A Comprehensive Analysis of Q-Learning

prepared by

ANIRUDDHA JAYANT KARAJGI	**2017A7PS0084P**
ATMADEEP BANERJEE	**2017A7PS0101P**

IN PARTIAL FULFILMENT OF THE COURSE
ARTIFICIAL INTELLIGENCE

(CS F407)

NOVEMBER 2020

Contents

Introduction

Reinforcement learning is an area of machine learning that is concerned with algorithms for training agents to perform some task within their environments. RL algorithms have a concept of rewards for agents. The agents are given rewards depending upon their performance on their tasks and the RL algorithms teach the agents to take actions that maximize overall reward. Q-Learning is a reinforcement learning algorithm that teaches agents to learn a value for every possible action in every state. The possible configurations of the agent and the environment are the states for the algorithm. Once the agent is properly trained, the action with the maximum Q-value at each state will lead the agent to the most optimal outcome.

The Q-Learning algorithm takes multiple iterations through every state-action pair to converge. Initially all the Q-values of every state-action pair are initialized to some low constant value. As the training progresses, states and actions that repeatedly lead to high rewards have their Q-values gradually increased. The Q-values take into account the long term rewards that the action will lead to. In a properly trained agent an action that leads to a high immediate reward but lower overall reward will have a lower Q-value than an action with low short term reward but higher overall reward.

Q-Learning is parameterized by three main parameters which affect how well the trained agent performs. The learning rate determines how much weight is assigned to new information received when performing a new iteration vs the old information from all the previous iterations. If this parameter is too high, there will be too much fluctuation in the Q-values with every iteration and the training will be unstable. If it is too low, the agent will take too long to learn proper Q-values.

The second parameter, discount factor, determines how much weight is given to future rewards as compared to immediate rewards. If the discount factor is high, the agent will more strongly favour immediate rewards while a low discount factor favours high rewards in the future.

The third parameter epsilon controls the degree of exploration. As the agent is being trained some actions may initially appear to lead to a high reward. If the agent always picks the action with the best Q-value, during training, it will not be able to visit all state-action pairs. The epsilon factor allows the agent to “explore” during training time by picking an action which appears to be sub-optimal in the hope that it will lead to high rewards in the future. A high epsilon means that the agent will perform too much exploration, leading to slower convergence while a low epsilon leads to low exploration which can result in actually optimal actions for each state not being found.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \times [r_{t+1} + \gamma \, max_a \, Q(s_{t+1}, a) - Q(s_t, a_t)]$$

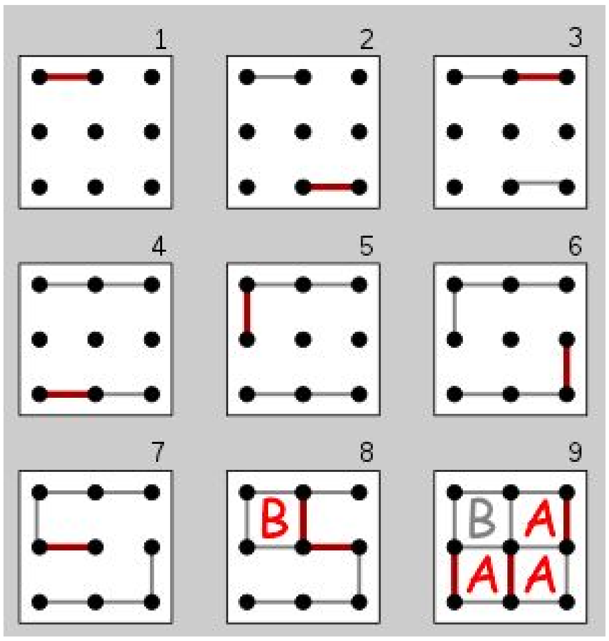
The algorithm for updating Q-values

Dots and Boxes

For this assignment we use Q-Learning to train an agent to play the game dots and boxes. This game consists of a board with a grid of dots. During each round a player connects two dots with a horizontal or vertical line. If a player draws a line that closes a box, they get a point. The player with the maximum number of points wins. For this game, every board configuration is a state for the Q-Learning algorithm and actions consist of drawing horizontal or vertical lines between available dots.

For our experiments we use a board with 3x3 dots. The board can have upto 12 distinct lines leading to 2^{12} states. With the mean number of lines being possible to draw being 6, there are a total of 6 x 2^{12} state-action pairs (about 25000).

Our implementation gives a positive reward to the agent when a game is won. We train the Q-Learning agent against random agents, simple heuristic based agents and other Q-learning agents. For every agent we play 500,000 games and we keep a track of the fraction of games won by the agent as the number of games increases. We vary the learning rate, discount factor and epsilon for the agents and plot the graphs of their performance.

