CS 280
Programming Assignment 1

Naive Bayes Spam Filter

Jessa Faye M. Rili
jessa.rili@eee.upd.edu.ph
CS280 ThuG-A

1. The Naive Bayes Algorithm

The Naive Bayes Algorithm is a probabilistic machine learning algorithm based on Bayes' theorem. Using prior knowledge about certain observed events or conditions, Bayes' theorem predicts the probability that a hypothesis will occur. Bayes' theorem is described in Equation 1.

Equation 1

$$P(hypothesis \mid observed\ evidence) = \frac{P(observed\ evidence \mid hypothesis)P(hypothesis)}{P(observed\ evidence)}$$

$$P(hypothesis \mid observed\ evidence) = \frac{P(observed\ evidence \mid hypothesis)P(hypothesis)}{P(observed\ evidence \mid hypothesis1)P(hypothesis1) + \cdots + P(observed\ evidence \mid hypothesisN)P(hypothesisN)}$$

Where:

- **P(hypothesis | observed event):** The probability that a hypothesis event will occur given that an observed evidence has occurred. Also called the **posterior probabilities**.
- **P(observed evidence | hypothesis):** The probability of how likely an observed evidence has occurred in the past given that the hypothesis has also occurred. Also called the **likelihood probabilities**.
- **P(hypothesis):** The probability of the hypothesis occurring at all based from prior knowledge. Also called the **prior probabilities**.
- **P(observed evidence):** The total probability that the observed evidence has occurred regardless of the outcome of the event being predicted/hypothesized. Also called the **marginal probabilities** because it can be expressed as the total sum of likelihood probabilities for all possible outcomes of the event being hypothesized (e.g. hypothesis1, hypothesis2, …. , hypothesisN)

In cases where there are multiple events being observed to predict the hypothesis, it is necessary to use joint probabilities, e.g. the probability that these multiple observed events all occur at the same time. We can then modify Equation 1 to come up with an extension of Bayes' theorem for multiple observed events or features shown in Equation 2.

Equation 2

$$P(hypothesis \mid observed\ evidence1, \dots, observent\ evidenceN) = \frac{P(observed\ evidence1, \dots, observed\ evidenceN \mid hypothesis)P(hypothesis)}{P(observed\ evidence1, \dots, observed\ evidenceN \mid hypothesis1)P(hypothesis1) + \cdots P(observed\ evidence1, \dots, observed\ evidenceN \mid hypothesisN)P(hypothesisN)}$$

Recall that if one or more events are independent from one another, then their join probability (i.e. the probability that these events will occur together) would simply be a product of their individual probabilities. Applying that to Equation 2:

Equation 3

$$P(hypothesis \mid observed\ evidence1, \dots, observent\ evidenceN) = \frac{P(observed\ evidence1|hypothesis) \dots P(observed\ evidenceN|hypothesis)P(hypothesis)}{\{P(observed\ evidence1|hypothesis) \dots P(observed\ evidenceN|hypothesis1)P(hypothesis1)\} + \cdots + \{P(observed\ evidence1|hypothesisN) \dots P(observed\ evidenceN|hypothesisN)P(hypothesisN)\}}$$

Putting this in context with our spam email classification problem, we can express Equation 3 for two hypotheses (i.e. spam or ham) as follows:

Equation 4 $$P(spam|w1, \dots, wN) = \frac{P(w1|spam)\dots P(wN|spam)P(spam)}{P(w1|spam)\dots P(wN|spam)P(spam) + P(w1|ham)\dots P(wN|ham)P(ham)}$$

Equation 5 $$P(ham|w1, \dots, wN) = \frac{P(w1|ham)\dots P(wN|ham)P(ham)}{P(w1|spam)\dots P(wN|spam)P(spam) + P(w1|ham)\dots P(wN|ham)P(ham)}$$

Where w1, ... , wN are the individual words of an email document.

