ID2222 – Data Mining

Assignment 4 - Graph Spectra

Course Coordinator:
Vladimar Vlassov & Sarunas Girdzijauskas

Teaching assistants:
Edward Tjörnhammer
Kambiz Ghoorchian
Mohamed Gabr

Group 2

Óttar Guðmundsson

Örn Arnar Karlsson

2018-12-2

Description

Clustering data is a really common and known technique used in data analysis. Data points are divided into several groups so that points in the same group are similar to each other and different from other groups.

In this assignment, we present an algorithm called Spectral clustering. It has become popular due to its simple implementation using linear algebra and good performance for graph-based clustering.

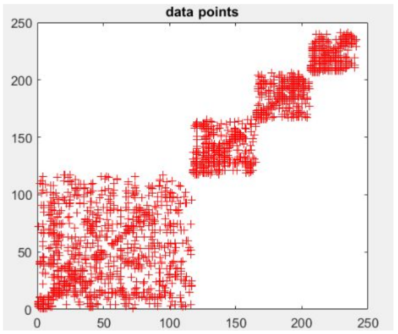
The steps taken are as follows. First, an adjacency matrix is created to find out the degree of each node in the graph. Then, the first k eigenvectors of its Laplacian matrix are computed to construct a feature vector for each object. Finally, k-means is processed on these features to separate the data points into k many classes.

How to run

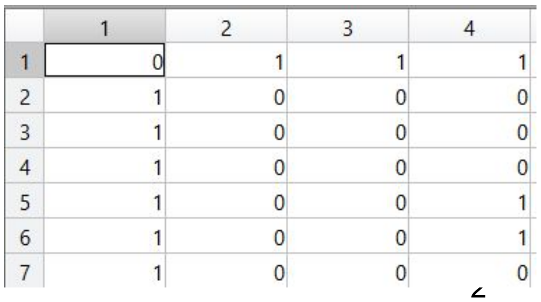
We wrote our solution in Matlab. To run the code, simply import the matlab2.mat file inside Matlab and make sure the files example1.dat and example2.dat are in the same folder. Select “Run section to compute all variables and plot up the clusters, based on the N that can be configured in the code.

Solution

To demonstrate our solution, we used the first data set provided (example1.dat). By plotting up the initial graph, we thought that four clusters could easily be identified. All code in ***italicbold*** is a Matlab code.

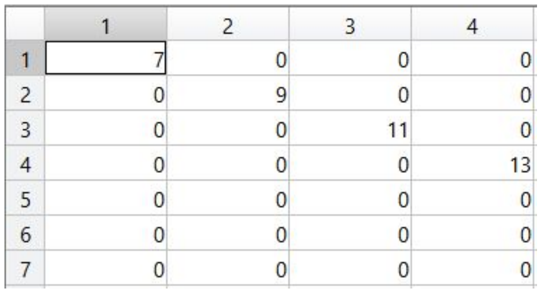


Using the provided code to create the adjacency matrix, we created a matrix where a connection between nodes existed



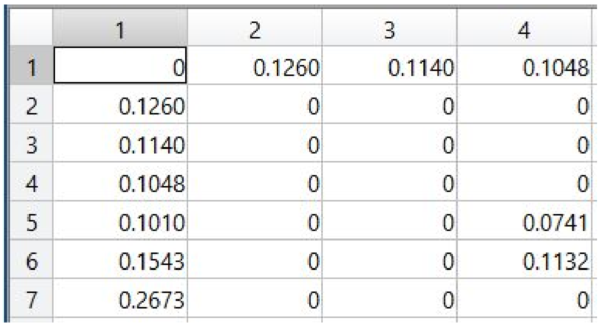
	1	2	3	4
1	0	1	1	1
2	1	0	0	0
3	1	0	0	0
4	1	0	0	0
5	1	0	0	1
6	1	0	0	1
7	1	0	0	0

Now we needed to implement our code. The first thing was to create a diagonal matrix that was the sum of all connections for every node, presented in a diagonal matrix.
D = diag(sum(A, 1));



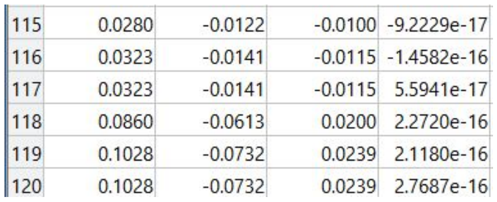
	1	2	3	4
1	7	0	0	0
2	0	9	0	0
3	0	0	11	0
4	0	0	0	13
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0

