Graph Signal Processing and Spectral GNNs

Michael Perlmutter

Department of Mathematics
Boise State University

Overview

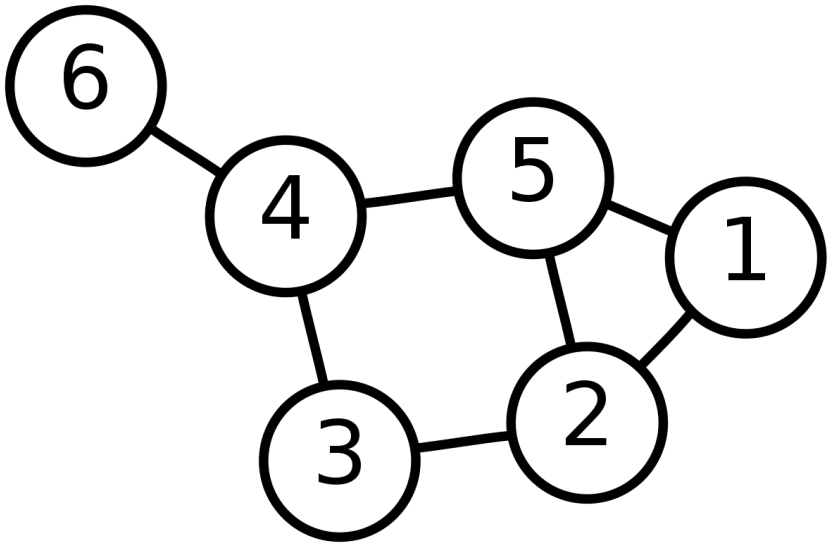
- Preliminaries
- Spectral Clustering
- Signal Processing
- Convolutions in Deep Learning
- Spectral GNNs
- Generalized Graphs

Preliminaries

Graphs

Defintion

A graph $G = (V, E)$ is a mathematical object consisting of vertices and edges. In applied settings, people sometimes say network instead of graph, node instead of vertex, and link instead of edge.



Geometric Deep Learning

- Convolutional Neural Networks utilize the structure of images (pixels are arranged in a grid).
- Recurrent Neural Networks utilize the sequential nature of time-series data
- How can we achieve similar successes for data with non-Euclidean structure such as graphs and manifolds?

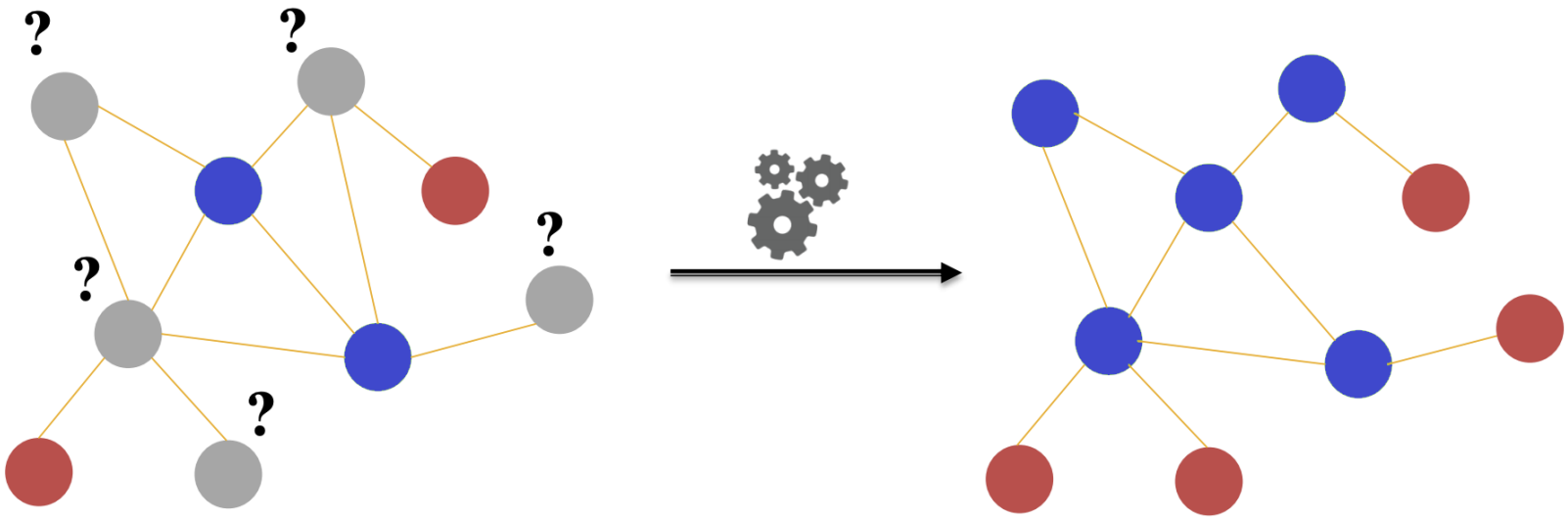
Node-Level Tasks

Node Classification

Is a member of a social network a Republican or a Democrat?

Node Clustering

Divide customers into meaningful subgroups.



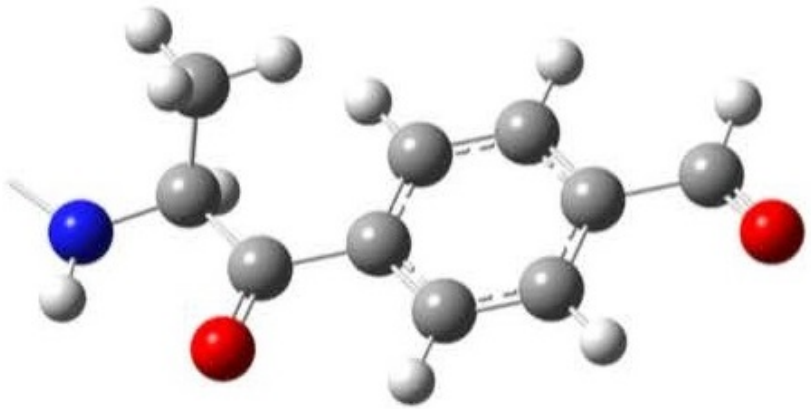
Graph-Level Tasks

Graph Classification

Is a molecule related to cancer?

Graph Regression

What is the formation energy of a molecule?



Could also have many different functions defined on the same graph, similar to image classification.

Notation

- $G = (V, E)$ is a graph, $V = \{v_1, \ldots, v_N\}$, $E \subseteq V \times V$.
- Adjacency matrix $\mathbf{A}$

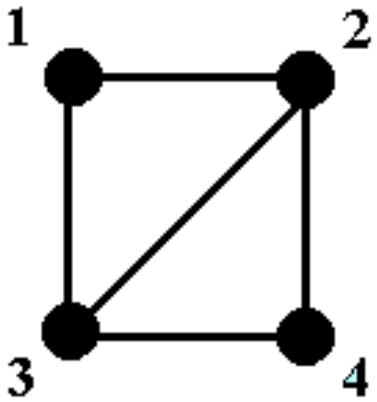
$$\mathbf{A}(j,k) = \begin{cases} 1 & \text{if } (v_j, v_k) \in E \\ 0 & \text{otherwise} \end{cases}.$$

- Degree vector and matrix

$$\mathbf{D} = \text{diag}(\mathbf{d}), \quad \mathbf{d}(j) = \text{degree of vertex } j.$$

- Note: $\mathbf{d}(j) = \sum_{k=1}^{N} A(j,k)1$, so $\mathbf{d} = \mathbf{A}\mathbb{1}$.
- This later definition generalizes to weighted graphs.
- Functions $\mathbf{x} : V \to \mathbb{R}$ are often called signals and can be thought of as N-dimensional vectors, $\mathbf{x}(j) = \mathbf{x}(v_j)$.

Example



$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}, \quad \mathbf{d} = (2, 3, 3, 2), \quad \mathbf{D} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}.$$

GNNs

Setup

- Input: $N \times N$ adjacency matrix $\mathbf{A}$ and $N \times F$ matrix $\mathbf{X}$.
- Output: Predictions about each node or the entire graph.

Two Perspectives

- Feature Perspective:
  - Each *row* of $\mathbf{X}$ contains a sequence of F attributes assigned to a given node.
- Signal Perspective:
  - Each *column*, $\mathbf{x}_i$ is a function (signal) defined on V.

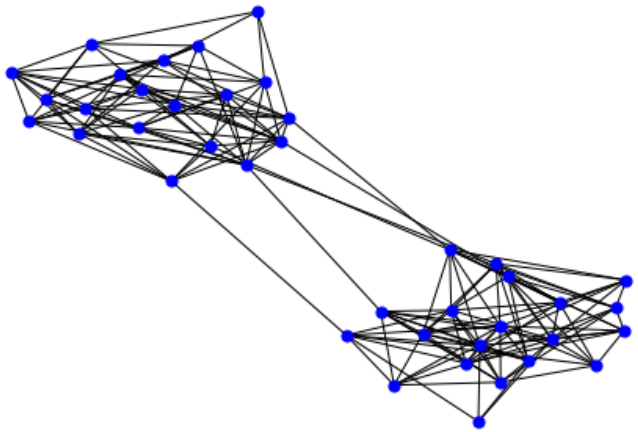
This talk is focused on the signal perspective.

Spectral Clustering

The Min-Cut Problem

Goal:

Partition the vertices into two roughly equally sized parts while cutting as few edges as possible.



More formally:

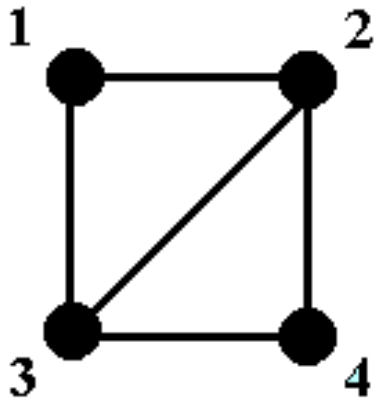
For $S \subset V$, let $e(S, S^C)$ be the number of edges from S to S^C, we want to minimize, over all sets $S \subseteq V$,

$$R(S) = \frac{e(S, S^C)}{|S||S^C|}.$$

Impossible to solve exactly in a reasonable amount of time.

Graph Laplacian: $\mathbf{L} = \mathbf{D} - \mathbf{A}$

Example:



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

Note: All rows sum to zero.

