University of Illinois at Urbana-Champaign

Signal Processing and Machine Learning on Graphs

Shrikant Venkataramani

12 November 2020

CS 598 PS
Machine Learning for Signal Processing

Motivation

- Lots of sensors and lots of data
  - Mean yearly temperature across a nation
  - Variation of traffic across a big city
  - Transfer of information through social networks
  - Neuro-imaging data and brain activations

- The affecting factors and their interactions are very complex
  - The resulting data has structure
    - But the structure is very “irregular” and involves complex relationships

- Audio and Images are sampled uniformly
  - Data or signals can be irregular too
    - Non-uniform sampling

- How do we understand the structure in such data?
  - Classical DSP assumes “regular” patterns in the data
  - Graph Signal Processing
    - Generalizes classical DSP to graphs

Outline

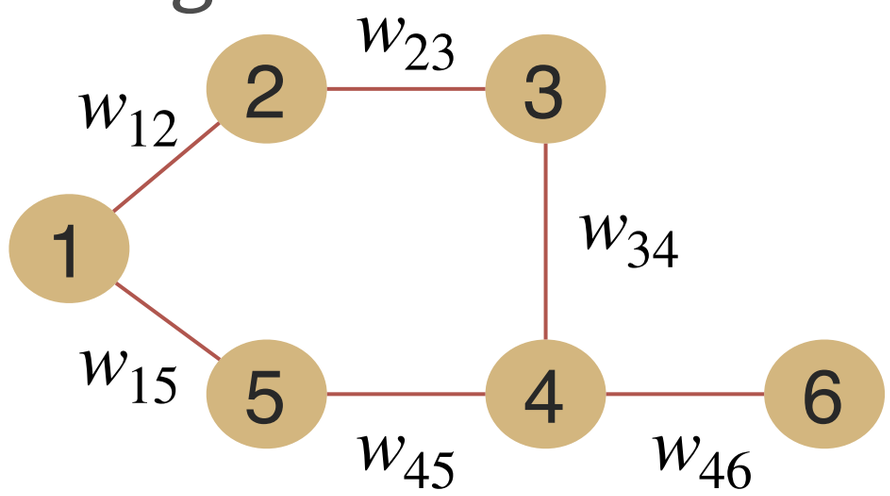
- What is a Graph
  - How to represent signals on graphs
  - How to represent graphs numerically
- Signal Processing on graphs
  - Generalizing classical DSP to graph signals
  - The Graph Fourier Transform
    - Applications: Filtering and Denoising
- What if signals don't have associated graphs?
  - Imposing a graph on signals
- Convolutions on Graphs
  - Graph Convolutional Networks (GCNs)
- Applications

A Graph

- A Graph is a set of vertices (nodes) and edges

Graph $\longrightarrow$ $G = \{V, E\}$

Vertices or Nodes — Edges

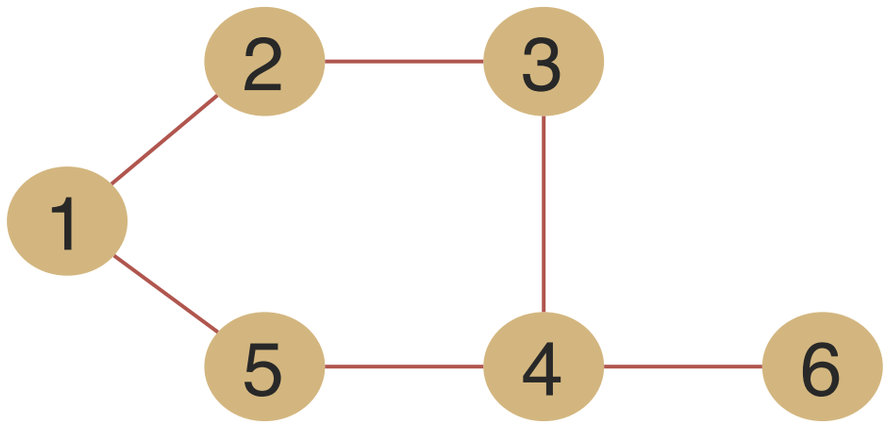


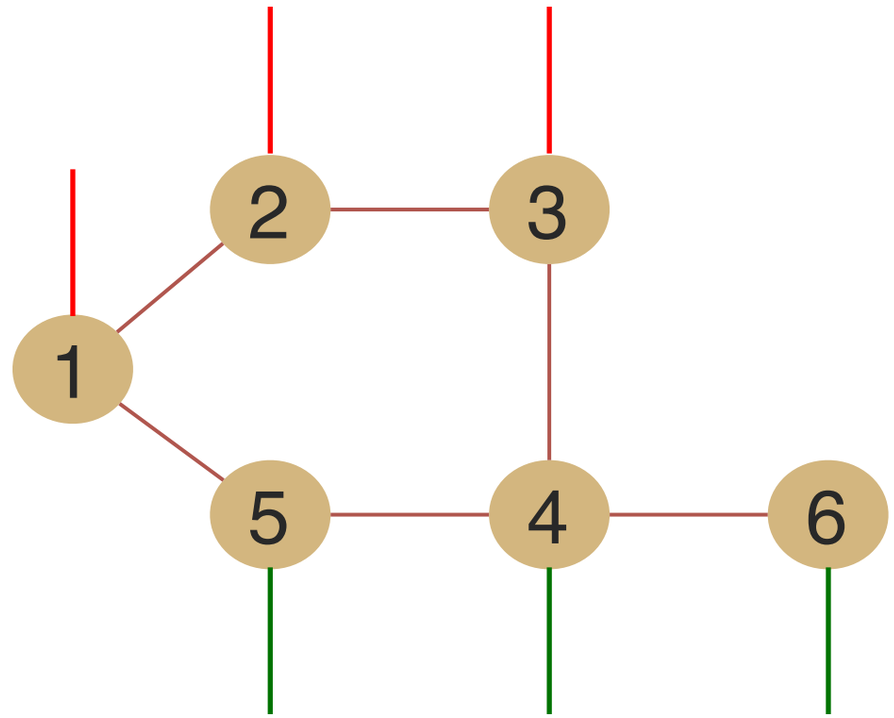
- Each node has a feature or message
  - Messages move through the nodes to distribute information
- Edges may have associated weights
  - Messages are weighted as they move through the edge
- Graphs can be directed or undirected
  - Undirected: 1 —> 2 is equivalent to 2 — > 1
  - Directed: 1 —> 2 and 2 — > 1 have different weights
- Efficient to represent pairwise relationships

Representing a signal as a graph

- Associate each sample of the signal to a node
  - Example

$$x = [\ \ 1 \quad 1 \quad 1 \quad\quad -1 \quad -1 \quad -1 \quad\quad]$$





- The signal values become node values or features
  - The sample relationships come from the graph
- Graphs are a good representation of irregular signals
  - Incorporates structure (edges) and data (values)

Graph Representations

- $G = \{V, E\}$
  - Completely define a graph

- How do we represent a graph numerically?
  - Construct a matrix $\mathbf{W}$ of size N x N
    - N is the number of vertices
  - The elements of $\mathbf{W}$ give the weights of the edges

- The Adjacency Matrix $\mathbf{W}$
  - Gives information about node connectivity & weights
  - Undirected graphs
    - Symmetric $\mathbf{W}$ (adjacency) matrix

Examples

Undirected Path Graph

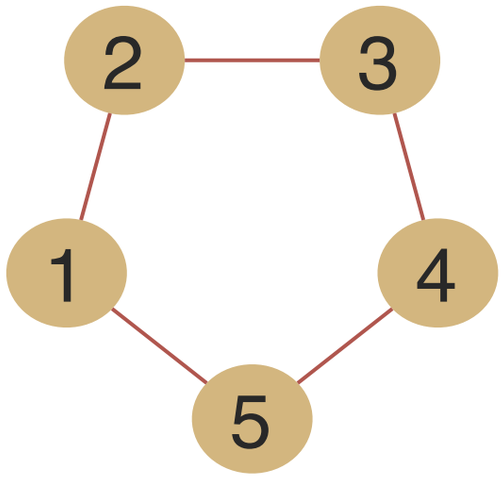
$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Directed Path Graph

