MS1b: SDM - Practical Sheet 1

Dimensionality Reduction and Clustering

In this practical we look at various exploratory data analysis (EDA), visualization, dimensionality reduction and clustering methods. Load the workspace `PracticalObjects` using something like

```
load(url("http://www.stats.ox.ac.uk/~teh/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.

Visualisation

The crabs data is available in the R package `MASS`.

1. Load the package `MASS` and examine the crabs dataset.

   ```
   library(MASS)
   crabs
   ```

   Use `summary()` and `str()` to see which variables are nominal. What are the levels of each of these variables? Identify the continuous variables in the dataset?

2. Lets look at the pairwise relationships between the continuous variables in the crabs dataset.

   ```
   pairs(crabs[,4:8])
   ```

   Note the command `crabs[,4:8]` is used to extract columns 4 through to 8 of the dataset. Are the pairwise linear associations between the variables strong or weak?

3. It is helpful to distinguish the species and sex of each observation in the plot. Observations in the crabs dataset can be categorised into four groups: 'Blue male', 'Blue female', 'Orange Male', and 'Orange female'. We create a vector `crabs.grp` to indicate the group that each observation belongs to.

   ```
   crabs.grp <- rep(1:4, each = 50)
   ```

   Various graphical parameters can be modified to change the plotting symbol `pch` and the colour of plotted points `col`.

   ```
   pairs(crabs[, 4:8], col=crabs.grp, pch=crabs.grp)
   ```

   To get information on any function, use the command `?pairs` or `help.start()`. For the latter, click on `Packages` and then choose the appropriate package and function.

4. The parallel coordinate plot can also be used to display the crabs data.

   ```
   parcoord(crabs[,4:8], col=crabs.grp)
   ```

   Is this visualisation method useful for the crabs data?

Principal Component Analysis

The function `princomp()` is used to perform principal components analysis and is found in `MASS`. We perform PCA on the crabs data but note that PCs depend on the scaling of the original variables.

1. Continuous variables in the crabs dataset have the same measurement unit (mm) but we inspect anyhow to what extent the scaling differs.

   ```
   boxplot(crabs[,4:8])
   ```

   The boxplots reveal that rescale may be necessary. A log transformation achieves this scaling the variables to fall roughly within the interval $[2, 4]$.

   ```
   boxplot(log(crabs[,4:8]))
   ```

2. A principal component analysis is performed on the transformed data, `lcrabs` with `princomp()`. The PCA objects have methods `loadings` and `predict` that extract the loadings and principal component score components of the object.

   ```
   lcrabs <- log(crabs[, 4:8])
   crabs.pca <- princomp(lcrabs)
   loadings(crabs.pca)
   crabs.pc <- predict(crabs.pca)
   crabs.pc[c(1:5,196:200),]
   ```

   The (overloaded) function `summary()` can be used to see what proportion of the total variation is explained by the first principal component.

   ```
   summary(crabs.pca)
   ```

3. The `lcrabs` dataset consists of $p = 5$ variables and $n = 200$ observations. As discussed in lectures, the PC scores give the projection of each of the original observations onto the principal components.

   ```
   library(MASS)
   eqscplot(crabs.pc[,1], crabs.pc[,2], xlab="1st principal component",
                                ylab="2nd principal component")
   ```

   Similar to the previous section, we can use pairs() to plot more than two variables at a time. Lets look at the first three PCs.

   ```
   pairs(crabs.pc[, 1:3], col=crabs.grp)
   ```

   Do the first three PCs give information about the known structure in the data?

Multidimensional Scaling

Consider the dataset `vanveer.4000` which contains microarray data on breast tumour. It contains 76 patients: 44 good and 32 poor. `outcome` indicates the prognosis group of a patient, furthermore, the dataset only contains the 4000 'best' genes of the original 24 189 column dataset. Lets apply MDS on this dataset on a smaller subset of this dataset, the " best" 20 genes say (correspond to the first 21 columns of `vanveer.4000`).