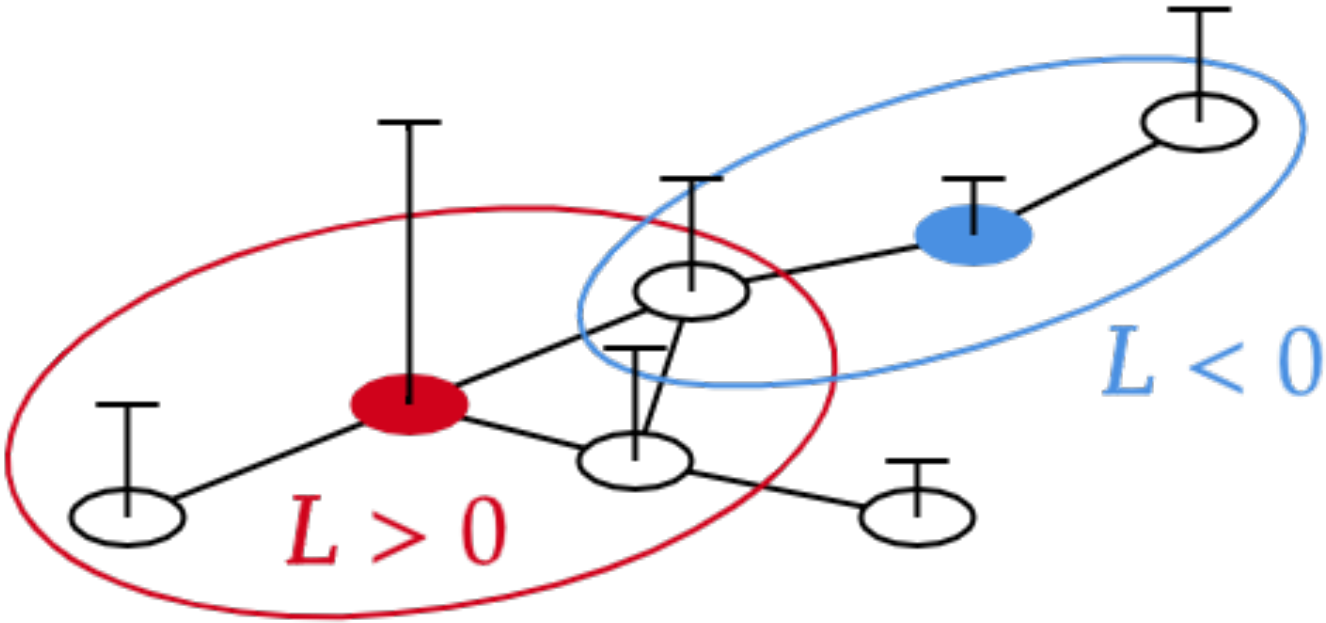
Eigenvectors of the Graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$

Facts:

- $\mathbf{L}$ is positive semidefinite.
- Orthonormal basis of eigenvectors
- $\mathbf{L}\mathbf{u}_j = \lambda_j \mathbf{u}_j$, $\lambda_j \leq \lambda_{j+1}$
- $\lambda_1 = 0, \mathbf{u}_1 = \frac{1}{\sqrt{N}}\mathbb{1}$
- $$\mathbf{x}^T \mathbf{L}\mathbf{x} = \frac{1}{2} \sum_{(j,k)\in E} |\mathbf{x}(j) - \mathbf{x}(k)|^2$$

If we think of a $\mathbf{x}$ as a function on V, $\mathbf{x}(j) = \mathbf{x}(v_j)$, the last fact means that $\mathbf{x}^T \mathbf{L}\mathbf{x}$ measures the smoothness of $\mathbf{x}$, where a function is smooth if it has similar values at neighboring vertices.

The Graph Laplacian Illustrated



Application to min-cut

Choice of vector

For $S \subseteq V$, let $r_S = \sqrt{|S^C|/|S|}$, and define $\mathbf{x}_S$ by

$$\mathbf{x}_S(j) = \begin{cases} r_S & \text{if } v_j \in \mathcal{S} \\ -1/r_S & \text{otherwise.} \end{cases}$$

Claim:

$\mathbf{x}_S^T \mathbf{L} \mathbf{x}_S$ is a constant multiple of $R(S)$.

Notes:

$$\mathbf{x}_S^T \mathbf{L} \mathbf{x}_S = \frac{1}{2} \sum_{(j,k) \in E} |\mathbf{x}_S(j) - \mathbf{x}_S(k)|^2$$

Many of the terms in the sum are zero, and all non-zero terms are equal to $(r_S + \frac{1}{r_S})^2$.

Application to the min-cut (continued)

Non-zero values of $|\mathbf{x}_S(j) - \mathbf{x}_S(k)|^2$

$$\left(r_S + \frac{1}{r_S}\right)^2 = r_S^2 + \frac{1}{r_S^2} + 2 = \frac{|S^C|}{|S|} + \frac{|S|}{|S^C|} + 2$$
$$= \frac{|S^C|^2 + |S|^2 + 2|S||S^C|}{|S||S^C|} = \frac{(|S| + |S^C|)^2}{|S||S^C|} = \frac{N^2}{|S||S^C|}$$

Quadratic Form

$$\mathbf{x}_S^T \mathbf{L} \mathbf{x}_S = \frac{1}{2} \sum_{(j,k)\in E} |\mathbf{x}_S(j) - \mathbf{x}_S(k)|^2 = \frac{1}{2} e(S, S^C) \frac{N^2}{|S||S^C|} = \frac{N^2}{2} R(S)$$

Approximate Solution

Equivalent Formulation

- Minimizing $R(S)$ is equivalent to minimizing $\mathbf{x}_S^T L \mathbf{x}_S$
- We've reframed the problem, but so far we haven't actually made it any easier.

Heuristic

- A "good" cut should have $|S| \approx |S^C| \Rightarrow r = \sqrt{|S^C|/|S|} \approx 1$
- This implies that $\sum_j \mathbf{x}_S(j) \approx 0$.
- Note: $S = \{v_j : \mathbf{x}_S(j) \geq 0\}$

New Idea:

Solve the optimization problem

$$\min_{\{\mathbf{x}:\sum_j \mathbf{x}(j)=0\}} \mathbf{x}^T \mathbf{L} \mathbf{x}$$

set then $S = \{v_j : \mathbf{x}(j) \geq 0\}$.

Solving the new problem

Observations:

$$\sum_j \mathbf{x}(j) = 0 \Leftrightarrow \mathbf{x}^T \mathbb{1} = 0$$

$$\Leftrightarrow \mathbf{x}^T \mathbf{u}_1 = 0$$

$$\Leftrightarrow \mathbf{x} \perp \mathbf{u}_1$$

New Goal:

Solve the optimization problem

$$\min_{\mathbf{x} \perp \mathbf{u}_1} \mathbf{x}^T \mathbf{L} \mathbf{x}$$

then set then $S = \{v_j : \mathbf{x}(j) \geq 0\}$.

Solving the new problem

New Goal:

Solve the optimization problem

$$\min_{\mathbf{x} \perp \mathbf{u}_1} \mathbf{x}^T \mathbf{L} \mathbf{x}$$

then set then $S = \{v_j : \mathbf{x}(j) \geq 0\}$.

Courant-Fischer Theorem:

For any symmetric matrix $\mathbf{A}$, the minimizer of $\mathbf{x}^T \mathbf{A} \mathbf{x}$ constrained to be orthogonal to the first k eigenvectors is given by the $k+1$st eigenvector.

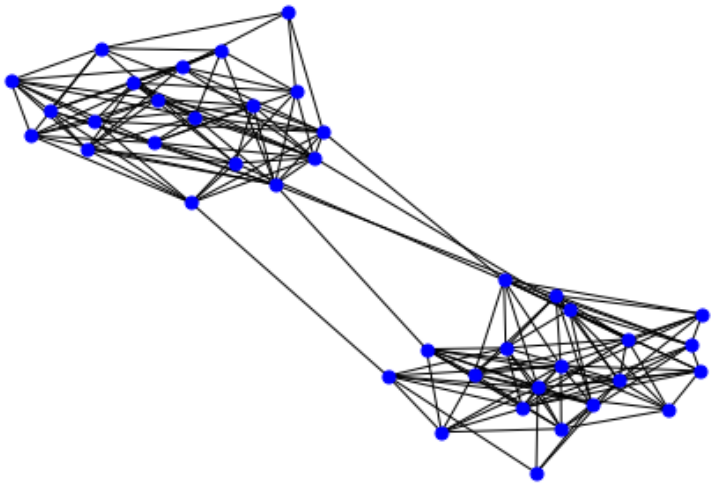
Therefore,

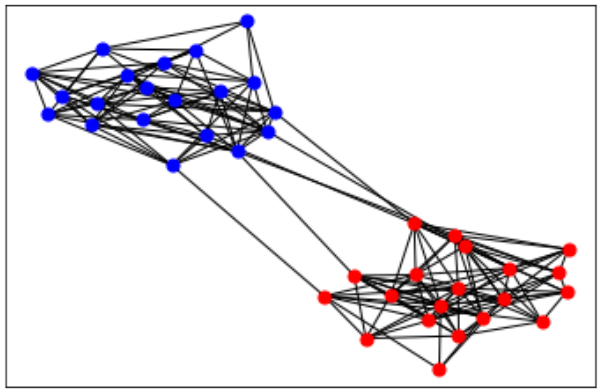
Our desired solution is given by $\mathbf{x} = \mathbf{u}_2$.

The Spectral Clustering Algorithm

- Spectral clustering for 2 clusters:
  1. Compute $\mathbf{d}(v_j)$ for $j = 1, \ldots, n$. Let $D = \text{diag}(\mathbf{d}(v_1), \ldots, \mathbf{d}(v_n))$.
  2. Compute Laplacian: $\mathbf{L} = \mathbf{D} - \mathbf{A}$.
  3. Compute **second** eigenpair $(\lambda_2, \mathbf{u}_2)$.
  4. Define S as
  $$S = \{v_j : \mathbf{u}_2(j) \geq 0\}$$
  5. **Output:** S, S^C.

Output of Spectral Clustering

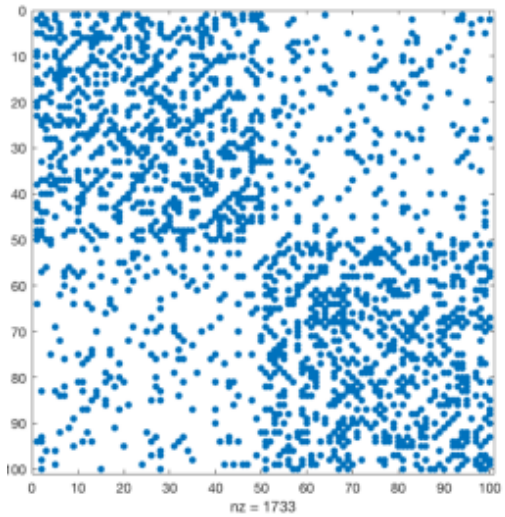




Stochastic Block Model

Vertex set V consists of two disjoint clusters C_1, C_2,

$$\mathbb{P}((j,k) \in E) = \begin{cases} p & \text{if } v_j \text{ and } v_k \text{ are in the same cluster} \\ q & \text{if } v_j \text{ and } v_k \text{ are in different clusters} \end{cases}$$



Theorem:

If p is much larger than q, spectral clustering will work well.

