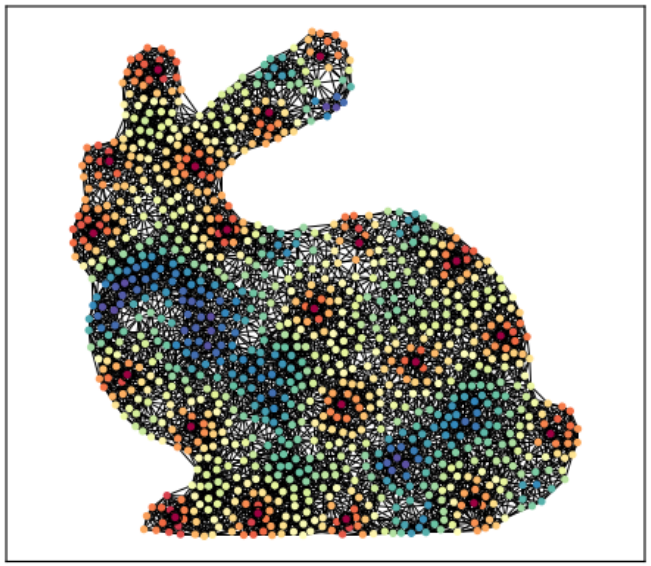
Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
Università degli Studi di Padova

Modern Computational Harmonic Analysis on Graphs and Networks

1. Introduction to spectral graph theory

Wolfgang Erb
University of Padova

C.I.M.E. Summer School
"Modern Perspectives in Approximation Theory: Graphs, Networks, quasi-interpolation and Sampling Theory"
July 21-24, 2025, Cetraro (CS), Italy



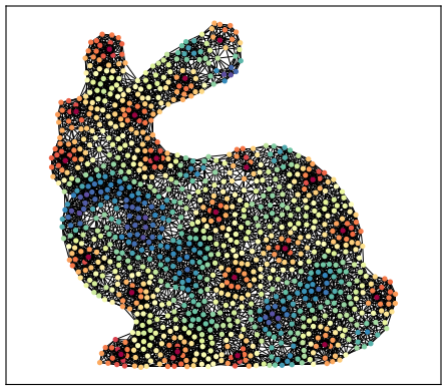
wolfgang.erb@unipd.it

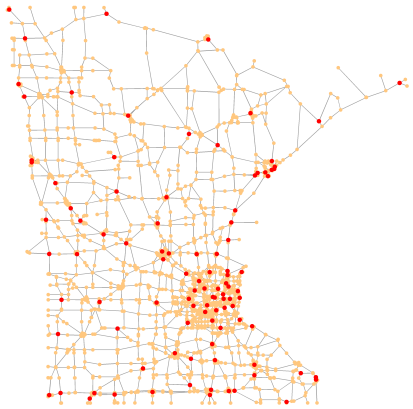
Goal of this presentation

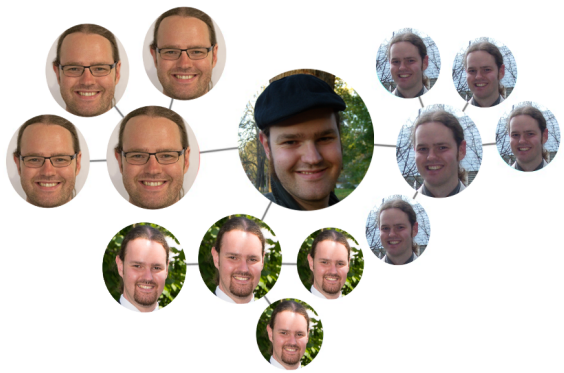
Introduction to computational methods for harmonic analysis on graphs.

1. Today: An introduction to spectral graph theory
   - The graph Laplacian and the Graph Fourier Transform (GFT).
   - Graph Convolution and graph signal processing
2. In the next days:
   - Spectral clustering and dimensionality reduction.
   - How to describe and compute uncertainty principles on graphs.
   - Kernel methods for approximation and classification on graphs based on positive definite graph basis functions.
   - Construction possibilities for adaptive wavelets on graphs.

Why are graphs interesting?







Graphs offer the possibility to model complex irregular structures and relations inside these structures.

Examples:

- Social networks: nodes = persons, edges = relations
- Transport networks: nodes = cities, edges = streets
- Meshes: nodes = mesh nodes, edges = edges of triangulation

Graphs

We consider simple and undirected graphs G given as a triplet

$$G = (V, E, \mathbf{A}),$$

with vertices

$$V = \{\mathrm{v}_1, \ldots, \mathrm{v}_n\}$$

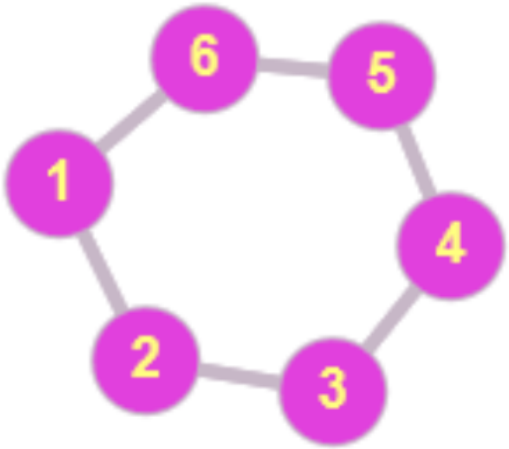
undirected edges $E \subset V \times V$ and a symmetric adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with non-negative entries

$$\begin{cases} \mathbf{A}_{i,j} > 0 \text{ if } e_{i,j} = (\mathrm{v}_i, \mathrm{v}_j) \in E, \\ \mathbf{A}_{i,j} = 0 \text{ otherwise.} \end{cases}$$

Standard $\mathbf{A}$: only entries 1 (if there is an edge) and 0.
The degree matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ is the diagonal matrix with entries

$$\begin{cases} \mathbf{D}_{i,i} = \sum_{j=1}^{n} \mathbf{A}_{i,j} \\ \mathbf{D}_{i,j} = 0 \text{ if } i \neq j. \end{cases}$$

Labeled graph

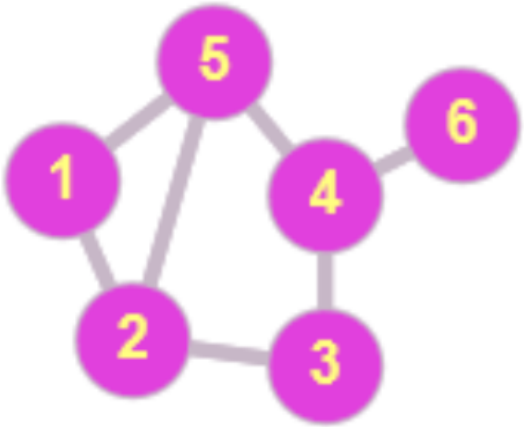


Degree matrix **D**

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

Adjacency matrix **A**

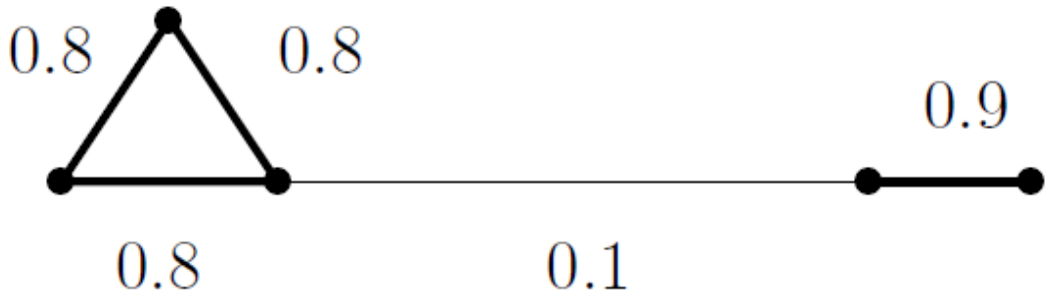
$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$



$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Example of a weighted graph



The adjacency matrix $\mathbf{A}$ for this graph is given as

$$\mathbf{A} = \begin{bmatrix} 0 & 0.8 & 0.8 & 0 & 0 \\ 0.8 & 0 & 0.8 & 0 & 0 \\ 0.8 & 0.8 & 0 & 0.1 & 0 \\ 0 & 0 & 0.1 & 0 & 0.9 \\ 0 & 0 & 0 & 0.9 & 0 \end{bmatrix}$$

For this graph we get the degree matrix $\mathbf{D} = \text{diag}(1.6, 1.6, 1.7, 1, 0.9)$.

