Part VI
Linear Discriminant Analysis – Using `lda()`

The function `lda()` is in the Venables & Ripley *MASS* package. It may have poor predictive power where there are complex forms of dependence on the explanatory factors and variables. In cases where it is effective, it has the virtue of simplicity. Covariates are assumed to have a common multivariate normal distribution.

The function `qda()` weakens the assumptions underlying `lda()` to allow different variance-covariance matrices for different groups within the data. This sets limits on the minimum group size.

Where there are two classes, (use `glm()`) has very similar properties to linear discriminant analysis using `lda()`. It makes weaker assumptions. The trade-off is that estimated group membership probabilities are conditional on the observed matrix.

With all these "linear" methods, the model matrix can replace columns of covariate data by a set of spline (or other) basis columns that allow for effects that are nonlinear in the covariates. Use `termplot()` with a `glm` object, with the argument `smooth=panel.smooth`, to check for hints of nonlinear covariate effects. Detection of nonlinear effects may require very extensive data.

A good first check on whether these "linear" methods seem to be effective is given by comparison with the highly nonparametric analysis of the function `randomForest()`, from the *randomForest* package. This function may do well when complex interactions are required to explain the dependence.

At this point, attention will mostly be limited to data where there are two groups only.

1 Accuracy for Classification Models – the `Pima` Data

After attaching the *MASS* package, type `help(Pima.tr)` to get a description of these data. They are relevant to the investigation of conditions that may pre-dispose to diabetes. All the explanatory variables can be treated as continuous variables; there are no factor columns, or columns (e.g. of 0/1 data) that might be regarded as factors.

1.1 Fitting an lda model

We will first try linear discriminant analysis. As a first try, we use the model formula `type` $\sim$ `..` The effect is to use as explanatory variables all columns of `Pima.tr` except `type`.

We run the calculations twice: (1) the first run of the calculations has `CV=TRUE`, to get predictions of class membership that are derived from leave-one-out cross-validation; (2) the second run of the calculations has `CV=FALSE` (the default), allowing us then to use `predict()` to obtain an object that includes discriminant scores.

```
> library(MASS)
> PimaCV.lda <- lda(type ~ ., data = Pima.tr, CV = TRUE)
> tab <- table(Pima.tr$type, PimaCV.lda$class)
> conCV1 <- rbind(tab[1, ]/sum(tab[1, ]), tab[2, ]/sum(tab[2, ]))
> dimnames(conCV1) <- list(Actual = c("No", "Yes"), "Predicted (cv)" = c("No",
+     "Yes"))
> print(round(conCV1, 3))
```

```
      Predicted (cv)
Actual    No   Yes
   No  0.864 0.136
   Yes 0.456 0.544
```

```
> Pima.lda <- lda(type ~ ., data = Pima.tr)
> Pima.hat <- predict(Pima.lda)
> tabtrain <- table(Pima.tr$type, Pima.hat$class)
```

```
> conTrain <- rbind(tab[1, ]/sum(tab[1, ]), tab[2, ]/sum(tab[2,
+     ]))
> dimnames(conTrain) <- list(Actual = c("No", "Yes"), "Predicted (cv)" = c("No",
+     "Yes"))
> print(round(conTrain, 3))

      Predicted (cv)
Actual    No   Yes
   No  0.864 0.136
   Yes 0.456 0.544
```

The argument `CV=TRUE` generates leave-one-out cross-validation predictions of the class. Notice that, here, the two accuracy measures are the same. In general, the training set accuracy can be optimistic.

Now plot the discriminant scores. As there are two groups only, there is just one set of scores.

```
> library(lattice)
> densityplot(~Pima.hat$x, groups = Pima.tr$type)
```

A function that calculates the confusion matrices and overall accuracy would be helpful:

```
> confusion <- function(actual, predicted, names = NULL, printit = TRUE,
+     prior = NULL) {
+     if (is.null(names))
+         names <- levels(actual)
+     tab <- table(actual, predicted)
+     acctab <- t(apply(tab, 1, function(x) x/sum(x)))
+     dimnames(acctab) <- list(Actual = names, "Predicted (cv)" = names)
+     if (is.null(prior)) {
+         relnum <- table(actual)
+         prior <- relnum/sum(relnum)
+         acc <- sum(tab[row(tab) == col(tab)])/sum(tab)
+     }
+     else {
+         acc <- sum(prior * diag(acctab))
+         names(prior) <- names
+     }
+     if (printit)
+         print(round(c("Overall accuracy" = acc, "Prior frequency" = prior),
+             4))
+     if (printit) {
+         cat("\nConfusion matrix", "\n")
+         print(round(acctab, 4))
+     }
+     invisible(acctab)
+ }
```

1.2 The model that includes first order interactions

It may be that the outcome is influenced by whether covariates increase or decrease together. One way to check this is to include the effects of all products of variable values such as `npreg*glu`, `npreg*bp`, etc. In this instance, it will turn out that this leads to a model that is over-fitted.

