Graph Signal Processing

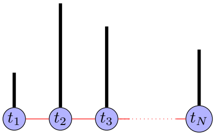
Rahul Singh

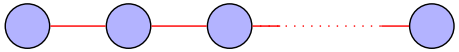
Sep 8, 2024

#0

- Next 90 $\pm$? minutes: signals processing over graphs
- About me
  - Signal Processing, Indian Institute of Space Science and Technology (graphs, Fourier, wavelets)
  - Electrical Engineering, Iowa State University (Image Processing)
  - Machine Learning, Georgia Tech (using structure in data)
  - Wu Tsai Postdoctoral Fellow, Yale (Computational Neuroscience)
    - Mentors: Smita Krishnaswamy and Joy Hirsch

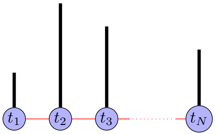
Classical Signal Processing

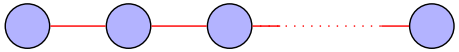




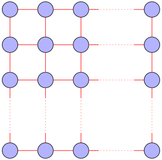
Structure behind time-series (speech, EEG, fMRI,. . .)

Classical Signal Processing





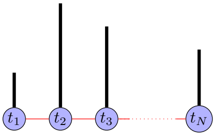
Structure behind time-series (speech, EEG, fMRI,. . .)

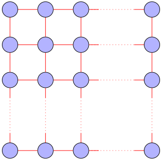


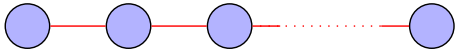
Structure behind image

Classical Signal Processing

- Translation, filtering, convolution, modulation, Fourier transform, sampling . . .



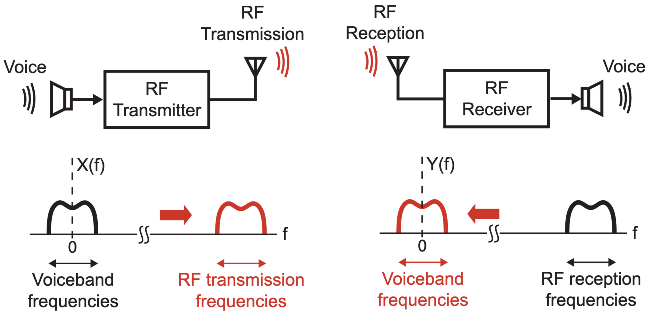




Structure behind time-series (speech, EEG, fMRI,. . .)

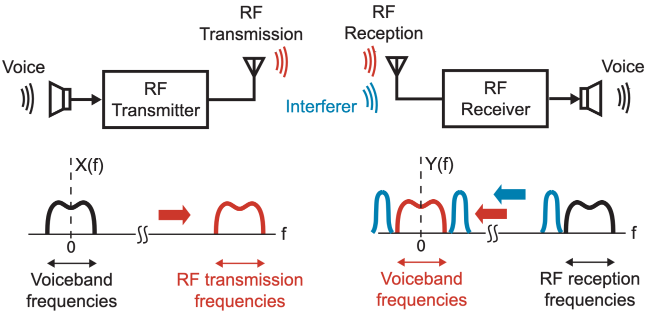
Structure behind image

Classical Signal Processing: Modulation



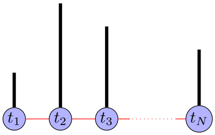
- Modulation is used to change the frequency band of a signal
  - Enables RF communication in different frequency bands
  - Used in cell phones, AM/FM radio, WLAN, cable TV, ...
  - Higher frequencies lead to smaller antennas

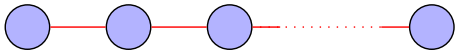
Classical Signal Processing: Filtering



- Filtering is used to remove undesired signals outside of the frequency band of interest
  - Enables selection of a specific radio, TV, WLAN, cell phone, cable TV
  - Also fundamental for denoising, smoothing, etc.

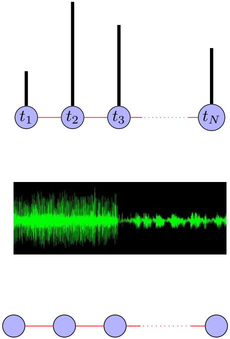
Classical Signal Processing



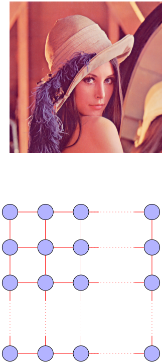


Structure behind time-series (speech, EEG, fMRI,...)

Classical Signal Processing



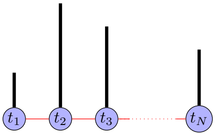
Structure behind time-series (speech, EEG, fMRI,. . .)

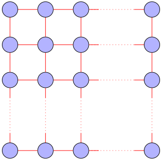


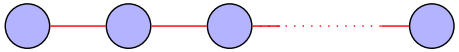
Structure behind image

Classical Signal Processing

- Translation, filtering, convolution, modulation, Fourier transform, sampling . . .



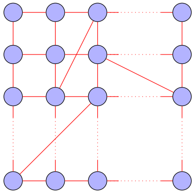




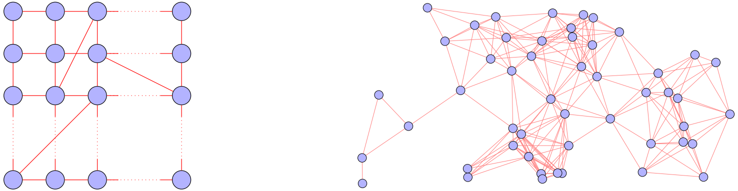
Structure behind time-series (speech, EEG, fMRI,. . .)

Structure behind image

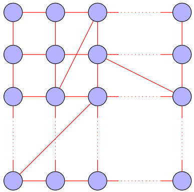
Graph Signal Processing

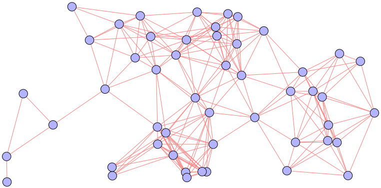


Graph Signal Processing

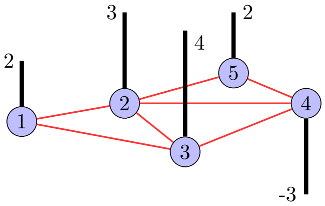


Graph Signal Processing

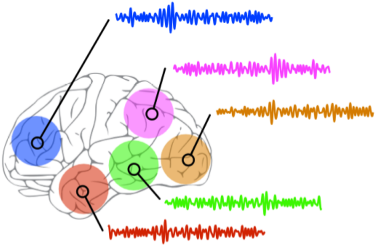




- A graph signal

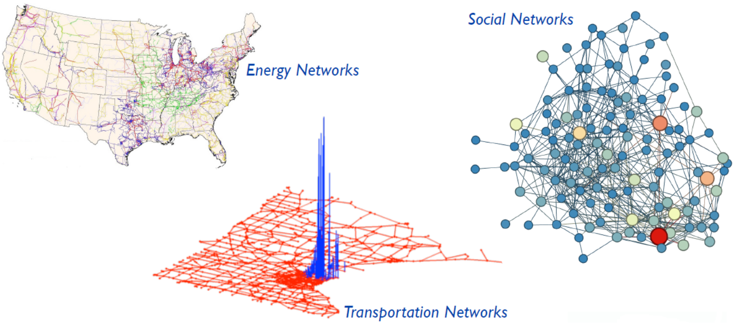


Graph Signal Processing



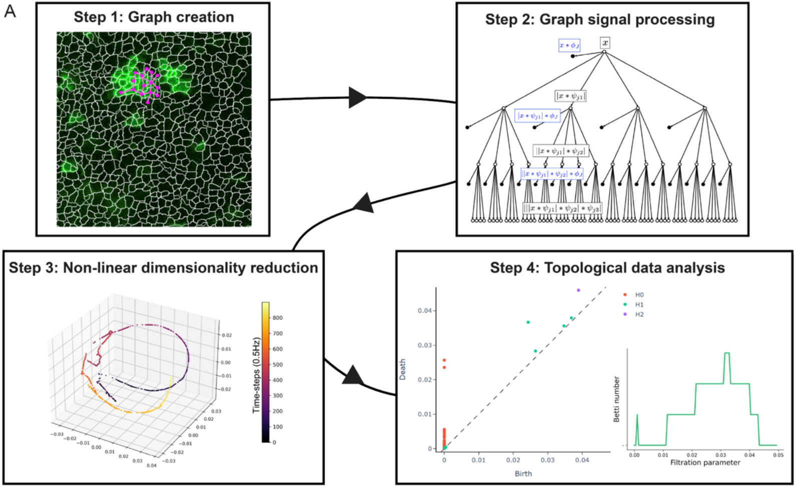
- Vertices: brain regions
- Edges: structural connectivity between brain regions
- Signal: blood-oxygen-level-dependent (BOLD) time series

Graph Signal Processing Applications



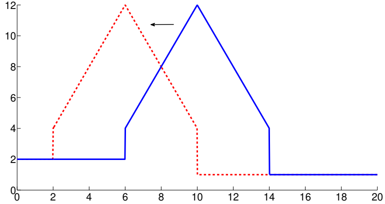
- Temperature/pressure recorded in a sensor network
- Number of followers of each user in a social network
- Traffic at each node in a road network
- Traffic at each node in a computer network

Geometric Scattering Trajectory Homology (GSTH)



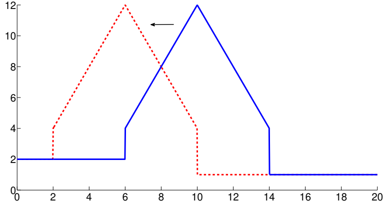
Difficulty in GSP

- Translation is simple in classical signal processing

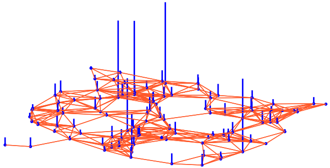


Difficulty in GSP

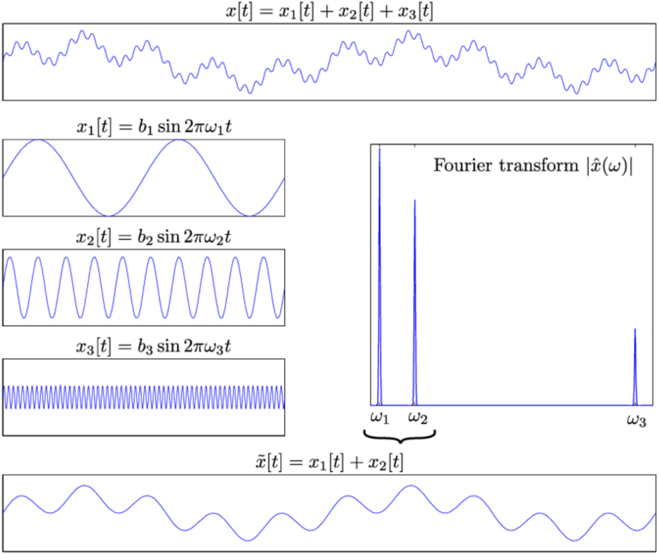
- Translation is simple in classical signal processing



