Lecture 7
Signal Processing on Graphs: Basic Concepts and Theory
Outline

• Motivation

• Graph signal processing (GSP): Basic concepts

• Spectral filtering: Basic tools of GSP

• Connection with literature
Outline

• Motivation

• Graph signal processing (GSP): Basic concepts

• Spectral filtering: Basic tools of GSP

• Connection with literature
Data are often structured
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

\[ f : V \rightarrow \mathbb{R}^N \]
Graphs are appealing tools

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

\[ f : \mathcal{V} \rightarrow \mathbb{R}^N \]

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: Number of 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 generalise classical signal processing tools on irregular domains such as graphs?

\[ f : \mathcal{V} \rightarrow \mathbb{R}^N \]
Graph signal processing

- Graph signals provide a nice compact format to encode structure within data
- Generalisation of classical signal processing tools can greatly benefit analysis of such data
- Numerous applications: Transportation, biomedical, social network analysis, etc.

\[ f : \mathcal{V} \rightarrow \mathbb{R}^N \]
Graph signal processing

- Graph signals provide a nice compact format to encode structure within data
- Generalisation 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
  - machine learning

\[ f : \mathcal{V} \rightarrow \mathbb{R}^N \]
Outline

- Motivation
- Graph signal processing (GSP): Basic concepts
  - Spectral filtering: Basic tools of GSP
- Connection with literature
Two paradigms

- The main approaches can be categorised into two families:
  - vertex (spatial) domain designs
  - frequency (graph spectral) domain designs
Two paradigms

- The main approaches can be categorised into two families:
  - vertex (spatial) domain designs
  - frequency (graph spectral) domain designs

Important for analysis of signal properties

\[ f : \mathcal{V} \rightarrow \mathbb{R}^N \]
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

A notion of frequency for graph signals:

We need the graph Laplacian matrix
Graph Laplacian

Weighted and undirected graph:

\[ \mathcal{G} = \{ \mathcal{V}, \mathcal{E} \} \]

\[
W = \\
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]
**Graph Laplacian**

Weighted and undirected graph:

$G = \{ \mathcal{V}, \mathcal{E} \}$

$D = \text{diag}(d(v_1), \ldots, d(v_N))$

\[
\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}

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

D

W
Graph Laplacian

Weighted and undirected graph:

\[ G = \{ \mathcal{V}, \mathcal{E} \} \]

\[ D = \text{diag}(d(v_1), \cdots, d(v_N)) \]

\[ L = D - W \quad \text{Equivalent to } G! \]

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

\[
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}
- \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1
\end{pmatrix}
- \begin{pmatrix}
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\
0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\
0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\
0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\
0 & 0 & -1 & 0 & -1 & 3 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1
\end{pmatrix}
\]
Graph Laplacian

Weighted and undirected graph:
\[ G = \{ \mathcal{V}, \mathcal{E} \} \]
\[ D = \text{diag}(d(v_1), \cdots, d(v_N)) \]
\[ L = D - W \quad \text{Equivalent to } G! \]
\[ L_{\text{norm}} = D^{-\frac{1}{2}} (D - W) D^{-\frac{1}{2}} \]

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

Why graph Laplacian?
Graph Laplacian

Why graph Laplacian?
- approximation of the Laplace operator

\[(Lf)(i) = 4f(i) - [f(j_1) + f(j_2) + f(j_3) + f(j_4)]\]

standard 5-point stencil for approximating \(-\nabla^2 f\)
Graph Laplacian

Why graph Laplacian?
- approximation of the Laplace operator

\[(L f)(i) = 4f(i) - [f(j_1) + f(j_2) + f(j_3) + f(j_4)]\]

standard 5-point stencil for approximating \(-\nabla^2 f\)

- converges to the Laplace-Beltrami operator given some conditions
- provides a notion of “frequency” on graphs
Graph Laplacian

Graph signal $f : \mathcal{V} \rightarrow \mathbb{R}^N$
Graph Laplacian

Graph signal \( f : \mathcal{V} \rightarrow \mathbb{R}^N \)

\[
Lf(i) = \sum_{j=1}^{N} W_{ij} (f(i) - f(j))
\]
Graph Laplacian

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

$$Lf(i) = \sum_{j=1}^{N} W_{ij} (f(i) - f(j))$$

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

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

\[
L = \begin{bmatrix}
\chi_0 & \cdots & \chi_{N-1}
\end{bmatrix} \begin{bmatrix}
\lambda_0 & 0 \\
0 & \ddots & \\
& \ddots & \lambda_{N-1}
\end{bmatrix} \begin{bmatrix}
\chi_0 & \\
& \chi_{N-1}
\end{bmatrix}
\]
Graph Laplacian

