



IP PARIS



Spatiotemporal Analysis with Graphs

APM_5DS30_TP, Machine Learning with Graphs

Jhony H. Giraldo (Assistant Professor).

jhony.giraldo@telecom-paris.fr

November 7, 2025

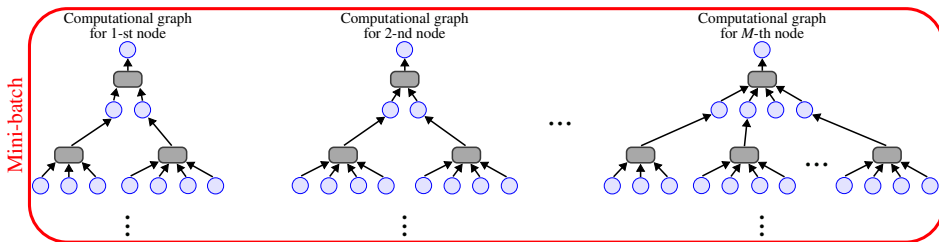


What do you remember from the last lecture(s)?

Recap

GraphSAGE Neighbor Sampling

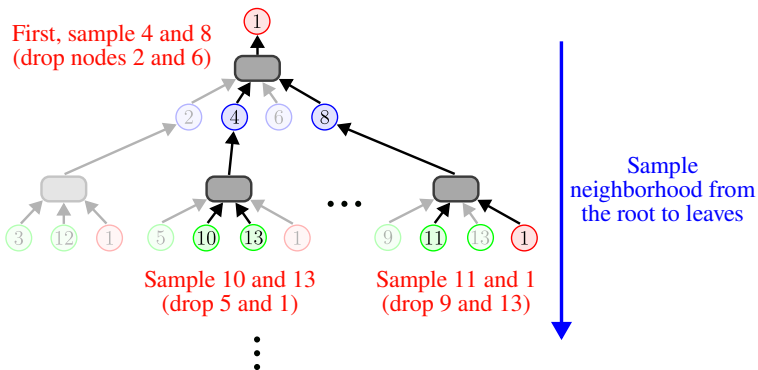
- **Key insight:** We need the **K-hop neighborhood** to compute the embedding of a single node
- Given a set of M different nodes in a mini-batch, we can generate their embeddings using M computational graphs (we can parallelize this using a GPU)



Recap

GraphSAGE Neighbor Sampling

- **Key idea:** Construct the computational graph by randomly sampling at most H neighbors at each hop
- Now we can use the pruned computational graph to do message-passing



Recap

Cluster-GCN

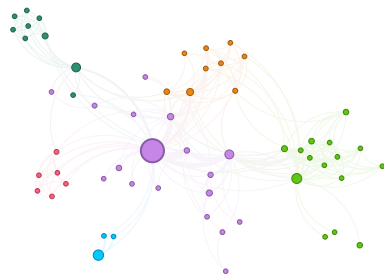
- **Pre-processing:** Given a large graph $G = (\mathcal{V}, \mathcal{E})$, we divide it into C sub-graphs $G_1, G_2, \dots, G_C$



Recap

Cluster-GCN

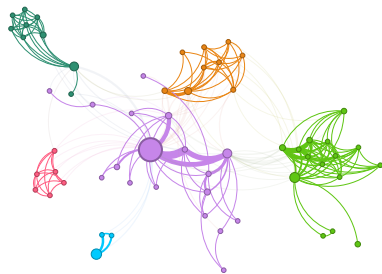
- ▶ **Pre-processing:** Given a large graph $G = (\mathcal{V}, \mathcal{E})$, we divide it into C sub-graphs $G_1, G_2, \dots, G_C$
- ▶ For doing so, we divide the set of nodes $\mathcal{V}$ into C partitions $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_C$ such that $\bigcup_{i=1}^C \mathcal{V}_i = \mathcal{V}$



Recap

Cluster-GCN

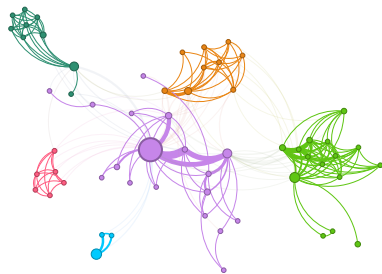
- ▶ **Pre-processing:** Given a large graph $G = (\mathcal{V}, \mathcal{E})$, we divide it into C sub-graphs $G_1, G_2, \dots, G_C$
- ▶ For doing so, we divide the set of nodes $\mathcal{V}$ into C partitions $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_C$ such that $\bigcup_{i=1}^C \mathcal{V}_i = \mathcal{V}$
- ▶ Each $\mathcal{V}_i$ induces a sub-graph $G_i = (\mathcal{V}_i, \mathcal{E}_i)$



Recap

Cluster-GCN

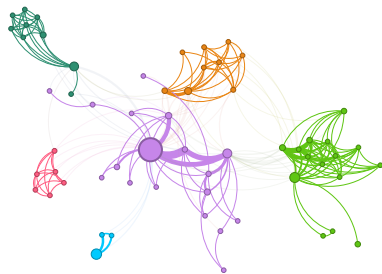
- ▶ **Pre-processing:** Given a large graph $G = (\mathcal{V}, \mathcal{E})$, we divide it into C sub-graphs $G_1, G_2, \dots, G_C$
- ▶ For doing so, we divide the set of nodes $\mathcal{V}$ into C partitions $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_C$ such that $\bigcup_{i=1}^C \mathcal{V}_i = \mathcal{V}$
- ▶ Each $\mathcal{V}_i$ induces a sub-graph $G_i = (\mathcal{V}_i, \mathcal{E}_i)$
- ▶ Notice that in general $\bigcup_{i=1}^C \mathcal{E}_i \neq \mathcal{E}$ since the **between-group** edges are not included in $G_1, \dots, G_C$



Recap

Cluster-GCN

- ▶ **Pre-processing:** Given a large graph $G = (\mathcal{V}, \mathcal{E})$, we divide it into C sub-graphs $G_1, G_2, \dots, G_C$
- ▶ For doing so, we divide the set of nodes $\mathcal{V}$ into C partitions $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_C$ such that $\bigcup_{i=1}^C \mathcal{V}_i = \mathcal{V}$
- ▶ Each $\mathcal{V}_i$ induces a sub-graph $G_i = (\mathcal{V}_i, \mathcal{E}_i)$
- ▶ Notice that in general $\bigcup_{i=1}^C \mathcal{E}_i \neq \mathcal{E}$ since the **between-group** edges are not included in $G_1, \dots, G_C$
- ▶ Once you have the C induced sub-graphs, you apply the regular GNN's layer-wise node update over G_i to obtain the node embedding $\mathbf{h}_v$ for each node $v \in \mathcal{V}_i$

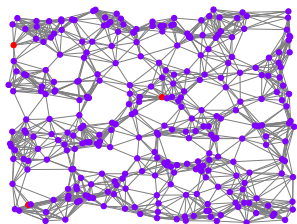


Recap

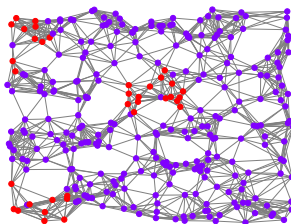
Simple Graph Convolution (SGC)

$$\bar{\mathbf{Y}} = f(\mathbf{X}; \mathbf{A}, \Theta) = \text{softmax} \left(\hat{\mathbf{A}}^l \mathbf{X} \mathbf{W} \right)$$

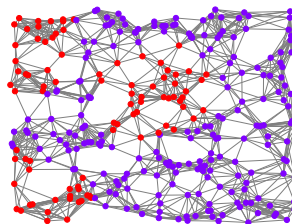
- ▶ $\hat{\mathbf{A}}^l \mathbf{X}$ diffuses node embeddings along the graph
- ▶ Remember the diffusion sequence in our first lecture, where the shift operator $\mathbf{S}$ is $\hat{\mathbf{A}}$ in this case



$$\mathbf{x}^{(0)} = \mathbf{x} = \mathbf{S}^0 \mathbf{x}$$

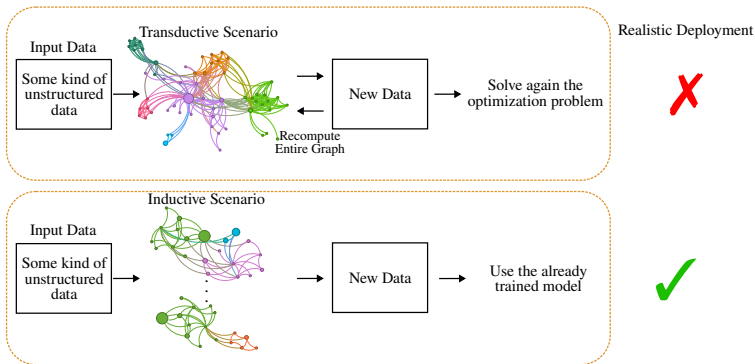


$$\mathbf{x}^{(1)} = \mathbf{S} \mathbf{x}^{(0)} = \mathbf{S}^1 \mathbf{x}$$



$$\mathbf{x}^{(2)} = \mathbf{S} \mathbf{x}^{(1)} = \mathbf{S}^2 \mathbf{x}$$

Question: How do we tackle an expanding graph? e.g., a new user on Instagram.