Signal Processing

The Graph Laplacian of the Cycle Graph

$$\mathbf{L} = \mathbf{D} - \mathbf{A} = \begin{pmatrix} 2 & -1 & 0 & \dots & 0 & -1 \\ -1 & 2 & -1 & 0 & \dots & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ -1 & 0 & \dots & 0 & -1 & 2 \end{pmatrix}.$$

Eigenvectors are Fourier modes:

$$\mathbf{u}_k(n) = e^{2\pi i n k/N} = \cos(2\pi nk/N) + i \sin(2\pi nk/N).$$

Fourier Modes at Different Frequencies

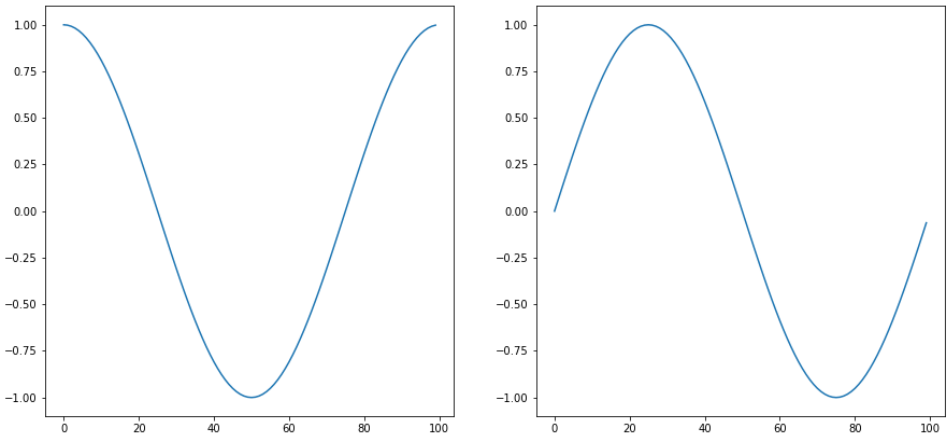


Figure: Real (left) and imaginary (right) parts, $k = 1$, $N = 100$

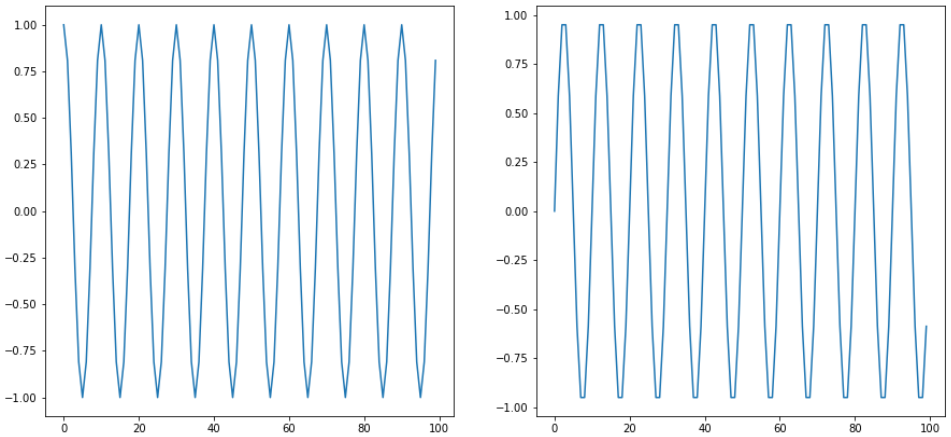


Figure: Real (left) and imaginary (right) parts, $k = 10$, $N = 100$

Audio Example

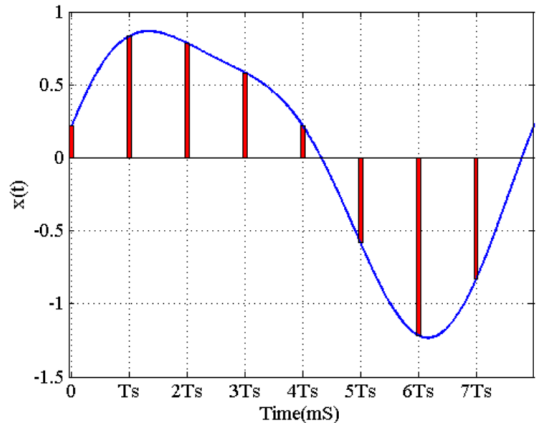


Figure: Taken from allaboutcircuits.com

$$\mathbf{x}(j) = \sin\left(\frac{2\pi}{8}j\right) + .125\sin\left(\frac{4\pi}{8}j\right) + .216\cos\left(\frac{4\pi}{8}j\right)$$

- Decompose a signal into sines and cosines at different frequencies.
- Can use this decomposition to, for example, turn up the low-frequency signals.

Discrete Fourier Transform

- Fourier modes: $\mathbf{u}_k(n) = e^{2\pi ink/N}$, $1 \leq k \leq N$.
  - Recall $e^{2\pi ink/N} = \cos(2\pi ink/N) + i\sin(2\pi ink/N)$.
- $\mathbf{u}_1, \ldots, \mathbf{u}_N$ form an orthonormal basis for $\mathbb{C}^N$.
- $$\mathbf{x} = \sum_{k=1}^{N} \hat{\mathbf{x}}(k)\mathbf{u}_k$$
- $\hat{\mathbf{x}}(k) = \langle \mathbf{u}_k, \mathbf{x} \rangle$ represents how much of $\mathbf{x}$ corresponds to frequency k.

Generalization to Graphs

On the cycle graph:

- Signal (e.g. sound) can be seen as function over time.
- Signals can be decomposed into waves/oscillations thanks to Fourier.

On graphs:

- Functions $f : V \to \mathbb{R}$ can be indentified with signals, $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{x}(k) := f(v_k)$.
- **Question:** Can we use Fourier on graph domains?
- **Yes:** We can use Fourier to study diffusion on graphs via a graph Laplacian.

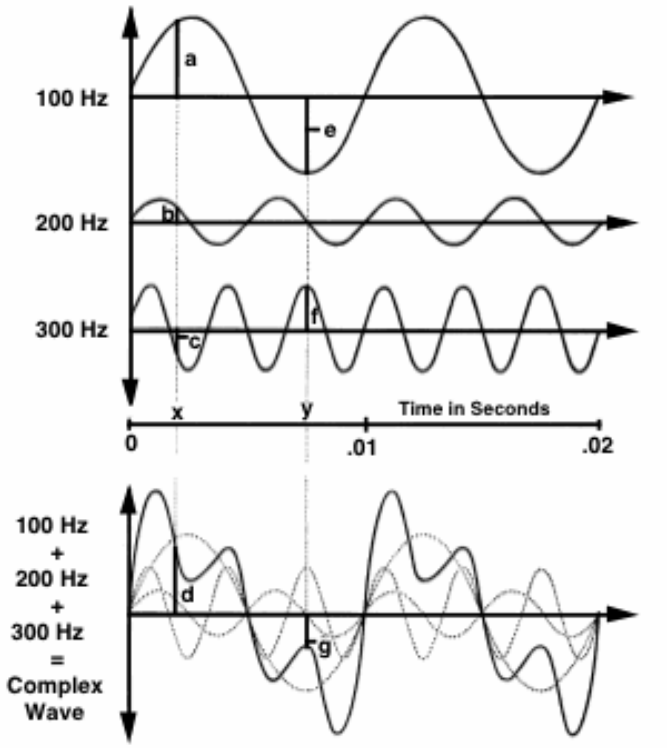


Figure: Frequency decomposition a sample signal (`hearinghealthmatters.org`)

Convolution

Definition:

$$(\mathbf{x} * \mathbf{y})(n) = \sum_m \mathbf{x}(n-m)\mathbf{y}(m).$$

Convolution Theorem:

Convolution is Fourier multiplication

$$\widehat{\mathbf{x} * \mathbf{y}}(k) = \widehat{\mathbf{y}}(k)\widehat{\mathbf{x}}(k).$$

In this graph setting, we take this theorem to be a defintion.

Convolutions in Deep Learning

Machine Learning for Images

- You have 500 photos of cats and 500 photos of dogs.
- Given a new image, how do you decide if it's a cat or a dog.

Pictures are Vectors

- A ten megapixel (greyscale) image is a collection of ten million numbers between 0 and 1.
- We can write this as a ten million dimensional vector $\mathbf{x} \in \mathbb{R}^N$, $N = 10,000,000$.
- $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$.

