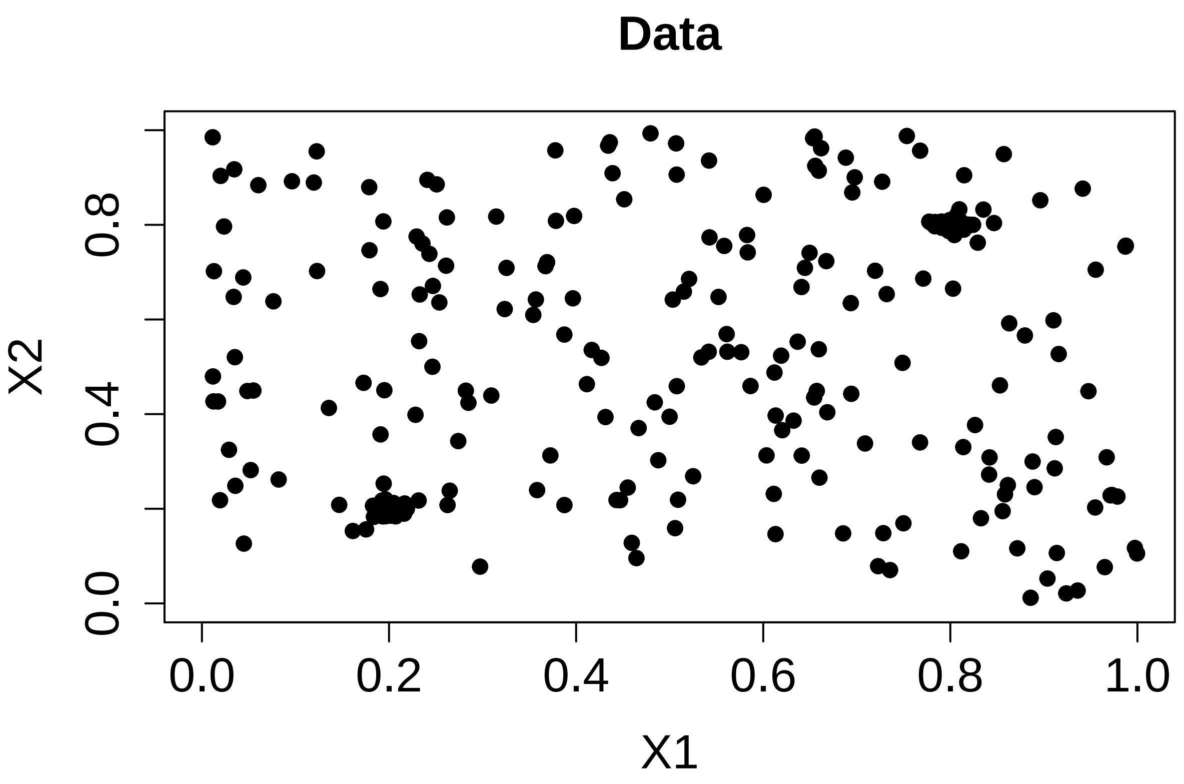
Final Projet

Ariane Ducellier

12/12/2019

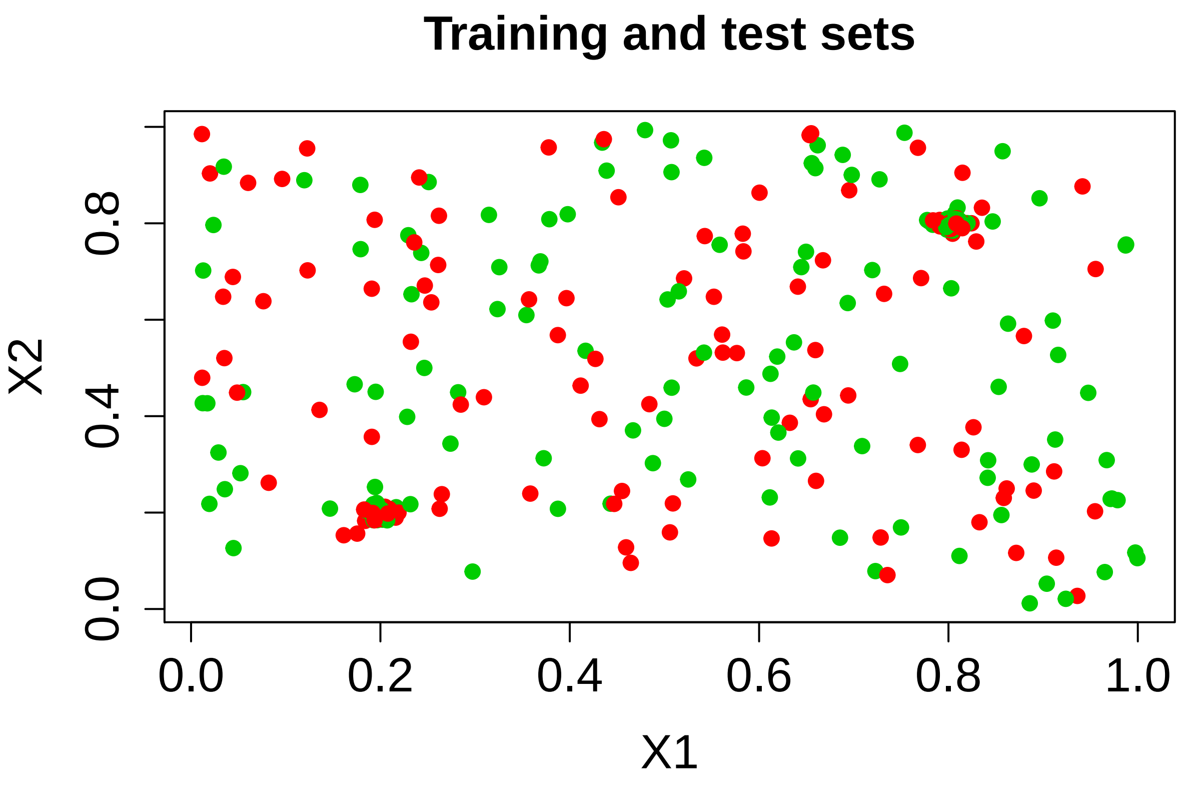
Data set

```r
set.seed(12122019)
N1 = 40
N2 = 40
N3 = 200
c1 = c(0.2, 0.2)
c2 = c(0.8, 0.8)
sigma = 0.01
X1 = vector(length = N1 + N2 + N3)
X2 = vector(length = N1 + N2 + N3)
X1[1 : N1] = rnorm(N1, c1[1], sigma)
X2[1 : N1] = rnorm(N1, c1[2], sigma)
X1[(N1 + 1) : (N1 + N2)] = rnorm(N2, c2[1], sigma)
X2[(N1 + 1) : (N1 + N2)] = rnorm(N2, c2[2], sigma)
X1[(N1 + N2 + 1) : (N1 + N2 + N3)] = runif(N3)
X2[(N1 + N2 + 1) : (N1 + N2 + N3)] = runif(N3)
X = matrix(c(X1, X2), nr=N1 + N2 + N3, nc=2)
par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
plot(X1, X2, xlab="X1", ylab="X2", pch=19, main="Data", xlim=c(0, 1), ylim=c(0, 1))
```



Training and test sets

```
index = runif(N1 + N2 + N3)
col_data = vector(length = N1 + N2 + N3)
train = which(index <= 0.5)
test = which(index > 0.5)
Xtr = X[train, ]
Xte = X[test, ]
ntr = dim(Xtr)[1]
nte = dim(Xte)[1]
col_data[train] <- 2
col_data[test] <- 3
par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
plot(X1, X2, col=col_data, xlab="X1", ylab="X2", pch=19, main="Training and test sets")
```