Question: What's the difference between vanilla Transformers and GNNs like GAT?

- ▶ We can use Transformers in graphs as well: Let $\mathbf{Q} \in \mathbb{R}^{N \times d}$, $\mathbf{K} \in \mathbb{R}^{N \times d}$, and $\mathbf{V} \in \mathbb{R}^{N \times d_v}$ be the query, key, and value matrices. The vanilla self-attention is given as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right) \mathbf{V}.$$

- ▶ Computational and memory complexity of $\mathbf{Q}\mathbf{K}^\top$ are dominated by $\mathcal{O}(N^2)$
- ▶ This is not even possible for medium-scale graphs. It's even worse for large-scale graphs

A Brief Story on Spatiotemporal Analysis

A Brief Story on Spatiotemporal Analysis

2020



A Brief Story on Spatiotemporal Analysis

2020



A Brief Story on Spatiotemporal Analysis

2020

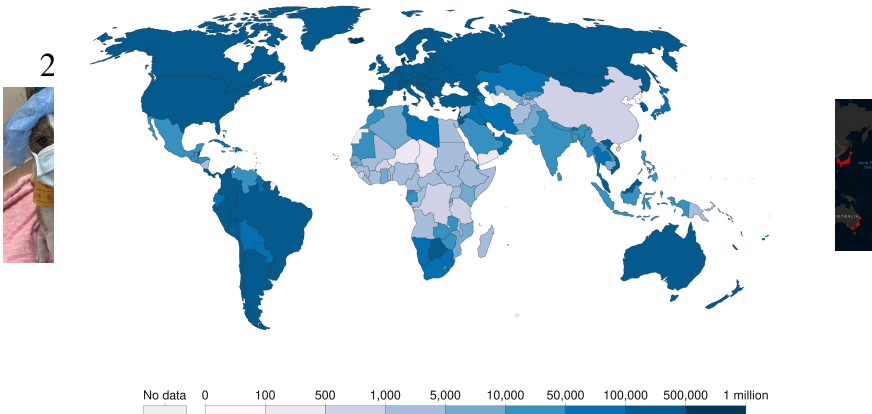


A Brief Story on Spatiotemporal Analysis

Cumulative confirmed COVID-19 cases per million people, Jun 20, 2022

Due to limited testing, the number of confirmed cases is lower than the true number of infections.

Our World
in Data



Source: Johns Hopkins University CSSE COVID-19 Data

CC BY

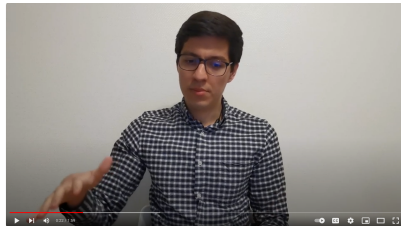
A Brief Story on Spatiotemporal Analysis

2020



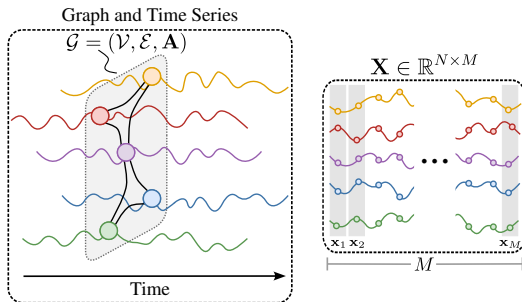
A Brief Story on Spatiotemporal Analysis

2020



Basic Definitions

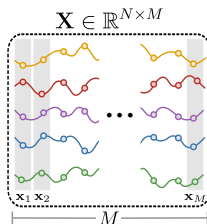
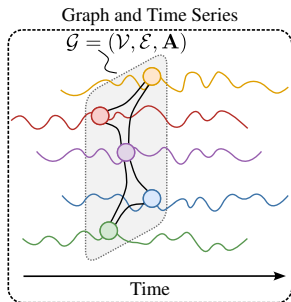
Time-varying Graph Signals



- $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M] \in \mathbb{R}^{N \times M}$ is a **time-varying graph signal**, where $\mathbf{x}_s \in \mathbb{R}^N$ is a graph signal in $\mathcal{G}$ at time s

Basic Definitions

Time-varying Graph Signals

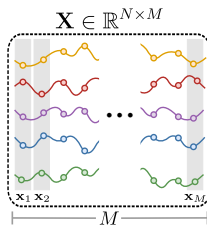
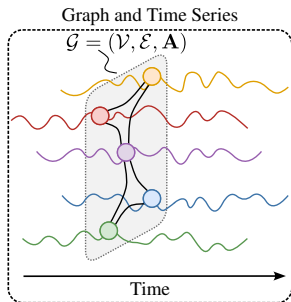


$$\mathbf{D}_h = \begin{bmatrix} -1 & & & & \\ 1 & -1 & & & \\ & 1 & \ddots & & \\ & & \ddots & -1 & \\ & & & 1 & \end{bmatrix} \in \mathbb{R}^{M \times (M-1)}$$

- ▶ $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M] \in \mathbb{R}^{N \times M}$ is a **time-varying graph signal**, where $\mathbf{x}_s \in \mathbb{R}^N$ is a graph signal in $\mathcal{G}$ at time s
- ▶ $\mathbf{D}_h$ is the temporal difference operator

Basic Definitions

Time-varying Graph Signals

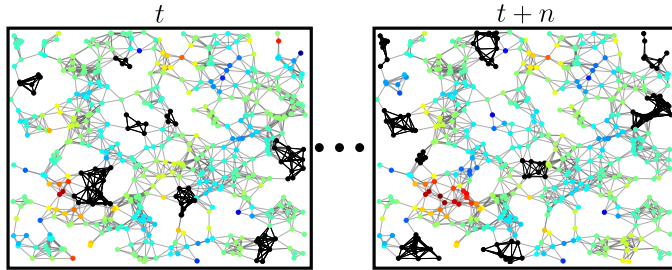


$$\mathbf{D}_h = \begin{bmatrix} -1 & & & & \\ 1 & -1 & & & \\ & 1 & \ddots & & \\ & & \ddots & -1 & \\ & & & 1 & \end{bmatrix} \in \mathbb{R}^{M \times (M-1)}$$

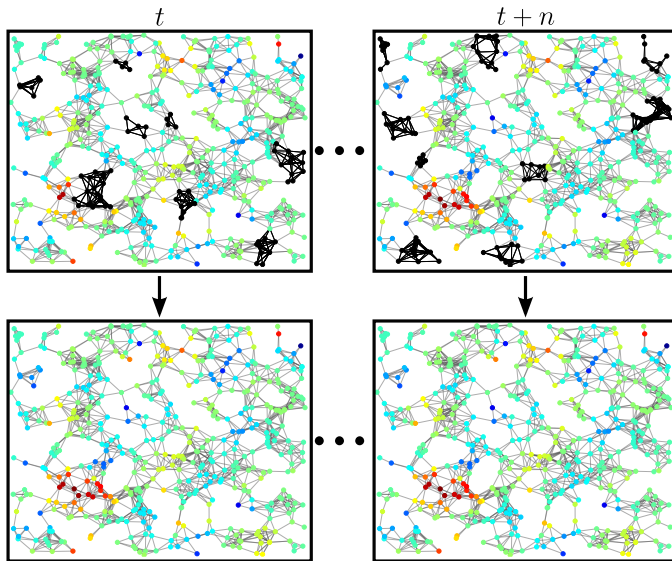
- ▶ $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M] \in \mathbb{R}^{N \times M}$ is a **time-varying graph signal**, where $\mathbf{x}_s \in \mathbb{R}^N$ is a graph signal in $\mathcal{G}$ at time s
- ▶ $\mathbf{D}_h$ is the temporal difference operator
- ▶ We obtain the temporal difference signal as $\mathbf{X}\mathbf{D}_h = [\mathbf{x}_2 - \mathbf{x}_1, \mathbf{x}_3 - \mathbf{x}_2, \dots, \mathbf{x}_M - \mathbf{x}_{M-1}]$

