Breda Marie Brady

Dr. Jiao

ECE 5400: Applied Machine Learning

October 29, 2019

Mini Project 1 Report

Overview

In this mini project, we were to apply Machine Learning methods to an erroneous 16x16 bit multiplier in order to predict which bits of the outputs would be produced incorrectly. We were to create 32 different models using our chosen classifier to correspond to each bit of the 32-bit output. To train our 32 models, we determined that our model’s X would consist of a 32 dimensional array with 150000 rows, which corresponds to each bit for each training input to be a separate feature. However, in order to create 32 different models for each bit of the output, our model’s Y would be only a 1 dimensional array in which only the values for a particular bit for all training input. Using this training data, we are then able to fit our models and predict the resulting our of testing data. Comparing the predicted and actual results of the testing data, we are then able to determine the accuracy, precision, and recall of our models.

Description of my code

First step was to import the training and testing data and labels. To do this, I created functions that would open the file, read the contents into a list, map each character into a separate list element, remove any tab or newline characters, converted the list into a numpy array, and lastly, changed converted it to an array containing integers. After importing the data, I created a while loop that loop 32 times – one for each model I created. In each iteration of the loop, I chose my classifier type as the Multi-layer Perceptron classifier method, and I fit my model with the 150000x32 training\_data array as X and 150000x1 training\_label (corresponding to only one bit position) as Y. After fitting my model, I used the 50000x32 testing\_data to produce my predicted Y output. Next, I generated a confusion matrix to display how my model performed in predicting the output. I also produced accuracy score, precision score, and recall score for my model’s prediction for a certain bit position compared against the true values for that bit position. After my while loop created all 32 models, I averaged my results for accuracy score, precision score, and recall score to find the overall mean values of these performance metrics for all 32 models. Lastly, I printed the running time of my program.

Designing my classifier

In working on this mini project, I trained and tested various different classification models in order to produce the best overall accuracy score. Some of the models I tried were Logistic Regression, Decision Tree, Random Forest, Gaussian Naive Bayes, Bernoulli Naïve Bayes, SVM, and Multi-layer Perceptron. I eliminated some of these models, such as Decision Tree and Logistic Regression, due to its relatively low accuracy score. I also eliminated some models, such as SVM, due to its very long running time. I ultimately chose the Multi-Layer Perceptron Classification model as it produced a high overall accuracy score in a reasonable running time.

The MLP Classifier performs classification using Neural Networks. For this classifier, I chose the following parameters: MLPClassifier(hidden\_layer\_sizes=(10,10), solver='adam', alpha=1e-5, random\_state=1).

Activation = logistic

The hidden\_layers\_sizes parameter refers to

My final mean accuracy/precision/recall for all bit positions

My program running time

My program output screenshot