ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

PROJECT 1: TIC TAC TOE

CS-456 ARTIFICIAL NEURAL NETWORKS

Yingxue Yu
Siran Li

6th June 2022

1 Q-Learning

Question 1

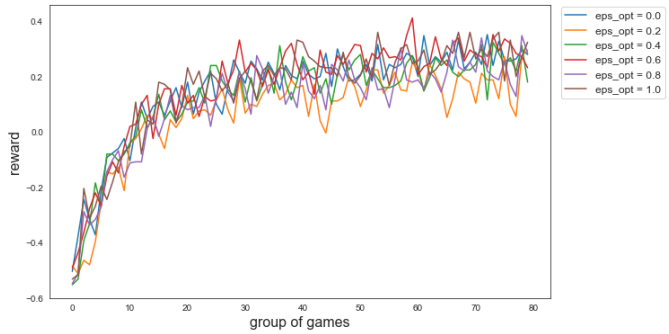


Figure 1: The agent learns to play Tic Tac Toe. The average rewards increase over time and get close to 0.4 with different ϵ in $[0.0, 0.2, 0.4, 0.8, 1.0]$.

Question 2

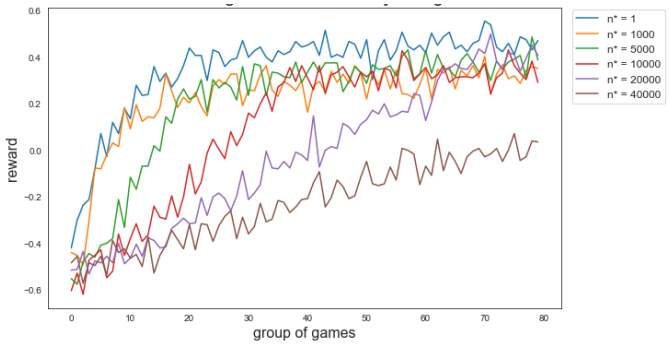


Figure 2: With different n^*, ϵ decreases as the epoch increases. With increasing n^*, the convergence time becomes longer. The best n^* found in the plot are 1 and 20000.

Question 3

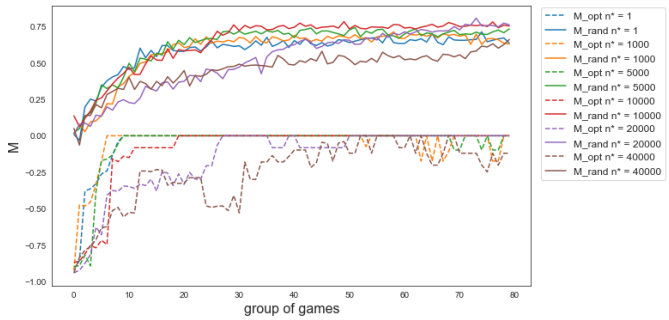


Figure 3: The curve of M_{opt}, M_{rand} are more stable and smooth compared with the curve of average rewards. M_{opt}, M_{rand} and average rewards can all reflect the performance of the training model.

Question 4

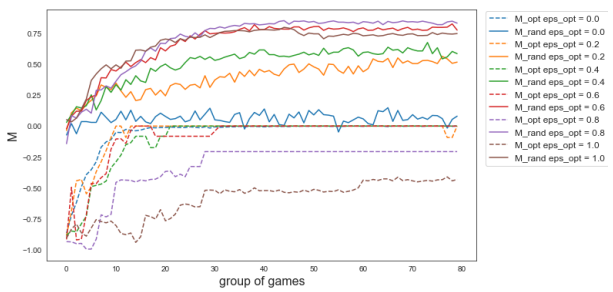


Figure 4: With $n^* = 10000$ and smaller $Opt(\epsilon_{opt})$, the 'test' curve for M_{opt} converges faster, and reaches 0. While for M_{rand}, with $Opt(\epsilon_{opt})$ over 0.5, the M_{opt} are higher, and the best $Opt(\epsilon_{opt})$ for M_{opt} is 0.8. If the agent trains more with optimal expert, it can perform better against an optimal player than practicing with a random decision player, vice versa.