As both A and D were now constructed, we could easily implement the Laplacian matrix with the given formula.
L = D^(-0.5) * A * D^(-0.5)



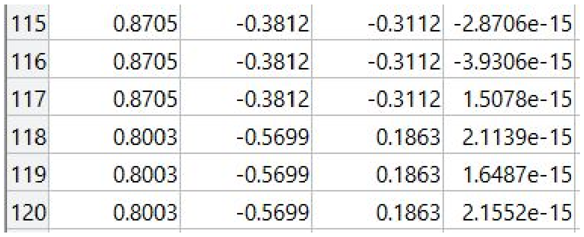
	1	2	3	4
1	0	0.1260	0.1140	0.1048
2	0.1260	0	0	0
3	0.1140	0	0	0
4	0.1048	0	0	0
5	0.1010	0	0	0.0741
6	0.1543	0	0	0.1132
7	0.2673	0	0	0

Now to compute the eigenvectors, we used the simple inbuilt method in Matlab to compute at K biggest eigenvectors, resulting in this array.
K = 4;[X, xE] = eigs(L, K);



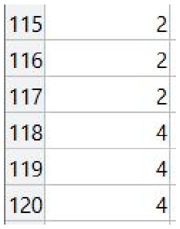
115	0.0280	-0.0122	-0.0100	-9.2229e-17
116	0.0323	-0.0141	-0.0115	-1.4582e-16
117	0.0323	-0.0141	-0.0115	5.5941e-17
118	0.0860	-0.0613	0.0200	2.2720e-16
119	0.1028	-0.0732	0.0239	2.1180e-16
120	0.1028	-0.0732	0.0239	2.7687e-16

The only thing left was to form the matrix Y, which was created by renormalizing the eigenvectors to a unit length
```
Y = zeros(size(X));
for i=1:max_ids
        for j=1:K
                Y(i,j) = X(i,j)/ norm(X(i,:));
        end
end
```



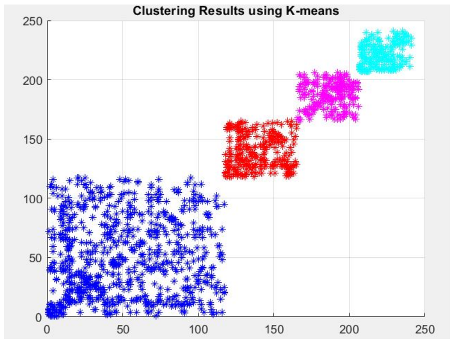
115	0.8705	-0.3812	-0.3112	-2.8706e-15
116	0.8705	-0.3812	-0.3112	-3.9306e-15
117	0.8705	-0.3812	-0.3112	1.5078e-15
118	0.8003	-0.5699	0.1863	2.1139e-15
119	0.8003	-0.5699	0.1863	1.6487e-15
120	0.8003	-0.5699	0.1863	2.1552e-15

Finally, calling kmeans on the normalized eigenvectors we compute the ID's of the clusters, grouping together the most similar eigenvectors of unit length.
[idx, ~] = kmeans(Y, K);



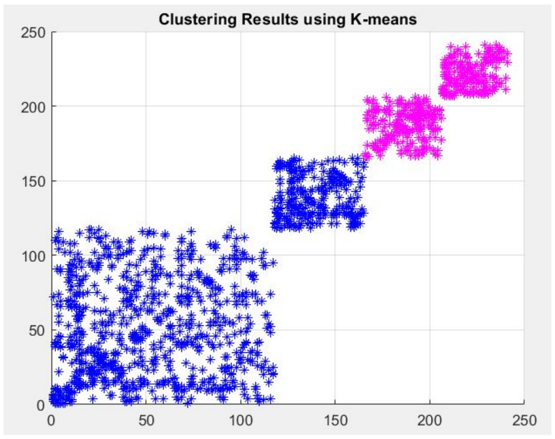
115	2
116	2
117	2
118	4
119	4
120	4

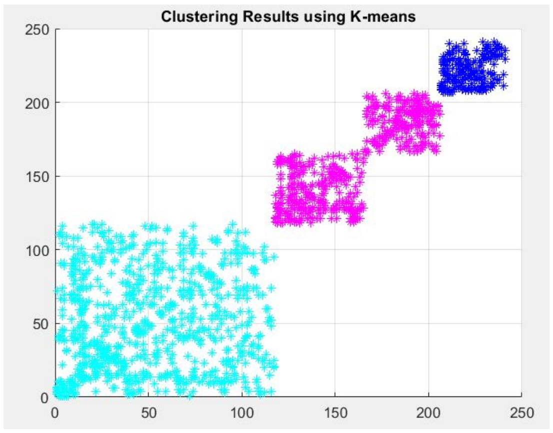
So now by plotting up the same graph as before, but changing the color of it based on the kmeans array, we end up with four different clusters.

