Report:PRML Assignment 3

Rituparna Adha (CS20M054)

Shree Kanti (CS20M062)

1 Introduction

The problem provided is to build a binary classifier to distinguish if an email is a spam or not. We have implemented a Naive Bayes classifier.

2 Dataset

http://www.aueb.gr/users/ion/data/enron-spam/preprocessed/enron1.tar.gz

3 Implementation

We have implemented a Naive Bayes classifier.

3.1 Preprocessing

- The preprocessing part includes tokenization (i.e.converting sentences into a list of words). we have used the nltk library to do that. These are done both for spam and ham emails separately.
- We have also made all the words in lower case such that two words-Subject and subject should not be considered as two different features.
- There are various words in emails like the, in, are etc which might appear many times in both spam and ham emails. They are not significant or can be said as they do not add value to the prediction.

These words can be denoted as stopwords and we have removed these stopwords from our list of words.

- The nltk library also provides a function for lemmatization. There are words like help, helped and helping, the core word here is help. Lemmatization helps in extracting the core word. This would help us combine words with same core word to be considered as a single feature.

- Lastly there are also some emails which contain numbers like phone numbers etc. Numbers are not a very valuable feature for classifying an email as spam or ham. So, We have removed them too.

3.2 Training

- We have created two different dictionaries containing the frequency of different features for both spam and ham emails. These two dictionaries are stored in a folder named model as 'spam.csv' and 'ham.csv'.

- We have also recorded the total number of words of both spam and ham category after the preprocessing.

- The probability of a word occuring in a spam mail or non-spam mail= $\frac{\text{frequency}}{\text{\# of words in spam/non-spam}}$

3.3 Prediction

- We are trying to predict $P(spam|e_1 \cap e_2 \cap \cdots \cap e_n)$ where e_i are the words in the email, E=$(e_1, e_2, ..., e_n)$.

- Now from Bayes theorem,
$P(spam|e_1 \cap e_2 \cap \cdots \cap e_n) = \frac{p(spam) \times P(e_1 \cap e_2 \cap \cdots \cap e_n|spam)}{P(e_1 \cup e_2 \cap \cdots \cap e_n)}$

- We considers all features are independent in naive bayes.Hence,

$$P(spam|e_1 \cap e_2 \cap \cdots \cap e_n) = \frac{p(spam) \times P(e_1|spam)p(e_2|spam)\ldots p(e_n|spam)}{P(e_1 \cap e_2 \cap \cdots \cap e_n)}$$

- We can rewrite the above as

$$P(spam|e_1 \cap e_2 \cap \cdots \cap e_n) \propto p(spam) \times P(e_1|spam)p(e_2|spam)\ldots p(e_n|spam)$$
$$P(ham|e_1 \cap e_2 \cap \cdots \cap e_n) \propto p(ham) \times P(e_1|ham)p(e_2|ham)\ldots p(e_n|ham)$$

- So to classify an email as spam or ham we need to find

  $P(spam|e_1 \cap e_2 \cap ... \cap e_n)$ and $P(ham|e_1 \cap e_2 \cap ... \cap e_n)$. Whichever probability is maximum, the email will go into that class.

  Here,
  $P(e_1) = \frac{\text{total number of } e_1 \text{ in dataset}}{\text{total number of words in dataset}}$

  $P(e_1|spam) = \frac{\text{total number of } e_1 \text{ in spam emails}}{\text{total number of words in spam emails}}$

- We might have certain words that might not be in the dictionary, we are avoiding them in the computation else the probability will result in 0. This issue has been solved by additive smoothing whose formulae is provided below.

$$P(e_k|spam) = \frac{n_k + 1}{n + |dictionary|}$$

  Where, n_k = number of e_k in spam messages and
  n = total words in spam messages.

- Lastly,we have computed both $P(spam|e_1 \cap e_2 \cap \cdots \cap e_n)$ and $P(ham|e_1 \cap e_2 \cap \cdots \cap e_n)$.Based on the value of them we classify the emails as spam or non-spam.

4 Conclusion

- On running train.py a directory named model will be auto created. Inside this directory, two csv files spam.csv and ham.csv will be created which will contain the frequency of each of the feature in spam and ham emails respectively.

- On running predict.py a text file will be created named as output.txt which will show the classification of each of the emails in test folder as spam or ham along with the name of the email file. Here, we have shown spam by 1 and ham(non-spam) by 0.

- Finally, for better understanding of the performance of our algorithm, we have calculated the accuracy for spam and ham.
  For our dataset, we are getting accuracy as-
  Spam Accuracy = 90% and
  Ham Accuracy = 99%.