- What does it mean to translate the signal to 'vertex 50'?
- Challenging in GSP

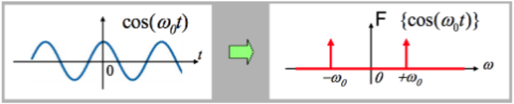


Need for Frequency

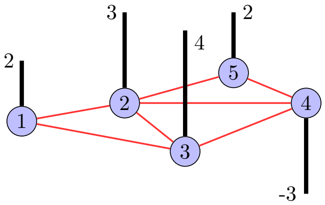


Need for Frequency

- Classical Fourier transform provides the frequency domain representation of signals

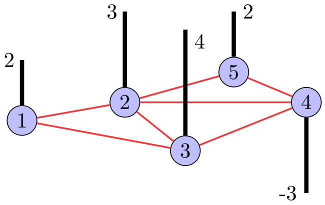


- A notion of frequency for graph signals?



- We need Laplacian Matrix

Notation



A graph signal $\mathbf{f}$

Graph $\mathcal{G} = (\mathcal{V},\ \mathbf{W})$

$$\mathbf{f} = \begin{bmatrix} 2 \\ 3 \\ 4 \\ -3 \\ 2 \end{bmatrix}$$

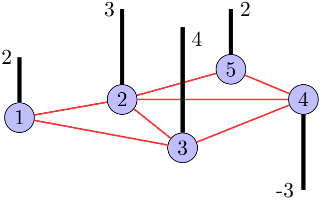
Weight matrix $\mathbf{W} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$

Degree matrix $\mathbf{D} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$

Laplacian matrix

$$\mathbf{L} = \mathbf{D} - \mathbf{W} = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$

Graph Laplacian



$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$

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

Multiplication by the Laplacian

- Signal $\mathbf{y} = \mathbf{L}\mathbf{x}$ results from multiplying $\mathbf{x}$ with the Laplacian
- $y_i = \sum_{j \in \mathcal{N}_i} w_{ij}(x_i - x_j)$
- Replaces x_i by weighted average of difference with neighbors
- Further Laplacian multiplications
  - $\mathbf{L}^2\mathbf{x}$ brings in features from 2-hop neighborhood
  - $\mathbf{L}^3\mathbf{x}$ brings in features from 3-hop neighborhood
  - $\mathbf{L}^k\mathbf{x}$ brings in features from k-hop neighborhood

Laplacian Quadratic Form

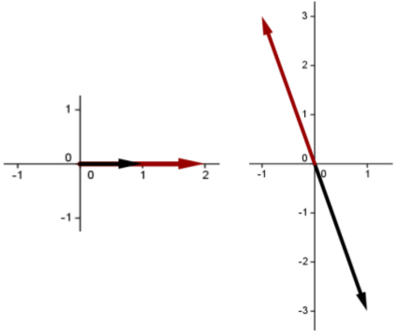
- The Laplacian quadratic form of graph signal $\mathbf{x}$ is $\mathbf{x}^T\mathbf{L}\mathbf{x}$

$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} w_{ij}(x_i - x_j)^2$$

- $\mathbf{x}^T\mathbf{L}\mathbf{x}$ quantifies the local variation of signal $\mathbf{x}$
  - Signals can be ordered depending on how much they vary
  - Will be used to order graph frequencies

Eigenvectors and Eigenvalues

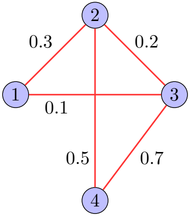
- For a square matrix $\mathbf{A}_{N\times N}$,
  - $\boxed{\mathbf{Au} = \lambda\mathbf{u}}$
  - $\mathbf{u}$ is an eigenvector
  - Scalar λ is the eigenvalue
- N eigenvalues and N eigenvectors
  - For $\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix}$,



$$\begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = (2)\begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ -3 \end{bmatrix} = (-1)\begin{bmatrix} 1 \\ -3 \end{bmatrix}$$

- $\lambda = 2$ and $\lambda = -1$ are eigenvalues.
- $\mathbf{u} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\mathbf{u} = \begin{bmatrix} 1 \\ -3 \end{bmatrix}$ are corresponding eigenvectors

Graph Spectrum



$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Eigenvalues (Graph Spectrum): $\{0, 0.8299, 2.6889, 4, 4.4812\}$

$$\mathbf{U} = [\mathbf{u}_0|\mathbf{u}_1|\mathbf{u}_2|\mathbf{u}_3|\mathbf{u}_4] = \begin{bmatrix} 0.4472 & 0.4375 & 0.7031 & 0 & 0.3380 \\ 0.4472 & 0.2560 & -0.2422 & 0.7071 & -0.4193 \\ 0.4472 & 0.2560 & -0.2422 & -0.7071 & -0.4193 \\ 0.4472 & -0.1380 & -0.5362 & 0 & 0.7024 \\ 0.4472 & -0.8115 & 0.3175 & 0 & -0.2018 \end{bmatrix}$$

Classical vs Graph Signal Processing

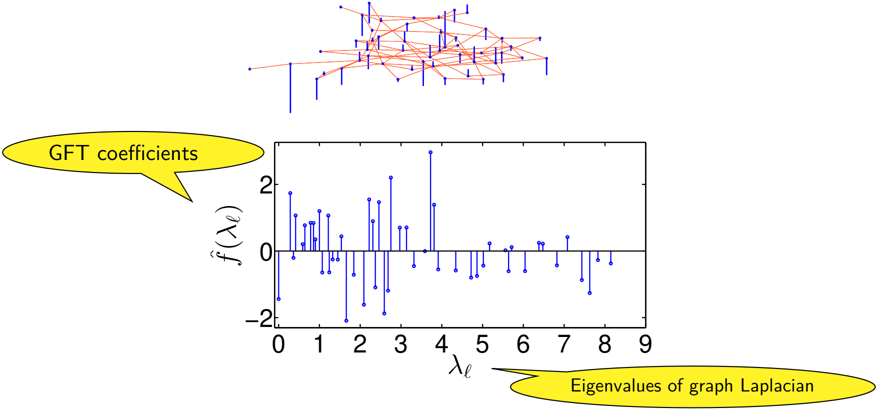
Operator/ Transform	Classical Signal Processing	Graph Signal Processing
Fourier Transform	▪ $\hat{x}(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt$ ▪ Frequency: ω can take any value ▪ Fourier basis: Complex exponentials $e^{j\omega t}$	▪ $\hat{\mathbf{f}}(\lambda_\ell) = \sum_{n=1}^{N} f(n)u_\ell^*(n)$ ▪ Frequency: Eigenvalues of the graph Laplacian (λ_ℓ) ▪ Fourier basis: Eigenvectors of the graph Laplacian $(\mathbf{u}_\ell)$
Convolution	▪ In time domain: $x(t) * y(t) = \int_{-\infty}^{\infty} x(\tau)y(t-\tau)d\tau$ ▪ In frequency domain: $\widehat{x(t) * y(t)} = \hat{x}(\omega)\hat{y}(\omega)$	▪ Defined through Graph Fourier Transform ▪ $\widehat{\mathbf{f} * \mathbf{g}} = \left(\hat{\mathbf{f}}.\hat{\mathbf{g}}\right)$
Translation	▪ Can be defined using convolution ▪ $T_\tau x(t) = x(t-\tau) = x(t) * \delta_\tau(t)$	▪ Defined through graph convolution ▪ $\mathbf{T}_i\mathbf{f}(n) = \sqrt{N}(f * \delta_i)(n)$ $= \sqrt{N}\sum_{\ell=0}^{N-1} \hat{f}(\lambda_\ell)u_\ell^*(i)u_\ell(n)$
Modulation	▪ Multiplication with the complex exponential ▪ $M_\omega x(t) = e^{j\omega t}x(t)$	▪ Multiplication with the eigenvector of the graph Laplacian ▪ $M_k\mathbf{f}(n) = \sqrt{N}\mathbf{u}_k(n)\mathbf{f}(n)$

Graph Fourier Transform

	Classical Signal Processing	Graph Signal Processing
Fourier Transform	■ $\hat{x}(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt$ ■ Frequency: ω can take any value ■ Fourier basis: Complex exponentials $e^{j\omega t}$	■ $\hat{\mathbf{f}}(\lambda_\ell) = \sum_{n=1}^{N} f(n)u_\ell^*(n)$ ■ Frequency: Eigenvalues of the graph Laplacian (λ_ℓ) ■ Fourier basis: Eigenvectors of the graph Laplacian $(\mathbf{u}_\ell)$

- Graph Fourier Transform
  - Graph Fourier basis are **Eigenfunctions** of the Laplacian matrix (operator)
  - Graph Frequencies: Eigenvalues of the Laplacian matrix $\mathbf{L}$
  - Graph Harmonics: Eigenvectors of the Laplacian matrix $\mathbf{L}$
  - $\boxed{\mathbf{L} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T}$ $\boxed{\mathbf{U} = [\mathbf{u}_0|\mathbf{u}_1|\mathbf{u}_2|\ldots]}$
  - GFT $\boxed{\hat{\mathbf{f}} = \mathbf{U}^\mathbf{T}\mathbf{f}}$, IGFT $\boxed{\mathbf{f} = \mathbf{U}\hat{\mathbf{f}}}$

