Project 1 Report: Tic Tac Toe

Yihang Chen (343713) [*1] **Yifei Song (335187)** [*1]

1. Q Learning

- **Answer 1**. See Figure 1. The agent learns to play Tic Tac Toe.

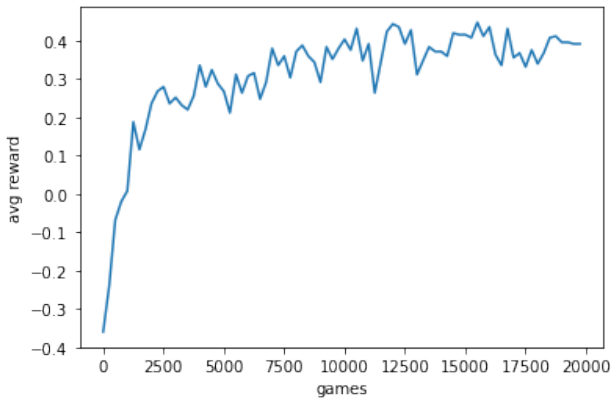

Figure 1. Average reward for every 250 games during training when fixing $\varepsilon = 0.2$.

- **Answer 2**. See Figure 2. ε is decaying from $\varepsilon_{\max}$ to $\varepsilon_{\min}$ during the first $n^\star$ steps linearly and for the rest, $\varepsilon = \varepsilon_{\min}$. Effect of $n^\star$: controls the speed of ε's decaying: the larger $n^\star$ is, the faster ε will decay to $\varepsilon_{\min}$. Decreasing ε helps training compared to having a fixed one. Setting $n^\star = 1$ amounts to fixing $\varepsilon = \varepsilon_{\min}$, whose performance takes the lead only during the early phase of training. Setting $n^\star$ to be too large, i.e. ε cannot be decayed to $\varepsilon_{\min}$ woll hinder the performance. Setting $\varepsilon = 10000, 20000$ will have better training reward in average.

- **Answer 3**. See Figure 3. Similarity: the average training and test reward will both increase during training. Difference: the agent can be guaranteed not to lose to the optimal player in the early stage of training. However, the agent is not optimal since the average reward against a random player is fluctuating. Different $n^\star$:

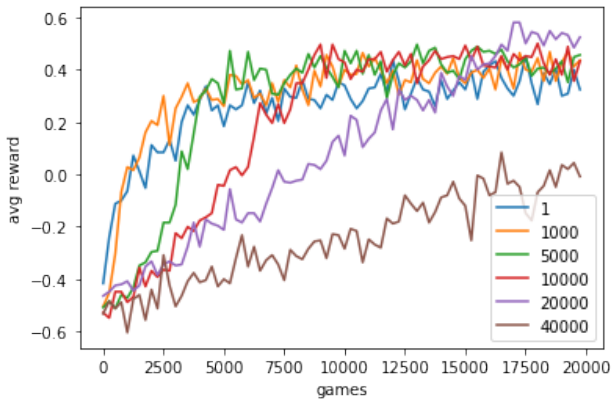

Figure 2. Average reward for every 250 games during training for different values $n^\star = \{1, 1000, 5000, 10000, 20000, 400000\}$.

lower $n^\star$ achieves better performance in the early training stage, middle size $n^\star$ achieves better performance in the later stage. Setting $n^\star$ too large is not beneficial as well.

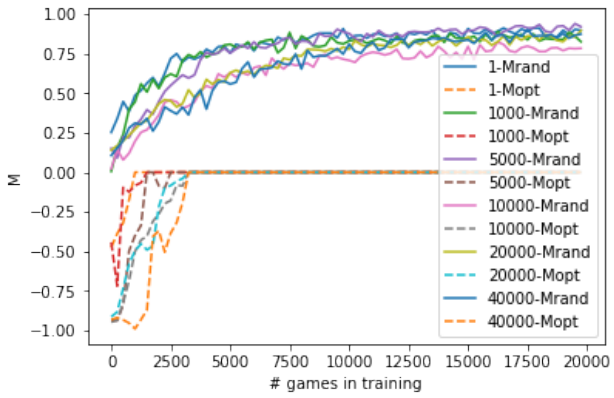

Figure 3. $M_{\text{rand}}, M_{\text{opt}}$ over time for different values of $n^\star = \{1, 1000, 5000, 10000, 20000, 400000\}$.

