Fiche MAD 4

TP 5 - Kernel PCA

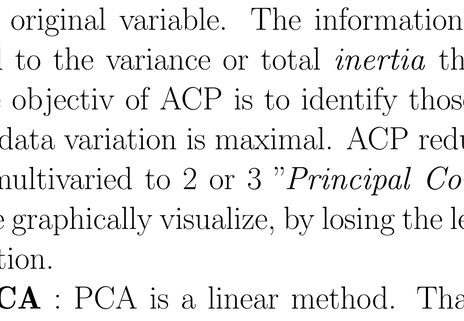
PCA : Principal Component Analysis is a method that consist to transform correlated variables into new variables decorrelated. These new variables are named "*Principal Component*" or principal axis. It allows resuming information by reducing the number of variables. Those new variables are a linear combinaison of original variable. The number of "*Principal Component*" is less or equal than number of original variable. The information in a data correspond to the variance or total *inertia* that he contains. The objectiv of ACP is to identify those direction where the data variation is maximal. ACP reduce dimension of a multivaried to 2 or 3 "*Principal Component*", who can be graphically visualize, by losing the less amount of information.

Kernel PCA : PCA is a linear method. That is it can only be applied to datasets which are linearly separable. But, if we use it to non-linear datasets, we might get a result which may not be the optimal dimensionality reduction. Kernel PCA uses a kernel function to project dataset into a higher dimensional feature space, where it is linearly separable. Kernel Method will help us mesure the similarity between variables.

Examples :

- Gaussian
- Polynomial
- linear

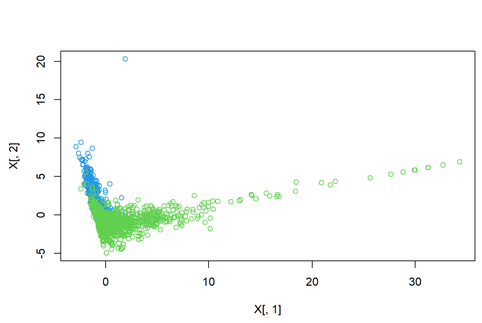
```
library(kernlab)
library(FactoMineR)
data(spam)
df <- spam[, 1:57] Retire a non numeric column
PCA(df)
```



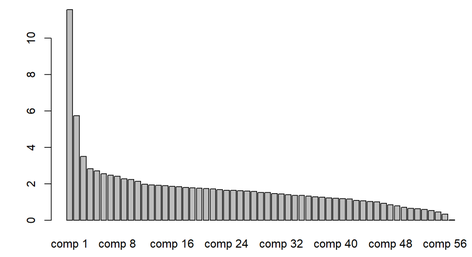
Be careful, before doing a PCA we need to normalize our data. Here we can see that percent of explained variance on axis are way too low, the sum of this 2 value is around 20 percent. The Correlation Circle shows us that the PSA won't work here.

P = percent

```
library(dplyr)
spam-quant <- spam P>P select(-"type")
spam-quant-norm <- scale(spam-quant)
res.PCA <- PCA(spam-quant-norm, graph = F)
eigvalues <- data.frame(res.PCADOLLeig)
barplot(eigvaluesDOLLpercentage.of.variance,
names.arg = row.names(eigvalues))
X <- res.PCADOLLindDOLLcoord
plot(X[,1], X[,2],
col = as.numeric(spamDOLLtype)+2)
```

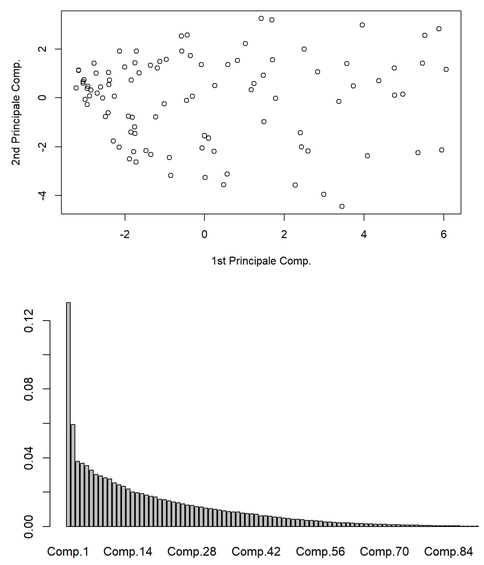


Our value here are represented with 2 colors and those colors depend on the attribute type (non numeric). We do plot our eigen values (still by decreasing order) and we see that it will be hard to choose a number of eigen value to keep in order to use it.



m = 100 N =nrow(spam-quant-norm)

```
Tirage <- sample(1:N, m, replace = FALSE)
spam-subset = spam-quant-norm[Tirage,]
kpc <- kpca( ., data = as.data.frame (spam-subset),
kernel = "rbfdot", kpar = list(sigma=0.01))
kpvc <- pcv(kpc)
plot(rotated(kpc)[,1:2], xlab = "1st Principale
Comp.", ylab = "2nd Principale Comp.")
barplot(eig(kpc)/sum(eig(kpc)))
```



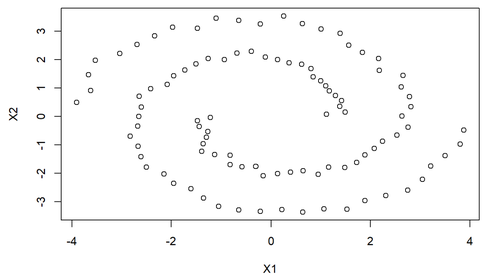
Here we have done kernel PCA with a '*RBF*' a linear model, with now need to work with different type of kernel to compare our result. Around 25 percent of variance is explained so we need to apply few more methods.