Note that the model formula `(a+b+c)^2` expands to `a+b+c+a:b+a:c+b:c`. Note the following:

- `a:a` is the same as `a`.

- If `a` and `b` are (different) factors, then `a:b` is the interaction between `a` and `b`, i.e., it allows for effects that are due to the specific combination of level of `a` with level of `b`.
- If `a` is a factor and `b` is a variable, the interaction `a:b` allows for different coefficients of the variable for different levels of the factor.
- If `a` and `b` are (different) variables, then `a:b` is the result of multiplying `a` by `b`, element by element.

Exercise 1
Try adding interaction terms to the model fitted above:

```
> PimaCV2.lda <- lda(type ~ .^2, data = Pima.tr, CV = TRUE)
> confusion(Pima.tr$type, PimaCV2.lda$class)
> Pima2.hat <- predict(lda(type ~ .^2, data = Pima.tr))$class
> confusion(Pima.tr$type, Pima2.hat)
```

Notice that the training set measure (*resubstitution* accuracy or *apparent* accuracy) is now substantially exaggerating the accuracy. The model that includes all interactions terms is in truth giving lower predictive accuracy; it overfits.

1.3 Proportion correctly classified

Consider the fit

```
> PimaCV.lda <- lda(type ~ ., data = Pima.tr, CV = TRUE)
> confusion(Pima.tr$type, PimaCV.lda$class)
```

```
   Overall accuracy  Prior frequency.No Prior frequency.Yes
              0.755               0.660                0.340

Confusion matrix
      Predicted (cv)
Actual     No    Yes
   No  0.8636 0.1364
   Yes 0.4559 0.5441
```

The overall accuracy is estimated as 0.755. If however we keep the same rule, but change the prior proportions of the two classes, the overall accuracy will change. If for example, the two classes are in the ratio 0.9:0.1, the overall accuracy will be $0.9 \times 0.8636 + 0.1 \times 0.5441 \simeq 0.83$. The No's are easier to classify; with more of them the classification accuracy increases.

However the classification rule that is optimal also changes if the prior proportions change. The function `lda()` allows specification of a prior, thus:

```
> prior <- c(0.9, 0.1)
> PimaCVp.lda <- lda(type ~ ., data = Pima.tr, CV = TRUE, prior = prior)
> confusion(Pima.tr$type, PimaCVp.lda$class, prior = c(0.9, 0.1))
```

```
   Overall accuracy  Prior frequency.No Prior frequency.Yes
             0.9104              0.9000               0.1000

Confusion matrix
      Predicted (cv)
Actual     No    Yes
   No  0.9773 0.0227
   Yes 0.6912 0.3088
```

If the rule is modified to be optimal relative to the new prior proportions, the accuracy thus increases to 0.91, approximately.

Exercise 2
Now assume prior proportions of 0.85 and 0.15. Repeat the above calculations, i.e.

- Estimate the accuracy using the rule that is designed to be optimal when the prior proportions are as in the sample.
- Estimate the accuracy using the rule that is designed to be optimal when the prior proportions are 0.85:0.15.

1.4 The ROC (receiver operating characteristic)

It is common to speak of sensitivity and specificity. With prior proportions as in the sample (0.755:0.245), the sensitivity (true positive rate) was estimated as 0.544; this is the probability of correctly identifying a person who is a diabetic as a diabetic. The false positive rate (1 - Specificity) was estimated as 0.136. There is a trade-off between sensitivity and specificity. The ROC curve, which is a plot of sensitivity against specificity, displays this trade-off graphically.

The analysis assumes that the cost of both types of mis-classification are equal. Varying the costs, while keeping the prior probabilities the same, is equivalent to keeping the costs equal, but varying the prior probabilities. The following calculation takes advantage of this equivalence.

Exercise 3
Run the following calculations:

```
> truepos <- numeric(19)
> falsepos <- numeric(19)
> p1 <- (1:19)/20
> for (i in 1:19) {
+     p <- p1[i]
+     Pima.CV1p <- lda(type ~ ., data = Pima.tr, CV = TRUE, prior = c(p,
+         1 - p))
+     confmat <- confusion(Pima.tr$type, Pima.CV1p$class, printit = FALSE)
+     falsepos[i] <- confmat[1, 2]
+     truepos[i] <- confmat[2, 2]
+ }
```

Now plot the curve.

```
> plot(truepos ~ falsepos, type = "l", xlab = "False positive rate",
+     ylab = "True positive rate (Sensitivity)")
```

Read off the sensitivity at a low false positive rate (e.g., 0.1), and at a rate around the middle of the range, and comment on the tradeoff.

