Homework 5

Federico Veronesi

2023-11-15

Point 1

Get data

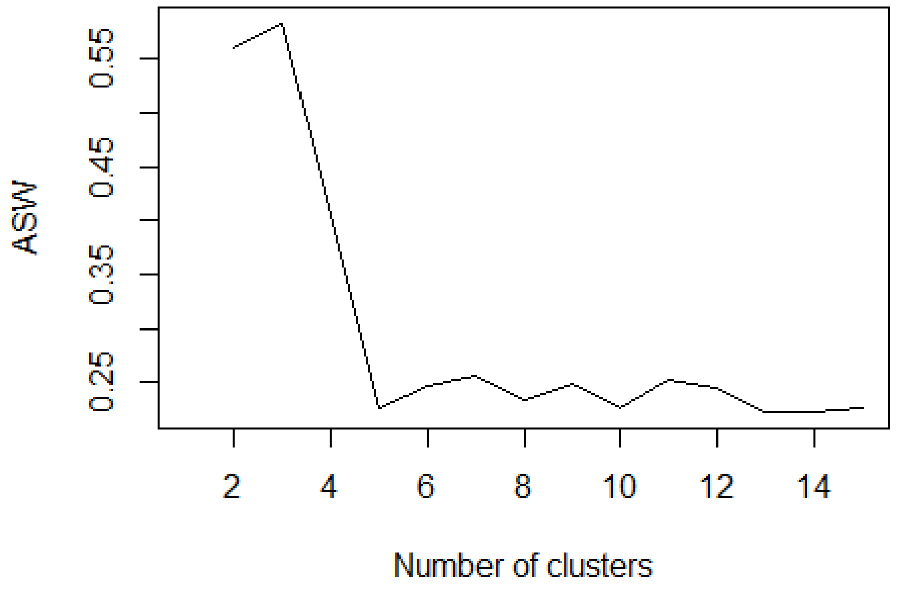
```
setwd("C:/Users/Veronesi/Desktop/uniBo/Magistrale/Modern Statistics and Big Data
Analytics/Datasets")
glass <- read.table("glass.dat",header=TRUE)
```

Method 1: Partitioning around medoids

Choice of k

Since we don't have any preliminar information, we will try PAM with K from 2 to 15 and decide through ASW what is our favourite n°of clusters.

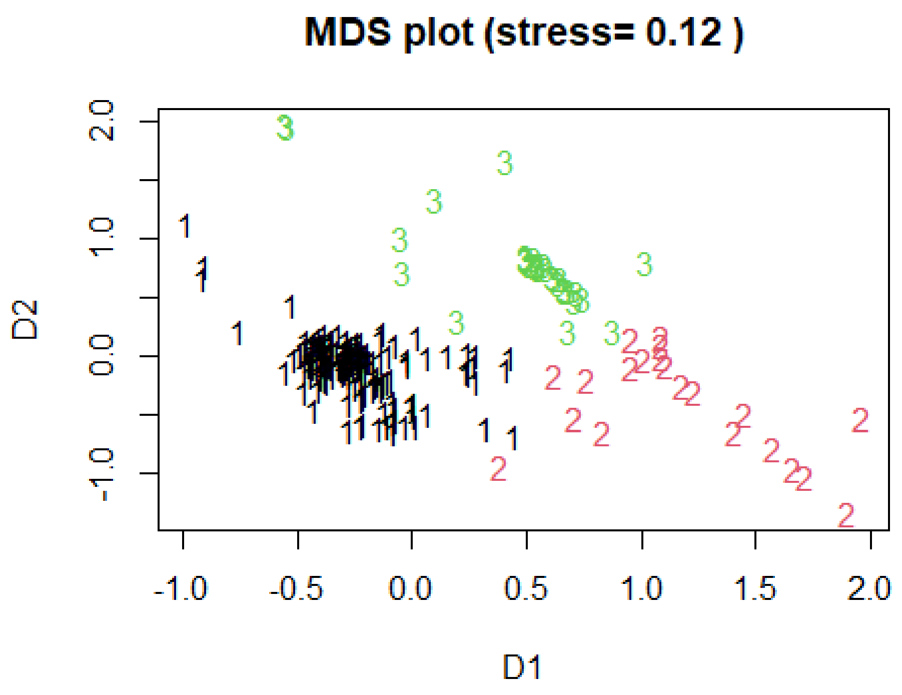
```
library(cluster)
distglass <- dist(glass, method="euclidean")
glass_pam <- list()
sil <- list()
asw <- c()
for (k in 2:15)
     {glass_pam[[k]] <- pam(distglass, k)
      sil[[k]] <- silhouette(glass_pam[[k]],dist=distglass)
      asw[k] <- summary(sil[[k]])$avg.width
     }

plot(1:15,asw,type="l",xlab="Number of clusters",ylab="ASW")
```



It seems that 3 is an appropriate n°of clusters. Local optimas are present when k is higher, but ASW is too low in those cases.