Graph Signal in Two Domains



A graph signal in vertex domain and spectral domain

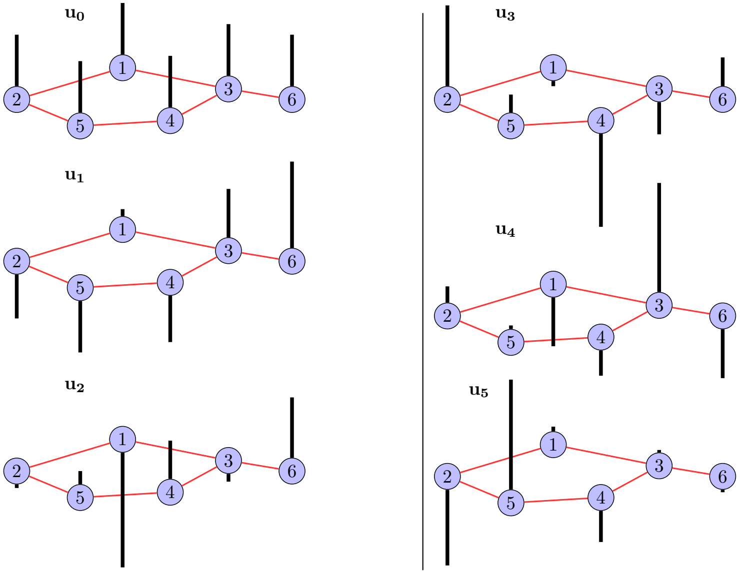
Frequency Ordering

- Use the Laplacian quadratic of $\mathbf{u}$ is $\mathbf{u}^T \mathbf{L} \mathbf{u}$
- $\mathbf{u}_0^T \mathbf{L} \mathbf{u}_0 = ?$
- $\mathbf{u}_1^T \mathbf{L} \mathbf{u}_1 = ?$

Frequency Ordering

- Use the Laplacian quadratic of $\mathbf{u}$ is $\mathbf{u}^T\mathbf{L}\mathbf{u}$
- $\mathbf{u}_0^T\mathbf{L}\mathbf{u}_0 =?$
- $\mathbf{u}_1^T\mathbf{L}\mathbf{u}_1 =?$
- $\mathbf{u}_k^T\mathbf{L}\mathbf{u}_k = \lambda_k$
- Small eigenvalues are low frequencies

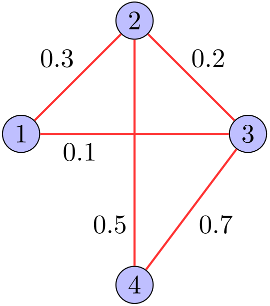
Laplacian Eigenvectors as GFT Basis



.

- Effect of Vertex Indexing on Graph Harmonics and Signal Representation!

Effect of Vertex Indexing



$$\mathbf{W} = \begin{bmatrix} 0 & 0.3 & 0.1 & 0 \\ 0.3 & 0 & 0.2 & 0.5 \\ 0.1 & 0.2 & 0 & 0.7 \\ 0 & 0.5 & 0.7 & 0 \end{bmatrix}$$

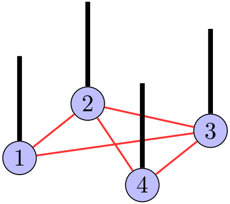
$$\mathbf{L} = \begin{bmatrix} 0.4 & -0.3 & -0.1 & 0 \\ -0.3 & 1 & -0.2 & -0.5 \\ -0.1 & -0.2 & 1 & -0.7 \\ 0 & -0.5 & -0.7 & 1.2 \end{bmatrix}$$

Frequencies λ: 0.0000, 0.4640, 1.2308, 1.9052

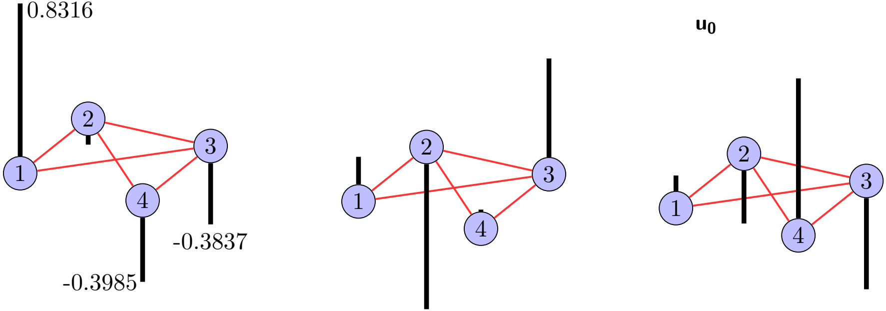
Harmonics
$$\mathbf{U} = \begin{bmatrix} 0.5000 & 0.8316 & 0.2185 & 0.1034 \\ 0.5000 & -0.0494 & -0.7942 & -0.3417 \\ 0.5000 & -0.3837 & 0.5669 & -0.5305 \\ 0.5000 & -0.3985 & 0.0088 & 0.7689 \end{bmatrix}$$

Effect of Vertex Indexing (cont'd...)

$$\mathbf{U} = \begin{bmatrix} 0.5000 & 0.8316 & 0.2185 & 0.1034 \\ 0.5000 & -0.0494 & -0.7942 & -0.3417 \\ 0.5000 & -0.3837 & 0.5669 & -0.5305 \\ 0.5000 & -0.3985 & 0.0088 & 0.7689 \end{bmatrix}$$



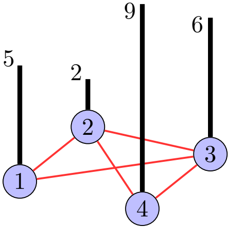
$\mathbf{u_0}$



$\mathbf{u_1}$ $\mathbf{u_2}$ $\mathbf{u_3}$

Effect of Vertex Indexing (cont'd...)

$$\mathbf{U} = \begin{bmatrix} 0.5 & 0.8316 & 0.2185 & 0.1034 \\ 0.5 & -0.0494 & -0.7942 & -0.3417 \\ 0.5 & -0.3837 & 0.5669 & -0.5305 \\ 0.5 & -0.3985 & 0.0088 & 0.7689 \end{bmatrix}$$



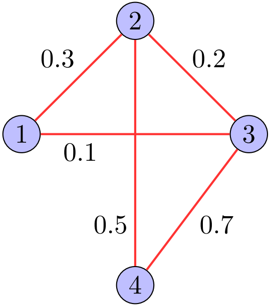
$\mathbf{f_1} = [5\ 2\ 6\ 9]^T$

- Graph signal as linear combination of the Harmonics

$$\mathbf{f_1} = \begin{bmatrix} 5 \\ 2 \\ 6 \\ 9 \end{bmatrix} = (11) \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} + (-1.83) \begin{bmatrix} 0.8316 \\ -0.0494 \\ -0.3837 \\ -0.3985 \end{bmatrix} + (2.98) \begin{bmatrix} 0.2185 \\ -0.7942 \\ 0.5669 \\ 0.0088 \end{bmatrix} + (3.57) \begin{bmatrix} 0.1034 \\ -0.3417 \\ -0.5305 \\ 0.7689 \end{bmatrix}$$

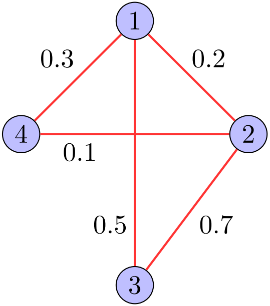
Effect of Vertex Indexing (cont'd...)

Case 1



$$\mathbf{L} = \begin{bmatrix} 0.4 & -0.3 & -0.1 & 0 \\ -0.3 & 1 & -0.2 & -0.5 \\ -0.1 & -0.2 & 1 & -0.7 \\ 0 & -0.5 & -0.7 & 1.2 \end{bmatrix}$$

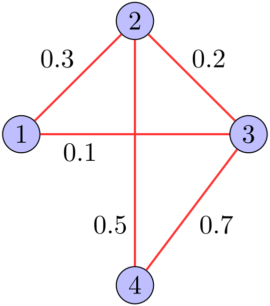
Case 2



$$\mathbf{L} = \begin{bmatrix} 1 & -0.2 & -0.5 & -0.3 \\ -0.2 & 1 & -0.7 & -0.1 \\ -0.5 & -0.7 & 1.2 & 0 \\ -0.3 & -0.1 & 0 & 0.4 \end{bmatrix}$$

Effect of Vertex Indexing (cont'd...)

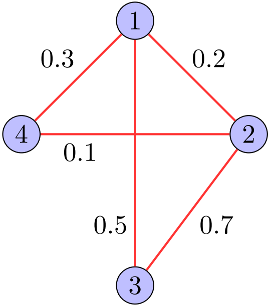
Case 1



λ: 0, 0.4640, 1.2308, 1.9052

$$\mathbf{U} = \begin{bmatrix} 0.5000 & 0.8316 & 0.2185 & 0.1034 \\ 0.5000 & -0.0494 & -0.7942 & -0.3417 \\ 0.5000 & -0.3837 & 0.5669 & -0.5305 \\ 0.5000 & -0.3985 & 0.0088 & 0.7689 \end{bmatrix}$$

Case 2

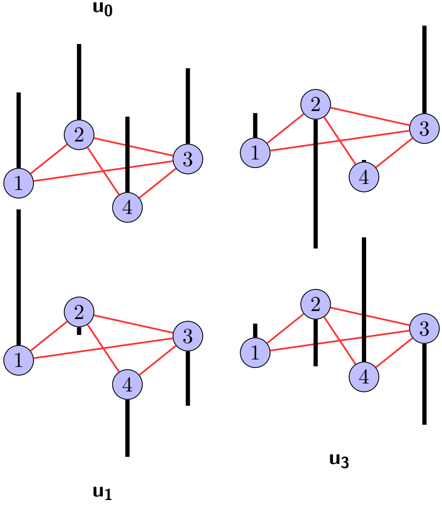


λ: 0, 0.4640, 1.2308, 1.9052

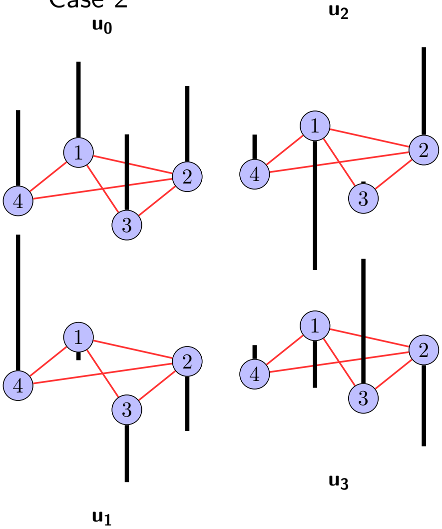
$$\mathbf{U} = \begin{bmatrix} 0.5000 & -0.0494 & -0.7942 & -0.3417 \\ 0.5000 & -0.3837 & 0.5669 & -0.5305 \\ 0.5000 & -0.3985 & 0.0088 & 0.7689 \\ 0.5000 & 0.8316 & 0.2185 & 0.1034 \end{bmatrix}$$

Effect of Vertex Indexing (cont'd. . .)

