NATIONAL SCHOOL OF STATISTICS AND INFORMATION ANALYSIS

Complement on Unsupervised Learning

Clustering Methods on Simulated Datasets

Lab 1&2

Wrote by:

Moustapha OUSMANE BAWA GAOH
Ashutosh KAUSHIK
Master of Science in Big Data Students

Supervised by :

Badhis Ghattas

Université d'Aix-marseille

06th December 2015

Table des matières

OUSMANE BAWA GAOH Moustapha
KAUSHIK Ashutosh
MSC in Big Data, ENSAI

Introduction

The objective of this report is to present the main points aborted in the course of complement on Unsupervised Learning.

In the following exercise we achieved at first to simulate third Datasets based on different simulations rules. In second place we apply on this third datasets The **Agnes**, the **hclust**, the **K-modes**, the **K-median**, the **DBSCAN** and the **CUBT** which are different clustering methods.

Then in order to see the clustering power and also for compare this different methods between them we compute the miss-classification and the miss-prediction error.

And finally considering the different cases (number of observations, number of variables) we will try to figure out the best model.

I. Clustering methods and Data Simulation

In this part we are going to present briefly the different clustering methods that we used and present the datasets that we simulated.

1. Clustering methods

Clustering is define like the task of grouping a set of objects in such a way that objects in the same cluster are more similar to each other than to those in other clusters. There are many different approaches which are used for make clustering. So there we are going to present the principal ones and the methods based on this approaches that we used.

Hierarchical clustering

Is based on the core idea of objects being more related to nearby objects than to objects farther away. These algorithms connect "objects" to form "clusters" based on their distance.[1]

- **Agnes**: The agnes-algorithm constructs a hierarchy of clusterings. At first, each observation is a small cluster by itself. Clusters are merged until only one large cluster remains which contains all the observations. At each stage the two nearest clusters are combined to form one larger cluster.[2]
- **Hclust:** This function performs a hierarchical cluster analysis using a set of dissimilarities for the n objects being clustered. Initially, each object is assigned to its own cluster and then the algorithm proceeds iteratively, at each stage joining the two most similar clusters, continuing until there is just a single cluster.

Centroid-based clustering (K-means

In centroid-based clustering, clusters are represented by a central vector. When the number of clusters is fixed to k, k-means clustering gives a formal definition as an optimization problem: find

[1] https://en.wikipedia.org/wiki/Cluster_analysis
[2] R help

the k cluster centers and assign the objects to the nearest cluster center, such that the squared distances from the cluster are minimized.

- **K-mode** is a variant of k-means in the case of the input variables are categorical.
- **K-median** is also a variant of k-means in the case of the input variables are ordinal.

Density-based clustering

In density-based clustering, clusters are defined as areas of higher density than the remainder of the data set. Objects in these sparse areas are usually considered to be noise and border points. These group of methods try to combine the advantages of the both previous methods.

- **DBSCAN:** in this method the number of cluster is automatically determined by the algorithm. Points in low-density regions are considered like noise and are omitted, it is the reason why DBSCAN does not produce a complete clustering.

CUBT

Is a top-down hierarchical clustering method that consists of three stages: Growing stage, Pruning stage and Joining stage.

2. Data Simulation

In order to apply the different methods presented above we generated three datasets based on three different simulation rules.

Linear Combination Clustering

Three clusters are defined, each characterized by a high frequency of one different level.

A tree model

Four clusters are defined by using Tree structure.

Another tree model

Four clusters are defined by using Tree structure and we had a parameter q=0.8 that controls the non-uniformity of the distribution of levels.

II. Presentation of the result

In this part we are going to present the miss classification and the miss prediction error of each methods on every dataset.

We perform **100** replicates for each model. And for each dataset we had n= {100; 300; 500; 1000} for the sample sizes.