Graph distance

The weight $\mathbf{A}_{i,j} > 0$ for the edge $e_{i,j}$ is a measure for the strength of the relation between v_i and v_j. As a respective distance, we can take the inverse $1/\mathbf{A}_{i,j}$ and define the length $\ell(e_{i,j})$ as

$$\ell(e_{i,j}) = \ell(\mathrm{v}_i, \mathrm{v}_j) = \frac{1}{\mathbf{A}_{i,j}} \quad \text{if } \mathrm{v}_i \text{ is connected with } \mathrm{v}_j.$$

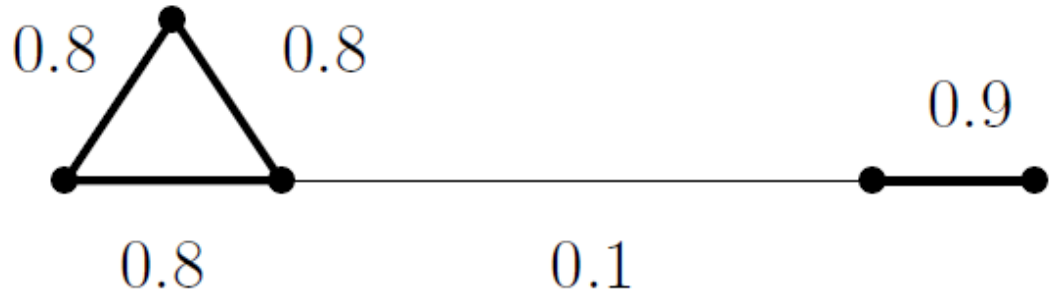
A path in G is a finite sequence $c = (\mathrm{w}_1, \ldots, \mathrm{w}_k)$ of distinct vertices for which $(\mathrm{w}_i, \mathrm{w}_{i+1}) \in E$. The length $\ell(c)$ of the path c is given by the sum of the edge lengths in the path, i.e.,

$$\ell(c) = \sum_{i=1}^{k-1} \ell((\mathrm{w}_i, \mathrm{w}_{i+1})).$$

The minimal length of a path connecting two nodes v and w is referred to as graph distance between v and w. The maximal graph distance between two nodes of G is referred to as diameter of G.

Example

We consider again the following weighted connected graph.



The three nodes on the left (and the three edges connecting them), and the two nodes on the right form connected subgraphs. The graph distance between the outer left and the outer right node of this graph is

$$\frac{1}{0.8} + \frac{1}{0.1} + \frac{1}{0.9} = 12.36111\ldots$$

This corresponds also already to the diameter of this graph.

Graph signals

Graph signals are mappings $x : V \to \mathbb{R}$ (or $x : V \to \mathbb{C}$)

- A signal x is defined on the vertices $\mathrm{v} \in V$ of the graph
- We can represent x as a vector

$$x = (x(\mathrm{v}_1), \ldots, x(\mathrm{v}_n))^* \in \mathbb{R}^n \quad (\in \mathbb{C}^n).$$

- The linear space of all signals is denoted by $\mathcal{L}(G)$.

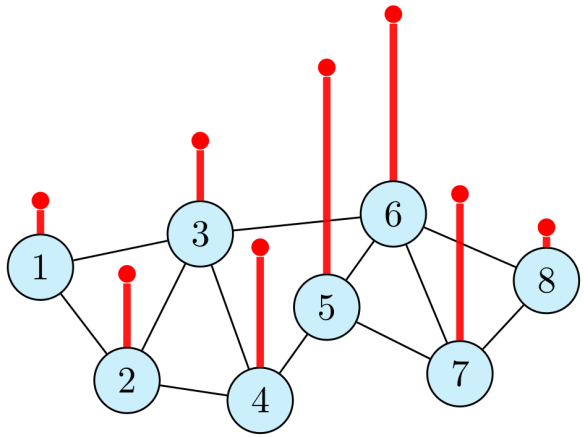


Fig.: Illustration of a graph signal x.

Graph signals

Example: brain connectivity networks:

- vertices: neural elements of the brain
- edges: pairwise relationships between elements
- graph signal: brain functional activity on vertices

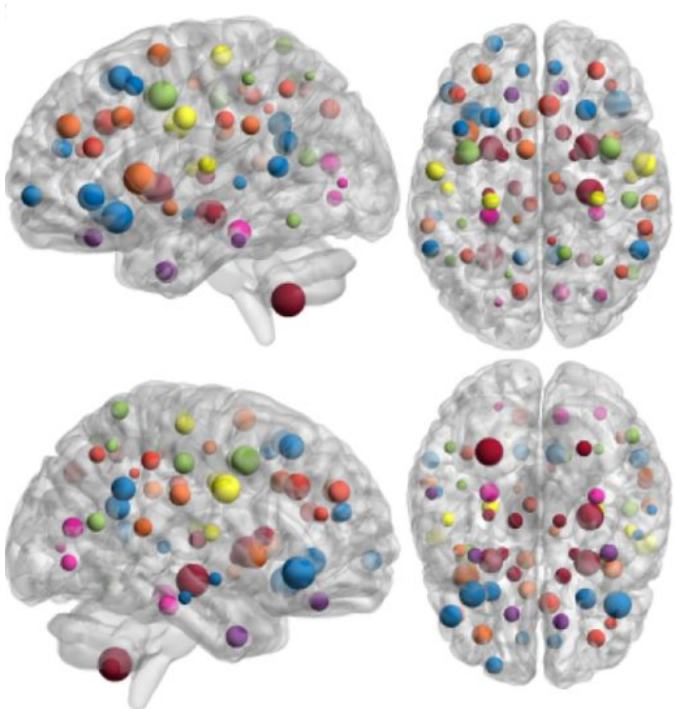
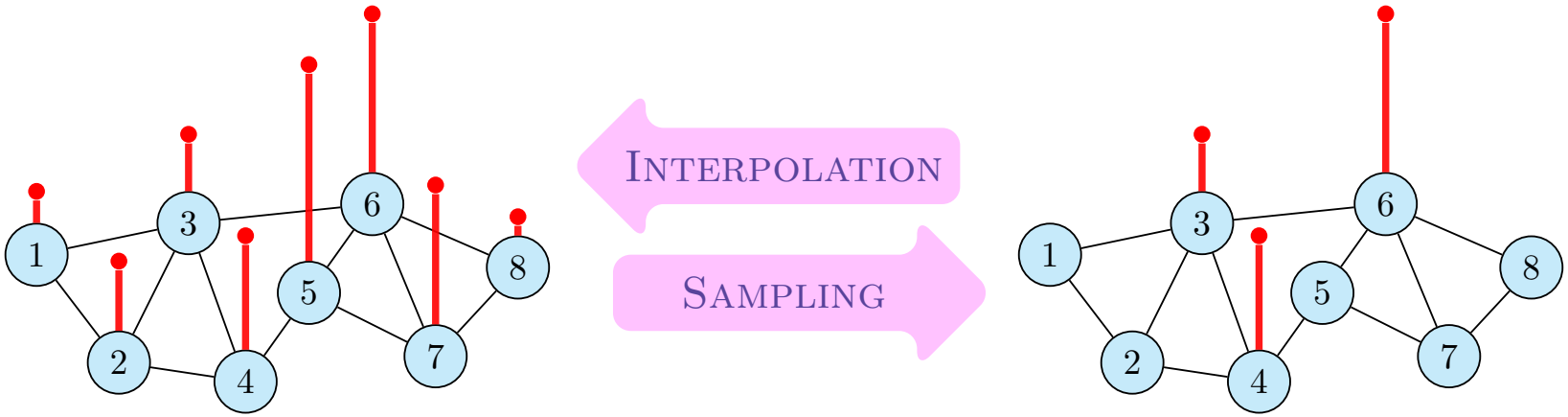


Fig.: Illustration of signals on a brain connectivity graph, Manjunatha et al. 2023

Graph signal processing (GSP)

In applications, graphs and graph signals might by huge. Necessity of efficient signal processing tools on graphs.



