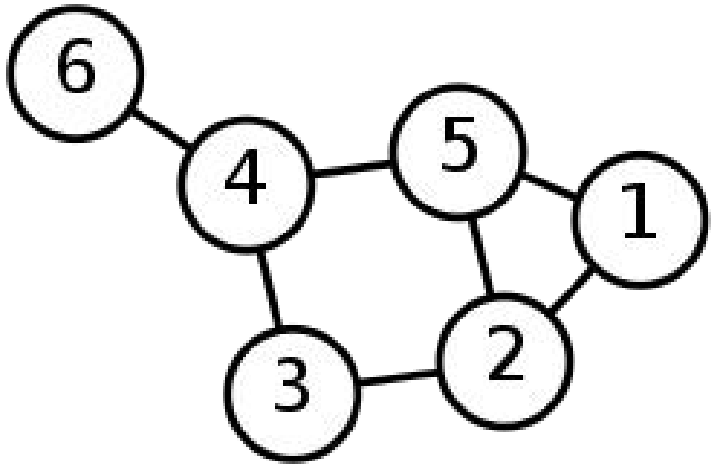
The Graph Laplacian and Graph Signal Processing

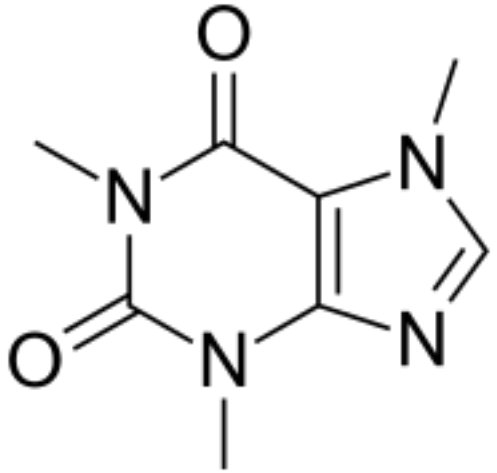
Graphs Review



- A set of vertices
  $V = \{v_1 \quad v_2 \quad \cdots\}$
- Edges that connect them
- $E \in VxV$
- Described by adjacency matrices A
- $A_{i,j}$ is the weighted connectivity between nodes *i and j*

Data Graphs: Natural vs Affinity Graphs

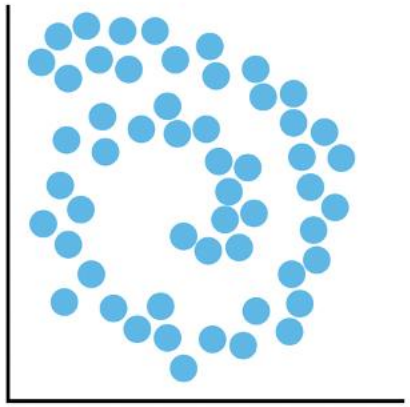
- Some data naturally occur as graphs



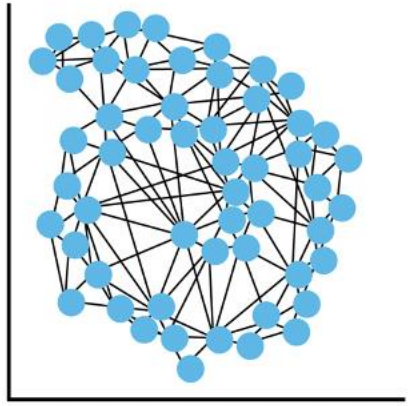
- Other types of data occur as points in n-dimensional space and a graph is induced on them

Graphs Induced by Data

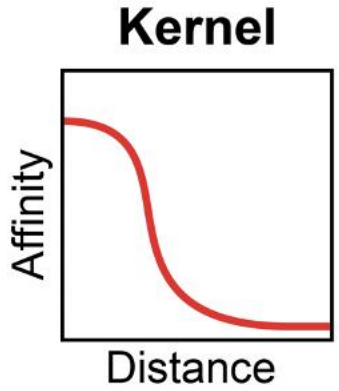
Data in two dimensions



Distances between all points are calculated



A kernel function calculates affinities from distance



Graph Laplacian L

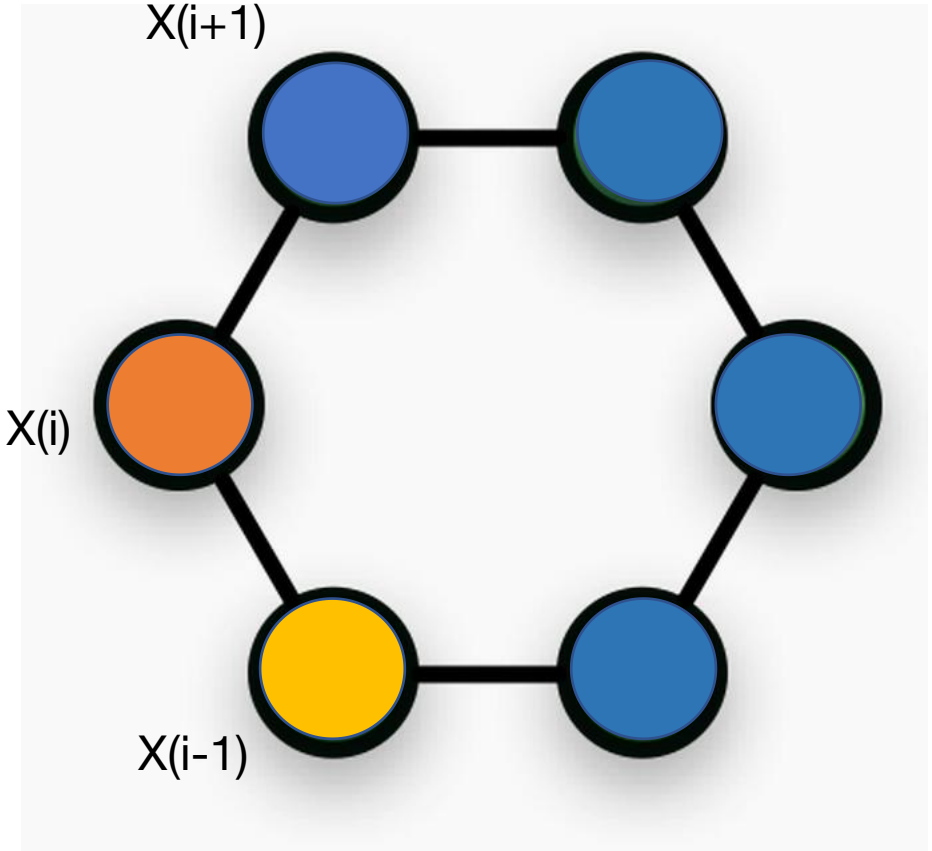
- A difference operator based on the graph adjacency matrix A.
- unnormalized Laplacian: $L = D - A$
  - Degree matrix: $D_{ii} = \sum_j A_{ij}$, 0 everywhere else
  - D is row sum of A
  - L isn't symmetric
- normalized Laplacian: $L = I - D^{-1/2} A D^{-1/2}$
  - I is identity
  - Measures how similar a point is to its neighbors
- random walk Laplacian: $L = I - D^{-1} A = I - M$
  - Related to Markovian matrix M

Laplacian vs Markov Matrix

- $L=I-D^{-1}A$ is Laplacian
- $M=D^{-1}A$ is the Markov affinity matrix
- Same eigenvectors
- The ordering of eigenvectors is flipped
- Eigenvalues are now $1-\lambda_i$

Second derivative operator

- 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)



Based on Adjacency Matrix

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

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

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

Multiply X by L can Estimate smoothness of X at each p

$= LX$ quadratic normal form $s = X^T LX$

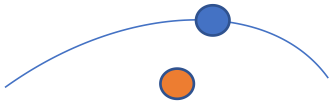
use as regularization in NN

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

Laplacian Measures Curvature

- Limit as you have infinitely many points goes to curvature

Average of neighbors is itself

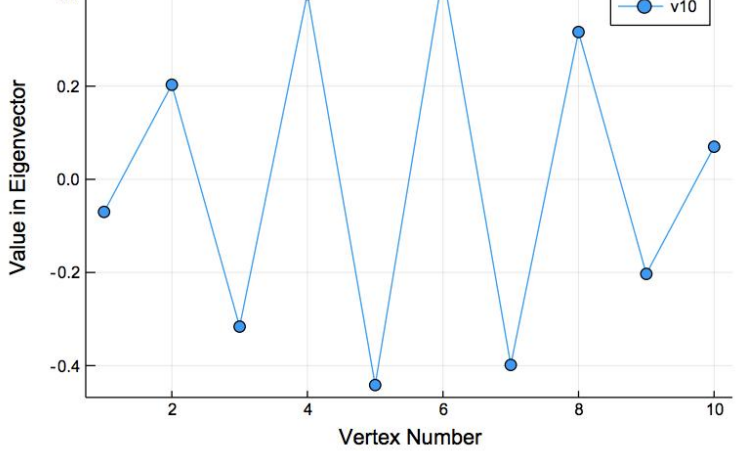


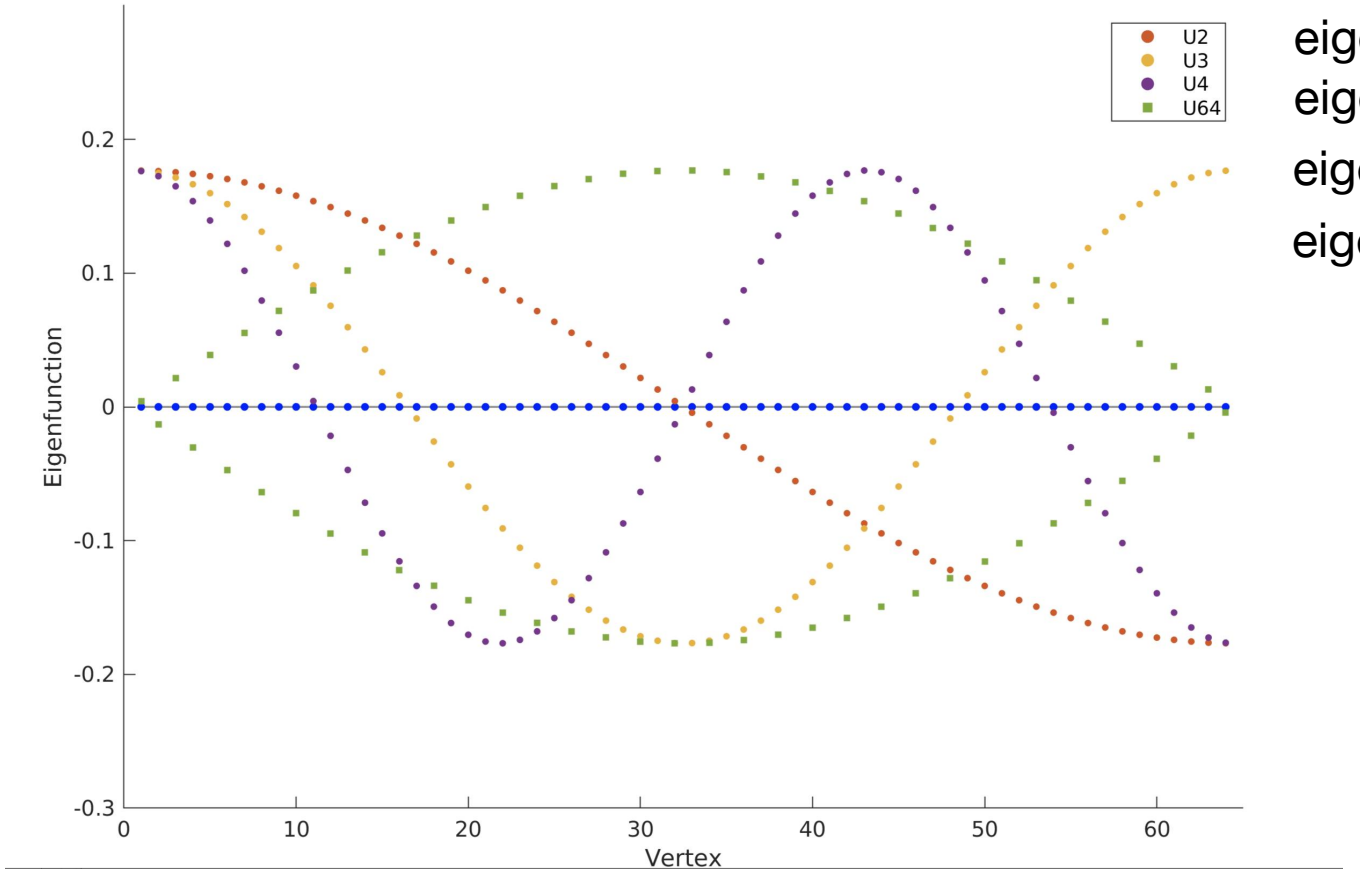
Average of neighbors not on the curve

Eigenvectors are Frequency Harmonics

- continuous y (signal)
  - The second derivatives is an operator $\frac{\partial^2 y}{\partial x^2}=\lambda y$
  - $y = \sin(\omega x)$ Similarly with y= cos(wx)
  - first derivative $\frac{dy}{dx} = \omega\cos(\omega x)$
  - second derivative is $\frac{\partial^2 y}{\partial x^2} = -\omega^2\sin(\omega x)$
- discrete y
  - Eigenvectors of Graph Laplacian Matrix are “spectrum of Fourier harmonics”
  - Eigenvalues would be similar to $-\omega^2$
  - eigenvectors are same as Markov matrix M, but eigenvalues are flipped

