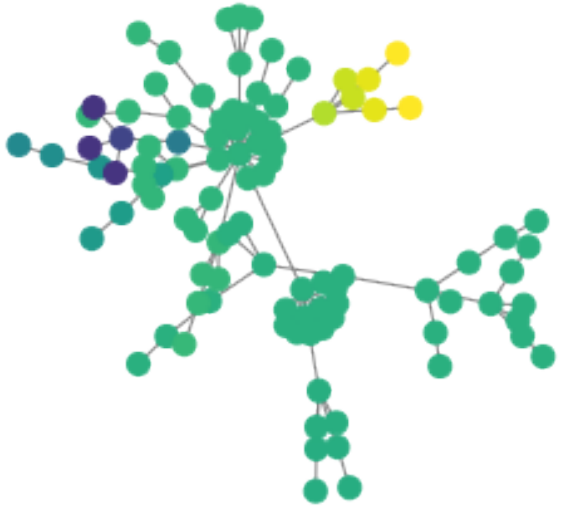


Regression on graphs with graph signal processing

Andreas Loukas

Signal Processing Laboratory (LTS2), EPFL

EPFL

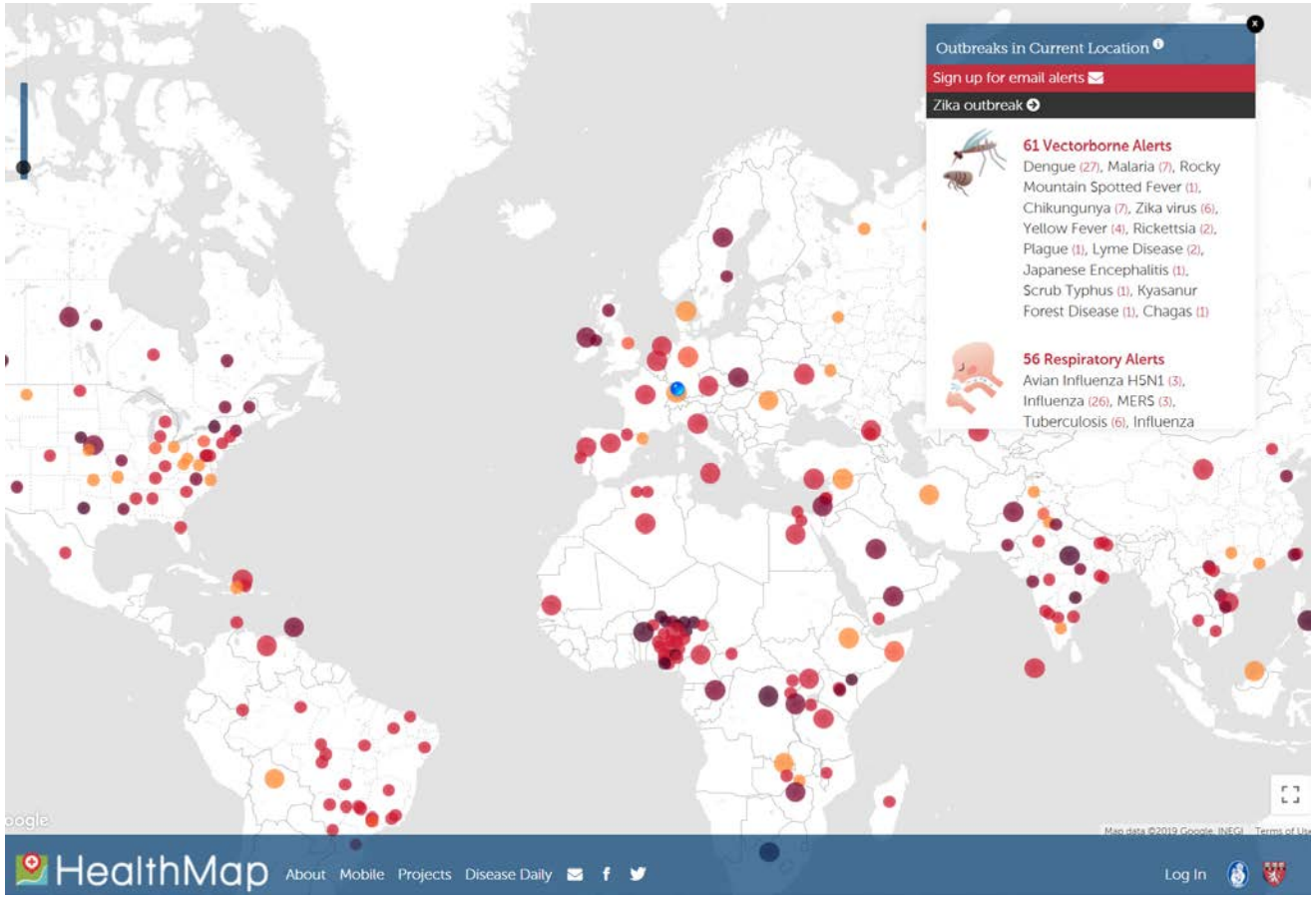
Graph data - beyond the graph structure

Road network

The actual data

Graph data - beyond the graph structure

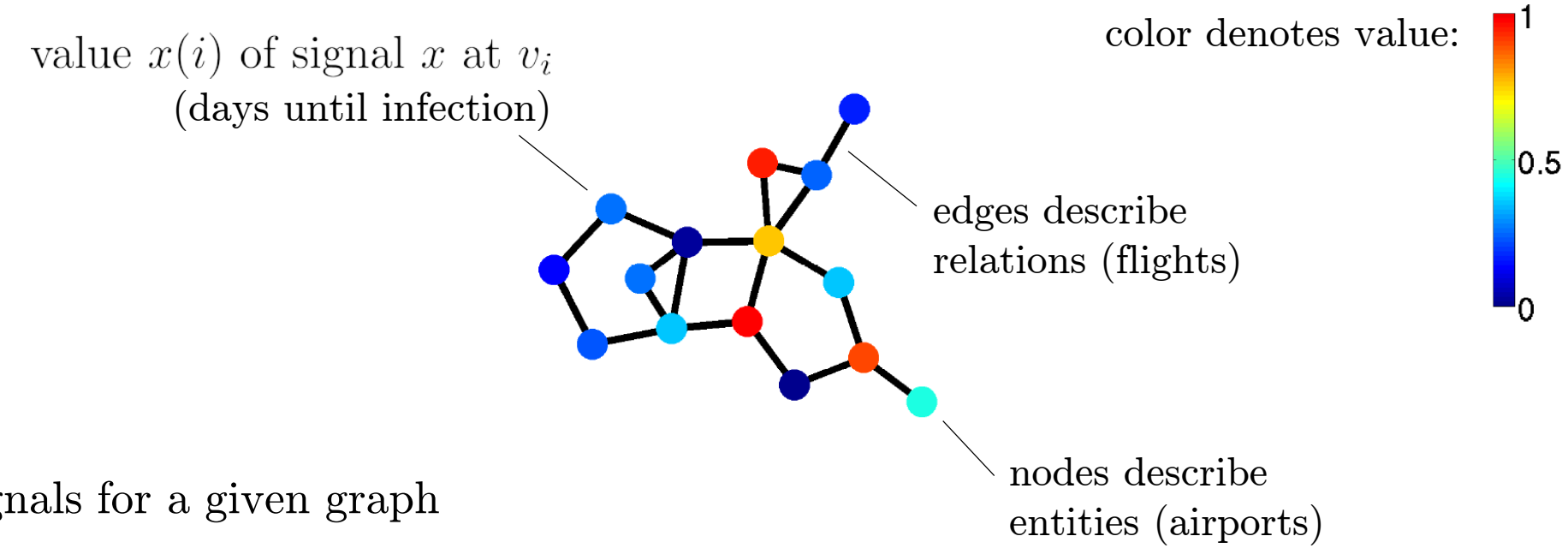
Flight transportation network



Disease outbreaks in the last week – HealthMap

Graph signal

In addition to graph $G = (V, E, W)$, we are given a vector $x \in \mathbb{R}^N$.



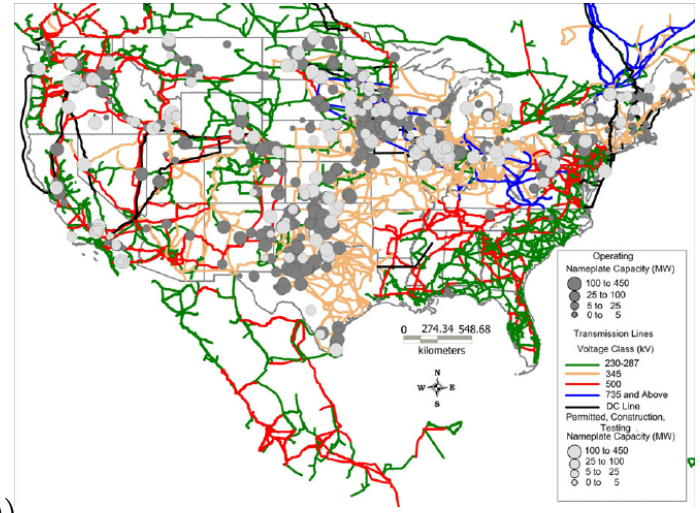
- Many signals for a given graph
- Discrete signals can be used to encode labels

Other examples

Wikipedia network (Yin et al. 2017)

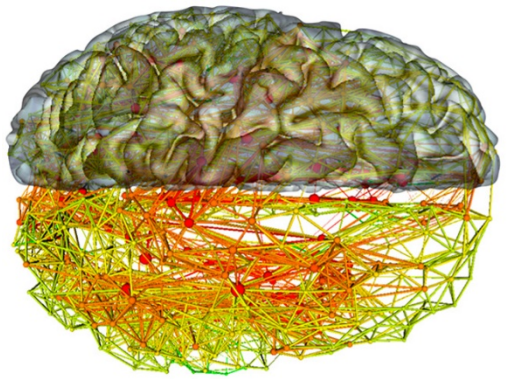
Power-grid (Stephens. et al. 2013)

Transmission organizations & lines.

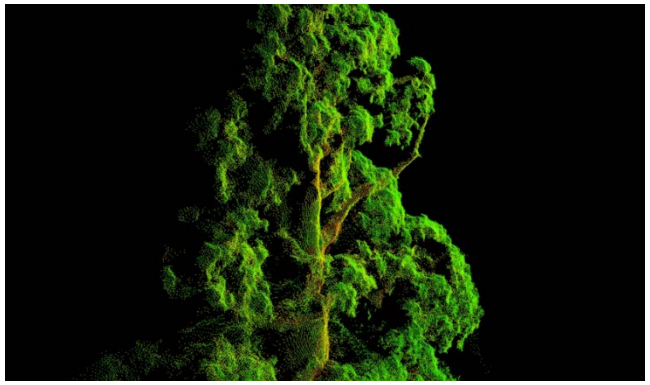


Connectome (Parisot et al. 2017)

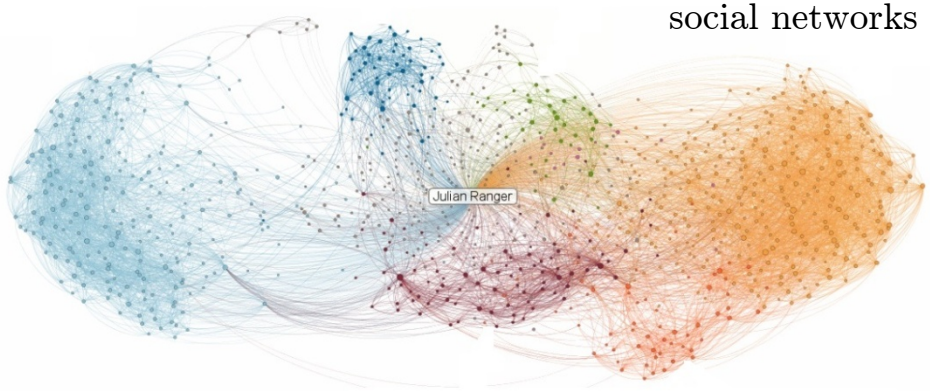
Image by Martijn van den Heuvel



Point clouds (Pix4D)

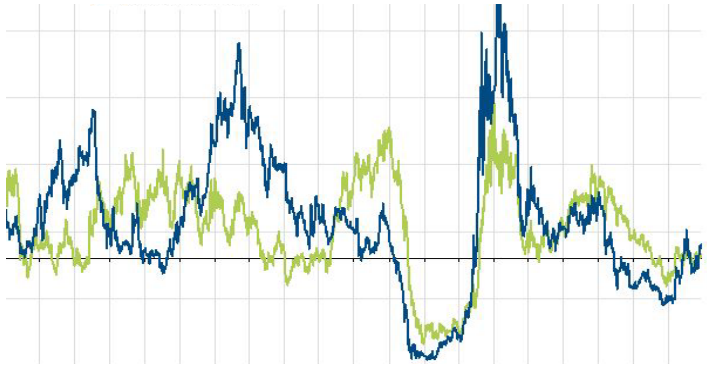


social networks

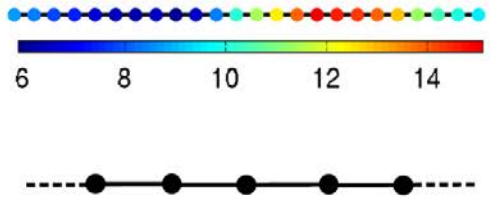


Interpretation

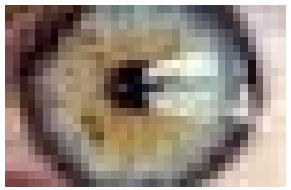
Graph captures the geometric structure of the domain where data live.

