

CLP Lab 7 Report

Albert Aparicio Isarn
`albert.aparicio.isarn@alu-etsetb.upc.edu`

Héctor Esteban Cabezos
`hect.esteban@gmail.com`

1. Programación de método de clasificación

El script `main_2_1` contiene el código del ejercicio 1. En la sección 4.1 se puede ver su código fuente.

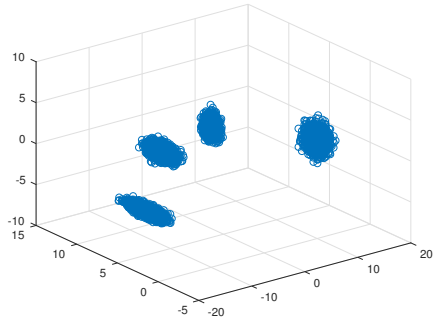
La sección 2 de este ejercicio corresponde a la función `CLP_Kmeans`, cuyo código fuente se muestra en la sección 4.3.

A continuación se muestran los resultados de este ejercicio. En la figura 1 se muestra la base de datos creada por `CLP_Generate` (código fuente en la sección 4.2) y el resultado de la clasificación con 4, 9 y 10 *clusters*.

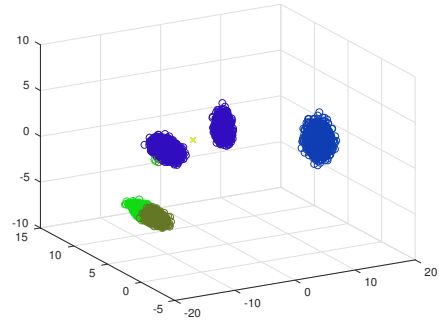
En los resultados de la clasificación de la figura 1 se pueden ver también algunas de las evoluciones de los centroides. No se ven todas porque algunos centroides quedan "ocultos" dentro de la nube de puntos del *cluster*.

El cálculo de $Trace(S_W^{-1}S_B)$ es el que más información da sobre el rendimiento del clasificador. Cuando la clasificación empeora, el valor de éste cálculo disminuye notablemente ($K = 4$ en la figura 2c), mientras que J y $Trace(S_T^{-1}S_W)$ aumentan en menor medida o se mantienen.

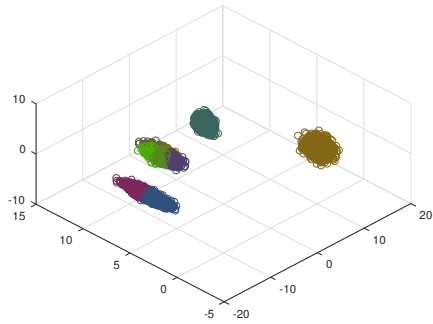
Si se usa la menor K que da mayor valor de $Trace(S_W^{-1}S_B)$, se consigue el número de clústeres óptimo para esa base de datos.



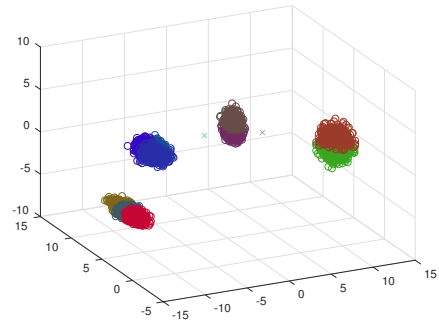
(a) Base de datos sintética



(b) 4 clusters

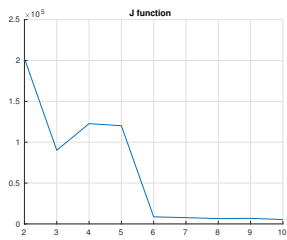


(c) 9 clusters

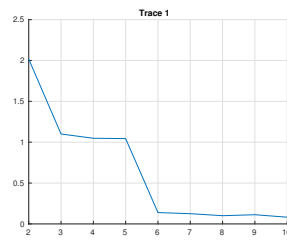


(d) 10 clusters

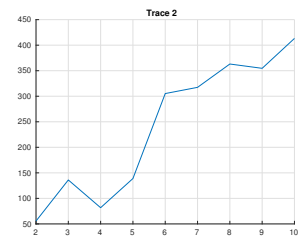
Figura 1: Clasificación de la base de datos sintética



(a) Evolución de la función de coste (J)



(b) Evolución de la métrica $Trace(S_T^{-1} S_W)$



(c) Evolución de la métrica $Trace(S_W^{-1} S_B)$

Figura 2: Evolución de las diversas métricas usadas en la clasificación

2. Cuantificación de imágenes

More text.

