**IAML LAB 3: SVM CLASSIFICATION AND EVALUATION**

In this lab we initially re-examine the spam filtering problem from Lab 1. This time, we train a Logistic Regression model and a linear Support Vector Machine for the spam or non-spam classification task. In the second part of the lab we examine both visualisation and more rigorous methods for feature selection.

**SPAM FILTERING**

1. Start up Weka, select the Explorer interface and load the preprocessed Spambase data set from Lab 1, where all attributes are converted to Boolean and randomize the instances.   
   **Cheat:** If you did not save this data set, download it [here](http://www.inf.ed.ac.uk/teaching/courses/iaml/lab/data/spambase_binary.arff).
2. Now it's time to train our classifiers. The task is to classify e-mails as spam or non-spam and we evaluate the performance of Logistic Regression and Support Vector Machines on this task. Go to the *Classify* tab and select *Choose > functions > SimpleLogistic*. Select the percentage split and set it to 10%. This is done in order to save us waiting while Weka works hard on a large data set.

Click *Start* to train the model. Examine the *Classifier output* frame to view information for the model you've just trained and try to answer the following questions:

* + What is the percentage of correctly classified instances?
  + How do the regression coefficients for class 1 relate to the ones for class 0? Can you derive this result from the form of the Logistic Regression model?
  + Write down the coefficients for class 1 for the attributes [word\_freq\_hp\_binarized] and [char\_freq\_$\_binarized]. Generally, we would expect the string $ to appear in spam, and the string hp to appear in non-spam e-mails, as the data was collected from HP Labs. Do the regression coefficients make sense given that class 1 is spam? *Hint:* Consider the sigmoid function and how it transforms values into a probability between 0 and 1. Since our attributes are boolean, a positive coefficient can only increase the total sum fed through the sigmoid and thus move the output of the sigmoid towards 1. What can happen if we have continuous, real-valued attributes?

1. We will now train a Support Vector Machine (SVM) on our classification task. In the *Classify* tab, select *Choose > functions > SMO* (SMO stands for Sequential Minimal Optimization, which is an algorithm for training SVMs). Use the default parameters and click *Start*. This will train a linear SVM (which is quite similar to logistic regression). Again, examine the *Classifier output* frame and try answering the following:
   * What is the percent of correctly classified instances? How does it compare to the result from Logistic Regression?
   * What are the coefficients for the attributes [word\_freq\_hp\_binarized] and [char\_freq\_$\_binarized]? Compare these to the ones you found with Logistic Regression.
   * How does a linear SVM relate to Logistic Regression? *Hint:* Consider the classification boundary learnt in each model.

**PERFORMANCE ASSESSMENT #1**

We will now look at a few ways of assessing the performance of a classifier. To do so we will introduce a new data set, the [Splice](http://archive.ics.uci.edu/ml/datasets/Molecular+Biology+(Splice-junction+Gene+Sequences)) data set. The classification task is to identify *intron* and *exon* boundaries on gene sequences. Read the description at the link above for a brief overview of how this works. The class attribute can take on 3 values: N, IE and EI. Now download the data sets below, converted into ARFF for you, and load the training set into Weka:

* [splice\_train.arff](http://www.inf.ed.ac.uk/teaching/courses/iaml/lab/data/splice_train.arff): training data
* [splice\_test.arff](http://www.inf.ed.ac.uk/teaching/courses/iaml/lab/data/splice_test.arff): test data

1. We'll also use a new classifier. Under the *Classify* tab, select *classifiers > lazy > IBk*. This is a K-nearest neighbour classifier.
2. In the *Test options* panel, select *Use training set* and hit *Start*.
3. Observe the output of the classifier and consider the following:
   * What is the classification accuracy?
   * Is this meaningful?
   * Why is testing on the training data a particularly bad idea for a 1-nearest neighbour classifier?
   * Do you expect the performance of the classifier on a test set to be as good?
4. Now evaluate the classifier on the test set and check your expectations. In the *Test options* panel, select *Supplied test set* and load the file *splice\_test.arff*. In the *Result list* panel, right-click on the classifier and select *Re-evaluate model on current test set*. Observe the output and consider the following:
   * What would be the accuracy of the classifier, if all points were labelled as *N*?   
     *Hint:* View the distribution of the *class* attribute of the test data. You can do this by loading the test data on the *Preprocess* tab, and selecting the *class* attribute in the *Attributes* panel.
5. Now explore the effect of the *k* parameter. To do this, train the classifier multiple times, each time setting the *KNN* option to a different value. Try 5, 10, 100, 1000 and 10000 and test the classifier on the test set. *Hint:* To change the *KNN* option you need to bring up the options panel of the classifier.
   * How does the *k* parameter effect the results?   
     *Hint:* Consider how well the classifier is generalising to previously unseen data, and how it compares to the base rate again.
   * Plot the results (k-value on the x-axis and PC on the y-axis), making sure to mark the axis. Can you conclude anything from observing the plot?