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

\[
L = \begin{bmatrix}
\chi_0 & \cdots & \chi_{N-1} \\
\vdots & \ddots & \vdots \\
\chi_{N-1} & \cdots & \chi_0
\end{bmatrix}
\begin{bmatrix}
\lambda_0 & 0 \\
0 & \ddots \\
0 & \cdots & \lambda_{N-1}
\end{bmatrix}
\begin{bmatrix}
\chi_0 \\
\vdots \\
\chi_{N-1}
\end{bmatrix}
\]

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

\[ \chi_0 \quad \chi_1 \quad \chi_{50} \quad [\text{Shuman13}] \]
Graph Fourier transform

\[ \chi_0 \quad \chi_1 \quad \chi_{50} \quad [\text{Shuman13}] \]
Graph Fourier transform

- Eigenvectors associated with smaller eigenvalues have values that vary less rapidly along the edges.
Graph Fourier transform

Graph Fourier transform:

\[ L = \chi \Lambda \chi^T \]

\[
\begin{align*}
\chi_0^T L \chi_0 &= \lambda_0 = 0 \\
\chi_{50}^T L \chi_{50} &= \lambda_{50}
\end{align*}
\]

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

[Shuman13, Hammond11]
Graph Fourier transform

\[ L = \chi \Lambda \chi^T \]

\[ \chi_0^T L \chi_0 = \lambda_0 = 0 \]

\[ \chi_{50}^T L \chi_{50} = \lambda_{50} \]

Graph Fourier transform:
[Hammond11]

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

Low frequency \[ \lambda_0 \lambda_1 \lambda_2 \lambda_3 \lambda_4 \cdots \lambda_{N-1} \]
High frequency

Low frequency
High frequency
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: $-\nabla^2$

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: $-\nabla^2$

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: $\chi_\ell$

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: $-\nabla^2$

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

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

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

graph Laplacian: $L$

- eigenvectors: $\chi_\ell$

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:
  \[
  \chi_\ell = \begin{bmatrix}
  1, \omega^\ell, \omega^{2\ell}, \ldots, \omega^{(N-1)\ell}
  \end{bmatrix}, \text{ where } \omega = e^{\frac{2\pi i}{N}}
  \]

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

[Vandergheynst11]
Two special cases

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

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

• Connection with literature
Classical frequency filtering

Classical FT:
\[
\hat{f}(\omega) = \int (e^{i\omega x})*f(x)\,dx \quad f(x) = \frac{1}{2\pi} \int \hat{f}(\omega)e^{i\omega x}\,d\omega
\]
Classical frequency filtering

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

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

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

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 \( f : \mathcal{V} \rightarrow \mathbb{R}^N \)
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 \( f : \mathcal{V} \rightarrow \mathbb{R}^N \)
Graph spectral filtering

\[ \hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi^*_\ell(i) f(i) \quad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i) \]

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

\[ \hat{g}(\Lambda) = \begin{bmatrix} \hat{g}(\lambda_0) & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \hat{g}(\lambda_{N-1}) \end{bmatrix} \]
Graph Laplacian revisited

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

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

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

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

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!

\[ f \xrightarrow{\text{GFT}} \chi^T f \xrightarrow{\hat{g}(\Lambda)} \hat{g}(\Lambda)\chi^T f \xrightarrow{\text{IGFT}} \chi\hat{g}(\Lambda)\chi^T f \]
Transforms and dictionaries can be designed through graph spectral filtering: Functions of graph Laplacian!

\[ f \xrightarrow{\text{GFT}} \chi^T f \xrightarrow{\hat{g}(\Lambda)} \hat{g}(\Lambda)\chi^T f \xrightarrow{\text{IGFT}} \chi\hat{g}(\Lambda)\chi^T f \]

\(\hat{g}(L)\): functions of \(L\)!
Graph transform/dictionary design

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

\[ f \xrightarrow{\text{GFT}} \chi^T f \xrightarrow{\hat{g}(\Lambda)} \hat{g}(\Lambda)\chi^T f \xrightarrow{\text{IGFT}} \chi\hat{g}(\Lambda)\chi^T f \]

\( \hat{g}(L) \): functions of \( L \)

- Important properties can be achieved by properly defining \( \hat{g}(L) \), such as localisation of atoms

- Closely related to kernels and regularisation on graphs [Smola03]
A simple example

\[ f \xrightarrow{\text{GFT}} \chi^T f \xrightarrow{\hat{g}(\Lambda)} \hat{g}(\Lambda) \chi^T f \xrightarrow{\text{IGFT}} \chi \hat{g}(\Lambda) \chi^T f \]

\[ \hat{g}(L) \]
A simple example

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

\[
y^* = \arg \min_y \{ ||y - f||_2^2 + \gamma y^T Ly \}
\]
A simple example

\[ f \xrightarrow{\text{GFT}} \chi^T f \xrightarrow{\hat{g}(\Lambda)} \hat{g}(\Lambda)\chi^T f \xrightarrow{\text{IGFT}} \chi\hat{g}(\Lambda)\chi^T f \]

\[ \hat{g}(L) \]

