ENPM 808F Robot Learning
Name: Suyash Yeotikar UID:116188456
Assignment 3

1) Answer to question 1:

States and actions:
The board state is represented as a string where ‘0’ or ‘1’ represents presence or absence of lines, since it doesnt matter which player has drawn the lines of box, but what matters is who draws the last line in the box (Instead of who has drawn lines, timing of drawing the last lines of box is of the essence). There are 2^12 states. Actions correspond to the indices in the state which can be replaced, to represent a line. An empty Q table is initialized as dictionary. The key to the dictionary is given by the tuple (string,action), with value as the Q value of the state and action pair.

Self-Play:
2 players use the concept of Q learning to play and learn against each other. Both players refer the same qtable to access and update qvalues, based on the bellman equation formula.

Epsilon greedy policy:
The epsilon greedy policy is implemented, to decide whether to explore new states or exploit existing ones. An epsilon value of 0.2 is set.A number is generated randomly between 0 and 1. If it is less than 1-epsilon then exploration is done or else exploitation is done. The idea of decaying epsilon as the number of states explored increases was also explored.

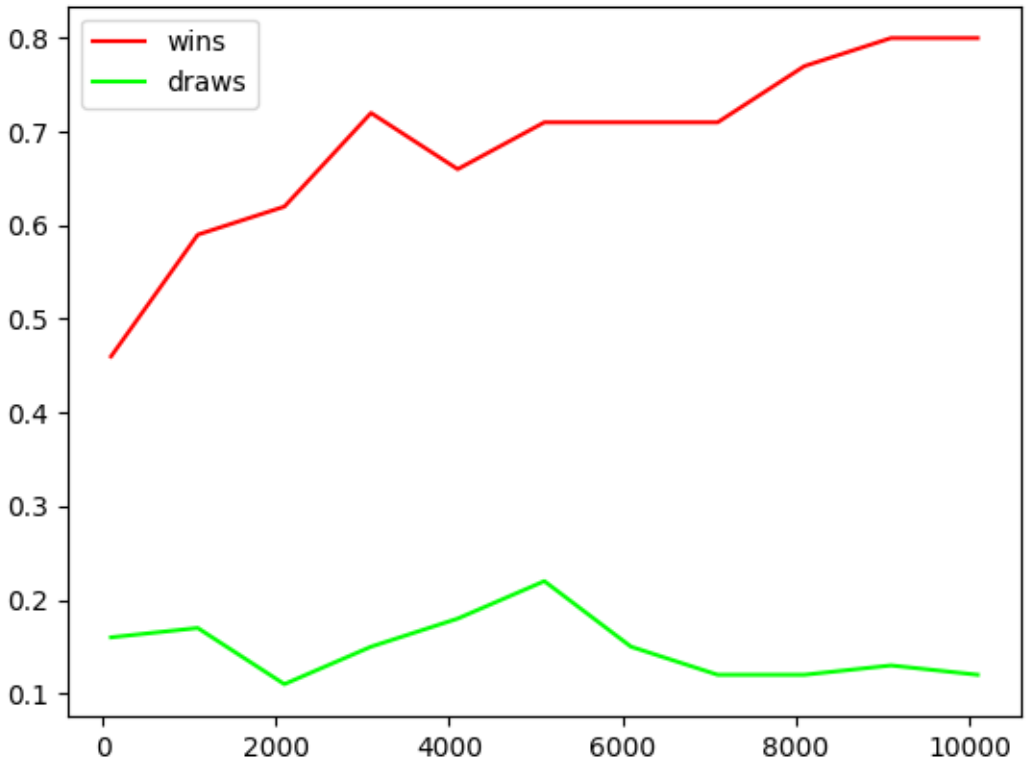
Random play:
Once, certain number of self play games are completed, the final Q-table is used to play against a random player, who chooses its actions randomly. All random play trials were done for 100 games.

Algorithm flow:
First an empty board and q table is initialized. The turn of a player is randomly decided in each game. Then the game is started. Q-learning function for each player is called as the game progresses. In this function, first the epsilon greedy policy is implemented. Depending on the value of epsilon either the player chooses a random action or an action that gives the maximum value from q table. (Since, initially no value is present in q table, a random action is chosen). The player then checks for the reward. If a reward was received the Q-table is updated, and the number of boxes for each player is incremented. (In case a game ends, the player with highest number of boxes receives an additional reward of +5). The game ends if the sum of boxes of both players is equal to 4. Then using the same Q-table a new game is initiated, and the Q-learning function is called again.

Finally, when a fixed number of games are played

Observation and Results:

The number of wins and draws against the random player were plotted as the function of number of games was plotted as below:



A steady increase in the number of wins as the number of games increases can be observed, with a few fluctuations. It can also be observed that after 6000 iterations the number of draws has reduced and remained consistent. If the total number of wins and draws together are considered then the learned player could have either won or had a draw match for a maximum of upto 93% (80 wins and 13 draws) times. Considering the high accuracy it can be considered that the agent has successfully learned.