- **Answer 4**. From Figure 3, $n^\star = 20000$ is the best. We set $n^\star = 20000$ here. See Figure 4 We observe: (1) When ε increases, the player will take longer time to achieve $M_{\text{opt}} = 0$. M_{opt} cannot be zero when

[*]Equal contribution. Alphabetical Order. [1]École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. Correspondence to: Yihang Chen <yihang.chen@epfl.ch>, Yifei Song <yifei.song@epfl.ch>.

$\varepsilon_{\text{opt}} = 0$. Because it is impossible to beat the optimal player, large ε_{opt} means the Q player cannot get positive reward during training. (2) When ε decreases, the player will take longer time to achieve high M_{rand}, i.e. beat the random player. Because small ε means the opponent is random at most time, it will not learn the strategy to beat the random player from a random player.

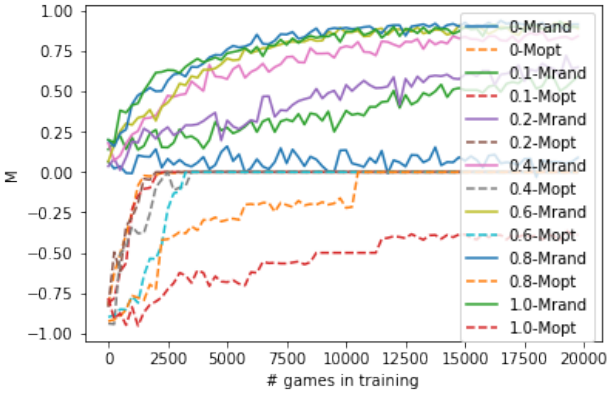

Figure 4. $M_{\text{rand}}, M_{\text{opt}}$ over time for different values of $\varepsilon_{\text{opt}} = \{0, 0.1, 0.2, 0.4, 0.6, 0.8, 1.0\}$.

- **Answer 5**. The maximum value of M_{opt} is 0.0. The maximum value of M_{rand} is 0.94.

- **Answer 6**. No. It is impossible to win a optimal adversary (get reward 1). Hence, during the learning of Q_2, the reward for the player will be no more than 0. From the update rule of Q function

$$Q^{\text{new}}(s_t, a_t) = (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \max_a Q(s_{t+1}, a_t)) \tag{1}$$

we obtain if $Q(s, a) \leq 0, \forall s, a, r_t \leq 0$, we get $Q^{\text{new}}(s, a) \leq 9, \forall s, a$. Since Q_2 is initialized to be zero, we obtain $Q_2(s, a) \leq 0, \forall s, a$.

On the other hand, it has a positive probability to win a random adversary, from which the player can get positive reward 1. If $Q_1 = Q_2 \leq 0$, then under that Q_1, the player could possibly win a random adversary, assuming the last step of the player is s_T, a_T, then $Q^{\text{new}}(s_T, a_T) - Q(s_T, a_T) = \alpha(1 - Q(s_T, a_T)) \geq \alpha$, which is contradictory to the fact that Q_1 converges.

In all, $\exists s, a, Q_1(s, a) \neq Q_2(s, a)$.

- **Answer 7**. See Figure 5. The agent learn to play Tic Tac Toe by playing against itself when setting $\varepsilon > 0$ during training. Effect of ε: ε is a balance between exploration and exploitation. When ε is too small (e.g. $\varepsilon = 0, 0.1, 0.2$ in Figure 5), the agent won't explore enough state and only obtain a local best solution; when ε is too large (e.g. $\varepsilon = 0.8$ in Figure 5), the randomization will be dominant and harm the agent's exploitation, especially against optimal player.

