Graph signal processing

Concepts, tools and applications

Xiaowen Dong

Department of Engineering Science

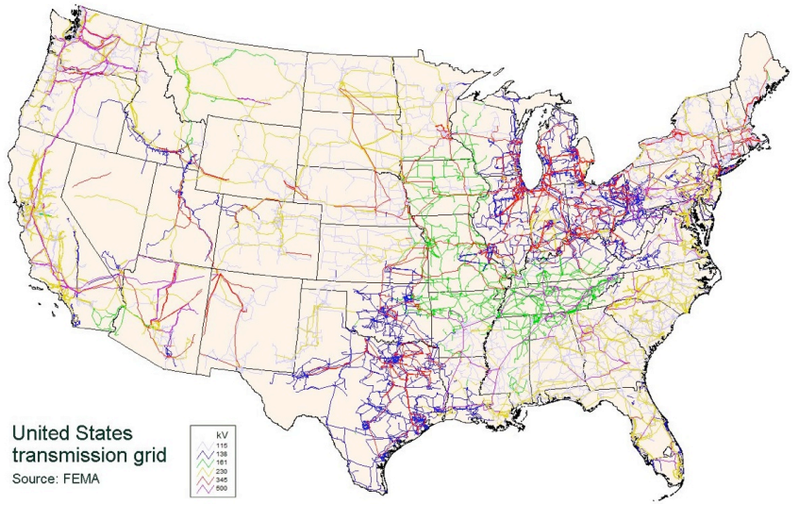
Outline

- Motivation
- Graph signal processing (GSP): Basic concepts
- Spectral filtering: Basic tools of GSP
- Applications and perspectives

Outline

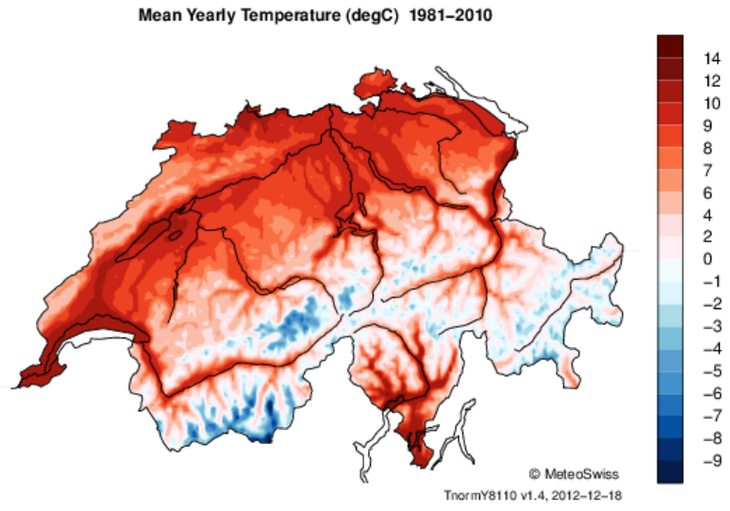
- Motivation
- Graph signal processing (GSP): Basic concepts
- Spectral filtering: Basic tools of GSP
- Applications and perspectives

Data are often structured

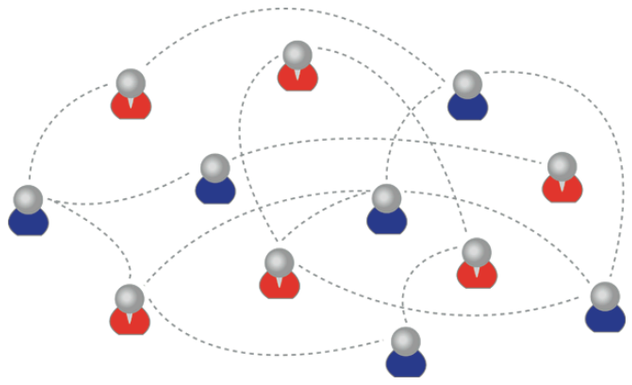


Electrical data

Traffic data

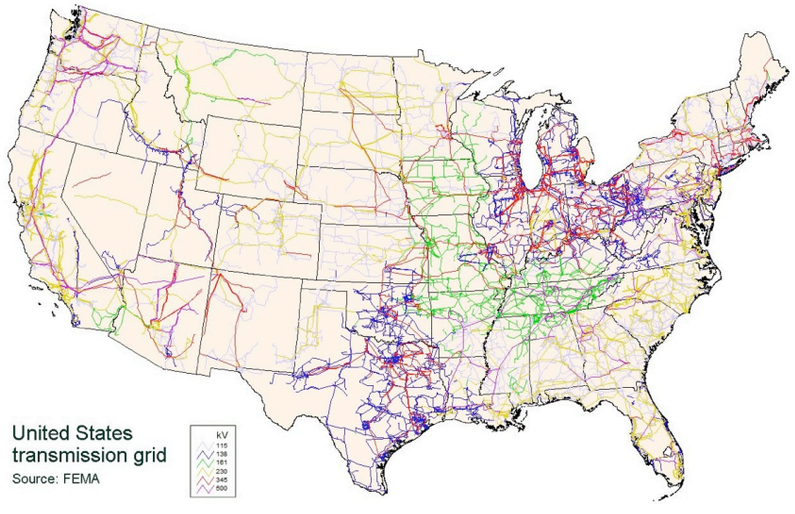


Temperature data



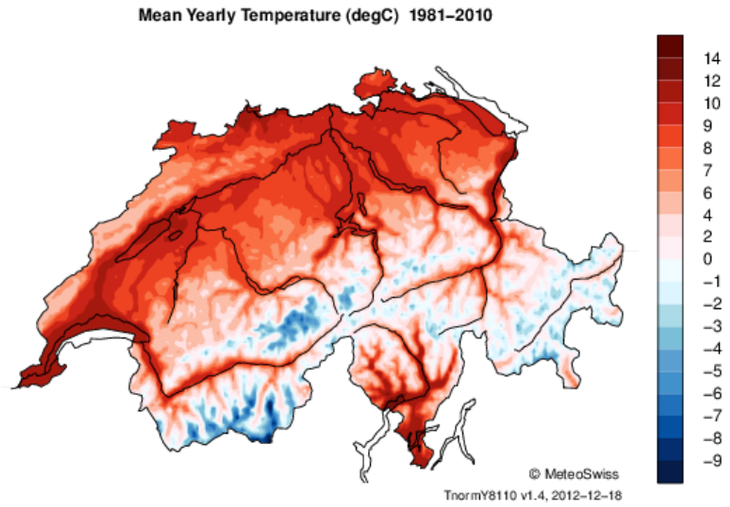
Social network data

Data are often structured

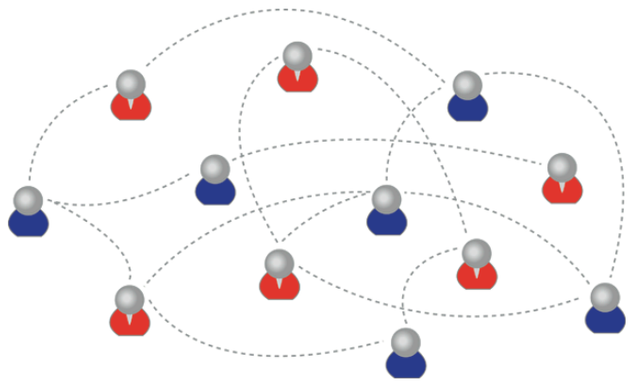


Electrical data

Traffic data



Temperature data



Social network data

We need to take into account the structure behind the data

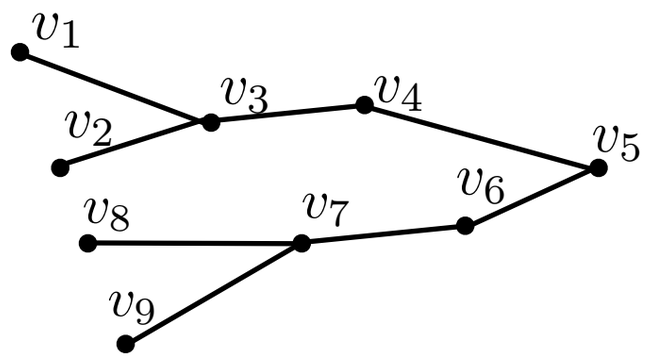
Graphs are appealing tools

- Efficient representations for **pairwise relations** between entities

The Königsberg Bridge Problem
[Leonhard Euler, 1736]

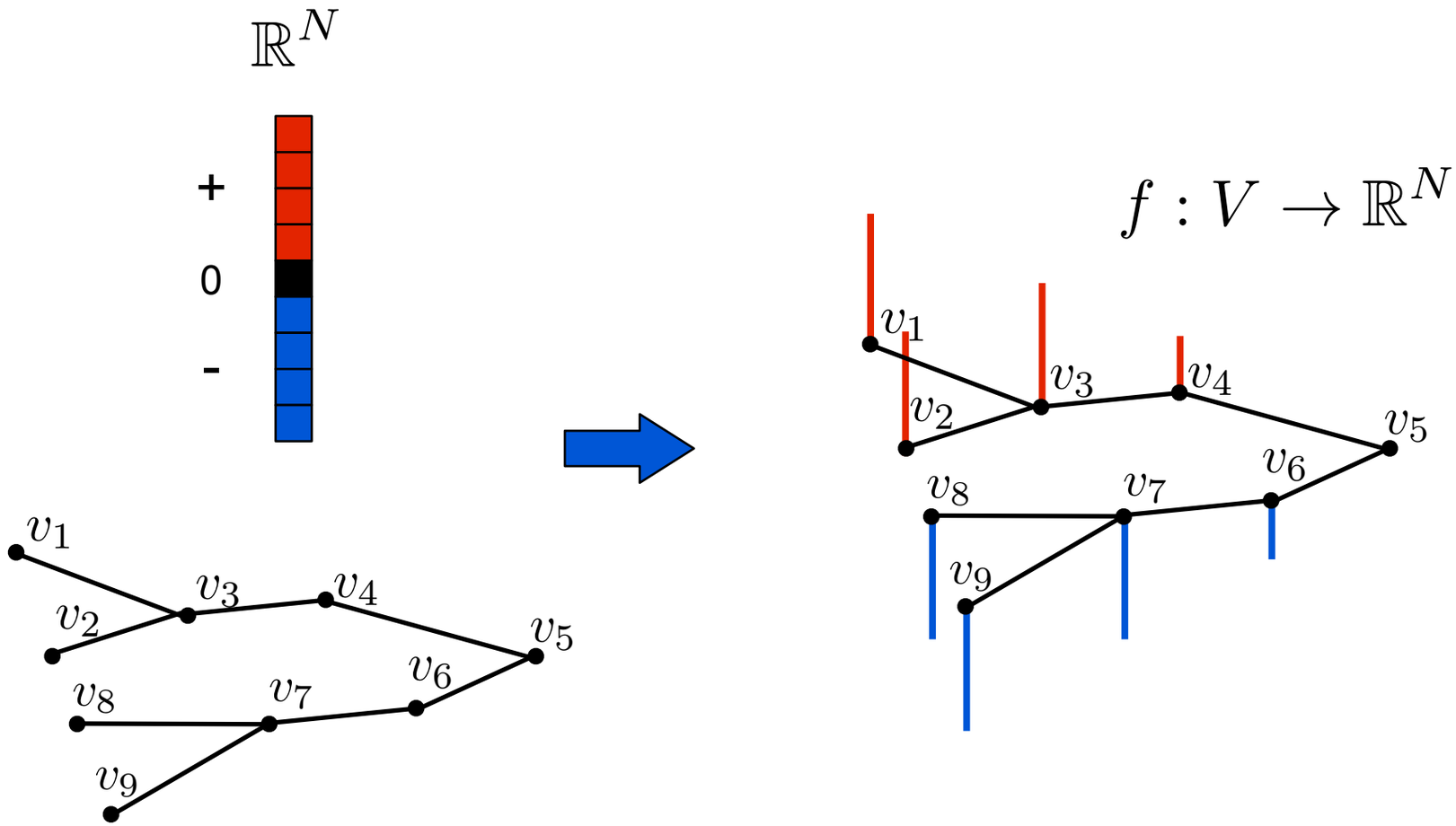
Graphs are appealing tools

- Efficient representations for pairwise relations between entities



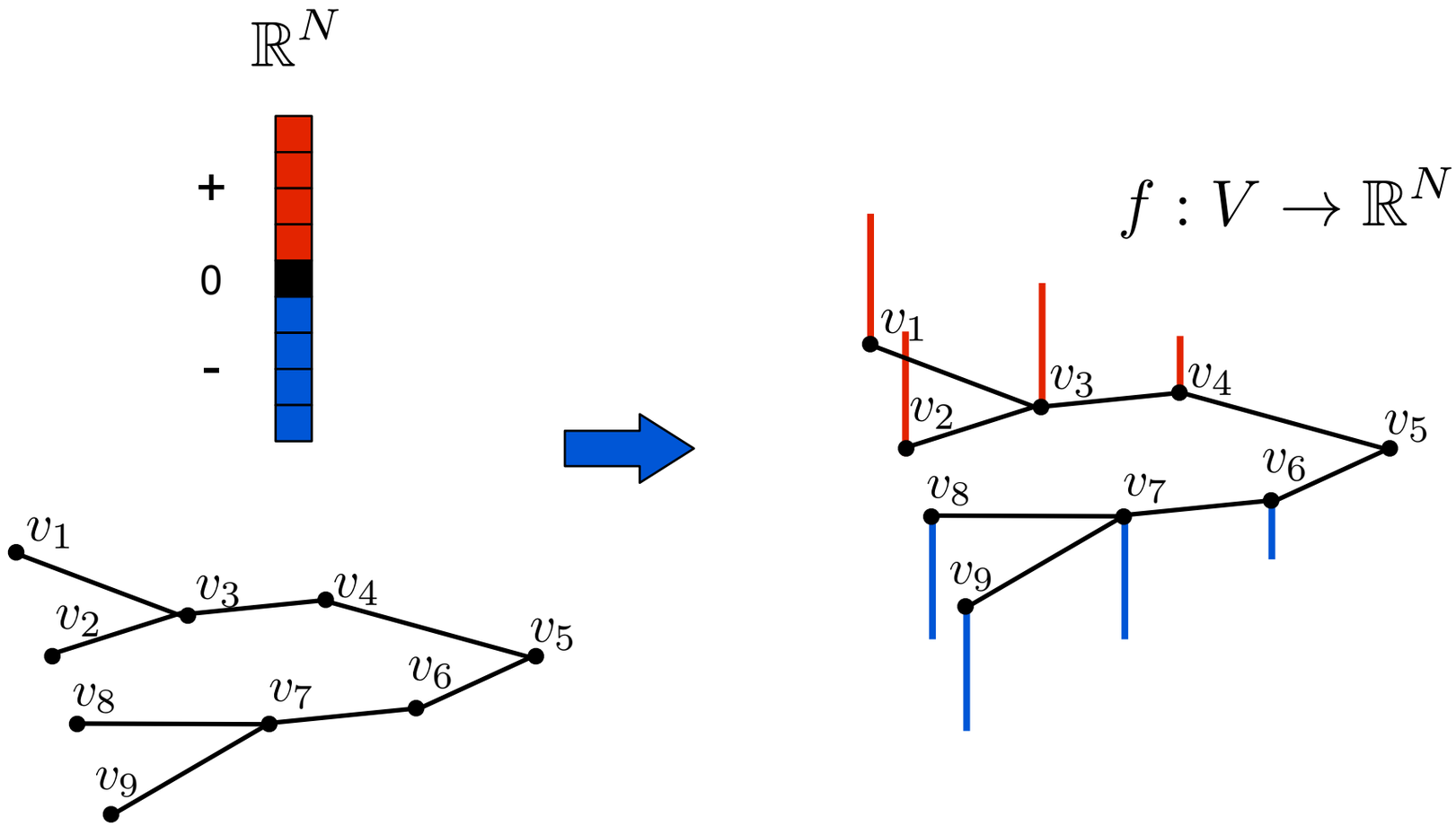
Graphs are appealing tools

- Efficient representations for pairwise relations between entities
- Structured data can be represented by **graph signals**



