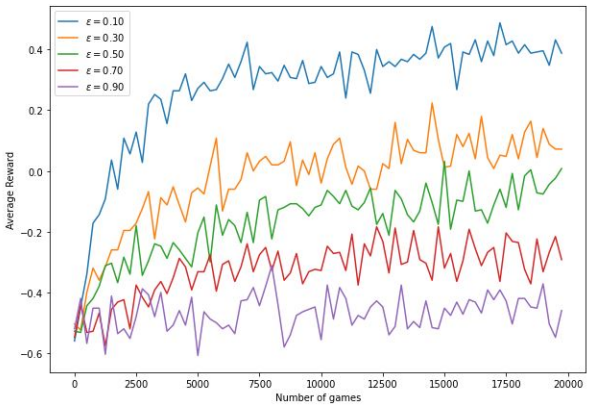
Mini-project 1: Tic Tac Toe

Jeremy Goumaz, Laurent Gürtler
CS-456 Artificial Neural Network, EPFL

I. Q-Learning

A. Learning from experts

Question 1: Taking into account different value of ϵ, we can see that the average reward depends on this value. The lower the ϵ is, the better the average reward will be, because the agent learns better against a good player which confirms that the agent learns to play Tic Tac Toe.



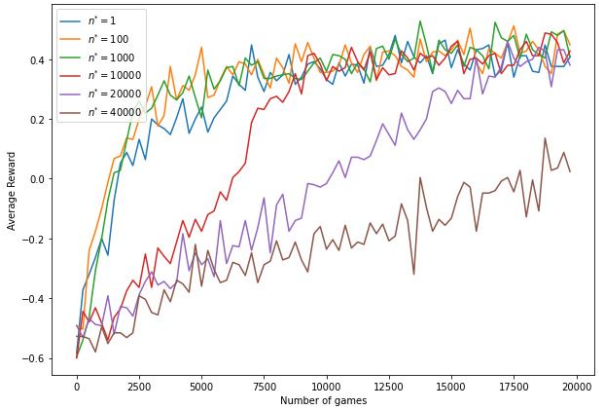
Q1: Average reward for different ϵ.

A1. *Decreasing exploration:*

$$\epsilon(n) = \max\left\{\epsilon_{min}, \epsilon_{max}\left(1 - \frac{n}{n^*}\right)\right\}$$

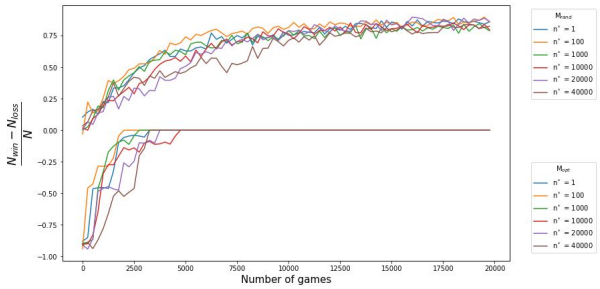
Question 2: Indeed, decreasing ϵ helps the agent to train the agent faster and more efficiently. By changing the value of n*, it decreases linearly the value of ϵ over time. For n* = 1, the value of ϵ decreases instantly to ϵ_{min} = 0.1 which is exactly the agent that we had for the question 1 with ϵ = 0.1 (which was a good agent). We can observe on the plot that higher n^* gives poorer results and slower convergence to good results. We can also observe that for lower n^* (between 1 and 1000), we have a similar quick convergence to a good average reward. We can conclude that the best n^* is probably 100 in our case.

Question 3: The test performance shows once again that the agent learns to play Tic Tac Toe. As we can see,



Q2: Average reward for different n^*.

after several games played, all the agents with different values of n* playing against Opt(0) obtain only ties. Against Opt(1), the agent reaches a high score of wins after several games, independently of the value of n* (not only wins because randomness induces ties). The difference with the last curves is that every n* value converges approximately to the same score with M_{opt}, M_{rand}, which is a better way to test the performance of the agent.

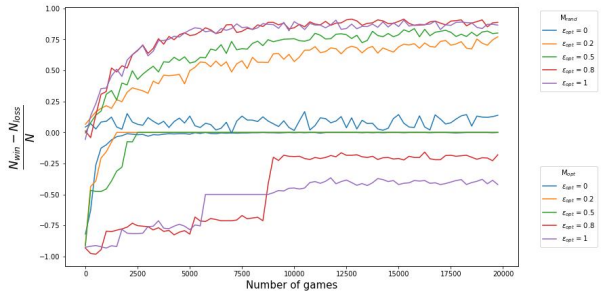


Q3: M_{rand} and M_{opt} for different n^*.

