**Lab Exercise 4**

**Introduction to data classification - Naïve Bayes classifier**

**Naïve Bayes classifier**

We will use Weka to train a Naïve Bayes classifier for the purposes of spam detection. .

The data set we will consider is the [Spambase](http://archive.ics.uci.edu/ml/datasets/Spambase) set, consisting of tagged emails from a single email account. Read through the description available for this data to get a feel for what you're dealing with. The data has been converted to ARFF format for you:

* [spambase.arff](http://www.inf.ed.ac.uk/teaching/courses/iaml/lab/data/spambase.arff)

Some simple preprocessing of the data will be required before it is ready for use. We can do this in Weka:

* From the *Preprocess* (default) tab in Weka, hit *Open file...* and select the spambase.arff file that you downloaded above.
* A full list of the attributes in this data set will appear in the "Attributes" frame.
* Delete the capital\_run\_length\_average, capital\_run\_length\_longest and capital\_run\_length\_total attributes by checking the box to their left and hitting the *Remove* button.
* The remaining attributes represent relative frequencies of various important words and characters in emails. We wish to convert these to Boolean values instead: 1 if the word or character is present in the email, 0 if not. To do this, select the *Choose* button in the *Filter* frame at the top of the window, and pick *filters > unsupervised > attribute > NumericToBinary*. Now hit the *Apply* button. All the numeric frequency attributes are now converted to Booleans. Each e-mail is now represented by a 55 dimensional vector representing whether or not a particular word exists in an e-mail. This is the so called [*bag of words*](http://en.wikipedia.org/wiki/Bag_of_words_model) representation (this is clearly a very crude assumption since it does not take into account the order of the words).
* Save this preprocessed data set for future use using the *Save...* button. You will need this for lab 2.

Given the data set we've just loaded, we wish to train a Naïve Bayes classifier to distinguish spam from regular email by fitting a distribution of the number of occurrences of each word for all the spam and non-spam e-mails. Under the *Classify* tab:

* Select *Choose* in the *Classifier* frame at the top and select *classifiers > bayes > NaiveBayes*.
* Leave the default settings and hit *Start* to build the classifier. Study the output produced, most importantly the percentages of correctly and incorrectly classified instances. You probably will notice that your classifer does rather well despite making a very strong assumption on the form of the data.
* Can you come up with a reason for the good performance? What would be the main practical problems we would face if we were not to make this assumption for this particular dataset?
* How long did your classifer take to train and classify? Given this, how scalable do you think the Naïve Bayes classifier is to large datasets? Can you come up with a good reason for this?
* Examine the classifier models produced by Weka (printed above the performance summary). Find the prior probabilities for each class.
* How does Naïve Bayes compute the probability of an e-mail belonging to a class (spam/not spam)?
* Compute the conditional probability of observing the word "3d" given that an e-mail is spam P(3d|spam) and that it is non-spam P(3d|non-spam). To do this, we need to use the counts of the built model that are produced within the *Classifier output* screen under the *Classify* tab. The general format of the Weka count output is (*Note:* this is a toy example. You will need to examine your Weka output to find the true counts for the word "3d".):

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **Class** | |
|  | | ***0*** | ***1*** |
|  | ***0*** | 1 | 2 |
| ***1*** | 3 | 4 |
| **total** | | 4 | 6 |

* This means that 4 instances (e.g. e-mails) contain that particular attribute value (e.g. the word "3d") in Class 1 (e.g. Is Spam). 2 instances didn't contain that value of the attribute in Class 1. 3 instances of Class 0 contained that attribute value, whilst 1 instance of Class 0 (e.g. Not Spam) didn't contain that attribute value. The totals reflect the number of instances belonging to both classes e.g. the number of e-mails that are Spam and not Spam.
* For the final part of this section we will now pretend we are spammers wishing to fool a spam checking system based on Naïve Bayes into classifying a spam e-mail as ham (i.e. a valid e-mail). We will now use all of the training data to train our classifier and apply the learnt classifer to a dedicated test set. Load the test set in Weka. Under the *Classify tab*, select *supplied test set > set > open file* and set the test file to the supplied [spambase\_test.arff](http://www.inf.ed.ac.uk/teaching/courses/iaml/lab/data/spambase_test.arff). This ARFF file contains the binary vector representing one spam e-mail. Run the Naïve Bayes classifer on this test set. Does the classifer classify the spam e-mail correctly?
* Open the test file [spambase\_ HYPERLINK "http://www.inf.ed.ac.uk/teaching/courses/iaml/lab/data/spambase\_test.arff"test.arff](http://www.inf.ed.ac.uk/teaching/courses/iaml/lab/data/spambase_test.arff) in emacs or another text editor. Identify good non-spam words and add these to the e-mail. *Important:* Leave the class label (last attribute value) in the test data file untouched. During testing, Weka will ignore this attribute and will instead use our previously trained classifier to predict the class label of this e-mail. Re-run the classifer on the modified test set. Has the class label (spam/non-spam) for this e-mail changed?

You've now managed to switch the predicted class label for that e-mail. Adding more “hammy” words to this e-mail has sufficiently increased the probability that this e-mail is ham so the classifier now outputs "ham" as the e-mail's class label (by changing the word content of the e-mail you have added extra evidence or "votes" towards this e-mail being classified as ham). This is the "stuffing" example given in the lectures and is directly caused by the independence assumption that is made by Naive Bayes. Each word contributes independently of each other to the final score. This is a reason that a lot of spam e-mails include random excerpts from the passages of books so as to effectively add “ hammy” words in the hope that the spam e-mail will bypass the spam filters. For this reason, in practice, many commercial e-mail systems (consider Gmail) likely use a lot more sophisticated spam detection models.