Project 4 Report

Eddie Tribaldos, et7226

In this report I will discuss the approach taken to complete the assignment, the general algorithm employed, instructions on running the system, and insightful discussion on experimental results obtained. This will include discussion on the following aspects of the results:

1. Comparative accuracy of the algorithms at different points on the learning curve for the training data.
2. Comparative accuracy of the algorithms at different points on the learning curve for the testing data.
3. Comparative running times of the algorithms in training and testing phases. Include a summary table of training and testing times for each algorithm, as reported by CVLearningCurve.

Approach and Algorithm

Both algorithms were implemented as described on the slides, except Rocchio had the added scaling for the max weight. For KNN, the training function put all the examples into an inverted index. The files were also stored in a HashMap that mapped the files to categories. The testing consisted of querying the inverted index using the example's hash vector, and returning the category with the most occurrences within the top k results. The training function for Rocchio consisted of first making an inverted index, creating an empty hash vector for each category and then adding the hash vectors of all examples mapped to a category to that category's vector. The vectors for each example are first normalized by the maximum weight of a token, and then the individual tokens are multiplied times it's idf.

Running

This program can be run by using the commands:

```
java ir.classifiers.TestKNN [-K k]
java ir.webutils.TestRocchio [-neg]
```

Results

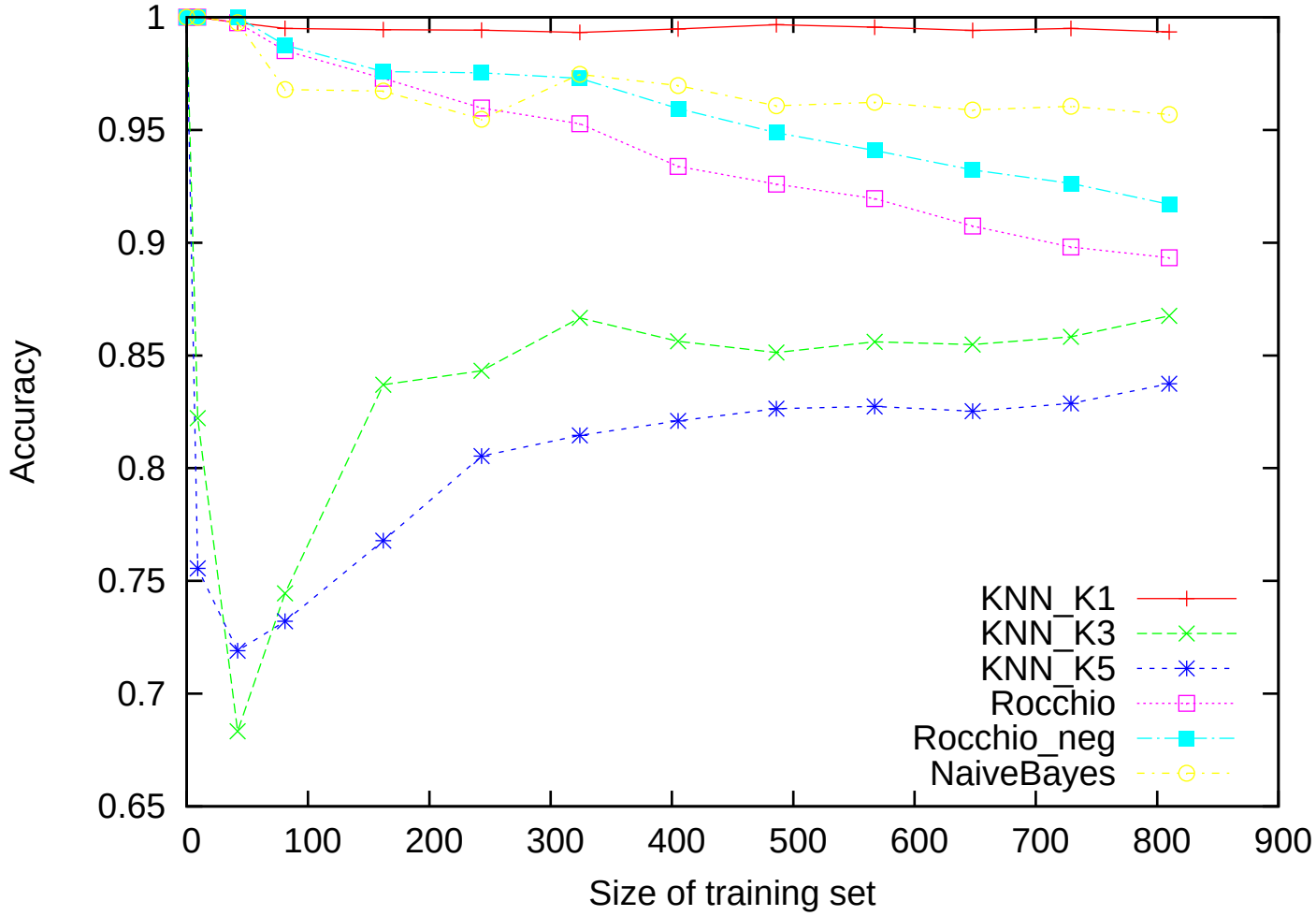
The learning curves for the training data for the Rocchio classifiers were similar to the Naive Bayes learning curve, but had slightly lower accuracy. The curves gradually got less and less accurate, which makes sense since we are sacrificing accuracy in training data for accuracy in unseen data. The KNN classifier for K=1 stays at consistently high accuracy, which might be an indicator of overfitting. For K=3 and K=5, the accuracy consistently gets better, which could be an indicator of underfitting.