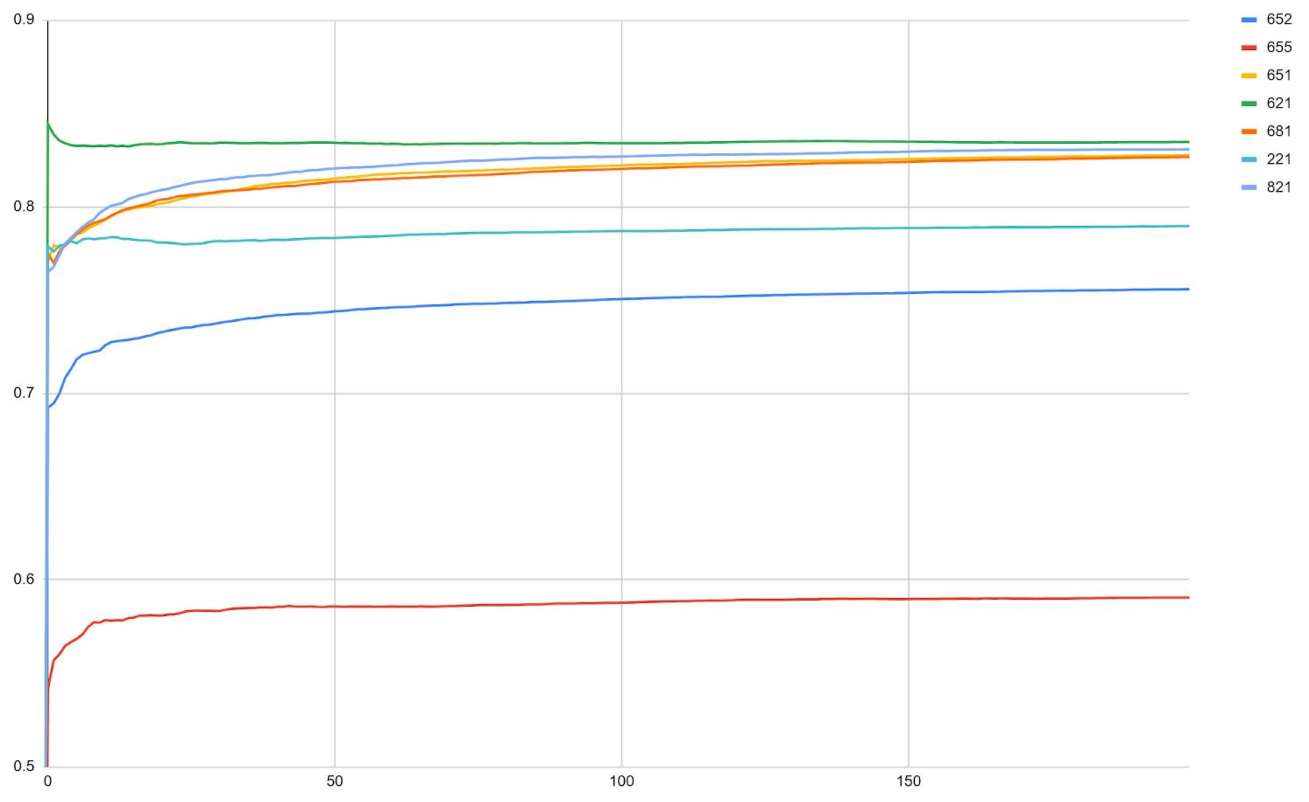


The dots and boxes game

Experiments

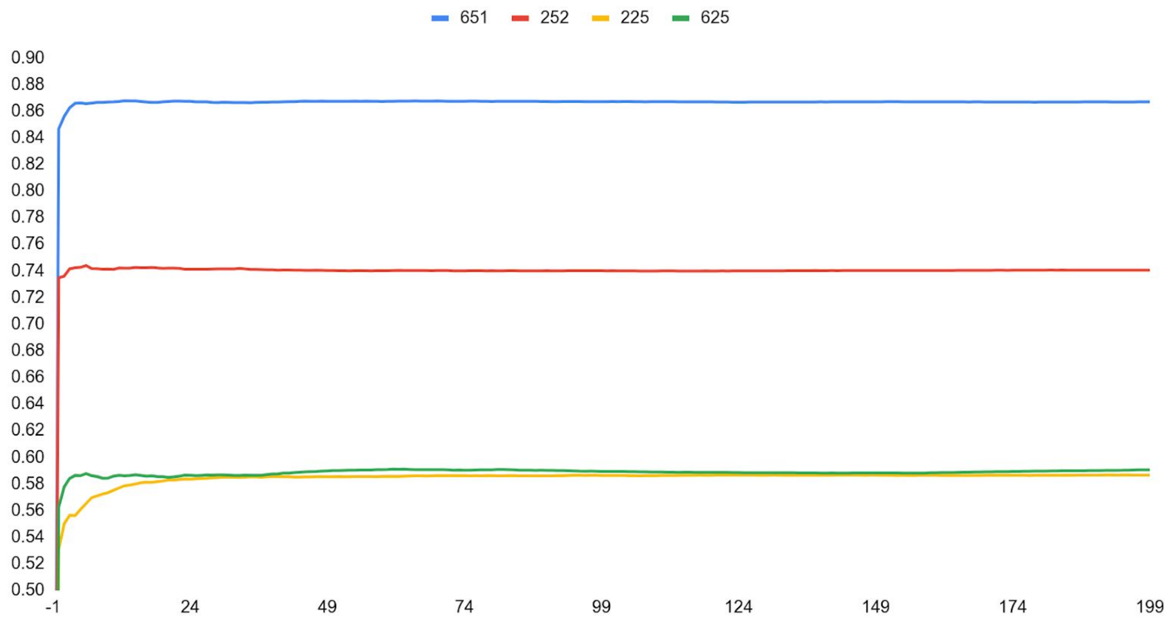
1. Q-learning agent vs Random agent (Q1)

