Exercise 5 Trashaj Alberto 1075402

alberto.trashaj

October 2023

1 Point 1

In the first point we were asked to cluster the stars5000 dataset provided in Virtuale.
As usual, let's see the descriptive statistics

```{r}
stars5000 <- read.table("~/Desktop/Universita /Unsupervised/stars5000.dat",head
summary(stars5000)
str(stars5000)
pairs(stars5000, pch=20, cex=0.1)
heatmap(cor(stars5000))
```

Let's fit a Gaussian mixture model on the standardized dataset: to standardized the dataset, as suggested, I used the MAD function and I took a subset of the original dataset with the sample function

```{r}
sampled_data <- stars5000[sample(nrow(stars5000), 1000, replace=F),]
mad_values <- numeric(ncol(sampled_data))

for (col in seq_along(sampled_data)) {
  mad_values[col] <- mad(sampled_data[, col])
}


standardized_data <- as.data.frame(lapply(
                            seq_along(sampled_data), function(col) {
  col_name <- colnames(sampled_data)[col]
  (sampled_data[, col] - median(sampled_data[, col]))
                                / mad_values[col]
}), col.names = colnames(sampled_data))
```

```
head(standardized_data)
```

Let's fit now the gaussian mixture model with the Mclust function

```{r}
set.seed(123)
sampled_data <- stars5000[sample(nrow(standardized_data),
                                 1000, replace=F),]
msampled <- Mclust(standardized_data, G = 1:10)
summary(msampled)
```

```
Gaussian finite mixture model fitted by EM algorithm
```

```
Mclust VVV (ellipsoidal, varying volume, shape, and orientation)
model with 7 components:


Clustering table:
  1   2   3   4   5   6   7
234 113 178 107 166  13 189
```

The results show that the BIC is maximised when the number of the components is 7.
Then with the smsn function we fitted a skew.t and a t mixture model on the subset dataset

```
set.seed(1234)

mix_mod_skew_t <- smsn.search(sampled_data, g.min = 3,
                              g.max = 5, nu = 5, family = "Skew.t")
mix_mod_t <- smsn.search(sampled_data, g.min = 1,
                              g.max = 3, nu = 3, family = "t")

mix_mod_skew_t$best.model
mix_mod_t$best.model
```

The results show that the optimal number of clusters is 4 for the skew.t distribution and 3 for the t distribution.

2 Point 2

In the second point I wrote a function that takes the entire dataset as input: it compute a subset and run an Gaussian mixture model on it. Then with predict.Mclust function I extended the fitted model to all the observations

```r
big_data_gmm <- function(data, ns = 2000) {

  subset_data <- data[sample(nrow(data), ns), ]

  model <- Mclust(subset_data, G = 1:10)

  predicted_clusters <- predict.Mclust(model, data = data)$classification

  return(predicted_clusters)
}

# Load stars5000 data from Exercise 1 (replace this with your actual data loadin
# stars5000 <- read.csv("path/to/stars5000.csv")

# Run big data method
system.time({
  big_data_results <- big_data_gmm(stars5000)
})

# Run Mclust on the entire data
system.time({
  mclust_results <- Mclust(stars5000, G = 1:10)$classification
})

# Compare the results
cat("Big Data Method Results:\n", table(big_data_results), "\n")
cat("Mclust Results:\n", table(mclust_results), "\n")
```

```
   user system elapsed
15.930 0.311 16.959

   user system elapsed
39.284 0.645 41.390
Big Data Method Results:
255 204 252 342 347 98 340 141 21
Mclust Results:
561 474 73 606 692 299 601 546 800 348
```

As we can see from the results, both model maximise the BIC with 10 components, although it's possible to see that the cluster proportion differs, the run time is way less with the big data method, rather than fitting all the observations in the Mclust function all in once.

3 Point 3

With 10 variables and 4 mixture components we have:
assuming that the number of weights is 3
(a) "VVV" Gaussian Mixture Model (Fully Flexible Covariance Matrices):
The total number of free parameters is $k \times \frac{(p(p+1))}{2} + 3 = 263$
Given by $k \times p$ means, $\frac{p \times (p+1)}{2}$ covariance matrix and $k-1$ mixture weights
(b)a "VII" Gaussian mixture model assuming spherical covariance matrices with potentially differing volumes:
The tot number of free parameters is $k \times p + k \times 1 + k - 1$ given respectivetly by the means, the spherical cov matrix and the free weights: 47

(c) a fully flexible skew-normal mixture:
The tot number of free parameters is $k \times p + k \times \frac{p(p+1)}{2} + k \times p + 3$: same argument for the means, the same covariance matrix as in the first point and last term is due to the λ vector: 303

(d) a fully flexible mixture of multivariate t distributions:
The tot number of free parameters is $k \times p + k \times \frac{p(p+1)}{2} + \nu + 3$: same argument for the mean and the variance as in the point (c), plus the degrees of freedom (4) and the weigths as usual: 267

(e) a mixture of skew-t distributions where skewness parameters, degrees of freedom and -matrices:
The tot number of free parameters is given by $k \times p + \frac{p \times (p+1)}{2} + \nu + 10 + 3$: here we have the means, a uniqe covariance matrix with 55 parameters, a unique value for the ν degrees of freedom, a vector of 10 parameters for the skewness and 3 weights: 109.

4 Point 4

In the last point, after implementing the artificial dataset as suggested in the problem we have used the flexmix function; then we computed the simple matching distance and used that distance in the hclust function to clusterize with the average method.
In order to compare the two clusters we used the table function to see how many different classifications has been done from the different methods.

```
    ```{r}
flexmix_result <- flexmixedruns(consumers, continuous = c(),
                                discrete = c(1,2), n.cluster = 1:5,
                                diagonal = FALSE, xvarsorted = FALSE)

# Get the optimal number of clusters based on BIC
optimal_clusters <- flexmix_result$optimalk
#install.packages("nomclust")
```
```

```
library(nomclust)
simple_match_dist <- sm(consumers)

avg_clust <- hclust(simple_match_dist, method ="average")
plot(avg_clust)

cutted_tree <- cutree(avg_clust, k=2)
table(cutted_tree, flexmix_result[["flexout"]][[2]]@cluster)
```

```
cutted_tree   1   2
           1   0 260
           2 434 306
```

We can see from the results that the two method differs since in the main diagonal we have just 306 observations clusterized in the same clusters.
Average Linkage may be problematic for these data because the simple matching distance is sensitive to the order of observations.