Machine Learning Assignment 1

Sourav Chakraborty

Raja Mummidi

Lata Yadav

Q1.

Task (T): Playing Tic-Tac-Toe

Performance (P): Percent of games one against opponents

Training experience (E): playing practice games against itself

Let the chosen board features be:

1. x_1 : number of x's
2. x_2 : number of o's
3. x_3 : number of two x's in one row, column, diagonal
4. x_4 : number of two o's in one row, column, diagonal
5. x_5 : number of x in a winning position
6. x_6 : number of o in a winning position

Let b denote the current board state and B denote the set of legal board states. The target value V (b) for an arbitrary board state b in B, as follows:

1. If b is a final board state that is won, then V (b) = 100
2. If b is a final board state that is lost, then V (b) = -100
3. If b is a final board state that is drawn, then V (b) = 0

4. If b is a not a final state in the game, then V (b) = V (b'), where b' is the best final board state that can be achieved starting from b and playing optimally until the end of the game.

We can define the learning function $\hat{V}$ (b) as a linear function:

$\hat{V}$ (b) = $w_0 + w_{1x_1} + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6$, w_0 through w_6 represent the weights chosen by the learning algorithm.

The training value of V_{train} (b) or any intermediate board state b to be $\hat{V}_{successor}$ (b) where $\hat{V}$ is the learner's current approximation to V and where successor (b) denotes the next board state following b for which it is again the program's turn to move.
Then,
V_{train} (b) $\leftarrow$ $\hat{V}_{successor}$(b)

Issues in setting up Tic-Tac-Toe Problem:

1) Determining value of V (b) for the particular board state requires searching ahead the optimal way of play, all the way till the end of the game. So this value V (b) is not efficiently computable.

Q2.
For the Tic-Tac-Toe problem, let us consider the following attributes:

1. top-left-square: {x, o, b}
2. top-middle-square: {x, o, b}
3. top-right-square: {x, o, b}
4. middle-left-square: {x, o, b}
5. middle-middle-square: {x, o, b}
6. middle-right-square: {x, o, b}
7. bottom-left-square: {x, o, b}
8. bottom-middle-square: {x, o, b}
9. bottom-right-square: {x, o, b}
10. Class: {positive, negative}

Here, x represents the square has 'x', o represents the square has 'o' and b represents the square is blank. Every square can have either of the 3 values.

There are 3*3*3*3*3*3*3*3*3 = 19683 distinct instances.
5*5*5*5*5*5*5*5*5 = 1953125 syntactically distinct hypotheses within H.

Every hypothesis containing one or more "∅" symbols represents the empty set of instances, so it classifies every instance as negative. The number of semantically distinct hypotheses is 1 + (4*4*4*4*4*4*4*4*4) = 1+ 262144 = 262145

Considering the size of the input data set (rows), the chance of generalization can be considered small.
However, the performance will also depend a lot on the ratio of positive to negative scenarios in the data set