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

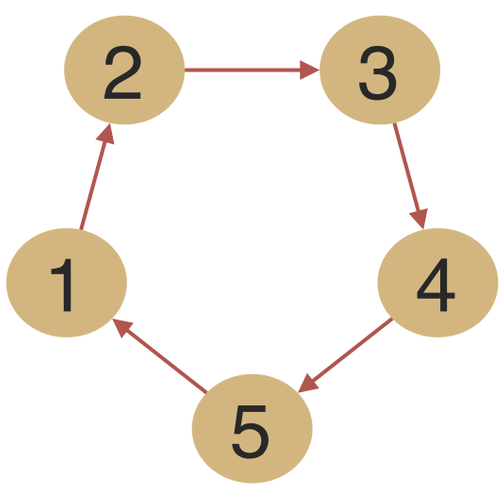
Examples

Undirected Ring Graph



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

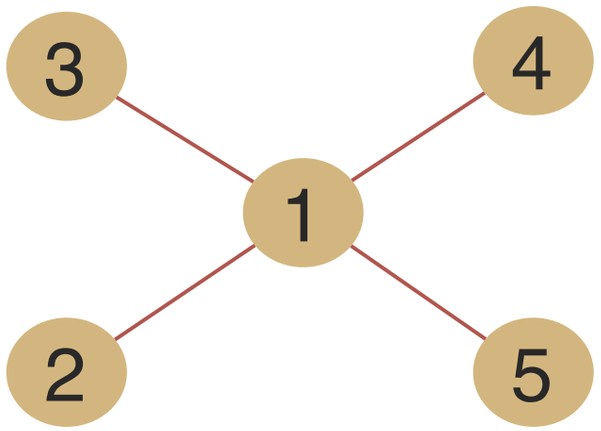
Directed Ring Graph



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

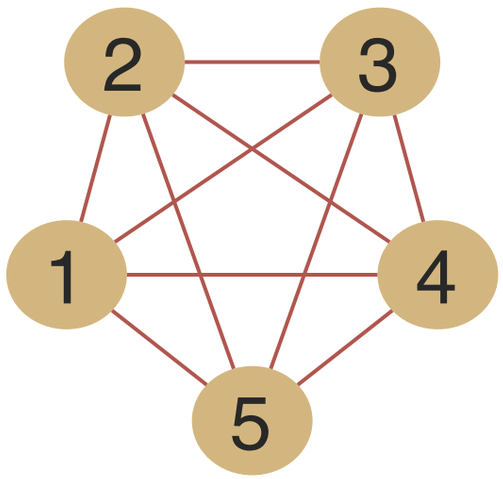
More Examples

Undirected Star Graph



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

Undirected Fully Connected Graph



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

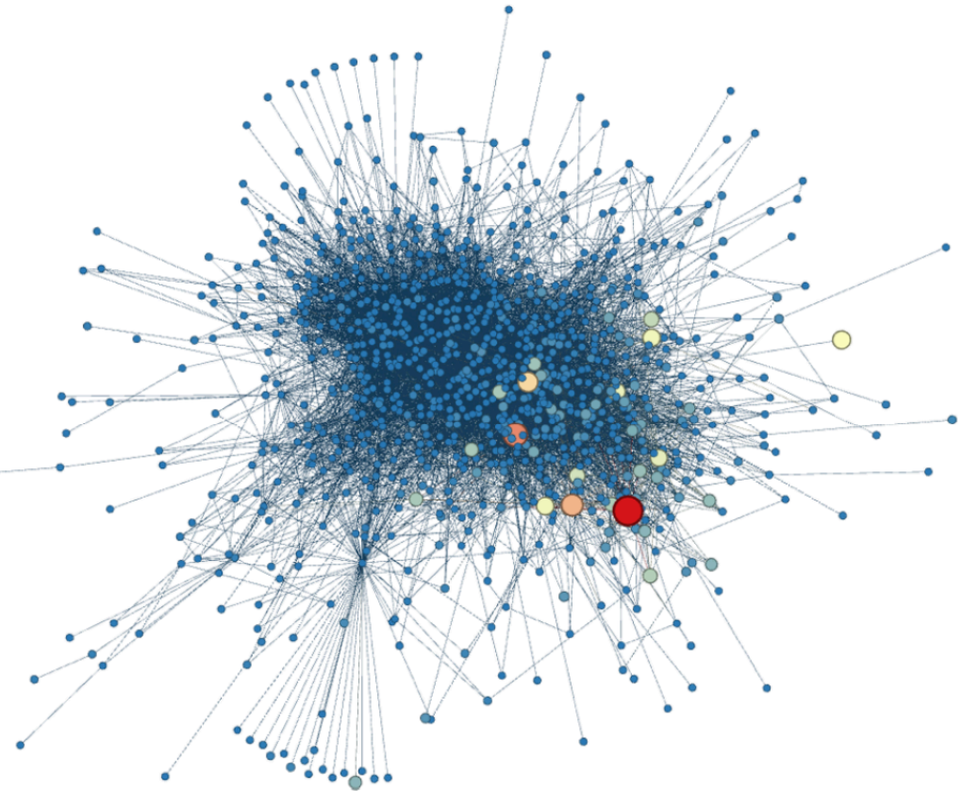
Real-world Examples

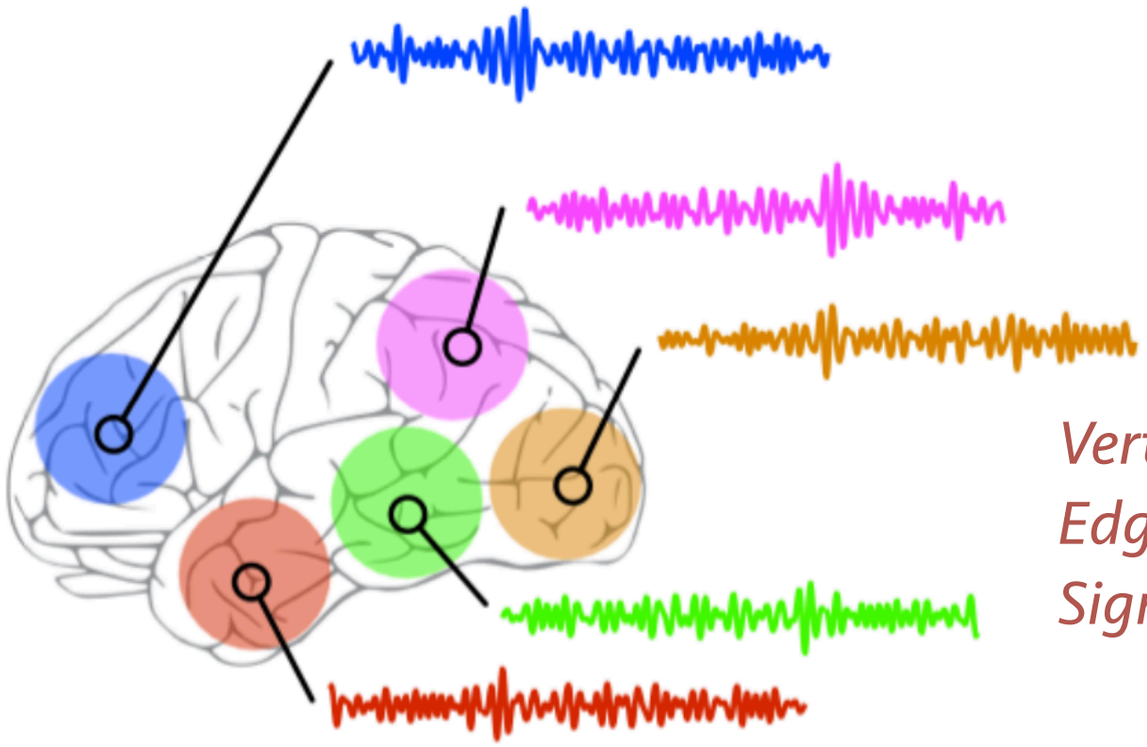
Social Network Graph

Vertices: 1000 Twitter users

Edges: Relationship between users

Signal: Apple-related hash-tags in a 6 week period





Brain Imaging

Vertices: Different brain regions

Edges: Structural connectivity between regions

Signal: Blood-oxygen-level series

Graph Signal Processing

- The road so far…
  - Graphs are a collection of vertices and edges
  - The adjacency matrix contains all the information

- How do we process such representations?
  - Graph Signal Processing

- Classical DSP assumes “regular” data
  - GSP generalizes classical DSP to graph signals

- Also helps propose Neural Nets on graphs
  - Make signal processing parameters trainable

Graph Signal Processing

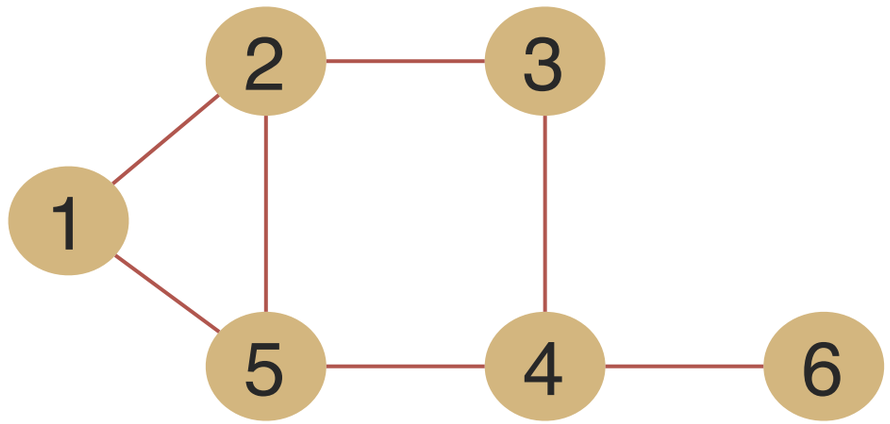
- Classical signal processing
  - Time domain
    - Natural representation using waveforms
    - Signal properties not easily visible
  - Frequency domain
    - Fourier Transforms, Cosine Transforms etc.
    - Signal properties can be easily analyzed

- Graph signal processing
  - Vertex (spatial) domain
    - Denoted using the Adjacency matrix
  - Graph Spectral domain
    - Graph Fourier Transform
    - Important for analysis of signal properties

- How do we move from spatial to spectral domain?
  - We need the Graph Laplacian matrix $\mathbf{L}$

The Graph Laplacian

- Consider an undirected graph
  - Assume that all the weights are equal to 1



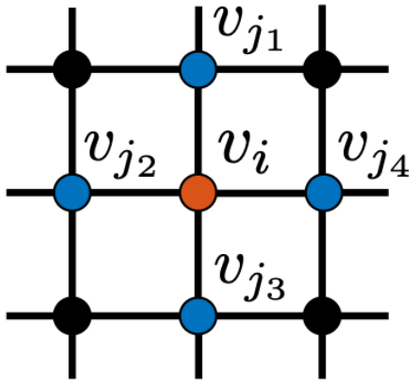
$$\begin{bmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 &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

Graph Laplacian ***Degree Matrix*** ***Graph Adjacency***

$$\mathbf{L} \qquad \mathbf{D} = \mathrm{diag}(\mathbf{W} \cdot \mathbf{1}_6) \qquad \mathbf{W}$$

The Graph Laplacian: Properties

- Approximates the point-wise Laplace operator



$$\mathbf{L}(v_i) = 4 \cdot v_i - v_{j_1} - v_{j_2} - v_{j_3} - v_{j_4}$$

- For undirected graphs, $\mathbf{L}$ is symmetric

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

- Off-diagonal entries are non-positive
- Rows add up to zero
  - This is due to the degree matrix $\mathbf{D}$
- Normalized Laplacian $\mathbf{L}_{norm} = \mathbf{D}^{-\frac{1}{2}} \cdot (\mathbf{D} - \mathbf{W}) \cdot \mathbf{D}^{-\frac{1}{2}}$
  - We will stick to using $\mathbf{L}$

The Graph Fourier Transform

- Given:

$$Graph \quad G = \{V, \quad E\}, \; signal \quad \mathbf{s}$$

- Get graph Laplacian $\mathbf{L}$
  - $\mathbf{L}$ has a complete set of orthonormal eigen-vectors
    - Eigen-values are sorted in increasing order

$$\mathbf{L} = \mathbf{V} \cdot \mathbf{\Lambda} \cdot \mathbf{V}^{-1}$$

Graph Fourier bases — $\mathbf{V}$; $\mathbf{\Lambda}$ — *Graph Fourier spectral coefficients*

- Transforming the signal

Signal Fourier coefficients $\longrightarrow \mathbf{S} = \mathbf{V}^{-1} \cdot \mathbf{s} \longleftarrow$ *Signal spatial coefficients*

Graph Fourier Transform — $\mathbf{V}^{-1}$

What does this mean?

- Eigen decomposition as an optimization

$$max \quad \mathbf{x}^{\mathbf{T}} \cdot \mathbf{L} \cdot \mathbf{x} \quad s.t \quad \mathbf{x}^{\mathbf{T}} \cdot \mathbf{x} = 1$$

$$\mathbf{x}^{\mathbf{T}} \cdot \mathbf{L} \cdot \mathbf{x} = \frac{1}{2} \cdot \sum_{i,j=1}^{N} w_{ij} \cdot (\mathbf{x}_i - \mathbf{x}_j)^2$$

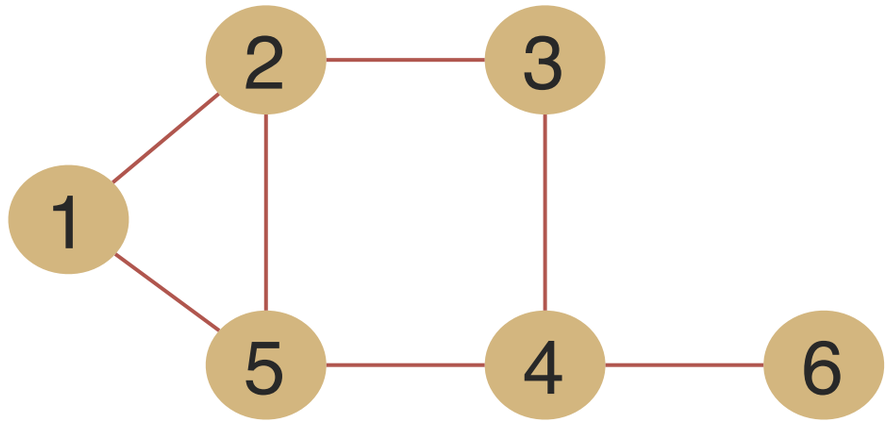
Adjacency matrix elements

Local Variation

- Bases capture "local variation"
  - "How much" does a signal change in a neighborhood
  - Lower value — smoother signal

Example

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



$\mathbf{x}$	$\frac{\mathbf{x}^T \cdot \mathbf{L} \cdot \mathbf{x}}{\mathbf{x}^T \cdot \mathbf{x}}$
[1, 1, 1, 1, 1, 1]	0
[0, 0, 1, 1, 0, 0]	3
[1, 0, 1, 0, 1, 0]	5

Higher the eigen-value, higher the local variation in the corresponding eigen-vector

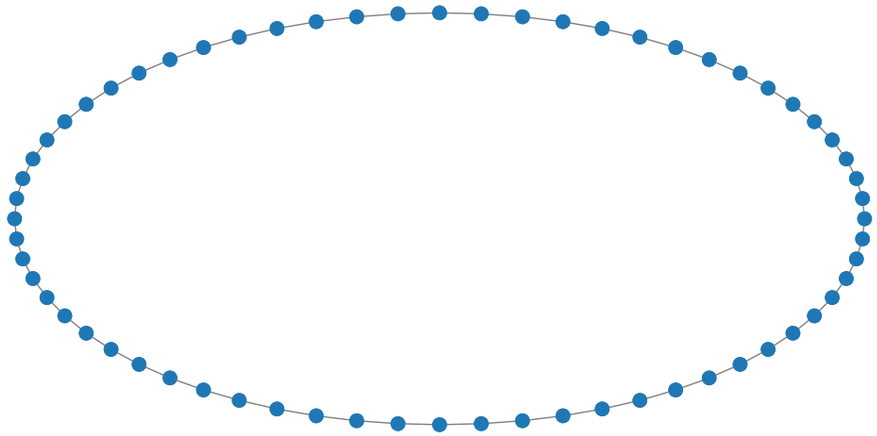
A notion of "Frequency"

- The eigen-values give the spectral coefficients
- Eigen-vectors are ranked by their eigen-values
  - The initial eigen-vectors are smooth and change slowly
  - The later eigen-vectors change quickly
  - Similar to frequency in classical DSP

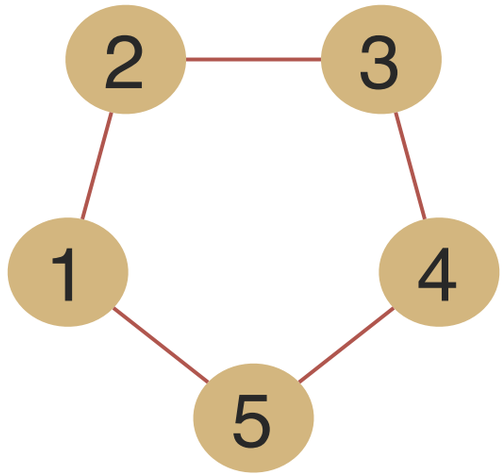
Random Sensor Network

Special Case I: DFT

- Consider an undirected Ring graph
  - Assume all the weights are equal to unity



Ring graph
64 Vertices



Ring graph
5 Vertices

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

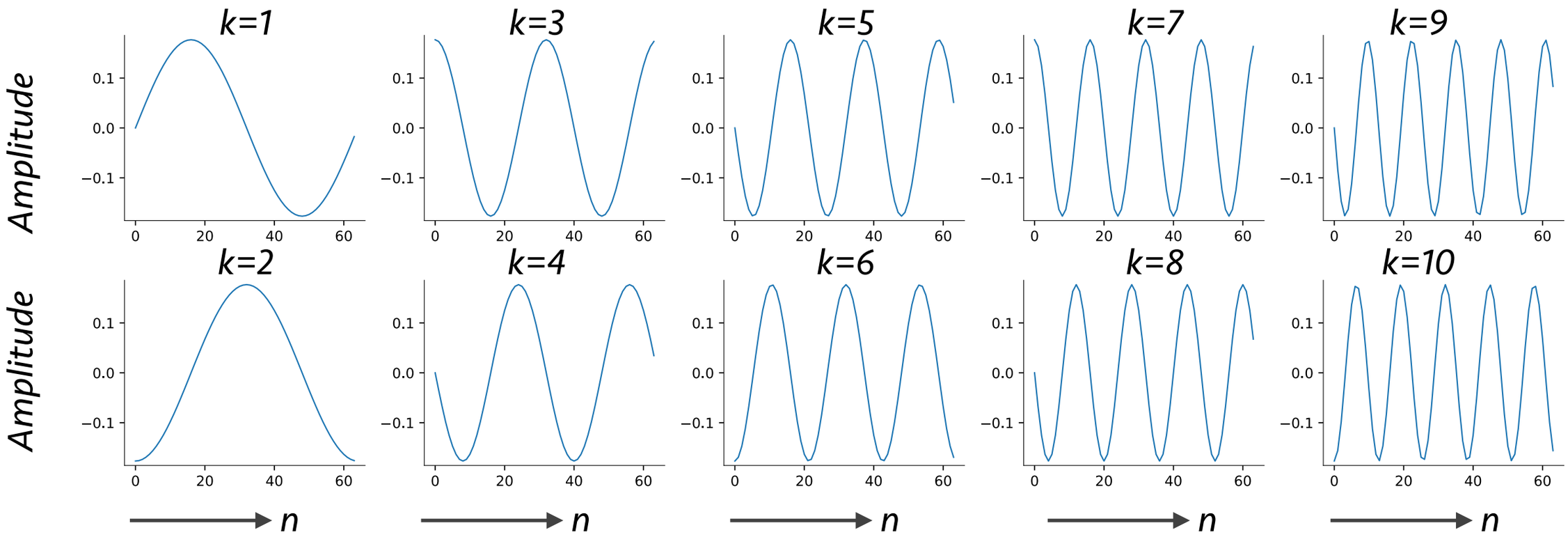
Adjacency
5 Vertices

- The adjacency matrix "Circulant"
  - We can define an orthonormal sinusoidal basis