Goal of GSP: study of graph signals and possible processing tools

- Analysis of smoothness of signals (spaces of signals)
- Decomposition of signals (Fourier, wavelets, frames, etc.)
- Denoising of signals (convolution filters)
- Sampling and interpolation of signals
- Uncertainty principles

Graph Laplacian

The main ingredient for GSP is the graph Laplacian.

The graph Laplacian $\mathbf{L} \in \mathbb{R}^{n \times n}$ is defined as the matrix

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \quad \mathbf{L}_{i,j} := \begin{cases} \mathbf{D}_{i,i} & \text{if } i = j \\ -\mathbf{A}_{i,j} & \text{if } i \neq j \end{cases}.$$

The normalized graph Laplacian $\mathbf{L}_N \in \mathbb{R}^{n \times n}$ is defined as

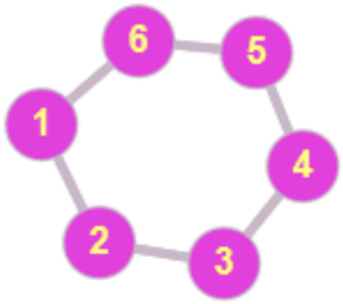
$$\mathbf{L}_N = \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} = \mathbf{I}_n - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}.$$

The random walk graph Laplacian $\mathbf{L}_{RW} \in \mathbb{R}^{n \times n}$ is defined as

$$\mathbf{L}_{RW} = \mathbf{D}^{-1} \mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-1} \mathbf{A}.$$

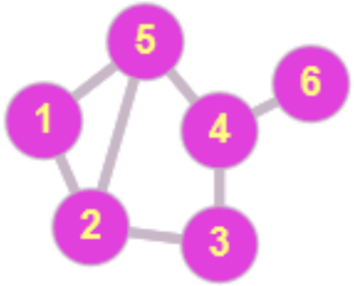
The matrices $\mathbf{L}$ and $\mathbf{L}_N$ are symmetric and positive semi-definite. The normalizations ensure that the spectrum of $\mathbf{L}_{RW}$ and $\mathbf{L}_N$ is in $[0, 2]$.

Labeled graph G



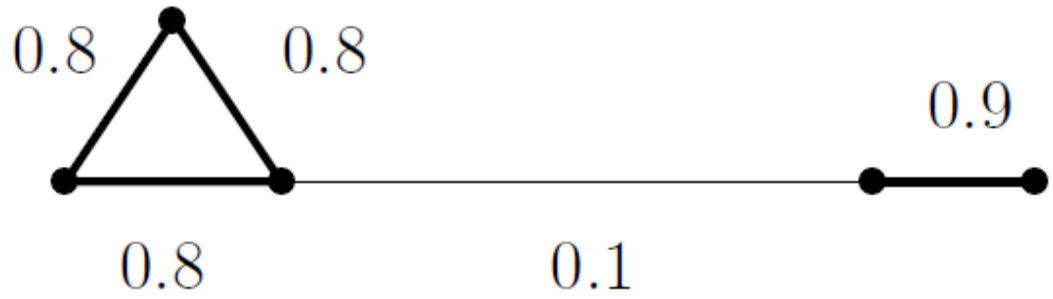
Graph Laplacian $\mathbf{L}$

$$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & 0 & -1 & 2 \end{pmatrix}$$



$$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

Example of a weighted graph



The graph Laplacian $\mathbf{L}$ for this graph is given as

$$\mathbf{L} = \begin{bmatrix} 1.6 & -0.8 & -0.8 & 0 & 0 \\ -0.8 & 1.6 & -0.8 & 0 & 0 \\ -0.8 & -0.8 & 1.7 & -0.1 & 0 \\ 0 & 0 & -0.1 & 1 & -0.9 \\ 0 & 0 & 0 & -0.9 & 0.9 \end{bmatrix}$$

Interpretation of the graph Laplacian

The graph Laplacian operates on a signal $x \in \mathcal{L}(G)$ as

$$\mathbf{L}x(\mathrm{v}_i) = \sum_{(\mathrm{v}_i,\mathrm{v}_j)\in E} \mathbf{A}_{i,j}(x(\mathrm{v}_i) - x(\mathrm{v}_j))$$

In many cases, $\mathbf{L}$ is a discretization of a continuous Laplacian

Example 1: the path graph P_n

Path graph P_n with n nodes and weights $\mathbf{A}_{i,j} = h^{-1}$.

Laplacian for P_n (interior nodes)

$$\mathbf{L}x(\mathrm{v}_i) = \frac{2x(\mathrm{v}_i) - x(\mathrm{v}_{i+1}) - x(\mathrm{v}_{i-1})}{h}$$

is a second order difference quotient that approximates $-\frac{\mathrm{d}^2x}{\mathrm{d}t^2}(t)$.

Basic properties of the graph Laplacian

Theorem 1

For the graph Laplacian $\mathbf{L} \in \mathbb{R}^{n \times n}$ *of a graph* $G = (V, E, \mathbf{A})$ *we have*

1. $\mathbf{L}$ *is symmetric;*
2. *All the rows (and columns) sum up to* 0, *i.e.* $\mathbf{L}e = 0$ *for* $e = [1, \ldots, 1]^*$. *This implies that* $\mathbf{L}$ *has an eigenvalue* $\lambda_1 = 0$ *with eigenvector* $e \in \mathbb{R}^n$.
3. *For every* $x \in \mathbb{R}^n$, *we have*

$$x^* \mathbf{L} x = \frac{1}{2} \sum_{(\mathrm{v}_i, \mathrm{v}_j) \in E} \mathbf{A}_{i,j} (x(\mathrm{v}_i) - x(\mathrm{v}_j))^2.$$

This implies that $\mathbf{L}$ *is positive semidefinite and its eigenvalues are all nonnegative:* $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

Proof of Theorem 1

The two properties 1) and 2) follow directly from the definition of the graph Laplacian. Property 3) is shown by the following short calculation:

$$\begin{aligned}\sum_{i,j=1}^{n} \mathbf{A}_{i,j} \left(x(\mathrm{v}_i) - x(\mathrm{v}_j)\right)^2 &= \sum_{i,j=1}^{n} \mathbf{A}_{i,j} x(\mathrm{v}_i)^2 + \sum_{i,j=1}^{n} \mathbf{A}_{i,j} x(\mathrm{v}_j)^2 - 2 \sum_{i,j=1}^{n} \mathbf{A}_{i,j} x(\mathrm{v}_i) x(\mathrm{v}_j) \\ &= \sum_{i=1}^{n} \mathbf{D}_{i,i} x(\mathrm{v}_i)^2 + \sum_{j=1}^{n} \mathbf{D}_{j,j} x(\mathrm{v}_j)^2 - 2 \sum_{i,j=1}^{n} \mathbf{A}_{i,j} x(\mathrm{v}_i) x(\mathrm{v}_j) \\ &= 2x^* \mathbf{D} x - 2x^* \mathbf{A} x \\ &= 2x^* (\mathbf{D} - \mathbf{A}) x = 2x^* \mathbf{L} x.\end{aligned}$$

