Use case of GFT and wavelet

① If you want to go into graph spectral domain and think about operations on that domain when we can have ad over localization is when we want to reason about the whole graph eg. Is the whole graph globally connected? Do signals between different vertices far away actually similar vs only signals nearby being similar?

② If we look at expression $\Psi_j = P^{2j-1} - P^{2j}$

If you diffuse to a small extend, then you are doing the neighborhood aggregation; If you diffuse to a higher power you are looking at far away neighborhood

The way that value range changes from being local to far away neighborhood said sth about your entire graph especially we don't have real signal, we just have dierect tells about connectivity on a graph that is more locally connected or globally has a scale free type of property

③ That's where wavelet or GFT based NN are useful

④ On the other hand. If we just reasoning about nodes, probably most relevant info of node is from its neighborhood. eg. Facebook friend graph, most info about you is contained in your 1st or 2nd circle of friends you don't need GFT for this kind of problem

① we can have direct GFT as well called Miggnetic Laplacian, they can store directionally in a complex value in entry

There's quite a lot freedom we have with GNN

Regular GNN are more used in direct setting.

② $\Psi \partial_i$ can center signal on certain vertex

∂_i is one hot encoding $\begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix} \rightarrow$ ith entry

$\Psi f = \begin{bmatrix} -\Psi_1 - \\ \vdots \end{bmatrix} \begin{bmatrix} | \\ f \\ | \end{bmatrix}$

Deep Learning Theory and Applications

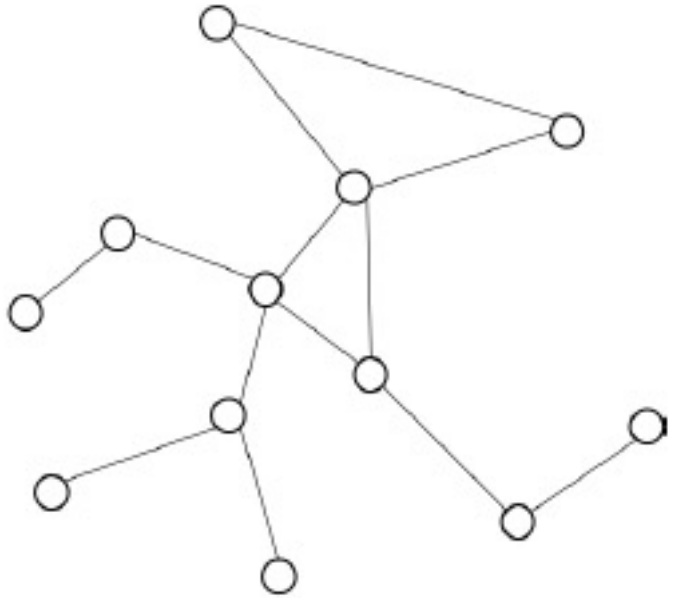
Spectral Graph Convolutional Networks

CPSC/AMTH 663

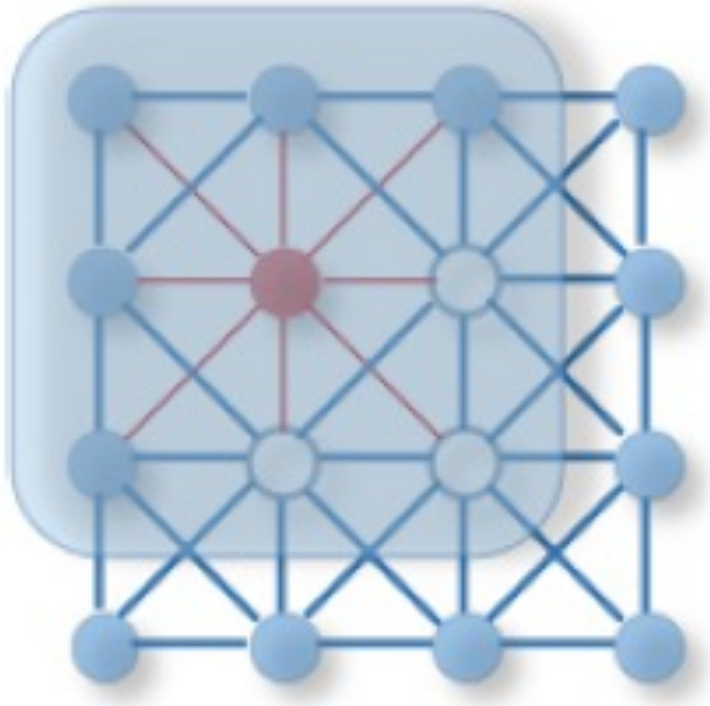
CPSC 452

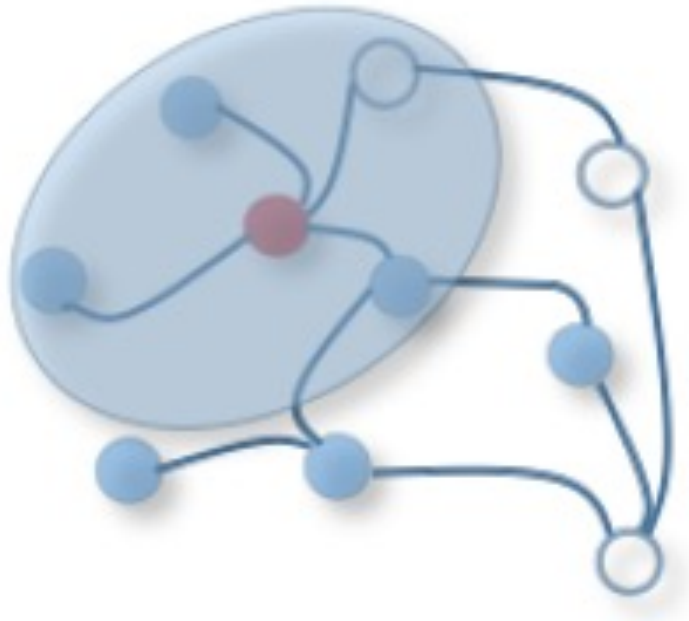
Yale

Images are a specific sort of graph—grid graph



More general graph structures





Recall: CNNs use three basic ideas

1. Local receptive fields
    1. Problem: There is no clearly defined direction or method of translation over a graph (cannot cleanly slide a convolution)
2. Shared weights
3. Pooling
    1. Could potentially be achieved with graph clustering!

2. hierarcical

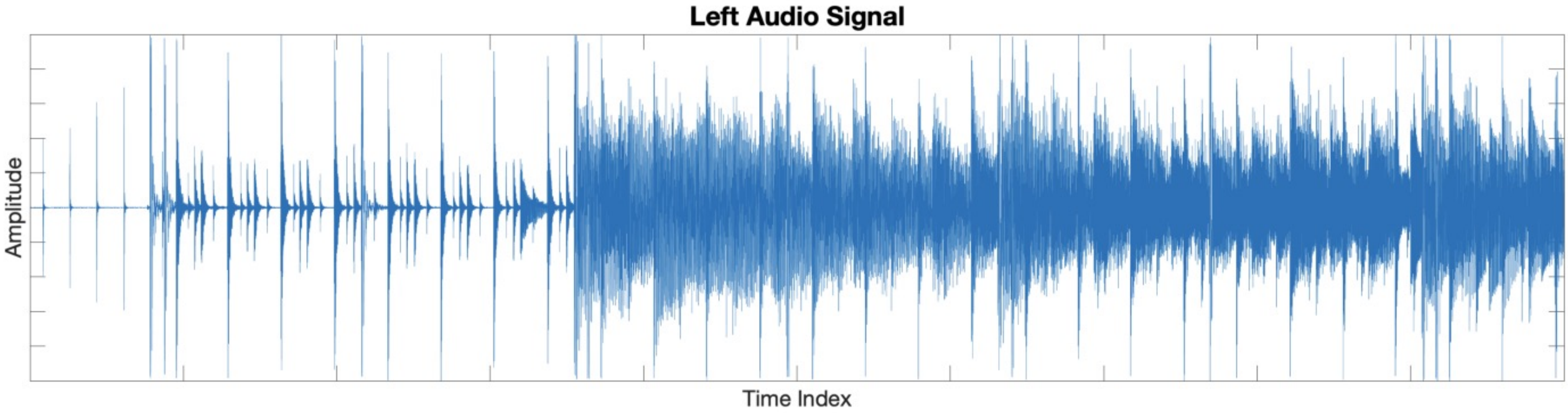
3. simple

How can we approximate these on more general non-grid graphs

- Spatial construction (Bruna, Hamilton)
  - Apply the same operator (aggregation) and move over the graph space in some fashion
  - Notion of locality based on neighbors of a node

- Spectral construction (Bruna, Defferard)
  - Define a convolutional operation in the graph spectral space
  - Uses signal processing theory to define filters in the graph Fourier domain
  - Notion of locality based on global properties of the filter

Time series signals



https://towardsdatascience.com/the-wavelet-transform-e9cfa85d7b34

Fourier Transform of a Signal

= integral of signal function weighted by signal harmonics

$$f(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx$$

harmonics

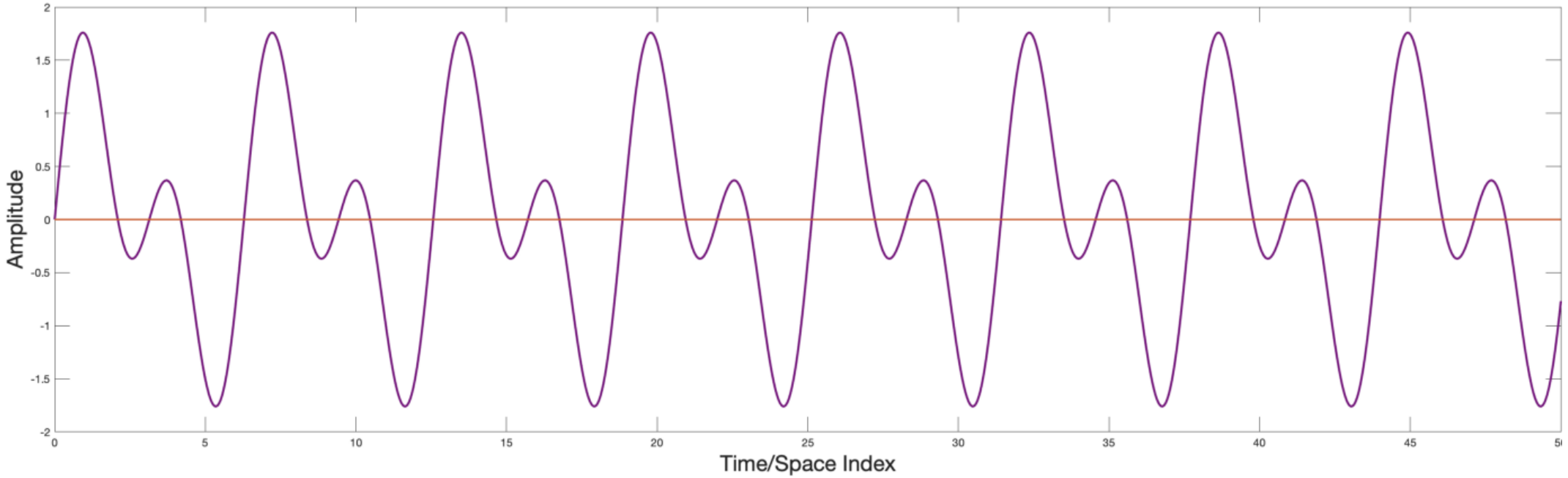
Recall Euler's formula:

$e^{-i\vartheta x} = \cos(\vartheta x) + i\sin(\vartheta x)$

Decomposes signal into frequences

components

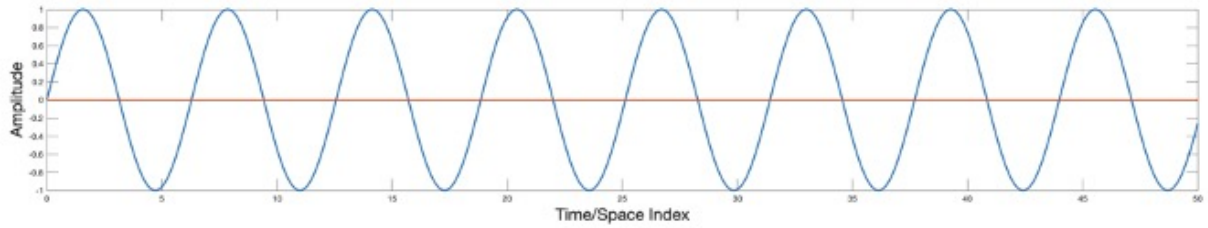
signal



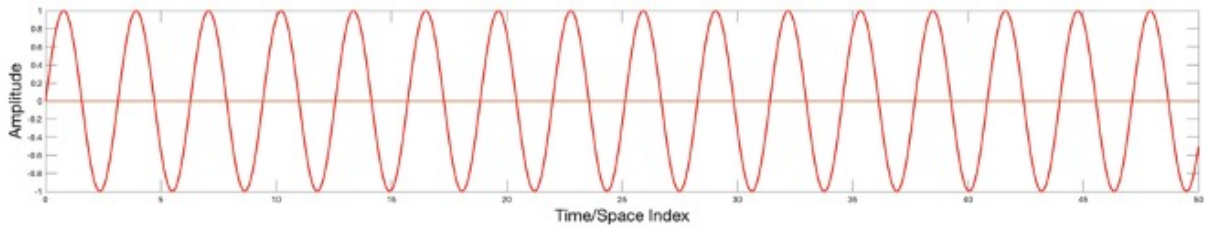
low freq

=

high freq



+



Discrete Fourier Transform

- Usually we don't have the functional form available for computing the integral, so we use discrete samples and discrete wavelengths

take inner product of signal and freq
if f_k is high: signal has high loading of that freq, verse versa

$$f(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx$$

FT

Discretize

$$f_k = \Sigma_0^{N-1} x_n e^{\frac{-2\pi i k n}{N}}$$

DFT

DFT is a matrix multiplication

coefficient at each freq

$$\begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{K-1} \end{pmatrix} = \begin{pmatrix} R_{0,0} & R_{0,1} & \cdots & R_{0,N-1} \\ R_{1,0} & R_{1,1} & & \vdots \\ \vdots & & & \vdots \\ R_{K-1,0} & \cdots & \cdots & R_{K-1,N-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix}$$

signal

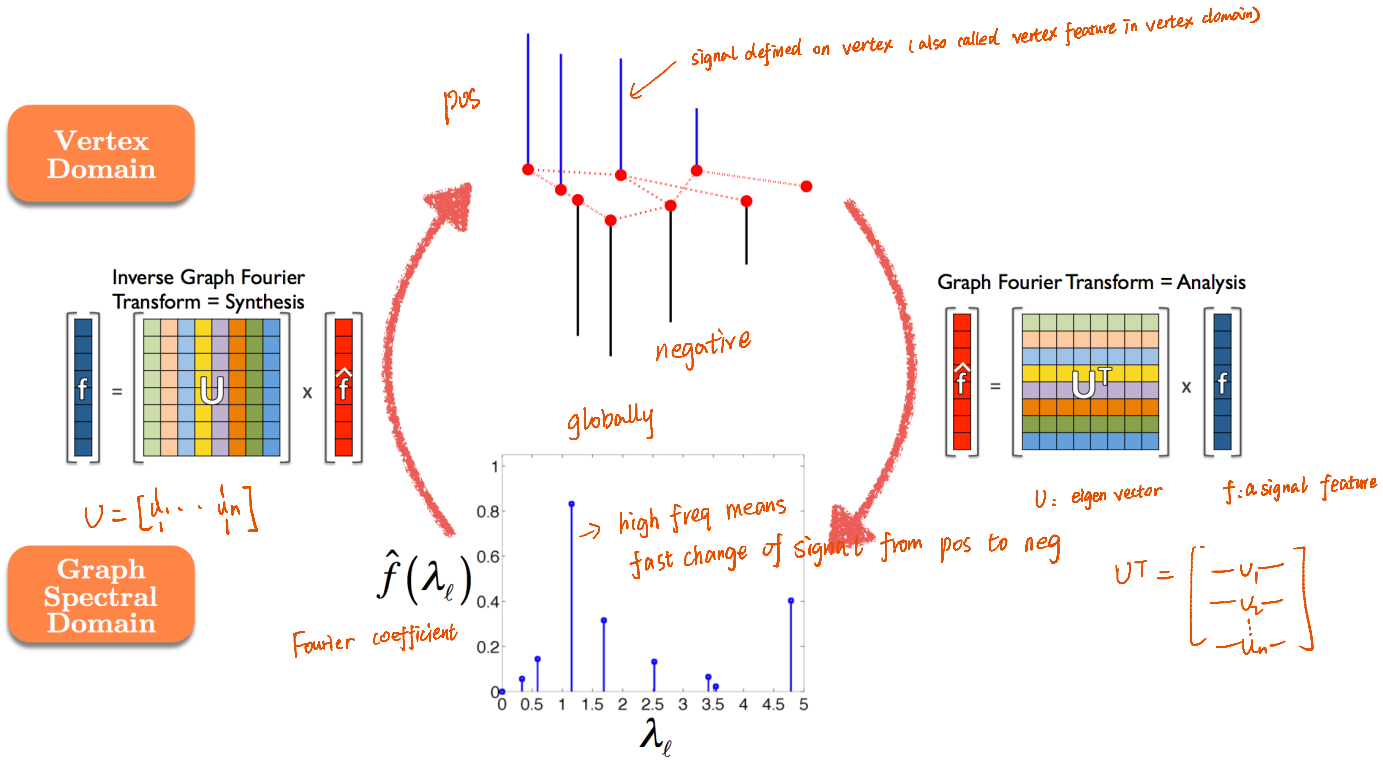
(time signal) $R_{k,n}$ is the value of the kth waveform at time index n $R_{kn} = e^{\frac{-2\pi ikn}{N}}$

