CIS 6930/4930 Deep Learning for Computer Graphics
Dr. Corey Toler-Franklin

Part I. Classifiers and Regressors
DUE: October 8th, 11:59 pm

Overview | Details | Resources | Getting Help | Submitting | Acknowledgements

Overview

Collaboration Rules: The entire course project may be completed in groups of one, two or three. Although you may discuss concepts between groups, each group must submit their own original code.

Course Project Part I: In this project, you will evaluate the performance of classifiers and regressors on a simple game of *Tic Tac Toe*. Your single classifier program will generate statistical accuracy and confusion matrices for several classifiers and regressors (linear regression, linear SVM, k-nearest neighbors, and multilayer perceptron) using provided datasets. Each game play session will be synthetically generated. The two players in your games are both AI players. Your program will learn from these games to produce reasonable player moves for one of the players in the game.

Details

Getting Started

Set Up Your Code Base: This project will be developed using Python and the scikit-learn module. Although scikit-learn is supported on Windows, macOS and Linux, we recommend that window users install Ubuntu Virtual Box (instructions posted to canvas) and work with Python and scikit-learn in a Linux environment. You will find information for installing Python here, and useful *getting started* information on scikit-learn here. Use the latest development version of scikit learn found here, which you must build from the source code. This will enable you to use the multilayer perceptron classes in scikit-learn.

Figure 1: Tic Tac Toe. Image Source: Wikipedia:Traced by User:Stannered

Learn Game Rules: If you are not familiar with the game of *Tic Tac Toe*, you can find background reading and game rules here. For a simple 3x3 Tic Tac Toe game, the optimal game play can be obtained trivially by using minimax described here. More complex games such as *Go* have larger state spaces that require machine learning to evaluate the quality of an intermediate game board. AlphaGo by Google is an example (see the Nature paper here), which recently beat professional Go player Lee Sedol.

Examine 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 $x_0, x_1 \cdots, x_8$, the states of each Tic Tac Toe square:

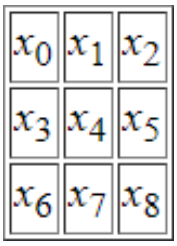

Figure 2: States

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: $x_0, x_1 \cdots, x_8$, 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 $x_0, x_1 \cdots, x_8$, 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 $x_0, x_1 \cdots, x_8$. The outputs are $y_0, y_1 \cdots, y_8$, 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 trivially load datasets into Python by loading the data matrix and then selecting out columns from it:

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

Figure 3: Loading data

Implementation

Report performance of these classifiers on the above two classification datasets: linear SVM, k-nearest neighbors, and multilayer perceptron. We have included links to the relevant classes in scikit-learn. You can manually choose the number of neighbors k, and the architecture of your

multilayer perceptron, although be aware this introduces some overfitting risk. You should use k-fold cross validation (using e.g. 10 folds).

Please write a single classifier program that outputs the statistical accuracy and confusion matrices for both datasets and for each of these classifiers. See canvas posting for details on confusion matrices. Please normalize the confusion matrices so that each row sums to one. Note that none of the datasets are randomly shuffled so you should randomly shuffle the order of the dataset before using cross-validation.

Run these regressors on the intermediate boards optimal play (multi label) dataset: k-nearest neighbors, linear regression and multilayer perceptron. For linear regression, implement the solution yourself via normal equations. Note that one linear regressor can be trained independently for each output. For simplicity, you can use a squared loss function.

Write a single regressor program which evaluates each of these regressors. Since this is really a multi-label classification problem that is being treated as a regression problem, please report the accuracy of the multi-label regressor as if it were a classifier, by rounding the output of each regression output to either 0 or 1 and comparing with the testing data. Use k-fold cross-validation also when assessing the accuracy for this part.

Evaluation: Record the accuracy and normalized confusion matrices for the 3 **classifiers, and the accuracy for the** 3 **regressors. Explain which method worked best for classification and for regression, and why. Investigate (and report) what happens to the accuracy of the classifiers if they are trained on** $\frac{1}{10}$ **as much data, or a substantial fraction of the ground truth labels are corrupted by random noise (this could be a simple way to model the inclusion of non-optimal gameplay in the training set). Explain why certain methods scale better to larger datasets than the others. Hint: the multilayer perceptron should work better than k-nearest neighbors regressors, but they both should have above 80% accuracy, and should play a decent game of Tic Tac Toe in the next step**.

Implement a simple command-line Tic Tac Toe game where a human (player X, i.e. $+1$) can play the best machine learning model (player O, i.e. -1). You can implement the computer player as follows: suppose the computer is given a current board. Then find the output of the regressor that is highest for the given board, which is not already taken by another player's move, and make the computer move that corresponds to the choice most preferred by the model. You can use the intermediate play classifier for the entire game. **Describe in your write up whether you can beat the computer, and how hard this is for the three regressors**.

Explain your code for us: Make a 5-7 minute video screen tour walking us through your code, explaining what you did in you own words. Also run your code and show us the program generating results at the command line. We will suggest programs for doing this in canvas. You may however use any program you like. These are only suggestions.

Optional extra credit: the above machine learning setup requires an expert human or computer player to observe and learn in a supervised manner from. You can explore the use of alternative techniques such as genetic algorithms.

Grading: You will be evaluated based on the following criteria (1) source code correctness and completeness 30%, (2) results from classifier and regressor evaluation (your written evaluation, and our output running your program) 30%, (3) the performance of your human gameplay program, your assessment of your ability to beat the computer and your assessment of the classifiers 20% (4) video screen tour of your code 10% (5) written report (all other items not scored in 2 or 3) 10%.

Resources

Instruction documents referenced in this assignment will be posted to canvas.

If you are not in the CISE department, and would like computer lab access, you can register for a CISE account at https://www.cise.ufl.edu/help/account. Linux systems are available for remote access top complete this assignment. **Remember to back-up your work regularly!!!** Use version control to store your work. DO NOT PUBLISIZE SOLUTIONS.

Getting Help

Send me an email at ctoler@cise.ufl.edu for any issues or questions.

Office Hours are conducted in my personal meeting room in Zoom. A link to this meeting room and available hours are posted on canvas.

Collaborating You may discuss this assignment with each other but each group should submit individual work. Remember to always credit outside sources you use in your code. University policies on academic integrity must be followed.

Submitting

Upload one zip file to Canvas. If your zip file is too large for a canvas upload, you must submit your written report to canvas by the due date and upload your zip file to a file share provided (see canvas) by your instructor by the due date. All submissions (Canvas or otherwise) must be completely uploaded by the due date and time. This assignment is subject to the late policy for course assignments.

Your submission should include everything needed to test, run and understand your code including:

- The complete source code including any third party libraries.
- A program that automatically prints the required classifier and regressor results.
- A program that automatically implements the *Tic Tac Toe* gameplay with a human.
- Any other input needed to run your code that is not mentioned in this list. For example some programs need images icons or other support data to run out-of-the-box.
- A one to two page written report that includes:
  - An explanation of your implementation.
  - Evaluation results requested in the assignment (see bold text).
  - Instructions on how to run your programs. Your code will be tested on all the provided datasets using scripts.
  - Anything you would like us to know (bugs, difficulties).
- 5-7 minute video screen tour walking us through your code and explaining your implementation. It should be located in a folder in your zip called videotour. You should explain your code and step through the key parts of your code. You also need to run the system so that we can see the results being generated. Program options will be posted to canvas but you may use any program you prefer.

Acknowledgements

Sources and Datasets:
UCI Machine Learning Repository
Connelly Barnes, UCI
David W. Aha