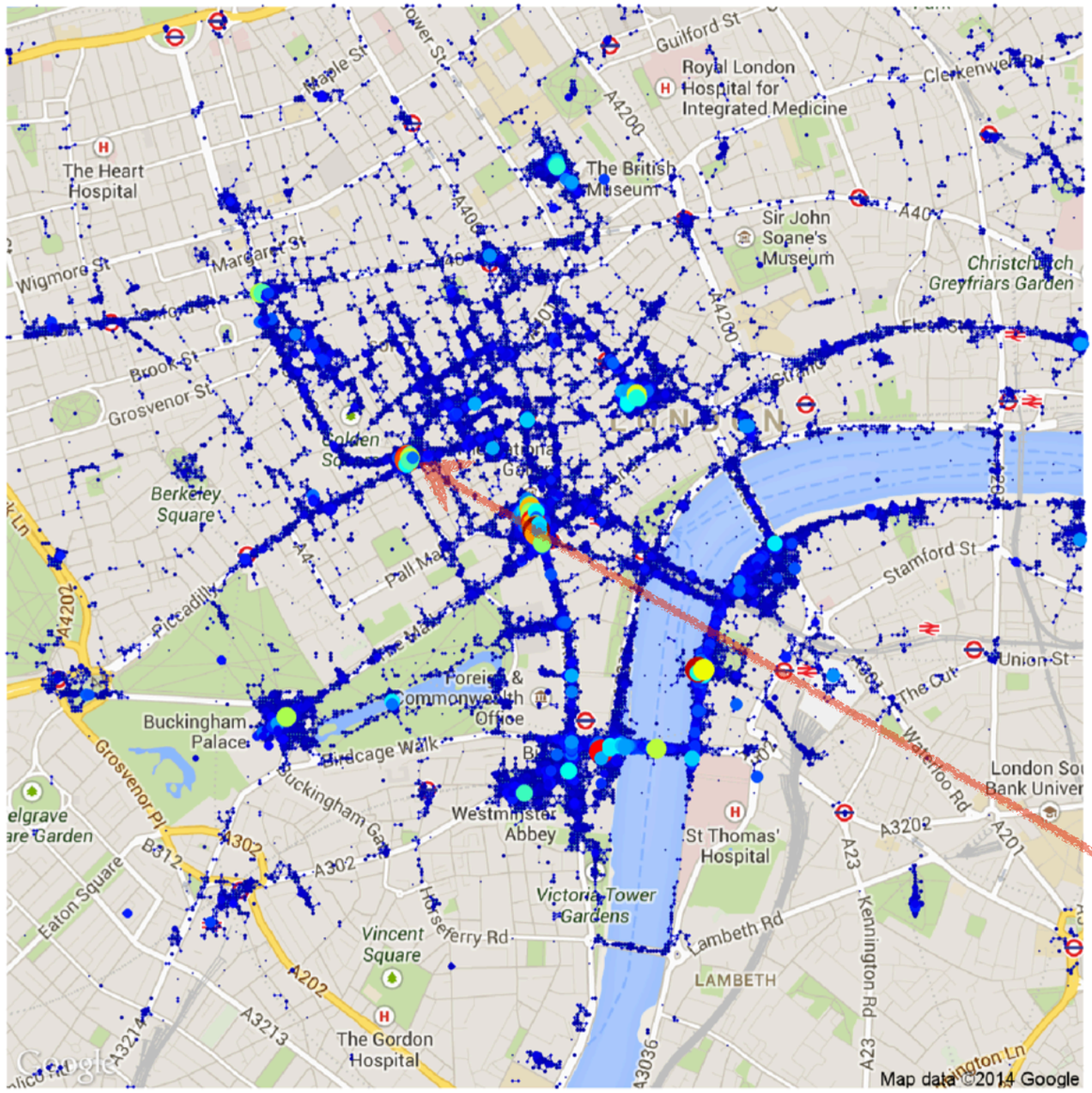
Graphs are appealing tools

- Efficient representations for pairwise relations between entities
- Structured data can be represented by **graph signals**

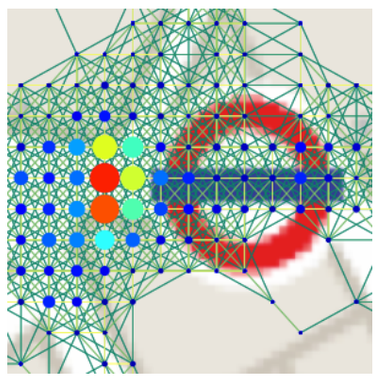


Takes into account both structure (edges) and data (values at vertices)

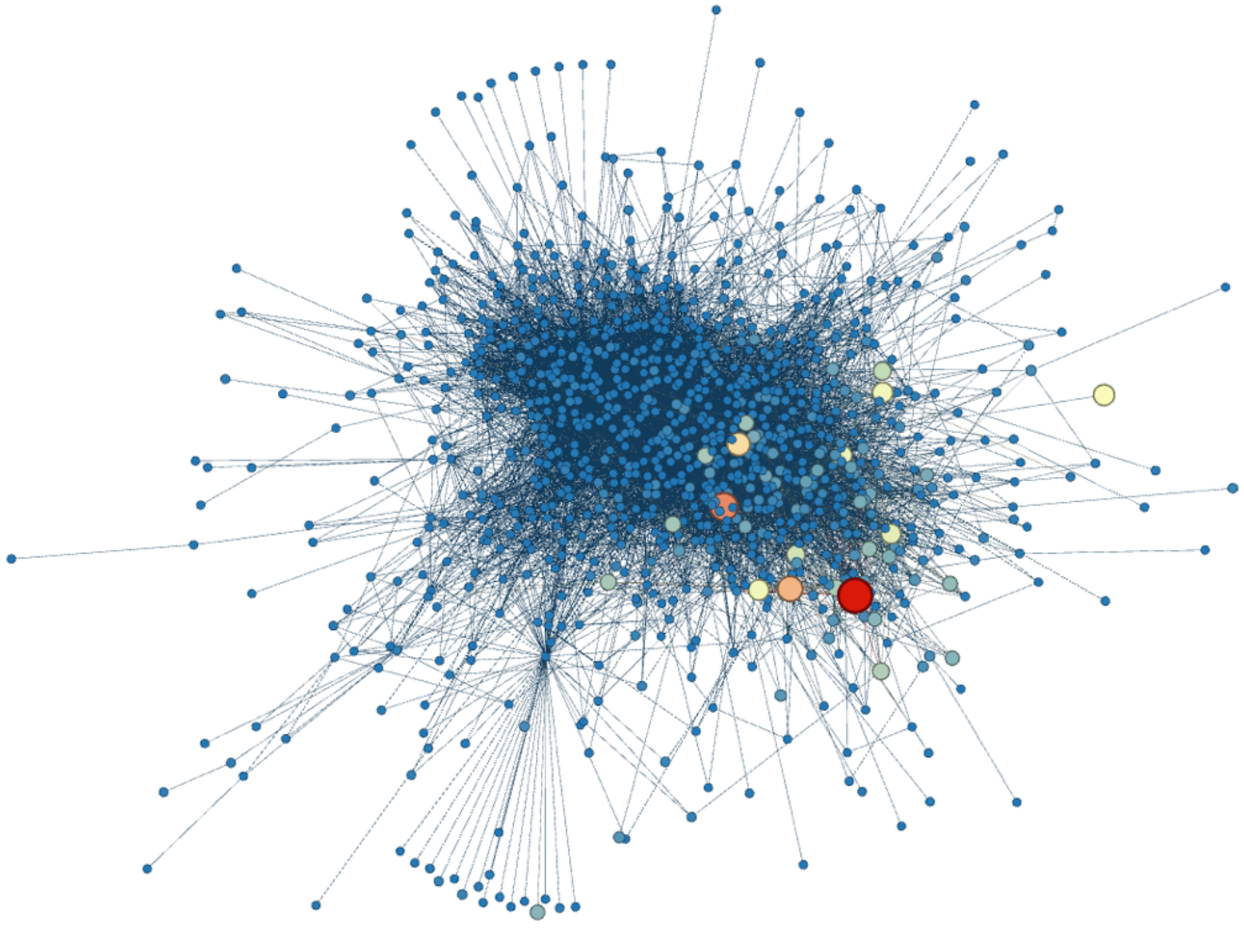
Graph signals are pervasive



- Vertices:
  - 9000 grid cells in London
- Edges:
  - Connecting cells that are geographically close
- Signal:
  - # Flickr users who have taken photos in two and a half year

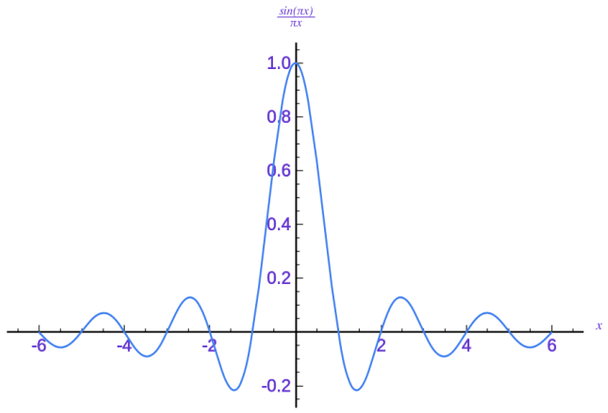


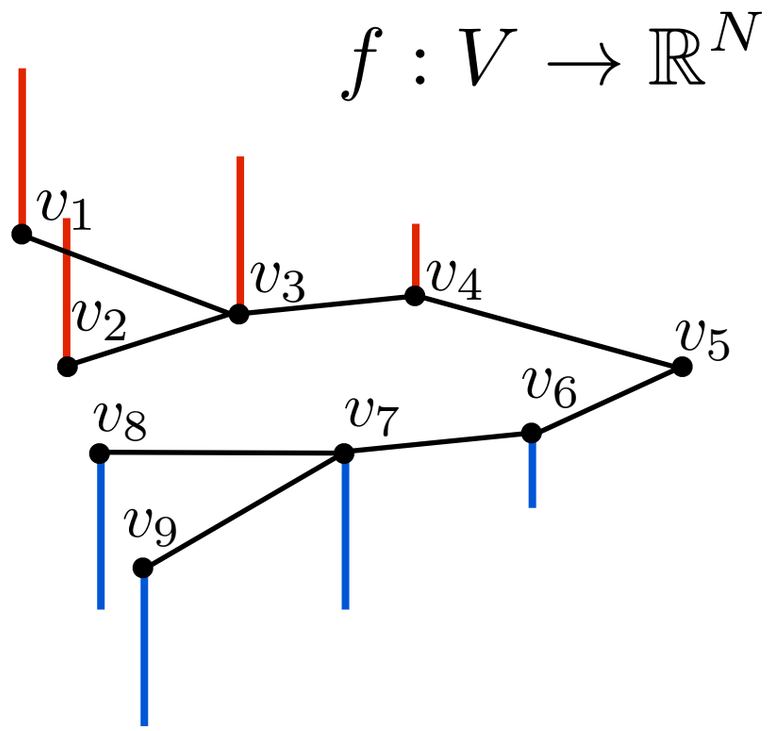
Graph signals are pervasive



- Vertices:
  - 1000 Twitter users
- Edges:
  - Connecting users that have following relationship
- Signal:
  - # Apple-related hashtags they have posted in six weeks

Research challenges

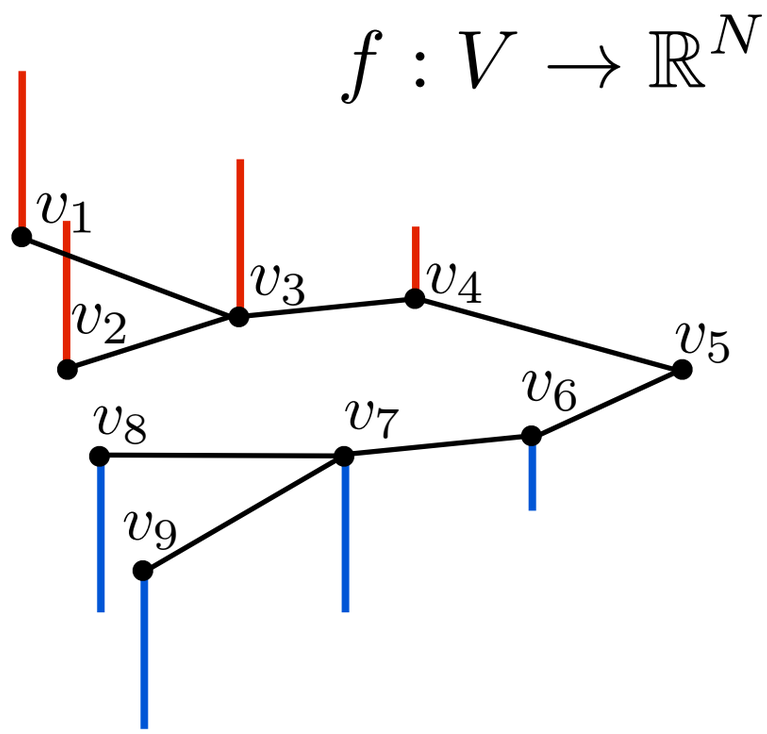




How to generalize classical signal processing tools on irregular domains such as graphs?

Graph signal processing

- Graph signals provide a nice compact format to encode structure within data
- Generalization of classical signal processing tools can greatly benefit analysis of such data
- Numerous applications: Transportation, biomedical, social network analysis, etc.
- An increasingly rich literature
  - classical signal processing
  - algebraic and spectral graph theory
  - computational harmonic analysis

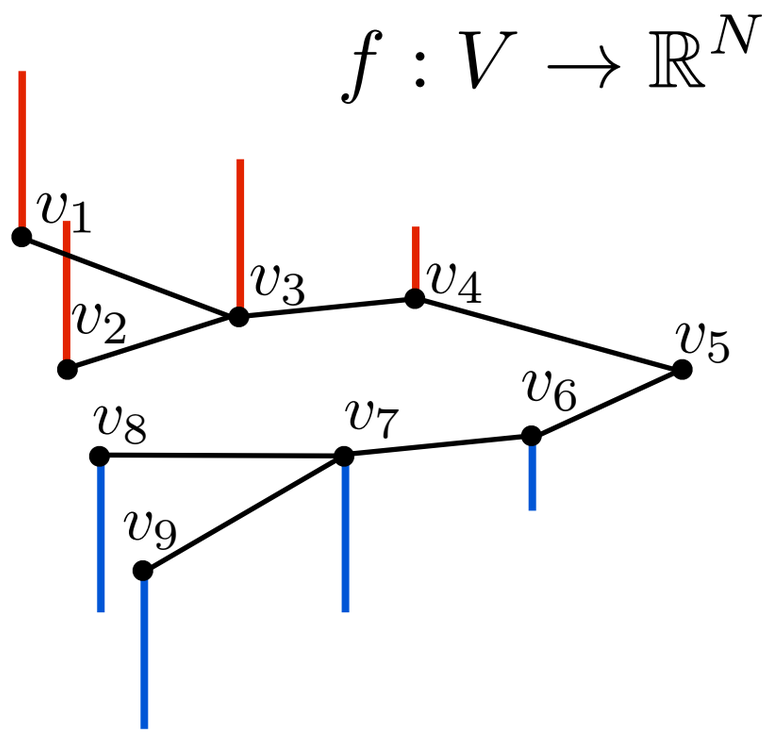


Outline

- Motivation
- Graph signal processing (GSP): Basic concepts
- Spectral filtering: Basic tools of GSP
- Applications and perspectives