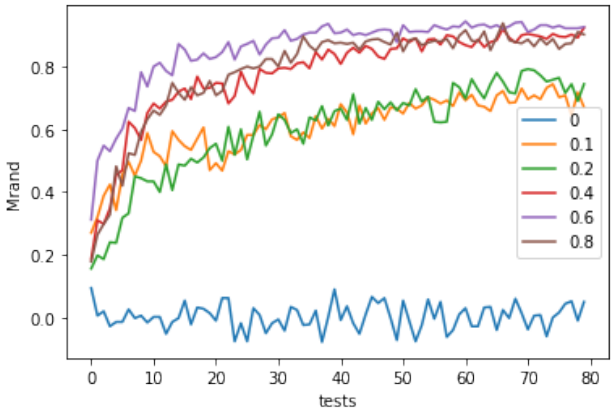

Figure 5. Self-practice: $M_{\text{rand}}, M_{\text{opt}}$ over time for different values of $\varepsilon = \{0, 0.1, 0.2, 0.4, 0.6, 0.8\}$ during training.

- **Answer 8**: See Figure 6. Decreasing ε helps training compared to having a fixed ε. Effect of $n^\star$: when $n^\star$ is too small, i.e. $\varepsilon = \varepsilon_{\min}$ for the most time, the agent is not well-explored and the performance against random adversary is bad. When n is too large, ε is not decayed enough, and the randomization will be dominant and harm the agent's exploitation, especially against optimal player. In all, the best performance would be $n^\star = 20000$.

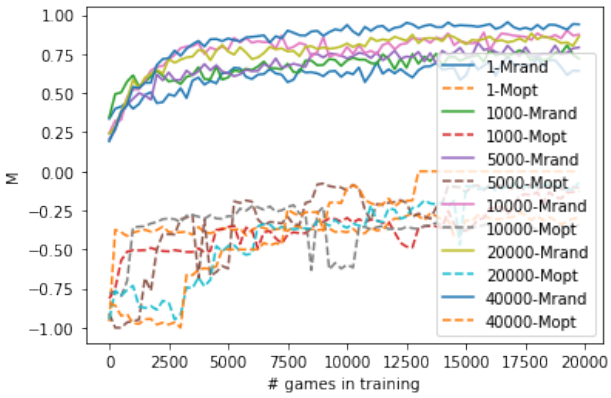

Figure 6. Self-practice: $M_{\text{rand}}, M_{\text{opt}}$ over time for different values of $n^\star = \{1, 1000, 5000, 10000, 20000, 400000\}$ during training.

- **Answer 9**. The maximum value of M_{opt} is -0.074. The maximum value of M_{rand} is 0.95.

- **Answer 10**. See Figure 7. The result largely makes sense, and the agent learn the game well. The best option is always the most reasonable one, for example, the agent should place 'X' or 'O' at the corner, which is the best choice. In case 2, when the first player choose the middle, the second one must choose the corner. However, there are some deficits: The Q some reasonable choice does not high Q-value, such as the top right corner in the first figure, which might be attributed to underexploration during self-play.

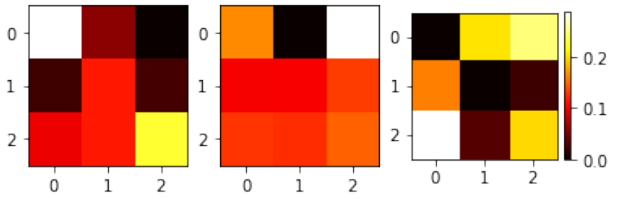

Figure 7. **1st**: Heat map of Q-values of available actions. empty grid. **2nd**: empty grid with position (1,1) being 'X'. **3st**: empty grid with position (0,0) being 'X' and position (1,1) being 'O'.

2. Deep Q Learning

- **Answer 11**: See Figure 8. The loss decreases and the agent learns to play the tic-tac-toe. The optimal choice of ε is 0.01.