□

Theorem 2

1) For the normalized graph Laplacian $\mathbf{L}_N \in \mathbb{R}^{n \times n}$ *we have*

- $\mathbf{L}_N$ *is symmetric and*

$$x^* \mathbf{L}_N x = \frac{1}{2} \sum_{(v_i, v_j) \in E} \mathbf{A}_{i,j} \left(\frac{x(v_i)}{\mathbf{D}_{i,i}^{1/2}} - \frac{x(v_j)}{\mathbf{D}_{j,j}^{1/2}} \right)^2 .$$

This implies that $\mathbf{L}_N$ *is positive semidefinite and its eigenvalues are all nonnegative.*

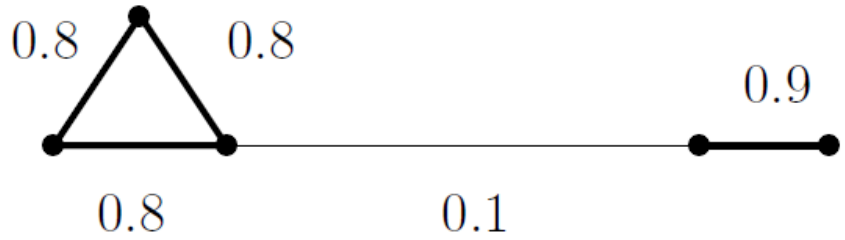
2) The eigenvalues of $\mathbf{L}_{RW}$ *correspond to the eigenvalues of* $\mathbf{L}_N$ *and*

(i) $\sum_{k=1}^{n} \lambda_i = n$

(ii) $\lambda_i \in [0, 2]$ *for all* $i \in \{1, \ldots, n\}$.

Example

The graph Laplacian $\mathbf{L}$ of the graph



is given as

$$\mathbf{L} = \begin{bmatrix} 1.6 & -0.8 & -0.8 & 0 & 0 \\ -0.8 & 1.6 & -0.8 & 0 & 0 \\ -0.8 & -0.8 & 1.7 & -0.1 & 0 \\ 0 & 0 & -0.1 & 1 & -0.9 \\ 0 & 0 & 0 & -0.9 & 0.9 \end{bmatrix}.$$

The eigenvalues of $\mathbf{L}$ are $0 < 0.0788 < 1.8465 < 2.4000 < 2.4747$. The eigenvalues of $\mathbf{L}_N$ and $\mathbf{L}_{RW}$ are $0 < 0.0693 < 1.4773 < 1.5000 < 1.9534$.

Example

We consider a modified graph (having two connected components) with the adjacency matrix

$$\mathbf{A} = \begin{bmatrix} 0 & 0.8 & 0.8 & 0 & 0 \\ 0.8 & 0 & 0.8 & 0 & 0 \\ 0.8 & 0.8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9 \\ 0 & 0 & 0 & 0.9 & 0 \end{bmatrix}$$

It is easy to check that the characteristic polynomial of the corresponding graph Laplacian is given as

$$\det(\lambda \mathbf{I} - \mathbf{L}) = \lambda^2(\lambda - 2.4)^2(\lambda - 1.8).$$

Thus, the graph Laplacian $\mathbf{L}$ of the modified graph has a repeated eigenvalue $\lambda_1 = \lambda_2 = 0$ with multiplicity 2 (which is equal to the number of connected components).

Theorem 3

The multiplicity of the zero eigenvalue of the Laplacians $\mathbf{L}$, $\mathbf{L}_N$ *and* $\mathbf{L}_{RW}$ *equals the number of connected components of the graph* G.

Interpretation

- This is a first example in which we see that the graph Laplacian contains relevant information about the geometric structure of a graph. The multiplicity of the zero eigenvalue gives automatically the number of connected components.
- If only one eigenvalue of $\mathbf{L}$ is zero and there are $k-1$ eigenvalues of $\mathbf{L}$ that are very close to zero, we get a computational indication that the graph G consists of k components that are only weakly connected between each other. This is a theoretical explanation of why spectral graph clustering works.

Proof. Part 1)

1) The multiplicity of $\lambda_1 = 0$ is at least the number k of connected components of G.

Assume that the connected components correspond to the partition of V into disjoint sets $V_1, \ldots, V_k$. Define k vectors $w_1, \ldots, w_k$ s.t.

$$w_i(\mathrm{v}_j) = 1/\sqrt{|V_i|} \text{ if } \mathrm{v}_j \in V_i, \text{ and } 0 \text{ otherwise.}$$

For $i \in \{1, \ldots, k\}$, we have $\|w_i\|_2 = 1$. Additionally, for $i \neq j$, we have $\langle w_i, w_j \rangle = 0$. Finally, $\mathbf{L}w_i = 0$ holds true.

Hence there is a set of k orthonormal vectors that are all eigenvectors of $\mathbf{L}$, with respect to the eigenvalue 0.

Proof, part 2)

2) To see that the multiplicity of $\lambda_1 = 0$ is at most k, we consider

$$x^* \mathbf{L} x = \frac{1}{2} \sum_{(v_i, v_j) \in E} \mathbf{A}_{i,j} (x(v_i) - x(v_j))^2.$$

This expression can only be zero if x is constant on every connected component. To see that there is no way of finding a $k+1$st vector x that is a zero eigenvector, orthogonal to $w_1, \ldots, w_k$, observe that any eigenvector x must be nonzero on some node. Hence, we can assume that x is nonzero on a node in V_i, and, thus, nonzero and constant on all nodes in V_i, in which case x can not be orthogonal to w_i. Therefore, there can be no $k+1$st eigenvector with eigenvalue 0. □

The Cheeger constant

The Cheeger constant h_G of a connected graph G is a measure on how well the vertices of G can be separated in two connected components by removing edges of G. If the constant h_G is small, the graph G has a bottleneck so that it is easy to split the graph in two parts.

- The volume of the set W is defined as

$$\mathrm{vol}(W) = \sum_{i\,:\,\mathrm{v}_i \in W} \mathbf{D}_{i,i}.$$

  The volume of the entire vertex set V is then $\mathrm{vol}(V) = \sum_{i=1}^{n} \mathbf{D}_{i,i}$.
- The set of edges of G having endpoints in W_1 and W_2 is

$$E(W_1, W_2) = \{(\mathrm{w}_1, \mathrm{w}_2) \in E \ : \ \mathrm{w}_1 \in W,\ \mathrm{w}_2 \in W_2)\} \subset E.$$

  The size of $E(W_1, W_2)$ is given as

$$|E(W_1, W_2)| = \sum_{i,j\,:\,\mathrm{v}_i \in W_1, \mathrm{v}_j \in W_2} \mathbf{A}_{i,j}.$$

  If $\mathbf{A}_{i,j} \in \{0, 1\}$, then $|E(W_1, W_2)|$ corresponds to the number of edges between W_1 and W_2.

The Cheeger constant

For a subset $W \subset V$, we define the Cheeger ratio of W as

$$h_W = \frac{|E(W, W^c)|}{\min\{\mathrm{vol}(W), \mathrm{vol}(W^c)\}},$$

where $W^c = V \setminus W$ is the complement of the set W in V. For the full vertex set $W = V$, we set $h_V = 1$.