Two paradigms

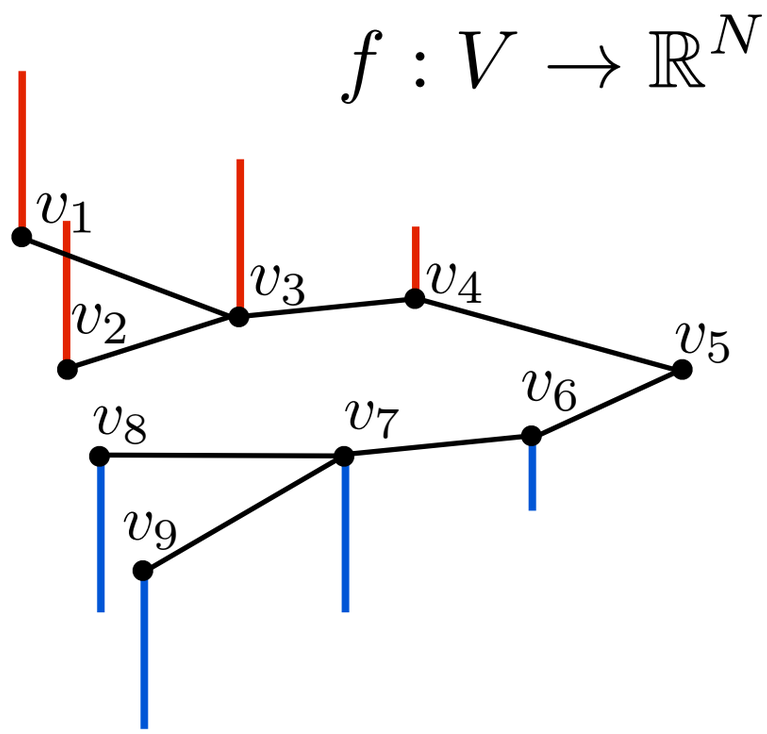
- The main approaches can be categorized into two families:
  - Vertex (spatial) domain designs
  - Frequency (graph spectral) domain designs



Two paradigms

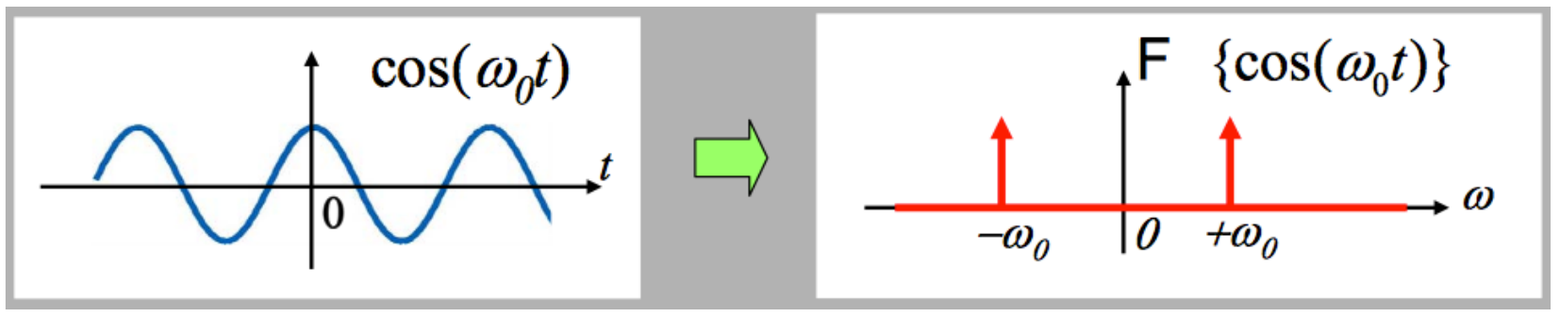
- The main approaches can be categorized into two families:
  - Vertex (spatial) domain designs
  - Frequency (graph spectral) domain designs

Important for analysis of signal properties

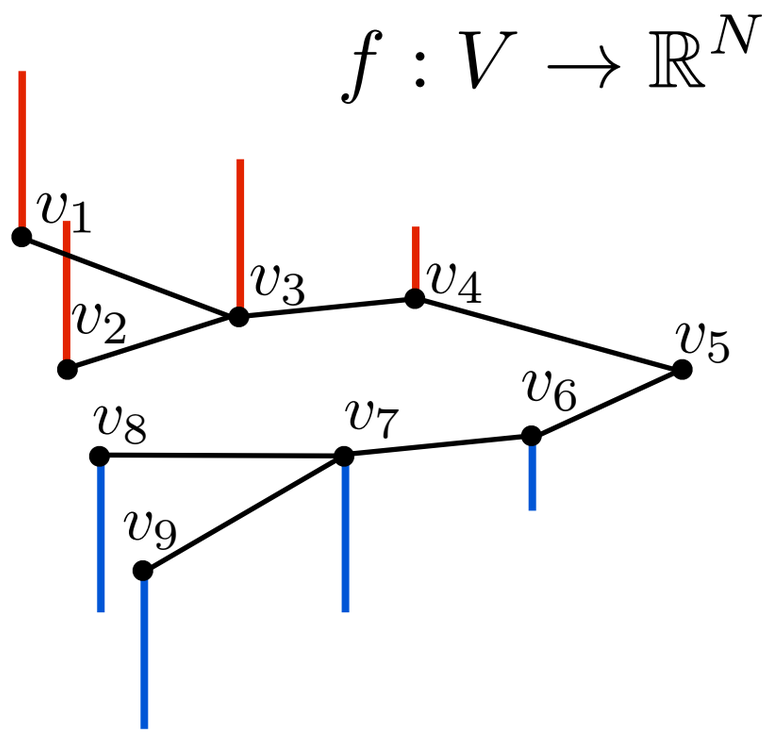


Need for frequency

- Classical Fourier transform provides the frequency domain representation of the signals

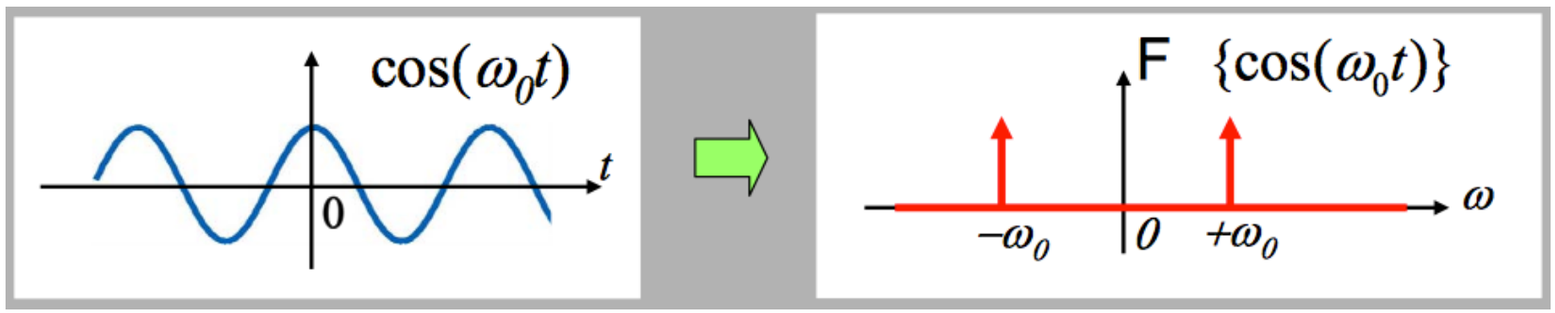


Source: http://www.physik.uni-kl.de



Need for frequency

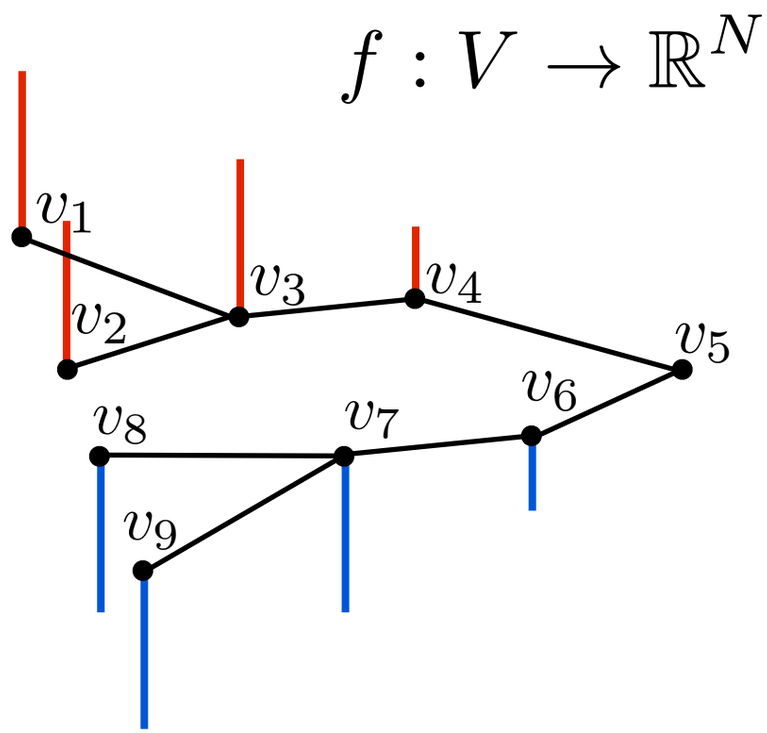
- Classical Fourier transform provides the frequency domain representation of the signals



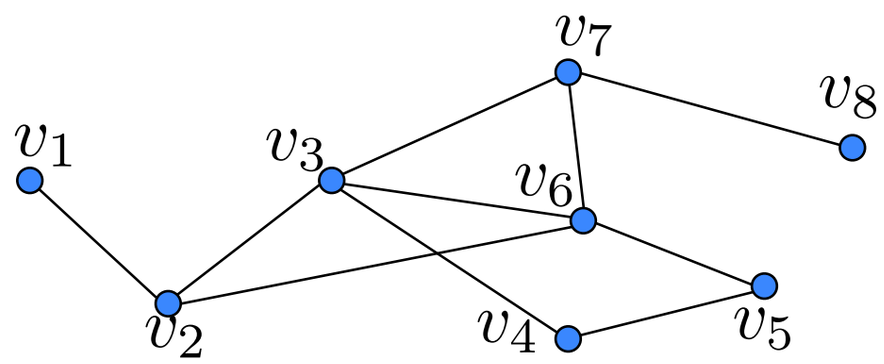
Source: http://www.physik.uni-kl.de

A notion of frequency for graph signals:

We need the graph Laplacian matrix



Graph Laplacian



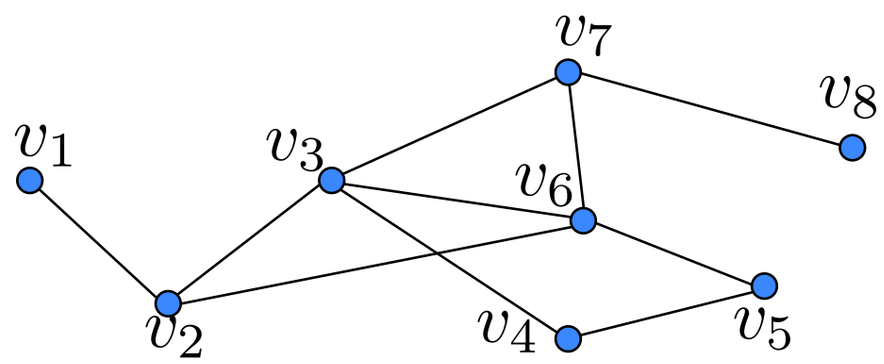
Weighted and undirected graph:

$G = \{V, E\}$

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

A

Graph Laplacian



Weighted and undirected graph:

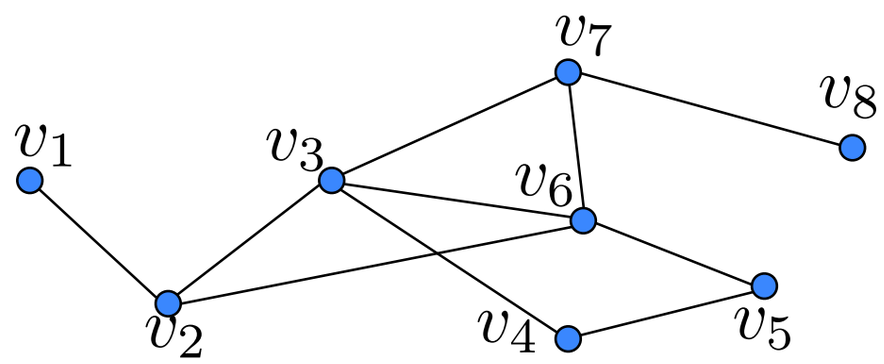
$G = \{V, E\}$

$D = \text{diag}(\text{degree}(v_1) \quad ... \quad \text{degree}(v_N))$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

D A

Graph Laplacian



Weighted and undirected graph:

$G = \{V, E\}$

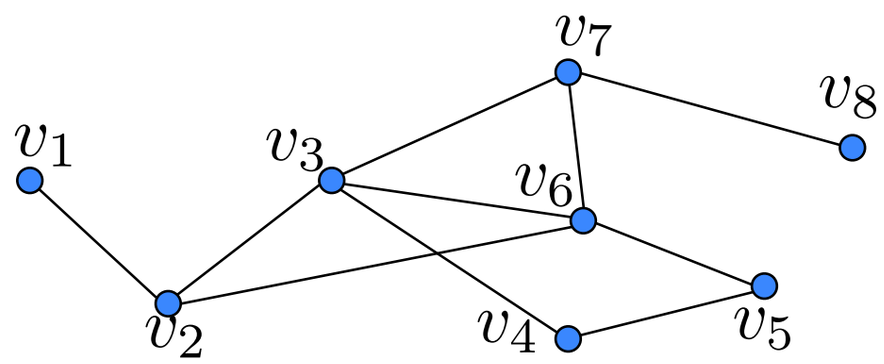
$D = \mathrm{diag}(\mathrm{degree}(v_1) \quad ... \quad \mathrm{degree}(v_N))$

$L = D - A$ **Equivalent to G!**

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

D A L

Graph Laplacian



Weighted and undirected graph:

$G = \{V, E\}$

$D = \mathrm{diag}(\mathrm{degree}(v_1) \quad ... \quad \mathrm{degree}(v_N))$

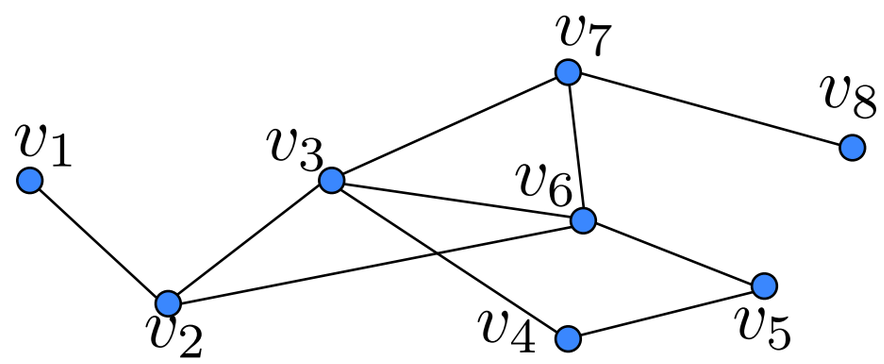
$L = D - A$ **Equivalent to G!**

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

D A L

- Symmetric
- Off-diagonal entries non-positive
- Rows sum up to zero

Graph Laplacian



Weighted and undirected graph:

$G = \{V, E\}$

$D = \mathrm{diag}(\mathrm{degree}(v_1) \quad ... \quad \mathrm{degree}(v_N))$

$L = D - A$ **Equivalent to G!**

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

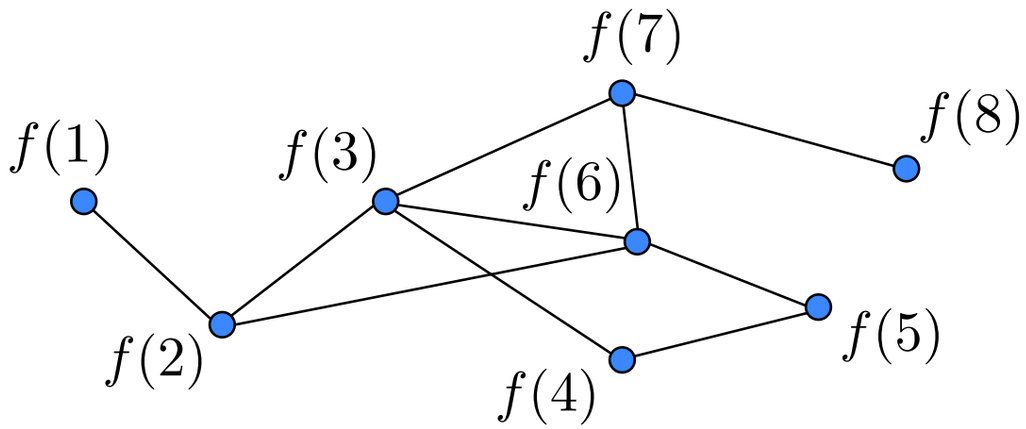
D A L

- Symmetric
- Off-diagonal entries non-positive
- Rows sum up to zero

Why graph Laplacian?

