Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
Università degli Studi di Padova

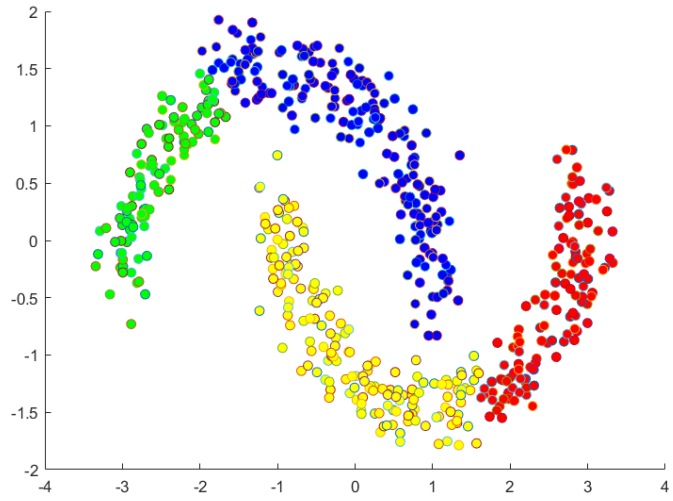
A Harmonic Journey trough Graph Wavelets

4. Spectral clustering

Wolfgang Erb
University of Padova

Frontiers and Applications of Approximation Theory
Modern Perspectives for Young Researchers

March 16-20, 2026
Lucian Blaga University of Sibiu, Romania



wolfgang.erb@unipd.it

Spectral clustering

Spectral clustering goes back to Donath and Hoffman (1973), who first suggested to construct graph partitions based on eigenvectors of the adjacency matrix. In the same year, Fiedler (1973) discovered that bi-partitions of a graph are closely connected with the second eigenvector of the graph Laplacian $\mathbf{L}$, and he suggested to use this eigenvector to partition a graph. Since then, spectral clustering has been discovered, re-discovered, and extended many times.

In the machine learning community, spectral clustering has been made popular by the works of Shi and Malik (2000) and Ng et al. (2002).

Idea of spectral clustering

Observation: we saw that the number of connected components of a graph corresponds to the multiplicity of the eigenvector $\lambda_1 = 0$. Moreover, the eigenvectors corresponding to the zero eigenvalue are piecewise constant on the connected components.

Main idea: If data points are strongly clustered a corresponding similarity graph can be almost split in several (lets say k) connected components and the eigenvectors corresponding to the k smallest eigenvalues are almost constant on these components. The entries of the k eigenvectors form therefore clusters in $\mathbb{R}^k$ which can be determined, for instance, by the use of k-means.

Advantages and drawbacks of spectral clustering

- Spectral clustering does not make strong assumptions on the form of the clusters. This is different in other clustering algorithms as, for instance, the k-means algorithm. In this case, the clusters form convex sets.
- Can be implemented efficiently for large graphs as long as the adjacency matrix $\mathbf{A}$ is sparse.
- Caveat: depends strongly on the similarity graph generated from the data points.

Algorithm 1: Unnormalized spectral clustering

Input: Point set $V = \{v_1, \ldots, v_n\}$, number $k \geq 1$ of clusters to construct.

1: Construct similarity graph G from the given data. For this, use for instance a *kNN* or ε-balls strategy. Apply weighting strategy (binary, Gaussian) to get the adjacency matrix $\mathbf{A}$.

2: Compute the standard graph Laplacian

$$\mathbf{L} = \mathbf{D} - \mathbf{A}.$$

3: Calculate the k smallest eigenvectors $u_1, \ldots, u_k$ of $\mathbf{L}$:

$$\mathbf{L}u_i = \lambda_i u_i, \quad i \in \{1, \ldots, k\}.$$

Store them in a matrix $\mathbf{U}_k = [u_1, \ldots, u_k]$.

4: Let $y_i \in \mathbb{R}^k$, $i \in \{1, \ldots, n\}$, denote the i-th row of $\mathbf{U}_k$.

5: Cluster $\{y_1, \ldots, y_n\} \subset \mathbb{R}^k$ with k-means into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{v_j \in V : y_j \in C_i\}$.

Algorithm 2: Shi/Malik normalized spectral clustering

Input: Point set $V = \{v_1, \ldots, v_n\}$, number $k \geq 1$ of clusters to construct.

1: Construct similarity graph G from the given data. For this, use for instance *kNN* or ε-balls strategy. Apply weighting strategy (binary, Gaussian) to get the adjacency matrix $\mathbf{A}$.

2: Compute the standard graph Laplacian

$$\mathbf{L} = \mathbf{D} - \mathbf{A}.$$

3: Calculate the k smallest eigenvectors $\nu_1, \ldots, \nu_k$ of the eigenvalue problem:

$$\mathbf{L}\nu_i = \lambda_i \mathbf{D}\nu_i, \quad i \in \{1, \ldots, k\},$$

i.e., the eigenvectors of the random walk Laplacian $\mathbf{L}_{RW}$ and store them in a matrix $\mathbf{V}_k = [\nu_1, \ldots, \nu_k]$.

4: Let $y_i \in \mathbb{R}^k$, $i \in \{1, \ldots, n\}$, denote i-th row of $\mathbf{V}_k$.

5: Cluster $\{y_1, \ldots, y_n\} \subset \mathbb{R}^k$ with k-means into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{v_j \in V : y_j \in C_i\}$.

Algorithm 3: Ng/Jordan/Weiss normalized spectral clustering

Input: Point set $V = \{v_1, \ldots, v_n\}$, number $k \geq 1$ of clusters to construct.

1: Construct similarity graph G from the given data. For this, use for instance *kNN* or ε-balls strategy. Apply weighting strategy (binary, Gaussian) to get the adjacency matrix $\mathbf{A}$.

2: Compute the normalized graph Laplacian

$$\mathbf{L}_N = \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}.$$

3: Calculate the k smallest eigenvectors $u_1, \ldots, u_k$ of the Laplacian $\mathbf{L}_N$. and store them in a matrix $\mathbf{U}_k = [u_1, \ldots, u_k]$.