Special Case I: DFT

- One set of orthonormal eigen-vectors

$$\left[\cos\left(\frac{2\pi kn}{N}\right)\right]_{n=0}^{N-1} \quad , \quad \left[\sin\left(\frac{2\pi kn}{N}\right)\right]_{n=0}^{N-1} \quad \forall \quad k \in \{0,1,\ldots,\frac{N}{2}-1\}$$



- Paired sine and cosine terms
  - Have the same eigen-values
  - Similar local variation

Special Case I: DFT

- We can group these paired eigen-vectors
  - Use as real and imaginary parts of complex numbers

$$\left[\cos\left(\frac{2\pi kn}{N}\right)\right]_{n=0}^{N-1} \quad , \quad \left[\sin\left(\frac{2\pi kn}{N}\right)\right]_{n=0}^{N-1} \quad \forall \quad k \in \{0,1,\ldots,\frac{N}{2}-1\}$$

- Another orthonormal set of eigen-vectors

Discrete Fourier Transform bases $\longrightarrow$ $\left[\exp\left(j\frac{2\pi kn}{N}\right)\right]_{n=0}^{N-1} \quad , \forall \quad k \in \{0,1,\ldots,N-1\}$

- Symmetry properties
  - Undirected graphs have symmetric $\mathbf{W}$ and $\mathbf{L}$ matrices
    - Eigen-values (spectral coefficients) will be real
    - "Symmetry — Real" duality for graphs

Special Case II: DCT

- Consider an undirected Path graph
  - Assume all the weights are equal to unity

Path graph
5 Vertices

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

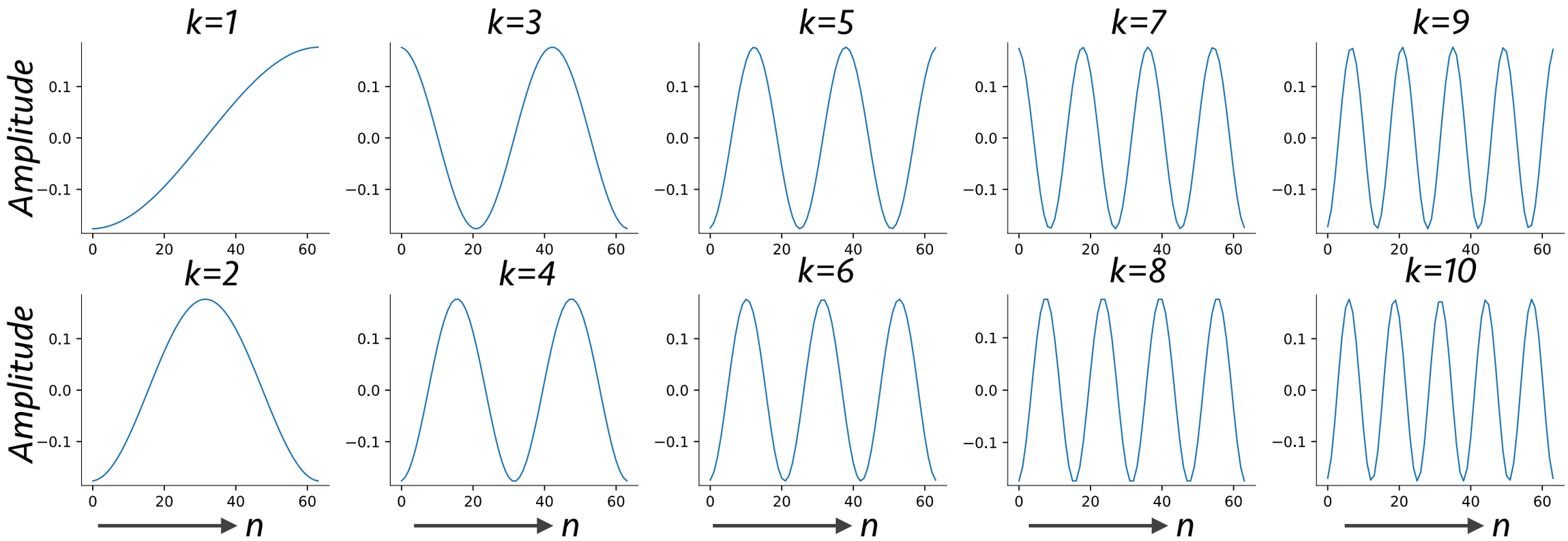
Adjacency
5 Vertices

- The adjacency matrix "Toeplitz"
  - As before we can define eigen-vectors to be sinusoidal

Special Case II: DCT

- One set of orthonormal eigen-vectors

$$\left[\sqrt{\frac{2}{N}} \cdot \cos\left(\frac{\pi k(n-0.5)}{N}\right)\right]_{n=0}^{N-1} \quad \forall \quad k \in \{0,1,\ldots,N-1\}$$



- Type 2 Discrete Cosine Transform
  - Real alternative to Fourier Transforms
  - Used for lossy JPEG image compression

Special Case III: PCA

- We can use other matrices in place of W (impose a graph)
  - The covariance matrix has similar properties too
    - Eigen-decomposition leads to PCA

$$cov(\mathbf{X}) = \frac{1}{N} \cdot \mathbf{X} \cdot \mathbf{X}^T$$

- Graph Fourier Transforms
  - Generalizes classical Fourier transforms
  - DFT when we assume a Ring graph
  - DCT when we assume a Path graph
  - PCA when we assume a graph based on Covariance

- We can use different adjacency matrices
  - Capture different characteristics in spectral domain

Imposing a Graph

- The road so far…
  - Graphs are a collection of nodes and edges
    - Graphs are defined by their adjacency matrices $\mathbf{W}$
  - The graph Laplacian $\mathbf{L}$ can be computed from $\mathbf{W}$
    - Used to compute the Graph Fourier Transform
    - Generalizes classical Fourier Transform theory

- We assumed that the data lived on an underlying graph
  - Can we learn a graph from the data itself?
  - Can audio / images be interpreted as a graph?

- Learning a Graph = Learning an adjacency matrix $\mathbf{W}$
  - Easy to learn adjacency matrices from the data
    - Use a suitable distance metric to get $\mathbf{W}$

Imposing a Graph

- An example:
  - Adjacency based on spatial location

$$\mathbf{w}_s(i,j) = \exp(-\alpha \cdot |s_i - s_j|)$$

    - Can be time-indices or spatial co-ordinates

  - Adjacency based on sample values

$$\mathbf{w}_v(i,j) = \exp(-\beta \cdot |v_i - v_j|)$$

    - Can be features, sample-values, image colors etc.

  - Overall Adjacency matrix

$$\mathbf{W} = \gamma \cdot \mathbf{W}_v + (1-\gamma) \cdot \mathbf{W}_s$$

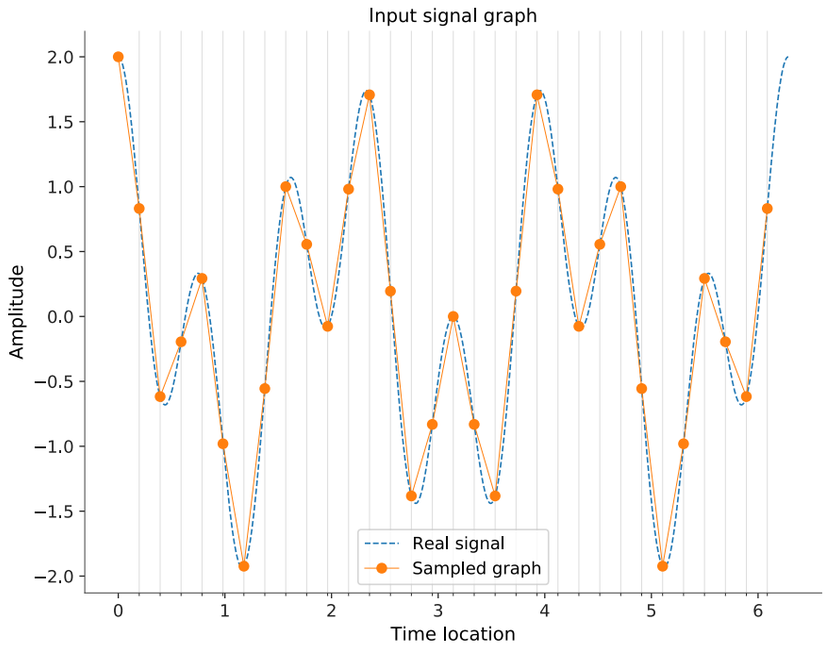
    - For undirected graphs, ensure symmetry
    - Scale the two adjacencies before summing

