Modern Statistics and Big Data Analysis, Exercises 6

Deadline: End of Friday 24 November.

1. (3.5 points) On the Virtuale page of the course you find the dataset `stars5000.dat`. You can read it using

   ```
   stars5000 <- read.table("stars5000.dat",header=TRUE)
   ```

   The data contain information about 5000 celestial objects (most of these are stars, but objects also include quasars, distant galaxies, and a certain amount of optical noise) of which spectra were obtained by mounting a prism in front of a telescope. The aim is to find classes of celestial objects with specific characteristics. There are six variables containing summary information of the spectra (original spectra contain 300 highly dependent variables; the original database contains about 4 million objects). These are

   casn Signal-to-noise ratio of the Calcium break (which is a characteristic discontinuity in spectra of galaxies)

   cacont Contrast of the Calcium break to smoothed version of the spectrum

   kl1 First principal component of smoothed spectrum

   kl2 Second principal component of smoothed spectrum

   xh1 "Half power point" in upper spectrum

   xh2 "Half power point" in lower spectrum (these are indicators of where the spectral density is concentrated)

   Obviously you are not expected to understand the meaning of these in any detail. What is interesting regarding applied statistics is that the astronomers, with some help from statisticians, used a judicious mix of subject-matter knowledge (knowing what kind of information in the spectra is important) and statistical techniques (such as principal component analysis and kernel density smoothing) in order to reduce the highly redundant and noisy high dimensional information in the original spectra. Such an approach is often better than relying on the data (or the scientists' knowledge) alone.

   Cluster these data using Gaussian mixtures, t-mixtures, skew-normal mixtures, and skew-t mixtures (using `smsn.search` you will have to decide what number of degrees of freedom ν to use for the t-distributions), and decide which clustering you find most convincing, with reasons. Although methods with flexible covariance/shape matrices can in principle handle variables with very different variances, value ranges here are vastly different, and standardisation may help, maybe in a robust manner (using median and MAD) because of the presence of outliers. If you have time, you can compare how much of a difference that makes.

   Hint: A pairs plot of 5000 observations will work better with smaller plot symbols, for example `pch=20,cex=0.1`.

2. (2 points) A very simple strategy for applying a mixture to a big data set (also mentioned in Fraley et al. 2005 as cited on the course slides) is the following:

   Step 1 Draw a random data subset of n_s observations (Fraley et al. recommend $n_s = 2000$).

   Step 2 Compute the mixture ML estimators using the EM-algorithm on that subset.

   Step 3 Extend the fitted model to all other observations by computing the estimated posterior probabilities p_{ik} for observation $\mathbf{x}_i$ to belong to cluster k (these can be used as usual by maximisation to find a clustering).

   For Gaussian mixtures, this can be done using `Mclust` in the following way:

   Step 1 as above.

   Step 2 Run `Mclust` on the subset.

   Step 3 Use function `predict.mclust` to extend the fitted model to all observations (read the help page for how exactly to do that).

   Implement the big data method explained above, and apply it to the `stars5000` data from Exercise 1 above. Use $n_s = 1000$ (for a data set of size 5000, using $n_s = 2000$ will not win that much time). Take the time (this can be done using the `system.time` command in R). Also take the time for running `Mclust` on those data. Compare the running times and the results of the big data method and of standard `Mclust` (it would be a good result for the big data method if results are very similar, but the run time is much faster).

   You may want to (but don't have to) write a single function for the big data method that takes the data as input and optionally some other parameters such as n_s or `Mclust`-input parameters, so that you can use it easily in other situations.

3. (1.5 points) In a situation with 10 variables and 4 mixture components, what is the number of free parameters for

   (a) a "VVV" Gaussian mixture model assuming fully flexible covariance matrices,

   (b) a "VII" Gaussian mixture model assuming spherical covariance matrices with potentially differing volumes,

   (c) a fully flexible skew-normal mixture,

   (d) a fully flexible mixture of multivariate t distributions,

   (e) a mixture of skew-t distributions where skewness parameters, degrees of freedom and Σ-matrices are assumed equal in all mixture components?

