README

The data provided in a7a.train and a7a.test are from the “Census Income” Dataset. The objective of the assignment was to predict whether the income of each tuple exceeded $50k/yr. The first column of the data was the class label which took the value of 1 or -1. For more information, please visit: http://archive.ics.uci.edu/ml/datasets/Adult

Each tuple has a total of 123 features. Given that this was a binary classification problem, the perceptron algorithm was implemented.

REQUIREMENTS:

Please have the numpy library installed and imported to run this program. The use of arrays is essential to complete dot products and array summations. Data for this is stored in /u/cs246/data/adult/ on the csug machines.

FUNCTIONS:
There are 5 functions defined in the file Yakubov_perceptron.py:

The first is the parse_line which will parse each line in the data and add another value into the list to incorporate the bias. Thus, each tuple will have a total of 124 values.

The second is parse_data which will return an array of the class labels of each tuple in the data and second, an array of the actual tuples within arrays.

The third function is the implementation of perceptron. The algorithm works by running through the number of iterations requested in the initial for-loop, followed by a for-loop that sweeps through each data point. The weight array is initialized to zero and is updated if and only if the dot product of the weight vector with the datafile multiple by its actual class label yields a number less than or equal to 0. Thus, adjustments are only made to the weight array when the prediction is incorrect. The final weight array is returned for the last iteration.

The fourth function is predict, which takes the weight array returned from the perceptron algorithm and runs on a test tuple with a given class label for that test tuple.

The fifth function, test_accuracy, then iterates through the test data and feeds it into the predict function. The output of the product function is either a 1 or a 0. 1 if the prediction is correct, and 0 if the prediction is wrong. These outputs are recorded in the accuracy function in a list, and then summed, and divided by the total to return the accuracy.

TESTS:

The data included a development set, which can be used. To run the data without using the development data, please run as follows: ./Yakubov_perceptron.py —nodev —-iterations ? —-lr

? . The question marks should be filled with the iterations and learning rate requested for the algorithm. This command defaults to using the training set, and then the testing set for creating the perceptron model.

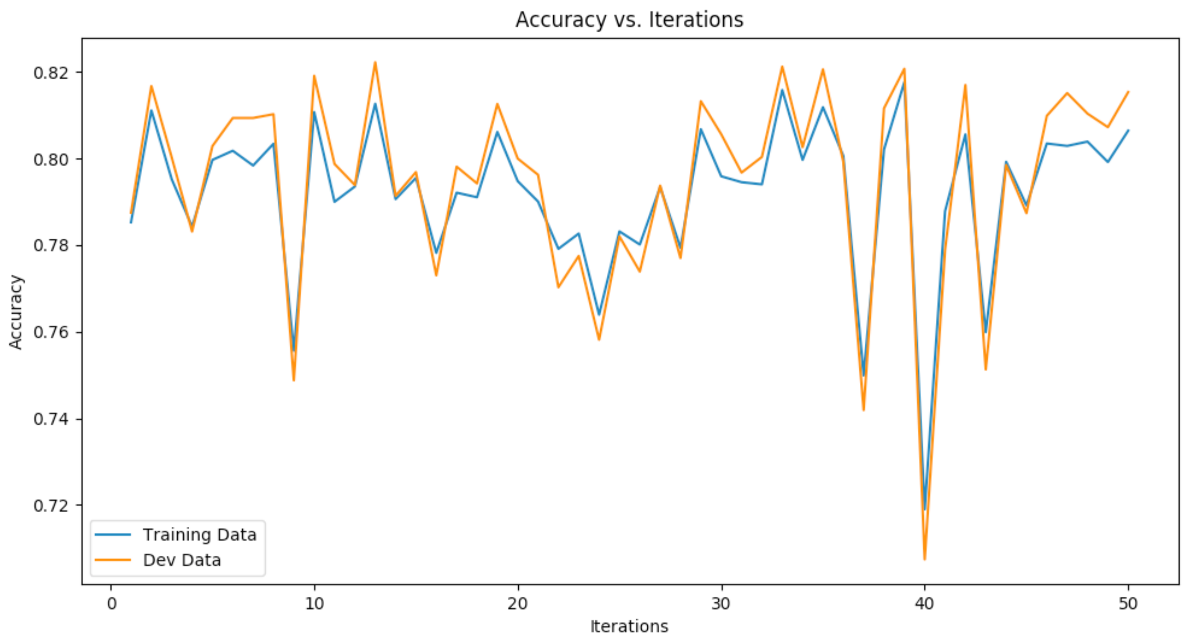
If running with the development set, simply run the following on terminal: ./Yakubov_perceptron.py —-iterations ? —lr ? . Again, the question marks should be filled with the iterations and learning rate requested for the algorithm. This command defaults to using the training set and then the testing set for creating the perceptron model. The development data accuracy for the last iteration will be returned along with the Test Accuracy and Feature weights.

DISCUSSION:

The perceptron algorithm works exceptionally well when the data provided to it requires a binary classification model and is Linearly Separable. If the data is not linearly separable, the algorithm will continue to run, and will not converge. This is because it will continue to search for the optimal decision surface, which will keep it running. Thus, the development data is used in which the algorithm is run to determine an optimal stopping number for iterations. This number of iterations is then used on the testing set. The development data is similar to the validation data in which the best performing approach is used on the testing data.

As you can see from the testing data and the development data accuracy points for a total iteration of 50, the accuracies are quite similar. This suggests that there is very minimal overfitting present. The development data is the validation set, and since this is data that the model has not seen, it is used to determine the strength of the parameters in the data. When comparing the validation set to the testing set, it is reasonable to believe that overfitting was minimal.

FIGURE 1


Accuracy vs. Iterations
Accuracy
Iterations
0.82
0.80
0.78
0.76
0.74
0.72
0
10
20
30
40
50
Training Data
Dev Data