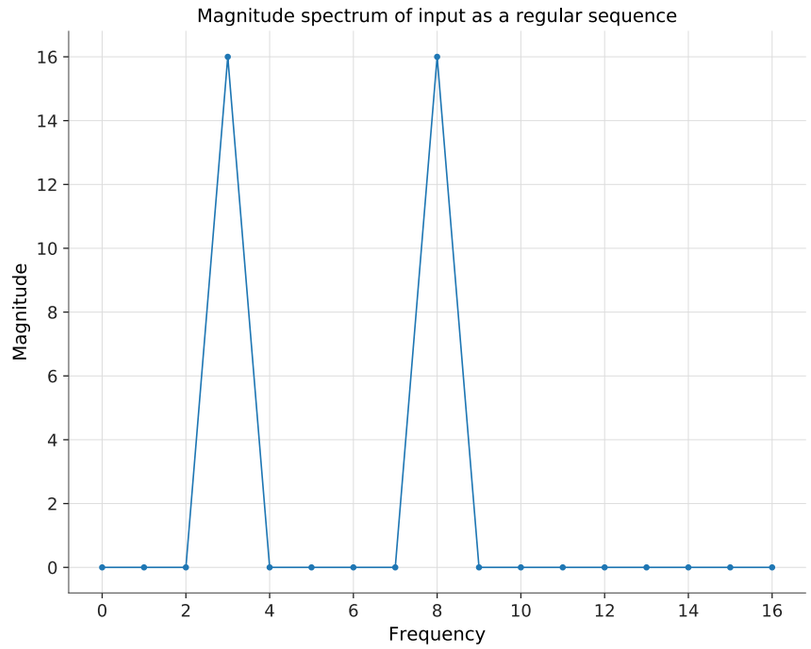
Filtering

- Filtering in classical signal processing

$$x(t) = \cos(3 \cdot t) + \cos(8 \cdot t)$$

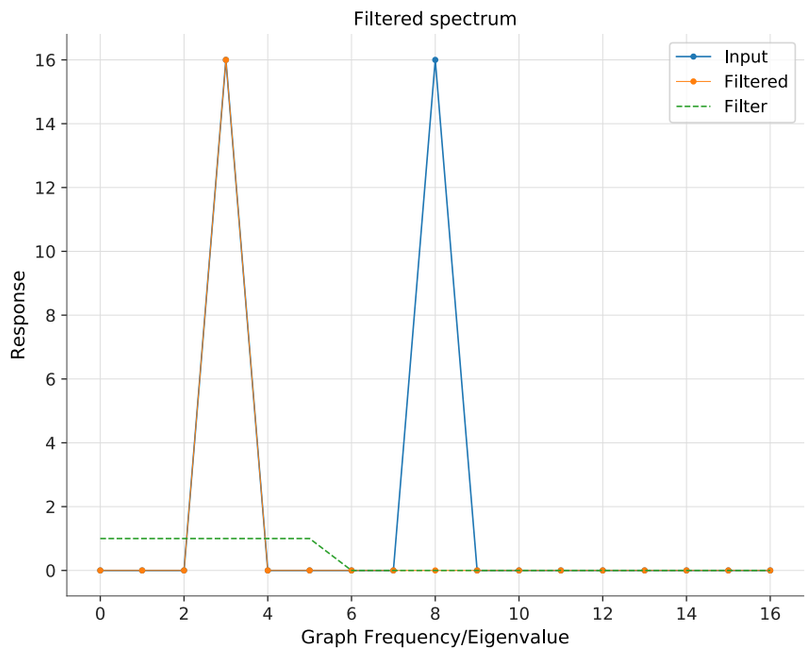
Transform signal into frequency domain

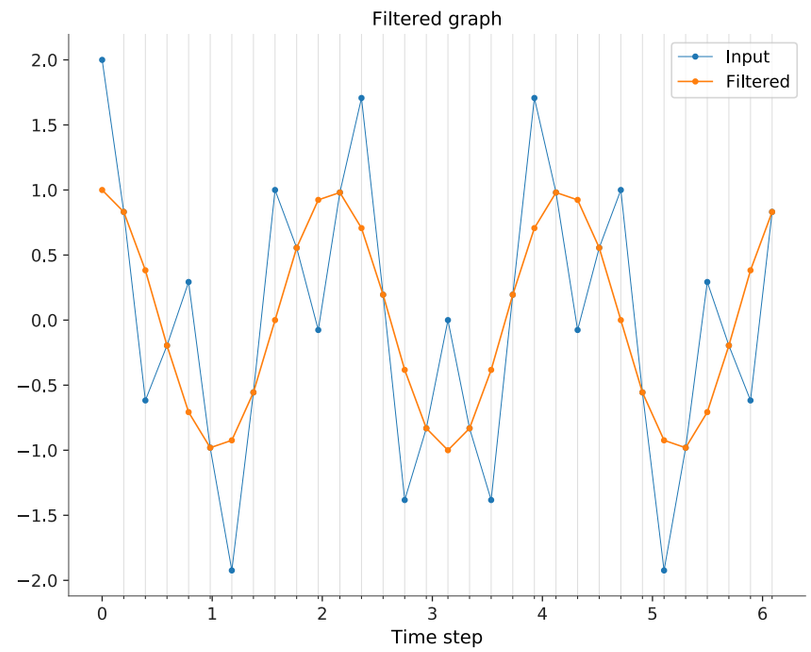




Design filter and apply

Invert to time-domain

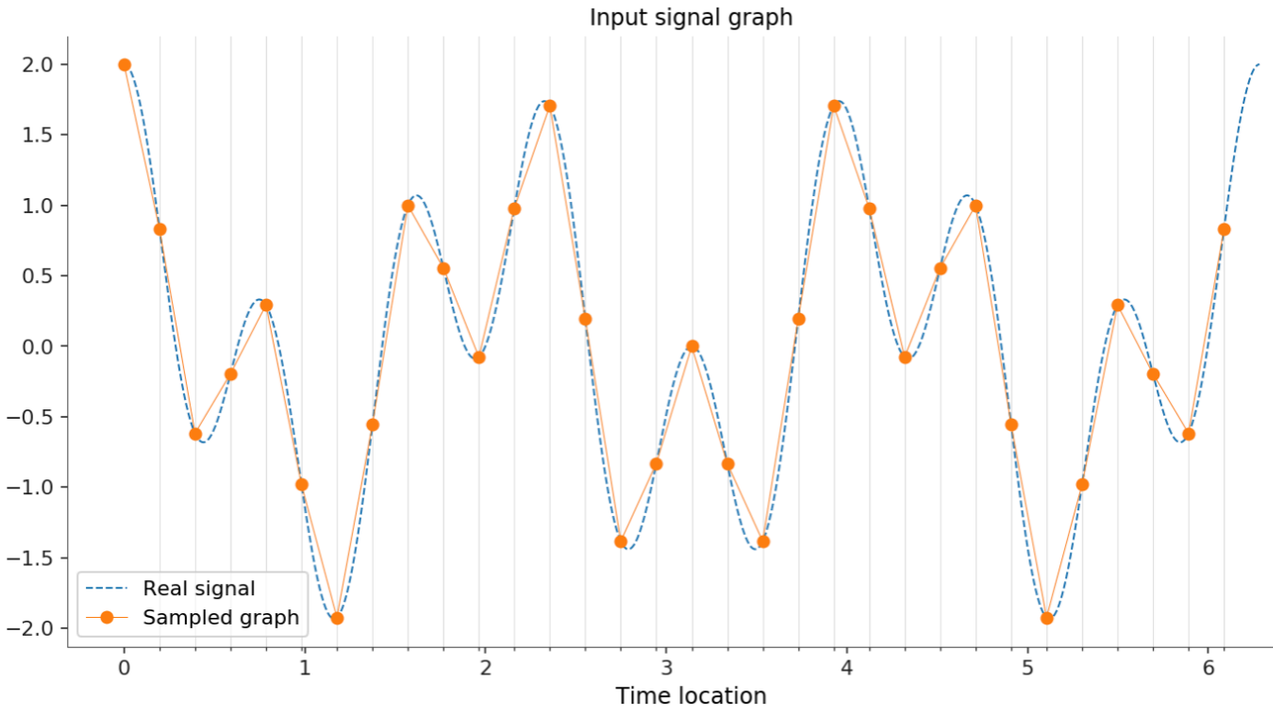




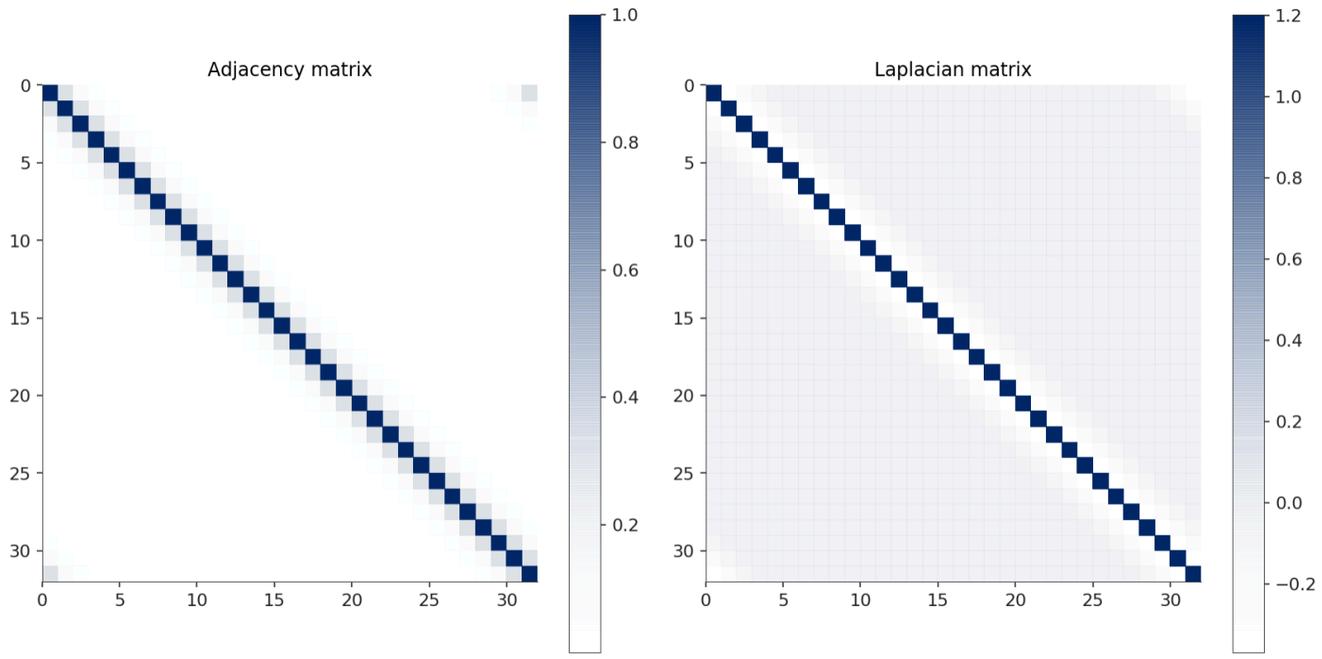
Filtering on Graphs

- Given a signal vector x

$$x(t) = \cos(3 \cdot t) + \cos(8 \cdot t)$$



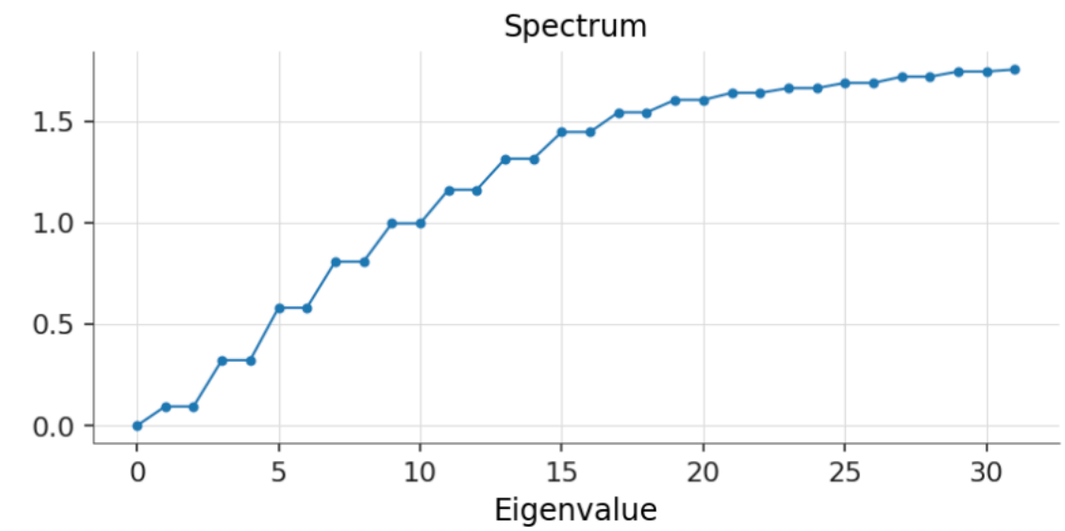
Compute Adjacency and Laplacian Matrices

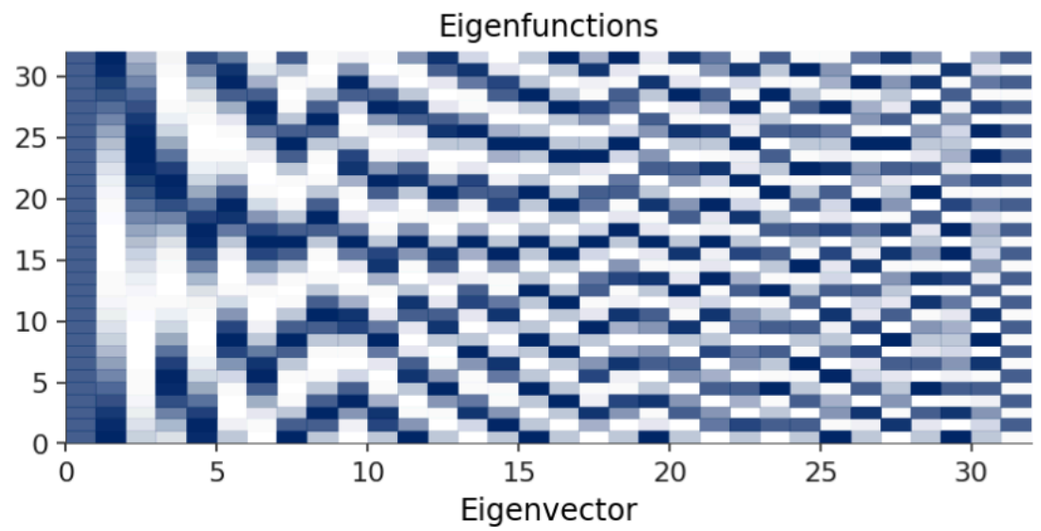


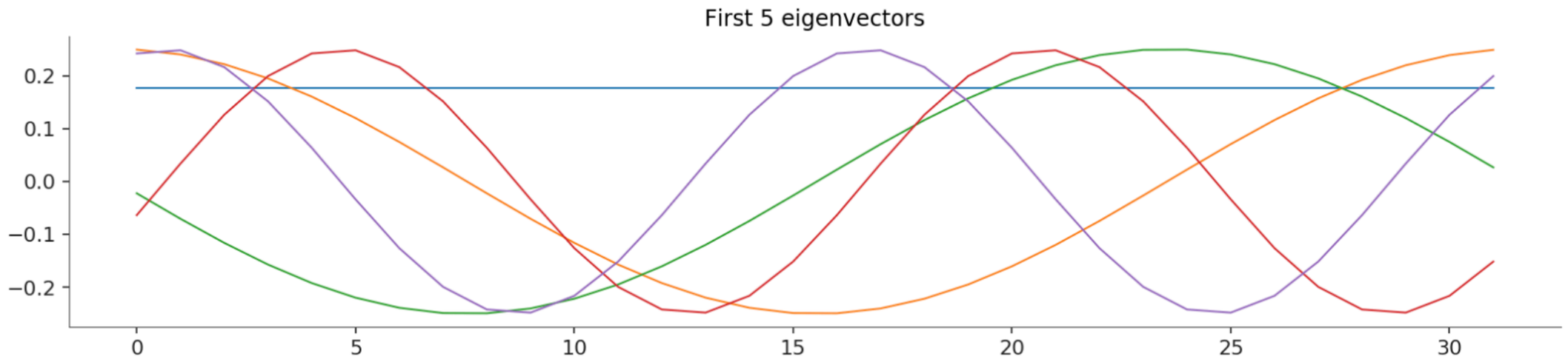
Step 1: Get Fourier Bases

- Eigen decomposition of the Graph Laplacian
  - Learn a suitable Graph Fourier transform