2) Answer to question 2:

For a 3x3 grid the flow of the algorithm is similar to that of question 1 except now the number of states have been increased to 2^24, as now 24 lines can be drawn. The learned Q-table from the 2x2 can be leveraged to initialize the Q-table for a 3x3 grid. For this purpose the Q-table of 2x2 grid is saved in a file.

Seeding 2x2 Q table to 3x3 Q table:

A 3x3 grid can be said to consist of overlapping multiple 2x2 grids placed at different locations.
Each state of the 2x2 grid Q-table can be placed (super-imposed) in these constituents 3x3 grids, with the other constituents of 3x3 grid as empty. The super-imposed 3x3 grid state also represents one of the valid 3x3 grid states. Not only, that the action providing maximum reward in these states would also correspond to the action that provides maximum reward in the 2x2 case. Hence, the modified state and action pair as keys, and exact same q values can be added to 3x3 grid Q-able.

Modification of keys and values:
Since the 2x2 state is represented as a string, it can be spliced into vertical and horizontal lines.

Example:
0,1 (bottom most horizontal lines) | 2,3,4 (bottom most vertical lines) | 5,6 (middle horizontal lines) | 7,8,9 (middle vertical lines) | 10, 11 (top most horizontal lines)

Here, 0 to 11 are indices of the string and ‘1’ at this index represents that the line is drawn and ‘0’ represents it is absent.
A state 01|110|11|001|00 can be appended with some zeros in between to get a state in 3x3 grid with similar representation. The zeros in inverted commas are the appended ones.
01’0’|110’0’|11’0’|001’0’|00’0’|’0000’|’000’
Here the 2x2 grid is imposed at the bottom left 2x2 grid in 3x3 grid.

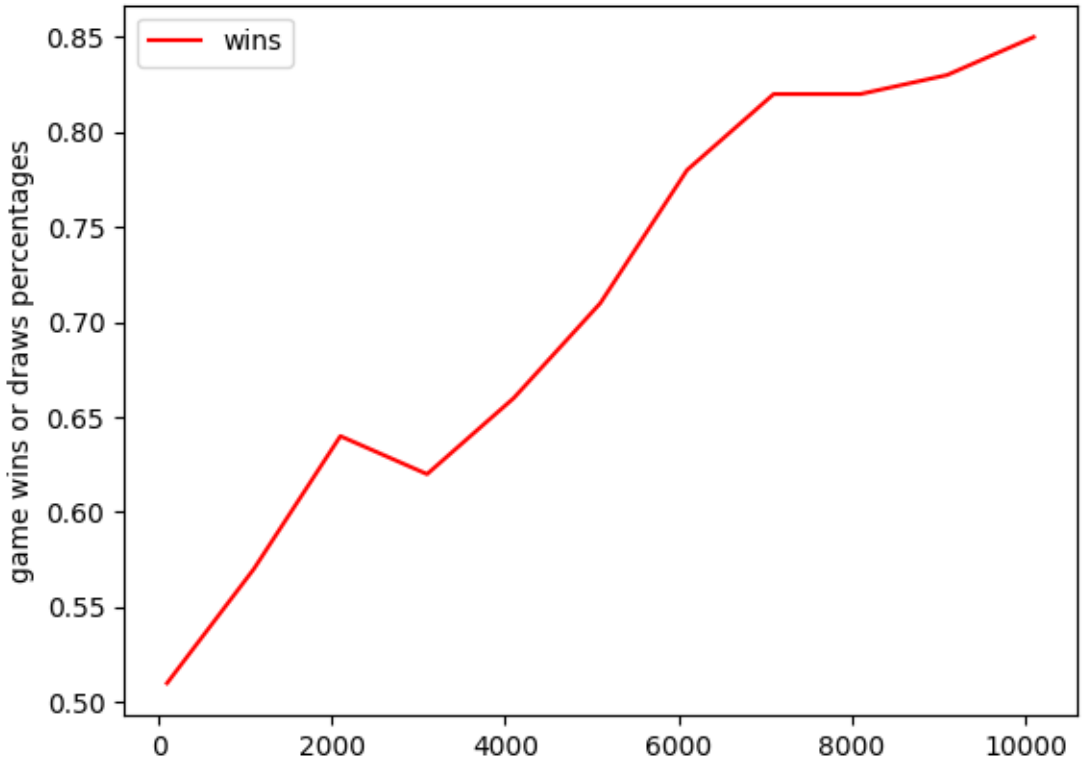
Hypothetically if the best action to perform in the 2x2 grid was at index 4 (110) then now the best action to perform will be at index 5. Hence, for each zero appended before the index of the best action has to be incremented.