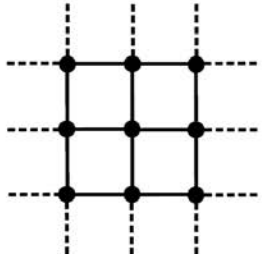


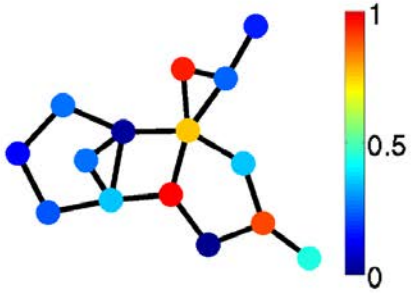
oil prices

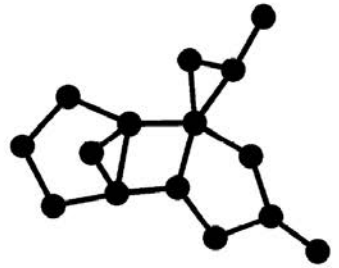


Afghan girl, Steve McCurry





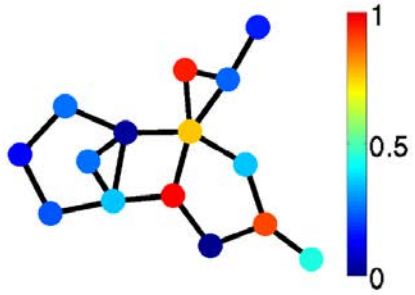




Outline

- Regression 101: Tikhonov and TV
- Graph Fourier Transform
- Graph filters

Linear inverse problems (on graphs)



Consider a graph signal $x \in \mathbb{R}^N$.

We observe y such that:

$$y = Ax + w,$$

with A an $M \times N$ observation matrix (usually $M < N$) and $w \in \mathbb{R}^M$ is the noise vector (usually white).

Objective: recover x (given y and A).

Examples:

denoising

$$A = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \\ 0 & 0 & \cdots & 1 \end{bmatrix} \quad \text{and} \quad w = N(0, I)$$

interpolation

$$A = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \end{bmatrix} \quad \text{and} \quad w = 0$$

Regularization

Key observation:

- In general, we cannot recover x, unless we know something about the structure of x.

Pose as a regression problem

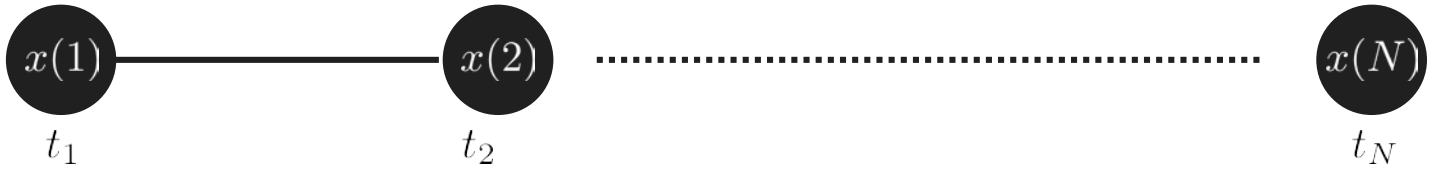
$$\tilde{x} = \arg\min_{x \in \mathcal{D}} \|y - Ax\|_2^2 + R(x)$$

Regularizer encodes prior knowledge about data and helps to distinguish between good and bad solutions:

- good solutions have small $R(x)$
- bad solutions have large $R(x)$

Tikhonov regularization

Suppose that $x(1), x(2), \ldots, x(N)$ is a regularly sampled timeseries corresponding to $t_1, t_2, \ldots, t_N$

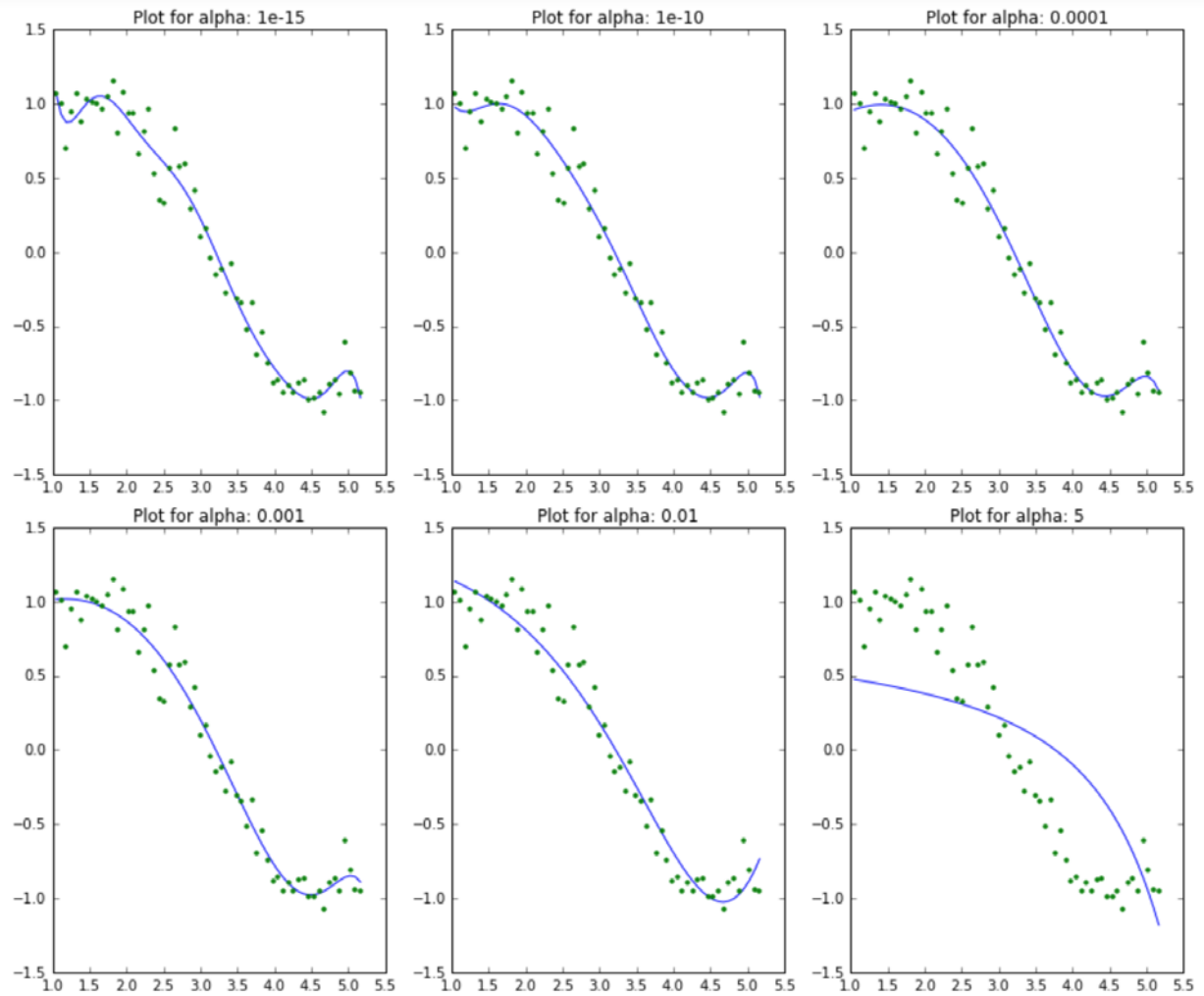


Tikhonov regularization (also referred to as ridge regression):

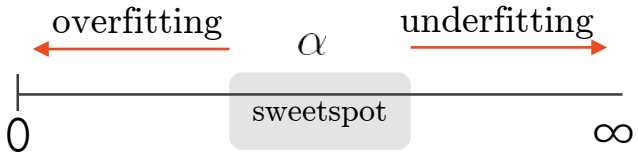
$$R_{\text{tk}}(x) = \alpha \sum_{t=1}^{n} (x(t+1) - x(t))^2$$

- penalizes solution that change rapidly (i.e., it is a smoothness prior)
- α is the strength of the regularization.

Illustration of Tikhonov regularization



α controls model complexity:



How to select α?

Comparison by Aarshay Jain at https://www.analyticsvidhya.com

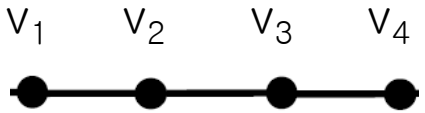
Graph Tikhonov regularization

Notice that if $G = (V, E)$ is a path graph of n nodes, then:

$$\sum_{t=1}^{n} (x(t+1) - x(t))^2 = \|Sx\|_2^2,$$

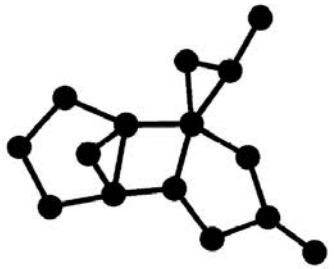
where $S \in |E| \times |V|$ is the incidence matrix (or gradient):

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



Direct extension for general graph $G = (V, E)$

$$\tilde{x} = \arg\min_{x \in \mathbb{R}^N} \|Ax - y\|_2^2 + R_{\text{tk}}(x; G),$$



- $R_{\text{tk}}(x; G) = \alpha \|Sx\|_2^2 = \alpha x^\top S^\top S x = \alpha x^\top L x = \alpha \sum_{w_{ij} \in E} w_{ij} (x(i) - x(j))^2$

Solution

To compute the solution of the convex problem:

$$\tilde{x} = \arg\min_{x \in \mathbb{R}^N} f(x) \quad \text{with} \quad f(x) = \|Ax - y\|_2^2 + \alpha \|Sx\|_2^2,$$

we equate the gradient to zero:

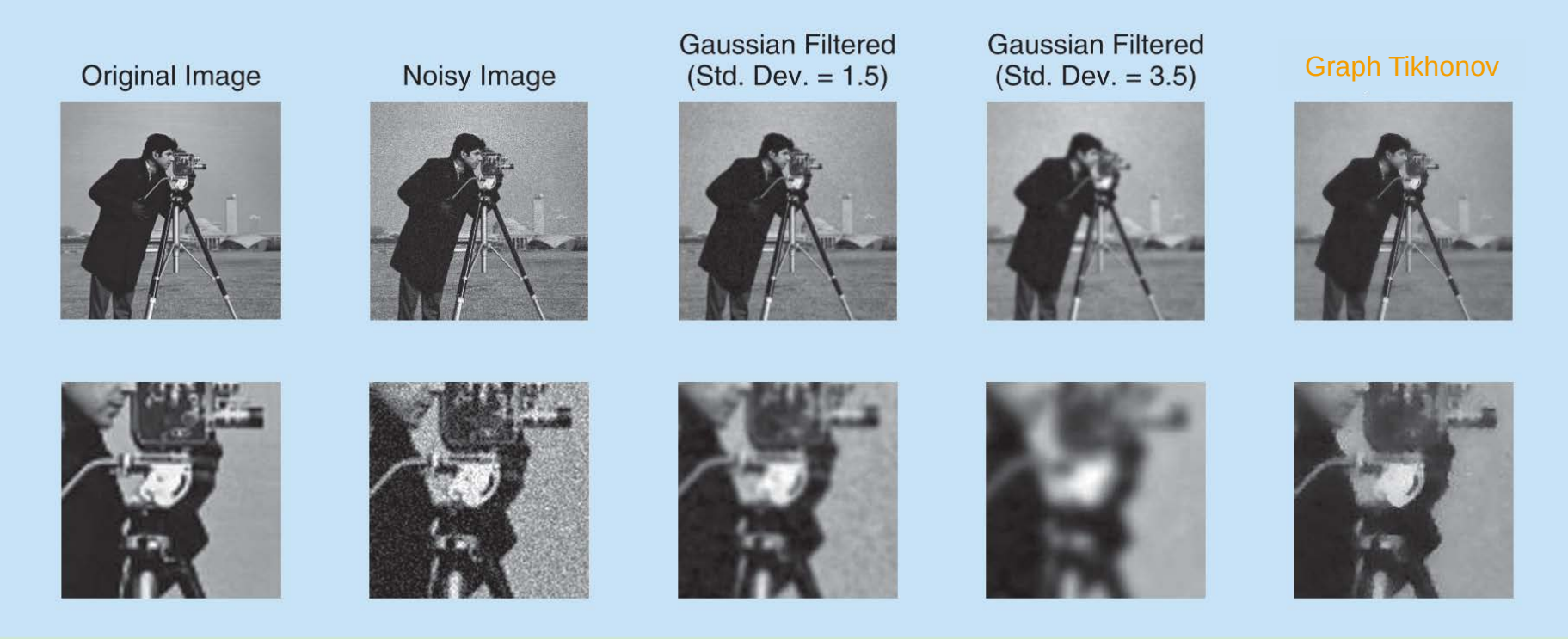
$$\begin{aligned}
\frac{df}{dx} &= d(x^\top A^\top A x + y^\top y - 2y^\top A x + \alpha x^\top L x)/dx \\
&= (A^\top A + A^\top A)x - 2A^\top y + \alpha(L + L^\top)x && \left(\tfrac{dy^\top x}{dx} = y \text{ and } \tfrac{dx^\top M x}{dx} = (M + M^\top)x\right) \\
&= 2A^\top A x + \alpha 2 L x - 2A^\top y && (A^\top A, L \text{ are symmetric}) \\
&= 2(A^\top A + \alpha L)x - 2A^\top y = 0
\end{aligned}$$