We train the Q-learning agent against a random agent. This agent randomly draws a line at one of the available locations. We plot the share of wins of the Q-learning agent every 2500 games. The different plots correspond to learning rate, discount factor and epsilon values of [0.6,0.5,0.2], [0.6,0.5,0.5], [0.6,0.5,0.1], [0.6,0.2,0.1], [0.6,0.8,0.1], [0.2,0.2,0.1] and [0.8,0.2,0.1]. We observe from this experiment that lowering the epsilon leads to significantly better performance. For the same learning rate, lowering the discount factor leads to slight improvements. Changing the learning rate did not have a significant effect. 0.6 gave best performance and both 0.8 and 0.2 performing slightly worse.



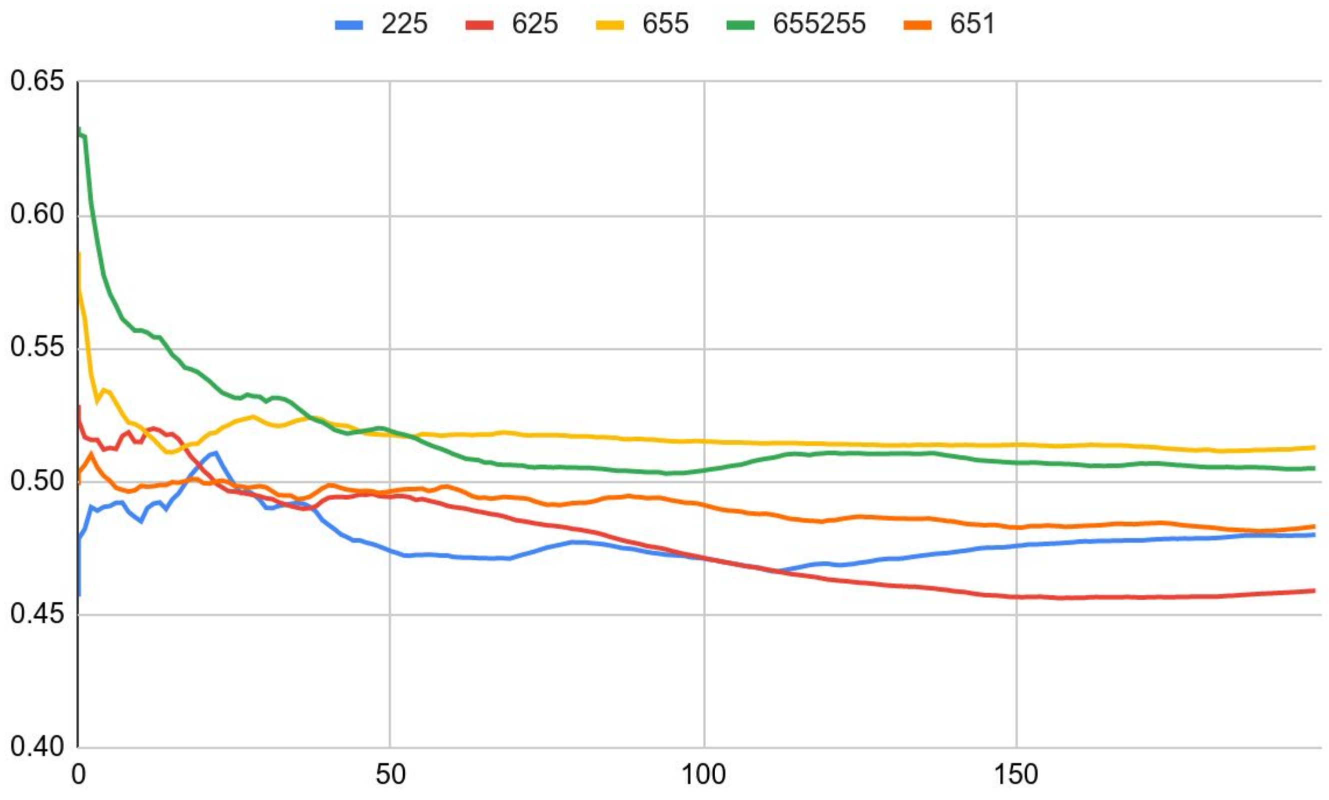
2. Q-learning vs Simple agent (Q3)