Visualization

```
#MDS
mdsglass <- mds(distglass, ndim=2)
# MDS-plot:
plot(mdsglass$conf,col=glass_pam[[3]]$cluster,pch=clusym[glass_pam[[3]]$cluster],
main=paste("MDS plot (stress=", round(mdsglass$stress, 2), ")"))
```



From the MDS plot visualization, we cannot be very happy about the performance of PAM. Cluster 1 is quite homogeneous but clusters 2 and 3 are sparse. Moreover, it seems that some points are outliers and we would struggle in assigning them to a cluster. Let's see if the gaussian mixture will solve some of these problems.

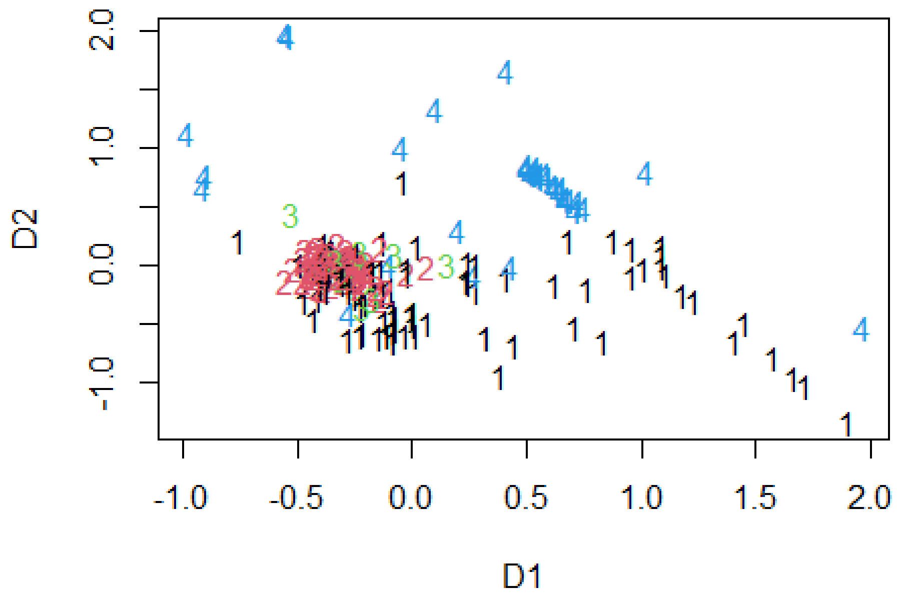
Method 2: Gaussian mixture

```
library(mclust)

## Package 'mclust' version 6.0.0
## Type 'citation("mclust")' for citing this R package in publications.

mixtglass <- Mclust(glass, G=2:15)
summary(mixtglass)

## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEV (ellipsoidal, equal shape) model with 4 components:
##
##  log-likelihood   n  df      BIC      ICL
##        2557.753 214 195 4069.141 4068.502
##
## Clustering table:
##  1  2  3  4
## 94 75  9 36

summary(mixtglass$BIC)
```

