Mini-project 1: Tic Tac Toe

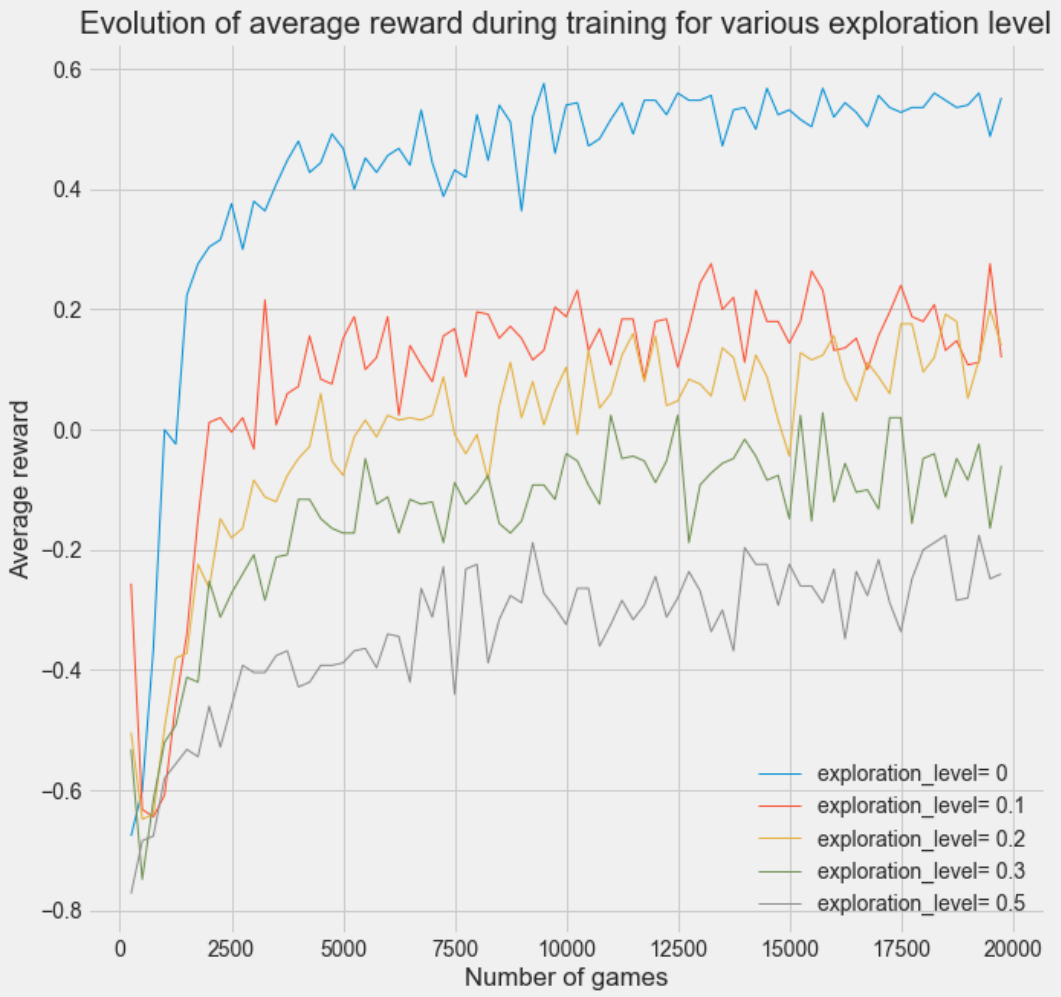
Henry Papadatos 284446,
Aitana Waelbroeck 282764