- standard stencil approximation of the Laplace operator
- leads to a Fourier-like transform

Graph Laplacian

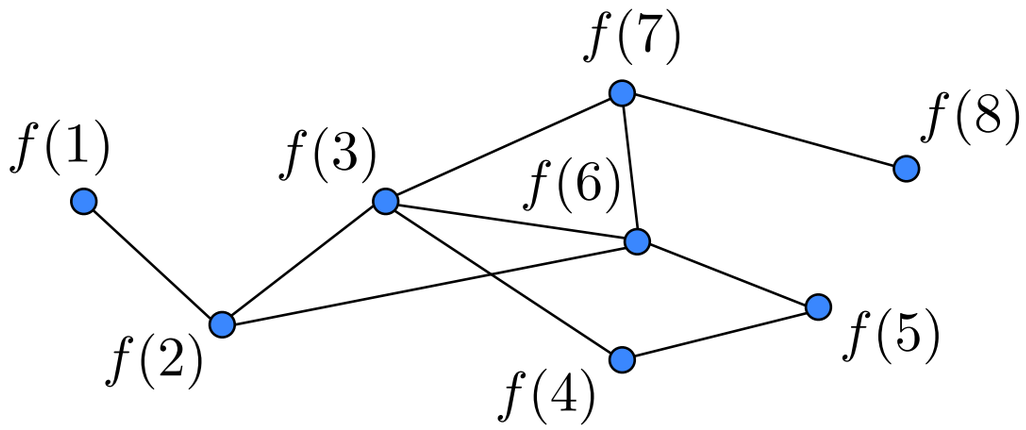


Graph signal $f : V \to \mathbb{R}^N$

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

L

Graph Laplacian



Graph signal $f : V \to \mathbb{R}^N$

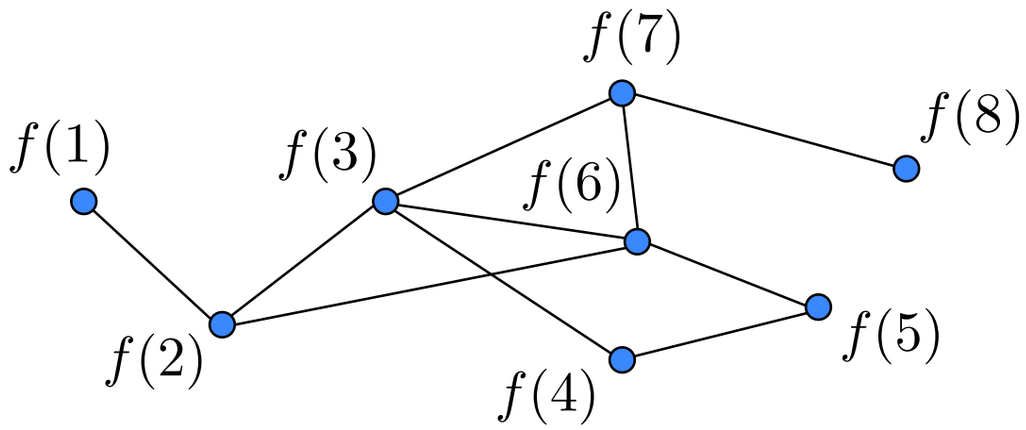
A difference operator:

$$Lf = \sum_{i,j=1}^{N} A_{ij} \left(f(i) - f(j) \right)$$

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{pmatrix}$$

L f

Graph Laplacian



$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{pmatrix}$$

L f

Graph signal $f : V \to \mathbb{R}^N$

A difference operator:

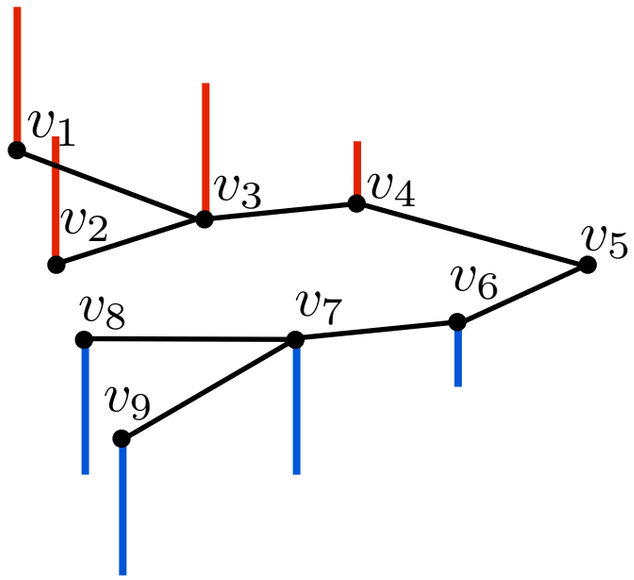
$$Lf = \sum_{i,j=1}^{N} A_{ij} \left(f(i) - f(j) \right)$$

Laplacian quadratic form:

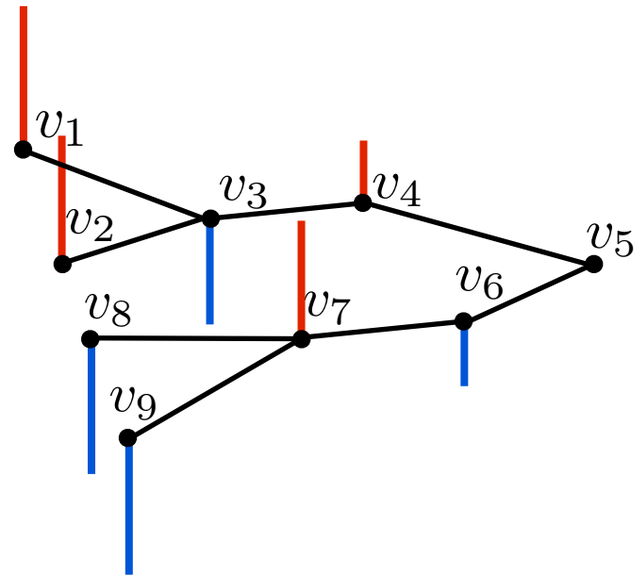
$$f^T L f = \frac{1}{2} \sum_{i,j=1}^{N} A_{ij} \left(f(i) - f(j) \right)^2$$

A measure of "smoothness" [Zhou04]

Graph Laplacian



$$f^T L f = 1$$



$$f^T L f = 21$$

Graph Laplacian

- L has a complete set of orthonormal eigenvectors: $L = \chi \Lambda \chi^T$

$$L = \begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N\text{-}1} \\ | & & | \end{bmatrix} \begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix} \begin{bmatrix} — & \chi_0 & — \\ & \cdots & \\ — & \chi_{N\text{-}1} & — \end{bmatrix}$$

$$\chi \qquad\qquad \Lambda \qquad\qquad \chi^T$$

Graph Laplacian

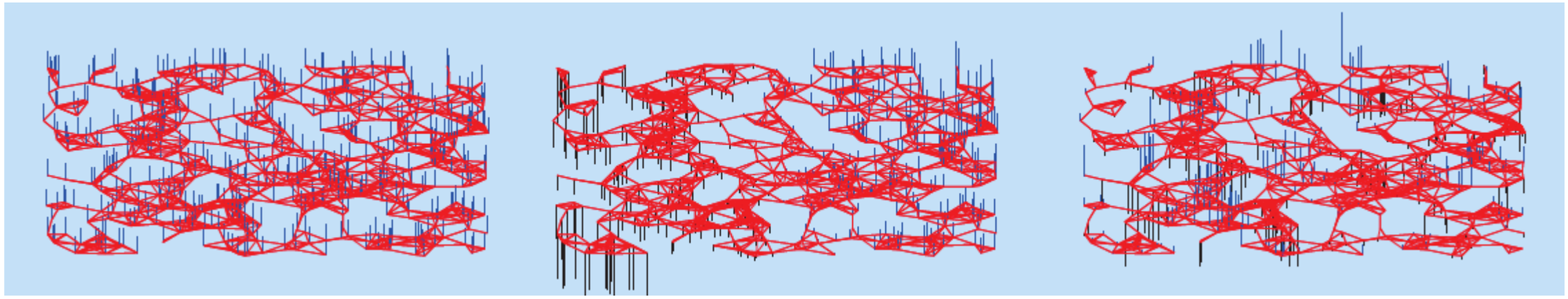
- L has a complete set of orthonormal eigenvectors: $L = \chi \Lambda \chi^T$

$$L = \begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N\text{-}1} \\ | & & | \end{bmatrix} \begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix} \begin{bmatrix} — & \chi_0 & — \\ & \cdots & \\ — & \chi_{N\text{-}1} & — \end{bmatrix}$$

$$\chi \qquad\qquad \Lambda \qquad\qquad \chi^T$$

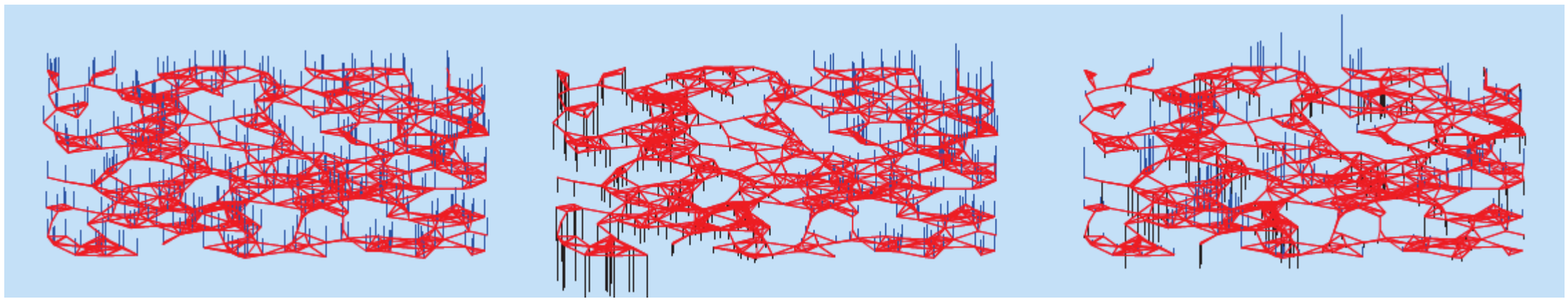
- Eigenvalues are usually sorted increasingly: $0 = \lambda_0 < \lambda_1 \leq \ldots \leq \lambda_{N-1}$

Graph Fourier transform



χ_0 χ_1 χ_{50} [Shuman13]

Graph Fourier transform



χ_0 χ_1 χ_{50} [Shuman13]

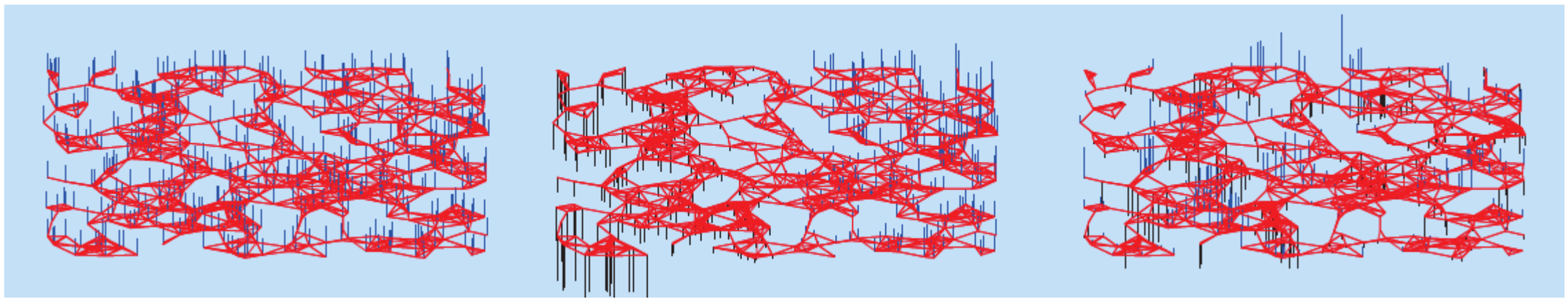
Low frequency → High frequency

$L = \chi \Lambda \chi^T$

$\chi_0^T L \chi_0 = \lambda_0 = 0$ $\chi_{50}^T L \chi_{50} = \lambda_{50}$

- Eigenvectors associated with smaller eigenvalues have values that vary less rapidly along the edges

Graph Fourier transform



χ_0 χ_1 χ_{50} [Shuman13]

Low frequency → High frequency

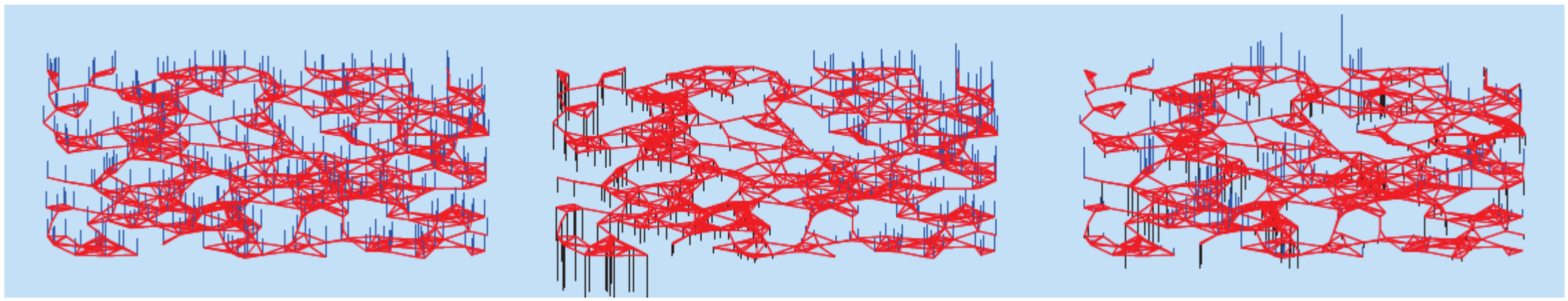
$L = \chi \Lambda \chi^T$

$\chi_0^T L \chi_0 = \lambda_0 = 0$ $\quad$ $\chi_{50}^T L \chi_{50} = \lambda_{50}$

Graph Fourier transform:
[Hammond11]

$$\hat{f}(\ell) = \langle \chi_\ell, f \rangle : \quad \begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N\text{-}1} \\ | & & | \end{bmatrix}^T \begin{bmatrix} | \\ f \\ | \end{bmatrix}$$

Graph Fourier transform



χ_0 χ_1 χ_{50} [Shuman13]