Equivalently,

$$2(A^\top A + \alpha L)x = 2A^\top y \quad \text{or} \quad \tilde{x} = (A^\top A + \alpha L)^{-1} A^\top y$$

Example: image denoising

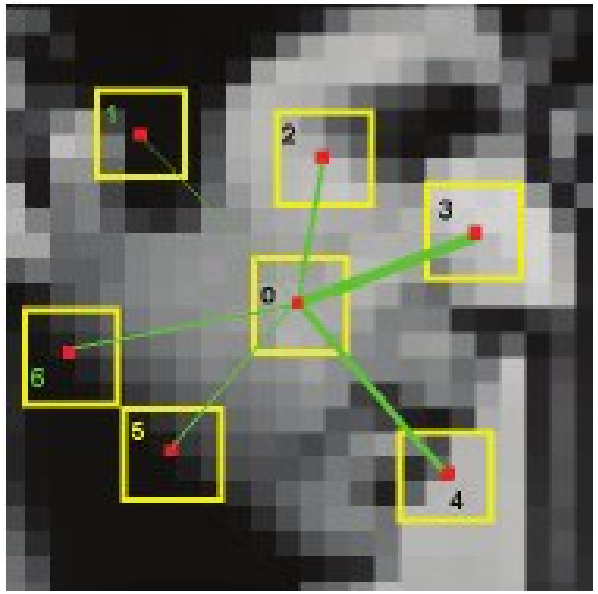
Rather than a pixel grid, use a graph that encodes pixel similarity



The emerging field of signal processing on graphs. Shuman, Narang, Ortega, Frossard, Vandergheynst. IEEE Signal Processing Magazine, 2013

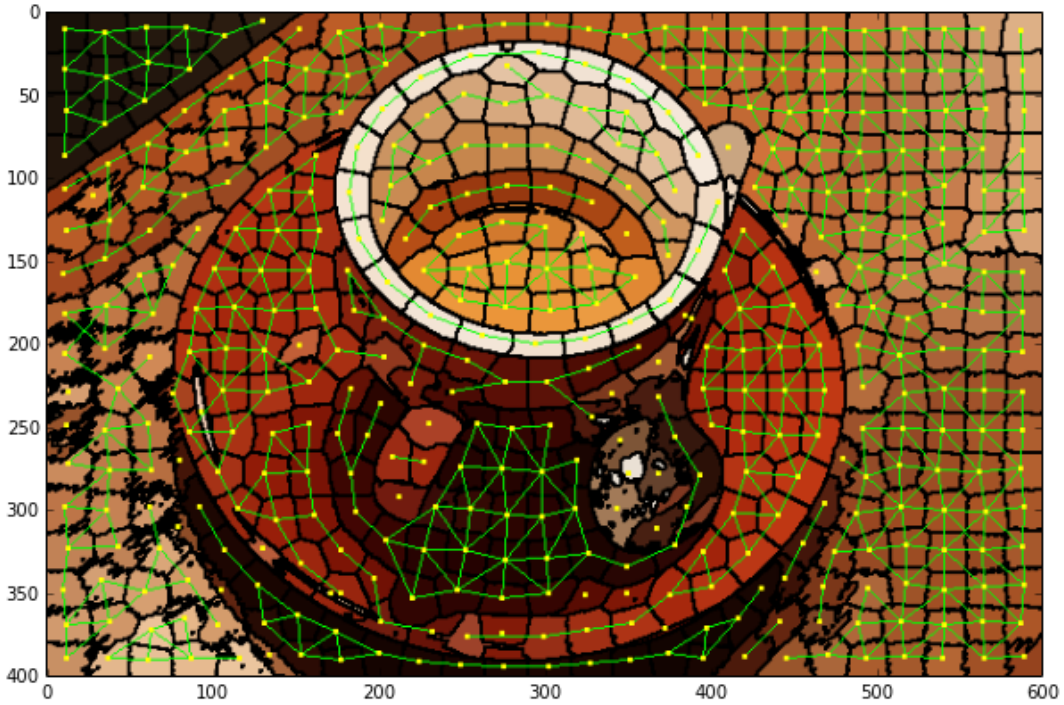
Example: denoising++

More advanced graph constructions



Rachid et al

patch-based similarity graph

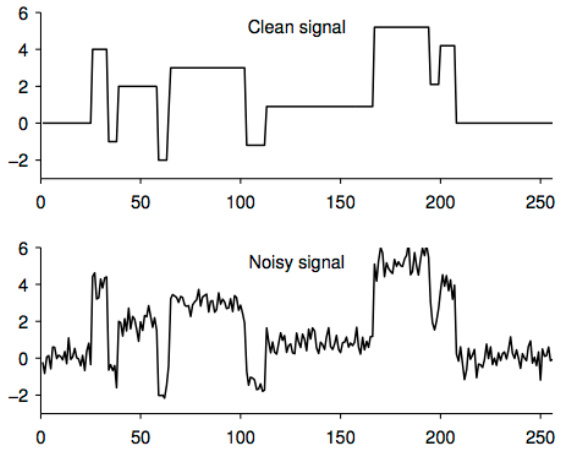


Scikit-image

superpixel-based similarity graph

Buades, Antoni (20–25 June 2005). "A non-local algorithm for image denoising". CVPR 2005
Achanta, Radhakrishna, et al. "SLIC superpixels compared to state-of-the-art superpixel methods." IEEE PAMI (2012)

Graph Total Variation (TV)

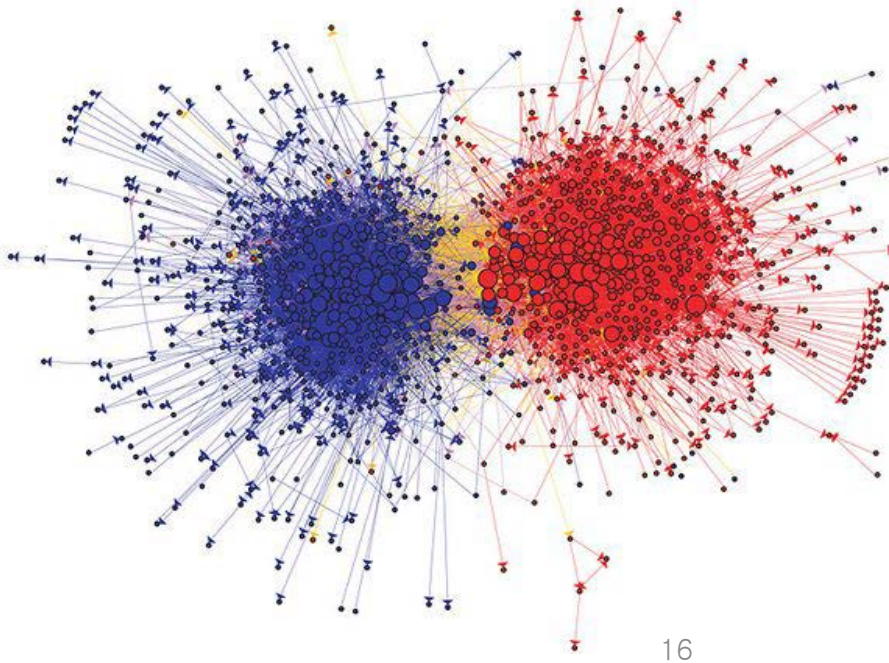


Instead of smoothness, ask for piece-wise constant solutions on graph

For general graph $G = (V, E)$, solve the following lasso-type problem:

$$\tilde{x} = \arg\min_{x \in \mathbb{R}^N} \|Ax - y\|_2^2 + R_{\text{tv}}(x; G),$$

- $R_{\text{tv}}(x; G) = \alpha \|Sx\|_1 = \alpha \sum_{w_{ij} \in E} \sqrt{w_{ij}} \, |x(i) - x(j)|$

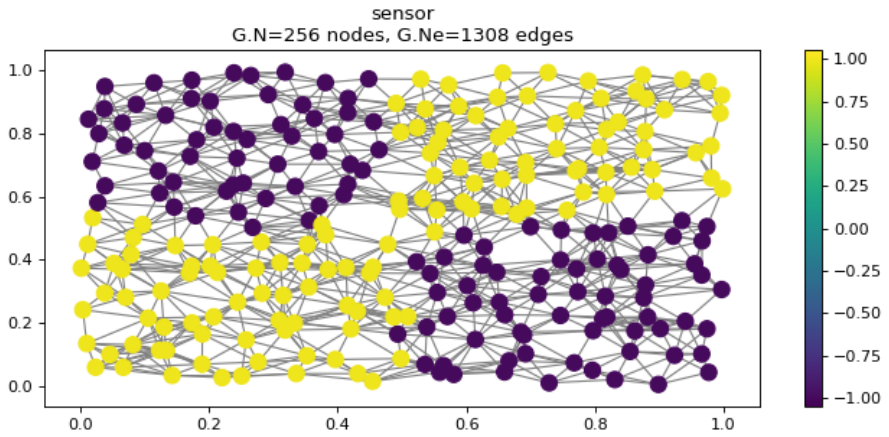


Useful, for recovering partially observed labels over a graph

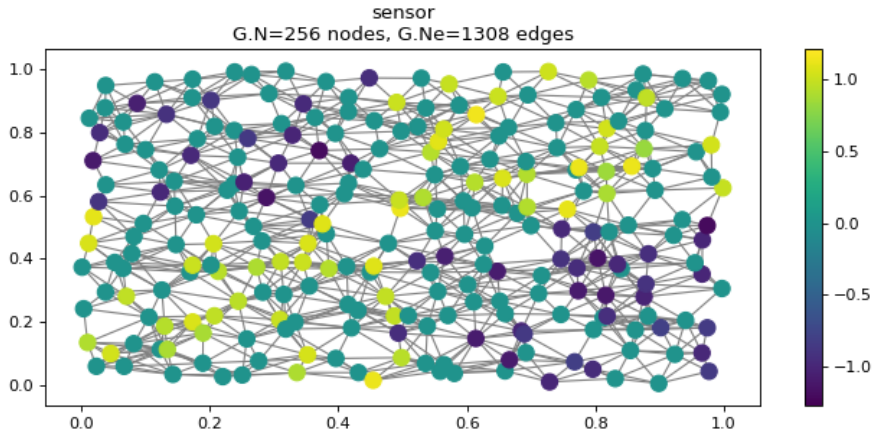
- Semi-supervised learning
- Continuous version of label propagation

Demonstration

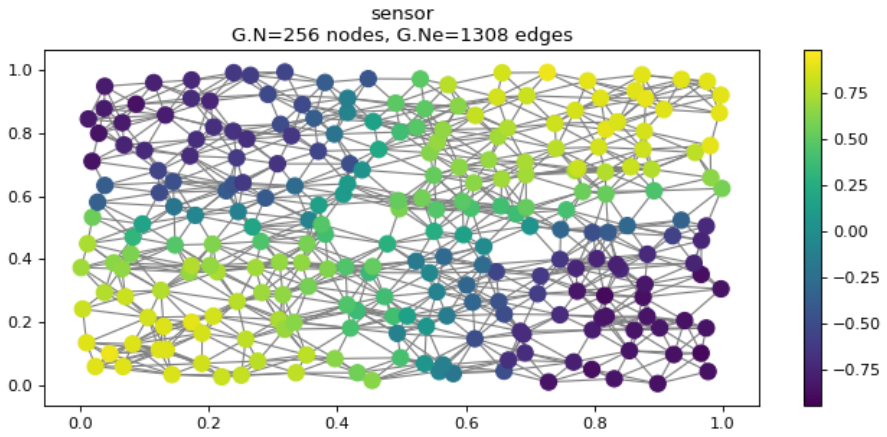
x (ground-truth)



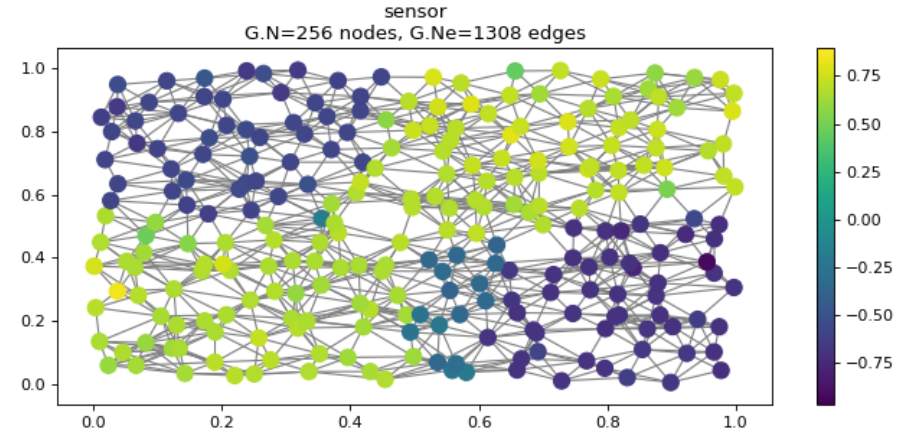
y (noisy and partially observed)