Case 1

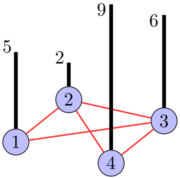


Case 2

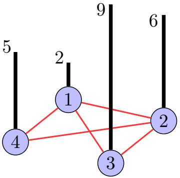


Effect of Vertex Indexing (cont'd...)

$$\mathbf{f_1} = \begin{bmatrix}5\\2\\6\\9\end{bmatrix} = (\mathbf{11})\begin{bmatrix}0.5\\0.5\\0.5\\0.5\end{bmatrix} + (-\mathbf{1.83})\begin{bmatrix}0.8316\\-0.0494\\-0.3837\\-0.3985\end{bmatrix} + (\mathbf{2.98})\begin{bmatrix}0.2185\\-0.7942\\0.5669\\0.0088\end{bmatrix} + (\mathbf{3.57})\begin{bmatrix}0.1034\\-0.3417\\-0.5305\\0.7689\end{bmatrix}$$



$\mathbf{f_1} = [5\ 2\ 6\ 9]^T$

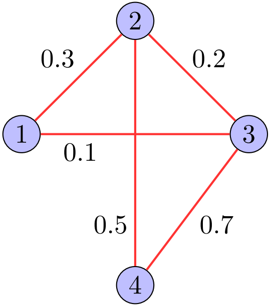


$\mathbf{f_2} = [2\ 6\ 9\ 5]^T$

$$\mathbf{f_2} = \begin{bmatrix}2\\6\\9\\5\end{bmatrix} = (\mathbf{11})\begin{bmatrix}0.5\\0.5\\0.5\\0.5\end{bmatrix} + (-\mathbf{1.83})\begin{bmatrix}-0.0494\\-0.3837\\-0.3985\\0.8316\end{bmatrix} + (\mathbf{2.98})\begin{bmatrix}-0.7942\\0.5669\\0.0088\\0.2185\end{bmatrix} + (\mathbf{3.57})\begin{bmatrix}-0.3417\\-0.5305\\0.7689\\0.1034\end{bmatrix}$$

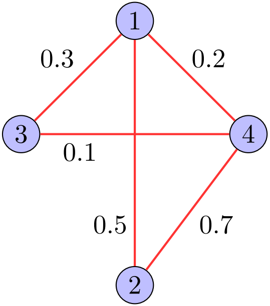
Effect of Vertex Indexing (cont'd...)

Case 1



$$\mathbf{L} = \begin{bmatrix} 0.4 & -0.3 & -0.1 & 0 \\ -0.3 & 1 & -0.2 & -0.5 \\ -0.1 & -0.2 & 1 & -0.7 \\ 0 & -0.5 & -0.7 & 1.2 \end{bmatrix}$$

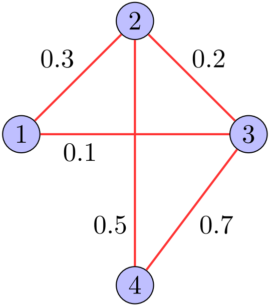
Case 3



$$\mathbf{L} = \begin{bmatrix} 1 & -0.5 & -0.3 & -0.2 \\ -0.5 & 1.2 & 0 & -0.7 \\ -0.3 & 0 & 0.4 & -0.1 \\ -0.2 & -0.7 & -0.1 & 1 \end{bmatrix}$$

Effect of Vertex Indexing (cont'd...)

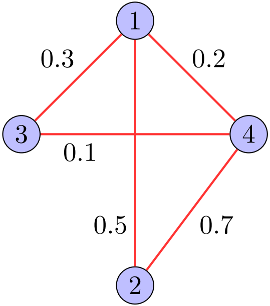
Case 1



λ: 0, 0.4640, 1.2308, 1.9052

$$\mathbf{U} = \begin{bmatrix} 0.5000 & 0.8316 & 0.2185 & 0.1034 \\ 0.5000 & -0.0494 & -0.7942 & -0.3417 \\ 0.5000 & -0.3837 & 0.5669 & -0.5305 \\ 0.5000 & -0.3985 & 0.0088 & 0.7689 \end{bmatrix}$$

Case 3

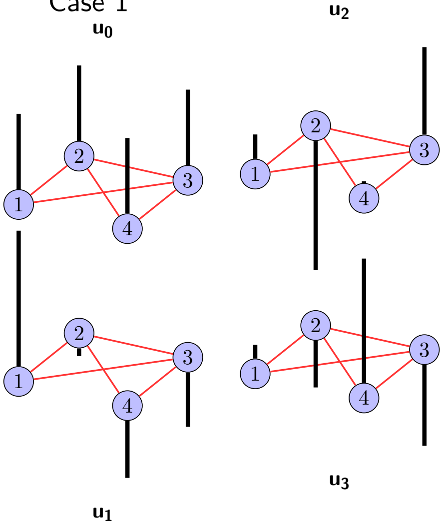


λ: 0, 0.4640, 1.2308, 1.9052

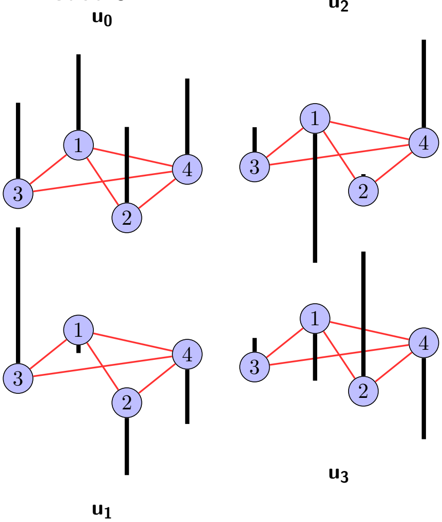
$$\mathbf{U} = \begin{bmatrix} 0.5000 & -0.0494 & -0.7942 & -0.3417 \\ 0.5000 & -0.3985 & 0.0088 & 0.7689 \\ 0.5000 & 0.8316 & 0.2185 & 0.1034 \\ 0.5000 & -0.3837 & 0.5669 & -0.5305 \end{bmatrix}$$

Effect of Vertex Indexing (cont'd. . .)

Case 1

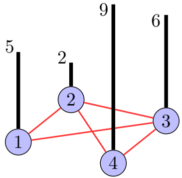


Case 3

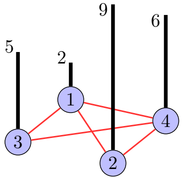


Effect of Vertex Indexing (cont'd. . .)

$$\mathbf{f_1} = \begin{bmatrix}5\\2\\6\\9\end{bmatrix} = (\mathbf{11})\begin{bmatrix}0.5\\0.5\\0.5\\0.5\end{bmatrix} + (-\mathbf{1.83})\begin{bmatrix}0.8316\\-0.0494\\-0.3837\\-0.3985\end{bmatrix} + (\mathbf{2.98})\begin{bmatrix}0.2185\\-0.7942\\0.5669\\0.0088\end{bmatrix} + (\mathbf{3.57})\begin{bmatrix}0.1034\\-0.3417\\-0.5305\\0.7689\end{bmatrix}$$



$\mathbf{f_1} = [5\ 2\ 6\ 9]^T$



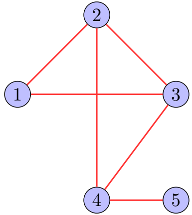
$\mathbf{f_3} = [2\ 9\ 5\ 6]^T$

$$\mathbf{f_3} = \begin{bmatrix}2\\9\\5\\6\end{bmatrix} = (\mathbf{11})\begin{bmatrix}0.5\\0.5\\0.5\\0.5\end{bmatrix} + (-\mathbf{1.83})\begin{bmatrix}-0.0494\\-0.3985\\0.8316\\-0.3837\end{bmatrix} + (\mathbf{2.98})\begin{bmatrix}-0.7942\\0.0088\\0.2185\\0.5669\end{bmatrix} + (\mathbf{3.57})\begin{bmatrix}-0.3417\\0.7689\\0.1034\\-0.5305\end{bmatrix}$$

Effect of Vertex Indexing (cont'd. . .)

- Change in vertex indexing
  - Alters signal representation in vertex domain: signal indexing changes
  - No change in frequency domain representation of the signal (GFT coefficients)

Graph Convolution

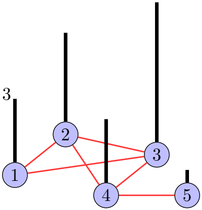


$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

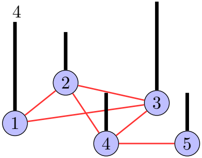
Eigenvalues: 0, 0.8299, 2.6889, 4, 4.4812

$$\mathbf{U} = \begin{bmatrix} 0.4472 & 0.4375 & 0.7031 & 0 & 0.3380 \\ 0.4472 & 0.2560 & -0.2422 & 0.7071 & -0.4193 \\ 0.4472 & 0.2560 & -0.2422 & -0.7071 & -0.4193 \\ 0.4472 & -0.1380 & -0.5362 & 0 & 0.7024 \\ 0.4472 & -0.8115 & 0.3175 & 0 & -0.2018 \end{bmatrix}$$

Graph Convolution(cont'd. . .)