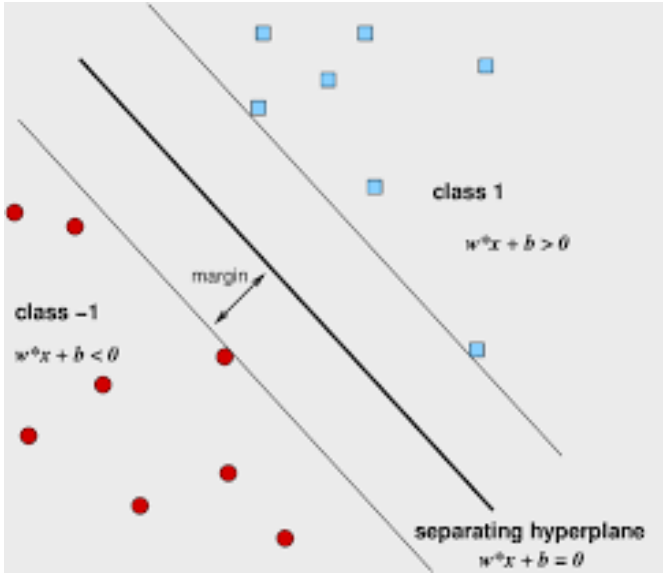
Deep Neural Networks

Hidden representation of each image

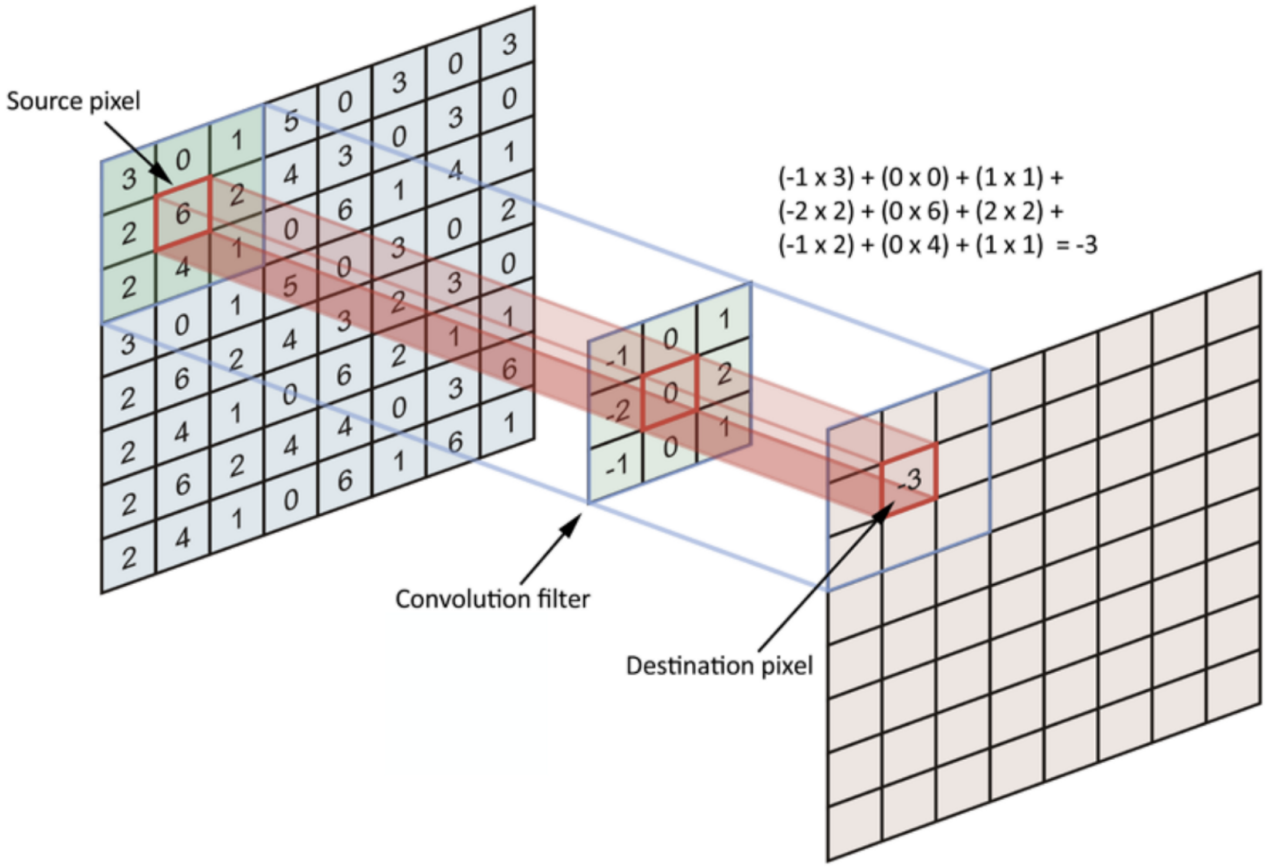
- Non-linear, multi-layered structure

$$\mathbf{x} \to A_1\mathbf{x} \to \sigma(A_1\mathbf{x}) \to A_2\sigma(A_1\mathbf{x}) \to \sigma(A_2\sigma(A_1\mathbf{x}))$$

- A_1, and A_2 are $p \times N$ and $m \times p$
- σ is an entrywise, nonlinear function, e.g. absolute value
- Use these hidden representations for various tasks



Convolutional Neural Networks

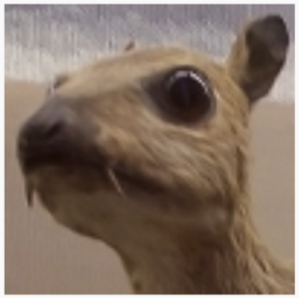


$$(\mathbf{x} * \mathbf{y})(\vec{n}) = \sum_{\vec{m}} \mathbf{x}(\vec{n} - \vec{m})\mathbf{y}(\vec{m}), \quad \vec{n} = (n_1, n_2), \vec{m} = (m_1, m_2)$$

Image Filtering

Edge detection

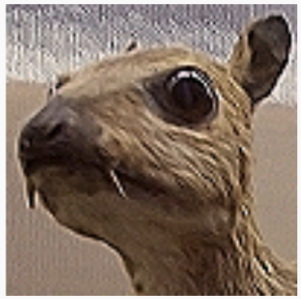
Kernel



$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$

Sharpen

$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$



Graph Convolutional Networks

- Key Challenge: Need a notation of convolution on graphs
- Naive definition doesn't make sense

$$(\mathbf{x} * \mathbf{y})(v) = \sum_{w} \mathbf{x}(v - w)\mathbf{y}(v)$$

- The subtraction of vertices $v - w$ doesn't make sense.
- Need something else!
- Once we have graph convolution, then things are (somewhat) the same as with images.

Eigenvectors to the rescue

- Recall from earlier: $\widehat{\mathbf{x} * \mathbf{y}}(k) = \widehat{\mathbf{y}}(k)\widehat{\mathbf{x}}(k)$
- The eigenvectors of $\mathbf{L} = \mathbf{D} - \mathbf{A}$, $\mathbf{L}\mathbf{u}_k = \lambda_k \mathbf{u}_k$ are called *generalized Fourier modes*.
- Definition: Fourier Coefficients

$$\widehat{\mathbf{x}}(k) = \langle \mathbf{x}, \mathbf{u}_k \rangle.$$

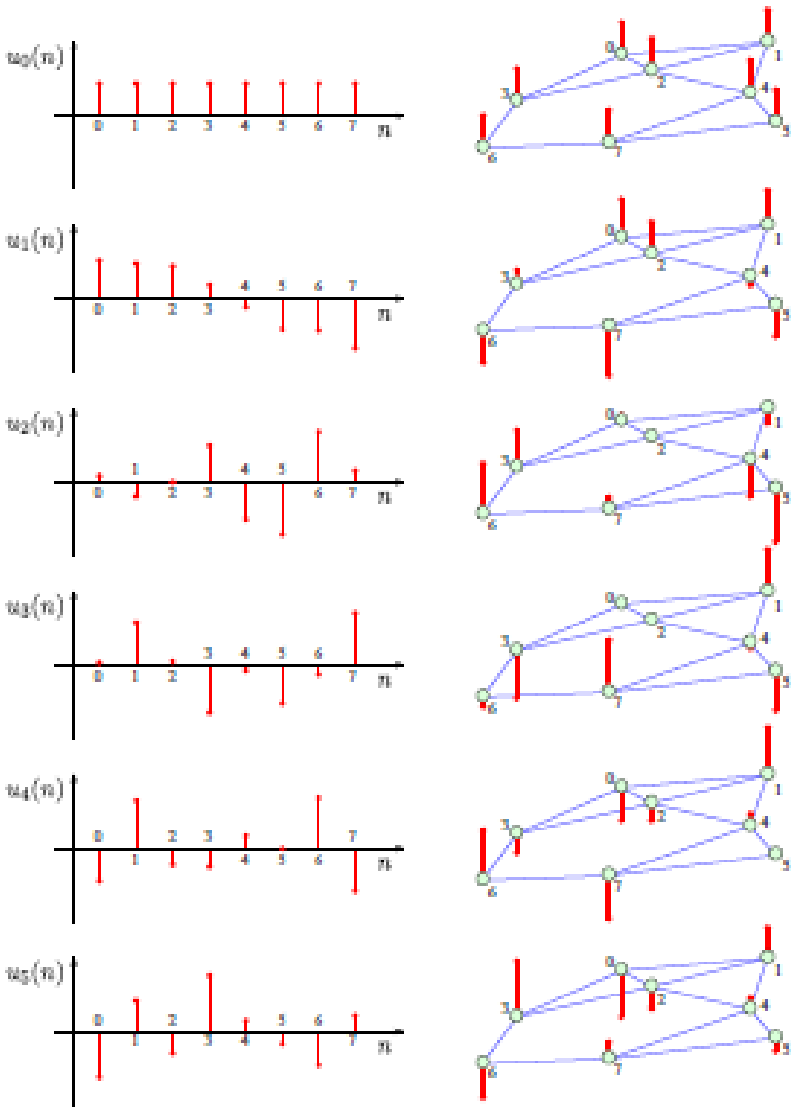
- Definition:

$$\widehat{\boldsymbol{y} \star \boldsymbol{x}}(k) := \hat{\boldsymbol{y}}(k)\hat{\boldsymbol{x}}(k).$$

So then

$$\boldsymbol{y} \star \boldsymbol{x} := \sum_k \widehat{\boldsymbol{y} \star \boldsymbol{x}}(k)\mathbf{u}_k.$$

Eigenvectors Illustrated



Spectral GNNs

Spectral GNNs

Generalized Convolutional Neural Networks

- Alternating sequence of linear filterings and pointwise activations
- Filters are generalized convoutions defined via the graph Fourier transform
- Reduce to classic CNNs when the graph is a cycle.

Problem

- Computing Eigenvectors is expensive
- $\mathcal{O}(N^3)$ operations

Avoid computing eigenvectors

First observe:

Inductively, we have $\mathbf{L}^n\mathbf{u}_k = \lambda_k^n\mathbf{u}_k$. Therefore if $p(\cdot)$ is any polynomial, we have

$$p(\mathbf{L})\mathbf{u}^k = p(\lambda_k)\mathbf{u}_k$$

Therefore:

If $\mathbf{q}$ is a vector with $\widehat{\mathbf{q}}(k) = p(\lambda_k)$, we have

$$\begin{aligned} p(\mathbf{L})\mathbf{x} &= p(\mathbf{L})\sum_{k=1}^{n}\widehat{\mathbf{x}}(k)\mathbf{u}_k \\ &= \sum_{k=1}^{n}\widehat{\mathbf{x}}(k)p(\mathbf{L})\mathbf{u}_k \\ &= \sum_{k=1}^{n}\widehat{\mathbf{x}}(k)p(\lambda_k)\mathbf{u}_k = \mathbf{q}\star\mathbf{x} \end{aligned}$$

ChebNet

ChebNet (Defferrard, Bresson, Vandergheynst)

Let T_k be the K-th Chebyshev polynomial and use filters:

$$\mathbf{y}_\theta \star \mathbf{x} = \sum_{k=0}^{K} \theta_k T_k(\mathbf{L})\mathbf{x}.$$

$$T_0(\mathbf{L}) = \mathbf{I},\ T_1(\mathbf{L}) = \mathbf{L},\ T_n(\mathbf{L}) = 2\mathbf{L}\,T_{n-1}(\mathbf{L}) - T_{n-2}(\mathbf{L}).$$

(This idea was previously applied to graph wavelets by Hammond et al.)

