The Spectral Clustering

Nataliya Sokolovska

Sorbonne University
Paris, France

Outline

Spectral Clustering

Implementation

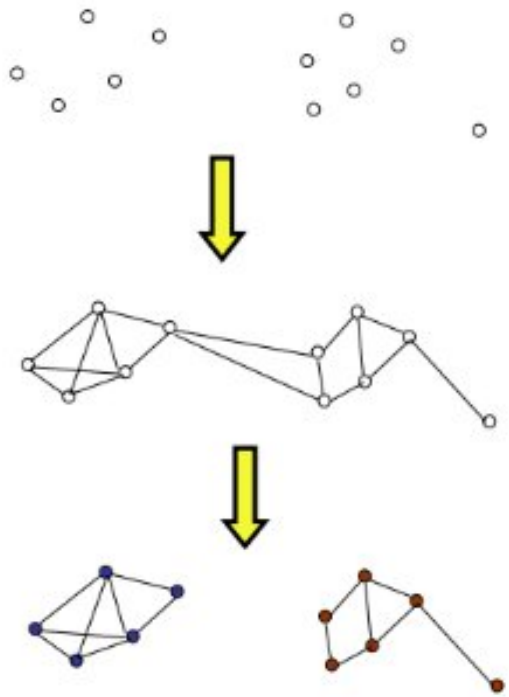
Silhouette score

Spectral Clustering

- *U. von Luxburg, "A tutorial on spectral clustering", Stat. Comp., 2007*
- One of the most popular clustering algorithms
- It can be proved that under very mild conditions, spectral clustering algorithms are statistically consistent. This means that is we assume that the data has been sampled randomly according to some probability distribution from some underlying space, and if we let the sample size increase to infinity, then the results of clustering converge (these results do not necessary hold of unnormalized spectral clustering).

Graph notation and similarity graphs

If we do not have more information than similarities between data points, a nice way of representing the data is in form of **similarity graph**. The vertices represent the data points. Two vertices are connected if the similarity between the corresponding data points is positive (or larger than a certain threshold), and the edge is weighted by the similarity.



Graphs and Cluster Assumption

The problem of clustering: we want to find a partition of the graph such that the edges between different groups have a very low weight.

"Cluster assumption": two points are likely to have the same class label if there is a path connecting them passing through regions of high density only. Or, the decision boundary should lie in regions of low density.

Graph notations

- $G = (V, E)$ is an undirected graph
- the graph is weighted: each edge between two vertices v_i and v_j has a weight $w_{ij} > 0$
- The weighted adjacency matrix W ($w_{ij} = 0$ mean that the vertices are not connected)
- Graph is undirected, $w_{ij} = w_{ji}$
- The degree of a vertex v_i is defined as $d_i = \sum_{j=1}^{n} w_{ij}$
- The degree matrix D

Graph notations Cont'd

- A subset of vertices A
- Two ways of measuring the size of A
  - $|A|$ – the number of vertices in A
  - $vol(A) = \sum_{i \in A} d_{ij}$ – measure the size of A by the weights of its edges
- a subset A is connected is any two vertices in A cab be joined by a path such that all intermediate points also lie in A.

Different similarity graphs (used in Spectral Clustering)

There are several popular constructions to transform a given set of data points into a graph. Most of them lead to a sparse representation $\Rightarrow$ computational advantages.

- The ϵ-neighborhood graph. We connect all points whose pairwise distances are smaller than ϵ. Usually considered as an unweighted graph.
- k-nearest neighbor graphs. We connect vertex v_i with vertex v_j if v_j is among the k nearest neighbors of v_i.
- The fully connected graph. We connect all points with positive similarity with each other, and we weight the edges by s_{ij}. The graph should model the local neighborhood relationships. An example of similarity function is the Gaussian similarity function $s(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$. The parameters σ controls the width of the neighborhoods.

Graph Laplacians

- The main tool for spectral clustering are graph Laplacian matrices
- In the literature, there is no unique convention which matrix exactly is called "graph Laplacian"
- The unnormalized graph Laplacian matrix is defined as

$$L = D - W$$

.