$$\mathbf{L} = \mathbf{V} \cdot \mathbf{\Lambda} \cdot \mathbf{V}^T$$







Step 2: Transform Data

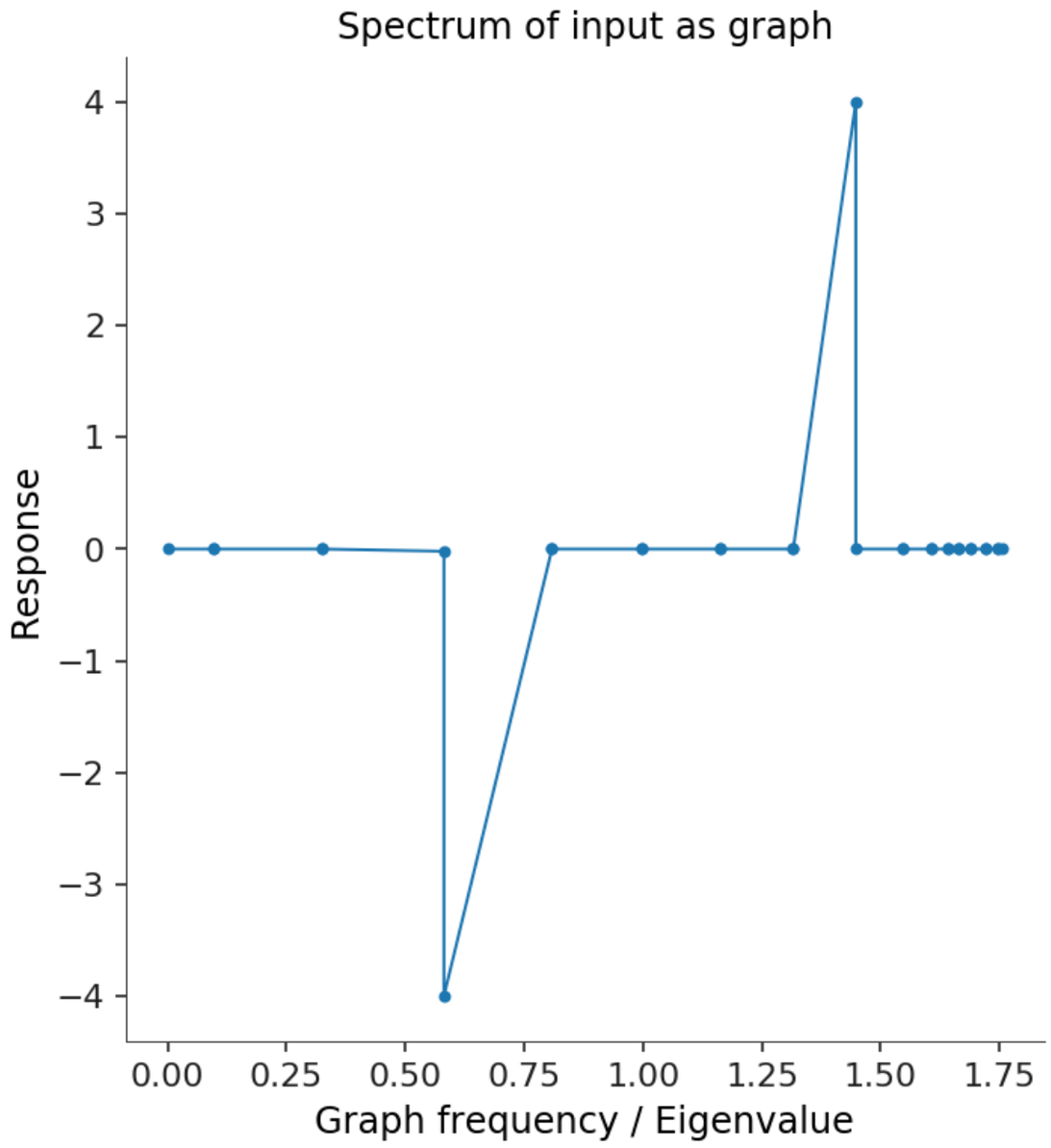
- Transform data into Fourier Domain
  - Use the eigen-vectors learned in Step 1

Signal Fourier coefficients

$$\mathbf{X} = \mathbf{V}^T \cdot \mathbf{x}$$

Graph Fourier bases

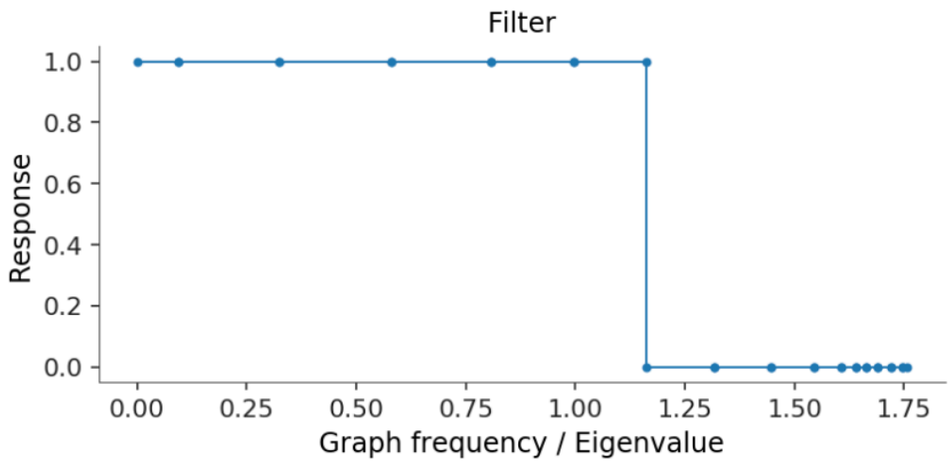
Graph Signal

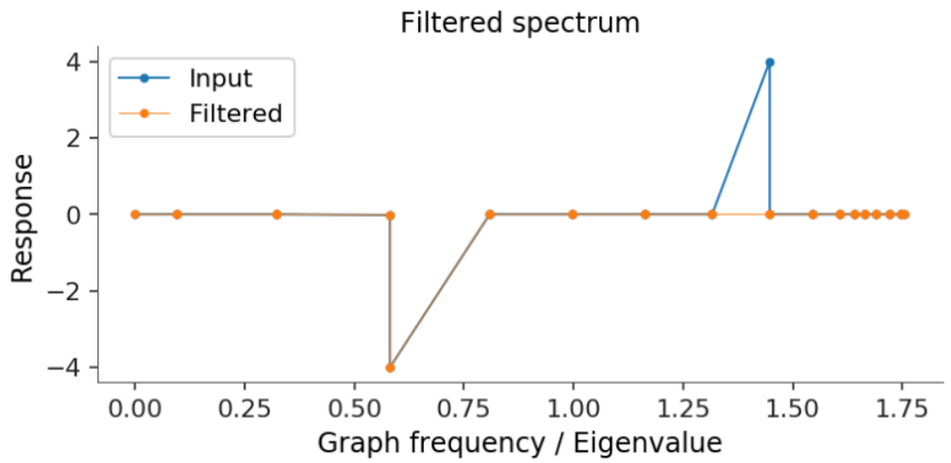


Step 3: Filter and Invert

- Design a low-pass filter
  - Multiply element-wise with signal in Fourier domain

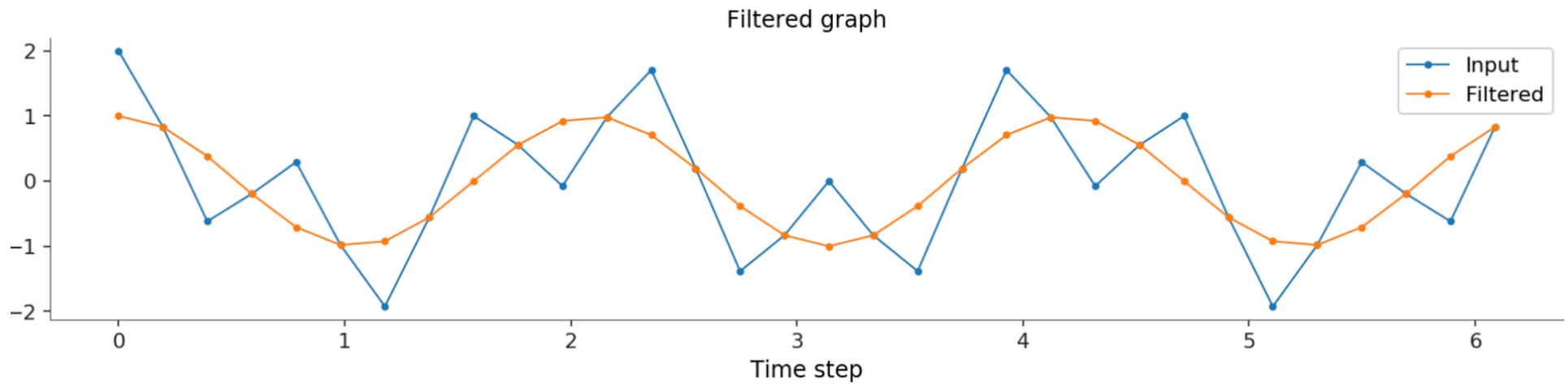
$$\mathbf{Z} = \mathbf{F} \odot \mathbf{X}$$





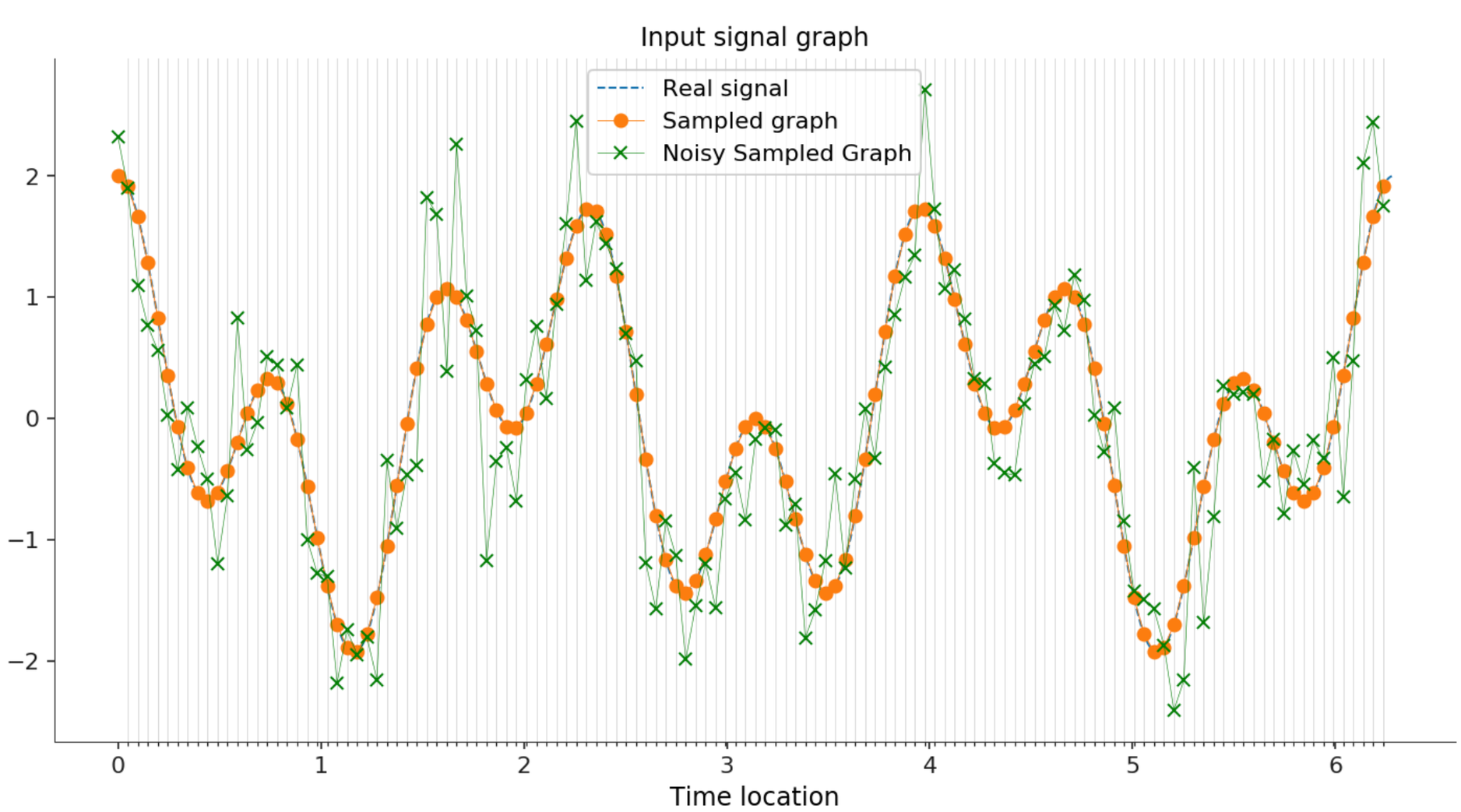
- Invert using Fourier bases

$$\mathbf{z} = \mathbf{V} \cdot \mathbf{Z}$$



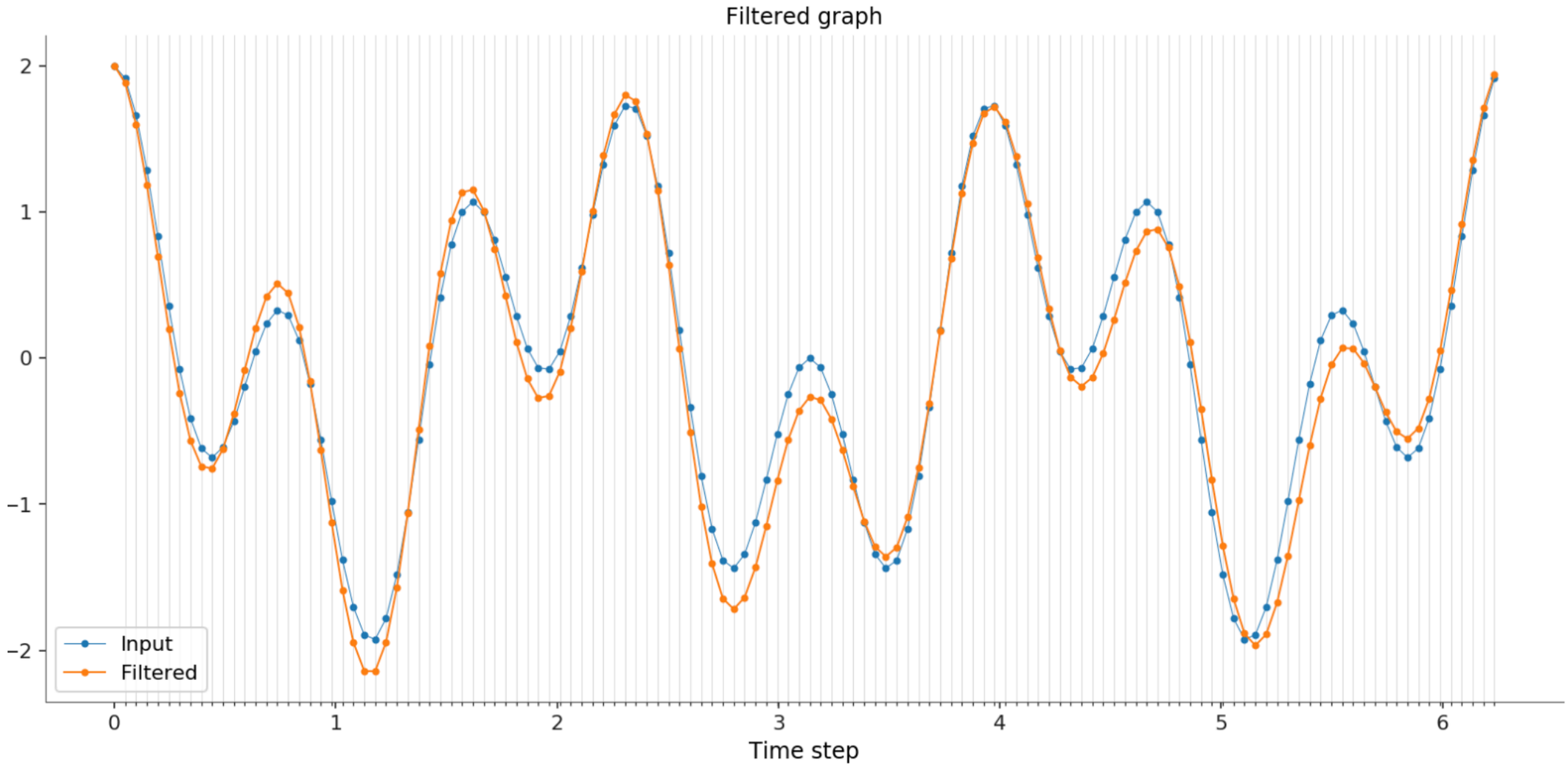
Denoising

- Consider a noisy signal
  - Can we recover the clean signal
- Noise typically has a higher local variation
  - Signal and noise could be separable in Fourier domain