Hierarchical clustering

Distance between observations

```
dist_obs <- function(X1, X2){
    d = sqrt((X1[1] - X2[1])^2 + (X1[2] - X2[2])^2)
    return(d)
}
Dtr = matrix(rep(0, ntr^2), nr = ntr, nc = ntr)
```

```
Dte = matrix(rep(0, nte^2), nr = nte, nc = nte)
for (i in 1 : ntr){
    for (j in 1 : ntr){
        Dtr[i, j] = dist_obs(Xtr[i, ], Xtr[j, ])
    }
}
for (i in 1 : nte){
    for (j in 1 : nte){
        Dte[i, j] = dist_obs(Xte[i, ], Xte[j, ])
    }
}
```

Distance between clusters

```
dist_clust <- function(D, C1, C2, type){
    if (type == "complete"){
        Dsub = matrix(rep(0, length(C1) * length(C2)), nr=length(C1), nc=length(C2))
        for (i in 1 : length(C1)){
            for (j in 1 : length(C2)){
                Dsub[i, j] = D[C1[i], C2[j]]
            }
        }
        return(max(Dsub))
    }
}
```

Iteration: We merge the clusters with the lowest distance between them

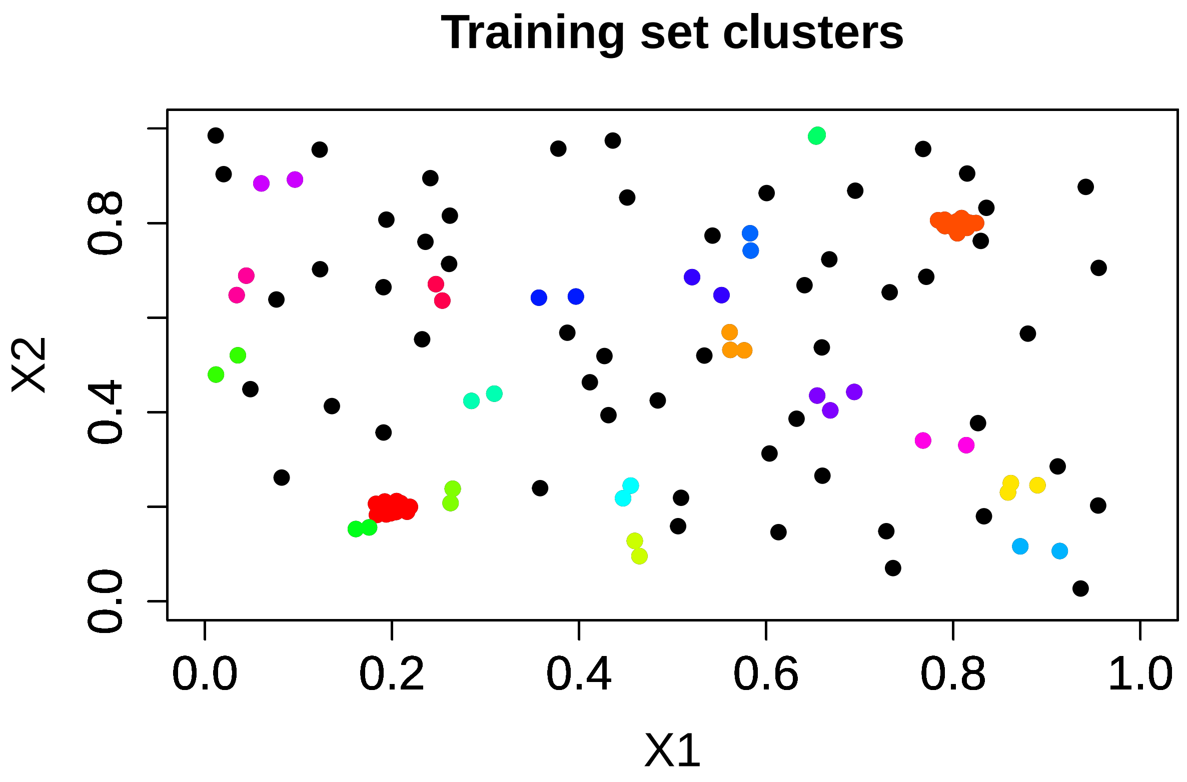
```
iteration <- function(D, clusters, type, h){
    # Distance between clusters
    Dclust = matrix(rep(0, length(clusters)^2), nr=length(clusters), nc=length(clusters))
    for (i in 1 : length(clusters)){
        for (j in 1 : length(clusters)){
            Dclust[i, j] = dist_clust(D, clusters[[i]], clusters[[j]], type)
        }
    }
    # Find minimum distance
    mask = upper.tri(Dclust)
    upperD = Dclust * mask
    upperD[upperD == 0] <- NA
    index = which(upperD == min(upperD, na.rm=TRUE), arr.ind=TRUE)
    i0 = index[1]
    j0 = index[2]
    # Merge clusters with minimum distance
    if (min(upperD, na.rm=TRUE) < h){
        clusters[[i0]] = c(clusters[[i0]], clusters[[j0]])
        clusters = clusters[-j0]
        return(list("clusters" = clusters, "state" = "continue"))
    } else {
        return(list("clusters" = clusters, "state" = "stop"))
    }
}
```