Low frequency → High frequency

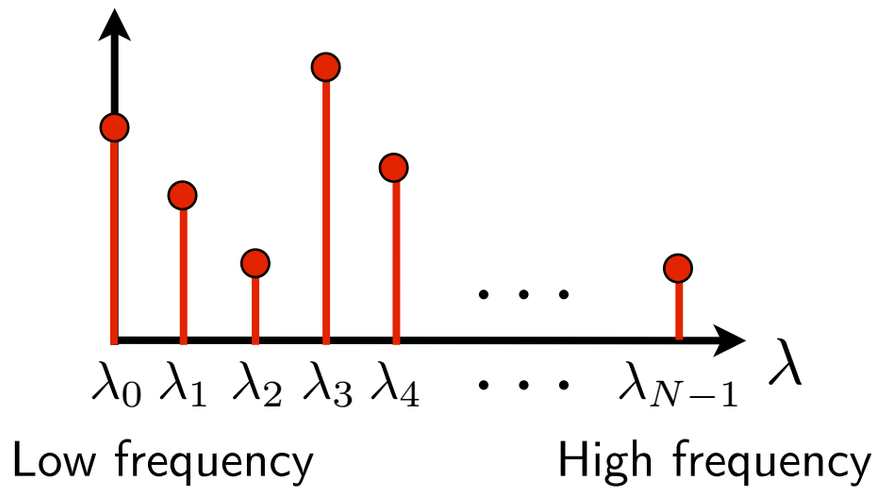
$L = \chi \Lambda \chi^T$

$\chi_0^T L \chi_0 = \lambda_0 = 0$

$\chi_{50}^T L \chi_{50} = \lambda_{50}$

Graph Fourier transform:
[Hammond11]

$$\hat{f}(\ell) = \langle \chi_\ell, f \rangle : \begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N-1} \\ | & & | \end{bmatrix}^T f$$



Graph Fourier transform

- The Laplacian L admits the following eigendecomposition: $L\chi_\ell = \lambda_\ell \chi_\ell$

Graph Fourier transform

- The Laplacian L admits the following eigendecomposition: $L\chi_\ell = \lambda_\ell \chi_\ell$

one-dimensional Laplace operator: $\dfrac{d^2}{dx^2}$

eigenfunctions: $e^{j\omega x}$

Classical FT: $\hat{f}(\omega) = \int (e^{j\omega x})^* f(x) dx$

$$f(x) = \frac{1}{2\pi} \int \hat{f}(\omega) e^{j\omega x} d\omega$$

Graph Fourier transform

- The Laplacian L admits the following eigendecomposition: $L\chi_\ell = \lambda_\ell \chi_\ell$

one-dimensional Laplace operator: $\frac{d^2}{dx^2}$

eigenfunctions: $e^{j\omega x}$

Classical FT:

$$\hat{f}(\omega) = \int (e^{j\omega x})^* f(x) dx$$

$$f(x) = \frac{1}{2\pi} \int \hat{f}(\omega) e^{j\omega x} d\omega$$

graph Laplacian: L

eigenvectors: χ_ℓ

$f : V \to \mathbb{R}^N$

Graph FT:

$$\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i)$$

$$f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$$

Graph Fourier transform

- The Laplacian L admits the following eigendecomposition: $L\chi_\ell = \lambda_\ell \chi_\ell$

one-dimensional Laplace operator: $\dfrac{d^2}{dx^2}$

eigenfunctions: $e^{j\omega x}$

Classical FT:

$$\hat{f}(\omega) = \int (e^{j\omega x})^* f(x) dx$$

$$f(x) = \frac{1}{2\pi} \int \hat{f}(\omega) e^{j\omega x} d\omega$$

graph Laplacian: L

eigenvectors: χ_ℓ

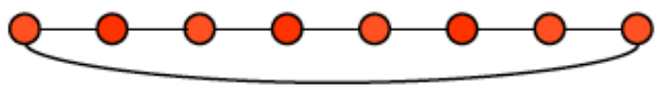
$f : V \to \mathbb{R}^N$

Graph FT:

$$\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i)$$

$$f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$$

Two special cases



- (Unordered) Laplacian eigenvalues: $\lambda_\ell = 2 - 2\cos\left(\frac{2\ell\pi}{N}\right)$
- One possible choice of orthogonal Laplacian eigenvectors:

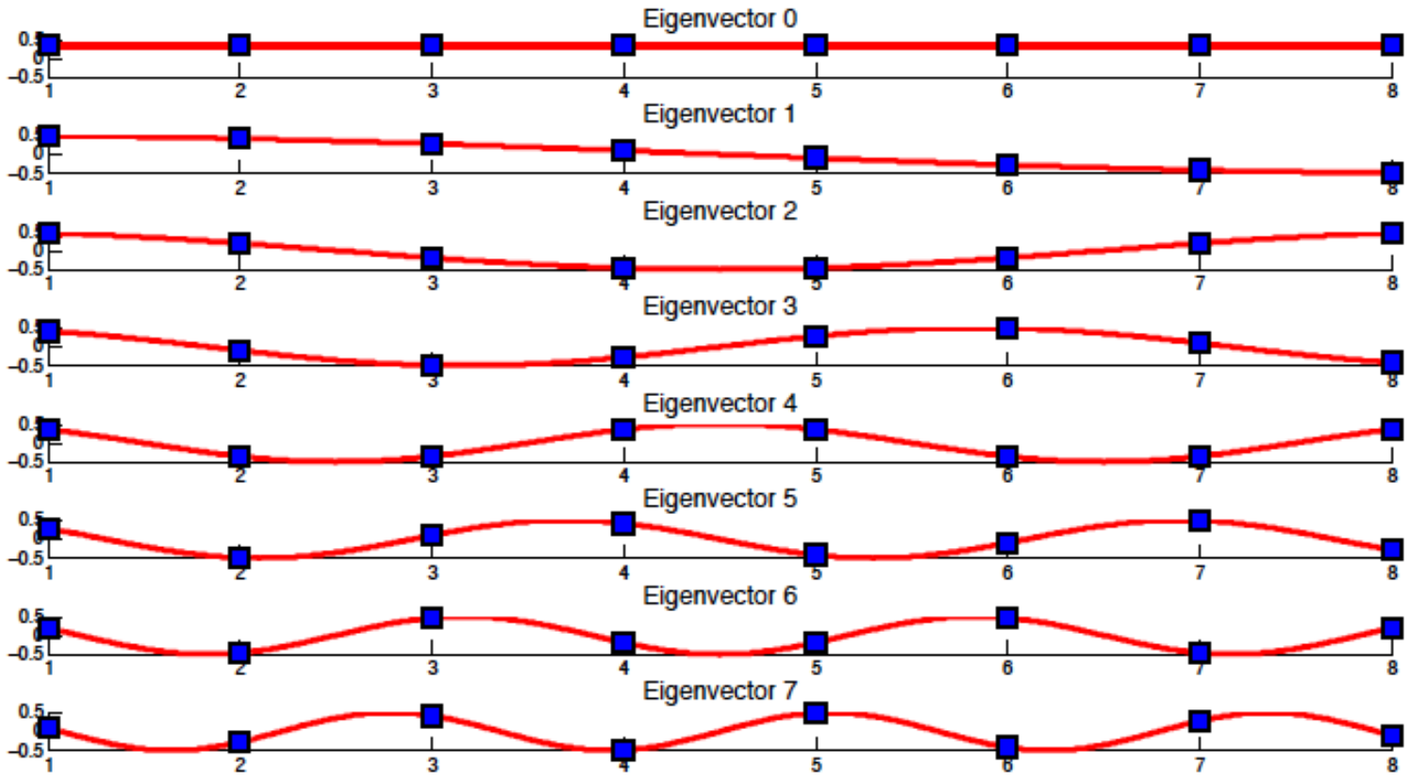
$$\chi_\ell = \left[1, \omega^\ell, \omega^{2\ell}, \ldots, \omega^{(N-1)\ell}\right], \text{ where } \omega = e^{\frac{2\pi j}{N}}$$

- $\begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N-1} \\ | & & | \end{bmatrix}$ is the Discrete Fourier Transform (DFT) matrix

[Vandergheynst11]

Two special cases

- $\lambda_\ell = 2 - 2\cos\left(\frac{\pi\ell}{N}\right)$
- $\chi_0(i) = \frac{1}{\sqrt{N}}$, $\chi_\ell(i) = \sqrt{\frac{2}{N}}\cos\left(\frac{\pi\ell(i-0.5)}{N}\right)$, $\ell = 1, 2, \ldots, N-1$



$$\begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N-1} \\ | & & | \end{bmatrix}$$ is the Discrete Cosine Transform matrix (DCT-II, Strang, 1999), which is used in JPEG image compression

[Vandergheynst11]

Outline

- Motivation
- Graph signal processing (GSP): Basic concepts
- Spectral filtering: Basic tools of GSP
- Applications and perspectives

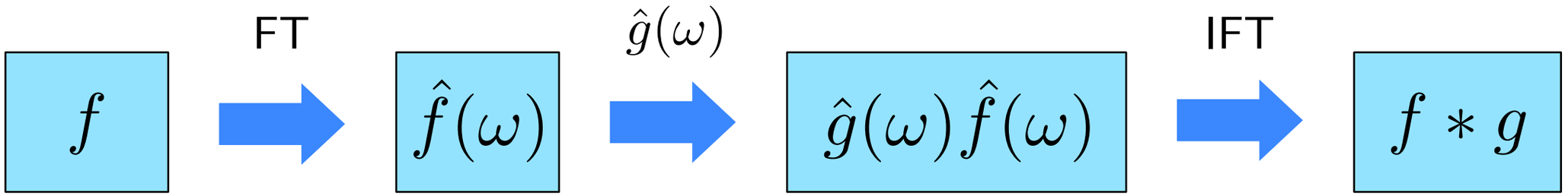
Classical frequency filtering

Classical FT: $\hat{f}(\omega) = \int (e^{j\omega x})^* f(x) dx \qquad f(x) = \frac{1}{2\pi} \int \hat{f}(\omega) e^{j\omega x} d\omega$

Classical frequency filtering

Classical FT: $\hat{f}(\omega) = \int (e^{j\omega x})^* f(x) dx \qquad f(x) = \frac{1}{2\pi} \int \hat{f}(\omega) e^{j\omega x} d\omega$

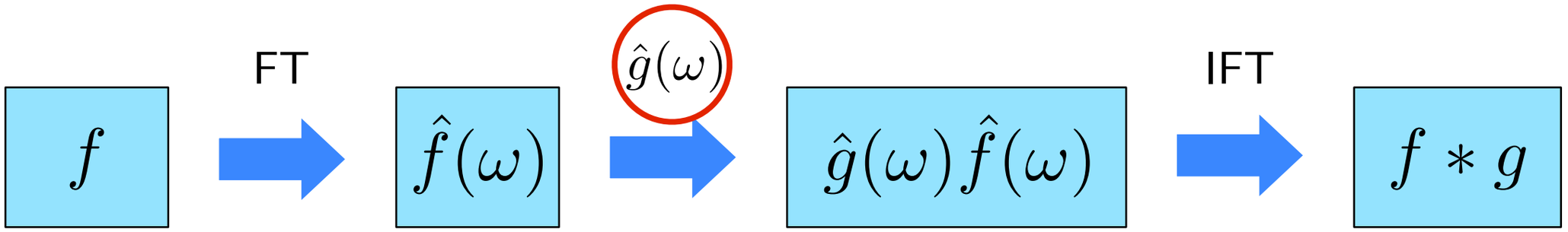
Apply filter with transfer function $\hat{g}(\cdot)$ to a signal f

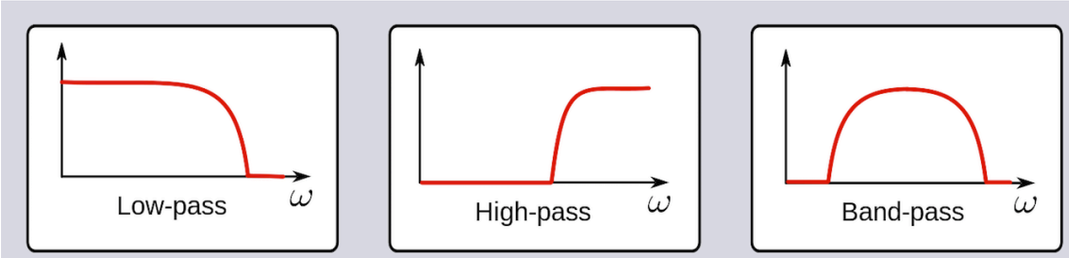


Classical frequency filtering

Classical FT: $\hat{f}(\omega) = \int (e^{j\omega x})^* f(x) dx \qquad f(x) = \frac{1}{2\pi} \int \hat{f}(\omega) e^{j\omega x} d\omega$

Apply filter with transfer function $\hat{g}(\cdot)$ to a signal f





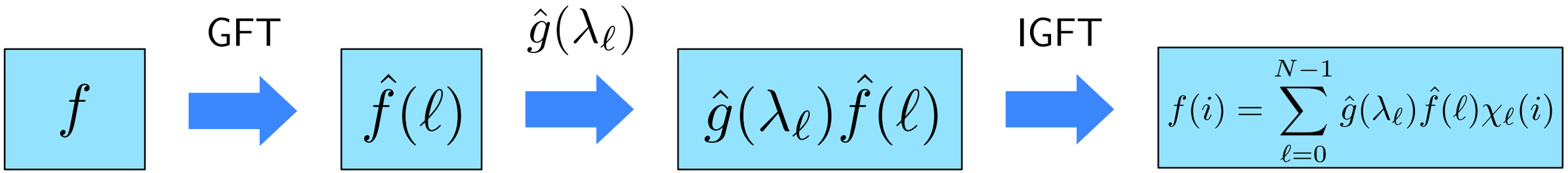
Graph spectral filtering

GFT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i)$ $\quad$ $f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$

Graph spectral filtering

GFT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i)$ $\quad$ $f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$

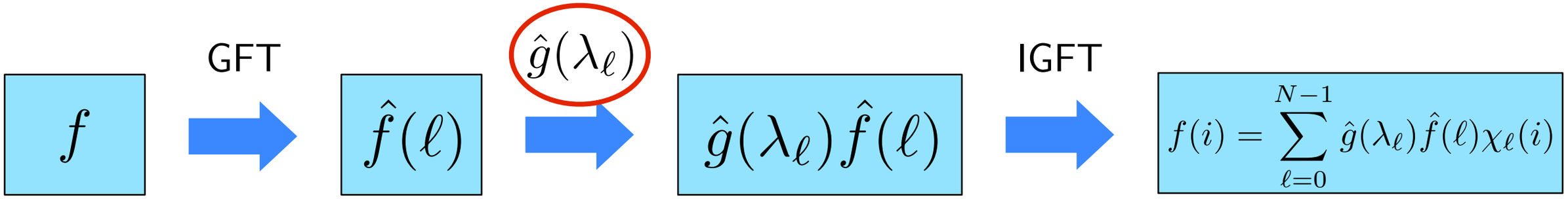
Apply filter with transfer function $\hat{g}(\cdot)$ to a graph signal $f : V \to \mathbb{R}^n$

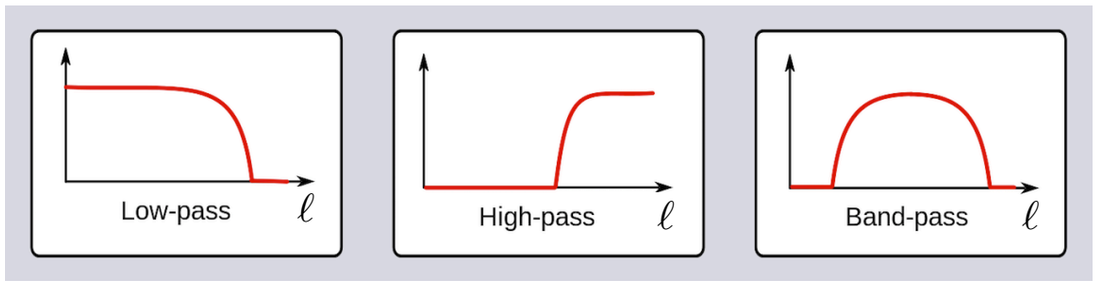


Graph spectral filtering

GFT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i)$ $\quad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$