- The normalized Laplacian

$$L = D^{-1/2}(D - W)D^{-1/2}$$

Properties of L

- For every vector $f \in \mathbb{R}^n$ we have

$$f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

- L is symmetric and positive semi-definite
- The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector 1.
- L has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$

Unnormalized Spectral Clustering

- Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct
  - Construct a similarity graph; W is its weighted adjacency matrix
  - Compute the unnormalized Laplacian L
  - Compute the first k eigenvectors $v_1, \ldots, v_k$ of L.
  - Let $V \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $v_1, \ldots, v_k$ as columns
  - For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of V
  - Cluster the points $(y_i)_{i=1,\ldots,n} \in \mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$
- Output: Clusters $A_1, \ldots, A_k$.

Normalized Spectral Clustering (Shi and Malik, 2000)

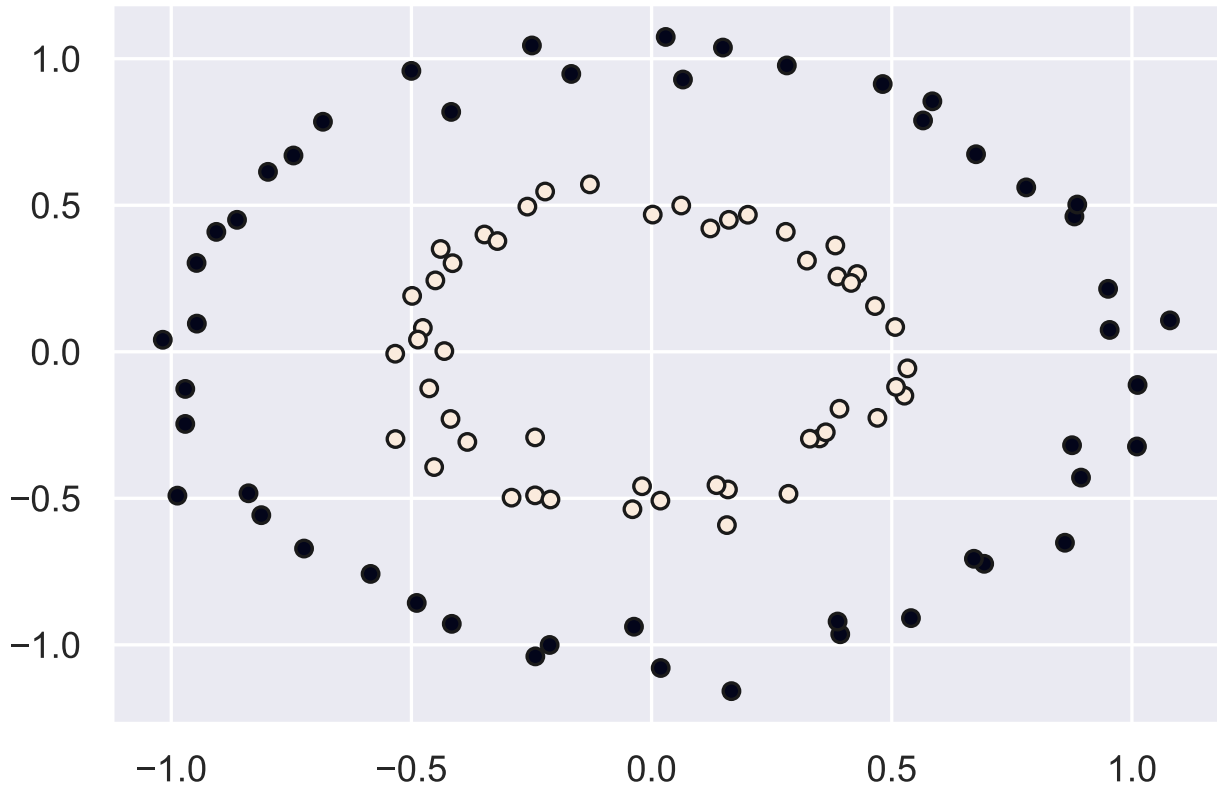
- Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct
  - Construct a similarity graph; W is its weighted adjacency matrix
  - Compute the unnormalized Laplacian L
  - Compute the first k eigenvectors $v_1, \ldots, v_k$ of the generalized eigenproblem $Lv = \lambda Dv$.
  - Let $V \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $v_1, \ldots, v_k$ as columns
  - For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of V
  - Cluster the points $(y_i)_{i=1,\ldots,n} \in \mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$
- Output: Clusters $A_1, \ldots, A_k$.