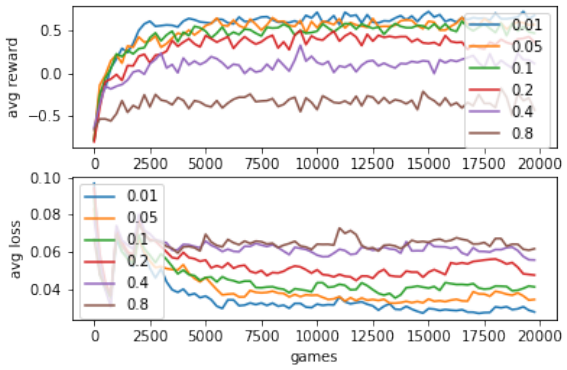

Figure 8. Average reward and average training loss for every 250 games during training when fixing $\varepsilon = 0.01, 0.05, 0.1, 0.2, 0.4, 0.8$.

- **Answer 12**: See Figure 9. We observe: The loss decreases and the agent learns to play the tic-tac-toe. But the performance of reward and loss are not as good as using the buffer and small batch. The average reward is much less and the average loss is much more.

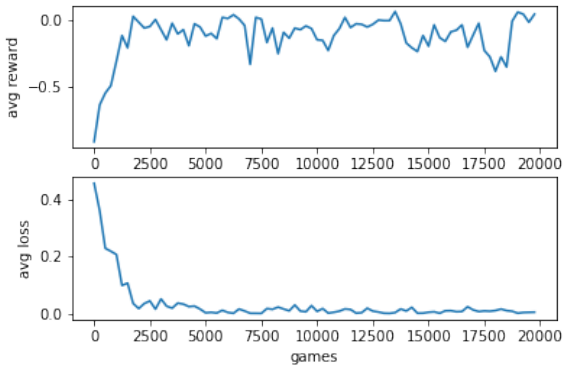

Figure 9. Average reward and average loss for every 250 games during training when fixing $\varepsilon = 0.01$ without buffer and only one batch.

- **Answer 13**: See Figure 10, 11. Decreasing ε helps training compared to having a fixed ε, but it is not very obvious. Effect of n^*: In the early stage of training, when n^* is small (e.g. $n^* = 1$), the agents do not perform very well, significantly worse than those with larger n^*. Since smaller $n^\star$ will make the player less explorative , thus hindering the learning in the early stages. In the middle and late stages of training, there is no significant difference in the performance of different n^*. It can be seen that when n^* is very large (e.g. $n^* = 40,000$), agent's performance against optimal(M_{opt}) has relatively large ups and downs.

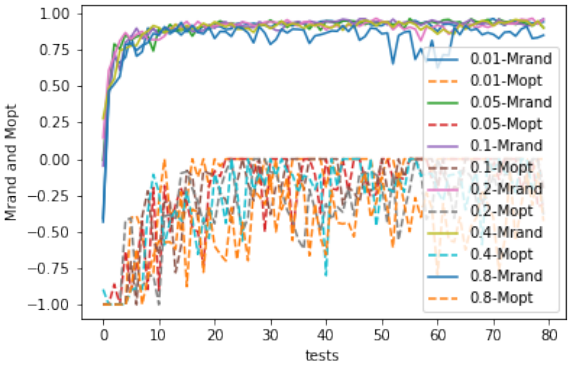

Figure 10. Mrand and Mopt over time for a fixed ε.