The Graph Convolutional Network

GCN (Kipf and Welling)

Setting $K = 1$, $\theta_0 = -\theta_1$, and apply a renormalization trick yields

$$\mathbf{y}_\theta \star \mathbf{x} = \theta \widetilde{\mathbf{D}}^{-1/2} \widetilde{\mathbf{A}} \widetilde{\mathbf{D}}^{-1/2} \mathbf{x} := \theta \widehat{\mathbf{A}} \mathbf{x}, \quad \widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}, \quad \widetilde{\mathbf{D}} = \widetilde{\mathbf{A}} \mathbf{1}.$$

With multiple channels and filters

$$\mathbf{Z} = \widehat{\mathbf{A}} \mathbf{X} \mathbf{\Theta}, \quad \mathbf{X} = (\mathbf{x}_1 \dots \mathbf{x}_F).$$

Important Note: $\mathbf{\Theta}$ is learned but $\widehat{\mathbf{A}}$ is designed.

Spectral vs Spatial

- Line between spectral and spatial GNNs is not clear
- GCN is both a spectral network and a spatial network

Semi-Supervised Node Classification

Setup

- All Vertices and Edges
- Node features $\mathbf{x}_1, \ldots, \mathbf{x}_F$ for all graph nodes
- Labels for some nodes ($\leq$5%)

- Popular methods such as GCN suffer from **oversmoothing** (Li 2018)

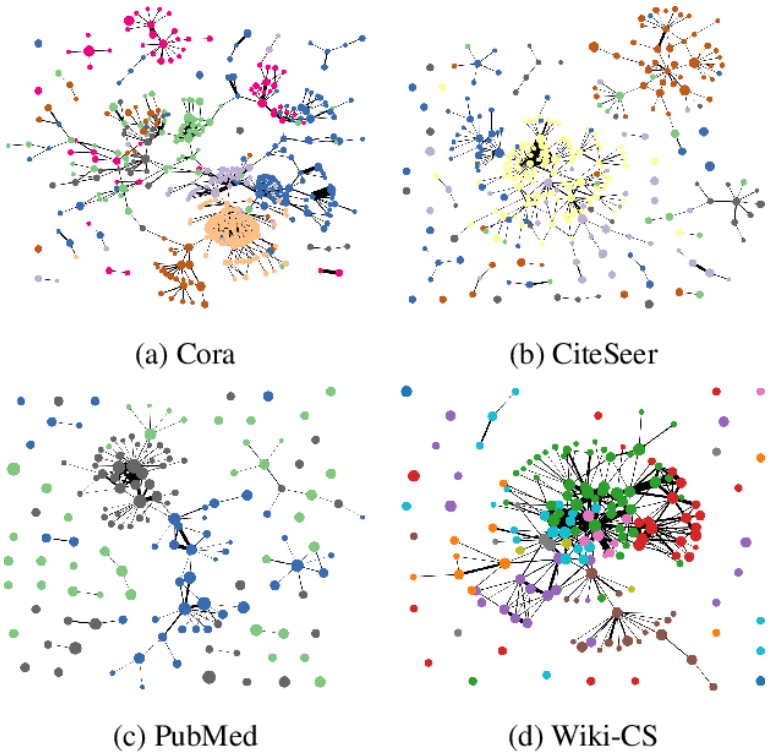


Figure: Visualizations of benchmarks, taken from Mernyei 2020.

Oversmoothing

Graph Convolutional Network (Kipf and Welling)

$$\mathbf{Z} = \widehat{\mathbf{A}}\mathbf{X}\mathbf{\Theta}$$

- The matrix $\widehat{\mathbf{A}}$ acts a local-averaging operator.
- Promotes smoothness, i.e., similarity amongst neighbors

Low-pass filter

- Multiplying by $\widehat{\mathbf{A}}$ leaves bottom eigenvector unchanged.
- All other frequencies are depressed.
- Repeated applications increasingly depress high-frequencies.
- "Deep" Graph Neural Nets typically use two layers.
- Can't process long-rang interactions.

Overcoming Oversmoothing

(Hybrid) Geometric Scattering Transforms

- Lazy-random walk matrix $\mathbf{P} = \frac{1}{2}(\mathbf{I} + \mathbf{D}^{-1}\mathbf{A})$.
  - Note $\mathbf{P} = \mathbf{I} - \frac{1}{2}\mathbf{L}_{RW}$, $\mathbf{L}_{RW} = \mathbf{D}^{-1}\mathbf{L}$
- Graph Wavelets:

$$\mathbf{W}_j = \mathbf{P}^{2^{j-1}} - \mathbf{P}^{2^j}.$$

- Apply many different types of filters in each layer (both low-pass and wavelet) and let the network learn how to balance them.
- Capture both low-frequency and high-frequency info
- Also capture both local and global graph structure

Combinatorial Optimization (P., Min, Wenkel, Wolf)

The Max Clique Problem

- Scattering based methods are faster than traditional solvers and more accurate than other GNNs
- Wavelets help detect the edges of the clique

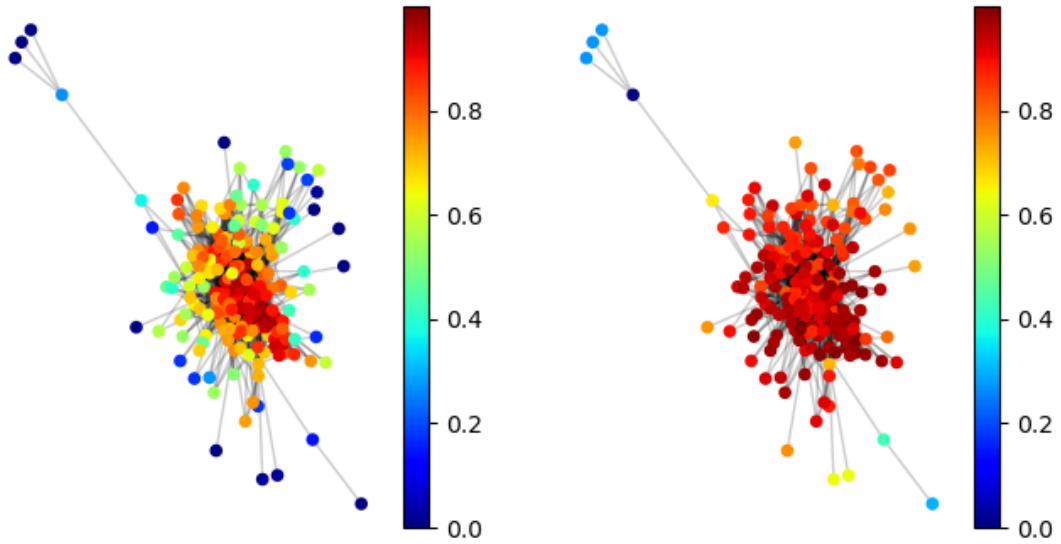


Figure: Hidden representations produced by a scattering-based network (shown on the left) and a traditional GCN (shown on the right)

Encoding robust representation for graph generation (Zou and Lerman 2019)

- Encoder E = Graph Scattering Transform
- Decoder D = Fully Connected Network
- $D \circ E = Id$
- Generate new graphs by adding noise in latent space

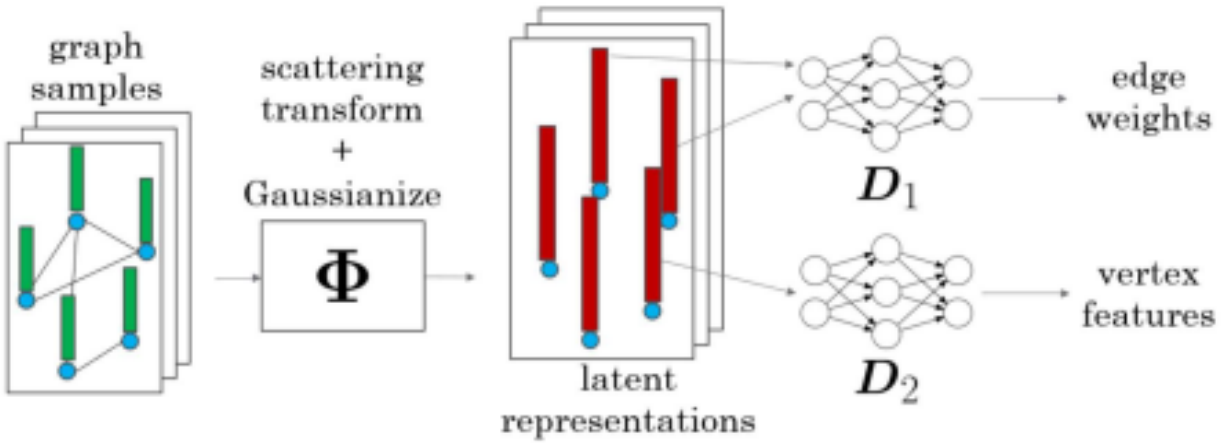


Figure: Scattering Encoder-Decoder Network

Molecular Graph Generation via Geometric Scattering (GRASSY) - Bhaskar, Grady, P., Krishnaswamy

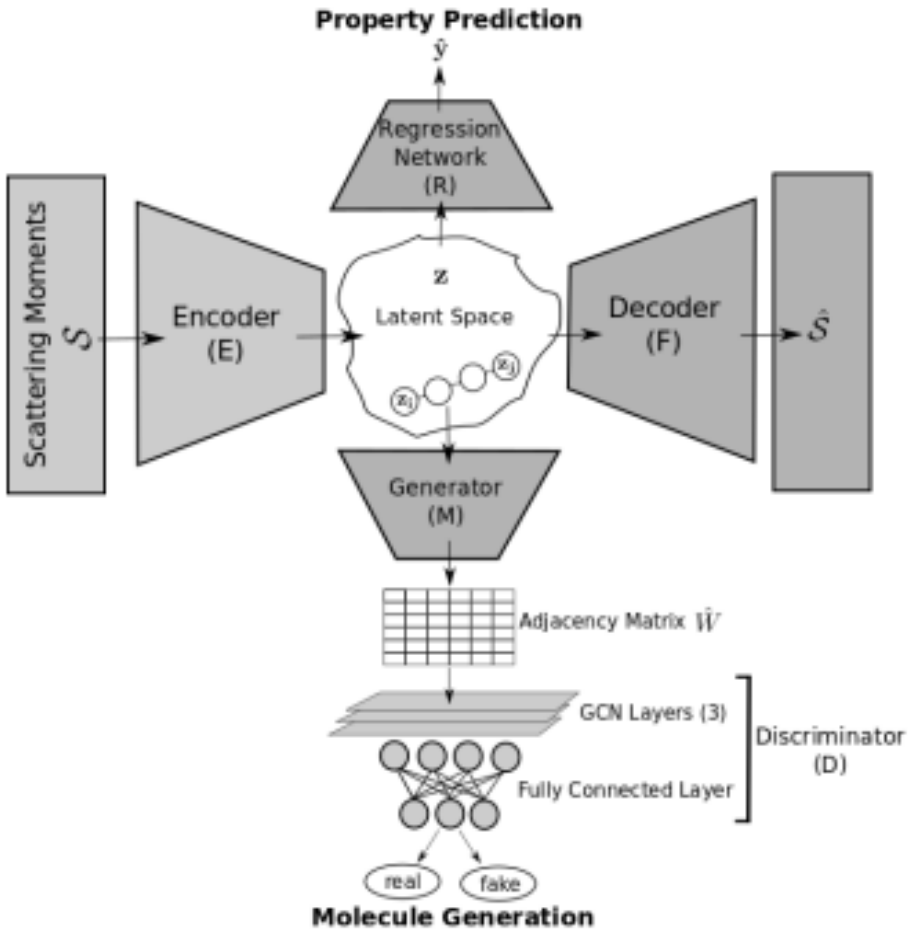


Figure: GRaph Scattering SYnthesis network

Generalized Graphs

High-dimensional (biomedical) point cloud data

- Given: $\{x_i\}_{i=1}^N \subseteq \mathbb{R}^D$
- Modeling assumption: the 'true' dimension of the data d is much lower than the ambient dimension D (Riemannian manifold).
- Turn data into a graph, $G_N = (V_N, E_N)$ $V_N = \{x_i\}_{i=1}^N$, edges between nearby points.
- Theory showing (that if things are nice and we do things smartly) that the eigenvalues of the graph Laplacian $\mathbf{L}_N$ converge to the eigenvalues of the continuous manifold Laplacian $\mathcal{L}$ as $N \to \infty$.
- Very recent results showing that spectral GNNs will also converge to a continuum limit.

