# LABORATORY EXPERIENCE #4 (Clustering)

## Introduction

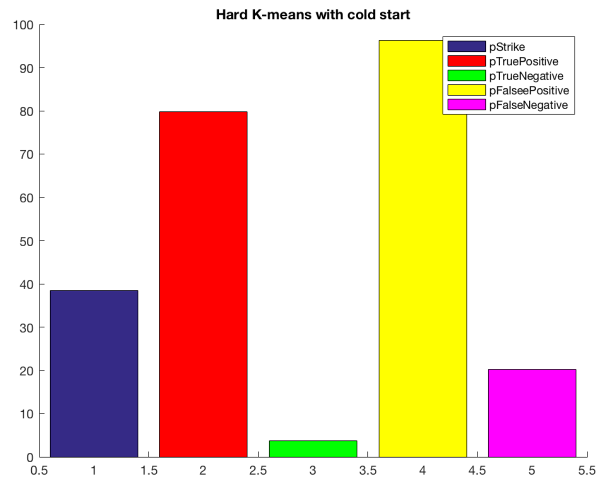
This laboratory experience required to perform **clustering algorithms** on the arrhythmia dataset studied in the previous experiment. Clustering is a class of unsupervised machine learning, where we want to map our measurements in some groups, about which we don't have neither the number, neither the characteristics of the centroids.
The algorithms to be implemented are **Hard K-means** and Soft K-means, two iterative algorithms that step-by-step update the values of the centroids of the clusters until convergence.

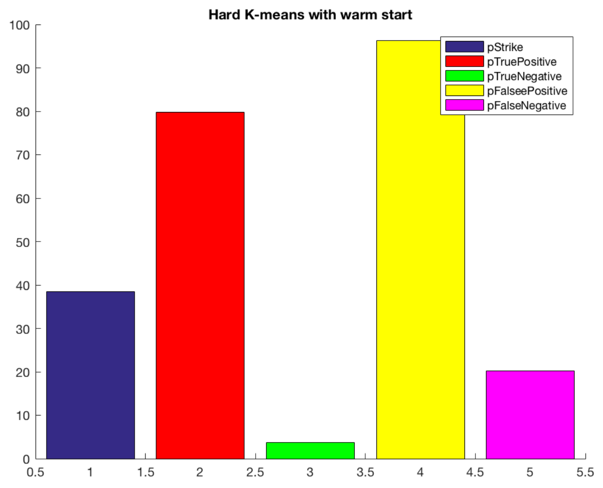
## Data preparation

The same as the previous laboratory experience.

## Algorithms implementation and results

The first step of the assignments requires to perform **Hard K-means** setting the number of clusters as two. The algorithms foresees a preliminary guess of the variables that will be updated during the iterative procedure. One of them is the vector of centroids, that are needed to start the clustering operations. Depending on the amount of information about the system we have, the algorithm can be triggered with a random guess of the centroids (**"cold start"**) or with an "aware" guess (**"warm start"**). For this purpose we divided the dataset between positive and negative clinical cases, in order to find out the **centroids** of the two classes to be used in the case of warm start.
Then, the experience was repeated both with the cold start and the warm start, imposing as stopping function a maximum number of iteration. In both cases the clustering results to be unbalanced: using a cold start we obtain different results depending on the initial guess of the centroids, but the final clusters happen to be very different in dimension. When a warm start is performed, we can compare the results with the score given by the medical doctor: the performance of the algorithm seems to be not very good, since the strike rate is under the 40% with a high rate of true positives, but also a very high rate of false positives. This result should not be necessarily interpreted as a bad implementation of the algorithm, as the clustering algorithm evaluates also features that the medical doctor is not taking into consideration and that may not be related to the disease.





Repeating the experiment with **4 clusters**, we cannot evaluate the performances in terms of specificity and sensitivity, but we could try to extract information from the elements of the clusters, for example evaluating the mean and the standard variation of their features. In particular, it may be interesting to find out if each cluster is represented by elements belonging to a specific subset of final scores, but the results didn’t show any relevant information about it.

## Table of Contents

## Data Preparation

```
clear all
close all
clc

load('arrhythmia.mat')

A=arrhythmia;

A(:, find(sum(abs(A)) == 0)) = []; % we erase the zero columns

class_id=A(:,end); % last vector of the matrix
class_id(find(class_id>1))=2; % all the values higher than 1 are put
 equal to 2
y=A;
y(:,end)=[]; % we put in y all the features but the last one
[N,F]=size(y);

%normalizing y
mean_y=mean(y,1);
stdv_y=std(y,1);

o=ones(N,1);% o is a column vector
y=(y-o*mean_y)./(o*stdv_y);% y is normalized

mean_y=mean(y,1); % checking that y matrix is properly normalized
var_y=var(y,1);


% we divide patients in two classes: with and without arrhythmia
y1=y(find(class_id==1),:); % patients without arrhythmia
y2=y(find(class_id==2),:); % patients with arrhythmias

n_healthy=sum(class_id==1);
n_ill=sum(class_id==2);

x_k(1,:)=mean(y1,1) % centroids that can be used for a "warm start"
x_k(2,:)=mean(y2,1)
```