Fourier transforms have numerous uses

- Power spectral analysis
- Solutions of differential equations (heat equation) physics
- Audio and image processing
- …much more

Signals on a graph substrate

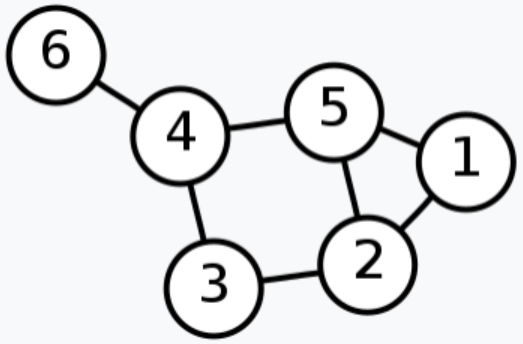
- These signals have an analogous Graph Fourier Transform
- Involves using the eigenvectors of the graph Laplacian as the waveforms



Graphs, Degree Matrix, Adjacency Matrix

$A_{ij} = \begin{cases} 1 & \text{ith node is connected to jth node} \\ 0 & \text{isn't} \end{cases}$

node 1 connects to 2 nodes

Labelled graph	Degree matrix	Adjacency matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$

sum over each row

then put value in diagonal

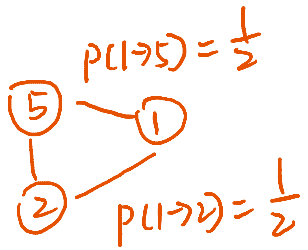
Graph Laplacian

- A difference operator based on the graph adjacency matrix A.
- (unnormalized Laplacian)$L = D - A$
  - Degree matrix: $D_{ii} = \sum_j A_{ij}$, 0 everywhere else
- (normalized Laplacian) $L = I - D^{-1/2}AD^{-1/2}$
  - I is identity
  - Measures how similar a point is to its neighbors

divide by degree, means proportional to its connectivity

$= (D-A)D^{-1}$

- (random walk Laplacian) $L = I - D^{-1}A = I - M$
  - Related to Markovian matrix M tells probability of random walk from one node to another

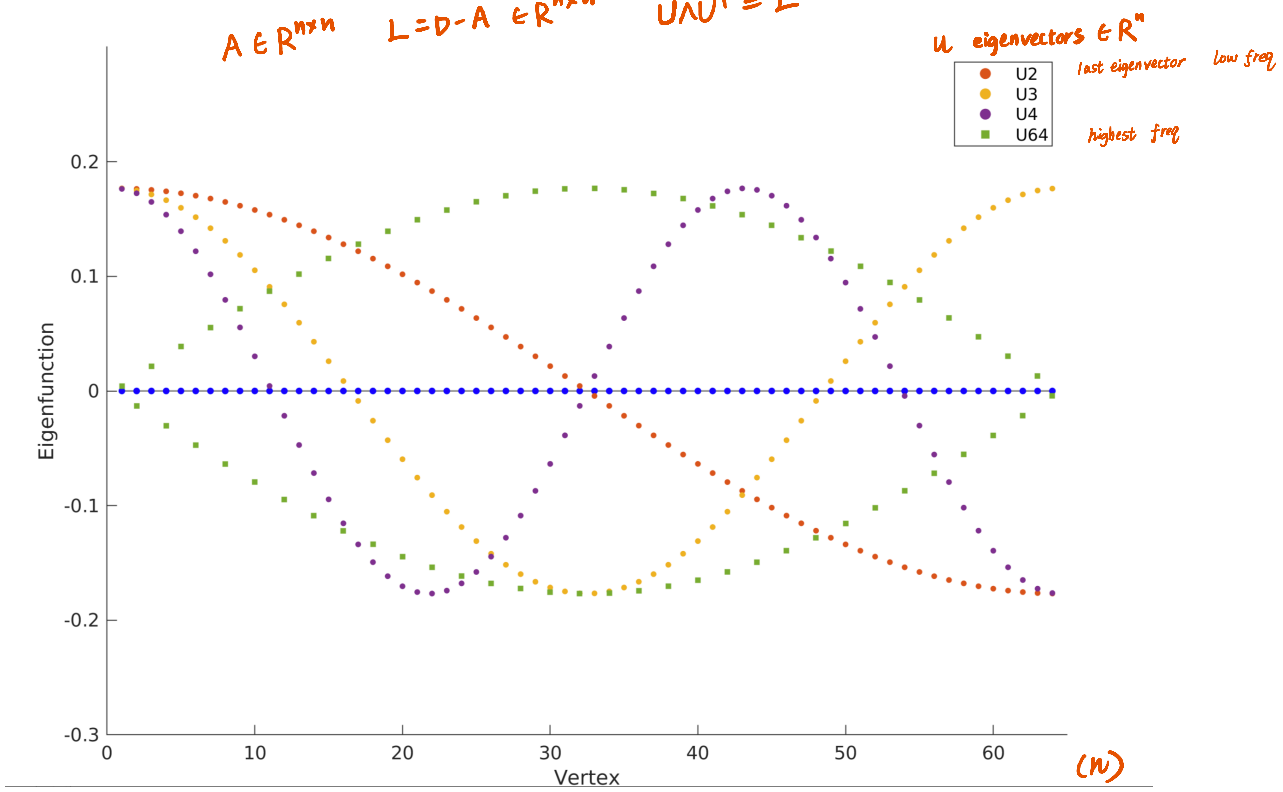


Eigenvectors form a Graph Fourier Spectrum

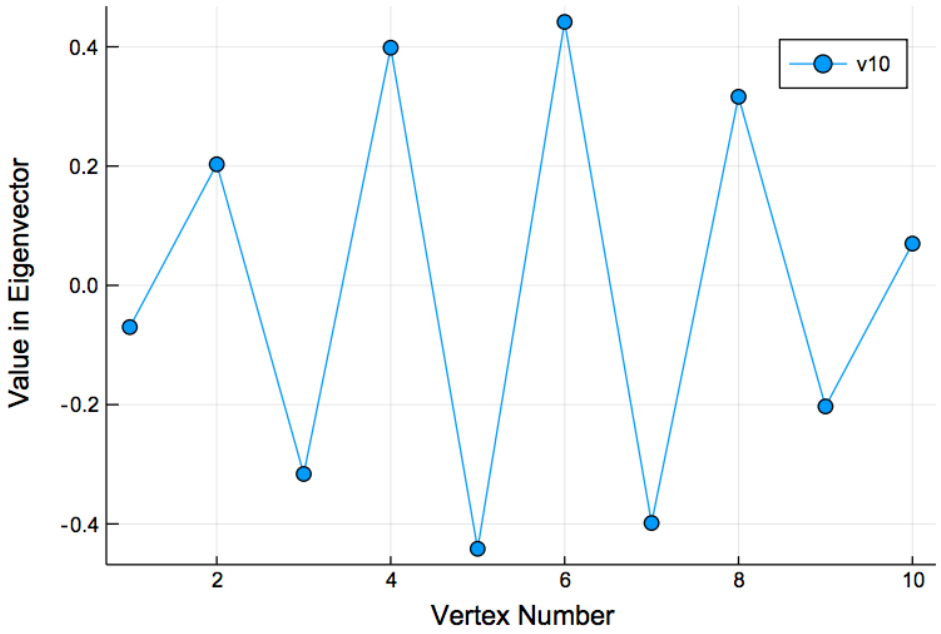
eigenvectors of Graph Laplacian is Fourier harmonics

eigendocompose Laplacian matrix L

$A \in R^{n \times n}$ $L = D - A \in R^{n \times n}$ $U \Lambda U^T = L$



this eigenvector is like U64

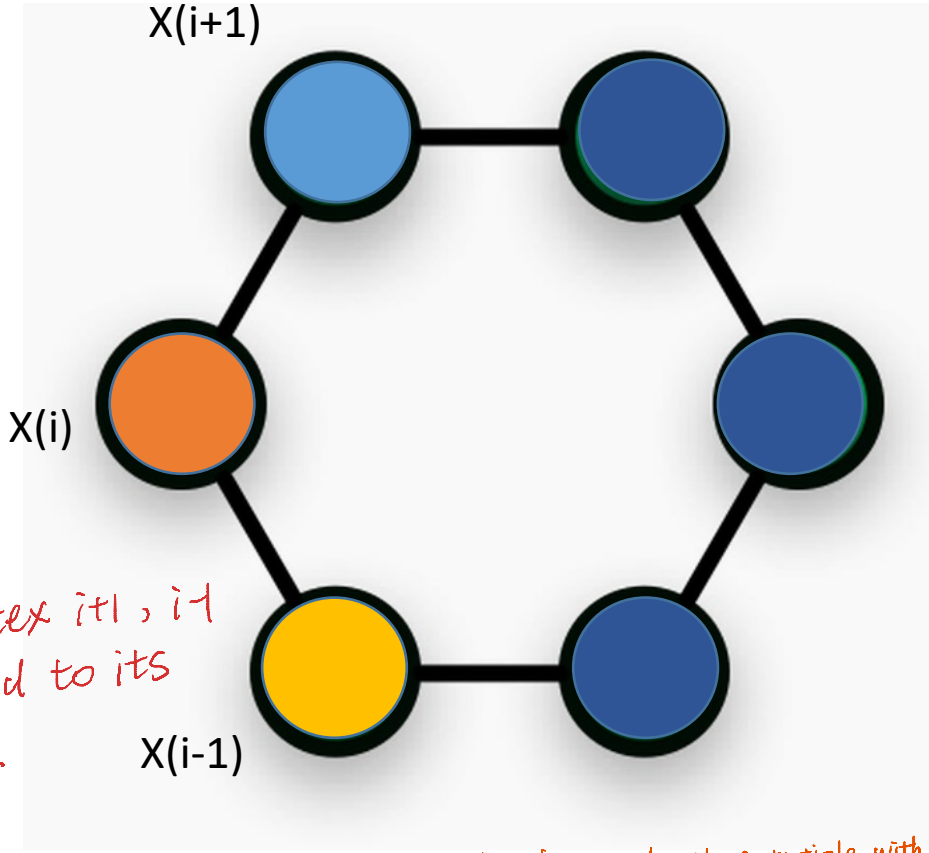


Graph Laplacian

is a second derivative on graph and its eigenfunction are sin and cos

discrete analogue of

- Imagine the Graph Laplacian is created from a dataset with features
- One feature is X
  - X(i) is X evaluate at node i
- First derivative in discrete graph setting is the difference:
  - d(i)= X(i)-X(i-1)
  - d(i+1)=X(i+1)-X(i)
- Second derivative
  - d(i+1)-d(i)
  - =X(i+1)-2X(i)+X(i-1)



vertex i+1, i-1 signal at node i is compared to its

meaning: check how similar

harmonic: second derivative of a func is itself $\frac{\partial^2 f}{\partial x^2} = cf$

eg. sinx, cosx

eigen func: apply an operator of a func and get a mutiple with itself

eigen value is freq square

$\sin(wx) \rightarrow w\cos(wx) \rightarrow -w^2\sin(x)$

Based on Adjacency Matrix

a cycle graph

A=
$$\begin{bmatrix} 0 & 1 & & & & & 1 \\ 1 & 0 & 1 & & & & \\ 0 & 1 & 0 & 1 & & & \\ & & 1 & 0 & 1 & & \\ & & & 1 & 0 & 1 & \\ & & & & 1 & 0 & 1 \\ 1 & & & & & 1 & 0 \end{bmatrix}$$

2nd derivative=
A-D
= -L
$$\begin{bmatrix} -2 & 1 & & & & & 1 \\ 1 & -2 & 1 & & & & \\ 0 & 1 & -2 & 1 & & & \\ & & 1 & -2 & 1 & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & 1 \\ 1 & & & & & 1 & -2 \end{bmatrix}$$

L=D-A
$$\begin{bmatrix} 2 & -1 & & & & & -1 \\ -1 & 2 & -1 & & & & \\ 0 & -1 & 2 & -1 & & & \\ & & -1 & 2 & -1 & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ -1 & & & & & -1 & 2 \end{bmatrix}$$
L

$$\begin{bmatrix} X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix}$$
X

$= LX$

X is a signal

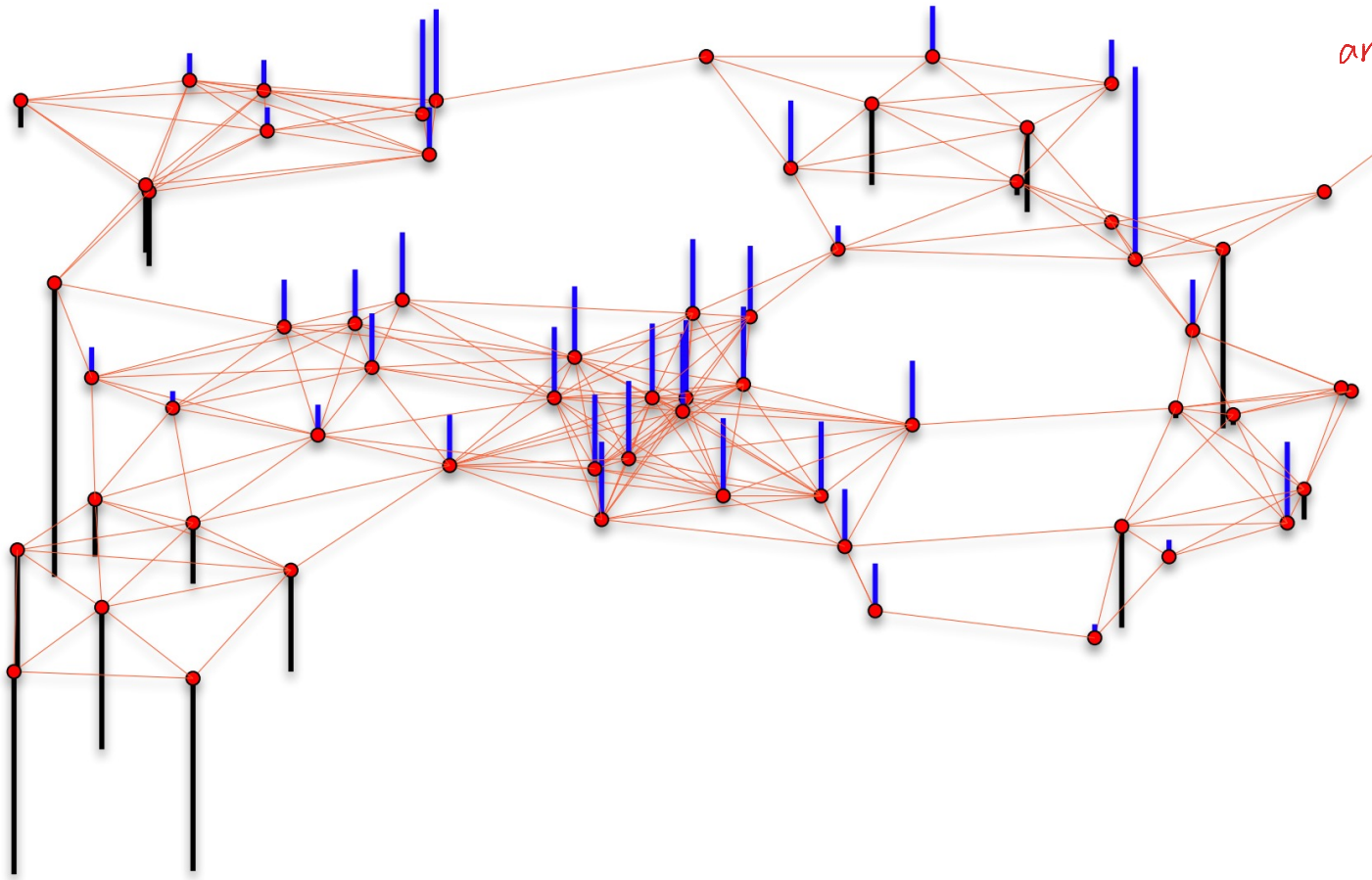
Estimates smoothness of X at each point

$$s = X^T LX$$

s gives smoothness as a score, lower s is more smooth

coz L is small means difference between nodes is small

Features are signals on a graph



reason for specifically use Laplacian is to get harmonics

coz eigen functions of graph Laplacian are sin and cos (second derivative operator)

discretize 离散化

If we use another matrix, it's not second derivative operator we won't get harmonics we will get sth else. we can use it if we want