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

\[ y^* = \arg\min_y \left\{ \|y - f\|_2^2 + \gamma y^T Ly \right\} \]

Data fitting term

“Smoothness” assumption
A simple example

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

\[
y^* = \arg\min_y \left\{ \|y - f\|_2^2 + \gamma y^T Ly \right\}
\]

\[
y^* = (I + \gamma L)^{-1} f
\]

Laplacian (Tikhonov) regularisation is equivalent to low-pass filtering in the graph spectral domain!
A simple example

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

\[ w_{ij} = \frac{1}{|f(i) - f(i)|} \]
A simple example

- 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

\[ f \rightarrow \chi^T f \rightarrow \hat{g}(\Lambda) \rightarrow \hat{g}(\Lambda)\chi^T f \rightarrow \chi\hat{g}(\Lambda)\chi^T f \]

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

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

Window kernel: Windowed graph Fourier transform \[ \text{[Shuman12]} \]
Example designs

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

Window kernel: Windowed graph Fourier transform  [Shuman12]

Shifted and dilated band-pass filters: Spectral graph wavelets \[
\hat{g}(sL)
\]  [Hammond11]
Example designs

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

Window kernel: Windowed graph Fourier transform  [Shuman12]

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

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

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

Window kernel: Windowed graph Fourier transform \([\text{Shuman12}]\)

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

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

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

• Motivation
• Graph signal processing (GSP): Basic concepts
• Spectral filtering: Basic tools of GSP
• Connection with literature
GSP and the literature

There is a rich literature about data analysis and learning on graphs
GSP and the literature

There is a rich literature about data analysis and learning on graphs
GSP and the literature

There is a rich literature about data analysis and learning on graphs.
GSP and the literature

There is a rich literature about data analysis and learning on graphs:

- **Network science**
- **Diffusion on graphs**
- **Unsupervised learning** (dimensionality reduction, clustering)
GSP and the literature

There is a rich literature about data analysis and learning on graphs

network science

unsupervised learning (dimensionality reduction, clustering)

diffusion on graphs

semi-supervised learning

[Zhou04]
Network centrality

**eigenvector centrality**

\[ Wx = \lambda_{\text{max}} x \]

**degree centrality**

\[ d = [d(v_1), \ldots, d(v_N)] \]
Network centrality

- Google’s PageRank is a variant of eigenvector centrality
- Eigenvectors of W can also be used to provide a frequency interpretation for graph signals [Sandryhaila13]

PageRank: http://www.ams.org/publicoutreach/feature-column/fcarc-pagerank
Diffusion on graphs

heat diffusion
Diffusion on graphs

\[
\frac{\partial x}{\partial \tau} - Lx = 0
\]

\[
x(v, 0) = x_0(v)
\]

\[
x(v, \tau) = e^{-\tau L} x_0(v)
\]
Diffusion on graphs

- heat diffusion on graphs is a typical physical process on graphs
- other possibilities exist (e.g., random walk on graphs) [Smola03]
- nearly all of them has an interpretation of filtering on graphs

Graph clustering (community detection)
Graph clustering (community detection)
Graph clustering (community detection)

\[ NCut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \overline{A_i})}{vol(A_i)} \]
Graph clustering (community detection)

- first $k$ eigenvectors of graph Laplacian minimise the graph cut
- eigenvectors of graph Laplacian enable a Fourier-like analysis for graph signals

$$NCut(A_1, ..., A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \overline{A_i})}{vol(A_i)}$$

Semi-supervised learning
Semi-supervised learning

\[
\min_{x \in \mathbb{R}^N} \| y - x \|^2_2 + \alpha x^T L x,
\]

\[
y : \begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
\vdots \\
0 \\
\end{bmatrix}
\]
Semi-supervised learning

- learning by assuming smoothness of predicted labels
- this is equivalent to a denoising problem for graph signal $y$
  (try it yourself during the lab session!)

GSP and the literature

centrality, propagating information, class membership, node labels (and features in general) can ALL be viewed as graph signals

network science

unsupervised learning (dimensionality reduction, clustering)

network diffusion

semi-supervised learning

[Zhou04]
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 tutorial/overview papers:

---

The Emerging Field of Signal Processing on Graphs

![Cover Image](image-url)

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

---

Discrete Signal Processing on Graphs

Aliaksei Sandalichuk, Member, IEEE, and Todor M. M. Milanov, Fellow, IEEE

Graph Signal Processing: Overview, Challenges, and Applications

This article presents methods to process data associated with graphs (graph signals) extending techniques (filters, sampling, and others) that are used for conventional signals.

By Artemes Orteg, Fellow IEEE, Paua Permut, Fellow IEEE, Rama Kezvavo, Fellow IEEE, Jose M. P. fortress, Fellow IEEE, and Prasant Vasudevan.
Resources

• Graph signal processing
  - MATLAB toolbox
    ▪ https://lts2.epfl.ch/gsp/
    ▪ https://github.com/STAC-USC/GraSP
  - Python toolbox

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