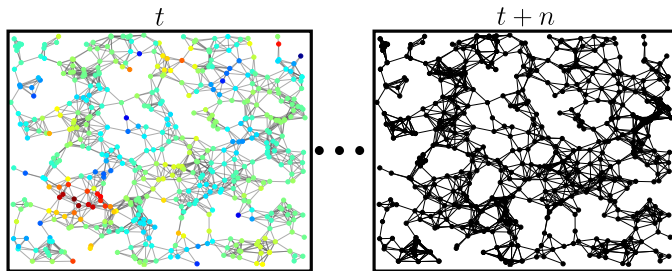
The Two Problems

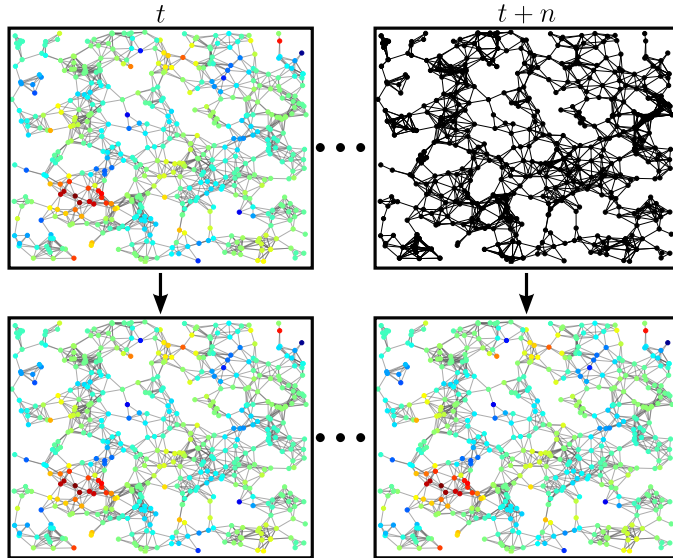
Reconstruction



Reconstruction







In this lecture, we will focus on two problems in spatiotemporal analysis with graphs: reconstruction and forecasting.

- ▶ Please discuss with your classmate next to you about possible applications for these two problems
- ▶ Please write down the one or two most important applications you see for each problem
- ▶ We'll have a small discussion about your thoughts

You have 3 minutes.

Spectral GNNs

- Graph Fourier Transform (GFT)
- Graph Filters in the Frequency Domain
- Chebyshev Graph Convolutions

Reconstruction of Time-varying Graph Signals

- Solving Reconstruction, the Optimization Way
- Solving Reconstruction, the Neural Network Way

Forecasting of Time-varying Graph Signals

- Auto-regressive Modeling
- Forecasting with GNNs

Spectral GNNs

Spectral GNNs

Why Introducing Spectral GNNs Today?

Spectral GNNs

Why Introducing Spectral GNNs Today?

- ▶ I've seen in practice that spectral GNNs outperform the regular MPNN in problems of spatiotemporal analysis

Spectral GNNs

Why Introducing Spectral GNNs Today?

- ▶ I've seen in practice that spectral GNNs outperform the regular MPNN in problems of spatiotemporal analysis
- ▶ Spectral GNNs leverage the **spectral domain of graphs**

Spectral GNNs

Why Introducing Spectral GNNs Today?

- ▶ I've seen in practice that spectral GNNs outperform the regular MPNN in problems of spatiotemporal analysis
- ▶ Spectral GNNs leverage the **spectral domain of graphs**
- ▶ We define the spectrum of a graph through their Laplacian eigenvalues and eigenvectors

Spectral GNNs

Why Introducing Spectral GNNs Today?

- ▶ I've seen in practice that spectral GNNs outperform the regular MPNN in problems of spatiotemporal analysis
- ▶ Spectral GNNs leverage the **spectral domain of graphs**
- ▶ We define the spectrum of a graph through their Laplacian eigenvalues and eigenvectors
- ▶ Spectral GNNs are rooted in graph signal processing (GSP) and operate by transforming graph signals using filters defined in the spectral domain

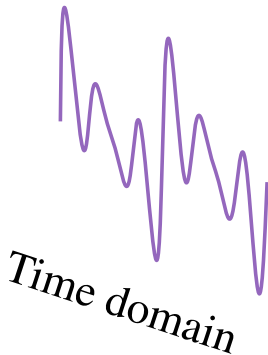
Spectral GNNs

Why Introducing Spectral GNNs Today?

- ▶ I've seen in practice that spectral GNNs outperform the regular MPNN in problems of spatiotemporal analysis
- ▶ Spectral GNNs leverage the **spectral domain of graphs**
- ▶ We define the spectrum of a graph through their Laplacian eigenvalues and eigenvectors
- ▶ Spectral GNNs are rooted in graph signal processing (GSP) and operate by transforming graph signals using filters defined in the spectral domain
- ▶ We'll see how to analyze graph signals in the spectral (**Fourier**) domain

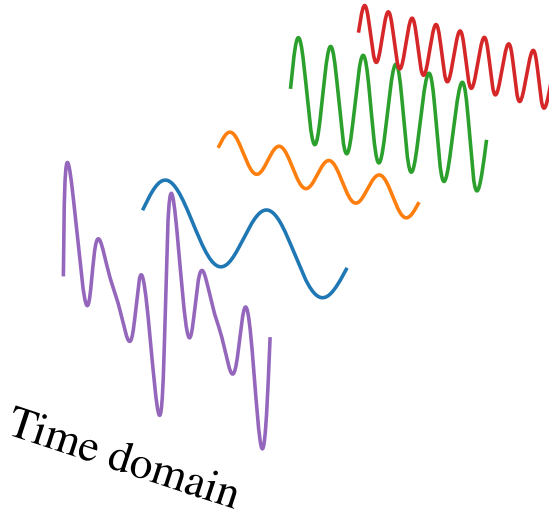
Spectral GNNs

Fourier Transform (Recap)



Spectral GNNs

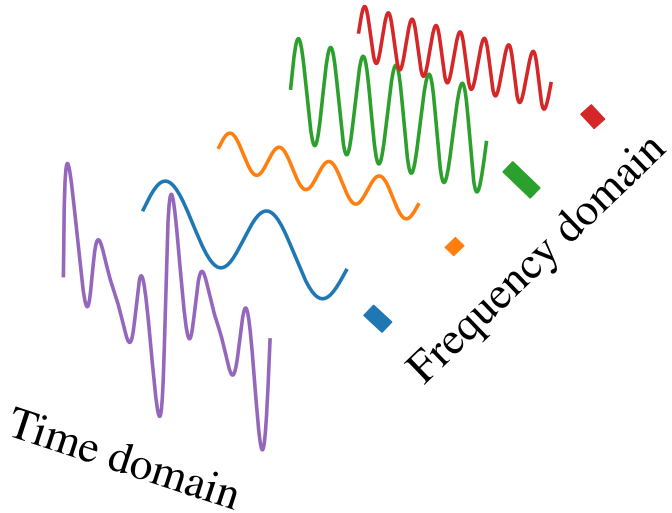
Fourier Transform (Recap)



Time domain

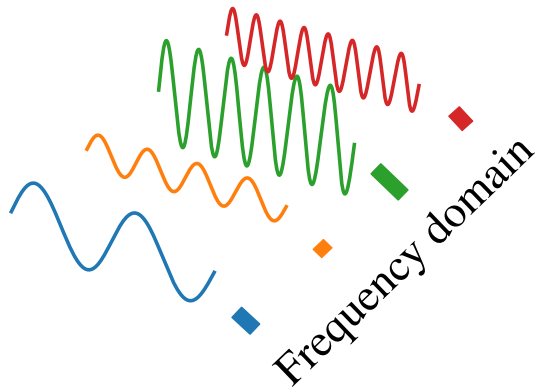
Spectral GNNs

Fourier Transform (Recap)



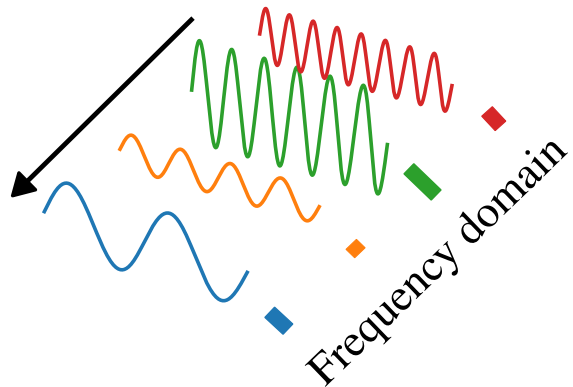
Spectral GNNs

Inverse Fourier Transform (Recap)



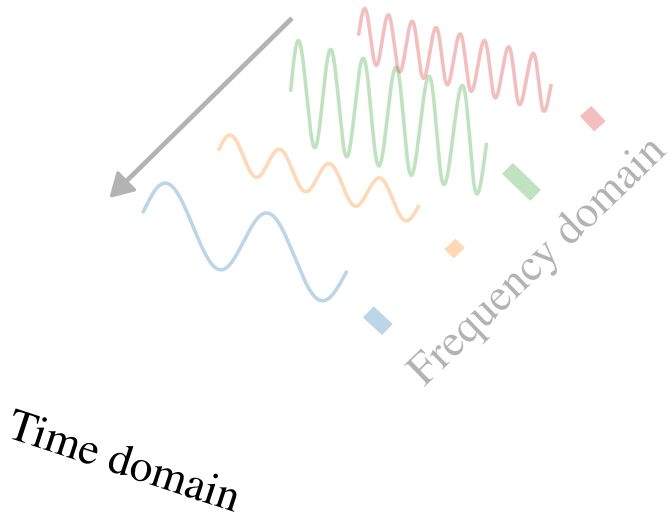
Spectral GNNs

Inverse Fourier Transform (Recap)