$\tilde{x}$ (graph Tikhonov)



$\tilde{x}$ (graph TV)

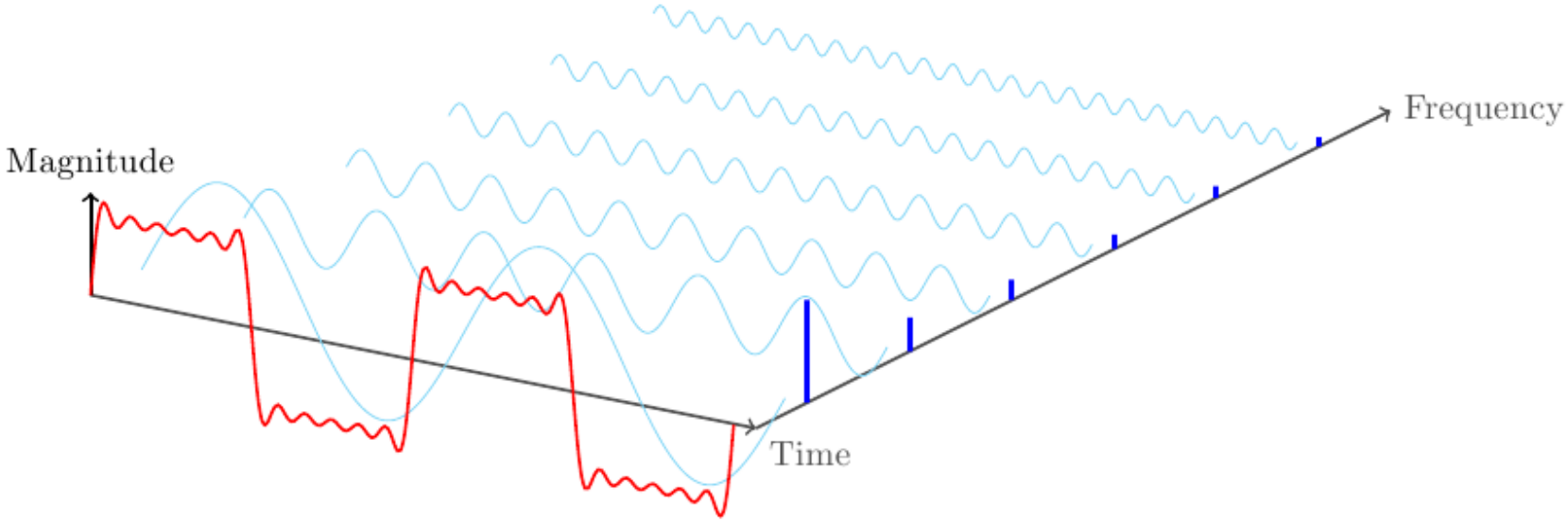


https://pygsp.readthedocs.io/en/stable/tutorials/optimization.html

Outline

- Regression 101: Tikhonov and TV
- Graph Fourier Transform
- Graph filters

a fundamental idea



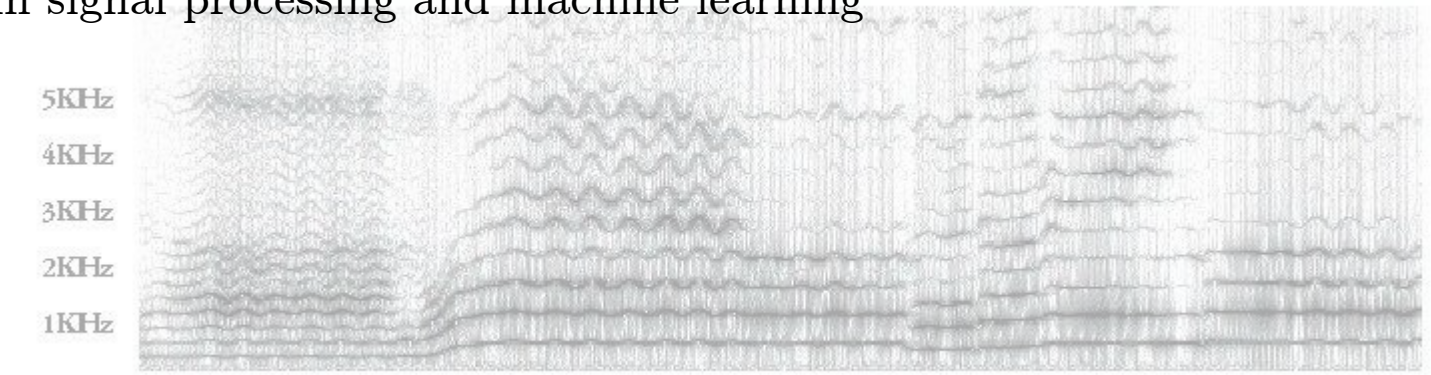
- "Express signals both in time and frequency."

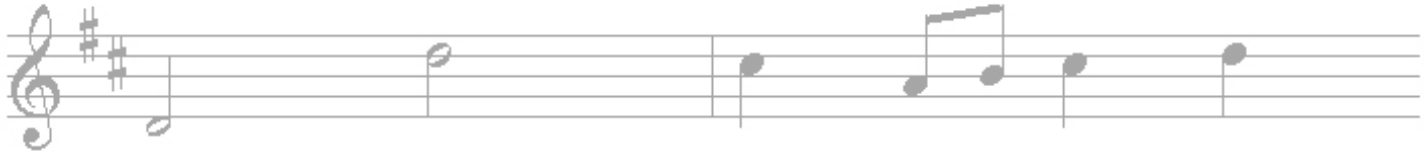
Joseph Fourier 1822.

Théorie analytique de la chaleur.

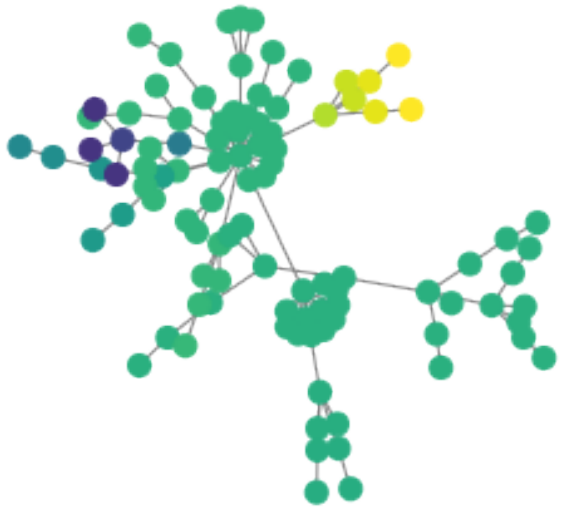
Fourier Transform benefits to data analysis

- Yields insight by transforming data into an equivalent, but more intuitive representation
  - Easy to express hypotheses/priors of data behavior in frequency domain
  - Used heavily in signal processing and machine learning

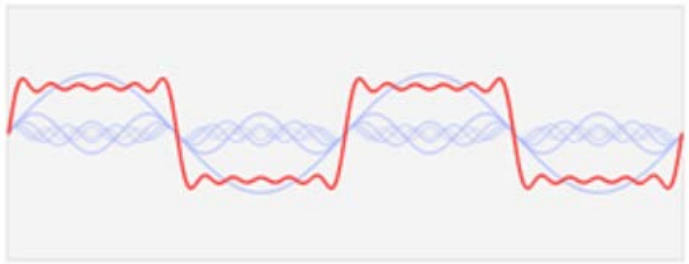




What is the frequency of a graph signal?



?



Fourier transform reminder

The Fourier transform of a continuous function x is

$$\hat{x}(\xi) = \int_{-\infty}^{\infty} x(t)\, e^{-2\pi j t \xi}\, dt \quad \text{and} \quad x(t) = \int_{-\infty}^{\infty} \hat{x}(\xi)\, e^{2\pi j t \xi}\, d\xi$$

for real frequency ξ (Hz).

The Discrete Fourier Transform (DFT) of a discrete function x (sequence) is

$$\hat{x}(k) = \sum_{i=1}^{N} x(i)\, e^{-2\pi j (k-1)\frac{i-1}{N}} \quad \text{and} \quad x(i) = \frac{1}{N} \sum_{k=1}^{N} \hat{x}(k)\, e^{2\pi j (i-1)\frac{k-1}{N}}$$

for $k = 1, \ldots, N$.

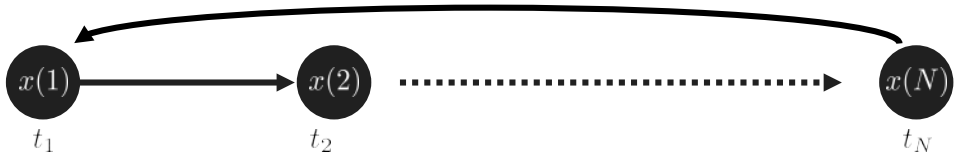
In vector form: $\hat{x}(k) = a_k^\top x$ and $x(i) = b_k^\top \hat{x}$ and, further, we have $\hat{x} = Ax$ and $x = B\hat{x} = A^{-1}\hat{x}$.

An alternative perspective

The DFT of a periodic signal x is nothing else than

$$\hat{x} = U_{\mathrm{DFT}}^{\top} x \quad \text{and} \quad x = U_{\mathrm{DFT}} \hat{x},$$

where the DFT matrix U_{DFT} is the eigenvector matrix of the ring adjacency matrix W_R which is exactly the cyclic time-shift operator:



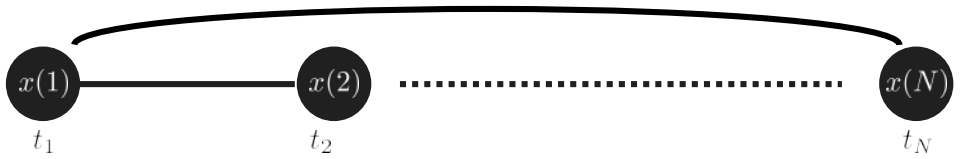
$$W_R = \begin{pmatrix} & & & 1 \\ 1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} = U_{\mathrm{DFT}} \begin{pmatrix} \lambda'_1 & & \\ & \ddots & \\ & & \lambda'_N \end{pmatrix} U_{\mathrm{DFT}}^{\top}$$

An alternative perspective

The DFT of a periodic *real* signal x is nothing else than

$$\hat{x} = U_{\text{DFT}}^{\top} x \quad \text{and} \quad x = U_{\text{DFT}} \hat{x},$$

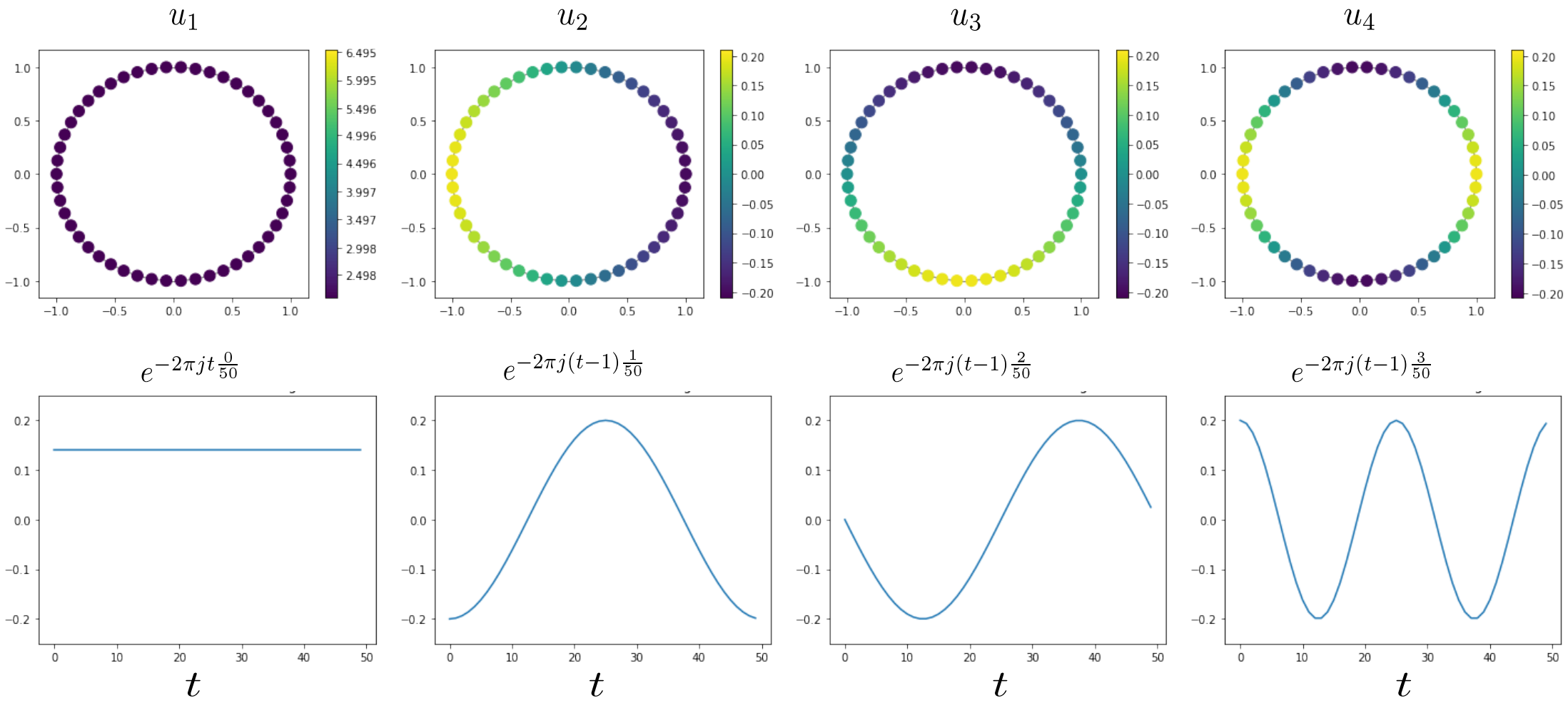
where the DFT matrix U_{DFT} is the eigenvector matrix assosiated with the Laplacian of the ring:



$$L_R = \begin{pmatrix} 2 & -1 & & & & -1 \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & -1 \\ -1 & & & & -1 & 2 \end{pmatrix} = U_{\text{DFT}} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{pmatrix} U_{\text{DFT}}^{\top}$$

Strang, G. (1999). The discrete cosine transform. SIAM review, 41(1), 135-147.

