Alexandre Allani

CID : 1797836

November 21, 2019

Reinforcement Learning Part 2: Coursework Part 1

1 Questions

Question 1 (i) : The variance of the loss is higher in the neural network with online learning rather than the one with a replay buffer. In online learning, the neural network trains with one sample and then "forgets" it. This means that the agent has to explore again that state with the same action to train on that situation. However, with batch learning, multiple samples are used at the same time to train the network and they are saved, with a probability to be used again afterward. This explains why the loss is going down with a lower variance for batch learning.

Question 1 (ii) : As a consequence of the first question, batch learning will be more effective and will converge faster than online learning. Online learning will eventually converge, but it will take a considerable amount of time since it is only training with one sample at a time. Plus it needs to re-experience a state to gain knowledge from it again. In the end, the sample used to train the online network will have great importance and "move" a lot the weight of the neural network. That is why the neural network with batch learning will converge faster.

Question 2 (i) : The bottom left part is more likely to have a more accurate prediction for the Q value since it is the region where the agent starts. For the moment, the actions taken by our agent during the training are random. So the farthest the region is, the less likely the agent is going to "visit". Hence the top right region might be off compared to the real Q-value of the problem.

Question 2 (ii) : The agent is not able to reach the policy. With the Q-value representation, we can already see that it kind of knows the direction, (the yellow triangles are directed toward the goal). However here the agent is blocked on the wall. There are multiple explanations. First, the agent is not trained enough. With more iterations, Q-value will be more precise, hence giving a better evaluation of the distance. A second explanation would be the discount factor of 0. Here the agent focuses only on the immediate reward, knowing that the deepQ network hasn't converged, it is likely that the states will have values of the same order in this area. This explains why he is hitting the wall.

Question 3 (i) : This time, we are predicting the Qvalue with a discount factor greater than 0. The overall shape seems normal with a decreasing mean of the loss. However, I have a strange decrease toward 100 steps. I would guess that my agent was experiencing the same states at the beginning and got better at it. However after a moment, it discovered new states, the loss went up. Since deepQ networks without target networks tend to increase Qvalue of the adjacent states it has not visited (because it predicts a continuous function), this would be a plausible explanation.

Question 3 (ii) : In this part, the overall loss is very noisy. I tried with more steps, and it seems to converge in the end. After each update of the network, the loss function goes up. Indeed, the learning network will train on the Qnetwork that is initialized with wrong values. And at the purpose is for the online network to overfit the values of the target network. Each time the target network is updated, the loss is increased, because the online network has to "overfit" the target network again. However each time there is an update, the overall predictions will get better. However, in this situation, we do not let enough steps for the learning network to converge resulting in this high variance.

Question 4 (i) : The optimal value for delta is toward $\delta = 0.2$. When $\delta = 0$, the agent is fully random. It will only visit an area quite close to the start region, and it is less likely to get toward the goal. This explains why the agent has a lower total sum of reward. To confirm this trend, multiple tests without a fixed seed should be done. (In the next question I'm expressing my opinion towards the result I got. I think this result of δ is heavily biased by the fixed seed)

Question 4 (ii) : When $\delta = 1$, the agent only takes the greedy action. Hence the agent is never exploring, only exploiting. The agent can get stuck in a local maximum for a long time, and I think the agent gets stuck in the wall for a certain amount of steps which gives a bad result. The value of $\delta = 0.2$ is better in the end because it gives at the beginning of the training a chance for the agent to explore. And at the end, when the agent kinds of know the world, it explores less, and exploit more. However I said in the previous question, I think this value of δ is a bit off. After 5 exploration, with $\delta = 0.2$, ϵ drops to 0.1. So the exploration part is done after 5 steps in a run of 500. This doesn't let the agent explore enough the world. I have two possible explanations:

- I start the epsilon greedy policy when the replay buffer is filled. So this means, within the 50 steps of full randomness, the agent explored enough. That is why a high value of δ, such as 0.2 is correct in this situation
- I got quite lucky in the initialization of the deepQ network, giving Qvalues that are not far from reality. Or at least that allows exploration.