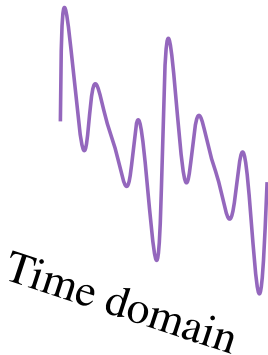
Spectral GNNs

Inverse Fourier Transform (Recap)



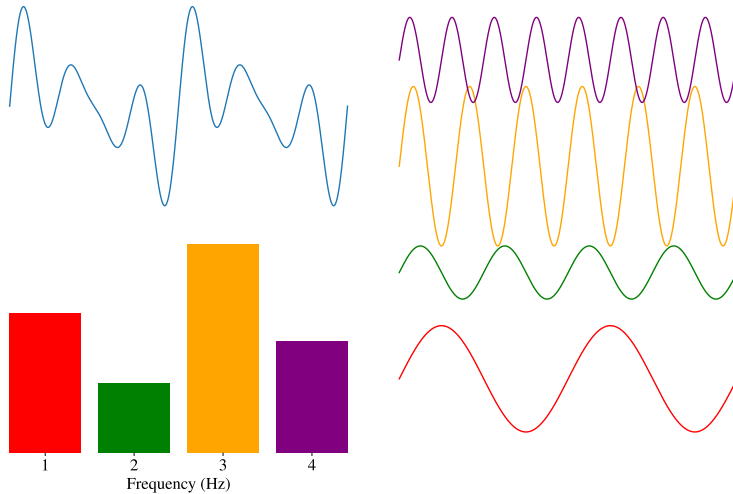
Spectral GNNs

Inverse Fourier Transform (Recap)



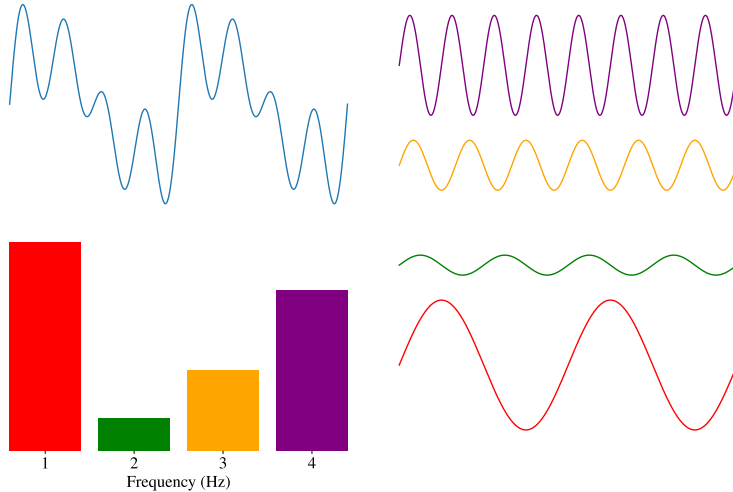
Spectral GNNs

Examples of Fourier Transform



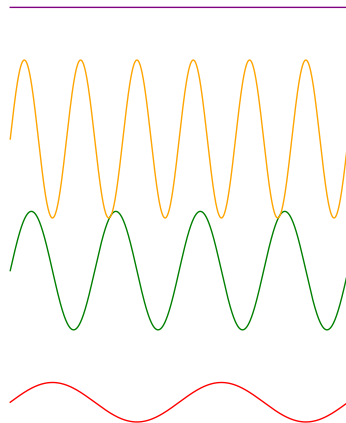
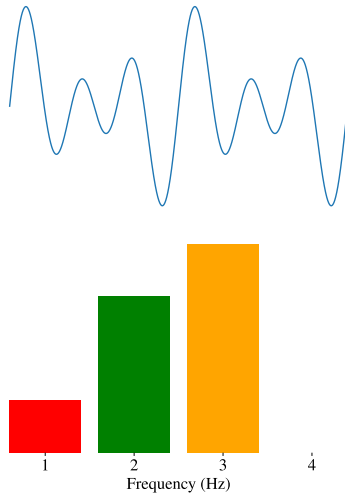
Spectral GNNs

Examples of Fourier Transform



Spectral GNNs

Examples of Fourier Transform



Spectral GNNs

Graph Fourier Transform (GFT)

- ▶ The GFT is a transformation of a graph signal $\mathbf{x}$
- ▶ We can use GFT to analyze a graph signal in the frequency domain

- ▶ The GFT is a transformation of a graph signal $\mathbf{x}$
- ▶ We can use GFT to analyze a graph signal in the frequency domain

Graph Fourier Transform Definition - Diagonalizable $\mathbf{S}$

Since $\mathbf{S}$ is diagonalizable, it has a full set of eigenvectors $\mathbf{u}_i$ that form a basis for $\mathbb{R}^N$. Therefore, we have:

$$\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1},$$

where $\mathbf{\Lambda}$ is the matrix of eigenvalues. Then we can write any $\mathbf{x}$ in terms of the basis vectors representing its graph frequencies:

$$\mathbf{x} = \mathbf{U}\hat{\mathbf{x}} \quad \text{with} \quad \hat{\mathbf{x}} = \mathbf{U}^{-1}\mathbf{x},$$

where $\hat{\mathbf{x}}$ is the GFT of $\mathbf{x}$.

Spectral GNNs

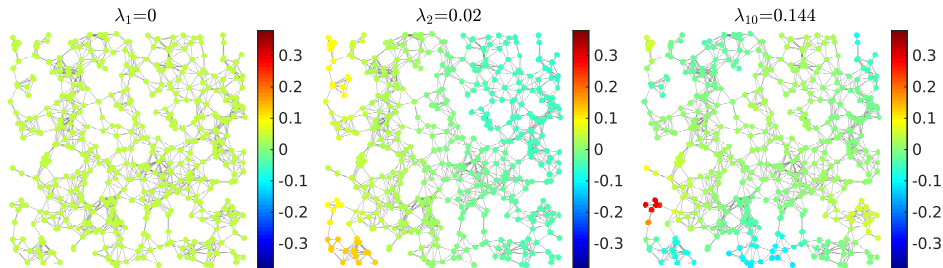
Graph Fourier Transform (GFT)

- ▶ When the graph is undirected, $\mathbf{S}$ is symmetric, and since any real symmetric matrix is guaranteed to have a full set of orthogonal eigenvectors, we have $\mathbf{U}^\top \mathbf{U} = \mathbf{U} \mathbf{U}^\top = \mathbf{I}$
- ▶ In this case, $\mathbf{x} = \mathbf{U} \hat{\mathbf{x}}$ with $\hat{\mathbf{x}} = \mathbf{U}^\top \mathbf{x}$

Spectral GNNs

Graph Fourier Transform (GFT)

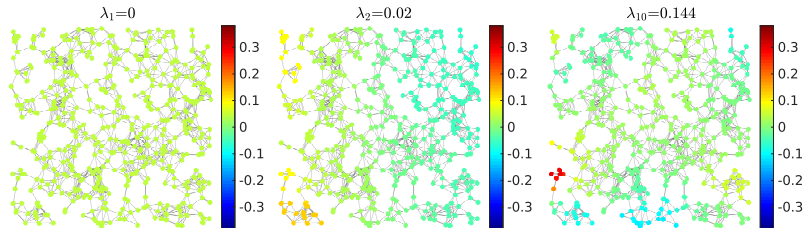
- ▶ When the graph is undirected, $\mathbf{S}$ is symmetric, and since any real symmetric matrix is guaranteed to have a full set of orthogonal eigenvectors, we have $\mathbf{U}^\top \mathbf{U} = \mathbf{U} \mathbf{U}^\top = \mathbf{I}$
- ▶ In this case, $\mathbf{x} = \mathbf{U} \hat{\mathbf{x}}$ with $\hat{\mathbf{x}} = \mathbf{U}^\top \mathbf{x}$
- ▶ Let's interpret the GFT:



Eigenvectors/eigenvalues for the Laplacian matrix of a sensor network

Spectral GNNs

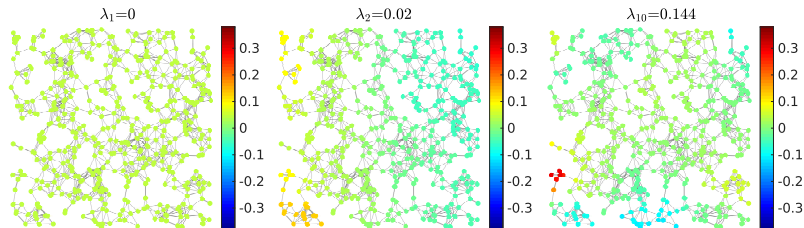
Graph Fourier Transform (GFT)