A2. *Good experts and bad experts:*

Question 4: The best n* found in the last part is 100. In this part, the agents will be trained against players with different $M_{opt} \in [0, 1)$. M_{opt} converges rapidly to zero with ϵ_{opt} = [0, 0.2, 0.5], which means a

majority of ties for the agent. But with ϵ_{opt} = [0.8, 1], the agent struggles to achieve ties against the player. This is due to the fact that training against a bad player won't help the agent to play good or tie against a good player. More precisely, the good player will play moves that the agent didn't learn during the training.
For M_{rand}, this is the inverse that happens. ϵ_{opt} = [0.5, 0.8, 1] get good M_{rand} scores contrary to ϵ_{opt} = [0, 0.2]. We can observe that if the agent trains against a player with random moves, he will also perform good in the test versus a player with random moves.
We can see that there is a trade-off between training against good players and bad players. Thus, the most optimal approach is to train against a player with a middle ϵ_{opt} = 0.5 as we do in the other sections. In conclusion, training against a random player makes the agent good against random players and training against good players makes the agent good against good players.



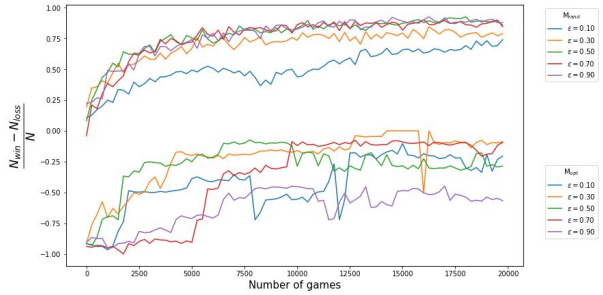
Q4: M_{rand} and M_{opt} for different ϵ_{opt}.

Question 5: The highest value of M_{opt} is 0. This is consistent with the fact that the agent is not supposed to win against a player with optimal moves and it leads to only ties.
The highest value of M_{rand} is 0.914. This means that the agent wins most games against a player with random moves. Note that it is impossible to achieve 1 because ties are frequent in Tic Tac Toe even with random moves.

Question 6: $Q_1(s,a)$ and $Q_2(s,a)$ have not the same values. As we have seen in the previous questions, learning against a random player (such as Opt(1)) makes the agent play specifically well against random players but not against good players. In the opposite way, learning against a good player (such as Opt(0)) makes the agent play specifically well against good players but not against random players. Therefore Agent 1 and Agent 2 Q-values will converge to different optimal values (optimized either against random players or good players) which makes them different. This is why it is important to use a trade-off and train against Opt(0.5) or similar.

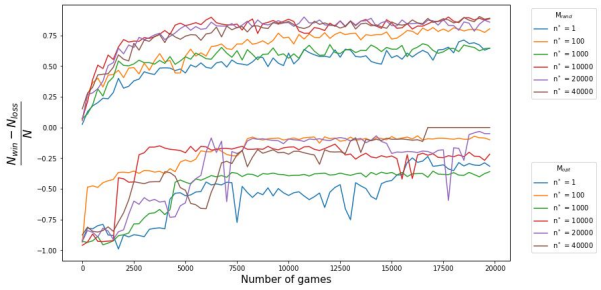
B. Learning by self-practice

Question 7: Yes, the agent learns to play correctly from self-practice but is less good against Opt(0.0) compared to the learning from experts. We can observe more oscillations in M_{opt} and the agent has more difficulty to approach $M_{opt} = 0$. It is mainly due to the fact that the agent plays against itself and has fewer opportunities to learn the "optimal" plays. Lower ϵ leads to lower M_{rand} and higher ϵ leads to lower M_{opt} (trade-off). It seems that the best ϵ is in the middle between 0.3 and 0.7.



Q7: M_{rand} and M_{opt} for different ϵ.

Question 8: There are not many differences with the plot of Question 7, but we can observe that small n^* lead to bad results in M_{rand} in the same way as low ϵ which is logical (low n^* means a low ϵ reached quickly). Higher n^* also seems to lead to better results in M_{opt} (M_{opt} is better when ϵ decreases). From our results, we can conclude that high n^* (10000 to 40000) helps training compared to fixed ϵ.