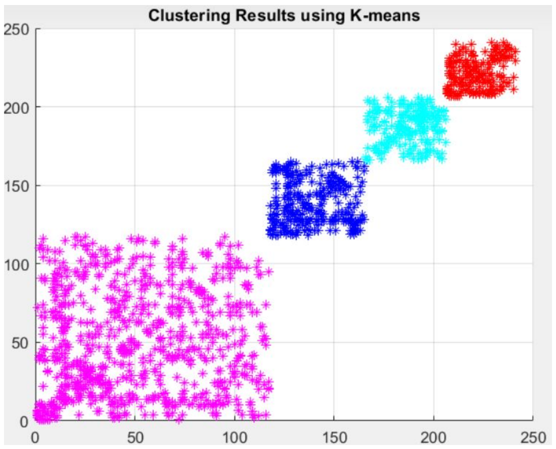


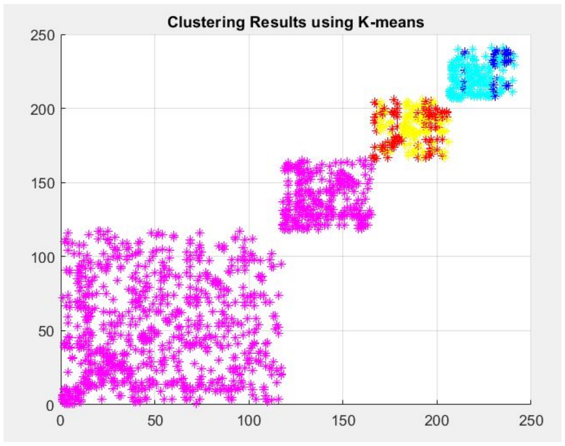
Results

Example1.dat (K= 2,3,4 and 5 respectively)



