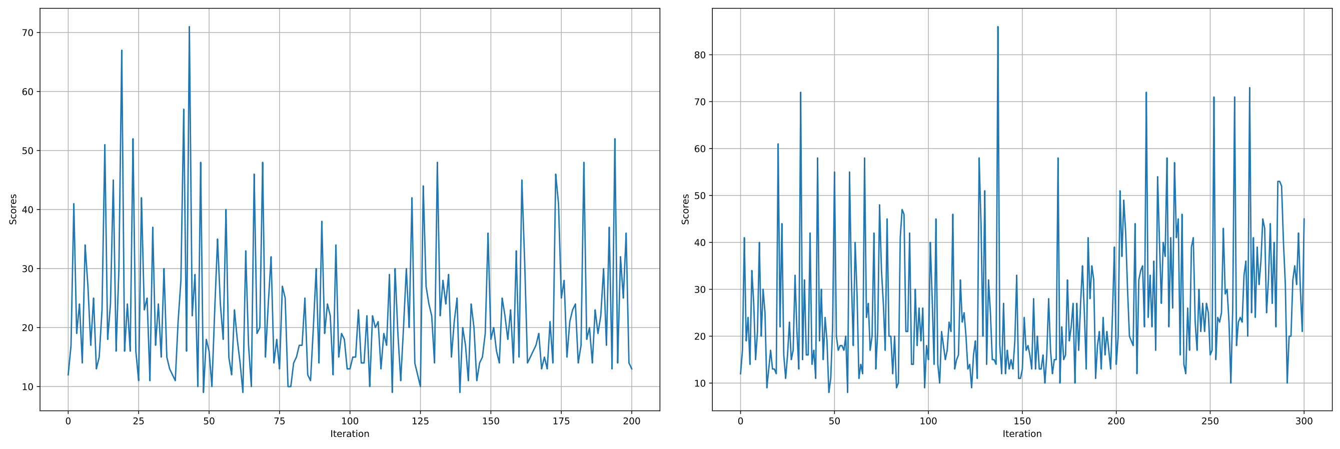
Finally, we test again the algorithm by setting the temperature to zero and we observe that the model is able to reach the best score at every trial.

V. APPENDIX



(a) (b)

FIG. 1 RL for Cart-Pole environment. (a) Scores profile for parameters: $\beta = 6$, $\gamma = 0.97$ and $\eta = 10^{-2}$. (b) Scores profile for parameters: $\beta = 15$, $\gamma = 0.98$ and $\eta = 10^{-1}$.



(a) (b)

FIG. 2 RL for Cart-Pole environment using screen pixels. Scores profile for parameters: $\beta = 15$, $\gamma = 0.98$ and $\eta = 10^{-1}$ with one input image. (b) Scores profile for parameters: $\beta = 15$, $\gamma = 0.98$ and $\eta = 10^{-1}$ with a sequence of four input images.

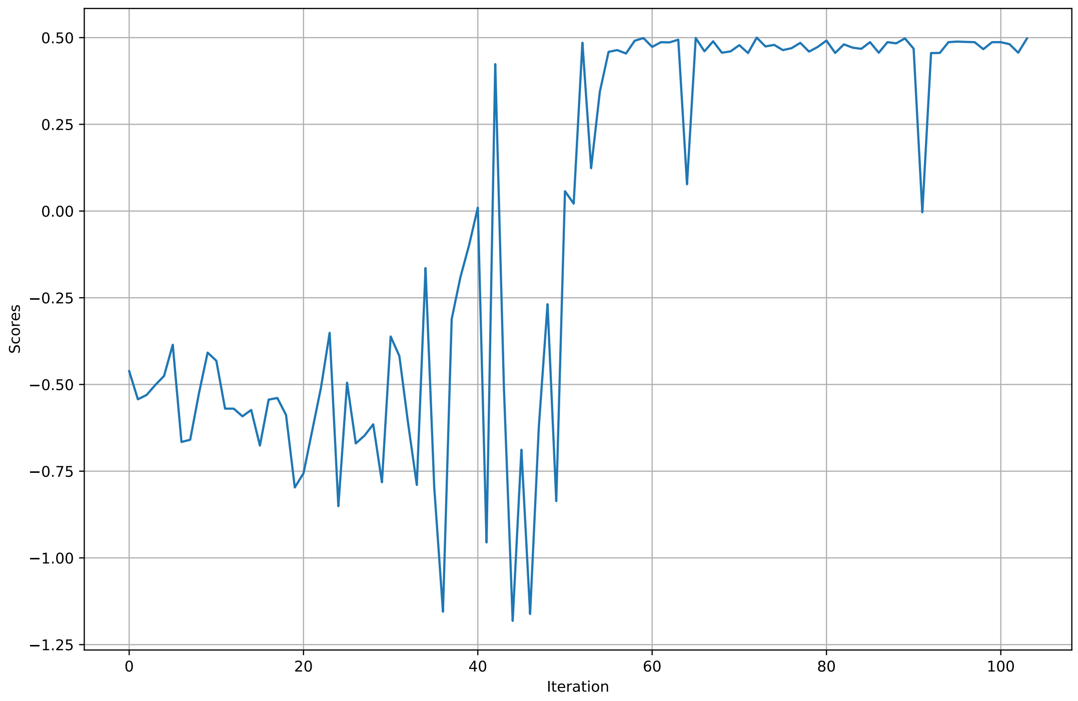


FIG. 3 RL for Mountain-Car environment. Scores profile for parameters: $\beta = 15$, $\gamma = 0.98$ and $\eta = 10^{-1}$.