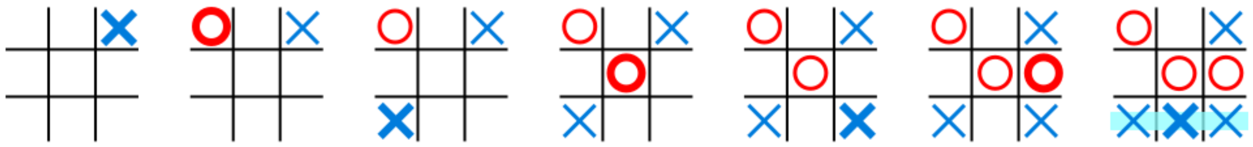
CIS 4930 - Intro to Machine Learning
Assignment 1

Overview: This is an individual assignment. You may discuss but must submit your own original work.

In this assignment, you will evaluate the performance of classifiers and regressors on a game of tic-tac-toe. Your single classifier will generate accuracy and confusion matrices of several classifiers and regressors using the provided datasets. Both players in our games will be AI players. Your program will learn from these games are produce reasonable player moves for one of the players in the game.

Details:

You will find details for installing python here, and useful starting information on scikit-learn here. You will find the latest version of scikit learn here.

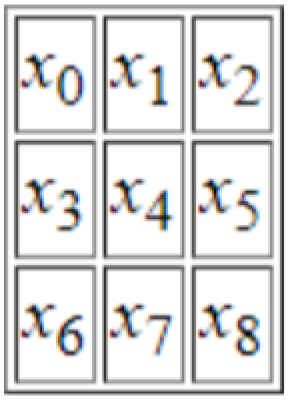


Rules:
You can get familiar with the rules of the TicTacToe game here. The optimal way of playing can be obtained here.

Datasets:
Your program will evaluate the performance of simple classifiers and regressors on the provided synthetically generated Tic Tac Toe games. There are two computer players. The X player is denoted as + 1. The O player is denoted as −1. Empty squares are denoted as 0. The players in the games make optimal moves. Your goal is to learn from these games to produce reasonable moves for player O (−1).

There are three datasets posted to canvas.

Final boards classification dataset:
The final boards classification dataset is a binary classification task for boards where the game is over. The input features are x_0,x_1,x_2….,x_8, the states of each Tic Tac Toe square:

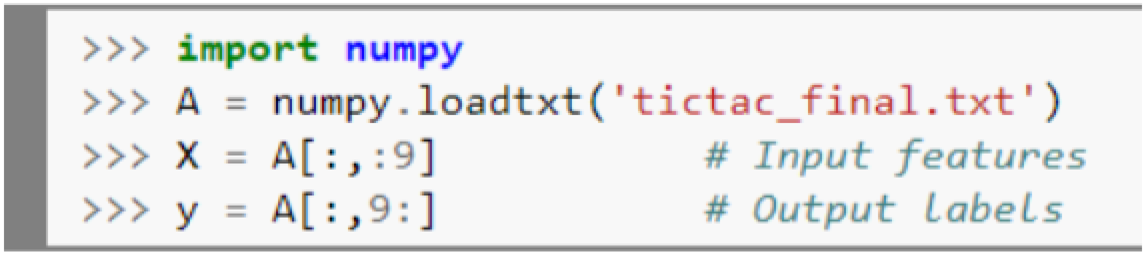


The output feature y is the winning player. Each line of the input file lists the input features followed by the single output feature, as:x_0,x_1,x_2….,$x_{8,}$ y.

Intermediate boards optimal play (single label):
This is a multi-class classification task where the board is set up so that it is O player's move, and the goal is to predict the next move that is optimal for the O player (i.e.player −1).
Inputs are x_0,x_1,x_2….,x_8 , the states of the Tic Tac Toe squares for a given board, and the output Y is the index of the best move for the O player (player −1).

Intermediate boards optimal play (multi label):
There are usually multiple optimal moves. This dataset can be viewed either as a regression or a classification problem with multiple output labels. Inputs are x_0,x_1,x_2….,$x_{8,}$. The outputs are y_0,$y_{1,}$... Y_8, where y_i is 1 if the given square is an optimal move for player O, otherwise it is 0. Each line lists the x_i inputs followed by the y_i outputs.

You can load these data sets into python by using numpy :



```
>>> import numpy
>>> A = numpy.loadtxt('tictac_final.txt')
>>> X = A[:,:9]              # Input features
>>> y = A[:,9:]              # Output labels
```

Implementation:
Report performance of these classifiers : linear SVM, K-nearest neighbors, and multilayer Perceptron. You can manually choose the k value in KNN. and the architecture of multilayer Perceptron. Be aware of overfitting risks. You should use k-fold cross validation (eg. using 10 folds)

Please write a single program that outputs accuracy and confusion matrices for both datasets and for all the classifiers. You will also need to randomly shuffle the order of dataset before using Cross-validation.

Report performance of these regressors: on the intermediate boards optimal play dataset: KNN, linear regression and multilayer perceptron.

Please write a single program that outputs accuracy and confusion matrices for both datasets and for all the regressors. You will also need to randomly shuffle the order of dataset before using Cross-validation.

Evaluation : record the accuracy and normalized confusion matrices for the 3 classifiers and accuracy for the 3 regressors. Explain which methods worked in both cases and why.
Implement a simple command line tic tac toe game : where a human (player X, i.e. +1) can play the best ML model (player O, i.e. -1). You can use the intermediate play classifier for the entire game. Describe whether you can beat the computer and how hard it is for the 3 regressors.

Extra Credit:

1. Implement Linear Regression using Normal equations. 10 points
2. Train the models on 1/10th of the data and explain what happens. 5 points
3. Investigate and report what happens when a substantial fraction of ground truth values are corrupted by random noise. Explain why certain models scale better to larger dataset than others. 5 points

Grading :
1. Source code and correctness (30 percent)
2. Results from classifier and regressor evaluation (written evaluation and program) (30 percent)
3. Gameplay performance of your program (20 percent)
4. Record a video tour of your code (5 minutes) (10 percent)

5. Written report (10 percent)

Submissions:
Submit your report, source code and the video in one zip file.

1. Complete source code including any 3rd party libraries.
2. A program that automatically prints the required classifier and regressor results.
3. A program that automatically implements tic tac toe gameplay with a human.
4. Any other input required to run your code that is not mentioned in this list.
5. A one or two page written report
    a. Explanation of implementation
    b. Evaluation results
    c. Instructions on how to run your code
    d. Any bugs, difficulties you would like us to know
6. 5 minute video tour of your code.