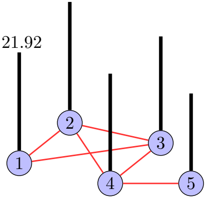


$\mathbf{f} = [3,\ 4,\ 6,\ 3,\ 1]^T$



$\mathbf{g} = [4,\ 2,\ 4,\ 2,\ 2]^T$

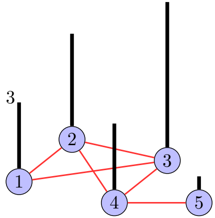
$$\mathbf{h} = \mathbf{f} * \mathbf{g} = \mathrm{IGFT}(\hat{\mathbf{f}}.\hat{\mathbf{g}})$$



$\mathbf{h} = [21.92,\ 23.92,\ 21.08,\ 21.72,\ 17.80]^T$

Building block for graph neural networks (GNNs)

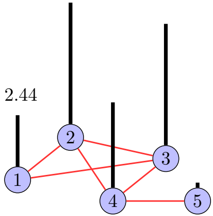
Graph Translation



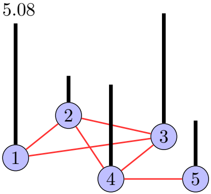
$\mathbf{f} = [3,\ 4,\ 6,\ 3,\ 1]^T$

Translation to node i :

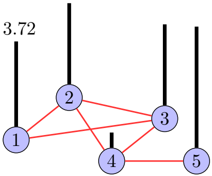
$$T_i(\mathbf{f}) = \sqrt{N}\ (\mathbf{f} * \delta_i) = \sqrt{N}\ \mathrm{IGFT}(\hat{\mathbf{f}}.\mathbf{U}^T(:, i))$$



$T_1\mathbf{f} = [2.44,\ 5.08,\ 5.08,\ 3.72,\ 0.69]^T$



$T_2\mathbf{f} = [5.08,\ 1.50,\ 4.66,\ 3.56,\ 2.21]^T$



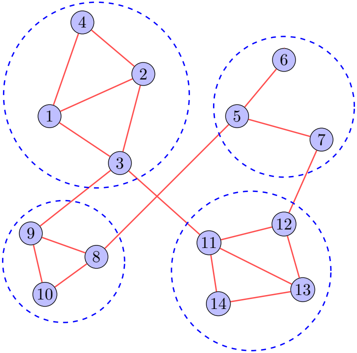
$T_4\mathbf{f} = [3.72,\ 3.56,\ 3.56,\ 1.08,\ 5.08]^T$

Classical vs Graph Signal Processing (Laplacian Based)

Operator/ Transform	Classical Signal Processing	Graph Signal Processing
Fourier Transform	■ $\hat{x}(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt$ ■ Frequency: ω can take any value ■ Fourier basis: Complex exponentials $e^{j\omega t}$	■ $\hat{\mathbf{f}}(\lambda_\ell) = \sum_{n=1}^{N} \mathbf{f}(n) \mathbf{u}_\ell^*(n)$ ■ Frequency: Eigenvalues of the graph Laplacian (λ_ℓ) ■ Fourier basis: Eigenvectors of the graph Laplacian $(\mathbf{u}_\ell)$
Convolution	■ In time domain: $x(t) * y(t) = \int_{-\infty}^{\infty} x(\tau) y(t-\tau) d\tau$ ■ In frequency domain: $\widehat{x(t) * y(t)} = \hat{x}(\omega)\hat{y}(\omega)$	■ Defined through Graph Fourier Transform ■ $\widehat{\mathbf{f} * \mathbf{g}} = \left(\hat{\mathbf{f}}.\hat{\mathbf{g}}\right)$
Translation	■ Can be defined using convolution ■ $T_\tau x(t) = x(t-\tau) = x(t) * \delta_\tau(t)$	■ Defined through graph convolution ■ $\mathbf{T}_i \mathbf{f}(n) = \sqrt{N}(f * \delta_i)(n)$ $= \sqrt{N} \sum_{\ell=0}^{N-1} \hat{f}(\lambda_\ell) u_\ell^*(i) u_\ell(n)$
Modulation	■ Multiplication with the complex exponential ■ $M_\omega x(t) = e^{j\omega t} x(t)$	■ Multiplication with the eigenvector of the graph Laplacian ■ $M_k \mathbf{f}(n) = \sqrt{N} \mathbf{u}_k(n) \mathbf{f}(n)$

Spectral Clustering of Complex Networks

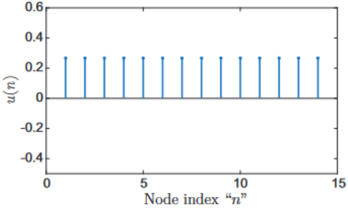
Spectral Clustering of Complex Networks



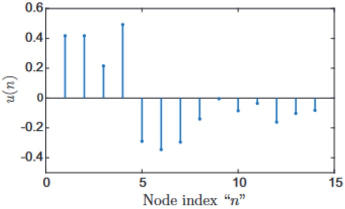
Network with four clusters

- Spectral graph clustering algorithm
- Can be used for community detection
- Eigenvectors of the graph Laplacian for clustering

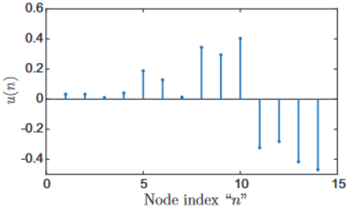
Spectral Clustering of Complex Networks



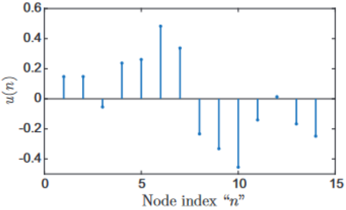
(a) $\mathbf{u}_0$



(b) $\mathbf{u}_1$

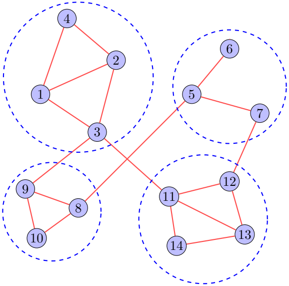


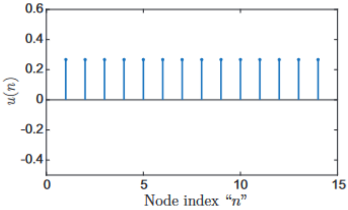
(c) $\mathbf{u}_2$



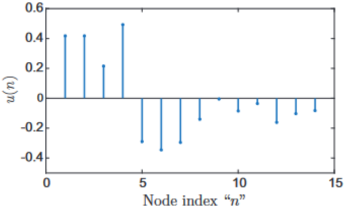
(d) $\mathbf{u}_3$

Spectral Clustering of Complex Networks



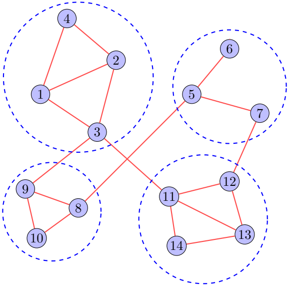


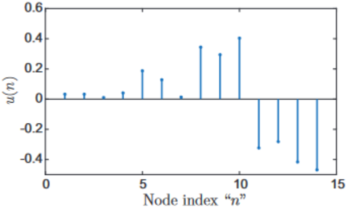
(a) $\mathbf{u}_0$



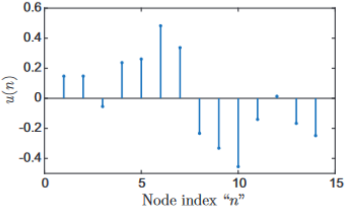
(b) $\mathbf{u}_1$

Spectral Clustering of Complex Networks





(c) $\mathbf{u}_2$



(d) $\mathbf{u}_3$

Spectral Clustering of Complex Networks[1]

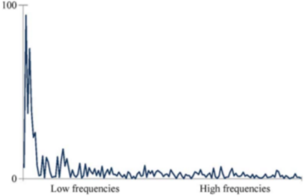
Algorithm 8.1 Algorithm for Spectral Graph Clustering

1: Compute the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$.
2: Compute the first k eigenvectors $\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{k-1}$ of $\mathbf{L}$.
3: Create the matrix $\mathbf{U}_k = [\mathbf{u}_0|\mathbf{u}_1|\ldots\mathbf{u}_{k-1}]$ whose columns are the k eigenvectors of L.
4: Let $\mathbf{z}_i \in \mathbb{R}^k$ be the i^{th} row of $\mathbf{U}_k$.
5: Cluster the N points $\{\mathbf{z}_i\}_{i=1,2,\ldots,N}$ into k clusters with k-means or any other algorithm.
6: Assign the node v_i to cluster j if and only if point $\mathbf{z}_i$ was assigned to cluster j.

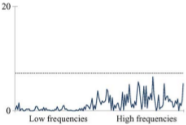
[1] B.S. Manoj, A. Chakraborty, and R. Singh. *Complex Networks: A Networking and Signal Processing Perspective*. Prentice Hall communications engineering and emerging technologies series. Prentice Hall, 2018. ISBN: 9780134786995.

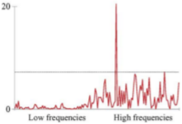
Example Application

- Sensor Malfunction Detection



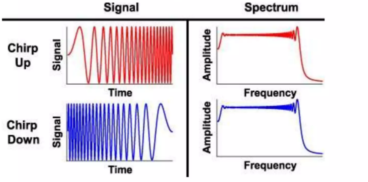
Temperature on a graph and its spectrum





True signal and Corrupted signal after HPF

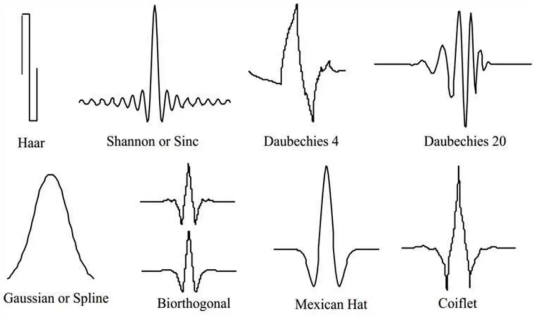
(Graph) Fourier Transform Limitations



- Different in time but same frequency representation!
- (Graph) Fourier Transform only gives "what" frequency components are present
- Cannot tell at what time (where in graph) the frequency components are present
- Simultaneous time frequency representation: **wavelets**

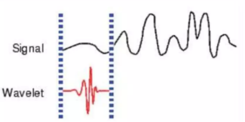
Classical Wavelets

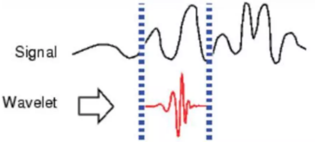
- Wavelet: a small wave
- Ability to provide time-frequency representation simultaneously

