Data Mining Lab 9: Random Forests

1 Introduction

In this final lab we are going to look at the last major topic on the course: random forests.

Since this is the final lab, it is going to be more open ended to allow you to practise all the statistical theory and practical `R` that you have learned throughout the semester. Therefore, Section 2 covers how to use random forests in `R` and then Section 3 invites you to experiment and use your imagination and ingenuity.

If you need to refer to previous labs or to download the data set, they will be on the course labs website:

`http://www.maths.tcd.ie/~louis/DataMining/`

2 Random Forests

2.1 Getting Setup

Exercise: Read in the German credit data and create a $\frac{2}{3} : \frac{1}{3}$ `train:test` split. (Hint: refer back to your lab 8 solutions if you've forgotten this)

```
>
>
>
>
```

Exercise: Load the `randomForest` package, which contains the functions to build classification trees in `R`. (Hint: see lab 2, §3.1 if you've forgotten)

```
>
```

2.2 Fitting the Model

We are now in a position to fit a random forest model to the data set, which can be done as follows:

```
> fit = randomForest(good_bad ~ ., data=train, ntree=500,
                     importance=TRUE, proximity=TRUE)
> fit
```

You will see that the output gives us an estimate of the error of the classifier and a guideline confusion matrix. However, we have a train and test split so we can verify these estimates ourselves (later).
Exercise: Look at `?tuneRF` and figure out how to use it to fit a tree where the number of variables randomly sampled as candidates at each split is automatically tuned. (Hint: you don't specify a formula here ... just pass the correct columns of the `train` matrix for x and y and have a look at the doBest option)

2.3 Variable Importance

The theory behind random forests provides a very natural way to rate the importance of variables, since we are permuting different variables being left out of the trees fitted in our forest. It is easy to extract this information in `R`:

```
> importance(fit)
> varImpPlot(fit)
```

2.4 Partial Dependence

Digging deeper than just which variable is important, we can actually examine the effect of different values of a given variable on the class prediction. In other words, we saw that under the mean decrease in Gini measure, the purpose of the loan was relatively high importance as a variable — but, what is the effect of the different loan purposes (car/furniture/etc) on the probability of default on the loan?

Looking at a partial dependence plot can tell us more:

```
> partialPlot(fit, test, purpose, "Good")
```

This bar chart tells us the marginal effect that each category of the loan purpose variable has on the probability of being classed as a *good* credit risk.
Exercise: If you are assessing the risk of a borrower who wants a loan for a car and you learn that the car is used, then considering only the marginal effect are you more likely to rate them as a good credit risk than if they had told you they were borrowing to buy a new car? (Hint: You might need to look up the meaning of codes A-J from the PDF file on the course website)
Exercise: Do a partial dependence plot for the marginal effect that the age variable has on the probability of being classed as a good credit risk. Note the interesting features of the plot, including the rapid increase in credit worthiness through the 20s and the rapid drop-off after retirement age.

2.5 Other Model Evaluation

The other kinds of model evaluation we have encountered on the course are equally valid for evaluating random forests.
Exercise: Compute the confusion matrix for the classifier you have created. (Hint: see lab 8, §2.2.1 if you've forgotten)

```
>
```

Exercise: Compute the ROC curve for the classifier you have created. (Hint: see lab 8, §2.2.2 if you've forgotten)

> **Note:** by default, using `predict()` on a random forest just returns the predicted classes and *not* the probabilities. Look at `?predict.randomForest` to figure out how to get the probabilities instead.

```
>
>
>
```

3 Comparing and Contrasting

You should have seen enough `R` now that you can build classification trees, neural networks and random forests for different classification problems. Therefore, we can explore these three techniques using the German credit risk data as the subject. If you are doing this section properly it is highly unlikely you'll finish within the lab time, but if you can do *any* of these without the structured guidance usual for the labs then it demonstrates a good grasp of working with `R`.

First, some things to bear in mind:

- Fit each model on the same training data so that you can directly compare the results you are getting. Don't just run the fitting command and forget it, but look at the output and make sure it is making sense. You might have to tune things (*eg* decay for neural networks).
- Store each model in different variables. In other words, don't just use `fit`, but use `fittree`, `fitnn` and `fitrf` so that you can work with all at once.
- If you want to compare plots side-by-side, remember the handy pre-plotting command `par(mfrow=c(1,3))`. The 1 is the number of rows and 3 the number of columns for the grid of plots: vary these as needed.

Exercise: Use the German credit data as a data set with which to compare all the approaches to classification you have learned on the course. Let your imagination run wild! Some ideas to get you started (each progressively harder):

1. Compare the confusion matrices for each technique.

2. See which technique gives the most convincing model evaluation plots (ROC / lift / etc).

3. If you were writing a program to perform classification on data arriving in real time, then speed of fitting and predicting may be an issue. Explore the `system.time()` command to compare the run-time of the different models.

4. We didn't explore varying the number of trees in our forest. Try out a for-loop to fit a forest of 100 to 1,000 trees in steps of 100 and plot the overall misclassification rate on the test data against forest size.

5. All the techniques allow tweaking how hard we penalise false positives or false negatives by adjusting where we take the probability cut-off (by default 0.5). Think about how a lender might setup the problem to minimise losses as a result of misclassification by adjusting the cut-off in each method.

   For example, imagine a new loan is being assessed and in this scenario the person defaulting on the loan may cost the lender €1,000 whereas turning down a person who would pay loses them €600 in charges.

   For all three techniques, tune the cut-off on the train data and see how much would be lost by the lender through the two types of misclassification when applied to the test set. Which classifier is best in this scenario?

6. Random forests are based on the idea of building a more powerful classifier from many smaller ones. We can do likewise! Build your own new classifier which is a model averaging of a tree, a neural network and a random forest.

   In other words, a classifier which predicts class membership probabilities as the average of the class probabilities given by each of the three techniques. Test your classifier using confusion matrices, ROC curves etc and compare it to the three individual models.

   A step up from this is weighted averaging where the weights could be chosen to max/min-imise some criterion.