- **Answer 14**: See Figure 12. We observe: (1) When ε increases, the player will take longer time to achieve $M_{\text{opt}} = 0$. M_{opt} can hardly be zero when $\varepsilon_{\text{opt}} = 0$. Because it is impossible to beat the optimal player,

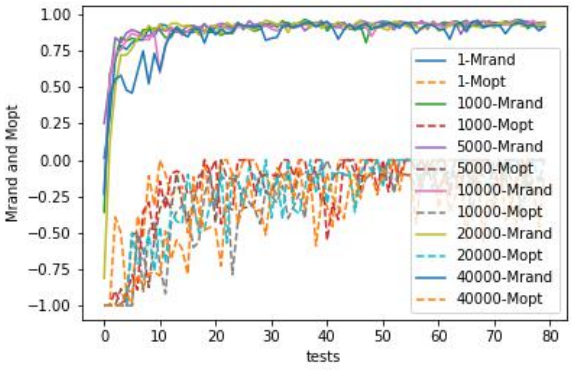

Figure 11. Mrand and Mopt over time for different values of n^* ={1, 1000, 5000, 10000, 20000, 40000}.

large ε_{opt} means the Q player cannot get positive reward during training. (2) When ε decreases, the player will take longer time to achieve high M_{rand}, i.e. beat the random player. Because small ε means the opponent is random at most time, it will not learn the strategy to beat the random player from a random player.

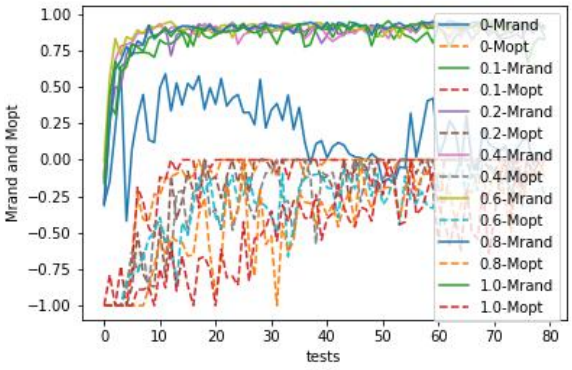

Figure 12. Mrand and Mopt over time for different values of ε_{opt} = {0, 0.1, 0.2, 0.4, 0.6, 0.8, 1.0}.

- **Answer 15**:

Highest value of M_{opt}	0.964
Highest value of M_{rand}	0

- **Answer 16**: See Figure 13. The agent learns to play Tic Tac Toe by playing against itself for different values of $\varepsilon \in \{0, 1\}$. Effect of ε: It is very obvious that when ε is too small (e.g. $\varepsilon = 0$) or ε is too large (e.g. $\varepsilon = 0.8$), the performances are terrible. Since setting $\varepsilon = 0$ means the agent cannot explore during training, and too large ε will not visit high reward actions frequently enough. The performances and trends are about the same for the rest ε values.

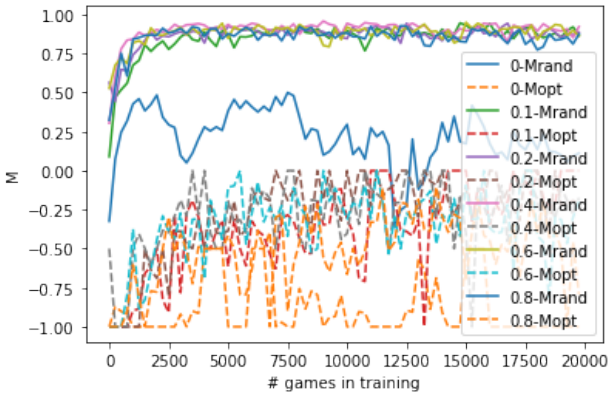

Figure 13. Self-practice : M_{rand} and M_{opt} over time for different values of ε ={0, 0.1, 0.2, 0.4, 0.6, 0.8} during training.

- **Answer 17**: See Figure 14. Decreasing ε helps training compared to having a fixed ε. Effect of $n^\star$: When facing to random policy, the performances of different $n^\star$ are almost the same. When facing to optimal policy, the large value of $n^\star$ (e.g. $n^\star$ = 40000) makes more fluctuations. Setting $n^\star = 10000$ would be a preferable choice.

