Janakiraman Kirthivasan
UID:116349086

Robot Learning Assignment 4

The idea behind building the states and actions of the system is as described below:

For a given board type of mxn, the number of possible Actions is given by
p = m*(n+1) + n*(m+1).

The number of states is determined by the fact that if a given line is being drawn or is empty.

Hence the total number of states = 2^p. Each state can be visualized in the binary format. For example, for the case of 2x2 board, with 12 lines possible being drawn. A state of all the boxes being filled is encoded as 0b111111111111.

On drawing a line in the 2x2 grid, the new state transition is determined by the following calculation given below:

New state = current state | (0x1 << (line drawn-1))

Example in case the board is in the states 0b000000000001. And the 12^{th} line has been drawn. Then the new state is determined by:

New state = 0b000000000001 | 1<< (12-1) = 0b100000000001

This ensured a faster calculation with lesser processing time. Based on the following encoding schema, the various tests are performed and detailed below

1)

A 2x2 board is been evaluated using Q table wherein the player is trained through Q learning. The table dimension is equal to numOfStates x numOfActions which in the case of 2x2 board equal to 4096 x 12.

The Q player is trained through self-play. After the training the player is tested by playing with a random player and the results are documented. The Q Learning parameters are set as below:
learning Rate = 0.5
discount Rate = 0.5
Epsilon value = 0.2

The player is trained for three cases of 100, 1000 and 10,000 games and played against an opponent. In the test phase the Q player plays 1000 games against a random player and the win % is calculated

It is observed as the number of games for which the player is trained is increased, the performance of the Q player also increases. Below are the results:

NumOfGames	Win %

100	40
1000	52
10,000	69

2. The player is now trained using Q Learning with function approximation. The RBF stochastic gradient descent is used as a function approximator. 5 RBF kernels are used with following parameter values:

RBF1- , standard deviation – 5.0, and number of components = 100

RBF2- , standard deviation – 2.0, and number of components = 100

RBF3- , standard deviation – 1.0, and number of components = 100

RBF4- , standard deviation – 0.5, and number of components = 100

The RBF stochastic gradient descend functionality is implemented using scikit learning library of python.

The Q learning parameters are the same that is used in section 1

The below table indicates the win % for 100, 1000 and 10,000 games:

NumOfGames	Win %
100	45
1000	50
10,000	53

The values seems to be quite close, this is because of the RBF not converging to the optimum and hence we do not see a satisfactory output for the number of iterations. An instability is also seen in the win percentage as the test games are executed repeatedly for different trials. The variation is +5 or -5

3.

The dots and boxes games are further extended to be played on a 3x3 board using QTable. Below the number of states and number of actions for a 3x3 board is indicated:

Number of Actions = 3*4 + 4*3 = 24

Number of States = 2^24 =

In this step the Q table trained using 2x2 is used to seed the 3x3 table. The procedure that is followed is the Q values of all the states and actions of the 2x2 board is copied into the 2x2 area of the 3x3 board.

Then the table is further trained for 100, 1000 and 10,000 games. The learning rate, discount rate and epsilon values are maintained the same as in the case of 2x2 board:

NumOfGames	Win %
100	43
1000	52
10,000	65

4.

The dots and boxes game are trained using Q Learning with function approximation for a 3x3 board. The functional approximation is performed using RBF as in the case of 2x2 board. The same kernel parameters along with the Q learning parameters are used. The results are detailed below:

NumOfGames	Win %
100	44
1000	49
10,000	52

Note:

1. **Dependencies for the project:**
    a. Python 2.7 or Python 3
    b. Sci Kit learn python library. In specific following python libraries has to be working and to be imported:
        i. Sklearn.pipeline
        ii. Sklearn.preprocessing

2. The repository downloads and execution instruction are given in github Readme.md file
3. The values in the table are rounded to the nearest integer and a median value had been picked up among the various tests that are performed