4: Generate $\mathbf{T}_k$ by normalizing the rows of $\mathbf{U}_k$:

$$(\mathbf{T}_k)_{i,j} = \frac{(\mathbf{U}_k)_{i,j}}{\sqrt{\sum_{j=1}^{k} |(\mathbf{U}_k)_{i,j}|^2}}.$$

5: Let $y_i \in \mathbb{R}^k$, $i \in \{1, \ldots, n\}$, denote the i-th row of $\mathbf{T}_k$.

6: Cluster $\{y_1, \ldots, y_n\} \subset \mathbb{R}^k$ with k-means into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{v_j \in V : y_j \in C_i\}$.

Spectral clustering algorithms - some details

- All three algorithms look rather similar, apart from the fact that they use the three different graph Laplacians $\mathbf{L}$, $\mathbf{L}_{RW}$ and $\mathbf{L}_N$.
- In all three algorithms, the vectors $y_i \in \mathbb{R}^k$ form representations of the data points v_i. Due to the properties of the graph Laplacians this representation turns out to be useful for the detection of the clusters.
- In particular, the *k*-means clustering algorithm has no difficulties to detect the clusters in this new representation.
- In the third Algorithm 3 an additional row normalization step is introduced which is not needed in the other algorithms. We'll clearify this normalization later.

The starting point: how to we get graphs from data?

In practice, we are often just given a set of vertices V (point cloud).

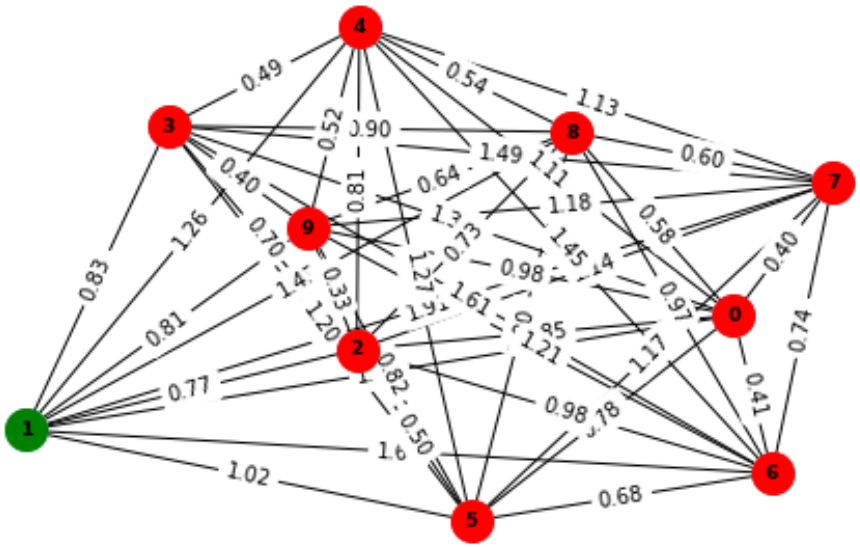
Question: how to set up edges between these vertices in order to get a similarity graph?

The following questions are relevant for setting up the edges:

1. Should each vertex be connected to every other vertex in order to get a complete graph (increasing the accuracy of our model)?
2. Should the graph have a sparse edge structure (increasing computational performance)?
3. What are the weights $\mathbf{A}_{i,j}$ assigned to each edge?

Strategy I: complete graphs

Idea: construct a complete graph K_n by mutually connecting all vertices.

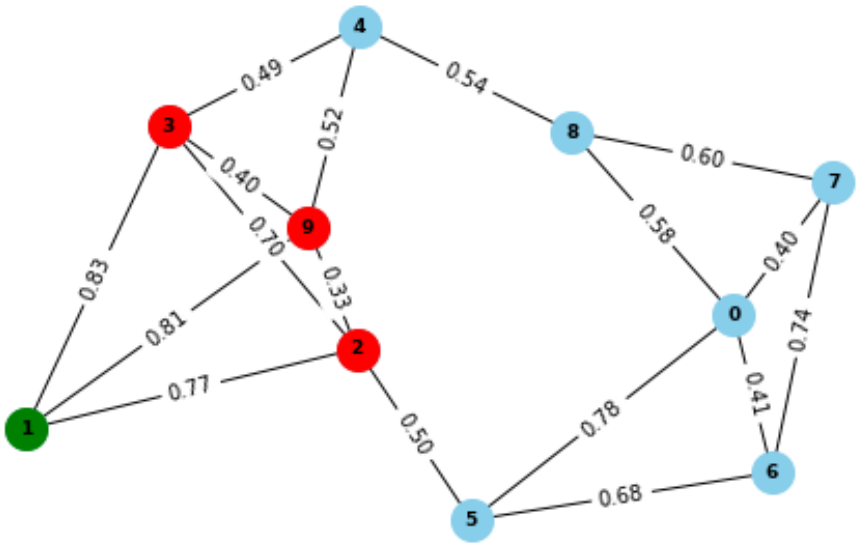


- Gaussian similarities $\mathbf{A}_{i,j} = e^{-\operatorname{dist}(\mathrm{v}_i, \mathrm{v}_j)^2/\epsilon^2}$ are often used as edge weights, where $\operatorname{dist}(\cdot,\cdot)$ is a metric distance, and ϵ a scale parameter.
- A drawback of this strategy is the large number of resulting edges, i.e., $|E(K_n)| = n(n-1)/2$, which leads to a dense graph Laplacian and possibly to cost-expensive matrix-vector operations.

Strategy II: k-nearest neighbor graph (kNN graph)

Directed kNN graph: connect v_i with v_j if v_j is among the k-nearest neighbors of v_i.

