MS1b: SDM - Practical Sheet 2

Parametric Classification

In this assignment we look at parametric classification methods. Load the workspace `PracticalObjects` using something like

```
load(url("http://www.stats.ox.ac.uk/%7Eteh/teaching/datamining/PracticalObjects.RData"))
```

This file contains functions and datasets that we will need use in these classes. Alternatively, download from the website and open locally with `load`. Where packages cannot be loaded, type `install.packages()` and follow the instructions to install a package.

Vector Quantization

We experiment with vector quantization ideas for a .png picture file.

Download it from `http://www.stats.ox.ac.uk/%7Eteh/teaching/datamining/Lenna.png`.

```
library(png)
library(matlab) #for the reshape() function
lenna <- readPNG("Lenna.png")
eqscplot(c(0, 512), c(0, 512), type = "n", xlab="", ylab="")
rasterImage(lenna,0,0,512,512)
```

This is a 512x512 pixel colour image of a model Lenna. It is a famous image used in the image compression/image processing communities. We turn the image into a data matrix, and apply K-means.

```
v <- dim(lenna)
X <- reshape(lenna,v[1]*v[2],v[3])
## cluster color vectors into 10 groups
km <- kmeans(X,centers=10, iter.max = 100, nstart = 10)
```

Examine how these clusters we found distributed by looking at the data in PC space.

```
i <- sort(sample(1:dim(X)[1],1000))
lenna.pc <- princomp(X[i,])
plot(lenna.pc$scores[,1:2],col=km$cluster[i],pch=km$cluster[i])
zeroX <- (km$centers-(1/dim(X)[1])*matrix(1,10,1)%*%matrix(1,1,dim(X)[1])%*%X)
points(zeroX%*%lenna.pc$loadings,pch="*",cex=3)
```

Using these cluster centres, we apply vector quantization ideas to compress the image.

```
Y <- km$centers[km$cluster,]
y <- reshape(Y,v[1],v[2],v[3])
eqscplot(c(0, 512), c(0, 512), type = "n", xlab="", ylab="")
rasterImage(y,0,0,512,512)
```

The compression rate is given by

```
(object.size(km$centers)+object.size(km$cluster))/object.size(X)
```

Linear and Quadratic Discriminant Analysis: Cushing's Syndrome Data

In this section we consider the Cushing's syndrome data, which is available in the package `MASS` and use the functions `lda()` and `qda()` to perform linear and quadratic discriminant analysis, respectively. These two functions are also available in the package `MASS`.

```
library(MASS)
Cushings
```

1. Note that there are three types of Cushing's syndrome distinguished coded as 'a', 'b', and 'c'. We remove the observations of type 'u' (unknown).

```
cush <- Cushings[Cushings$Type!="u",]
cush[,3] <- as.factor(as.character(cush[,3]))
cush.type <- cush[,3]
```

The vector `cush.type` indicates the syndrome type of each observation in the reconstructed dataset `cush`. The second line gets rid of the level `u` in the vector (in a dirty way).

The log transform of the continuous variables ensure that their distributions are more symmetrical.

```
boxplot(cush[,1:2])
cush[,1] <- log(cush[,1])
cush[,2] <- log(cush[,2])
boxplot(cush[,1:2])
```

2. First, we perform a linear discriminant analysis on the Cushing's data. The function `lda()` accepts the continuous variables as its first argument and the class labels as its second argument.

   ```
   cush.lda <- lda(cush[,1:2], cush.type)
   cush.lda
   ```

   The command `cush.lda` displays some of the discriminant analysis results. Note that the linear discriminants, a_1 and a_2, are displayed.

   We use the method `predict()` to project observations onto the linear discriminants. The argument `dimen` specifies the number of discriminant components onto which the data will be projected.

   ```
   cush.pred <- predict(cush.lda, dimen=2)
   ```

   After the object `cush.pred` is created we need to extract the scores (projected observations) on the first `dimen` discriminant variables.

   ```
   cush.pr <- cush.pred$x
   ```

   Let us look at the data projected onto the first two discriminant components.

   ```
   eqscplot(cush.pr, type="n", xlab="first linear discriminant",
       ylab="second linear discriminant", main="LDA")
   text(cush.pr, labels=as.character(cush.type), cex=0.8)
   ```

3. We can create a plot that indicates the linear discriminant analysis decision boundaries of the dataset `cush[,1:2]`. The function `dec.bound.plot()`, in the workspace `PracticalObjects`, is used to create such a plot. This function only accepts datasets with two continuous variables and one categorical (or nominal) variable that indicate the group or class of each observation. The first argument of this function is the object created by using `lda()` (or `qda()`), and the second and third arguments are the data (continuous variables) and the class labels, respectively.

   ```
   dec.bound.plot(cush.lda, cush[,1:2], cush.type,
       "Tetrahydrocortisone", "Pregnanetriol", "LDA")
   ```

   Examining the plot, it is clear that the fourth, fifth, and sixth arguments of `dec.bound.plot` are the x axis label, y axis label, and the title of the plot, respectively.