Question 5 From the above result, the highest M_{opt} is 0.85, and highest M_{rand} is 0.0.

Question 6

$Q_1(s,a)$ and $Q_2(s,a)$ should have different values, since they would have different rewards in the same states based on their opponents' actions. After we update the Q values according to $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q\left(S', a\right) - Q(S,A)\right]$ the obtained $Q_1(s,a)$ and $Q_2(s,a)$ should also be different.

Question 7

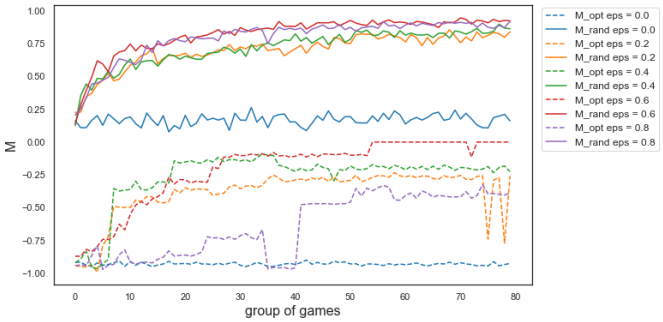


Figure 5: The agent is learning to play. Increasing ϵ in $[0.0, 0.2, 0.4, 0.8]$, both M_{opt} and M_{rand} first increase and then decrease.M_{opt} and M_{rand} are highest when $\epsilon = 0.6$. ϵ can help balance the model's training process. If it is too low, the model overfits the optimal policy, while if it's too high, the model is underfitting.

Question 8

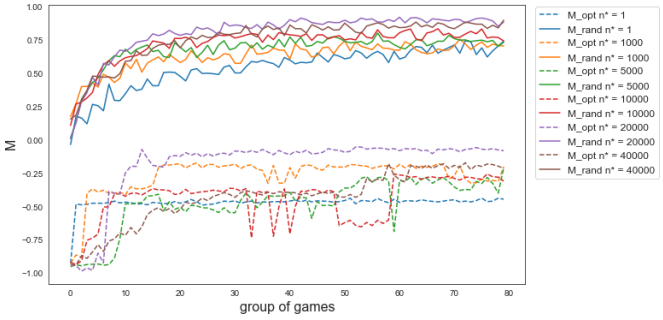
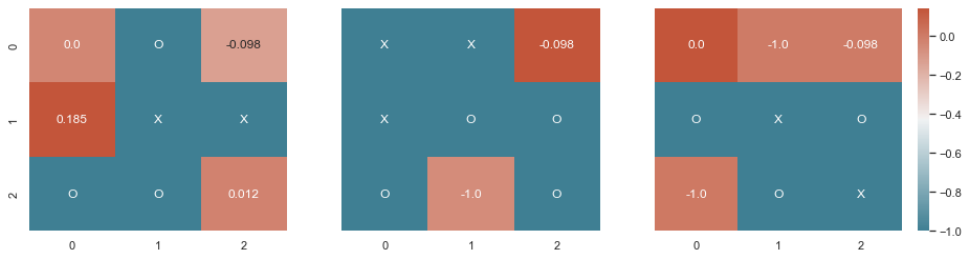


Figure 6: Decreasing ϵ during a moderate number of phases improved performance. Increasing n^* in $[1, 1000, 5000, 10000, 20000, 40000]$, both 'test' M_{opt} and M_{rand} first increase and then decrease. With $n^* = 20000$, the M_{opt} and M_{rand} are highest. The higher the n^*, the longer the exploration time.

Question 9 During the experiments, the obtained curves are random as with different random seeds. Generally speaking, the highest values got from fixed $Opt(\epsilon)$ and decreasing ϵ methods are very similar. While for M_{opt} curves from the decreasing ϵ method are more smooth and stable. The highest M_{opt} is 0.00, when $Opt(\epsilon)$ is fixed to 0.6, and for both fixed $Opt(\epsilon)$ and decreasing ϵ method, we get the same highest M_{rand}: 0.948.

Question 10

Figure 7: We used our best training model described in Question 9 for getting the best Q-values. The results make sense, since it gave the highest Q-values for the winning positions.