We train the Q-learning agent against a simple agent. This agent uses a simple heuristic and always picks the smallest available line. We plot the share of wins of the Q-learning agent every 2500 games. The different plots correspond to learning rate, discount factor and epsilon values of [0.6,0.5,0.1], [0.2,0.5,0.2], [0.2,0.2,0.5], and [0.6,0.2,0.5]. We observe from this experiment that a higher learning rate with a higher discount factor performs better.



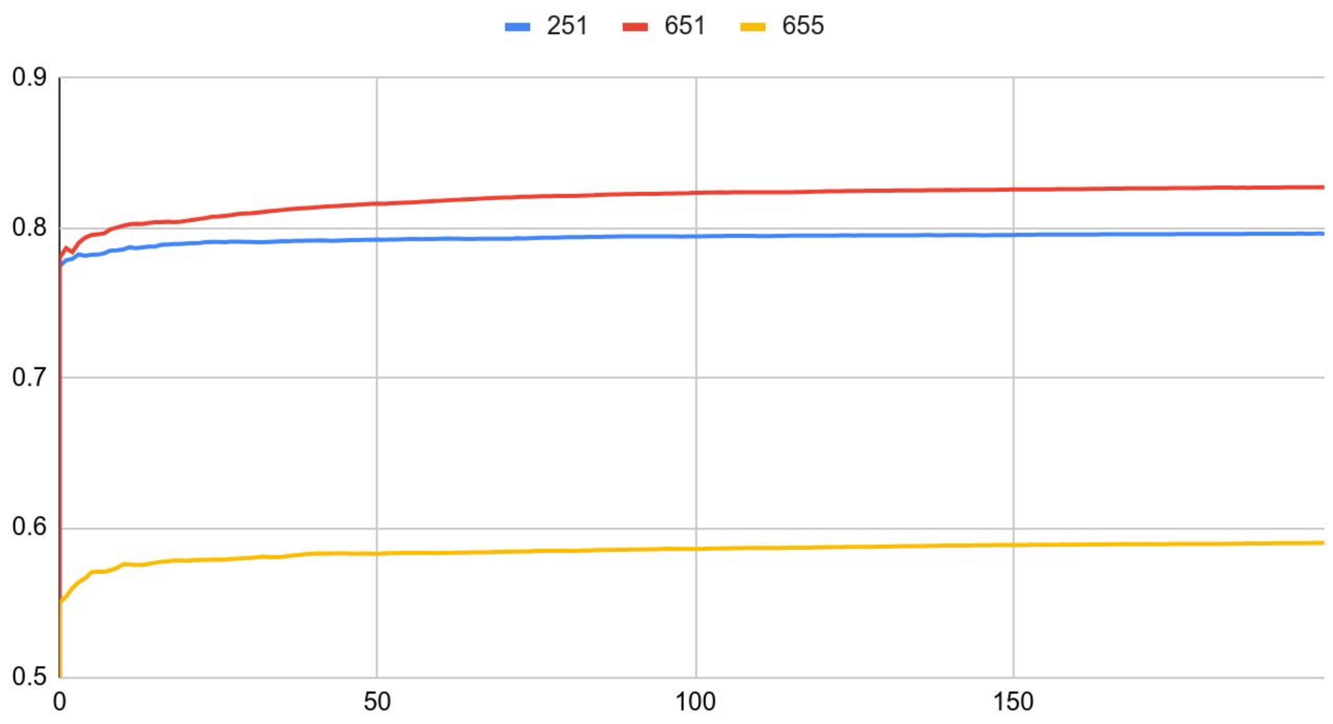
3. Q-learning vs Q-learning (Q2)

We train the Q-learning agent against another Q-learning agent, with both agents initialized to all q-values as 0 initially. We plot the share of wins of the first Q-learning agent every 2500 games. The different plots correspond to learning rate, discount factor and epsilon values of [0.2,0.2,0.5], [0.6,0.2,0.5], [0.6,0.5,0.5], [0.6,0.5,0.1] for both agents, and [0.6,0.5,0.5] for first and [0.2,0.5,0.5] for second agent. We observe from the experiment with different parameters that higher learning rate leads to a slightly better performance. For the experiments with the same parameters for both agents we see that the agents always converge to a win rate of almost 0.5, meaning that neither agent is better or worse.