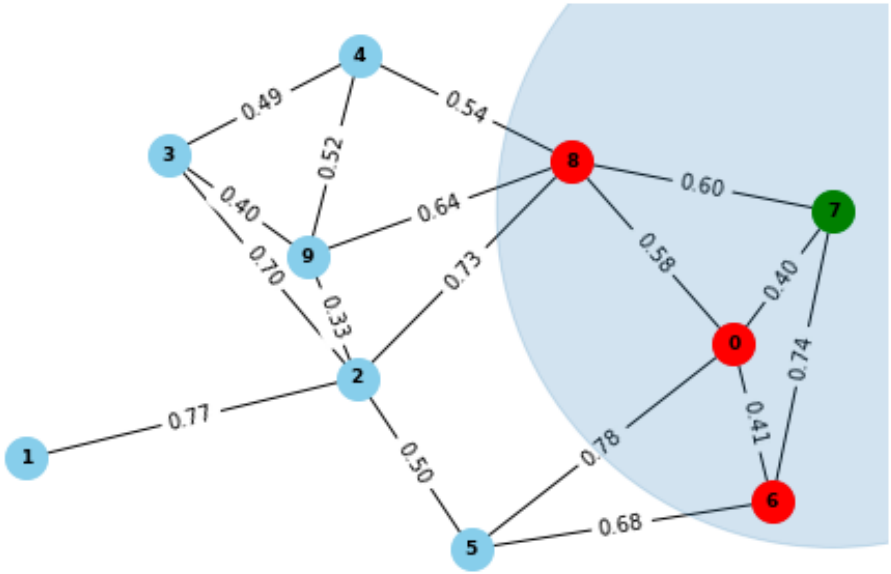
Undirected kNN graph: create an edge $e = (\mathrm{v}_i, \mathrm{v}_j)$ if v_i is among the k-nearest neighbors of v_j or if v_j is among the k-nearest neighbors of v_i.



kNN graphs are usually sparse. Relevant parameters are the distance metric between the vertices and the number k. The edge weights $\mathbf{A}_{i,j}$ are usually chosen as $\mathbf{A}_{i,j} \in \{0, 1\}$ (unweighted graph) or Gaussian weights.

Strategy III: ε-ball graph

Idea: connect all vertices whose pairwise distances are smaller than $\varepsilon > 0$.



For ε-ball graphs, the edge weights $\mathbf{A}_{i,j}$ are usually chosen to be $\mathbf{A}_{i,j} \in \{0, 1\}$ (unweighted graph). The important parameters are the distance metric between the vertices and the value of ε.

Spectral clustering - 1D toy example

- We draw 300 points $\{v_1, \ldots, v_{300}\}$ in $\mathbb{R}$ randomly according to a mixture of three Gaussians.
- As similarity graphs, we consider a complete graph as well as the *kNN*-graph with $k = 10$.
- As edge weights we choose the Gaussian similarity $\mathbf{A}_{i,j} = e^{-|v_i - v_j|^2}$.
- We consider the Laplacian $\mathbf{L}$ and the random-walk Laplacian $\mathbf{L}_{RW}$.

In the following two figures:

- We plot a histogram of the sampled values
- In Fig. 1, we plot the eigenpairs of $\mathbf{L}$ and $\mathbf{L}_{RW}$ for the *kNN*-graph. For the eigenvalues, we plot the index i vs. $\lambda_i/\lambda_{\text{max}}$. For the eigenvectors u_k, we plot v_i against the i-th. component of u_k.
- In Fig. 2, we plot the eigenpairs of $\mathbf{L}$ and $\mathbf{L}_{RW}$ for the complete graph.

