MS1b: SDM - Practical Sheet 4

Decision trees and ensembles

The purpose of this example is to introduce you to classification and regression trees (in a practical sense), to see the role V-fold classification plays in model-selection and get a feeling for Bagging, Random Forests, and out-of-bag error estimation.

We begin by using the first 100 data samples of the Boston Housing data set

```
library(rpart)
library(MASS)

data(Boston)
price <- Boston[1:100,14]  ## response variable
X <- Boston[1:100,-14]     ## predictor variables
```

Recall that splits are based on some measure of impurity $R(T)$. For regression, this would measure the residual variance in the leaf nodes of the tree T. Pruning is based $R_\alpha(T) = R(T) + \alpha\,\text{size}(T)$ say. The `CP` parameter of `rpart` is $\alpha/R(T)$ where T is the tree we are pruning. In other words, a node is only splitting if the loss $R(T)$ is reduced by at least a factor `CP`.

Use the `rpart()` function to fit a regression tree (using `method="anova"` instead of `method="class"`). Since the data set is small splits of leaves with 5 or more data vectors in each non-empty class (`minsplit=5`) are allowed. Note we can prune back if we have grown the tree too large. The `rpart` function does V-fold cross validation, as part of its analysis, so we specify V (`xval=10`).

We use the Boston Housign dataset of the lectures.

```
boston.tree <- rpart(price ~ ., data=X,
                     control = rpart.control(minsplit=2,xval=10),
                     method="anova")
```

There are some useful visualization tools

```
plot(boston.tree, margin=0.1)
text(boston.tree)
```

Examine the output from the following and ask for help interpreting it

```
summary(boston.tree)
```

Now the pruning. Prune at the smallest tree with `xerror` within 1 std. dev. of `CP` value giving minimum `xerror` we achieved.

```
(boston.cpt <- printcp(boston.tree))
plotcp(boston.tree)
prunedtree <- prune(boston.tree,cp=0.051)
plot(prunedtree); text(prunedtree)
```

Prices can be predicted on the training data by

```
predictedvalues <- predict(prunedtree)
```

You could look at the fit on the training data by i.e.

```
plot(predictedvalues,price)
```

Now you can also predict the data on the left-out samples, the test set, by

```
predicttest <- predict(prunedtree,newdata=Boston[101:500,1:13])
```

and compute the mean squared error

```
print(  mean( (Boston[101:500,14] - predicttest)^2 ))
```

The training error goes to zero as the tree is grown larger (and we are forcing splits even for leaves with relatively few data vectors). However the cross-validated (`x-val`) error identifies the overfit. To grow the tree very large initially, first repeat the tree fit with a low `cp` value.

```
boston.tree <- rpart(price ~ ., data=X,
   control = rpart.control( minsplit=2, xval=10,cp=0.001),method="anova")

boston.cpt <- printcp(boston.tree)
plotcp(boston.tree)

par(xaxt="n")
plot(1:nrow(boston.cpt),boston.cpt[,3],type='l',xlab="CP",ylab="error,xerror")
par(xaxt="s")
points(1:nrow(boston.cpt),boston.cpt[,4],type='b')
axis(1, at = 1:nrow(boston.cpt), labels = formatC(boston.cpt[,1], format="fg"))
axis(3, at = 1:nrow(boston.cpt), labels = formatC(boston.cpt[,2]+1, format="fg"))
```

The last two lines print the `CP` values on the x-axis and the number of leaf nodes on the top axis.

Repeat the analysis for the entire Boston Housing dataset.

```
price <- Boston[,14]
X <- Boston[,-14]
```

What is the cross-validated estimate of the error rate?

Now, fit a Random Forest classifier to the same dataset.

```
library(randomForest)
rf <- randomForest(X,price,ntree=200,nodesize=5,mtry=5)
```

Plot the out-of-bag estimates and compute the out-of-bag error rate.

```
plot(predict(rf), price)
abline(c(0,1))
print( mean( (predict(rf)-price)^2 ) )
```

Repeat for various values of `nodesize` (minimal nodesize) and `mtry` (number of randomly chosen variables over which to chose the best split). How does the error rate depend on `mtry` and `nodesize`? Make a plot of variable importance using the `varImpPlot` function.

New observations can be predicted as for trees by

```
predictrf <- predict(rf,newdata= Xnew)
```

where `Xnew` would be a data frame containing the same column names as the original data set `X`.

Lastly, bagging with out-of-bag prediction. First, create a matrix `OOB` which specifies if the sample is in or out of the bootstrap sample for each bootstrap iteration and another matrix `OOBpredict` which records the prediction for each bootstrap iteration. Use the value of `cp` found to be optimal under cross-validation.

```
nboot <- 100
n <- nrow(X)
OOB <- matrix(0,nrow=nboot,ncol=n)
OOBpredict <- matrix(0,nrow=nboot,ncol=n)
for (boot in 1:nboot){
  subsample <- sample(1:n,n,replace=TRUE)
  boston.treeboot <- rpart(price[subsample] ~ .,data=X[subsample,],cp=0.051,
                           control = rpart.control(minsplit=2,xval=10),
                           method="anova")
  OOB[boot, ! (1:n)%in%subsample] <- 1
  OOBpredict[boot,] <- predict(boston.treeboot,newdata=X)
}
```

Look at the matrices for example by using function `image.plot` in library `fields`,

```
image.plot(OOB)
image.plot(OOBpredict)
```

Compute the bagged estimator and find the out-of-bag estimation of the test error

```
OOBresult <- numeric(n)
for (i in 1:n){
```

```
  whichoob <- which(OOB[,i]==1)
  OOBresult[i] <- mean( OOBpredict[ whichoob  ,i])
}
print( mean((OOBresult-price)^2) )
```

Compare with the out-of-bag estimation of the test error when using Random Forests.