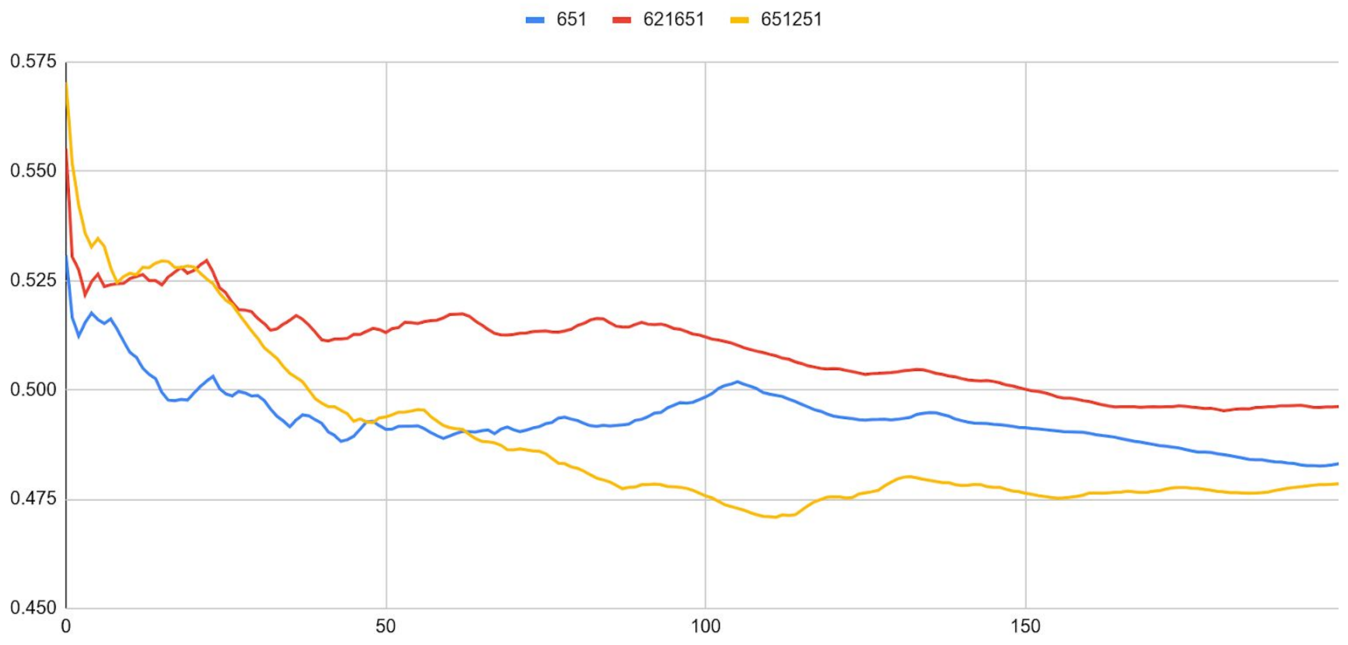
4. Q2 vs Random Agent (Q5)

We train the agent already trained against another Q-learning agent against a random agent. The plots correspond to learning rate, discount factor and epsilon values of [0.2,0.5,0.1], [0.6,0.5,0.1] and [0.6, 0.5, 0.1]. Q2 performs similar to Q1 in this experiment and higher learning rate and lower epsilon leads to better results.

