Machine Learning

Lab 2: Naive Bayes Spam Detector 27 September, 2012

By:
Paris Mavromoustakos
Georgios Methenitis
Marios Tzakris

Introduction

The purpose of this lab assignment is to create an e-mail spam filter by implementing a naive Bayes classifier. In order to create the classifier, we used code we were provided with, which helped us process the contents of e-mails and extract the number of appearances of a specific word in one or more e-mails. E-mails were separated in two classes, **Spam**, which might contain advertisements or scam messages, and **Ham**, which are important e-mails that users should receive. This means that a good spam detector must not discard any ham mails.

Part 1

First, we had to process all the e-mails' text bodies, and count the occurrence of each single word in the spam and ham folders' e-mails.

After the application of the functions mentioned above on both training sets (spam/train and ham/train), the result we got was two two-dimensional cell arrays each containing all the different words found in the spam and ham folders' e-mails and their occurrences in ascending order.

After we had made a complete list of words and their occurrences, we picked some words like "ABOUT",HOWEVER,ISSUES,THANKS,BOB,BETWEEN,NEED,HTTP,SG,'MR,MEDS,PILLS, which were representative of both classes, and used them to implement the first version of our classifier.

Additionally, we created an automatic selection of these words. For each word, we computed the difference of the probability between its appearances in the ham and the spam mails respectively. After that, we normalized this probability in respect to the summation of these two probabilities as follows. For each word, w_i:

$$ProbDiff = \frac{|P(w_i|spam) - P(w_i|ham)|}{P(w_i|spam) + P(w_i|ham)}$$

Furthermore, we added an additional feature in each word, which was the ratio between the number of the appearances in two mail sets and the total number of mails in our training set. So, we considered as important and highly representative words, those which had an important difference in their probabilities and the number of their appearances in the mail set was not negligible. This function (`word_selection()`), takes as input arguments the list of the words and the number of their appearances in the mail list, as well as, the number of the words we want to return as a result. With this technique, we found a good way to choose informative words for our classifier.

Part 2

In this part of the lab exercise, we built a naive bayes classifier using the features we chose in the previous part. First, we split our data (training mails) into two parts. The first part which was the

90% of all the mails (ham and spam) had been used as train data, and the second part which contained the remaining mails had been used as the test data. First, we check in how many mails our selected words are presented. To train our classifier, we computed the probability of a mail being spam or ham, given the number of appearances in the above computation, normalizing them with a laplace smoothing to avoid zeros in probabilities. For each word w_i in our selection we did:

$$P(spam|w_i) = \frac{no.\ of\ appearances + k}{Total\ no.\ of\ mails + k \times d}$$

,where $k = 1$, and $d = 2$, the number of classes.

Then we took the second part of the training data, the test set, and we computed the likelihood of each mail being spam or ham. To accomplish that, we should check each mail if it contains the words that have been selected in the training part of the classifier. The function `presentredir()` return an array which contains ones for words that exist in this mail and zeros otherwise. Then, we computed the product of each line replacing ones with the probability that the corresponding word belongs to spam or ham and zeros with ones in order not to get zero probability. Finally, the posterior probability is computed for each mail taking into consideration the prior probability of a mail being spam or ham. We have set the prior probabilities as 0.8 for spam and 0.2 for ham mails instead of learning them from the data, as in reality there is much more spam than legitimate email on the internet.

Part 3

In the final part of the exercise we used the data in the test directory to to evaluate the classification rate with the features we have chosen and the parameters the classifier has learnt from the training data. We have plotted the roc curve for different features and parameters, found the area under curve (AUC) to compare the most efficient combination and found the most efficient threshold which afterwards used in the final test data.

TEST RESULTS:

First, the roc curve for the manually selected words does not have pretty good results. This happened because those words which we selected infer spam messages for sure, but there are spam messages which have other characteristics, difficult to derive manually. Figure **??**, presents the roc curve for 20 manually selected words by us.