The ROC curve allows assessment of the effects of different trade-offs between the two types of cost.

1.5 Accuracy on test data

There is an additional data set – `Pima.te` – that has been set aside for testing. The following checks the accuracy on these "test" data.

```
> Pima.lda <- lda(type ~ ., data = Pima.tr)
> testhat <- predict(Pima.lda, newdata = Pima.te)
> confusion(Pima.te$type, testhat$class)

   Overall accuracy  Prior frequency.No Prior frequency.Yes
             0.7982              0.6717               0.3283

Confusion matrix
      Predicted (cv)
Actual     No    Yes
   No  0.8879 0.1121
   Yes 0.3853 0.6147
```

This is an improvement on the leave-one-out CV accuracy on the training data. The difference in the prior proportions is too small to have much effect on the overall accuracy. The apparent improvement might be a result of chance. Another possibility that has to be contemplated is that the division of the data between `Pima.tr` and `Pima.te` may not have been totally random, and `Pima.te` may have fewer hard to classify points. There are two checks that may provide insight:

- Swap the roles of training and test data, and note whether there the relative accuracies are similar.
- Repeat the calculations on a bootstrap sample of the training data, to get an indication of the uncertainty in the accuracy assessment.

Exercise 4
Try the effect of swapping the role of training and test data.

```
> swapCV.lda <- lda(type ~ ., data = Pima.te, CV = TRUE)
> confusion(Pima.te$type, swapCV.lda$class)

   Overall accuracy  Prior frequency.No Prior frequency.Yes
             0.7831              0.6717               0.3283

Confusion matrix
      Predicted (cv)
Actual     No    Yes
   No  0.8879 0.1121
   Yes 0.4312 0.5688

> swap.lda <- lda(type ~ ., data = Pima.te)
> otherhat <- predict(Pima.lda, newdata = Pima.tr)
> confusion(Pima.tr$type, otherhat$class)

   Overall accuracy  Prior frequency.No Prior frequency.Yes
               0.77                0.66                 0.34

Confusion matrix
      Predicted (cv)
Actual     No    Yes
   No  0.8712 0.1288
   Yes 0.4265 0.5735
```

Note that, again, the accuracy is greater for `Pima.te` than for `Pima.tr`, but the difference is smaller.

Exercise 5
Now check the accuracy on a bootstrap sample:

```
> prior <- table(Pima.tr$type)
> prior <- prior/sum(prior)
> index <- sample(1:dim(Pima.tr)[1], replace = TRUE)
> boot.lda <- lda(type ~ ., data = Pima.tr[index, ], CV = TRUE)
> cmat <- confusion(Pima.tr[index, "type"], boot.lda$class, printit = FALSE)
> print(c(acc = round(prior[1] * cmat[1, 1] + prior[2] * cmat[2,
+     2], 4)))

acc.No
0.7978
```

The calculations should be repeated several times. The changes in the predictive accuracy estimates are substantial.
Note the need to relate all accuracies back to the same prior probabilities, to ensure comparability.
Annotate the code to explain what it does.

(From running this code five times, I obtained results of 0.77, 0.74, 0.72, 0.71 and 0.82.)

2 Logistic regression – an alternative to `lda`

As the Pima data have only two classes (levels of `type`) the calculation can be handled as a regression problem, albeit with the reponse on a logit scale, i.e., the linear model predicts $\log(\frac{\pi}{1-\pi})$, where π is the probability of having diabetes.

Exercise 6
Fit a logistic regression model and check the accuracy.

```
> Pima.glm <- glm(I(unclass(type) - 1) ~ ., data = Pima.tr, family = binomial)
> testhat <- round(predict(Pima.glm, newdata = Pima.te, type = "response"))
> confusion(Pima.te$type, testhat)
```

Compare the accuracy with that obtained from `lda()`.
A cross-validation estimate of accuracy, based on the training data, can be obtained thus:

```
> library(DAAG)
> CVbinary(Pima.glm)

Fold:  2 9 6 7 3 10 8 1 4 5
Internal estimate of accuracy = 0.775
Cross-validation estimate of accuracy = 0.745
```

This should be repeated several times. How consistent are the results?