4. The function `qda()` can be used to perform a quadratic discriminant analysis and `dec.bound.plot()` enables us to visualise the quadratic decision boundaries.

   ```
   cush.qda <- qda(cush[,1:2], cush.type)
   dec.bound.plot(cush.qda, cush[,1:2], cush.type,
       "Tetrahydrocortisone", "Pregnanetriol", "QDA")
   ```

5. The function `visu.lqda()`, which is also in the workspace `PracticalObjects`, can be used to display the estimated Gaussian distributions for each class in addition to the linear or quadratic decision boundaries. For LDA the orientation and spread of the estimated densities for each class are similar.

   ```
   library(mvtnorm)
   visu.lqda(cush.lda, cush)
   ```

   Compare this to the spread and orientation of the estimated Gaussian distributions for QDA. Are there any differences?

   ```
   visu.lqda(cush.qda, cush)
   ```

Note that the second argument of the function `visu.lqda()` is the full dataset (continuous variables and grouping factor).

6. The reduced rank LDA does not use all projection directions. For the Cushing's data we can use a maximum of two projection directions. The function `visu.rr1.lda()` can be used to fit a reduced rank LDA with rank 1 to the `cush` dataset, and to visualise the decision boundaries.

   ```
   visu.rr1.lda(cush.lda, cush)
   ```

   The solid black lines indicate the decision boundaries, while the coloured dashed lines indicate the projection of the data points onto one dimension.

Linear and Quadratic Discriminant Analysis: Vanveer Data

We consider the breast tumour data again and perform discriminant analyses on different subsets of the genes. For linear discriminant analysis we will use smaller subsets of the genes to ensure that the estimated covariance matrices are of full rank.

```
vanv.10 <- vanveer.4000[,1:11]
vanv.20 <- vanveer.4000[,1:21]
vanv.prog <- vanveer.4000[,1]
```

1. In this section we will look more closely at the proportion of observations that are accurately classified when we used LDA. We perform an LDA on the dataset containing the subset of 10 "best" genes. As above we use the function `predict()` to extract information about the projection of the data onto the linear discriminant components.

   ```
   vanv.lda.10 <- lda(vanv.10[,2:10],vanv.prog)
   vanv.pred.10 <- predict(vanv.lda.10)
   ```

   Calculate the proportion of patients that are accurately classified. The function `table()` counts the number of patients with prognosis "poor" that are classified as "good" or "poor", and the number of patients with prognosis "good" that are classified as "good" or "poor".

   ```
   table(vanv.prog, vanv.pred.10$class)
   ```

   How many patients are misclassified?

   We can adjust the above commands to see how many patients are misclassified when we use the subset of 20 "best" genes.

   ```
   vanv.lda.20 <- lda(vanv.20[,2:21],vanv.prog)
   vanv.pred.20 <- predict(vanv.lda.20)
   table(vanv.prog, vanv.pred.20$class)
   ```

2. Note that we used the same data to fit and assess the classification performance. A more accurate estimate of the misclassification rate can be obtained by dividing the dataset into a training set and a test set. We randomly select patients for the training set (roughly 60% of the patients), and the remaining patients constitute the test set.

   ```
   train <- runif(nrow(vanv.10))<0.60
   test <- !train
   ```

   The vectors `train` and `test` are indicator vectors—they indicate which observations in the dataset `vanv.10` belong to the training set, and which belong to the test set.

   We perform a linear discriminant analysis on the data in the training set. However, we predict the data in the test set on the linear discriminant components. To calculate the number of misclassifications in the test set we compare the predicted classes of the observations in the test set with their true classes.

   ```
   vanv.lda.10 <- lda(vanv.10[train,2:11],vanv.prog[train])
   vanv.pred.10 <- predict(vanv.lda.10, vanv.10[test,2:11])
   table(vanv.prog[test], vanv.pred.10$class)
   ```

   How many patients in the test set are misclassified, for the subset of 10 "best" genes?

Exercises

1. *LDA and QDA*
   Adjust the commands given in point 2 (LDA and QDA: Vanveer Data section) and calculate the proportion of misclassifications in the test set for the subsets of 15 and 20 "best" genes. Compare these proportions to the proportion of misclassifications calculated at point 1 when we used the whole dataset to perform an LDA and to assess its performance. Do you notice any differences?

2. We can use the same training set and test set as defined above (point 2) and assess the performance of quadratic discriminant analysis. The quadratic decision boundaries are more flexible than the linear decision boundaries (as seen in the plots constructed for the Cushing's data). This can create the expectation that QDA will lead to a lower rate of misclassification.

   Adjust the above code to compute the proportion of misclassified patients for the datasets containing the subsets of 10 and 15 "best" genes, respectively. Compare these proportions to the corresponding proportions of misclassifications obtained by using LDA. Does QDA perform better than LDA, when we compare the error rates? Dinstinguish between error rates on training and test sets. Is the proportion of misclassifications high overall?