COMP24111 – Ex 3:
Naïve Bayes Classifier for Spam Filtering

Deadline: See the teaching website.
Extension policy: **NO EXTENSIONS** (EVEN ON REQUEST)

In general, a typical pattern recognition system consists of three modules; i.e., pre-processing, feature extraction and classification. As a pattern recognition task, a spam filterer needs to pre-process text messages for cleaning up data, extract salient features for distinguishing between normal and spam mails that form a feature vector for an email message, and classify email messages based on feature vectors for making a decision. While pre-processing and feature extraction require knowledge in the field of natural language processing and computational linguistics, classification needs to find a proper classifier to provide the underpinning technique for decision-making.

Bayesian methods have become very widely used for spam filtering, and you may have such a filter on your computer. Mozilla has a Bayesian filter, Spam Assassin uses Bayesian methods, and even Google uses a Bayesian filter to prevent "search-engine spammers" from dominating search results. In this lab, you are going to implement naïve Bayes classifiers for spam filtering emails based on a UCI machine learning benchmark database named `spambase` (http://archive.ics.uci.edu/ml/datasets/Spambase) where **57** attributes (features) have already been extracted for each email message and all instances were labelled as "**0**" (normal) or "**1**" (spam). The purposes of this lab is two-fold: 1) to use Matlab to implement the naïve Bayes algorithms learned from the module, as underpinning techniques, for developing a spam filter working in different situations, and 2) to enhance your understanding of the naïve Bayes classifier, its application to spam filtering and relevant performance evaluation. **In this lab, you are NOT allowed to use any Matlab naiveBayes build-in functions in your code.**

In order to apply the discrete-valued naïve Bayes classifier for spam filtering, the `spambase` database has been modified by the further discretisation of continuous attribute values. Also some instances have been re-labelled as "2" to indicate "uncertain", a newly added class.

You need to do three things:

1) Set up your path (and ensure it stays set up for every Matlab session)
2) Copy datafiles

then (and ONLY then) …

3) Do the lab exercises.

SET UP YOUR PATH
The following directory needs to be added to your Matlab path:

```
/opt/info/courses/COMP24111/ex3
```

You have to do this *within* Matlab - either with the "addpath" function (type: help addpath) which will modify the path only for the current session, or you can change it permanently with the "pathtool" (type: help pathtool).

COPY DATAFILES
Go to the directory: **`/opt/info/courses/COMP24111/ex3`**
you will find **four data files with the suffix `.mat` and the original `spamebase.data`** as follows:

- `av2_c2` - a dataset for binary classification where each attribute has two discrete values.
- `av3_c2` - a dataset for binary classification where each attribute has three discrete values.
- `av7_c3` - a dataset for three-class classification where each attribute has seven discrete values.
- `avc_c2` - a dataset for binary classification where all attributes have continuous values.
- `Spambase.data` – the original UCI spambase database (for the detailed information, see http://archive.ics.uci.edu/ml/datasets/Spambase).

Apart from `Spambase.data`, each of the aforementioned datasets contains the same data format as follows:

```
AttributeSet     –  a M*57 matrix where each row represents the
                    feature vector of an training example, M is
                    the number of training examples.
LabelSet         –  a M*1 column vector where elements
                    represent the labels of corresponding to
                    training examples in AttributeSet.
testAttibuteSet  –  a N*57 matrix where each row represent the
                    feature vector an test instance, N is the
                    number of test instances.
validLabel       -  a N*1 column vector where elements
                    represent the labels corresponding to
                    test instances in testAttributeSet (used
                    to achieve the accuracy of a classifier).
```

In addition, a function (`main.m`) is also provided to load a dataset for training and testing your implementation. As a result, you need to embed your code into this `main` function for evaluation. After embedding your code to this `main` function, you need to run this function only.
A typical scenario for machine learning is to create a learning system by training it on a given training data set. Later on the system will be applied to different test data sets. As a result, *you must write two generic functions; i.e., one for training a naïve Bayes classifier and the other for test based on the trained naïve Bayes classifier so that they can be applied to* ***any datasets*** *with the* `main` *function.* A typical format is shown below where "Parameter List" stands for a set of data structures (e.g., vector and matrix) used to store conditional probabilities and prior probabilities to be learned. You need to design such data structures yourself for this purpose.

```
% for NB training
[ "Parameter List" ] = NBTrain(AttributeSet, LabelSet)
% for NB test
[predictLabel, accuracy] = NBTest("Parameter List",
                                  testAttributeSet, validLabel)
```

