CS5691: Pattern Recognition and Machine Learning

Assignment 3

Course Instructor : Arun Rajkumar

Muhammed Tharikh
cs22m058

DATASET

Here I used the dataset "enron" from kaggle. It contains 6 datasets, and I used 5 of them for training and the last one for testing. The training dataset (enron1, enron2, enron3, enron4, enron5) contains a total 27k+ emails in the form of txt file- containing 15k+ ham email and 12k+ spam email. And for testing (enron6) it contains 4500 spam and 1500 hams.

PREPROCESSING

I have one universal dictionary which I will use to hold all words encountered so far. Also for each data i read, i create a dictionary to count how many times that word was encountered, all these will be done in the preprocessing step.

For preprocessing the data, first I read each of the emails and only take the words containing digits, "$" symbol (so as to get money symbol) , and letters, rest of the symbols like comma and all i skipped. Then add (all in lowercase) to the universal dictionary, also a local dictionary of that data along with how much time I encountered this. if it is a number, i add a new word "NUMBER" to represent all numbers, so whenever a number is encountered i increment the count for "NUMBER".

I used the **nltk** library for stemming (porter stemmer) the words to make it to its base form, so that the dimension of the data might not be too long. Also the word swim and swimming conveys almost the same meaning, so stemming is done to make it a base word. Also i used a **enchant** dictionary so that, when a word is encountered, i check if it is in the english dictionary, if it is, then stem it and add to dictionary with count incremented, if it is not in dictionary, i added new word "UNKNOWN" to mark as illegal word (in the hope that spam email might contain many unknown word - gibberish words) and increment its count whenever illegal word comes.

Also, the universal dictionary that I created will also contain the index of each word (in the order added to the dictionary) so that we can vectorise each local data (word will be key and index will be value). First I will take a local dictionary which contains all words in that data with its frequency in it. Then turn to a vector whose length is the length of the universal dictionary i created, and if word is present then put its count based on the value of the word in the dictionary (or put 1 as i have implemented 2 methods in which one uses binary value and other the frequency). So on doing so we get a vectorized form of data, i have not added the vectorized form of data into the zip as it is in gb's, so not possible to add that much data.

This data is given to training data. Here 2 matrices are there, one which contains vectors of spam emails and other vectors of ham emails.Also done laplace smoothing indirectly that if prob is 0, then add 1/count onto it, so that it doesn't become 0.

TRAINING

Here i have used 3 methods (i will only take 1 which gave better accuracy in test data)

- Gaussian naive bayes: i used the vectorized form and frequency count to implement gaussian naive bayes. But i encountered a problem that, as the dataset contains nearly 30k data, on encountering number of words I got approx 21k (after stemming and removing illegal words) which is very large (it increased the dimension), and on finding the inverse of the covariance matrix even though it might seem a reasonable model. It took more time so as to compute the label for 1 test data. Also, the PC got hung up by using the model, so I have not used this dataset for final (i have attached the code i did for the same in the folder).
- Normal Naive bayes: here instead of storing count, i only store the presence(i.e, whether that word is present or not). And used the normal bayes theorem for training and prediction
- Frequency counting naive bayes: here instead of just checking if a word is present or not, i count all the word occurrences, then based on the count, i give more probability (w.r.t total number of words).

TESTING

Here i used "enron6" dataset to test out the accuracy of the dataset, and computed 3 accuracy, the spam accuracy (out of total spam, how much it discovered as spam - true positive), ham accuracy (out of total ham, how much it discovered as ham -true negative). Then for the 2 cases, the accuracy are:

Normal Naive bayes :
spam accuracy :- **0.8155555555555556**
ham accuracy :- **0.806**
Overall accuracy :- **0.8131666666666667**

Frequency counting naive bayes :
spam accuracy :- **0.9817777777777777**
ham accuracy :- **0.568**
Overall accuracy :- **0.8783333333333333**

So overall, I am getting around 80% accuracy for normal naive bayes and 87% for frequency based naive bayes. But frequency based naive bayes, try to create more false positives, i.e, telling more data that it is spam. Also i tried to combine these two, but didn't go any exceptional results (i preferred simplicity over some increase in accuracy), so in the final answer, i will attach the code for simple naive bayes, as it gives accuracy of both spam and non-spam reasonably well,

also it is simple in nature as compared to the frequency method (both have similar kind of accuracy, but this is simple and the other complex).
I have attached code for each of the models used, and to run specific models from scratch-goto that folder and run the from preprocessing part by including data in corresponding dataset as the vector representation is not available (this is for trying out the code only for testing for new data just run the main file) .

But to test the final output, I created a folder named test, you just need to **add data onto the test folder**, then **run the main file**, which prints the label for each data file in the terminal in the form of a dictionary.

Run The Main File To Test Datas (By Putting Files Inside Test Folder)