Naive Bayes Classifier

In this experiment, from a given dataset of emails, we want to learn whether the email is spam or not.
For this, we use a Naive Bayes Classifier.

We have implemented this in Python. It could be run by:

python naive.py

Datasets used:

Training- nbctrain.txt (consisting of 5000 email instances)

Testing- nbctest.txt(consisting of 1000 email instances)

1. Training

The classifier runs on 5000 instances of emails to calculate the prior probabilities (ham and spam) and the likelihood.

Calculating prior probabilities, P(ham) and P(spam)

By spanning through every instance in training data, counting number of ham and spam mails.

	Total word frequency	Probability
HAM	2598	0.5196
SPAM	2402	0.4804

Table 1: Prior probabilities and their respective counts

Calculating likelihood, P(x|ham) and P(x|spam)

By spanning through every instance in training data, we create a dictionary of unique words and count the frequency of ham and spam for each word. The table contains the five fields: Spam frequency, Ham frequency, Total frequency, Likelihood spam, Likelihood ham. This can be seen below:

Word	Spam	Ham	Total	Prob_spam, P(x|spam)	Prob_ham P(x|ham)
limited	259	306	565	0	0
all	1447	2419	3866	0	0
jason	4	234	238	0	0

Table 2: Frequency table with initial likelihood 0

Initially, likelihood is 0 corresponding to each word. So after the creation of frequency table we observed:

Total spam words=792238

Total ham words=1158206

Getting the frequency table, we calculate the likelihood by m-estimate. Initially, m is chosen to be the distinct words in dictionary, i.e the length of dictionary. For every word xi, likelihood is calculated by:

$P(x|ham) = (n_{ham} + m*p)/(n+m)$

where n_{ham} denotes frequency of ham word x in frequency table

n total frequency of ham

m is vocabulary i.e. distinct number of words

p is chosen as 1/m

From our training set we observed:

Vocabulary = 996

So, for our m-estimate,

m=996 and

p=1/996=0.00100401606426

Similarly, likelihood can be calculated for P(x|spam).

Thus, the frequency table is updated with the likelihood.

Word	Spam	Ham	Total	Prob_spam, P(x|spam)	Prob_ham P(x|ham)
limited	259	306	565	2.0171E-05	0.05596091
All	1447	2419	3866	0.03671804	0.04066591
jason	4	234	238	0.02275117	0.03682792

Table 3: Frequency table

Out of the 996 distinct words, the top 5 words for likelihood of ham and spam are:

	Spam	Ham	Total	Prob_spam	Prob_ham
Aaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaa aaaaaaaaaaaa	15	64869	64884	0.0000201705927	0.0559609110405
Enron	29125	47139	76264	0.0367180428474	0.0406659063735
the	18046	42690	60736	0.0227511680034	0.0368279212769
to	15256	30742	45998	0.0192339208859	0.0265208307094
A	19289	21850	41139	0.0243181709306	0.0188500364906

Decreasing

Table 4: Frequency table of top 5 likelihood of ham

Word	Spam	Ham	Total	Prob_spam	Prob_ham
Enron	29125	47139	76264	0.0367180428474	0.0406659063735
A	19289	21850	41139	0.0243181709306	0.0188500364906
The	18046	42690	60736	0.0227511680034	0.0368279212769
Corp	16577	14483	31060	0.020899255453	0.0124948024589
To	15256	30742	45998	0.0192339208859	0.0265208307094

Decreasing

Table 5: Frequency table of top 5 likelihood of spam

2. Testing

Calculating posterior probabilities

By spanning through every instance in test data, for each word we use the likelihood probability of ham and spam calculated. Thus, calculating both posterior probabilities as P(spam|x) and P(ham|x) .