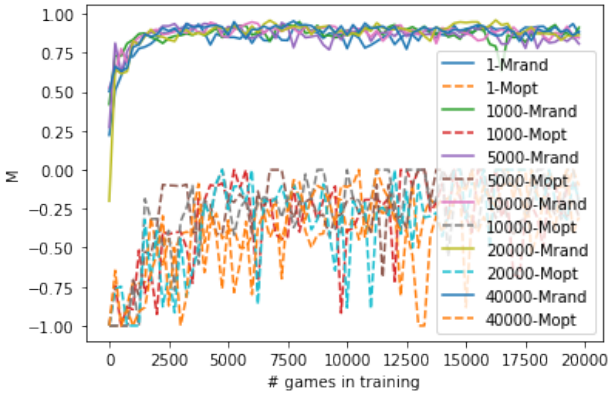

Figure 14. Self-practice : M_{rand} and M_{opt} over time for different values of ε ={0, 0.1, 0.2, 0.4, 0.6, 0.8} during training.

- **Answer 18**:

Highest value of M_{opt}	0.96
Highest value of M_{rand}	0.0

- **Answer 19**: See Figure 15. The result largely makes sense, and the agent learn the game well. For example, in case 2, when the first player choose the middle, the second one must choose the corner. And we can see the

player won't choose positions that have been chosen (black). However, there are some deficits: The Q some reasonable choice does not high Q-value, such as the corners in the first figure, the first player had better choose the corner or middle to maximize its winning chance, which might be attributed to under-exploration during self-play.

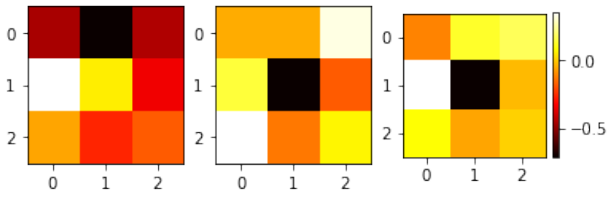

Figure 15. 1st: Heat map of Q-values of available actions. empty grid. 2nd: empty grid with position (1,1) being 'X'. 3st: empty grid with position (0,0) being 'X' and position (1,1) being 'O'. White position is the suggested position.

3. Comparing Q-Learning with Deep Q-Learning

- **Answer 20**: See Table 1

Comparison table						
	Expert			Self		
Method	M_{opt}	M_{rand}	T_{train}	M_{opt}	M_{rand}	T_{train}
Q	0.944	0.0	15000	0.95	-0.074	10000
DQN	0.964	0.0	2000	0.96	0.0	2000

Table 1. Table of the best performance of Q-Learning and DQN.

- **Answer 21**:
  - In terms of M_{rand}, Q learning converges much slower. Training a Q table is expensive for the agent, specially, in the beginning steps. Because, every state-action pair should be visited frequently in order to converge to the optimal policy. However, the expressiveness of neural network is much stronger, hence the convergence's of Deep Q is much faster. T_{train} for Deep Q is thus much smaller than Q.
  - In terms of M_{opt}, Q learning is stable and converges fast, while Deep Q's M_{opt} is very unstable even in the late stage of training. The TD update of Q learning, traversing backward from a episode, can solve the sparse reward problem better than the Deep Q learning, where Q network is updated based on the sampled transaction tuples. even we have improved the Deep Q's performance by propagating the final reward backward in each episode with discount 1/2.
  - Both Q and Deep Q will require a reasonable size ε. However, when learning from experts, optimal ε for Q learning (≈ 0.2) is usually larger than ε for Deep Q (≈ 0.01). When learning from self-play, both method require ε withn 0.2-0.6.
  - Both Q and Deep Q will get low M_{rand} from small ε_{opt}, low M_{opt} from large ε_{opt}. The difference is that Q learning require a narrow range to get optimal M_{rand}, i.e. $\varepsilon_{\text{opt}} \approx 0.5$. Deep Q can achieve high M_{rand} for wider range of ε_{opt}.
  - In summary, Q performs better in terms of M_{opt}, Deep Q performs better in terms of M_{rand}.