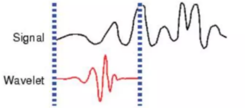


Classical Wavelets Cont'd

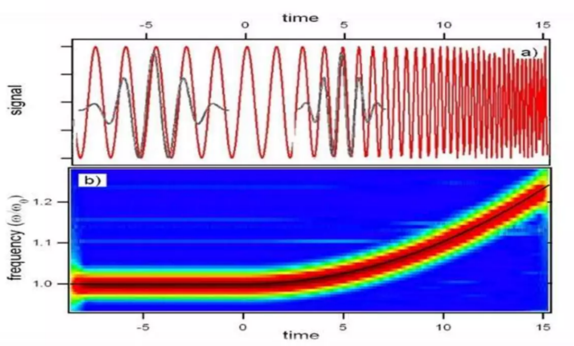
- Wavelets at different "shifts" and "scales"







Classical Wavelets Cont'd



Spectral Graph Wavelet Transform

- Classical wavelets
  - Wavelets at different scales and locations are constructed by scaling and translating a single "mother" wavelet ψ
  - $\boxed{\psi_{s,a}(x) = \frac{1}{s}\psi\left(\frac{x-a}{s}\right)}$
  - Scaling in Fourier domain $\boxed{\psi_{s,a}(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{j\omega x}\hat{\psi}(s\omega)e^{-j\omega a}d\omega}$
  - Scaling ψ by $1/s$ corresponds to scaling $\hat{\psi}$ with s
  - Term $e^{-j\omega a}$ comes from localization of the wavelet at location a
- Spectral Graph Wavelets
  - Graph wavelet at scale t and centered at node n

$$\psi_{t,n}(m) = \sum_{\ell=0}^{N-1} u_\ell(m)g(t\lambda_\ell)u_\ell^*(n)$$

  - Frequency ω is replaced with eigenvalues of graph Laplacian λ_ℓ
  - Translating to node n corresponds to multiplication by $u_\ell^*(n)$
  - g acts as a scaled bandpass filter, replacing $\hat{\psi}$

Matrix Form of SGWT[2]

- Wavelet basis at scale t = collection of N number of wavelets (each wavelet centered at a particular node of the graph)

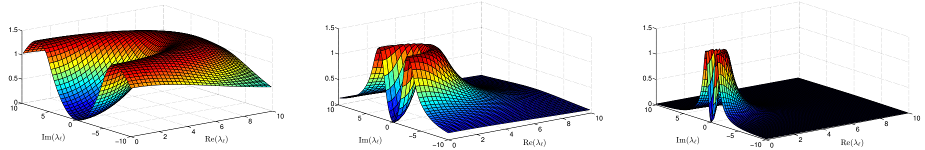
$$\boldsymbol{\Psi}_t = [\psi_{t,1}|\psi_{t,2}| \ \dots \ |\psi_{t,N}] = \mathbf{U}\mathbf{G_t}\mathbf{U}^\mathbf{T}$$

- Wavelet coefficient at scale t and centered at node n of a graph signal f

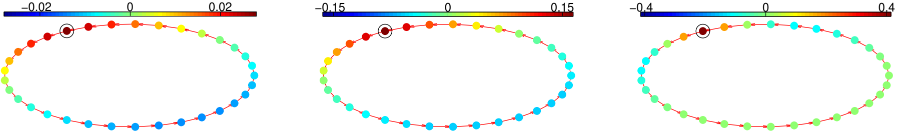
$$W_f(t,n) = \langle \psi_{t,n}, \mathbf{f} \rangle = \psi_{t,n}^T \mathbf{f}$$

[2] B.S. Manoj, A. Chakraborty, and R. Singh. *Complex Networks: A Networking and Signal Processing Perspective*. Prentice Hall communications engineering and emerging technologies series. Prentice Hall, 2018. ISBN: 9780134786995.

SGWT Example



Kernel at scale $t = 0.2$ Kernel at scale $t = 0.5429$ Kernel at scale $t = 1.4736$



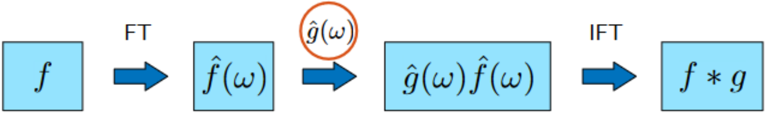
Wavelet at scale $t = 19.8244$ Wavelet at scale $t = 7.3034$ Wavelet at scale $t = 2.6906$

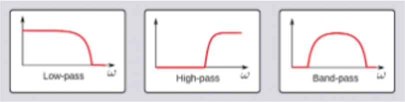
$$\mathbf{\Psi}_t = [\psi_{t,1}|\psi_{t,2}| \ \dots \ |\psi_{t,N}] = \mathbf{V} \begin{bmatrix} g(t\lambda_0) & & & \\ & g(t\lambda_1) & & \\ & & \ddots & \\ & & & g(t\lambda_{N-1}) \end{bmatrix} \mathbf{V}^{-1} = \mathbf{V}\mathbf{G_t}\mathbf{V}^{-1}$$

Spectral Filtering of Graph Signals

Classical vs Graph Spectral Filtering

	Classical Signal Processing	**Graph Signal Processing**
Fourier Transform	• $\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$ • Frequency: ω can take any value • Fourier basis: Complex exponentials $e^{j\omega t}$	• $\hat{\mathbf{f}}(\lambda_\ell) = \sum_{n=1}^{N} f(n)u_\ell^*(n)$ • Frequency: Eigenvalues of the graph Laplacian (λ_ℓ) • Fourier basis: Eigenvectors of the graph Laplacian $(\mathbf{u}_\ell)$





- Replace ω by λ_ℓ in graph spectral filtering

Different GSP Frameworks

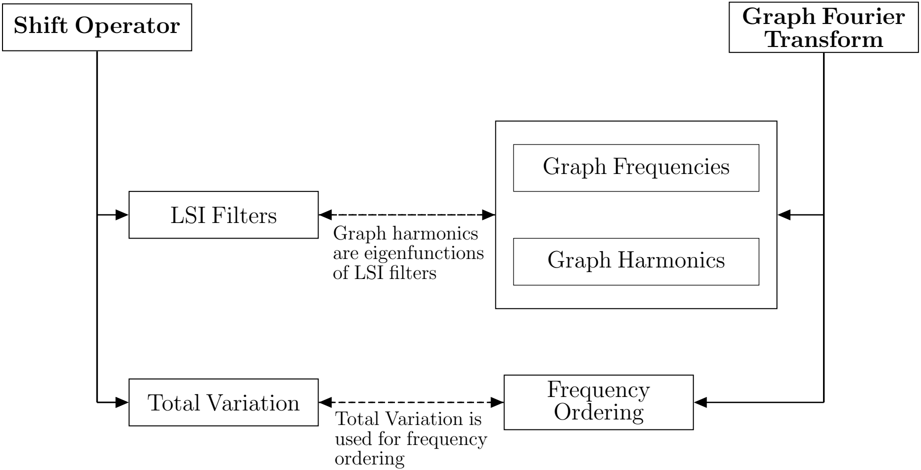
Existing Graph Signal Processing (GSP) Frameworks

- Discrete Signal Processing on Graphs (DSP_G) framework

	GSP based on Laplacian	**DSP_G Framework**	
		Based on Weight Matrix	**Based on Directed Laplacian**
Shift Operator	Not defined	The weight matrix $\mathbf{W}$	Derived from directed Laplacian $(\mathbf{I} - \mathbf{L})$
LSI Filters	Not applicable	Applicable	Applicable
Applicability	Only undirected graphs	Directed graphs	Directed graphs
Frequencies	Eigenvalues of the Laplacian (real)	Eigenvalues of the weight matrix	Eigenvalues of the directed Laplacian
Harmonics	Eigenvectors of the Laplacian matrix (real)	Eigenvectors of the weight matrix	Eigenvectors of the directed Laplacian
Frequency Ordering	Laplacian quadratic form (natural)	Total variation (not natural)	Total variation (natural)

DSP_G Framework

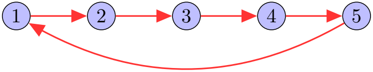
Discrete Signal Processing on Graphs (DSP$_G$) Framework



Concepts in the DSP$_G$ framework

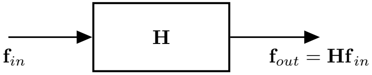
DSP$_G$ Framework: Weight Matrix

- Shift operator
  - Weight matrix $\mathbf{W}$ of the graph
- Shifted graph signal $\boxed{\tilde{\mathbf{f}} = \mathbf{W}\mathbf{f}}$
- Example: shifting discrete-time signal (one unit right)



$$\mathbf{x} = [9,\ 7,\ 5,\ 0,\ 6]^T$$

$$\tilde{\mathbf{x}} = \mathbf{W}\mathbf{x} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 9 \\ 7 \\ 5 \\ 0 \\ 6 \end{bmatrix} = \begin{bmatrix} 6 \\ 9 \\ 7 \\ 5 \\ 0 \end{bmatrix}$$



A linear graph filter

- Linear Shift Invariant (LSI) filters
  - $\mathbf{H}(\mathbf{W}\mathbf{f}_{in}) = \mathbf{W}(\mathbf{H}\mathbf{f}_{in})$
  - Polynomials in $\mathbf{W}$

$$\mathbf{H} = h(\mathbf{W}) = \sum_{m=0}^{M-1} h_m \mathbf{W}^m$$
$$= h_0\mathbf{I} + h_1\mathbf{W} + \ldots + h_{M-1}\mathbf{W}^{M-1}$$

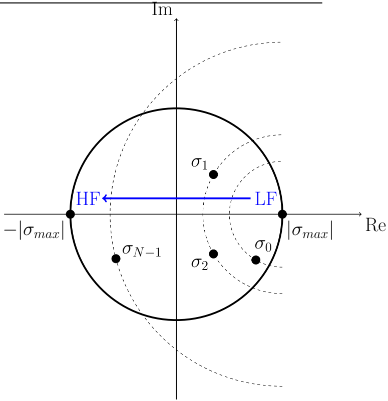
DSP$_{\mathrm{G}}$ Framework: Weight Matrix

- Analogy from classical signal processing
  - Classical Fourier basis: Complex exponentials
  - Complex exponentials are **Eigenfunctions** of Linear Time Invariant (LTI) filters
- Graph Fourier Transform
  - Graph Fourier basis are **Eigenfunctions** of Linear Shift Invariant (LSI) graph filters
  - Graph Frequencies: Eigenvalues of the weight matrix $\mathbf{W}$
  - Graph Harmonics: Eigenvectors of the weight matrix $\mathbf{W}$
  - $\boxed{\mathbf{W} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^{-1}}$
  - GFT $\boxed{\hat{\mathbf{f}} = \mathbf{V}^{-1}\mathbf{f}}$, IGFT $\boxed{\mathbf{f} = \mathbf{V}^{-1}\hat{\mathbf{f}}}$