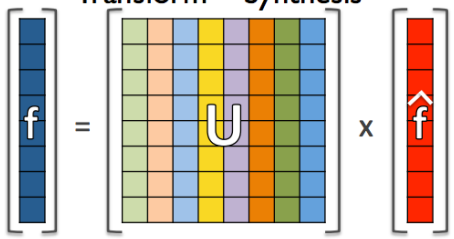
It's important if you want to do freq domain analysis. coz Fourier analysis is all about loading signals to sin and cos of different frequency, then you want them on your graph, they are eigenvector of Laplacian

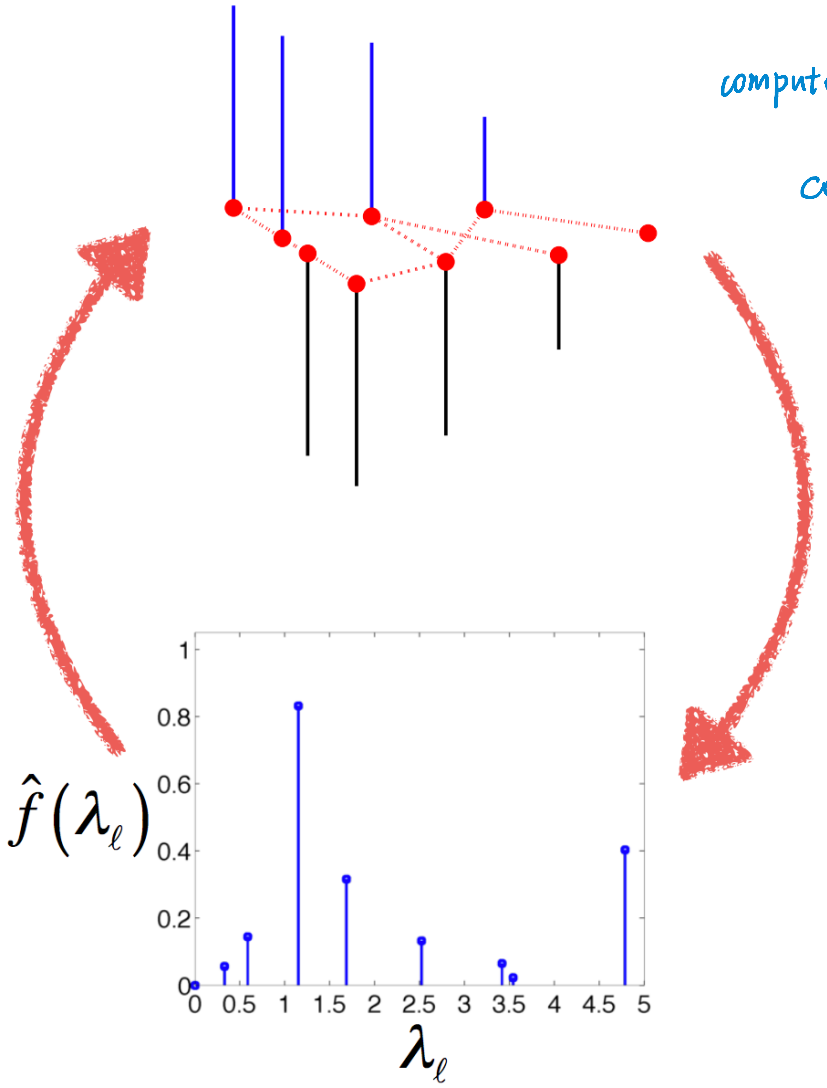
Graph Fourier Transform

Vertex Domain

Graph Spectral Domain

Inverse Graph Fourier Transform = Synthesis





compute ajacency matrix A, degree matrix D

↓

compute graph Laplacian matrix $L = D - A$

↓

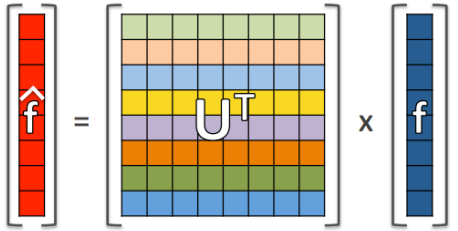
eigen decompose $L = U \Lambda U^{-1}$

L is pos-semi-definite

$0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{n-1}$

$\lambda_i \geq 0$ for all i

Graph Fourier Transform = Analysis



$f \in R^n$ is a feature vector

$U \in R^{n \times n} = \begin{bmatrix} | & & | \\ v_1 & \cdots & v_n \\ | & & | \end{bmatrix}$

ith column is ith eigenvector of L

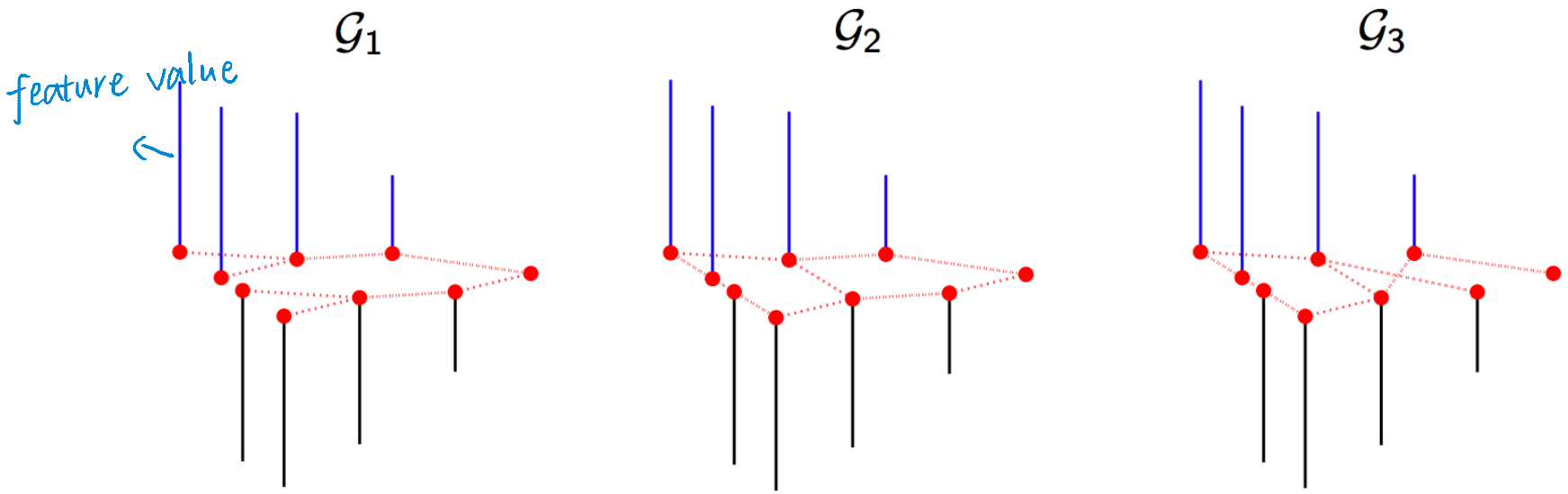
$\Lambda \in R^{n \times n} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}$

diagonal matrix

$\Lambda_{ii} = \lambda_i$

We have 2 ways of representing signals on vertex of a graph

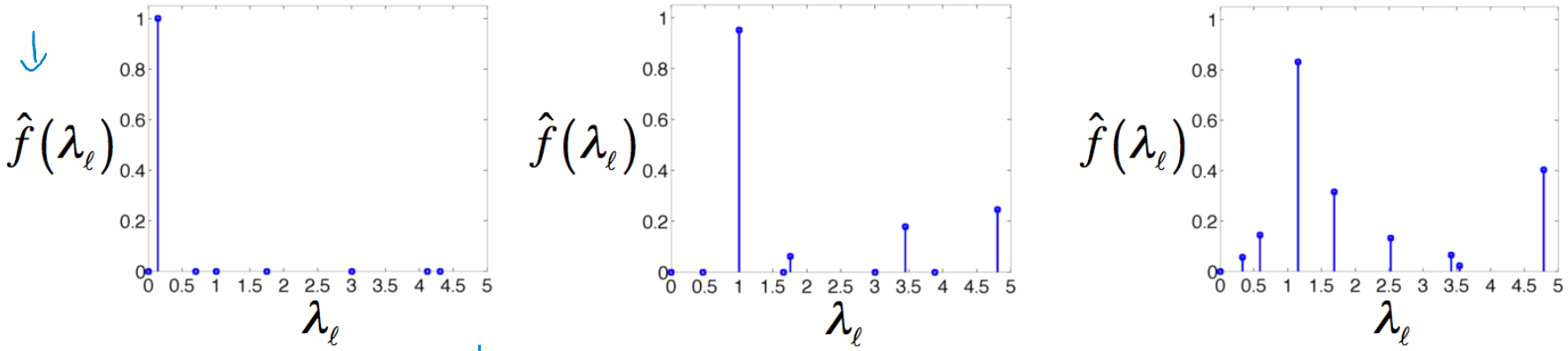
Vertex Domain



Graph Fourier Transform

graph Fourier coefficient of a signal

Graph Spectral Domain

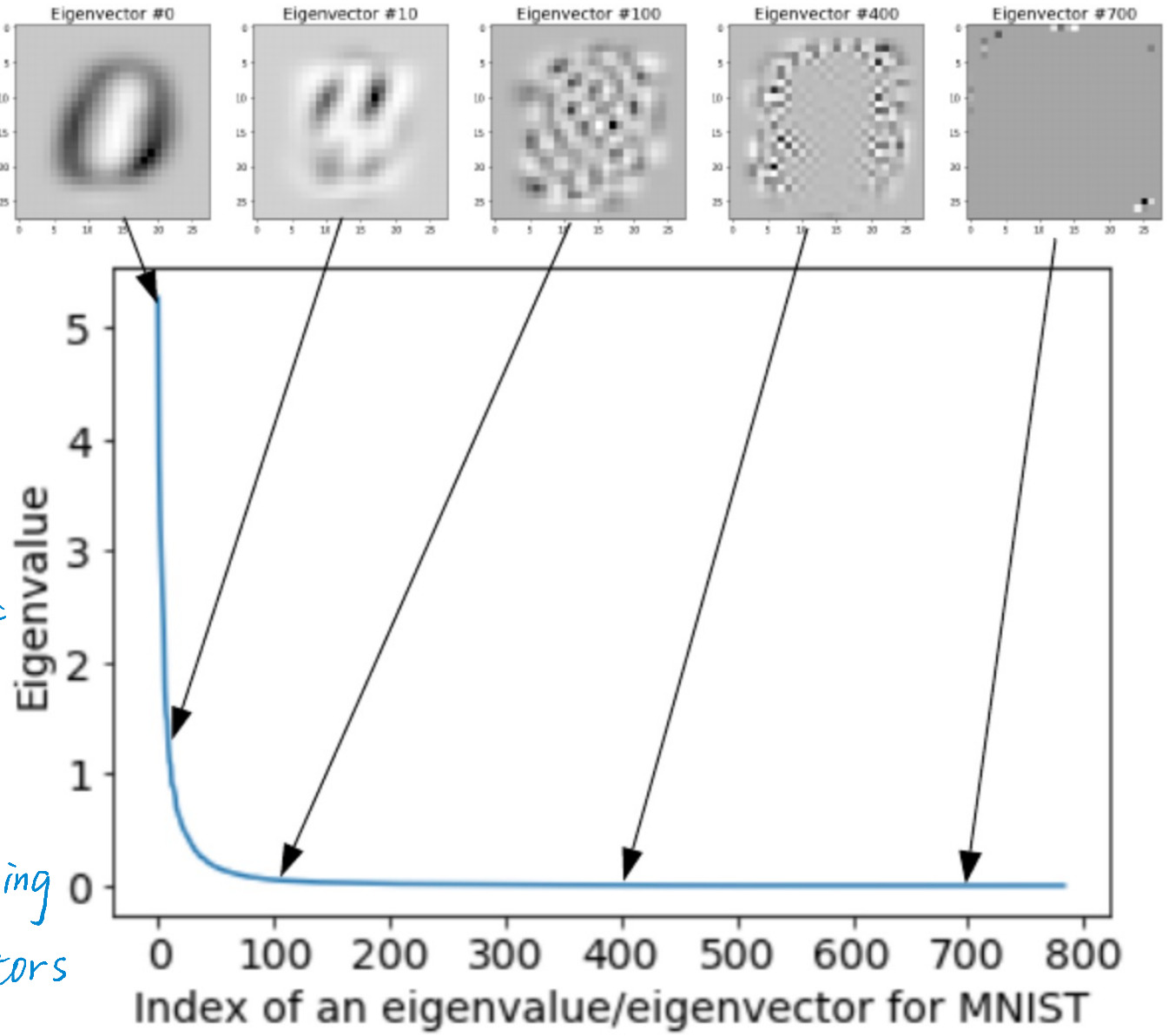


Laplacian Eigenvectors

eigenvector captures global trend on your graph and get more and more specific as increase index of eigenvector

eg: graph is a grid graph

MNIST image dataset



eigenvalue is discrete of freq

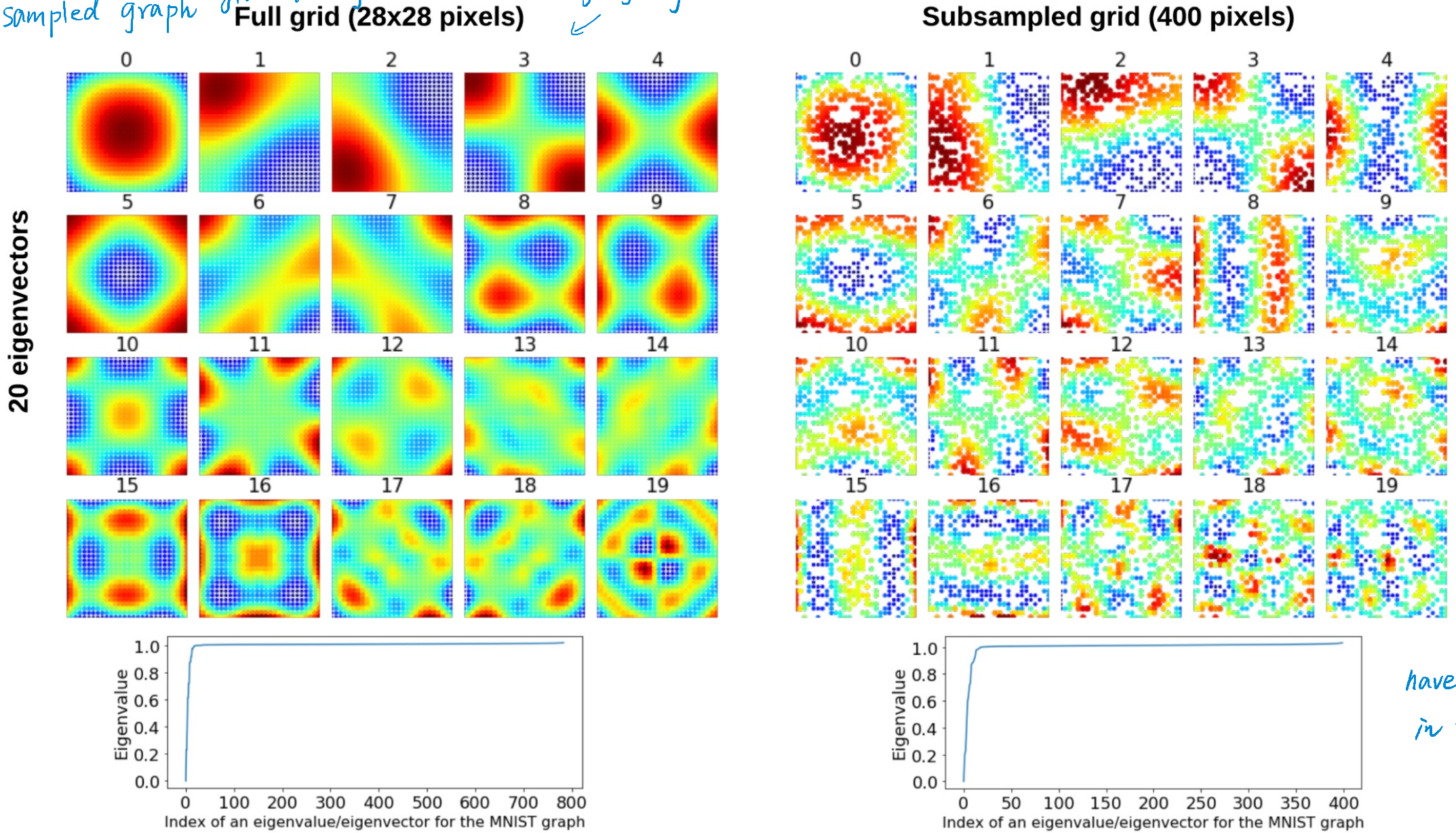
you can't get all the freq on a discrete graph, coz some of freq don't have vertex

you get some of discrete freq

go from low to high by looking at eigenvalues and eigenvectors

Subsampled Graph

highly sampled graph given lots of vertices and high finity with one or another we can get continuous shape



