Part XI
Data Exploration and Discrimination – Largish Dataset

Note: These computations are at the limit, or beyond, what a machine with 512MB memory and running Microsoft Windows is able to handle. A reboot may be necessary in order to get the calculations to run. If you cannot get calculations to run at all, try taking every second observation in the relevant data frames, and working with these reduced datasets.

One a 512MB Mac system running OS X, calculations have run without problem. The same is almost certainly true on a Unix or Linux system (I have not tested this),

The data used in this laboratory gives forest cover type (seven different types), for 581,012 sites in the United States. There are 54 columns of explanatory features (variables or dummy variables that code for qualitative effects). The 55th column holds the cover type, given as an integer in the range 1 to 7. Data, stored in the R image file **covtype.RData**, are available from
`http://www.maths.anu.edu.au/~johnm/r/misc-data/`
[To obtain the data as a text file, go to the UCI Machine Learning Repository at
`http://www.ics.uci.edu/~mlearn/MLRepository.html`

1 Data Input and Exploration

As available from the repository, data are comma delimited, and (assuming that the file has been placed in the working directory; if accessible from another location, the path must be included) can be read in with

```
covtype <- read.csv("covtype.data", header=FALSE)
```

For purposes of this laboratory, data have been placed in the image file **covtype.RData**.

The image file `covtype.RData` holds all 581,012 records. The first 11340 records have been sampled in some systematic way from an original total data set, for use as a training set. The next 3780 records, again sampled in some systematic way from the total data, were used as test data. Below, the first 11340 records will be extracted into the data frame `covtrain`, while the next 3780 records will be extracted into the data frame `covtest`.

1.1 Extraction of subsets of interest

The following extracts these data, plus a systematic sample of every 50th record, taken right through the 565,982 records that remain after the training and test sets have been extracted (if these calculations prove troublesome on your system, skip them and obtain the data sets from a web page or from a CD). It is assumed that the file **covtype.RData** is in your working directory; if it is elsewhere you must of course include the path:

```
> attach("covtype.RData")
> covtrain <- covtype[1:11340, ]
> covtest <- covtype[11340 + (1:3780), ]
> every50th <- seq(from = 11340 + 3780 + 1, to = 581012, by = 50)
> covsample <- covtype[every50th, ]
```

> **Note:** These computations may, on some systems, be unreasonably time-consuming. On my 1.25GHz G4 Powerbook with 512MB of RAM, the elapsed time for extraction of `covtrain` was 74 seconds while that for extraction of `covtest` was 35 seconds (try, e.g., `system.time(covtest <- covtype[11340+(1:3780),])` – the third of these numbers is the elapsed time). On less well endowed systems, the calculations may take an unreasonable

amount of time, or may not run at all. I would be glad to have feedback on experience with such systems.

The reason that these apparently simple operations are so time-consuming is that because data are stored columnwise, extraction of records that lie within specific ranges requires substantial manipulation within memory.

An alternative, for the input of `covtrain` and `covtest`, is:

```
covtrain <- read.csv("covtype.data", header=FALSE, nrows=11340)
covtest <- read.csv("covtype.data", header=FALSE, skip=11340, nrows=3780)
```

(Use these on Windows machines with less than 512MB of random access memory.)

Question: Which of the following is preferable?

```
every50th <- seq(from=11340+3780+1, to=581012, by=50)
every50th <- seq(from=15121, to=581012, by=50)
every50th <- 15121+(0:((581012-15121) %/% 50))*50
```

Which is more consistent with notions of literate programming? Is computational efficiency a consideration?
(The symbol `%/%` is the integer division operator. Try, e.g., `11 %/% 3`, `12 %/% 3`, etc. Try also `11 %% 3`, `12 %% 3`, which give the remainders after division.)

1.2 Image files

Having extracted these data, they can be saved to image files, which can then be attached so that they are available as required:

```
> save(covtrain, file = "covtrain.RData")
> rm(covtrain)
> save(covtest, file = "covtest.RData")
> rm(covtest)
> save(covsample, file = "covsample.RData")
> rm(covsample)
```

Now attach these image files, making `covtrain`, `covtest` and `covsample` available as required:

```
> attach("covtrain.RData")
> attach("covtest.RData")
> attach("covsample.RData")
```

1.3 Data exploration

Next, we extract some basic statistical information about these data:

```
> options(digits = 3)
> tab.all <- table(covtype$V55)
> tab.train <- table(covtrain$V55)
> tab.test <- table(covtest$V55)
> tab.sample <- table(covsample$V55)
> tab.all/sum(tab.all)
> tab.sample/sum(tab.sample)
> tab.train/sum(tab.train)
> tab.test/sum(tab.test)
```

What is interesting is that the proportions of the different cover types, in both the training and the test data, are equal. That is not the case for the data as a whole, and in this respect the "training" and "test" data are untypical.

