CSE 537 – Project 05 - Machine Learning

Group Members (41):
Pratik Zambani 111500248:
Siddhartha Chhabra: 111461745
Sweta Kumari: 111497926
Swetha Tatavarthy:111326521

Contribution:
Pratik Zambani: 25%
Worked on question 1, decision tree construction,entropy, Q2 testing
Siddhartha Chhabra: 25%
Worked on question 1 pruning and prediction, Q2 code review.
Sweta Kumari:25%
Worked on question 2. Classification of data on the basis of training models.Worked on extra credit feature ideas.
Code reviewed for question 1.
Swetha Tatavarthy:25%
Worked on question 2. Building word count for spam and ham emails.Worked on extra credit feature ideas.Testing for question 1.

1. **Accuracy for threshold values.**

Due to our threshold we have a tradeoff between accuracy and runtime so to improve accuracy we can increase threshold but it would take a lot of time due to bigger tree.

Threshold	**Accuracy**
0.01	0.72
0.05	0.72

1	0.72

2. Explain which options work well and why?

Ideal threshold should be around 0.01 to 1.
If we use a threshold below this our predictions won't be accurate due to excessively pruned tree.
If we keep our threshold above 1, size of our tree will be huge due to which our predictions would be right but it would take a lot of time to make and traverse our decision tree.

2. Spam Filter

What: In this project, we implemented Naive Bayes algorithm to classify spam emails. We also implemented Laplace Smoothing.
Input: Training data and Test data
Output: A csv file containing two columns as <ID> <Spam/Ham> delimited by space.
How:

1. We preprocessed the training data and isolated different kinds of data, viz. count of each spam and ham words, total number of spam and ham words .
2. Using the information we have about training data, we calculated the probability of the emails in test data being spam based on the given words using Naive Bayesian Classification.
3. To accommodate the scenario where there is a word in test data which never appeared in training data, we used smoothing. We used Laplace smoothing in particular.

Result: Correct Mails identified: 909
Accuracy Rate: 90.9

Observations and Optimizations:

We have tried two optimization:

1. Since the probabilities are very small, so used logarithms.
2. We tried variations of smoothing for different smoothing factors ranging from 1-100. We got the best result by using the standard Laplace smoothing i.e add-1 smoothing.

How to run:

```
python q2_classifier.py -f1 <train_dataset> -f2 <test_dataset> -o <output_file>
```

where, q2_classifier.py is the name of the python script

<train_dataset> is the name of the training data file

<test_dataset> is the name of the test data file

<output_file> is the name of the output file to be generated

Bayesian classifier for extra credit:

We attempted building the classifier with original files provided in the problem statement.

We adopted some additional features checks for example:

1.Excluded the commonly occurring genuine words like ‘Thanks’,’Dear’,’Regards’,’Respected’,’Hello’ etc from affecting the probability classification.

2.Inflated the probability of spam for emails containing words ‘!’,’$$$’ and large number of uppercase characters, words like ‘lottery’ etc.

3. Inflated the probability of spam for emails with email Id containing more than 50% upper-case characters.

Results : Accuracy achieved:58.4

With better parsing logic to read the email message body and the message labels etc, we can prevent the classifier from picking pronouns etc from training the classifier.

Running the file:

Note: Please note that you have to place the input data in the same folder containing the classifier and as follows:

Command: python q2_classifer_extra_credit.py -f1 train.index -f2 test.index -o output.csv