1. Linear Combination Clustering

With this data set we remark clearly three group of method. In first part we have the Agnes, the Hclust and the k-mode which had the least miss classification and the miss prediction error. For agnes and HCA the prediction.error is around 5% for n=100 and it decrease when the number of observation is increased. K-mode is stable and give low errors between 0.5% and 2% for every number of observations. The K-median is in the second group of method. In this case it is a stable method and all the errors that it produced each number of observation is always around 10%. Finally the methods which are giving the must high errors upper than 30% are the DBSCAN and the CUBT.

Board 1: Miss classification and Miss Prediction error by N (%) on linear combination data

N=100	`` $`N=100`$M.prediction.error ``
	`Agnes       HCA    K_mode    DBSCAN K_median      CUBT`
	`5.66       5.32      1.69     33.54      8.97     30.43`
	`` $`N=100`$M.classification.error ``
	`Agnes       HCA    K_mode    DBSCAN K_median      CUBT`
	`1.69       0.69      1.82     33.58      9.38     27.62`

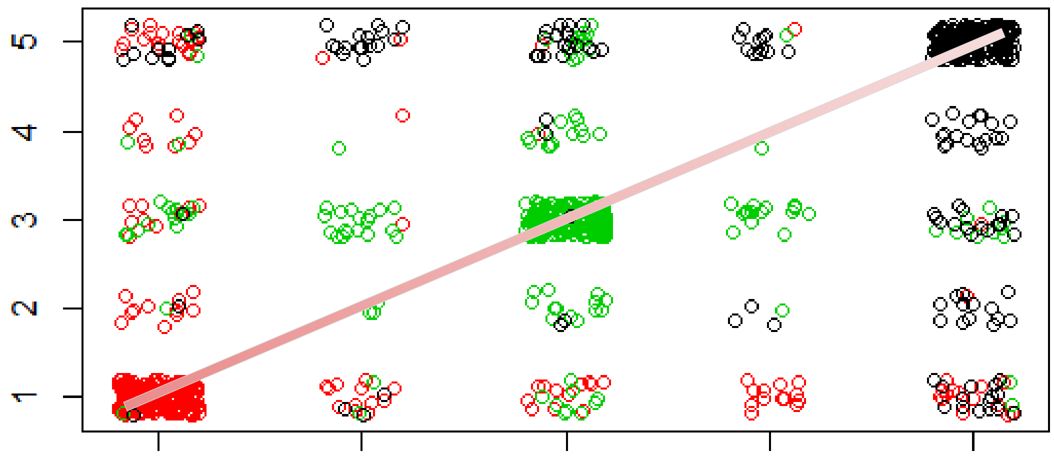
N=300	**`M.prediction.error`**
	`Agnes       HCA    K_mode    DBSCAN K_median      CUBT`
	`0.13       0.12      0.77     33.45      9.77     33.13`
	`M.classification.error`
	`Agnes       HCA    K_mode    DBSCAN K_median      CUBT`
	`1.17       0.46      0.80     33.47      9.79     30.50`

N=500	**`M.prediction.error`**
	`Agnes       HCA    K_mode    DBSCAN K_median      CUBT`
	`0.12       0.12      1.12     33.47      9.86     32.24`
	`M.classification.error`
	`Agnes       HCA    K_mode    DBSCAN K_median      CUBT`
	`1.00       0.47      1.14     33.43     10.01     30.01`

N=1000	**`M.prediction.error`**
	`Agnes       HCA    K_mode    DBSCAN K_median      CUBT`
	`0.10       0.10      0.43     33.42      9.84     32.56`
	`M.classification.error`
	`Agnes       HCA    K_mode    DBSCAN K_median      CUBT`
	`0.93       0.47      0.43     33.41     10.03     30.97`

Also we remark that when we plot the different points of our dataset by using the different obtained clusters, it is like we can make a linear combination between the centres of each cluster.

