COURSE MM 3.1

MULTIMEDIA DOCUMENTS INDEXING AND RETRIEVAL

# Mini-Test:
# Clustering with K-means

**Lecturer:** Dr. DOAN Nhat Quang
*USTH ICTLab*

**Student:** VU Anh Tuan
Quentin Viotto

Hanoi, October 21, 2015

# Contents

# List of Figures

# 1 Clustering with K-means

**Q.1.** *Write a function* `M = distance(X, C)` *which computes Euclidean distance between vectors of the matrix* `X` *and the matrix* `C` *is formed by* `K` *centroids.*

Given:

- a matrix `X` of points which composes `N` vector of `d` dimension: $X = (x_1\ x_2\ \dots\ x_N)^T \in \mathbb{R}^{N*d}$ where $x_i \in \mathbb{R}^d$.
- a matrix `C` of centroids which composes `K` vector of `d` dimension: $C = (w_1\ w_2\ \dots\ w_K)^T \in \mathbb{R}^{K*d}$ where $w_i \in \mathbb{R}^d$.

The Euclidean distance matrix `M` between `X` and `C` composes $N * K$ elements, where an element in the row $i$ and the column $k$ is the Euclidean distance between the point $x_i$ and the centroid $w_k$:

$$\begin{aligned} M_{ik} &= \sqrt{\sum_{j=1}^{d}(x_{ij} - w_{kj})^2} \\ &= \sqrt{(x_i - w_k) * (x_i - w_k)^T} \end{aligned} \tag{1.1}$$

Therefore, the formula 1.1 is used to compute elements of the distance matrix `M` from matrices `X` and `C`. The code source of the function `distance` is shown in the A.1.

**Q.2.** *Write a function* `list = assignment(M)` *which returns the list of clusters to which* `N` *vectors are assigned.*

We can see that elements of a row `i` in the distance matrix `M` are distances from the point $x_i$ to centroids $w_k$ $(k = \overline{1, K})$ respectively. As we assign a point to the closest centroid, a point $x_i$ hence belongs to the cluster $k$, where $d(x_i, w_k)$ is minimum value in the row `i` of the distance matrix `M`.

The code source to find the list of clusters to which points are assign is shown in the code A.2.

**Q.3.** *Compute the intra-class error* `E` *of the clusters which is minimized by K-means.*

The intra-class error `E` is the total of intra-distance of clusters, where an intra-distance

of a cluster is the total distances of points in that cluster to its centroid.

$$E = \sum_{k=1}^{K} \sum_{i \in C_k} d(x_i, w_k)$$
$$= \sum_{i=1}^{N} d(x_i, w_k) \tag{1.2}$$

where $w_k$ is the centroid of cluster in which $x_i$ belongs to.

Therefore, the formula 1.2 is used to compute the intra-class error. For simply, we compute the intra-class error in the function `assignment`. The code soure is shown in the code A.2.

**Q.4.** *Write a function* `C = newCentroid(X, list)` *which re-calculates new centroids according to the new assignments to new clusters.*

After assignment of points to clusters, the centroid of clusters is re-calculated. The value of new centroid of a cluster is mean value of points belongs to that cluster.

The code source is shown in the code A.3. In this code we add more 2 parameters: `K` - the number of cluster, and `cC` - current centroids used to assign to the new centroids in the initial step.

**Q.5.** *Write a function* `[C, list, E] = kmeans(X, K)` *where* `X` *is the input data,* `K` *is the number of clusters.*

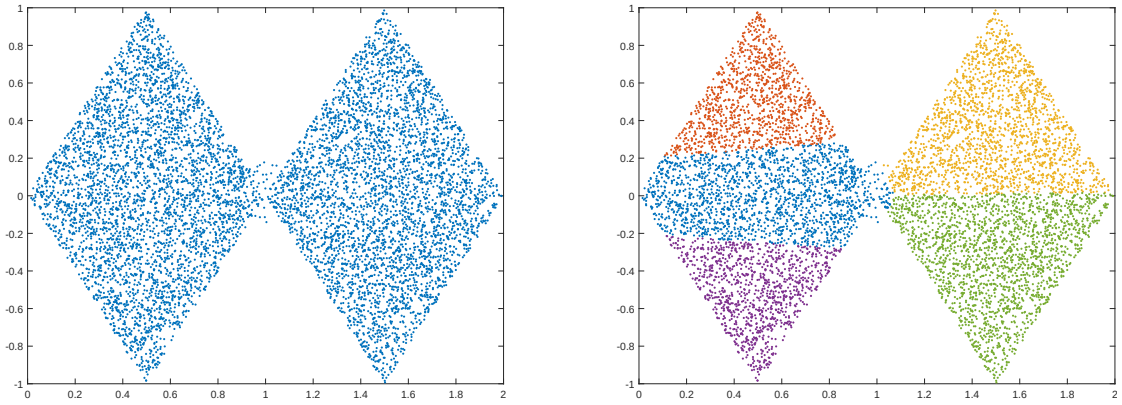
The function `kmeans` realizes clustering algorithm K-means on the dataset `X` with `K` clusters. In our implementation, we use 2 stopping criterion: number of iterations and intra-class error. Therefore, two more parameters are added to this function:

- *threshold*: if the change of intra-class error of two consecutive iterations is less than this value, the algorithm is stopped.
- *maxn*: the maximum number of iterations. The algorithm is stopped after this number of iterations regardless of the change of intra-class error.

The code source is shown in the code A.4.

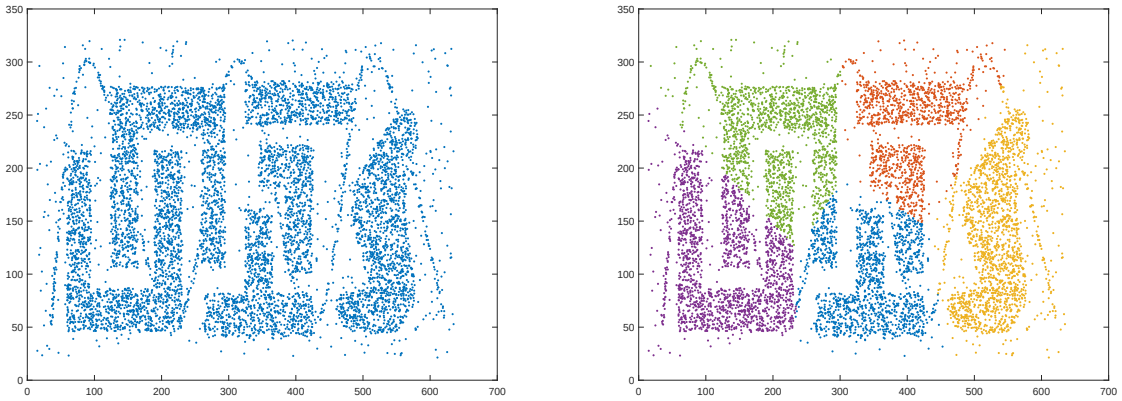
**Q.6.** *Test this function with many datasets.*

As all testing datasets has points of 2 dimensions, we create a function named plotClusters in order to plot dataset before and after clustering. Then, we use script in the code A.6 is to test K-means algorithm with given datasets. Results of tests is shown in figures: fig.1, fig.2, fig.3, fig.4 and fig.5.



(a) before clustering (b) after 5-means clustering

Figure 1: Test with dataset `ds2.dat`



(a) before clustering (b) after 5-means clustering

Figure 2: Test with dataset `ds3.dat`

**Q.7.** *Compute different quality criteria.*

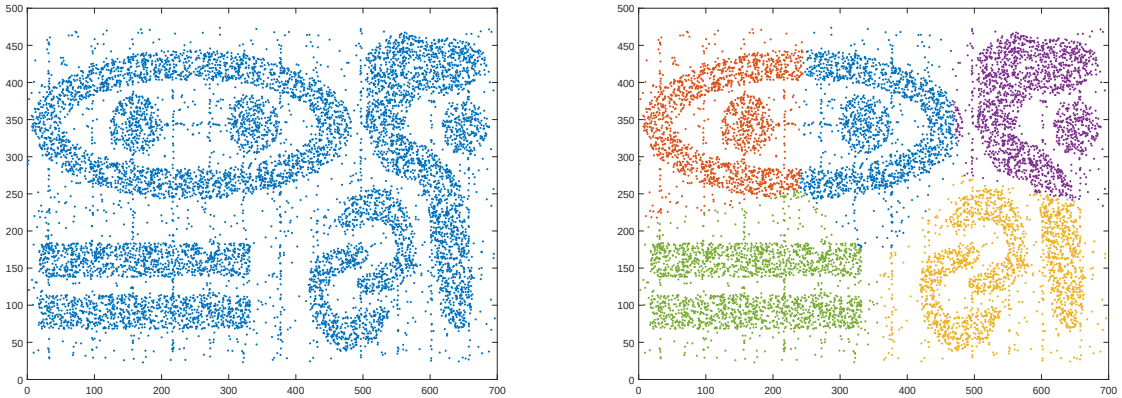
As do not have any class labels to analyze how close is a clustering to a reference, we use internal validity measures: Dunn index and Davies-Bouldin index.