I think, that the first explanation is more plausible. It might be also a combination of the two, a good initialization with more steps thanks to the first 50 steps.

Question 5 (i) : The reward function that I have chosen is $reward = 1 - d^3$. What I wanted to achieve, is giving a higher reward the closer the agent is to the end. In the original function, this was done linearly, so the reward would increase the same way when going from a distance of 0.4 to 0.5 and a distance of 0.5 to 0.6 (increase in the reward of 0.1). With my function, the agent gets more rewards the closer to the goal he is. So the agent will struggle more at the beginning, and not at the end. However, within 500 steps, the agent is more likely to explore more and then predict more precisely the starting area. This explains why it seems to converge faster (on the figure) with this reward function.

Question 5 (ii) : The sparse function would give worse performance than my reward function, and the "basic" reward function. Even though our agent is farsighted ($\gamma = 0.9$), it would have to first experience randomly the final state, to start predicting correctly the Q values. However, this is achieved randomly. Moreover, since we are using an ϵ-greedy policy with $\delta = 0.2$, the more steps the agent is taking, the less likely it is to reach the goal randomly. A sparse reward function, in this case, would be one of the worst reward function. The agent will most probably experience the same state and think that a loop is the best action to take. As explained in the previous question, a continuous reward function over all the states is better.

2 Figures

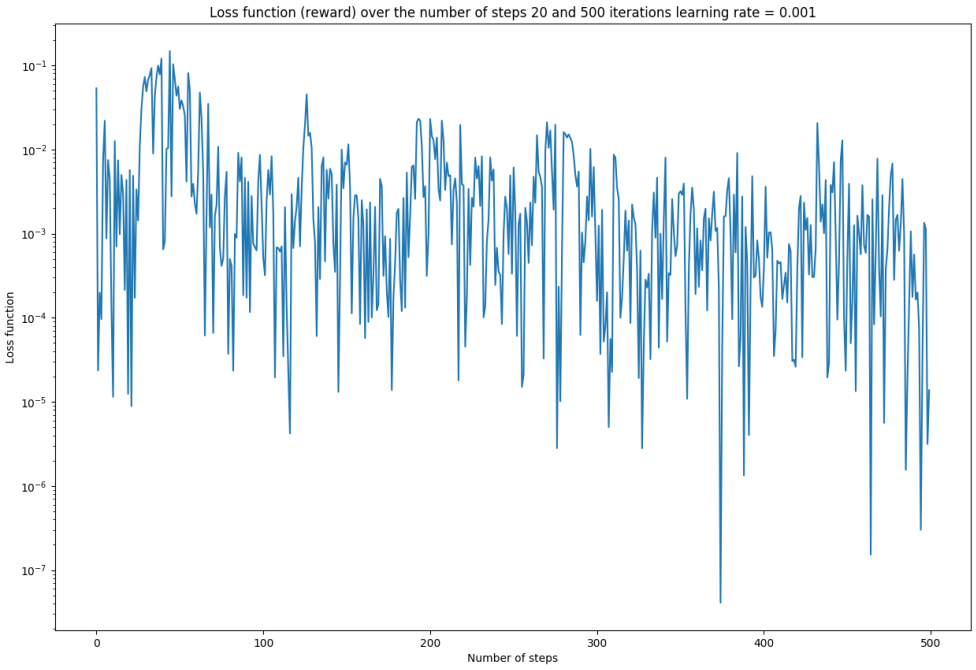

Figure 1(a): Online Loss 1

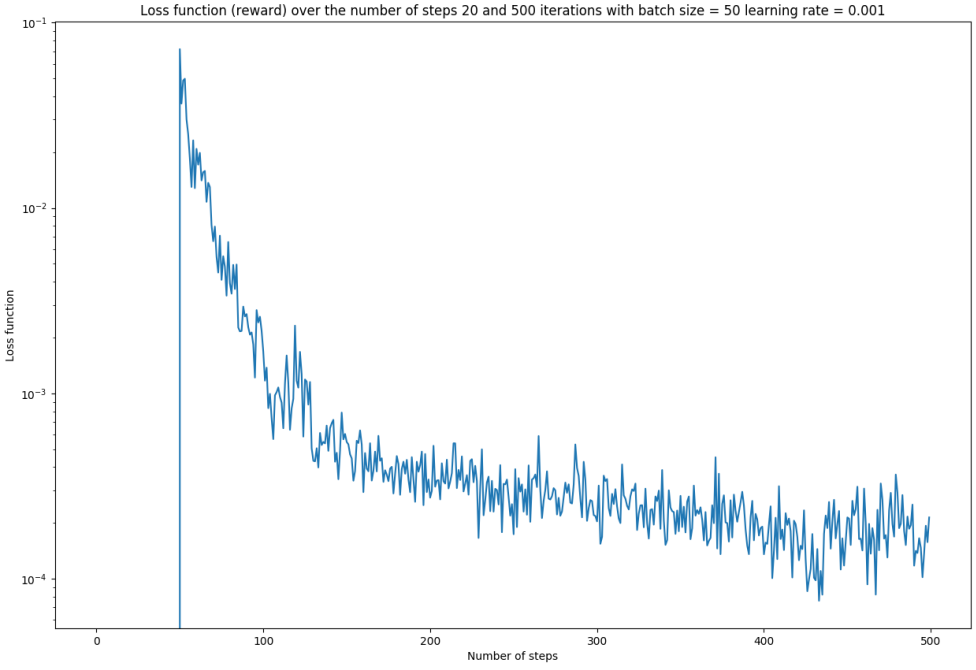

Figure 1(b): Batch Loss 1

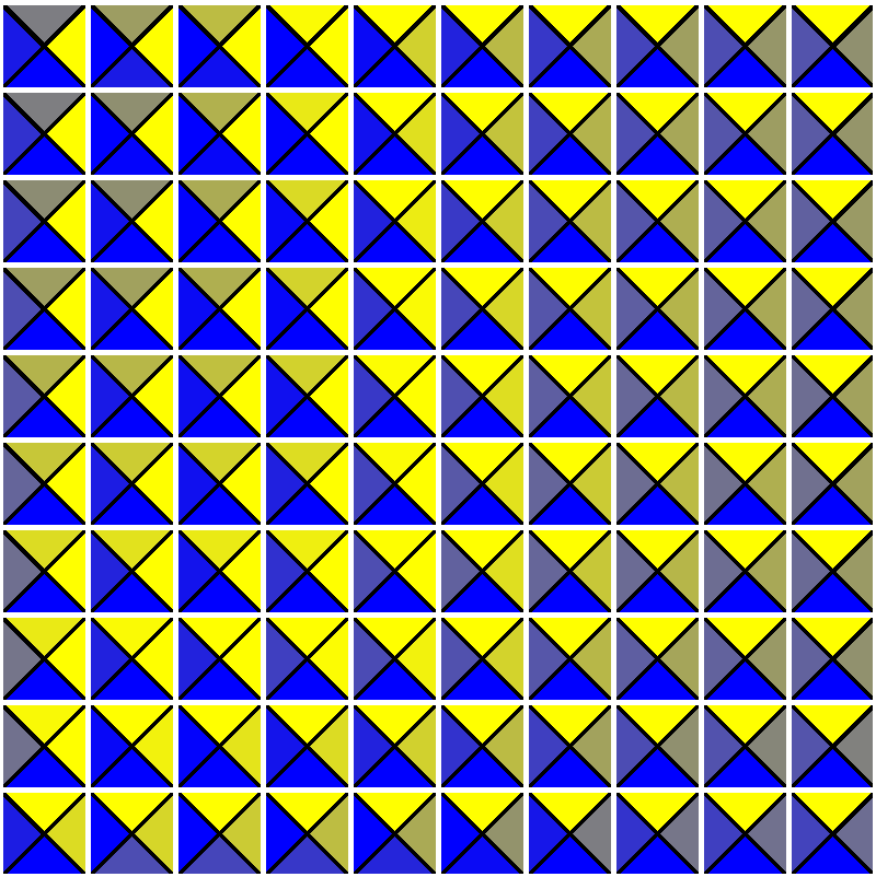

Figure 2(a): Q-value representation2