June 6, 2022

1 Introduction

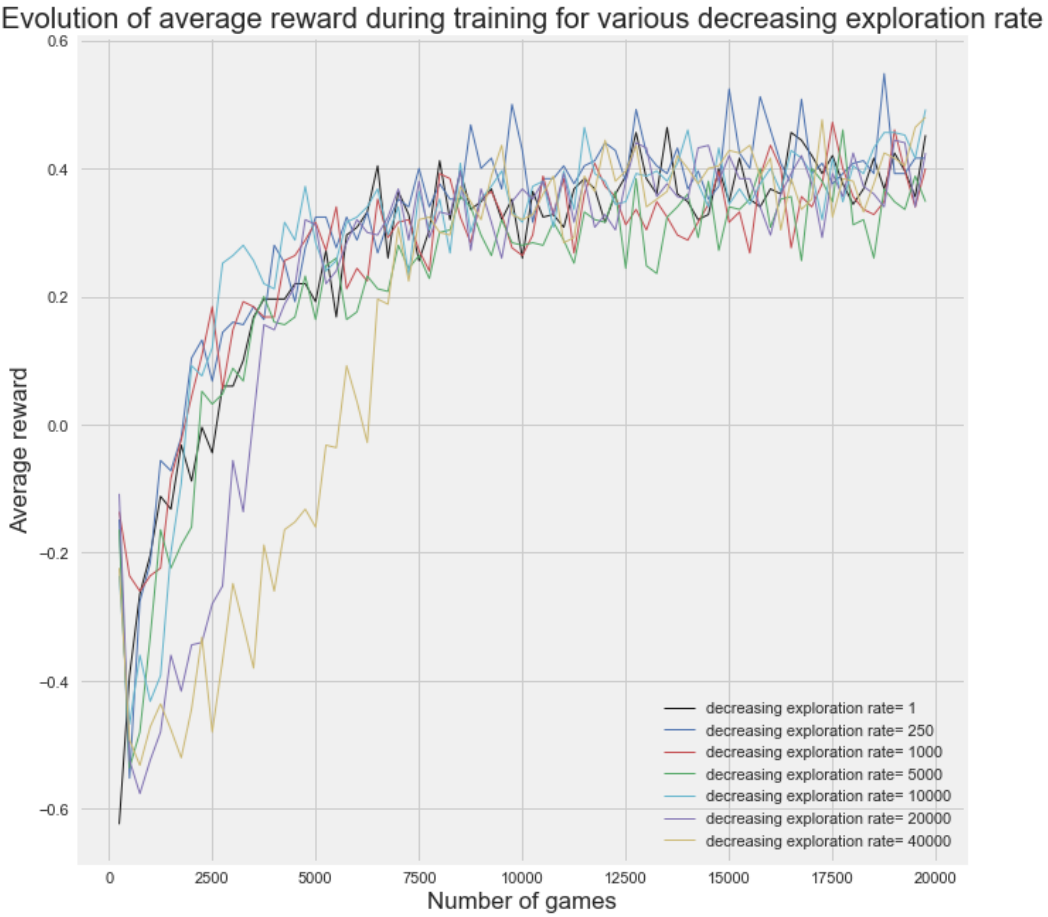
2 Q-Learning

2.1 Learning from experts

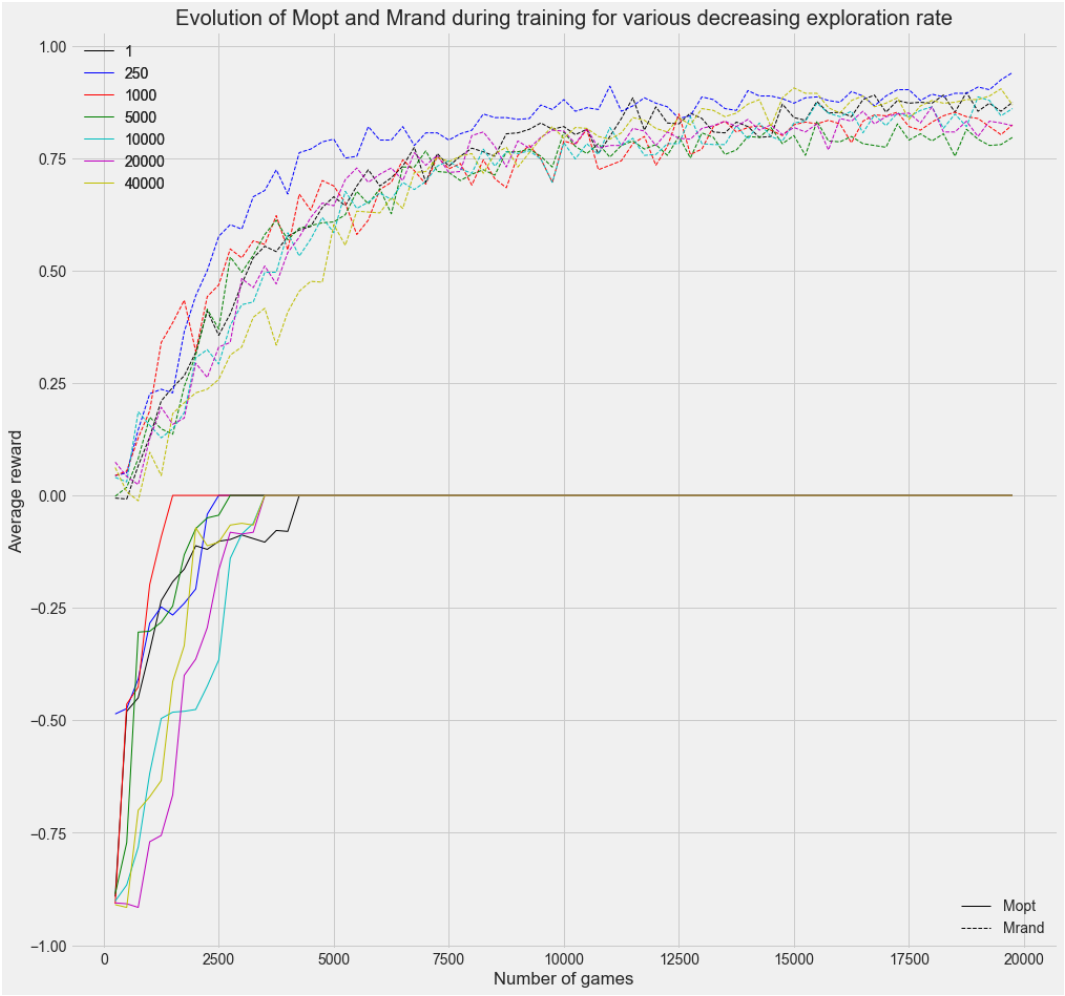


Question 1 For any value of ϵ , the average reward over 250 games quickly increases, so the agent learns quickly. During training, ϵ will bias the results, and thus this graph is not a good metric to choose ϵ. We have found using the Mopt and Mrand metric that $\epsilon = 0.1$ is a good choice