DFT basis as ring Laplacian eigenvectors



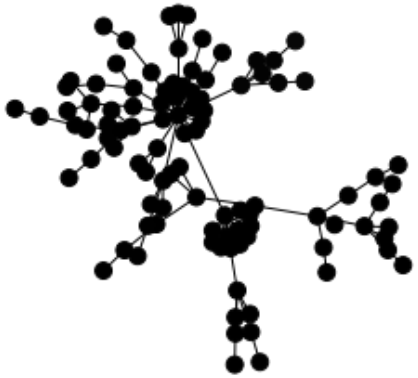
Graph Fourier Transform

We define the Graph Fourier Transform (GFT) of a graph signal x w.r.t. graph G as

$$\hat{x} = U^\top x \quad \text{and} \quad x = U\hat{x},$$

where $U \in \mathbb{R}^{N \times N}$ is the eigenvector matrix assosiated with a graph shift matrix GS:

$$GS = U\Lambda U^\top = U \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{pmatrix} U^\top$$

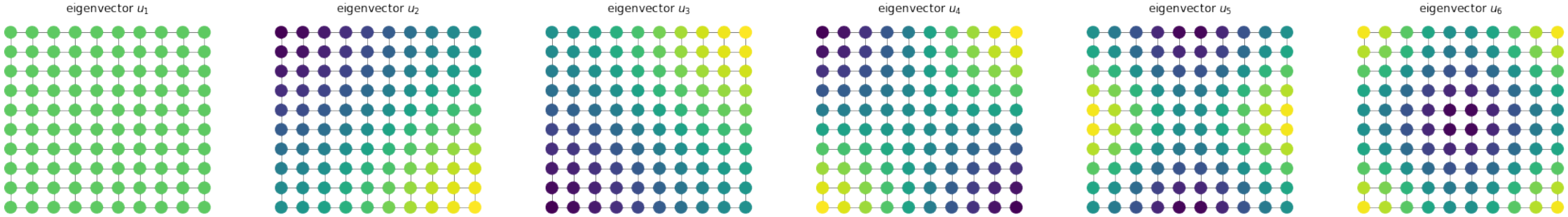


Two competing perspectives:

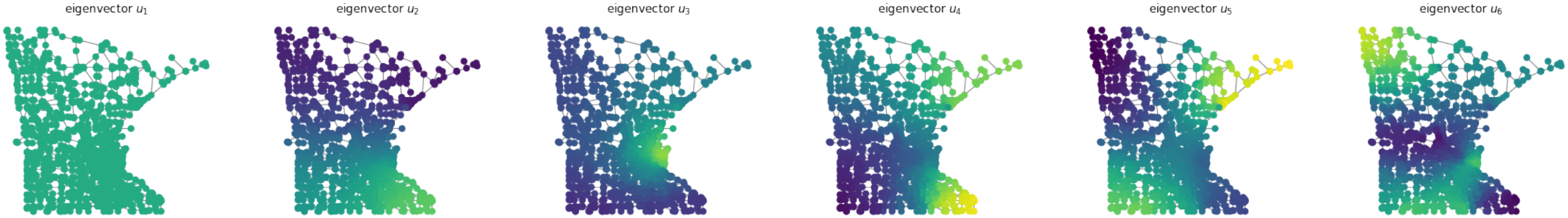
- GS should be the weighted adjacency matrix W
  - allows for directed graphs, but well-defined only for undirected graphs (Jordan decomposition is unstable).
- GS should be a Laplacian matrix L (we are going to use this next).
  - L is symmetric, so GFT always well-defined

Some example GFT bases

- regular 10x10 grid

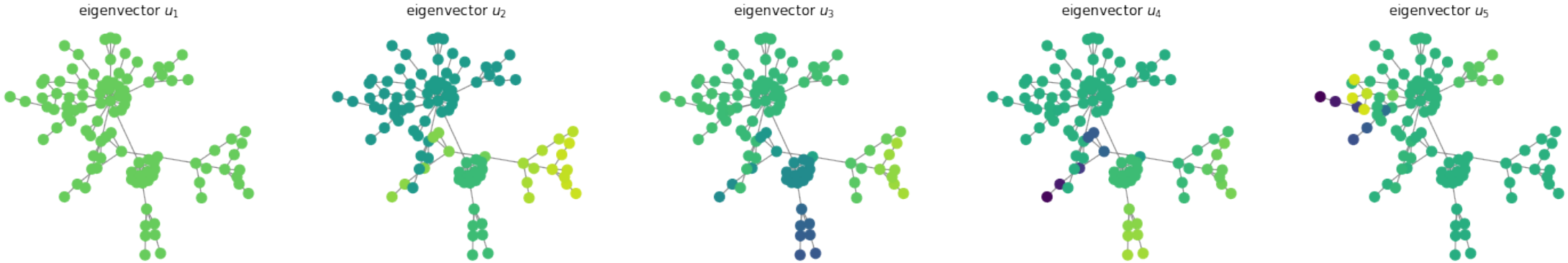


- Minnesota road graph

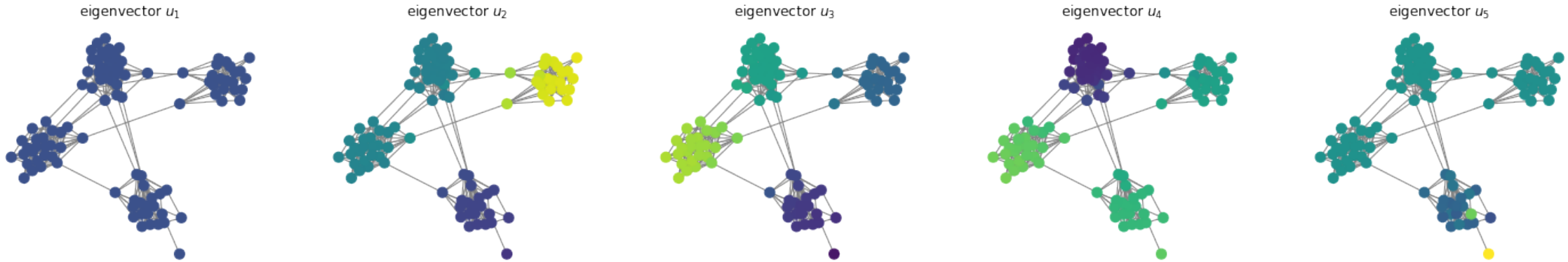


Some example GFT bases

- Barabási-Albert scale-free network



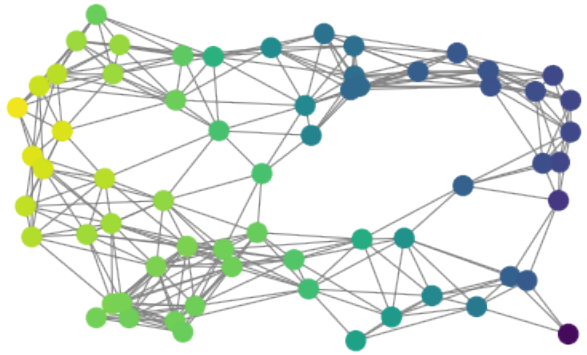
- Stochastic block model

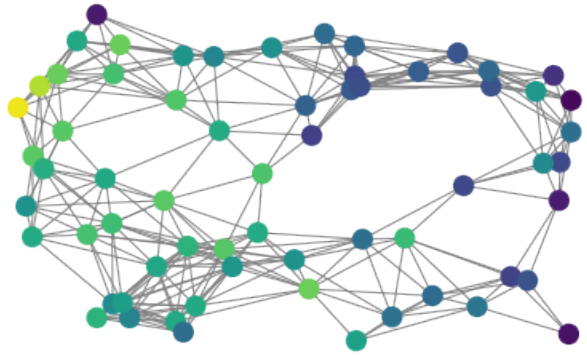


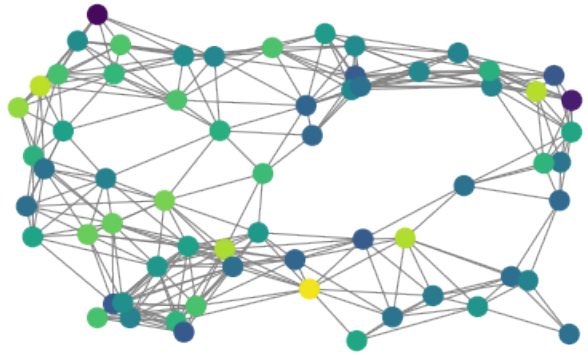
GFT of graph signals

vertex domain

$x(i)$

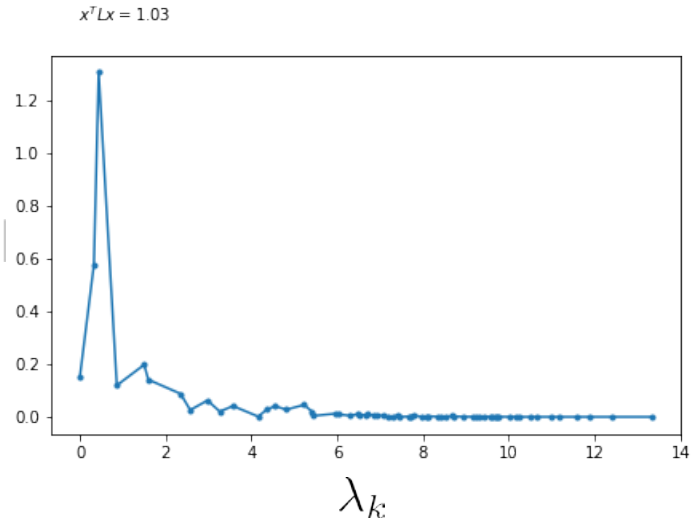


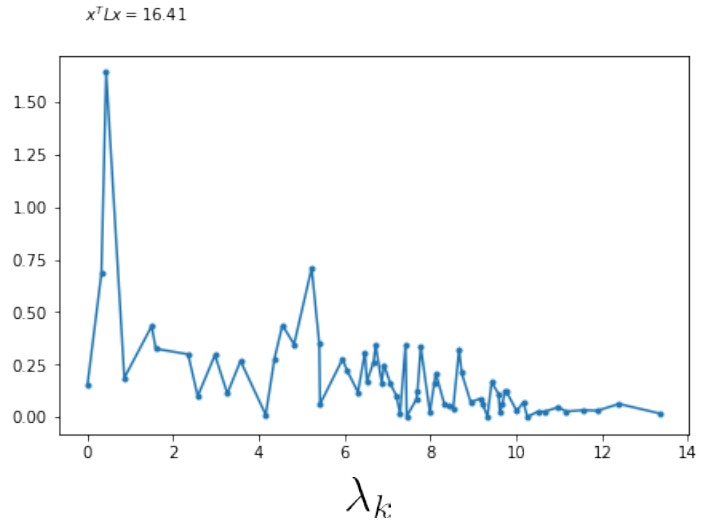


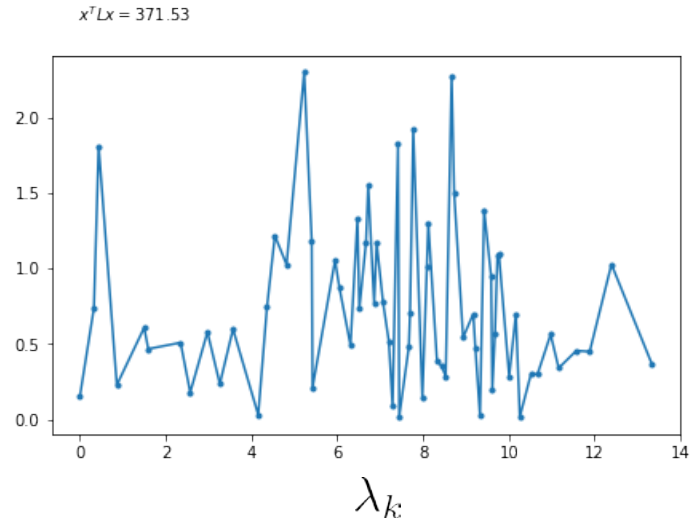


graph spectral domain

$|\hat{x}(k)|$







Properties of GFT

By the min-max theorem, $u_1, \dots, u_N$ form the basis of minimum smoothness:

$$u_k = \underset{\|x\|_2=1}{\arg\min}\ x^\top L x \quad \text{subject to} \quad x \perp u_1, \dots x \perp u_{k-1}.$$

