Report on
Naive Bayes

Team

1. Sai Satvik Vuppala (2017B4A71449H)
2. Shreya Kotagiri (2017B3AA0296H)
3. Dhanush Karupakula(2017B3A71011H)

Introduction

We implement Naive Bayes algorithm to classify mails to be spam mails or not, using 7-fold cross validation for training and testing the model. Stop words, commas, fullstops, numbers, hyphens, brackets, exclamation marks and any other single/double letter words have been discarded, since they do not contribute to the context of the mail. Laplacian smoothing has been used to get over the case of zero probability occurrence. By doing this, we ensure every word occurs at least once in the spam and not spam classes.

Libraries Used

1. Numpy
2. Pandas
3. RE (regular expression)
4. Statistics

Implementation

The naive bayes algorithm correlates tokens (certain words) to spam mails and non-spam mails, and then calculates the probability that a mail is spam (or not spam). We remove any stopwords, punctuations, characters etc., since they do not provide any information about the mail being spam or not, so words that have a high probability to be in spam mails and that are not usually found in non-spam mails, or vice versa can be focussed on for better accuracy. All words have been converted to lowercase to make sure 'the', 'The' are treated to be the same word.

Laplacian smoothing is important in implementation of this algorithm. It ensures that a mail is not wrongly categorized, due to the occurrence of a word in non-spam mail, for example is zero. Laplacian smoothing allots a small positive value instead of zero probability for the word to occur in a non-spam mail, so as to ensure that the probabilities of other words in the mail is not nullified, thus preventing wrong classification. The idea is that there is a very less probability of such a word being in a non-spam mail but not zero probability.

1. Data Preprocessing:

   Inorder to improve the accuracy of the data we performed data preprocessing which includes the following:

   a. Using the regular expression, special characters were eliminated.
   b. All the words were converted to lowercase, as there should be no distinction between words such as MACHine and machine.
   c. Stopwords (the list of most frequent words) used were removed, as they do not contribute to the aim of the experiment.
   d. Laplacian smoothening was done in order to avoid the division by zero.
2. Then the data is divided into 7 folds to perform cross-validation and the obtained train data is fed to the main naive bayes algorithm.
3. The naive bayes algorithm is as follows

Here we have two classes namely not-spam (1) and spam (0)

We aim to find if given a mail content check if it is spam or not. So we need to find the probability of spam mail given a mail sentence, i.e, the class with the highest probability of the two, given the text.

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d)$$

Where c denotes spam or not, d is the sentence in the mail.

We can further simplify it using Bayes theorem, and as the main aim is to maximise the above equation we can ignore the denominator as it does not contribute to it.

On rewriting it we obtain the equation in the form of

$$\text{Posterior} \propto \text{Prior} * \text{Likelihood}$$

Where likelihood is the probability of having a document from the class c and prior is the probability that d belongs to a class c, given c.

It is assumed that each and every feature is independent, and on considering that our equation turns out to be

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|c)$$

This assumption enables us to substitute P(d/c) as a product of probability of individual features.

For computational purposes we maximise the logarithmic value of the function. Leading us to the final equation used:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in positions} \log P(w_i|c)$$

Where P(c) is the probability of finding a document of a class c in the dataset and $P(w_i/c)$ is the probability of word w_i occurring in a document belonging to c.

Here we also handle the case of zero probability, that is the case where there are words used that are not part of the dictionary of the training data, to handle such cases we use laplacian smoothing.

$$\hat{P}(w_i|c) = \frac{count(w_i,c)+1}{\sum_{w \in V}(count(w,c)+1)} = \frac{count(w_i,c)+1}{(\sum_{w \in V} count(w,c)) + |V|}$$

|V| = denotes the size of the vocabulary.

Accuracy

The accuracy of the naive bayes algorithm over each fold is presented below along with the average accuracy:

```
Accuracies over each fold: [82.3943661971831, 73.23943661971832,
85.91549295774648, 82.3943661971831, 79.5774647887324, 83.80281690140845,
80.98591549295774]

Average Accuracy: 81.187
```

Limitations of Naive Bayes Algorithm

1. Assuming all the features are independent of each other. This is not the case with the real world, consider two features like weight and height, they cannot be considered as independent, as BMI another feature is both dependent on height and weight making them dependent.
2. Features are given equal importance which is not a practical assumption. In real time, we may not find values for all the features everytime, so giving equal importance to each and every feature is not a viable option.
3. Zero Frequency, which means if any categorical variable is not present in the training data set then zero probability is assigned there. To tackle this limitation we use laplacian smoothing.