Mini-project 1: Tic Tac Toe

Mihaela Berezantev (287768), Pruñonosa Soler Guillem (350999)

June 2022

1 Question 1

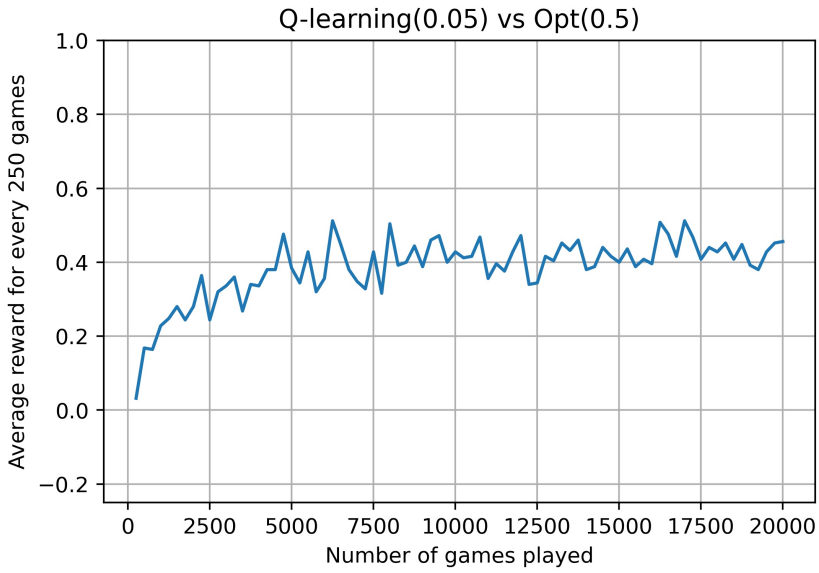

Figure 1: We can see that the average reward progressively increases over time which indicates that the player learns to maximise the total reward. A low epsilon (our choice is 0.5) ensures a high probability that the player takes an action corresponding to the optimised Q-values instead of a randomly chosen one.

2 Question 2

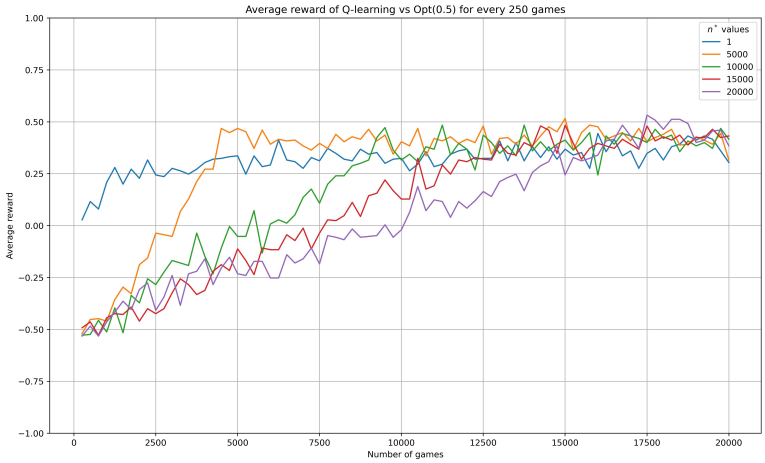

Figure 2: By favouring a high ϵ, we benefit from exploration, i.e. the agent often chooses an action at random and improves its knowledge about various possible moves which could show a long-term benefit. When ϵ is lower, we benefit from exploitation as the agent uses the estimated values to choose an action and get the most reward. In a perfect scenario, we would like to stop exploration as soon as the best solution has been identified, but this is impossible. Decreasing ϵ over time mimics this behaviour and is therefore a better solution than a fixed ϵ. n^* defines the number of exploratory games, so if it is high , the agent reaches a high average reward slower. With a lower n^*, we reach the exploitation phase quicker and therefore start maximising reward faster. It can be noted that for $n^* = 1$ there is almost no exploration. Therefore, the agent quickly reaches an approximate average of 0.4, but later, it is penalised due to the lack of exploration. On the other hand, with $n \geq 10000$, the exploration phase is too long and the convergence is slow. It looks like there is a good compromise between exploration and exploitation for $n = 5000$ in this case.