Eigenvectors on a line graph





eigenvector of smallest eigenvalue

eigenvector of 2nd smallest eigenvalue

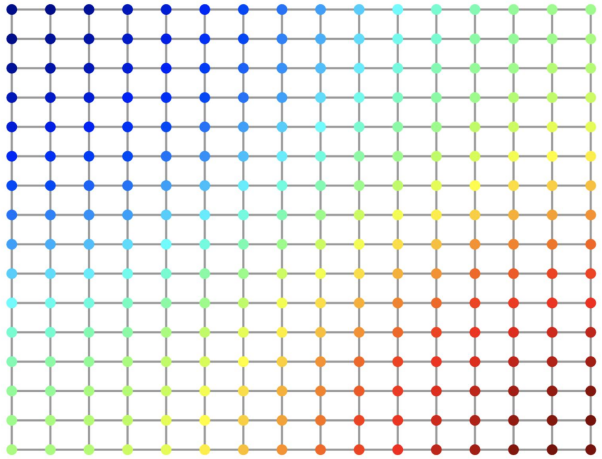
eigenvector of 5th eigenvalue

eigenvector of 65th eigenvalue

eigenvector of Adjacent matrix

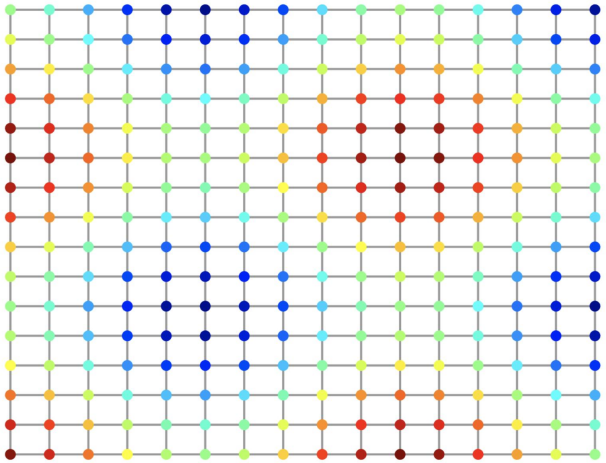
Laplacian Eigenvectors on a grid graph

Low frequency



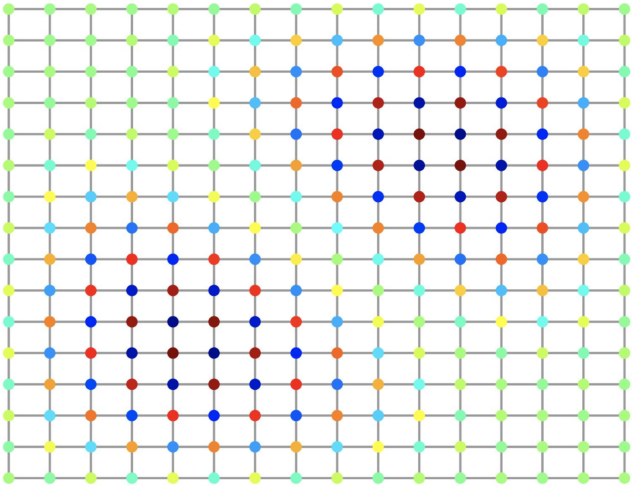
2nd lowest Eigenvector

Medium frequency



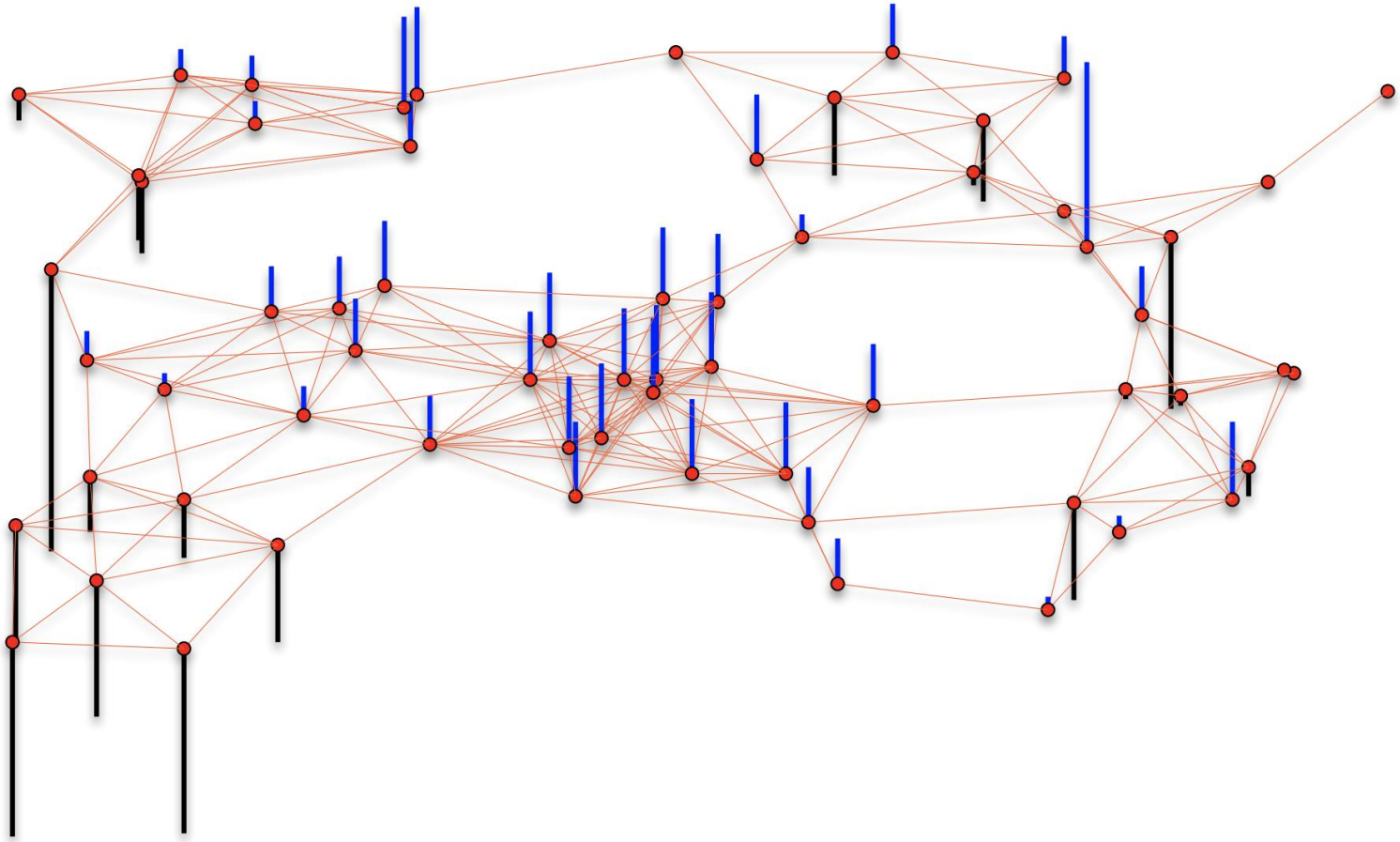
10th lowest Eigenvector

High frequency



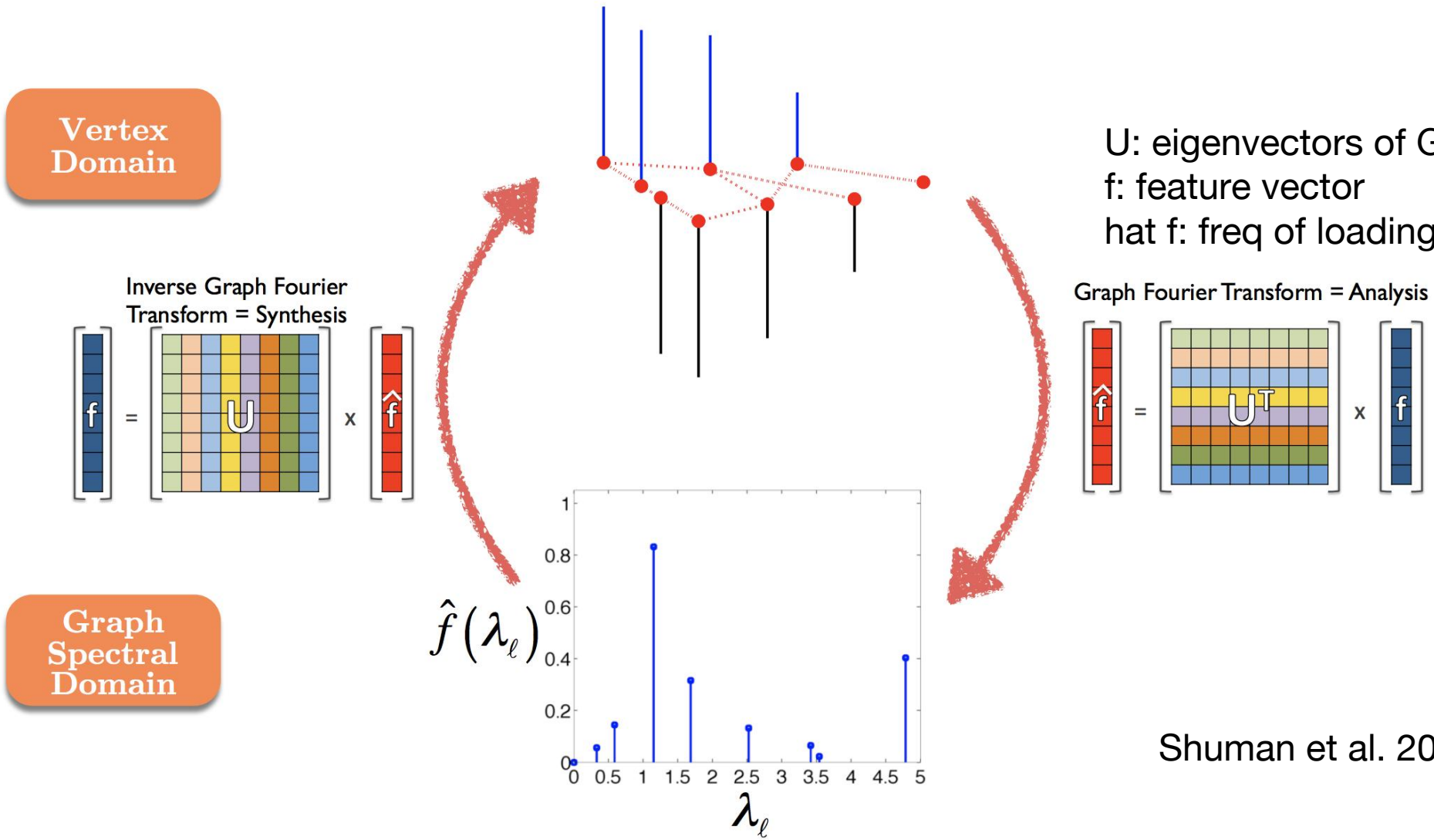
2nd highest eigenvector

Features are signals on a graph



Graph Fourier Transform

- is a linear transform



U: eigenvectors of Graph Laplacian
f: feature vector
hat f: freq of loading

Shuman et al. 2013

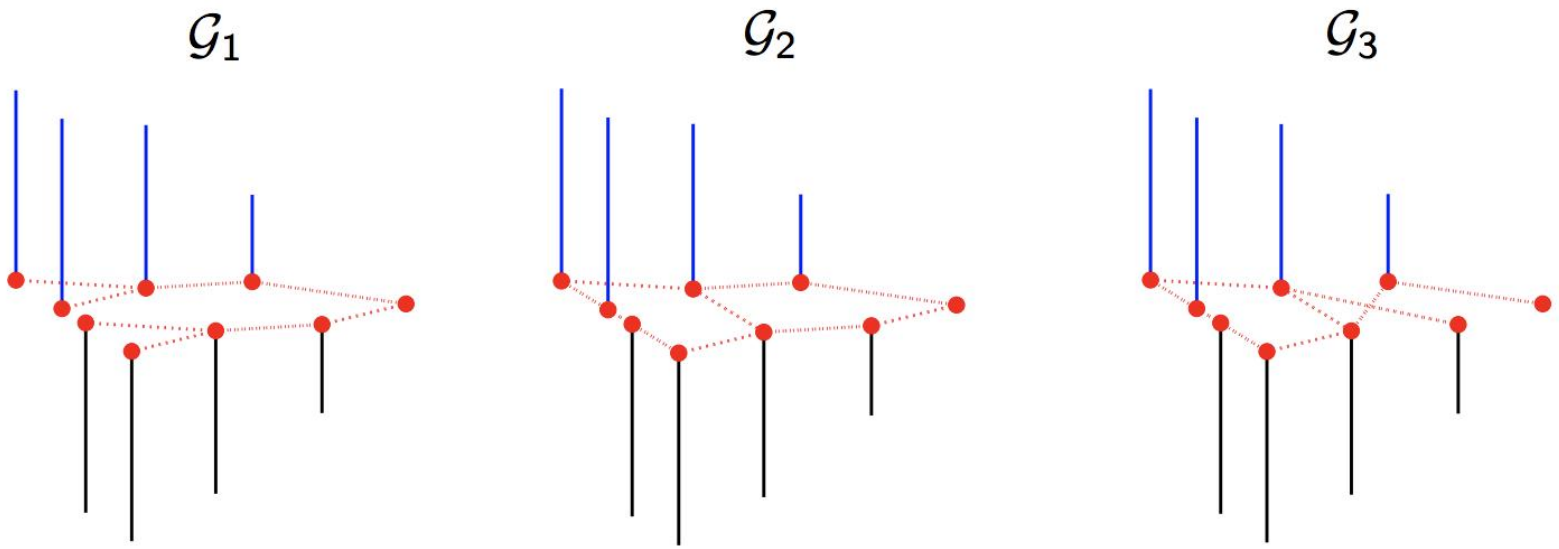
Regular Signal Processing vs Graph Signal Processing

have gap between freqs

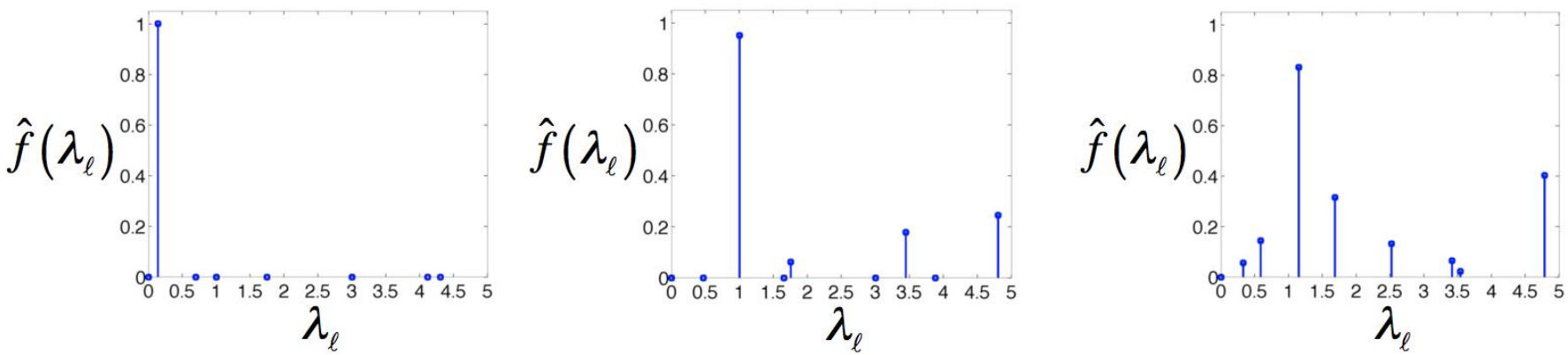
	Regular	Graph
Similar	Signals are observations	Signals are features of data points
	Signals over time (continuous axis) e.g. audio, radio	Signals over a graph (discrete point set)
	Signals is decomposed into sine/cosine	Signals is decomposed into eigenvector basis of graph Laplacian matrix
	Filter is a modification of Fourier coefficients over sine/cosines of different frequencies	Filter is a modification of Graph Fourier coefficients over eigenvectors of different frequencies
Different	Clean notion of translation	Less clean notion of translation Hard to interpret what means to translate regular/irregular graph

a regular graph: **each vertex has the same number of neighbors**;
i.e. every vertex has the same degree or valency

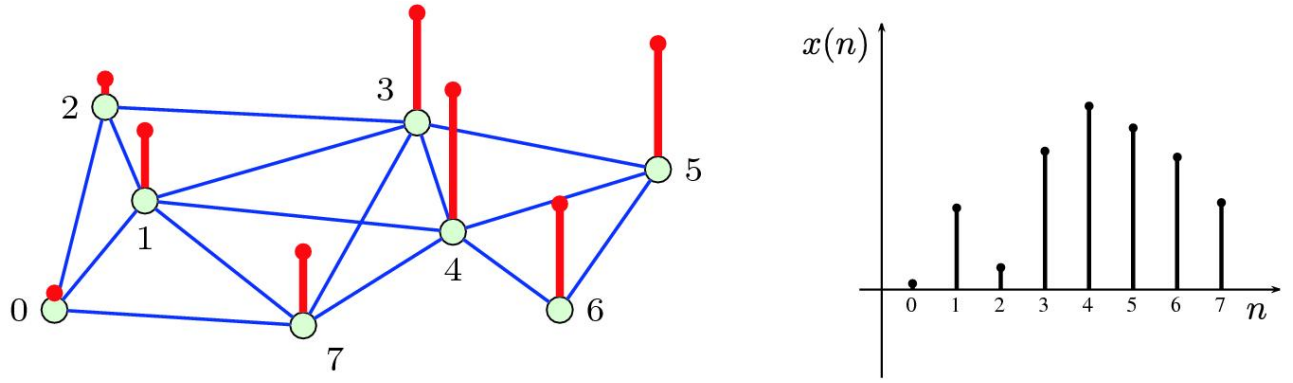
Vertex Domain



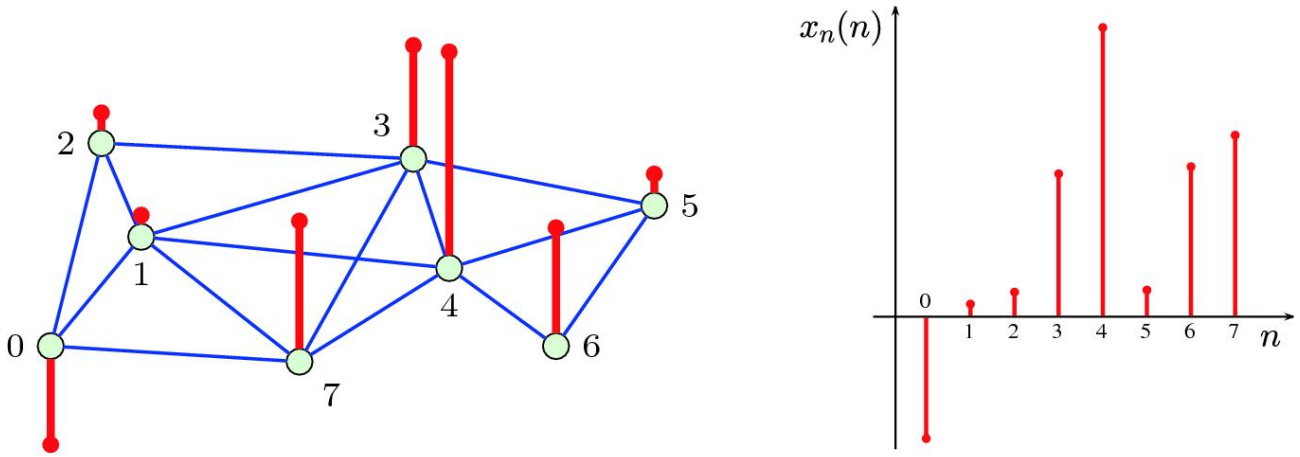
Graph Spectral Domain



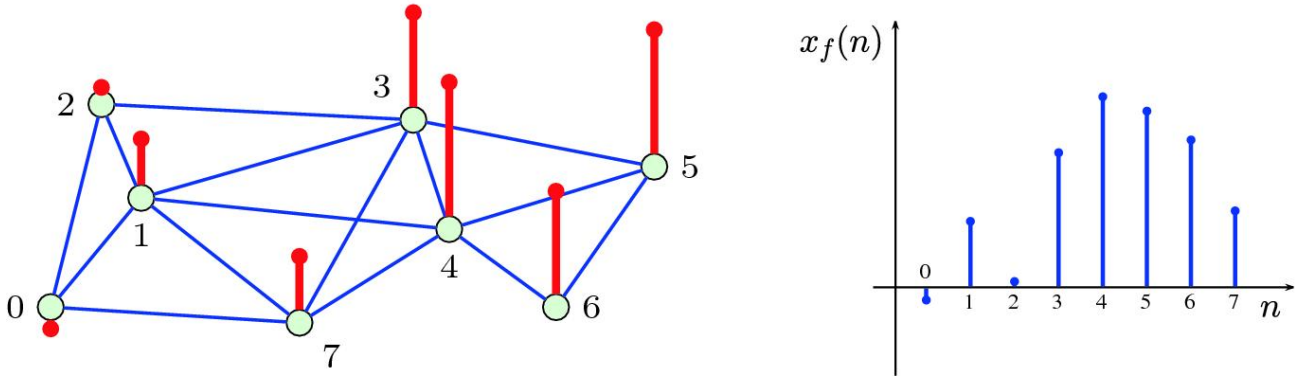
Graph Filtering



(a) original signal



(b) noisy signal



(c) filtered signal

Stankovic et al. 2018

Exercise

- Take the dataset you created for diffusion, and compute a Laplacian
- Visualize the eigenvectors
- Do the same for the swiss roll

Graph Filtering and Denoising

General Filter Construction

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

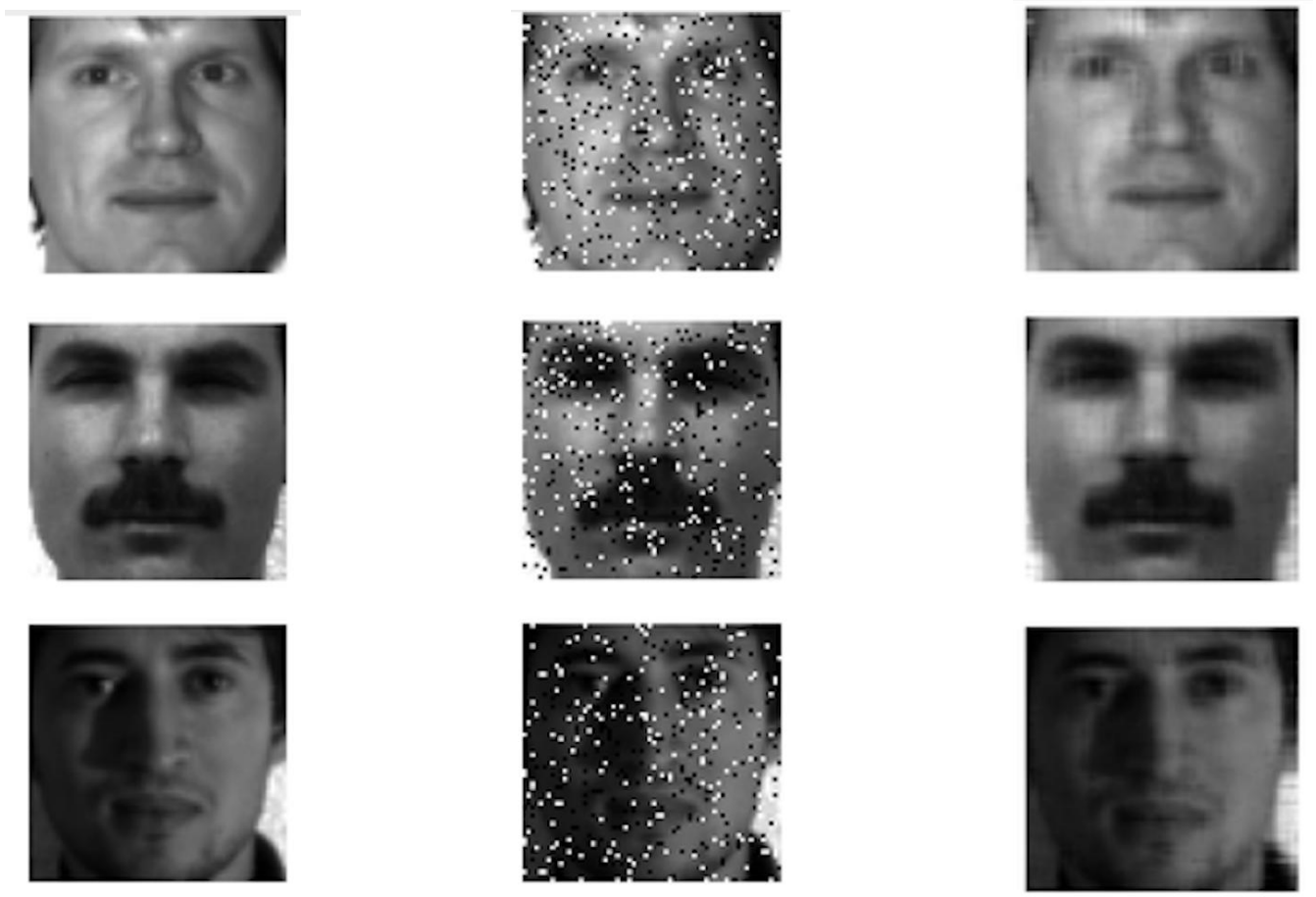
- Modulate and alter eigenvalues $L = UH(\Lambda)U^T$
- $$\begin{bmatrix} & L & \end{bmatrix} = \begin{bmatrix} | & | & | \\ U_1 & U_2 & U_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} H(\lambda_1) & & \\ & H(\lambda_2) & \\ & & H(\lambda_3) \end{bmatrix} \begin{bmatrix} | & | & | \\ U_1 & U_2 & U_3 \\ | & | & | \end{bmatrix}^{-1}$$