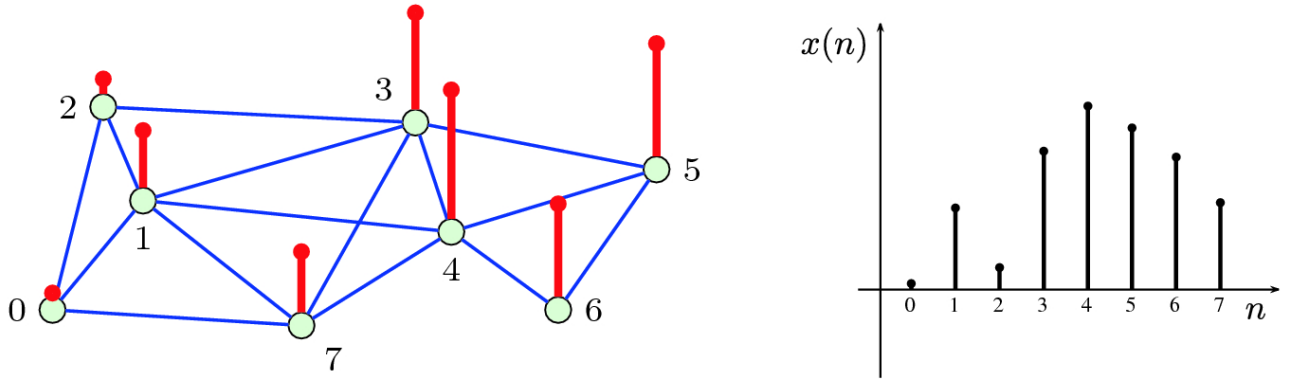
usually graph has gap in regular connectivity we can still get a good idea of main trend in a graph, eg how to compose some signal on a graph looks more regular and will have different freq in the range

Spectral Graph Filtering

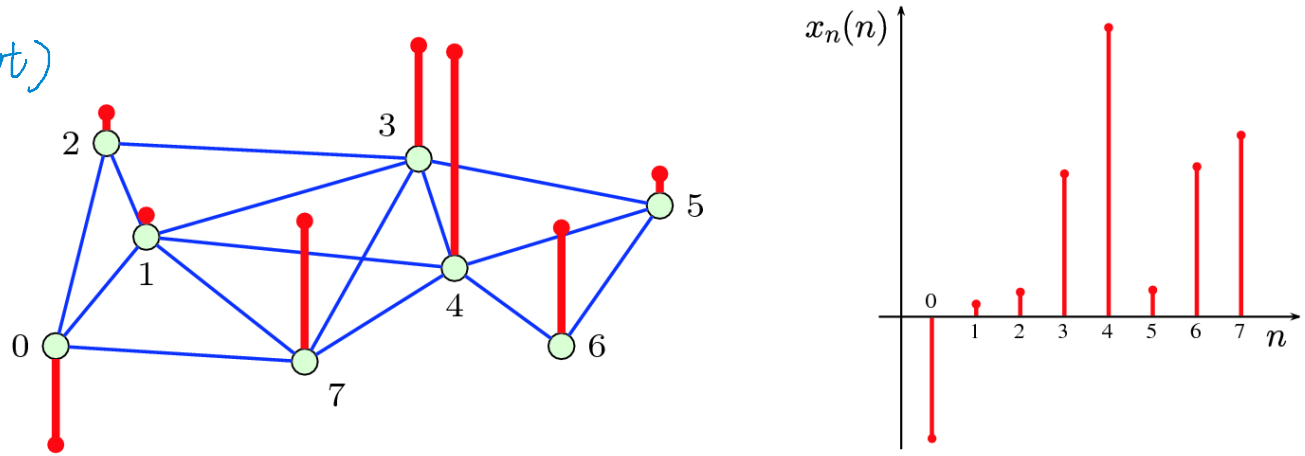
create a filter / feature detector in a spectral graph domain

what we do is rescale freq (coefficient)

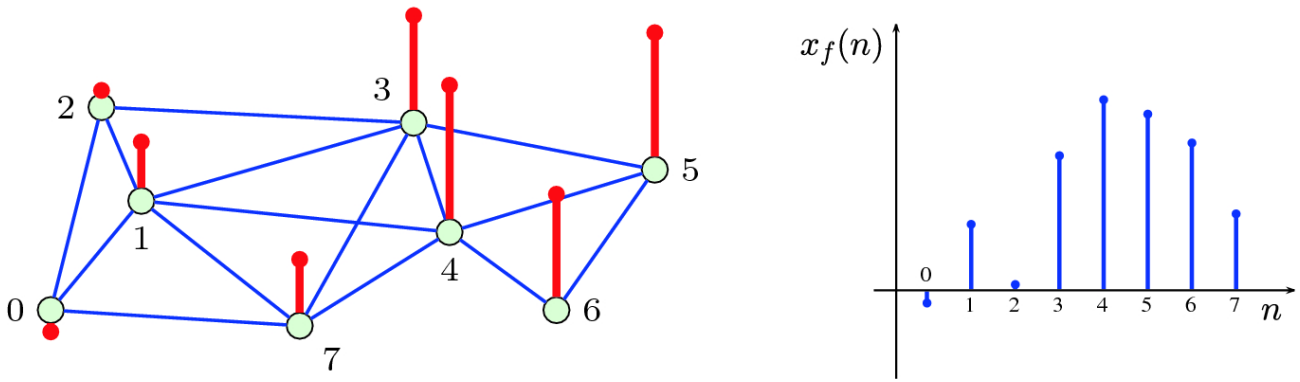
High pass filter : smoothing , cut off low freq features



(a) original signal



(b) noisy signal



(c) filtered signal

Stankovic et al. 2018

General Filter Construction

- Eigendecompose Graph Laplacian
- $$\begin{bmatrix} & L & \end{bmatrix} = \begin{bmatrix} | & | & | \\ U_1 & U_2 & U_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \lambda_3 \end{bmatrix} \begin{bmatrix} | & | & | \\ U_1 & U_2 & U_3 \\ | & | & | \end{bmatrix}^{-1}$$

  eigen vectors　　eigenvalues

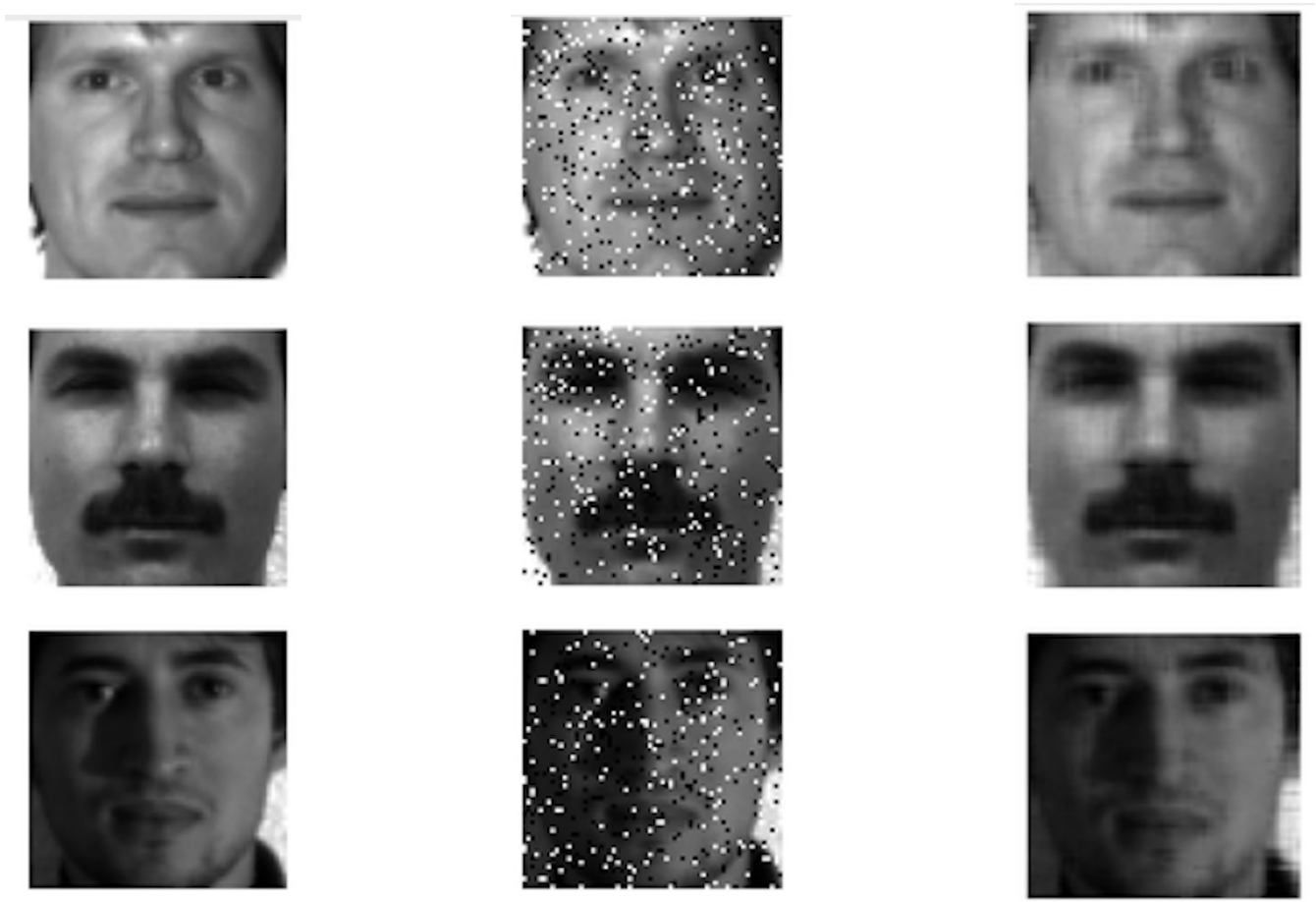
- Modulate and alter eigenvalues　　by function H　　i.e. modulate freq
- $$\begin{bmatrix} & L & \end{bmatrix} = \begin{bmatrix} | & | & | \\ U_1 & U_2 & U_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} H(\lambda_1) & & \\ & H(\lambda_2) & \\ & & H(\lambda_3) \end{bmatrix} \begin{bmatrix} | & | & | \\ U_1 & U_2 & U_3 \\ | & | & | \end{bmatrix}^{-1}$$

  ③ inverse back in spatial domain　　① Graph Fourier Harmonics　　High pass filter: H of high freq = 0

- Apply it to the signal $UH(\Lambda)U^T X$

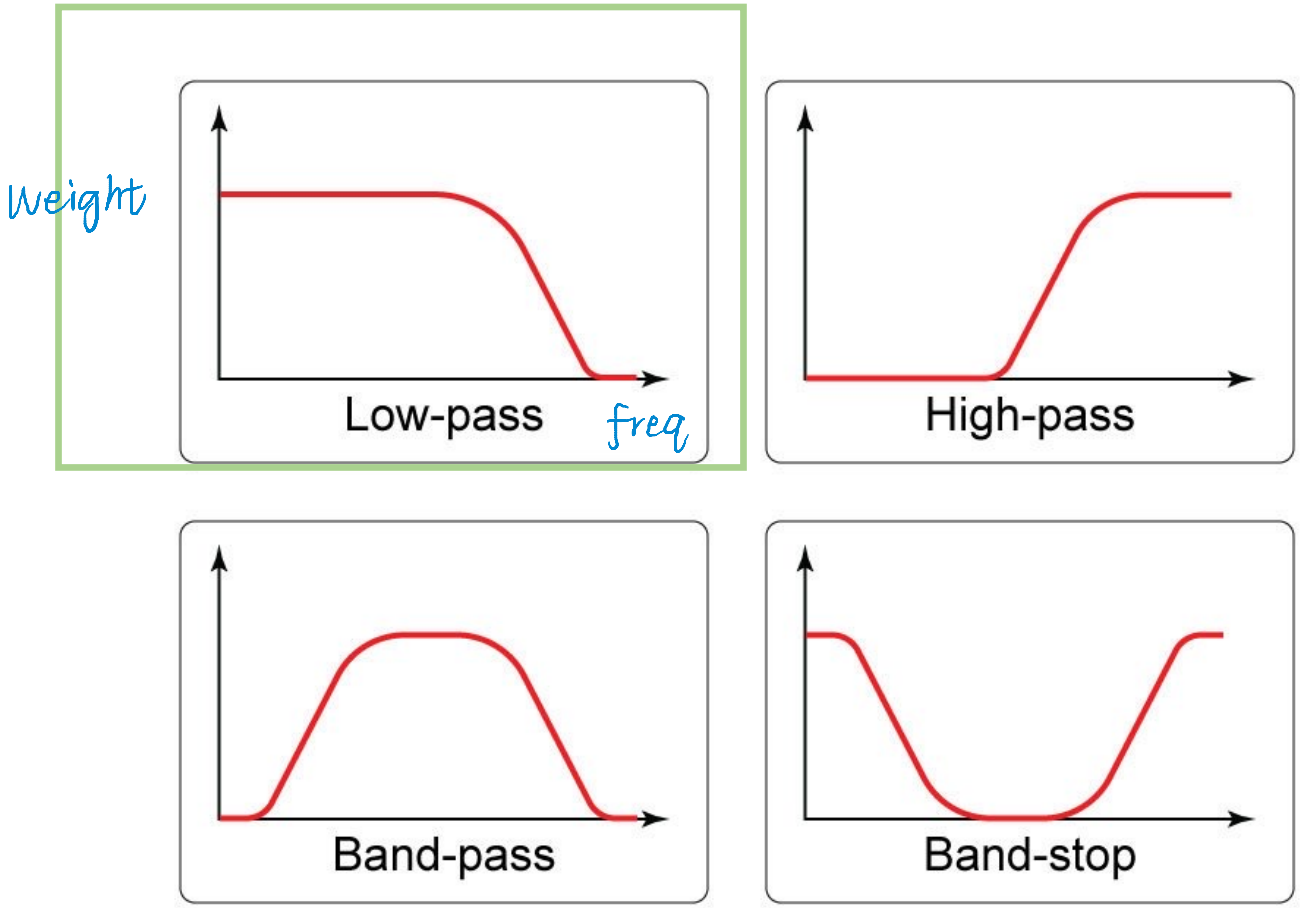
  3 steps computation　　② change coefficient　　signal value

Data signals are low frequency, Noisy can be high frequency



Can take off noise by taking off high frequency eigenvectors

Low-pass filter



Example of a low pass filter

- $$\begin{bmatrix} | & | & | \\ U_1 & U_2 & U_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} H(\lambda_1) & & \\ & H(\lambda_2) & \\ & & H(\lambda_3) \end{bmatrix} \begin{bmatrix} | & | & | \\ U_1 & U_2 & U_3 \\ | & | & | \end{bmatrix}^{-1}$$

- $H(\lambda) = e^{-\lambda}$ called heat filter mimics diffusion of heat

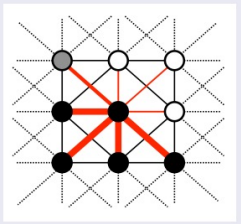
- Applying $UH(\Lambda)U^T X$ would diminish high frequency components

Image Denoising

another kind of low pass filter derive from take derivative of loss

$$f(n) \longrightarrow \boxed{\text{GFT}} \longrightarrow \hat{f}(\lambda_\ell) \longrightarrow \boxed{\hat{g}} \longrightarrow \hat{g}(\lambda_\ell)\hat{f}(\lambda_\ell) \longrightarrow \boxed{\text{IGFT}} \longrightarrow \Phi f(n)$$