One advantage of use of `glm()` is that asymptotic standard error estimates are available for parameter estimates:

```
> round(summary(Pima.glm)$coef, 3)
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-9.773	1.770	-5.520	0.000
npreg	0.103	0.065	1.595	0.111
glu	0.032	0.007	4.732	0.000
bp	-0.005	0.019	-0.257	0.797

```
skin          -0.002      0.022  -0.085    0.932
bmi            0.084      0.043   1.953    0.051
ped            1.820      0.666   2.735    0.006
age            0.041      0.022   1.864    0.062
```

These results suggest that `npreg`, `bp` and `skin` can be omitted without much change to predictive accuracy. Predictive accuracy may actually increase. There is however, no guarantee of this, and it is necessary to check. Even though there is individually no detectable effect, the combined effect of two or more of them may be of consequence.

Using this logistic regression approach, there is no built-in provision to adjust for prior probabilities. Users can however make their own adjustments.

One advantage of `glm()` is that the `termplot()` function is available to provide a coarse check on possible nonlinear effects of covariates. Use `termplot()` with a `glm` object as the first argument, and with the argument `smooth=panel.smooth`. The resulting graphs can be examined for hints of nonlinear covariate effects. Detection of nonlinear effects may require very extensive data.

3 Data that are More Challenging – the `crx` Dataset

The data can be copied from the web:

```
> webpage <- "http://mlearn.ics.uci.edu/databases/credit-screening/crx.data"
> webn <- "http://mlearn.ics.uci.edu/databases/credit-screening/crx.names"
> test <- try(readLines(webpage)[1])
> if (!inherits(test, "try-error")) {
+     download.file(webpage, destfile = "crx.data")
+     crx <- read.csv("crx.data", header = FALSE, na.strings = "?")
+     download.file(webn, destfile = "crx.names")
+ }
```

Column 16 is the outcome variable. Factors can be identified as follows:

```
> if (exists("crx")) sapply(crx, function(x) if (is.factor(x)) levels(x))
```

These data have a number of factor columns. It will be important to understand how they are handled.

3.1 Factor terms – contribution of the model matrix

As with normal theory linear models, the matrix has an initial column of ones that allows for a constant term. (In all rows, the relevant parameter is muitiplied by 1.0, so that the contribution to the fitted value is the same in all rows.) For terms that correspond directly to variables, the model matrix incoporates the variable directly as one of its columns. With the default handling of a factor term

- Implicitly there is a column that corresponds to the initial level of the factor, but as it has all elements 0 it can be omitted;
- For the third and any subsequent levels, the model matrix has a column that is zeros except for rows where the factor is at that level.

A factor that has only two levels will generate a single column, with 0s correponding to the first level, and 1s for the second level. The Pima data has, except for the response variable `type`, no binary variables.

3.2 Fitting the model

Exercise 7
Now fit a linear discriminant model:

```
> if (exists("crx")) {
+     crxRed <- na.omit(crx)
+     crxCV.lda <- lda(V16 ~ ., data = crxRed, CV = TRUE)
+     confusion(crxRed$V16, crxCV.lda$class)
+ }
```

Note the message

```
Warning message:
In lda.default(x, grouping, ...) : variables are collinear
```

Now, for comparison, fit the model using `glm()`. This is one way to get details on the reasons for collinearity. Also, for using `glm()`, the argument `na.action=na.exclude` is available, which omits missing values when the model is fit, but then places `NA`s in those positions when fitted values, predicted values, etc., are calculated. This ensures that predicted values match up with the rows of the orginal data.

```
> if (exists("crx")) {
+     crx.glm <- glm(V16 ~ ., data = crx, family = binomial, na.action = na.exclude)
+     alias(crx.glm)
+     confusion(crx$V16, round(fitted(crx.glm)))
+     summary(crx.glm)$coef
+ }
```

From the output from `alias(crx.glm)`, what can one say about the reasons for multi-collinearity?

Now display the scores from the linear discriminant calculations:

```
> if (exists("crx")) {
+     crxRed <- na.omit(crx)
+     crx.lda <- lda(V16 ~ ., data = crxRed)
+     crx.hat <- predict(crx.lda)
+     densityplot(~crx.hat$x, groups = crxRed$V16)
+ }
```

This plot is actually quite interesting. What does it tell you?

4 Use of Random Forest Results for Comparison

A good strategy is to use results from the random forests method for comparison. The accuracy of this algorithm, when it does give a worthwhile improvement over `lda()`, is often hard to beat. This method has the advantage that it can be applied pretty much automatically. It is good at handling situations where explanatory variables and factors interact in a relatively complex manner.

Here are results for `Pima.tr` as training data, at the same time applying predictions to `Pima.te` as test data. Notice that there are two confusion matrices, one giving the OOB estaimtes for `Pima.tr`, and the other for `Pima.te`.

