Spectral clustering

Contents:

- Matrices from the similarity graph
- 1D spectral embedding & clustering
- Unnormalized and normalized spectral clustering
- Important choices

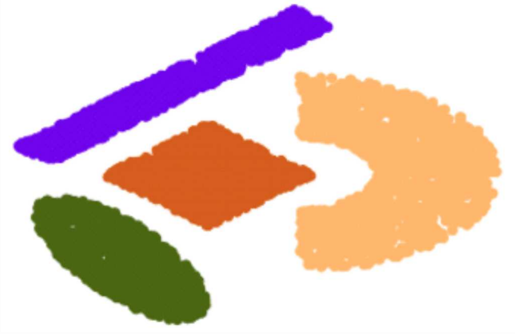
Book: Sections 2.4.4.3, 6.7, 19.3.4

Recommended external material:
von Luxburg (2007): A Tutorial on Spectral Clustering.

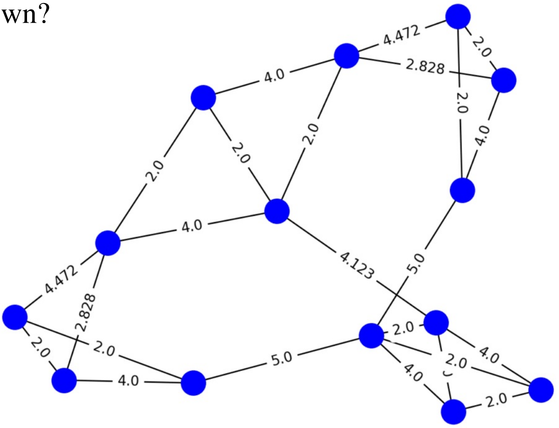
Presemo: `https://presemo.aalto.fi/mdm2023`

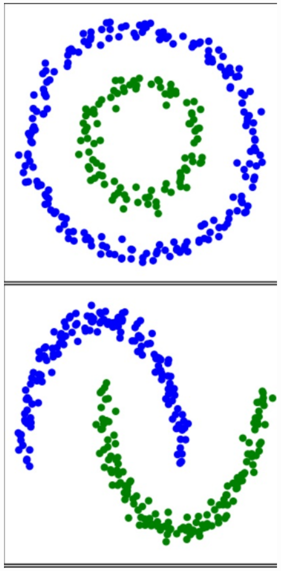
Recap: How could you cluster these?

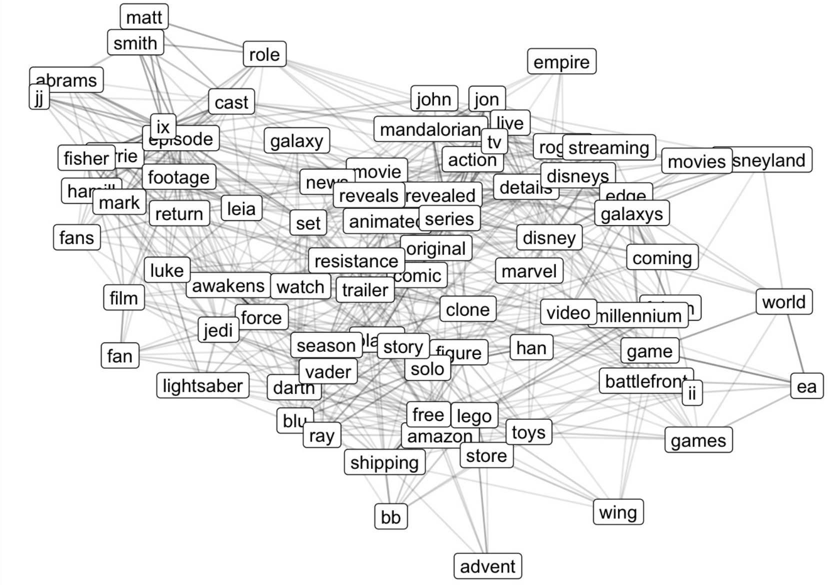
Arbitrary shapes?



Only similarity graphs known?







Images: White (2019) https://www.markhw.com/blog/word−similarity−graphs, Cooper (2021) https://spin.atomicobject.com/2021/09/07/spectral−clustering/, Park & Kim (2020) https://doi.org/10.1115/DETC2020−22642, Scikit−learn documentation https://ogrisel.github.io/scikit−learn.org/sklearn−tutorial/auto_examples/cluster/plot_cluster_comparison.html

General idea of graph-based clustering

1. Present data as a similarity (neighbourhood) graph $\mathbf{G}$
2. Cluster nodes of G with a network clustering or community detection algorithm

+ can detect arbitrary-shaped clusters

+ even varying cluster densities (given k nearest neighbour similarity graph)

+ for any data type (if pairwise similarity/distance defined)

