CS 760: Homework 1
Experimental Methodology, Feature Selection, *k*-NN and Naive Bayes

Sajika Galege

Login: sgallege

2/17/2010

Part 1: Cross Validation

The CV.java creates the files for cross validation

CV.java sample.names sample.data n
sample.names = name definitions file
sample.data = Data set file
n = Number of folds

The program takes in the above arguments and creates n training files and n test files from the randomly ordered data set file named

train0.data … train9.data
test0.data … test9.data
for n=10 in the root directory.

Part 2: Implementing the k Nearest-Neighbor and Naive Bayes

k Nearest-Neighbor

The Distance function for the features are implemented as follows

Continuous features: The Euclidean distance between 2 points is calculated as

```
Distance += Math.sqrt((X-Y)*(X-Y));
```

Discrete features: A specified weight is added if the discrete values are not the same

```
if(!x[i].equals(y[i]))Distance+=w[i];
```

Hierarchical features: A specified weight is added if the values are not sidling (from same parent)

```
if(!x.parent.equals(y.parent))Distance+=w;
```

All the values are added to the Distance and if 2 data points are have same values the distance would be 0. This approach is a good measure of the similarity between 2 data samples as all of the features contribute to the Distance and the changing the weight would allow emphasize a particular feature. This also uses a very simple method to calculate the distance reducing computation time.

Naive Bayes

The p value is set to 1/#class and m=30. The continuous vales are divided in to bins of discrete set with the bin size set 100, any hierarchical values also can be treated as discrete values . The Learning algorithm runs through all training samples and calculates

```
prob(feature_i = value_j | category)
```

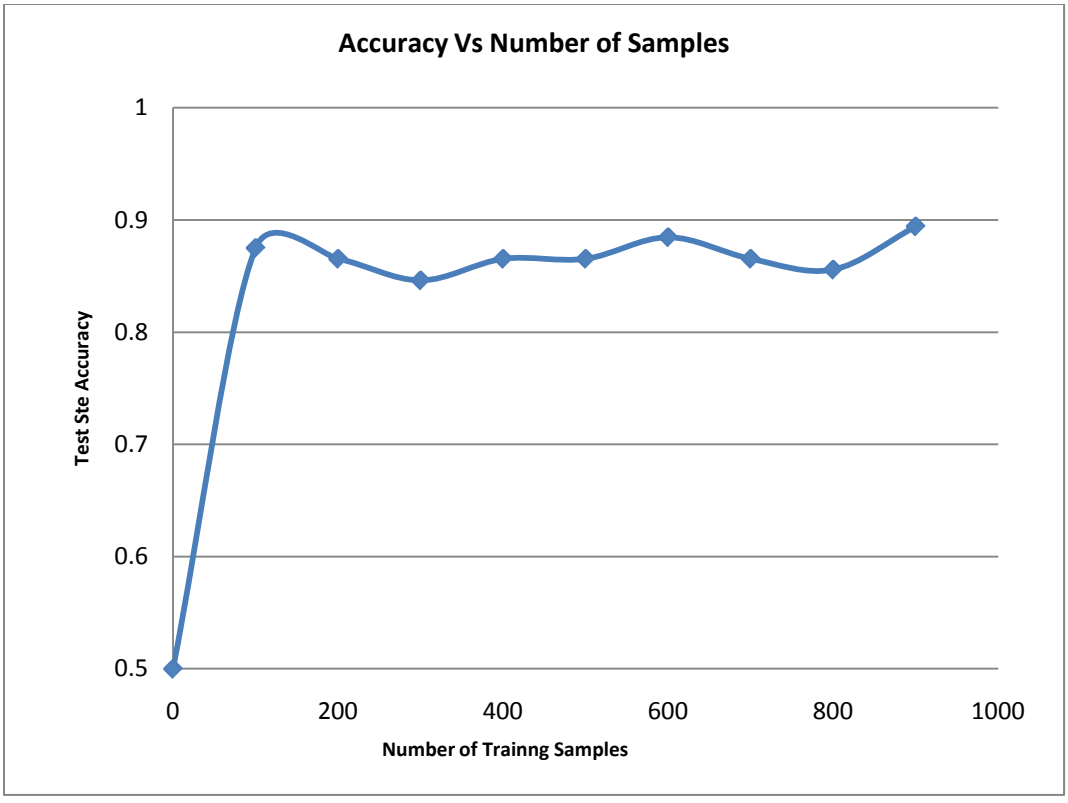
After the learning is complete new samples are classified by adding up the conditional probabilities in Log and the most likely outcome is predicted. The implementation works for more than 2 output classes

```
log_nbP[j]=Math.log(outCount[j]) - Math.log(trainSize);

log_nbP[j]+= Math.log(currentFeature[testSample[i]][j]+(m*p[i]))
             - Math.log(outCount[j]+m);
```

Part 3: Learning Curve for Naive Bayes

The Learning Curve for Naïve Bayes is created unig the train/test fold #1. This is a graph where the X-axis is the number of training examples and the Y-axis is the accuracy on the test set.

