Section 7: Discriminant Analysis.

Ensure you have completed all previous worksheets before advancing.

1 Linear Discriminant Analysis

To perform Linear Discriminant Analysis in `R` we will make use of the `lda` function in the package `MASS`.

Load this package and have a look at the function's help file (we will also need the package `MASS` in order to use the `qda` function for Quadratic Discriminant Analysis).

Remember from lectures that LDA assumes the data within each group follows a Multivariate Normal distribution and that each such group distribution shares the same covariance matrix.

Task: Download the Salmon data that is available at:

`http://www.cs.tcd.ie/~houldinb/Index/MVA.html`

Save this file into `R` under the name `salmon` using the `read.table` command.

A plot of this data is available by entering the following:

```
> plot(salmon[,-1],col=as.factor(salmon[,1]))
```

In order to use LDA, we need to first split the data into a part used to train the classifier, and another part that will be used to test the classifier. For this example we will try an 80:20 split (we will again ignore the validation step for the purposes of exposition):

```
> strain=salmon[c(1:40,51:90),]
```

```
> stest=salmon[c(41:50,91:100),]
```

We are then able to train our classifier in the following way:

```
> lsol=lda(strain[,c(2,3)],grouping=strain[,1])
```

The above has told `R` to train the classifier on the data available in the second and third columns of the data set `strain` using the knowledge that this data is grouped in the way described by the first column of `strain`.

If you enter the following into `R` you will be returned with a list of summary information concerning the computation:

```
> lsol$prior

Alaska Canada
   0.5    0.5

> lsol$means

Group means:
       Freshwater  Marine
Alaska    100.550 422.275
Canada    138.625 368.650
```

The first of these provides the prior probability of group membership that is used in the analysis, which by default, is taken to be the class proportions in the training data. The second section provides the estimated group means for each of the two groups (which is no different from that which would be found by applying the `mean` function on the appropriate subset).

Additional information that is not provided, but may be important, is the single covariance matrix that is being used for the various groupings. You can find this manually by calculating the following:

$$\Sigma = \frac{(N_1 - 1)\Sigma_1 + (N_2 - 1)\Sigma_2}{N_1 + N_2 - 2}$$

In the above Σ is the estimated common covariance matrix, Σ_i is the estimated covariance matrix for specific group i, whilst N_i is the number of data points in group i. This formula extends naturally for the case of k groups:

$$\Sigma = \frac{\sum_{i=1}^{k}(N_i - 1)\Sigma_i}{(\sum_{i=1}^{k} N_i) - k}$$

To request the covariance matrix in `R` for the case of the `salmon` data enter the following:

```
> alaskasalmon=salmon[c(1:40),c(2,3)]

> canadasalmon=salmon[c(51:90),c(2,3)]

> singlecov=(39/78)*(cov(alaskasalmon)+cov(canadasalmon))
```

Remember from lectures that the classification rule for LDA is as follows:

$$\log\left(\frac{P(k|\mathbf{x})}{P(l|\mathbf{x})}\right) = \log\left(\frac{\pi_k}{\pi_l}\right) + \log\left(\frac{f(\mathbf{x}|k)}{f(\mathbf{x}|l)}\right) = 0$$

The Multivariate Normal assumption with common covariance then leads to the following:

$$\log\left(\frac{f(\mathbf{x}|k)}{f(\mathbf{x}|l)}\right) = \mathbf{x}^T\Sigma^{-1}(\mu_k - \mu_l) - 1/2(\mu_k^T\Sigma^{-1}\mu_k - \mu_l^T\Sigma^{-1}\mu_l)$$

$$\Rightarrow \log\left(\frac{f(\mathbf{x}|k)}{f(\mathbf{x}|l)}\right) = (\mathbf{x} - 1/2(\mu_k + \mu_l))^T\Sigma^{-1}(\mu_k - \mu_l)$$

The term $1/2(\mu_k+\mu_l)$ gives the average of the group means, and so $\mathbf{x}-1/2(\mu_k+\mu_l)$ gives the difference of the observation to this value. Assuming the prior probabilities are equal, $(\mathbf{x} - 1/2(\mu_k + \mu_l))^T\Sigma^{-1}(\mu_k - \mu_l)$ determines the classification by whether it is positive or negative (if positive then group k).