Clustering of training and test data sets

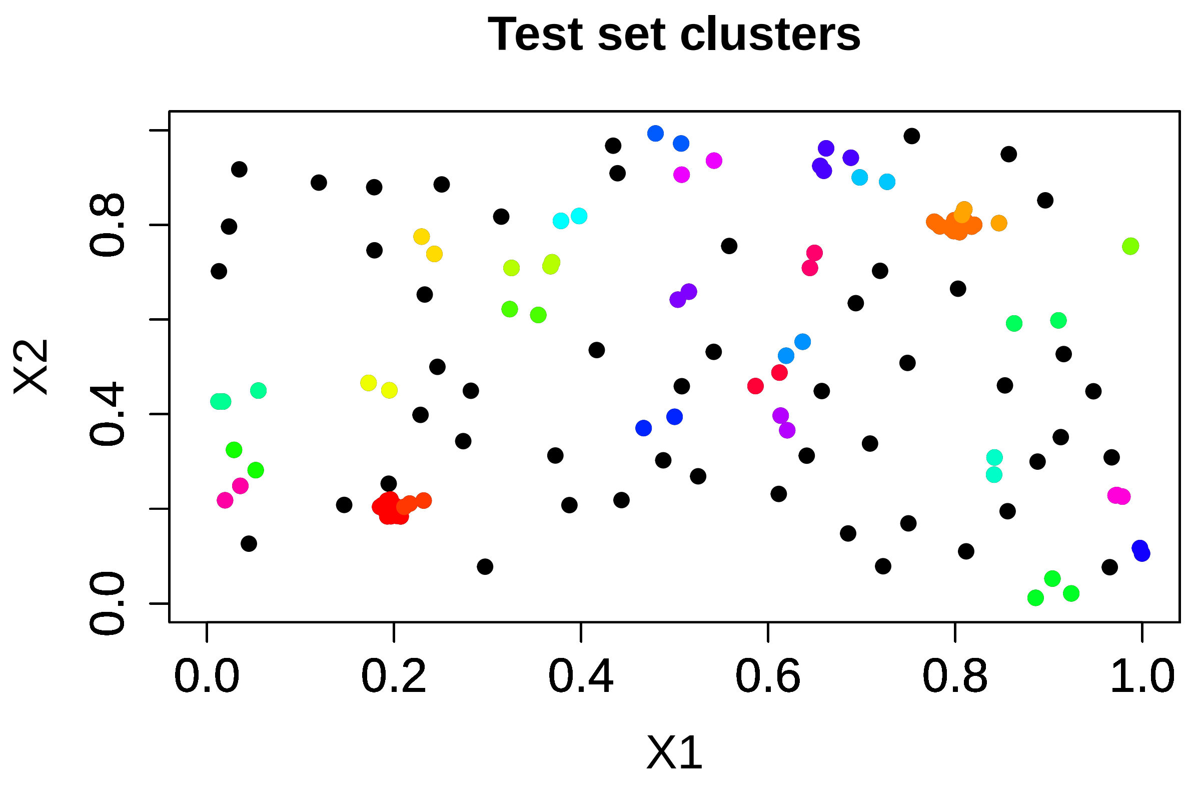
```
h = 0.05
type = "complete"
# Hierarchical clustering of training set
state = "continue"
clusters_tr = list()
for (i in 1:ntr){
    clusters_tr = c(clusters_tr, c(i))
}
while (state == "continue"){
    result = iteration(Dtr, clusters_tr, type, h)
    clusters_tr = result$clusters
    state = result$state
}
# Hierarchical clustering of test set
state = "continue"
clusters_te = list()
for (i in 1:nte){
    clusters_te = c(clusters_te, c(i))
}
while (state == "continue"){
    result = iteration(Dte, clusters_te, type, h)
    clusters_te = result$clusters
    state = result$state
}
```