– computationally costly

– many parameter choices

Spectral clustering: Idea

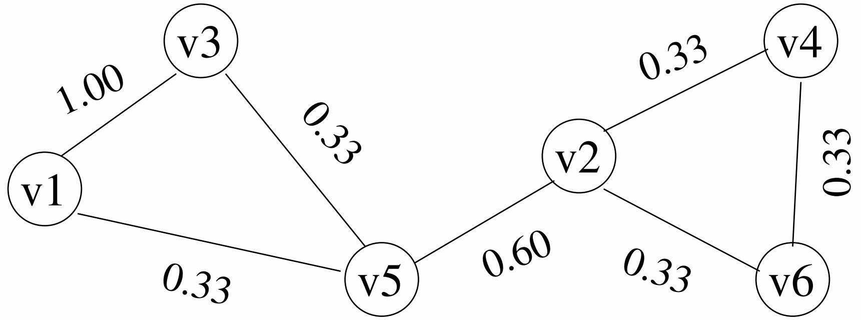
1. Create **similarity graph G**
   - node v_i for the ith data point ($i = 1, \ldots, n$)
   - edge weight w_{ij} = similarity between nodes v_i and v_j
2. **Present data in** (low-dimensional) **vector space** (i.e., find vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n$) such that **local similarity/clustering structure** is preserved
   - idea: choose $\mathbf{Y}$ to minimize $cost(\mathbf{G}, \mathbf{Y}) = \sum \sum w_{ij} L_2^2(\mathbf{y}_i, \mathbf{y}_j)$
   - intuition: large w_{ij} tends to produce small $d(\mathbf{y}_i, \mathbf{y}_j)$
   - → easy after reformulation with a Laplacian matrix
3. **Cluster $\mathbf{y}_i$**s with K-means (etc.)

What is needed?

From $\mathbf{G}$ derive:

1. weight matrix $\mathbf{W}$
2. diagonal degree matrix $\mathbf{\Lambda}$
3. Laplacian matrix $\mathbf{L} = \mathbf{\Lambda} - \mathbf{W}$
4. normalized Laplacian matrices $\mathbf{L}_{rw}$, $\mathbf{L}_{sym}$ if desired)

Similarity graph and weight matrix $\mathbf{W}$



$$\begin{bmatrix} 0.00 & 0.00 & 1.00 & 0.00 & 0.33 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.33 & 0.60 & 0.33 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.33 & 0.00 \\ 0.00 & 0.33 & 0.00 & 0.00 & 0.00 & 0.33 \\ 0.33 & 0.60 & 0.33 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.33 & 0.00 & 0.33 & 0.00 & 0.00 \end{bmatrix}$$

- $\mathbf{W}$ adjacency matrix of a weighted graph
- $W_{ij} = w_{ij}$ (similarity between nodes v_i and v_j)
- if unweighted graph, use weights 1 (edge) or 0

Diagonal degree matrix $\boldsymbol{\Lambda}$ *(*$\Lambda_{ii} = \sum_{j=1}^{n} W_{ij}$*)*

$$\boldsymbol{\Lambda} = \begin{bmatrix} 1.33 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.26 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.33 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.66 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.26 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.66 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 0.00 & 0.00 & 1.00 & 0.00 & 0.33 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.33 & 0.60 & 0.33 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.33 & 0.00 \\ 0.00 & 0.33 & 0.00 & 0.00 & 0.00 & 0.33 \\ 0.33 & 0.60 & 0.33 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.33 & 0.00 & 0.33 & 0.00 & 0.00 \end{bmatrix}$$

(Unnormalized) Laplacian matrix $\mathbf{L} = \mathbf{\Lambda} - \mathbf{W}$

$$\mathbf{L} = \begin{bmatrix} 1.33 & 0.00 & -1.00 & 0.00 & -0.33 & 0.00 \\ 0.00 & 1.26 & 0.00 & -0.33 & -0.60 & -0.33 \\ -1.00 & 0.00 & 1.33 & 0.00 & -0.33 & 0.00 \\ 0.00 & -0.33 & 0.00 & 0.66 & 0.00 & -0.33 \\ -0.33 & -0.60 & -0.33 & 0.00 & 1.26 & 0.00 \\ 0.00 & -0.33 & 0.00 & -0.33 & 0.00 & 0.66 \end{bmatrix}$$

⇒ normalized Laplacian matrices:

- Random-walk Laplacian $\mathbf{L}_{rw} = \mathbf{\Lambda}^{-1}\mathbf{L}$
- Symmetric Laplacian $\mathbf{L}_{sym} = \mathbf{\Lambda}^{-0.5}\mathbf{L}\mathbf{\Lambda}^{-0.5}$

