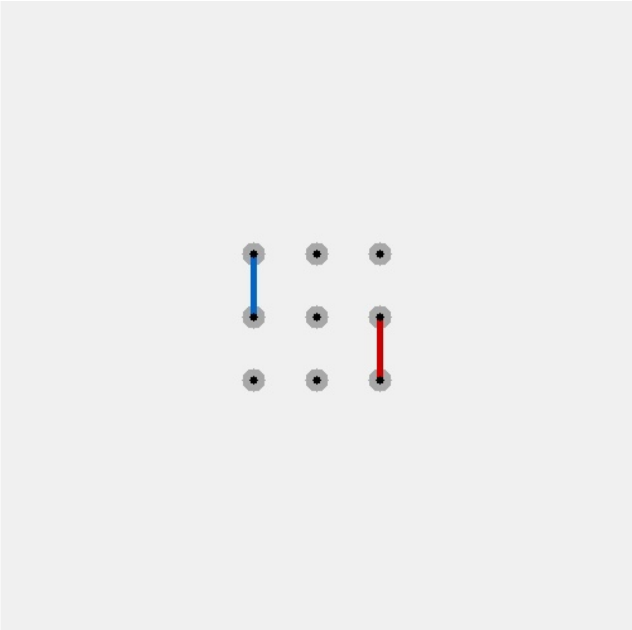
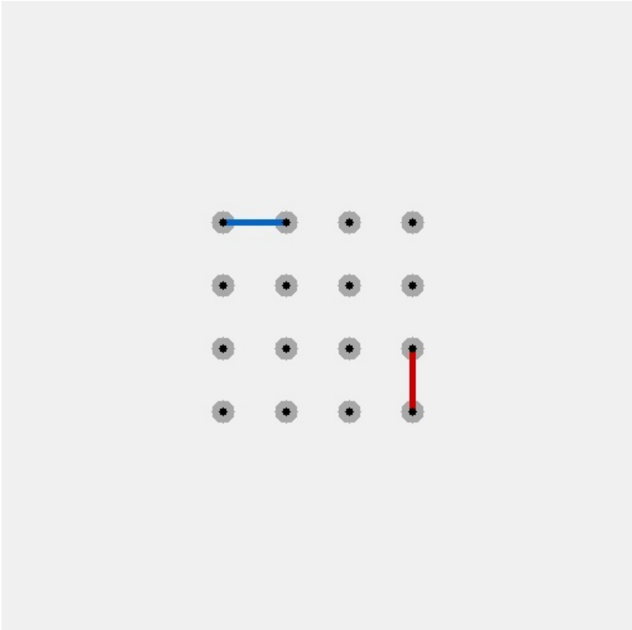
Robot Learning
Dots and Boxes

Q - Learning

Section 1: Game Play

I have created the “Dots and Boxes” game in python, with all the rules embedded. It is just a game which can be played between two human players with a good looking GUI as an interface.





Section 2: 2*2 Grid, Q-Learning and Function Approximation

The Q-Learning algorithm was implemented to make computer learn about the Game and try to win against any human player. It was trained for 100, 1000 and 10,000 times and the performances against a random player has been recorded.

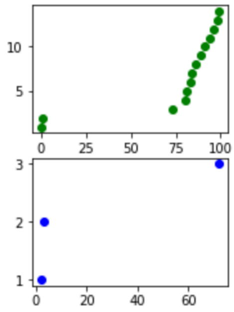
For Implementing the Q-Learning I have used a virtual board for recording states and actions and not the GUI.

I have tried playing with the parameters such a learning rate, discount factor and epsilon and found the best results came from the combination of learning rate = 0.09, discount factor = 0.8 and epsilon = 0.9

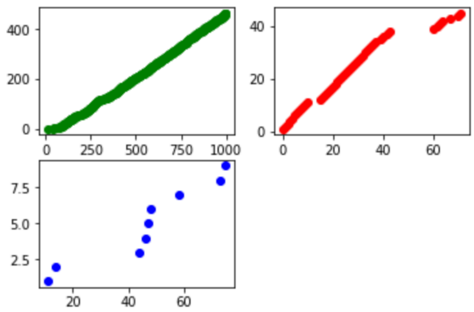
Table for performance against computer while training.