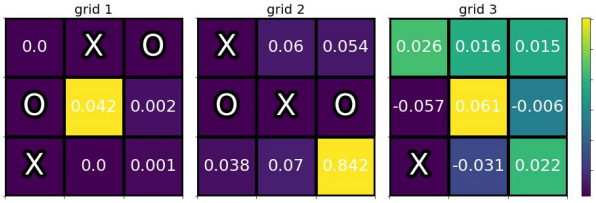


Q8: M_{rand} and M_{opt} for different n^*.

Question 9: Highest M_{opt} achieved is 0 which means only ties against Opt(0) and highest M_{rand} achieved is 0.904 which is slightly better than the last part (learning

against experts) which means that self-practice is a very good way for training an agent for Tic Tac Toe.

Question 10: The results make sense and the agent learned the game well. Indeed we can observe that the Q-values are valuating good actions in all the three boards. In the first one, we can observe that the higher value is in the middle (probably the best way to get a tie since it seems difficult to win). In the second board, it's the turn of X to play and by playing the best Q-value action, it leads to a win which is a proof of correctly handled reward-learning. In the third board, we are in an opening position and a lot of positions are equivalent but the middle seems to be the more attractive position which is correct. It seems that the agent learned correctly the game since high Q-value are linked to good positions in these cases. We can interpret positive Q-values as winning positions and negative Q-values as losing positions. Q-values close to 0 are linked to neutral positions (leading to ties). In our three boards, positive and negative Q-values are both presents which means that the agent learned correctly when winning or losing.



Q10: Q-values for three board arrangements for Q-Learning.

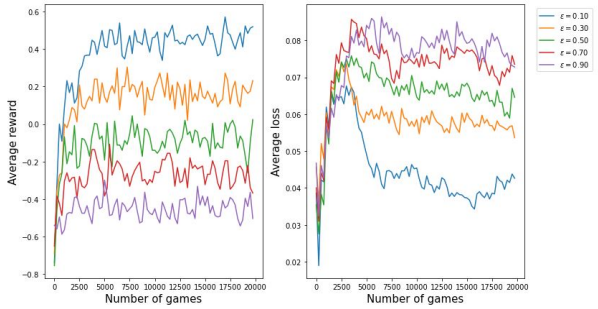
II. Deep Q-Learning

A. Learning from experts

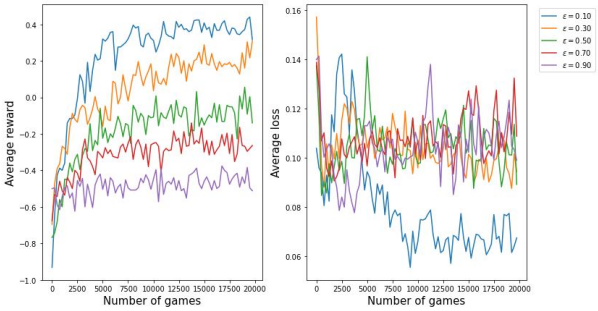
Question 11: Yes, the loss decreases but after an initial increase in the first 5000 games (probably due to the time it takes to stabilize memory buffer and target policy). The average reward also quickly stabilizes. In conclusion, the agent learn to play correctly. The best ϵ is 0.1.

Question 12: We can observe that the performances are much worse. The average rewards are lower, the average losses are much higher and the losses oscillate a lot (due to the variance of batch size = 1). However we can still observe the same trends in the plots but the convergence seems slow.

Question 13: It seems that decreasing ϵ does not really help compared to fixed ϵ. As we can observe on the plots, it seems that n^* has no real effect on



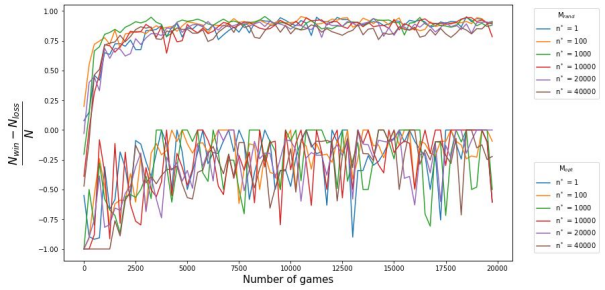
Q11: Average reward and average loss for different ϵ.