Plot the results

```
# Training set
par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
plot(Xtr[, 1], Xtr[, 2], col=1, xlab="X1", ylab="X2", pch=19, main="Training set clusters", xlim=c(0, 1
nc_tr = 0
for (i in 1:length(clusters_tr)){
    if (length(clusters_tr[[i]]) > 1){
        nc_tr = nc_tr + 1
    }
}
color_tr = rainbow(nc_tr)
ic = 1
for (i in 1:length(clusters_tr)){
    if (length(clusters_tr[[i]]) > 1){
        indices = clusters_tr[[i]]
        par(new=TRUE)
        plot(Xtr[indices, 1], Xtr[indices, 2], col=color_tr[ic], xlab="", ylab="", pch=19, main="", xli
        ic = ic + 1
    }
}
```



```
# Testing set
par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
plot(Xte[, 1], Xte[, 2], col=1, xlab="X1", ylab="X2", pch=19, main="Test set clusters", xlim=c(0, 1), yl
nc_te = 0
for (i in 1:length(clusters_te)){
    if (length(clusters_te[[i]]) > 1){
        nc_te = nc_te + 1
    }
}
color_te = rainbow(nc_te)
ic = 1
for (i in 1:length(clusters_te)){
    if (length(clusters_te[[i]]) > 1){
        indices = clusters_te[[i]]
        par(new=TRUE)
        plot(Xte[indices, 1], Xte[indices, 2], col=color_te[ic], xlab="", ylab="", pch=19, main="", xli
        ic = ic + 1
    }
}
```



K-means clustering

Centroids of the training set

```
centroids = matrix(rep(0, length(clusters_tr) * 2), nr=length(clusters_tr), nc=2)
for (i in 1:length(clusters_tr)){
    indices = clusters_tr[[i]]
    centroids[i, 1] = mean(Xtr[indices, 1])
    centroids[i, 2] = mean(Xtr[indices, 2])
}
```