2 DEEP Q-LEARNING

Question 11

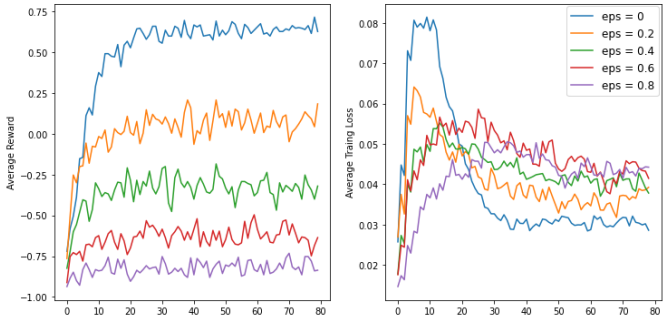


Figure 8: The loss first increases for a short period and then decreases. The reward increases over time and reaches an average of 0.7 when ϵ is 0. The agent has learned to play the game over time.

Question 12

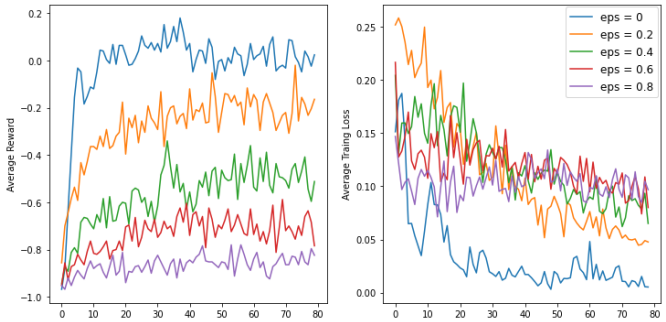


Figure 9: The loss still decreases and the reward still increases. The best ϵ is still 0. But the performance is worse, i.e. it has lower reward and higher training loss.

Question 13

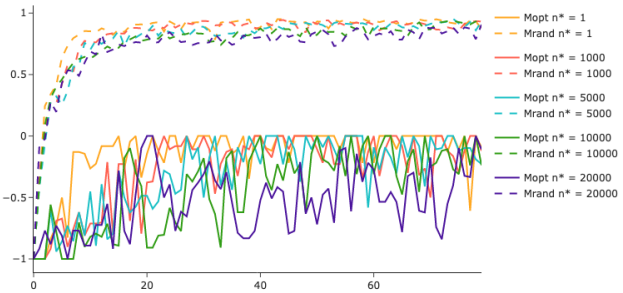


Figure 10: In the plot, $n^* = 1$ performs the best. Decreasing ϵ doesn't seem to help training compared with a fixed ϵ.Increasing n^* does not improve, if not worsen the performance. Higher n^* means more exploration time instead of exploitation time. Thus, it takes longer to reach the optimal performance against both $Opt(1)$ and $Opt(0)$ with larger n^*.

Question 14

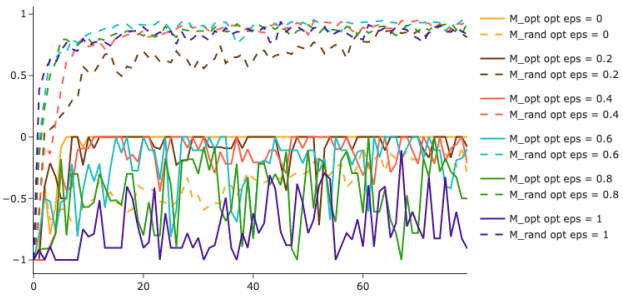


Figure 11: n^* is chosen to be 0. M_{opt} gets higher as ϵ_{opt} gets lower. Because the agent needs to learn from a competent player first to achieve good performance against the optimal player. M_{rand} is high as long as we don't learn from $Opt(0)$. Because there's no randomness in $Opt(0)$, our agent can't learn from it how to deal with a completely random player.

Question 15 The highest M_{opt} we could achieve is 0. And the highest M_{rand} 0.954 is achieved by setting $\epsilon = 0.1$ with $Opt(0.5)$.