2.2 Decreasing exploration

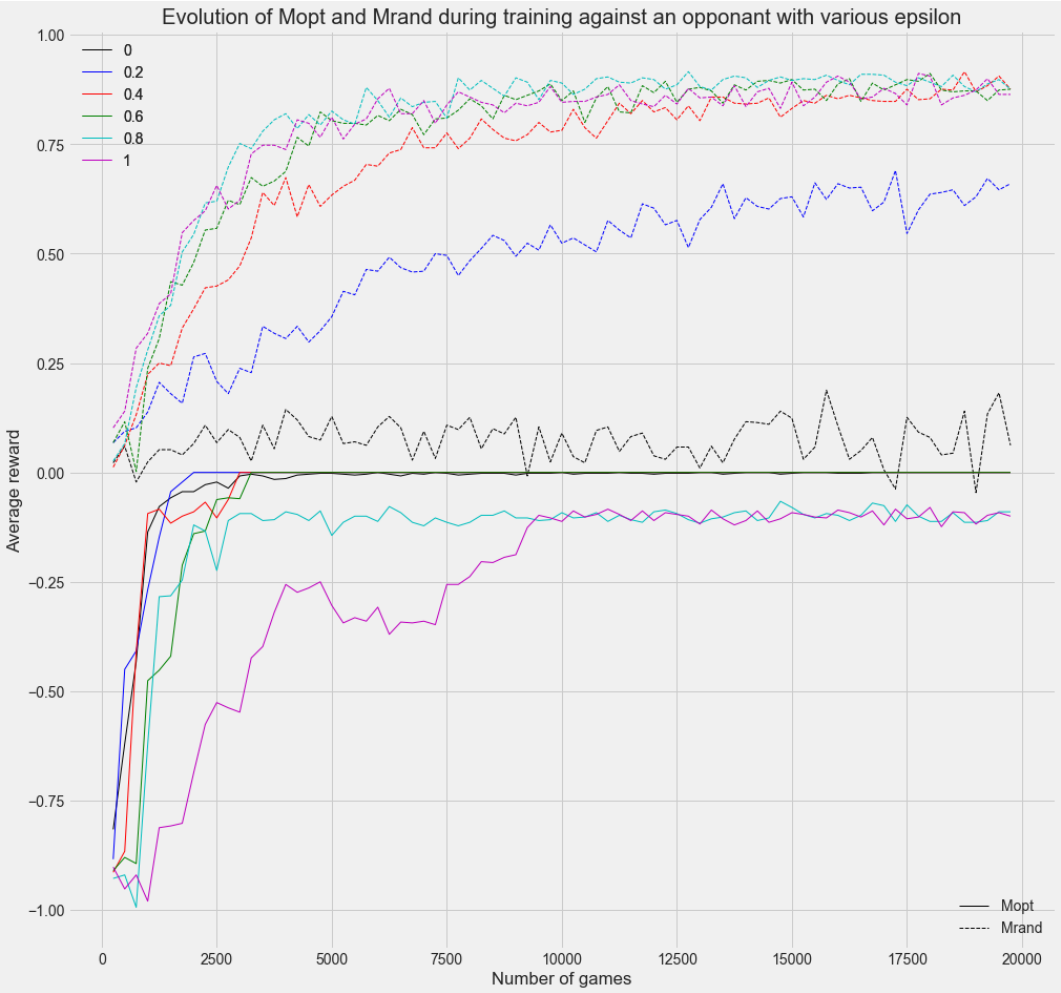


Question 2 A high n^* will lead to a slower decrease of ϵ over the number of games. Therefore, over the last games, ϵ will still be relatively high leading to more random actions played by the agent. Thus, for a high n^* of 40000, the agent stabilizes slower, as it is still exploring and learning while the others are more greedy already. In practice, we can see that an n^* of 1 (equivalent to fix epsilon, $\epsilon = \epsilon_{min} = 0.1$), reaches around the same performance as the other n^*'s, even if an n^* of 5000 seems to reach higher values. It is actually hard to compare since the plots are a bit noisy even after stabilization. Thus, we can't really conclude that adding a decreasing rate is better, but we can say that a slow decreasing rate (high n^*), leads to slower learning.



Question 3 Here,we can really asses the performances of our agents against an optimal and random policy. When playing against a random player, we can see that an n^* of 250 leads to better results overall. Against the optimal player, higher n^* values perform better than $n^* = 1$, and $n^* = 10000$ gives the best results followed by $n^* = 250$. Therefore, sticking to a value of $n^* = 250$ is a good compromise.