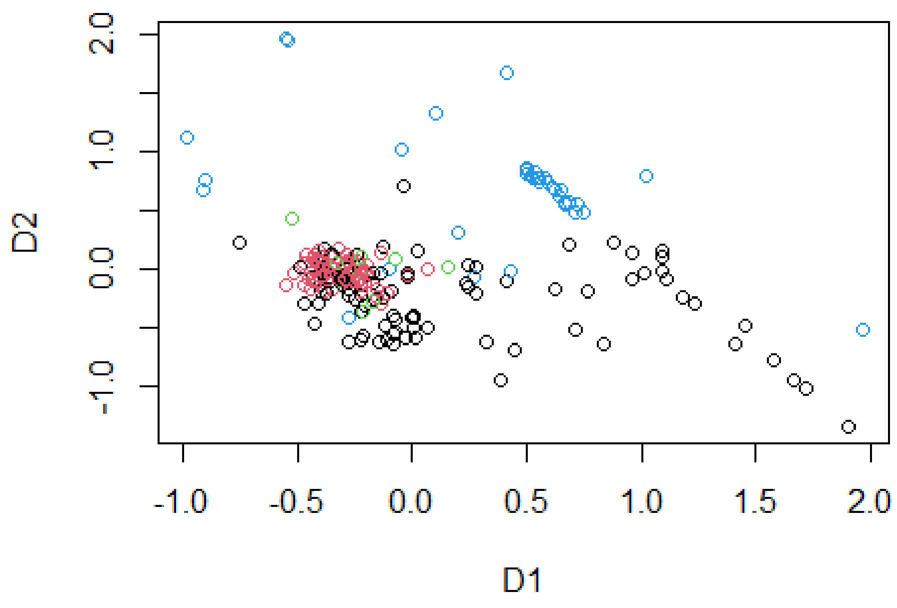
```
## Best BIC values:
##              VEV,4      VEV,3      VEV,2
## BIC      4069.141 4028.56162  3066.734
## BIC diff    0.000  -40.57902 -1002.407
```

VEV with 4 components is chosen.

```
plot(mdsglass$conf,col=mixtglass$classification,pch=clusym[mixtglass$classificatio
n])
```



```
plot(mdsglass$conf,col=mixtglass$classification)
```



Gaussian mixtures struggles in finding meaningful clusterings, and couldn't be considered a good method for these data. A potential problem is that, since dissimilarities between units are produced by different compositions of materials, we would want to find clusters that are quite well separated, while a Gaussian mixture is likely to find different concepts of clustering, since it's based on Gaussian density. As it is shown in the plot, here there's overlapping between clusters 1, 2 and 3 and that is not desirable in this case.

Other considerations:

Here I didn't apply any trasformation to the data. To be more precise, we have to say that the first variable (RI) has a different measurement from the other variables. But since all the values for RI are very similar for all the observations, using RI in euclidean distance is quite similar to remove the variable. To extend the analysis, we could have removed RI or could have tried to standardized the data (wr. to the mean only, or wr. to the mean and variance), and look for the new results.

Exercise 3

Question 3a

The aim of DBSCAN algorithm is to find clusters and "noise" points (i.e. points that don't belong to any cluster). Starting from a point, the algorithm discovers all the points which are density-reachable from that point with respect to the tuning parameters Eps and MinPts. These two quantities are often settled for the "thinnest" (i.e. least dense) cluster and used as global parameters.
Exploiting the definition of density-reachable, this algorithm find clusters associated with the concepts of neighboorhood and density of points. DBSCAN introduces also another useful

concept: the noise. When a point isn't density reachable to any other point it's classified as noise and not assigned to any cluster.