Normalized spectral clustering (Ng, Jordan, and Weiss, 2002)

- Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct
  - Construct a similarity graph; W is its weighted adjacency matrix
  - Compute the normalized Laplacian L_{sym}
  - Compute the first k eigenvectors $v_1, \ldots, v_k$ of L_{sym}.
  - From the matrix $U \in \mathbb{R}^{n \times k}$ from V by normalizing the row sums to have norm 1, that $u_{ij} = v_{ij} / (\sum_k v_{ik}^2)^{1/2}$
  - For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of V
  - Cluster the points $(y_i)_{i=1,\ldots,n} \in \mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$
- Output: Clusters $A_1, \ldots, A_k$.

Let us consider an example

A typical case where the Spectral Clustering outperforms other methods

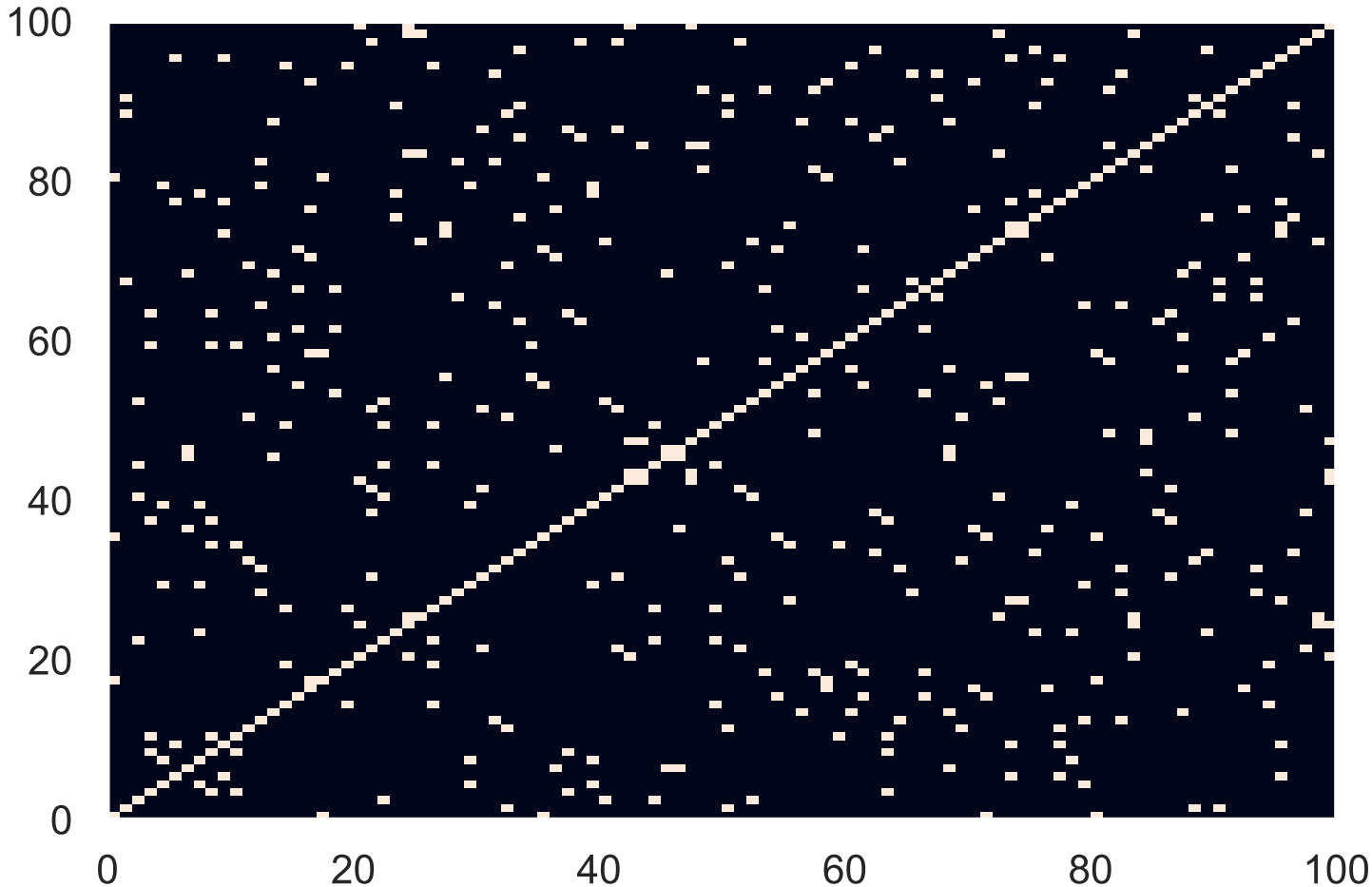


Number of points = 100, number of clusters = 2

Constructing the Adjacency Matrix

```
def constructAdjMatrix(dataFrame, k_neighbors):

    m = NearestNeighbors(n_neighbors =
     k_neighbors, algorithm='ball_tree')

    m.fit(dataFrame.values)

    distArray, indArray = m.kneighbors(dataFrame.values)

    affinity = m.kneighbors_graph(dataFrame.values)

    return  affinity.toarray()
```