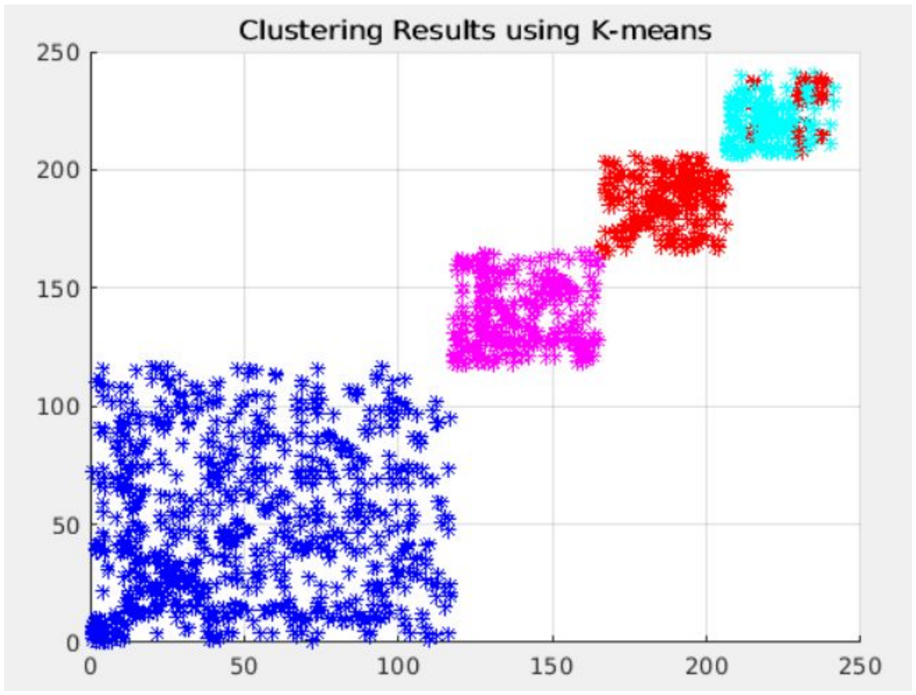






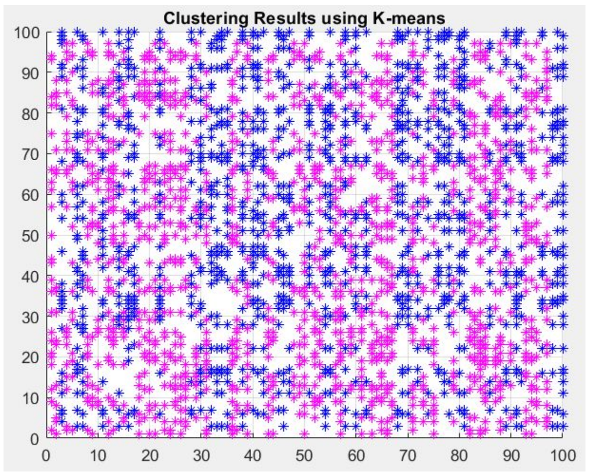
We can clearly see that for different size of k, we manage to cluster similar data points together. However, as k size gets bigger than the actual clusters, we see worse performance.

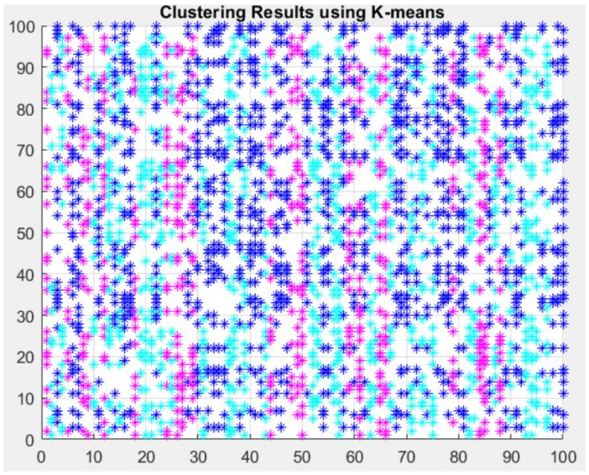
Note that in some cases for k=4, we got this result

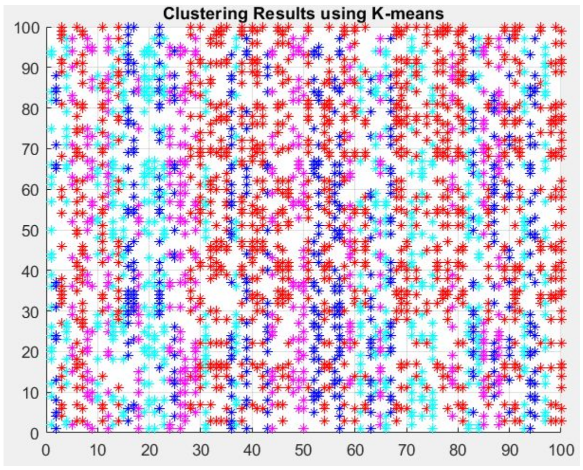


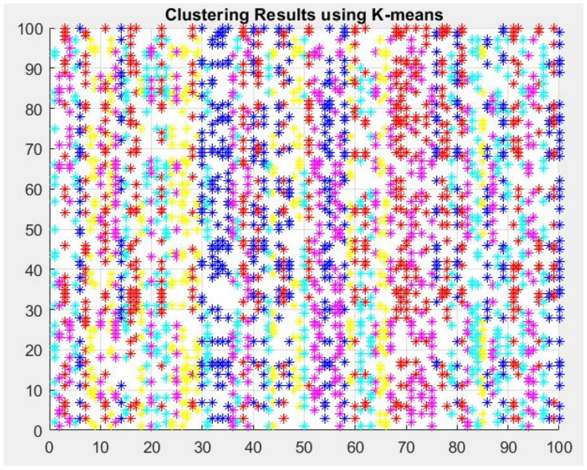
As one can see, the results from this matrix are not really good cluster representation. This was due to the randomness of the k-means algorithm applied to the normalized eigenvectors, as the starting point of the cluster was random for every iteration. This can be countered by finding good initial positions for the k-means algorithm by using something more sophisticated such as self-organizing neural network maps or a radial basis function, but this is outside the scope of this assignment.

Example2.dat (K= 2,3,4 and 5 respectively)









For the second data set, no size of K seemed to work out for us. It seemed like this data set was very different and based on its connectivity, there were no actual well-defined clusters.