ENPM 808F: Robot Learning

Project 4

Bala Murali Manoghar Sai Sudhakar (116150712)

Overview:

This report applies the reinforcement learning algorithm Q-learning to the simple pen and pencil game dots & boxes, also known as La Pipopipette. The goal is to analyse how different parameters and opponents affect the performance of the Q-learning algorithm. Simple computer opponents play against the Q-learning algorithm and later Q-learning algorithms with different experience play against each other.

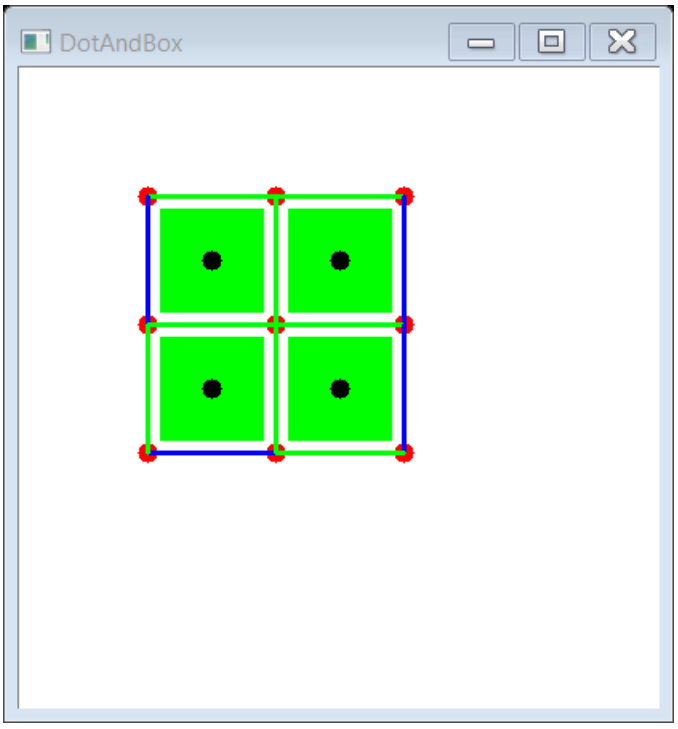
Q Table Implementation on 2x2 grid:

Used epsilon greedy policy.

Learning Rate- 0.6

Discount Factor- 0.7

Epsilon- 0.6



Green – Q Agent

Blue – Random Agent

For 100 Games self-play learning phase:

Agent1 – Q Learning Agent

Agent2 – Q Learning Agent

Both agents update same Q table

Agent1 wins- 46

Agent1 wins- 39

Number of Draw Games- 15

Time consumed- 1 Second

Results with random agent after 100 games training:

Agent1 – Random Agent

Agent2 – Q Learning Agent

Used Q table trained with 100 games of self-play

Random Agent wins- 6

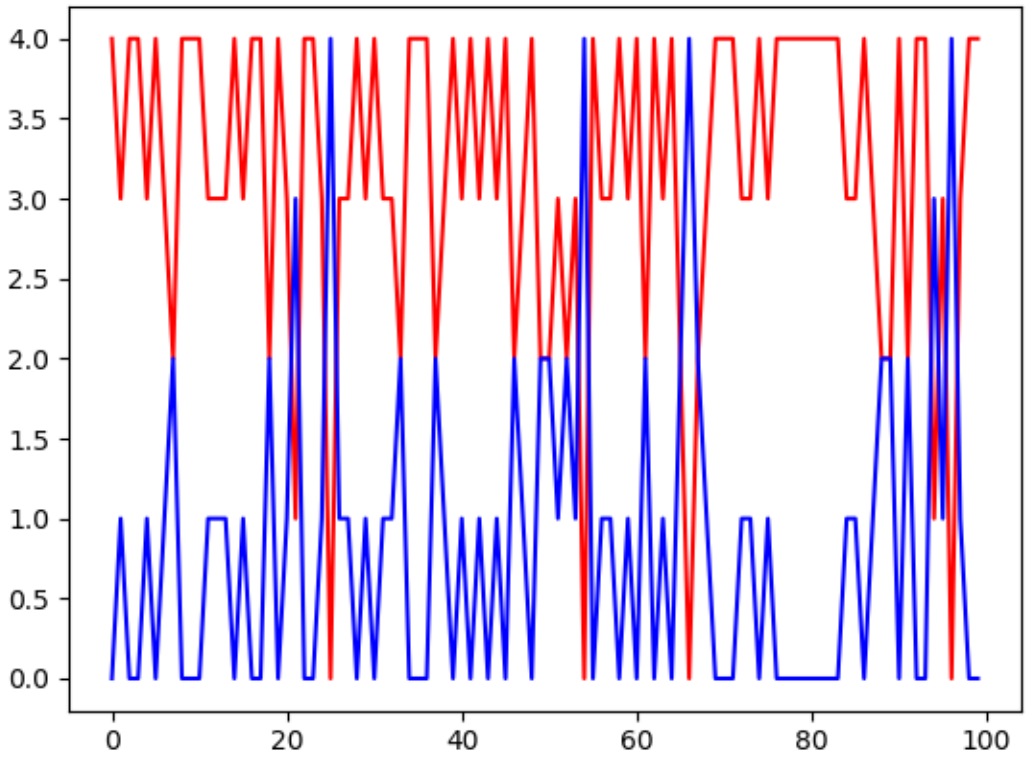
Q learning agent wins- 79

Number of Draw Games- 15

Average Score of Q Learning agent – 3.16

Average Score of Random agent – 0.84

Time consumed- 5 Seconds



Score Vs Iterations (100 training iterations)

For 1000 Games self-play learning phase:

Agent1 – Q Learning Agent

Agent2 – Q Learning Agent

Both agents update same Q table

Agent1 wins- 428

Agent1 wins- 352

Number of Draw Games- 220

Time consumed- 10 Seconds

Results with random agent after 1000 games training:

Agent1 – Random Agent

Agent2 – Q Learning Agent

Used Q table trained with 1000 games of self-play

Random Agent wins- 3

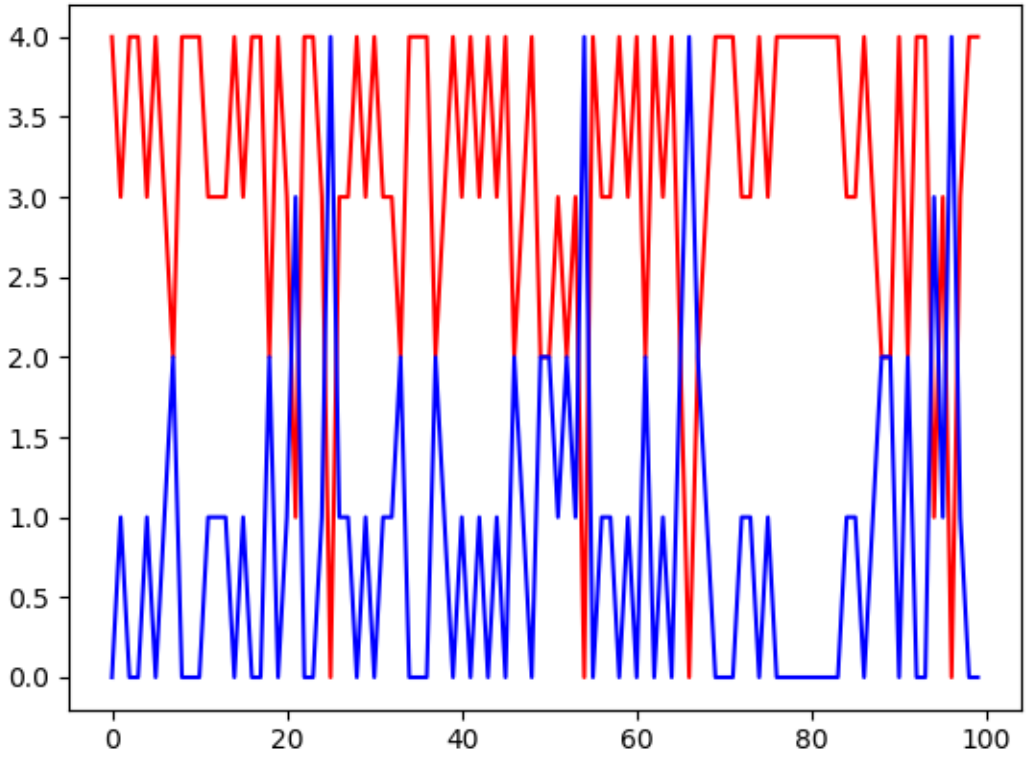
Q learning agent wins- 85

Number of Draw Games- 12

Average Score of Q Learning agent – 3.31

Average Score of Random agent – 0.69

Time consumed- 5 Seconds



Score Vs Iterations (1000 training iterations)

For 10000 Games self-play learning phase:

Agent1 – Q Learning Agent

Agent2 – Q Learning Agent

Both agents update same Q table

Agent1 wins- 4333

Agent1 wins- 3627

Number of Draw Games- 2040

Time consumed- 30 Seconds

Results with random agent after 10000 games training:

Agent1 – Random Agent

Agent2 – Q Learning Agent

Used Q table trained with 10000 games of self-play

Random Agent wins- 7

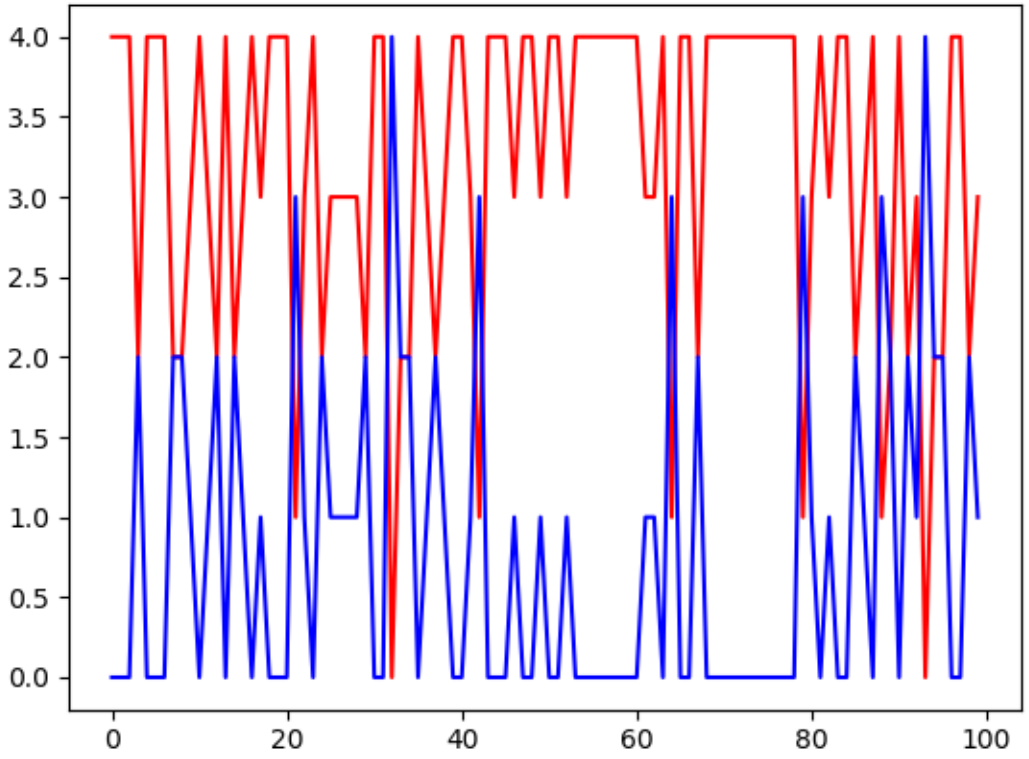
Q learning agent wins- 76

Number of Draw Games- 17

Average Score of Q Learning agent – 3.21

Average Score of Random agent – 0.76

Time consumed- 5 Seconds



Score Vs Iterations (10000 training iterations)

Training Loop	Random Agent	Q Learning Agent
100	6%	79%
1000	3%	85%
10000	7%	76%

Table : Win Percentage

Training Loop	Random Agent	Q Learning Agent
100	0.84	3.16
1000	0.69	3.31
10000	0.76	3.2

Table : Average Score

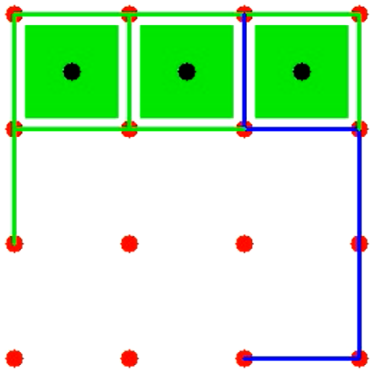
Q Table Implementation on 3x3 grid:

Used epsilon greedy policy.