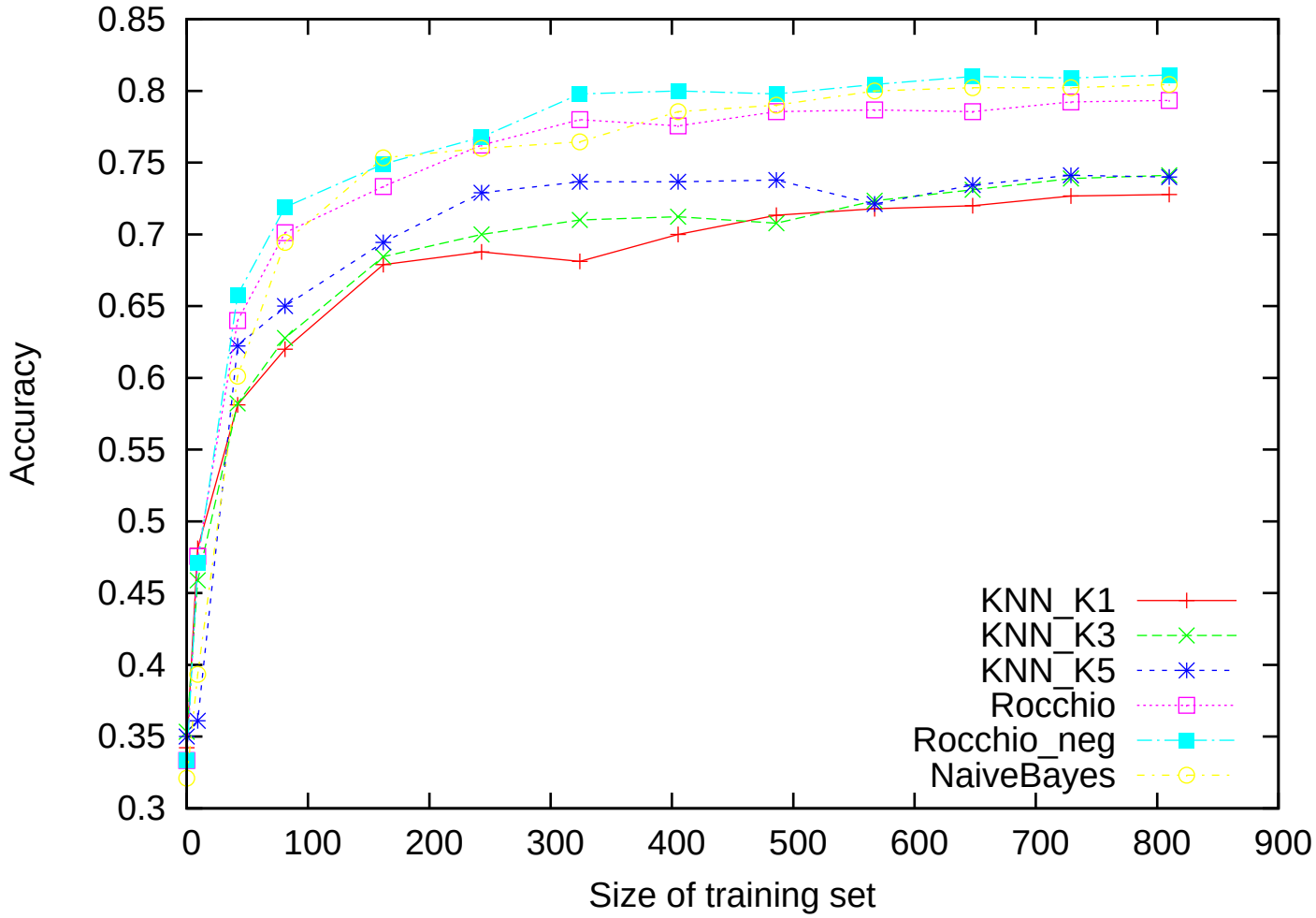
The learning curve for the training data followed a similar pattern for every classifier. The accuracy increases as the size of the training set increases, which makes sense. The KNN classifiers are less accurate than the Rocchio and Bayes classifiers, which might be due to the overfitting and underfitting. It does seem however that K=3 tests better than K=1 and K=5. The other classifiers were within the same accuracy, though the neg Rocchio had slightly higher accuracy than Naive Bayes, and Naive Bayes had slightly accuracy than normal Rocchio.

Classifier	Total training time	Testing time per example
Naive Bayes	2.446	0.04
KNN K=1	1.841	0.18
KNN K=3	1.875	0.18
KNN K=5	1.969	0.19
Rocchio	3.653	0.48
Rocchio neg	6.775	1.25

The fastest classifier to test was KNN and the slowest was Rocchio with the negative version being slower. Testing for Naive Bayes was the fastest and the slowest again was Rocchio with Rocchio neg being slowest.

I would say that in general, Naive Bayes is the best since it has the fastest testing time and among the best accuracy. It also has a faster training time than KNN.


Accuracy
Size of training set
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0
100
200
300
400
500
600
700
800
900
KNN_K1
KNN_K3
KNN_K5
Rocchio
Rocchio_neg
NaiveBayes


0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
Accuracy
0
100
200
300
400
500
600
700
800
900
Size of training set
KNN_K1
KNN_K3
KNN_K5
Rocchio
Rocchio_neg
NaiveBayes