2.3 Good experts and bad experts



Question 4 Here we use $n^* = 250$. When varying the exploration ϵ_{opt} of the opponent during training, the performance of our agent changes. Mopt is better for lower ϵ_{opt} values used in the training, since the model has been trained against an opponent that was already almost optimal. Indeed the agent trained against the optimal model ($\epsilon_{opt} = 0$) and tested on an optimal opponent quickly reaches an average reward of 0, which was to be expected. However, interestingly, $\epsilon_{opt} = 0.2$ gives even better results. When taking the opposite case, an agent trained against a random player $\epsilon_{opt} = 1$, learned some moves that made it win sometimes against the optimal player during testing, but not enough to have a really good performance. Mrand performance increases with the ϵ_{opt} value. For $\epsilon_{opt} = 0$ the model was only trained against an optimal player and when tested against a random player, the performance is really bad. This is because the optimal player has a fixed strategy that always uses the same moves, and thus the agent only learns these specific moves, and is lost infront of the random moves of the random player. With a bigger ϵ_{opt}, the agent is trained against an opponent closer to a random policy, and thus Mrand results are better.

Question 5 The maximum values achieved after 20000 games were 0.0 for Mopt (for all the decreasing rates) and 0.904 for Mrand, for a training done with $n^* = 20000$.

Question 6 The actions of the optimal player are systematic, and therefore playing against him leads to a large panel of unvisited states. Thus many states in our graph will not be visited by our agent. However, when training against a random player, the random behavior will make sure we visit a larger part of the graph. Therefore, for a same action and state, Q1 might be 0 (because unvisited), while Q2 may have been visited. Therefore Q2 might have non-zero values.

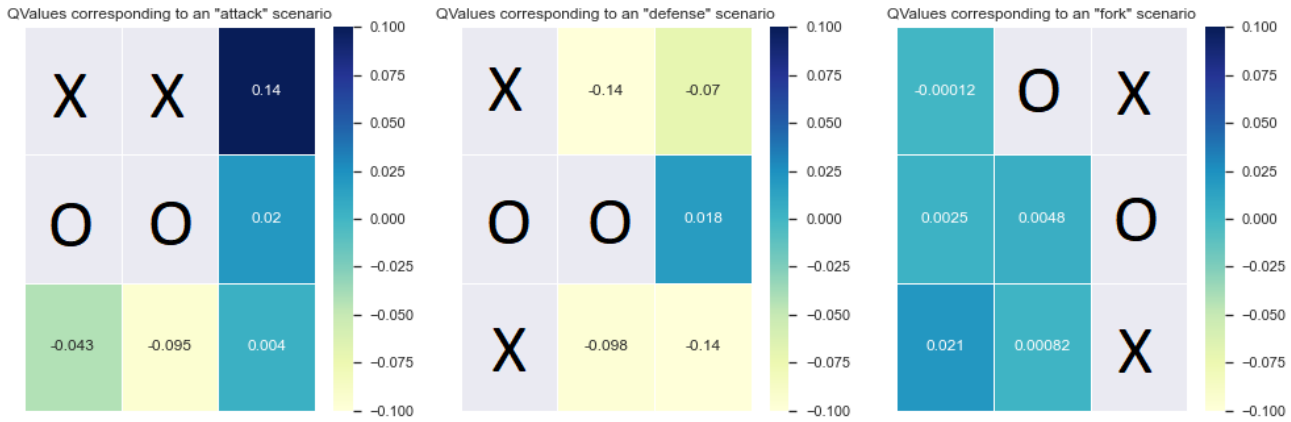
Another difference, is that it is impossible to win against an optimal player. Therefore, during the training the reward can only be zero or negative which will lead to negative or null Q values too. This is not true for Q2 since it could win by chance and thus the reward can be positive leading to positive Q2 too.

2.4 Learning by self-practice

Question 7 When training with two optimal players the agent doesn't learn, and doesn't perform well against either a random or optimal opponent. As in Question 4, it might be because the optimal player has a fixed strategy and thus always the sames moves are played by the agent and this blocks the training. This is a bit better for $\epsilon = 0.2$, here the agent learns, but really slowly. When training with an $\epsilon = 0.8$, the agent has the best results both against the optimal and random opponents

Question 8 The performances with a decreasing learning rate are worse for Mopt. The average reward reaches lower levels than for a fixed $\epsilon = 0.8$. Mrand is also worse in general, even if all the agents learn (compare to the last question, where some fixed epsilon values lead to really bad learning). The average reward reached is overall lower than before (0.75 vs 0.9)

Question 9 The highest values of Mopt is -0.066 for $n^* = 40000$, and Mrand is 0.796 for $n^* = 20000$

