Homework 4: Naïve Bayes Classification

Joe Arriaga
CS 445: Machine Learning

March 19, 2019

1 Description

A Naïve Bayes Classifier using the Gaussian probability distribution was implemented in the Python programming language with the intent of identifying spam emails from the University of California at Irvine Spambase data set [1] as "spam" or "not spam". Training data was split into "spam" and "not spam" examples; 39.4% of the training data was spam, 60.6% of the data was not spam. For each class the mean and standard deviation for each feature was calculated. These values were then used to classify a new example from the test set by calculating the conditional probability of the data of the example in a Gaussian distribution given each class; all features were assumed to be independent. The example was classified as whichever class yielded the higher probability.

2 Results

The classifier achieved an accuracy of 65%, a precision of 53%, and a recall of 91%; the results from the test set are given in Figure 1. The classifier appears to be too sensitive to spam since it correctly classified almost all of the spam messages but misclassified just over half of the legitimate messages. This error is the reason for the low precision score and low overall accuracy. A hypothesis for these errors is that a legitimate email does not have enough strong indicators to rise above the background noise of the 57 features. If the features were not useful for classification at all one would expect a 60 - 40 split, following the a priori distribution, but the examples were separated almost 50 - 50 with slightly more classified as "spam", contradicting the a priori distribution. This suggests that the features measured are biased to classify examples as spam. More focused study of the features would be necessary to determine the causes of such bias.

The Support Vector Machine (SVM) from Homework 3 performed much better with an accuracy of 90%, a precision of 86%, and a recall of 89%. The mechanism of the SVM seems much more suited to this task since it relies on finding the key features that are most predictive rather than using the input of every feature, whether or not it has shown to be predictive.

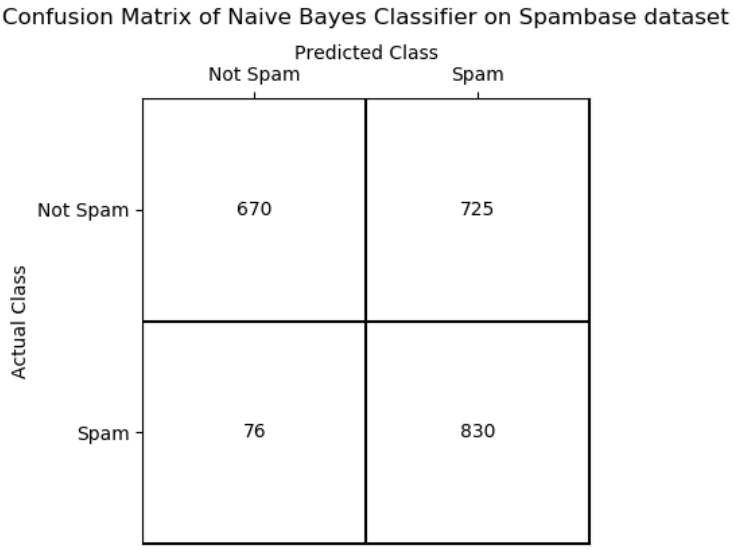

Figure 1: Confusion matrix of results from Naïve Bayes Classifier

The attributes are almost certainly not independent. Intuition says that some words would tend to be seen together frequently. The SVM supports this notion as some of the support vectors had similar weights, which was likely due to both words appearing in many of the legitimate emails. The Naïve Bayes Classifier performed poorly but the independence assumption does not appear to be a primary cause of the error. A model that accounted for the correlation among certain words could have improved performance but would also substantially increase the complexity of the model. A simpler solution may be to be more judicious about the weight given to each feature rather than taking the predictions of all features to be equal.

One other reason the Naïve Bayes Classifier might have done poorly is that it is too easy for a legitimate email to contain a few spam-like words, such as `free` or `credit`, and then be scored as highly likely to be spam. Whereas it is unlikely for a spam email to contain the words associated with legitimate emails such as `george` or `hp`. This means that the features are biased towards predicting spam as an email of random words from a dictionary is more likely to be classified as spam.

3 Conclusion

The Naïve Bayes Classifier did quite poorly because it seemed biased towards predicting "spam". The two main causes were likely that the features selected did not adequately differentiate between spam and legitimate emails, and that giving all features equal weight in the prediction created a such high level of noise in the predictions that any truly predictive features were drowned out. One solution may be to analyze the spam and legitimate emails for the frequency of each word over all messages in each class. One would then be able to see which words were highly correlated with one class and not the other, these words would likely be highly predictive. Another way to improve performance would be to analyze the results of the probability densities to determine which features were most used to predict examples, determine if they were predicting correctly or incorrectly, and weight the contribution of each feature to the final prediction accordingly; this could possibly be done using a neural network.

References

[1] Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt. Spambase data set. https://archive.ics.uci.edu/ml/datasets/Spambase.