#Train Samples	Test Set Accuracy
100	0.875
200	0.8653846153846154
300	0.8461538461538461
400	0.8653846153846154
500	0.8653846153846154
600	0.8846153846153846
700	0.8653846153846154
800	0.8557692307692307
900	0.8942307692307693



Part 4: Selecting a Special *k*

The algorithm runs through *k*: 1, 3, 5, 13, 15, 21, 27, 51 with leave one out testing to select the k value that makes the best prediction. The implementation works for more than 2 output classes. If there is a tie between classes when taking the majority vote, the value that appears first on the .names file is favored.

If there is a tie between 2 k values the greater value is favored. The accuracies for the folds 1-3 are given below

```
Fold# 1
1-NN a=0.9208556149732621
3-NN a=0.9155080213903743
5-NN a=0.9229946524064171
13-NN a=0.9165775401069519
15-NN a=0.9176470588235294
21-NN a=0.8994652406417112
27-NN a=0.9048128342245989
51-NN a=0.8877005347593583
Best k =5

Fold# 2
1-NN a=0.9315508021390374
3-NN a=0.9251336898395722
5-NN a=0.9251336898395722
13-NN a=0.9133689839572192
15-NN a=0.9112299465240642
21-NN a=0.8973262032085562
27-NN a=0.8983957219251337
51-NN a=0.893048128342246
Best k =1

Fold# 3
1-NN a=0.9294117647058824
3-NN a=0.9112299465240642
5-NN a=0.9229946524064171
13-NN a=0.9101604278074866
15-NN a=0.9058823529411765
21-NN a=0.8855614973262033
27-NN a=0.9037433155080213
51-NN a=0.8855614973262033
Best k =1
```

For the given data set it seems to prefer k=1 over the higher values, which is somewhat counter intuitive. However since the accuracy is around 90% for most k's there is not a large effect on the selected k.

Part 5: Using Backward Selection to Select Good Features for k-NN

The program implements a greedy backward selection to choose a good subset of the features. The Train data is shuffled and split in to a 80% train set and a 20% tune set before the feature selection is done. After each feature is dropped the new accuracy on the 20% tuning set is calculated and if the new accuracy is worse than the original the original then the feature is added on again. But if the new accuracy is better the next feature is dropped looping through all the features.

