Voted Perceptron

Cashton Holbert

Outline:

In this project, we are asked to use a voted perceptron model to classify given test data points as either labels 0 or 1. This algorithm works by using multiple weighted perceptron margins. It creates a new perceptron with weights after each iteration being incorrectly classified. Each time it correctly classifies it will simply increment the value of the "*c"* weight for that perceptron and continue using the same perceptron as before. In our case, after training the model, it makes a prediction on each new test data point by utilizing the weight value "*c*" and the sign function in the algorithm outlined below …

$$\hat{y} = sign(\sum_{k=1}^{K} c_k sign(w_k x))$$

… where w is the array of weights, x is some testing data point in n-dimensional space, k is the number of misclassifications as we trained, c_k is the ranking weights of v when we classified correctly, and y_hat is the predicted output.

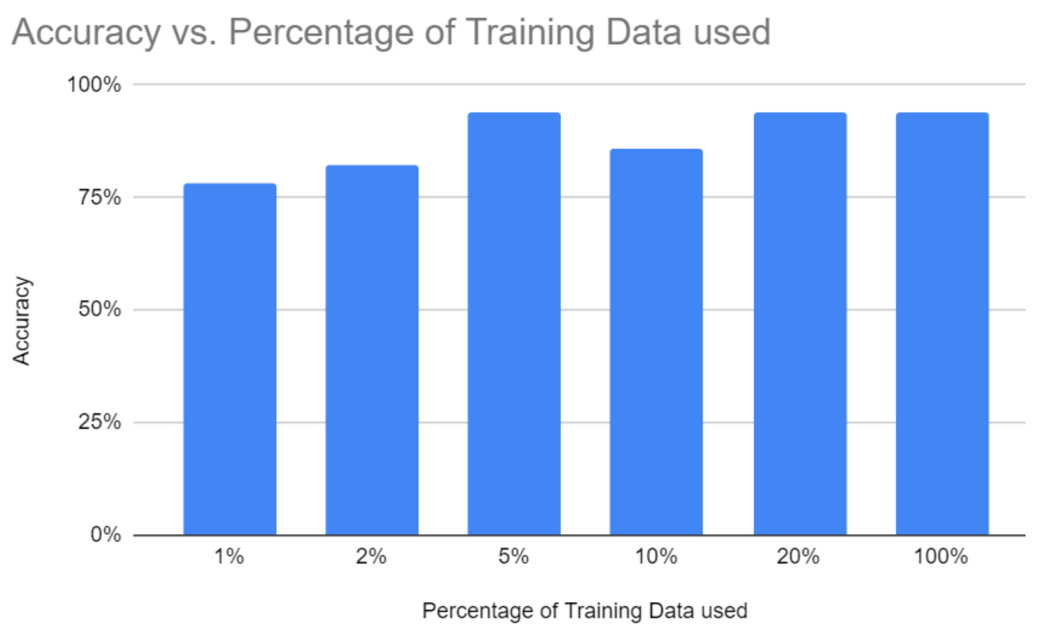
How is it done?

The main program loads the files Xtrain.csv, Ytrain.csv, test_file.csv, and pred_file.csv then calls run(...), where the parameters of run are all those loaded files. I first change the labels to be -1 if it was previously 0 as to make the sign function arithmetic work. I then initialize the necessary variables, k, v, and c, to be 0 as to update them as needed. Note: the weight will start at 0, and will likely need to be updated on its first pass. From there I train the model on the training data and make predictions on the testing data doing the same process as I outlined above. I then change the -1 labels back to zero to match the desired output and save those predictions into pred_file.csv.

Given data

Xtrain.csv: list of n-dimensional training data points

Ytrain.csv: list of the corresponding classification labels of Xtrain.csv



Analysis:

The graph above shows the change in accuracy of the data set as split from the given Xtrain,Ytrain set. In this graph the program was run on increments of the first 90% of the training data; 1%, 2%, 5%, 10%, 20%, where the testing data was the last 10% of the original training data. It can be seen from the graph that as we get larger and larger training data, the accuracy of the model likely increases. The size of the training set seems to have diminishing returns at higher values.