Knowing Equation 4 and Equation 5, the Naive Bayes Algorithm is simply all about computing the posterior probabilities for all possible hypotheses and then choosing the hypothesis that yields the highest posterior probability.

It is worth noting however why the algorithm is called 'naive' is because of a naive assumption made especially for Equation 3 – that is that each of the observed evidences are independent from one another. This, of course, is not always true in all cases. Say for instance, in spam email classification, if the word 'France' occurs in the input document, there could be a fairly high chance that the word 'French' would also occur. Nonetheless, that is the assumption for this assignment – that each word appearing in an email document is independent from each other.

2. Implementation

The algorithm described in Section 1 is implemented in Python 3.6 in the file `PA1_RILI.ipynb`.

Looking at Equation 4 and Equation 5, all that is needed to make a prediction are the likelihood and prior probabilities. But before obtaining that, we need first some way to represent text data from the email documents as numerical data that the algorithm can understand. Apart from

that, there are also improvements to the algorithm implemented to raise the precision and recall scores.

More thorough implementation details are narrated in the jupyter notebook file.

a. Computing the prior probabilities

The prior probabilities $P(ham)$ and $P(spam)$ are obtained as follows, after going through all the given training data labels.

Equation 6 $$P(ham) = \frac{\#\ of\ training\ documents\ labeled\ as\ "ham"}{total\ \#\ of\ training\ documents}$$

Equation 7 $$P(spam) = \frac{\#\ of\ training\ documents\ labeled\ as\ "spam"}{total\ \#\ of\ training\ documents}$$

b. Constructing the vocabulary

It is necessary to construct a vocabulary i.e. a set of words that is assumed by the classifier as the set of all possible words that can be encountered inside and outside of training.

- Access each training document line per line
- Get the words from each line using the regex expression:
  (?<=\s)[a-zA-Z]+[\-\']?[\.|\,|\s]
  - When a white space is encountered, do a *lookahead* to the subsequent characters
  - The subsequent characters after the white space should be alphabetic and could contain dashes (-) and apostrophe's (')
  - Just beyond the semi-alphabetic sequence of characters, there should either be a period (.), a comma (,), or another white space.
  - Returns the semi-alphabetic sequence (i.e. excluding the leading white space)
- Exclude search hits that have less than a set maximum number of letters.
  This is more for reasons of memory limitations. If done without setting a maximum word length, the memory usage of the program bloats up to the machine's limit (8GB). If that happens, the notebook runs slower then eventually freezes.

Search hits (with the trailing periods(.), comas(,) and apostrophe's(') stripped) are stored in a vocabulary vector.

c. Representing the documents

The vocabulary vector obtained previously would then be used to create a representation of input email documents in the form of a word-presence vocabulary vector, wherein each row corresponds with a word in the vocabulary vector. The elements of this vector representation would either be a '1' or a '0' which would indicate whether a certain word is present or not in the document.

d. Getting the likelihood matrices

Now that the means to represent with numbers in the form of vocabulary vectors are available for our text documents, the likelihood probabilities can now be computed. This is done by computing the individual likelihood probabilities like so:

Equation 8 $$P(w1|ham) = \frac{\#\ times\ w1\ appeared\ in\ ham\ documents}{total\ number\ of\ ham\ documents}$$

Equation 9 $$P(w1|spam) = \frac{\#\ times\ w1\ appeared\ in\ spam\ documents}{total\ number\ of\ spam\ documents}$$

Firstly, it is needed to count the occurrence of each word in all documents of a certain label. This is done by getting the vocabulary vector representation of all the training documents per label as column vectors, and then concatenating them with each other to create a likelihood matrix with shape $|v| \times D$ where $|v|$ is the size of the vocabulary and D is the number of documents belonging to a class i.e. ham or spam. Two such matrices are produced, one for each class. Afterwards, the word counts are obtained by summing along the rows of the matrix. The resulting vectors for each label after summing is then divided by the total number of documents of the specific label.

e. Computing the posterior probabilities and making predictions

The posterior probabilities are easily computed using Equation 4 and Equation 5 as long as the prior and likelihood probabilities are obtained. To avoid underflow, however, it is good to note here that instead of performing multiplication, the implementation performed a sum of logarithms of the likelihood probabilities.