## Hard K-Means Algorithm with warm start

```
% starting conditions
K=2; % number of clusters
```

```
var_k = ones(1,K); % we start from a set of variance values = 1
pi_k = (1/K)*ones(1,K); % we set all the prior probabilities pi_k to
 1/K

for iteration=1:7

    % evaluating the distance
    for j=1:N
        dist(j,1)=norm(y(j,:)-x_k(1,:)).^2;
        dist(j,2)=norm(y(j,:)-x_k(2,:)).^2;
    end

    % defining the regions
    R_1 = (pi_k(1)/((2*pi*var_k(1))^(N/2)))*exp(-dist(:,1)/
(2*var_k(1)));
    R_2 = (pi_k(2)/((2*pi*var_k(2))^(N/2)))*exp(-dist(:,2)/
(2*var_k(2)));

    % taking the decision
    ind_1=find(R_1>=R_2); % indexes of vectors belonging to region 1
    ind_2=find(R_1<R_2); % indexes of vectors belonging to region 2

    w_1 = y(ind_1,:); % vectors belonging to region 1
    w_2 = y(ind_2,:); % vectros belonging to region 2

    N_1 = length(ind_1);
    N_2 = length(ind_2);

    % updating the prior probabilities
    pi_k(1) = N_1 / N;
    pi_k(2) = N_2 / N;

    % updating each x_k taking the mean of the vectors assigned to the
    % corresponding region

    x_k(1,:) = mean(w_1,1);
    x_k(2,:) = mean(w_2,1);

    % updating the set of variances
    var_k(1)=0;
    var_k(2)=0;
     for i = 1:N_1
            var_k(1) = var_k(1) + norm(w_1(i,:)-x_k(1,:)).^2;
     end
    var_k(1) = var_k(1)/((N_1 - 1)*F);

     for i = 1:N_2
            var_k(2) = var_k(2) + norm(w_2(i,:)-x_k(2,:)).^2;
     end
    var_k(2) = var_k(2)/((N_2 - 1)*F);

end

n_false_negative=length(find(class_id(ind_1)==2));
```

```
n_false_positive=length(find(class_id(ind_2)==1));
n_true_negative=length(find(class_id(ind_1)==1));
n_true_positive=length(find(class_id(ind_2)==2));


p_true_positive=100*n_true_positive/n_ill;
p_true_negative=100*n_true_negative/n_healthy;
p_false_positive=100*n_false_positive/n_healthy;
p_false_negative=100*n_false_negative/n_ill;

p_strike=100*(n_true_positive+n_true_negative)/N

%mses=[p_strike,p_true_positive,p_true_negative,p_false_positive,p_false_negative;
figure
hold on
b=bar(1,p_strike);
b2=bar(2,p_true_positive,'r');
b3=bar(3,p_true_negative,'g');
b4=bar(4,p_false_positive,'y');
b5=bar(5,p_false_negative,'m');

title('Hard K-means with warm start')
legend('pStrike','pTruePositive','pTrueNegative','pFalseePositive','pFalseNegative
```

## Hard K-means algorithm with cold start

```
% starting conditions
K=2; % number of clusters
var_k = ones(1,K); % we start from a set of variance values = 1
pi_k = (1/K)*ones(1,K); % we set all the prior probabilities pi_k to
 1/K

% we can decide to start with x_k random, or taken as the mean of the
 two
% known classes. In a clustering case, the random choice is more
 realistic
rng('shuffle')
x_k = rand(K,F);


x_k

for iteration=1:7

    % evaluating the distance
    for j=1:N
        dist(j,1)=norm(y(j,:)-x_k(1,:)).^2;
        dist(j,2)=norm(y(j,:)-x_k(2,:)).^2;
    end

    % defining the regions
    R_1 = (pi_k(1)/((2*pi*var_k(1))^(N/2)))*exp(-dist(:,1)/
(2*var_k(1)));
```

```
    R_2 = (pi_k(2)/((2*pi*var_k(2))^(N/2)))*exp(-dist(:,2)/
(2*var_k(2)));

    % taking the decision
    ind_1=find(R_1>=R_2); % indexes of vectors belonging to region 1
    ind_2=find(R_1<R_2); % indexes of vectors belonging to region 2

    w_1 = y(ind_1,:); % vectors belonging to region 1
    w_2 = y(ind_2,:); % vectros belonging to region 2

    N_1 = length(ind_1);
    N_2 = length(ind_2);

    % updating the prior probabilities
    pi_k(1) = N_1 / N;
    pi_k(2) = N_2 / N;

    % updating each x_k taking the mean of the vectors assigned to the
    % corresponding region

    x_k(1,:) = mean(w_1,1);
    x_k(2,:) = mean(w_2,1);

    % updating the set of variances
    var_k(1)=0;
    var_k(2)=0;
     for i = 1:N_1
            var_k(1) = var_k(1) + norm(w_1(i,:)-x_k(1,:)).^2;
     end
    var_k(1) = var_k(1)/((N_1 - 1)*F);

     for i = 1:N_2
            var_k(2) = var_k(2) + norm(w_2(i,:)-x_k(2,:)).^2;
     end
    var_k(2) = var_k(2)/((N_2 - 1)*F);

end

n_false_negative=length(find(class_id(ind_1)==2));
n_false_positive=length(find(class_id(ind_2)==1));
n_true_negative=length(find(class_id(ind_1)==1));
n_true_positive=length(find(class_id(ind_2)==2));

p_true_positive=100*n_true_positive/n_ill;
p_true_negative=100*n_true_negative/n_healthy;
p_false_positive=100*n_false_positive/n_healthy;
p_false_negative=100*n_false_negative/n_ill;

p_strike=100*(n_true_positive+n_true_negative)/N

%mses=[p_strike,p_true_positive,p_true_negative,p_false_positive,p_false_negative;
figure
hold on
b=bar(1,p_strike);
```

```
b2=bar(2,p_true_positive,'r');
b3=bar(3,p_true_negative,'g');
b4=bar(4,p_false_positive,'y');
b5=bar(5,p_false_negative,'m');

title('Hard K-means with cold start')
legend('pStrike','pTruePositive','pTrueNegative','pFalseePositive','pFalseNegative
```

