REPORT

COMPLEMENTS ON UNSUPERVISED LEARNING

MENTOR :--
BADIH GHATTAS

MEMBERS INCLUDED:--
1) LUCAS DE LA BROSSE
2) SHISHIR DUBEY

LAB 1

École nationale de la statistique
et de l'analyse de l'information

AIM :

1) To compare different clustering methods over three categorical datasets distribued according three differents models

Main function used : **compare**(n , p , l , q , b , which_data , k_cluster)

inputs:

l :number of labels of levels
q :prob
n :number of observations
p :number of variables
b :number of bootstrap samples
which_data :allows to select a dataset 1:LC ect....
k_cluster :the number of supposed cluster

outputs :

A data.frame containing: _The mean of the miss classification rate over b bootstrap samples for the three method used : k-modes, k-medians and cubt.
_The mean of the classification error rate over b bootstrap samples for the three method used : k-modes, k-medians and cubt.

Note: For the k-modes prediction, the rule is to allocate the cluster which has the closest centroids according to simple matching distance.

I)Clustering on the First Dataset

Comments on the dataset : For the First Model Dataset we analyzed that the data in the real world within LC-simulations can be manipulated as it depends on the way we are following

compare(n=… , p = 9 , l = 5 , q = 0.8 , b =100 , which_data = 1 , k_cluster = 3) for n=99,300,800

	percentage_miss_classification	percentage_prediction_error	nb_of_observations
mean_k_modes	1.555556	1.484848	99
mean_k_medians	11.010101	10.393939	99
mean_cubt	19.252525	27.171717	99
mean_k_modes	0.4566667	0.4466667	300
mean_k_medians	7.0033333	6.8066667	300
mean_cubt	20.5700000	25.6166667	300
mean_k_modes	0.4398496	0.433584	798
mean_k_medians	7.9674185	7.927318	798
mean_cubt	24.1027569	26.607769	798

II) Clustering on the second dataset:

compare(n=… , p = 3 , l = 6 , b =100 , which_data = 2 , k_cluster = 4) for n=100,300,800

	percentage_miss_classification	percentage_prediction_error	nb_of_observations
mean_k_modes	58.53	60.27	100
mean_k_medians	63.61	64.48	100
mean_cubt	62.96	64.72	100
mean_k_modes	62.87333	63.54	300
mean_k_medians	66.46000	66.92	300
mean_cubt	63.67333	63.97	300
mean_k_modes	64.41000	64.23125	800
mean_k_medians	67.97250	68.23750	800
mean_cubt	68.37875	68.05250	800

III)Clustering on the third dataset:

compare(n=… , p = 3 , l = 6 , q = 0.8 , b =100 , which_data = 3 , k_cluster = 4) for n=99,300,800

	percentage_miss_classification	percentage_prediction_error	nb_of_observations
mean_k_modes	32.80	32.01	100
mean_k_medians	51.33	53.43	100
mean_cubt	48.73	49.39	100
mean_k_modes	35.67333	35.26000	300
mean_k_medians	53.88000	54.70667	300
mean_cubt	51.27333	52.17333	300
mean_k_modes	36.45875	36.25875	800
mean_k_medians	56.56625	56.97375	800
mean_cubt	53.46625	53.55625	800