Semi-Local Graph

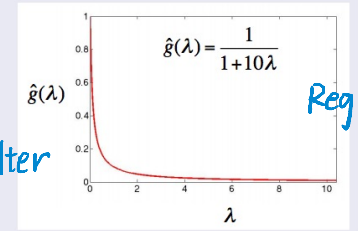


Tikhonov Regularization

$$\underset{f}{\operatorname{argmin}} \left\{ \|f - y\|_2^2 + \gamma f^{\mathrm{T}} \mathcal{L} f \right\}$$

$$\Longrightarrow \hat{g}(\lambda_\ell) = \frac{1}{1+\gamma\lambda_\ell}$$

low pass filter



L : graph lapalacian

Reg term $f^T L f$ is for compute smoothness

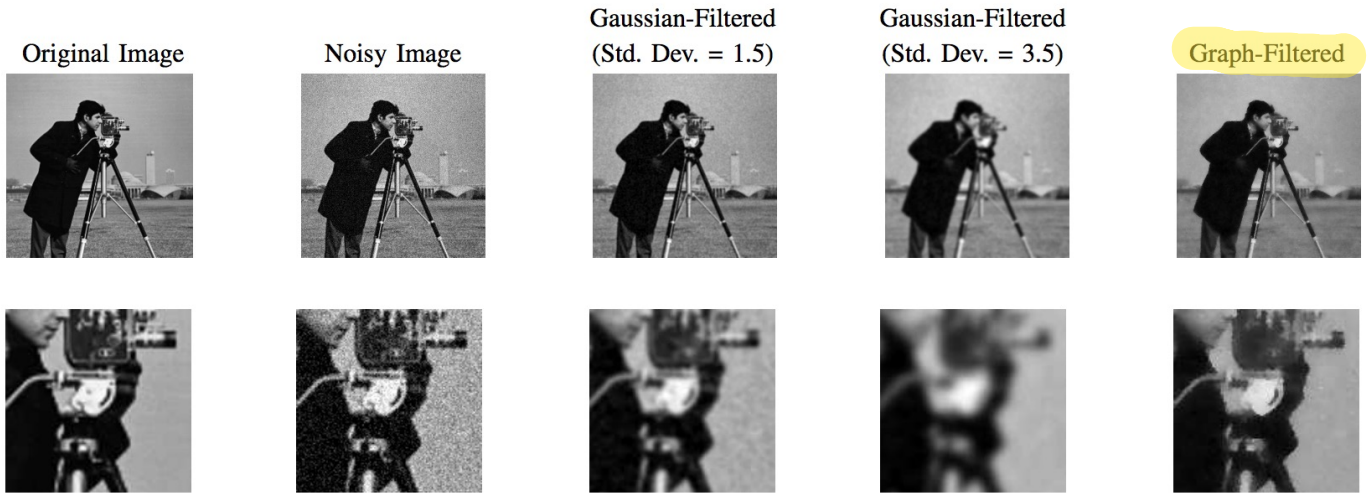
Loss will penalize signals on graph be not smoth i.e. have noise

f : modified signal

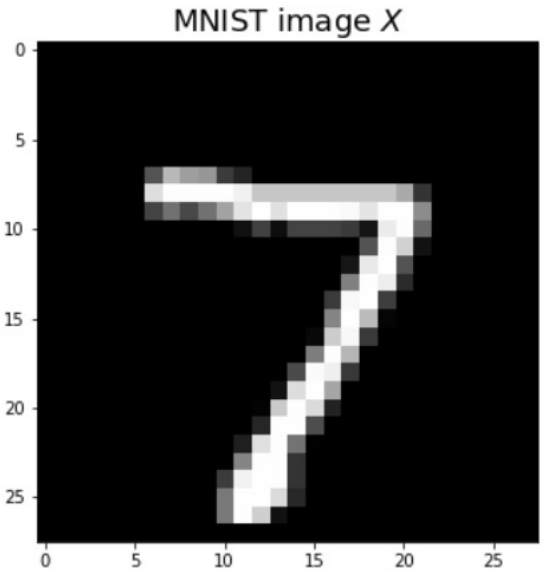
y : original signal

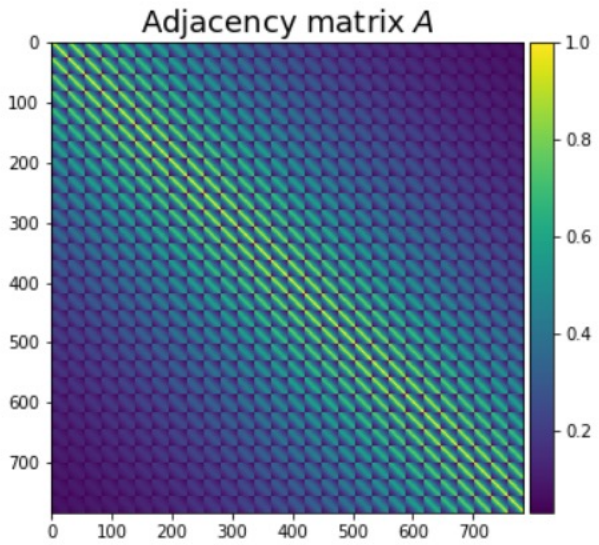
λ_l : eigenvalue

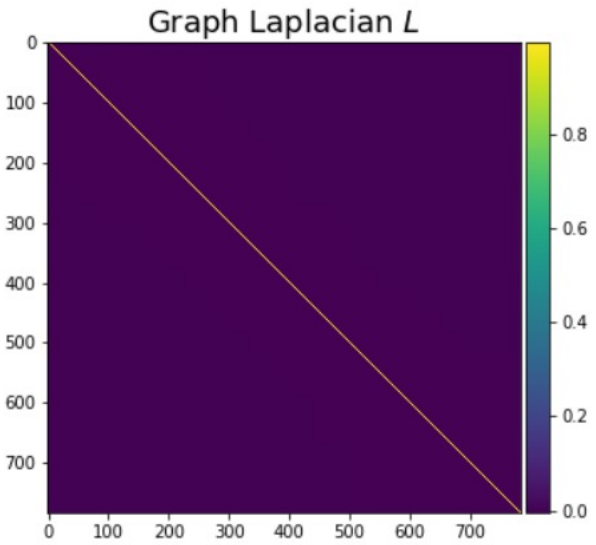
γ : smooth factor



Graph Spectral Convolution MNIST

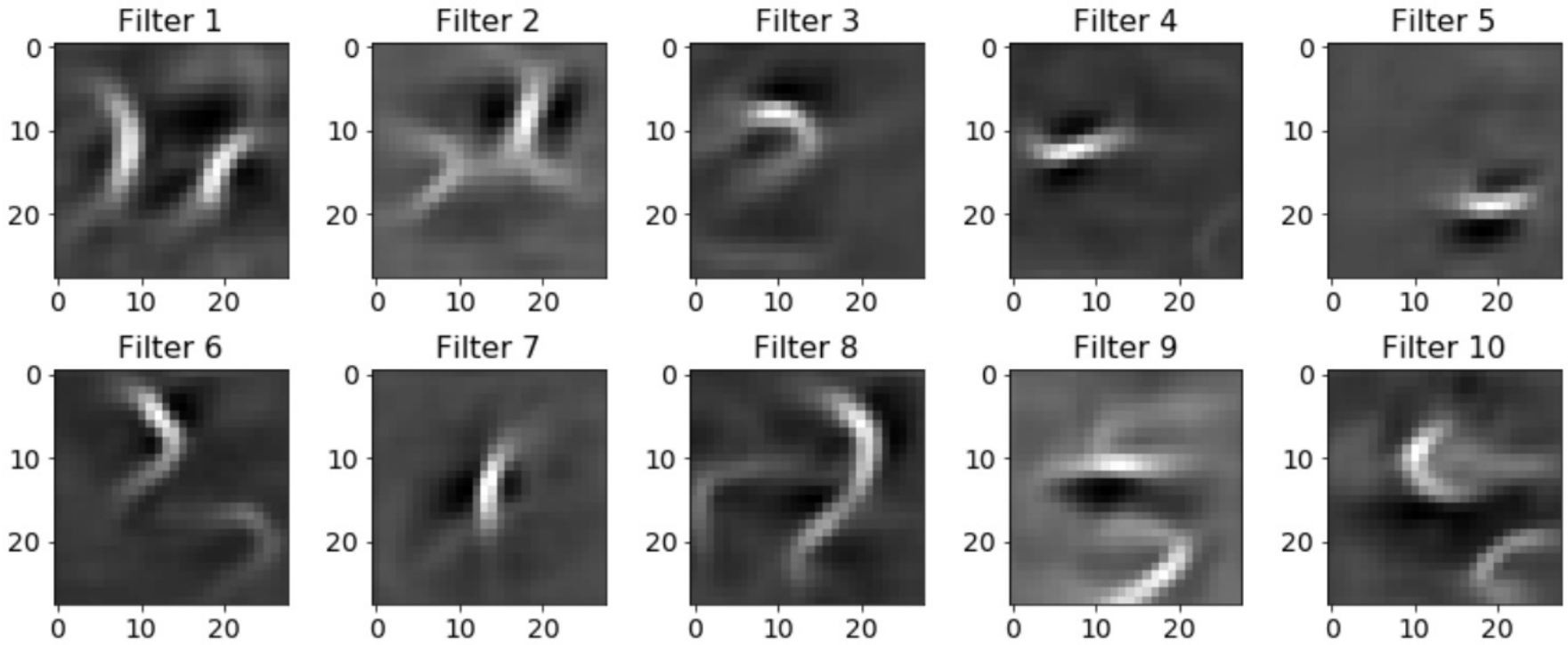






Learned Graph Spectral Filters

filters can be learned



Problems

- **Problem 1**: The filters are no longer localized operations
  - They are operating globally based on frequency characteristics
- Fix: To localize in the spatial domain you need smoothness in the Fourier domain
  - Smoothness achieved by restricting filters to polynomial filters with K coefficients
  - K coefficients gives K-node localization!
- Fix 2: Use wavelet filters instead of fourier
  - Wavelets are localized in time and space
- **Problem 2:** Fourier transform is expensive, eigendecomposition of an NxN matrix each time
- Fix: Chebyshev approximation
- Fix2: Diffusion based operations

A tradeoff spacial domain vs spectral domain

use polynomial of Laplacian power matrix is faster then matrix decomposition

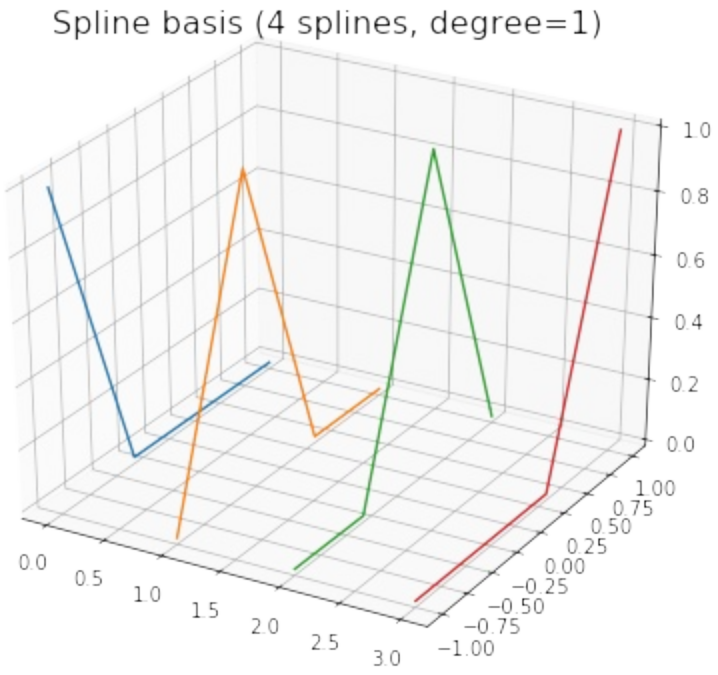
Diffusion matrix with random walk approx

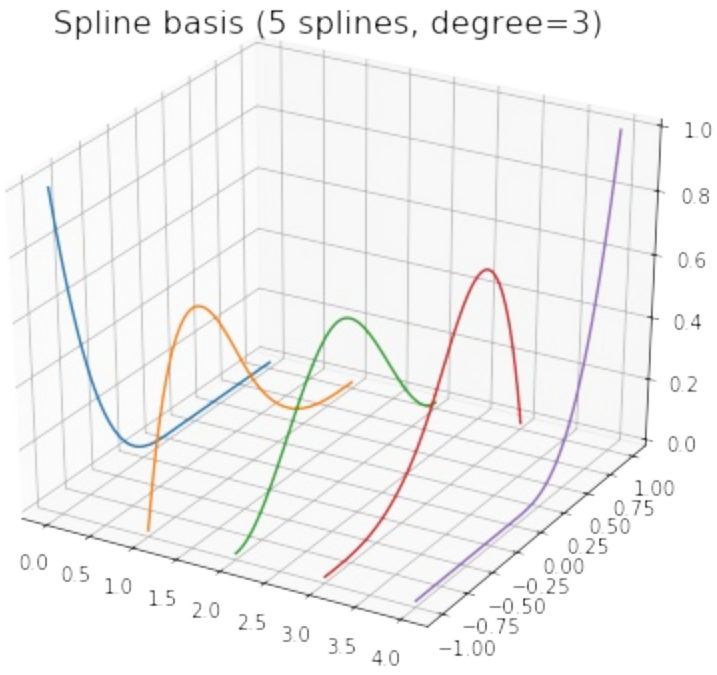
smooth in spacial domain → localized in spectral

smooth in spectral domain → smooth in spacial

K filter coefficients

a spline is a func defined piecewise by polynomials





coefficient of each filter

↑

$$W_{\text{spectral}}^{(l)} \approx \sum_{k=1}^{K} \alpha_k f_k \qquad (4)$$

Chebyshev Polynomials

- Avoiding eigendecomposition which is expensive
- Chebyshev polynomials of the graph Laplacian defined recursively

often degree =3

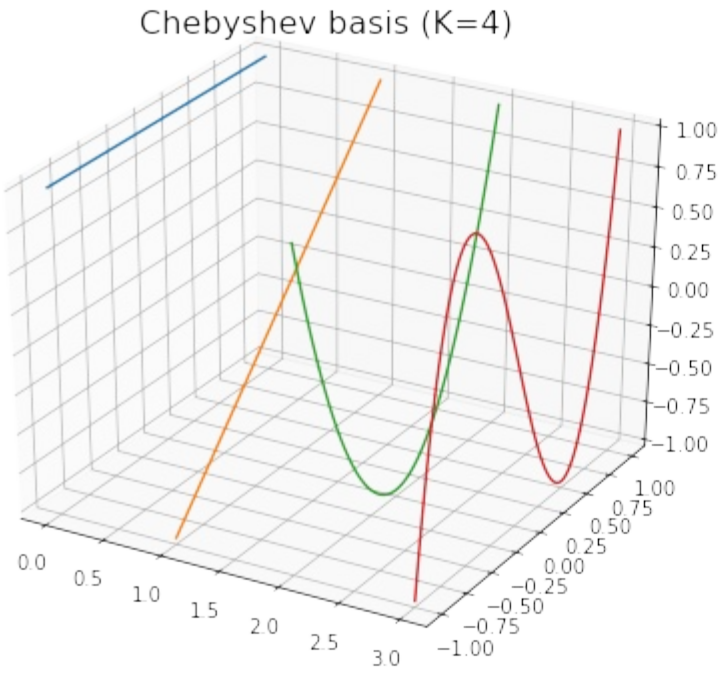
$$T_k(y) := \begin{cases} 1, & \text{if } k = 0 \\ y, & \text{if } k = 1 \\ 2yT_{k-1}(y) - T_{k-2}(y), & \text{if } k \geq 2 \end{cases}.$$

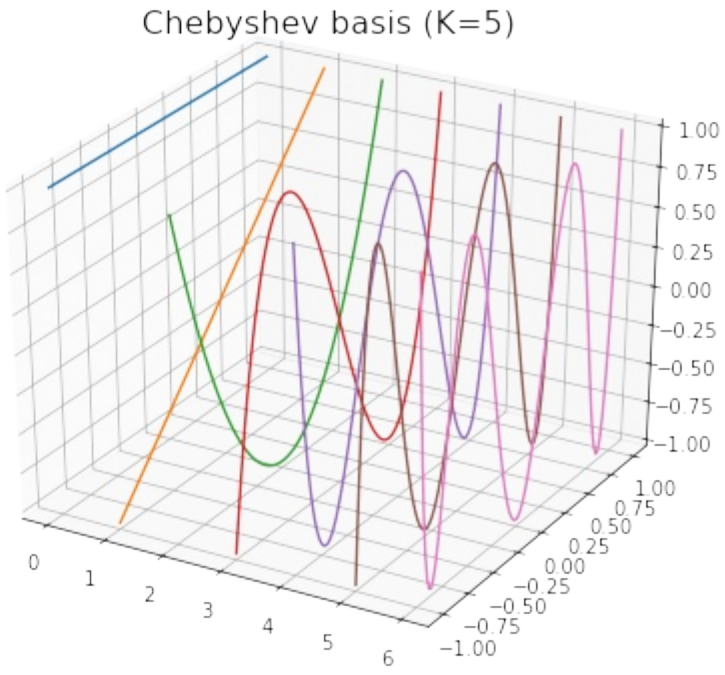
Original Spectral Filters $$y = g_\theta(L)x = g_\theta(U\Lambda U^T)x = Ug_\theta(\Lambda)U^T x.$$

