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.

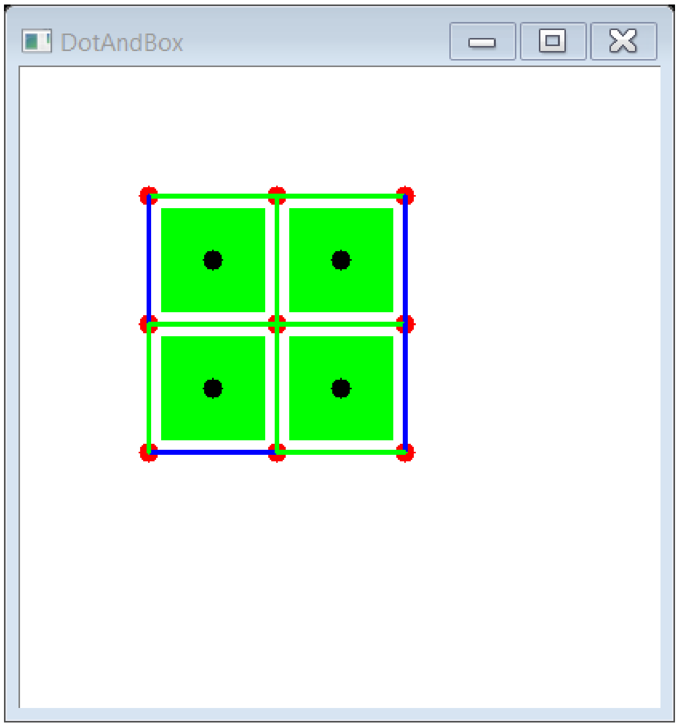
Q Table Implementation on 2x2 grid:

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- 5

Q learning agent wins- 75

Number of Draw Games- 15

Time consumed- 5 Seconds

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- 6

Q learning agent wins- 83

Number of Draw Games- 11

Time consumed- 5 Seconds

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 1000 games of self-play

Random Agent wins- 6

Q learning agent wins- 79

Number of Draw Games- 15

Time consumed- 5 Seconds

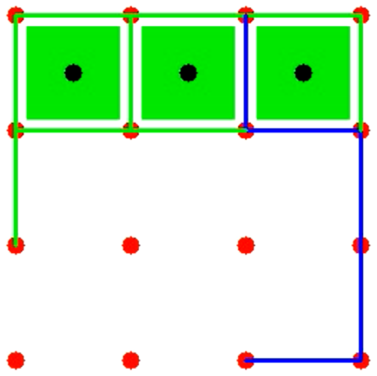
<u>Q Table Implementation on 3x3 grid:</u>

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

Time consumed- 5 Seconds

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- 10

Q learning agent wins- 90

Number of Draw Games- 0

Time consumed- 5 Seconds

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 1000 games of self-play

Random Agent wins- 8

Q learning agent wins- 92

Number of Draw Games- 0

Time consumed- 5 Seconds

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.

Attempts have made to implement a neural network with 1 input layer, 2 hidden layer and an output layer is implemented to train the game.

Randomly few states are selected from the q-table whose q-values are not equal to zero as the features to train the network. But promising results are not obtained