3. Identificación de clústeres en una base de datos propia

4. Código fuente

A continuación se encuentra el código fuente generado para la resolución de este laboratorio

4.1. main_2_1

```
%% Exercise 2.1 script of the K-Means classifier
clear
close all

% Switch to activate or deactivate the plotting of all classifiers
plot_clusters = 1;

%% Section 1
% Generate DB

% Initialize parameters
L = 4;
N = 10000;
d = 3;
th = 0.0005;

% Compute a priori probabilities for each cluster
probabilities = rand(L,1);
probabilities = probabilities./sum(probabilities);

% Generate DB samples
[DB, Nnew] = CLP_Generate(L,N,d,probabilities);

% Draw clusters
scatter3(DB(1,:), DB(2,:), DB(3,:))%, hold on

%% Section 3
% Classify with K-Means clustering

% Preallocate results from the classifier
J = cell(9,1);
minimized_J = zeros(9,1);

trace1 = zeros(9,1);
trace2 = zeros(9,1);

Sw = zeros(d,d,9);
Sb = zeros(d,d,9);

for K=2:10
    [Centroides, Labels, n , J{K-1}, trace1(K-1), trace2(K-1), ...
     Sw(:,:,K-1), Sb(:,:,K-1)] = CLP_Kmeans(DB(1:d, :),K, d, th);

    minimized_J(K-1) = J{K-1}(end);
```

```

if plot_clusters
    % Plot DB with color labeling
    figure, hold on
    for i=1:K
        c = rand(1,3);
        scatter3(DB(1,Labels==i), DB(2,Labels==i), DB(3,Labels==i),
        ...
                'MarkerEdgeColor', c/sum(c))
        scatter3(Centroides(1,i,:), Centroides(2,i,:), Centroides(3,i
        ,:),...
                'x', 'MarkerEdgeColor', 1 - c/sum(c))
    end
    grid on
    hold off
end
end

%% Section 4
% Analyze classification metrics

figure, hold on
plot(2:10, minimized_J)
grid on
title('J function')
hold off

figure, hold on
plot(2:10, trace1)
grid on
title('Trace 1')
hold off

figure, hold on
plot(2:10, trace2)
grid on
title('Trace 2')
hold off

% Trace 2 is great, Trace 1 not good, because it decays constantly. It
% does so
% because as we increase the number of clusters, each of them is more
% compact,
% so the "within" metric improves.
%
% The J function behaves sometimes roughly like Trace 1
%
% We want to minimize Trace 1 and maximize Trace 2
%
% Trace 2 increases greatly when different clusters are classified as
% such,
% while Trace 1 does slightly decrease in the same situation.
%