► Please notice that $\hat{\mathbf{x}}(i) = \langle \mathbf{u}_i, \mathbf{x} \rangle$

Spectral GNNs

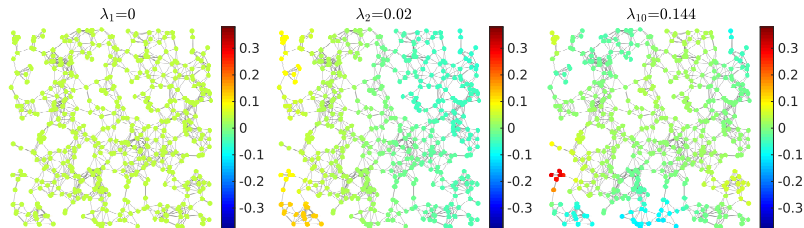
Graph Fourier Transform (GFT)



- ▶ Please notice that $\hat{\mathbf{x}}(i) = \langle \mathbf{u}_i, \mathbf{x} \rangle$
- ▶ In other words, the i th component of the GFT of $\mathbf{x}$ is measuring “**how similar**” the eigenvector $\mathbf{u}_i$ is to $\mathbf{x}$
- ▶ For example, if $\hat{\mathbf{x}}$ has more of its component in the first eigenvectors, it means the signal is quite **smooth** (low-frequency components)

Spectral GNNs

Graph Fourier Transform (GFT)



- ▶ Please notice that $\hat{\mathbf{x}}(i) = \langle \mathbf{u}_i, \mathbf{x} \rangle$
- ▶ In other words, the i th component of the GFT of $\mathbf{x}$ is measuring “**how similar**” the eigenvector $\mathbf{u}_i$ is to $\mathbf{x}$
- ▶ For example, if $\hat{\mathbf{x}}$ has more of its component in the first eigenvectors, it means the signal is quite **smooth** (low-frequency components)
- ▶ We’re stopping this discussion here, but all these concepts from GSP are quite important in machine learning with graphs. One interesting book in this domain is [Ortega(2022)]

Spectral GNNs

Graph Filters in the Frequency Domain

- ▶ We saw in the second lecture how to compute graph filters as polynomials of the shift operator $\mathbf{z} = \sum_{i=0}^k h_i \mathbf{S}^i \mathbf{x}$
- ▶ The interpretation in the Fourier domain is as follows

- ▶ We saw in the second lecture how to compute graph filters as polynomials of the shift operator $\mathbf{z} = \sum_{i=0}^k h_i \mathbf{S}^i \mathbf{x}$
- ▶ The interpretation in the Fourier domain is as follows

Graph Filters in Frequency Domain

Any graph filter $\mathbf{H}$, defined as a polynomial of $\mathbf{S}$, can be written as:

$$\mathbf{H} = h(\mathbf{S}) = \sum_{i=0}^k h_i \mathbf{S}^i = \mathbf{U} h(\mathbf{\Lambda}) \mathbf{U}^{-1}, \quad (1)$$

where $h(\mathbf{\Lambda})$ is a diagonal matrix with j th entry $h(\lambda_i) = \sum_{i=0}^k h_i \lambda_j^i$.

Proof: See [Ortega(2022)].

► From a frequency perspective, we can thus apply a graph filter as follows:

1. Compute the GFT: $\hat{\mathbf{x}} = \mathbf{U}^{-1}\mathbf{x}$
2. Apply a filter in the Fourier domain: $\hat{\mathbf{y}} = h(\mathbf{\Lambda})\hat{\mathbf{x}}$
3. Compute the inverse GFT: $\mathbf{y} = \mathbf{U}\hat{\mathbf{y}}$

- ▶ From a frequency perspective, we can thus apply a graph filter as follows:
 1. Compute the GFT: $\hat{\mathbf{x}} = \mathbf{U}^{-1}\mathbf{x}$
 2. Apply a filter in the Fourier domain: $\hat{\mathbf{y}} = h(\mathbf{\Lambda})\hat{\mathbf{x}}$
 3. Compute the inverse GFT: $\mathbf{y} = \mathbf{U}\hat{\mathbf{y}}$
- ▶ Great, now we can avoid the issue of computing powers of $\mathbf{S}$ or the recursive implementation

- ▶ From a frequency perspective, we can thus apply a graph filter as follows:
 1. Compute the GFT: $\hat{\mathbf{x}} = \mathbf{U}^{-1}\mathbf{x}$
 2. Apply a filter in the Fourier domain: $\hat{\mathbf{y}} = h(\mathbf{\Lambda})\hat{\mathbf{x}}$
 3. Compute the inverse GFT: $\mathbf{y} = \mathbf{U}\hat{\mathbf{y}}$
- ▶ Great, now we can avoid the issue of computing powers of $\mathbf{S}$ or the recursive implementation
- ▶ What's the issue with this approach?

- ▶ From a frequency perspective, we can thus apply a graph filter as follows:
 1. Compute the GFT: $\hat{\mathbf{x}} = \mathbf{U}^{-1}\mathbf{x}$
 2. Apply a filter in the Fourier domain: $\hat{\mathbf{y}} = h(\mathbf{\Lambda})\hat{\mathbf{x}}$
 3. Compute the inverse GFT: $\mathbf{y} = \mathbf{U}\hat{\mathbf{y}}$
- ▶ Great, now we can avoid the issue of computing powers of $\mathbf{S}$ or the recursive implementation
- ▶ What's the issue with this approach?
- ▶ With your classmate next to you discuss *what are the trade-offs of spectral vs. spatial GNNs*. Please discuss it for 2 minutes

- ▶ From a frequency perspective, we can thus apply a graph filter as follows:
 1. Compute the GFT: $\hat{\mathbf{x}} = \mathbf{U}^{-1}\mathbf{x}$
 2. Apply a filter in the Fourier domain: $\hat{\mathbf{y}} = h(\mathbf{\Lambda})\hat{\mathbf{x}}$
 3. Compute the inverse GFT: $\mathbf{y} = \mathbf{U}\hat{\mathbf{y}}$
- ▶ Great, now we can avoid the issue of computing powers of $\mathbf{S}$ or the recursive implementation
- ▶ What's the issue with this approach?
- ▶ With your classmate next to you discuss *what are the trade-offs of spectral vs. spatial GNNs*. Please discuss it for 2 minutes
- ▶ The eigenvalue decomposition typically has complexity $\mathcal{O}(N^3)$

- ▶ From a frequency perspective, we can thus apply a graph filter as follows:
 1. Compute the GFT: $\hat{\mathbf{x}} = \mathbf{U}^{-1}\mathbf{x}$
 2. Apply a filter in the Fourier domain: $\hat{\mathbf{y}} = h(\mathbf{\Lambda})\hat{\mathbf{x}}$
 3. Compute the inverse GFT: $\mathbf{y} = \mathbf{U}\hat{\mathbf{y}}$
- ▶ Great, now we can avoid the issue of computing powers of $\mathbf{S}$ or the recursive implementation
- ▶ What's the issue with this approach?
- ▶ With your classmate next to you discuss *what are the trade-offs of spectral vs. spatial GNNs*. Please discuss it for 2 minutes
- ▶ The eigenvalue decomposition typically has complexity $\mathcal{O}(N^3)$
- ▶ You can work with this tool if your graphs are small

Spectral GNNs

Chebyshev Graph Convolutions

- ▶ Depending on the size of your graph, you might want to choose using the spectrum of the graph or the polynomial of $\mathbf{S}$ to implement spectral GNNs
- ▶ The most famous spectral GNN in the literature, called Chebyshev graph convolutions (ChebConv), was proposed by [Defferrard et al.(2016)]
- ▶ ChebConv is the predecessor of GCN, so it's quite important in our field
- ▶ Some people in the graph learning community continued working on spectral GNNs, but this is arguably a less explored topic than MPNN (it's very active though)

Spectral GNNs

Chebyshev Graph Convolutions

- We can define a ChebConv layer as follows:

$$\mathbf{H}^{(l+1)} = \sum_{i=1}^k \mathbf{Z}_i \mathbf{W}_i, \quad (2)$$

where $\mathbf{W}_i$ is a linear transformation, $\mathbf{Z}_1 = \mathbf{H}^{(l)}$, $\mathbf{Z}_2 = \tilde{\mathbf{L}}\mathbf{H}^{(l)}$, and $\mathbf{Z}_i = 2\tilde{\mathbf{L}}\mathbf{Z}_{i-1} - \mathbf{Z}_{i-2}$, with $\tilde{\mathbf{L}} = \frac{2\mathbf{L}}{\lambda_N} - \mathbf{I}$

Spectral GNNs

Chebyshev Graph Convolutions

- We can define a ChebConv layer as follows:

$$\mathbf{H}^{(l+1)} = \sum_{i=1}^k \mathbf{Z}_i \mathbf{W}_i, \quad (2)$$

where $\mathbf{W}_i$ is a linear transformation, $\mathbf{Z}_1 = \mathbf{H}^{(l)}$, $\mathbf{Z}_2 = \tilde{\mathbf{L}}\mathbf{H}^{(l)}$, and $\mathbf{Z}_i = 2\tilde{\mathbf{L}}\mathbf{Z}_{i-1} - \mathbf{Z}_{i-2}$, with $\tilde{\mathbf{L}} = \frac{2\mathbf{L}}{\lambda_N} - \mathbf{I}$

- This recursion has been derived using the Chebyshev polynomials, and that's why this model is called ChebConv¹

¹ If you're interested on the details, please see [Defferrard et al.(2016)].