IV)CODES

```r
#install.packages("klaR")   ##kmodes
library(klaR)

#install.packages("combinat") ## kcca
library(combinat)

#install.packages("clue")
library(clue)

#install.packages("RWeka")
library(RWeka)


#install.packages("flexclust")
library(flexclust)

#install.packages("RWekajars")
library(RWekajars)

#install.packages("rJava")
library(rJava)


#####################################################
#####################################################

# l number of labels of levels
# q prob
# n number of observations
# p number of variables
LC=function(n=300,p=9,l=5,q=0.9){

cluster1=sapply(c(1:p),function(x) sample(c(1:l),prob=c(q,rep((1-q)/(l-1),(l-1))),T,size=n/3))
cluster2=sapply(c(1:p),function(x) sample(c(3,1:l)[-4],prob=c(q,rep((1-q)/(l-1),(l-1))),T,size=n/3))
cluster3=sapply(c(1:p),function(x) sample(c(5,1:l)[-6],prob=c(q,rep((1-q)/(l-1),(l-1))),T,size=n/3))

cluster=rep(1,n/3)
dataset=rbind(cbind(cluster1,cluster),cbind(cluster2,2),cbind(cluster3,3))
#dataset=dataset[sample(1:n,n),]

}
```

```
# l number of labels of levels
# n number of observations
M2=function(n=300,l=6){

cluster1=sapply(c(1:3),function(x)
if(x==1 || x==2){ sample(seq(from=1,to=l,by=2),prob=rep(2/l,trunc((l+1)/2)),T,size=n/4) }
else if (x==3) {sample(c(1:l),prob=rep(1/l,l),T,size=n/4)} )

cluster2=sapply(c(1:3),function(x)
if(x==2){ sample(seq(from=2,to=l,by=2),prob=rep(2/l,trunc(l/2)),T,size=n/4) }
else if (x==1) {sample(seq(from=1,to=l,by=2),prob=rep(2/l,trunc((l+1)/2)),T,size=n/4)}
else if (x==3){sample(c(1:l),prob=rep(1/l,l),T,size=n/4)} )

cluster3=sapply(c(1:3),function(x)
if(x==1){ sample(seq(from=2,to=l,by=2),prob=rep(2/l,trunc(l/2)),T,size=n/4) }
else if (x==3) {sample(seq(from=1,to=l,by=2),prob=rep(2/l,trunc((l+1)/2)),T,size=n/4)}
else if (x==2){sample(c(1:l),prob=rep(1/l,l),T,size=n/4)} )

cluster4=sapply(c(1:3),function(x)
if(x==1 || x==3){ sample(seq(from=2,to=l,by=2),prob=rep(2/l,trunc(l/2)),T,size=n/4) }
else if (x==2) {sample(c(1:l),prob=rep(1/l,l),T,size=n/4)} )

cluster=rep(1,n/4)
dataset=rbind(cbind(cluster1,cluster),cbind(cluster2,2),cbind(cluster3,3),cbind(cluster4,4))
dataset=dataset[sample(1:n,n),]

}
```

```
# n number of observations
# l number of labels
# q probability
M3=function(n=300,l=6,q=0.8){

cluster1=sapply(c(1:3),function(x)
if(x==1 || x==2){ sample(seq(from=1,to=l,by=2),prob=c(q,rep(((1-q)/(trunc((l+1)/2))),trunc((l+1)/2)-1)),T,size=n/4) }
else if (x==3) {sample(c(1:l),prob=rep(1,l),T,size=n/4)} )

cluster2=sapply(c(1:3),function(x)
if(x==2){sample(seq(from=2,to=l,by=2),prob=c(q,rep(((1-q)/(trunc((l+1)/2))),trunc((l+1)/2)-1)),T,size=n/4)}          ## creer vecteur 1 pour cluster 2
else if (x==1){sample(seq(from=1,to=l,by=2),prob=c(q,rep(((1-q)/(trunc((l+1)/2))),trunc((l+1)/2)-1)),T,size=n/4) }  ## creer vecteur 2 pour cluster 2
else if (x==3){sample(c(1:l),prob=rep(1,l),T,size=n/4)} )                          ## creer vecteur 3 pour cluster 2

cluster3=sapply(c(1:3),function(x)
if(x==1){ sample(seq(from=2,to=l,by=2),prob=c(q,rep(((1-q)/(trunc((l+1)/2))),trunc((l+1)/2)-1)),T,size=n/4) }
else if (x==3) {sample(seq(from=1,to=l,by=2),prob=c(p,rep(((1-q)/(trunc((l+1)/2))),trunc((l+1)/2)-1)),T,size=n/4)}
else if (x==2){sample(c(1:l),prob=rep(1,l),T,size=n/4)} )

cluster4=sapply(c(1:3),function(x)
if(x==1 || x==3){ sample(seq(from=2,to=l,by=2),prob=c(q,rep(((1-q)/(trunc((l+1)/2))),trunc((l+1)/2)-1)),T,size=n/4) }  ## creer vecteur 1 et 2 pour cluster 1
else if (x==2) {sample(c(1:l),prob=rep(1,l),T,size=n/4)} )                         ## creer vecteur 3 pour cluster 1

cluster=rep(1,n/4)
dataset=rbind(cbind(cluster1,cluster),cbind(cluster2,2),cbind(cluster3,3),cbind(cluster4,4))
#dataset=dataset[sample(1:n,n),]

}
```

```
###################
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(nbcl > nbclusters){
    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)
    }
  }
  return(res=c(res,nbclusters))
}