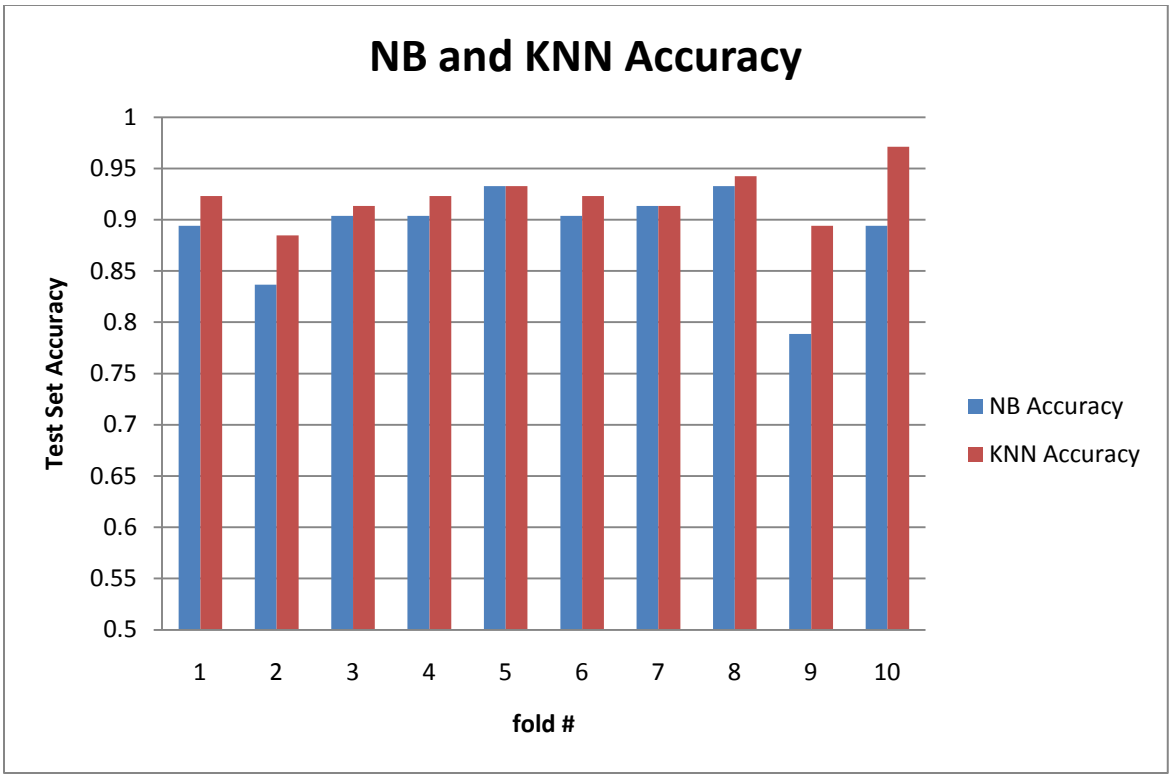
```
All Features
 Manufacture_Country
 Engine_Displacement
 Num_Cylynders
 Num_Gears
 Trans_Type
 Drive_Type
 Vehicle_Class
 Variable_Valve_Timing
 Air_Aspiration_Method

Test Accuracy 5-NN = 0.9230769230769231
Selected Features:
 Engine_Displacement
 Num_Cylynders
 Trans_Type
 Drive_Type
 Vehicle_Class
 Variable_Valve_Timing
Test Accuracy 5-NN (FS'd)= 0.9326923076923077

Test Accuracy 1-NN = 0.8846153846153846
Selected Features:
 Manufacture_Country
 Engine_Displacement
 Num_Cylynders
 Vehicle_Class
 Variable_Valve_Timing
 Air_Aspiration_Method
Test Accuracy 1-NN (FS'd)= 0.8942307692307693

Test Accuracy 1-NN = 0.9134615384615384
Selected Features:
 Manufacture_Country
 Engine_Displacement
 Num_Cylynders
 Num_Gears
 Trans_Type
 Drive_Type
 Vehicle_Class
 Variable_Valve_Timing
 Air_Aspiration_Method
Test Accuracy 1-NN (FS'd)= 0.9038461538461539
```

The selected feature subset seems to perform better against the test set, the feature selection is dependent on the random selection of the train/tune sets even on the same fold.

Part 6: Running Your Algorithms

The test-set accuracies and the mean and standard deviation for *k*-NN and Naive Bayes. Also for each fold, the chosen value of *k* and (for folds #1-3) the chosen feature subset.

Fold#	NB Accuracy	KNN Accuracy	K	Selected Features	FS KNN Accuracy
1	0.894230769	0.923076923	5	Engine_Displacement Num_Cylynders Trans_Type Drive_Type Vehicle_Class Variable_Valve_Timing	0.932692308
2	0.836538462	0.884615385	1	Manufacture_Country Engine_Displacement Num_Cylynders Vehicle_Class Variable_Valve_Timing Air_Aspiration_Method	0.894230769
3	0.903846154	0.913461538	1	Manufacture_Country Engine_Displacement Num_Cylynders Num_Gears Trans_Type Drive_Type Vehicle_Class Variable_Valve_Timing Air_Aspiration_Method	0.903846154
4	0.903846154	0.923076923	1		
5	0.932692308	0.932692308	1		
6	0.903846154	0.923076923	1		
7	0.913461538	0.913461538	1		
8	0.932692308	0.942307692	5		
9	0.788461538	0.894230769	1		
10	0.894230769	0.971153846	1		
mean	0.890384615	0.922115385			
St Dev	0.044688269	0.024176951			



Part 7: Judging the Statistical Significance

Test Fold	Error(NB)	Error(KNN)	Error(NB)- Error(KNN)
1	0.105769	0.076923	0.028846
2	0.163462	0.115385	0.048077
3	0.096154	0.086538	0.009615
4	0.096154	0.076923	0.019231
5	0.067308	0.067308	0
6	0.096154	0.076923	0.019231
7	0.086538	0.086538	0
8	0.067308	0.057692	0.009615
9	0.211538	0.105769	0.105769
10	0.105769	0.028846	0.076923
Mean			**0.031730**
St Dev			**0.035125**
$T_{95,10}$			**2.23**

$P = m +/- t_{n,k-1} S_m$

P = 0.0317308 + (2.23 * 0.035125) = 0.395639

Or P = 0.0317308 - (2.23 * 0.035125) = 0.238981

P= 2.38% lies within the first 2.5% of the t-paired test thus we can reject the null hypothesis for the NB and KNN algorithms feather more we can say that the KNN algorithm is better for the given cars data set