Idea of 1D spectral embedding & clustering

Goal: find embedding $\mathbf{y} = (y_1, \ldots, y_n)^T$, where each y_i corresponds v_i and $cost(G, \mathbf{y})$ minimal.

$$\boxed{cost(G, \mathbf{y}) = \sum \sum w_{ij}(y_i - y_j)^2 = 2\mathbf{y}^T \mathbf{L} \mathbf{y}}$$

- we want to avoid trivial solution $\forall i : y_i = 0 \rightarrow$
- scaling constraint (e.g.) $\mathbf{y}^T \mathbf{y} = 1$ (i.e., $\sum_i y_i^2 = 1$)
- $\mathbf{L}$ is positive semidefinite (eigenvalues λ_i real, $\lambda_i \geq 0$)
- solution smallest non-trivial eigenvector of $\mathbf{L}$

Extra: Why eigenvectors $\mathbf{y}$ of $\mathbf{L}$ would be the solution?

Task: Find $\mathbf{y}$ such that $2\mathbf{y}^T\mathbf{L}\mathbf{y}$ minimal given constraint $\mathbf{y}^T\mathbf{y} = 1$

Method of Lagrange multipliers:

1. Reformulate as a Lagrangian function $\mathcal{L}(\mathbf{y}, \lambda) = \mathbf{y}^T\mathbf{L}\mathbf{y} - \lambda(\mathbf{y}^T\mathbf{y} - 1)$
2. Set the partial derivatives (with respect to $\mathbf{y}$ and λ) as 0
3. Reduces to $\mathbf{L}\mathbf{y} = \lambda\mathbf{y}$ **Eigenvalue & -vector definition!**

Idea of 1D spectral embedding & clustering

- solution smallest non-trivial eigenvector $\mathbf{y}$ of $\mathbf{L}$
- $cost = 2\mathbf{y}^T\mathbf{L}\mathbf{y} = 2\mathbf{y}^T\lambda\mathbf{y} = 2\lambda(y_1^2 + \ldots + y_n^2) = 2\lambda$ (λ eigenvalue)
- $cost$ minimal, when λ minimal (recall $\mathbf{y}^T\mathbf{y} = 1$)
- but **skip trivial solution** $\lambda = 0$ with $\mathbf{y}$ (proportional to) $\mathbf{1} = (1, \ldots, 1)^T$
  - exists always when $\mathbf{G}$ connected
- → optimal solution **eigenvector corresponding to the 2nd smallest** λ
- cluster elements of $\mathbf{y}$ with K-means

Example

Unnormalized Laplacian L

$$\begin{bmatrix} 1.33 & 0.00 & -1.00 & 0.00 & -0.33 & 0.00 \\ 0.00 & 1.26 & 0.00 & -0.33 & -0.60 & -0.33 \\ -1.00 & 0.00 & 1.33 & 0.00 & -0.33 & 0.00 \\ 0.00 & -0.33 & 0.00 & 0.66 & 0.00 & -0.33 \\ -0.33 & -0.60 & -0.33 & 0.00 & 1.26 & 0.00 \\ 0.00 & -0.33 & 0.00 & -0.33 & 0.00 & 0.66 \end{bmatrix}$$

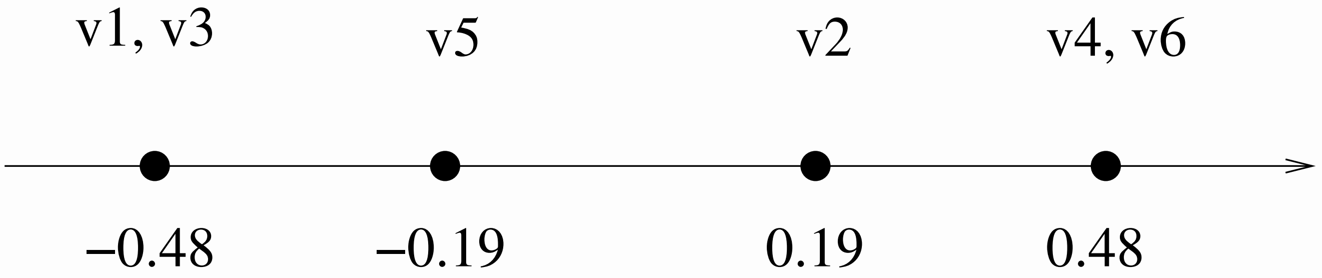
Eigenvalues:

$\approx$0, 0.20, 0.99, 0.99, 1.99, 2.33 *

Second smallest eigenvector:

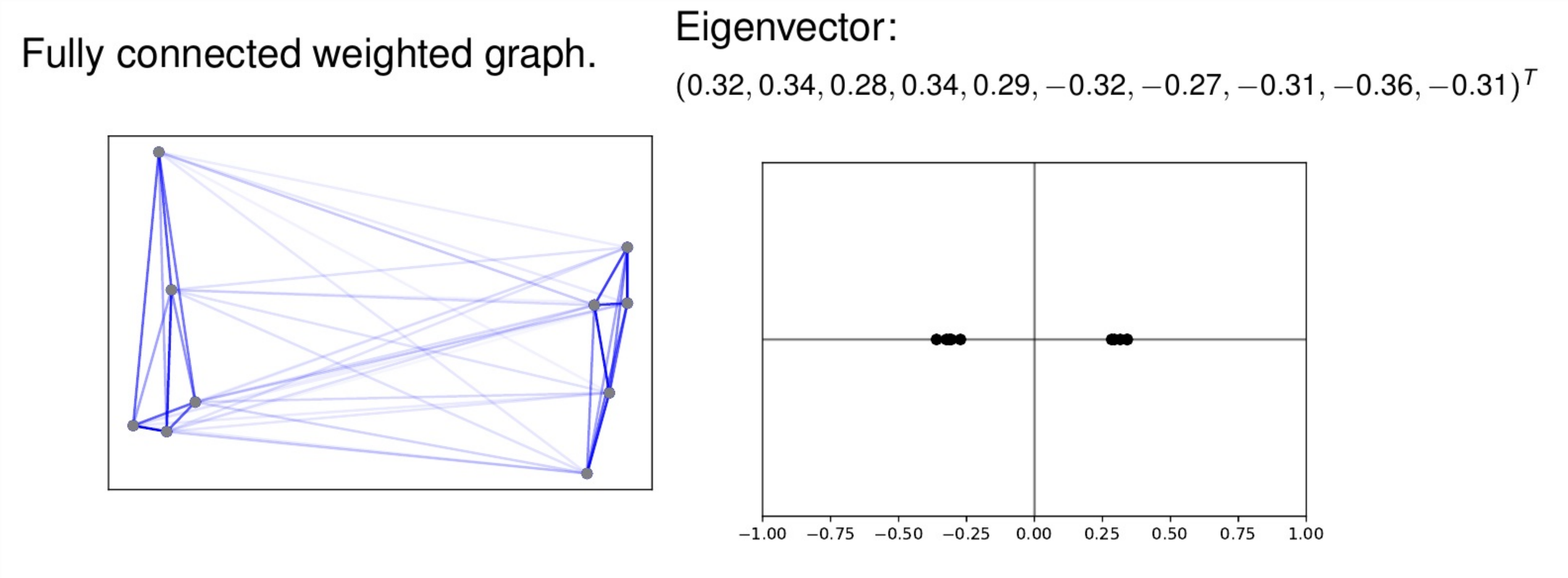
$(0.48, -0.19, 0.48, -0.48, 0.19, -0.48)^T$

The new representation can be clustered by K-means:



* 1st eigenvalue 1.9e-16 due to imprecision (should be 0)

Another example with 1D embedding



Fully connected weighted graph.

Eigenvector:

$(0.32, 0.34, 0.28, 0.34, 0.29, -0.32, -0.27, -0.31, -0.36, -0.31)^T$

$n = 10$

Example by Bruno Ordozgoiti, MDM 2020

Generalization with multidimensional embedding

Unnormalized spectral clustering
Input: Graph $\mathbf{G}$ with adjacency matrix $\mathbf{W}$, number of clusters K.

1. Compute the Laplacian $\mathbf{L} = \mathbf{\Lambda} - \mathbf{W}$
2. Compute the **eigenvectors** $\mathbf{y}_1, \ldots, \mathbf{y}_k$ of $\mathbf{L}$ corresponding to the k smallest eigenvalues (excluding $\lambda = 0$)
3. Present the data as matrix $\mathbf{Y}$ whose columns are $\mathbf{y}_1, \ldots, \mathbf{y}_k$.
4. Cluster $\mathbf{Y}$ with K-means.

Note: Usually $k = K$ or $k < K$. Eigengap $|\lambda_{k+1} - \lambda_k|$ can be used to choose k.

Eigengap heuristic for choosing k

Choose k such that $\lambda_1, \ldots, \lambda_k$ small but λ_{k+1} relatively large.