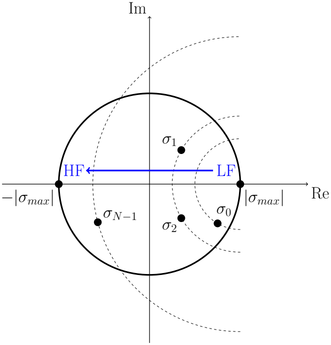
DSP$_{\mathrm{G}}$ Framework: Weight Matrix

- Total Variation in classical signal processing
  - $\boxed{\mathrm{TV}(\mathbf{x}) = \sum_n x[n] - x[n-1] = ||\mathbf{x} - \tilde{\mathbf{x}}||_1}$, where $\tilde{x}[n] = x[n-1]$
- Analogy from classical signal processing
- Total Variation on graphs $\boxed{\mathrm{TV}_{\mathcal{G}}(\mathbf{f}) = ||\mathbf{f} - \tilde{\mathbf{f}}||_1 = ||\mathbf{f} - \mathbf{W}\mathbf{f}||_1}$

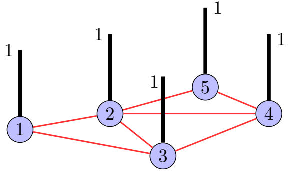
- Frequency ordering: Based on Total Variation
- Eigenvalue with largest magnitude: Lowest frquency



Problems in Weight Matrix based DSP$_G$



- Constant graph signal



$$\hat{\mathbf{f}} = \begin{bmatrix} 0 \\ \boxed{0.36} \\ \boxed{0.16} \\ 0 \\ 2.20 \end{bmatrix}$$

Graph frequencies:
-1.62, -1.47, -0.46, 0.62, 2.94

- Weight matrix based DSP$_G$
  - Does not provide "natural" frequency ordering
  - Even a constant signal has high frequency components

Graph Fourier Transform based on Directed Laplacian

Graph Fourier Transform based on Directed Laplacian[3]

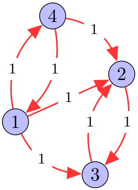
- Redefines Graph Fourier Transform under DSP$_{\mathrm{G}}$
  - Shift operator: Derived from directed Laplacian
  - Linear Shift Invariant filters: Polynomials in the directed Laplacian
  - Graph frequencies: Eigenvalues of the directed Laplacian
  - Graph harmonics: Eigenvectors of the directed Laplacian
- "Natural" frequency ordering
- Better intuition of frequency as compared to the weight matrix based approach
- Coincides with the Laplacian based approach for undirected graphs

[3] Rahul Singh, Abhishek Chakraborty, and BS Manoj. "Graph Fourier transform based on directed Laplacian". In: *2016 International Conference on Signal Processing and Communications (SPCOM)*. IEEE. 2016, pp. 1–5.

Directed Laplacian Matrix

- Basic matrices of a directed graph
  - Weight matrix: $\mathbf{W}$
    - w_{ij} is the weight of the directed edge from node j to node i
  - In-degree matrix: $\mathbf{D}_{\text{in}} = \text{diag}(\{d_i^{in}\}_{i=1,2,\dots,N})$, $\quad d_i^{in} = \sum_{j=1}^{N} w_{ij}$
  - Out-degree matrix: $\mathbf{D}_{\text{out}} = \text{diag}(\{d_i^{out}\}_{i=1,2,\dots,N})$, $\quad d_i^{out} = \sum_{i=1}^{N} w_{ij}$

- Directed Laplacian matrix $\boxed{\mathbf{L} = \mathbf{D}_{\text{in}} - \mathbf{W}}$
  - Sum of each row is zero
  - $\lambda = 0$ is surely an eigenvalue



A directed graph

$$\mathbf{W} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Weight matrix

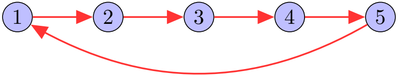
$$\mathbf{D}_{\text{in}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In-degree matrix

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}$$

Directed Laplacian matrix

Shift Operator



A directed cyclic (ring) graph

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Laplacian matrix

- A signal $\mathbf{x} = [9\ 7\ 1\ 0\ 6]^T$; shifted by one unit to the right $\tilde{\mathbf{x}} = [6\ 9\ 7\ 1\ 0]^T$

$$\tilde{\mathbf{x}} = \mathrm{S}\mathbf{x} = (\mathrm{I} - \mathbf{L})\mathbf{x} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 9 \\ 7 \\ 1 \\ 0 \\ 6 \end{bmatrix} = \begin{bmatrix} 6 \\ 9 \\ 7 \\ 1 \\ 0 \end{bmatrix}$$

- $\boxed{\mathrm{S} = (\mathrm{I} - \mathbf{L})}$ is the **shift operator**

- Shifted graph signal: $\boxed{\tilde{\mathbf{f}} = \mathrm{S}\mathbf{f} = (\mathrm{I} - \mathbf{L})\mathbf{f}}$

LSI Filters

$$\mathbf{f}_{in} \longrightarrow \boxed{\mathbf{H}} \longrightarrow \mathbf{f}_{out} = \mathbf{H}\mathbf{f}_{in}$$

A linear graph filter

- Linear Shift-Invariant (LSI) filter: $\boxed{\mathbf{S}(\mathbf{f}_{out}) = \mathbf{H}(\mathbf{S}\mathbf{f}_{in})}$

Theorem

A graph filter $\mathbf{H}$ *is LSI if the following conditions are satisfied.*

1. *Geometric multiplicity of each distinct eigenvalue of the graph Laplacian is one.*
2. *The graph filter* $\mathbf{H}$ *is a polynomial in* $\mathbf{L}$*, i.e., if* $\mathbf{H}$ *can be written as*

$$\mathbf{H} = h(\mathbf{L}) = h_0\mathbf{I} + h_1\mathbf{L} + \ldots + h_m\mathbf{L}^m$$

where, $h_0, h_1, \ \ldots \ , h_m \in \mathbb{C}$ *are called filter taps.*

Graph Fourier Transform based on Directed Laplacian

- Jordan decomposition of the directed Laplacian: $\boxed{\mathbf{L} = \mathbf{V}\mathbf{J}\mathbf{V}^{-1}}$
- **Graph Fourier basis**: Columns of $\mathbf{V}$ (Jordan Eigenvectors of $\mathbf{L}$)
- **Graph frequencies**: Eigenvalues of $\mathbf{L}$ (diagonal entries of Jordan blocks in $\mathbf{J}$)
- GFT $\boxed{\hat{\mathbf{f}} = \mathbf{V}^{-1}\mathbf{f}}$ and IGFT: $\boxed{\mathbf{f} = \mathbf{V}\hat{\mathbf{f}}}$
- **Frequency Ordering**: based on Total Variation

- Total Variation: $\mathrm{TV}_{\mathcal{G}}(\mathbf{f}) = ||\mathbf{f} - \mathbf{S}\mathbf{f}||_1 = ||\mathbf{f} - (\mathbf{I} - \mathbf{L})\mathbf{f}||_1$

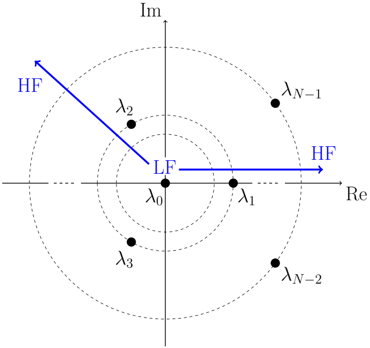
$$\boxed{\mathrm{TV}_{\mathcal{G}}(\mathbf{f}) = ||\mathbf{L}\mathbf{f}||_1}$$

Theorem

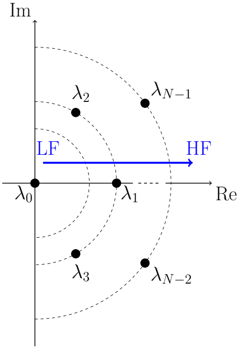
TV of an eigenvector $\mathbf{v}_r$ is proportional to the absolute value of the corresponding eigenvalue

$$\mathrm{TV}(\mathbf{v}_r) \propto |\lambda_r|$$

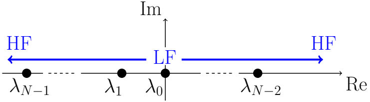
Frequency Ordering



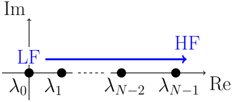
Arbitrary graph



Graph with positive edge weights



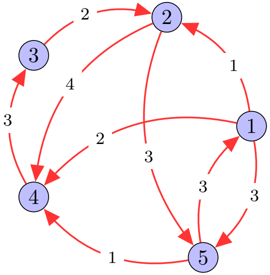
Undirected graph with real edge weights.



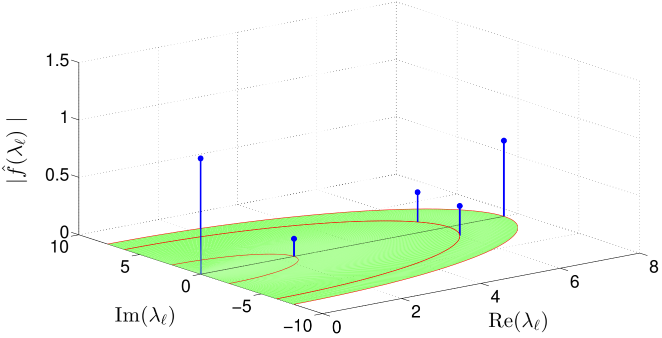
Undirected graph with real and non-negative edge weights.