Note that the `accuracy` of a naïve Bayes classifier is defined as follows:

$$accuracy = \frac{\text{number of correctly classified test instances}}{\text{number of test instances}}$$

For example, if you are given a test dataset of 100 instances and your code correctly predicts labels (i.e., predicted labels are as same as their true labels) for 85 instances, the accuracy is 85/100 = 0.85. Apart from the accuracy measure, you also need to report test results with a confusion matrix as described in lab 2 by appending the corresponding code to the end of the `main` function. If you wrote your code for calculating a confusion matrix for lab 2, you can reuse it here.

Now… the lab exercises…..

PART 1 – Implementation for discrete input attribute values

The main task of this exercise is to implement a naïve Bayes classifier for discrete input attribute values (see lecture notes of Naïve Bayes Classifier for details). Your code must be able to handle all possible situations; i.e., different attributes could have a various number of values (e.g., the first attribute has two values, the second one has four values and so on) and a problem could be either a binary or a multi-class classification task. If you do not have a clear idea of implementing a generic naïve Bayes classifier at the beginning, you might want to start coding a simplified naïve Bayes classifier for a simple situation, e.g., all attributes have only two values and the task is binary

classification. After gaining the experience on simple situations, you should be able to work on a generic naïve Bayes classifier. With the `main` function, you should test your code based on three datasets of discrete input attribute values. In terms of functionalities, three goals are set for Part 1:

1) your code works for binary attribute values and binary classification tasks,
2) your code works for multiple attribute values and binary classification tasks
3) your code works for different attribute values and multi-class classification tasks.

PART 2 – Implementation for continuous input attribute values

Upon the completion of the task of Part 1 described above, you need to further implement a naïve Bayes classifier for continuous input attribute values where attributes are assumed to be subject to Gaussian distribution (see lecture notes of Naïve Bayes Classifier for details). As same as required in Part 1, you must have two generic functions; one for training and the other test. With the same `main` function, you are still able to test your code with the given dataset of continuous input attribute values. Based on this continuous naïve Bayes implementation, you need to further write a programme for conducting a **10-fold cross validation** test with a proper setting on the original `spambase.data`. Based on your cross-validation setting, you will report the **mean & standard deviation of accuracy**.

DELIVERABLES

By the deadline, you should submit two files, using the **`submit`** command as you did last year:

```
ex3code.zip      - all your .m files, zipped up
ex3report.pdf    - your report, as detailed below
```

A report should be submitted, on **two one-side A4 pages** as a PDF file**. Marking will be based on both the report and the code submitted.** In other words, a new version of the report inconsistent with the submitted one is NOT accepted. The lab ex. will NOT be marked if only the code is submitted. Anything beyond 2 one-side A4 pages will be ignored**. In your report, it is essential for you to answer the following questions:**

1) What are those "parameters" that need learning in discrete and continuous naïve Bayes?
2) How do you store the learned "Parameter List" in your programme for Parts 1 & 2?
3) What are test results on all the given data sets described in Parts 1 & 2?
4) Wat are your test results described in Part 2? What are the motivation and setting(s) in your cross-validation experiments?
5) Based on your observation and analysis on experimental results achieved in Parts 1 & 2, can you grasp any non-trivial implication? If any, ***in your report***, you must ***explicitly*** describe your experimental evidence or theoretical justification that leads to such an implication.

Take note, we are not interested in the details of your code, e.g., what Matlab functions are called, what they return etc. This course unit is about machine learning algorithms and is indifferent to how you programme them. **The marking will be based on not only the performance/implementation but also, more importantly, the understanding of naïve Bayes as well as any implication that could be explored based on experimental results, the essence underlying machine learning. Anything inconsistent with the submitted report will not be counted during marking.**

The lab is marked out of 15 that will be allocated as follows:

- Implementation for naïve Bayes of discrete input attribute values and test **(8 marks)**
- Implementation for naïve Bayes of continuous input attribute values and test **(5 marks)**
- Analysis of experimental results and Implications **(2 marks)**

Marking will be done in Lab. Detailed marking criteria are on the basis of:

- How informative in your report about your implementation and experiments
- Functionality of your program and rigorous experimentation
- Knowledge displayed when talking to the TA marker