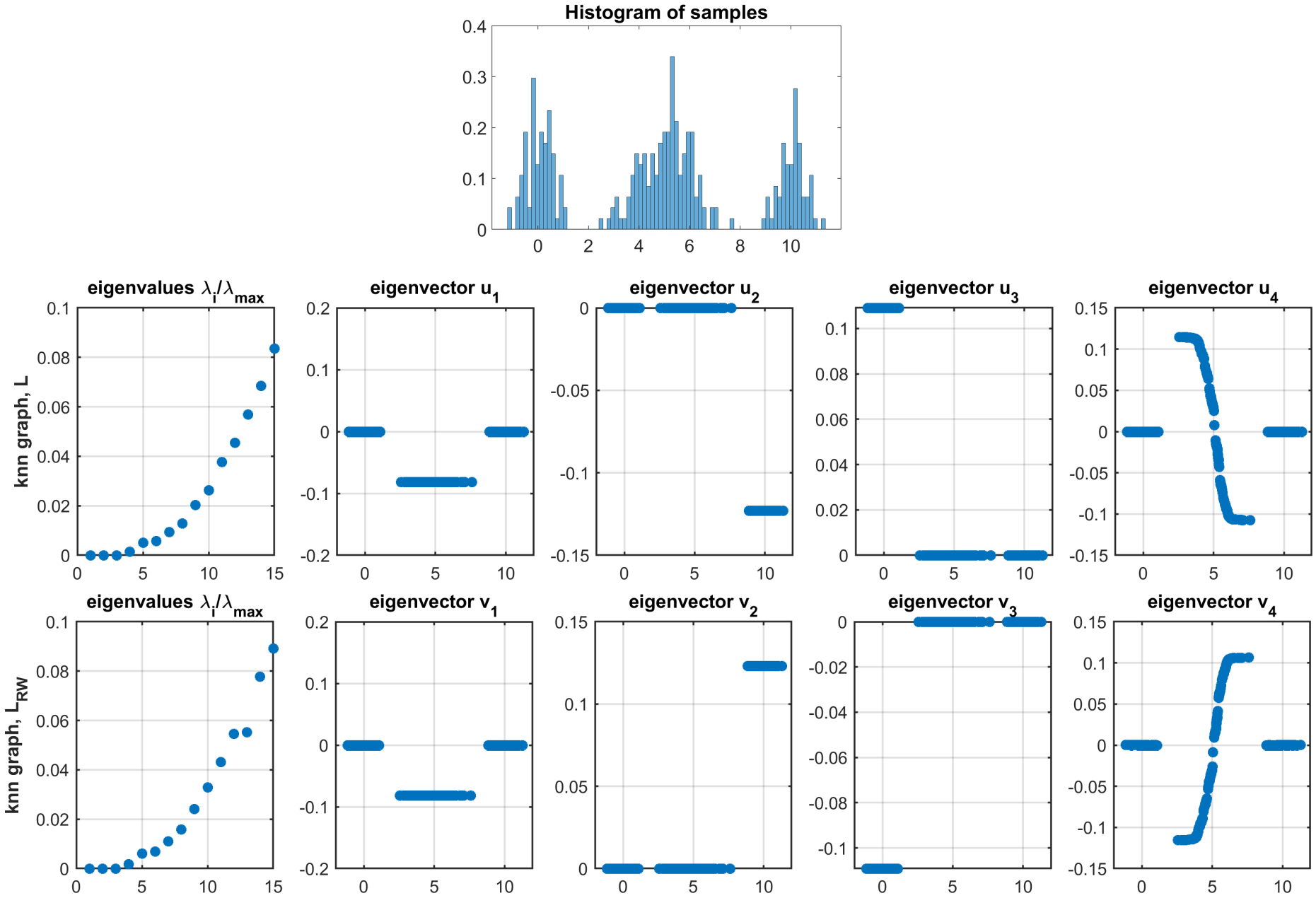


Figure 1: Eigenpairs of $\mathbf{L}$ and $\mathbf{L}_{RW}$ based on a *kNN* graph with $k = 10$.

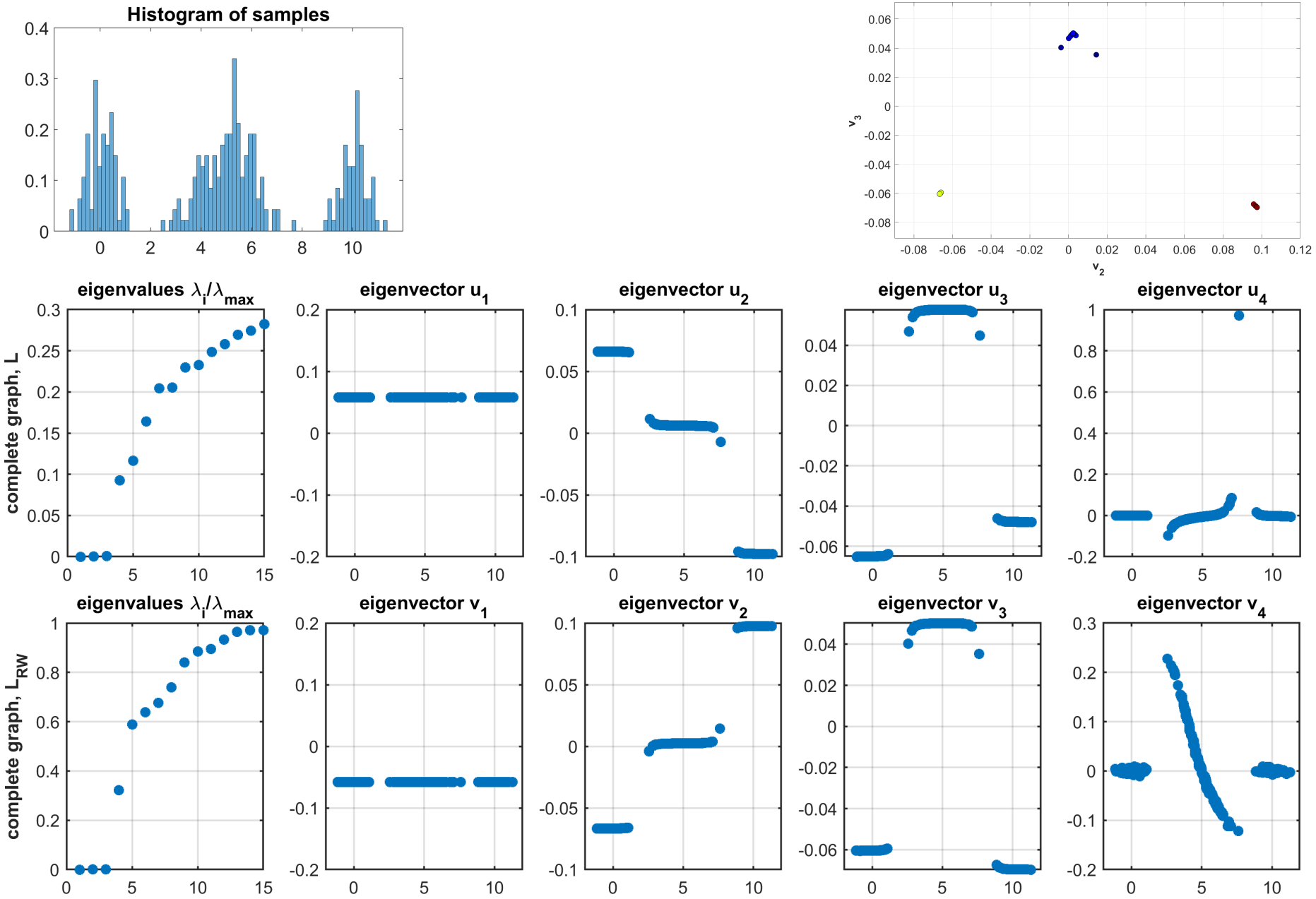


Figure 2: Eigenpairs of $\mathbf{L}$ and $\mathbf{L}_{RW}$ based on a complete graph.

Relation between spectral clustering and graph cuts

Intuition of clustering for graphs: find a partition such that the edges between different parts have a very low weight (points in different clusters are dissimilar from each other) and the edges within the same part have high weight (points within the same cluster are similar to each other).

Simple idea for clustering: For $k \in \mathbb{N}$, find clusters $A_1, \ldots, A_k$ that minimize the graph cut

$$\mathrm{cut}(A_1, A_2, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} |E(A_i, A_i^c)|,$$

where

$$|E(A_i, A_i^c)| = \sum_{v_j \in A_i, v_k \in A_i^c} \mathbf{A}_{j,k},$$

and $A_i^c = V \setminus A_i$ denotes the complement of A_i in V.

Spectral clustering and graph cuts

Problem with graph cut: in many cases the solution separates just one individual vertex from the rest of the graph.