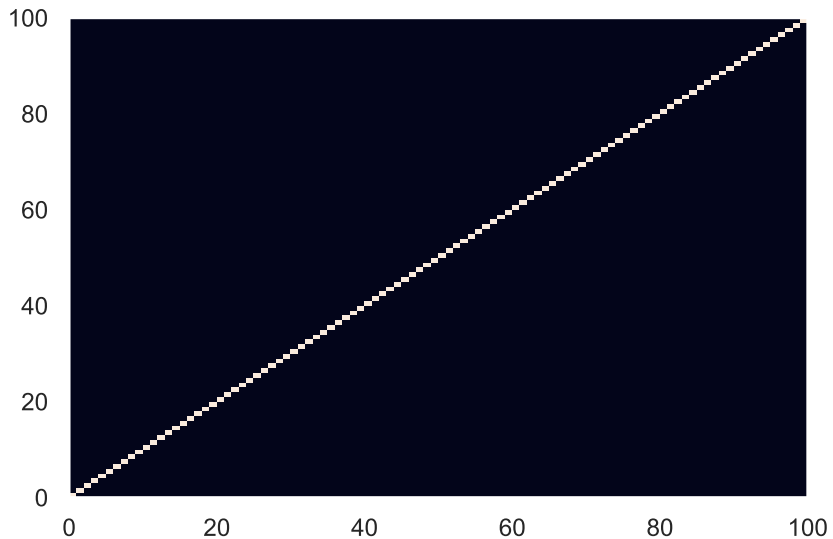
An adjacency matrix from our example



Constructing the degree matrix

```
def constructDegreeMatrix(adjMatrix):
    return  np.diag([sum(row) for row in adjMatrix])
```

If the number of the nearest neighbours is fixed to k, then the degree matrix contains k on its diagonal.