```
vanv.20 <- vanveer.4000[,2:21]
vanv.progn <- vanveer.4000$outcome
```

1. *Classical MDS*
   `cmdscale()` performs classical multidimensional scaling on `vanv.20` allowing us to view this 76×20 dataset. `dist()` calculates a distance matrix D with Euclidean distance between all pairs.

   ```
   vanv.dist <- dist(vanv.20)
   ```

2. Look at `?cmdscale`. Taking a distance matrix as its first argument, it calculates a set of points $z_1, \ldots, z_{76}$ in $\mathbb{R}^k$ such that the distances between the points best match the distances in `vanv.dist`, the argument `k` specifies the dimension of the reconstructed space.

   ```
   vanv.clas <- cmdscale(vanv.dist, k=2)
   ```

   The object `vanv.clas` becomes a matrix with 2 columns whose rows give the coordinates of the reconstructed points. By plotting an 'empty plot' with `eqscplot()` using `type=''n''` and by overlaying text with `text`, we get ourselves a nice plot.

   ```
   windows()  ## use command X11() on Linux/Unix/Mac and quartz() on a Mac
   eqscplot(vanv.clas, type="n", main=" Metric Scaling: 20 best genes")
   text(vanv.clas, labels=as.character(vanv.progn),
       col=1+as.numeric(vanv.progn), cex=0.8)
   ```

3. Lets compute the stress for this MDS representation, `dist()` helps us compute the distance matrix of the reconstructed points `vanv.clas`.

   ```
   vanv.clas.dist <- dist(vanv.clas)
   classical.stress <- sum( (vanv.dist - vanv.clas.dist)^2 )/sum((vanv.dist)^2)
   ```

4. *Sammon's non-linear mapping*
   We can examine the data with different stress functions. The function `sammon()` takes a distance object its the first argument and finds a two-dimensional configuration to minimise the Sammon stress function returning an object with two components: a vector of the fitted configuration `points` and the final stress achieved (`stress`).

   ```
   vanv.sam <- sammon(vanv.dist)
   names(vanv.sam)
   vanv.sam$points
   vanv.sam$stress
   ```

   We plot the representation found to compare it to that found with classical MDS stress.

   ```
   windows() ## use command X11() on Linux/Unix/Mac and quartz() on a Mac
   eqscplot(vanv.sam$points, type="n", main="Sammon Mapping: 20 best genes")
   text(vanv.sam$points, labels=as.character(vanv.progn),
       col=1+as.numeric(vanv.progn), cex=0.8)
   ```

5. *Kruskal's non-metric multidimensional scaling*
   And finally, we implement Kruskal's non-metric multidimensional scaling with `isoMDS()`.

   ```
   vanv.iso <- isoMDS(vanv.dist)
   vanv.iso$stress


   windows() ## use command X11() on Linux/Unix/Mac and quartz() on a Mac
   eqscplot(vanv.iso$points, type="n", main="Kruskal's MDS: 20 best genes")
   text(vanv.iso$points, labels=as.character(vanv.progn),
       col=1+as.numeric(vanv.progn), cex=0.8)
   ```

   How do the three configurations obtained above differ? The plots obtained using classical MDS and Kruskal's non-metric MDS are rather similar. It seems like there is a reasonable amount of overlap between the good and poor prognosis groups.

K-means

Lets look at K-means clustering. Remembering that K (the number of clusters) must be pre-specified, clusters are chosen to minimise the within-class sums of squares from cluster centres. As K-means is based on Euclidean distances, scaling of variables is important, `scale` achieves this.

```
ft.stand <- scale(ft[,1:21])
```

1. `kmeans()` performs K-means clustering and has argument `centers` which allows us to either specify the number of clusters (or to initialise cluster centres, they are random if not specified). The function returns a matrix storing the cluster centres and a vector indicating the class label of the corresponding observations in the data matrix. The dataset `ft` contains a league table comparing the performance of several UK universities.

   ```
   ft.km <- kmeans(ft.stand,4)
   ft.km$centers
   ft.km$cluster
   ```

2. As the dataset contains 22 variables, we'll need to reduce the dimensionality of the data to view the results of K-means clustering. We'll plot the data onto its first two PC.

   ```
   library(MASS)
   ft.label <- dimnames(ft.stand)[[1]]
   ft.pca <- princomp(ft.stand)
   ft.pc <- predict(ft.pca)
   eqscplot(ft.pc[,1:2], type="n", xlab="first principal component",
                              ylab="second principal component")
   text(ft.pc[,1:2], labels=ft.label)
   ```

   `dimnames()` is used to retrieve rows names in the dataset which are needed to add informative label to the plot. How accurate is the above two-dimensional presentation of the dataset?

   Adjusting the plot, we indicate the cluster/group of the observations.

   ```
   ft.centers <- predict(ft.pca, ft.km$centers)
   eqscplot(ft.pc[,1:2], type="n", xlab="first principal component",
                           ylab="second principal component")
   text(ft.pc[,1:2], col=ft.km$cluster, labels=ft.label)
   points(ft.centers[,1:2], pch=3, cex=2)
   ```

   Here we first predict the cluster centres `ft.km$centers` onto the first two PC, `points()` then added crosses to the plot to indicate the centres. It's also possible to create a similar plot where names of the observations are only made available by clicking observation on the graph using the function `identify()`.

   ```
   eqscplot(ft.pc[,1:2], type="n", xlab="first principal component",
                           ylab="second principal component")
   points(ft.pc[,1:2], col=ft.km$cluster)
   points(ft.centers[,1:2], pch=3, cex=2)
   identify(ft.pc[,1:2], cex=0.4, labels=ft.label)
   ```

3. `visu.kmeans()` allows us to visualise each iteration of the K-means algorithm. This function requires the package `deldir` to run. We start off by simulating a dataset containing data points from four multivariate distributions with `rmvnorm()` from the package `mvtnrom`.

   ```
   library(mvtnorm)
   data.1 <- rmvnorm(25,c(-1,2), matrix(c(0.5,0.1,0.,0.5),nrow=2))
   data.2 <- rmvnorm(25,c(0.5,1),matrix(c(0.5,0.15,0.15,0.5),nrow=2))
   ```

```
data.3 <- rmvnorm(25,c(0,-1.2),matrix(c(0.5,-0.12,-0.12,0.5),nrow=2))
data.4 <- rmvnorm(25,c(-2,0),matrix(c(0.5,0,0,0.5),nrow=2))
simu.data <- rbind(data.1, data.2, data.3, data.4)
```

Though these are unclassified data, we know the distribution from which they came. We store these in `simu.labels` for later reference.