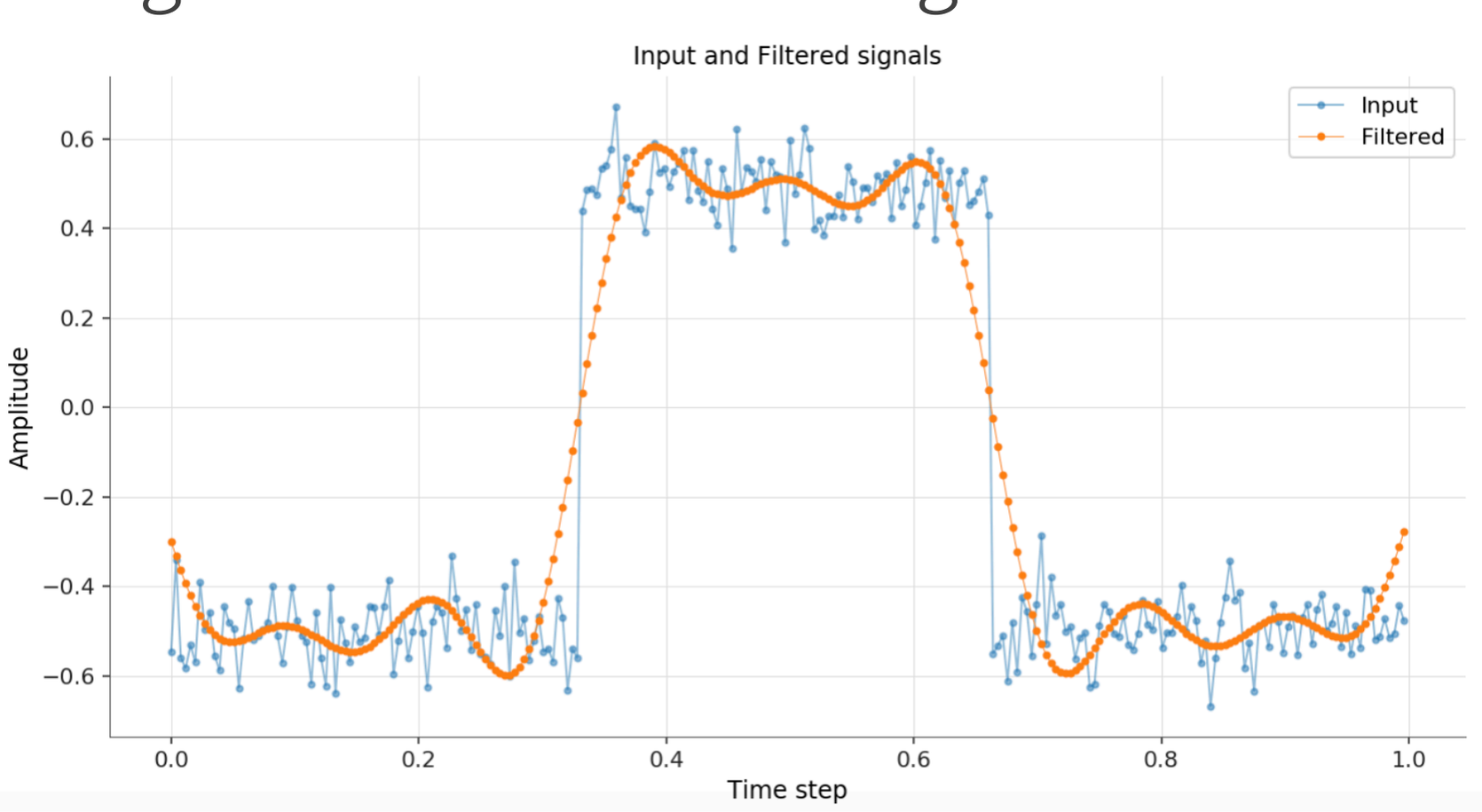
Denoising

- Steps
  - Transform signal into Fourier Domain
  - Design and apply a low-pass filter
  - Invert back into the spatial domain



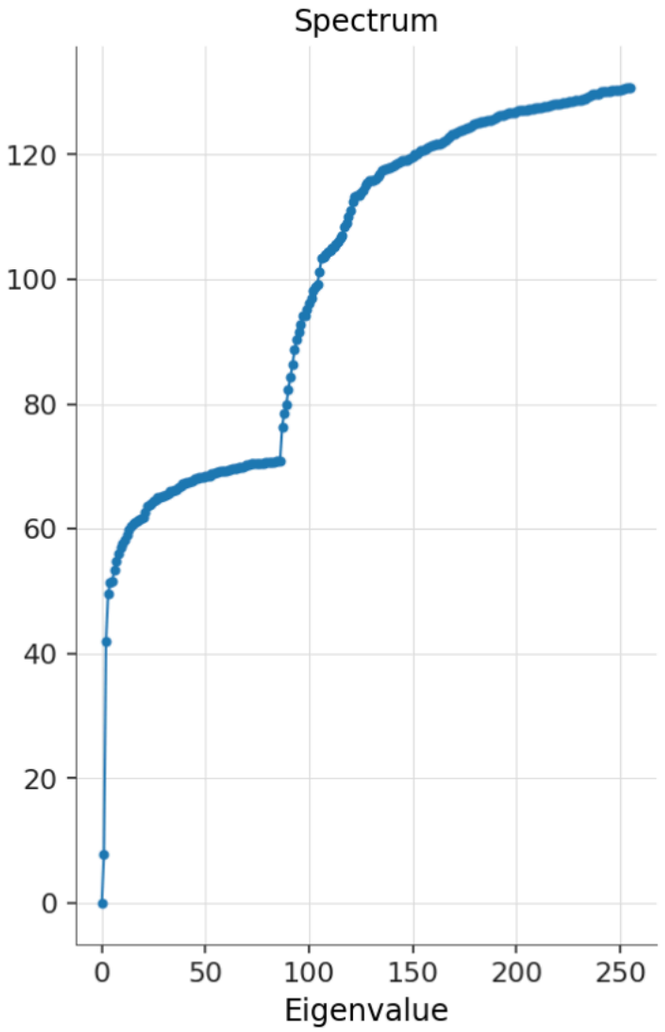
Edge preserving filters

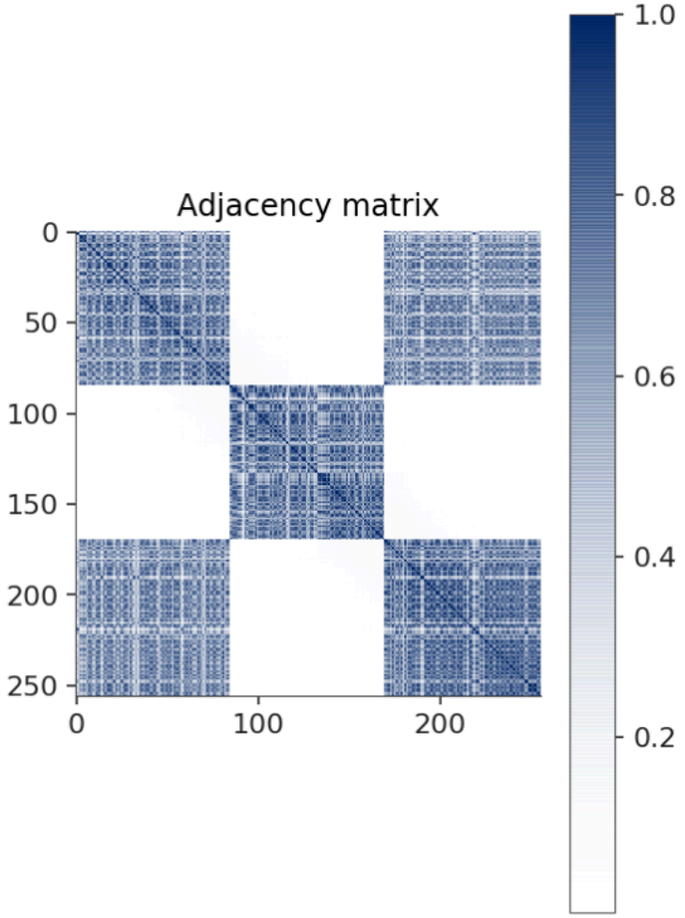
- Consider a noisy input with steep edges
  - Denoise the signal without affecting the edges
- Classical DSP does not account for structure
  - Edges are treated as high-frequency components
  - Filtering smoothes out the edges

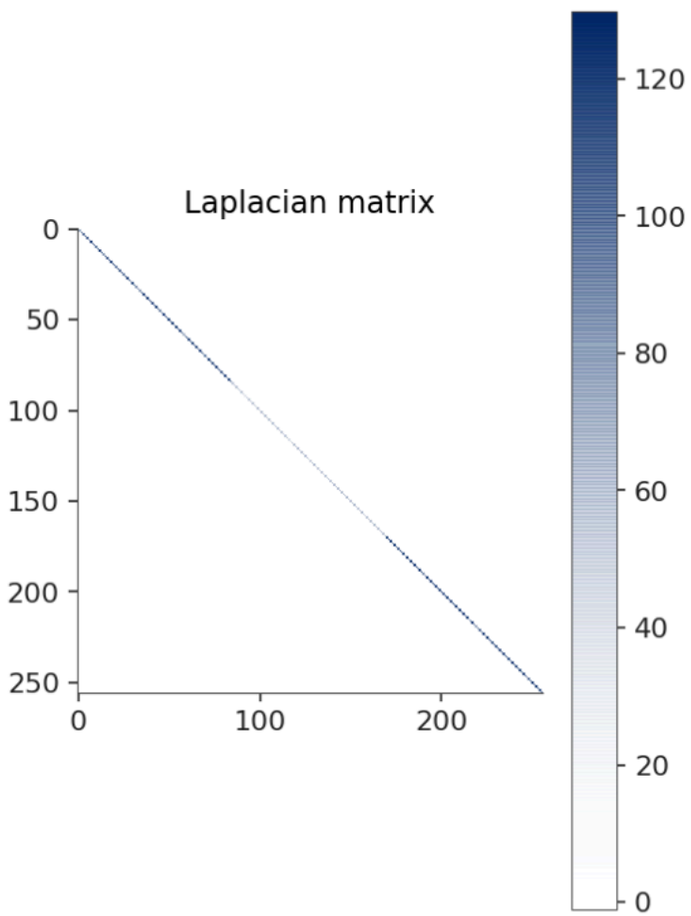


Edge preserving filters

- The signal is quantized
  - The values are more important than spatial location
    - Assign a higher weight to value based adjacency
- Representation understands structure