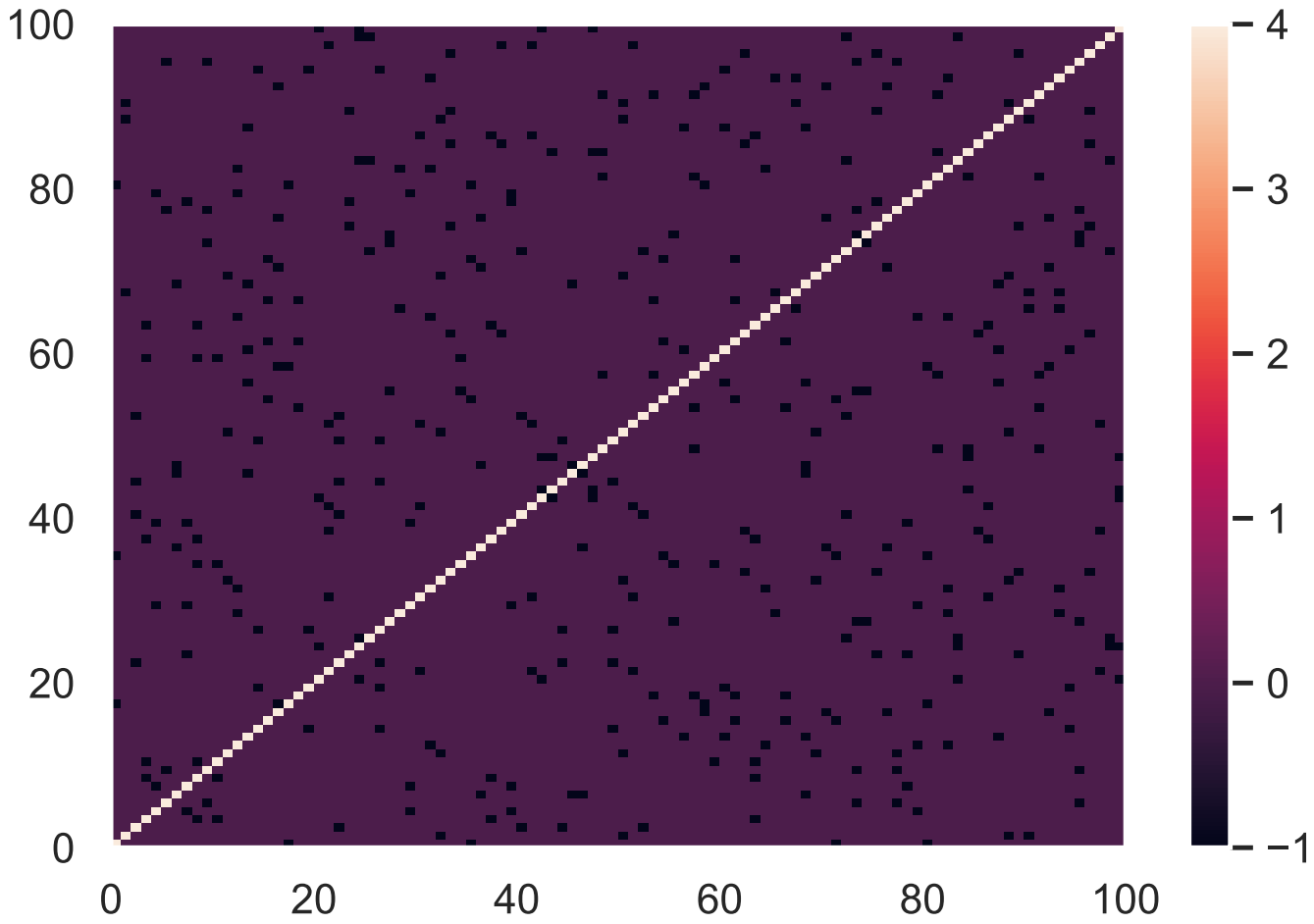


Construct the Laplacian matrix

- There is a number of variants
- Here we compute an unnormalised Laplacian matrix

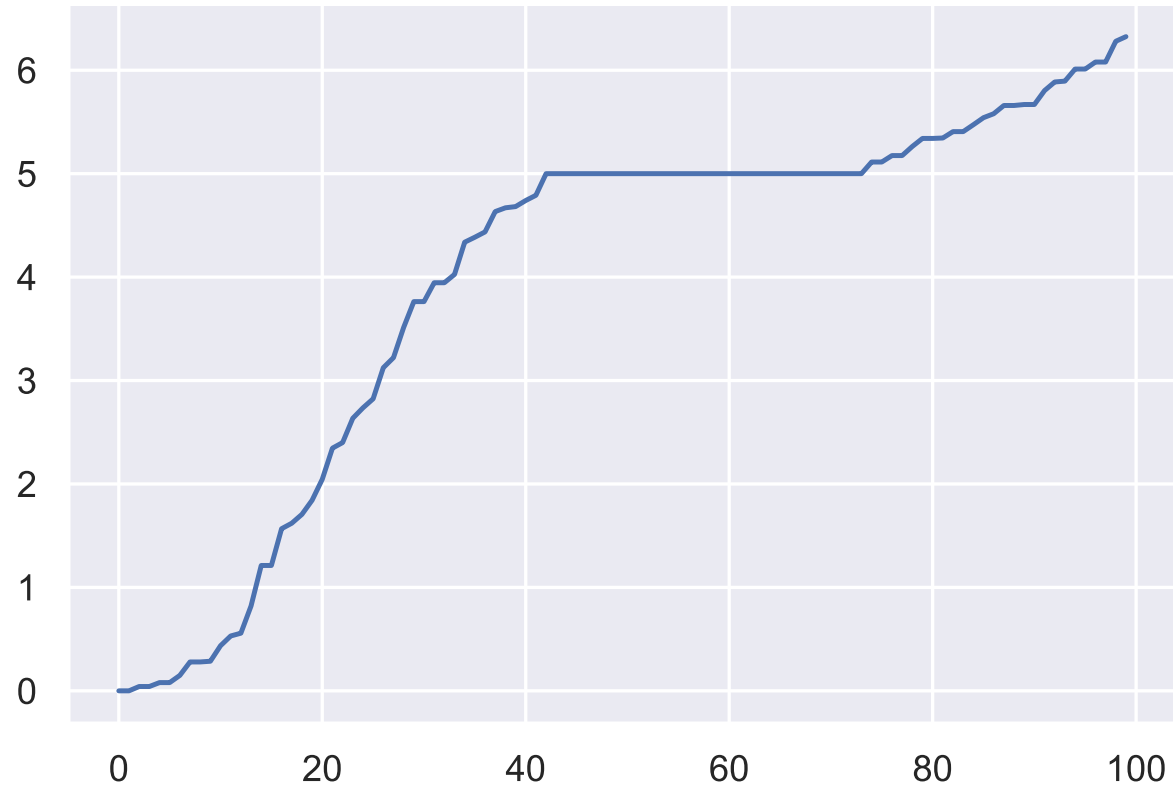
```
def constructLaplacianMatrix(degreeMatrix, adjMatrix):
    return degreeMatrix - adjMatrix
```