The Cheeger constant of the graph G is now given as the minimum possible ratio

$$h_G = \min_{W \subset V} h_W.$$

The Cheeger inequality

Theorem 4 (Cheeger inequality)

Let G be connected. Then, the Cheeger constant h_G and the second eigenvalue $\lambda_2 > 0$ of the normalized graph Laplacian $\mathbf{L}_N$ are linked as

$$\frac{h_G^2}{2} \leq \lambda_2 \leq 2h_G.$$

The Cheeger inequality for graphs is a discrete analog of the classical Cheeger inequality for Riemannian manifolds. It relates the Cheeger constant h_G to the second smallest eigenvalue $\lambda_2 > 0$ of $\mathbf{L}_N$. From this result, we see that the size of the second eigenvalue $\lambda_2 > 0$ of $\mathbf{L}_N$ provides relevant information on how easily a graph can be split in two connected components by removing edges of G.

Proof of the upper estimate

We assume that the minimum Cheeger ratio h_G is attained by a set W with $\mathrm{vol}(W) \leq \mathrm{vol}(V)/2$ such that

$$h_G = h_W = \frac{|E(W, W^C)|}{\mathrm{vol}(W)}.$$

Consider the graph signal x given as

$$x(\mathrm{v}_i) = \chi_W(\mathrm{v}_i) - \frac{\mathrm{vol}(W)}{\mathrm{vol}(V)}.$$

Then, $x^* \mathbf{D} e = \chi_W^* \mathbf{D} e - \frac{\mathrm{vol}(W)}{\mathrm{vol}(V)} e^* \mathbf{D} e = \mathrm{vol}(W) - \mathrm{vol}(W) = 0$. Thus, the Rayleigh characterization of the eigenvalue λ_2 yields

$$\lambda_2 \leq \frac{x^* \mathbf{L} x}{x^* \mathbf{D} x} = \frac{\chi_W^* \mathbf{L} \chi_W}{\mathrm{vol}(W) - \frac{\mathrm{vol}(W)^2}{\mathrm{vol}(V)}} = \frac{|E(W, W^c)|}{\mathrm{vol}(W)(1 - \frac{\mathrm{vol}(W)}{\mathrm{vol}(V)})} \leq 2h_G.$$

This proves the upper bound in the Cheeger inequality. □

The Graph Fourier Transform

Spectral decomposition of the Laplacian

For the symmetric positive-semidefinite matrix $\mathbf{L}$ there exists a set

$$\{u_1, u_2, \ldots, u_n\}$$

of real-valued orthonormal eigenvectors with respect to the eigenvalues $0 = \lambda_1 \leq \cdots \leq \lambda_n$. This set is an orthonormal basis of the space $\mathcal{L}(V)$ endowed with the inner product $\langle x, y \rangle = \sum_{i=1}^{n} x(v_i)\overline{y(v_i)}$

In other words, the graph Laplacian $\mathbf{L}$ has a spectral decomposition

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^*,$$

where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix with the increasingly ordered eigenvalues of $\mathbf{L}$ on the diagonal and $\mathbf{U}$ is a unitary matrix given by

$$\mathbf{U} = [u_1 \; u_2 \; \ldots \; u_n].$$

Similar decompositions hold true for the normalized matrices $\mathbf{L}_N$ and $\mathbf{L}_{RW}$.

The Laplacian and the Fourier transform on graphs

Graph Fourier transform

Idea: use the orthonormal eigenvectors

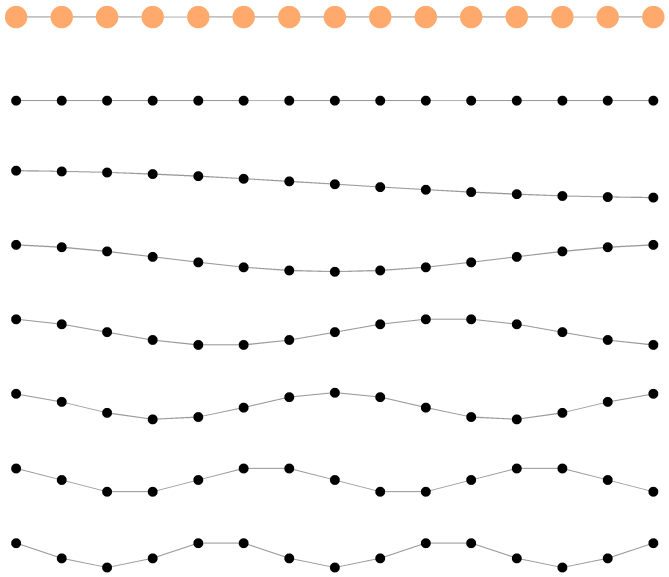
$$\mathbf{L}u_k = \lambda_k u_k, \quad k \in \{1, \dots, n\},$$

of the graph Laplacian $\mathbf{L}$ as Fourier basis and set

$$\hat{x}_k = \langle x, u_k \rangle = \sum_{i=1}^{n} x(\mathrm{v}_i)\overline{u_k(\mathrm{v}_i)}.$$

Note: also on the real line the Fourier components of a function f can be interpreted as the inner product of the function f with the eigenfunctions of the Laplace operator.

Example 1: the path graph P_n



Path graph

1. eigenvector
2. eigenvector
3. eigenvector
4. eigenvector
5. eigenvector
6. eigenvector
7. eigenvector

The eigenvalues and eigenfunctions can be written explicitly as

$$\lambda_k = 2 - 2\cos\left(\frac{(k-1)\pi}{n}\right)$$

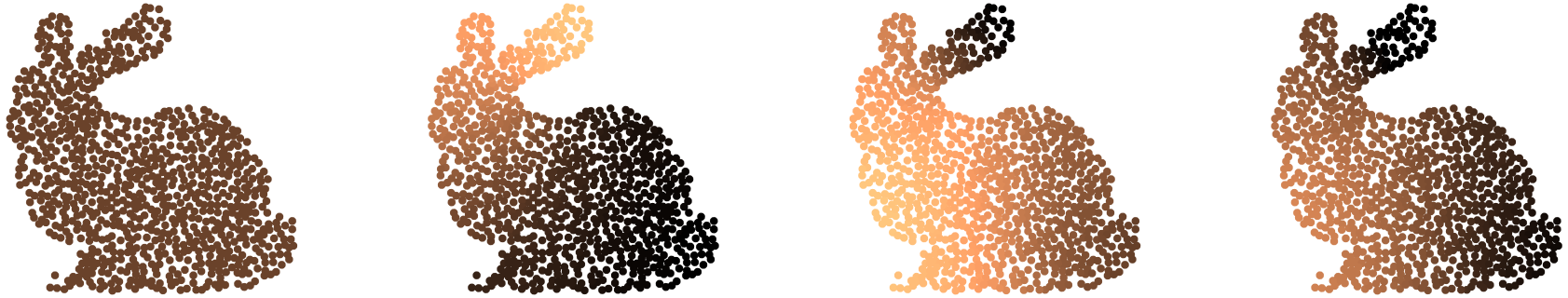
and

$$u_1 = \frac{1}{\sqrt{n}}, \quad u_k(v_i) = \sqrt{\frac{2}{n}}\cos\left(\frac{(k-1)\pi(i-0.5)}{n}\right), \; k \geq 2.$$

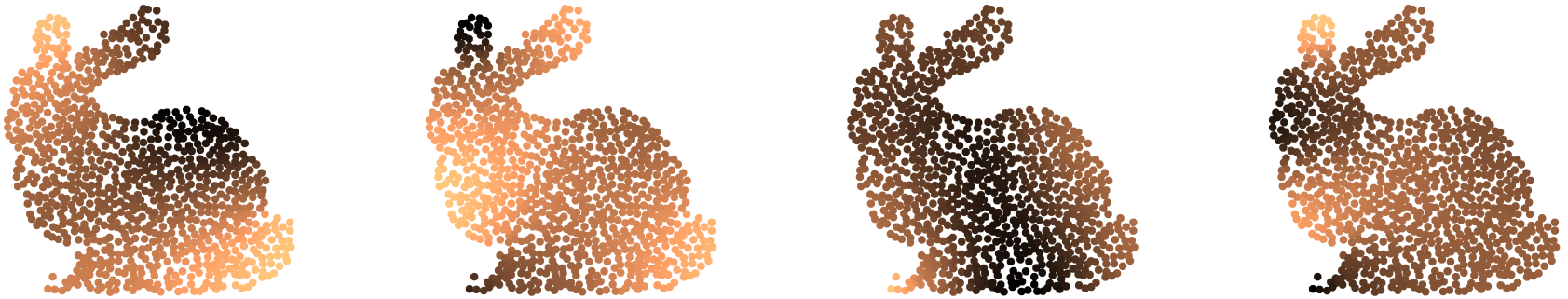
The graph Fourier transform corresponds in this case to the Discrete Cosine Transform (DCT II).