- Apply it to the signal $UH(\Lambda)U^T X$
- Example of a low pass filter $H(\lambda) = e^{-\lambda}$
- Applying $UH(\Lambda)U^T X$ would diminish high frequency components

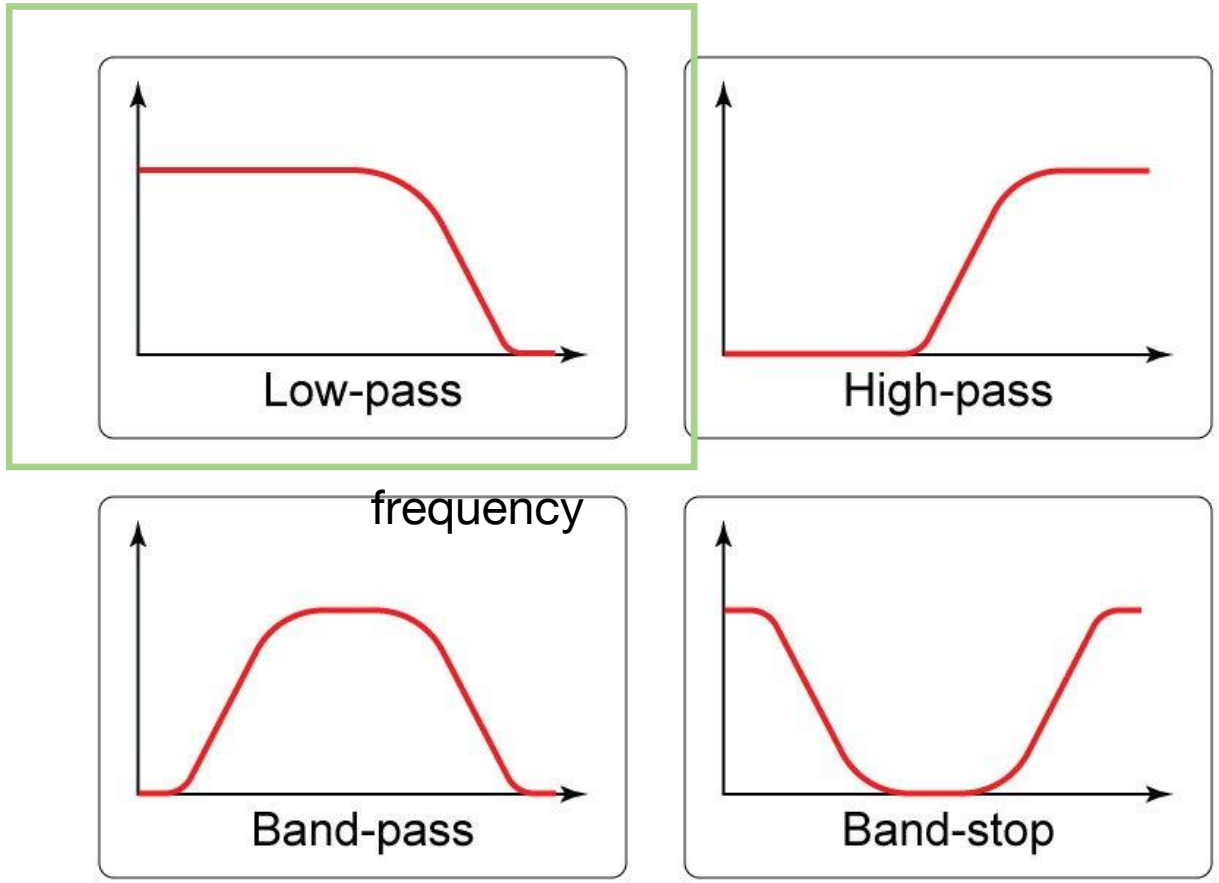
Filtering can be used to denoise

- graph filter: take off noise by taking off high frequency eigenvectors (with high eigenvalues)
  - reason: Data signals are low frequency, Noisy can be high frequency
- VS. PCA denoising: take off eigenvectors with low eigenvalue

add noise



Low-pass filter: keep low freqs and remove high freqs

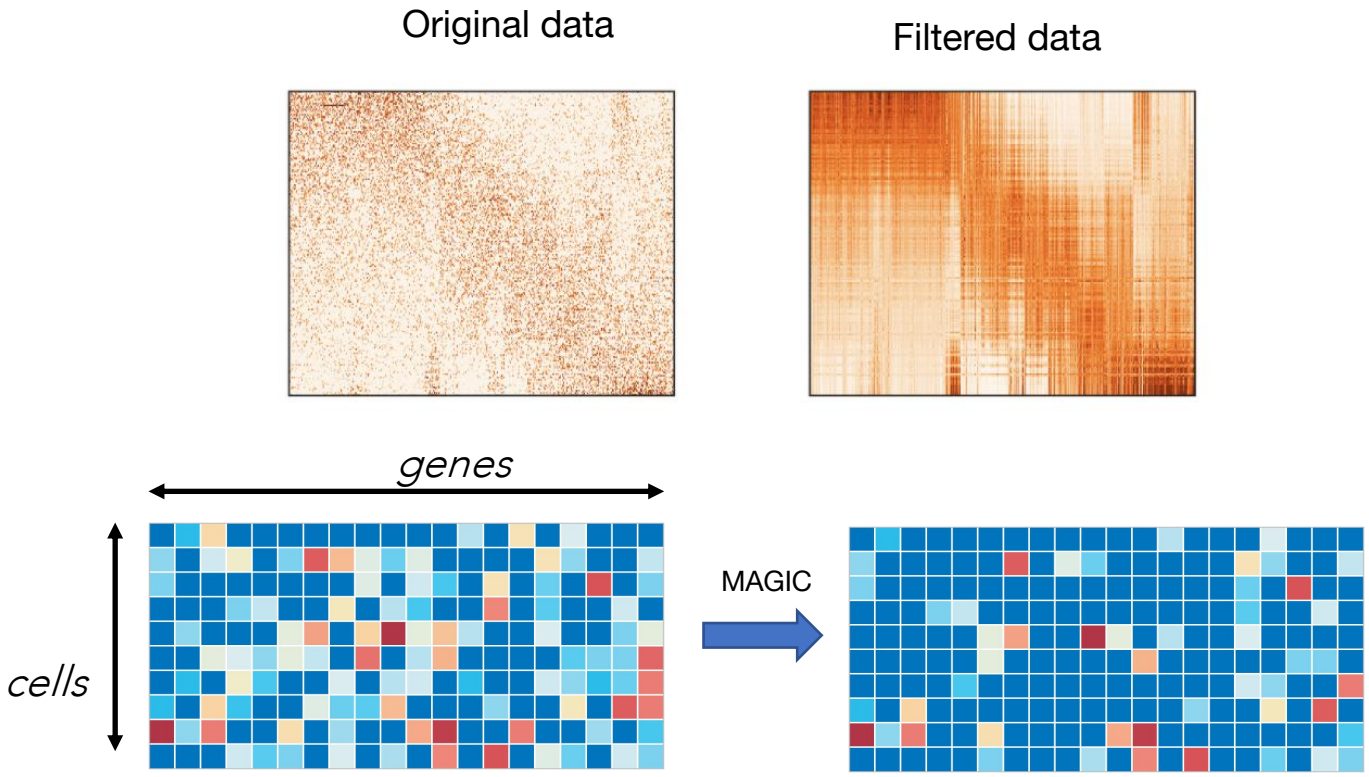


MAGIC-Manifold Denoising

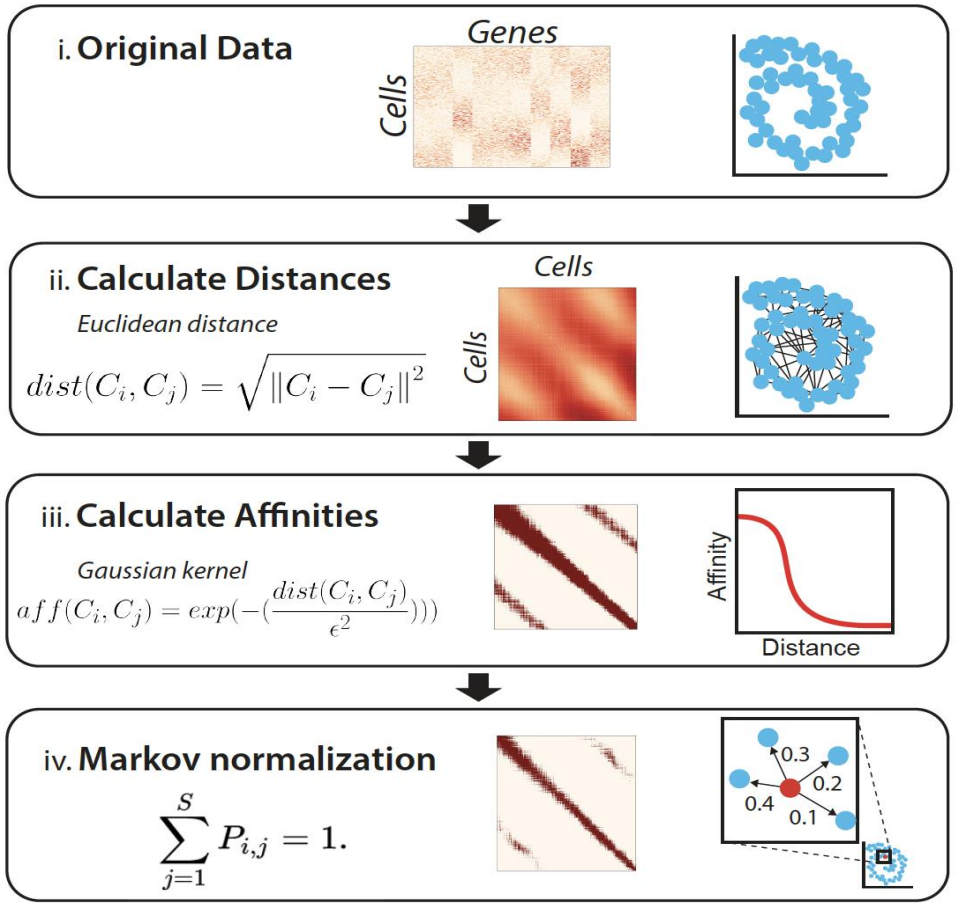
Markov Affinity-based Graph Imputation of Cells
(van Dijk et al. Cell 2018)

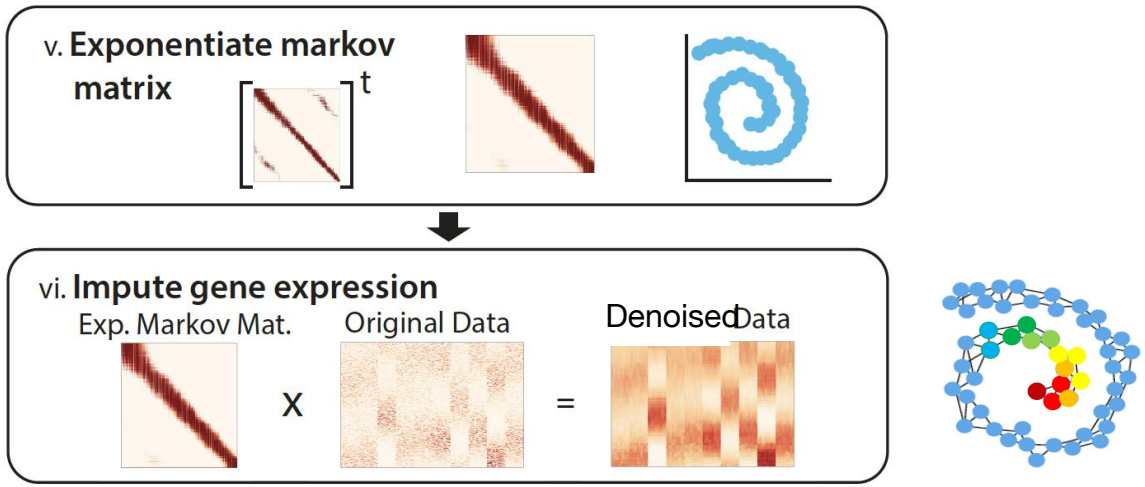
Main idea: scRNA-seq Data is noisy and sparse

Use a **low-pass filter** to deniose gene signals on a cell-cell graph.

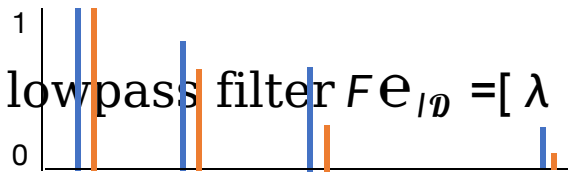


Graph filtering

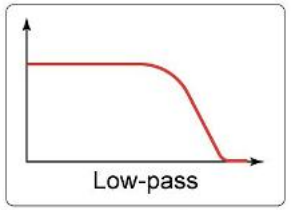




filtered signal $S_{filtered}$ = $IGFT(F_{l} * GFT(S))$



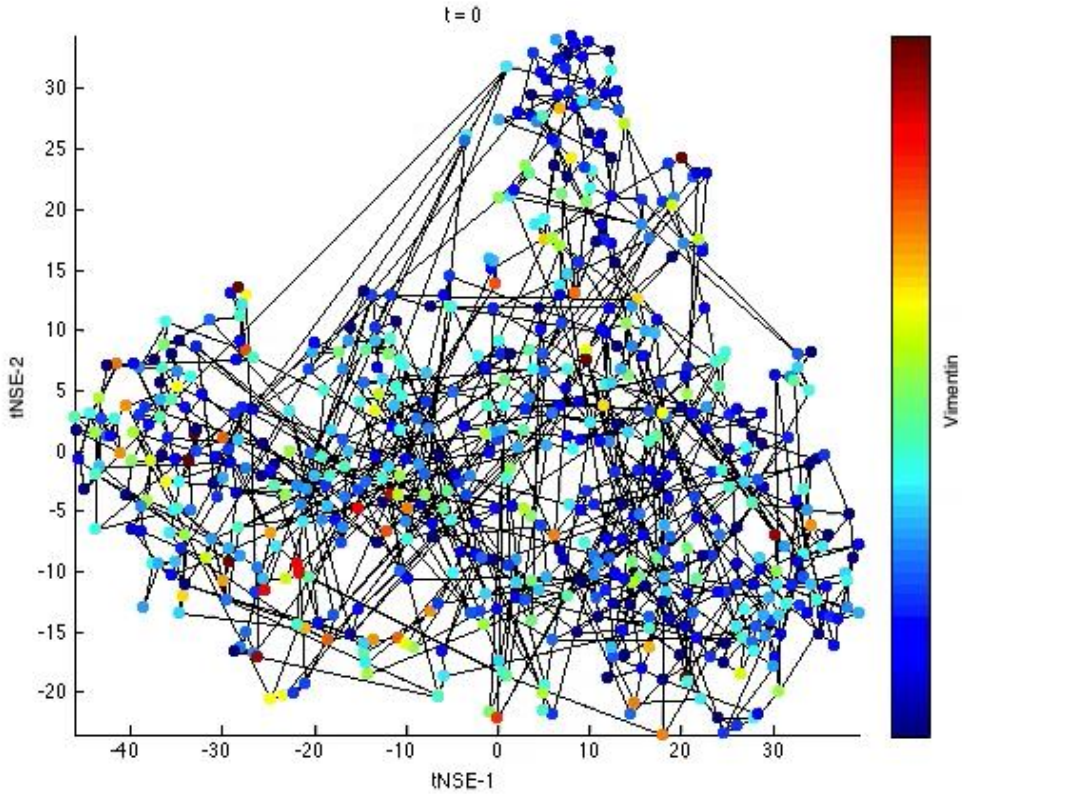
lowpass filter $Fe_{l\mathcal{D}}$ =[λ_0^t, λ