Spectral GNNs

Chebyshev Graph Convolutions

- We can define a ChebConv layer as follows:

$$\mathbf{H}^{(l+1)} = \sum_{i=1}^k \mathbf{Z}_i \mathbf{W}_i, \quad (2)$$

where $\mathbf{W}_i$ is a linear transformation, $\mathbf{Z}_1 = \mathbf{H}^{(l)}$, $\mathbf{Z}_2 = \tilde{\mathbf{L}}\mathbf{H}^{(l)}$, and $\mathbf{Z}_i = 2\tilde{\mathbf{L}}\mathbf{Z}_{i-1} - \mathbf{Z}_{i-2}$, with $\tilde{\mathbf{L}} = \frac{2\mathbf{L}}{\lambda_N} - \mathbf{I}$

- This recursion has been derived using the Chebyshev polynomials, and that's why this model is called ChebConv¹
- If you change the Chebyshev by other family of polynomials, you can get other methods like GegenConv (Gegenbauer polynomials) [Castro-Correa et al.(2024)] or JacobiConv [Wang and Zhang(2022)] (Jacobi polynomials)

¹ If you're interested on the details, please see [Defferrard et al.(2016)].

Spectral GNNs

Summary of Spectral GNNs

- ▶ The GFT is an important tool for analyzing graph signals in the **frequency domain**

- ▶ The GFT is an important tool for analyzing graph signals in the **frequency domain**
- ▶ Spectral GNNs operate in the spectral domain of graphs, leveraging the graph Laplacian's spectrum

- ▶ The GFT is an important tool for analyzing graph signals in the **frequency domain**
- ▶ Spectral GNNs operate in the spectral domain of graphs, leveraging the graph Laplacian's spectrum
- ▶ Working directly on the spectrum could be computationally expensive because the eigenvalue decomposition is required

Spectral GNNs

Summary of Spectral GNNs

- ▶ The GFT is an important tool for analyzing graph signals in the **frequency domain**
- ▶ Spectral GNNs operate in the spectral domain of graphs, leveraging the graph Laplacian's spectrum
- ▶ Working directly on the spectrum could be computationally expensive because the eigenvalue decomposition is required
- ▶ ChebConv is a key spectral GNN model that approximates graph convolutions using Chebyshev polynomials

Reconstruction of Time-varying Graph Signals

Reconstruction of Time-varying Graph Signals

Formal problem definition for reconstruction

Problem Definition in Reconstruction

Given a graph G with an associated time-varying graph signal $\mathbf{X} \in \mathbb{R}^{N \times M}$, we have some missing elements in $\mathbf{X}$ at some positions $i \in \mathcal{V}$ and some instant of time $j \in \mathcal{M}$, where $\mathcal{M} = \{1, 2, \dots, M\}$. In other words, we have a **known** set of missing elements $\mathcal{S} = \{(i, j) \mid i \in \mathcal{V}, j \in \mathcal{M}\}$. Given the set $\mathcal{T} = (\mathcal{V} \times \mathcal{M}) - \mathcal{S}$, i.e., the set of known (not missing) graph signals, we want to reconstruct the graph signals in $\mathcal{S}$.

Solving Reconstruction, the Optimization Way

Reconstruction of Time-varying Graph Signals

Solving Reconstruction, the Optimization Way (Roadmap)

- We define a measure of **smoothness** on time-varying graphs

Reconstruction of Time-varying Graph Signals

Solving Reconstruction, the Optimization Way (Roadmap)

- ▶ We define a measure of **smoothness** on time-varying graphs
- ▶ We design an optimization problem with an **error loss** regularized with the smoothness function

Reconstruction of Time-varying Graph Signals

Solving Reconstruction, the Optimization Way (Roadmap)

- ▶ We define a measure of **smoothness** on time-varying graphs
- ▶ We design an optimization problem with an **error loss** regularized with the smoothness function
- ▶ We solve the optimization problem with some **gradient descent** algorithm

Reconstruction of Time-varying Graph Signals

Solving Reconstruction, the Optimization Way (Roadmap)

- ▶ We define a measure of **smoothness** on time-varying graphs
- ▶ We design an optimization problem with an **error loss** regularized with the smoothness function
- ▶ We solve the optimization problem with some **gradient descent** algorithm
- ▶ These ideas come from [Qiu et al.(2017), Giraldo et al.(2022)]

Reconstruction of Time-varying Graph Signals

Measuring Smoothness in Graph Signals

- ▶ A common function to measure how smooth a signal is in the graph is called the **quadratic forms**

Reconstruction of Time-varying Graph Signals

Measuring Smoothness in Graph Signals

- ▶ A common function to measure how smooth a signal is in the graph is called the **quadratic forms**
- ▶ The so-called Laplacian quadratic form is widely used in machine learning with graphs

Reconstruction of Time-varying Graph Signals

Measuring Smoothness in Graph Signals

- ▶ A common function to measure how smooth a signal is in the graph is called the **quadratic forms**
- ▶ The so-called Laplacian quadratic form is widely used in machine learning with graphs

Laplacian Quadratic Form

The Laplacian quadratic form quantifies the **total graph signal variation**:

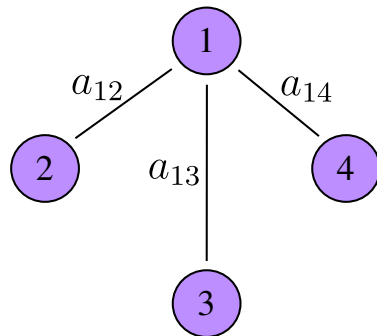
$$\Delta_{\mathbf{L}}(\mathbf{x}) = \mathbf{x}^{\top} \mathbf{L} \mathbf{x} = \sum_{i \sim j} a_{ij} (x_i - x_j)^2, \quad (3)$$

where the summation is over all edges. Each edge contributes to the variation as a function of its weight, a_{ij} , and of the difference between the signal values at its end-points, $(x_i - x_j)^2$.

Reconstruction of Time-varying Graph Signals

Measuring Smoothness in Graph Signals

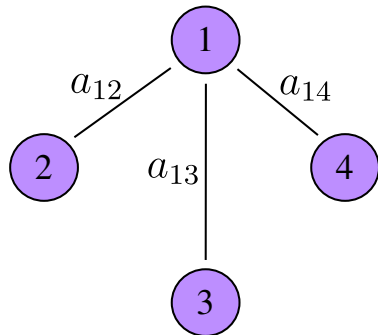
- ▶ Let's compute the Laplacian quadratic form for the graph in the figure, where $\mathbf{x} = [x_1, x_2, x_3, x_4]^\top$ and $\mathbf{x}^\top \mathbf{L} \mathbf{x} = \sum_{i \sim j} a_{ij} (x_i - x_j)^2$. You have 2 minutes



Reconstruction of Time-varying Graph Signals

Measuring Smoothness in Graph Signals

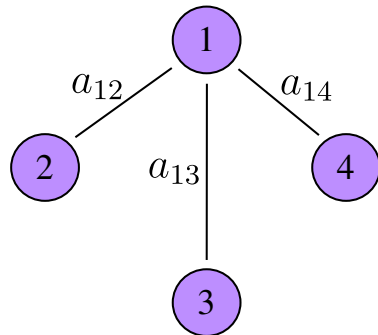
- ▶ Let's compute the Laplacian quadratic form for the graph in the figure, where $\mathbf{x} = [x_1, x_2, x_3, x_4]^\top$ and $\mathbf{x}^\top \mathbf{L} \mathbf{x} = \sum_{i \sim j} a_{ij} (x_i - x_j)^2$. You have 2 minutes
- ▶ $\mathbf{x}^\top \mathbf{L} \mathbf{x} = a_{12}(x_1 - x_2)^2 + a_{13}(x_1 - x_3)^2 + a_{14}(x_1 - x_4)^2$



Reconstruction of Time-varying Graph Signals

Measuring Smoothness in Graph Signals

- ▶ Let's compute the Laplacian quadratic form for the graph in the figure, where $\mathbf{x} = [x_1, x_2, x_3, x_4]^\top$ and $\mathbf{x}^\top \mathbf{L} \mathbf{x} = \sum_{i \sim j} a_{ij} (x_i - x_j)^2$. You have 2 minutes
- ▶ $\mathbf{x}^\top \mathbf{L} \mathbf{x} = a_{12}(x_1 - x_2)^2 + a_{13}(x_1 - x_3)^2 + a_{14}(x_1 - x_4)^2$
- ▶ We see that a signal with different values in two nodes contributes to the variation only if the corresponding nodes are connected (if $x_3 \neq x_4$ this has no direct effect on variation)