Example 2: the bunny graph

$\lambda_1 = 0$ $\lambda_2 = 0.0554$ $\lambda_3 = 0.1595$ $\lambda_4 = 0.2350$



$\lambda_5 = 0.2976$ $\lambda_6 = 0.3913$ $\lambda_7 = 0.6129$ $\lambda_8 = 0.6147$



The first 8 eigenfunctions of the graph Laplacian **L** on the bunny graph.

Example 3: circle graphs

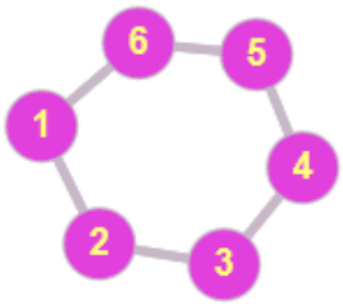
We consider the circle graph $C_n = \{1, \ldots, n\}$ with the set of edges given as

$$E = \{(i,j) \in C_n \times C_n : |i-j| = 1 \mod n\}.$$

In fact C_n can also be considered as a group (the cyclic group $\mathbb{Z}/n\mathbb{Z}$ with generating element 1.)
As adjacency matrix we have

$$\mathbf{A}_{i,j} = \begin{cases} 1 \text{ if } (i,j) \in E, \\ 0 \text{ otherwise.} \end{cases}$$

Cyclic group C_6



Graph Laplacian $\mathbf{L}$ of C_6

$$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & 0 & -1 & 2 \end{pmatrix}$$

Example 3: circle graphs

The graph Laplacian $\mathbf{L}$ of C_n is a circulant matrix and the normalized characters

$$u_k(i) = \frac{1}{\sqrt{n}} \exp\left(\frac{\imath 2\pi ik}{n}\right), \quad k \in \{1, \ldots, n\},$$

form a complete orthonormal eigenbasis of $\mathbf{L}$ with respect to the eigenvalues

$$\lambda_k = 2 - 2\cos\left(\frac{2\pi k}{n}\right).$$

The graph Fourier transform corresponds in this case to the discrete Fourier transform. Note that

$$\lambda_k = \lambda_{n-k},$$

i.e., for cyclic groups the eigenspaces of $\mathbf{L}$ are degenerate. In particular, also for general graphs G we can not expect to have unique basis elements u_k for the Fourier transform.

Interpretation of the graph Fourier transform

The GFT can be interpreted similarly as a classical Fourier transform.

- the eigenvalues of the Laplacians $\mathbf{L}$, $\mathbf{L}_N$, can be interpreted as frequencies, i.e., the larger the eigenvalue the higher the frequency of the respective eigenvector.
- The eigenvectors associated with large eigenvalues oscillate rapidly while the eigenvectors associated with small eigenvalues vary slowly.
- The eigenvector associated to the eigenvalue 0 is constant (for $\mathbf{L}$).

The Fourier transform $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_n)^*$ can be interpreted as the decomposition of a graph signal x into its single frequency components

$$x(\mathrm{v}_i) = \sum_{k=1}^{n} \hat{x}_k \, u_k(\mathrm{v}_i).$$

- The lower frequencies compose smooth part of the signal.
- The higher frequencies build the noisy part of the signal.

Matrix formulation of the GFT

As the grah Laplacian $\mathbf{L}$ is symmetric and positive semi-definite, we can write its eigendecomposition as

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^*,$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_n)$ contains the eigenvalues of $\mathbf{L}$ (increasingly ordered) and the unitary matrix $\mathbf{U} = [u_1 \; u_2 \; \cdots \; u_n]$ the corresponding eigenvectors.

Then, we can write the Graph Fourier transform of x as

$$\hat{x} = \mathbf{U}^* x, \quad \text{with } k\text{-th. entry} \quad \hat{x}_k = u_k^* x = \langle x, u_k \rangle.$$

The inverse Fourier transform is correspondingly given as

$$x = \mathbf{U}\hat{x}.$$

Numerical calculation of the GFT

Computational costs for the full GFT:

- $\mathcal{O}(n^3)$ for the calculation of the spectral decomposition of $\mathbf{L}$. Numerically, the Schur decomposition of $\mathbf{L}$ is calculated which, as $\mathbf{L}$ is symmetric, corresponds to the spectral decomposition;
- $\mathcal{O}(n^2)$ for GFTs and inverse GFTs (matrix-vector products).

For large graphs ($n >> 1$), the calculation of $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^*$ is too expensive. The following strategies allow to reduce the costs:

- If only the eigenpairs with respect to the smallest eigenvalues are required, we can use Krylov subspace methods;
- For particular operations, it is possible to avoid the spectral decomposition (e.g. in case of a convolution);
- For particular graphs (path graph, circle graph) the spectral decomposition of the Laplacian is explicitly known and fast algorithms are available (FFT, DCT, DST).

Convolution on graphs

Convolution in $\mathbb{R}$

$$(x * y)(s) = \int_{\mathbb{R}} x(t)y(s-t)\mathrm{d}t.$$

In the Fourier domain:

$$\widehat{(x * y)}(\omega) = \hat{x}(\omega)\hat{y}(\omega)$$

Graph convolution

No translation available
Idea: define convolution
via graph Fourier transform

$$\widehat{(x * y)}_k = \hat{x}_k \hat{y}_k$$

We define the graph convolution as

$$y * x := \mathbf{U}\mathbf{M}_{\hat{y}}\hat{x} = \mathbf{U}\mathbf{M}_{\hat{y}}\mathbf{U}^* x, \quad \text{where } \mathbf{M}_{\hat{y}} = \operatorname{diag}(\hat{y}_1, \dots \hat{y}_n).$$

Further, we define the convolution matrix $\mathbf{C}_y \in \mathbb{R}^{n \times n}$ as

$$\mathbf{C}_y = \mathbf{U}\mathbf{M}_{\hat{y}}\mathbf{U}^*.$$

Note: the graph convolution depends on the choice of the basis u_k.

Convolution on graphs

We can use the graph convolution to define filter functions on graphs. For a filter $f \in \mathcal{L}(V)$ and $x \in \mathcal{L}(V)$ the filtered signal $f * x$ is given as

$$(f * x)(v_i) = \mathbf{C}_f x(v_i) = \sum_{k=1}^{n} \hat{f}_k \, \hat{x}_k \, u_k(v_i).$$

- A filter function f with $\hat{f}_k = 0$ for $k \geq N < n$ is called low-pass filter.
- A filter function f with $\hat{f}_k = 0$ for $k < N < n$ is called high-pass filter.

Note: The calculation of $f * x$ via the GFT might be too costly. For particular filter functions f we can avoid the usage of the GFT.

Convolution on graphs via graph Laplacian

If the filter function is defined in terms of a polynomial of $\mathbf{L}$, the filtered signal can be obtained without the usage of the spectral decomposition.