clear gap

no gap

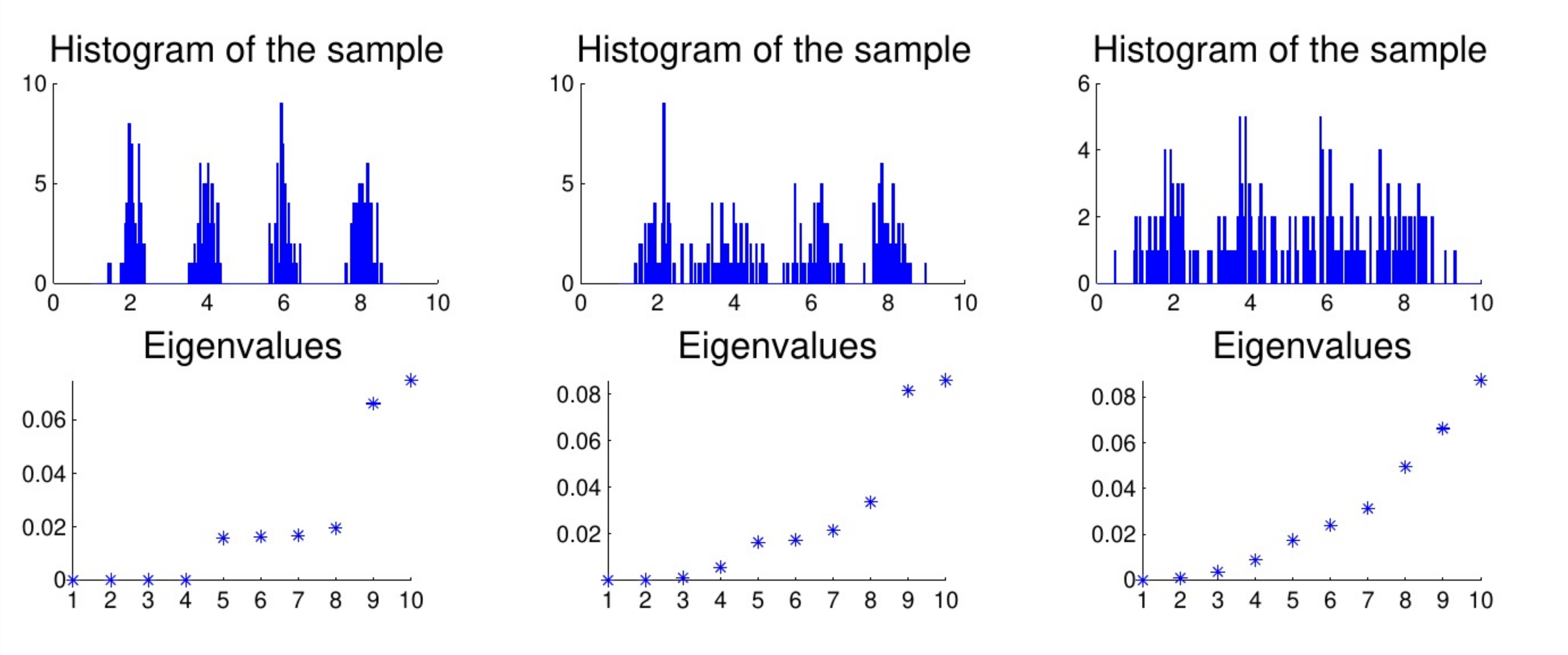


Image source: Fig 4 by von Luxburg (2006)

Normalized spectral clustering using random walk Laplacian $\mathbf{L}_{rw}$

Input: Graph $\mathbf{G}$ with adjacency matrix $\mathbf{W}$, number of clusters K.

1. Compute the **random walk** Laplacian $\mathbf{L}_{rw} = \mathbf{\Lambda}^{-1} L$
2. Compute the right eigenvectors $\mathbf{y}_1, \ldots, \mathbf{y}_k$ of $\mathbf{L}_{rw}$ corresponding to the k smallest eigenvalues (excluding $\lambda = 0$)
3. Present the data as matrix $\mathbf{Y}$ whose columns are $\mathbf{y}_1, \ldots, \mathbf{y}_k$.
4. Normalize the columns of $\mathbf{Y}$ to unit norm.
5. Cluster $\mathbf{Y}$ with K-means.

Normalized spectral clustering using symmetric normalized Laplacian $\mathbf{L}_{sym}$

Input: Graph $\mathbf{G}$ with adjacency matrix $\mathbf{W}$, number of clusters K.

1. Compute the **symmetric normalized** Laplacian $\mathbf{L}_{sym} = \mathbf{\Lambda}^{-1/2} L \mathbf{\Lambda}^{-1/2}$
2. Compute the eigenvectors $\mathbf{y}_1, \ldots, \mathbf{y}_k$ of $\mathbf{L}_{sym}$ corresponding to the k smallest eigenvalues (excluding $\lambda = 0$)
3. Present the data as matrix $\mathbf{Y}$ whose columns are $\mathbf{y}_1, \ldots, \mathbf{y}_k$.
4. Normalize the rows of $\mathbf{Y}$ to unit norm.
5. Cluster $\mathbf{Y}$ with K-means.