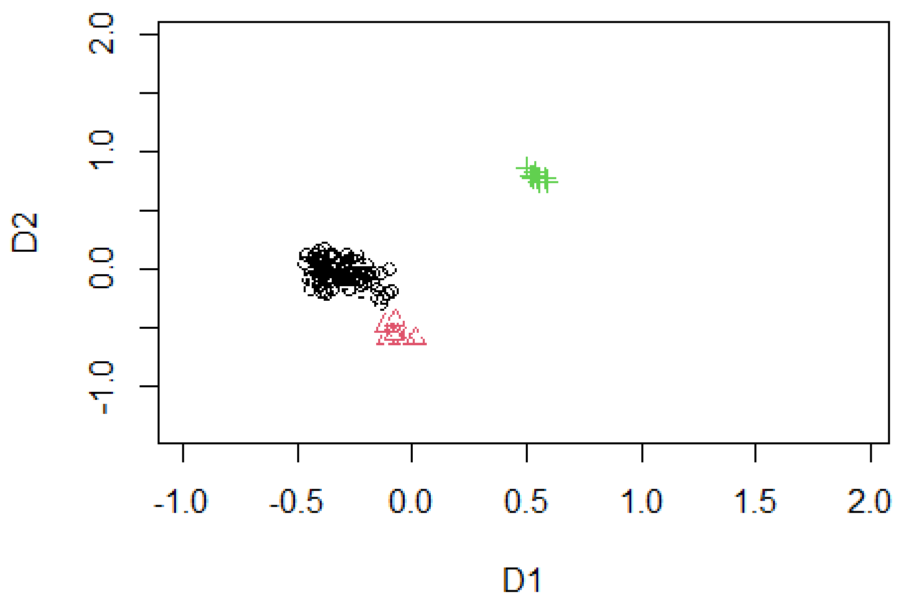
Question 3b

The principal advantages of DBSCAN over CLARANS (and other traditional methods):
- Basing on the concepts of neighboorhood and density DBSCAN is able to identify all the clusters, even those of different dimensions, artificially originated. On the contrary, CLARANS produces some confusion, in particular it splits the clusters of big dimensions, or misclassifies border points.
- The run time is consistently smaller than CLARANS. Especially if we have a lot of observations, this feature could represent an important advantage.
- DBSCAN introduces the concept of noise (not present in CLARANS and in all the "canonical" methods). This is an interesting feature since we avoid to "force" distant points into a cluster.

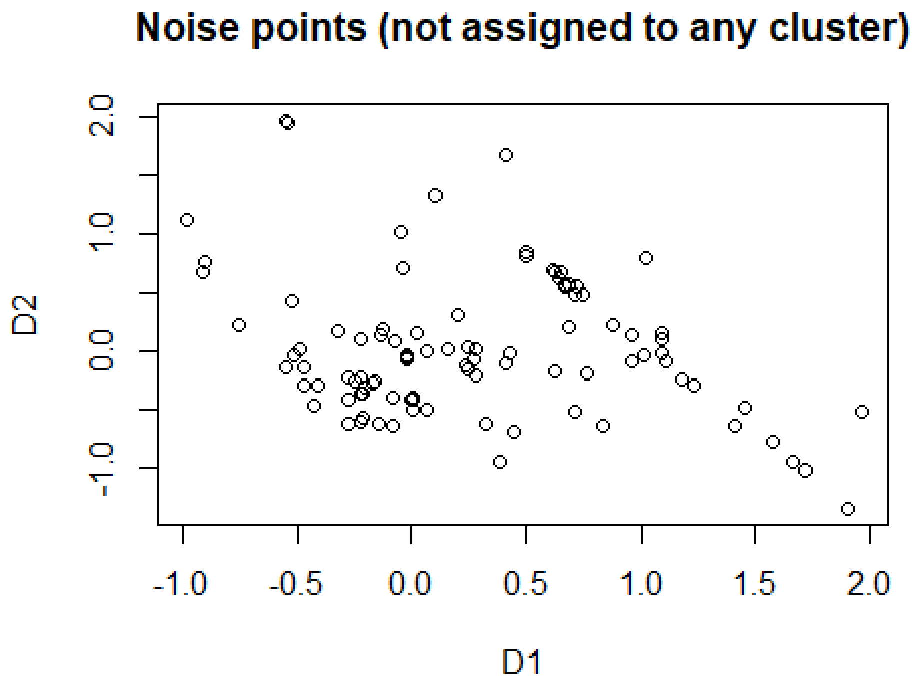
I think that, for the given task (spatial data), the authors' arguments are quite convincing since there are evident advantages over traditional algorithms and CLARANS. However, I have some doubts about the results:
- Doubt 1: the performances of the two algorithms are evaluated on 3 artificial datasets. In order to have more convincing results it would've been interesting to run them on a higher number of simulated datasets.
- Doubt 2: The concept of noise is interesting and useful, but in some real applications we may want, for various reasons, to assign each observation to a cluster. In DBSCAN this may not happen.

DBSCAN algorithm on glass data

```r
library(fpc)

# Fitting DBScan clustering Model
set.seed(200)
Dbscan_glass <- dbscan(glass, eps = 0.45, MinPts = 5)
Dbscan_glass

## dbscan Pts=214 MinPts=5 eps=0.45
##         0   1 2 3
## border 96   6 6 6
## seed    0  95 2 3
## total  96 101 8 9

plot(mdsglass$conf,col=Dbscan_glass$cluster,pch=Dbscan_glass$cluster)
```