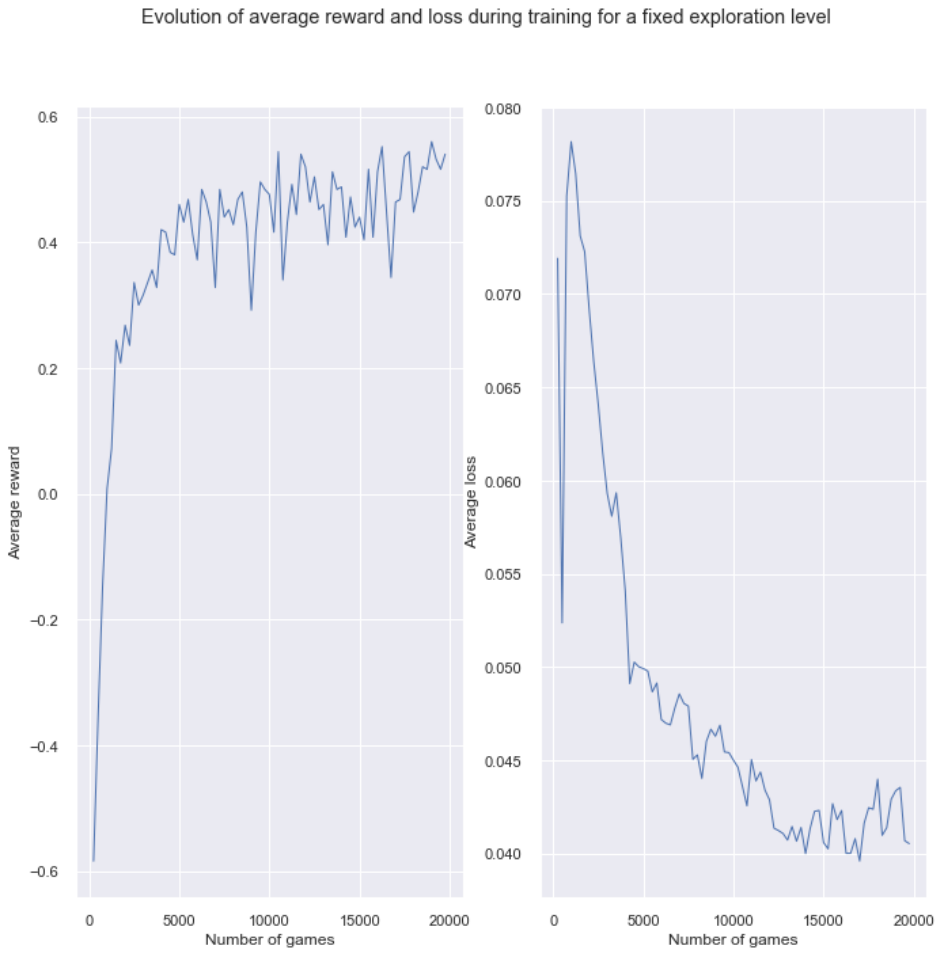


Question 10 In all of these heatmaps the player that has to play the next action is $'X'$. The first situation is an attack. Indeed, we can see that the position with the highest Q value is the position where $'X'$ could win. Interestingly, the second position with the highest Q value correspond to a position where the player would block the $'O'$ player from winning, which makes sense. The second situation is a defense. To avoid losing, the player should choose indeed the position with the highest Q value. The third situation is more complex. It is a fork scenario where the player can choose between two different positions, and any of these two choices will ensure he wins in the next turn. Indeed, the two highest Q values indicate these two positions.

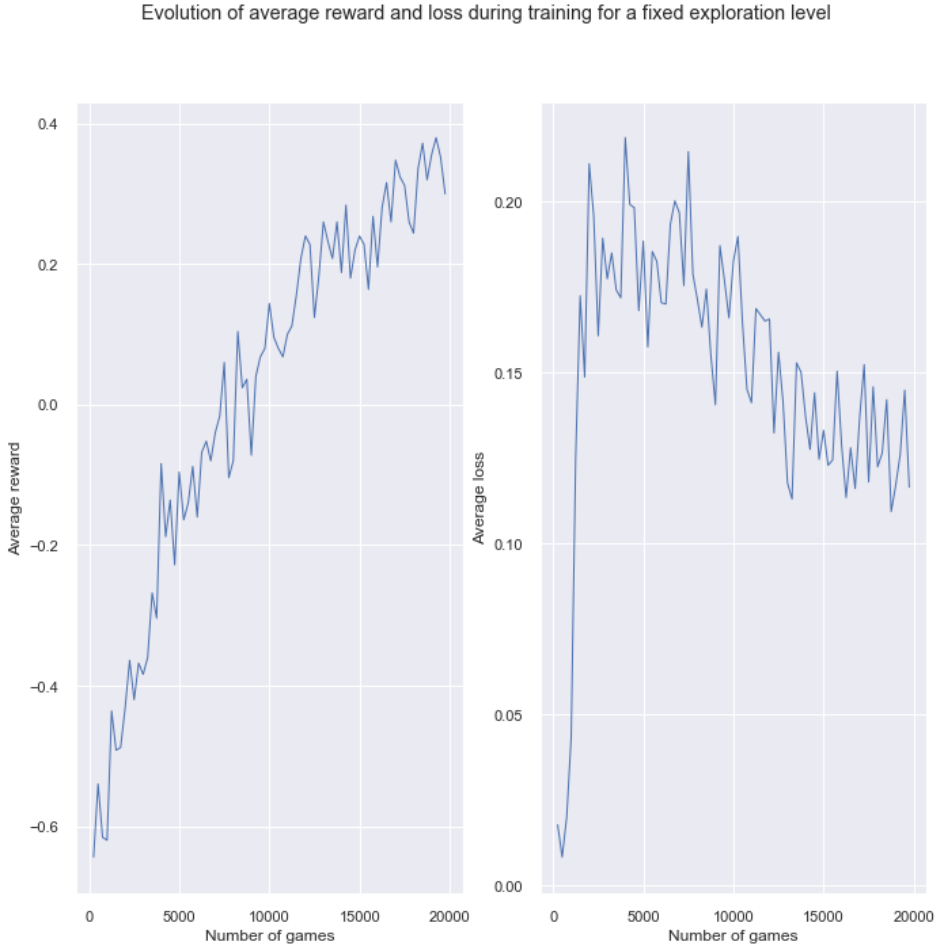
3 Deep Q-Learning

NOTE: For all the following graphs we use a learning rate a bit smaller than the one given as a starting point: $1 * 10^{-4}$.