Important choices

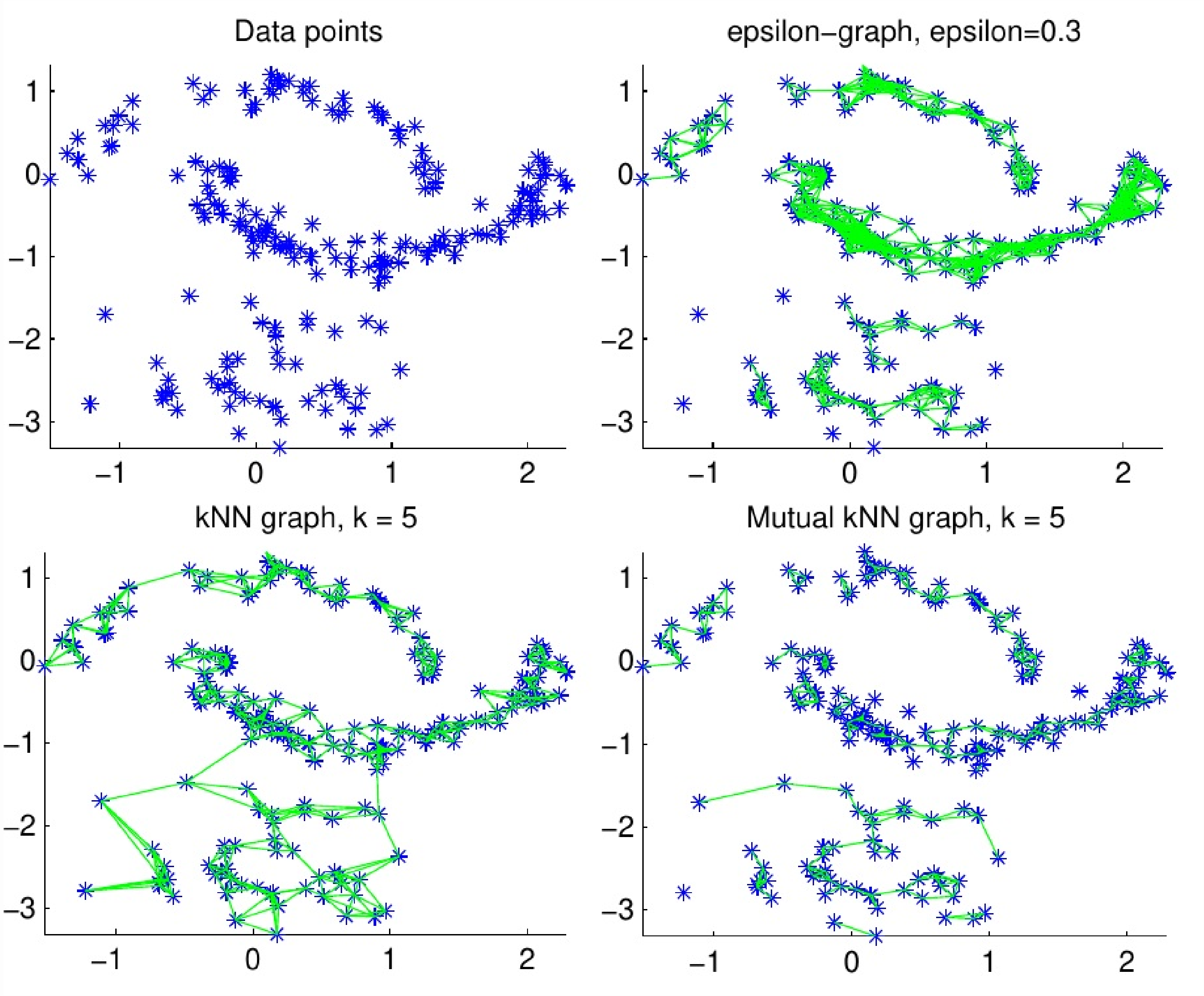
- **Method**: Unnormalized, random walk or symmetric normalized?
  - Usually normalization helps. Suggestion: try random walk first.
- **Similarity measure**
  - should measure local similarity reliably (close neighbours)
  - for numeric data, Gaussian similarity $exp\left(\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right)$ often used
- **Similarity graph** and its parameters
  - this has a strong effect on results!

Common choices for the similarity graph

General goal: sparse but connected graph (or number of connected components $<< K$)

1. **ϵ-neighbourhood** graph: keep only $w_{ij} \geq \epsilon$
   - problems if clusters of different densities
2. **k-nearest neighbour** graph: v_i among k nearest neighbours of v_j **or** vice versa
   - often a good first choice
   - can break the graph into disconnected components
3. **mutual k-nearest neighbour** graph: v_i among k nearest neighbours of v_j **and** vice versa

Similarity graph examples (von Luxburg, Fig 3)



Similarity graph (cont)