Figure 1: different points by cluster



2. A tree model

On this dataset we remark that all the different methods are very bad on prediction and also on also for classification. For n<1000 the DBSCAN is the worst one prediction.error= classification.error= 75%, it means that using this method and using à randomly clustering is the same. For n=1000, for the cubt method we got prediction.error= 75.98 % classification.error= 76.10%, in this case it means that it is better to use a randomly clustering method than to use cubt. All the other methods get errors around 58 % and 65 %

Board 2: Miss classification and Miss Prediction error by N (%) on tree model Data set

N=100	M.prediction.error						
		Agnes	HCA	K_mode	DBSCAN	K_median	CUBT
		58.86	58.42	59.62	75.00	64.08	67.09
	M.classification.error						
		Agnes	HCA	K_mode	DBSCAN	K_median	CUBT
		63.99	63.60	59.41	75.00	63.72	67.90
N=300	M.prediction.error						
		Agnes	HCA	K_mode	DBSCAN	K_median	CUBT
		60.68	60.94	63.79	75.00	66.95	72.58
	M.classification.error						
		Agnes	HCA	K_mode	DBSCAN	K_median	CUBT
		67.01	66.32	63.07	75.00	66.96	72.90
N=500	M.prediction.error						
		Agnes	HCA	K_mode	DBSCAN	K_median	CUBT
		61.37	61.19	64.11	75.00	67.11	73.82
	M.classification.error						
		Agnes	HCA	K_mode	DBSCAN	K_median	CUBT
		67.67	67.17	63.53	75.00	67.29	73.98
N=1000	M.prediction.error						
		Agnes	HCA	K_mode	DBSCAN	K_median	CUBT
		61.60	61.46	64.44	75.00	67.96	75.98
	M.classification.error						
		Agnes	HCA	K_mode	DBSCAN	K_median	CUBT
		68.30	67.77	64.34	75.00	68.09	76.10

3. Another tree model

On this tree dataset the errors are lower than on the last one but still high. It seems like the introduction of the non-uniformity make that the different methods become better. It appear also that globally all the different methods are better on prediction than on classification. The DBSCAN still the worst one prediction.error= classification.error= 75%, it means that using this method and using à randomly clustering method is the same. The prediction of Agnes and HCA become better with the number of observations.

Board 3: Miss classification and Miss Prediction error by N (%) on other tree model Data set

		Agnes	HCA	K_mode	DBSCAN	K_median	CUBT
N=100	**M.prediction.error**	**41.43**	**41.30**	**42.14**	**75.00**	**51.94**	**55.74**
	M.classification.error	**59.22**	**56.54**	**42.34**	**75.00**	**51.30**	**56.35**
N=300	M.prediction.error	40.93	41.28	42.33	75.00	53.75	63.45
	M.classification.error	62.67	60.80	41.50	75.00	53.17	63.77
N=500	M.prediction.error	41.13	40.20	42.51	75.00	53.23	65.92
	M.classification.error	63.86	62.74	42.74	75.00	53.21	66.28
N=1000	M.prediction.error	40.28	40.81	42.58	75.00	54.46	68.36
	M.classification.error	64.08	63.32	42.91	75.00	54.45	68.83

Conclusion

At the end of this work we achieved to apply six different clustering methods (Agnes, HCA, K_mode, DBSCAN, K_median and CUBT) on tree different datasets.

It globally appear that **Agnes, HCA** and **K_mode** are the ones which gives the best results follow by **K-median**, then we have **cubt** and at the last position **DBSCAN**. We also remark that all this methods was very bad when we used them on tree model dataset.

And before ending this report, we just want to say that the bad performances of DBSCAN and CUBT can also be due to a bad setting of parameter that we might did.