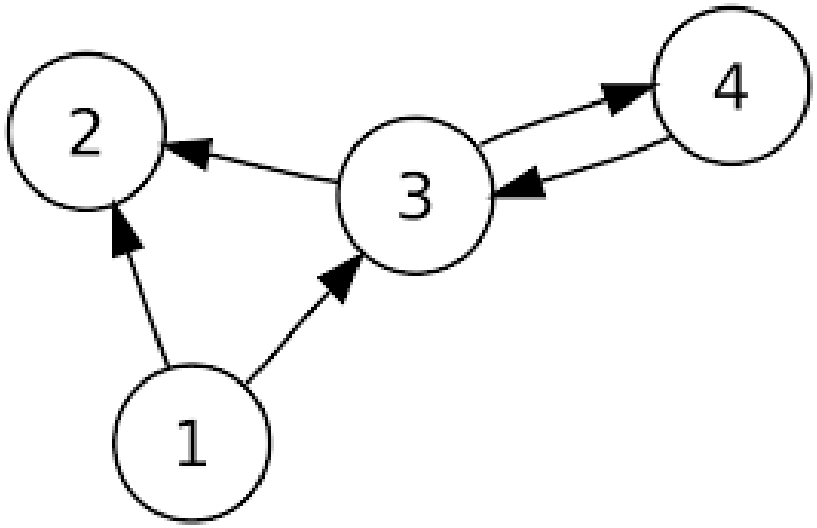
Directed Graphs

Setup:

- Adjacency matrix **A** is asymmetric,
  - $A(j,k) \neq 0 \Leftrightarrow (v_j, v_k) \in E$.

Examples:

Email, Citations, Traffic, Websites, Twitter, Sports



Challenge:

Naive definition of graph Laplacian is not symmetric.

Does Direction Matter?

Common Approach: Symmetrization

- Preprocess data by symmetrizing the adjacency matrix

Sometimes this is reasonable:

- Node Classification on Academic Citation Networks
  - Papers with the label 'data science' are likely to both cite, and be cited by, other data science papers.

Other times its not:

- Node Classification on Email Networks
- Link Prediction on Citation Networks
  - A paper with many citations is not the same as a paper with many references

Hermitian Adjacency Matrix

$$\mathbf{H}(j,k) = \begin{cases} 1 & \text{if } (v_j, v_k) \in E \text{ and } (v_k, v_j) \in E \\ i & \text{if } (v_j, v_k) \in E \text{ and } (v_k, v_j) \notin E \\ -i & \text{if } (v_k, v_j) \in E \text{ and } (v_j, v_k) \notin E \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{H}$ is Hermitian meaning $\mathbf{H}(j,k) = \overline{\mathbf{H}(k,j)}$
- An unreciprocated link from v_j to v_k is treated as the negative of a link in the opposite direction.

The Magnetic Laplacian

Definition

$$\mathbf{L}_{(\text{mag})} := \mathbf{D}_{(\text{und})} - \mathbf{H},$$

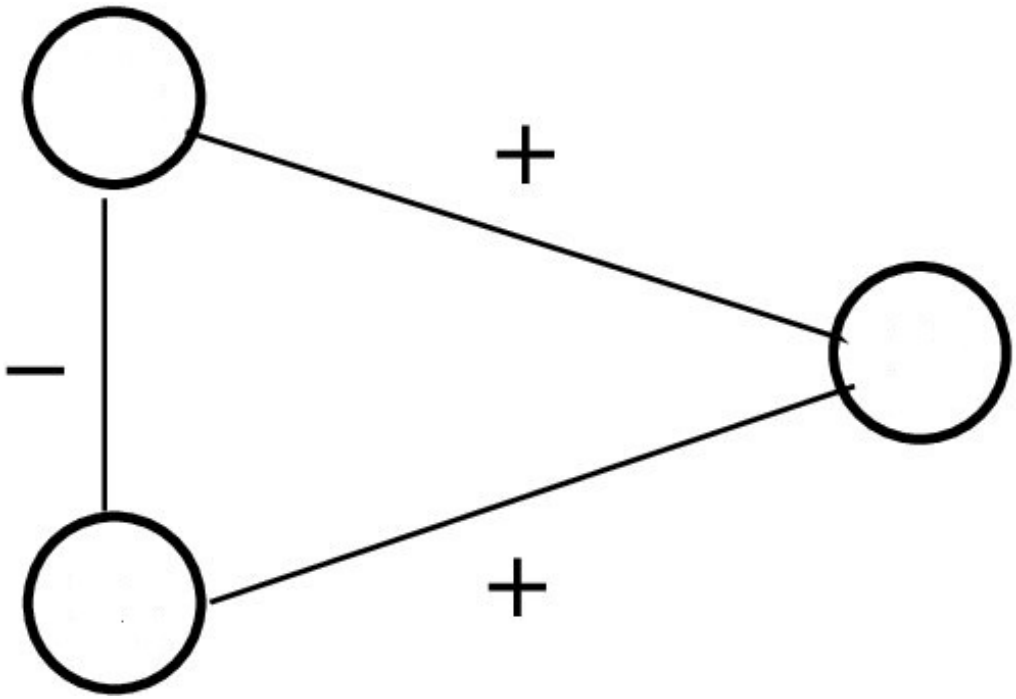
where $\mathbf{D}_{(\text{und})}$ is the degree matrix of the undirected version of G.

History

- First appears in physics literature in 1993 (Lieb and Loss)
- Ongoing research in the graph signal processing community
  - Detect certain substructures (motifs)
- Has been used for
  - Clustering
  - Community Detection
  - Directed Graph Neural Nets

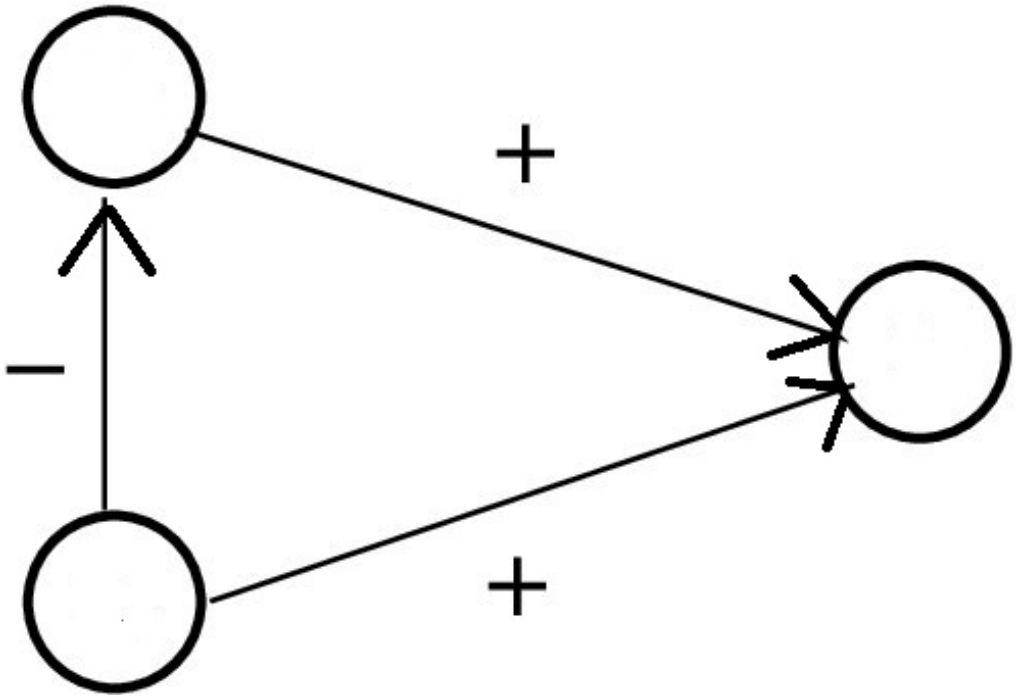
Signed Graphs

- Signed Laplacian: $\mathbf{L}_{\text{signed}} = \mathbf{D}_{\text{total}} - \mathbf{A}$
- Related to signed min-cut problem (keeps friends together and enemies apart).



Signed and Directed Graphs

- Models for e.g., influence on social networks and of financial time series (e.g., stocks)
- Magnetic Signed Laplacian: $\mathbf{L}_{\text{MS}} = \mathbf{D}_{\text{total}} - \mathbf{H}$
- Reduces to the magnetic laplacian or signed laplacian as appropriate.



Convolution on Signed and Directed Graphs

- All of these generalized Laplacians can be used to define generalized convolutions.
- Can be incorporated into spectral graph neural networks such as MagNet (2021) and MSGNN (2022).
- Essentially just re-use ChebNet architecture, but with different Laplacian.

Conclusion

- The eigenvectors of the graph Laplacian can be used to solve the min-cut problem.
- Extend signal processing to graphs.
- Spectral Graph Neural Networks.
- Generalized Laplacians for signed and/or directed graphs.

THANK YOU!