**a) Dunn Index:** measure dense and well-separated clusters. It is computed by the formula:

$$Dunn = \min_{i=1...K}\left(\min_{j=1...K, j\neq i}\left(\frac{d(w_i, w_j)}{\max_{k=1...K}\Delta(C_k)}\right)\right) \tag{1.3}$$

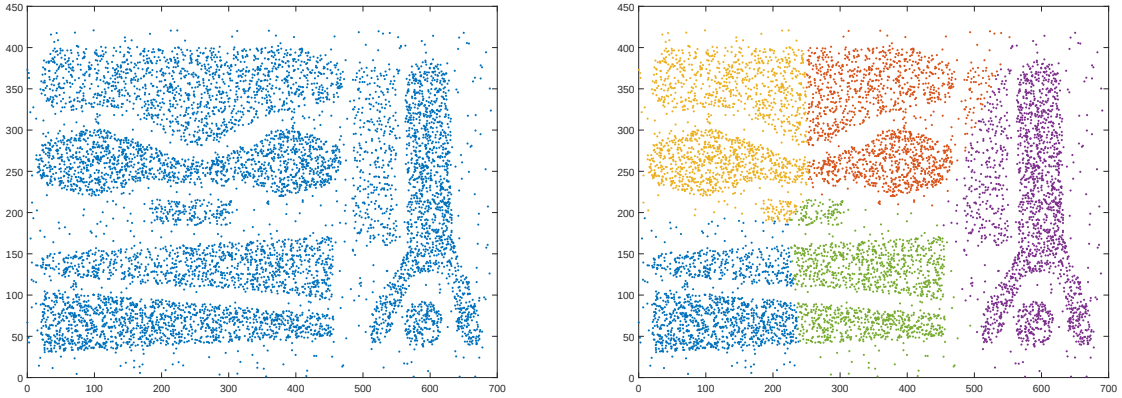
where: $\Delta(C_k)$ is the maximum distance between two points in the cluster $k$.

Based on the formula 1.3, some functions are used to compute Dunn index of a clustering. First of all, function getCluster is written to get a cluster. Then, this function is used in the function Delta to find the maximum distance between two points in a cluster. Function distanceClusters is used to create a distance matrix between two clusters. Lastly, function DunnIndex uses two functions Delta and distanceClusters to find the Dunn index of clustering.



(a) before clustering (b) after 5-means clustering

Figure 3: Test with dataset `ds4.dat`



(a) before clustering (b) after 5-means clustering

Figure 4: Test with dataset `ds5.dat`

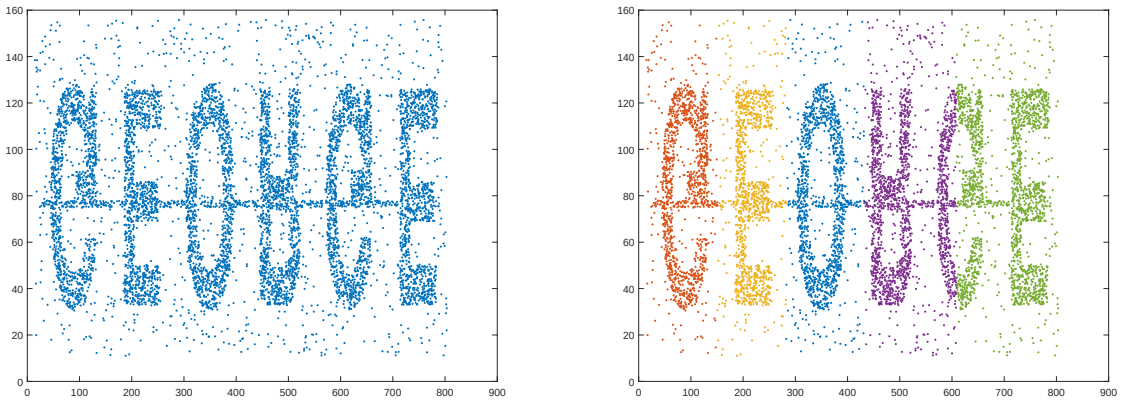
**b) Davies-Bouldin Index:** measures the correlation between two clusters. It is computed by the formula:

$$DB = \frac{1}{K}\sum_{i=1}^{K} \max_{k=1\ldots K, k\neq i} \frac{\mu_i + \mu_k}{d(w_i, w_k)} \tag{1.4}$$

where: $\mu_k$ is means distance between points in the cluster $k$ and its centroid $w_k$.