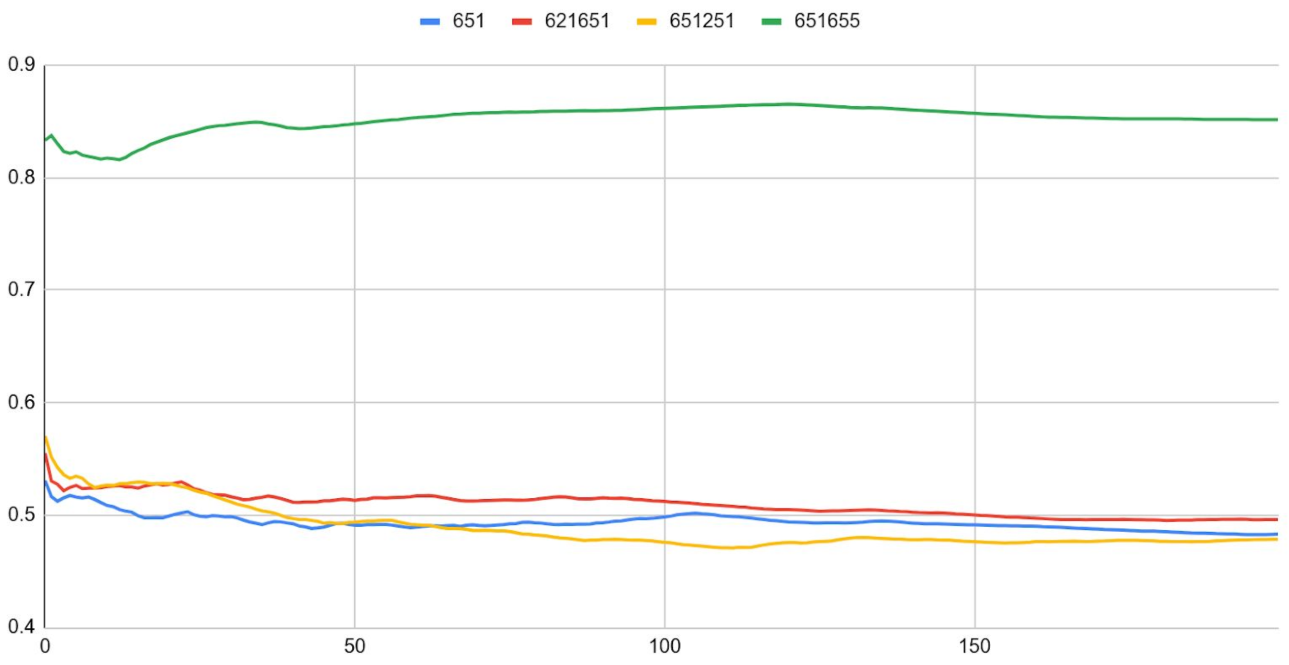


5. Q1 vs Q-learning agent (Q4)

We train Q1 against a new Q-learning agent with all Q-values initialized to 0. The plots correspond to learning rate, discount factor and epsilon values of [0.6,0.5,0.1] for both, [0.6,0.2,0.1] and [0.6,0.5,0.1], and [0.6,0.5,0.1] and [0.2,0.5,0.1]. Q1 performs similar to the new agent and final win rates are close to 0.5. When using different parameters for both agents we see that lowering the learning rate for the new agent improves its performance slightly.

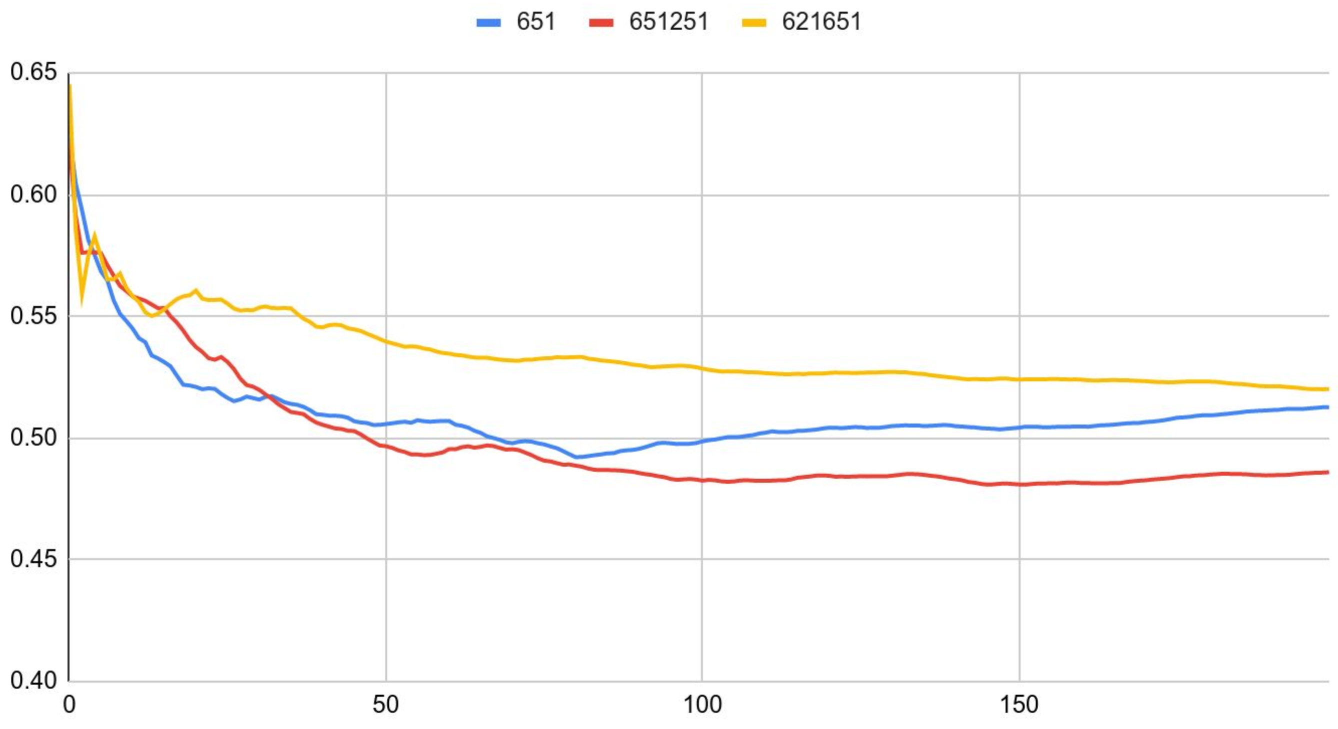


When varying the epsilon of the new agent from 0.1 to 0.5, there is a huge drop in its performance and Q1 has a win rate of over 0.8. The fact that low epsilon leads to better performance is also confirmed by previous experiments. So, for further experiments we do not vary the epsilon.