Question 16

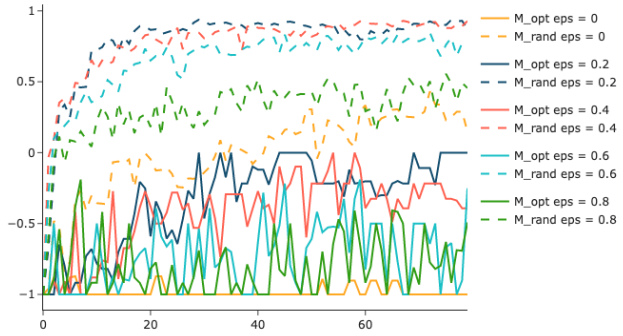


Figure 12: M_{opt} and M_{rand} increase over time. The agent is learning to play. When ϵ is too small or too big, there is not a right balance between exploration or exploitation, thus the agent is not learning well.

Question 17

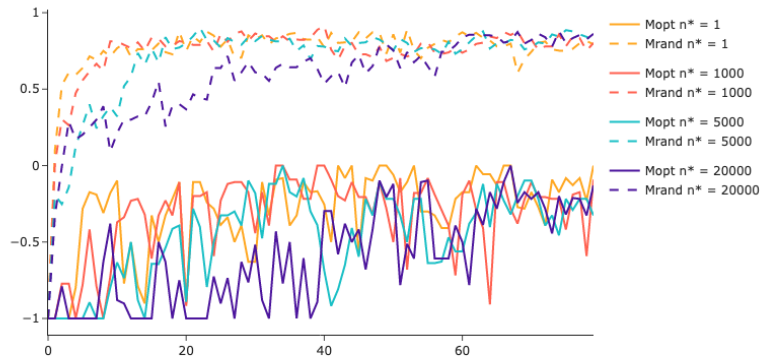
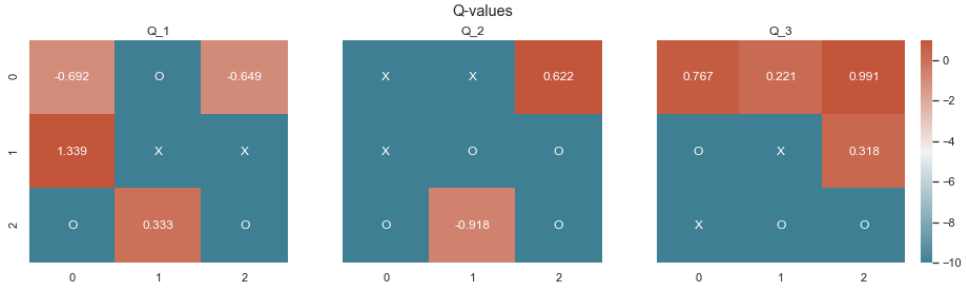


Figure 13: Decreasing ϵ seems to help achieve a higher M_{opt} compared with a fixed ϵ when n^* is small($n^* < 5000$). Higher n^* means more exploration time instead of exploitation time. Thus, it takes longer to reach the optimal performance against both $Opt(1)$ and $Opt(0)$ with larger n^*.

Question 18 The highest M_{opt} we could achieve is 0. And the highest M_{rand} 0.914 is achieved by decreasing exploration with $n^* = 1000$ and $Opt(0.4)$.

Question 19

Figure 14: We used the best model described in Question 18 to get the Q-values. For three board arrangements, Q-values at available positions are visualized. The results make sense, since it gave the highest Q-values for the winning positions.



3 Comparing Q-Learning with Deep Q-Learning

Question 20

	Q-Expert	Q-Self	DeepQ-Expert	DeepQ-Self
M_{rand}	0.85	0.948	0.954	0.914
M_{opt}	0	0	0	0
Training Time	6000	4000	750	3000

Table 1: Best Performance of Q-Learning and DQN.

Question 21

From Q20, we can see that DQN learns more quickly than Q-learning, and learns better with experts.

Q-learning, on the other hand, achieves better performance when it learns by playing with itself. From experiment plots, we can see its performance is more stable compared with DQN.

Q-learning and DQN share many similarities. They both learn better how to play with the optimal player when learning more with the optimal player. And they both play badly against the random player when only learning from the optimal player. They both performs better with a ϵ that is neither too big nor too small. During self-learning, setting the right n^* can help improve performance of both.