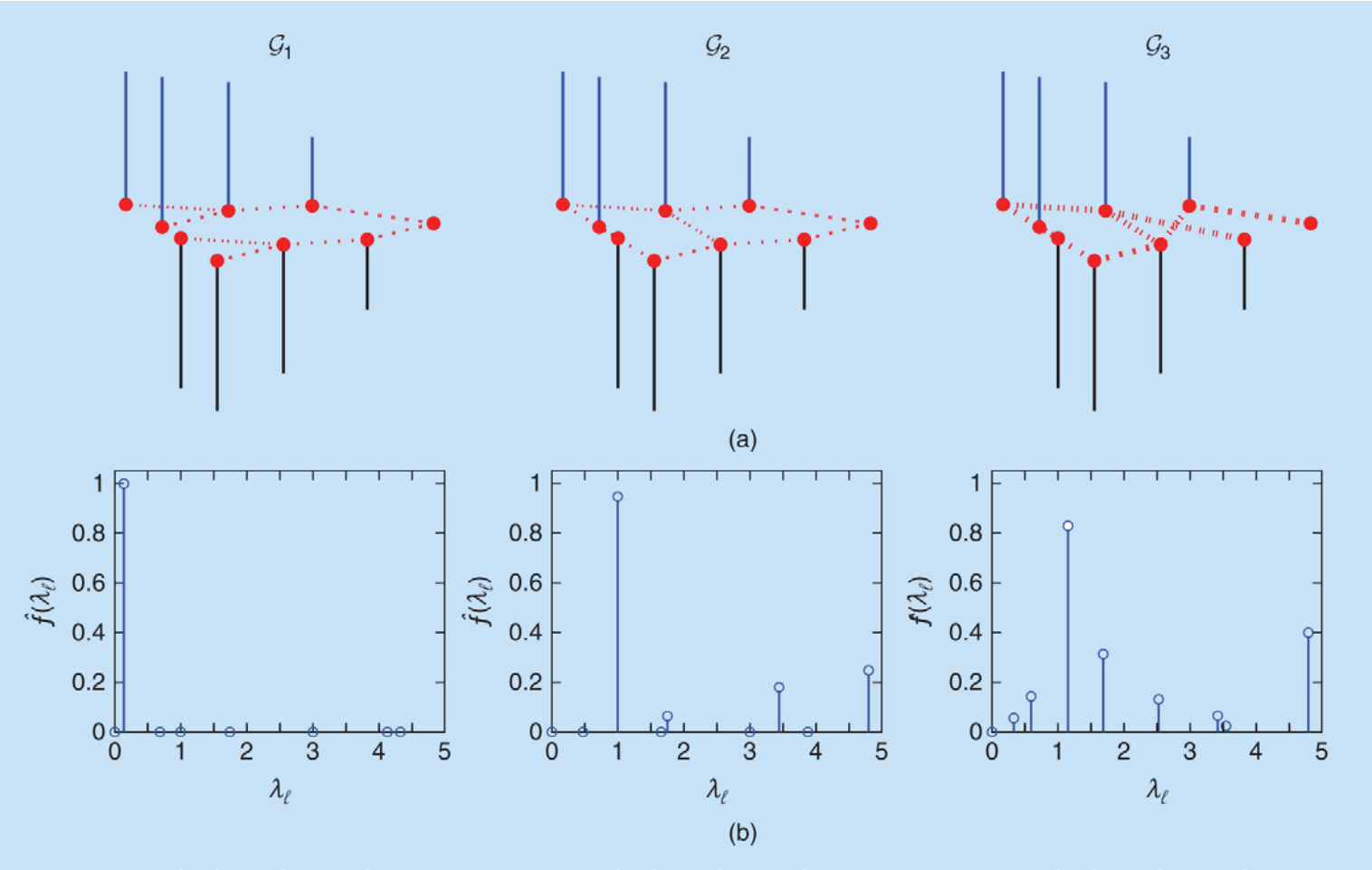
It follows that

- the lowest frequency $\lambda_1 = 0$ captures the signal's DC-offset
- all frequency modes u_k with $k > 1$ have zero mean
- the higher the graph frequency λ_k, the faster the frequency mode will vary/oscillate on the graph

More properties:

- Parseval theorem: $\|\hat{x}\|_2 = \|x\|_2$
- Linearity: if $x = a + b$ then $\hat{x} = \hat{a} + \hat{b}$
- $\hat{\delta}_i = U(i,:)^\top$ and $\hat{u_i} = \delta_i$

GFT depends on the graph!



The emerging field of signal processing on graphs. Shuman, Narang, Ortega, Frossard, Vandergheynst. IEEE Signal Processing Magazine, 2013

Outline

- Regression 101: Tikhonov and TV
- Graph Fourier Transform
- Graph filters

Revisiting graph Tikhonov regularization

What does the TK solution mean?

$$\tilde{x} = (A^\top A + \alpha L)^{-1} A^\top y$$

For simplicity, set $A = I$ (denoising) and let $L = U \Lambda U^\top$ be the GS matrix of G:

$$\begin{aligned}
\tilde{x} &= (I + \alpha L)^{-1} y \\
&= (U U^\top + \alpha U \Lambda U^\top)^{-1} y && U \text{ is orthogonal, so } U U^\top = I \\
&= (U(I + \alpha \Lambda) U^\top)^{-1} y && \text{for invertible } A, B, C : (ABC)^{-1} = C^{-1} B^{-1} A^{-1} \\
&= U (I + \alpha \Lambda)^{-1} U^\top y && \text{Matrix } I + \alpha \Lambda \text{ is invertible} \\
&= U (I + \alpha \Lambda)^{-1} \hat{y} && \hat{y} = U^\top y \\
&= U \left(h_{\text{tk}} \circ \hat{y} \right),
\end{aligned}$$

where $h_{\text{tk}} = [\frac{1}{1+\alpha\lambda_1}, \cdots, \frac{1}{1+\alpha\lambda_N}]$.

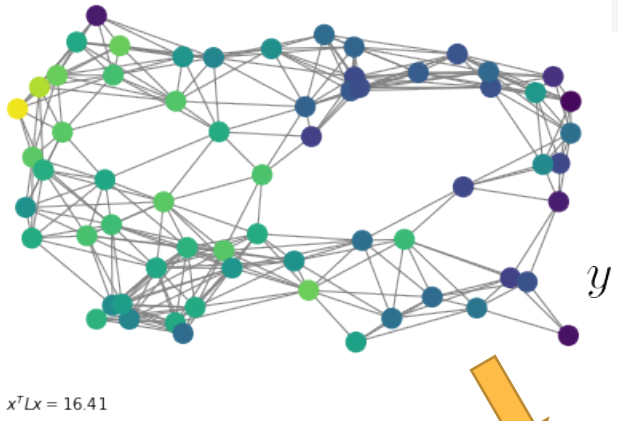
Revisiting graph Tikhonov regularization

In the graph spectral domain:

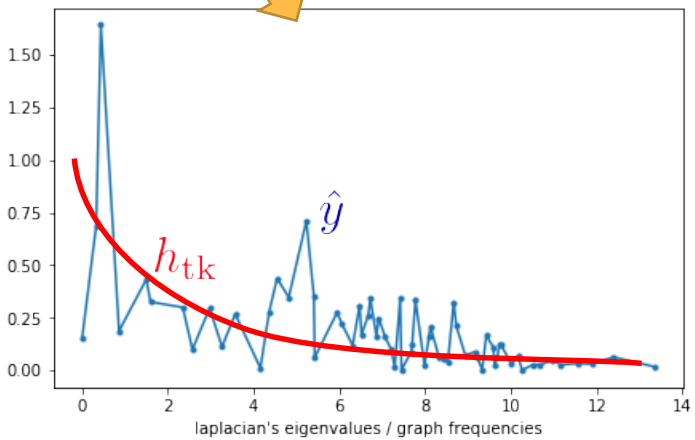
frequency response

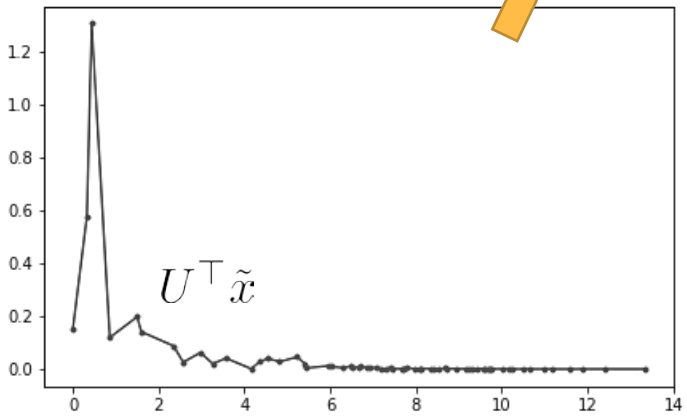
$$U^\top \tilde{x} = h_{\text{tk}} \circ \hat{y}$$

vertex domain

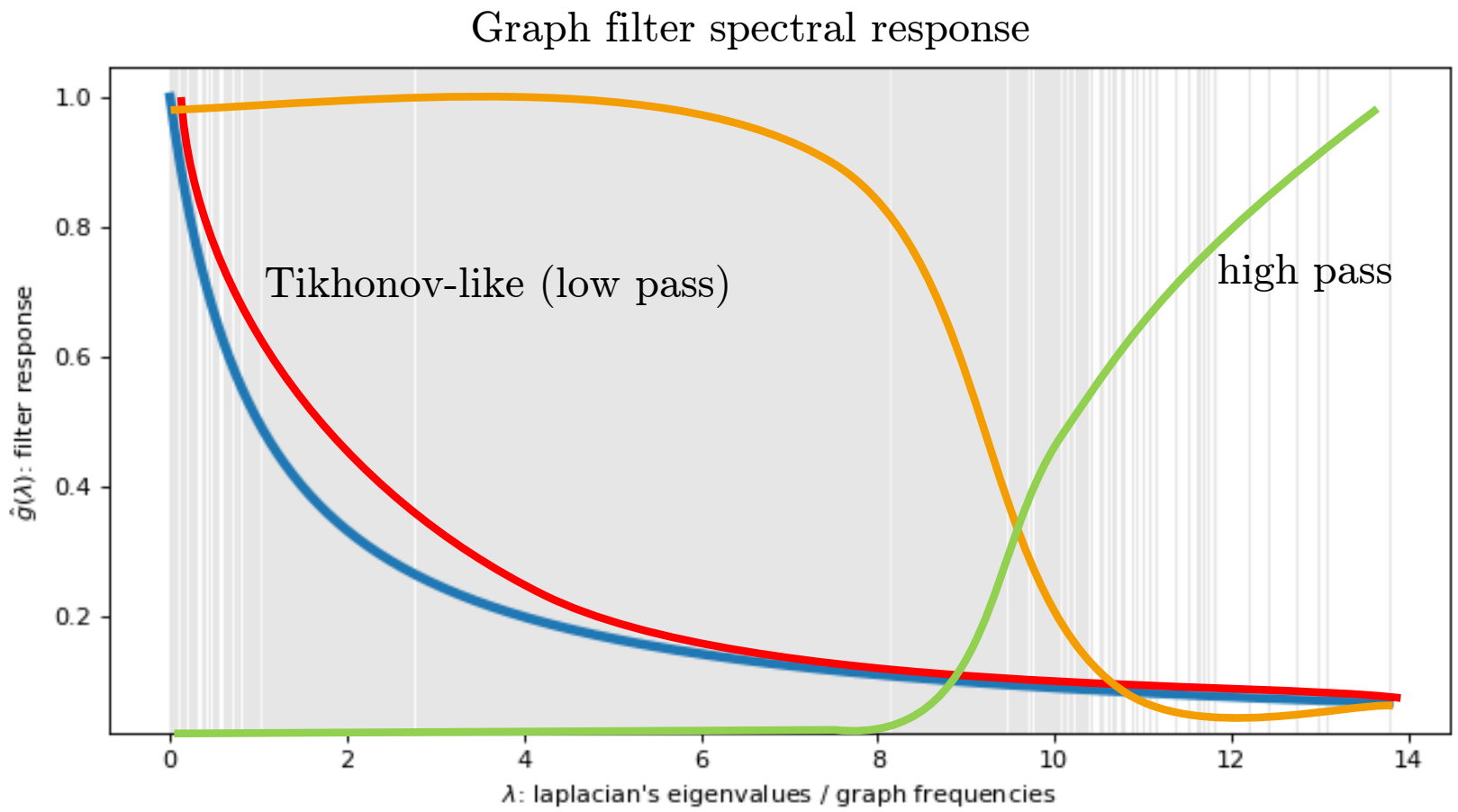


graph spectral domain





Graph filters



Graph filters

Let $h : \mathbb{R} \to \mathbb{R}$ be a scalar function.