```
> library(randomForest)
> Pima.rf <- randomForest(type ~ ., xtest = Pima.te[, -8], ytest = Pima.te[,
+     8], data = Pima.tr)
> Pima.rf
```

```
Call:
 randomForest(formula = type ~ ., data = Pima.tr, xtest = Pima.te[,      -8], ytest = Pima.te[, 8])
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 28.5%
Confusion matrix:
     No Yes class.error
No  110  22   0.1666667
Yes  35  33   0.5147059
                Test set error rate: 23.49%
Confusion matrix:
     No Yes class.error
No  192  31   0.1390135
Yes  47  62   0.4311927
```

Look at the OOB estimate of accuracy, which is pretty much equivalent to a cross-validation estimate of accuracy. This error will be similar to the error on test data that are randomly chosen from the same population.

The accuracy is poorer than for `lda()`. As before, the error rate is lower on `Pima.te` than on `Pima.te`. Note however the need to re-run the calculation several times, as the accuracy will vary from run to run.

Here are results for `crx`.

```
> if (exists("crxRed")) {
+     crx.rf <- randomForest(V16 ~ ., data = crxRed)
+     crx.rf
+ }

Call:
 randomForest(formula = V16 ~ ., data = crxRed)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 12.71%
Confusion matrix:
    +   - class.error
+ 253  43   0.1452703
-  40 317   0.1120448
```

Accuracy is similar to that from use of `lda()`.

5 Note – The Handling of NAs

The assumption that underlies any analysis that omits missing values is that, for purposes of the analysis, missingness is uninformative. This may be incorrect, and it is necessary to ask: Are the subjects where there are missing values different in some way?

The missing value issue is pertinent both to the Pima data and to the `crx` data. There is a further dataset, `Pima.tr2`, that augments `Pima.tr` with 100 subjects that have missing values in one or more of the explanatory variables. The question then arises: Is the pattern of missingness the same for those without diabetes as for those with diabetes?

The following shows the numbers of missing values for each of the variables

```
> if (exists("Pima.tr2", where = ".GlobalEnv", inherits = FALSE)) rm(Pima.tr2)
> sapply(Pima.tr2, function(x) sum(is.na(x)))

npreg   glu    bp  skin   bmi   ped   age  type
    0     0    13    98     3     0     0     0

> sum(!complete.cases(Pima.tr2))

[1] 100
```

Note that the variable `skin` accounts for 98 of the 100 subjects where there is one or more missing value.

A first step is to check whether the subjects with one or more missing values differ in some systematic manner from subjects with no missing values. The major issue is for values that are missing for `skin`. We start by creating a new variable – here named `complete` – that distinguishes subjects with missing values for `skin` from others. We omit observations that are missing on any of the other variables.

```
> newPima <- subset(Pima.tr2, complete.cases(bp) & complete.cases(bmi))
> newPima$NOskin <- factor(is.na(newPima$skin), labels = c("skin",
+     "NOskin"))
> newPima$skin <- NULL
```

The argument `labels=c("skin","NOskin")` takes the values (here `FALSE` and `TRUE`) in alphanumeric order, then making `skin` and `NOskin` the levels. Omission of this argument would result in levels `FALSE` and `TRUE`.[2]

We now do a linear discriminant analysis in which variables other than `skin` are explanatory variables.

```
> completeCV.lda <- lda(NOskin ~ npreg + glu + bp + bmi + ped +
+     age + type, data = newPima, CV = TRUE)
> confusion(newPima$NOskin, completeCV.lda$class)

      Overall accuracy   Prior frequency.skin Prior frequency.NOskin
                0.6937                 0.7042                 0.2958

Confusion matrix
        Predicted (cv)
Actual     skin NOskin
  skin   0.9700 0.0300
  NOskin 0.9643 0.0357
```

A linear discriminant analysis seems unable to distinguish the two groups. The overall accuracy does not reach what could be achieved by predicting all rows as complete.

5.1 Does the missingness give information on the diagnosis?

If there is a suspicion that it does, then a valid analysis may be possible as follows. Missing values of continuous variables are replaced by 0 (or some other arbitrary number). For each such variable, and each observation for which it is missing, there must be a factor, e.g. with levels `miss` and `nomiss`, that identifies the subject for whom the value is missing. Where values of several variables are missing for the one subject, the same factor may be used. This allows an analysis in which all variables, together with the newly created factors, are "present" for all subjects.

[2] NB also `factor(is.na(newPima$skin), levels=c(TRUE, FALSE))`; levels would then be `TRUE` and `FALSE`, in that order.