Annexe

```
####  UNSUPERVISED LEARNING
### Commented installation of packeges
# install.packages("clue")
# install.packages("klaR")
# install.packages("flexclust")
# install.packages("fpc")
# install.packages("RWeka")
# install.packages("divclust")
# install.packages("partitions")
# require(cubt)

library(clue)
library(cluster)
library(e1071)
library(flexclust)
library(klaR)
library(fpc)  ### function dbscan
library(RWeka)  ## function cobweb
library(cubt)
## First creation of the simulated data set

# simulation of the different cluster
LC<-function(n=300,q=0.8,var=9){
  c1=sapply(1:var, function(x) sample(1:5,n/3,T,c(q,rep((1-q)/4,4))))
  c2=sapply(1:var, function(x) sample(1:5,n/3,T,c(rep((1-
q)/4,2),q,rep((1-q)/4,2))))
  c3=sapply(1:var, function(x) sample(1:5,n/3,T,c(rep((1-q)/4,4),q)))
  data1=rbind(c1,c2,c3)
  y<-c(rep(1,n/3),rep(2,n/3),rep(3,n/3))
  mod1<-list(Model=as.data.frame(data1),cluster=y)
  return(mod1)
}

### second data set

Model3<-function(n=400){
  C1=
cbind(sample(c(1,3,5),n/4,T),sample(c(1,3,5),n/4,T),sample(1:6,n/4,T))
  C2=
cbind(sample(c(1,3,5),n/4,T),sample(c(2,4,5),n/4,T),sample(1:6,n/4,T))
  C3=
cbind(sample(c(2,4,6),n/4,T),sample(1:6,n/4,T),sample(c(1,3,5),n/4,T))
  C4=
cbind(sample(c(2,4,6),n/4,T),sample(1:6,n/4,T),sample(c(2,4,6),n/4,T))
  data3=rbind(C1,C2,C3,C4)
  y<-c(rep(1,n/4),rep(2,n/4),rep(3,n/4),rep(4,n/4))
  mod3<-list(Model=as.data.frame(data3),cluster=y)
  return(mod3)
}

### Third DATASET
Model4<-function(n=400,q=0.8){
  C1= cbind(sample(c(1,3),n/4,T,prob=c(q,1-
q)),sample(c(1,3),n/4,T,prob=c(q,1-q)),sample(1:4,n/4,T))
  C2= cbind(sample(c(1,3),n/4,T,prob=c(q,1-
q)),sample(c(2,4),n/4,T,prob=c(q,1-q)),sample(1:4,n/4,T))
  C3= cbind(sample(c(2,4),n/4,T,c(q,1-
q)),sample(1:4,n/4,T),sample(c(1,3),n/4,T,prob=c(q,1-q)))
```

```
  C4= cbind(sample(c(2,4),n/4,T,c(q,1-
q)),sample(1:4,n/4,T),sample(c(2,4),n/4,T,prob=c(q,1-q)))
  data4=rbind(C1,C2,C3,C4)
  y<-c(rep(1,n/4),rep(2,n/4),rep(3,n/4),rep(4,n/4))
  mod4<-list(Model=as.data.frame(data4),cluster=y)
  return(mod4)
}

### computation of the centroids of each cluster for the methods when he
doesn't exist
centroids=function(k=3,training.data,result.cluster){
  centroids<-matrix(NA,nrow=k,ncol=length(names(training.data)))
  for (j in 1:k){
    for (i in 1:length(names(training.data) )){
      centroids[j,i]<-
which.max(table(training.data[,i][result.cluster==j]))
    }
  }
  return(centroids)
}

#### Computation of the prediction
prediction=function(centroids,datatopredict){
  predicty<-rep(NA,length(datatopredict[,1]))
  err<-rep(NA,length(centroids[,1]))

  for(j in 1:length(datatopredict[,1])){
    for (i in 1:length(centroids[,1])){
      err[i]<-sum(datatopredict[j,]!=centroids[i,])
    }
    predicty[j]<-which.min(err)
  }
  return(predicty)
}
###################### Give by Badih Ghattas ##########################
error = function(pred=prev,obs=dd[,1],print=F)
{
  # computes a prediction error
  # uses index defined in our paper
  # proportion of observations not being together within the
  # bigger clusters
  if(length(obs) != length(pred)) stop("obs and pred different length")
  n = length(obs)
  nbcl = length(unique(obs))
  nbclusters = length(unique(pred))
  tab = table(obs,pred)
  if(nbcl <= nbclusters) {
    y = solve_LSAP(tab,maximum=T)
    #print(y)
    tr = sum(tab[cbind(seq_along(y), y)])
    if(print) print(tab)
    res = 1 - (tr / n)
  } else {
    if(nbclusters == 1) {