Learning Rate- 0.6

Discount Factor- 0.7

Epsilon- 0.6



Green – Q Agent

Blue – Random Agent

For 100 Games self-play learning phase:

Agent1 – Q Learning Agent

Agent2 – Q Learning Agent

Both agents update same Q table

Agent1 wins- 51

Agent1 wins- 49

Number of Draw Games- 0

Time consumed- 4 Seconds

Results with random agent after 100 games training:

Agent1 – Random Agent

Agent2 – Q Learning Agent

Used Q table trained with 100 games of self-play

Random Agent wins- 12

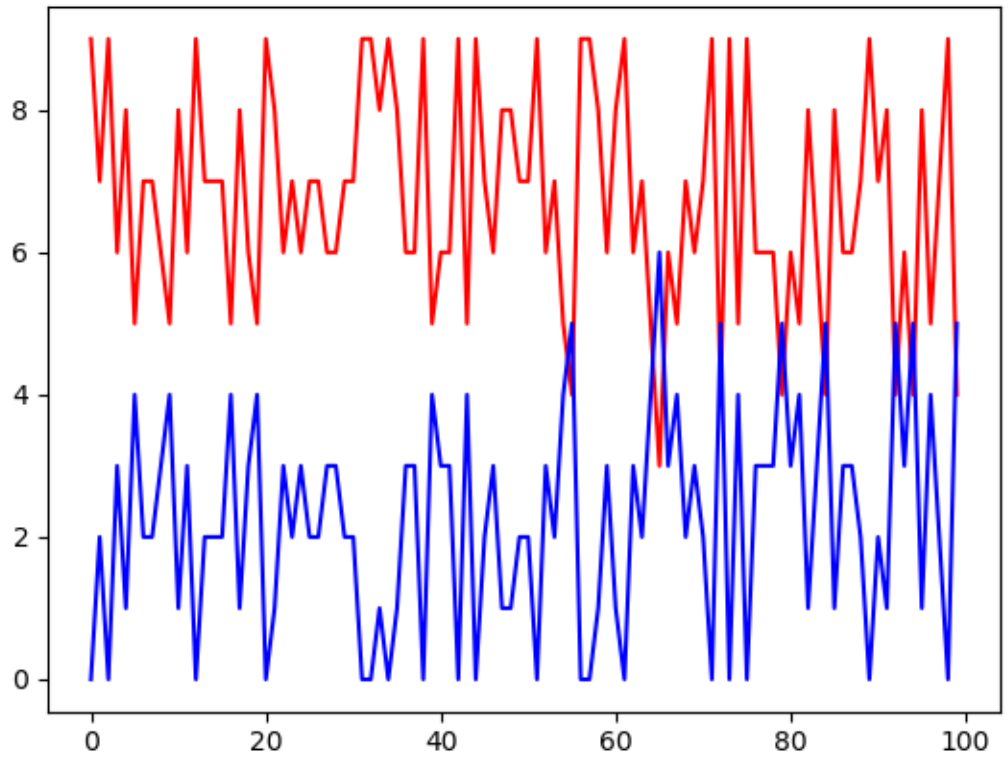
Q learning agent wins- 88

Number of Draw Games- 0

Average Score of Random agent – 2.71

Average Score of Q Learning agent – 5.89

Time consumed- 5 Seconds



Score Vs Iterations (100 training iterations)

For 1000 Games self-play learning phase:

Agent1 – Q Learning Agent

Agent2 – Q Learning Agent

Both agents update same Q table

Agent1 wins- 485

Agent1 wins- 515

Number of Draw Games- 0

Time consumed- 30 Seconds

Results with random agent after 1000 games training:

Agent1 – Random Agent

Agent2 – Q Learning Agent

Used Q table trained with 1000 games of self-play

Random Agent wins- 9

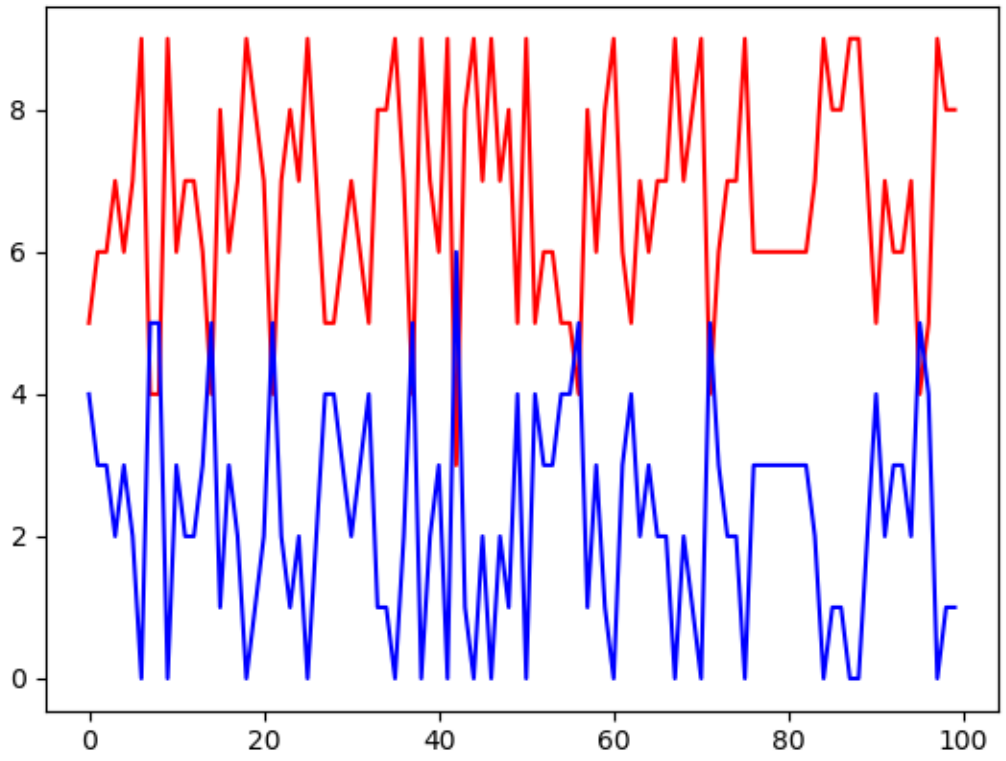
Q learning agent wins- 91

Number of Draw Games- 0

Average Score of Random agent – 2.24

Average Score of Q Learning agent – 6.76

Time consumed- 5 Seconds



Score Vs Iterations (1000 training iterations)

For 10000 Games self-play learning phase:

Agent1 – Q Learning Agent

Agent2 – Q Learning Agent

Both agents update same Q table

Agent1 wins- 4922

Agent1 wins- 5078

Number of Draw Games- 0

Time consumed- 100 Seconds

Results with random agent after 10000 games training:

Agent1 – Random Agent

Agent2 – Q Learning Agent

Used Q table trained with 10000 games of self-play

Random Agent wins- 8

Q learning agent wins- 92

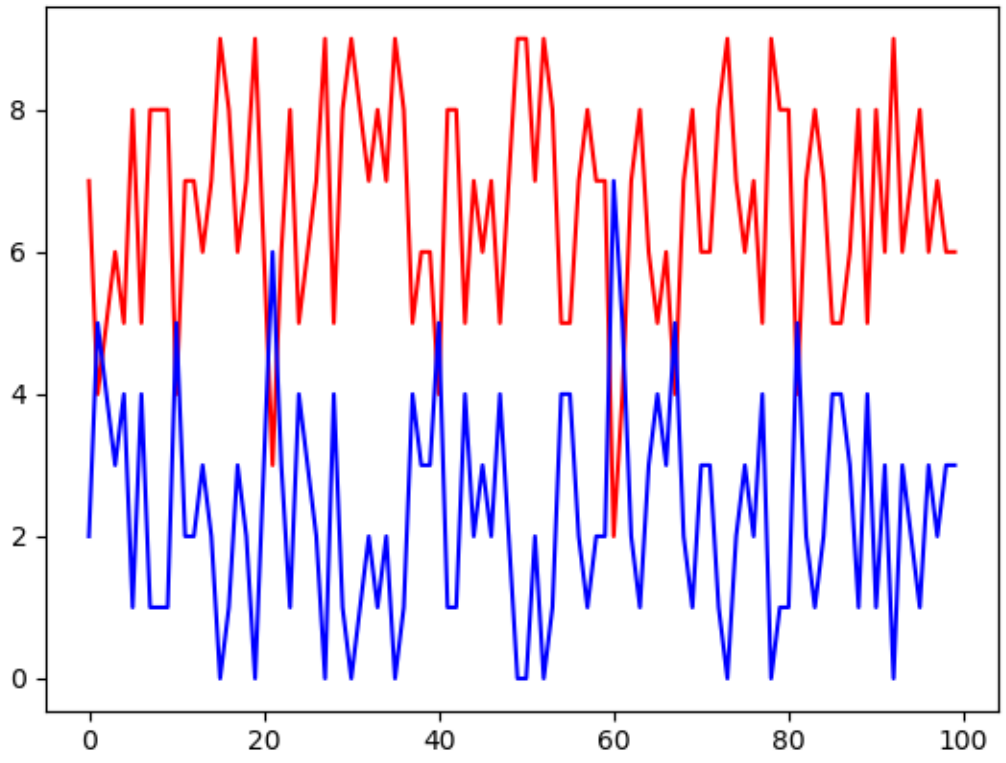
Number of Draw Games- 0

Time consumed- 5 Seconds

Average Score of Random agent – 2.32

Average Score of Q Learning agent – 3.68

Time consumed- 5 Seconds



Score Vs Iterations (10000 training iterations)

Training Loop	Random Agent	Q Learning Agent
100	12%	88%
1000	9%	91%
10000	8%	92%

Table : Win Percentage

Training Loop	Random Agent	Q Learning Agent
100	2.71	5.89
1000	2.24	6.76
10000	2.32	6.68

Table : Average Score

The results can be improved by running for longer iterations and reducing learning rate

Learning rate is the weight given to past experiences

Discount factor is the weight given to future rewards

Q Learning with functional approximation:

The disadvantage of having a look up table is that the size of the table increases with number of states. Often not all states are encountered during training phase. For 2x2 grid there are 2^{12} states and for 3x3 grid there are 2^{24} states. Thus, the size increases exponentially with problem. To overcome this, we do functional value approximation for Q learning.

Functional Approximation Implementation on 2x2 grid:

Hidden Layers: 150 x 150 x 150 x 150

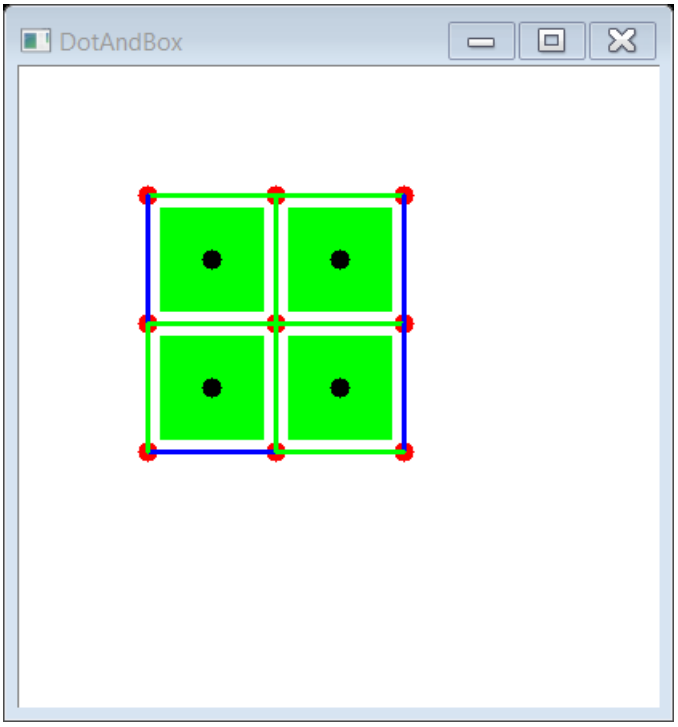
Reward Strategy:

Completed Box = +1

Win = +5

Invalid Action = -1

Reward for invalid action is needed since the neural net has to learn that invalid actions are not possible. Initially negative reward was not given, and the neural net did not provide promising results.



Green – Functional Approximation Agent

Blue – Random Agent

Results with random agent after 100 games training:

Agent1 – Random Agent

Agent2 – Functional Agent

Used trained neural net with 100 games of self-play

Random Agent wins- 54

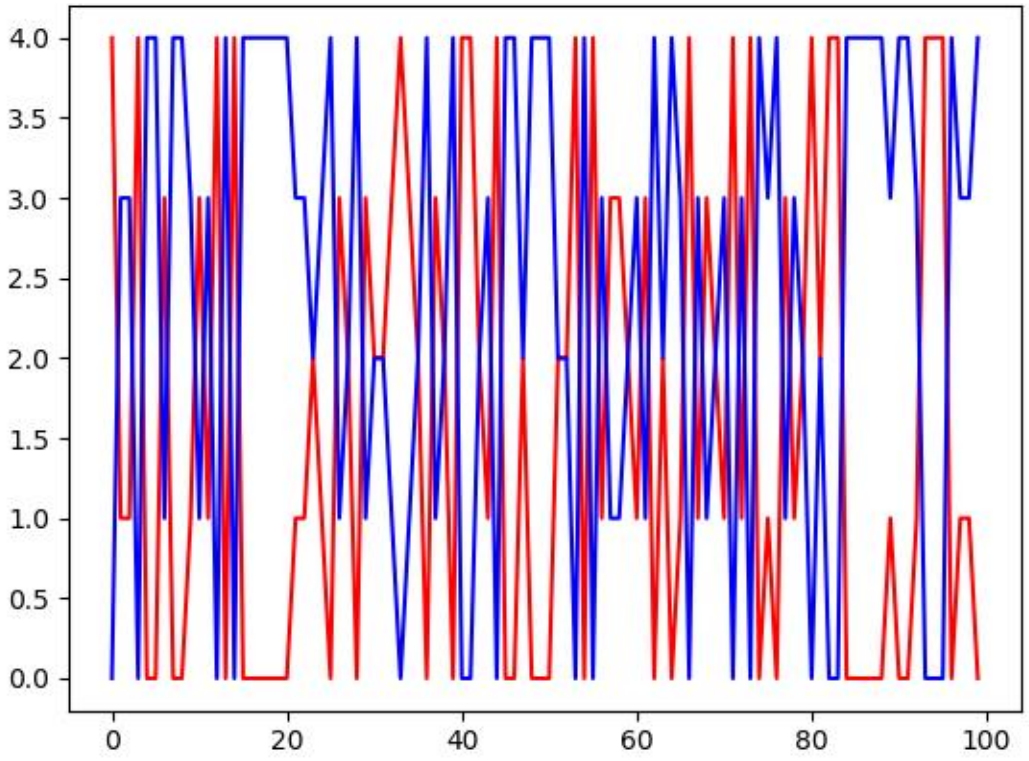
Functional Agent wins- 31

Number of Draw Games- 15

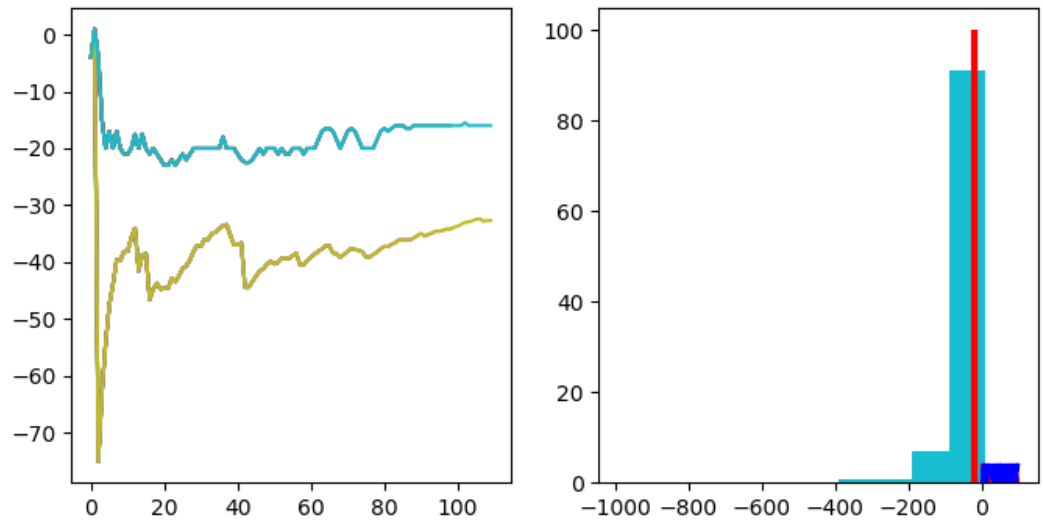
Average Score of Functional Agent – 1.62

Average Score of Random Agent – 2.38

Time consumed- 50 Seconds



Score Vs Iterations (100 training iterations)



Reward Vs Iterations and Histogram of total reward graphs

Results with random agent after 1000 games training:

Agent1 – Random Agent

Agent2 – Functional Agent

Used trained neural net with 1000 games of self-play

Random Agent wins- 60

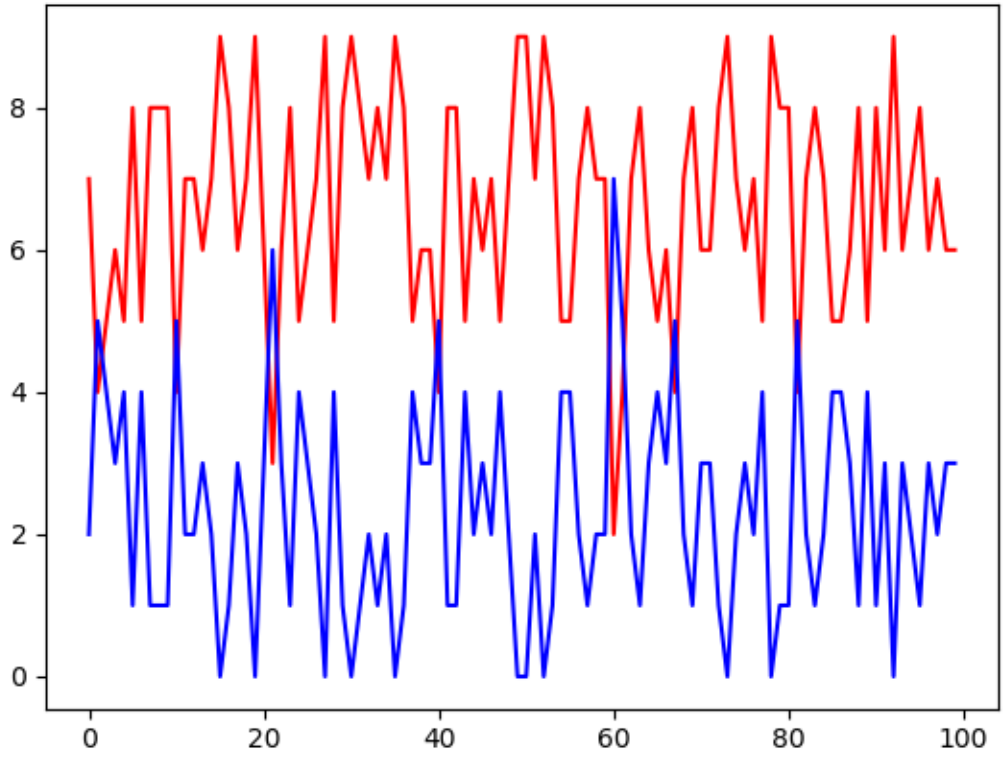
Functional Agent wins- 28

Number of Draw Games- 12

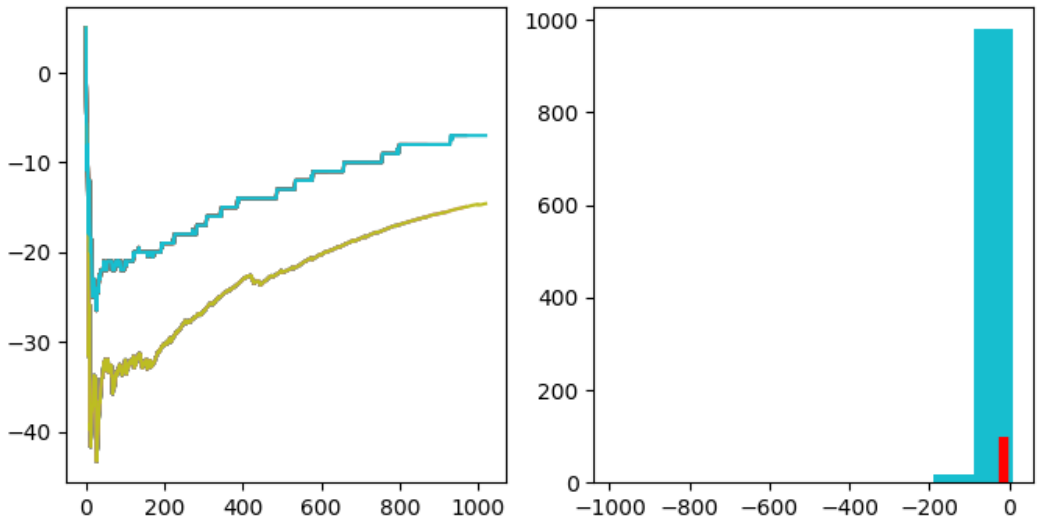
Average Score of Functional Agent – 2.51

Average Score of Random Agent – 1.49

Time consumed- 70 Seconds



Score Vs Iterations (1000 training iterations)