The Laplacian from our data



Eigenstructure decomposition of the Laplacian

```
eigenValues, eigenVectors = np.linalg.eig(laplacian)
sortedInd =  np.argsort(eigenValues.real)
```



The sorted eigenvalues

Sorting the eigenvectors

```
eigenValues, eigenVectors = np.linalg.eig(laplacian)
sortedInd =  np.argsort(eigenValues.real)

sortedEigenVectors = eigenVectors[:,sortedInd].real
sortedEigenVectors = sortedEigenVectors[:,:2]
```

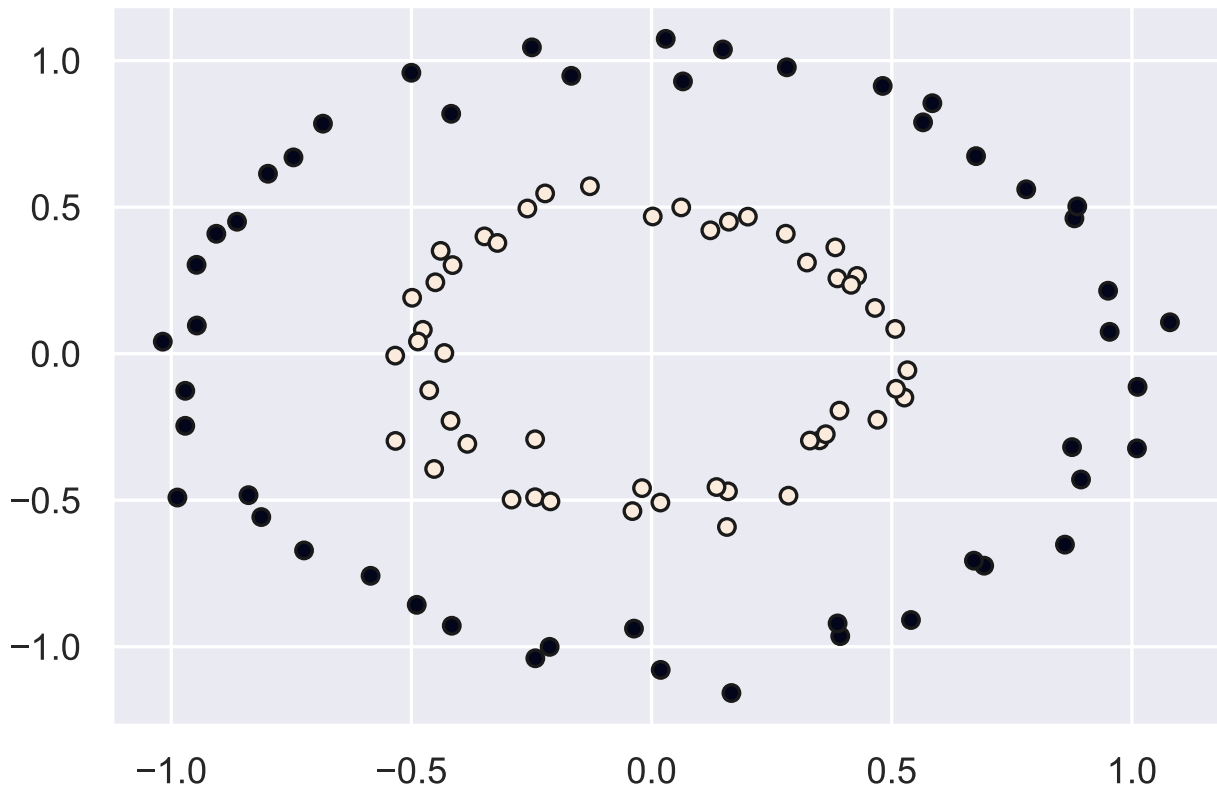
- We get the re-arranges eigenvectors, corresponding to the sorted eingenvalues

