Exercise 4

Marco Di Francesco - 100632815
ELEC-E8125 - Reinforcement Learning

October 10, 2022

Task 1

Training performance plots:

- Task a: look at 1
- Task b: look at 2

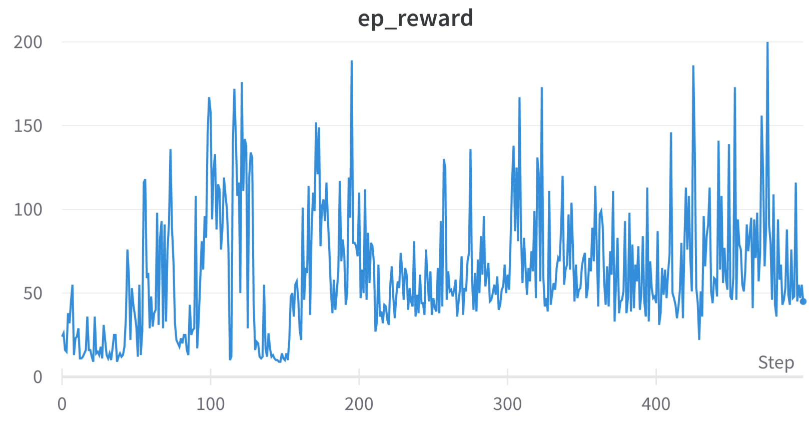

Figure 1: Training performance plot for Exercise 1 task a

Question 1.1

Would it be possible to learn accurately Q-values for the Cartpole problem using linear features (by passing the state directly to a linear regressor)? Why/why not?

No, it would not be possible to accuratly learn linear features, there may be multiple movitations for it:

- The features may not lineraly learnable: having only the states as features may not yeald learnable by using a linear regressor, for instance we may need more dimensions to learn the best parameters for all the states

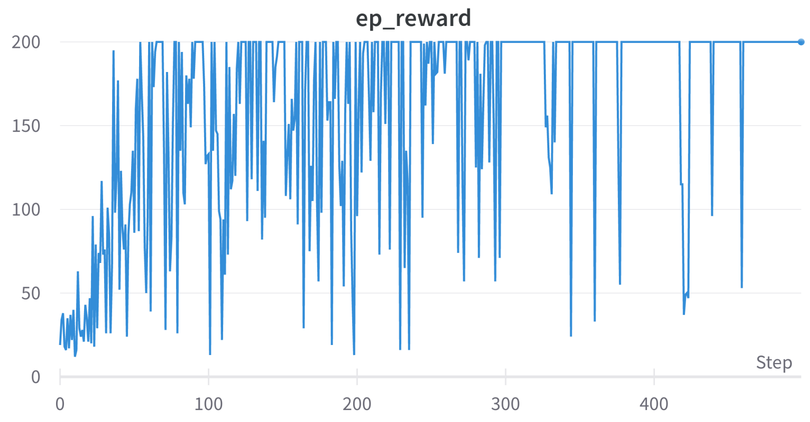

Figure 2: Training performance plot for Exercise 1 task b

- With stochastic gradient descent we may get stuck to a local minima

Question 1.2

Why do we use the experience replay buffer, how does it affect the learning performance?
Genearally speaking is used to generalize better

- Training performance is faster because we don't try to learn all the possible states but just a subset, thus making computation like loss computations on less values
- Avoiding overfitting: taking randomly from all the states in such a way that some states are not overrepresented

Why do we sample the mini-batches randomly?
There may be multiple motivations:

- In order to brake the correlation of succession of states: when we add sample to the replay buffer these samples are ordered by time in order to forget them after a while (for instance we want to consider the last 1000 steps), but the problem is that is we have 15 ordered samples the network may start learning temporal correlation, and this is something that we don't want
- We may not want to learn all the samples: we may want to learn every once in a while, for instance every 100 samples taken we may take 10 random sample and learn from them

Question 1.3

Are grid-based methods sample-efficient compared to the RBF function approximation methods? Why/why not?

Grid-based methods are not sample-efficient compared to RBF methods. Starting with saying that the state is not specified to be quantized with the grid-based methods, this means we may need to learn all the possible infinite states so it may be infinitly more difficult to compute all the possible states. Genearlly speaking also if we quantize the space in a reasonale number of states (e.g. 32), it takes way more time to learn all the possible states compared to a RBF function approximation method. On a side note, only in the edge case we quantize the states in very few state (e.g. 4) with few features the grid-based method may be faster.