#######################################

manhattan_distance=function(x,y){
  if(length(x) != length(y)) {stop("x and y different length")}
  return(sum(abs(x-as.numeric(y))))
}

simple_matching_distance=function(x,y){
    if(length(x) != length(y)) {stop("x and y different length")}
  return(length(which(x!=y))/length(x))
}
```

```
##################
##################


# l number of labels of levels
# q prob
# n number of observations
# p number of variables
# b number of bootstrap samples
# which_data allows to select a dataset 1:LC ect....
# k_cluster the number of supposed cluster
compare=function(n=300,p=9,l=5,q=0.8,b=200,which_data=1,k_cluster=3){

  ####### vector which stocks the miss classification rate and prediction error rate over b samples
  miss_classification_rate_k_modes=c()
  prediction_error_k_modes=c()

  miss_classification_rate_k_medians=c()
  prediction_error_k_medians=c()

  miss_classification_rate_cubt=c()
  prediction_error_cubt=c()

  ######## starts bootstrap loop
  for(i in 1:b){

    ######## create a training and a test sample according to the selected model
    if(which_data==1){
      X_test=LC(n,p,l,q)
      X_training=LC(n,p,l,q)
    }else if (which_data==2){
      X_test=M2(n,l)
      X_training=M2(n,l)
    }else{
      X_test=M3(n,m=l,p=q)
      X_training=M3(n,m=l,p=q)
    }

    ######## to avoid dimension issues
    n=nrow(X_training)
    p=ncol(X_training)-1

    ####### k_modes
    mod_k_modes=kmodes(X_training[,1:p],k_cluster)
    cluster_centroids=mod_k_modes$modes
    nb_Cluster=nrow(cluster_centroids)

    ####### An empty matrix, will be fill up with the distance from each observation to the cluster
    cluster_Distances=matrix(NA,nrow=n,ncol=nb_Cluster)

    ####### loop : compute the distance between each cluster's centroid and the observation
    for(j in 1:nb_Cluster){
      cluster_Distances[,j]=apply(X_test[,1:p],1,simple_matching_distance,y=cluster_centroids[j,])
    }

    ####### affect each observation to the closest cluster
    cluster_Belonging=apply(cluster_Distances,1,which.min)

    ######## using the error function, stocks for each simulation the
    miss_classification_rate_k_modes[i]=error(mod_k_modes$cluster,X_training[,'cluster'])[1]
```

```
  prediction_error_k_modes[i]=error(cluster_Belonging,X_test[,'cluster'])[1]

  ####### k-medians
  mod_k_medians=kcca(X_training[,1:p],k_cluster,family=kccaFamily("kmedians"))

  ######## using the error function, stocks for each simulation the
  miss_classification_rate_k_medians[i]=error(slot(mod_k_medians,"cluster"),X_training[,'cluster'])[1]
  prediction_error_k_medians[i]=error(predict(mod_k_medians,newdata=X_test[,1:p]),X_test[,'cluster'])[1]

  ####### cubt
  mod_cubt=cubt(X_training[,1:p],critopt ='entropy',minsplit = 0.8*(n/k_cluster) ,minsize = trunc(log(n)), mindev =
0.001)
  mod_cubt=prune.cubt(mod_cubt,X_training[,1:p])
  mod_cubt=join.cubt(mod_cubt,X_training[,1:p], nclass = 3, crit0 = 'entropy')

  ######## using the error function, stocks for each simulation the
  cluster_Belonging=where(mod_cubt)
  miss_classification_rate_cubt[i]=error(cluster_Belonging,X_training[,'cluster'])[1]
  cluster_Belonging=where(predict(mod_cubt,X_test[,1:p]))
  prediction_error_cubt[i]=error(cluster_Belonging,X_test[,'cluster'])[1]
 }

 ######## returns a list of data frames containing the mean of the miss classification rate
 ######## and the mean of the prediction error rate for each method
 data.frame(
  percentage_miss_classification=c(
   mean_k_modes=mean(miss_classification_rate_k_modes*100),
   mean_k_medians=mean(miss_classification_rate_k_medians*100),
   mean_cubt=mean(miss_classification_rate_cubt*100)
  ),
  percentage_prediction_error=c(
   mean_k_modes=mean(prediction_error_k_modes*100),
   mean_k_medians=mean(prediction_error_k_medians*100),
   mean_cubt=mean(prediction_error_cubt*100)
  ),
  nb_of_observations=c(n,n,n)
 )
}
```