K-means classifier

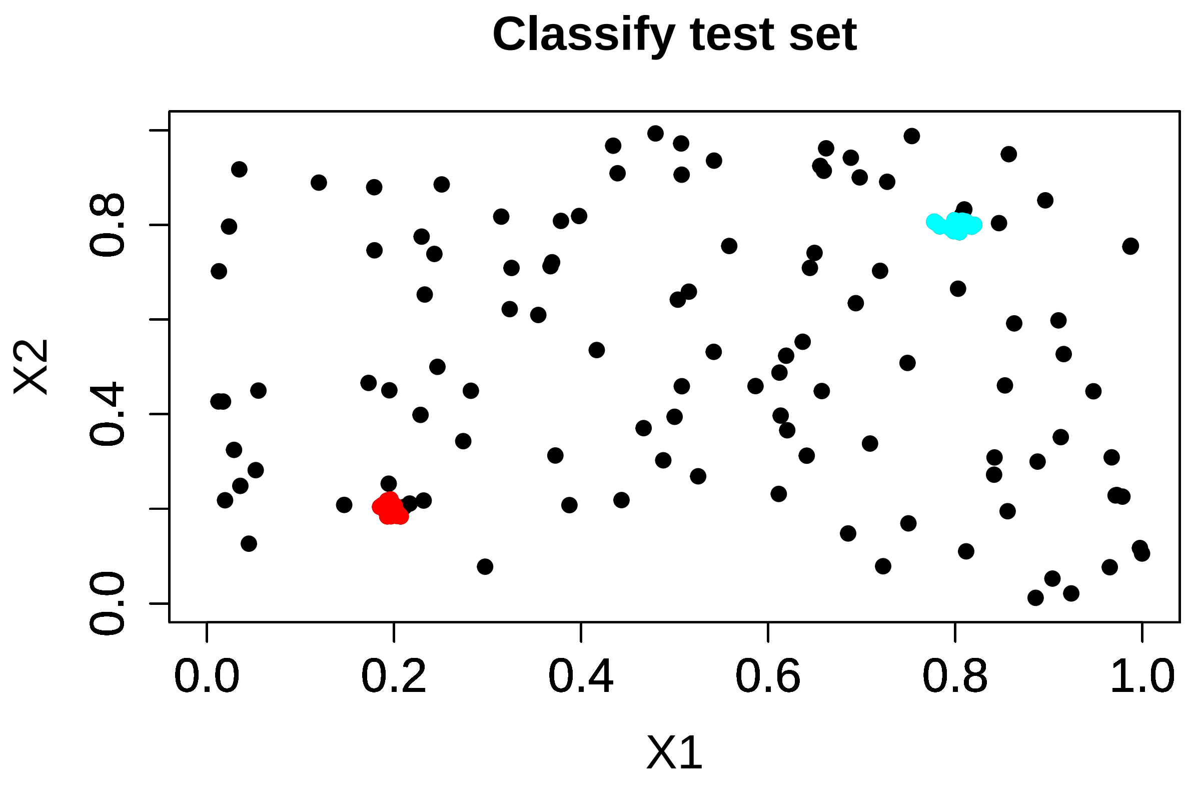
```
kmeans <- function(centroids, Xte, i){
    distance = sqrt((centroids[, 1] - Xte[i, 1])^2 + (centroids[, 2] - Xte[i, 2])^2)
    index = which.min(distance)
    return(index)
}
```

Classify observations from the test set

```
classification = rep(0, nte)
for (i in 1:nte){
    classification[i] = kmeans(centroids, Xte, i)
}
```

PLot class of observations in big clusters

```
min_nb_el = 5
# Plot all observations
par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
plot(Xte[, 1], Xte[, 2], col=1, xlab="X1", ylab="X2", pch=19, main="Classify test set", xlim=c(0, 1), yl
# Look for big clusters
clusters = list()
for (i in 1:length(clusters_te)){
    if (length(clusters_te[[i]]) > min_nb_el){
        clusters = c(clusters, clusters_te[i])
    }
}
# List classes of observations in big clusters
allclasses = c()
for (i in 1:(length(clusters))){
    cluster = clusters[[i]]
    allclasses = c(allclasses, classification[cluster])}
listclass = unique(allclasses)
# Define colors
color_cl = rainbow(length(listclass))
# Plot classes of observations in big clusters
for (i in 1:(length(clusters))){
    cluster = clusters[[i]]
    for (j in 1:(length(cluster))){
        par(new=TRUE)
        index = which(listclass == classification[cluster[j]])
        plot(Xte[cluster[j], 1], Xte[cluster[j], 2], col=color_cl[index], xlab="", ylab="", pch=19, mai
    }
}
```



Computation of preditcion strength

```
ps_clust = vector(length=length(clusters))
# Are pairs of observations in the same cluster?
for (j in 1:length(clusters)){
    cluster = clusters[[j]]
    nkj = length(cluster)
    D = 0
    for (i1 in 1:nkj){
        for (i2 in 1:nkj){
            if (i1 != i2){
                if (classification[cluster[i1]] == classification[cluster[i2]]){
                    D = D + 1
                }
            }
        }
    }
    ps_clust[j] = D / (nkj * (nkj - 1))
}
ps = min(ps_clust)
```

With a maximum distance to merge two clusters equal to 0.05, and when considering only the clusters from the test set with more than 5 elements in them, the prediction strength is equal to 1.