Reward Vs Iterations and Histogram of total reward graphs

Results with random agent after 10000 games training:

Agent1 – Random Agent

Agent2 – Functional Agent

Used trained neural net with 10000 games of self-play

Random Agent wins- 10

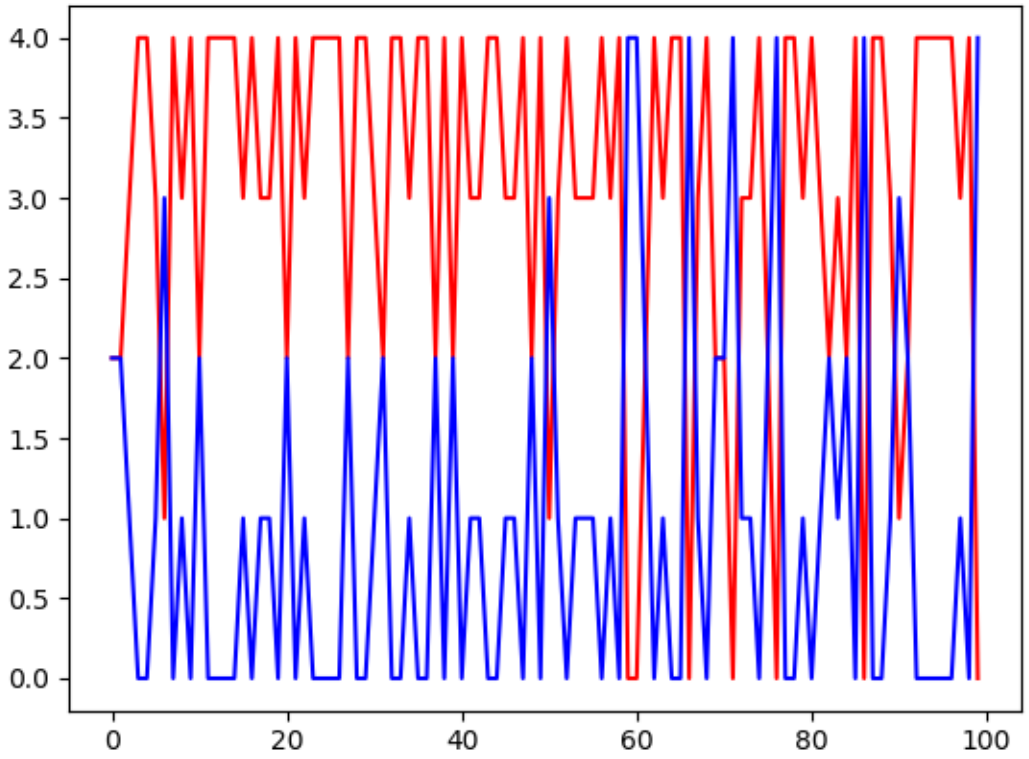
Functional Agent wins- 74

Number of Draw Games- 16

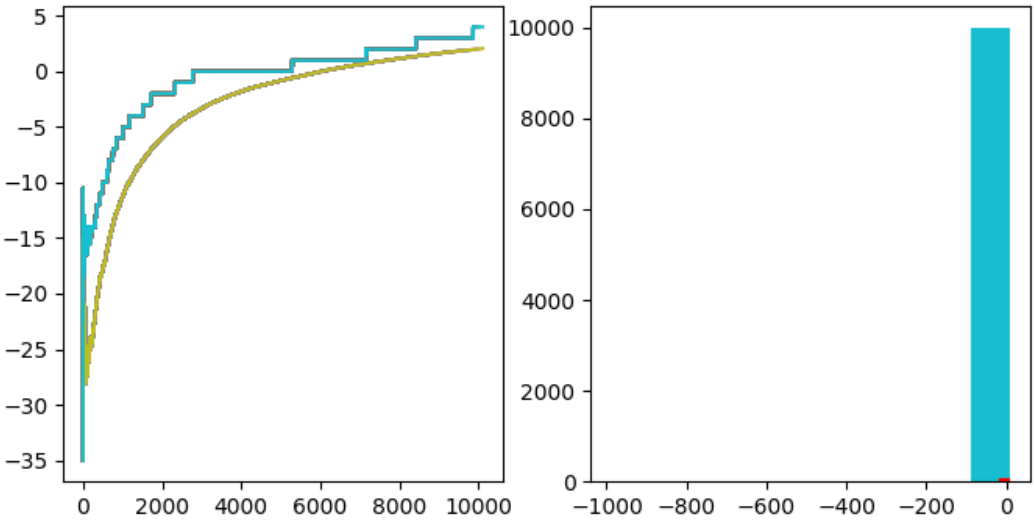
Average Score of Functional Agent – 3.04

Average Score of Random Agent – 0.96

Time consumed- 500 Seconds



Score Vs Iterations (10000 training iterations)



Reward Vs Iterations and Histogram of total reward graphs

Training Loop	Random Agent	Q Learning Agent
100	54%	31%
1000	60%	28%
10000	10%	74%

Table : Win Percentage

Training Loop	Random Agent	Q Learning Agent
100	2.38	1.62
1000	2.51	1.49
10000	0.96	3.04

Table : Average Score

Q Table Implementation on 3x3 grid:

Hidden Layers: 150 x 150 x 150 x 150

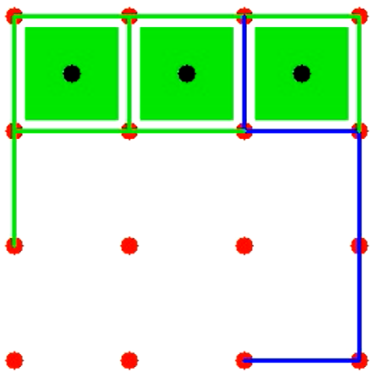
Reward Strategy:

Completed Box = +1

Win = +5

Invalid Action = -1

Reward for invalid action is needed since the neural net has to learn that invalid actions are not possible. Initially negative reward was not given, and the neural net did not provide promising results.