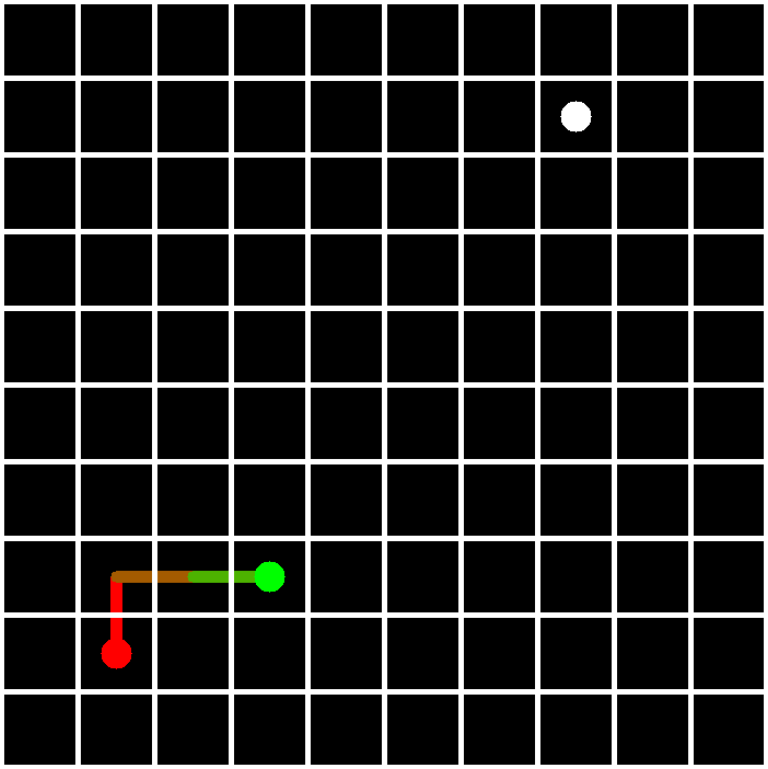

Figure 2(b): Path representation2

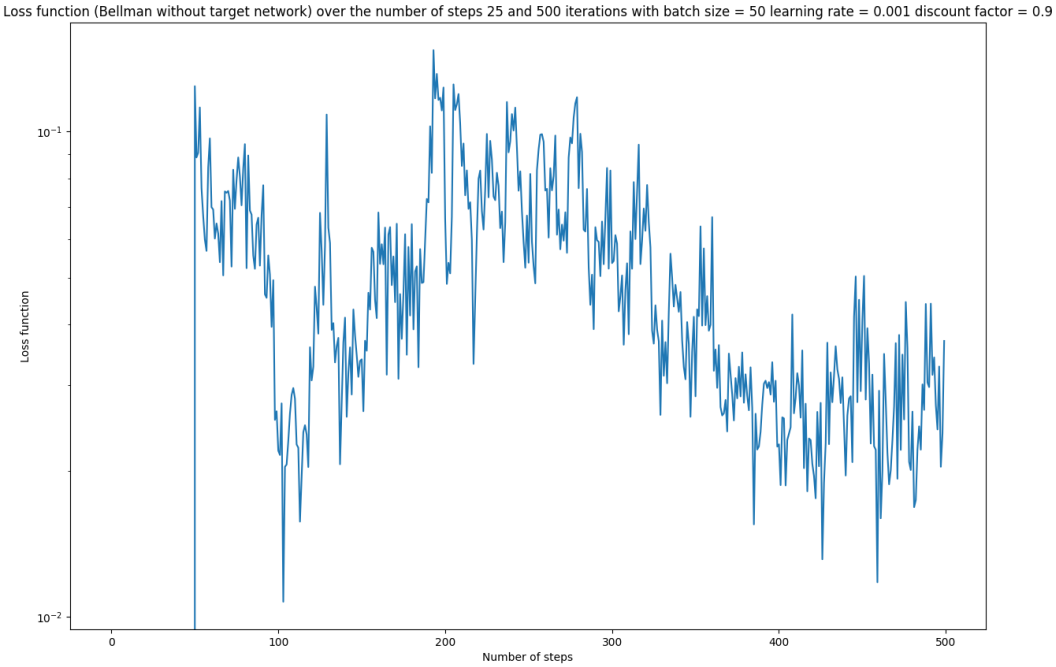

Figure 3(a): Loss with Q-network 3

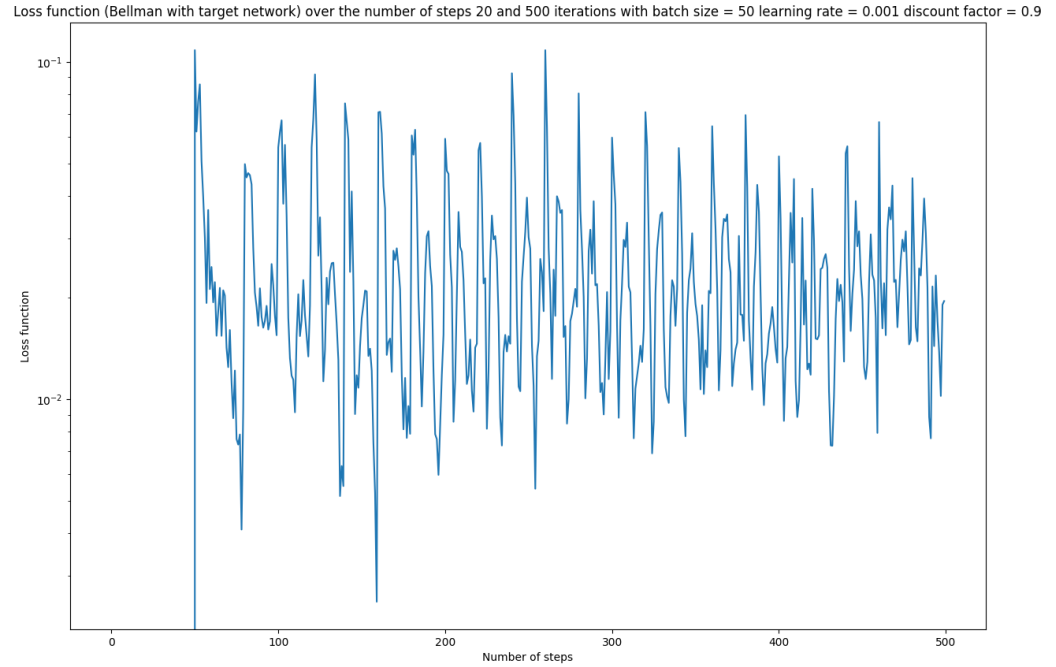

Figure 3(b): Loss with target network 3

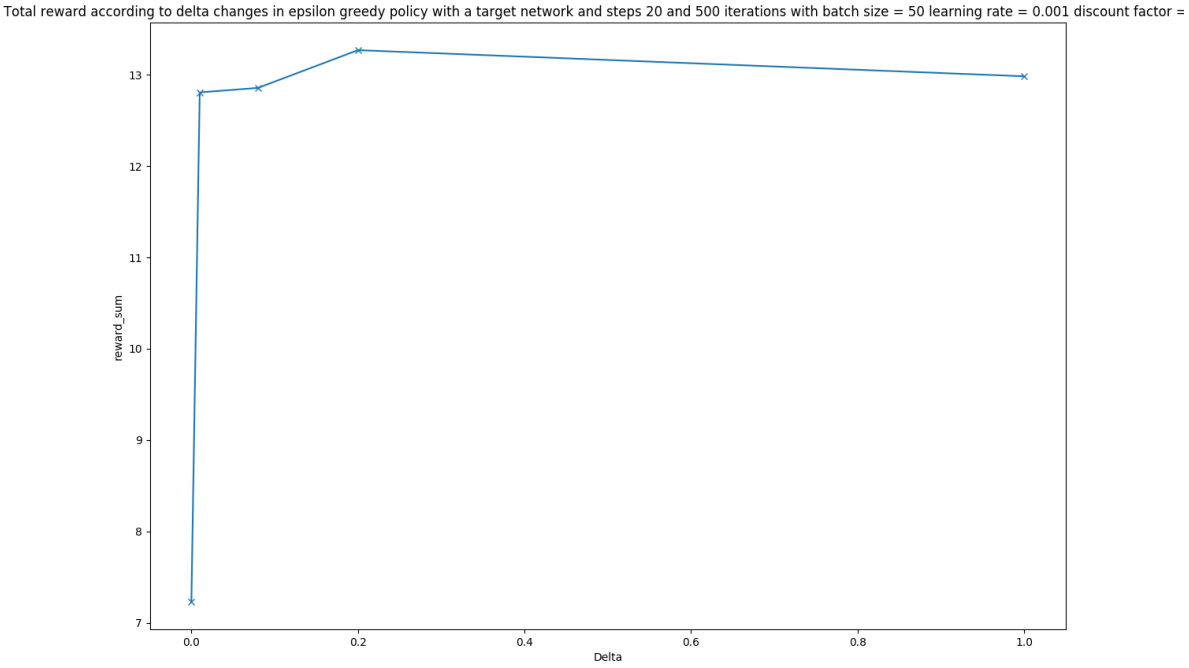

Figure 4: Total reward according to delta changes in epsilon greedy policy

Figure 5(a): Reward function : $1 - d^3$ 5

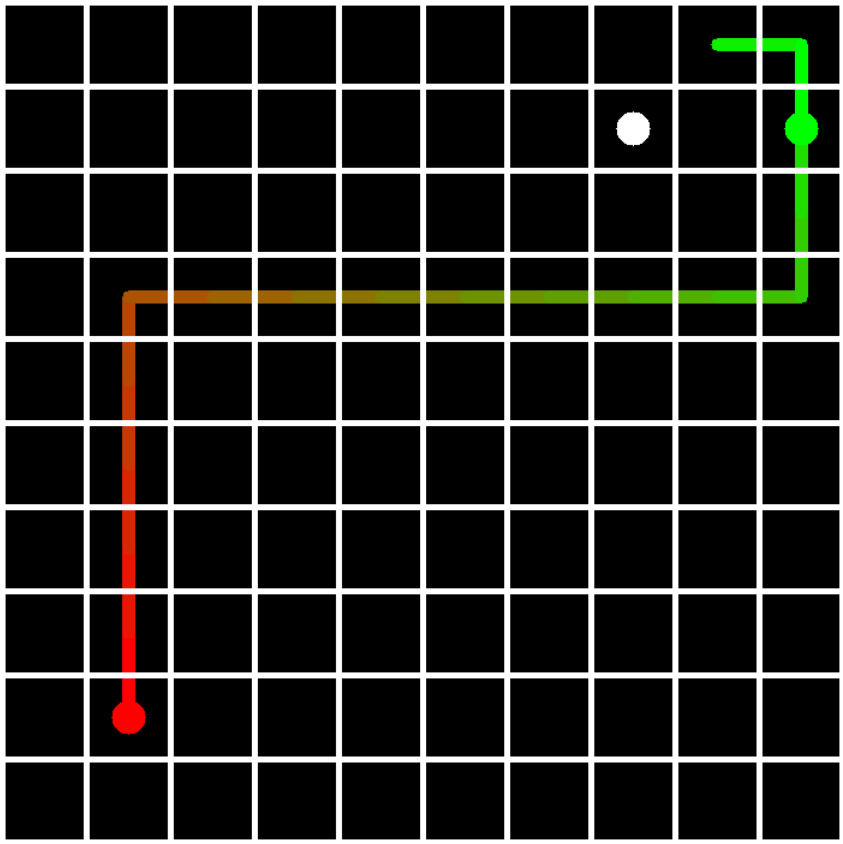

Figure 5(b): Path for reward function : $1 - d^3$ 5