Running a clustering algorithm (k-means) on the newly obtained data

```
km=KMeans(init='k-means++', n_clusters=2)
km.fit(sortedEigenVectors)
print(km.labels_)
```

- The number of clusters should be fixed
- How many eingenvectors to choose?
  - Number of clusters
  - Eigengap (elbow) method

What we are supposed to obtain



Problems to solve:

- Implement the Spectral clustering in Object Oriented Python

  `class spectralClustering:`

  `....`
- Introduce your eigenstructure implementation in the code
- Introduce your k-means implementation
- Test the sklearn package

Silhouette score: to what extent a point belongs to its cluster

- *Rousseeuw, 1987* introduced a measure of partitioning (does not depend on the number of clusters)
- Definition:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

  Describes the degree of belonging to its cluster by comparing the average distance to instances of its cluster with the average distance to the nearest cluster; $s(i)$ is independent of K (number of classes), we consider the distance to the nearest (cluster) neighbor!

- Interpretation:
  - $s(i) \to 1$: the point is well-clustered
  - $s(i) \approx 0$: the point is between 2 clusters
  - $s(i) \to -1$: the point is closer to another cluster than to its cluster

Silhouette score: definition

- Average distance of the point i to the points of its cluster (the number of points in the cluster is n_a):

$$a(i) = \frac{1}{n_a - 1} \sum_{j=1, j \neq i}^{n_1} d(i,j)$$

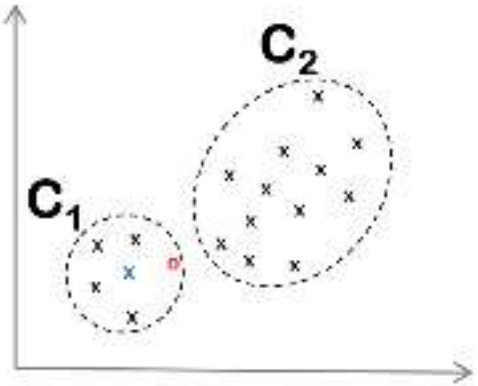
- Average distance of the point i to the points of another cluster (number of points in this other cluster n_k):

$$d(i, C_k) = \frac{1}{n_k} \sum_{j=1}^{n_k} d(i,j)$$

- Distance to the closest cluster:

$$b(i) = \min_{k \neq a} d_{i, C_k}$$

Silhouette score: examples



$s(x) > s(0)$: x is more central (it is even the center of the cluster C_1); o is relatively close to cluster C_2

- For one cluster (compactness and separability):

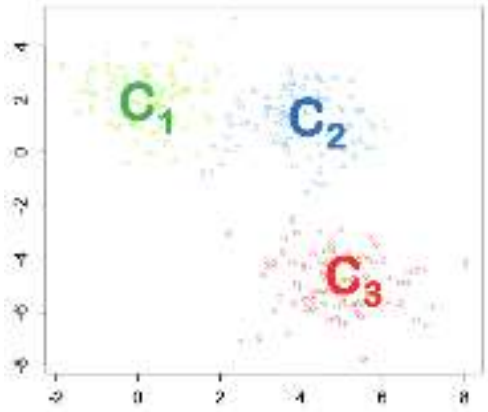
$$\bar{s}_k = \frac{1}{n_k} \sum_{i \in C_k} s(i)$$

- For the whole partition (K clusters):

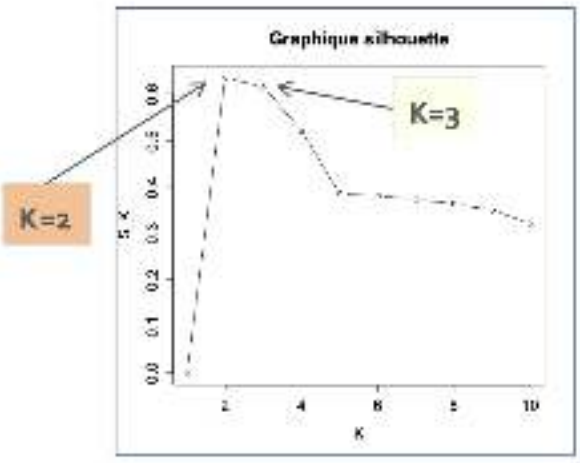
$$C_K = \frac{1}{n} \sum_{k=1}^{K} n_k \bar{s}_k$$