Q12: Average reward and average loss for different ϵ without replay buffer.

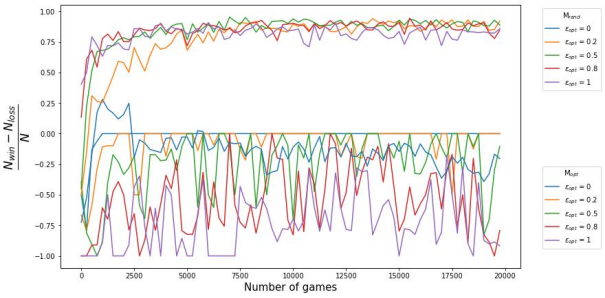
M_{opt} and M_{rand}. Whatever n^* is, the performances are nearly the same. Since we know that a very small n^* is approximately equivalent to a fixed ϵ (after n^* games), we can safely state that fixed ϵ (very low n^*) and decreasing ϵ (higher n^*) seem to give the same results. n^* seems to have a minor effect on M_{opt} and M_{rand} since all plots are approximately superimposed. In the case of M_{rand}, we can observe a very good convergence to a high score and in the case of M_{opt}, we can observe a relatively similar behaviour independently of n^* (tendency to flirt with M_{opt}=0 with a lot of variance and oscillations). Regardless of n^*, the results are very good and it is one more proof that this Deep Q-Learning Network can definitely learn to play Tic Tac Toe.
Note: higher n^* leads to more exploratory games with more randomness (on the contrary, lower n^* leads to less exploratory games). But here it does not seem to matter a lot.

Question 14: For M_{rand}, we can observe very good results with $\epsilon_{opt} \in [0.2, 0.5, 0.8, 1.0]$ but very bad results for $\epsilon_{opt} = 0.0$. It is consistent with the fact that training against a random player helps to be better against random players and leads to a higher M_{rand}. The specific case of $\epsilon_{opt} = 0.0$ gives very bad results because the training is



Q13: M_{rand} and M_{opt} for different n^*.

done against a perfectly optimal policy ($\epsilon_{opt} = 0.0$ means absolutely no randomness and no exploratory games). An agent trained against a perfectly optimal policy will not be able to do anything if the opponent makes bad moves (occurring with random moves) and will even lose some games (often due to illegal moves). The losses explain the negative M_{rand} shown in the plot for $\epsilon_{opt} = 0.0$. For M_{opt}, we can observe the opposite ($\epsilon_{opt} = 0.0$ gives the best M_{opt}). Indeed, if we train an agent with a lower ϵ_{opt} (which means more optimal moves), we will end up with an agent performing very well against optimal policies like Opt(0) which will lead to very good M_{opt} around 0 (only ties). It means that lower ϵ_{opt} give better M_{opt} results and higher ϵ_{opt} give poorer M_{opt} results.

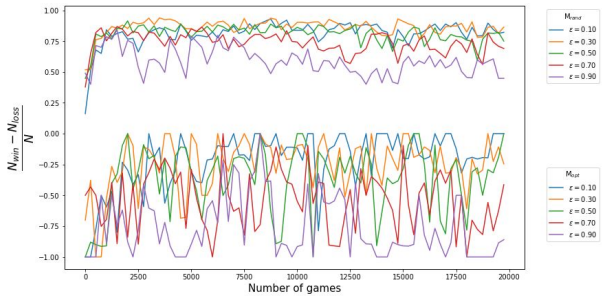


Q14: M_{rand} and M_{opt} for different ϵ_{opt}.

Question 15: Like the previous cases, the best M_{opt} is still 0. However, the best M_{rand} took a leap thanks to Deep Q-Learning and has reached a maximum of 0.954 which is very high. It means that Deep Q-Learning (trained against experts) is very efficient in our case and gives very good results on Tic Tac Toe.

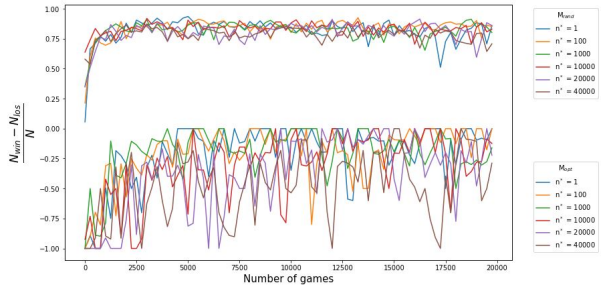
B. Learning by self-practice

Question 16: Yes, the agent learns to play well as shown in the plots (M_{rand} around 0.75 and M_{opt} sometimes reaching 0). The agent seems to learn a lot in the first games and quickly stabilizes (after only 2500 games). We can observe that lower ϵ gives better results on the two metrics and higher ϵ is less good. It is due to the fact that higher ϵ induces more randomness. Since the agent is self-practising against itself, it means that if he plays randomly he will not learn anything. Therefore lower ϵ with fewer random moves are better.