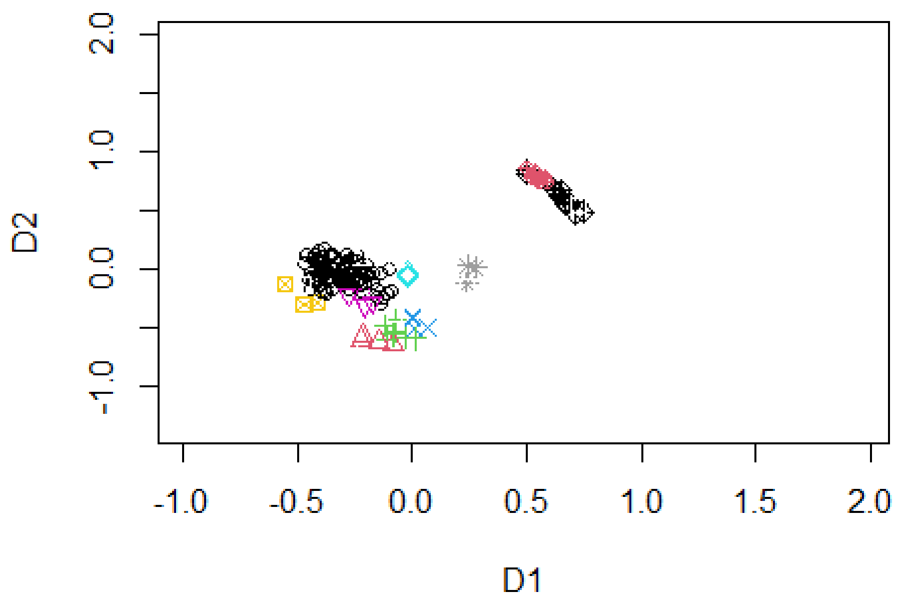


```
glass$dbscanclus <- Dbscan_glass$cluster
mds_data <- as.data.frame(mdsglass$conf)
mds_data$dbscanclust <- Dbscan_glass$cluster
plot(mds_data[mds_data$dbscanclust==0, 1:2], main = "Noise points (not assigned to
any cluster)")
```



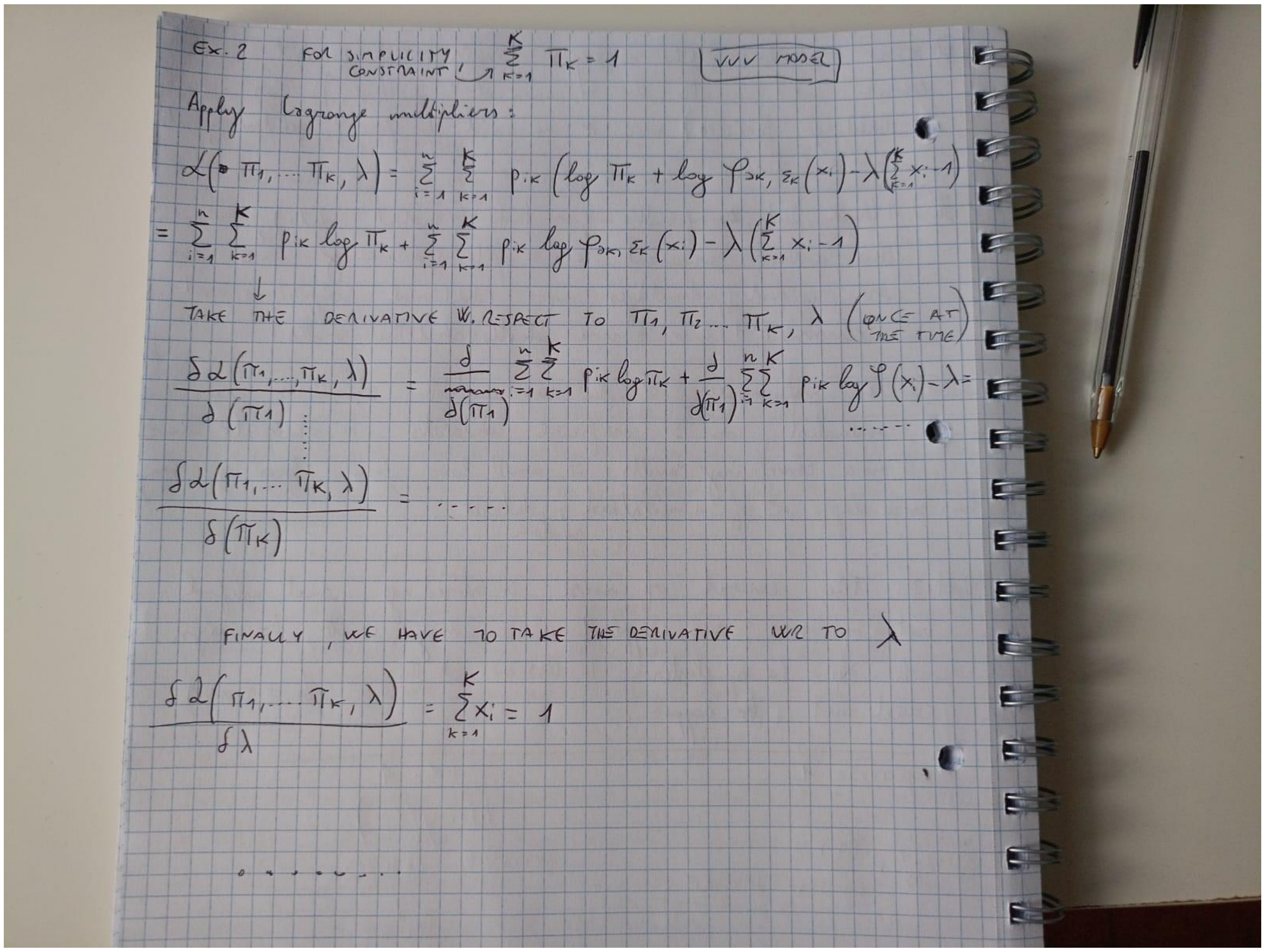
The results are good for what we want to do. The extreme points are classified as noise and not forced to belong to clusters. Now let's try with less "strict" parameters (higher neighborhood radius and smaller Minpts) and see if something changes.

```
set.seed(250)
Dbscan_glass2 <- dbscan(glass, eps = 0.60, MinPts = 3)
Dbscan_glass2

## dbscan Pts=214 MinPts=3 eps=0.6
##         0   1 2 3 4 5 6 7 8  9 10
## border 63   0 2 0 0 0 1 2 0  2  0
## seed    0 101 2 8 5 3 3 1 3  9  9
## total  63 101 4 8 5 3 4 3 3 11  9

plot(mdsglass$conf,col=Dbscan_glass2$cluster,pch=Dbscan_glass2$cluster)
```