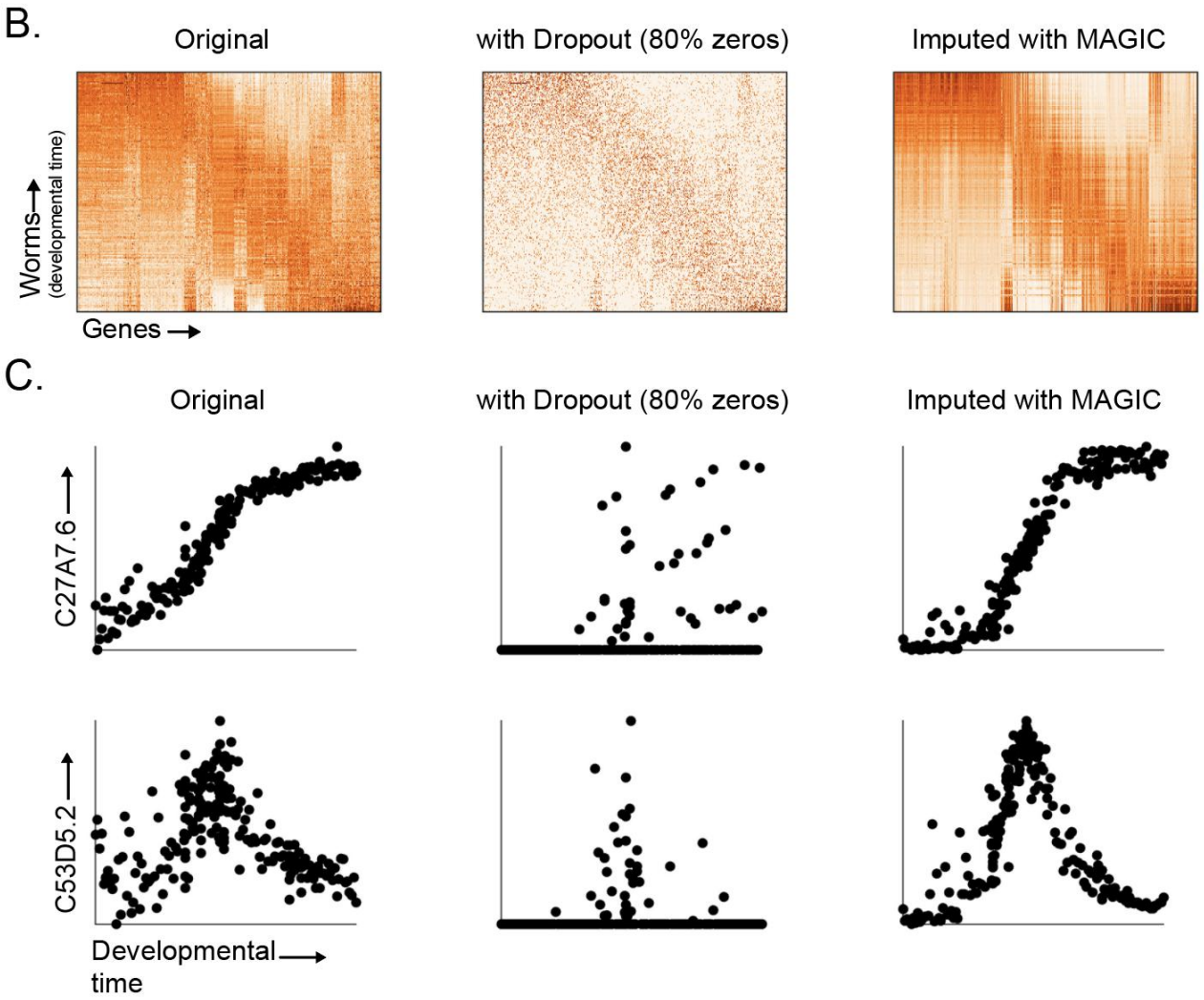
power eigenva

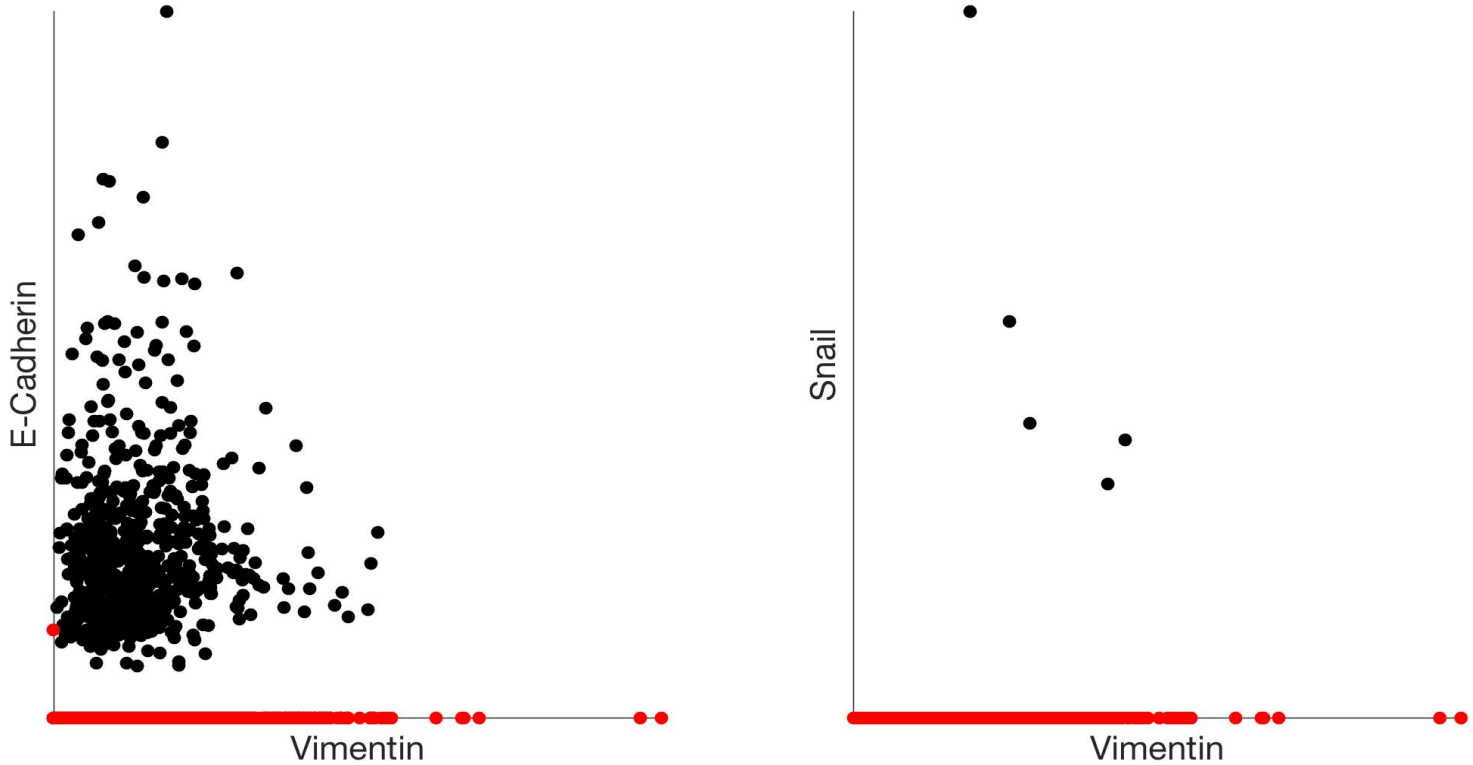
Vertex Domain

- Smooths signal on graph
- Takes weighted average of neighbor

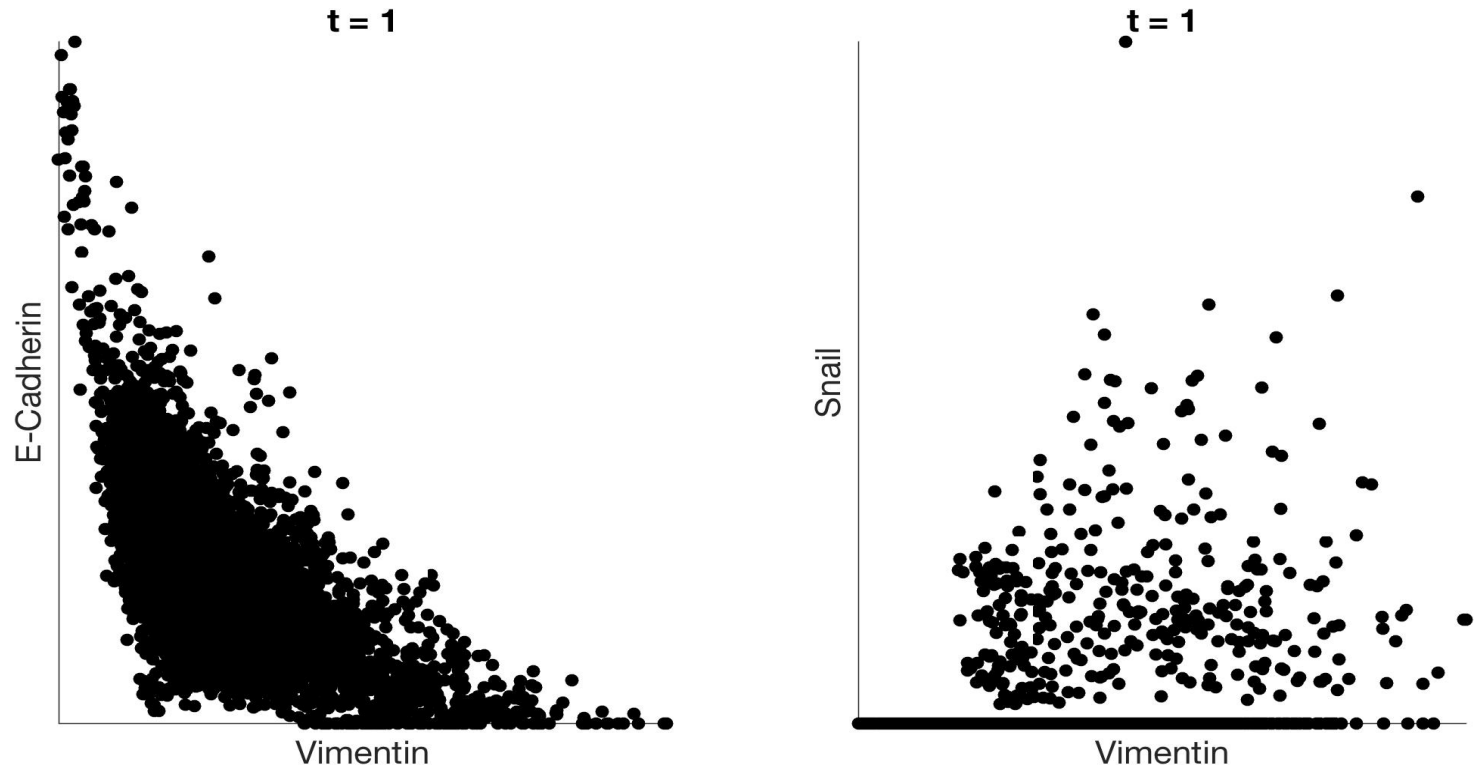


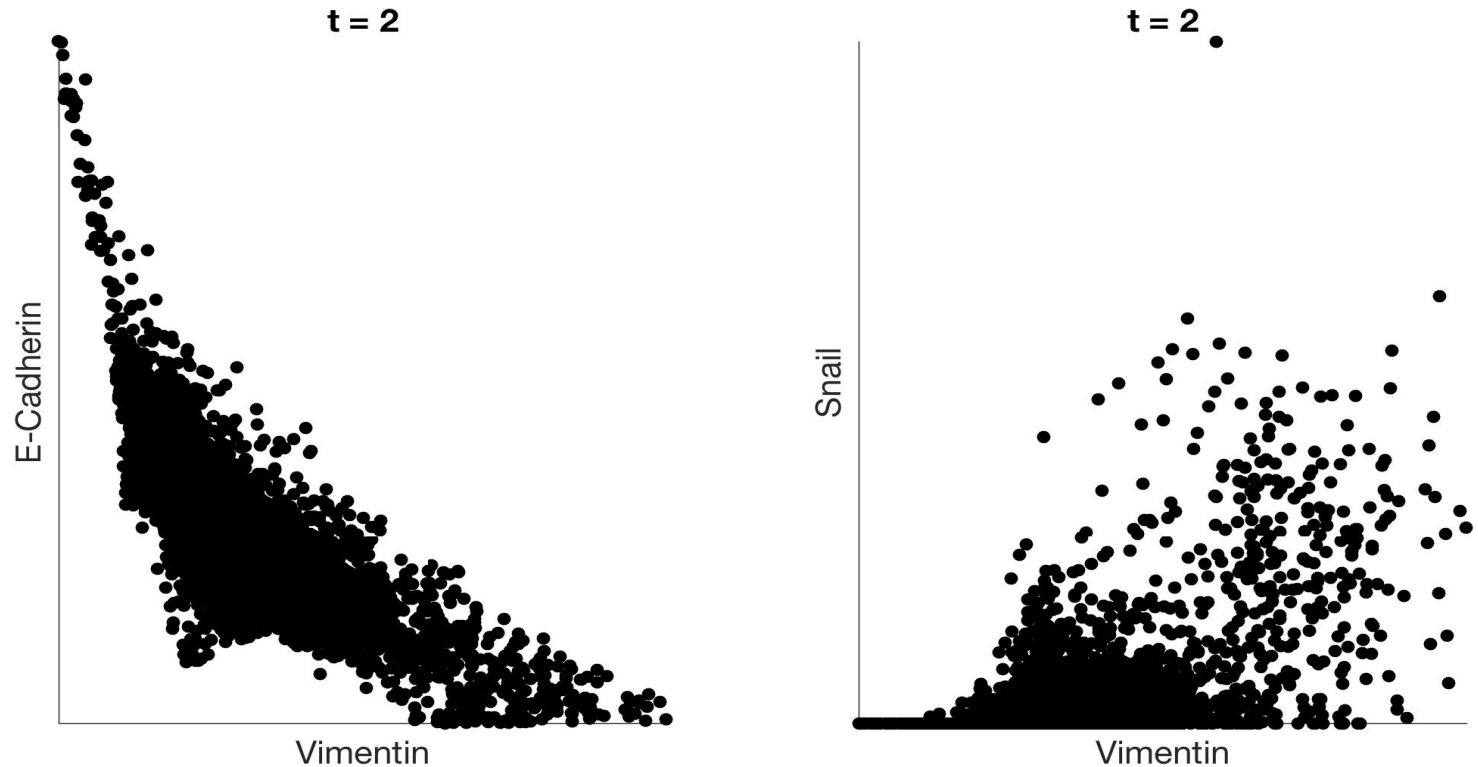
MAGIC recovers gene-gene relationships in an artificially dropped-out dataset

