Data Mining Lab 6: Random Forests

1 Introduction

In this lab we are going to look at random forests. This is only a very brief overview of the R package `randomForest`. We are going to use the churn dataset to illustrate the basic commands and plots.

2 Random Forests

2.1 Getting Setup

Exercise: Load the `randomForest` package, which contains the functions to build classification trees in `R`. Save the pdf file which explains how to run the package. Then load the churn dataset.

```
>
```

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(churn~.,data=data[3:18],ntree=1000,
   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. Again remember to see what else is included in object `fit` by using the commands `str` or `names(fit)`

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)
```

The above plots the average variable importance. As we saw in lectures we can plot this for each of the groups as well.

```
varImpPlot(fit,main=" Average Importance plots")
varImpPlot(fit,class="Yes",main=" Class= Yes Importance plots")
varImpPlot(fit,class="No",main=" Class= No Importance plots")
```

The above also shows how we can add titles to our plots.

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. Looking at a partial dependence plot can tell us more. The following plots for variable `svc.calls` for the `No` category.

```
> partialPlot(fit, data, svc.calls, "No",main="For the No category")
```

2.5 Other plots

To plot the margins do the following:

```
 margins.rf=margin(fit,churn)
plot(margins.rf)
hist(margins.rf,main="Margins of Random Forest for churn dataset")
boxplot(margins.rf~data$churn,
  main="Margins of Random Forest for churn dataset by class")
```

The error rate over the trees is obtained as follows:

```
plot(fit, main="Error rate over trees")
```

The multidimensional scaling plot is obtained as follows where k = number of dimensions.

```
MDSplot(fit, data$churn, k=2)
```

2.6 Other parameters used as input to rarndomForest

- mtry= number of variables selected at each node - default = $\sqrt{no.of variables}$
- ntree = number of trees to grow: default = 500
- nodesize=minimum size of terminal nodes default =1

2.7 Other Model Evaluation

The other kinds of model evaluation we have encountered on the course are equally valid for evaluating random forests. You can compute predicted values as follows.

```
> pred1=predict(fit,type="prob")
```

Again this creates two columns - where the first column is the probability of a "NO" and the second the probability of a "YES". You can use these to generate ROC curves etc.

2.8 Tuning a Random Forest

You may want to see what happens when you try different values for `mtry` the number of variables selected at each node. There is a function called tuneRF which tries different values of `mtry`. You could also program this yourself.

```
>
>
>
```