4. (3 points) Here is some R-code that will generate an artificial data set with 1000 observations from the illustrative model presented on p.273-278 of the course slides:

```
prodcat <- c("PR","MB","AB","N") # product categories
eventcat <- c("S","M","C","P","N") # event categories
```

```
prodp <- eventp <- list()

# Category probabilities within the two mixture components (clusters)
prodp[[1]] <- c(0.4,0.4,0.1,0.1)
prodp[[2]] <- c(0.2,0.1,0.4,0.3)
eventp[[1]] <- c(0.5,0.2,0.1,0.1,0.1)
eventp[[2]] <- c(0.1,0.1,0.1,0.3,0.4)

# The first 400 observations from component 1, then 600 from component 2:
n1 <- 400
n2 <- 600
n <- n1+n2

# Initialisation (provide a data frame of the correct type and size
# without already having the data):
consumers <- data.frame(prod=factor(c(prodcat,rep(NA,n-4)),levels=prodcat),
                  event=factor(c(eventcat,rep(NA,n-5)),levels=eventcat))

# Generation of the data; you may want to set a seed here.
consumers$prod[1:n1] <- sample(prodcat,n1,replace=TRUE,prob=prodp[[1]])
consumers$event[1:n1] <- sample(eventcat,n1,replace=TRUE,prob=eventp[[1]])
consumers$prod[(n1+1):n] <- sample(prodcat,n2,replace=TRUE,prob=prodp[[2]])
consumers$event[(n1+1):n] <- sample(eventcat,n2,replace=TRUE,prob=eventp[[2]])

# You can run table(consumers) or str(consumers) to see what you got.
```

Generate a `consumers`-data set in this way, and estimate a latent class mixture model using the `flexmixedruns`-function, including estimating the number of clusters (it is enough to go up to 5). Compare the results with the truth that you know about these data (this can concern the clustering vector, but also the estimated parameters).

Also compute the simple matching distance and cluster the data using any distance based clustering method (just one), and compare the clustering with the same number of clusters with the optimal clustering from `flexmixedruns`. You should find that Average Linkage with Simple Matching is somewhat problematic here. Why is this not a good method for these data?

See the Appendix below for visualisation ideas (although you are not asked to try these out).

Appendix

This is just for information for those interested; you are not expected to understand, learn, or use this (although obviously you can). My explanation will not be complete.

The plots on p.274-278 of the course slides have been produced by `ggplot`. These are based on frequency data; in order to use them for probabilities I have created a matrix with integer numbers (frequencies) that add up to 1000 and then divided the entries by 1000 in order to get the probabilities. It may be that this can be done more directly and elegantly but I don't know how.

The good thing is that the ggplot command should also apply to frequency data, i.e., the data set you generate for Ex. 6.4 above.

```
# This makes use of objects defined above:
library(ggplot2)

pi1 <- n1/n
pi2 <- n2/n

thetable <- c1table <- c2table <- matrix(NA,nrow=4,ncol=5)
for(i in 1:4)
  for(j in 1:5){
    thetable[i,j] <- pi1*prodp[[1]][i]*eventp[[1]][j]+
      pi2*prodp[[2]][i]*eventp[[2]][j]
    c1table[i,j] <- prodp[[1]][i]*eventp[[1]][j]
    c2table[i,j] <- prodp[[2]][i]*eventp[[2]][j]
  }
# thetable is the table for the overall distribution,
# c1table and c2table for the two mixture components/clusters

thetabled <- as.table(thetable*1000) # Transform probabilities to counts here.
attr(thetabled,"dimnames") <- attr(table(consumers),"dimnames")
# This is not needed if you want to visualise the data in comsumers
# directly. attr makes sure that the category names are in the right place.

thetabled <- as.data.frame(thetabled) # ggplot needs a data frame.

ggplot(thetabled)+geom_point(mapping = aes(x = prod, y = event, size = Freq),
    shape=15,show.legend=FALSE)+scale_size_area(max_size=40)+
    geom_text(aes(x = prod, y = event, label=Freq/1000),color="yellow",size=5)
# I won't document this in detail. Read help pages and/or play around to
# find out what is doing what.
# Anyway, if I'm not mistaken, this will work directly with as.data.frame(consumers)

# The same was done with c1table and c2table.
# If you have a clustering vector called cluster for example,
# you could have a within-cluster data object as
# consumers[cluster==1,], consumers[cluster==2,]
```

Another suitable visualisation method for such data is `mosaicplot`.