$$\mu_k = \frac{1}{n_k} \sum_{j=1}^{n_k} d(x_j, w_k)$$

Based on the formula 1.4, some function are built to compute Davies-Bouldin index of a clustering. To begin with, the function computeMu is written to compute $\mu$ (distance between points and its centroid) of clusters. Then we construct the function getMax to returns $Fmax$ of clusters. Where $Fmax$ of a cluster `i` is maximum value of fractions $\frac{\mu_i + \mu_k}{d(w_i, w_k)}$ $(k = \overline{1, K}, k \neq i)$. Lastly, these functions are used in the function DBIndex to find the Davies-Bouldin index of a clustering.

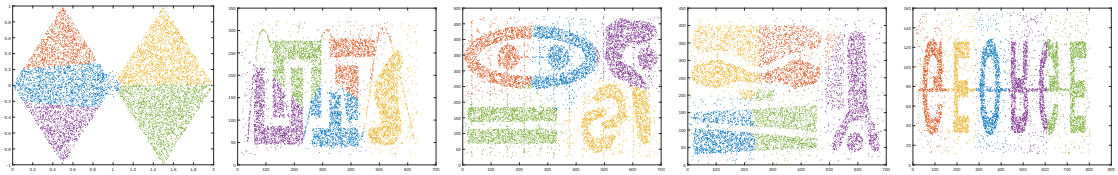


(a) before clustering (b) after 5-means clustering

Figure 5: Test with dataset `george.dat`

**Q.8.** *Explain why K-means gives bad results on some datasets.*

To evaluate the result of clustering we use two criterion: Davies-Bouldin index and Dunn index which are presented in the Q.7. All datasets are tested with the same parameters of threshold (0.01), maximum number of iterations (200) and number of clusters (K = 5). Obtained clusterings are shown in the figure 6.



(a) `ds2` (b) `ds3` (c) `ds4` (d) `ds5` (e) `george`

Figure 6: Result of clustering with 5-means

**a) Evaluation by using Davies-Bouldin index**. The testing script is shown in the code A.14. The result is shown in the figure 7. We know that the Davies-Bouldin

index measures the correlation between two clusters. The formula 1.4 shows that the smaller index is, the better of distinctness of clusters is. Therefore, the clustering gets the best partition.

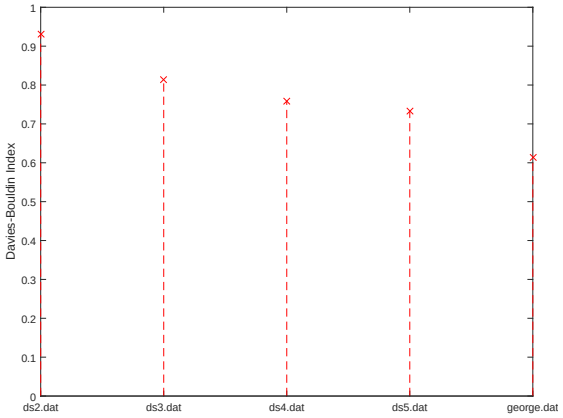



Figure 7: Evaluation by using Davies-Bouldin index

Result in the figure 7 shows that the clustering of the dataset `ds2` is the worst, and the one of the dataset `george` is the best. It is easy to understand. By observing, we easily find that the similarity between clusters of the dataset `ds2` is very high, otherwise the one of the dataset `george` is low.

**b) Evaluation by using Dunn index**. The testing script is shown in the code A.15. The result is shown in the figure 8. By observing the result of clustering in the figure 6, we can see that the separation between clusters of the dataset `ds3` is blur, the dense of points in a cluster is also low. Therefore, the Dunn index of the dataset `ds3` is the worst as the Dunn index measures the dense and well-separated clusters. The higher Dunn index is, the better of separation and dense is.

