CS570: Artificial Intelligence and Machine Learning

Programming Assignment 1

Due Date: Thursday 23:59, April 3rd

T.A: Bongseok Goh (bsgoh@ai.kaist.ac.kr)

Summary

In this assignment you will implement an algorithm for estimating the probabilities used by a naive Bayes classifier and will apply this to a data set which came from a set of real email. The assignment involves developing two functions in MATLAB and performing some experiments using those functions.

1. Write a function called ***nbayes_learn.m*** that take in training data and return the probability tables for a naïve Bayes classifier
2. Write a function called ***nbayes_predict.m*** that takes a set of test data vectors and return a set of class label predictions {spam, non-spam} for each vector.

Template for both of these function is given. Please remain comment on your code. **You should upload your MATLAB code for these two function along with summary document of your experiments (studentid_name.pdf) within zip file to KLMS.**

Ex) 20142014_BongseokGoh.zip

(nbayes_learn.m, nbayes_predict.m and 201421014_BongseokGoh.pdf are in this zip file)

Delayed submission won't be allowed. Please submit before the due time.

(Sending to T.A. won't be also allowed.)

Spam Email Datasets

For your assignment you will be working with a data set that was created a few years at Hewlett Packard Research Labs as a testbed data set to test different spam email classification algorithms. The HP researchers gathered a set of both spam and non-spam emails and computed a set of "feature values" (e.g., based on the presence of certain words) for each email. The idea is for an algorithm to try to predict whether an email is spam or not based on the features.

On the class web page you find pointers to three files.

1. Features: the first file, called ***spam_features.txt*** is an ascii file containing n = 4601 rows and d = 57 columns. Each row corresponds to a particular email and each column indicates the presence or absence of that feature in the email. The jth column thus gives the outcomes of a binary variable $X_j \in \{1,2\}$ for the set of emails.
2. Labels: the second data file, ***spam_labels.txt*** consists of n = 4601 values which are the class labels for each of the emails. The label 2 corresponds to emails that were identified as "spam" and the label 1 corresponds to emails that were identified as "non-spam"

If you are interested in finding out more about where the data came from, you can read more here http://mlearn.ics.uci.edu/databases/spambase/spambase.DOCUMENTATION which describes in detail how the data was collected, what the features mean, basic statistics about the data, and so on. Note that the original features are real-valued (many of them are expressed as the percentage of words in an email that correspond to a particular word). So to create discrete-valued variables that we can use in a naive Bayes classifier, we converted each of the original real-valued features to a binary random variable by making all values less than or equal to the median (across all 4601 rows for that feature) be 1 and all values above be 2.

Testing the classifier performance

We would like to see how accurate our classifier is at making good predictions. To calculate the accuracy, we can run the classifier on a set of examples where we know the true labels, and then calculate the percentage of the test examples for which the classifier made a correct prediction. Perform the following set of experiments and include the requested items in your report:

1. Train the classifier using only the first 1500 training examples and labels and then use the remaining rows (1501~4601) in order to compute the accuracy. To calculate the accuracy,

run *nbayes_predict.m* on the remaining rows and compare the predicted class labels it returns to the true class labels, reporting accuracy as a percentage of correct predictions.

2. Repeat this experiment but now train using only the first 50 examples and test on rows (1501~4601). Again report accuracy.

3. Repeat again, but now train using only the first 10 examples and test on rows 1501-4601. How does the accuracy vary across these 3 different training conditions?

4. Suppose a classifier randomly guessed the class labels with equal probabilities of 0.5. How accurate would you expect it to be? How accurate would a classifier be that always predicted the class label which occurred most commonly in the training set (i.e. if the majority of examples 1-1500 were class i, it would always report class i regardless of input)? How do these two simple strategies compare to the performance of your system measured in parts 1-3?

5. How do you improve the results of parts 2-3? Write the method and apply it. Report accuracy.

Source: www.ics.uci.edu/~fowlkes/class/cs177