Q16: M_{rand} and M_{opt} for different ϵ.

Question 17: Yes, decreasing ϵ helps compared to fixed ϵ. It can be easily observed in the plots. Decreasing ϵ improve the values of M_{rand} and M_{opt} and also helps to stabilize the results with a better convergence (due to ϵ_{min}=0.1). Decreasing ϵ is efficient because the agent ends (after some games) with a very low ϵ which is good for self-practising. n^* seems to have little effect on the final score but slightly modifies the convergence speed. Exploratory games are not very useful for self-practising here, therefore n^* has not a great impact on the model.

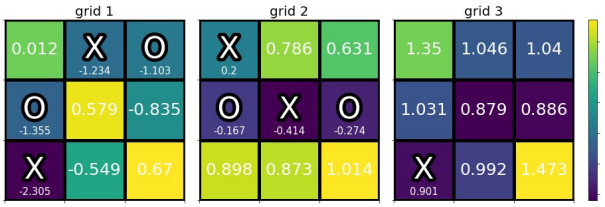


Q17: M_{rand} and M_{opt} for different n^*.

Question 18: As always, the best M_{opt} is 0. For M_{rand}, the best is 0.936 which is slightly lower than Deep Q-Learning trained by experts. But this result is still very good and it means that the model is able to learn very well by itself.

Question 19: We can observe that the results make sense in most of the cases. We can observe some very negative Q-values where the positions are taken which means that the model has learned to play legal moves and

to avoid positions already taken. The first and the second board have relatively good Q-values (approximately similar to the ones found in Question 10 with some variability). It seems that the third board has not been learned very well but it is probably due to the fact that this board is an early game and it takes more time to learn the earlier stages of the game (compared to the late stages where the reward is given to the model). Nevertheless, it seems that the Deep Q-Learning Network has brilliantly learned Tic Tac Toe and is able to play well and avoid illegal moves. The network is also able to win as shown in the second board which is another proof of its success.



Q19: Q-values for three board arrangements for Deep Q-Learning.

III. Comparing Q-Learning with Deep Q-Learning

Question 20:

Methods	**Learn from experts**			**Learn from self-practice**		
	M_{opt}	M_{rand}	T_{train}	M_{opt}	M_{rand}	T_{train}
Q-Learning	0.0	0.914	3750	0.0	0.904	3500
Deep Q-Learning	0.0	0.954	750	0.0	0.936	1500

TABLE I: Table (Q20) with result comparisons between the different models.

Question 21: All the models we have created gave us very good results. We got $M_{opt} = 0$ and more than $M_{rand} = 0.9$ for the four models. During our tests, that 'Deep Q-Learning' gave us better results than 'Q-Learning' and 'learning from experts' gave us better results than 'learning from self-practice'. It means that our best model is a Deep Q-Learning Network with 'training by experts' (OptimalPlayer). We can also notice that T_{train} is greater for Q-Learning models than for Deep Q-Learning models. It is very likely that the basic Q-Learning algorithm takes more time and updates to reach good results than Deep Q-Learning algorithm. It is consistent with the complexities of the models and the fact that Deep Q-Learning model is an improved Q-Learning model using neural networks.

A possible way to improve the model would be to improve the metrics functions. Indeed, we observed that when playing against Opt(0) and Opt(1) for metrics evaluations, most of the games played were similar (two deterministic models against each other often lead to the same games and the same results). Optimizing the metrics would be a good idea to analyze the results.

There are a lot of other possibilities to improve these models which are not assessed in this project. We only tried one network architecture for Deep Q-Learning but we could have tried many other architectures with different tuning setups to improve the results. Another point is that we have not really tested many given parameters (initial Q-values, learning rate, discount factor, buffer size, batch size and many more). Nevertheless, the results are very good and the Q-Learning models have clearly been able to learn to play Tic Tac Toe with success.