Polynomial Filters $$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k,$$

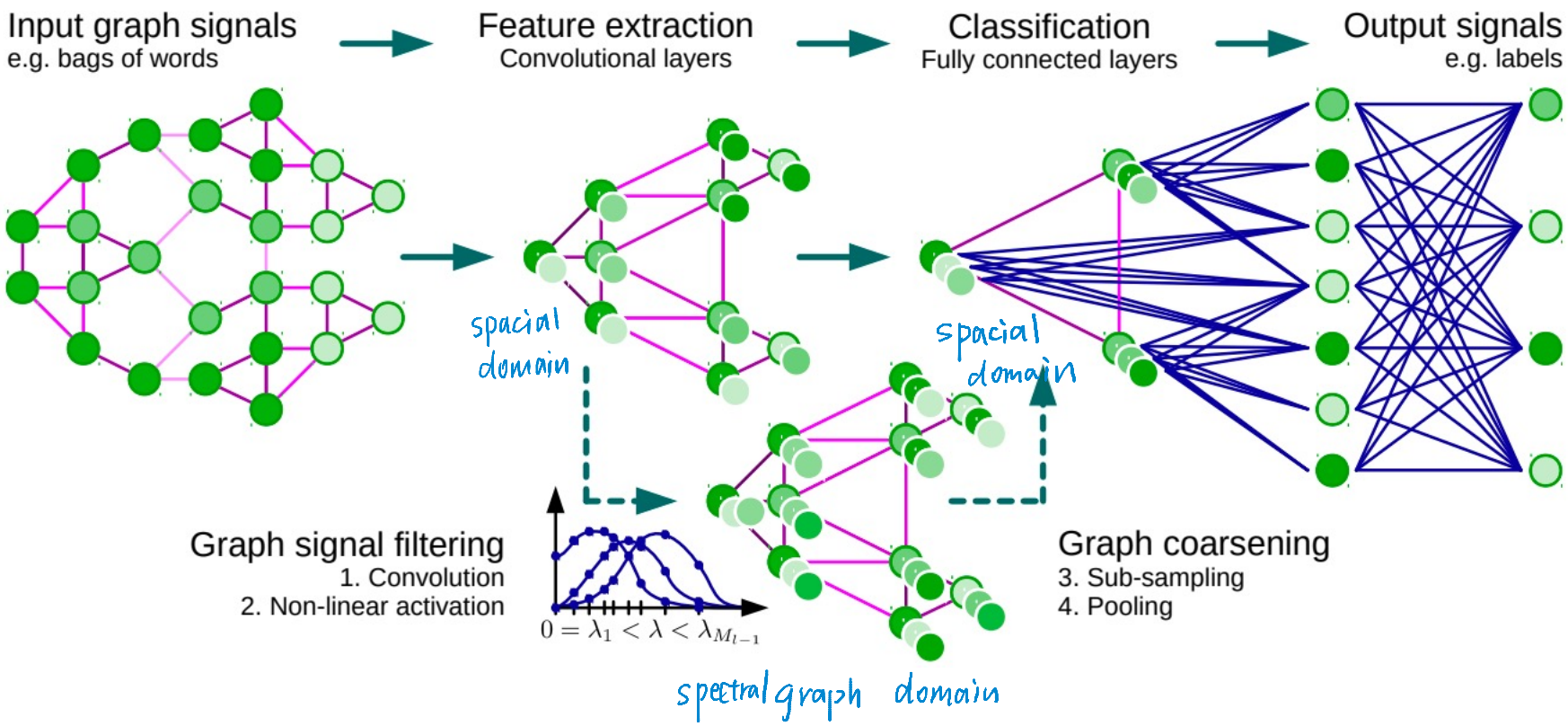
Chebyshev Filters $$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}),$$

Chebyshev Basis



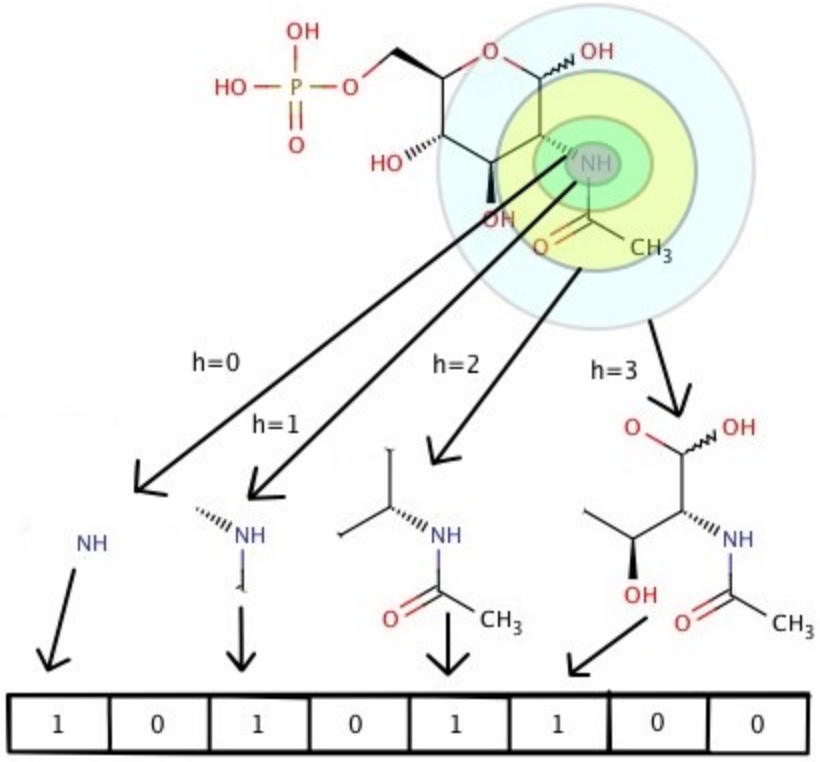


Summarized Architecture of GCN

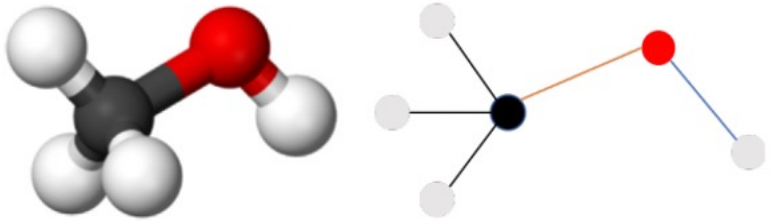


Applications

- Predicting Molecular properties by encoding molecules as graphs



features : connectivity, edge degree, atom name (one-hot)
variety of different features,



(b) molecule

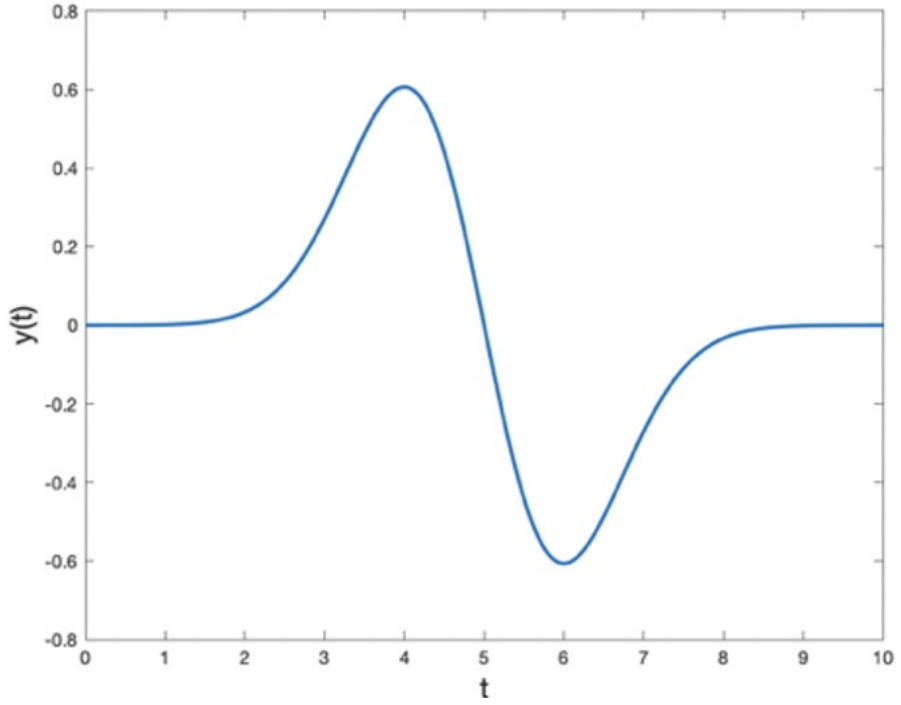
Global vs Local Frequency Domain Analysis

- Fourier transforms capture global frequency information, frequencies that persist over the entire length of the signal
- What about small bursts of frequency changes?
- Uneven frequency distributions over time?
- These might benefit from a more localized frequency domain analysis

use wavelet

Wavelets

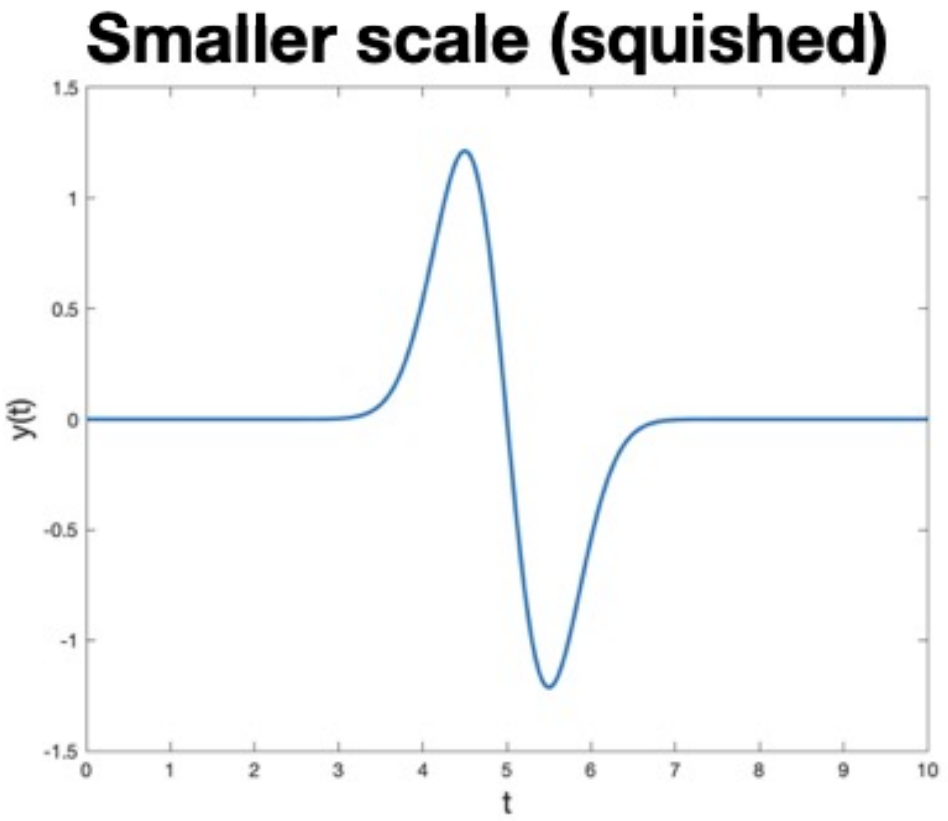
- Unlike a sine or cosine function, a wavelet is a localized oscillation