Similarly, by appending 0’s at other locations the 3x3 grid states can be obtained from 2x2 grids, and thus since the best action to do would be similar in the 2x2 grid case, the same set of q values can be used for the modified state action pair and appended to the Q table.

The Q-table from the 2x2 grid is saved in the modified state and action pair in the csv file (refer readme) and thus initialized instead of an empty Q table.

Apart from the above modification in Q table initialization and the game completion condition which is now that the number of boxes should be equal to nine is, the rest of the flow of the training and test algorithm is the same.

Observations and results:
A steady increase in the number of wins can be observed in the 3x3 grid case. In the 3x3 grid case there are no draws, hence a significantly higher increase in the number of wins can be observed. There are still some fluctuations observed, but they are significantly low.



Answer to Question 3:

Function approximation for 2x2 grid Q table:

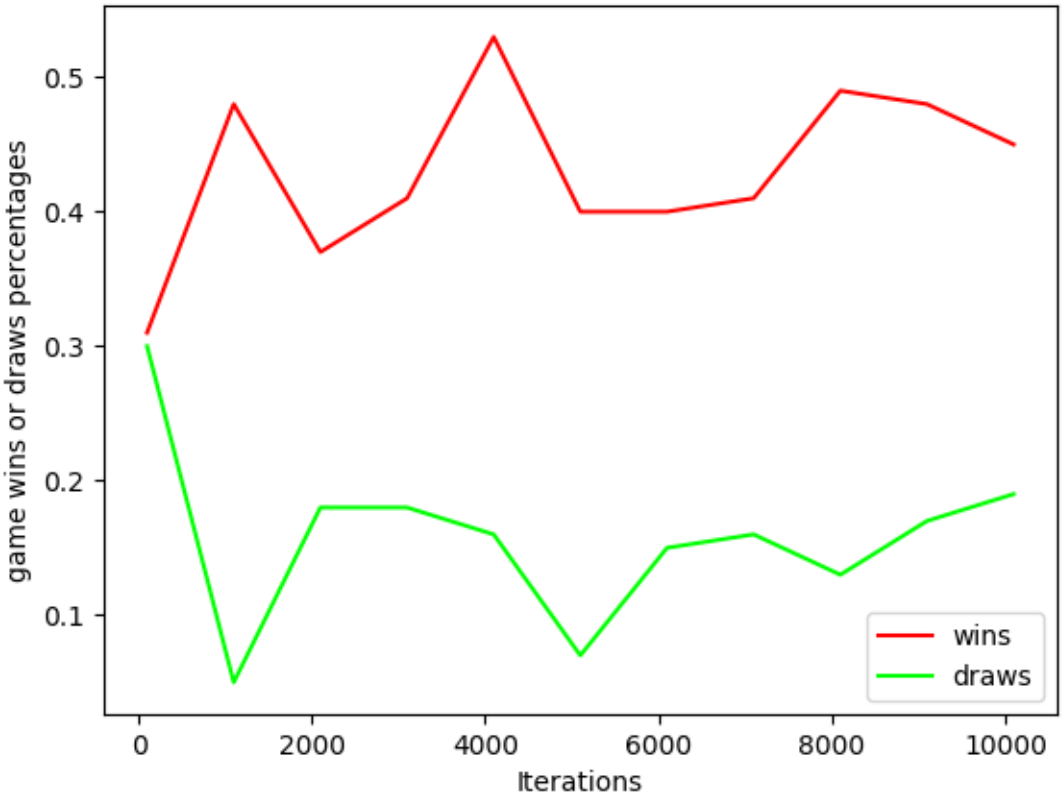
The Q-table for the 2x2 grid was replaced with an Artificial neural network (multi-layer perceptron). The following were its characteristics:
1) Architecture:
Input: The state is now converted an array of 1's and 0's. The actions are also one-hot encoded (ie. an action of drawing line corresponding to index 1 in string will be encoded 010000000000). The state and action arrays are appended and provided as input to the neural network. (Size: 1x24)

Activation function: Since the output of neural network has to be a Q value for a given action which can be any real number, the activation function cannot be sigmoid. The activation function for the neural network was chosen to be the linear activation function (f(x) = x), since it is not bounded in terms of its output.

Layers: The neural network has three layers: input, output and a hidden layer. Input layer has 24 neuron, hidden layer has 80 neurons and output layer has 1 neuron. The hidden layer has more number of neurons than the input layer. It can be thought of as a decoder that expands the state,action pair and depending upon the decoding combines them to give a Q value. The biases for each layer are set to be 0 and thus not considered while weights are initialized randomly and updated using the back propagation algorithm.