No.of Trainees	Wins	Loses	Ties	Invalid
100	14	56	3	27
1000	464	45	9	482
10000	9022	59	18	901

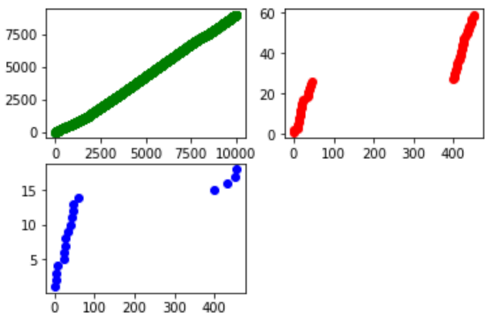
```
win 14
lose 56
tie 3
invalid 27
```



```
win 464
lose 45
tie 9
invalid 482
```



```
win 9022
lose 59
tie 18
invalid 901
```



As the number of trains increases no.of wins recorded is increasing, and because of the invalid moves the it is learning not to make such kind of moves in future, you can see this by the ratio of trains to that of invalid moves.
Win = Green; Lose = Red; Tie = Blue
And the loss-ratio is also significantly decreasing per no-of trains.

This results are against computer trained against itself and while training other computer player is also learning but I have programmed it to play more random moves so that original computer player learns better and faster.

Table for performance against Random Player for 100 games played.

No.of Trainees	Wins	Loses	Ties	Not trained states
100	22	32	46	551
1000	21	22	57	548
10000	31	24	45	554

As the training increases computer decreases the chances to loosing against a random player. As there is a random player involved the computer encounters many untrained states. But the when it encounters a trained state it tends to win with the increased training process. Because of the random player moves the numbers shown vary each time the program is executed.

As the program encounters many un trained states it couldn't perform drastically better than the random and few times 1/10th of a chance even the wins are lesser than that of loses. This is because of the random player and there are around 531441 states to be covered.

I have used a **CMAC** as a function approximate and trained it 1000 times.
Given the state and action the CMAC neural network will give us the Q-value and then we can decide upon which action to perform for maximum reward and again as the CMAC neural net is playing against random function the numbers showed will vary with every run.

Table for performance against Random Player for 100 games played with a Function Approximation.

No.of Trainees	Wins	Loses	Ties	Not trained states
100	20	34	46	705
1000	25	23	52	640
10000	26	25	49	600

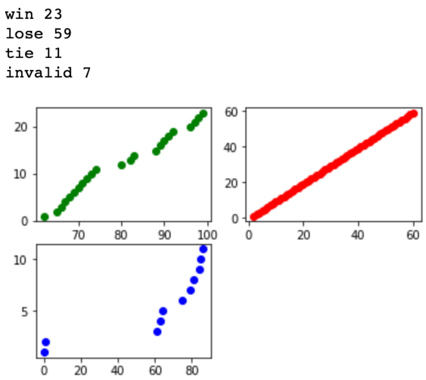
Conclusion:
- Performance of Q-Learning increases as number of trainers increases.
- Neural Network will be trained closer to the Q-Table with more number of training's.

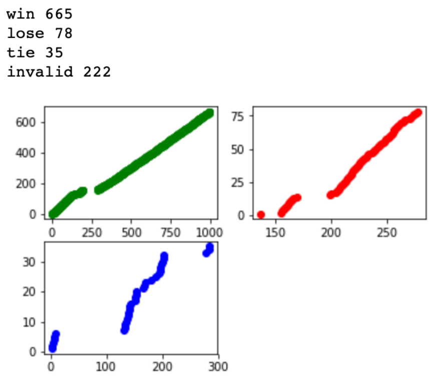
Section 3: 3*3 Grid, Q-Learning and Function Approximation

The Process of implementation is same as that of 2*2 Grid.

Table for performance against computer while training.

No.of Trainees	Wins	Loses	Ties	Invalid
100	23	59	11	7
1000	665	78	35	222
10000	8810	100	9	1081





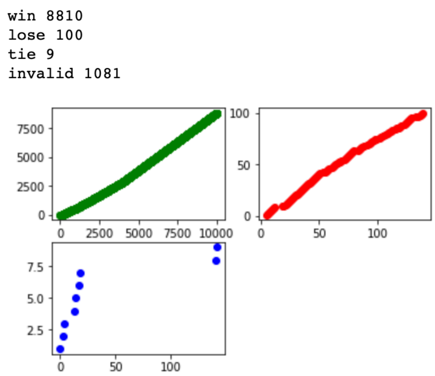


Table for performance against Random Player for 100 games played.

No.of Trainees	Wins	Loses	Ties	Not Trained States
100	24	28	48	1151
1000	26	26	48	1155
10000	29	20	51	1109

As the program encounters many un trained states it couldn't perform drastically better than the random and few times 1/10th of a chance even the wins are lesser than that of loses. This is because of the random player and there are around 282429536481 states to be covered and the numbers shown vary each time the program is executed.

Table for performance against Random Player for 100 games played with a Function Approximation.

No.of Trainees	Wins	Loses	Ties	Not trained states
100	24	26	50	1200
1000	27	27	46	1110
10000	28	25	47	1150

Conclusion:

- Performance of Q-Learning increases as number of trainers increases.
- Neural Network will be trained closer to the Q-Table with more number of training's.