Making predictions is done by computing the posterior probabilities for all labels (i.e. spam and ham) and choosing the class with the higher result.

f. Evaluating the performance

Performance is measured using the Precision and Recall scores which are computed as follows:

Equation 10 $$Precision = \frac{TP}{TP+FP}$$

Equation 11 $$Recall = \frac{TP}{TP+FN}$$

Where:

- TP (true positive): number of spam messages CLASSIFIED as spam
- TN (true negative): number of ham message CLASSIFIED as ham
- FP (false positive): number of ham messages MISCLASSIFIED as spam
- FN (false negative): number of spam messages misclassified as ham

Obtaining each of the four values above is obtained by a simple index-by-index comparison of the true labels and the predicted labels.

g. Applying Lambda Smoothing

If there was a word which had a likelihood probability of zero, the posterior probabilities described by Equation 4 and Equation 5 will always result to zero. To remedy that, lambda smoothing is performed by adding a few extra terms to Equation 8 and Equation 9 as follows:

Equation 12 $$P(w1|ham) = \frac{\#\ times\ w1\ appeared\ in\ ham\ documents + \lambda}{total\ number\ of\ ham\ documents + \lambda|V|}$$

Equation 13 $$P(w1|spam) = \frac{\#\ times\ w1\ appeared\ in\ spam\ documents + \lambda}{total\ number\ of\ spam\ documents + |V|}$$

h. Improved vocabulary selection

According to the reference paper by Hovold [1], the frequency of occurrence of the words in the training data should be considered when building the vocabulary for the filter. More specifically:

- Exclude the words with less than 3 occurences in the whole training data set
- Exclude 100 to 200 most frequently occuring words in the whole training data set

This is followed by the implementation by first re-parsing the training data but this time taking note of the number of times each word has appeared for each of the training documents. The resulting initial vocabulary is then sorted by descending order of word frequency. As per suggestion, the words with less than 3 occurrences were not included and the most frequently occurring words were also removed. Different values for the number of most frequently occurring words were tried, and the one that yielded the best score was chosen. Additionally, as per instruction, only 200 words were included from this improved vocabulary. More details are found in Section 3.

3. Analysis and discussion of results

a. The vocabulary

Using the method described previously, a vocabulary vector made of

$$|v| = 46{,}388$$

words was obtained. It was attempted to have no word length maximum limit imposed, but considering that two $|v| \times D$ matrices are to be created internally by the classifier, the processing was simply too much for the machine used if no word length limitation is imposed.

b. Vocabulary statistics

The prior probabilities obtained from the training data set are as follows:

$$P(ham) = 0.3539108192488$$
$$P(spam) = 0.6468075117370892$$

There is noticeably a big gap between documents labelled as 'spam' as compared to the documents labelled as 'ham'. The observed fact that there are more 'spam' emails than 'ham' emails makes sense considering that spam emails are faster to "produce" due to automation than the human-generated ham emails.

c. Results using the 46,388-word vocabulary without smoothing:

$$Precision = 0.7236634333408527$$
$$Recall = 0.8643017512348451$$

Expectedly, the performance was pretty bad without smoothing. This is due to the cases where there are words that have zero likelihood probabilities, e.g. the document contains words that were not encountered in any spam document during training but is part of the vocabulary because it was encountered in a ham document.
For this part, those cases are ignored, effectively having a likelihood probability of 1 – which is obviously incorrect.

d. Results using the 46,388-word vocabulary for different values of λ:

Results are shown in Table 1. The highest precision score was obtained using a value of λ=0.5, while the highest recall score was obtained using a value of λ=0.005.