Training the neural network:
The neural network is trained using the back propagation algorithm. Given a state and action pair the expected output of the neural network after forward propagation is the updated Q-value as per the bellman equation for that particular state and action. Hence, the neural network is trained on the error (difference) between forward propagated value (old Q value) and updated Q value . The training is done till the error is less than a threshold value. Thus when forward propogation will be run the neural network will give a value close to the updated Q value. This is the incremental training format. The error is taken as their difference implying that the loss function for the neural network is mean squared loss function. The learning rate for the neural network was set to 0.001

Observation and Results:
The plot of wins and draws (y axis) vs. the number of iterations (x axis) is shown below.



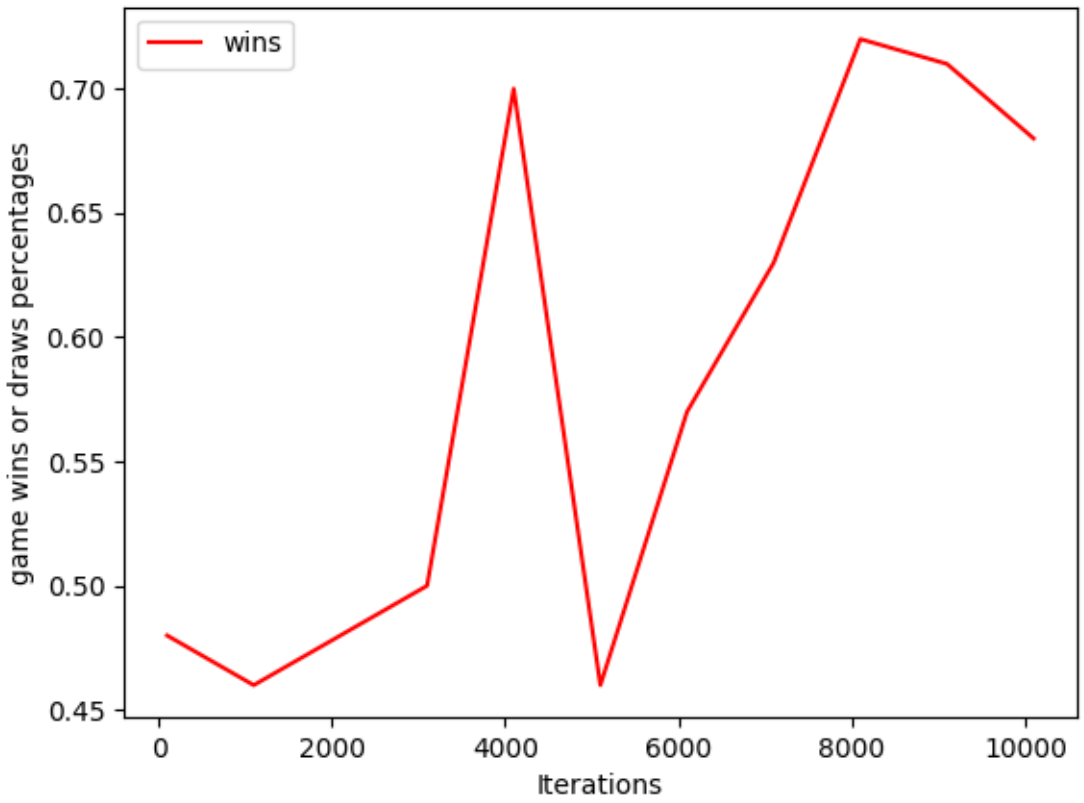
It is observed that there is an increase in the performance (on an average) as the number of iterations increase, however a lot of fluctuations can be observed in the performance (wins) as compared to the case of using a Q table. It can also be observed that the level of performance is not as good as using a Q table. There are higher percentages of games that were draw as compared to the Q table case. The observations can be attributed to the speculation that since the neural network is a function

approximator it cannot give correct Q values which would have been there in the table in some cases of forward propagation. Another speculation is that the neural network to give a performance as good as the Q-table requires higher number of iterations. Other aspects that could effect the performance are number of neurons in hidden layer, and the number of layers, the epsilon greedy policy, etc.

Function approximation for a 3x3 grid:

The function approximation for 3x3 grid Q-table is similar as the above case with a change in the number of hidden layer neurons to 100 and the input vector to neural network is now a 48 size vector with 1st 24 1's and 0's corresponding state and the last 24 1's and 0's corresponding to one hot encoded actions. The rest of the parameters are similar to the 2x2 grid Q table function approximation case.

Observations and Results:
The plot of number of wins vs. number of iterations for the neural network approximation of Q table is shown below:



Huge fluctuations in the performance of the neural network can be observed. After about 5000 iterations, the performance increases. It remains relatively high. As it was observed in the 2x2 grid function approximation of Q table, significant fluctuations can be observed, which can be attributed to similar reasons as mentioned in the 2x2 grid case. The only difference that can be observed here is that the here the performance of the neural network after 10000 iterations is comparable to the 3x3 grid Q table. This could possibly be because of no draw games, and the wins tipping in favor of one player.