3 Question 3

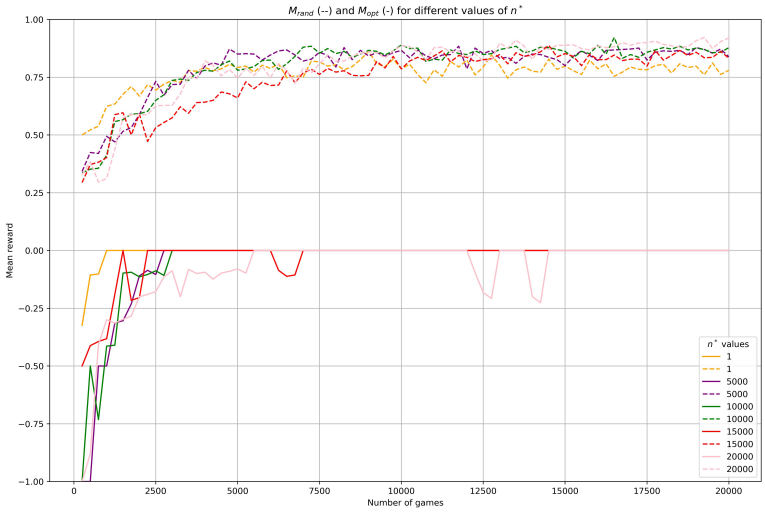

Figure 3: It can be noted that for M_{rand}, the curves corresponding to different values of n^* follow a pattern similar to the previous question. However, the mean reward is higher which is due to the fact that in this case the agent plays against a random player (Opt(1)) instead of Opt(0.5). As for M_{opt}, all the curves converge to 0 and the number of games played until this happens depends on the value of n^*. In this case, the agent plays against Opt(0), so most of the games finish with a draw as both players make the best moves possible.

4 Question 4

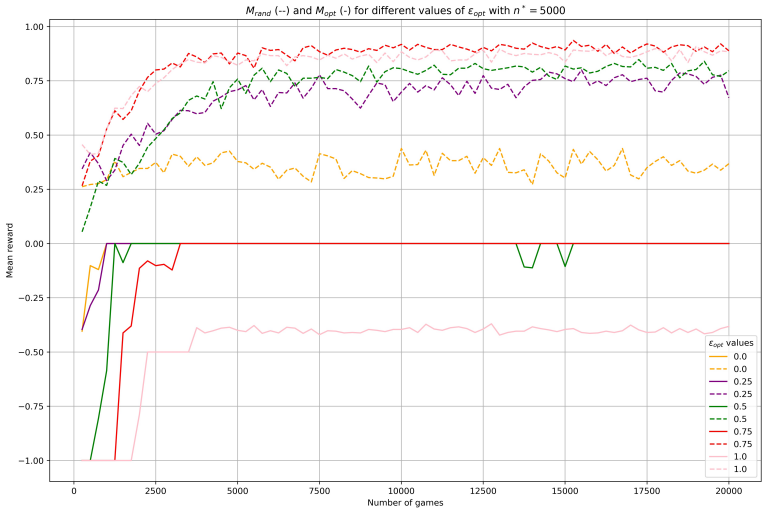

Figure 4: When testing against the random player, we observe that M_{rand} is always ≥ 0 which means that the Q-learning agent wins more than it looses. For $\epsilon_{opt} > 0$, the curves are above 0.6, which means that the agent's performance is good. The curve is lowest when the agent is trained by playing against Opt(0) and highest if he trains against Opt(0.75). This is due to the fact that the agent learns how to beat its specific opponent. Therefore, when facing a random player after playing against an optimal one, it doesn't have the optimal strategy. As for the case when we test against an optimal player, we see that M_{opt} is negative when the agent is trained against a completely random player ($\epsilon_{opt} = 1$). In the other cases, it manages to win and loose the same amount of games (or finish on a draw), after it has been trained sufficiently enough. The amount of games needed for obtaining $M_{opt} = 0$ depends on the randomness of the opponent during training.

5 Question 5

The highest M_{opt} value is 0. It is obtained when the Q-learning agent is trained against Opt(0), Opt(0.25), Opt(0.5) and Opt(0.75). As for the highest M_{ran}, its value is 0.92 and was obtained when training against Opt(0.75).

6 Question 6

The Q-value Q(s, a) is an estimation of how good is it to take the action a at the state s and is computed based on the old value, the reward, and an estimate of the optimal future value. When the Q-learning agent plays against the random player, it quickly reaches an average reward that is higher than when it plays against the optimal player. As Q(s, a) depends on the reward, the computed Q-values are not the same. In a way, the agent trains its Q-values to be optimal against its respective opponent.