Table 1 Results using the 46,388-word vocabulary for different values of λ

λ	Precision	Recall
2.0	0.98279422	0.89770992
1.0	0.98731212	0.90848675
0.5	0.99007664	0.90498428
0.1	0.98752009	0.93803323
0.005	0.97801280	0.94674450

e. Results using the 200 word vocabulary of most frequent words:

It was instructed to use the "best value" of λ from the previous section, but it is rather difficult to tell which was the best. Instead, multiple values are tried and tabulated.
Another value that was varied is the number of most frequent words removed in the vocabulary.
Results are shown in Table 2.

The best precision score is yielded when λ=2.0 and removing the 150 most frequent words from the vocabulary. Meanwhile, the best recall score is yielded when λ=0.005 and only removing the 100 most frequent words.

Table 2 Results using the 200-word vocabulary of most frequent words for different values of λ

λ	Number of most frequent words removed	Precision	Recall
2.0	200	0.89955687	0.92977099
1.0	200	0.89848419	0.93156713
0.5	200	0.89772432	0.93174674
0.1	200	0.89729403	0.93210597
0.005	200	0.89593580	0.93246520
2.0	150	0.90456644	0.94997755
1.0	150	0.90287371	0.95087562
0.5	150	0.90201754	0.95159407
0.1	150	0.90060318	0.95204311
0.005	150	0.89988122	0.95249214
2.0	100	0.89916038	0.95213291
1.0	100	0.89729501	0.95330040
0.5	100	0.89613567	0.95383925
0.1	100	0.89556136	0.95491693
0.005	100	0.89458186	0.95491693

f. Using a 1000-word vocabulary of most frequent words

Out of curiosity, the classifier was attempted to run while including more words for the vocabulary. For this part, a 1000-word vocabulary was used coming from the improved vocabulary described in Section 2.h. Results are detailed in Table 3.

It appears that the precision and recall scores have improved overall as compared to the results in Table 2.

Table 3 Results using a 1000-word vocabulary of most frequent words for different values of λ

λ	Number of most frequent words removed	Precision	Recall
2.0	200	0.96355631	0.92366412
1.0	200	0.95908588	0.93471037
0.5	200	0.95738844	0.94027840
0.1	200	0.95591584	0.94252357
0.005	200	0.95577953	0.94530759
2.0	150	0.96734961	0.92860350
1.0	150	0.96251846	0.93632690
0.5	150	0.96055046	0.94027840
0.1	150	0.95833333	0.94396048
0.005	150	0.95838645	0.94521778
2.0	100	0.96609073	0.93902110
1.0	100	0.96142974	0.94692411
0.5	100	0.95852201	0.95051639
0.1	100	0.95629843	0.95312079
0.005	100	0.95597089	0.95545577

4. Conclusions

A Naive Bayes Classifier for classifiying spam email documents was implemented. Bad performance was noted as-is, but the introduction of smoothing improved the classification scores significantly. The effect of varying the λ value for smoothing proved to be different for the precision and recall scores. In general, it seems that a higher λ yields better precision scores at the expense of worse recall scores. Meanwhile, it seems that a lower λ yields better recall scores at the expense of the classifier performance suffering in terms of the precision metric. It has also been noted that choosing the vocabulary contents (i.e. the "features") properly by choosing the words based on frequency of occurrence might have improved the classifier. But in the results outlined above, it is hard to tell since only 200 words were instructed to be included in the vocabulary which resulted to worse precision and recall scores. The performance were also measured using the improved vocabulary but including more words for the vocabulary, to which an overall improvement in classification performance was observed.

For future work, it is recommended to further increase the number of vocabulary words to be included from the improved vocabulary based on word occurrence.

5. References

[1] J. Hovold, "Naive Bayes Spam Filtering using Word Position Attributes," in *Conference on Email and Anti-Spam*, Stanford University, 2005.