Calling `lsol` in the `R` console also provides additional information, in particular:

```
Coefficients of linear discriminants:
                   LD1
Freshwater  0.04390178
Marine     -0.01806237
```

These are a (scaled) version of $\Sigma^{-1}(\mu_k - \mu_l)$, and hence can be used for classifying a new observation (though you need to note that it is the second class, here `Canada`, that is associated with group k).

Task: Determine the classification for an observation with a Freshwater recording of 120 and a Marine recording of 380.

```
>
```

To automatically predict such an observation enter the following:

```
> predict(lsol,c(120,380))
```

To automatically predict the test data enter the following:

```
> predict(lsol,stest[,c(2,3)])
```

Remember that when $\pi_k \neq \pi_l$, the term $\log(\pi_k/\pi_l)$ no longer disappears (though we can still use the `predict` function).

Of course we can always classify by hand by assigning to group k that maximises:

$$\log(\pi_k) + (\mathbf{x} - 1/2(\mu_k))^T\Sigma^{-1}(\mu_k)$$

2 Cross-Validation

Rather than splitting the data into a training and testing split (or a training, validation, and testing split for the case that different models are being considered, *e.g.*, if both LDA and QDA are being considered), an alternative technique for measuring the performance of the model is to ask `R` to perform cross-validation. For the `lda` function this is achieved by incorporating the argument `CV=TRUE`:

```
> lsolcv=lda(salmon[,c(2,3)],grouping=salmon[,1],CV=TRUE)
```

You should now find that the output from entering `lsolcv` includes a list of how the data points were classified when they were the point left out. Additionally a vector of group membership probability is returned.

In order to visualise the performance of LDA under cross-validation we can request a plot of the following form:

```
> plot(salmon[,c(2,3)],col=as.factor(salmon[,1]),pch=as.numeric(lsolcv$class))
```

The above command plots the two numeric variables of the `salmon` data with colouring being determined by true classification and symbols being determined by the resulting classification of 'leave-one-out' LDA.

3 Quadratic Discriminant Analysis

The function `qda` within the package `MASS` performs Quadratic Discriminant Analysis within `R`. Remember the difference between QDA and LDA is that the former permits each group distribution to have its own covariance matrix, whilst the latter assumes a common covariance matrix for all group distributions. The usage of `qda` within `R` is just the same as the usage of `lda`, *e.g.*:

```
> qsol=qda(strain[,c(2,3)],grouping=strain[,1])

> predict(qsol,stest[,c(2,3)])
```

Again you will notice in an 80:20 training:testing split we have achieved 100% correct classification. The output returned from now entering `qsol` provides details of the prior probability of group membership (again determined by the proportion of data points classified in that group by default), and the mean vectors for each group. To find the covariances for the two groups enter the following:

```
> alaskasalmon=salmon[c(1:40),c(2,3)]

> cov(alaskasalmon)

> canadasalmon=salmon[c(51:90),c(2,3)]

> cov(canadasalmon)
```

Task: Assess the performance of QDA for the `salmon` data set under cross-validation and produce a plot of your results:

```
>

>
```

Task: Assess the performance of the best of LDA and QDA under a 50:25:25 training:validation:testing split of the `salmon` data (remember that this means you should use 50% of the data to train the models, 25% of the data to assess which trained model appears to be the better classifier, and a further 25% of the data to more accurately assess the true classification rate of the better model):

```
>

>

>

>
```

Exercise: Assess the performance of LDA and QDA for classifying the `iris` data by species.

Exercise: Assess the performance of LDA and QDA for classifying the `olive` data by either region of origin, or specific area of origin.

Finally, to do something a little more fancy using the `ellipse` function in package `ellipse` (can you understand what it is doing?):

```
> plot(salmon[,c(2,3)],col=as.factor(salmon[,1]),xlim=c(50,190),
+ ylim=c(290,530))

> lines(ellipse(cov(salmon[c(1:50),c(2,3)]),
+ centre=mean(salmon[c(1:50),c(2,3)]),level=0.5))

> lines(ellipse(cov(salmon[c(51:100),c(2,3)]),
+ centre=mean(salmon[c(51:100),c(2,3)]),level=0.5),col=2)
```