Apply filter with transfer function $\hat{g}(\cdot)$ to a graph signal $f : V \to \mathbb{R}^n$

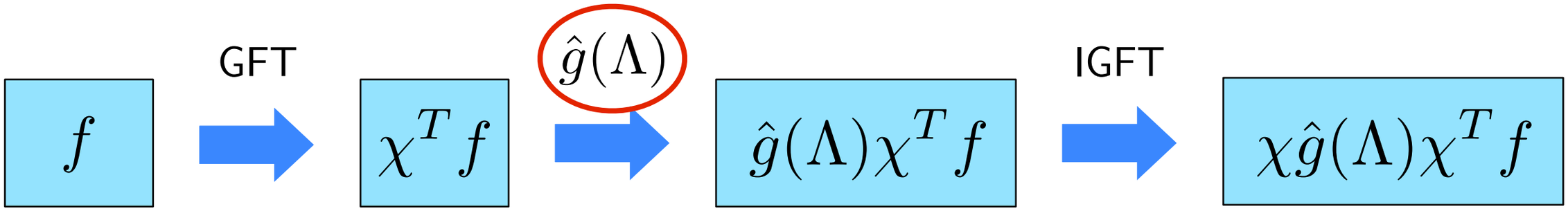




Graph spectral filtering

GFT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i)$ $\quad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$

Apply filter with transfer function $\hat{g}(\cdot)$ to a graph signal $f : V \to \mathbb{R}^n$



$$\hat{g}(\Lambda) = \begin{bmatrix} \hat{g}(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_{N-1}) \end{bmatrix}$$

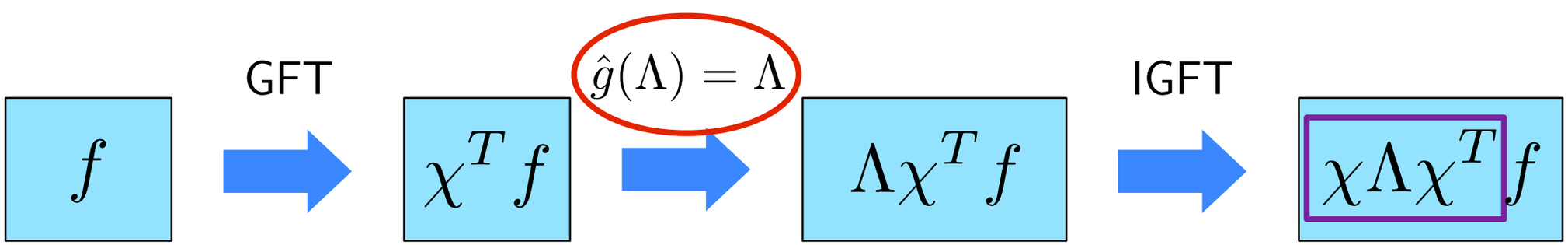
Graph Laplacian revisited

$$\text{GFT:} \quad \hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i) \qquad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$$

Graph Laplacian revisited

GFT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i)$ $\quad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$

The Laplacian L is a difference operator: $Lf = \chi \Lambda \chi^T f$



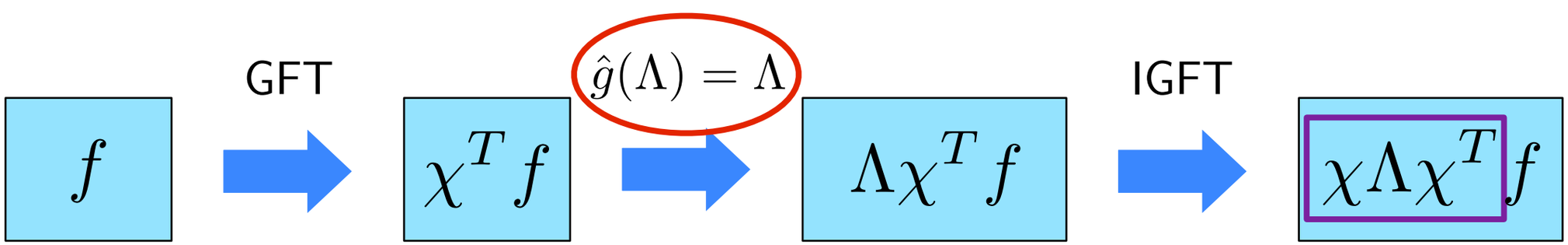
$$\begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix}$$

The Laplacian operator filters the signal in the spectral domain by its eigenvalues!

Graph Laplacian revisited

GFT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i)$ $\quad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$

The Laplacian L is a difference operator: $Lf = \chi \Lambda \chi^T f$



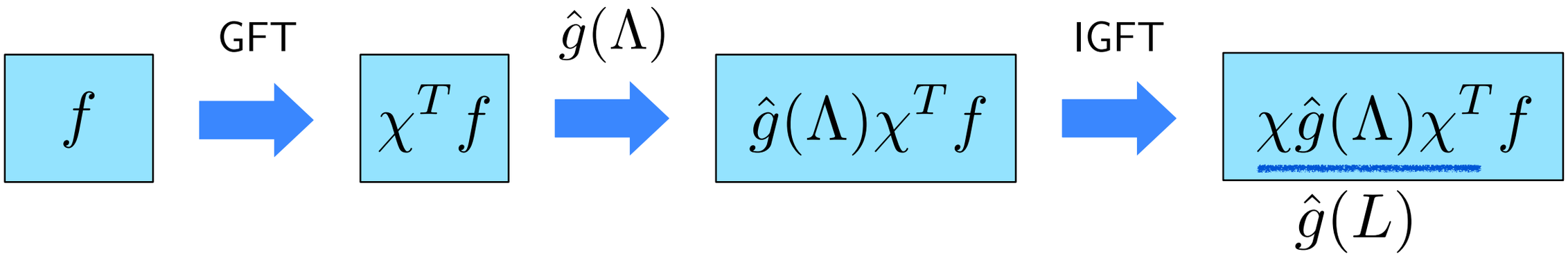
$$\begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix}$$

The Laplacian operator filters the signal in the spectral domain by its eigenvalues!

The Laplacian quadratic form: $f^T L f = ||L^{\frac{1}{2}} f||_2 = ||\chi \Lambda^{\frac{1}{2}} \chi^T f||_2$

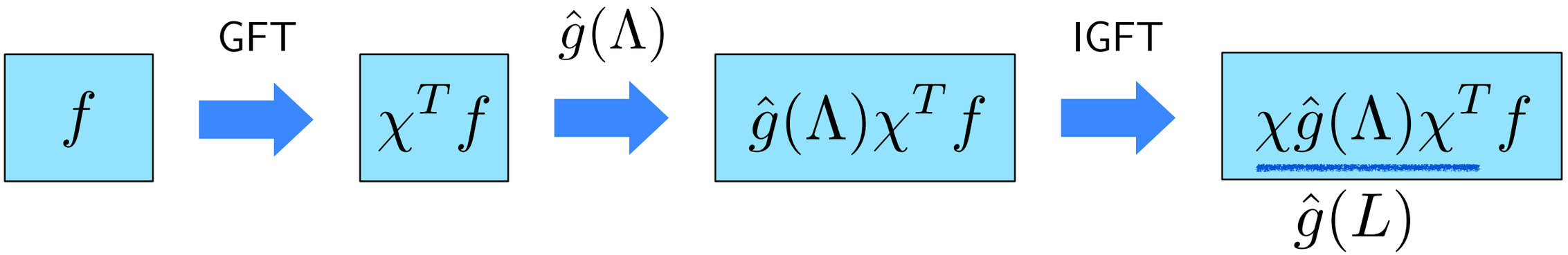
Graph transform/dictionary design

- Transforms and dictionaries can be designed through graph spectral filtering: Functions of graph Laplacian!



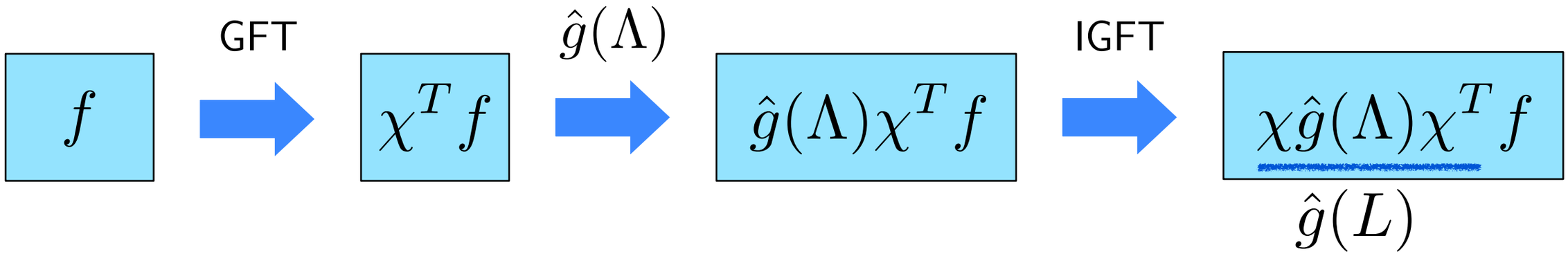
Graph transform/dictionary design

- Transforms and dictionaries can be designed through graph spectral filtering: Functions of graph Laplacian!

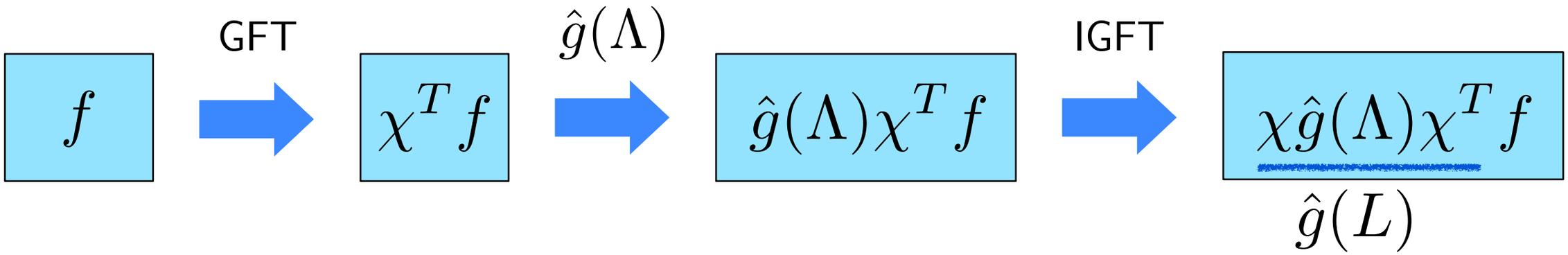


- Important properties can be achieved by properly defining $\hat{g}(L)$, such as localization of atoms
- Closely related to kernels and regularization on graphs [Smola03]

A simple example



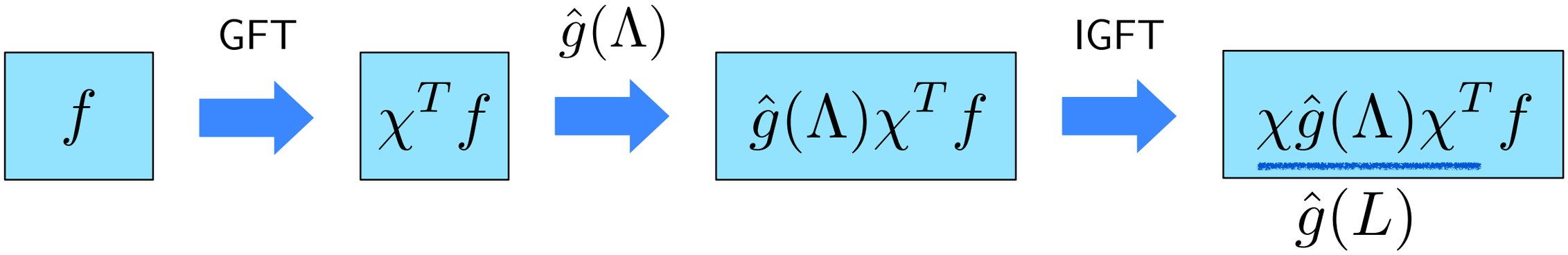
A simple example



Problem: We observe a noisy graph signal $f = y_0 + \eta$ and wish to recover y_0

$$y^* = \arg\min_y \{||y - f||_2^2 + \gamma y^T L y\}$$

A simple example



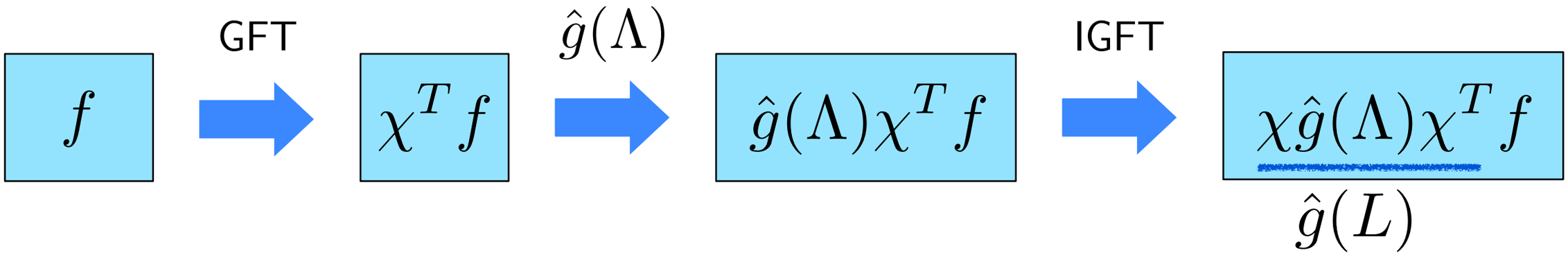
Problem: We observe a noisy graph signal $f = y_0 + \eta$ and wish to recover y_0

$$y^* = \arg\min_y \{||y - f||_2^2 + \gamma y^T L y\}$$

Data fitting term

"Smoothness" assumption

A simple example



Problem: We observe a noisy graph signal $f = y_0 + \eta$ and wish to recover y_0

$$y^* = \arg\min_y \{||y - f||_2^2 + \gamma y^T L y\}$$

Data fitting term

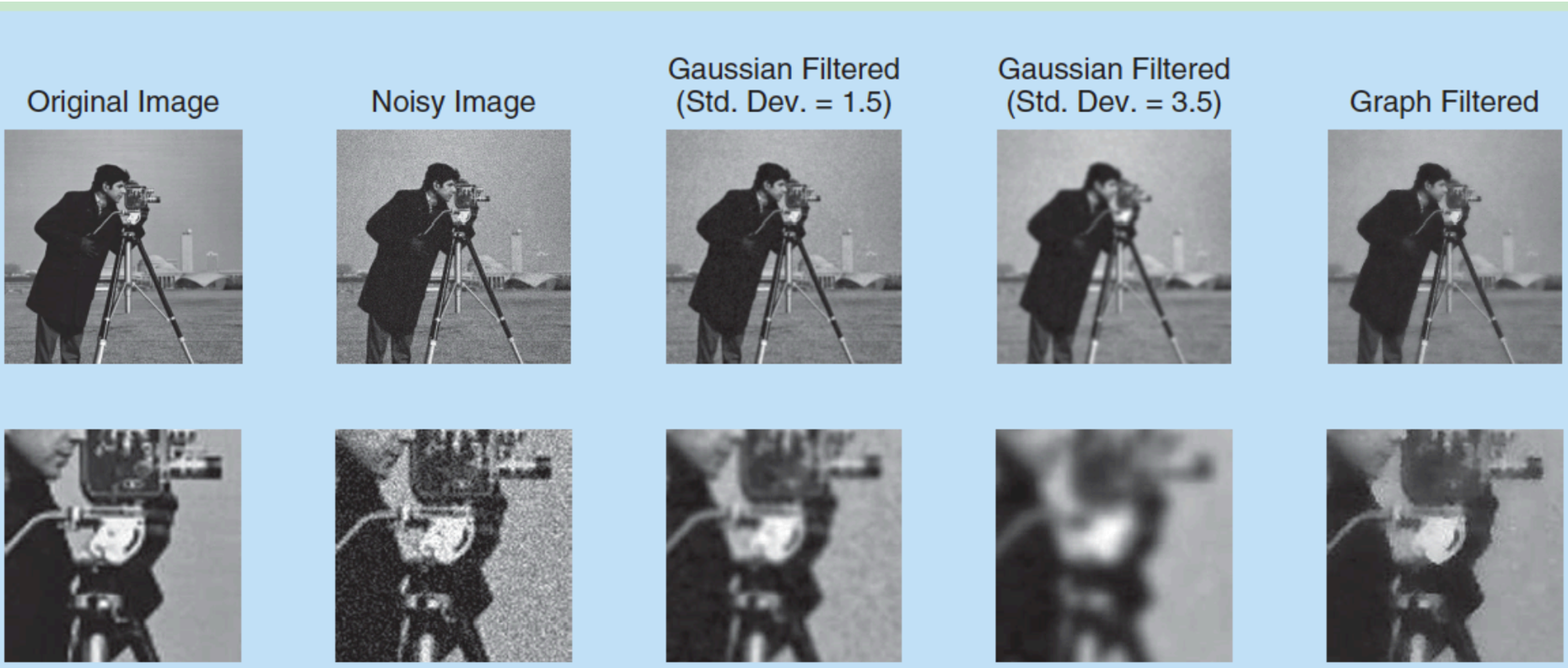
"Smoothness" assumption

$$y^* = \underbrace{(I + \gamma L)^{-1} f}_{\hat{g}(L)}$$

Laplacian (Tikhonov) regularization is equivalent to low-pass filtering in the graph spectral domain!