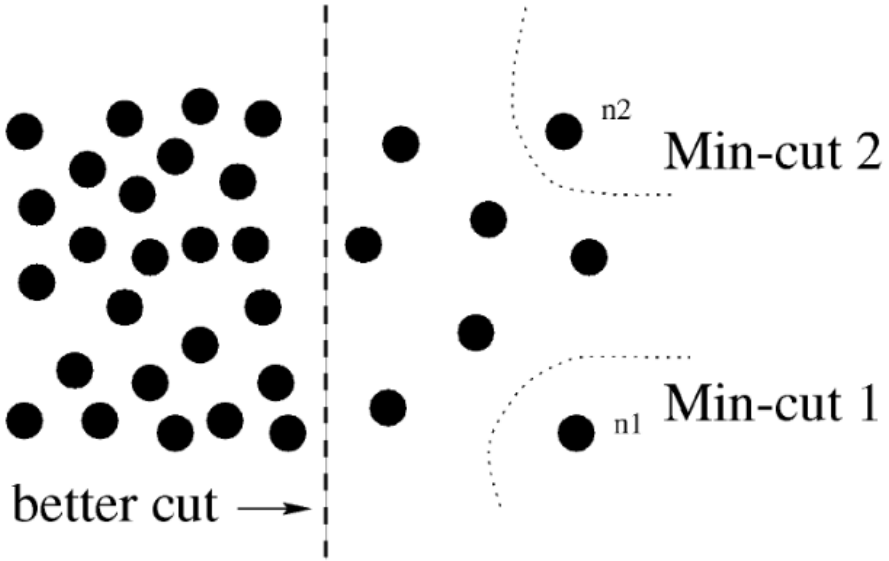


Figure 3: Possible problems when minimizing the graph cut (Shi & Malik, 2000).

Spectral clustering and graph cuts

Alternative approach: enforce the sets $A_1, \ldots, A_k$ to be reasonably large. The two most common objective functions to encode this are RatioCut (Hagen, Kahng, 1992) and the normalized cut Ncut (Shi, Malik, 2000).

$$\mathrm{RatioCut}(A_1, A_2, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{|E(A_i, A_i^c)|}{|A_i|} = \sum_{i=1}^{k} \frac{\mathrm{cut}(A_i, A_i^c)}{|A_i|},$$

$$\mathrm{Ncut}(A_1, A_2, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{|E(A_i, A_i^c)|}{\mathrm{vol}(A_i)} = \sum_{i=1}^{k} \frac{\mathrm{cut}(A_i, A_i^c)}{\mathrm{vol}(A_i)},$$

where $\mathrm{vol}(A_i)$ corresponds to the sum of the weights of the edges in A_i (i.e. of the edges of the reduced graph with node set A_i).

Spectral clustering and graph cuts

Note: both objective functions take a small value if the clusters A_i are not too small. In particular, the minima of the functions

$$\sum_{i=1}^{k} \frac{1}{|A_i|}, \quad \sum_{i=1}^{k} \frac{1}{\mathrm{vol}(A_i)}$$

are achieved if all $|A_i|$ and all $\mathrm{vol}(A_i)$ coincide, respectively. So what both objective functions try to achieve is that the clusters are balanced, as measured by the number of vertices or edge weights, respectively. Unfortunately, introducing balancing conditions turn the previously simple graph cut problem into NP hard problems (Wagner, Wagner, 1993). Spectral clustering is a way to solve relaxed versions of those problems. We will see that relaxing Ncut leads to normalized spectral clustering, while relaxing RatioCut leads to unnormalized spectral clustering.

Approximating ratio cut with $k = 2$

For $k = 2$, minimizing the ratio cut can be written as

$$\min_{A \subset V} \text{RatioCut}(A, A^C).$$

We rewrite the problem in a more convenient form. Given a subset $A \subset V$ we define the vector $f = (f_1, \ldots, f_n)^* \in \mathbb{R}^n$ with the entries

$$f_i = \begin{cases} \sqrt{\frac{|A^c|}{n|A|}} & \text{if } v_i \in A \\ -\sqrt{\frac{|A|}{n|A^c|}} & \text{if } v_i \in A^c \end{cases}.$$

Approximating ratio cut with $k = 2$

We can now rewrite the ratio cut using the standard graph Laplacian:

$$
\begin{aligned}
f^* \mathbf{L} f &= \frac{1}{2} \sum_{i,j=1}^{n} \mathbf{A}_{i,j} (f_i - f_j)^2 \\
&= \frac{1}{2n} \sum_{v_i \in A, v_j \in A^c} \mathbf{A}_{i,j} \left(\sqrt{\frac{|A^c|}{|A|}} + \sqrt{\frac{|A|}{|A^c|}} \right)^2 \\
&\qquad + \frac{1}{2n} \sum_{v_i \in A^c, v_j \in A} \mathbf{A}_{i,j} \left(-\sqrt{\frac{|A^c|}{|A|}} - \sqrt{\frac{|A|}{|A^c|}} \right)^2 \\
&= \frac{1}{n} \operatorname{cut}(A, A^c) \left(\frac{|A^c|}{|A|} + \frac{|A|}{|A^c|} + 2 \right) \\
&= \frac{1}{n} \operatorname{cut}(A, A^c) \left(\frac{|A| + |A^c|}{|A|} + \frac{|A| + |A^c|}{|A^c|} \right) \\
&= \operatorname{RatioCut}(A, A^c).
\end{aligned}
$$

Approximating ratio cut with $k = 2$

Additionally, we have

$$n \sum_{i=1}^{n} f_i = \sum_{v_i \in A} \sqrt{\frac{|A^c|}{|A|}} - \sum_{v_i \in A^c} \sqrt{\frac{|A|}{|A^c|}} = |A| \sqrt{\frac{|A^c|}{|A|}} - |A^c| \sqrt{\frac{|A|}{|A^c|}} = 0.$$

and

$$\|f\|_2^2 = \sum_{i=1}^{n} f_i^2 = \frac{|A|}{n} \frac{|A^c|}{|A|} + \frac{|A^c|}{n} \frac{|A|}{|A^c|} = \frac{|A^c| + |A|}{n} = 1,$$

i.e., the vector f is normalized and orthogonal to the constant vector e. Altogether, we can see that the problem of minimizing the ratio cut can be equivalently rewritten as

$$\min_{A \subset V} f^* \mathbf{L} f \text{ subject to } f^* e = 0, f \text{ piecew. const. on } A \text{ and } A^c, \|f\|_2 = 1.$$

This is a discrete optimization problem (the entries of the solution f are only allowed to take two values), and it is still NP hard.