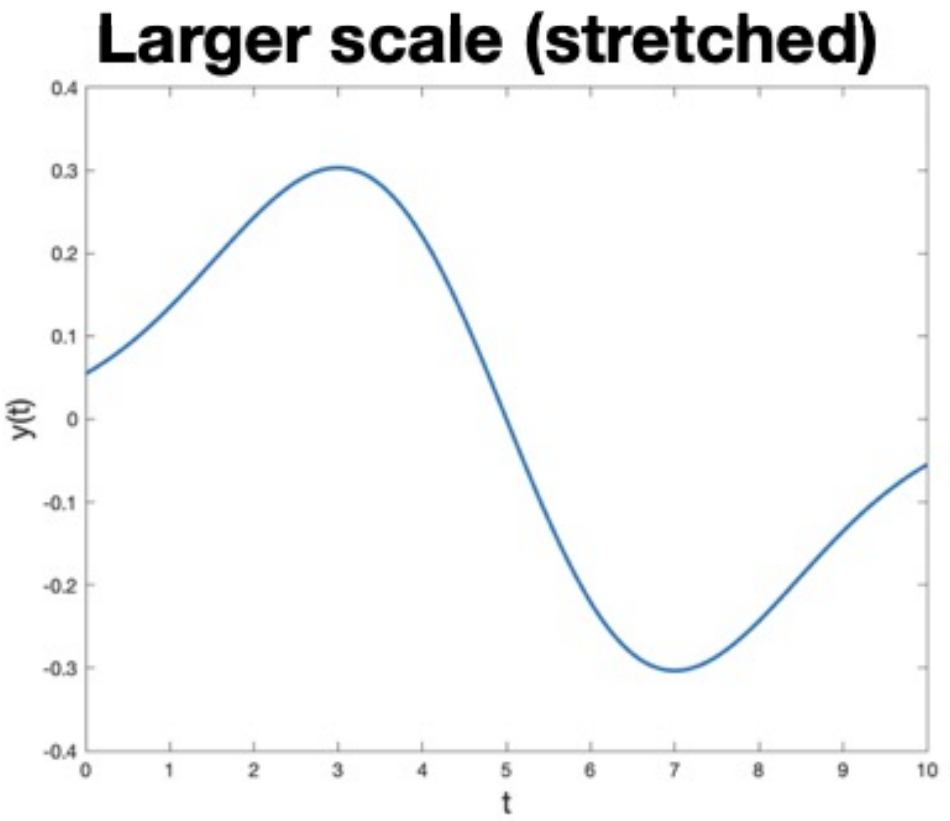


both localized in time and freq

Scaling wavelets

not localized in time but in freq

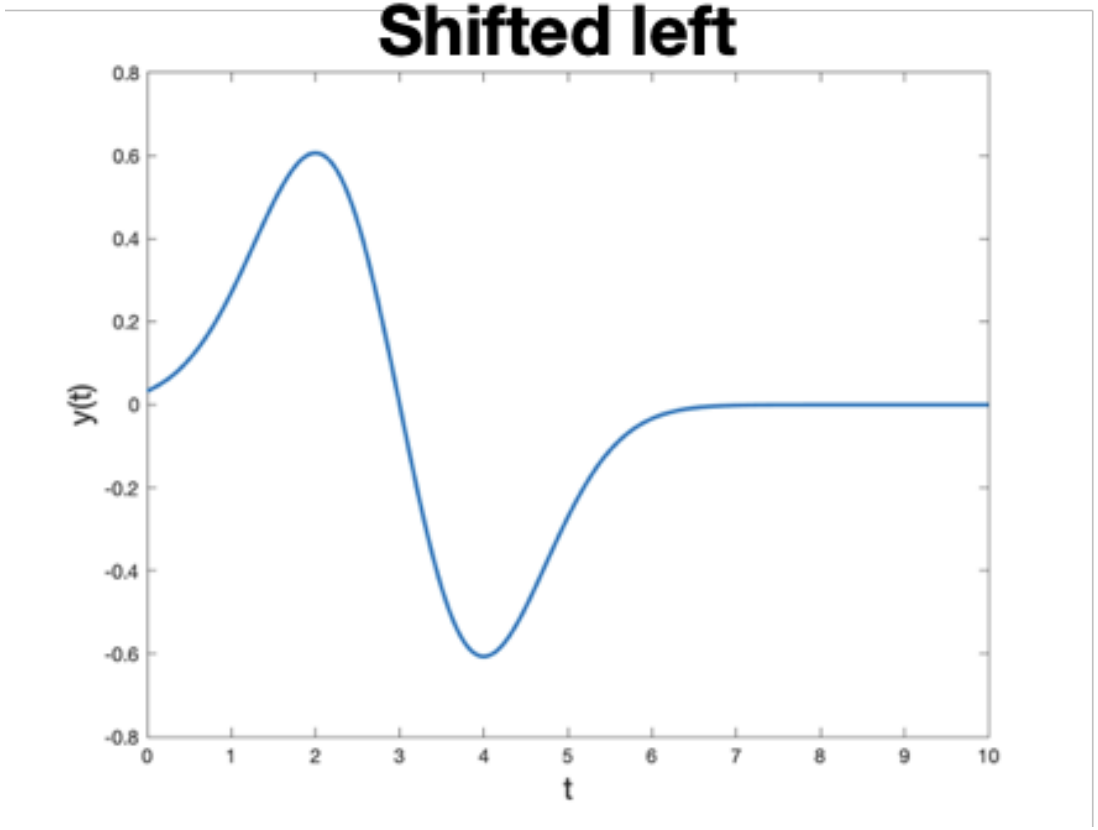


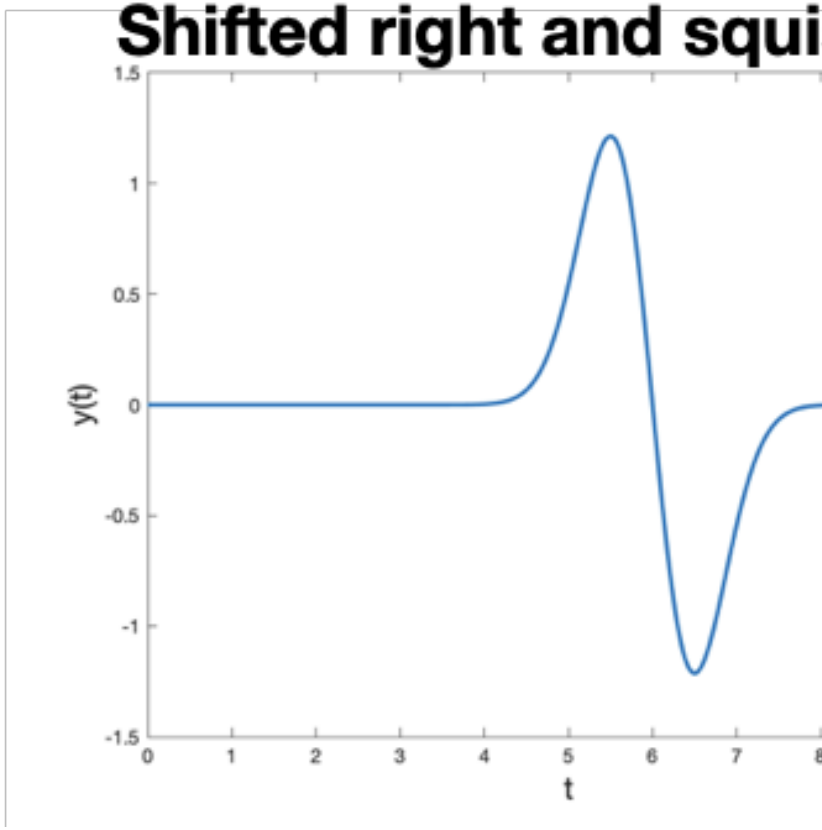


a

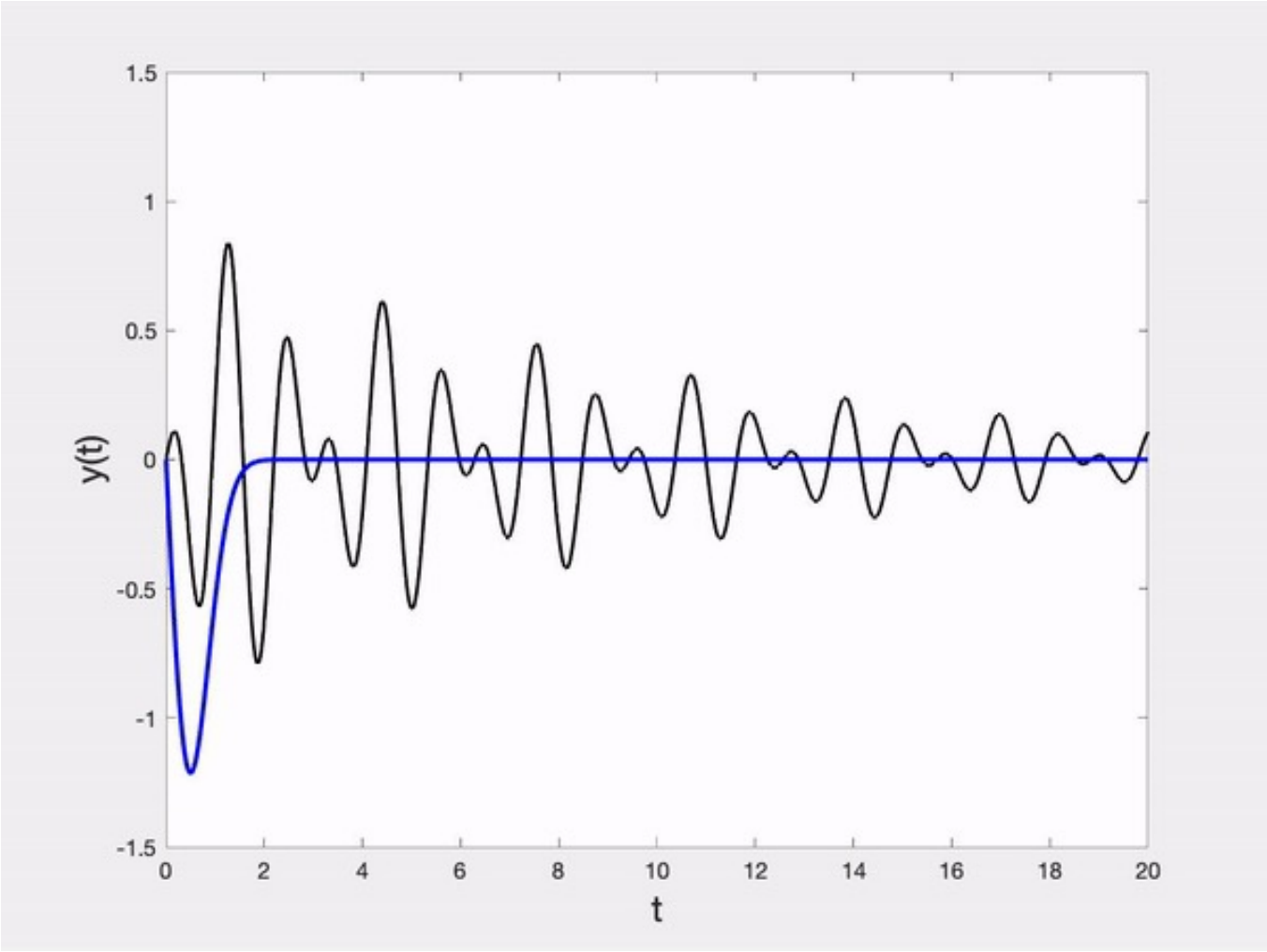
Shifting wavelets in time

to be centered on different vertice





Discrete Wavelet Transform



Wavelet Transform

Continuous Wavelet Transform (CWT)

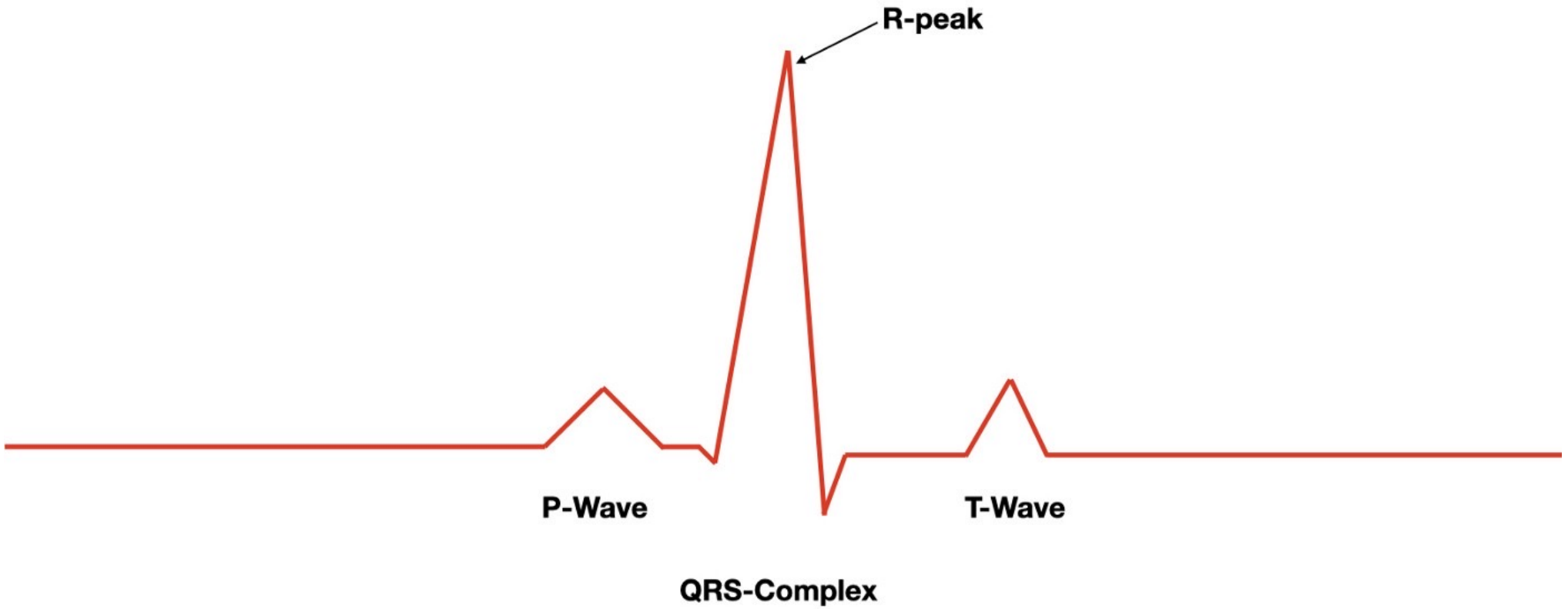
$$T(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\, \psi^* \frac{(t-b)}{a} dt$$

Discrete Wavelet Transform (DWT)

$$T_{m,n} = \int_{-\infty}^{\infty} x(t)\, \psi_{m,n}(t)\, dt$$

Signal

Sample Application peak detection in EEG

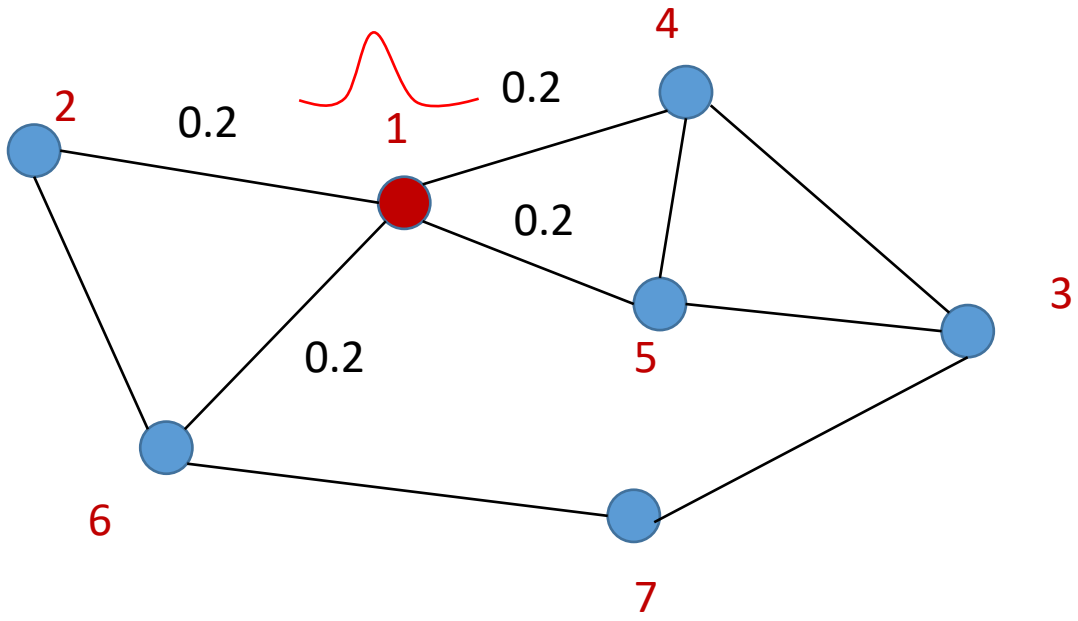


create wavelet by

Diffusion Operator

adjacency matrix A =

0	1	0	1	1	1	0
1	0	1	1	1	0	0
0	1	0	1	1	0	1
1	1	1	0	1	0	0
1	1	1	1	0	0	0
1	0	0	0	0	0	1
0	0	1	0	0	1	0



$$P = D^{-1}A$$

↑ Markov matrix

$D^{-1}A$: row normalized adjacency matrix

Same eigenvectors as the graph laplacian

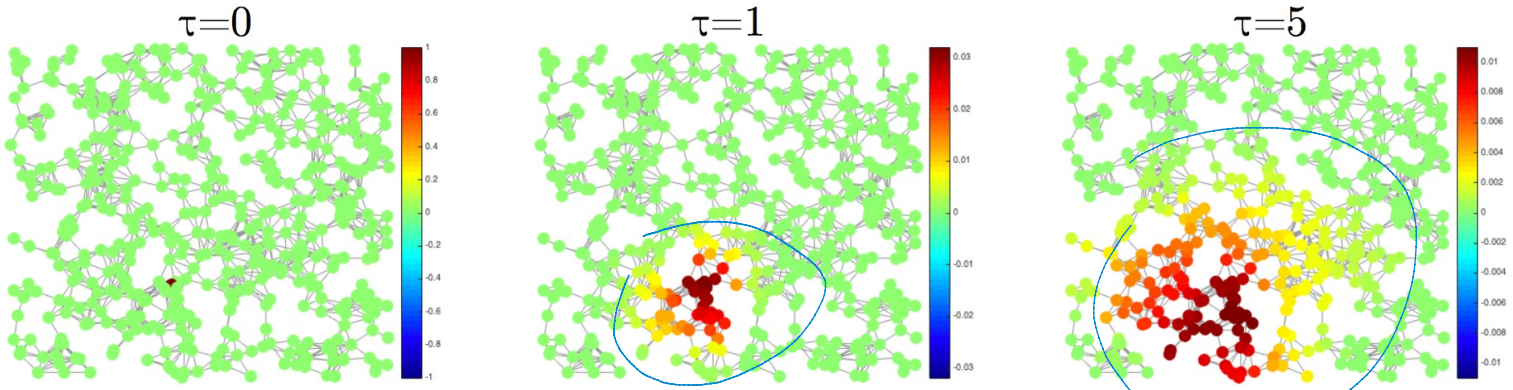
because $L = I - D^{-1}A$ I just flip eigenvalues

Diffusions can be used to create wavelets

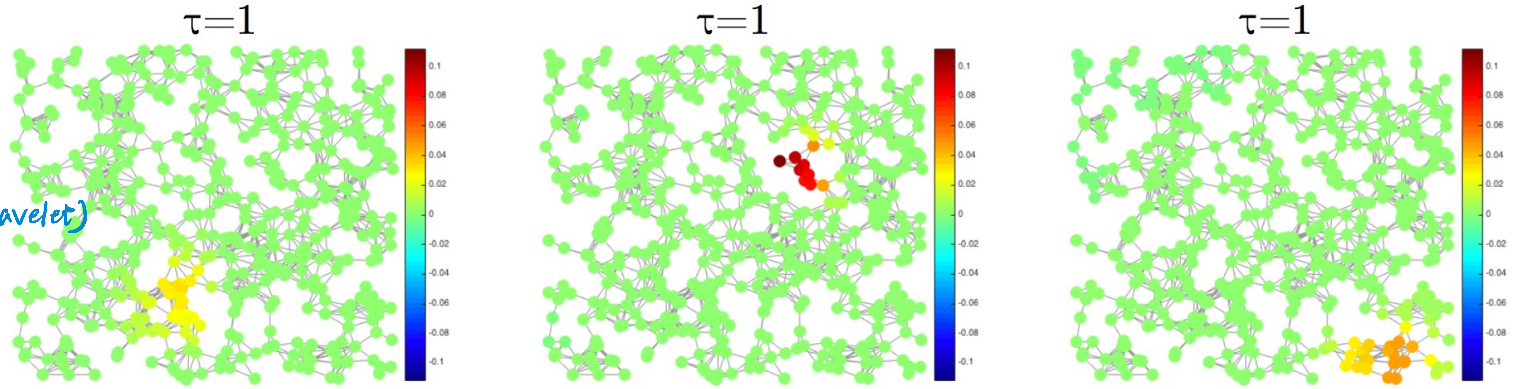
- Diffusion Atoms
- P^t s gives diffused version of S
- Diffusion based filters don't require eigendecomposition

Start with direct signal and apply diffusion operator → signal to diffuse

- Start with a unit of energy at a single vertex and let it diffuse:



- How much it diffuses over a fixed time depends on the graph structure:



Difference between 2 scales of diffusion are well known class of wavelet (diffusion wavelet)