Figures from `http://eric.univ-lyon2.fr/~ricco/cours/slides/classif_k_medoides.pdf`

Silhouette score: an example



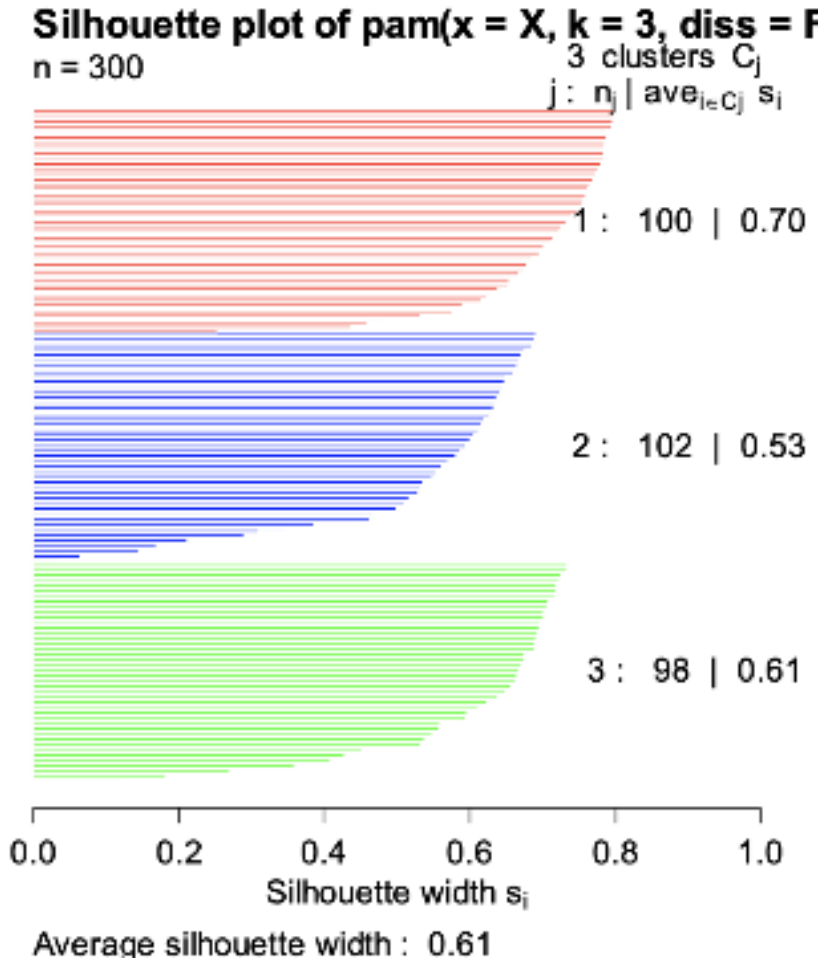
The silhouette values for each cluster and for the global repartition:
$\bar{s}_1 = 0.6, \bar{s}_2 = 0.53, \bar{s}_3 = 0.7,$
$S_{K=3} = 0.61.$

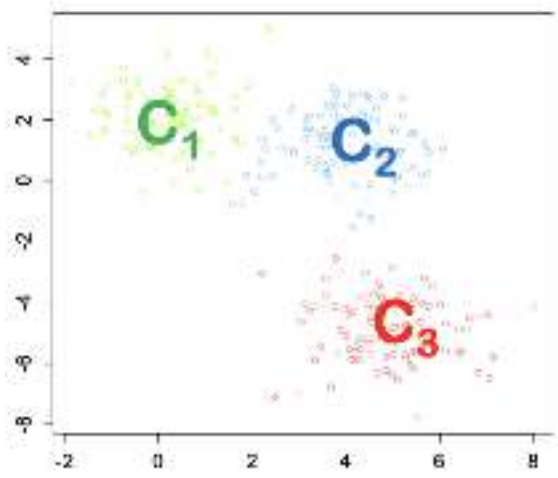


To find an optimal K, test different number of clusters. The silhouette value here identifies optimal $K = 2$.

Figures from `http://eric.univ-lyon2.fr/~ricco/cours/slides/classif_k_medoides.pdf`

Graphical representation of a silhouette





Figures from `http://eric.univ-lyon2.fr/~ricco/cours/slides/classif_k_medoides.pdf`