Example designs

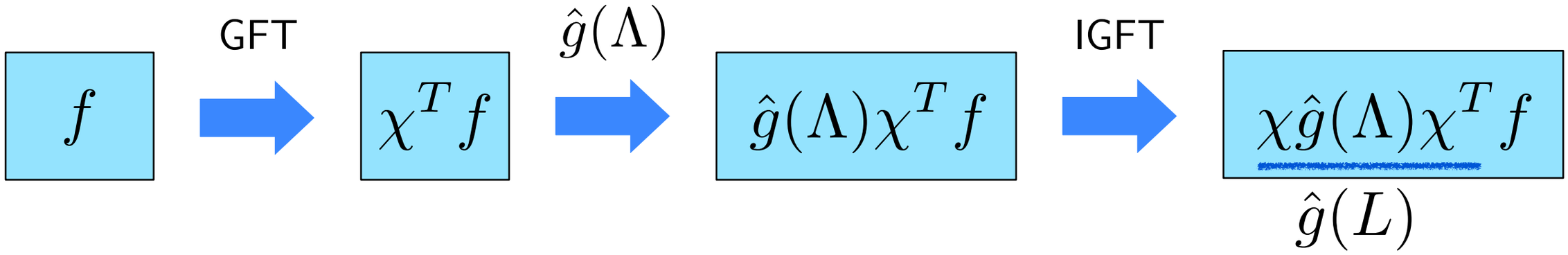
- Consider a noisy image as the observed noisy graph signal
- Consider a regular grid graph (weights inv. prop. to pixel value difference)



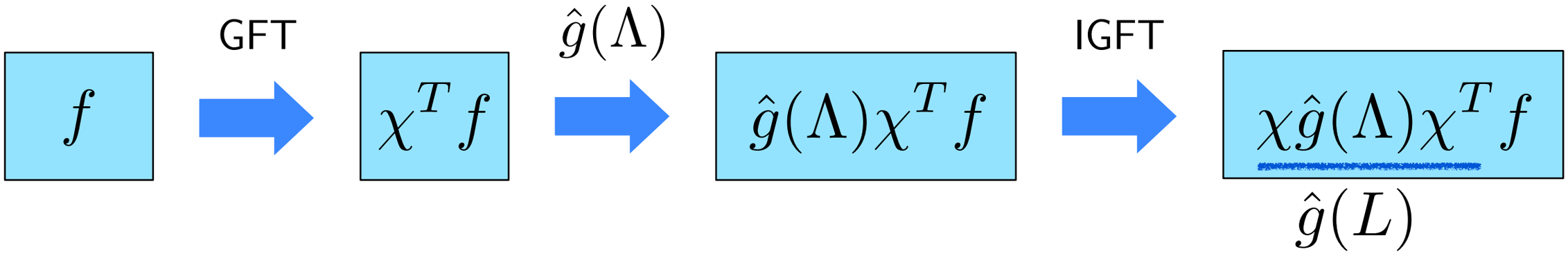
[FIGS2] Image denoising via classical Gaussian filtering and graph spectral filtering.

[Shuman13]

Example designs



Example designs



Low-pass filters: $\hat{g}(L) = (I + \gamma L)^{-1} = \chi(I + \gamma\Lambda)^{-1}\chi^T$

Shifted and dilated band-pass filters: Spectral graph wavelets $\hat{g}(sL)$ [Hammond11, Shuman11, Dong13]

Window kernel: Windowed graph Fourier transform [Shuman12]

Parametric polynomials: $\hat{g}_s(L) = \sum_{k=0}^{K} \alpha_{sk} L^k = \chi(\sum_{k=0}^{K} \alpha_{sk}\Lambda^k)\chi^T$ [Thanou14]

Adapted kernels: Learn values of $\hat{g}(L)$ directly from data [Zhang12]

Spectral graph wavelets

$$\psi_{s,a}(x) = \frac{1}{s}\psi\left(\frac{x-a}{s}\right)$$

$$W_f(s,a) = \int_{-\infty}^{\infty} \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right) f(x)\,dx$$

$$\bar{\psi}_s(x) = \frac{1}{s}\psi^*\left(\frac{-x}{s}\right)$$

$$(T^s f)(a) = \int_{-\infty}^{\infty} \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right) f(x)\,dx = \int_{-\infty}^{\infty} \bar{\psi}_s(a-x)\, f(x)\,dx$$
$$= (\bar{\psi}_s \star f)(a)$$

$$\widehat{T^s f}(\omega) = \hat{\bar{\psi}}_s(\omega)\hat{f}(\omega) = \hat{\psi}^*(s\omega)\hat{f}(\omega)$$

$$(T^s f)(a) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{i\omega a}\hat{\psi}^*(s\omega)\hat{f}(\omega)\,d\omega$$

Fourier multiplier operator: scaled kernel $\hat{\psi}^*(s\omega)$

Spectral graph wavelets

$$\psi_{s,a}(x) = \frac{1}{s}\psi\left(\frac{x-a}{s}\right)$$

$$W_f(s,a) = \int_{-\infty}^{\infty} \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right) f(x)\,dx$$

$$\bar{\psi}_s(x) = \frac{1}{s}\psi^*\left(\frac{-x}{s}\right)$$

$$(T^s f)(a) = \int_{-\infty}^{\infty} \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right) f(x)\,dx = \int_{-\infty}^{\infty} \bar{\psi}_s(a-x)\, f(x)\,dx$$

$$= (\bar{\psi}_s \star f)(a)$$

$$\widehat{T^s f}(\omega) = \hat{\bar{\psi}}_s(\omega)\hat{f}(\omega) = \hat{\psi}^*(s\omega)\hat{f}(\omega)$$

$$(T^s f)(a) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{i\omega a}\hat{\psi}^*(s\omega)\hat{f}(\omega)\,d\omega$$

$$\widehat{T_g^s f}(\ell) = g(s\lambda_\ell)\hat{f}(\ell)$$

$T_g^s = g(sL)$

Fourier multiplier operator: scaled kernel $\hat{\psi}^*(s\omega)$

Spectral graph wavelets

$$\psi_{s,a}(x) = \frac{1}{s}\psi\left(\frac{x-a}{s}\right)$$

$$W_f(s,a) = \int_{-\infty}^{\infty} \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right) f(x)\,dx$$

$$\bar{\psi}_s(x) = \frac{1}{s}\psi^*\left(\frac{-x}{s}\right)$$

$$(T^s f)(a) = \int_{-\infty}^{\infty} \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right) f(x)\,dx = \int_{-\infty}^{\infty} \bar{\psi}_s(a-x)\, f(x)\,dx$$

$$= (\bar{\psi}_s \star f)(a)$$

$$\widehat{T^s f}(\omega) = \hat{\bar{\psi}}_s(\omega)\hat{f}(\omega) = \hat{\psi}^*(s\omega)\hat{f}(\omega)$$

$$(T^s f)(a) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{i\omega a}\hat{\psi}^*(s\omega)\hat{f}(\omega)\,d\omega$$

$$T_g^s = g(sL)$$

$$\widehat{T_g^s f}(\ell) = g(s\lambda_\ell)\hat{f}(\ell)$$

$$(T_g^s f)(i) = \sum_{\ell=0}^{N-1} g(s\lambda_\ell)\hat{f}(\ell)\chi_\ell(i)$$

Fourier multiplier operator: scaled kernel $\hat{\psi}^*(s\omega)$

Spectral graph wavelets

$$\psi_{s,a}(x) = \frac{1}{s}\psi\left(\frac{x-a}{s}\right)$$

$$W_f(s,a) = \int_{-\infty}^{\infty} \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right) f(x)\,dx$$

$$\bar{\psi}_s(x) = \frac{1}{s}\psi^*\left(\frac{-x}{s}\right)$$

$$(T^s f)(a) = \int_{-\infty}^{\infty} \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right) f(x)\,dx = \int_{-\infty}^{\infty} \bar{\psi}_s(a-x)\, f(x)\,dx$$

$$= (\bar{\psi}_s \star f)(a)$$

$$\widehat{T^s f}(\omega) = \hat{\bar{\psi}}_s(\omega)\hat{f}(\omega) = \hat{\psi}^*(s\omega)\hat{f}(\omega)$$

$$(T^s f)(a) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{i\omega a}\hat{\psi}^*(s\omega)\hat{f}(\omega)\,d\omega$$

$$T_g^s = g(sL)$$

$$\widehat{T_g^s f}(\ell) = g(s\lambda_\ell)\hat{f}(\ell)$$

$$(T_g^s f)(i) = \sum_{\ell=0}^{N-1} g(s\lambda_\ell)\hat{f}(\ell)\chi_\ell(i)$$

Fourier multiplier operator: scaled kernel $\hat{\psi}^*(s\omega)$

Spectral graph wavelets

$$\psi_{s,a}(x) = \frac{1}{s}\psi\left(\frac{x-a}{s}\right)$$

$$W_f(s,a) = \int_{-\infty}^{\infty} \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right) f(x)\,dx$$

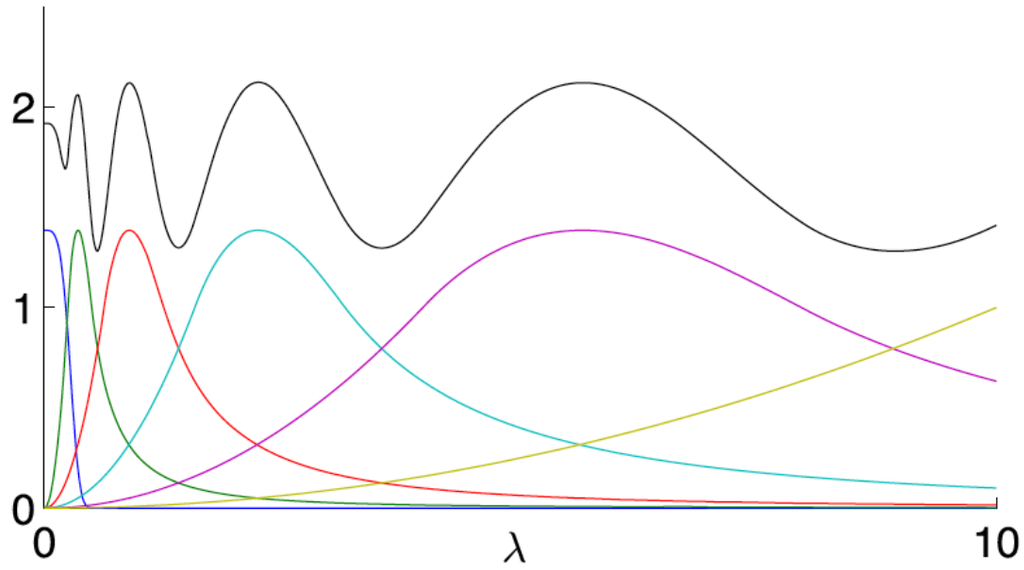
$$\bar{\psi}_s(x) = \frac{1}{s}\psi^*\left(\frac{-x}{s}\right)$$

$$(T^s f)(a) = \int_{-\infty}^{\infty} \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right) f(x)\,dx = \int_{-\infty}^{\infty} \bar{\psi}_s(a-x)\, f(x)\,dx$$

$$= (\bar{\psi}_s \star f)(a)$$

$$\widehat{T^s f}(\omega) = \hat{\bar{\psi}}_s(\omega)\hat{f}(\omega) = \hat{\psi}^*(s\omega)\hat{f}(\omega)$$

$$(T^s f)(a) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{i\omega a}\hat{\psi}^*(s\omega)\hat{f}(\omega)\,d\omega$$



$$\widehat{T_g^s f}(\ell) = g(s\lambda_\ell)\hat{f}(\ell)$$

$$T_g^s = g(sL)$$

$$(T_g^s f)(i) = \sum_{\ell=0}^{N-1} g(s\lambda_\ell)\hat{f}(\ell)\chi_\ell(i)$$

Fourier multiplier operator: scaled kernel $\hat{\psi}^*(s\omega)$

Spectral graph wavelets

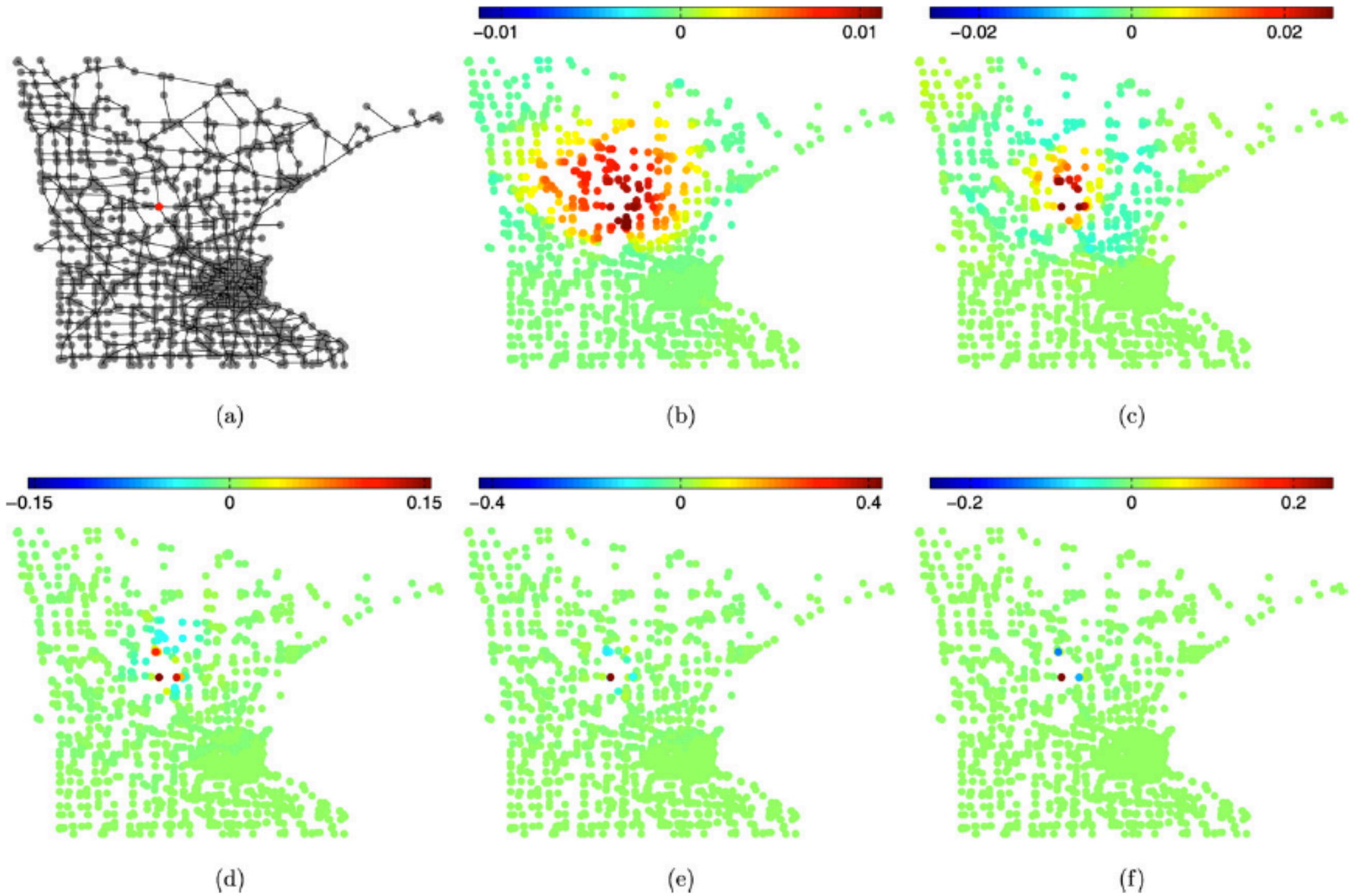


Fig. 4. Spectral graph wavelets on Minnesota road graph, with $K = 100$, $J = 4$ scales. (a) Vertex at which wavelets are centered, (b) scaling function, (c)–(f) wavelets, scales 1–4.