Encoding robust representation for graph generation (Zou and Lerman 2019)

- Encoder E = Graph Scattering Transform
- Decoder D = Fully Connected Network
- $D \circ E = Id$
- Generate new graphs by adding noise in latent space

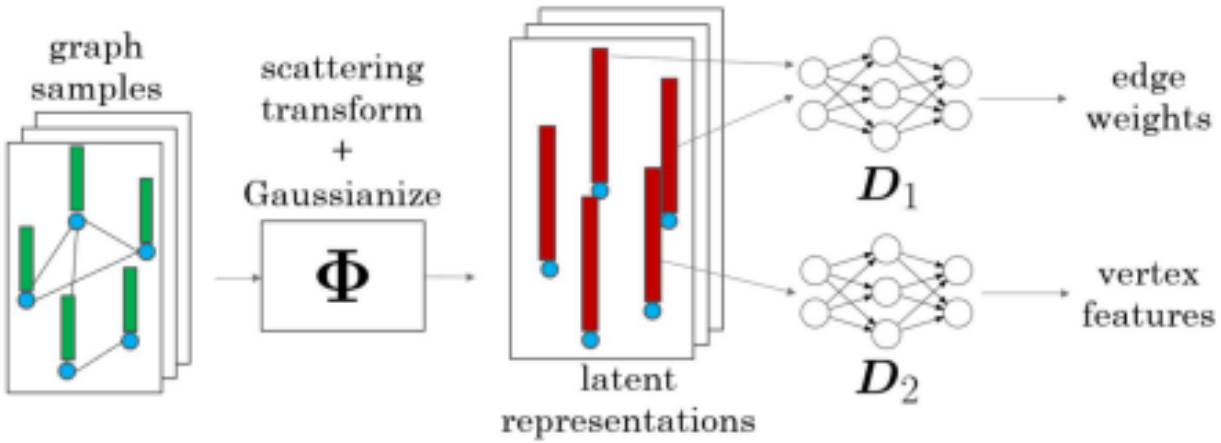


Figure: Scattering Encoder-Decoder Network

Molecular Graph Generation via Geometric Scattering (GRASSY) - Bhaskar, Grady, P., Krishnaswamy

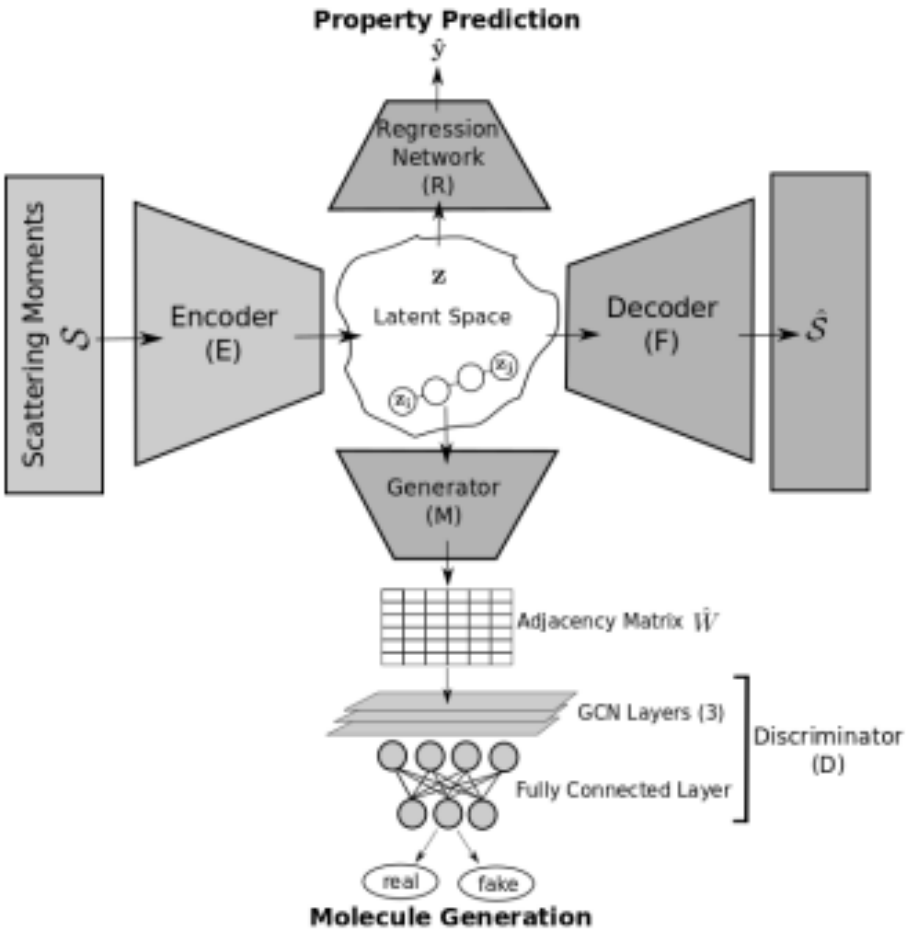
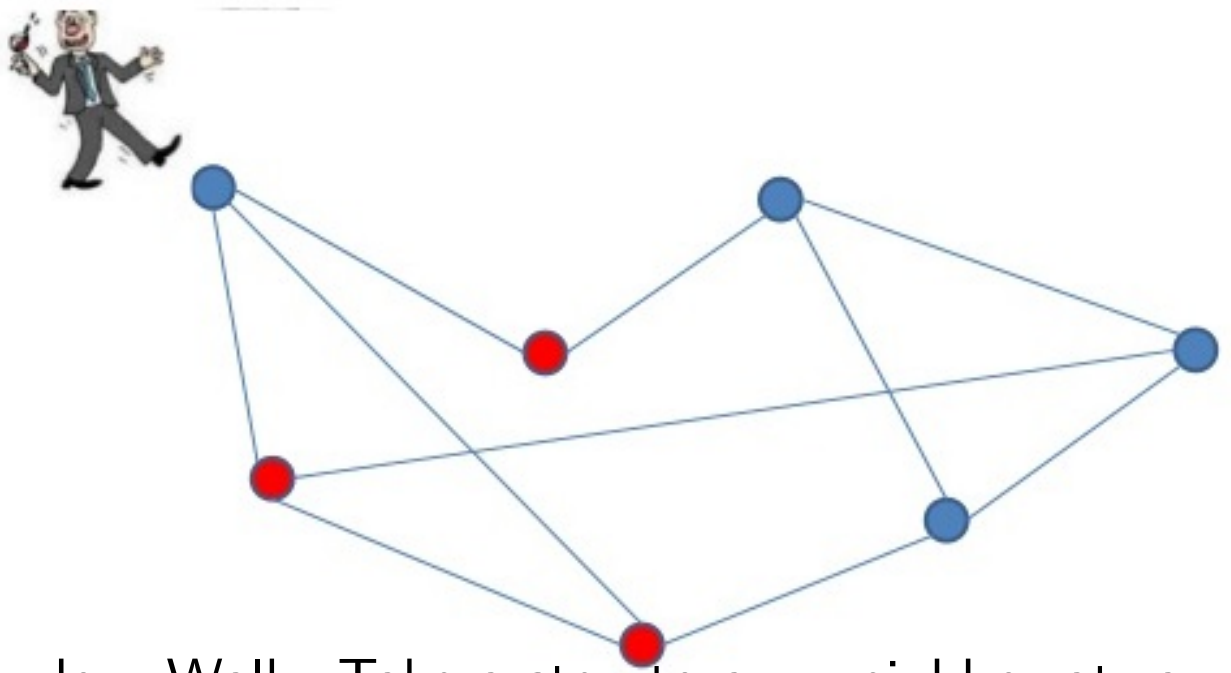


Figure: GRaph Scattering SYnthesis network

Random Walks and PageRank Approaches

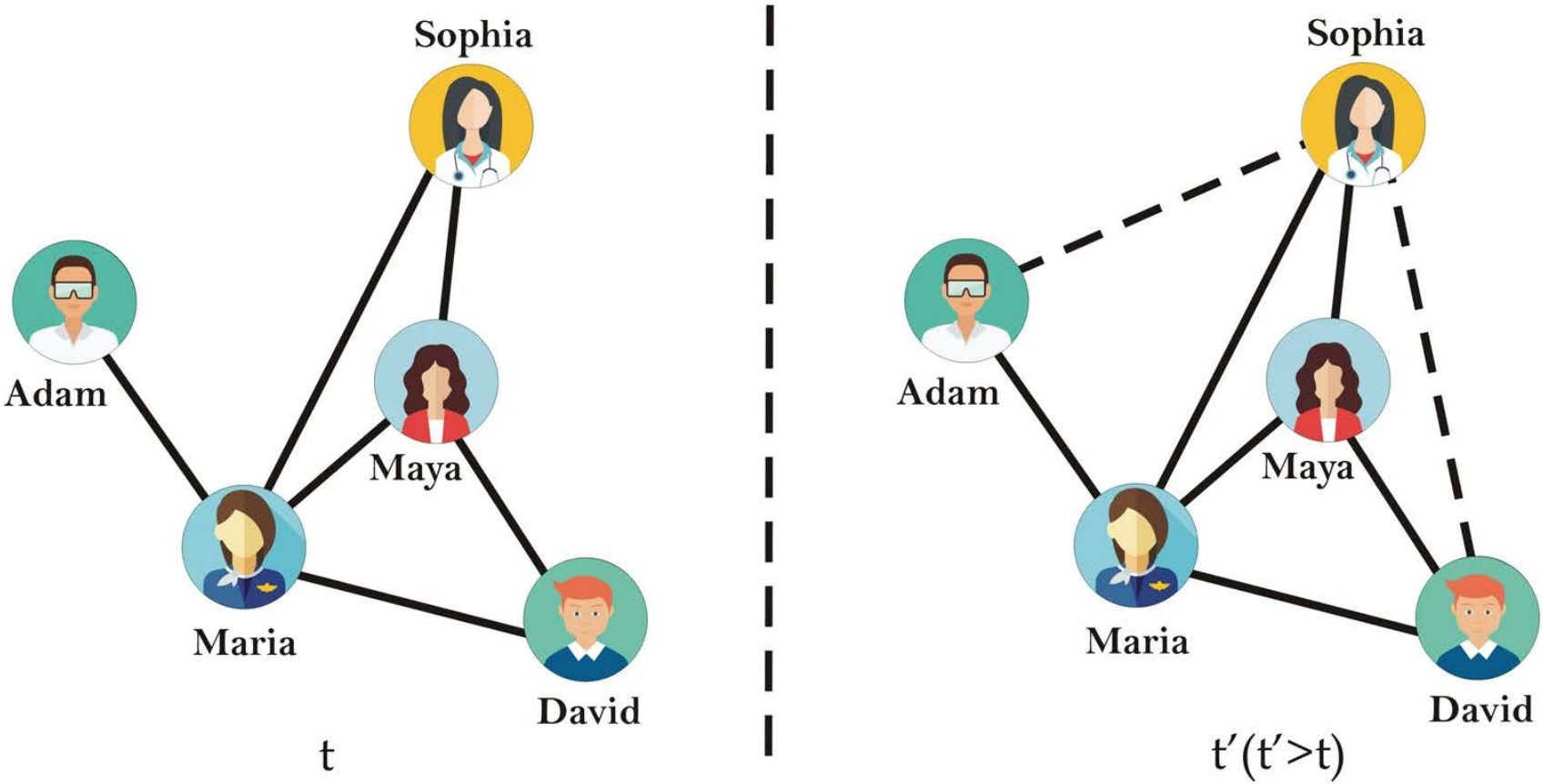


- Random Walk: Take a step to any neighbor at random
- One can derive an alebraic relationship between Random Walk matrices and Graph Laplacians
- PageRank: Teleport to random location with probability α, otherwise perform random walk step
- Use random walks / page rank to define spectral GNNs (Tong, Liang, Sun, Li, Rosenblum, Lim) AND (Ma, Hao, Yang, Li, Jin)