      res = 1 - (max(tab)/n)
    }else {
      zz= combn(nbcl,nbclusters)
      nn = ncol(zz)
      res = rep(NA,nn)
      for(j in 1:nn)    {
        tabp = tab[zz[,j],]
        y = solve_LSAP(tabp,maximum=T)
        tr = sum(tabp[cbind(seq_along(y), y)])
        if(print) print(tabp)
        res[j] = 1 - (tr / n)
      }
      res = min(res)
    }
  }
  c(res,nbclusters)
}
########################################################################
#######################################
########################################################################
#######################################

bootstr<-function(nboot=20,method,numobs){
  errmatt<-matrix(NA,nrow=nboot,ncol=6)
  errmattclasse<-matrix(NA,nrow=nboot,ncol=6)
  for(bs in 1:nboot){

    if (method==1){
      ####### generating of working datasets
      training<-LC(n=numobs,q=0.8,var=9)
      test<-LC(n=numobs,q=0.8,var=9)
      training.data<-training$Model
      test.data<-test$Model
      clusY<-training$cluster
      observedY<-test$cluster
      k<-3
    } else if (method==2){
      ####### generating of working datasets
      training<-Model3(n=numobs)
      test<-Model3(n=numobs)
      training.data<-training$Model
      test.data<-test$Model
      clusY<-training$cluster
      observedY<-test$cluster
      k<-4
    }else if (method==3){
      ####### generating of working datasets
      training<-Model4(n=numobs,q=0.8)
      test<-Model4(n=numobs,q=0.8)
      training.data<-training$Model
      test.data<-test$Model
      clusY<-training$cluster
      observedY<-test$cluster
      k<-4
    }
    ##########################################################
    ## With the function Agnes present in the package cluster ##
    ##########################################################
    result.agnes=agnes(training.data,method="ward")
    #plot(result.agnes)
    cluster.agnes=cutree(result.agnes,h=25,k=k)

    ## finding the center of this methods
    center<-
centroids(k,training.data=training.data,result.cluster=cluster.agnes)
    ## Predicting the clusters' of each data set
    predict.y<-prediction(center,test.data)
    Agnes=error(pred = predict.y, obs = observedY,print = F )[1]
    Agnes_MCE<-error(pred = cluster.agnes, obs = clusY,print = F )[1]


    ##########################################################
    # With functions available from basic installation of R #
    ##########################################################
    ## by using hclust function ward method
    distance=dist(training.data, "manhattan")
    obs=row.names(training.data)
    result.hclust=hclust(distance,method="ward.D")

    #plot(result.hclust,labels=obs,ylab="Distance",main="Dendrogram")

    # To obtain a certain level clustering
    cluster.hclust<-cutree(result.hclust,k=k)

    center.ward<-centroids(k,training.data,cluster.hclust)
    predict.y<-prediction(centroids =
center.ward,datatopredict=test.data)

    HCA<-error(predict.y,observedY)[1]
    HCA_MCE<-error(pred = cluster.hclust, obs = clusY,print = F )[1]
    ##########################################################
    # With the Kmode present in the package klar #
    ##########################################################
    result.kmodes<-kmodes(training.data,k,iter.max = 10)
    obs=row.names(training.data)

   #     plot(jitter(as.matrix(training.data)),
col=result.kmodes$cluster)