Example

- Graph signal $\mathbf{f} = [0.1189\ 0.3801\ 0.8128\ 0.2441\ 0.8844]^T$ defined on the directed graph



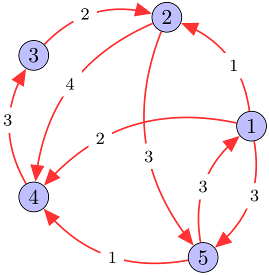
A weighted directed graph



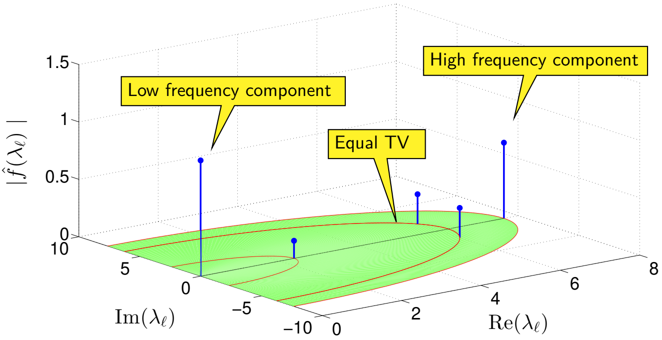
Spectrum of the signal $\mathbf{f} = [0.1189\ 0.3801\ 0.8128\ 0.2441\ 0.8844]^T$

Example

- Graph signal $\mathbf{f} = [0.1189\ 0.3801\ 0.8128\ 0.2441\ 0.8844]^T$ defined on the directed graph



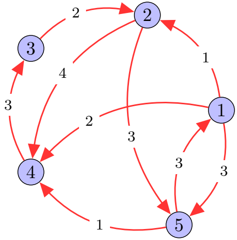
A weighted directed graph



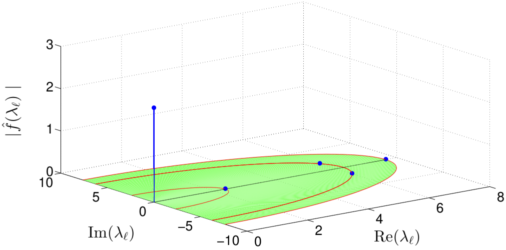
Spectrum of the signal $\mathbf{f} = [0.1189\ 0.3801\ 0.8128\ 0.2441\ 0.8844]^T$

Example: Zero Frequency

- Eigenvector corresponding to λ_0 is $\mathbf{v}_0 = \frac{1}{\sqrt{N}}[1,\ 1, \dots\ 1]^T$
  - TV of $\mathbf{v}_0$ is zero
- For a constant graph signal $\mathbf{f} = [k,\ k, \dots]^T$, GFT is $\hat{\mathbf{f}} = [(k\sqrt{N}),\ 0, \dots]^T$
  - Only zero frequency component



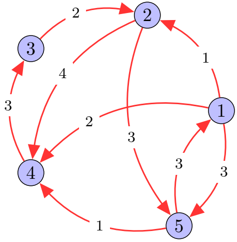
A weighted directed graph



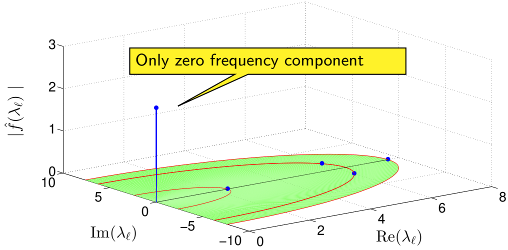
Spectrum of the constant signal $\mathbf{f} = [1\ 1\ 1\ 1\ 1]^T$

Example: Zero Frequency

- Eigenvector corresponding to λ_0 is $\mathbf{v}_0 = \frac{1}{\sqrt{N}}[1,\ 1, \dots\ 1]^T$
  - TV of $\mathbf{v}_0$ is zero
- For a constant graph signal $\mathbf{f} = [k,\ k, \dots]^T$, GFT is $\hat{\mathbf{f}} = [(k\sqrt{N}),\ 0, \dots]^T$
  - Only zero frequency component
- The **weight matrix based approach** of GFT **fails** to give this basic intuition



A weighted directed graph



Spectrum of the constant signal $\mathbf{f} = [1\ 1\ 1\ 1\ 1]^T$

Comparison of the GSP Frameworks

	GSP based on Laplacian	DSP_G Framework	
		Based on Weight Matrix	Based on Directed Laplacian
Shift Operator	Not defined	The weight matrix $\mathbf{W}$	Derived from directed Laplacian $(\mathbf{I}-\mathbf{L})$
LSI Filters	Not applicable	Applicable	Applicable
Applicability	Only undirected graphs	Directed graphs	Directed graphs
Frequencies	Eigenvalues of the Laplacian (real)	Eigenvalues of the weight matrix	Eigenvalues of the directed Laplacian
Harmonics	Eigenvectors of the Laplacian matrix (real)	Eigenvectors of the weight matrix	Eigenvectors of the directed Laplacian
Frequency Ordering	Laplacian quadratic form (natural)	Total variation (not natural)	Total variation (natural)

Filtering in Spectral Domain

- $\mathbf{x} \in \mathbb{R}^N$ be a single-channel input signal on the graph
- Graph convolution of the input graph signal $\mathbf{x}$ with a filter $\mathbf{g}$ is

$$\mathbf{x} * \mathbf{g} := \mathbf{U}\left((\mathbf{U}^T\mathbf{x}) \odot (\mathbf{U}^T\mathbf{g})\right) = \mathbf{U}\hat{\mathbf{G}}\mathbf{U}^T\mathbf{x},$$

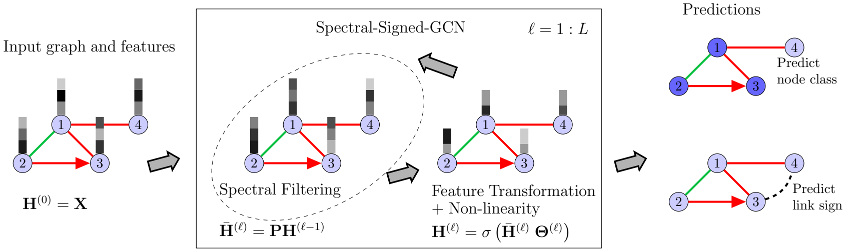
$$\hat{\mathbf{G}} := \mathrm{diag}(\hat{\mathbf{g}}) = \mathrm{diag}\{\hat{g}_1, \ldots, \hat{g}_N\}$$

- Spectral-GNN[4] learn all the filter coefficients (expensive)
- Approximate via K^{th} order polynomials of the graph frequencies ($K << N$)
  - $\hat{g}(\lambda_j) = \sum_{i=0}^{K} \theta_i \lambda_j^i, \quad \boldsymbol{\theta} \in \mathbb{R}^{K+1}$ are filter coefficients

$$\mathbf{x} * \mathbf{g} \approx \mathbf{U}\left(\sum_{i=0}^{K} \theta_i \boldsymbol{\Lambda}^i\right)\mathbf{U}^T\mathbf{x} = \sum_{i=0}^{K} \theta_i \mathbf{L}_{\mathrm{n}}^i \mathbf{x}.$$

[4] Joan Bruna et al. "Spectral networks and deep locally connected networks on graphs". In: *2nd International Conference on Learning Representations, ICLR*. 2014.

Spectral Graph Neural Networks[5]



The spectral GNN filters and transforms the features repeatedly throughout L layers and then applies a linear prediction.

[5] Rahul Singh and Yongxin Chen. "Signed Graph Neural Networks: A Frequency Perspective". In: *Transactions on Machine Learning Research* (2023). ISSN: 2835-8856. URL: https://openreview.net/forum?id=RZveYHgZbu.

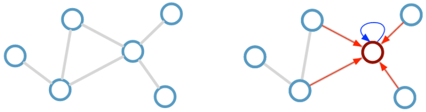
GCN[6]

- First order polynomial filter ($K = 1$) with $\theta_0 = 2\theta$ and $\theta_1 = -\theta$

$$\mathbf{x} * \mathbf{g} \approx \theta \left(2\mathbf{I} - \mathbf{L}_{\mathrm{n}}\right) \mathbf{x} = \theta \left(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right) \mathbf{x}$$

- With self-loop $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$

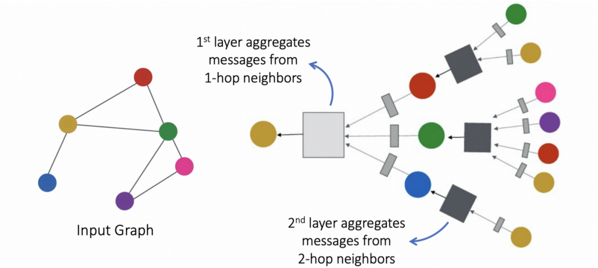
$$\mathbf{x} * \mathbf{g} \approx \theta \left(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\right) \mathbf{x}$$



Update rule: $$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i} \frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

[6] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations (ICLR)*. 2017.

Graph Neural Networks



- GNNs learn latent node representations via
  - Feature Aggregation and Feature Transformation
- Vanilla GCN: ℓ^{th} layer reads

$$\mathbf{H}^{(\ell)} = \sigma\left(\mathbf{P}\ \mathbf{H}^{(\ell-1)}\ \mathbf{\Theta}^{(\ell)}\right)$$

  - $\mathbf{H}^{(0)} = \mathbf{X}$, $\mathbf{P} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ is the low-pass feature aggregation filter, $\mathbf{\Theta}^{(\ell)}$ is a learnable transformation matrix

Takeaways

- Introduction to Graph Signal Processing (GSP)
- Graph Fourier Transform
  - Laplacian Eigenvalues as graph frequencies
  - Laplacian eigenvectors as graph Fourier basis
- Graph wavelets

References

- Review Papers[7][8][9]
- DSP_G framework[10]

[7] David I Shuman et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *Signal Processing Magazine, IEEE* 30.3 (2013), pp. 83–98.

[8] Antonio Ortega et al. "Graph signal processing: Overview, challenges, and applications". In: *Proceedings of the IEEE* 106.5 (2018), pp. 808–828.

[9] Geert Leus et al. "Graph Signal Processing: History, development, impact, and outlook". In: *IEEE Signal Processing Magazine* 40.4 (2023), pp. 49–60.

[10] B.S. Manoj, A. Chakraborty, and R. Singh. *Complex Networks: A Networking and Signal Processing Perspective*. Prentice Hall communications engineering and emerging technologies series. Prentice Hall, 2018. ISBN: 9780134786995, Aliaksei Sandryhaila and José MF Moura. "Discrete signal processing on graphs". In: *IEEE transactions on signal processing* 61.7 (2013), pp. 1644–1656.

Thank You.

r.singh@yale.edu