Task 2

Look at 3

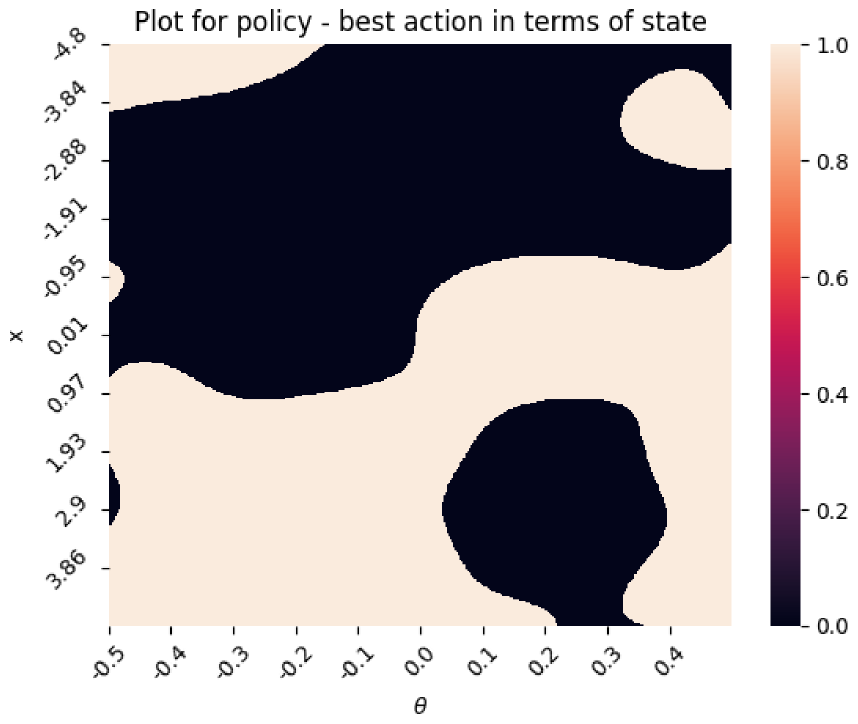

Figure 3: Plot for Task 2

Task 3

Plots for:

a) Environment CartPole, look at 4

b) Environment LunarLander, look at 5

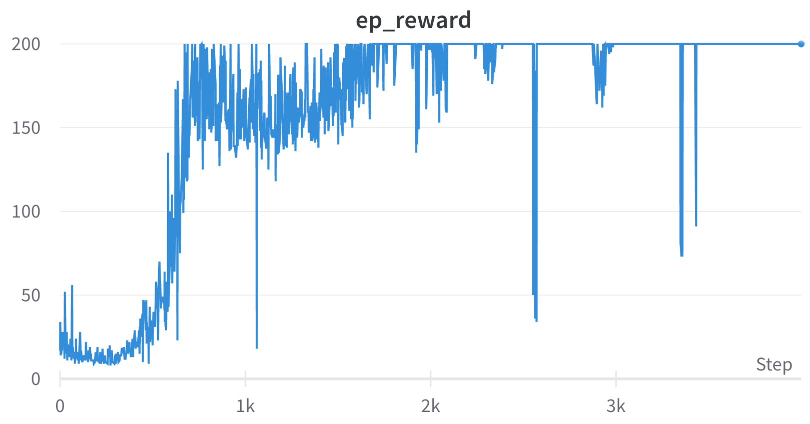

Figure 4: Plot for Task 3 part a

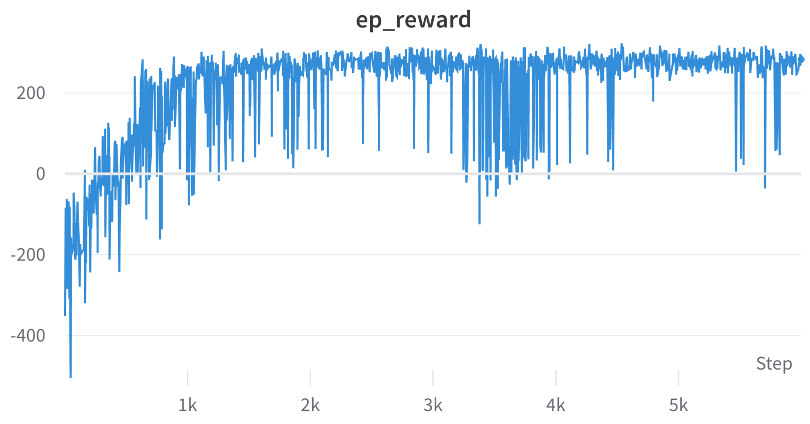

Figure 5: Plot for Task 3 part b

Question 3.1

Can Q-learning be used directly in environments with continuous action spaces?

No, without discretization it is not possbile. This is because if we take infinite largly points values we are not able to make an approximation of those values, thus making it infinitly difficult to learn all the possible states.

Question 3.2

Infinitly large values in the computation would make all the actions difficult to compute, for instance max and sum, but the real difference is when computing the derivatives in the backpropagation because we would have huge gradients. We can solve this by using gradient clipping, in this way it's possible to rescale all the values by taking the norm to be at most a particular value, in this way we can clip the gradients to a set threshold.

Question 3.3

In DQN, we use an additional target network to calculate the target Q value. Why we need this? Can we just use the same network in calculating both Q(s, a) and max a (Q(s, ·))?

No, we cannot use the same network for calculating $Q(s, a)$ and $max_a(Q(s,))$. If we would have the same network this may effect the action values of the next states S_{t+1} instead of making them stable before the update.

Also, what will happen if we do not stop gradient of the target Q value?

If we do not stop gradient of the target Q value we would update also the target network. More in particular in the code we use the detach function in order not to update the weights during the backpropagation function.

In litterature this is known as catastrophic forgetting [1], that happens where we take decision based on our previous actions, thus worsening our results until we take completly wrong decisions.

References

[1] Wikipedia, "Catastrophic interference — Wikipedia, the free encyclopedia." `http://en.wikipedia.org/w/index.php?title=Catastrophic%20interference&oldid=1111369428`, 2022. [Online; accessed 11-October-2022].