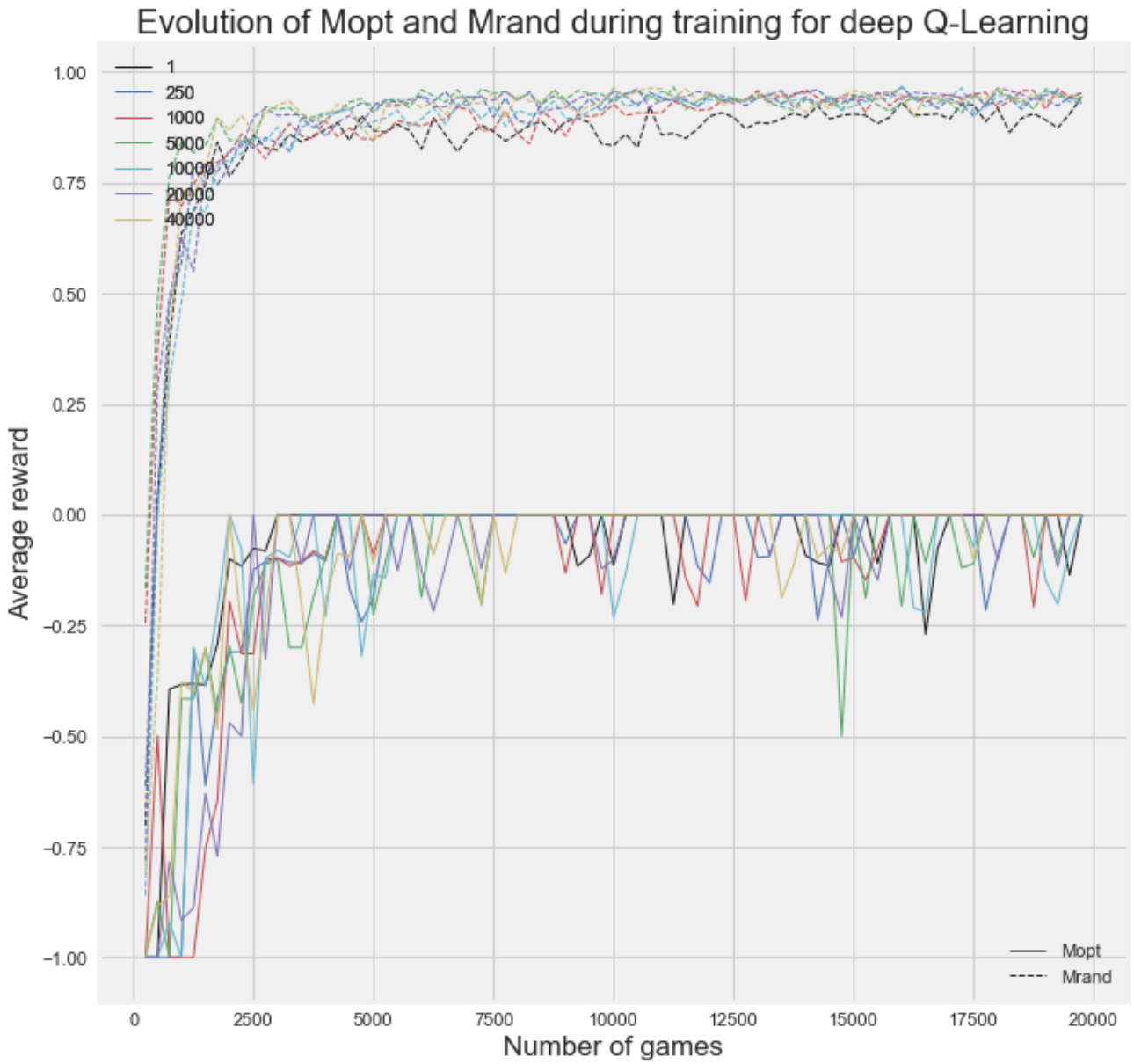
3.1 Learning from experts



Question 11 $\epsilon = 0.1$ The average reward increases quickly, indicating that the agent learns. The average loss also decreases drastically. We can see that both the loss and the average reward seem to reach a plateau. NOTE: the negative loss peak at the beginning is due to the first filling of the buffer.



Question 12 The agent struggles to learn and learns slower. Since the buffer only contains the last transition, the learning will be less stable. At each step, the model will tend to overfit the current board situation.