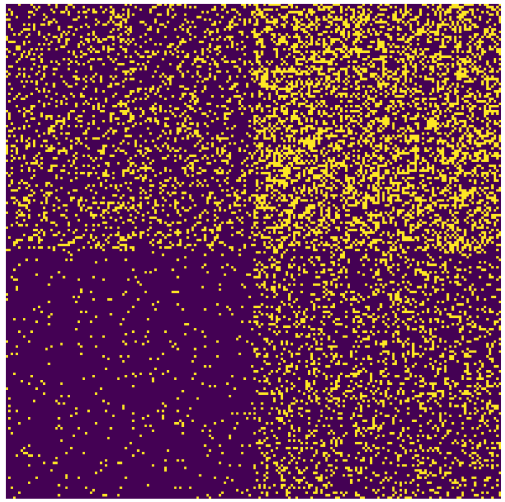
Edge-level tasks

Link Prediction

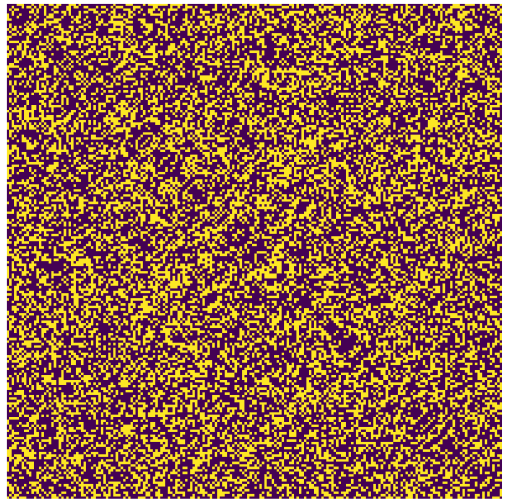
Suggest "people you may know" on Facebook, LinkedIn, etc.



Directed Stochastic Block Model.



(a) Directed adjacency



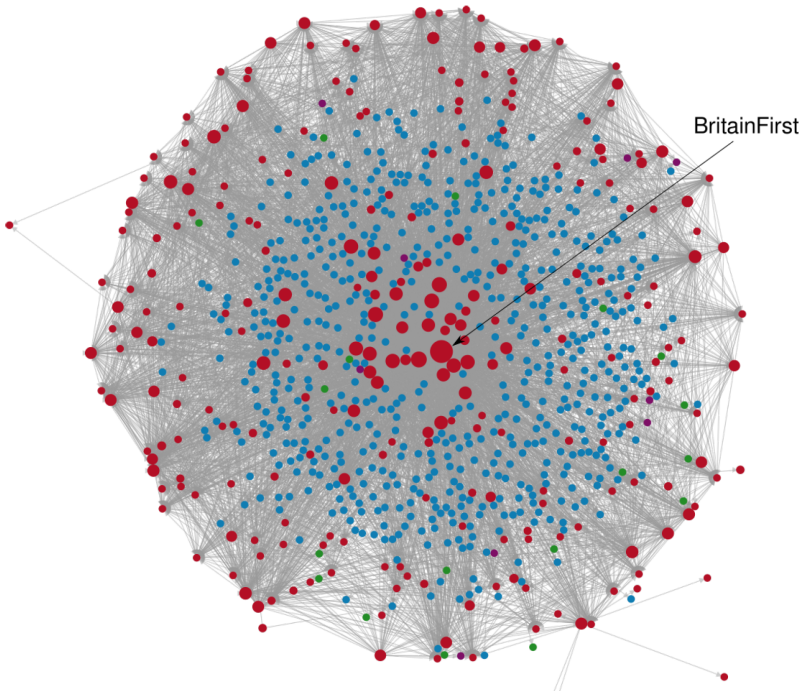
(b) Symmetrized adjacency

Figure: Directed edges point from cluster 1 to cluster 2 90% of the time.

Telegram Dataset

Activity of the far right

- Bovet and Grindrod constructed a directed graph of "influence" on Telegram.
- Analyzed core-peripherary structure (channels in red, websites in blue).



Eigenvectors Illustrated

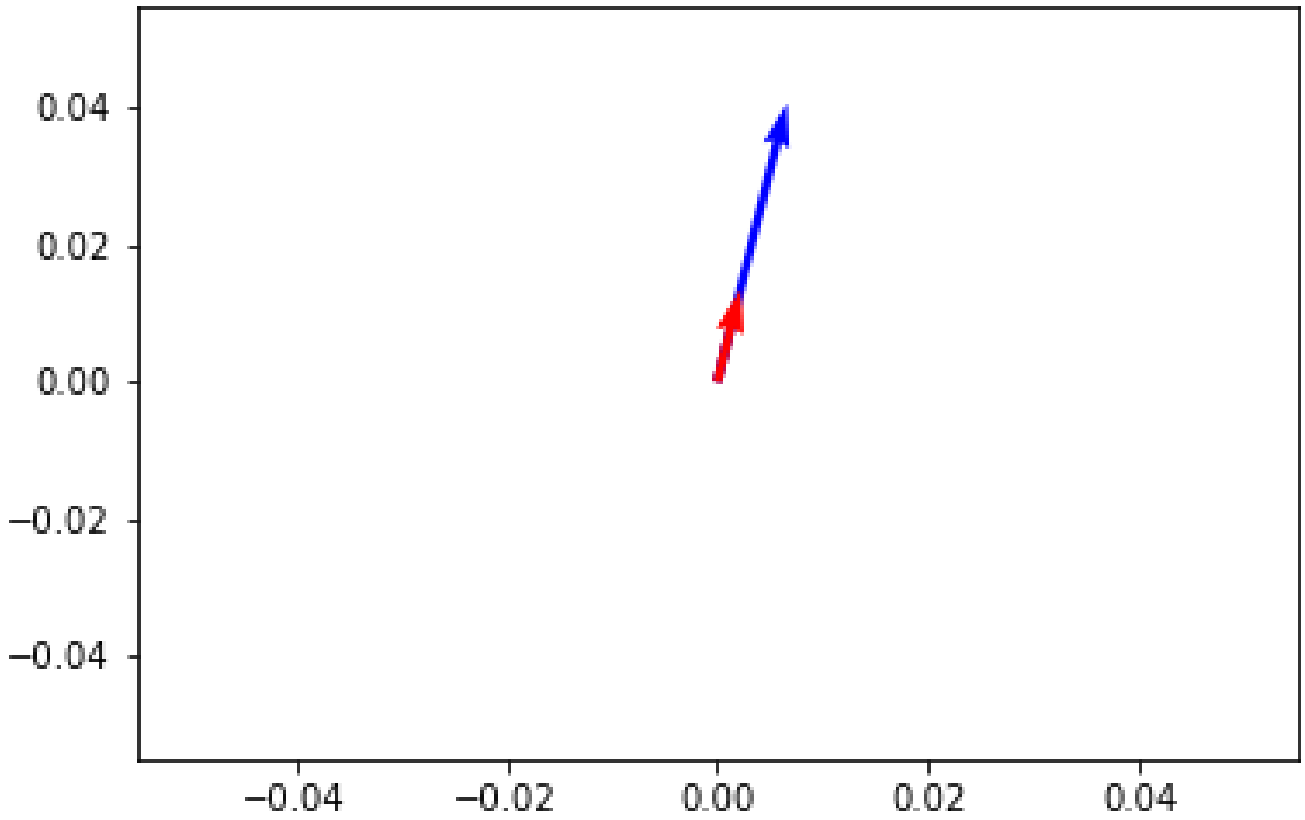
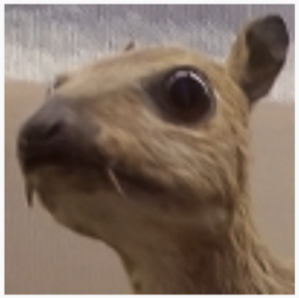


Figure: $\mathbf{v} = (1, 4)$ in red, $\mathbf{Av} = (6, 24)$ in blue

Image Filtering

Edge detection

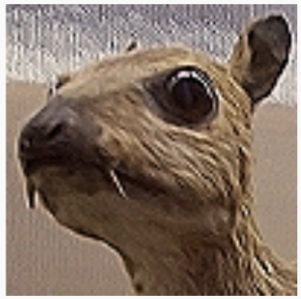
Kernel



$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$

Sharpen

$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$



Some Linear Algebra

(Orthonormal) Basis

A collection of N-dimensional vectors $\mathbf{u}_1, \ldots, \mathbf{u}_N$ is called a *basis* if every N-dimensional vector can be unique written as a linear combination of the $\mathbf{u}_j$

$$\mathbf{x} = c_1\mathbf{u}_1 + \ldots c_N\mathbf{u}_N.$$

The basis is called *orthonormal* is each $\mathbf{u}_j$ has length 1 and

$$\mathbf{u}_j \cdot \mathbf{u}_k = 0$$

for all $j \neq k$.

Example

$$\mathbf{u}_1 = (1, 0, 0),\ \mathbf{u}_2 = (0, 1, 0),\ \mathbf{u}_3 = (0, 0, 1)$$

$$(4, 5, 7) = 4\mathbf{u}_1 + 5\mathbf{u}_2 + 7\mathbf{u}_3$$

Some More Linear Algebra

Eigenvectors

A vector $\mathbf{v}$ is called an *eigenvector* of a matrix $\mathbf{A}$ if

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

for a scalar λ which is called the *eigenvalue*.

Example

$$\begin{pmatrix} -6 & 3 \\ 4 & 5 \end{pmatrix} \begin{pmatrix} 1 \\ 4 \end{pmatrix} = \begin{pmatrix} -6 \times 1 + 3 \times 4 \\ 4 \times 1 + 5 \times 4 \end{pmatrix} = \begin{pmatrix} 6 \\ 24 \end{pmatrix} = 6 \begin{pmatrix} 1 \\ 4 \end{pmatrix}.$$