**Ivancich Stefano 1227846**

**Neural Network and deep learning course 2020/21**

**Homework 3**

1. **Introduction**

In this homework we have to implement and test…..

A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled 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.

………………………………

In this case we will use the Huber loss as loss function (https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html). The Huber loss uses a squared term if the absolute element-wise error falls below beta and an L1 term otherwise. It is less sensitive to outliers than the MSELoss and in some cases prevents exploding gradients.

To overcome the exploration-exploitation dilemma, we will be using the epsilon-greedy approach to slowly decrease the randomization factor overtime. This will ensure that our agent will have a wide variety of state-action training samples and in the later part of the training, it will allow the agent to follow it’s own “trained strategy” as opposed to random actions.

Technically, in the code, we will be using a temperature term to smooth the probability of actions, and epsilon to decide between whether to take a random action or the predicted action output from the policy.

1. **Speeding up learning Convergence**

…..

gamma = 0.97 # gamma parameter for the long term reward

replay\_memory\_capacity = 10000 # Replay memory capacity

lr = 1e-2 # Optimizer learning rate

target\_net\_update\_steps = 10 # Number of episodes to wait before updating the target network

batch\_size = **256** # Number of samples to take from the replay memory for each update

bad\_state\_penalty = 0 # Penalty to the reward when we are in a bad state (in this case when the pole falls down)

min\_samples\_for\_training = 1000 # Minimum samples in the replay memory to enable the training

REACH 500 at: 790 episodes

####################################################à

gamma = 0.97 # gamma parameter for the long term reward

replay\_memory\_capacity = 10000 # Replay memory capacity

lr = 1e-2 # Optimizer learning rate

target\_net\_update\_steps = 10 # Number of episodes to wait before updating the target network

batch\_size = **256** # Number of samples to take from the replay memory for each update

bad\_state\_penalty **= 0.1** # Penalty to the reward when we are in a bad state (in this case when the pole falls down)

min\_samples\_for\_training = 1000 # Minimum samples in the replay memory to enable the training

REACH 500 at: 800

#####################################

gamma = **0.9** # gamma parameter for the long term reward

replay\_memory\_capacity = 10000 # Replay memory capacity

lr = 1e-2 # Optimizer learning rate

target\_net\_update\_steps = 10 # Number of episodes to wait before updating the target network

batch\_size = 128 # Number of samples to take from the replay memory for each update

bad\_state\_penalty = 0 # Penalty to the reward when we are in a bad state (in this case when the pole falls down)

min\_samples\_for\_training = 1000 # Minimum samples in the replay memory to enable the training

REACH 500 at: never

#####################################

gamma = **0.98** # gamma parameter for the long term reward

replay\_memory\_capacity = 10000 # Replay memory capacity

lr = 1e-2 # Optimizer learning rate

target\_net\_update\_steps = 10 # Number of episodes to wait before updating the target network

batch\_size = 128 # Number of samples to take from the replay memory for each update

bad\_state\_penalty = 0 # Penalty to the reward when we are in a bad state (in this case when the pole falls down)

min\_samples\_for\_training = 1000 # Minimum samples in the replay memory to enable the training

REACH 500 at: 720

#####################################

1. **Control CartPole using screen pixels**

…..

1. **??? Gym Environment**

The ??? gym environment consist of ….

* 2 pt: extend the notebook used in Lab 07, in order to study how the exploration profile (either using eps-greedy or softmax) impacts the learning curve. Try to tune the model hyperparameters or tweak the reward function in order to speed-up learning convergence (i.e., reach the same accuracy with fewer training episodes).
* 3 pt: extend the notebook used in Lab 07, in order to learn to control the CartPole environment using directly the screen pixels, rather than the compact state representation used during the Lab (cart position, cart velocity, pole angle, pole angular velocity). This will require to change the “observation\_space”.
* 3 pt: train a deep RL agent on a different Gym environment. You are free to choose whatever Gym environment you like from the available list, or even explore other simulation platforms: https://gym.openai.com/envs