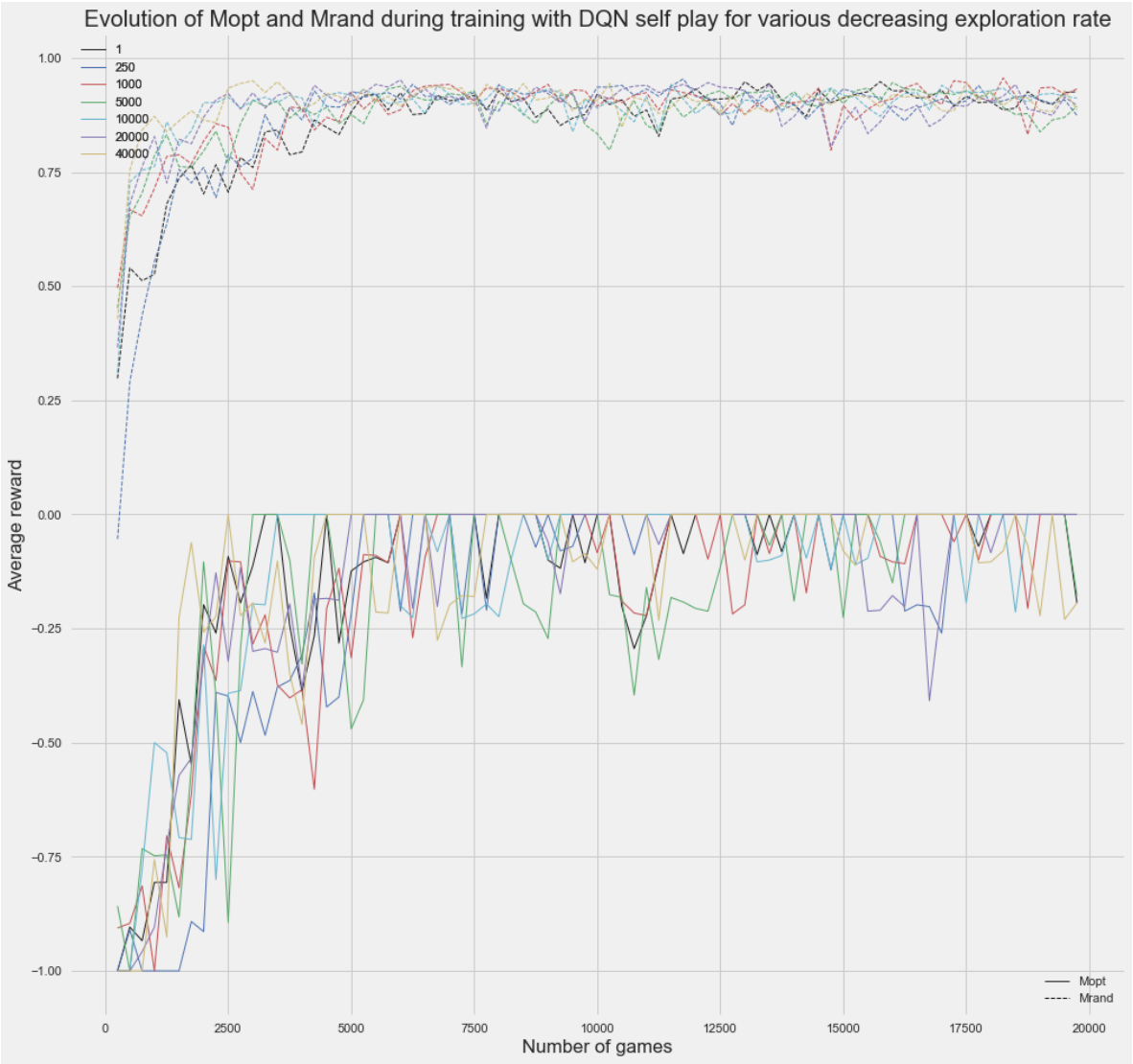
Question 13 We can see that for any decreasing rate the agent learns, suggesting that the exploration scheme is not as important in Deep Q-Learning, since now we use a neural network to predict the Qvalues. The performances on Mopt are good and reach 0 for all the decreasing rates, however, none of the agents is as stable as in Q learning. Here there is continuously more noise. For Mrand, all decreasing rates work well, except for the $n^* = 0$ which is a little worse. This makes sense because training with a highly decreasing rate, is more similar to playing against an almost optimal player. Thus, results when playing against a random player would be worse, for the same reasons as in the Q-learning part.

Question 14 We can see that the choice of ϵ_{opt} has a big impact on the learning. An ϵ_{opt} of 0.4 seems the have good results for both the Mopt and Mrand. From the plot in Question 13, we can see that $\epsilon_{opt} = 0.5$ performed even better.
Interestingly, for a training against $\epsilon_{opt} = 0$ the agent doesn't manage to win against a random player. This is again, because an optimal player only plays really specific moves compared to a random player, and thus the model can't generalize. The average reward Mrand is negative over the 20000 games. It doesn't perform well against an optimal player either. However, lower ϵ_{opt} seem to lead to better results against an optimal player, which makes sense.
Similarly, for a training against a random player $\epsilon_{opt} = 1$, the agent struggles to win against an optimal player (low Mopt). This is true for low ϵ_{opt} values. We can also see that the learning in general seems to be more unstable for really slow values and really high values of ϵ_{opt}. Therefore, middle values such as 0.4 or 0.5 are better. However, even with these middle values, the learning is less stable than with Q-learning, specially for Mopt.