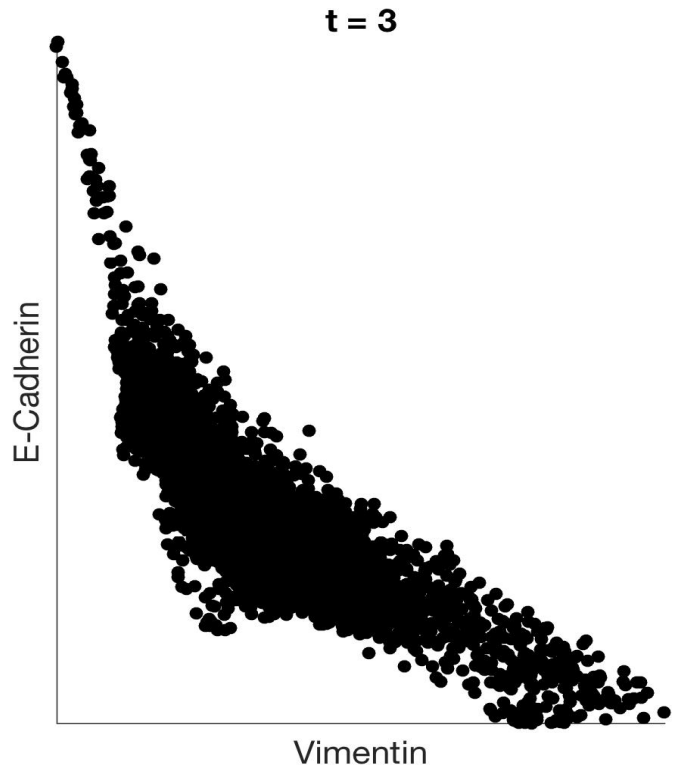


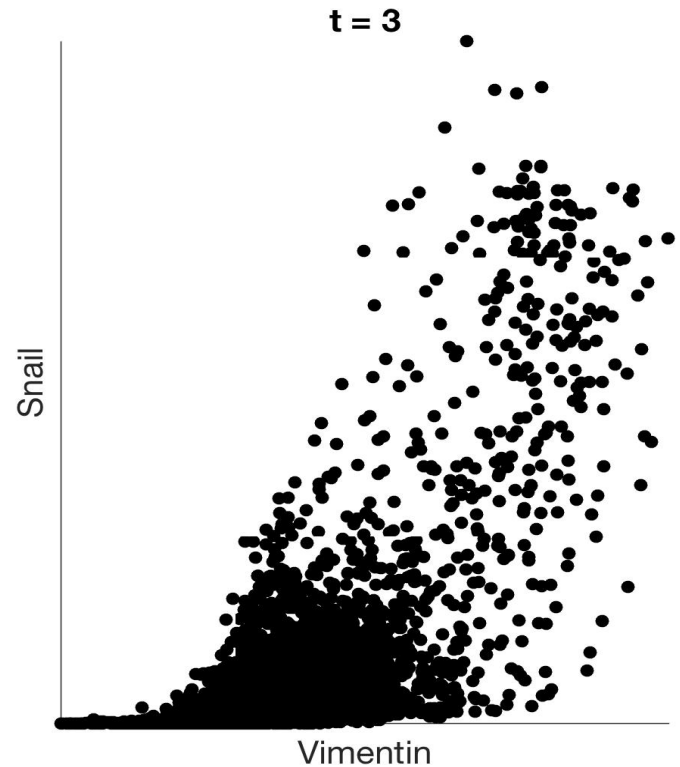


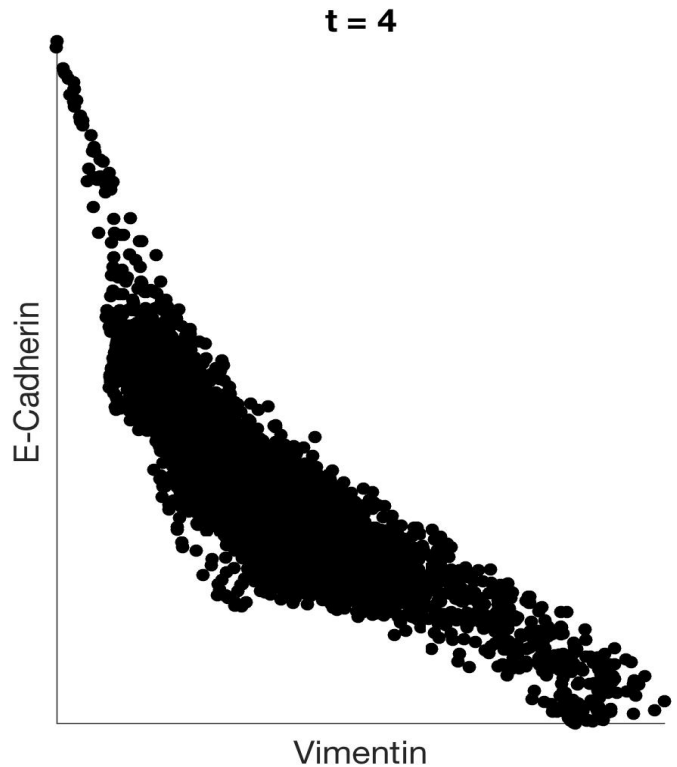
original data with dropout

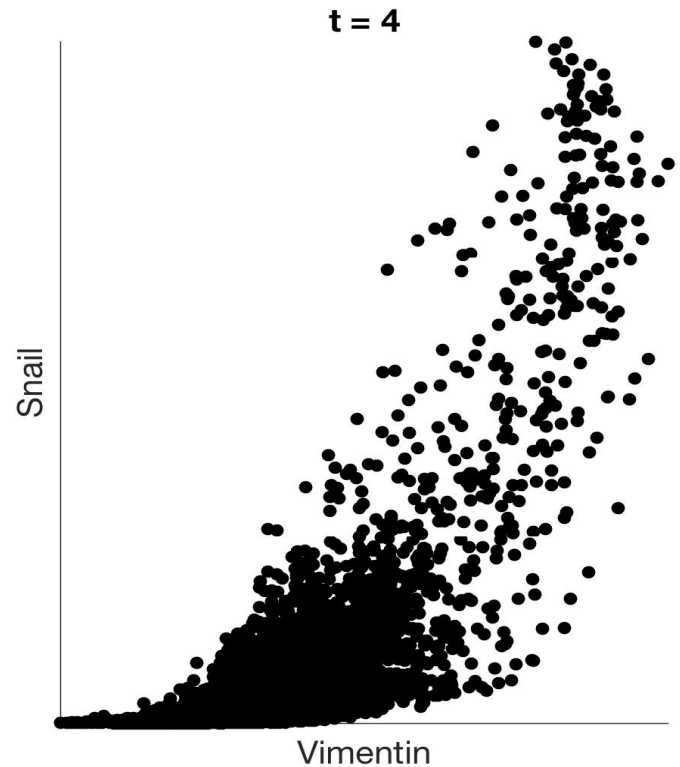

t = 1
E-Cadherin
Vimentin
t = 1
Snail
Vimentin

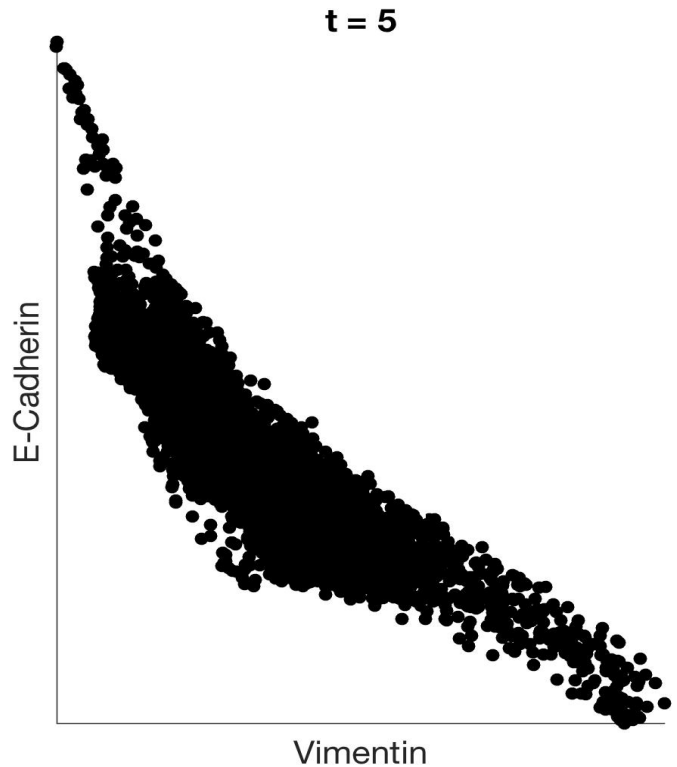

t = 2
E-Cadherin
Vimentin
t = 2
Snail
Vimentin

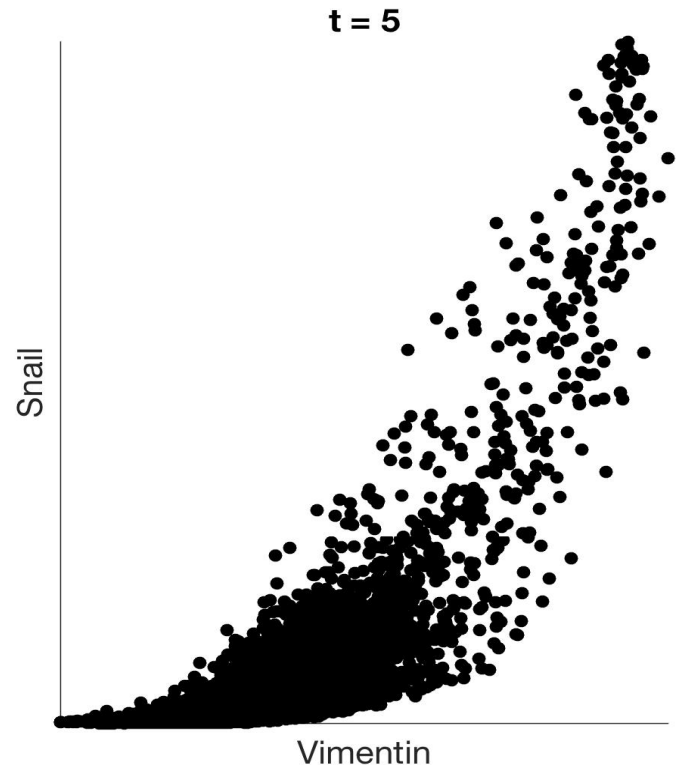

t = 3
E-Cadherin
Vimentin

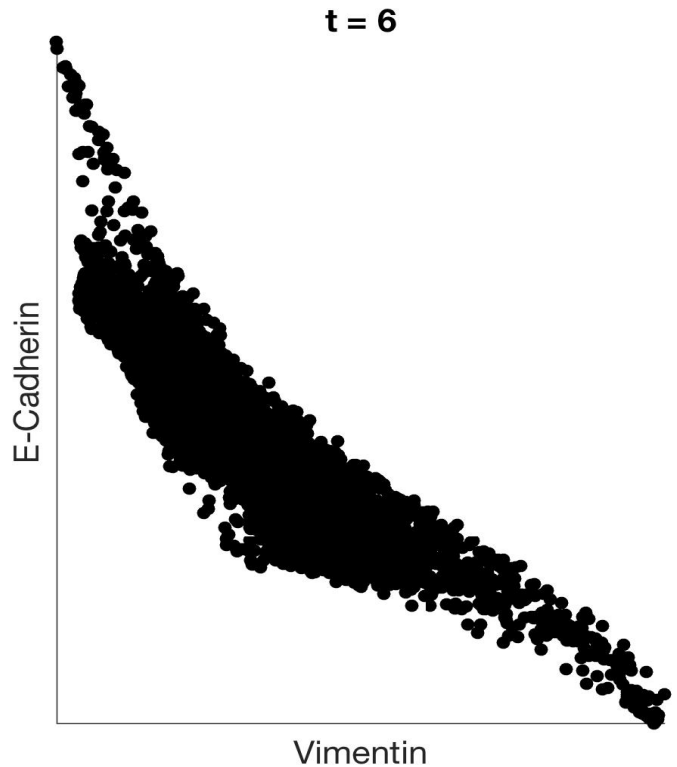

t = 3
Snail
Vimentin

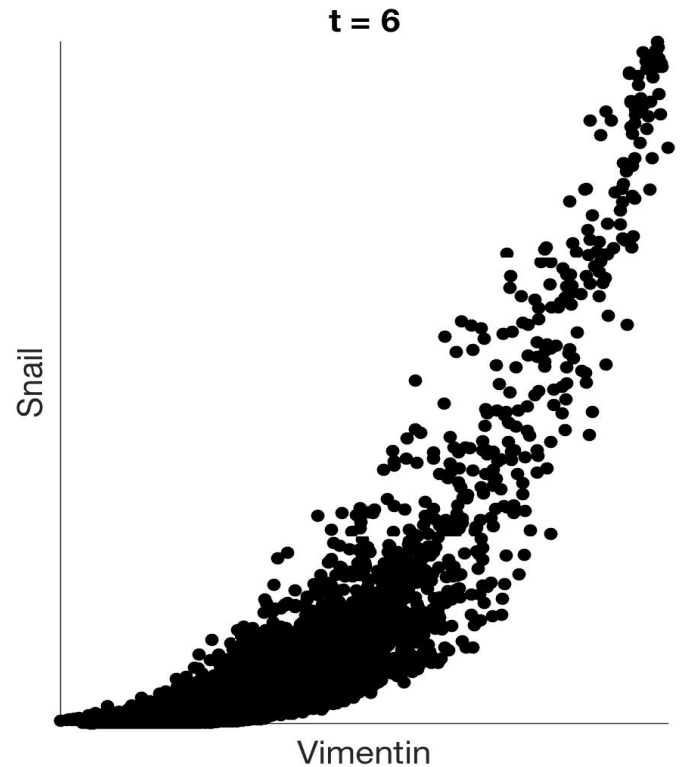

t = 4
E-Cadherin
Vimentin


t = 4
Snail
Vimentin


t = 5
E-Cadherin
Vimentin


t = 5
Snail
Vimentin


t = 6
E-Cadherin
Vimentin

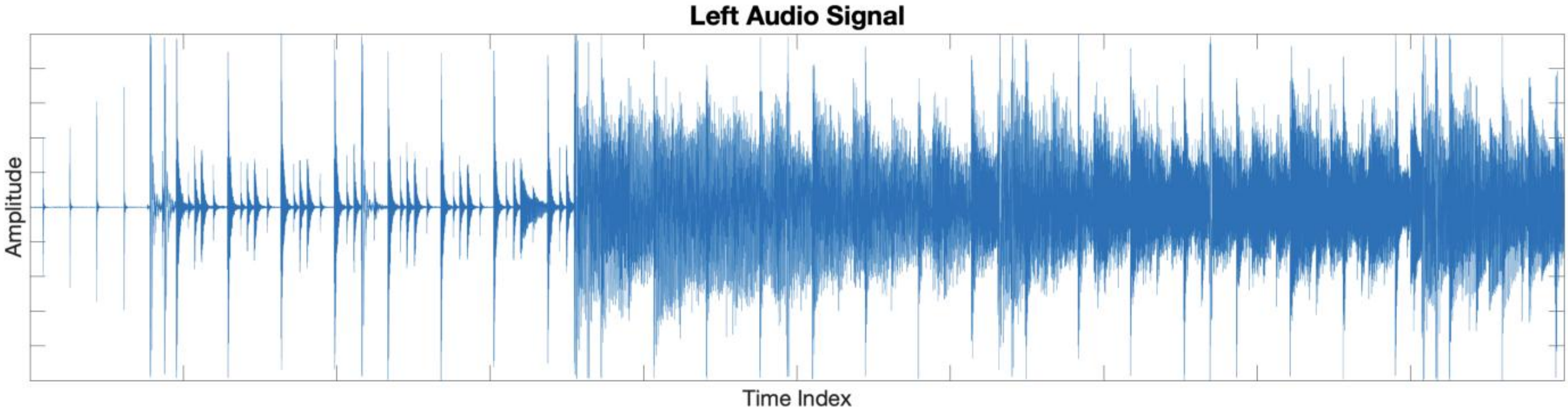

t = 6
Snail
Vimentin

Exercise

- Take the swiss roll and EB datasets compute a Laplacian
- Visualize the eigenvectors
- Low pass filter a feature and visualize it by coloring a PHATE or tSNE plot

Fourier and Wavelet Transforms

Time series signals



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

Fourier Transform of a Signal

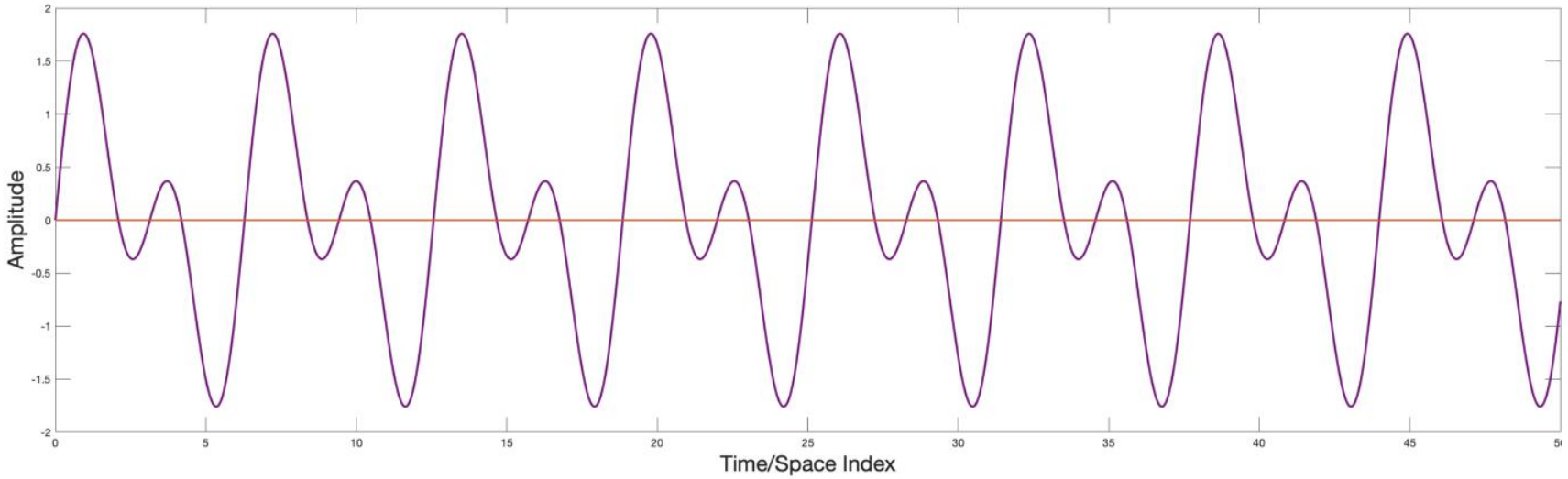
$$f(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx$$

Recall Euler's formula:

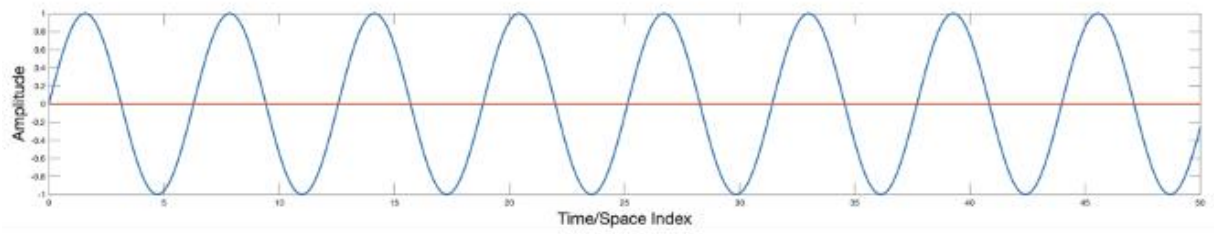
$e^{-i\vartheta x} = \cos(\vartheta x) + i\sin(\vartheta x)$

decompose signals in terms of sine and cosine

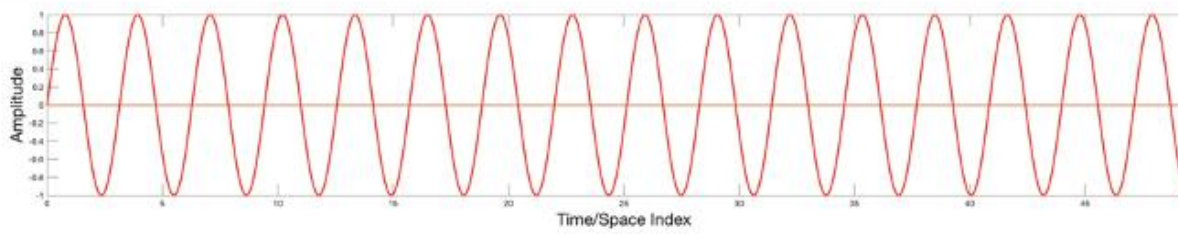
Decomposes signal into frequences



signal = lower freq signals + higher freq signals



+



Discrete Fourier Transform

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

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

FT

Discretize

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

DFT

DFT is a matrix multiplication

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

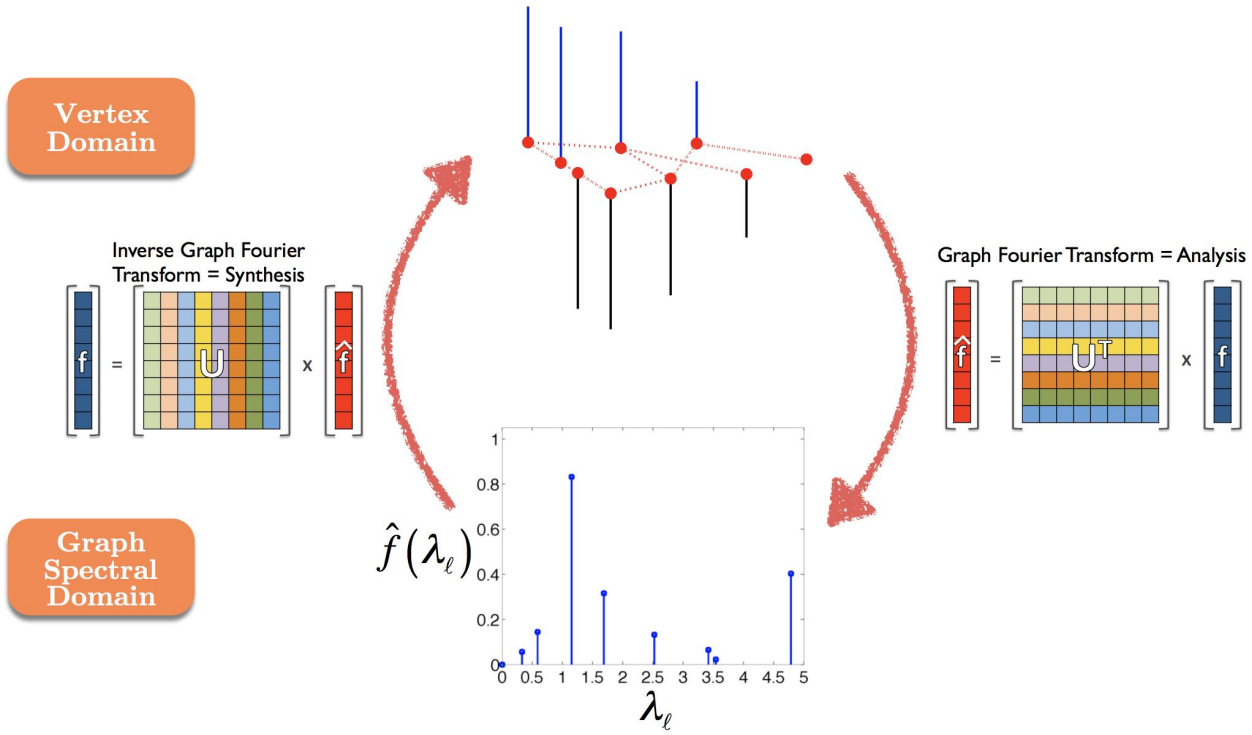
$R_{k,n}$ is the value of the kth waveform at time index n $R_{kn} = e^{\frac{-2\pi ikn}{N}}$