For this, define the filter f in terms of the Fourier transform:

$$\hat{f}_k = \sum_{l=0}^{N} \alpha_l \lambda_k^l = p_N(\lambda_k), \quad N \leq n.$$

Then, the filtered signal is given as

$$\begin{aligned} f * x &= \mathbf{U}\,\mathrm{diag}(\hat{f})\mathbf{U}^* x = \mathbf{U}\,\mathrm{diag}(p_N(\lambda_1), \ldots, p_N(\lambda_1))\mathbf{U}^* x \\ &= p_N(\mathbf{U}\,\mathrm{diag}(\lambda_1, \ldots, \lambda_n)\mathbf{U}^*)x \\ &= p_N(\mathbf{L})x = \sum_{l=0}^{N} \alpha_l \mathbf{L}^l x, \end{aligned}$$

i.e., the filtered signal $f * x$ is an element of the Krylov space $\mathcal{K}^{(N+1)}(\mathbf{L}, x)$. To calculate the filtered signal, only matrix-vector products are necessary.

Design of filters in the Krylov space $\mathcal{K}^{(N+1)}(\mathbf{L}, x)$

Strategy for the construction of the filters:

- Construct the polynomial $p_N(\lambda)$ in the spectral domain (i.e. on the eigenvalues of $\mathbf{L}$) such that it satisfies all desired filtering properties (low-pass, high-pass, band-pass etc.) when multiplied to the Fourier coefficients $\hat{x}$ of a signal x.
- Use a "good" basis to represent the polynomial. The monomial basis λ^l, $l \in \{0, \dots, N\}$ is in general not that well suited for this purpose, better use a properly dilated and shifted Chebyshev basis $T_l(\lambda)$.

Calculate the filtered signal as

$$f * x = p_N(\mathbf{L})x = \sum_{l=0}^{N} \beta_l T_l(\mathbf{L})x.$$

The vector $T_l(\mathbf{L})x$ can be generated efficiently by a three-term recurrence.

Application: denoising of graph signals

Question: how can we remove the noisy part of a signal?

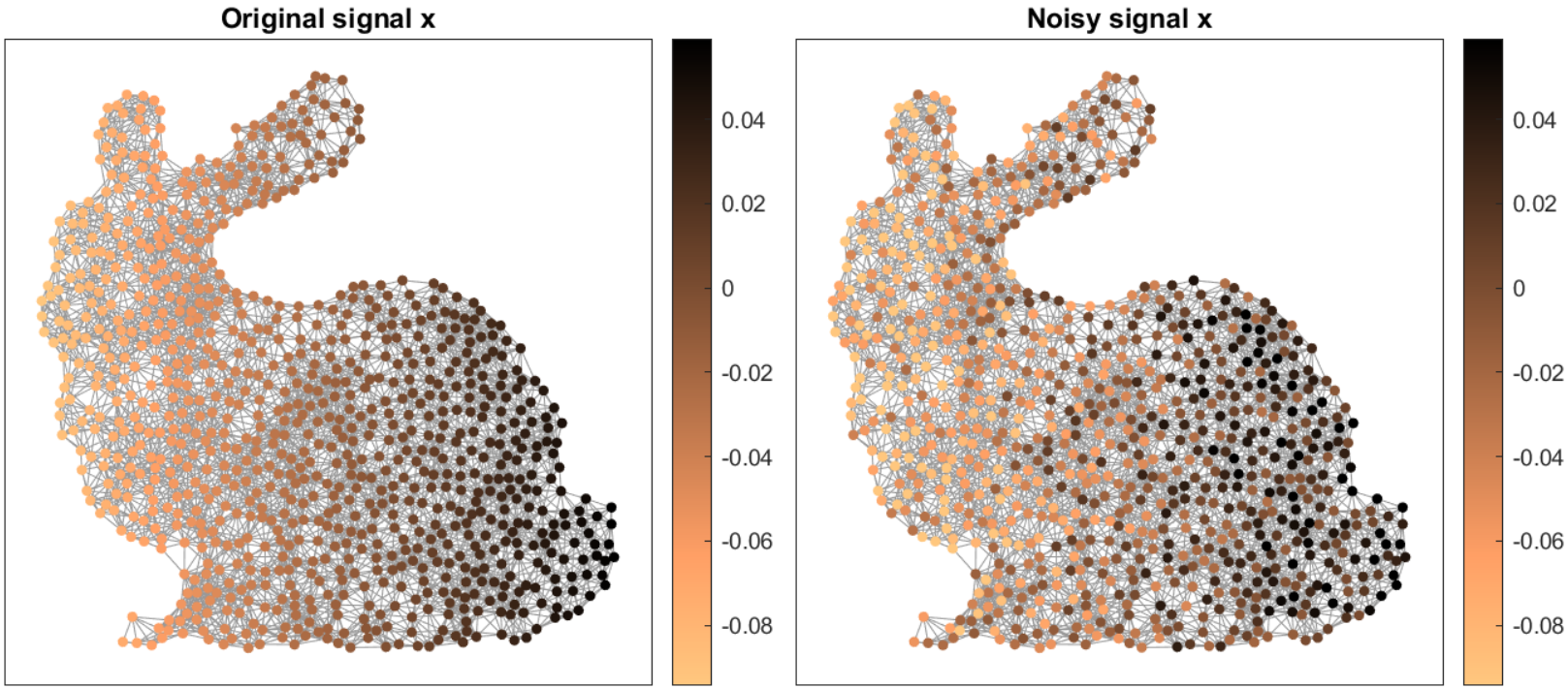


Figure 1: The original signal x and a noisy signal x_{noise} on the bunny graph.

Application: denoising of graph signals

To identify the noisy part of the signal x_{noise}, let's have a look at the Graph Fourier transform of x and x_{noise}.

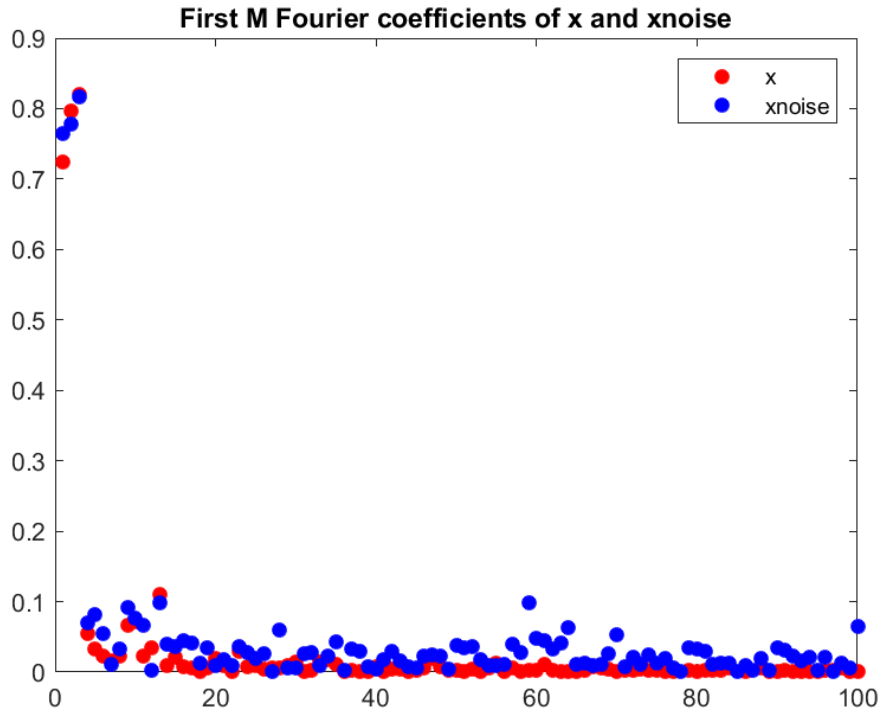


Figure 2: The size of the first 100 Fourier coefficients of the original signal x and the noisy signal x_{noise} on the bunny graph. We see that starting from $k \approx 15$, the frequency components $(\hat{x}_{\text{noise}})_k$ are much larger than for $\hat{x}_k$.