```
simu.labels <- rep(1:4,each=25)
plot(simu.data,pch=simu.labels)
```

```
library(deldir)
simu.stand <- scale(simu.data)
visu.kmeans(simu.data,k=4, iter.max=20)
```

Repeat the above K-means algorithm a few times and compare the resulting clustering.

Hierarchical Clustering Methods

We turn our attention to hierarchical clustering methods.

1. The hierarchical clustering methods discussed in the lectures depend on measures of similarity or dissimilarity so data should be standardised first. We use the scaled `ft.stand` data from before

2. To implement agglomerative hierarchical clustering, the function `agnes()` in the package `cluster` is used (you can alternatively use function `hclust`. The argument `method` specifies the linkage method used allowing `"average"` (group average method), `"single"` (single link) or `"complete"` (complete link).

   ```
   library(cluster)
   agn.ave <- agnes(ft.stand,method="average")
   plot(agn.ave)
   agn.sin <- agnes(ft.stand,method="single")
   plot(agn.sin)
   agn.com <- agnes(ft.stand,method="complete")
   plot(agn.com)
   ```

   The dendrograms indicate a small number of universities that are highly dissimilar to the others. It is worthwhile investigating the difference between these universities with respect to the others. Note that the height axis allows us to assess dissimilarity between two branches by reading off the height value at the level where two branches join.

3. The function `diana` in the cluster package can be used to implement divisive hierarchical clustering. Here looking at data about research activities and other charactersitic features of several UK universities.

   ```
   ft.dia <- diana(ft.stand)
   plot(ft.dia)
   ```

   Compare the dendrogram with those resulting from agglomerative clustering.

Exercises

1. *PCA*
   We consider the gene expression data discussed in lectures. The dataset `Cho.dat` contains 384 observations (genes) where for each gene, measurements are taken at 17 time points. The vector `Chodat.phases` contains the phases corresponding to each observation. We normalise the data

to have zero mean and unit variance using the function `scale()` and verify whether `gene.dat` has been normalised appropriately.

```
gene.dat <- Cho.dat
gene.norm <- scale(gene.dat)
apply(gene.norm,2,mean)
diag(var(gene.norm))
```

Note that if we set the argument `cor` of the function `princomp()` to `TRUE`, we do not need to normalise the data in advance.

Perform PCA on the `gene.norm` dataset and plot the first two principal components against each other using different symbols for each of the classes found in `Chodat.phases`. Use a scatterplot matrix to visualise relationships between the first four principal components. Which PCs are able to separate the classes reasonably?

What proportion of the total variation is explained by the first two principal components? Compare this proportion to the proportion of the variation explained for the crabs dataset.

2. *PCA*
The `Virus` dataset contains the measurements on 39 Tobamoviruses which we want to investigate. We are interested in whether subgroups among the viruses can be distinguished. As virus 7 and 20 have identical scores, we remove the virus 7.

```
virus.unq <- rbind(Virus[1:6,],Virus[8:39,])
```

The function `rbind()` takes two separate parts of the Virus dataset and combines them by rows. To get an idea of the distributions of the variables, a boxplot can be constructed for each variable. Should we scale the data before further analysis?

```
boxplot(virus.unq[,1:18])
virus.norm <- scale(virus.unq[,1:18])
```

Perform PCA on the normalised data and the original data to examine the structure of the dataset. Use the scatterplot matrix of the first few principal components to see whether we can detect structure in the dataset.

3. *MDS*
We consider the breast tumour dataset `vanveer.4000` again. Lets create datasets that contain the set of 50 and 100 'best' genes,

```
vanv.50  <- vanveer.4000[,2:51]
vanv.100 <- vanveer.4000[,2:101]
vanv.progn <- vanveer.4000$outcome
```

Use classical MDS, Sammon's non-linear mapping and Kruskal's MDS to create two dimensional presentations of the data and record the stress for each of these methods. Compare graphical presentations for the data containing the subset of 50 best genes and the subset of 100 best genes.

4. *K-means*
The choice of starting values (initial cluster centres) for K-means significantly influence the clustering results. Lets verify this on the `ft` dataset by randomly choosing four observations from the dataset as starting values.

```
random.ind1 <- sample(1:nrow(ft.stand), 4)
random.start1 <- ft.stand[random.ind1,]
ft.km <- kmeans(ft.stand,centers=random.start1)
```

Create a plot to indicate the clustering of the observations as well as the positions of the cluster centres using PCA. Using different starting values, create the corresponding plots that indicates the clustering of the data. Do different starting values lead to substantially different clustering results?