   #     points(result.kmodes$modes, col = 1:5, pch = 8)
   #     plot(training.data,col=(result.kmodes$cluster+1),pch=20,cex=2)
    centers<-result.kmodes$modes

    predict.y<-prediction(centroids = centers,datatopredict = test.data)

    K_mode<-error(predict.y,observedY)[1]
    Kmode_MCE<-error(pred = result.kmodes$cluster, obs = clusY,print = F
)[1]

    ##########################################################
    # With the DSCAN present in the package fpc #
    ##########################################################
    result.dbscan<-dbscan(training.data, eps=3.5, method = "raw", MinPts
=  0.8*(numobs/k))

    predict.y<-predict(result.dbscan,training.data,test.data)

    DBSCAN<-error(predict.y,observedY)[1]
    DBSCAN_MCE<-error(pred = result.dbscan$cluster, obs = clusY,print = F
)[1]

    #plot(jitter(training.data), col=result.dbscan$cluster)
    #print.dbscan(result.dbscan,training.data)

    ##########################################################
    # With the KCCA function present in the package flexclust#
    ##########################################################

    result.median<-kcca(training.data,k = k,family =
kccaFamily("kmedians"))
```

```
    kmed.cluster<-slot(result.median,"cluster")
    predict.y<-predict(result.median,newdata=test.data)

    K_median<-error(predict.y,observedY)[1]
    Kmedian_MCE<-error(pred = kmed.cluster, obs = clusY,print = F )[1]
    ##########################################################
    # With the CUBT function present in the package CUBT     #
    ##########################################################

    result.cubt<-cubt(as.matrix(training.data), critopt =
"entropy",minsplit = 0.8*(numobs/k),
                       minsize = log(numobs),mindev=0.001)
    #vv<-prune.cubt(result.cubt,training.data)
    #join.cubt(vv,training.data,nclass = 3)
    #plot(result.cubt,type="u")
    #text(result.cubt)
    cubt.cluster<-where(result.cubt)
    predict.y<-where(predict(result.cubt,test.data))
    cubt<-error(predict.y,observedY)[1]
    cubt_MCE<-error(pred = cubt.cluster, obs = clusY,print = F )[1]


    #### Putting the different result in a matrix
    errmatt[bs,]<-c(Agnes,HCA,K_mode,DBSCAN,K_median,cubt)
    errmattclasse[bs,]<-
c(Agnes_MCE,HCA_MCE,Kmode_MCE,DBSCAN_MCE,Kmedian_MCE,cubt_MCE)
  }
  MPE<-round(colMeans(errmatt)*100,2)
  names(MPE)<-c("Agnes","HCA","K_mode","DBSCAN","K_median","CUBT")
  MCE<-round(colMeans(errmattclasse)*100,2)
  names(MCE)<-c("Agnes","HCA","K_mode","DBSCAN","K_median","CUBT")
  Error2<-list(M.prediction.error=MPE,M.classification.error=MCE)
  return(Error2)
}

n300M1<-bootstr(nboot = 100,method = 1,numobs = 300)
n300M2<-bootstr(nboot = 100,method = 2,numobs = 300)
n300M3<-bootstr(nboot = 100,method = 3,numobs = 300)

n500M1<-bootstr(nboot = 100,method = 1,numobs = 500)
n500M2<-bootstr(nboot = 100,method = 2,numobs = 500)
n500M3<-bootstr(nboot = 100,method = 3,numobs = 500)

n100M1<-bootstr(nboot = 100,method = 1,numobs = 100)
n100M2<-bootstr(nboot = 100,method = 2,numobs = 100)
n100M3<-bootstr(nboot = 100,method = 3,numobs = 100)

n1000M1<-bootstr(nboot = 100,method = 1,numobs = 1000)
n1000M2<-bootstr(nboot = 100,method = 2,numobs = 1000)
n1000M3<-bootstr(nboot = 100,method = 3,numobs = 1000)

Result1<-
list("N=100"=n100M1,"N=300"=n300M1,"N=500"=n500M1,"N=1000"=n1000M1)
Result2<-
list("N=100"=n100M2,"N=300"=n300M2,"N=500"=n500M2,"N=1000"=n1000M2)
Result3<-
list("N=100"=n100M3,"N=300"=n300M3,"N=500"=n500M3,"N=1000"=n1000M3)
```