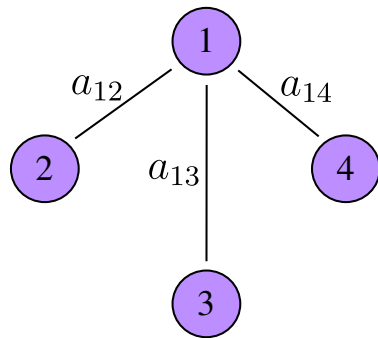
Reconstruction of Time-varying Graph Signals

Measuring Smoothness in Graph Signals

- As a low-variation signal, we can choose

$$x_1 = x_2 = x_3 = x_4 = 1, \text{ then } \mathbf{x}^\top \mathbf{L} \mathbf{x} =$$

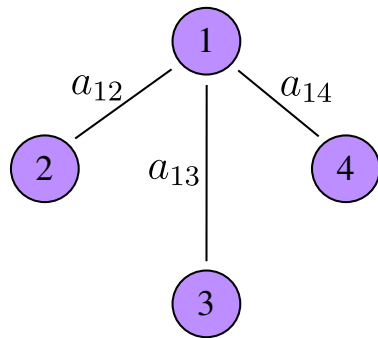
$$a_{12}(x_1 - x_2)^2 + a_{13}(x_1 - x_3)^2 + a_{14}(x_1 - x_4)^2 = 0$$



Reconstruction of Time-varying Graph Signals

Measuring Smoothness in Graph Signals

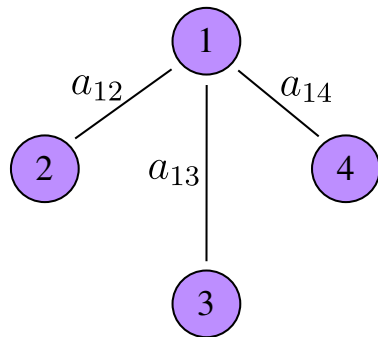
- ▶ As a low-variation signal, we can choose $x_1 = x_2 = x_3 = x_4 = 1$, then $\mathbf{x}^\top \mathbf{L} \mathbf{x} = a_{12}(x_1 - x_2)^2 + a_{13}(x_1 - x_3)^2 + a_{14}(x_1 - x_4)^2 = 0$
- ▶ Since all the terms are positive (because the weights $a_{ij} \geq 0$), this has to be the minimal possible variation



Reconstruction of Time-varying Graph Signals

Measuring Smoothness in Graph Signals

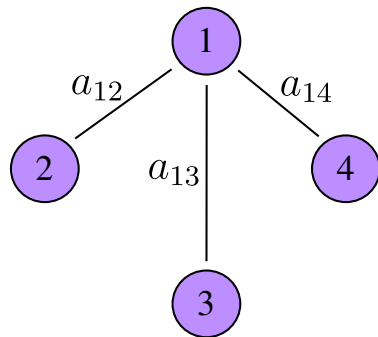
- ▶ As a low-variation signal, we can choose $x_1 = x_2 = x_3 = x_4 = 1$, then $\mathbf{x}^\top \mathbf{L} \mathbf{x} = a_{12}(x_1 - x_2)^2 + a_{13}(x_1 - x_3)^2 + a_{14}(x_1 - x_4)^2 = 0$
- ▶ Since all the terms are positive (because the weights $a_{ij} \geq 0$), this has to be the minimal possible variation
- ▶ For higher variation, we can choose signals with values that are different across edges in the graph.



Reconstruction of Time-varying Graph Signals

Measuring Smoothness in Graph Signals

- ▶ As a low-variation signal, we can choose $x_1 = x_2 = x_3 = x_4 = 1$, then $\mathbf{x}^\top \mathbf{L} \mathbf{x} = a_{12}(x_1 - x_2)^2 + a_{13}(x_1 - x_3)^2 + a_{14}(x_1 - x_4)^2 = 0$
- ▶ Since all the terms are positive (because the weights $a_{ij} \geq 0$), this has to be the minimal possible variation
- ▶ For higher variation, we can choose signals with values that are different across edges in the graph.
- ▶ For example, we can choose a signal that maximizes the number of zero crossings (changes in sign between the signal values at the ends of an edge) like $x_1 = 1$ and $x_2 = x_3 = x_4 = -1$, thus $\mathbf{x}^\top \mathbf{L} \mathbf{x} = 4(a_{12} + a_{13} + a_{14})$



Reconstruction of Time-varying Graph Signals

How to Measure Smoothness in Time-varying Graph Signals

- We can measure the smoothness of a time-varying graph signal as

$$E(\mathbf{X}) = \sum_{s=1}^M \mathbf{x}_s^T \mathbf{L} \mathbf{x}_s = \text{tr}(\mathbf{X}^T \mathbf{L} \mathbf{X})$$

Reconstruction of Time-varying Graph Signals

How to Measure Smoothness in Time-varying Graph Signals

- ▶ We can measure the smoothness of a time-varying graph signal as $E(\mathbf{X}) = \sum_{s=1}^M \mathbf{x}_s^T \mathbf{L} \mathbf{x}_s = \text{tr}(\mathbf{X}^T \mathbf{L} \mathbf{X})$
- ▶ Qiu *et al.* (2017) found that in real-world data $\mathbf{X} \mathbf{D}_h$ exhibits better smoothness properties than $\mathbf{X}$ alone

Reconstruction of Time-varying Graph Signals

How to Measure Smoothness in Time-varying Graph Signals

- ▶ We can measure the smoothness of a time-varying graph signal as $E(\mathbf{X}) = \sum_{s=1}^M \mathbf{x}_s^\top \mathbf{L} \mathbf{x}_s = \text{tr}(\mathbf{X}^\top \mathbf{L} \mathbf{X})$
- ▶ Qiu *et al.* (2017) found that in real-world data $\mathbf{X} \mathbf{D}_h$ exhibits better smoothness properties than $\mathbf{X}$ alone
- ▶ This means that we have $E(\mathbf{X} \mathbf{D}_h) = \sum_{s=2}^M (\mathbf{x}_s - \mathbf{x}_{s-1})^\top \mathbf{L} (\mathbf{x}_s - \mathbf{x}_{s-1})$

Reconstruction of Time-varying Graph Signals

How to Measure Smoothness in Time-varying Graph Signals

- ▶ We can measure the smoothness of a time-varying graph signal as $E(\mathbf{X}) = \sum_{s=1}^M \mathbf{x}_s^\top \mathbf{L} \mathbf{x}_s = \text{tr}(\mathbf{X}^\top \mathbf{L} \mathbf{X})$
- ▶ Qiu *et al.* (2017) found that in real-world data $\mathbf{X} \mathbf{D}_h$ exhibits better smoothness properties than $\mathbf{X}$ alone
- ▶ This means that we have $E(\mathbf{X} \mathbf{D}_h) = \sum_{s=2}^M (\mathbf{x}_s - \mathbf{x}_{s-1})^\top \mathbf{L} (\mathbf{x}_s - \mathbf{x}_{s-1})$
- ▶ Intuitively, we're measuring the **smoothness in the difference signals** and not on each static signal itself. When we have a high sampling rate, $\mathbf{x}_s - \mathbf{x}_{s-1}$ should be close to zero (typically, things don't change abruptly in nature)

Reconstruction of Time-varying Graph Signals

How to Measure Smoothness in Time-varying Graph Signals

- ▶ We can measure the smoothness of a time-varying graph signal as $E(\mathbf{X}) = \sum_{s=1}^M \mathbf{x}_s^T \mathbf{L} \mathbf{x}_s = \text{tr}(\mathbf{X}^T \mathbf{L} \mathbf{X})$
- ▶ Qiu *et al.* (2017) found that in real-world data $\mathbf{X} \mathbf{D}_h$ exhibits better smoothness properties than $\mathbf{X}$ alone
- ▶ This means that we have $E(\mathbf{X} \mathbf{D}_h) = \sum_{s=2}^M (\mathbf{x}_s - \mathbf{x}_{s-1})^T \mathbf{L} (\mathbf{x}_s - \mathbf{x}_{s-1})$
- ▶ Intuitively, we're measuring the **smoothness in the difference signals** and not on each static signal itself. When we have a high sampling rate, $\mathbf{x}_s - \mathbf{x}_{s-1}$ should be close to zero (typically, things don't change abruptly in nature)
- ▶ Therefore, this difference ($\mathbf{X} \mathbf{D}_h$) has better smoothness properties than $\mathbf{X}$ alone

Reconstruction of Time-varying Graph Signals

Why Do We Want to Use this Smoothness Function?