## Soft K-means algorithm

```
%
% % starting conditions
% K=2; % number of classes
% var_k = ones(1,K); % we start from a set of variance values = 1
% pi_k = (1/K)*ones(1,K); % we set all the prior probabilities pi_k to
 1/K
%
% % we can decide to start with x_k random, or taken as the mean of
 the two
% % known classes. In a clustering case, the random choice is more
 realistic
% x_k = rand(K,F);
% % x_k(1,:)=mean(y1,1);
% % x_k(2,:)=mean(y2,1);
%
%
% % evaluating the distance
% for j=1:N
%     dist(j,1)=norm(y(j,:)-x_k(1,:)).^2;
%     dist(j,2)=norm(y(j,:)-x_k(2,:)).^2;
% end
%
% % evaluating the responsibilities r1 and r2
% f1 = (pi_k(1)/((2*pi*var_k(1))^(F/2)))*exp(-dist(:,1)/(2*var_k(1)));
% f2 = (pi_k(2)/((2*pi*var_k(2))^(F/2)))*exp(-dist(:,2)/(2*var_k(2)));
% r1=f1./(f1+f2);
% r2=f2./(f1+f2);
%
% soft_ind_1=find(r1>=r2); % indexes of vectors belonging to region 1
% soft_ind_2=find(r1<r2); % indexes of vectors belonging to region 2
```

*Published with MATLAB® R2016a*

## Data Preparation

```
clear all
close all
clc

load('arrhythmia.mat')

A=arrhythmia;

A(:, find(sum(abs(A)) == 0)) = []; % we erase the zero columns
class_id=A(:,end); % last vector of the matrix
y=A;
y(:,end)=[]; % we put in y all the features but the last one
[N,F]=size(y);

%normalizing y
mean_y=mean(y,1);
stdv_y=std(y,1);

o=ones(N,1);% o is a column vector
y=(y-o*mean_y)./(o*stdv_y);% y is normalized

mean_y=mean(y,1); % checking that y matrix is properly normalized
var_y=var(y,1);


% we make a list of classes
classes = sort(unique(class_id));
C=length(classes);
```

## Hard K-Means Algorithm

```
% starting conditions
K=4; % number of classes
var_k = ones(1,K); % we start from a set of variance values = 1
pi_k = (1/K)*ones(1,K); % we set all the prior probabilities pi_k to
 1/K

% we can decide to start with x_k random, or taken as the mean of the
 two
% known classes. In a clustering case, the random choice is more
 realistic
rng('shuffle')
x_k = rand(K,F);

for iteration=1:12

    % evaluating the distance
    for n=1:N
        for k=1:K
            dist(n,k)=norm(y(n,:)-x_k(k,:)).^2;
        end
    end

    for k=1:K
        R_k(:,k) = (pi_k(k)/((2*pi*var_k(k))^(N/2)))*exp(-dist(:,k)/
(2*var_k(k)));
    end

    [M,decision]=max(R_k,[],2); % taking the decision
    %'decision' is an array of length N with the corresponding closest
 region
    % for each element (patient)

    for k=1:K
        w_k=y(find(decision==k),:);
        N_k(k) = size(w_k,1);
        pi_k(k) = N_k(k) / N;
        x_k(k,:) = mean(w_k,1);
        var_k(k)=0;
        for i = 1:N_k(k)
            var_k(k) = var_k(k) + norm(w_k(i,:)-x_k(k,:)).^2;
        end
        var_k(k) = var_k(k)/((N_k(k) - 1)*F);

        m_class(k)=mean(class_id(find(decision==k))); % trying to
 extract information from the elements of the cluster
        std_class(k)=std(class_id(find(decision==k)));
    end

end
```

*Published with MATLAB® R2016a*