Fourier transforms have numerous uses

- Power spectral analysis
- Solutions of differential equations (heat equation)
- 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

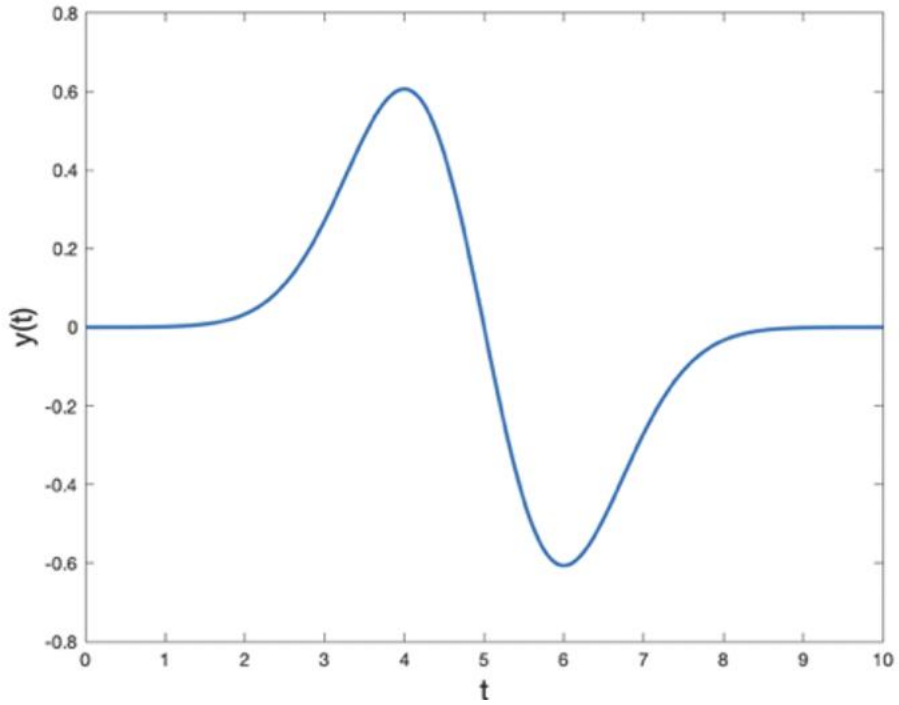


Global vs Local Frequency Domain Analysis

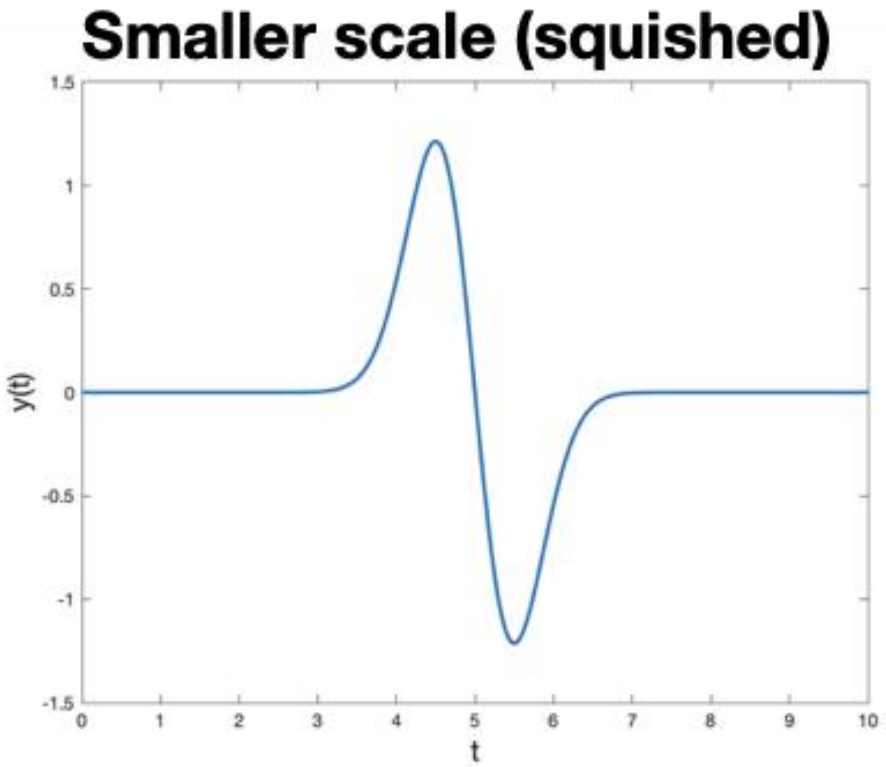
- Fourier transforms capture global frequency information, frequencies that persist over the entire length of the signal
- What about small bursts of frequency changes?
- Uneven frequency distributions over time?
- These might benefit from a more localized frequency domain analysis

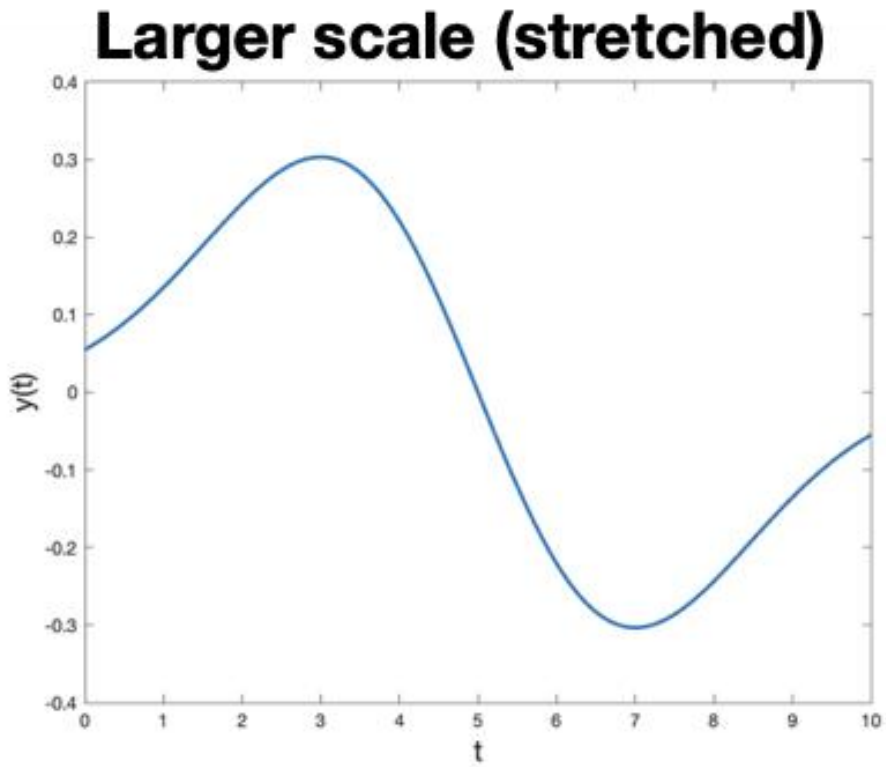
Wavelets

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



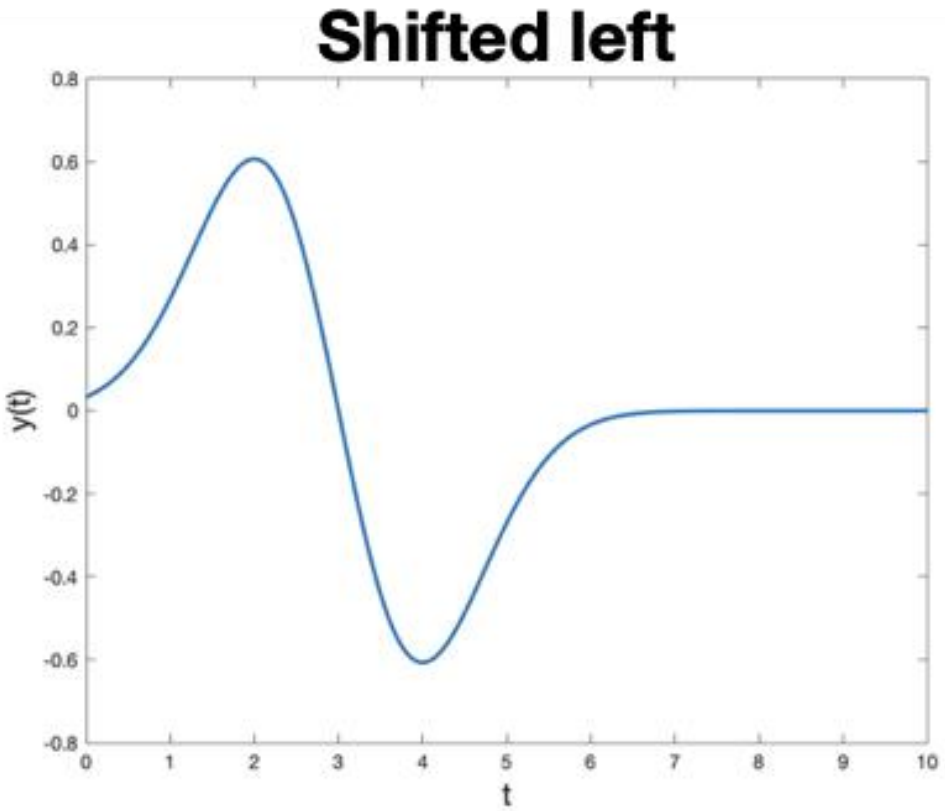
Scaling wavelets

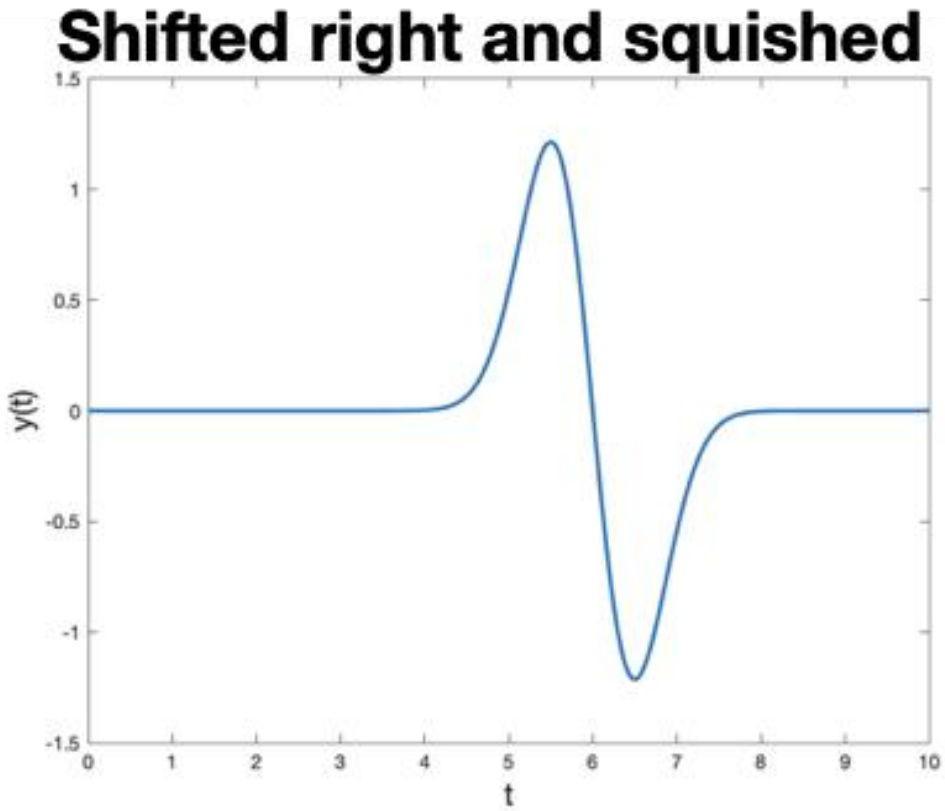




a

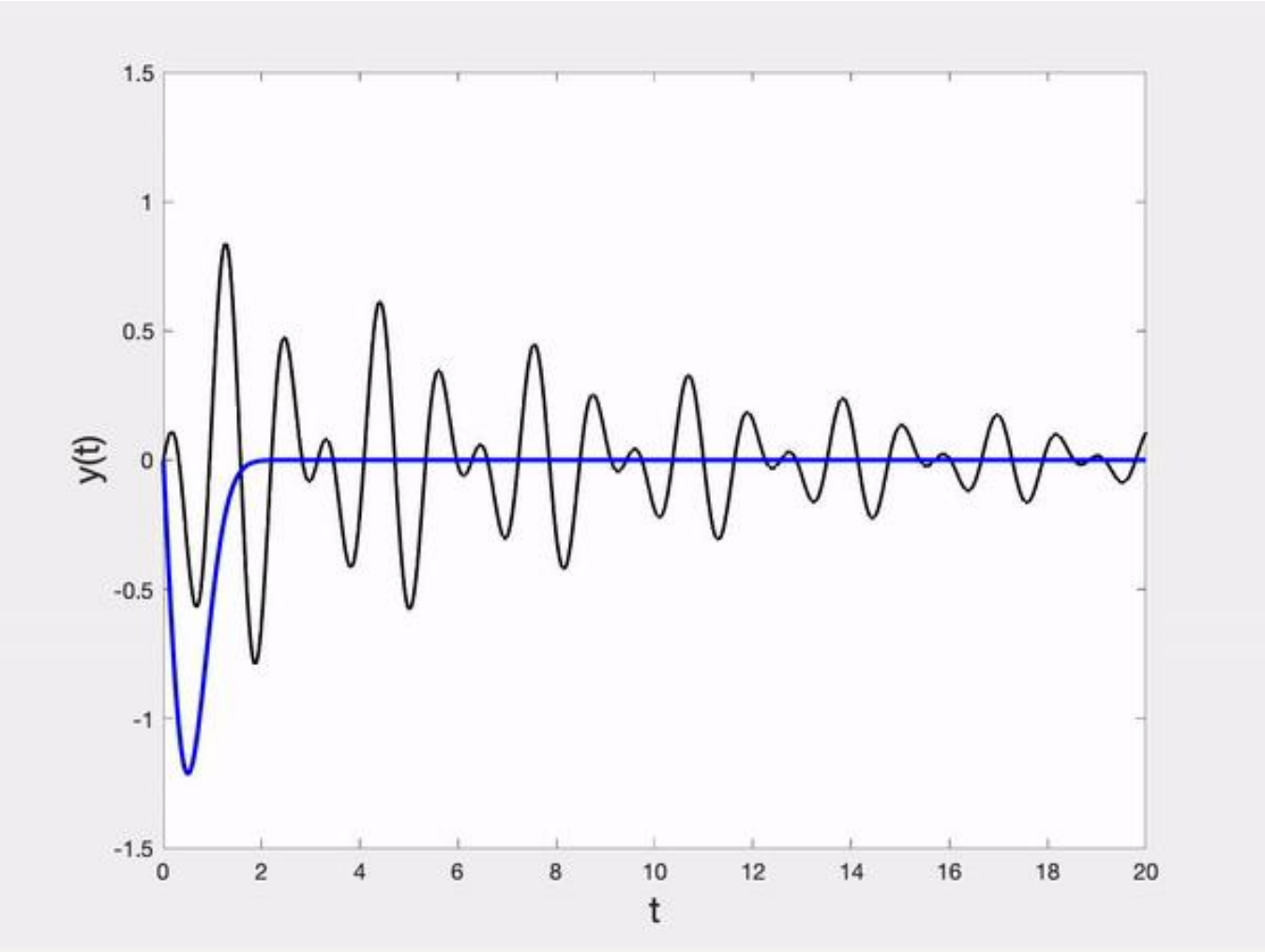
Shifting wavelets in time





b

Discrete Wavelet Transform



Wavelet Transform

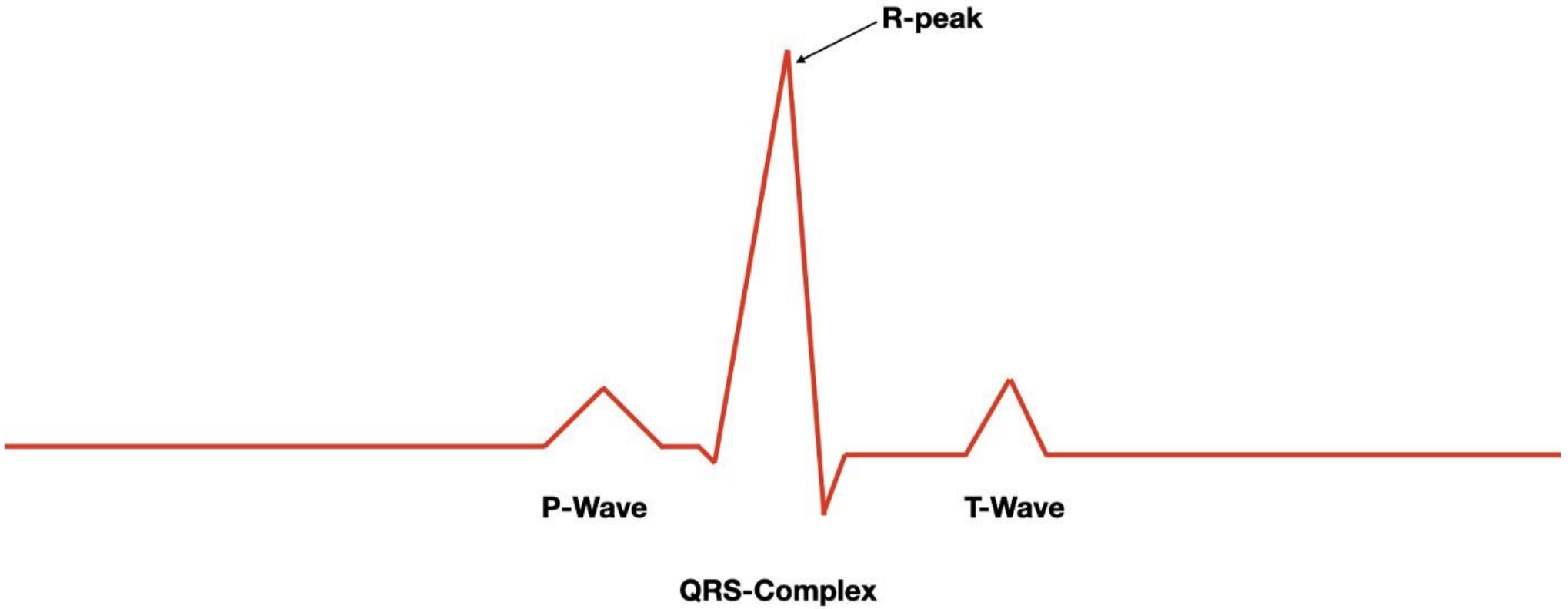
Continuous Wavelet Transform (CWT)

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

Discrete Wavelet Transform (DWT)