The above suggests that, in forming the training and test data, observations were taken from the original main body of data until cover types 3-7 were largely "used up". It might be suspected that the proportions of the different cover types will vary systematically as one moves through data. The following function, which models the occurrence of the specified forest cover as a function of distance (as a proportion) through the data, can be used to check this:

```
> library(splines)
> runningprops <- function(df = covtrain, type = 1) {
+     n <- dim(df)[1]
+     propthru <- (1:n)/(n + 1)
+     occurs <- as.integer(df$V55 == type)
+     print(table(occurs))
+     cov.glm <- glm(occurs ~ bs(propthru, 6), family = binomial)
+     hat <- predict(cov.glm, type = "response")
+     cbind(propthru, hat)
+ }
> hat.train <- runningprops()
> hat.test <- runningprops(df = covtest)
> hat.sample <- runningprops(df = covsample)
> print(range(c(hat.train[, 2], hat.test[, 2], hat.sample[, 2])))
```

Next, plot this information:

```
> plot(hat.train[, 1], hat.train[, 2], ylim = c(0, 0.65))
> lines(hat.test[, 1], hat.test[, 2], col = 2)
> lines(hat.sample[, 1], hat.sample[, 2], col = 3)
```

What does this plot suggest?

Exercise 1: Repeat the above plots, but now for forest cover type 2.

Exercise 2: Another way to estimate `hat` would be as a moving average of values of the variable `occurs` in the above function. Write a function to calculate moving averages, with the window `wid` as one of its parameters.

Exercise 3: What other preliminary explorations of these data might be useful?

2 Tree-Based Classification

Now fit a tree-based model for `covtrain`:

```
> library(rpart)
> train.rpart <- rpart(V55 ~ ., data = covtrain, cp = 1e-04, method = "class")
> train.rpart <- prune(train.rpart, cp = 0.0048)
> trainhat.train <- xpred.rpart(train.rpart, cp = 0.0048)
> testhat.train <- predict(train.rpart, newdata = covtest, type = "class")
> samplehat.train <- predict(train.rpart, newdata = covsample,
+     type = "class")
```

Next, we will define a function that calculates error rates for each different cover type:

```
> errs.fun <- function(obs, predicted) {
+     tab <- table(obs, predicted)
+     grosserr <- 1 - sum(tab[row(tab) == col(tab)])/sum(tab)
+     errs <- 1 - tab[row(tab) == col(tab)]/apply(tab, 1, sum)
+     names(errs) <- paste(1:length(errs))
+     print("Overall error rate (%)")
+     print(round(100 * grosserr, 1))
+     print("Error rate (%), broken down by forest cover type")
+     print(round(100 * errs, 2))
+     cat("\n")
+     invisible(errs)
+ }
```

Now apply the function, first to `trainhat.train`, then to `testhat.train`, and finally to `samplehat.train`:

```
> errs.fun(covtrain$V55, trainhat.train)
> errs.fun(covtest$V55, testhat.train)
> errs.fun(covsample$V55, samplehat.train)
```

Exercise 4: Explain: calculations:

- What data have been used, in each case, to test the model?
- Explain the notation used for the different sets of fitted values (`trainhat.train`, `testhat.train`, `samplehat.train`.)
- Why is it that, although the three sets of error rates are relatively similar when broken down by cover type, are there are major differences in the overall error rate?
- Which, if any, of these overall error rates is it reasonable to quote in practical application of the results? If none of them seem appropriate, what error rate do you consider should be quoted?

Exercise 5: Do the following calculation and comment on the results:

```
> sample.rpart <- rpart(V55 ~ ., data = covsample, cp = 1e-04,
+     method = "class")
> sample.rpart <- prune(sample.rpart, cp = 0.001)
> samplehat.sample <- xpred.rpart(sample.rpart, cp = 0.001)
> samplehat.sample <- factor(samplehat.sample, levels = 1:7)
> trainhat.sample <- predict(sample.rpart, newdata = covtrain,
+     type = "class")
> testhat.sample <- predict(sample.rpart, newdata = covtest, type = "class")
> errs.fun(covtrain$V55, trainhat.sample)
> errs.fun(covtest$V55, testhat.sample)
> errs.fun(covsample$V55, samplehat.sample)
```

3 Use of `randomForest()`

For use of `randomForest()`, calculations speed up greatly if explanatory variables are input as columns of a matrix, while (for classification) the response variable is input as a vector of class factor.

Exercise 6: Run the following computations, and interpret the output, comparing it with the relevant output from `rpart()`. Suggest why the error rates may be so much lower than those from `rpart()`:

```
> library(randomForest)
> xcovtrain <- as(covtrain[, 1:54], "matrix")
> ycovtrain <- covtrain[, 55]
> xsampletrain <- as(covsample[, 1:54], "matrix")
> ysampletrain <- covsample[, 55]
> ycovtrain <- factor(covtrain[, 55])
> ysampletrain <- factor(covsample[, 55])
> covtrain.rf <- randomForest(x = xcovtrain, y = ycovtrain, xtest = xsampletrain,
+     ytest = ysampletrain)
> covtrain.rf
```

4 Further comments

This has been a preliminary exploration of these data. The model that is optimal changes as one moves through the data. This can be verified by selecting successsive subsets of perhaps 10,000 successive observations at various points through the data (e.g.: 1-10,000, 101,000-110,000, ...), and comparing: (1) gross error rate for that subset as calculated by using `xpred.rpart()` and `errs.fun()`, with (2) the gross error rate when `train.rpart()` or (more appropriately) `sample.rpart()` is used determine fitted values for that subset.

Thus, after the first 15,120 (11,340 + 3780) records, a reasonable guess is that the order of records reflects an ordering of the data according to geographical location. The ordering holds information that can be used, even without knowledge of more exact geographical locations, to get improved model predictions.