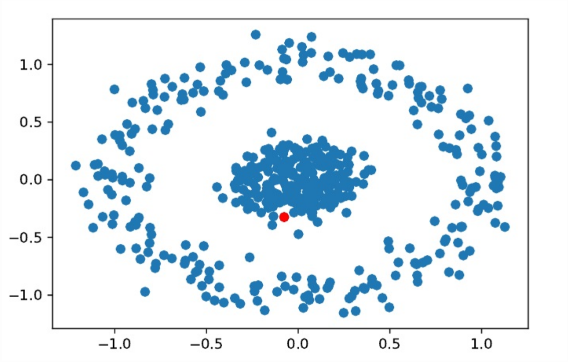
4. **fully connected graph**

- often with Gaussian similarity $\kappa(\mathbf{x}_i, \mathbf{x}_j) = exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$ (radial basis function, RBF)
- how to choose σ?
- Note: in scikitlearn parameter $\gamma = \frac{1}{2\sigma^2}$
- graph not sparse → heavy computation

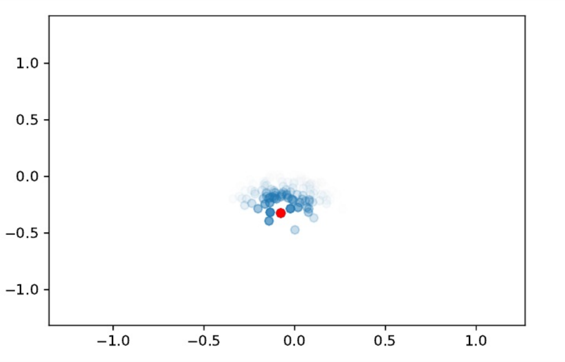
Choice of parameters (ϵ, k, σ) affects a lot, too!

Example: neighbourhood with $\kappa(\mathbf{x}_i, \mathbf{x}_j) = exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$