The algorithm finds a higher number of clusters (10 vs 3) and less points (63 vs 96) are classified as noise. However, for this task I would want more separation between cluster and, since in the previus analysis (PAM and Gaussian Mixture) we saw the presence of a lot of noise, I would prefere to have stricter parameters, and as a consequence,to have more points classified as noise. For these reason i think that the initial choice of parameters (EPS = 0.45 and MinPts = 5) is good for this task.

Exercise 2



Ex. 2 FOR SIMPLICITY, CONSTRAINT $\nearrow$ $\sum_{k=1}^{K} \pi_k = 1$ $\boxed{\text{VVV MODEL}}$

Apply Lagrange multipliers:

$$\mathcal{L}(\pi_1, \dots \pi_K, \lambda) = \sum_{i=1}^{n} \sum_{k=1}^{K} p_{ik} \left(\log \pi_k + \log \varphi_{\mu_k, \Sigma_k}(x_i) - \lambda \left(\sum_{k=1}^{K} x_i - 1 \right) \right)$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} p_{ik} \log \pi_k + \sum_{i=1}^{n} \sum_{k=1}^{K} p_{ik} \log \varphi_{\mu_k, \Sigma_k}(x_i) - \lambda \left(\sum_{k=1}^{K} x_i - 1 \right)$$

$\downarrow$

TAKE THE DERIVATIVE W. RESPECT TO $\pi_1, \pi_2 \dots \pi_K, \lambda$ (ONCE AT THE TIME)

$$\frac{\delta \mathcal{L}(\pi_1, \dots, \pi_K, \lambda)}{\partial(\pi_1)} = \frac{\partial}{\partial(\pi_1)} \sum_{i=1}^{n} \sum_{k=1}^{K} p_{ik} \log \pi_k + \frac{\partial}{\partial(\pi_1)} \sum_{i=1}^{n} \sum_{k=1}^{K} p_{ik} \log \varphi(x_i) - \lambda =$$

$\dots\dots$

$\vdots$

$$\frac{\delta \mathcal{L}(\pi_1, \dots \pi_K, \lambda)}{\delta(\pi_K)} = \dots\dots$$

FINALLY, WE HAVE TO TAKE THE DERIVATIVE WR TO λ

$$\frac{\delta \mathcal{L}(\pi_1, \dots \pi_K, \lambda)}{\delta \lambda} = \sum_{k=1}^{K} x_i = 1$$

$\dots\dots\dots$