Question 15 The maximum value of Mopt is 0.0 (for several ϵ_{opt}) and 0.788 for Mrand (for $\epsilon_{opt} = 0.8$)

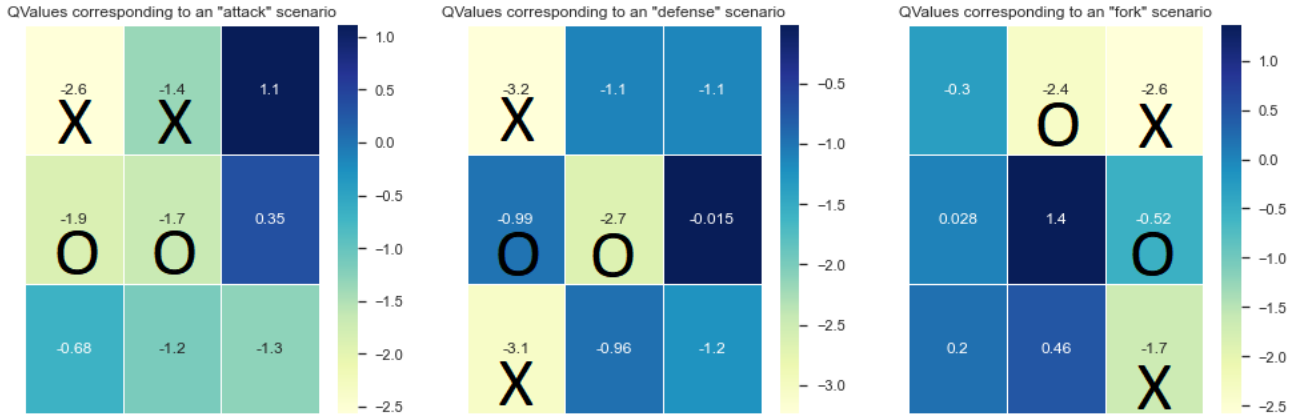
3.2 Learning by self-practice

Question 16 When training with optimal players $\epsilon = 0$, or random players $\epsilon = 1$ the agent doesn't seem to learn. Results are bad when testing against an optimal player and a random player with the twp training approaches. Interestingly, for $\epsilon = 1$, the agent doesn't even learn to avoid unavailable actions, since Mrand is negative. Therefore the balance between exploration and exploitation is important. For middle values ϵ we obtain good results both for Mopt and Mrand



Question 17 With any exploration rate the agent learns.Results seems to be almost independent of the exploration rate. However, we can notice that the average reward is less stable than with Q-Learing, and specially for Mopt.

Question 18 The maximum value of Mopt is 0.0 (for several decreasing rates) and for Mrand is 0.956 (for $n^\star = 1000$).



Question 19 In all of these heatmaps, the player that has to play the next action is X. The first situation is an attack action. Indeed, the position with the highest Q value is the position where X could win. Interestingly, the second position with the highest Q value is a position where the player would block the O player from winning, which makes sense. The second situation is a defense. To avoid losing the player should choose indeed the position with the highest Q-value. The third situation is more complex. It is a fork scenario where the player can choose between two different positions, and any of these two actions (4 or 6) will ensure he wins in the next turn. However, the two highest Q values proposed are actions 4 and 7. The agent manages to predict position 4 was better but not 6. This result is nevertheless already really good. Position 4 will make the agent win and playing in the center gives more possibilities for the next round. This shows that our algorithm generalizes well. Note that in all the situations presented, the non-available actions have low values. The agent learned not to play them.

4 Comparing Q-Learning with Deep Q-Learning

Question 20

Agent type	Mopt	Mrand	Training time
Q-L	0.0	0.904	3000
Q-L$_{self}$	-0.066	0.796	3000
DQN	0.0	0.968	1500
DQN$_{self}$	0.0	0.956	2500

Question 21 First of all, in order to obtain the results presented in this project we used a learning rate of $1 * 10^{-4}$, lower than the one given as a starting point. Moreover, we applied a clipping between (-1,1) of the gradient as suggested in the pytorch link given in the assignment.

From the table above, we can conclude that both Q-Learning and DQN agents managed to learn, including during selfplay. We can see from the training times that overall, DQN learned faster than Q-learning. For Q learning each unvisited state, will have null Qvalues, so the agent won't know what to play. Whereas, for DQN, the model will extrapolate Q values even for unvisited states. Therefore, it needs a less systematic search of the states landscape, and thus leads to faster convergence.

However, the average rewards obtained with Q-Learning converged towards more stable values than the ones obtained with DQN. The goal of using a Deep-Q network is to approximate the state graph of Q-Learning. Meaning that, once all the states in the graph have been visited by Q-Learning, Q-Learning is the best approximation of itself, and will be more stable after convergence than DQN.

Looking at the maximum average reward when playing against the optimal player, we see that selfplay Q-Learning doesn't learn as well as simple Q-Learning. This might be because with self playing, states are explored in a more hazardous way . That has a strong impact on Q-Learning, but not on DQN, due to the generalization power of deep networks (that we already discussed above).