Green – Q Agent

Blue – Random Agent

Results with random agent after 100 games training:

Agent1 – Random Agent

Agent2 – Functional Agent

Used trained neural net with 100 games of self-play

Random Agent wins- 49

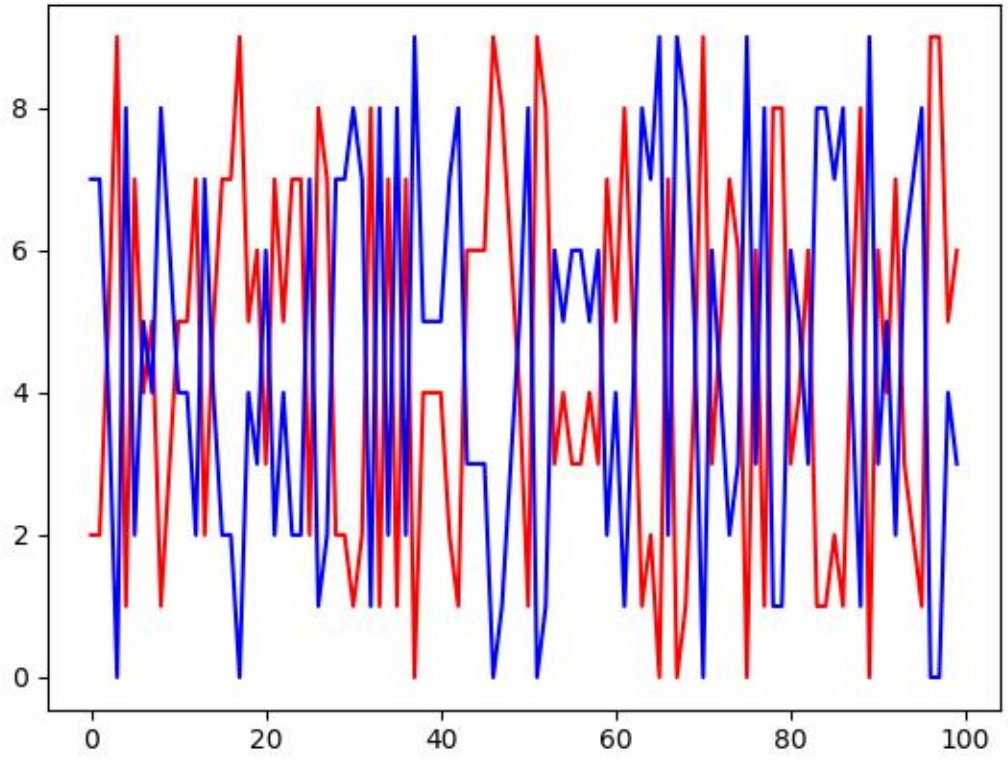
Functional Agent wins- 51

Number of Draw Games- 0

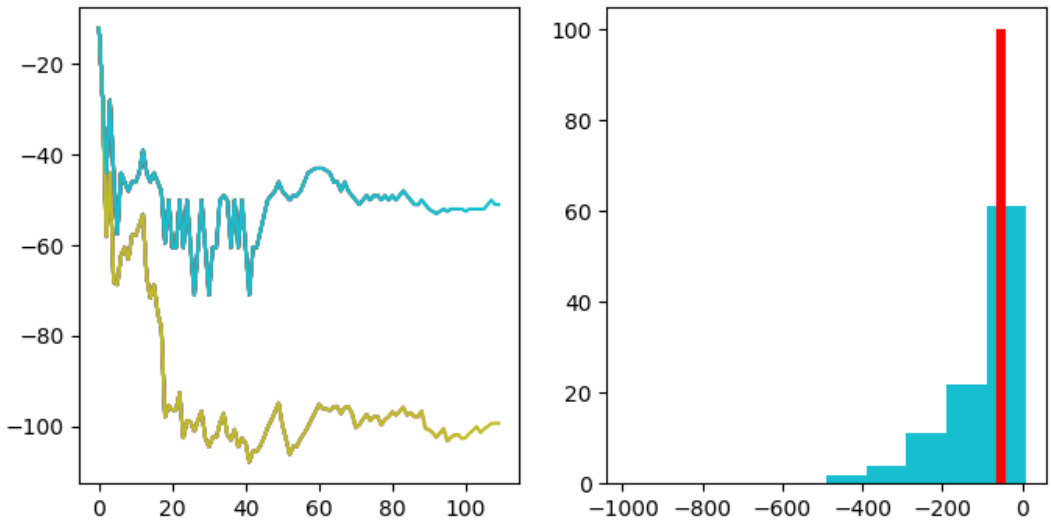
Average Score of Functional Agent – 4.52

Average Score of Random Agent – 4.48

Time consumed- 70 Seconds



Score Vs Iterations (100 training iterations)



Reward Vs Iterations and Histogram of total reward graphs

Results with random agent after 1000 games training:

Agent1 – Random Agent

Agent2 – Functional Agent

Used trained neural net with 1000 games of self-play

Random Agent wins- 49

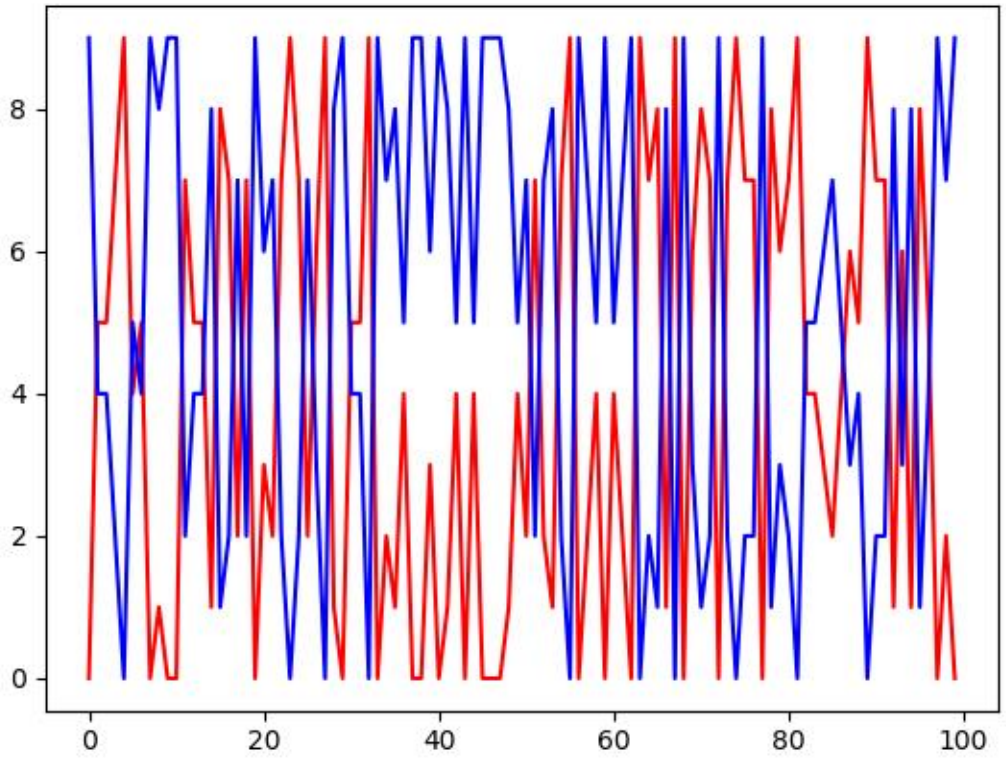
Functional Agent wins- 51

Number of Draw Games- 0

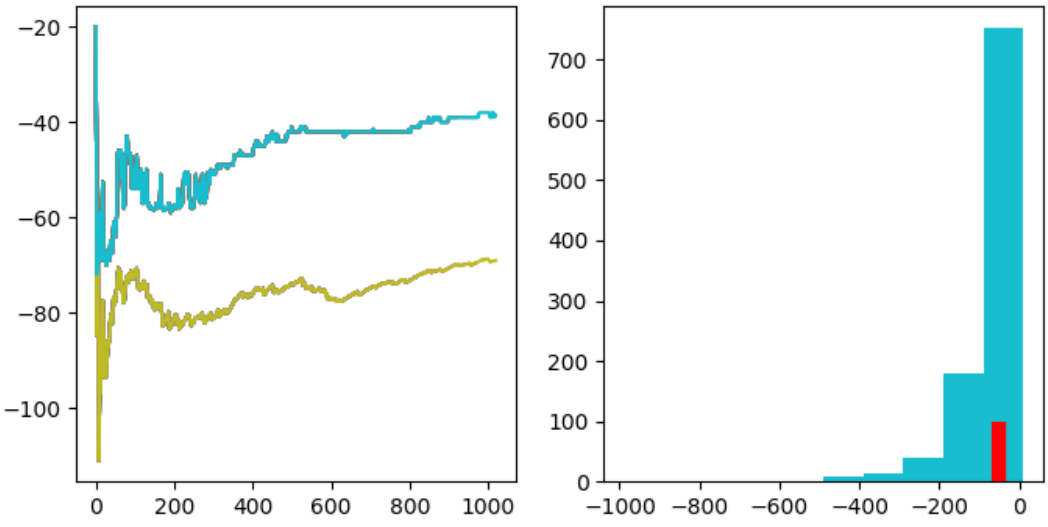
Average Score of Functional Agent – 4.48

Average Score of Random Agent – 4.52

Time consumed- 700 Seconds



Score Vs Iterations (1000 training iterations)



Reward Vs Iterations and Histogram of total reward graphs

Results with random agent after 10000 games training:

Agent1 – Random Agent

Agent2 – Functional Agent

Used trained neural net with 10000 games of self-play

Random Agent wins-

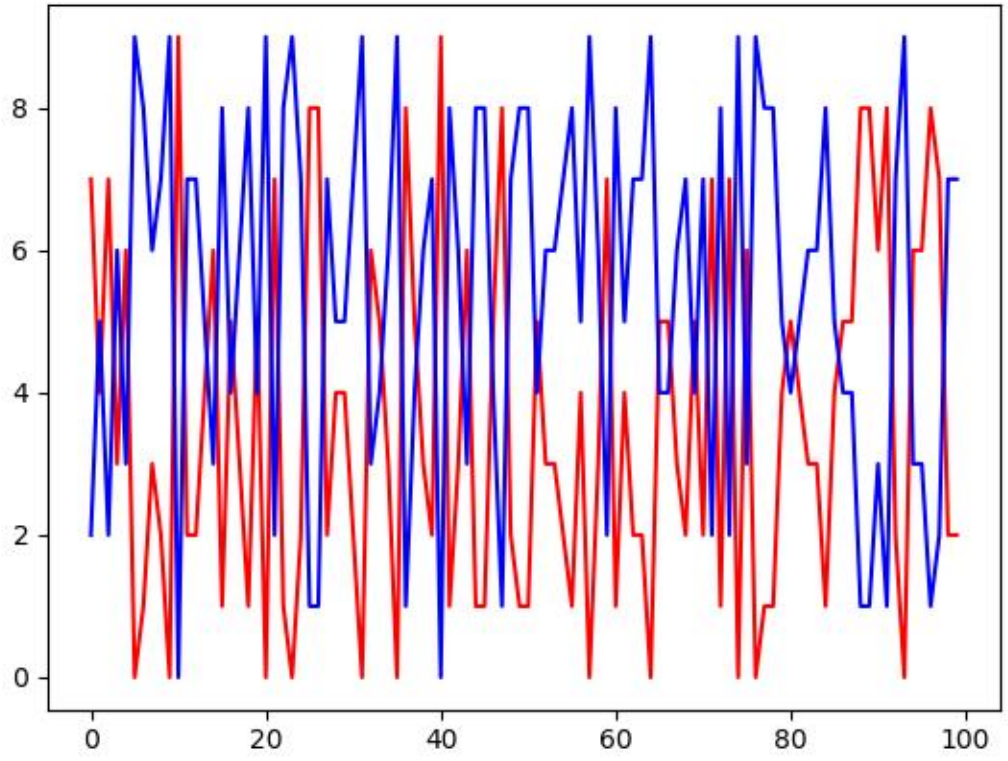
Functional Agent wins-

Number of Draw Games-

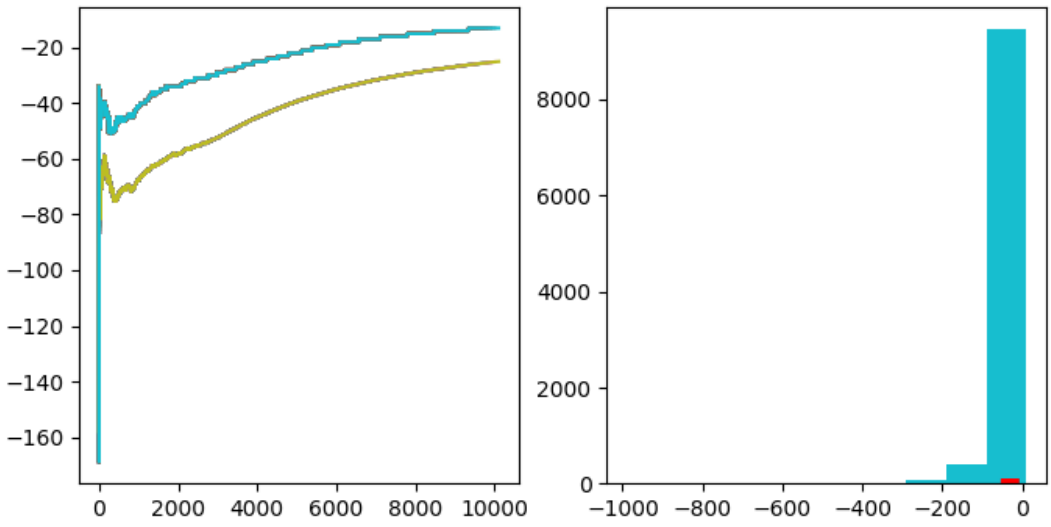
Average Score of Functional Agent –

Average Score of Random Agent –

Time consumed- 1000 Seconds



Score Vs Iterations (10000 training iterations)



Reward Vs Iterations and Histogram of total reward graphs

Training Loop	Random Agent	Q Learning Agent
100	49	51
1000	55	45
10000	63	37

Table : Win Percentage

Training Loop	Random Agent	Q Learning Agent
100	4.52	4.48
1000	5.04	3.96
10000	5.42	3.58

Table : Average Score

As seen from the graph, the total reward did not saturate.

In case of 2x2 grid game, the reward increased to positive value and can be seen that the Functional agent wins at the end of training after 10000 iterations.

But in case of 3x3 grid game, the reward is not positive even after 10000 iterations the reward is not positive. This is because not all state – action combination is visited in this case. Training for more iterations can help in improving the functional agent