[Hammond11]

Outline

- Motivation
- Graph signal processing (GSP): Basic concepts
- Spectral filtering: Basic tools of GSP
- Applications and perspectives

Applications of GSP

- Signal processing and machine learning tasks
  - Denoising [Graichen15, Liu16]
  - Semi-supervised learning / Classification [Kipf16, Manessi17]
  - Clustering [Tremblay14, Tremblay16]
  - Dimensionality reduction [Rui16]

Applications of GSP

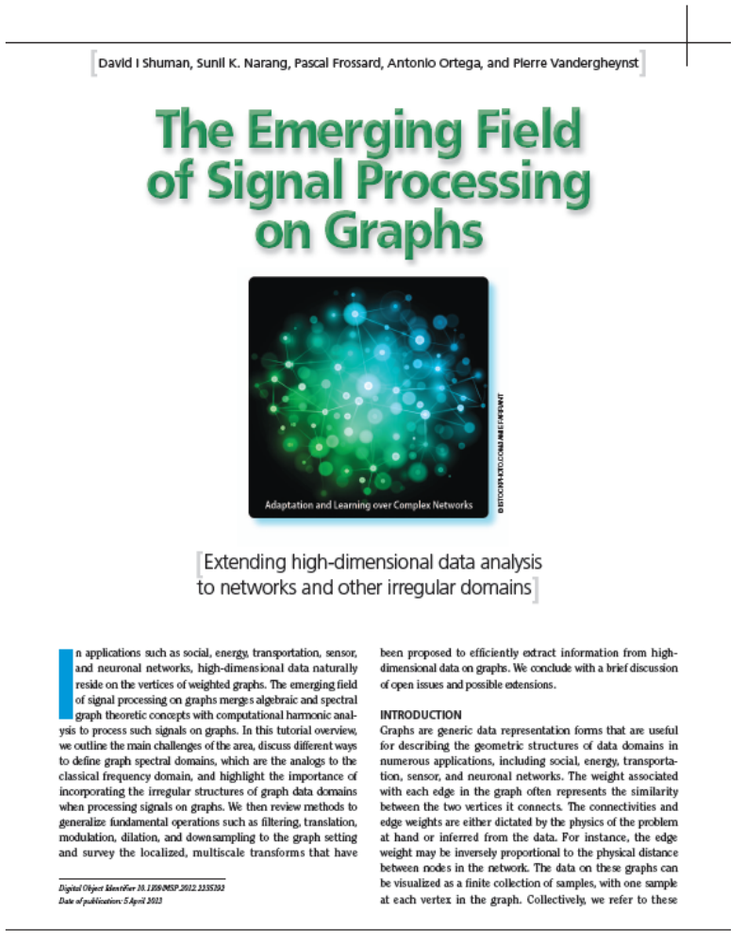
- Signal processing and machine learning tasks
  - Denoising [Graichen15, Liu16]
  - Semi-supervised learning / Classification [Kipf16, Manessi17]
  - Clustering [Tremblay14, Tremblay16]
  - Dimensionality reduction [Rui16]
- Application domains
  - Neuroimaging / Brain activity analysis [Huang16, Smith17, Ktena17]
  - Social network analysis (e.g., community detection [Bruna17], recommendation [Monti17], link prediction [Schlichtkrull17])
  - Urban computing (e.g., mobility inference [Dong13])
  - Computer graphics [Monti16, Yi16, Wang17, Simonovsky17]
  - Geoscience and remote sensing [Bayram17]

Future of GSP

- Mathematical models for graph signals
  - global and local smoothness / regularity
  - underlying physical processes
- Graph construction
  - how to infer topologies given observed data?
- Fast implementation
  - fast graph Fourier transform
  - distributed processing
- Connection to / combination with other fields
  - statistical machine learning
  - deep learning (on graphs and manifolds)
- Applications

References

- Three review papers:



David I Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst

The Emerging Field of Signal Processing on Graphs

Extending high-dimensional data analysis to networks and other irregular domains

In applications such as social, energy, transportation, sensor, and neuronal networks, high-dimensional data naturally reside on the vertices of weighted graphs. The emerging field of signal processing on graphs merges algebraic and spectral graph theoretic concepts with computational harmonic analysis to process such signals on graphs. In this tutorial overview, we outline the main challenges of the area, discuss different ways to define graph spectral domains, which are the analogs to the classical frequency domain, and highlight the importance of incorporating the irregular structures of graph data domains when processing signals on graphs. We then review methods to generalize fundamental operations such as filtering, translation, modulation, dilation, and downsampling to the graph setting and survey the localized, multiscale transforms that have been proposed to efficiently extract information from high-dimensional data on graphs. We conclude with a brief discussion of open issues and possible extensions.

Digital Object Identifier 10.1109/MSP.2012.2235192
Date of publication: 5 April 2013

INTRODUCTION

Graphs are generic data representation forms that are useful for describing the geometric structures of data domains in numerous applications, including social, energy, transportation, sensor, and neuronal networks. The weight associated with each edge in the graph often represents the similarity between the two vertices it connects. The connectivities and edge weights are either dictated by the physics of the problem at hand or inferred from the data. For instance, the edge weight may be inversely proportional to the physical distance between nodes in the network. The data on these graphs can be visualized as a finite collection of samples, with one sample at each vertex in the graph. Collectively, we refer to these

Discrete Signal Processing on Graphs

Aliaksei Sandryhaila, *Member, IEEE*, and José M. F. Moura, *Fellow, IEEE*

***Abstract*—In social settings, individuals interact through webs of relationships. Each individual is a node in a complex network (or graph) of interdependencies and generates data, lots of data. We label the data by its source, or formally stated, we *index* the data by the nodes of the graph. The resulting signals (data indexed by the nodes) are far removed from time or image signals indexed by well ordered time samples or pixels. DSP, discrete signal processing, provides a comprehensive, elegant, and efficient methodology to describe, represent, transform, analyze, process, or synthesize these well ordered time or image signals. This paper extends to *signals on graphs* DSP and its basic tenets, including filters, convolution, z-transform, impulse response, spectral representation, Fourier transform, frequency response, and illustrates *DSP on graphs* by classifying blogs, linear predicting and compressing data from irregularly located weather stations, or predicting behavior of customers of a mobile service provider.**

***Index Terms*—Graph Fourier transform, graphical models, Markov random fields, network science, signal processing.**

I. Introduction

THERE is an explosion of interest in processing and analyzing large datasets collected in very different settings, including social and economic networks, information networks, internet and the world wide web, immunization and epidemiology networks, molecular and gene regulatory networks, citation and coauthorship studies, friendship networks, as well as physical infrastructure networks like sensor networks, power grids, transportation networks, and other networked critical infrastructures. We briefly overview some of the existing work.

Many authors focus on the underlying *relational structure* of the data by: 1) inferring the structure from community relations and friendships, or from perceived alliances between agents as abstracted through game theoretic models [1], [2]; 2) quantifying the connectedness of the world; and 3) determining the relevance of particular agents, or studying the strength of their interactions. Other authors are interested in the *network function* by quantifying the impact of the network structure on the diffusion of disease, spread of news and information, voting trends, imitation and social influence, crowd behavior, failure propagation, global behaviors developing from seemingly random local interactions [2]–[4]. Much of these works either develop or assume network models that capture the interdependencies among the data and then analyze the structural properties of these networks. Models often considered may be deterministic like complete or regular graphs, or random like the Erdős-Rényi and Poisson graphs, the configuration and expected degree models, small world or scale free networks [2], [4], to mention a few. These models are used to quantify network characteristics, such as connectedness, existence and size of the giant component, distribution of component sizes, degree and clique distributions, and node or edge specific parameters including clustering coefficients, path length, diameter, betweenness and closeness centralities.

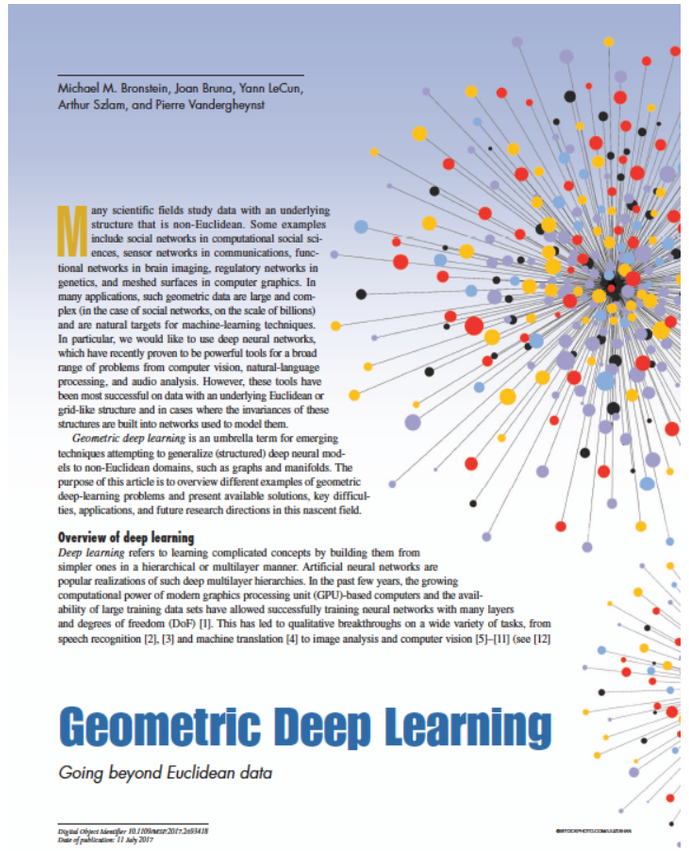
Another body of literature is concerned with inference and learning from such large datasets. Much work falls under the generic label of graphical models [5]–[10]. In graphical models, data is viewed as a family of random variables indexed by the nodes of a graph, where the graph captures probabilistic dependencies among data elements. The random variables are described by a family of joint probability distributions. For example, directed (acyclic) graphs [11], [12] represent Bayesian networks where each random variable is independent of others given the variables defined on its parent nodes. Undirected graphical models, also referred to as Markov random fields [13], [14], describe data where the variables defined on two sets of nodes separated by a boundary set of nodes are statistically independent given the variables on the boundary set. A key tool in graphical models is the Hammersley-Clifford theorem [13], [15], [16], and the Markov-Gibbs equivalence that, under appropriate positivity conditions, factors the joint distribution of the graphical model as a product of potentials defined on the cliques of the graph. Graphical models exploit this factorization and the structure of the indexing graph to develop efficient algorithms for inference by controlling their computational cost. Inference in graphical models is generally defined as finding from the joint distributions lower order marginal distributions, likelihoods, modes, and other moments of individual variables or their subsets. Common inference algorithms include belief propagation and its generalizations, as well as other message passing algorithms. A recent block-graph algorithm for fast approximate inference, in which the nodes are non-overlapping clusters of nodes from the original graph, is in [17]. Graphical models are employed in many areas; for sample applications, see [18] and references therein.

Extensive work is dedicated to discovering efficient data representations for large high-dimensional data [19]–[22]. Many of these works use spectral graph theory and the graph Laplacian [23] to derive low-dimensional representations by projecting the data on a low-dimensional subspace generated by a small subset of the Laplacian eigenbasis. The graph Laplacian approximates the Laplace-Beltrami operator on a compact manifold [21], [24], in the sense that if the dataset is large and sam-

Manuscript received October 15, 2012; revised December 16, 2012; accepted December 17, 2012. Date of publication January 11, 2013; date of current version March 08, 2013. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Xiang-Gen Xia. This work was supported in part by AFOSR grant FA95501210087.

The authors are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213-3890 USA (e-mail: asandryh@andrew.cmu.edu; moura@ece.cmu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.
Digital Object Identifier 10.1109/TSP.2013.2238935



Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst

Many scientific fields study data with an underlying structure that is non-Euclidean. Some examples include social networks in computational social sciences, sensor networks in communications, functional networks in brain imaging, regulatory networks in genetics, and meshed surfaces in computer graphics. In many applications, such geometric data are large and complex (in the case of social networks, on the scale of billions) and are natural targets for machine-learning techniques. In particular, we would like to use deep neural networks, which have recently proven to be powerful tools for a broad range of problems from computer vision, natural-language processing, and audio analysis. However, these tools have been most successful on data with an underlying Euclidean or grid-like structure and in cases where the invariances of these structures are built into networks used to model them.

Geometric deep learning is an umbrella term for emerging techniques attempting to generalize (structured) deep neural models to non-Euclidean domains, such as graphs and manifolds. The purpose of this article is to overview different examples of geometric deep-learning problems and present available solutions, key difficulties, applications, and future research directions in this nascent field.

Overview of deep learning

Deep learning refers to learning complicated concepts by building them from simpler ones in a hierarchical or multilayer manner. Artificial neural networks are popular realizations of such deep multilayer hierarchies. In the past few years, the growing computational power of modern graphics processing unit (GPU)-based computers and the availability of large training data sets have allowed successfully training neural networks with many layers and degrees of freedom (DoF) [1]. This has led to qualitative breakthroughs on a wide variety of tasks, from speech recognition [2], [3] and machine translation [4] to image analysis and computer vision [5]–[11] (see [12]

Geometric Deep Learning

Going beyond Euclidean data

Digital Object Identifier 10.1109/MSP.2017.2693418
Date of publication: 11 July 2017

Resources

- Graph signal processing
  - MATLAB toolbox: https://lts2.epfl.ch/gsp/
  - Python toolbox: https://pygsp.readthedocs.io/en/stable/

- Spectral graph wavelet transform
  - MATLAB toolbox: https://wiki.epfl.ch/sgwt
  - Python toolbox: https://github.com/aweinstein/pysgwt

- Topology inference
  - Tutorial: http://web.media.mit.edu/~xdong/presentation/GSPW_GraphLearning.pdf

- Geometric deep learning
  - Workshops, tutorials, papers and code: http://geometricdeeplearning.com

contact: xiaowen.dong@eng.ox.ac.uk