Application: denoising of graph signals

Idea for the filter function y: generate y such that all frequencies $(\hat{x}_{\text{noise}})_k$, $k \geq 21$, of x_{noise} are removed, i.e., calculate

$$x_{\text{denoised}} = y * x_{\text{noise}},$$

where y is a low-pass filter that cuts off all frequencies larger than $k = 20$.

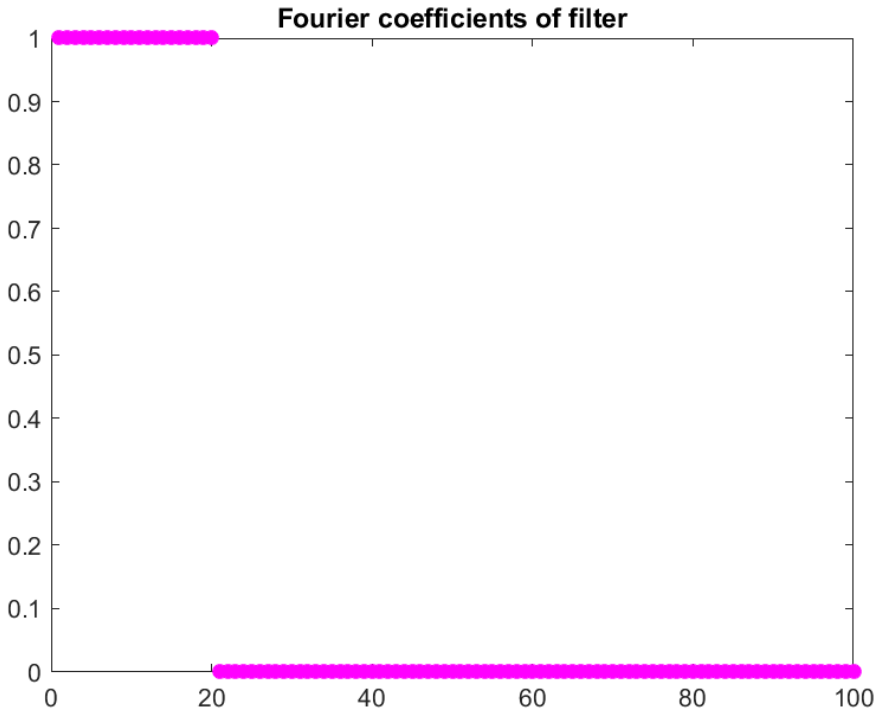


Figure 3: The Fourier coefficients of the low-pass filter y.

Application: denoising of graph signals

Result: denoised signal x_{denoised} calculated in terms of the Fourier coefficients

$$(\hat{x}_{\text{denoised}})_k = \begin{cases} \hat{x}_k & \text{for } k \leq 20, \\ 0 & \text{for } k > 20. \end{cases}$$

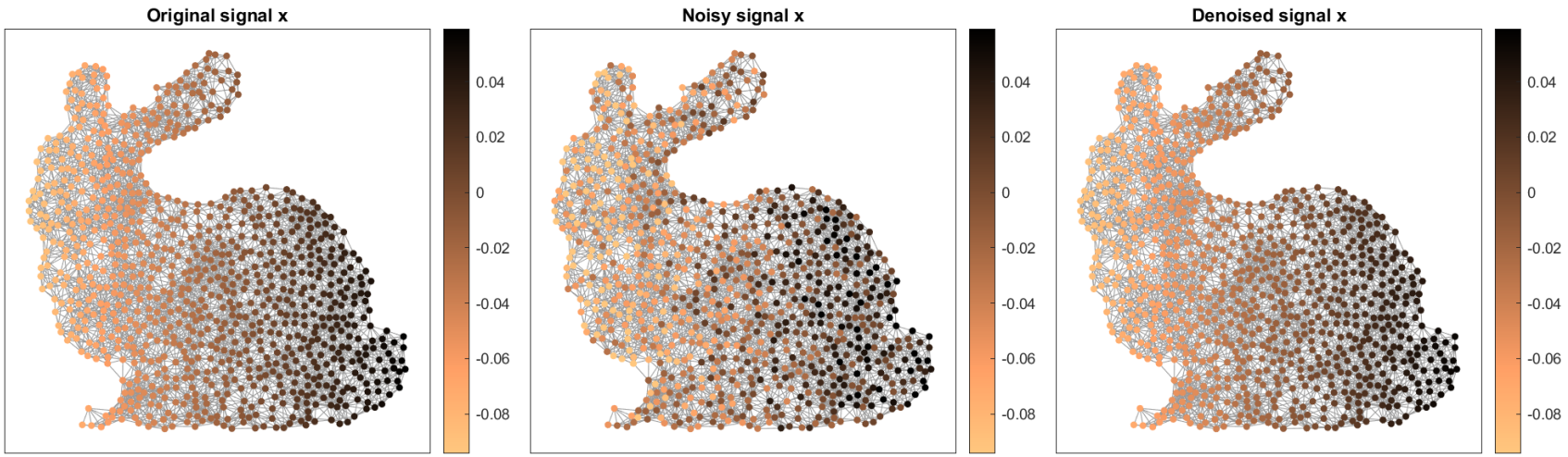


Figure 4: The original signal x, the noisy signal x_{noise} and the denoised x_{denoised}.

Application II: graph convolutional neural networks (GCNs)

Graph convolutional neural networks (GCNs) are deep neural networks on graphs in which the single layers are given as follows:

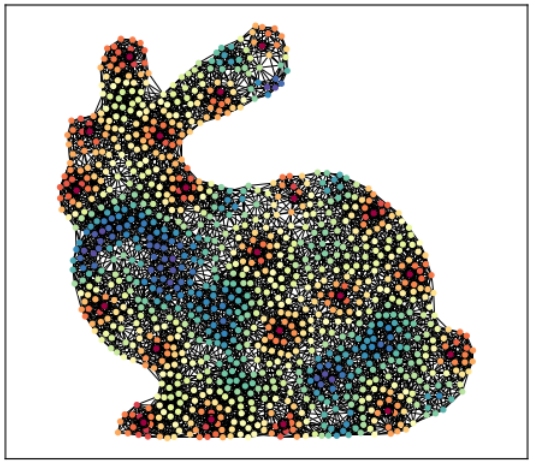
$$\mathbf{H}^{(k+1)} = \mathrm{ReLU}\left(\sum_{l=0}^{N} \alpha_l^{(k)} \mathbf{L}^l \mathbf{H}^{(k)} \Theta^{(k)}\right), \quad 0 \leq k \leq K-1,$$

where

- $\mathbf{H}^{(k)} \in \mathbb{R}^{n \times m}$ is the output of the k-th. layer.
- $\mathbf{H}^{(0)} = \mathbf{X}$ corresponds to the input (set of m feature vectors).
- $\Theta^{(k)} \in \mathbb{R}^{m \times m}$ are weights to be learned.
- $\alpha_i^{(k)} \in \mathbb{R}$ are coefficients to be learned.

In each layer, a graph convolution is applied to the previous output $\mathbf{H}^{(k)}$. The filters of the GCN are obtained by a learning procedure.

Thanks a lot for your attention!



General introductions to spectral graph theory and GSP

[1] F. Chung, *Spectral Graph Theory*, 2th edition, AMS, Providence, RI, 1997 (Chapter 1)

[2] A. Ortega, *Introduction to Graph Signal Processing*, Cambridge University Press (2022)

[3] D. Shuman et al., *The emerging field of signal processing on graphs. Extending high-dimensional data analysis to networks and other irregular domains*, IEEE Signal Processing Magazine 30,3 (2013), 83-98.