For an instance if

P(spam|x)>P(ham|x)

Then label =spam

P(ham|x)>P(spam|x)

Then label = ham

- As, for each word in a given instance, the likelihood probabilities are multiplied. Thus sometimes, the value becomes so less that probabilities of both ham and spam were getting equal to 0. Thus, to avoid this problem we **calculated log of the posterior probability**.
- Also, we observed that words like ‘a’, ‘the’, ‘to’, ’of’, ’for’ are more likely in spam or ham mails. Although, these words are common and do not contain any information. Thus, are not relevant to any of the category. To avoid this, we could sort the entire training data according to the frequency of each word. So **top n words of that list could be removed** and should not be used for decision of spam or ham.

- There are **4 words in test data that have not been seen in the vocabulary** of training set and are calculated using m estimate.

  5762

  msmsw06p

  msmsw05p

  msmsw04p

Running our classifier on training and test instances,

For m=996	Total instances	Correct instances	Incorrect instances	Error
Train	5000	4563	437	8.74%
Test	1000	901	99	9.9%

Table 6: Training and test error

Varying m for m-estimate

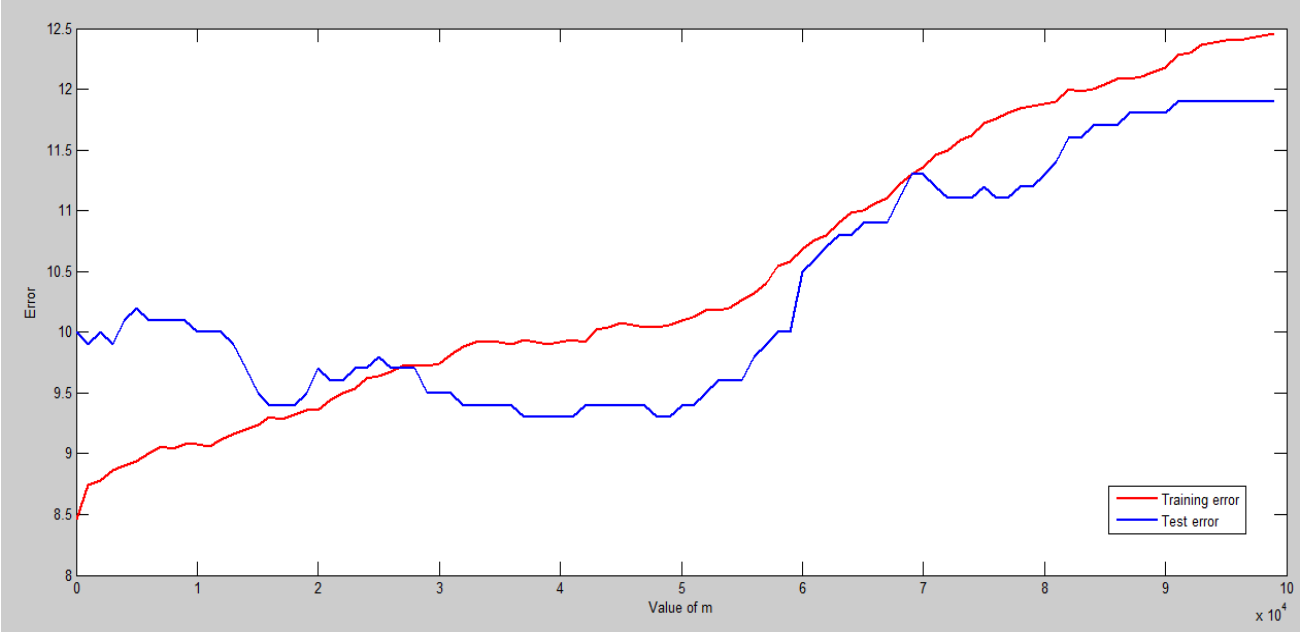

Figure 1 Variation of m with train and test error

As the value of m increases, the test error remains almost same until some point. But if value of m is further increased, the test and train error increases. This is because if value of m is very large, then the value of m dominates over the number of words for ham or spam. Thus, the values for likelihood are approximately the same, producing incorrect results.