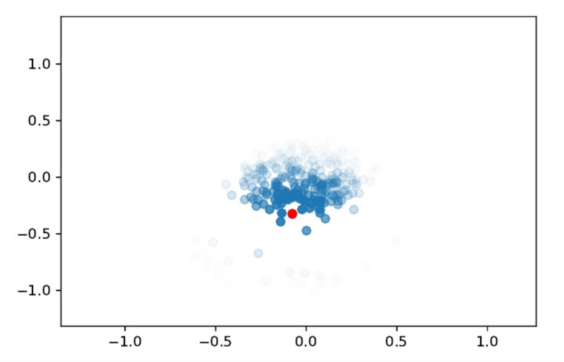
Data, query point red



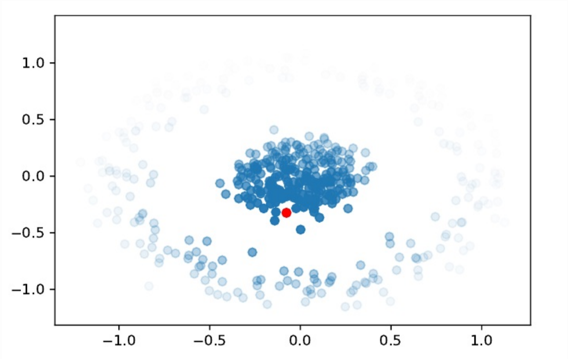
$\sigma = 0.1$



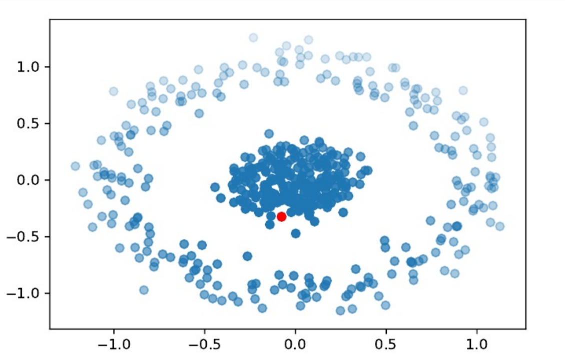
$\sigma = 0.2$



$\sigma = 0.4$



$\sigma = 0.8$



$\sigma = 1.6$

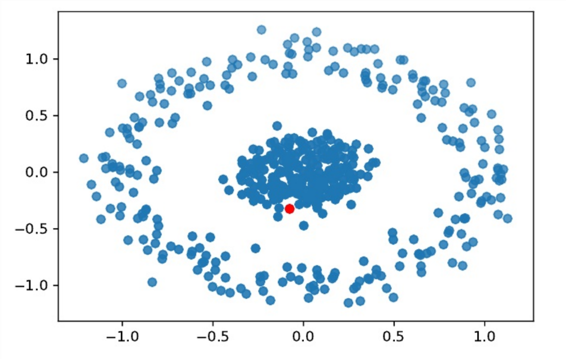
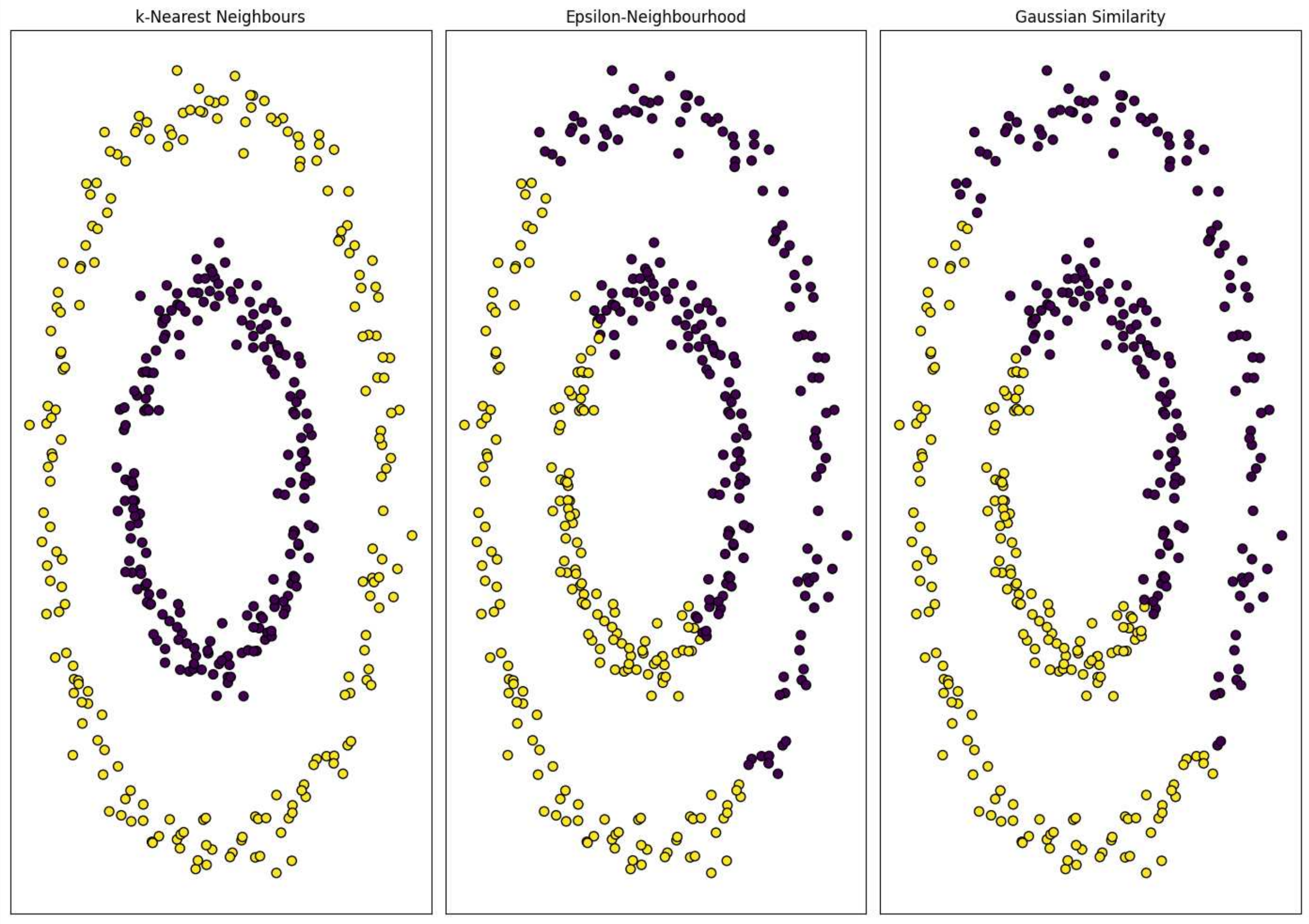


Image source: Bruno Ordozgoiti, MDM 2020 slides

Example: Different clustering results with different similarity graphs



Parameters: $k = 2$, $\epsilon = 0.3$, RBF with $\gamma = 10$ (i.e., $\sigma = \sqrt{5}$)

Experiment by Lai Khoa for MDM 2023

Summary

Idea: similarity graph → low-dimensional VS presentation (eigenvectors) → clustering (K-means etc.)

+ very powerful (virtually any datatype, arbitrary shapes)

– computationally expensive

- creating similarity graph $\mathcal{O}(n^2)$, spectral decomposition $\mathcal{O}(n^3)$

– many important parameter choices

Further reading

von Luxburg: A Tutorial on Spectral Clustering. Statistics and Computing, vol. 17, pp. 395–416, 2007.

Reading guide: Sec 2 overview, 2.2, Sec 3 overview + definitions of Laplacian matrices from 3.1-3.2, Sec 4, Sec 5 overview, (possibly Sec 6 overview), Sec 8. [a]

[a] section overview = text before subsections