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

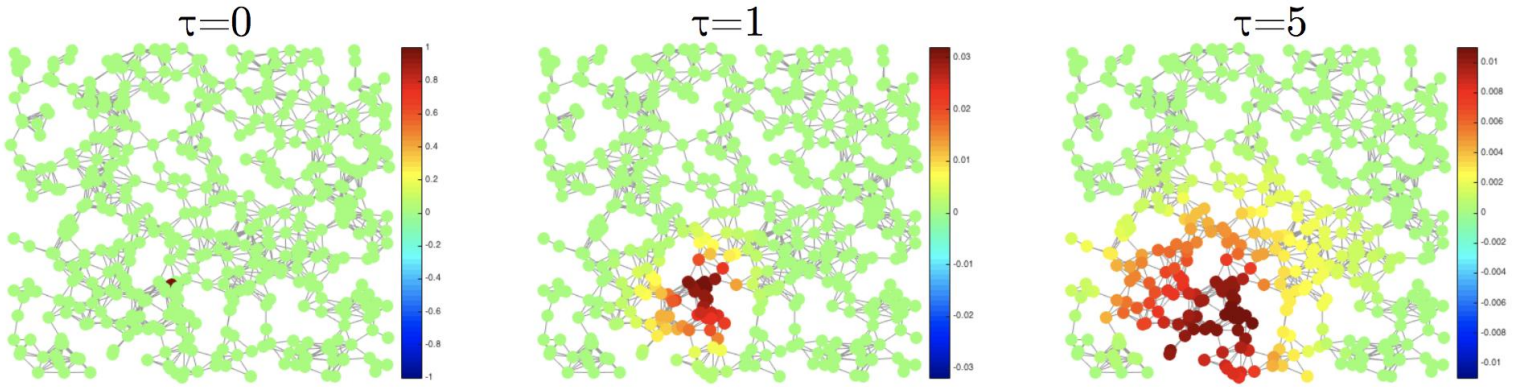
Sample Application peak detection in EEG



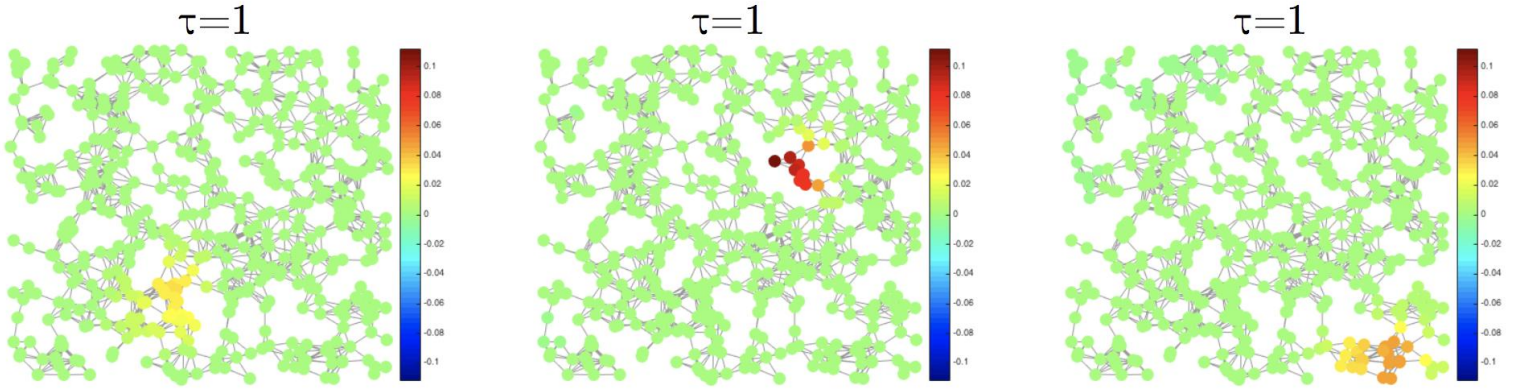
Diffusions can be used to create wavelets

- Diffusion Atoms

- 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:

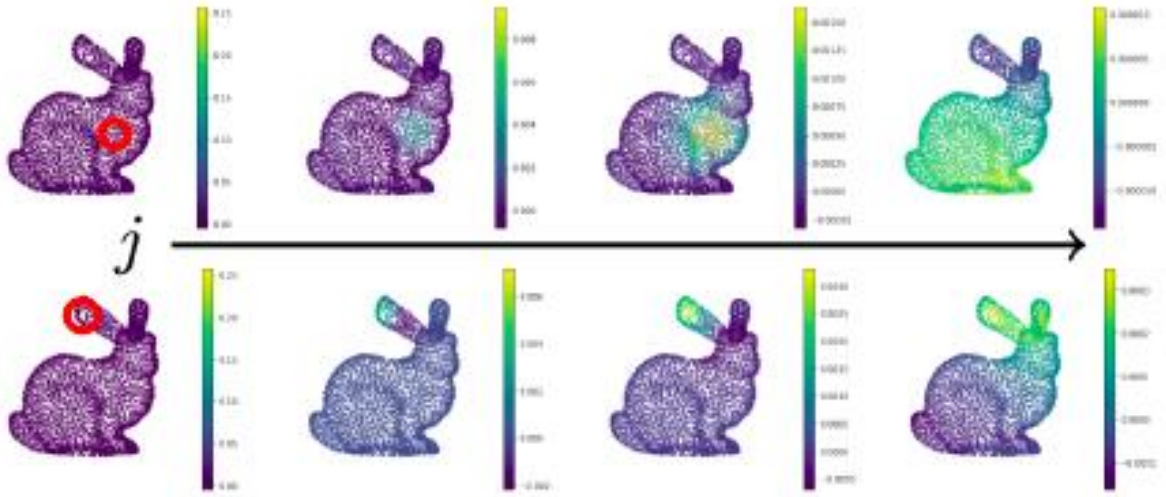


Coifman and Lafon, Diffusion maps, ACHA, 2006

Diffusion wavelets: Differences between lazy Random walks

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

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



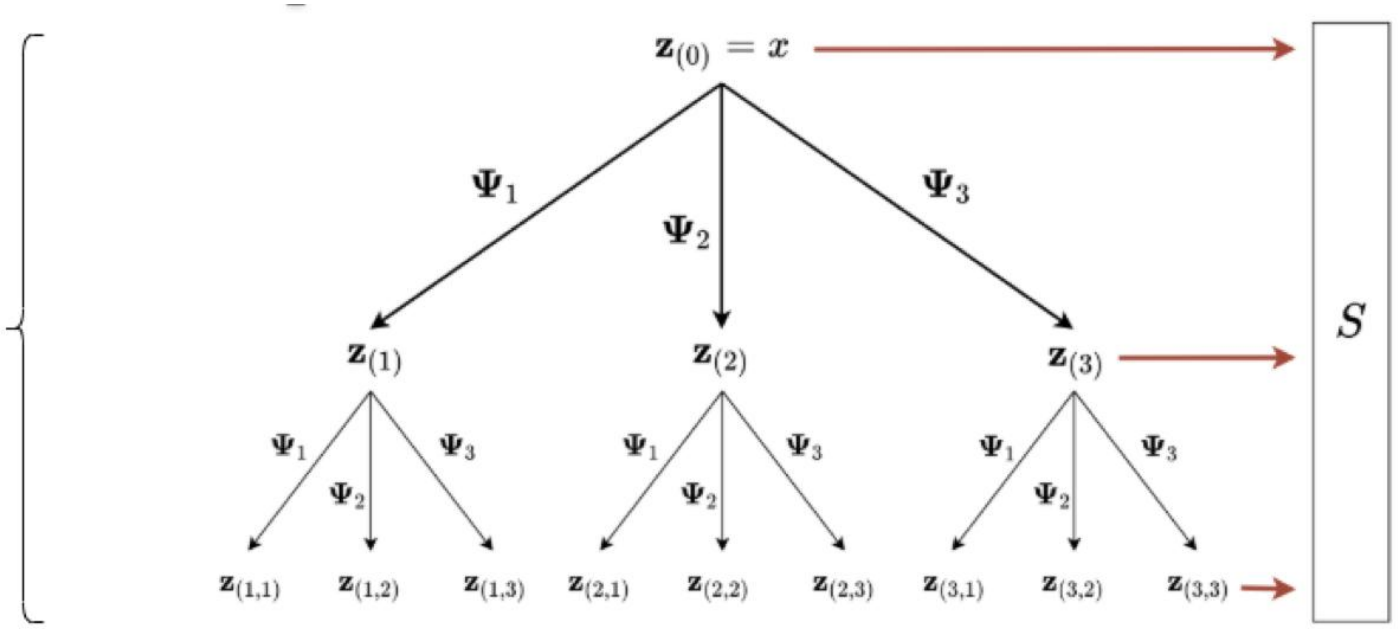
Coifman and Maggioni

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	

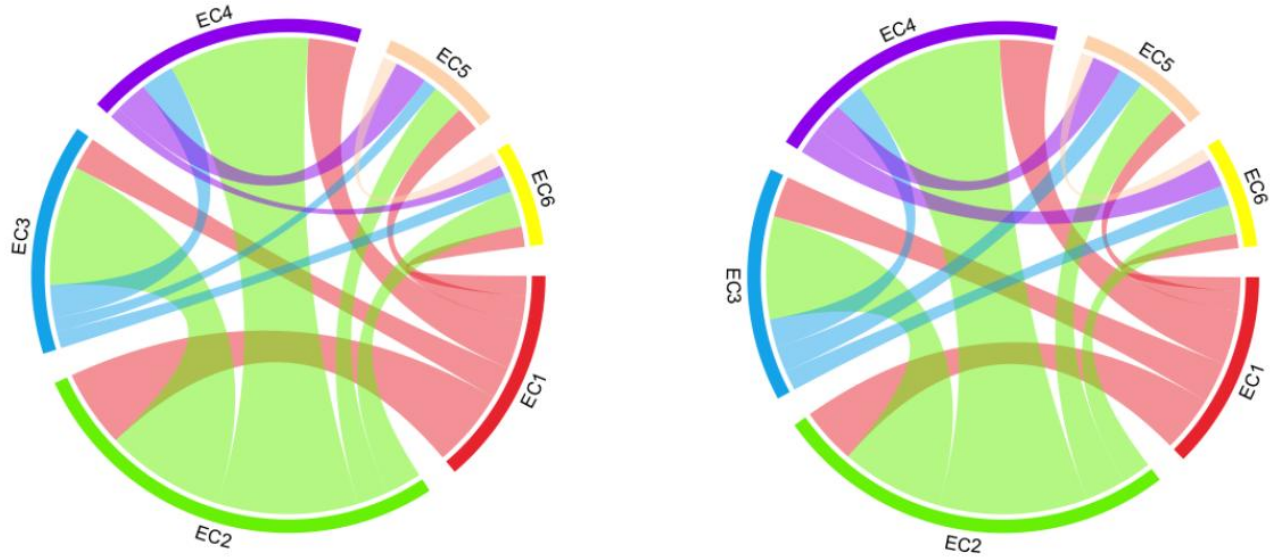
Scattering transform

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



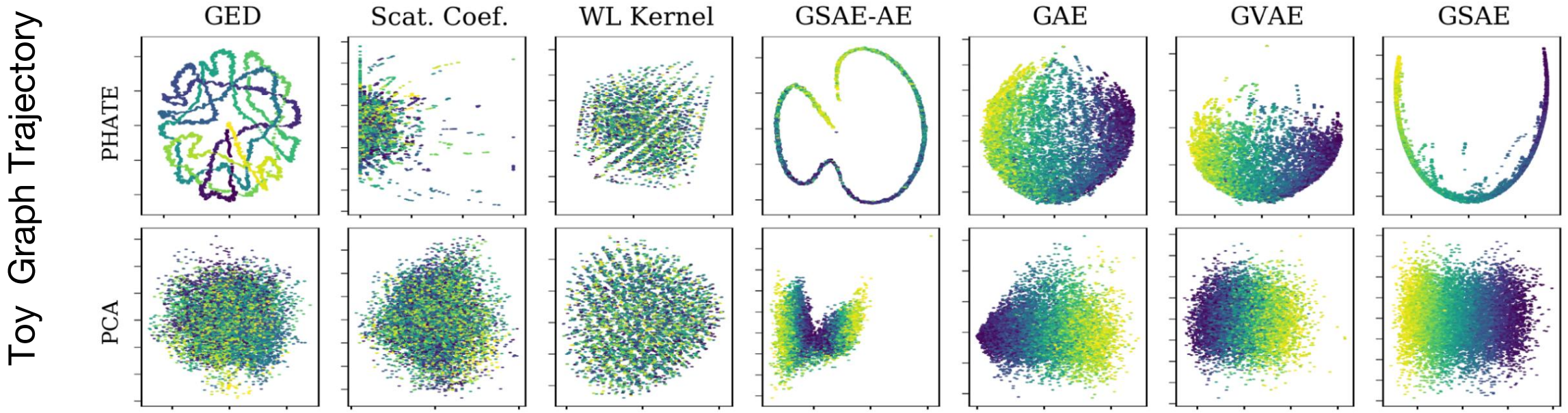
Gao et al. 2019

Embedding Analysis



(a) Observed (b) Inferred

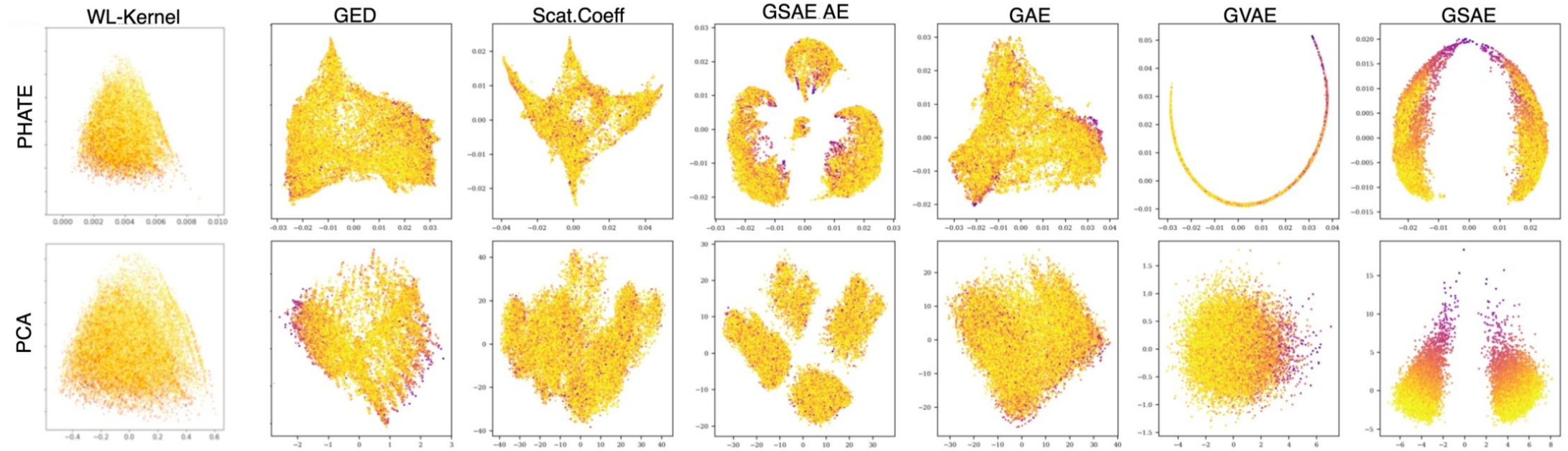
Embeddings



Castro et al. 2019

Bistable RNA Structure

SEQ3



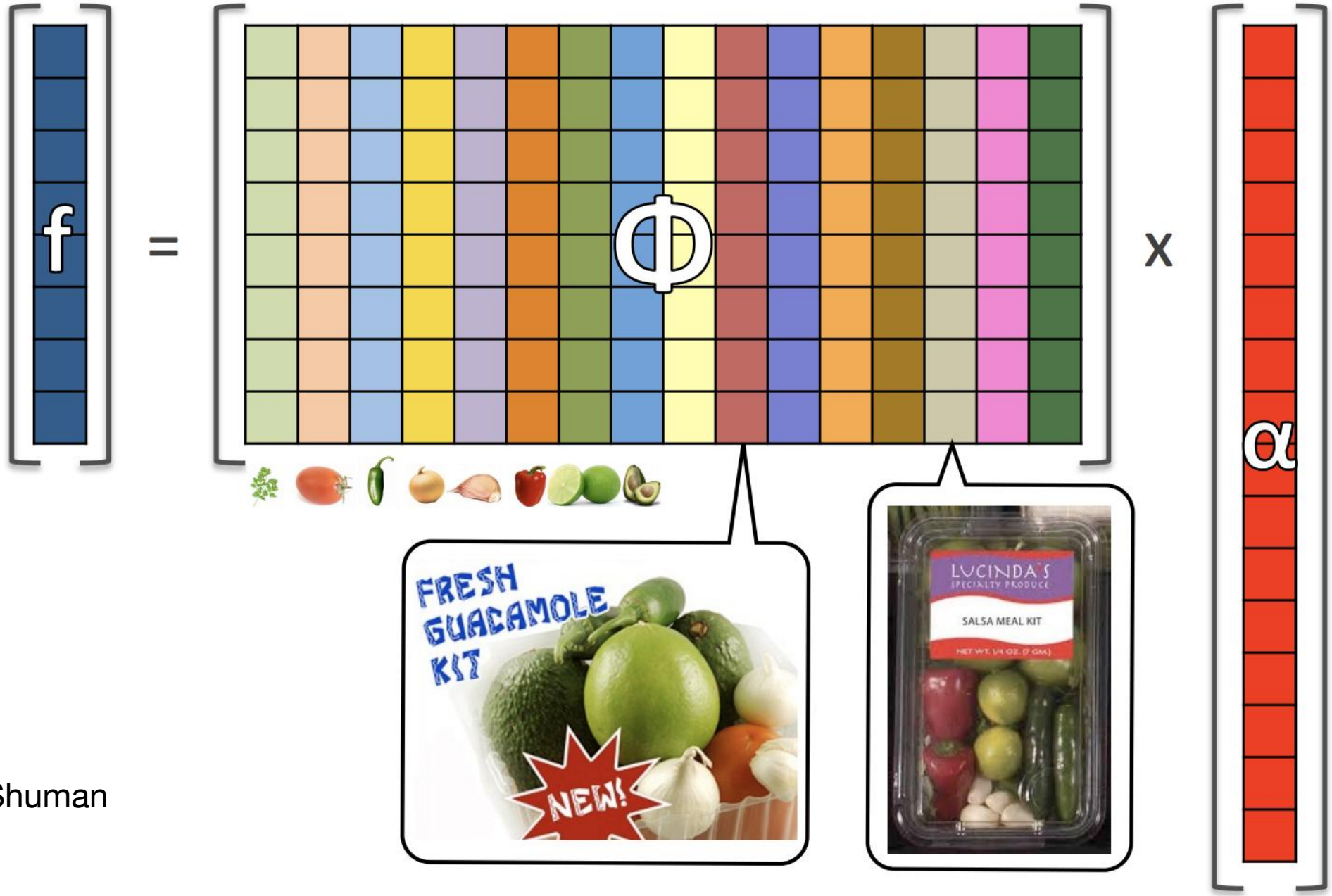
Exercise

- Generate a library of diffusion wavelets starting with a dirac signal
- Look at different scales