A graph filter $h(L)$ is an $N \times N$ matrix that takes the form:

$$h(L) = U h(\Lambda) U^\top = U \begin{pmatrix} h(\lambda_1) & & \\ & \ddots & \\ & & h(\lambda_N) \end{pmatrix} U^\top$$

Filtering, therefore, amounts to multiplication in the spectral domain (convolution theorem):

$$h(L)x = U h(\Lambda) U^\top x = U(\mathrm{diag}(h(\Lambda)) \circ \hat{x}) = \sum_{k=1}^{N} (h(\lambda_k) \circ \tilde{x}(k))\, u_k$$

Properties of graph filters

- They are linear operators
$$h(L)(x+y) = h(L)x + h(L)y$$
- They are symmetrix operators
$$h(L)^\top = h(L)$$
- The order of application does not matter
$$h(L)g(L)x = g(L)h(L)x$$
- They are only invertible when $h(\lambda) \neq 0$

Polynomial graph filters (FIR)

General graph filter $h(L)$

- N degrees of freedom
- Computing $h(L)x$ requires full eigendec. and takes $O(N^3)$ time

In practice, we usually prefer to work in the vertex domain.

Polynomial graph filters (order P monomial):

$$h_P(L) = \sum_{p=0}^{P} a_p L^p \quad \rightarrow \quad h_P(\lambda) = \sum_{p=0}^{P} a_p \lambda^p$$

- $P+1$ degrees of freedom
- computing $h_P(L)x$ requires P+1 sparse matrix-vector multiplications and takes $O(PM)$ time (for M edges).

Localization of polynomial filters

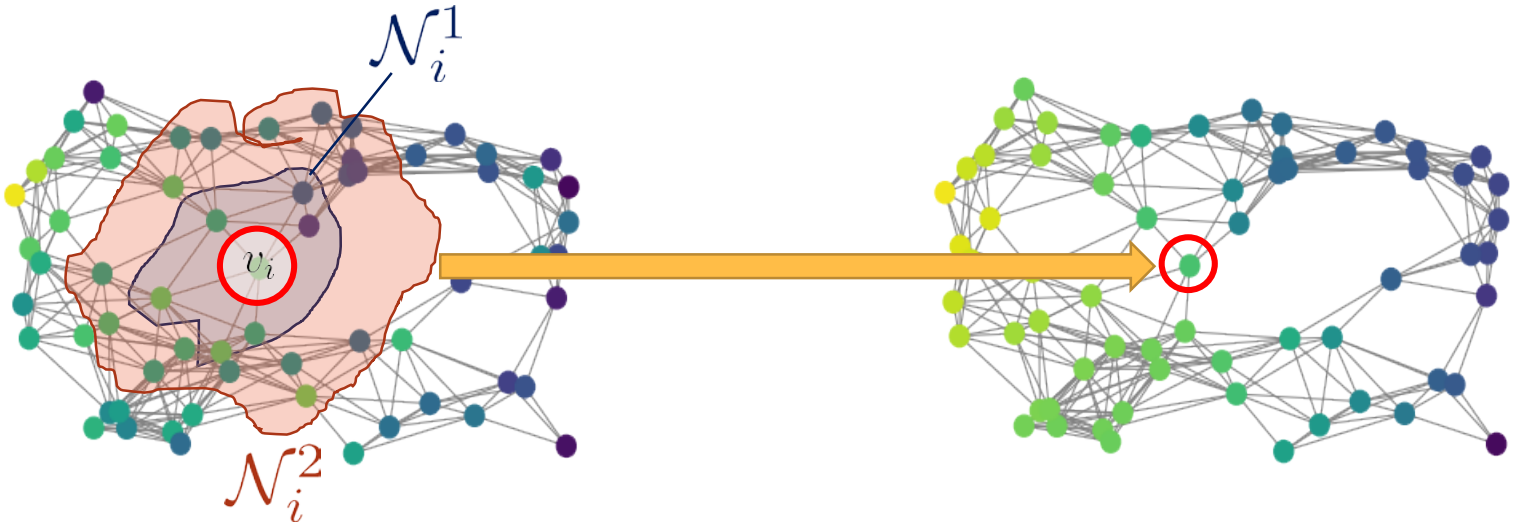
Hint:

$$y(i) = [\sum_{p=0}^{P} a_p L^p x](i)$$
$$= \sum_{p=0}^{P} a_p [L^p x](i)$$

A P-th order polynomial $h_P(L)$ is P-localized in G:

$$y(i) = [h_P(L)x](i) = \sum_{v_j \in \mathcal{N}_i^P} c_j \, x(j)$$

Why?



FIR approximate any graph filter

Polynomial fitting:

$$\min_{\alpha} \| \sum_{k=1}^{N} \left(h(\lambda_k) - a_0 + a_1 \lambda_k + a_2 \lambda_k^2 + \cdots + a_P \lambda_k^P \right) \|_2$$

If λ_k are unknown, choose equally spaced points in $[0, \lambda_N = \|L\|_2]$.

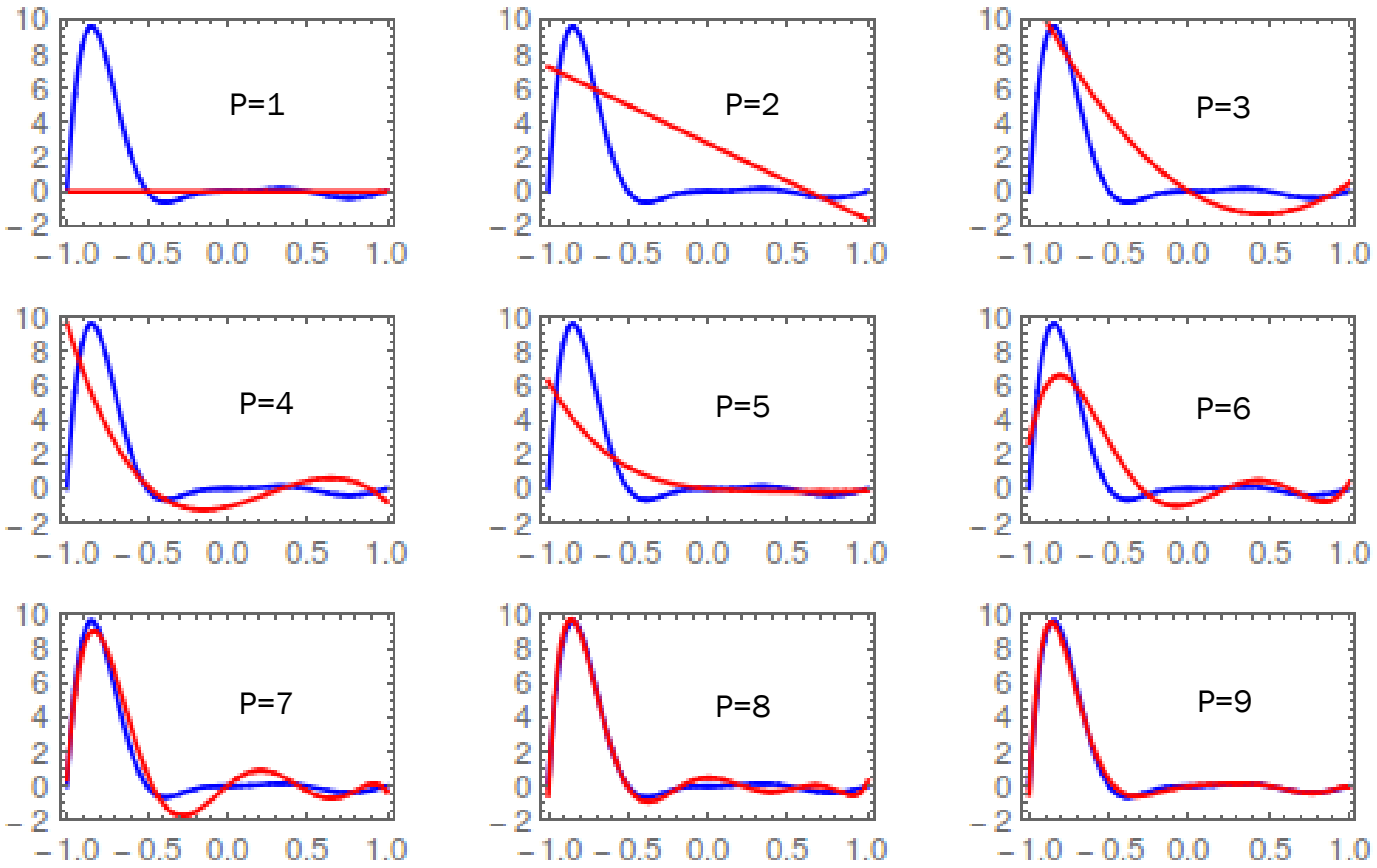
Expressed in matrix form

Vandermonde matrix

$$\min_{\alpha} \| \begin{bmatrix} h(\lambda_1) \\ h(\lambda_1) \\ h(\lambda_2) \\ \vdots \\ h(\lambda_N) \end{bmatrix} - \begin{bmatrix} 1 & \lambda_1 & \lambda_1^2 & \dots & \lambda_1^P \\ 1 & \lambda_2 & \lambda_2^2 & \dots & \lambda_2^P \\ 1 & \lambda_3 & \lambda_3^2 & \dots & \lambda_3^P \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_N & \lambda_N^2 & \dots & \lambda_N^P \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_P \end{bmatrix} \|_2 = \min_{\alpha} \|\text{diag}(h(\Lambda)) - Va\|_2$$

Polynomial coefficients (using ordinary least squares) are $a = (V^\top V)^{-1} V^\top \text{diag}(h(\Lambda))$.

Role of polynomial order to approximation



Higher P gives

- Better fit
- Higher complexity
- Less localization

The less smooth a function is, the more difficult to approximate it

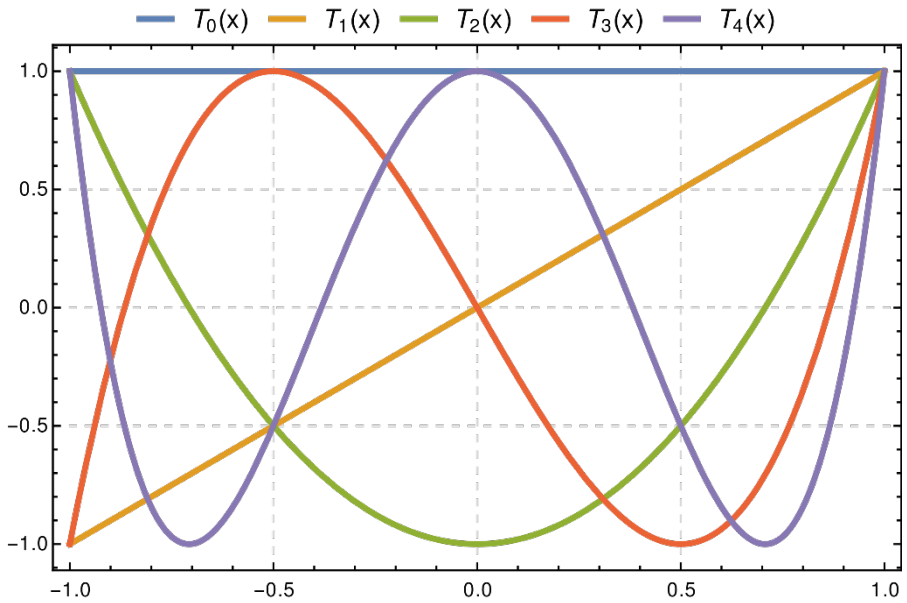
- Would need P = 100, 200, ..
- Ill conditioning problem

$$\begin{bmatrix} 1 & \lambda_1 & \lambda_1^2 & \dots & \lambda_1^P \\ 1 & \lambda_2 & \lambda_2^2 & \dots & \lambda_2^P \\ 1 & \lambda_3 & \lambda_3^2 & \dots & \lambda_3^P \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_N & \lambda_N^2 & \dots & \lambda_N^P \end{bmatrix}$$

Chebyshev polynomials

- Monomial design suffers from conditioning

- Most popular solution are Chebyshev polynomials
  - avoid least-squares formulation
  - rely on approximation theory



High-level idea

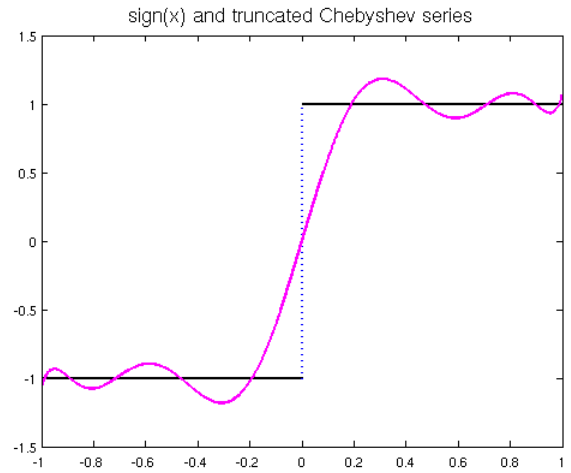
- Construct a sequence of orthogonal polynomials $T_t(\lambda)$ where $t = 0, 1, \ldots, P$
- Build the corresponding graph filters $T_t(L)$
- Compute $x_t = T_t(L)x$
- Find a_t such that

$$y = h(L)x \approx \sum_{t=0}^{P} a_t x_t.$$

Approximating functions with Chebyshev polynomials

Chebyshev polynomials $T_t(\lambda)$ of the first kind are defined by the recurrence relations

$$T_0(\lambda) = 1 \quad \text{and} \quad T_1(\lambda) = \lambda \quad \text{and} \quad T_{t+1}(\lambda) = 2\lambda T_t(\lambda) - T_{t-1}(\lambda).$$



A function $h(\lambda)$ in $-1 \leq \lambda \leq 1$ can be expressed via the expansion

$$h(\lambda) = \sum_{t=0}^{\infty} a_t T_t(\lambda) \approx \sum_{t=0}^{P} a_t T_t(\lambda).$$

The coefficients a_t are found by an inner product.