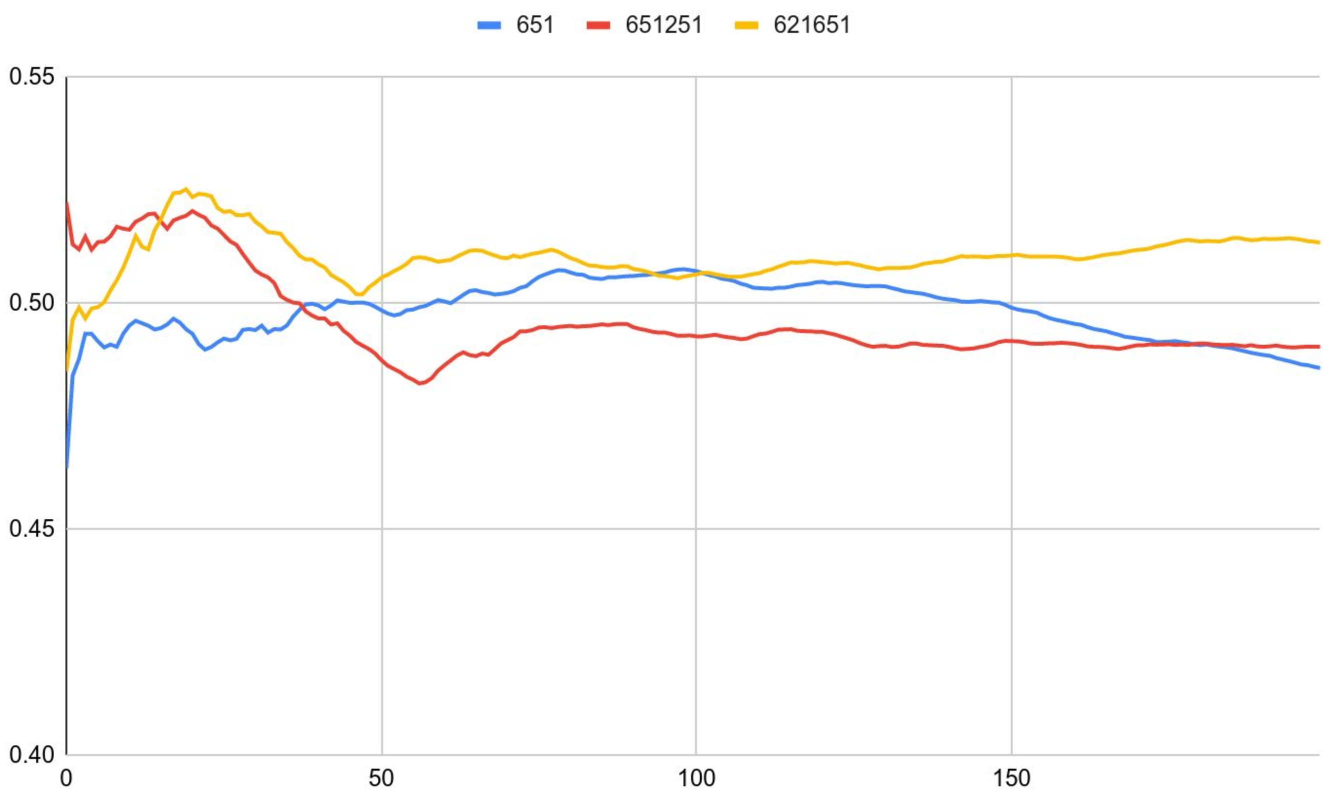
6. Q4 vs Q5 (Q6)

We train Q4 which is the result of training an agent against a random agent followed by a Q-learning agent against Q5 which is the result of training an agent a Q-learning agent followed by a random agent. The plots show that a win rate of around 0.5 is reached, so order of training against Q-learning and random agents does not matter. Lowering the learning rate and discount factor slightly improves performance.