Approximating ratio cut with $k = 2$

To relax this minimization problem, we can drop the condition that f is piecewise constant and allow arbitrary values $f_i \in \mathbb{R}$. Then, we get

$$\min_{f \in \mathbb{R}^n} f^* \mathbf{L} f \text{ subject to } f^* e = 0, \|f\|_2 = 1.$$

We know that the solution of this relaxed minimization problem is given by the eigenvector u_2 of $\mathbf{L}$ with respect to the second largest eigenvalue λ_2 of $\mathbf{L}$. To obtain a partition of the graph we need to turn u_2 into a discrete indicator vector. The simplest way is to use the sign of u_2 as indicator function, i.e.,

$$\begin{cases} v_i \in A & \text{if } f_i \geq 0 \\ v_i \in A^c & \text{if } f_i < 0 \end{cases}.$$

For $k > 2$, this heuristic is however too simple. We can then use the k-means clustering algorithm to split the values f_i into two clusters C and C^c. This leads to the following clustering of V:

$$\begin{cases} v_i \in A & \text{if } f_i \in C \\ v_i \in A^c & \text{if } f_i \in C^c \end{cases}.$$

Approximating Ncut for $k = 2$

Similar to RatioCut, normalized spectral clustering (Shi/Malik) can be seen as a relaxed Ncut problem. Let,

$$f_i = \begin{cases} \sqrt{\frac{\mathrm{vol}(A^c)}{\mathrm{vol}(A)\ \mathrm{vol}(V)}} & \text{if } v_i \in A \\ -\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(A^c)\,\mathrm{vol}(V)}} & \text{if } v_i \in A^c \end{cases}.$$

Then, as before one can see that $(\mathbf{D}f)^* e = 0, f^* \mathbf{D} f = 1$, and $f^* \mathbf{L} f = \mathrm{Ncut}(A, A^c)$. Thus, Ncut can be rewritten as

$$\min_A f^* \mathbf{L} f \text{ subject to } f \text{ piecew. const. on } A \text{ and } A^c, \mathbf{D} f^* e = 0, f^* \mathbf{D} f = 1.$$

Again, by dropping the condition that f is piecewise constant, we get

$$\min_{f \in \mathbb{R}^n} f^* \mathbf{L} f \text{ subject to } \mathbf{D} f^* e = 0, f^* \mathbf{D} f = 1.$$

The solution f to this minimization problem is given by the eigenvector ν_2 w.r.t. the second largest eigenvalue λ_2 of $\mathbf{L}_{RW}$.

Some comments

- If the graph consists of two connected components, the solutions of the relaxed problems correspond to the solutions of the original minimization problems.
- In general, there is no guarantee on the quality of the relaxed solutions compared to the exact solutions of Ncut and RatioCut. For instance, the minimal value of the relaxed problem can differ arbitrarily from the minimal value of RatioCut (cf. tutorial of von Luxburg).
- There are other relaxation possibilities for Ncut and RatioCut. The popularity of the spectral relaxation is due to the fact that eigenvalues/eigenvectors are well-understood objects in numerical linear algebra.

Practical details

Which similarity graph should be chosen?

- The outcome of the 3 spectral clustering algorithms depends strongly on the chosen similarity graph.
- There is no theory how an "optimal" similarity graph looks like.
- In principle, all strategies seen before can be used to create the similarity graph. The selection of proper parameters is typically a critical issue. It is recommendable to use a sparse graph structure.
- A simple and relatively stable strategy is to generate a kNN-graph. For clusters with different point densities one can additionally use Gaussian weights for the edges.