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

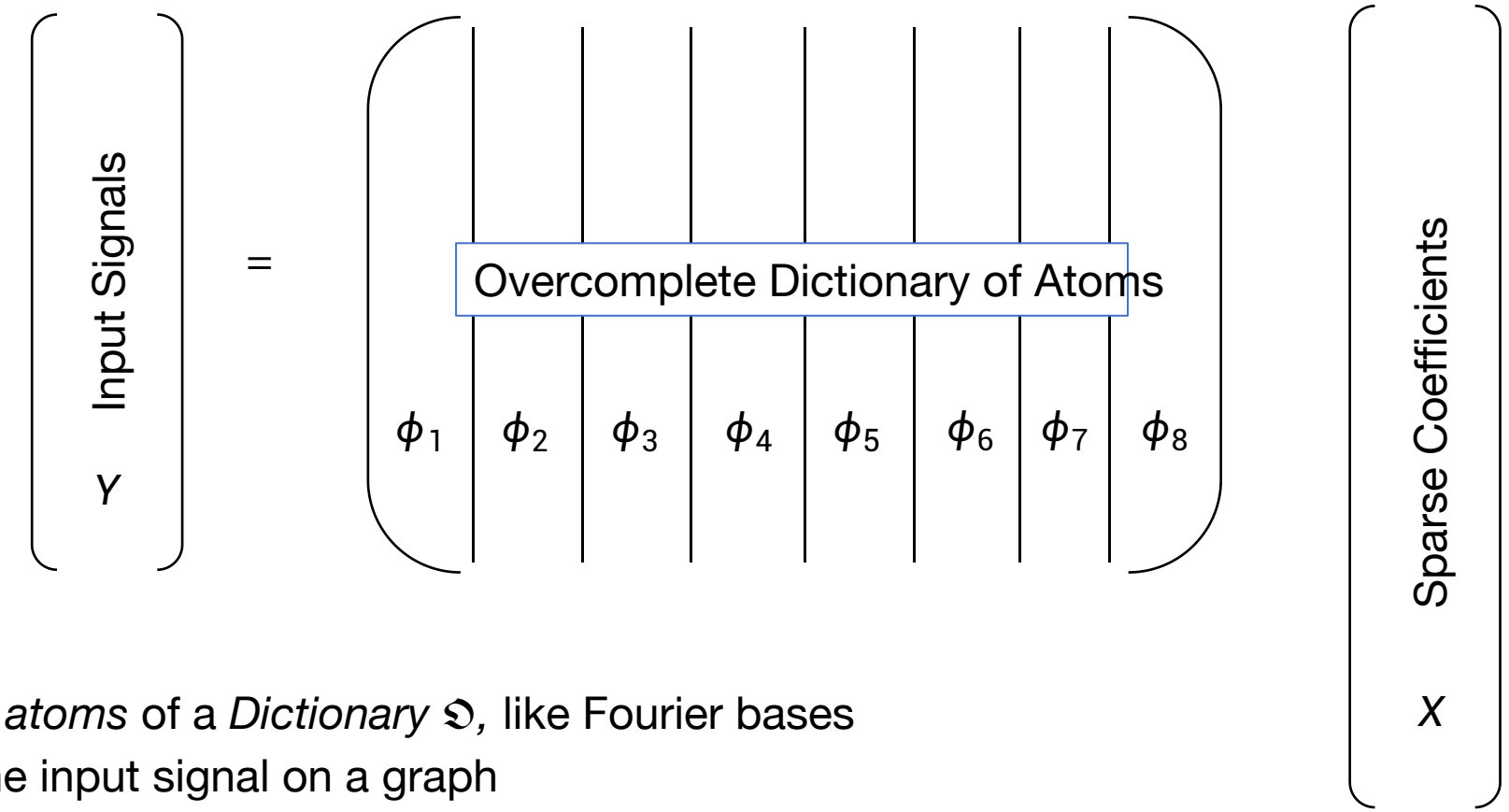
Sparse Coding and Dictionary Learning on Graphs

variants of sparse coding: smooth coding, dict learning



Courtesy Shuman

What is sparse coding?



ϕ_1 are *atoms* of a *Dictionary* $\mathfrak{D}$, like Fourier bases

Y is the input signal on a graph

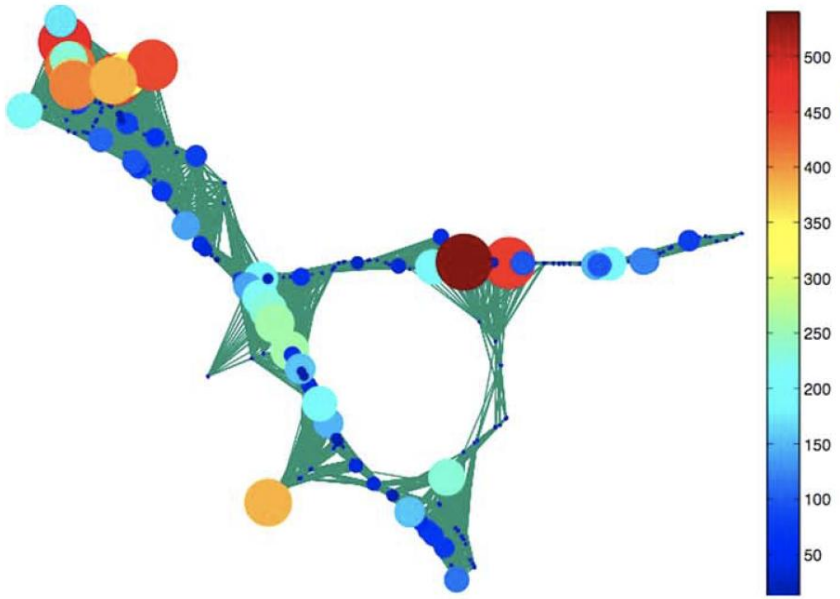
X is the re-representation of Y in terms of $\mathfrak{D}$, a set of sparse coefficients

overcomplete*: Dictionary* $\mathfrak{D}$ does not have to be orthonormal or be eigen basis

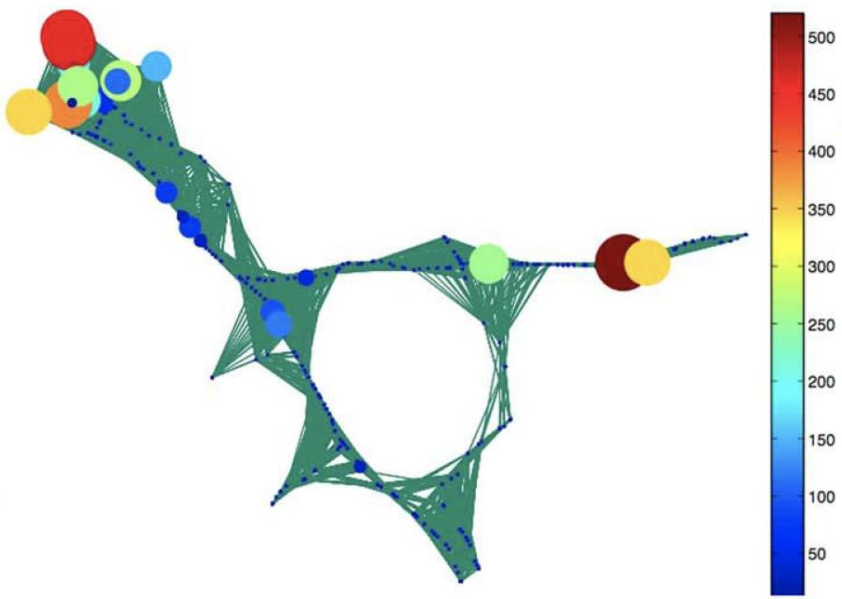
The Dictionary

- The dictionary contains *chosen* or *learned* atoms
- Examples of atoms for graph signals
  - Eigenvectors of graph Laplacian
  - Graph wavelets
  - Subdictionaries of polynomials of the graph Laplacian

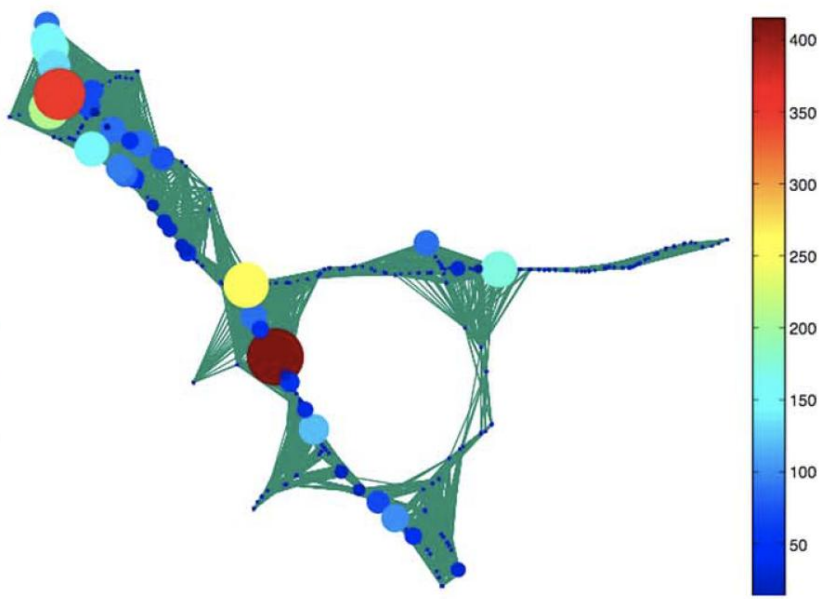
Examples of Atoms



(a)



(b)



(c)

Sparse Coding Optimization

minimize reconstruction error: recreating signal Y as a linear combination of atoms in dict D with sparse coefficient X

$$argmin_X ||Y - \mathfrak{D}X||^2 \quad \text{(convex)}$$

Subject to l_0 norm $||x_i||_0 \leq T$ (non-convex)

This is a non-convex optimization. Often solved by a method called matching pursuit.

1. Find the atom that loads the highest to the signal
2. Compute a "residual" by using an orthogonal projection to the atoms selected so far

Repeats step 1

Proceeds until convergence

Dictionary Learning

K-SVD: one of best-known dict learning algorithms
learn a dict D and sparse coding X simultaneously

Aharon et al. 2006

Task: Find the best dictionary to represent the data samples $\{\mathbf{y}_i\}_{i=1}^N$ as sparse compositions, by solving

$$\min_{\mathbf{D},\mathbf{X}} \left\{ \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \right\} \quad \text{subject to} \quad \forall i,\ \|\mathbf{x}_i\|_0 \leq T_0.$$

Initialization : Set the dictionary matrix $\mathbf{D}^{(0)} \in \mathbb{R}^{n \times K}$ with ℓ^2 normalized columns. Set $J = 1$.
Repeat until convergence (stopping rule):

- *Sparse Coding Stage*: Use any pursuit algorithm to compute the representation vectors $\mathbf{x}_i$ for each example $\mathbf{y}_i$, by approximating the solution of

$$i = 1, 2, \ldots,\ N, \quad \min_{\mathbf{x}_i} \left\{ \|\mathbf{y}_i - \mathbf{D}\mathbf{x}_i\|_2^2 \right\} \quad \text{subject to} \quad \|\mathbf{x}_i\|_0 \leq T_0.$$

- *Codebook Update Stage*: For each column $k = 1, 2, \ldots, K$ in $\mathbf{D}^{(J-1)}$, update it by
  - Define the group of examples that use this atom, $\omega_k = \{i \mid 1 \leq i \leq N,\ \mathbf{x}_T^k(i) \neq 0\}$.
  - Compute the overall representation error matrix, $\mathbf{E}_k$, by

$$\mathbf{E}_k = \mathbf{Y} - \sum_{j \neq k} \mathbf{d}_j \mathbf{x}_T^j.$$

  - Restrict $\mathbf{E}_k$ by choosing only the columns corresponding to ω_k, and obtain $\mathbf{E}_k^R$.
  - Apply SVD decomposition $\mathbf{E}_k^R = \mathbf{U}\boldsymbol{\Delta}\mathbf{V}^T$. Choose the updated dictionary column $\tilde{\mathbf{d}}_k$ to be the first column of $\mathbf{U}$. Update the coefficient vector $\mathbf{x}_R^k$ to be the first column of $\mathbf{V}$ multiplied by $\boldsymbol{\Delta}(1,1)$.
- Set $J = J + 1$.

What happens when an atom is removed?

$$
\begin{aligned}
\|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2 &= \left\|\mathbf{Y}-\sum_{j=1}^{K}\mathbf{d}_j\mathbf{x}_T^j\right\|_F^2 \\
&= \left\|\left(\mathbf{Y}-\sum_{j\neq k}\mathbf{d}_j\mathbf{x}_T^j\right)-\mathbf{d}_k\mathbf{x}_T^k\right\|_F^2 \\
&= \left\|\mathbf{E}_k-\mathbf{d}_k\mathbf{x}_T^k\right\|_F^2 .
\end{aligned}
$$

Graph Denoising Coding Optimization

$$argmin_X ||Y - \mathfrak{D}X||^2 + \gamma (DX)^T L(DX)$$

This is convex, can be solved faster by SGD

$||Y - \mathfrak{D}X||^2$: reconstruction error

$\gamma (DX)^T L(DX)$: constrain signals to be smooth instead of be sparse, quadratic form

γ: regularization parameter, determines how important constraint is in loss

Polynomial Dictionary

$$\underset{\alpha \in \mathbb{R}^{(K+1)S}}{\operatorname{argmin}} \quad \left\{ \|Y - \mathcal{D}X\|_F^2 + \mu \|\alpha\|_2^2 \right\}$$

$$\text{subject to} \quad \mathcal{D}_s = \sum_{k=0}^{K} \alpha_{sk} \mathcal{L}^k, \quad \forall s \in \{1, 2, \ldots, S\}$$

$$0 \preceq \mathcal{D}_s \preceq cI, \quad \forall s \in \{1, 2, \ldots, S\}$$

$$(c - \epsilon_1) I \preceq \sum_{s=1}^{S} \mathcal{D}_s \preceq (c + \epsilon_2) I.$$

Thanou et al. 2014

learn a dict D as concatenation of subdicts
[D1, …DS]

additional constraint to ensure subdicts have nice forms

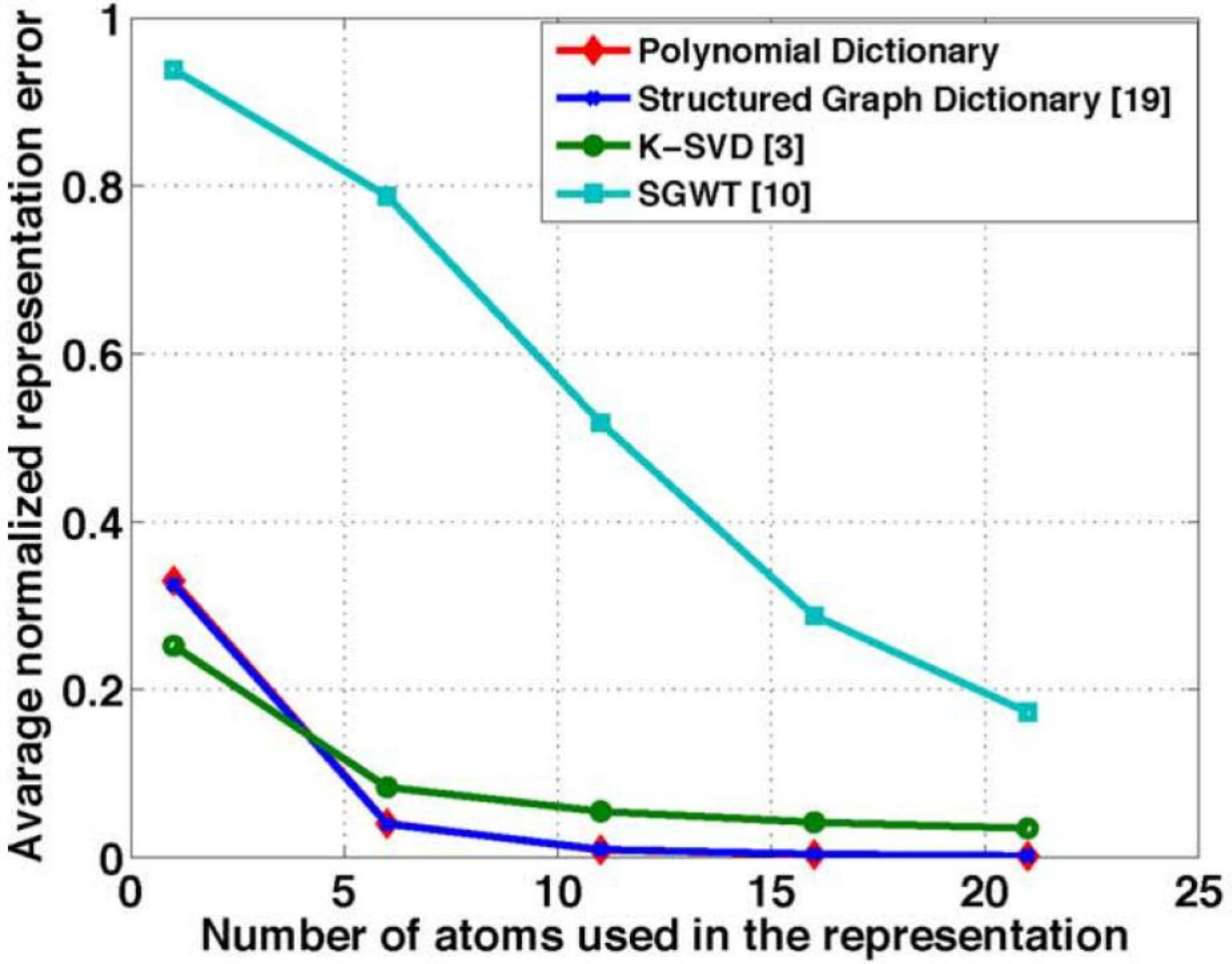
Compose a dictionary of subdictionari $[\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_S]$ whose atoms are Laplacian polynomials

each subdict D_s is polynomial of Graph Laplacian

$D_s = \hat{g_s}(L) = \theta \sum_0^K \alpha_{sk} \Lambda^K \theta^T = \sum_0^K \alpha_{sk} L^K$

α_{sk}: coefficient. Λ: eigenvalue, θ: conjugated eigenvector

Fewer atoms, less error in approximation



Applications of Sparse Coding

- Compressed representations of signals on graph
- Ease of comparing and interpreting differences of signals on graphs
- Ex. Flow of traffic on city street graph
- Ex2. Flow of brain state activity on neuronal graph