7 Question 7

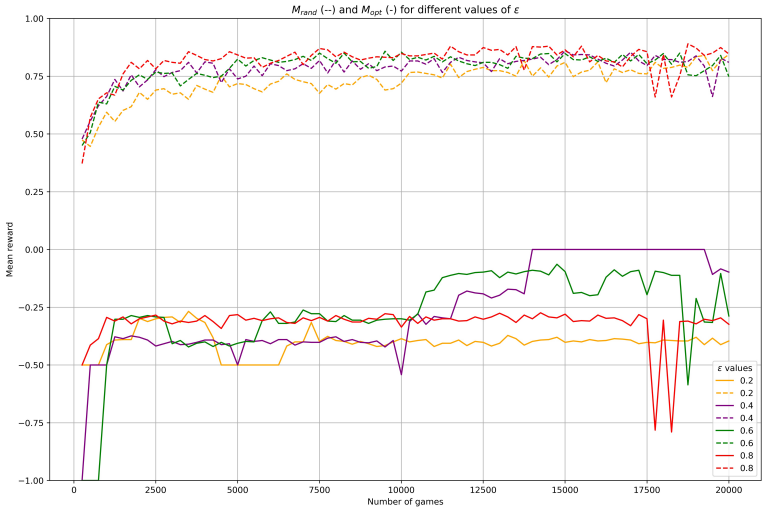

Figure 5: When testing against the random player, we see that M_{rand} quickly achieves a value close to 1, which means that the agent is learning. We see that M_{rand} is highest when the agent is trained against Q-learning(0.8) which is close to the totally random opponent during testing (Opt(1)). When testing against the optimal player, we see that the agent is learning, but achieves a value of $M_{opt} = 0$ only when it is trained against Q-learning(0.4). Also, the process is rather slow (value achieved after 13000 games). ϵ affects the level of exploration of the models and therefore influences the average reward.

8 Question 8

Figure 6: When testing against the random player, we see that M_{rand} behaves the same as previously - it quickly achieves a value close to 1. Nevertheless, there is an exception, when we define $n^* = 1$: there isn't enough exploration and we clearly achieve worse measures. When testing against the optimal player, we see that M_{opt} fluctuates more. A big n^* means that the exploration decreases slowly, so both have more games to learn the best actions for different situations, while with small n^* they are always training the same actions that are not the optimal ones indeed.

9 Question 9

When testing against a random player M_{rand} we reached a measure of 0.9 for a fixed ϵ of 0.8. This means that our player learns more playing against itself if it does a lot of random moves. Nevertheless, we got best performance measures decreasing the exploration level. We reached 0.94 with a value of $n^* = 20'000$ and $n^* = 15'000$. The highest M_{opt} we get when testing against the optimal player is 0 and we get it also with $n^* = 15'000$ and $20'000$, or with 0.4 in case of fixing ϵ.

10 Question 10

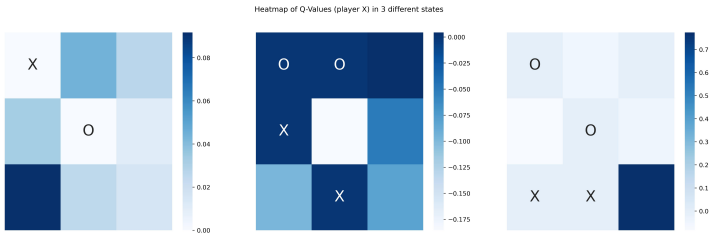

Figure 7: All 3 results make sense as the proposed actions are the optimal ones: in the first case the X player has more chances to win if he chooses the position (2, 0), in the second case player X has to choose position (0, 2) if he doesn't want to loose and finally, in the third case player X has to choose position (2, 2) if he wants to win. From this example we conclude that the agent learnt the game well.

11 Question 11

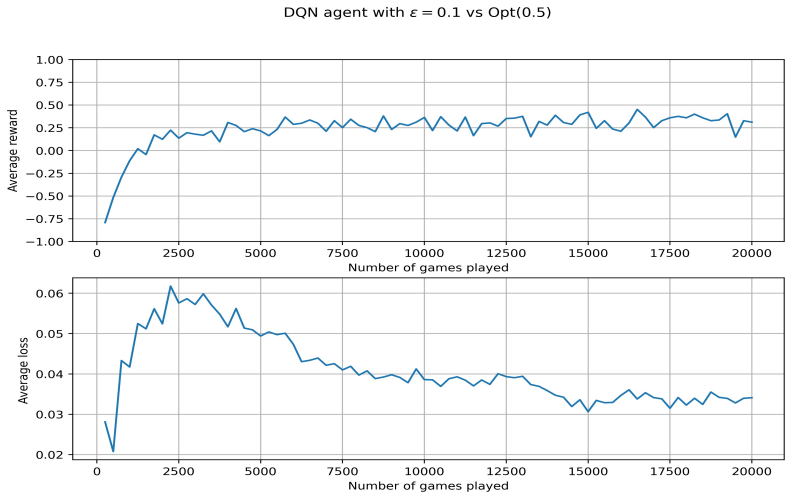

Figure 8: It can be noted that the agent learns how to play because the average reward increases over time. Also, the loss decreases over time.