Practical details

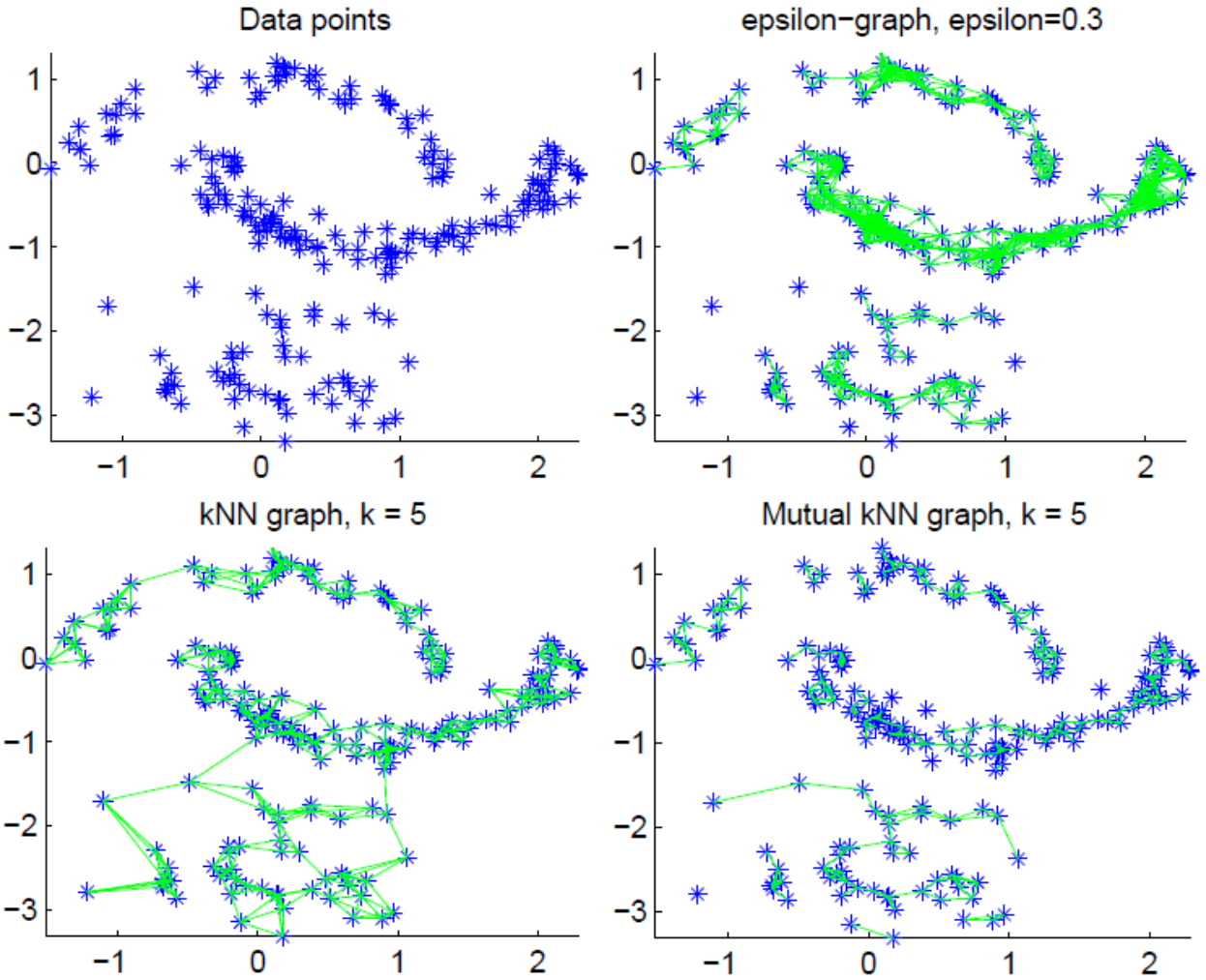


Figure 4: Different strategies to generate a similarity graph when the clusters have varying density. Not all strategies are equally useful.

Practical details

Computation of the eigenvectors

- Krylov subspace methods are particularly useful to calculate the first k eigenvectors of the graph Laplacian $\mathbf{L}$, as $\mathbf{L}$ might be large but sparse (if we choose the kNN strategy). The speed of convergence depends on the size of the eigengap $|\lambda_{k+1} - \lambda_k|$.
- It is in general not a problem if the graph is disconnected. The computed eigenvectors to the multiple root λ_1 of $\mathbf{L}$ might not be unique and depend on the implementation. However, as they are orthogonal to each other and piecewise constant on the clusters, the k-means algorithm will be able to separate them and to detect the clusters.

Practical details - the number k of clusters

- There are several statistical methods to guess the parameter k, for instance, the log-likelihood of the data.
- We can also use a heuristic for the eigengap $|\lambda_{k+1} - \lambda_k|$.

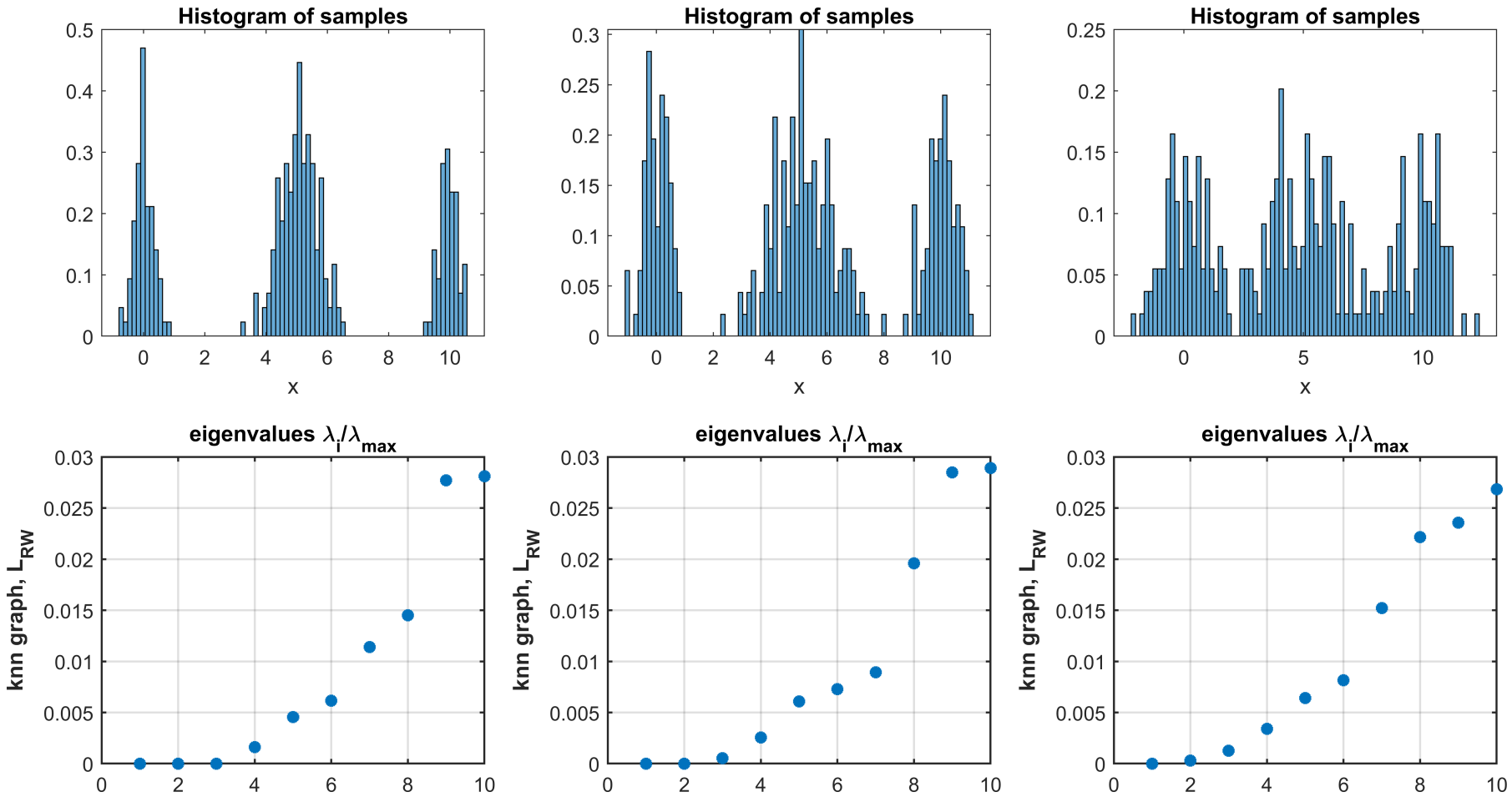


Figure 5: Behavior of first eigenvalues of $\mathbf{L}_{RW}$ for 3 different point distributions.