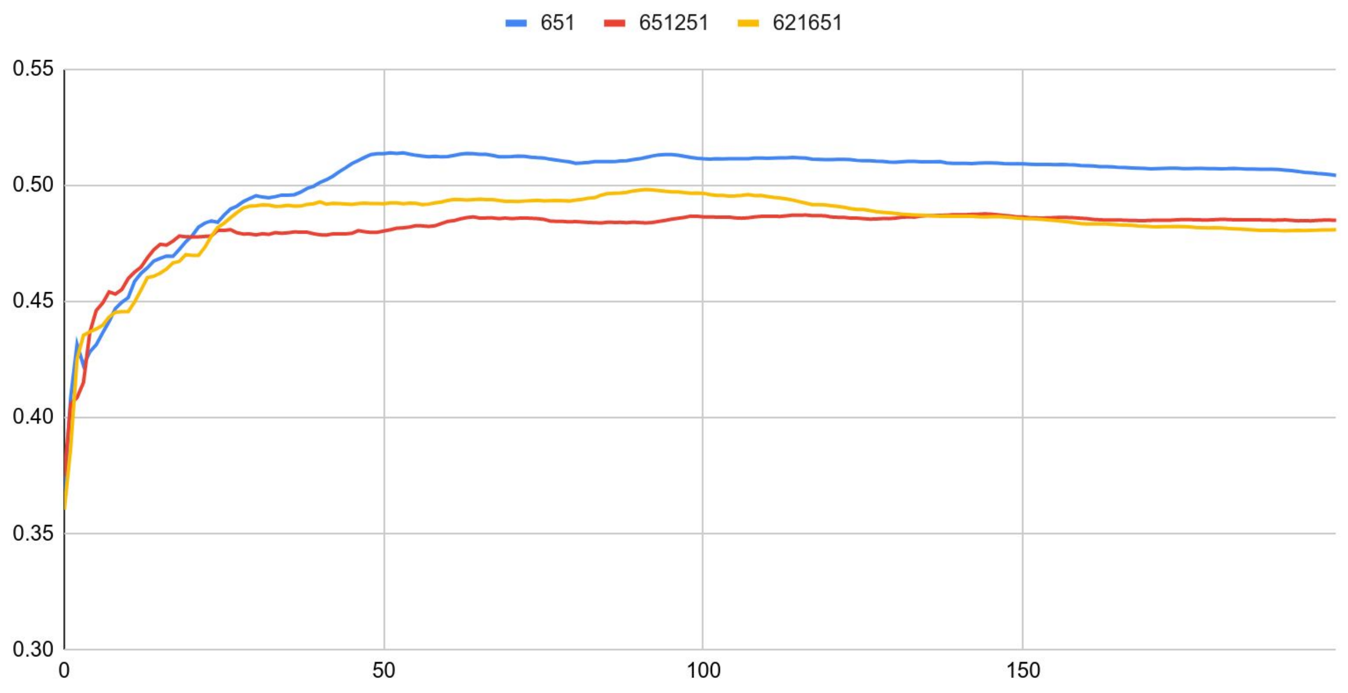
7. Q3 vs Q1 (Q7)

We train Q3 which has been trained against a simple agent against Q1 which has been trained against a random agent. According to the plots the models converge to around 0.5 win rate but Q1 has a better performance initially. Lowering the learning rate and discount factors slightly improves performance. Also, Q1 has a slightly better performance initially.



8. Q1 vs Q2 (Q8)

We train Q1 which has been trained against a random agent against Q2 which has been trained against another Q-learning agent. Initially Q1 performs worse than Q2 but they ultimately converge to about 0.5 win rate. This shows that training against a Q-learning agent creates a better agent than training against a random agent.



Findings

From our experiments we find the following:

1. A high epsilon is very harmful towards agent performance. Optimal performance is achieved at around 0.1 with rapid degradation on going up to 0.5.
2. The effect of learning rate and discount factor varies from case to case. When training two Q-learning agents against one another, a lower learning rate slightly improves final performance.
3. When training two Q-learning agents with different levels of previous experience, both converge to equivalent final performance. Also, the order in which experience is gathered does not matter much.
4. In experiment 7, we see that Q1 performs slightly better than Q3 initially and in experiment 8 we see that Q2 performs significantly better than Q1 initially. This indicates that training against a random agent is better than simple agent while training against another Q-learning agent gives the best performance.

Notation and Definitions

The notation used in the plot legends refers to the three parameters used to describe each agent. They are the learning rate, discount factor and epsilon.

Learning Rate: The rate at which the model updates its parameters

Discount Factor: The importance given to future rewards

Epsilon: The probability of the agent exploring vs exploiting.

For example, 651621 would mean that two agents were trained with the following parameters.

	Agent1	Agent2
Learning Rate	0.6	0.6
Discount Factor	0.5	0.2
Epsilon	0.1	0.1

Conclusion

In this report we present our results for training agents to play the game dots and boxes using Q-learning. We train Q-learning agents against random agents, simple heuristic agents and other Q-learning agents. We also vary the different parameters of the Q-learning algorithm and observe their effects on the final performance of the agents.

For our experiments performance is measured in terms of share of games won by the Q-learning agent. We conclude from our experiments that training against another Q-learning agent with a low epsilon factor gives best performance.

References

1. TowardsDataScience: Q Learning Tutorial
2. https://en.wikipedia.org/wiki/Q-learning
3. An introduction to Q-Learning: reinforcement learning
4. Q-Learning in Python
5. Artificial Agents Foundations: Q Learning
6. Artificial Intelligence: A Modern Approach