- The Chebyshev series converges to $h(\lambda)$ if the function is piecewise smooth and continuous.
- At a discontinuity, the series will converge to the average of the right and left limits.
- Shifting and normalization is needed to approximation functions in $[0, \lambda_{max}]$.

Chebyshev graph filtering

Finally, to filter we do:

$$h(L)x \approx \sum_{t=0}^{P} a_t T_t(L)x = \sum_{t=0}^{P} a_t x_t.$$

$x_{t+1} = T_{t+1}(L)x$ can be computed recursiverly as follows:

$$x_0 = 1 \quad \text{and} \quad x_1 = Lx \quad \text{and} \quad x_{t+1} = 2Lx_t - x_{t-1}.$$

In total, only P sparse matrix-vector multiplications (Lx_t) are needed.

"Chebyshev Polynomial Approximation for Distributed Signal Processing". David I Shuman, Pierre Vandergheynst, and Pascal Frossard. DCOSS.2013

$\text{ARMA}_{\text{P,Q}}$ graph filters

Rational graph frequency response (ratio of polynomials)

$$h_{P,Q}(\lambda) = \frac{b_P(\lambda)}{a_P(\lambda)} = \frac{\sum_{q=0}^{Q} b_q \lambda^q}{1 + \sum_{p=1}^{P} a_p \lambda^p}$$

$$\begin{aligned} h_{P,Q}(L) &= a_P(L)^{-1} b_Q(L) x \\ &= U a_P(\Lambda)^{-1} U^\top U b_Q(\Lambda) U x \\ &= U a_P(\Lambda)^{-1} b_Q(\Lambda) U x \\ &= U \text{diag}(a_P(\Lambda)^{-1} b_Q(\Lambda)) \hat{x} \end{aligned}$$

- P+Q+1 degrees of freedom
- Non-localized
- Higher modeling capacity
- Direct implementation:
  - solve linear system using sparse conjugate gradient (fastest)

"Distributed Autoregressive Moving Average Graph Filters". Andreas Loukas, Andrea Simonetto, and Geert Leus. Signal Processing Letters 2015

Tikhonov is an $\text{ARMA}_{1,0}$ graph filter

Recall Graph Tikhonov:

$$x_{\text{tk}} = (I + \alpha L)^{-1} y = h_{\text{tk}}(L) y \quad \text{with } h_{\text{tk}}(\lambda) = \frac{1}{1 + \alpha\lambda}$$

- It amounts to filtering y by an $\text{ARMA}_{1,0}$ with $b_0(\lambda) = 1$ and $a_1(\lambda) = 1 + \alpha\lambda$.

Consider the autoregressive equation:

$$s_{t+1} = y - \alpha L s_t \quad \text{with} \quad s_0 = y$$

After the above converges (if it does), we should have $s_{t+1} = s_t = s$ or equivalently

$$s = y - \alpha L s \Leftrightarrow (I + \alpha L) s = y \Leftrightarrow s = (I + \alpha L)^{-1} y$$

In other words, $s = \lim_{t \to \infty} s_t = x_{\text{tk}}$!

The convergence happens when $\lambda_{\max}(I + \alpha L) < 1 \;\Leftrightarrow\; 1 + \alpha \lambda_{\max}(L) < 1 \;\Leftrightarrow\; \alpha < \frac{1}{\lambda_{\max}(L)}$.

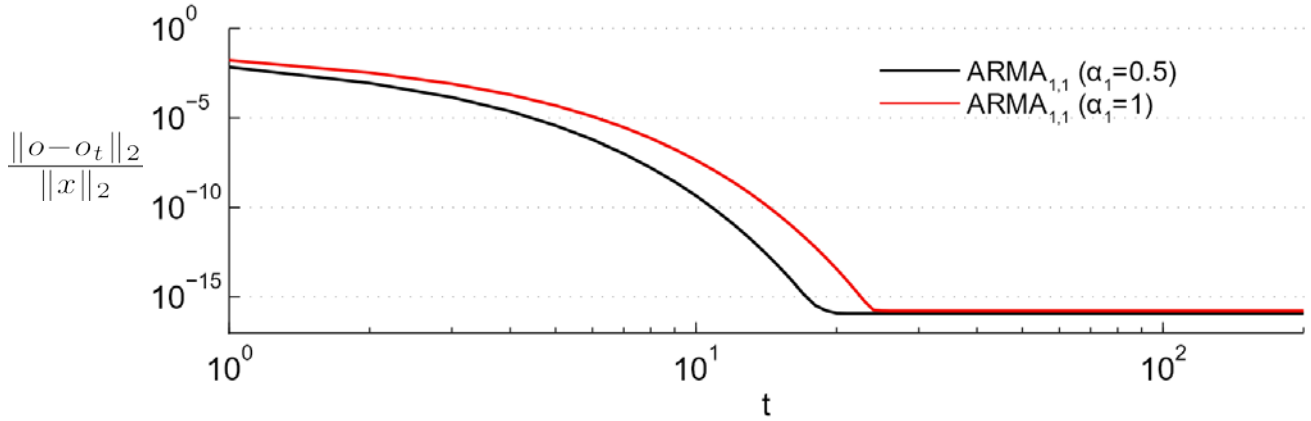
Simple $\text{ARMA}_{1,1}$ implementation

At steady-state, the autoregressive equation

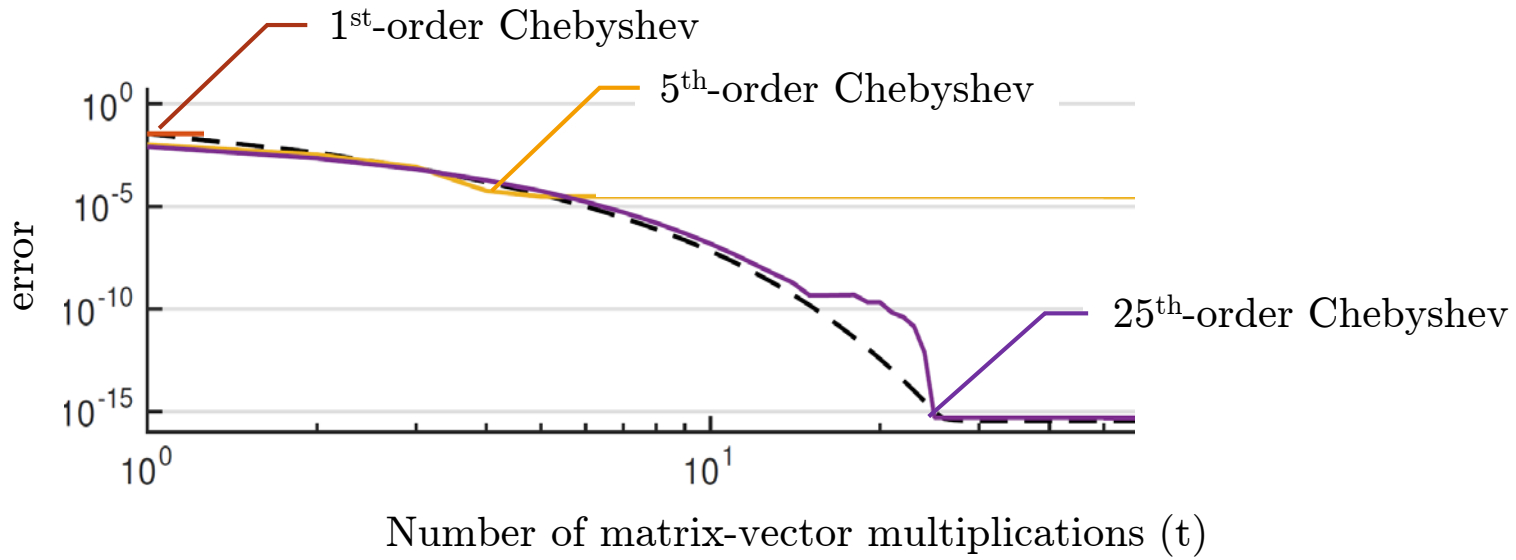
$$\begin{aligned} \text{graph filter state} \quad & s_{t+1} = a_1 L s_t + \left(b_0 + \frac{b_1}{a_1} \right) x \\ \text{graph filter output} \quad & o_{t+1} = s_{t+1} - \frac{b_1}{a_1} x, \end{aligned} \qquad (\text{ARMA}_{1,1})$$

computes $o = \lim_{t \to \infty} o_t = h_{1,1}(L)x$ with $h_{1,1}(\lambda) = \dfrac{b_0 + b_1 \lambda}{1 + a_1 \lambda}$.

For $a_1 < 1/\lambda_{\max}(L)$, the above converges with rate $\frac{\|o - o_t\|}{\|x\|} = O(e^{-(1-a_1)t})$.

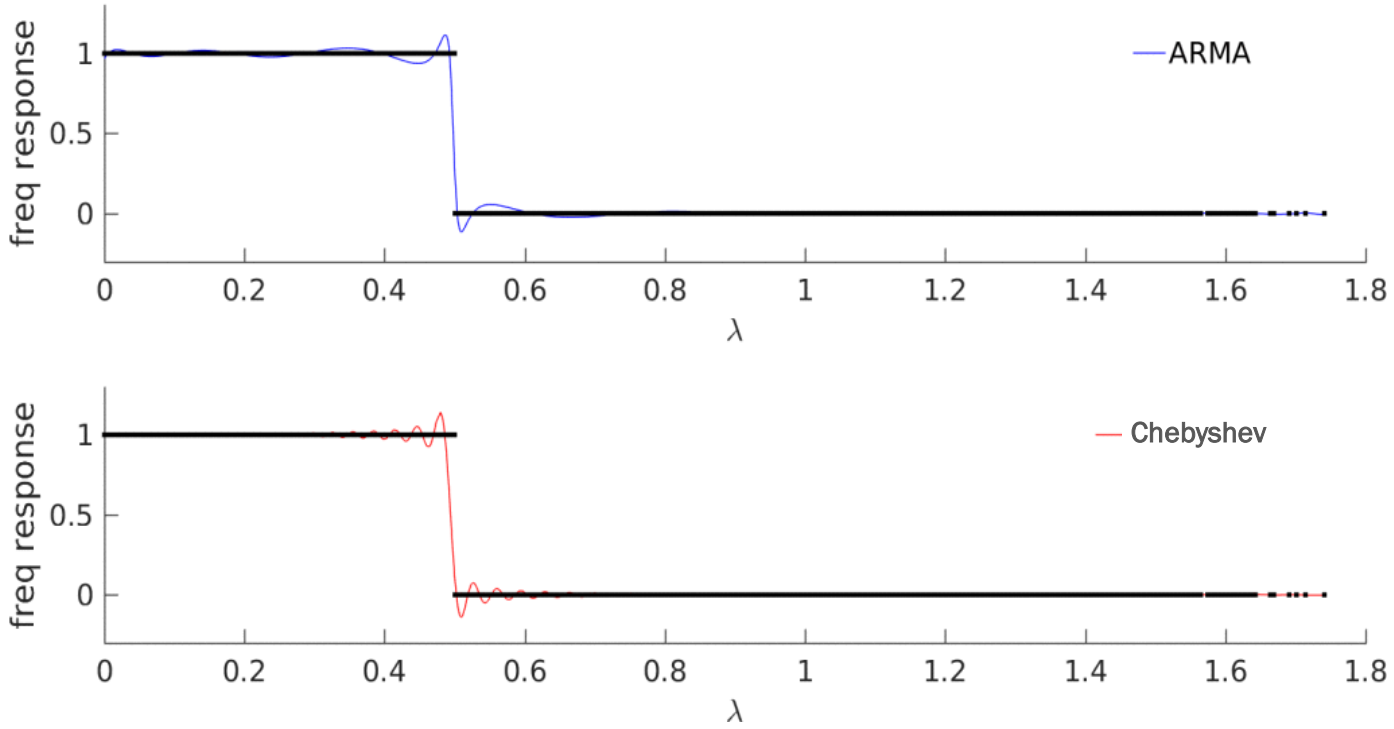


Comparison: solving graph Tikhonov



- Chebyshev give good approximation for P=25
- ARMA gives exact solution (P=1, Q=0) but needs to converge first

Comparison: ideal low-pass



- Similar approximation,
- but different parameter complexity
  - ARMA: P=2, Q=18
  - Chebyshev: P = 150

- Chebyshev design optimizes worst-case error and suffers from ripple effect

Summary

Graph signal processing:
- Study of graph signals
- Graph is interpreted as the domain where data live in

Direct generalization of key concepts from regular grids to graphs
- Regression problems (interpolation and denoising)
- Frequency
- Filters

Active research topics:
- Sampling theory for graphs
- Graph stationarity
- Graph inference
- Forecasting for time-series on graphs
- … and many others