Practical details - which Laplacian should we choose?

- The first eigenvectors of $\mathbf{L}$ are relaxed solutions of RatioCut, while the first eigenvectors of $\mathbf{L}_{RW}$ are relaxed solutions of Ncut.
- For a large within-cluster similarity, the volume term $\mathrm{vol}(A_i)$ is more suited than the size $|A_i|$. Therefore, minimizing Ncut has an advantage compared to RatioCut. Using the Laplacian $\mathbf{L}_{RW}$ has therefore also an advantage compared to $\mathbf{L}$.
- If a graph has k connected components A_i, the first eigenvectors of the normalized Laplacian $\mathbf{L}_N$ are given by $\mathbf{D}^{1/2}\chi_{A_i}$. If the degrees of the graph are varying a lot, this can result in problems for the clustering with k-means. The additional row-normalization in Algorithm 3 is therefore necessary.
- Also with additional row-normalization, $\mathbf{L}_{RW}$ (i.e. Shi/Malik clustering) has a slight advantage compared to $\mathbf{L}_N$.

Spectral clustering vs k-means++

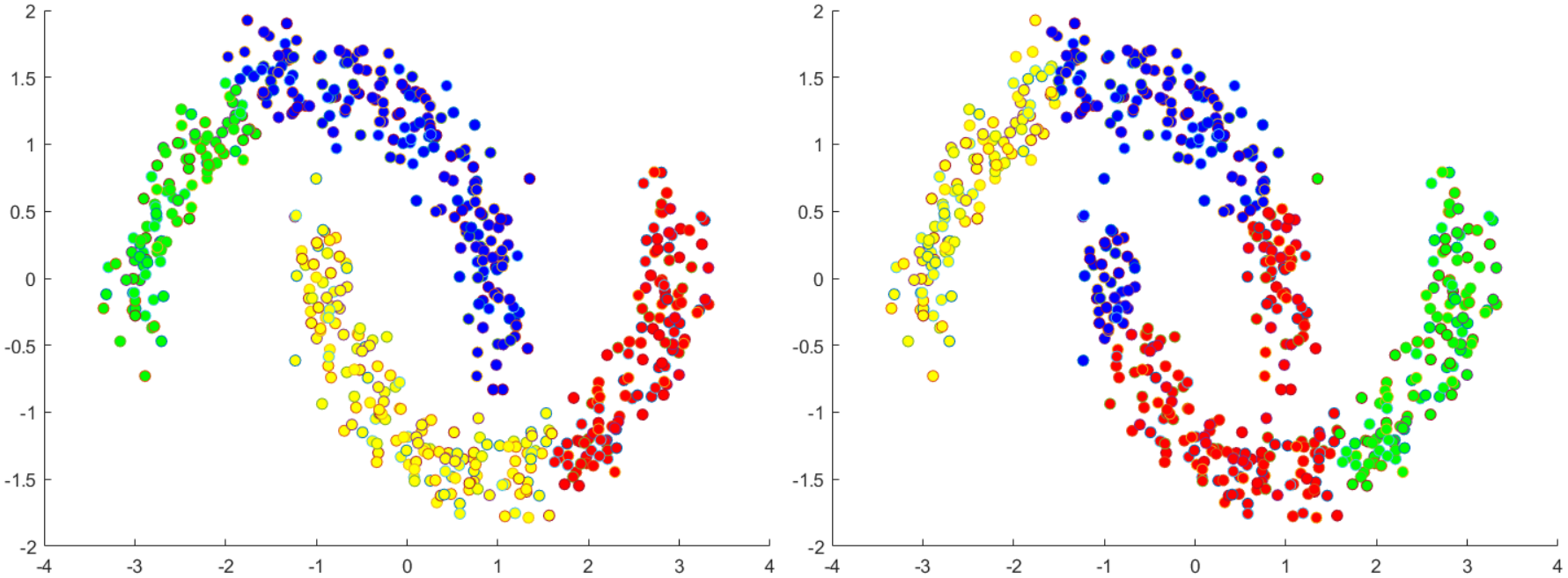
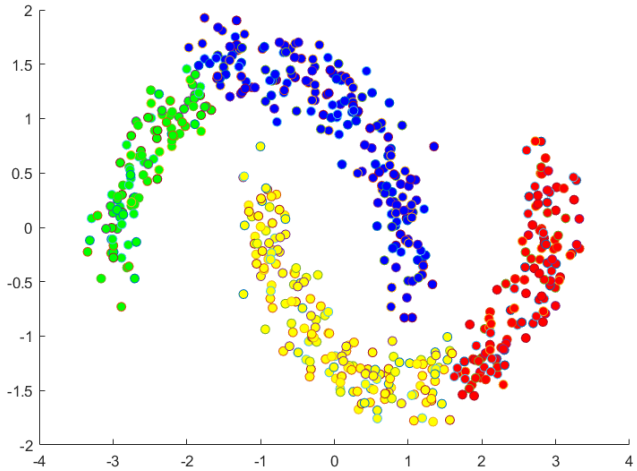


Figure 6: $k = 4$ calculated clusters with Shi/Malik spectral clustering (left) and k-means++ algorithm (right) for a 2-moon data set.

Thanks a lot for your attention!



[1] J. Shi and J. Malik.
"Normalized Cuts and Image Segmentation,"
IEEE Trans Pattern Anal. Mach. Intell., Vol. 22, No. 8, pp. 888–905, 2000.

[2] U. von Luxburg.
"A Tutorial on Spectral Clustering,"
arXiv:0711.0189v1 [cs.DS], 2007.