```

```
% Because of this, Trace 2 is the best metric
../src/main_2_1.m
```

4.2. CLP_Generate

```
function [ DB, N_new ] = CLP_Generate( L, N, d, priori_prob )
%CLP_Generate Generate a synthetic gaussian database
%
% [ DB, Nnew ] = CLP_Generate( L, N, d, probabilities )
%
% This function generates L clusters of gaussian-distributed d-
% dimensional
% vectors.
%
% The L clusters contains a number of samples <= N. The total number
% of samples (N_new) may be less than N. This could happen due to the
% rounding
% process in which each cluster is assigned a number of vectors,
% following the
% a priori probabilities vector (priori_prob).
%
% INPUTS:
%
% =====
%
% - L: Number of clusters to be generated
% - N: Total number of samples to be generated
% - d: Number of dimensions in the vectors
% - priori_prob: Vector of a priori probabilities for the clusters
%
% Notice that L == length(priori_prob)
%
% OUTPUTS:
%
% =====
%
% - DB: Matrix containing the generated database (first d rows contain
% the
%       vector components, last row contains the labels)
% - N_new: Actual total number of generated vectors (N_new <= N)
%
% TODO: Insert license notice

%% Check the input parameters for errors
assert(length(priori_prob) == L, strcat('Input parameter size mismatch.',
...
    'Please, make sure the vector priori_prob has L parameters'))

%% Initialize parameters
% Compute the number of samples in each cluster
values = round(N*priori_prob);

% Compute the total number of samples in the generated clusters
```

```

N_new = sum(values);

% Preallocate mean and variance matrices
matriu_mitjes = zeros(d,L);
matriu_sigma = zeros(d,d,L);

% The last row contains labeling information
DB = zeros(d +1, N_new);

% Initialize the range of the clusters' mean
a = -10;
b = 10;

%% Compute cluster vectors
index = 1;
for i=1:L
    % Compute means of the current cluster
    matriu_mitjes(:,i) = (b-a).*rand(d,1) + a;
    % Compute variances of the current cluster
    matriu_sigma(:,:,i) = diag(rand(d,1));
    % Get random values for the current cluster
    DB(1:d, index:index+values(i)-1) = ...
        mvnrnd(matriu_mitjes(:,i), matriu_sigma(:,:,i), values(i))';
    % Save "label" of current cluster
    DB(d+1, index:index+values(i)-1) = i;
    % Increase the index (it is used to access the columns in the DB that
    % correspond to a single cluster
    index = index+values(i);
end

% Shuffle database columns
DB=DB(:,randperm(length(DB)));
end

```

../src/CLP_Generate.m

4.3. CLP_Kmeans

```

function [Centroids, Labels, n, J, tr1, tr2, Sw, Sb] = CLP_Kmeans(DB, K,
    d, th)
%CLP_Kmeans Classify matrix with a K-Means algorithm
%
% [ Centroids, Labels, n, J, tr_1, tr_2, Sw, Sb ] = CLP_Kmeans(DB, K, d
% )
%
% Detailed explanation goes here

%% Initialize centroids and labels matrices

Centroids = datasample(DB, K, 2, 'Replace', false);

%% Classify database
% threshold = 0.0005;

```

```

Labels = zeros(length(DB), 1);
n = 1;

J = zeros(50,1);

% Iterate while the cost function variates enough
condition = n <= 2;
while condition == 1

    J(n) = 0; % Cost function

    % Classify database
    for i = 1:length(DB)
        norms = sum(abs(repmat(...
            double(DB(:,i)), 1, K) - double(Centroids(:, :, end))).^2, 1);
        [Minimum_value, Labels(i)] = min(norms);
        J(n) = Minimum_value + J(n);
    end

    % Update centroids
    for i = 1:K
        Centroids(:, i, n) = mean(double(DB(:, Labels==i)), 2);
    end

    if n > 1
        diff = J(n-1) - J(n);
        condition = (diff) > th;
    end

    n = n+1;
end

% Take actual values of J (eliminate the rests of preallocated data)
J = J(1:n-1);

%% Compute error metrics
% Within-cluster scatter matrix
Sw = zeros(d,d);
ni = zeros(1,K);

for i = 1:length(DB)
    Sw = Sw + (double(DB(:,i))-double(Centroids(:,Labels(i),end)))*...
        (double(DB(:,i))-double(Centroids(:,Labels(i),end)))';
    ni(Labels(i)) = ni(Labels(i)) + 1; % Add one sample to detected class
end

% Between-cluster scatter matrix
Sb = zeros(d,d);

for j = 1:K
    m = (1/length(DB))*ni*double(Centroids(:, :, end))';
    Sb = Sb + ni(j)*(double(Centroids(:, j, end))-m)*(double(Centroids(:, j

```



```

        ,end)) -m'))';
end

% Total scatter matrix
St = Sb+Sw;

% Trace metrics
tr1 = trace(St\Sw);
tr2 = trace(Sw\Sb);

end

```

../src/CLP_Kmeans.m