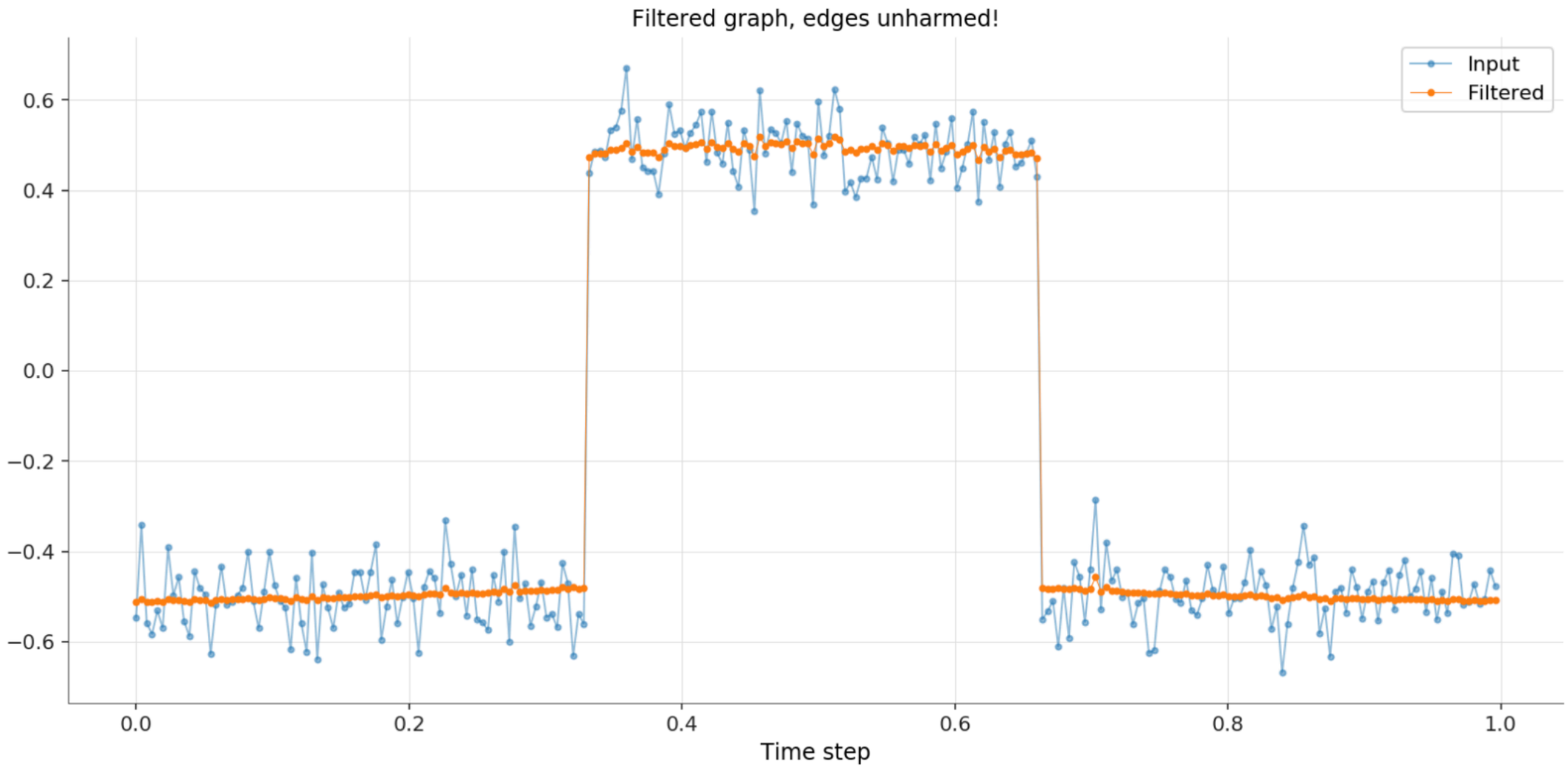






Edge preserving filters

- Noise has a higher local variation
  - Signals and Noise are easily separable
    - Graphs allow additional modeling flexibility

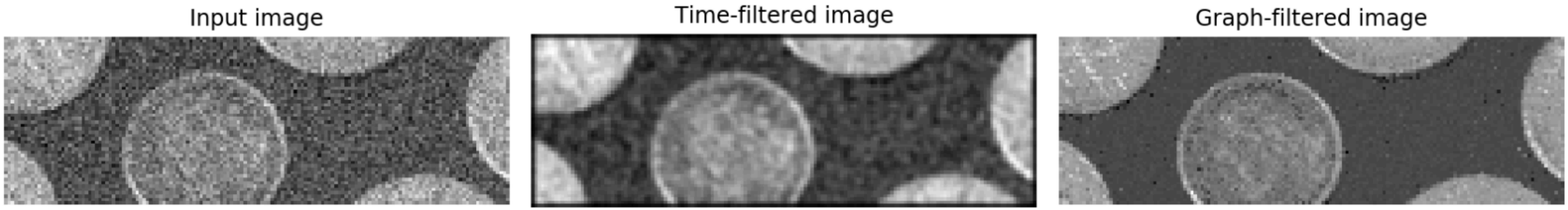


Edge preserving filters: 2D Example

- Consider a noisy image
  - Sample value is given by pixel intensity
  - Spatial location is given by pixel co-ordinates

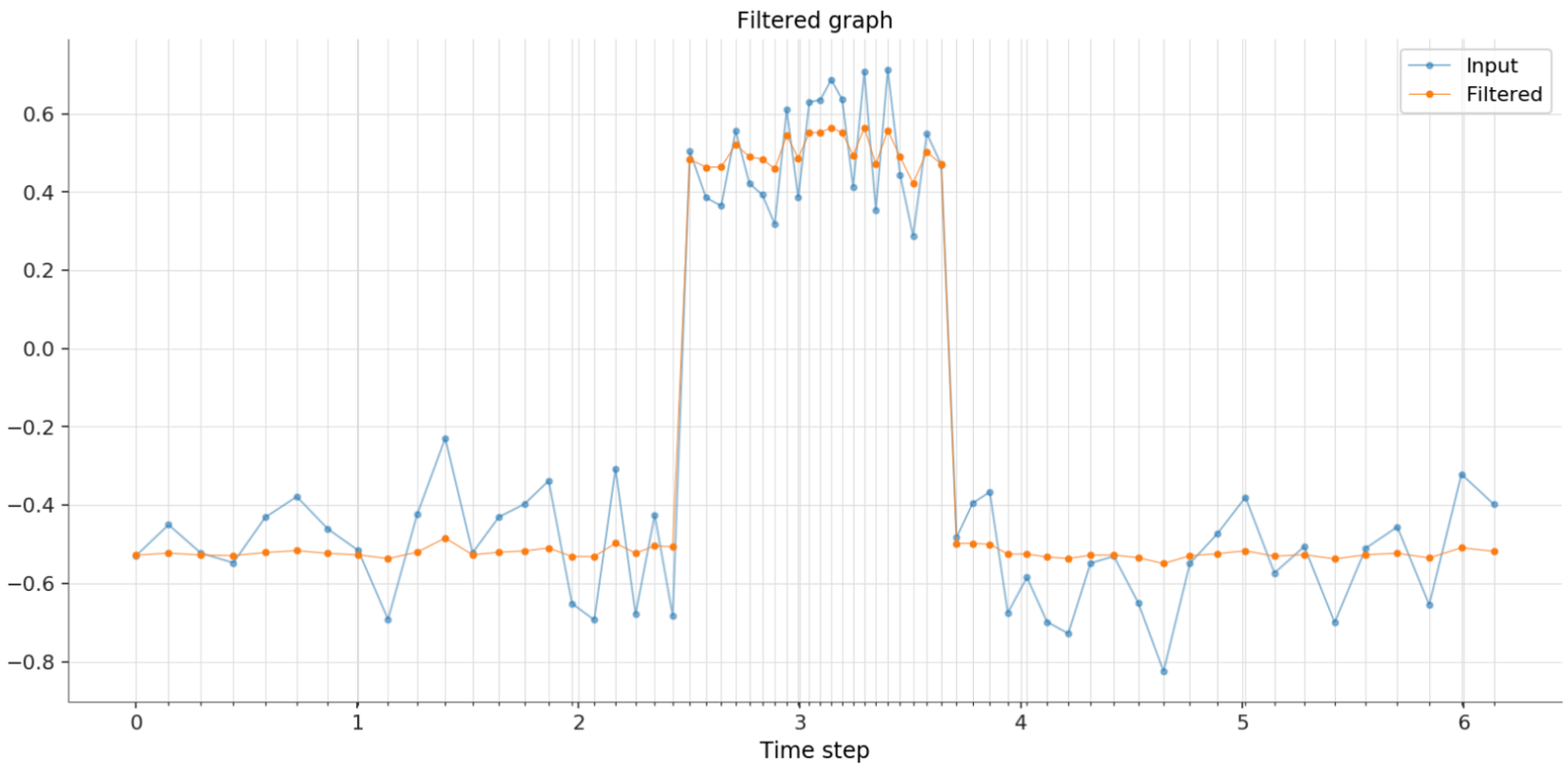
- Vectorize the samples and spatial locations
  - We can use the vec() operator
  - Everything else stays the same

- As before, spatial filtering smoothes the edges



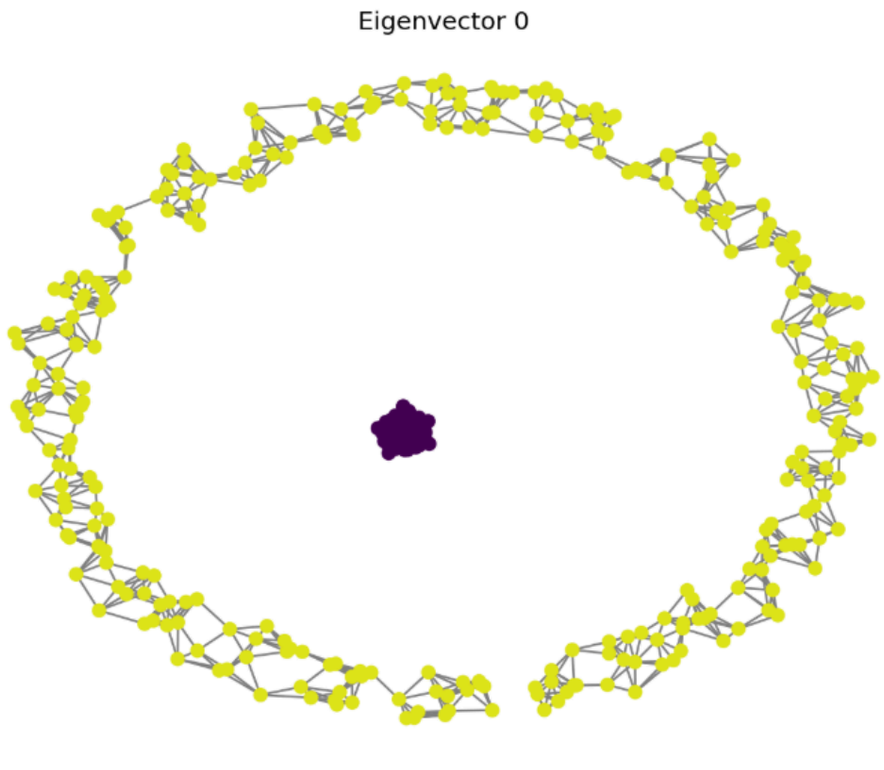
Handling Data Irregularities

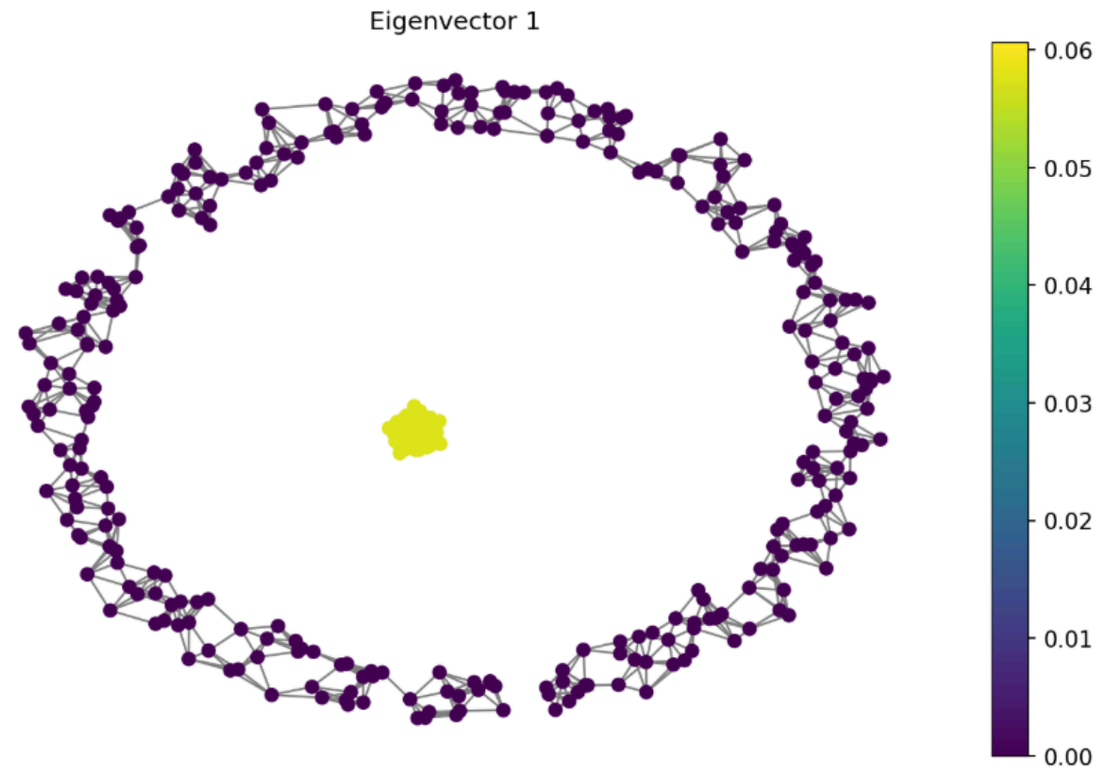
- Graphs interpret data as a cloud of points
  - Data can have a continuous range of values
  - Data can be positioned randomly in space
- Irregularly sampled data
  - Classical DSP does not have the tools
  - Graphs can naturally handle such signals



Using KNN Graphs

- K-Nearest Neighbor graphs
  - Alternative way to impose a graph on data
  - Every vertex is connected to only K of its closest points
- Fourier Transforms on KNN graphs
  - Eigen decomposition on KNN adjacency matrix
  - Spectral Clustering: ISOMAPs



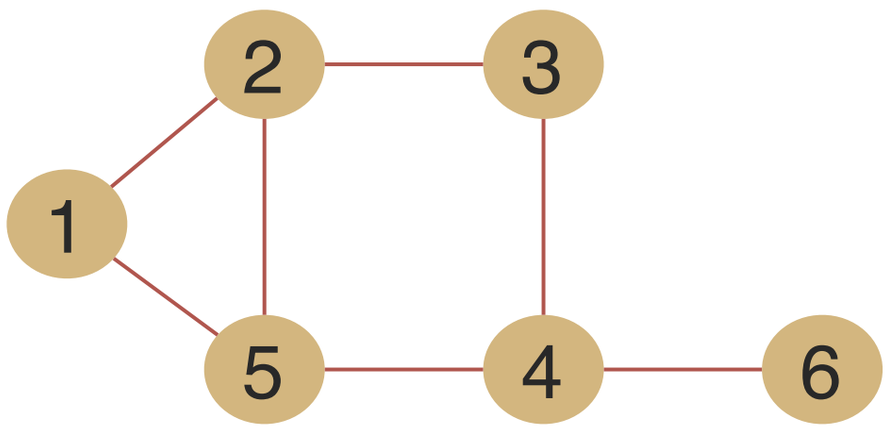


Filtering in the spatial domain

- Classical DSP: filtering in time vs frequency
  - Frequency: Multiplication
  - Time: Convolution
  - Can we have such interpretations for graphs