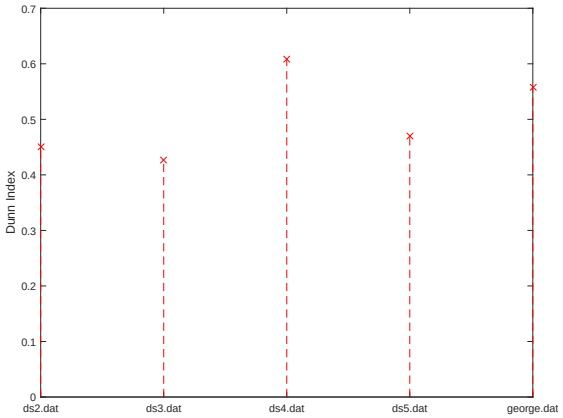



Figure 8: Evaluation by using Dunn index

# Code Source



## A.1. function `distance`

This function returns an Euclidean distance matrix between points `X` and centroids `C`.

```
function M = distance(X, C)

N = size(X, 1); K = size(C, 1);
M = zeros(N, K);

for i=1:N
    for k = 1:K
        T = X(i,:) - C(k,:);
        M(i,k) = sqrt(T*T');
    end
end
```

Code A.1: function `distance`

## A.2. function `assignment`

This function returns the list of clusters to which points are assigned.

```
function [list, E] = assignment(M)

[N, ~] = size(M);
list = zeros(N,1); % list of clusters
E = 0; % intra-class error

for i = 1:N
    [d_ik, list(i)] = min(M(i,:));
    % d_ik: the distance between the point x_i and the centroid w_k
    % of cluster in which it belongs to
    E = E + d_ik;
end
```

Code A.2: function `assignment`

## A.3. function `newCentroid`

This function returns new centroids according to the assignment of clusters.

```
function C = newCentroid(X, list, K, cC)

C = cC; % initially, new centroids are the same current centroids
for k = 1:K
    I = find(list==k); % indexes of points belong to cluster k
    if ~isempty(I)
        C(k,:) = mean(X(I,:));
    end
end
```

Code A.3: function `newCentroid`

## A.4. function `kmeans`

This function returns centroids, list of clusters in which points are assigned and intra-class error after K-means algorithm is realized.

```
function [C, list, E] = kmeans(X, K, threshold, maxn)

[~, d] = size(X);
C = rand(K, d);
E = 0; i = 1;

while 1
    M = distance(X, C);
    [list, E2] = assignment(M);
    C2 = newCentroid(X, list, K, C);
    ch = abs(E2-E); % change of intra-class error
    E = E2; C = C2; i = i+1;
    if (ch < threshold) || (i > maxn)
        break;
    end
end
```

Code A.4: function `kmeans`

## A.5. Test with different dataset

We use function `plotClusters` below to plot a dataset of 2 dimensions before and after k-means clustering.

```
function plotClusters(X, K, list, dname)

figure; plot(X(:,1), X(:,2),'.');
pos = strfind(dname, '.');
fname = strcat(dname(1:pos), 'eps');
print(fname, '-depsc');
```

```
7
8  figure;
9  for k = 1:K
10     I = find(list==k); % indexes of points belong to cluster k
11     if ~isempty(I)
12         Xk = X(I,:); % points belong to cluster k
13         plot(Xk(:,1), Xk(:,2),'.'); hold on;
14     end
15 end
16 fname = sprintf('-%d.', K);
17 fname = strcat(dname(1:pos-1), fname, 'eps');
18 print(fname, '-depsc');
```

Code A.5: function `plotClusters`

In the script below, we realize a test with 5 different datasets. The number of cluster is set to 5, the maximum number of iterations is 200 and the threshold to stop iteration is 0.01.

```
1  clc; clear; close all;
2
3  datasets = {'ds2.dat', 'ds3.dat', 'ds4.dat',...
4      'ds5.dat', 'george.dat'};
5  K = 5; threshold = 0.01;
6  maxn = 200; % maximum number of iteration
7  for i = 1:length(datasets)
8      X = load(datasets{i});
9      [~, list, ~] = kmeans(X, K, threshold, maxn);
10     plotClusters(X, K, list, datasets{i});
11 end
```