12 Question 12

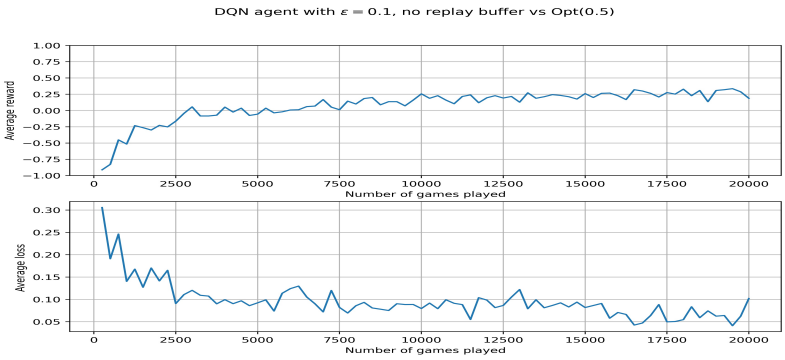

Figure 9: We observe that the agent learns as the reward grows and the loss diminishes over time. However, the highest reward is lower than previously and the loss is higher. This might be due to the similarity of states s and s' that have only one step between them.

13 Question 13

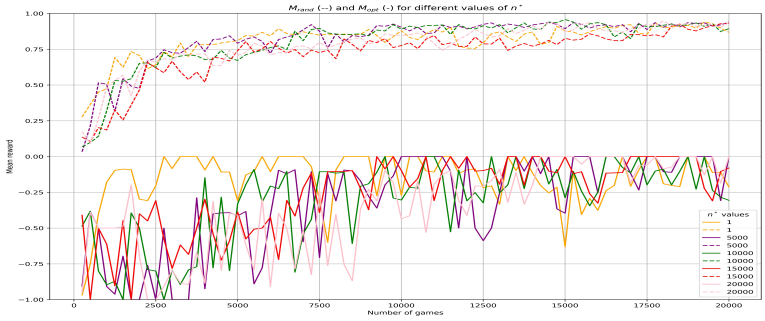

Figure 10: We can see that M_{rand} is getting closer to 1 as the number of games played increases. We can see that decreasing ϵ helps training in this case as the highest curve is obtained for $n^* = 5000$. As for M_{opt}, we can see that the agent is struggling to converge towards 0. Nevertheless, we see a clear increase of the number of wins over time. A possible explanation could be that the algorithm needs more training in order to converge.

14 Question 14

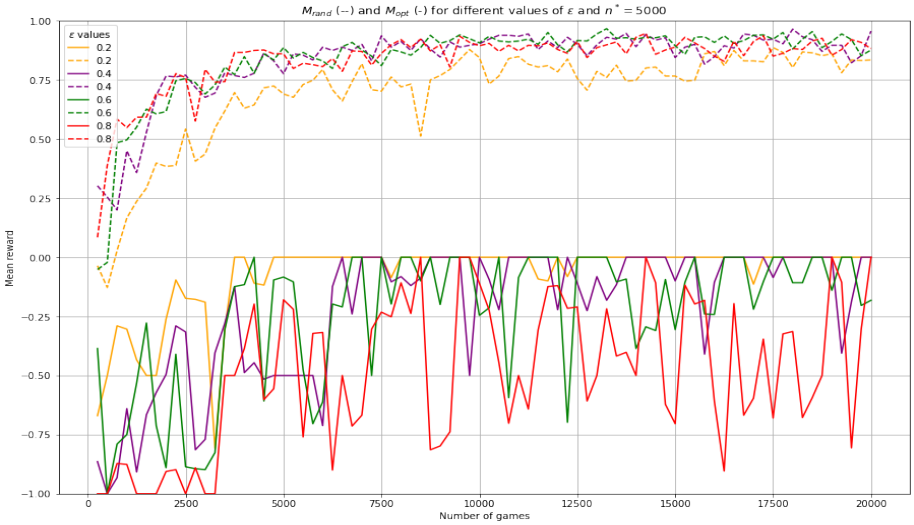

Figure 11: As previously, M_{rand} approaches 1 over time for all the values of ϵ. The lowest curve is that of $\epsilon = 0.2$ which is similar to Figure 5 in Question 7. The curves of M_{opt} oscillate considerably, especially the one with $\epsilon = 0.8$ which corresponds to learning from a highly random player. However, we see that the curves are higher over time which indicates that the policy is getting better.

15 Question 15

The highest M_{opt} is 0 and the highest M_{rand} is 0.956.

16 Question 20

	QL experts	**QL self**	**DQL experts**	**DQL self**
M_{opt}	0	0	0	NA
M_{ran}	0.91	0.944	0.956	NA
T_{train}	2250	1500	2750	NA

17 Question 21

Our results in Q-learning combined with ϵ-greedy policy have been successful. We have been able to achieve the optimal policy in both training against experts

and in self-learning training. We have also managed to win almost all the games against the random player, so that means that our player can learn how to play against any type of player. We have also achieved the optimal policy in Deep Q-learning, and although our player needed more time training to achieve its best results, at the end the results were even slightly better than with the regular Q-learning.