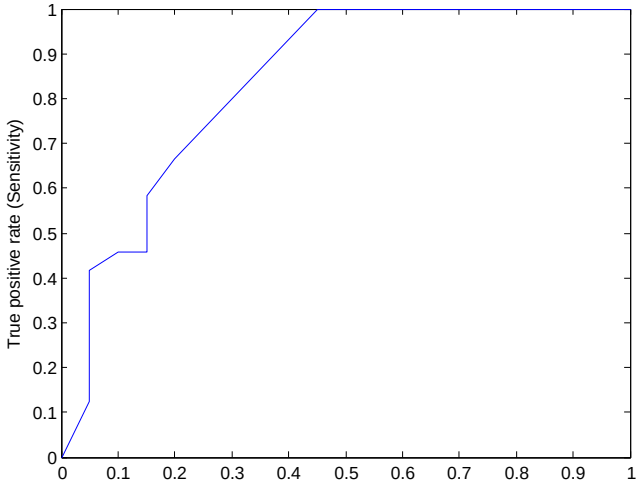

Figure 1: Roc curve, with 20 manually selected words, performed on the test set.

For the automatically selected words things are better. We performed a number of tests with different number of words to test our classifier. We tested for $5, 10, 15, ..., 200$ words. Overall, the results were far better making use of the word selection function. The best result derived for 50 selected words. The roc curve for the 50 words spam detector is depicted in Figure **??**. The area under the curve is, $AUC = 0.9458$ which determines the quality of our classifier.

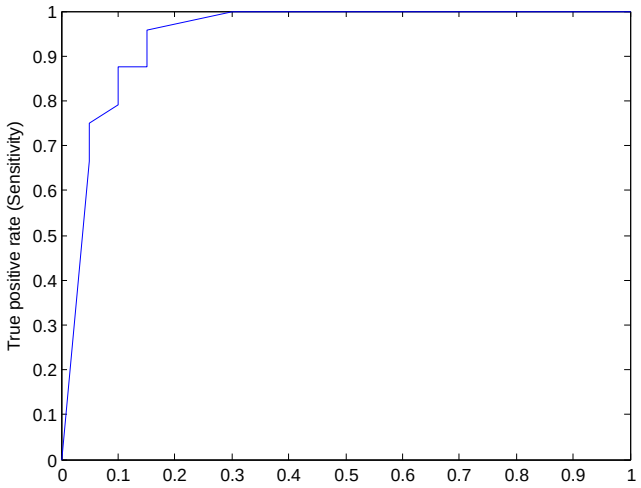

Figure 2: Roc curve, with 50 automatically selected words, performed on the test set.

The above figure indicates that the threshold we should use, by varying the decision threshold between zero and one is 0.76. And this is happening because the closest point of the curve to the point(0,1) is for threshold 0.76. With this threshold the trial in the final test data gave us:

- For the Ham test data: **134** correct and **10** misclassified mails.
- For the Spam test data: **50** correct and **4** misclassified mails.

This may be acceptable in an another algorithm, but not in a spam detector. As we said in the introduction it is crucial for people who use mail to communicate with others, not to lose their incoming messages due to spam detector. So, in a real world scenario what it matters the most is not to classify ham mails as spam. We think, that we could "sacrifice" a low error rate in spam test data, in order to get a better accuracy in ham test data. For that reason we have found the exact point when the True positive rate becomes one in our roc curve and the threshold there, is 0.80. After testing that threshold in the test set we get the following results:

- For the Ham test data: **144** correct and **0** misclassified mails.
- For the Spam test data: **42** correct and **12** misclassified mails.

To evaluate the expected true error of our classifier we could use **cross validation**. With this method we could train our data set better as the great advantage of cross-validation is that all the cases in the available set are used for testing, and almost all the cases are also used for training the classifier.

Conclusion

In this lab exercise we implemented a spam detector. This detector uses a naive bayes approach to evaluate the probability that a mail belongs to spam or to ham class. Through the possibility of each contained word this probability is easy to be computed. Our final result is a little annoying (some but not many spam messages are classified as ham) but reliable spam detector. With the term reliable we mean that the probability that a ham mail classified as spam is close to zero.

Source Code

Source code is in the same directory with this report. The main m- file is the `spam_classification.m`.