Code A.6: test script

## A.6. Dunn Index

The function `getCluster` below returns all points in a cluster.

```
1  function Xk = getCluster(X, list, k)
2
3  I = find(list==k); % indexes of points belong to cluster k
4  if ~isempty(I)
5      Xk = X(I,:); % points belong to cluster k
6  else
7      Xk = [];
8  end
```

Code A.7: function `getCluster`

The function `Delta` returns the maximum distance between two points in a cluster.

```
function d = Delta(X, list, k)

Xk = getCluster(X, list, k);
if ~isempty(Xk)
    M = distance(Xk, Xk);
    d = max(M(:));
else
    d = -1;
end
```

Code A.8: function `Delta`

The function `distanceClusters` returns a matrix distance between two clusters.

```
function D = distanceClusters(C)

D = distance(C, C);
for i = 1:size(C,1)
    D(i,i) = inf;
end
```

Code A.9: function `distanceClusters`

The function `DunnIndex` returns Dunn index of a clustering.

```
function di = DunnIndex(X, list, C)

maxD = -1; % max delta
for k = 1:size(C, 1)
    d = Delta(X, list, k);
    if maxD < d
        maxD = d;
    end
end

D = distanceClusters(C);
D = D./maxD;
di = min(D(:));
```

Code A.10: function `DunnIndex`

## A.7. Davies-Bouldin Index

The function `computeMu` returns mean distances of clusters.

```
function Mu = computeMu(X, list, C)

K = size(C, 1);
Mu = zeros(K, 1);
```

```
for k = 1:K
    Xk = getCluster(X, list, k);
    if ~isempty(Xk)
        M = distance(Xk, C(k,:));
        Mu(k) = mean(M);
    end
end
```

Code A.11: function `computeMu`

The function `getMax` returns $Fmax$ of clusters. Where $Fmax$ of a cluster `i` is maximum value of fractions $\frac{\mu_i + \mu_k}{d(w_i, w_k)} \quad (k = \overline{1, K}, k \neq i)$.

```
function Fmax = getMax(Mu, C)

K = size(C,1);
D = distance(C, C);
Fmax = zeros(K, 1);
for i = 1:K
    for k = 1:K
        if k ~= i
            f = (Mu(i) + Mu(k))/D(i,k);
            if Fmax(i) < f
                Fmax(i) = f;
            end
        end
    end
end
```

Code A.12: function `getMax`

The function `DBIndex` returns Davies-Bouldin index of a clustering.

```
function db = DBIndex(X, list, C)

Mu = computeMu(X, list, C);
Fmax = getMax(Mu, C);
db = sum(Fmax)/size(C, 1);
```

Code A.13: function `DBIndex`

## A.8. Evaluation

The script below is to evaluate the result of clustering by using Davies-Bouldin index.

```
clc; clear; close all;

datasets = {'ds2.dat', 'ds3.dat', 'ds4.dat',...
    'ds5.dat', 'george.dat'};
```

```
K = 5; threshold = 0.01;
maxn = 200;
len = length(datasets);
DB = zeros(len, 1);

for i = 1:len
    X = load(datasets{i});
    [C, list, E] = kmeans(X, K, threshold, maxn);
    DB(i) = DBIndex(X, list, C);
end
figure; stem(DB,'xr--');
set(gca,'XTick',1:5);
set(gca,'xticklabel',datasets); ylabel('Davies-Bouldin Index');
print('db.eps', '-depsc');
```

Code A.14: evaluation by using Davies-Bouldin index

The script below is to evaluate the result of clustering by using Dunn index.

```
clc; clear; close all;

datasets = {'ds2.dat', 'ds3.dat', 'ds4.dat',...
    'ds5.dat', 'george.dat'};
K = 5; threshold = 0.01;
maxn = 200;
len = length(datasets);
DI = zeros(len, 1);

for i = 1:len
    X = load(datasets{i});
    [C, list, E] = kmeans(X, K, threshold, maxn);
    DI(i) = DunnIndex(X, list, C);
end
figure; stem(DI,'xr--');
set(gca,'XTick',1:5);
set(gca,'xticklabel',datasets); ylabel('Dunn Index');
print('di.eps', '-depsc');
```

Code A.15: evaluation by using Dunn index

# Bibliography

[1] Jean-Marc Ogier, DOAN Nhat Quang, NGHIEM Thi Phuong. *Course Multimedia Documents Indexing and Retrieval.* University of Science and Technology of Hanoi (USTH), October, 2015.