Coifman and Lafon, Diffusion maps, ACHA, 2006

Diffusion wavelets: Differences between lazy Random walks

lazy walk = self-loop at vertice

D

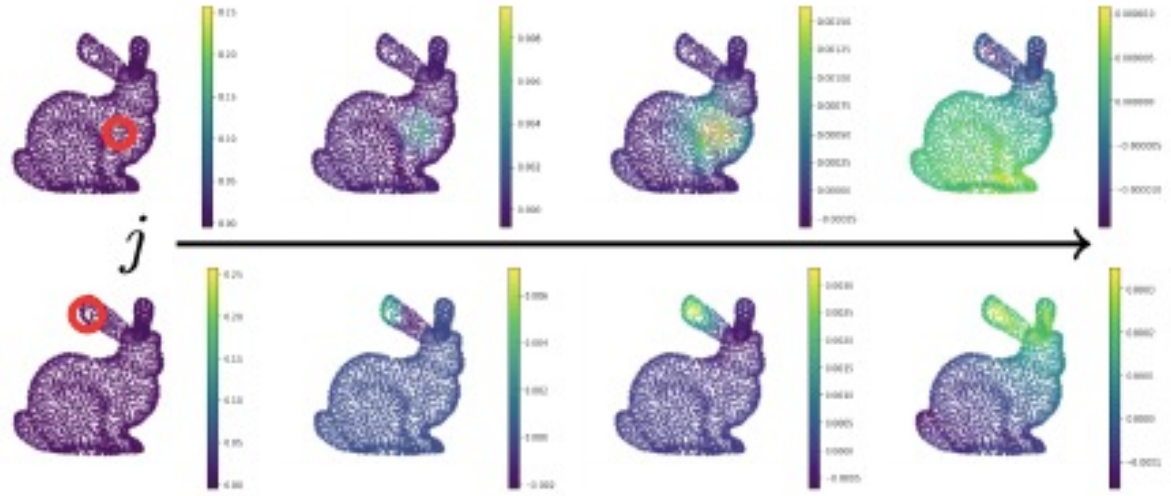
$$\mathbf{P} = \frac{1}{2}\left(\mathbf{I} + \mathbf{A}\mathbf{D}^{-1}\right)$$

wavelet

$$\mathbf{\Psi}_j = \mathbf{P}^{2^{j-1}} - \mathbf{P}^{2^j} = \mathbf{P}^{2^{j-1}}\left(\boldsymbol{I} - \mathbf{P}^{2^{j-1}}\right)$$

diffuse diadic steps

wavelets mark body



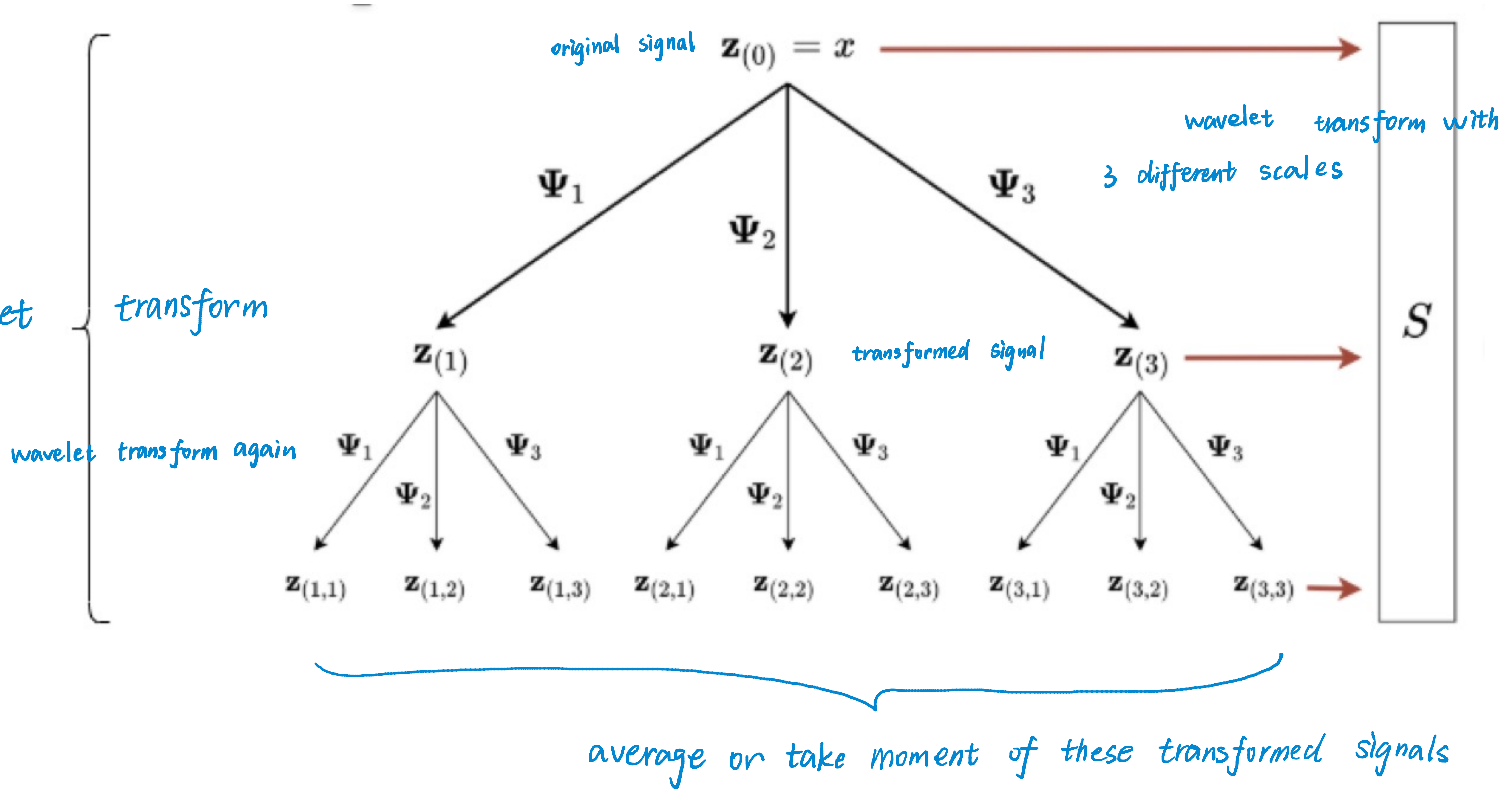
wavelets marked ear

Coifman and Maggioni

Scattering transform

Neural network

- Multiscale Wavelet transform
- Coefficients collected via a non-linearity



Graph Classification

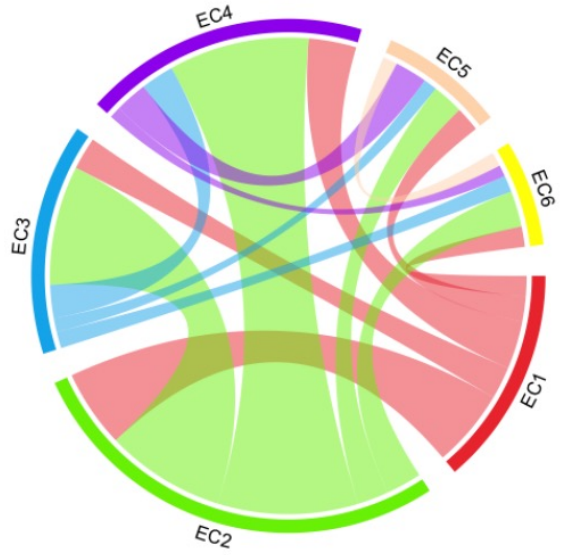
	COLLAB	IMDB-B	IMDB-M	REDDIT-B	REDDIT-5K	REDDIT-12K	
WL	77.82 ± 1.45	71.60 ± 5.16	N/A	78.52 ± 2.01	50.77 ± 2.02	34.57 ± 1.32	Graph kernel
Graphlet	73.42 ± 2.43	65.40 ± 5.95	N/A	77.26 ± 2.34	39.75 ± 1.36	25.98 ± 1.29	
WL-OA	80.70 ± 0.10	N/A	N/A	89.30 ± 0.30	N/A	N/A	
DGK	73.00 ± 0.20	66.90 ± 0.50	44.50 ± 0.50	78.00 ± 0.30	41.20 ± 0.10	32.20 ± 0.10	
DGCNN	73.76 ± 0.49	70.03 ± 0.86	47.83 ± 0.85	N/A	48.70 ± 4.54	N/A	Deep learning
2D CNN	71.33 ± 1.96	70.40 ± 3.85	N/A	89.12 ± 1.70	52.21 ± 2.44	48.13 ± 1.47	
PSCN ($k = 10$)	72.60 ± 2.15	71.00 ± 2.29	45.23 ± 2.84	86.30 ± 1.58	49.10 ± 0.70	41.32 ± 0.42	
GCAPS-CNN	77.71 ± 2.51	71.69 ± 3.40	48.50 ± 4.10	87.61 ± 2.51	50.10 ± 1.72	N/A	
S2S-P2P-NN	81.75 ± 0.80	73.80 ± 0.70	51.19 ± 0.50	86.50 ± 0.80	52.28 ± 0.50	42.47 ± 0.10	
GIN-0 (MLP-SUM)	80.20 ± 1.90	75.10 ± 5.10	52.30 ± 2.80	92.40 ± 2.50	57.50 ± 1.50	N/A	
GS-SVM	79.94 ± 1.61	71.20 ± 3.25	48.73 ± 2.32	89.65 ± 1.94	53.33 ± 1.37	45.23 ± 1.25	

Graph Classification

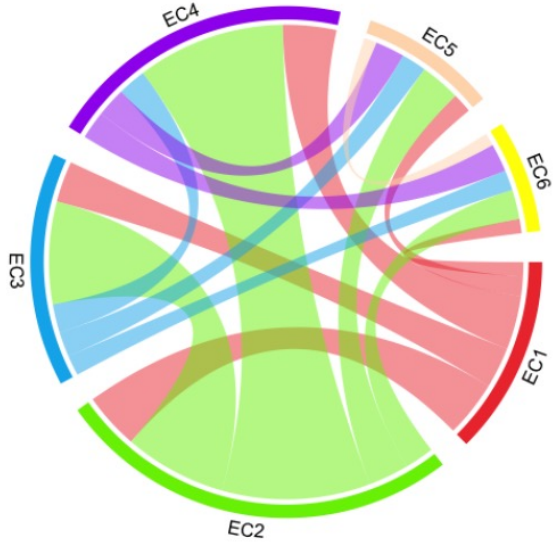
	COLLAB	IMDB-B	IMDB-M	REDDIT-B	REDDIT-5K	REDDIT-12K	
WL	77.82 ± 1.45	71.60 ± 5.16	N/A	78.52 ± 2.01	50.77 ± 2.02	34.57 ± 1.32	Graph kernel
Graphlet	73.42 ± 2.43	65.40 ± 5.95	N/A	77.26 ± 2.34	39.75 ± 1.36	25.98 ± 1.29	
WL-OA	80.70 ± 0.10	N/A	N/A	89.30 ± 0.30	N/A	N/A	
DGK	73.00 ± 0.20	66.90 ± 0.50	44.50 ± 0.50	78.00 ± 0.30	41.20 ± 0.10	32.20 ± 0.10	
DGCNN	73.76 ± 0.49	70.03 ± 0.86	47.83 ± 0.85	N/A	48.70 ± 4.54	N/A	Deep learning
2D CNN	71.33 ± 1.96	70.40 ± 3.85	N/A	89.12 ± 1.70	52.21 ± 2.44	48.13 ± 1.47	
PSCN ($k = 10$)	72.60 ± 2.15	71.00 ± 2.29	45.23 ± 2.84	86.30 ± 1.58	49.10 ± 0.70	41.32 ± 0.42	
GCAPS-CNN	77.71 ± 2.51	71.69 ± 3.40	48.50 ± 4.10	87.61 ± 2.51	50.10 ± 1.72	N/A	
S2S-P2P-NN	81.75 ± 0.80	73.80 ± 0.70	51.19 ± 0.50	86.50 ± 0.80	52.28 ± 0.50	42.47 ± 0.10	
GIN-0 (MLP-SUM)	80.20 ± 1.90	75.10 ± 5.10	52.30 ± 2.80	92.40 ± 2.50	57.50 ± 1.50	N/A	
GS-SVM	79.94 ± 1.61	71.20 ± 3.25	48.73 ± 2.32	89.65 ± 1.94	53.33 ± 1.37	45.23 ± 1.25	

Embedding Analysis

embedding = scattering coefficient



(a) Observed



(b) Inferred

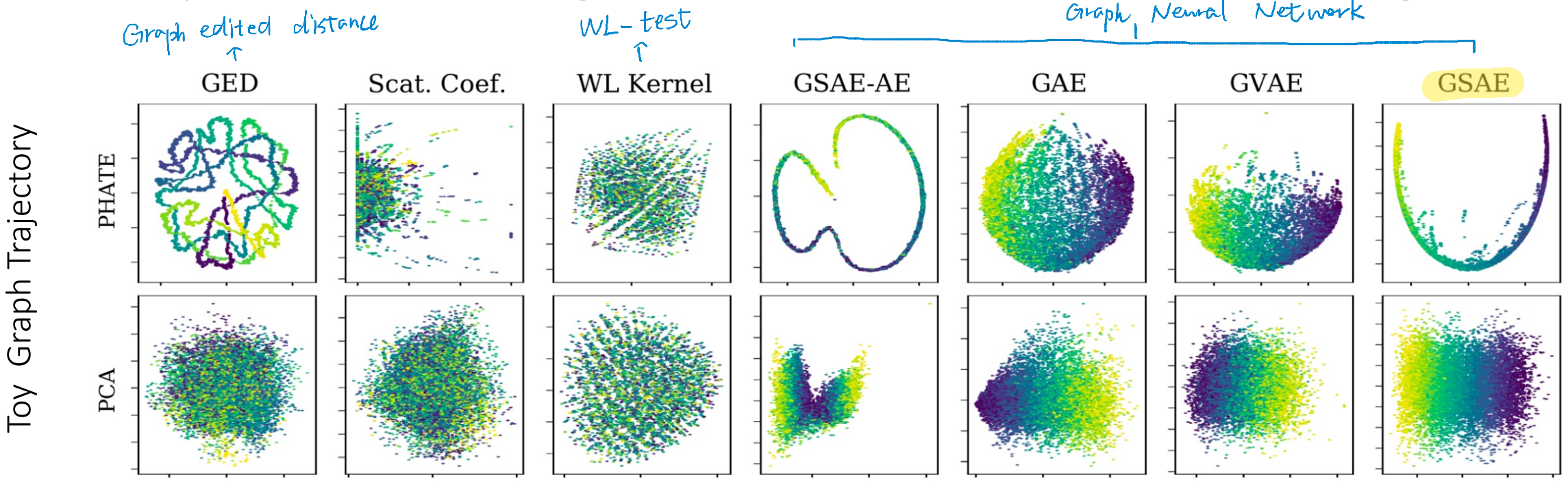
ab are similar

:)

Graph scattering transform faithfully capture structure of molecular

GSAE

Graph Scattering Autoencoder Embeddings



high dim

input : graph scattering coefficients

↓

Autoencoder → low dim embeddings

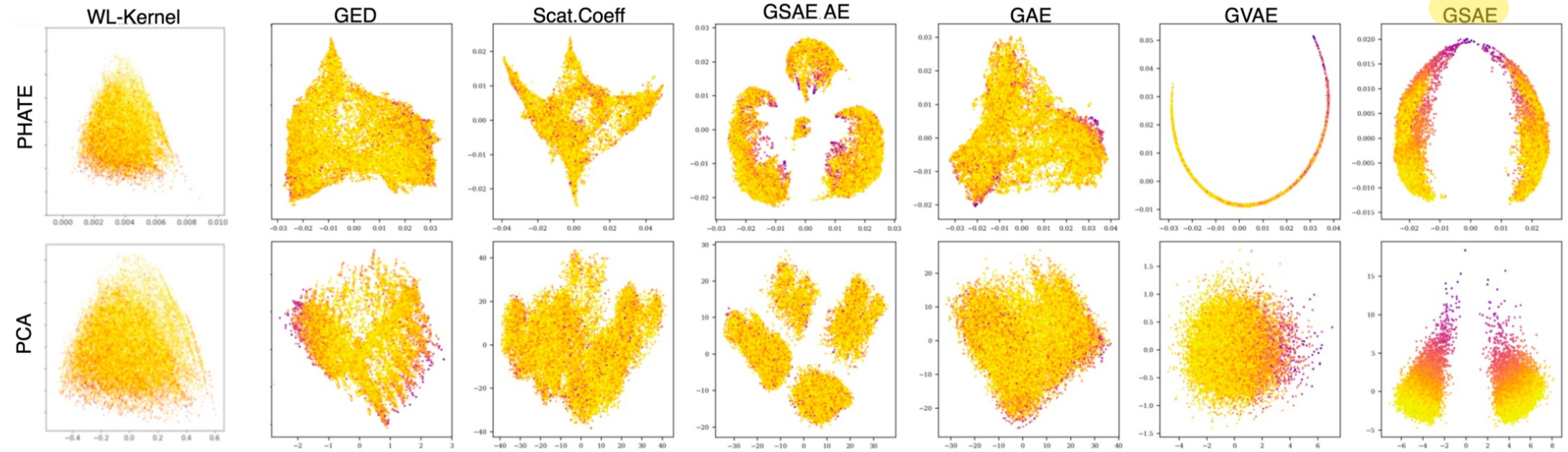
Castro et al. 2019

Bistable RNA: non-coding RNA with 2 stable folds (low energy)

Bistable RNA Structure

clearly show 2 freq structure

SEQ3



Reading list

- Defferrard et al. Convolutional Neural Networks on Graphs

- Kipf & Welling Semisupervised Graph Classification

- https://towardsdatascience.com/spectral-graph-convolution-explained-and-implemented-step-by-step-2e495b57f801
- Graph wavelet neural network https://arxiv.org/pdf/1904.07785.pdf

- Graph scattering Autoencoder: https://arxiv.org/abs/2006.06885
- Learnable scattering network: https://arxiv.org/abs/2010.02415

- Diffusion wavelets: https://www.sciencedirect.com/science/article/pii/S106352030600056X