- Filter outputs are weighted sums of time-shifted inputs
  - We use the delay operator to define filters
    - Shifts a sequence by 1 sample

- The graph-shift matrix
  - Non-zero on diagonal and connected edges
  - Examples: Adjacency, Degree, Laplacian



$$S = \begin{bmatrix} S_{11} & S_{12} & 0 & 0 & S_{15} & 0 \\ S_{21} & S_{22} & S_{23} & 0 & S_{25} & 0 \\ 0 & S_{32} & S_{33} & S_{34} & 0 & 0 \\ 0 & 0 & S_{43} & S_{44} & S_{45} & S_{46} \\ S_{51} & S_{52} & 0 & S_{54} & S_{55} & 0 \\ 0 & 0 & 0 & S_{64} & 0 & S_{66} \end{bmatrix}$$

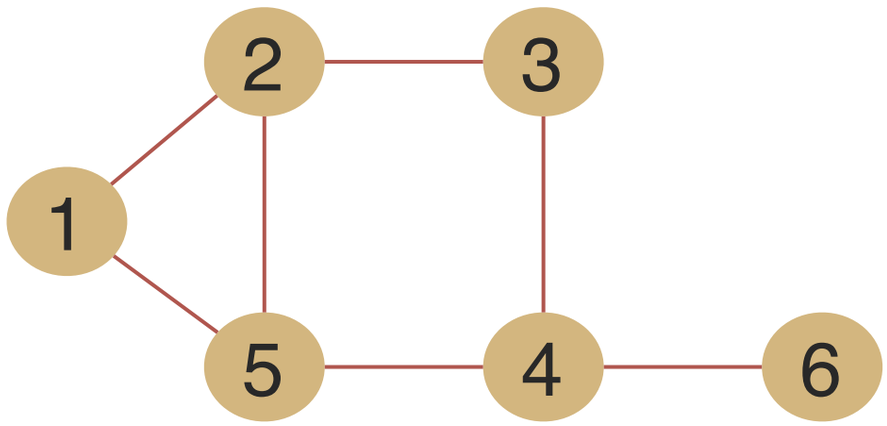
Filtering in the spatial domain

- The graph shift can be computed locally
  - $\mathbf{S} \cdot \mathbf{x}$ can be computed from current and 1-hop nodes
  - $\mathbf{S}^2 \cdot \mathbf{x}$ require all 2-hop nodes only
  - Used to define filters

- Example

$\mathbf{x} = [-1,2,0,0,0,0]^T$, $\mathbf{h} = [1,1,0.5]^T$, $\mathbf{S} = \mathbf{W}$ (Adjacency matrix)

$$\mathbf{y} = h[0] \cdot \mathbf{x} + h[1] \cdot \mathbf{S} \cdot \mathbf{x} + h[2] \cdot \mathbf{S}^2 \cdot \mathbf{x}$$

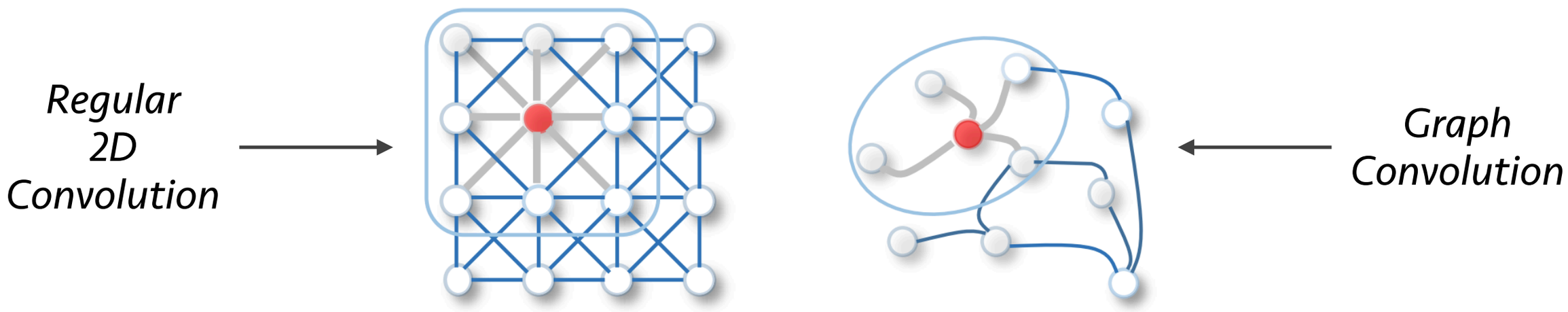


$$\mathbf{x} = \begin{bmatrix} -1 \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{S} \cdot \mathbf{x} = \begin{bmatrix} 2 \\ -1 \\ 2 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{S}^2 \cdot \mathbf{x} = \begin{bmatrix} 0 \\ 5 \\ -1 \\ 3 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 1 \\ 3.5 \\ 1.5 \\ 1.5 \\ 1.5 \\ 0 \end{bmatrix}$$

Filters to Neural Networks

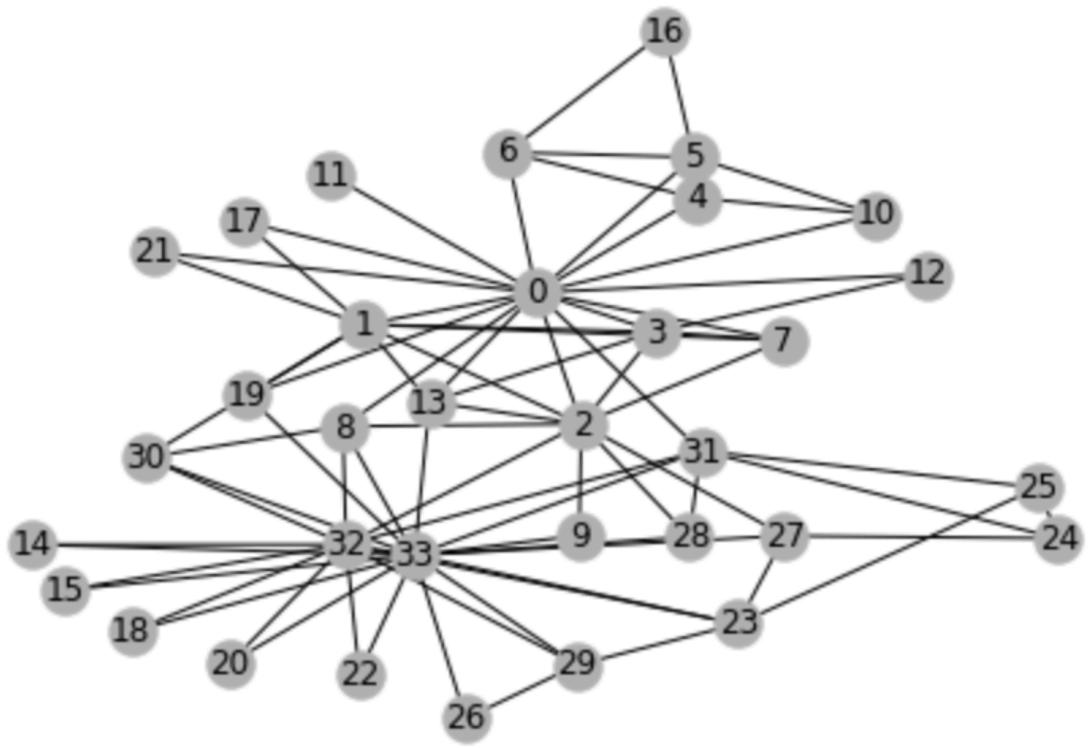
- $\mathbf{S} \cdot \mathbf{x}$ propagates data $\mathbf{x}$ through the nodes
  - We can generalize this to construct neural networks

- Layer propagation rule
  - $\mathbf{H}^{l+1} = \sigma(\ \mathbf{S} \cdot \mathbf{H}^{l} \cdot \mathbf{W}^{(l)}\ )$
    - $\mathbf{W}^{(l)}$: Weight matrix for l^{th} layer
    - σ : Non-linear activation function
    - $\mathbf{H}^{0} = \mathbf{X}$ (the network input)
    - $\mathbf{H}^{L} = \mathbf{Y}$ (the network output)



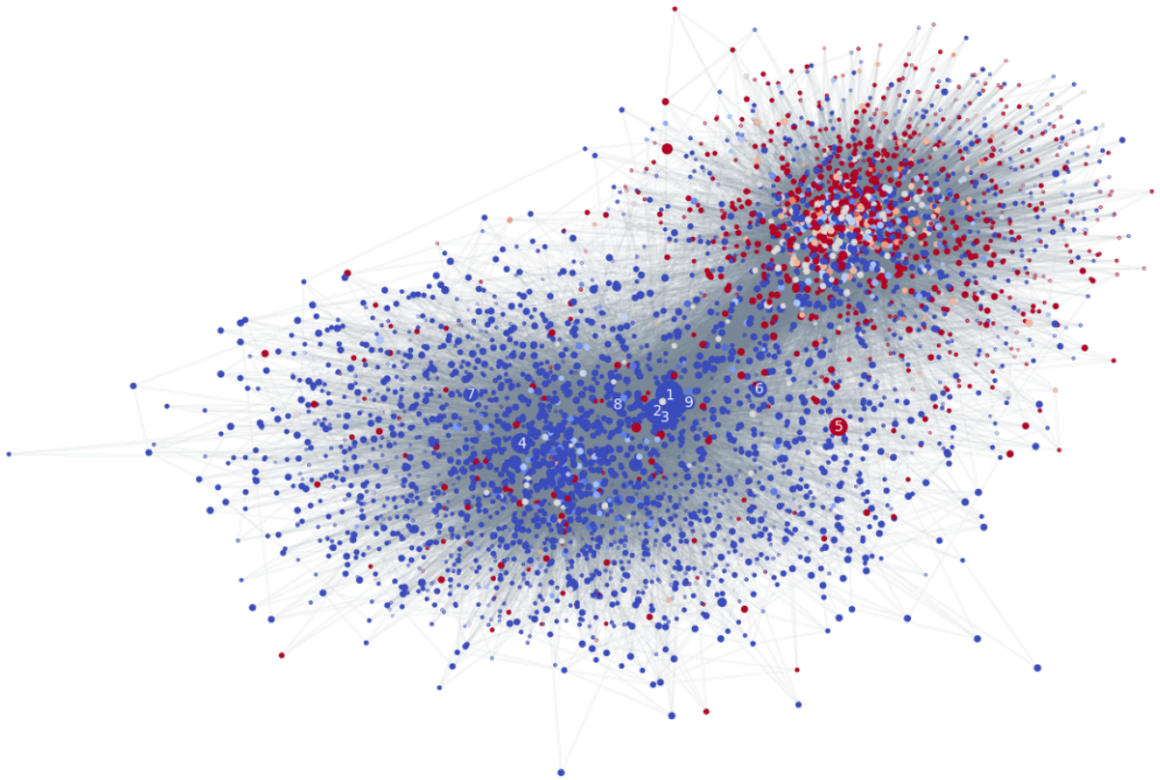
Semi-supervised Node classification

- Karate Club problem
  - Vertices: 34 members
  - Edges: Members who interact outside the club
- Club splits into two communities
  - Node 0 and Node 33
- Which member joins which community?



Semi-supervised Node classification

- Network: GCN with 2 hidden layers
  - We know labels for node 0 and node 33
  - Estimate labels for other nodes
  - Cost function: Cross entropy
  - Demo
- Application: Spread of misinformation on Twitter
  - Uses tweets, social circles and other aspects
- We know some credible and some shady users
  - How credible are the other unknown users?



Vertices: Twitter users

Gray Edges: Social connections

Color: Blue (credible), Red (shady)

Circle sizes: Number of followers

3D Object Classification

- Lot of recent interest in 3D modeling
  - 3D CAD models, Quality inspection, Animation, VR/AR
- A 3D object is a set of points in space
  - The positions are on a continuous space
    - Until recently, no good way of dealing with such data
    - Quantize on a grid and train models
- A point cloud as a graph
  - Each point on the cloud is a node
- EdgeConv
  - Since we don't have a graph yet, we impose a graph
    - We can use KNN to get local neighbors
  - Followed by a Graph Convolutional Network
- For classification, max-pool over all nodes
  - Not required for segmentation

Automatic Code Debuggers

- Novel application in Programming
  - Automatically detect variable misuse
    - Detect and correct incorrect variable name
- Represent a program as a graph
  - A program is a list of tokens from the syntax
    - Augment with Abstract syntax tree and next-token info
      - Link all occurrences of a variable
  - Variable/Function names and other relationships
- Train to predict variable at an empty slot
  - 85% accuracy
  - Fixed real world bugs in open source projects
  - Can possibly be used in an IDE

Recap

- Interpreting signals as a graph is beneficial
  - Graphs understand the underlying structure
  - Flexible: We can use different distance metrics
- Signal Processing on Graph signals
  - Generalizing classical DSP approaches to graph signals
    - The Graph Fourier Transform
  - Applications: Filtering and Denoising
    - Preserves semantic structure in the data better
    - Edge preserving filters
- Graph convolutional networks
  - Generalize passage of information through graphs
    - Several novel applications now possible

THANK YOU