Spectral Clustering

Spectral Clustering : In multivariate statistics, spectral clustering techniques make use of the spectrum (eigenvalues) of the similarity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions. The similarity matrix is provided as an input and consists of a quantitative assessment of the relative similarity of each pair of points in the dataset. For example, you can apply it to a population, if you want to promote your product to a large part of the population you can use spectral clustering to target people who are influent in different communities and then use them in order to promote.

It happens when we can't use *K-means* algorithm, because they are no oblivious group.

```
library(mlbench)
set.seed(111)
obj <- mlbench.spirals(100,1,0.025)
my.data <- data.frame(4 * objDOLLx)
names(my.data)<-c("X1","X2")
plot(my.data)
```



Here we can't use *K-Means* because group are not easy to find. We now need to use Kernel PCA in order to find new group.

$$K_\phi(X_i, X_j) = exp(-\frac{1}{2\sigma^2} * ||X_i - X_j||_2^2)$$
$$= exp(-\sigma||X_i - X_j||)$$
$$\text{with : } \sigma = 1$$
$$K_\phi(X_i, X_j) =< \phi(X_i), \phi(X_j) >$$

We use this to project those value in a certain Hilbert's Space. σ was given here, if sigma is a too big variable (100) are correlated a lot and we can't find anything and if it s too low nothing (is correlated and we can't find anything too. We now need to find A the matrix of similarity K_ij express the similarity between x_i and x_j. The bigger the value is, those value are closest. A is symmetric and semi-define positive. A is a matrix of adjacent it's created with the K matrix, and u build it with 2 methods

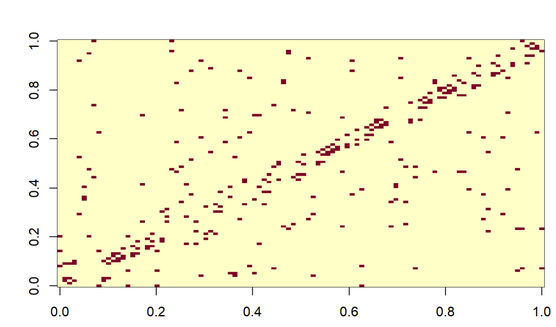
- *K-closest Neighbor* : We look into each line to the biggest value and we choose the K closest value.
- *Threshold* : We choose a threshold and we keep all the value beyond this one.

With this affinity matrix, clustering is replaced by a graph-partition problem, where connected graph components are interpreted as clusters. The graph must be partitioned such that edges connecting different clusters should have low weights, and edges within the same cluster must have high values. Spectral clustering tries to construct this type of graph.

```
diag(K ) = 0
N= 100
A <- matrix(0,N, N)
neighbor <- 3
for(i in 1:N)
idx <- order(K(i,), decreasing= T)
(1: neighbor)
A[i, idx] = 1
image(A)
```

OR

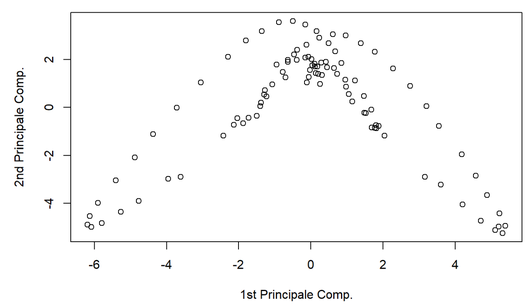
```
A <- K>0.5
diag(A) = 0
image(A)
```



We now need to find the D matrix the matrix of degrees, D_ii represent the degree of node i. With the k-closest neighbor we will always have K on the diag. THe unnormalized graph Laplacian U = D A and/or a normalized version.

```
D <- diag(N)
D <- rowSums(A)
U = D - A
L <- scale(U)
L-value <- eigen(L, symmetric = TRUE)
k <- 2
Z <- L-valueDOLLvectors[,(ncol(L-
valueDOLLvectors)-
k+1):ncol(L-valueDOLLvectors)]
prendre 4 derniers vecteurs et voir bien deux groupes
plot(Z)
my.data<-as.matrix(my.data)
kpc-data <- kpca( ., data = as.data.frame (my.data),
kernel = "rbfdot", kpar = list(sigma=1))

kpvc-data <- pcv(kpc-data)
plot(rotated(kpc-data)[,1:2],
xlab = "1st Principale Comp.",
ylab = "2nd Principale Comp.")
barplot(eig(kpc-data)/sum(eig(kpc-data)))
```



We do have a decomposition and we can work with those values and try to use some linear clustering.