$$\text{tr} \left((\tilde{\mathbf{X}}\mathbf{D}_h)^\top \mathbf{L} \tilde{\mathbf{X}}\mathbf{D}_h \right).$$

Reconstruction of Time-varying Graph Signals

Why Do We Want to Use this Smoothness Function?

$$\text{tr} \left((\tilde{\mathbf{X}}\mathbf{D}_h)^\top \mathbf{L} \tilde{\mathbf{X}}\mathbf{D}_h \right).$$

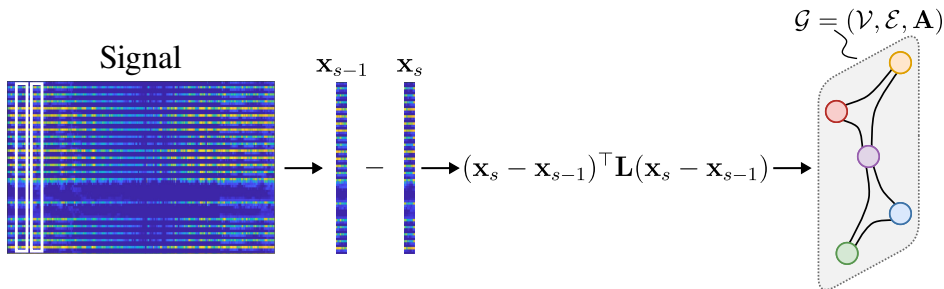
- We use the smoothness term $\text{tr} \left((\tilde{\mathbf{X}}\mathbf{D}_h)^\top \mathbf{L} \tilde{\mathbf{X}}\mathbf{D}_h \right)$ in an optimization function because it encourages the difference time signals to be smooth in the graph

Reconstruction of Time-varying Graph Signals

Why Do We Want to Use this Smoothness Function?

$$\text{tr} \left((\tilde{\mathbf{X}} \mathbf{D}_h)^\top \mathbf{L} \tilde{\mathbf{X}} \mathbf{D}_h \right).$$

- We use the smoothness term $\text{tr} \left((\tilde{\mathbf{X}} \mathbf{D}_h)^\top \mathbf{L} \tilde{\mathbf{X}} \mathbf{D}_h \right)$ in an optimization function because it encourages the difference time signals to be smooth in the graph



Reconstruction of Time-varying Graph Signals

Defining the Optimization Problem

- Now, if we want to define an optimization problem for reconstructing time-varying graph signals, we should define a **measure of error**, and we can regularize it with our smoothness function

Reconstruction of Time-varying Graph Signals

Defining the Optimization Problem

- ▶ Now, if we want to define an optimization problem for reconstructing time-varying graph signals, we should define a **measure of error**, and we can regularize it with our smoothness function
- ▶ A possible measure of error is given by $\|\mathbf{J} \circ \tilde{\mathbf{X}} - \mathbf{Y}\|_F^2$, where $\mathbf{J} \in \{0, 1\}^{N \times M}$ denotes the **sampling matrix**, $\tilde{\mathbf{X}}$ is the matrix to be reconstructed, and $\mathbf{Y}$ is the matrix with the values we know

Reconstruction of Time-varying Graph Signals

Defining the Optimization Problem

- ▶ Now, if we want to define an optimization problem for reconstructing time-varying graph signals, we should define a **measure of error**, and we can regularize it with our smoothness function
- ▶ A possible measure of error is given by $\|\mathbf{J} \circ \tilde{\mathbf{X}} - \mathbf{Y}\|_F^2$, where $\mathbf{J} \in \{0, 1\}^{N \times M}$ denotes the **sampling matrix**, $\tilde{\mathbf{X}}$ is the matrix to be reconstructed, and $\mathbf{Y}$ is the matrix with the values we know
- ▶ $\mathbf{J}_{ij}$ is equal to 1 if we know the value of the signal at node i and time j , this means that for using this method we should know where and when we have missing values

Reconstruction of Time-varying Graph Signals

Defining the Optimization Problem

- ▶ Now, if we want to define an optimization problem for reconstructing time-varying graph signals, we should define a **measure of error**, and we can regularize it with our smoothness function
- ▶ A possible measure of error is given by $\|\mathbf{J} \circ \tilde{\mathbf{X}} - \mathbf{Y}\|_F^2$, where $\mathbf{J} \in \{0, 1\}^{N \times M}$ denotes the **sampling matrix**, $\tilde{\mathbf{X}}$ is the matrix to be reconstructed, and $\mathbf{Y}$ is the matrix with the values we know
- ▶ $\mathbf{J}_{ij}$ is equal to 1 if we know the value of the signal at node i and time j , this means that for using this method we should know where and when we have missing values
- ▶ Qiu *et al.* (2017) proposed a time-varying graph signal reconstruction (TGSR) method as follows:

$$\min_{\tilde{\mathbf{X}}} \frac{1}{2} \|\mathbf{J} \circ \tilde{\mathbf{X}} - \mathbf{Y}\|_F^2 + \frac{\nu}{2} \text{tr} \left((\tilde{\mathbf{X}} \mathbf{D}_h)^\top \mathbf{L} \tilde{\mathbf{X}} \mathbf{D}_h \right).$$

Reconstruction of Time-varying Graph Signals

Why Do We Want to Minimize the Error Function?

$$\frac{1}{2} \|\mathbf{J} \circ \tilde{\mathbf{X}} - \mathbf{Y}\|_F^2.$$

Reconstruction of Time-varying Graph Signals

Why Do We Want to Minimize the Error Function?

$$\frac{1}{2} \|\mathbf{J} \circ \tilde{\mathbf{X}} - \mathbf{Y}\|_F^2.$$

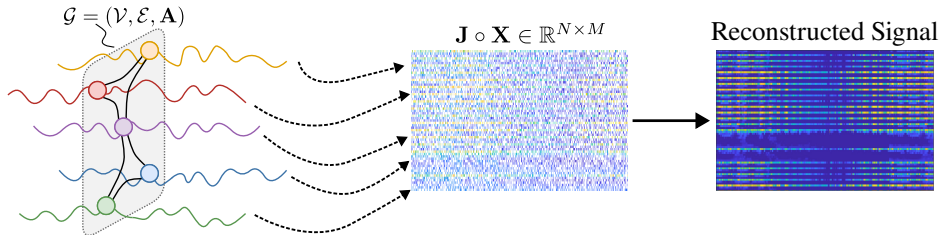
- ▶ Minimizing the first term $\|\mathbf{J} \circ \tilde{\mathbf{X}} - \mathbf{Y}\|_F^2$ encourages the solution $\tilde{\mathbf{X}}$ to be as close as possible in the know values (the positions where $\mathbf{J}$ is equal to 1) to the non-missing information $\mathbf{Y}$

Reconstruction of Time-varying Graph Signals

Why Do We Want to Minimize the Error Function?

$$\frac{1}{2} \|\mathbf{J} \circ \tilde{\mathbf{X}} - \mathbf{Y}\|_F^2.$$

- Minimizing the first term $\|\mathbf{J} \circ \tilde{\mathbf{X}} - \mathbf{Y}\|_F^2$ encourages the solution $\tilde{\mathbf{X}}$ to be as close as possible in the know values (the positions where $\mathbf{J}$ is equal to 1) to the non-missing information $\mathbf{Y}$



Reconstruction of Time-varying Graph Signals Pipeline

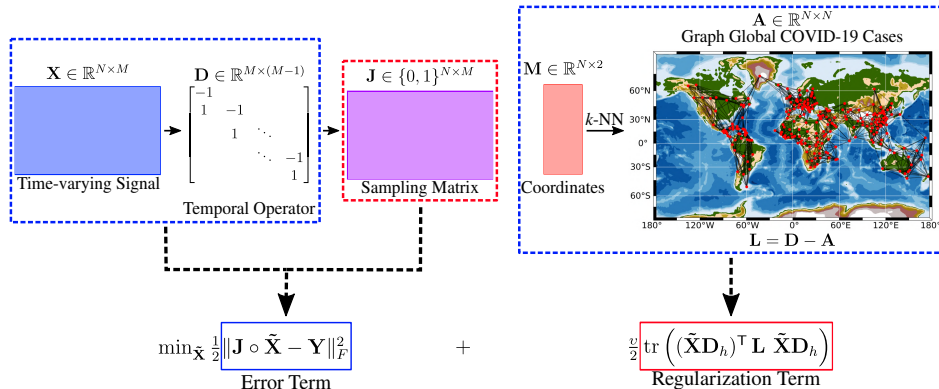


Figure: Source: Modified from [Giraldo et al.(2022)]

Reconstruction of Time-varying Graph Signals

Solving the Optimization Problem

- ▶ I'm gonna give you only the intuitions on how to solve the optimization problem

Reconstruction of Time-varying Graph Signals

Solving the Optimization Problem

- ▶ I'm gonna give you only the intuitions on how to solve the optimization problem
- ▶ Indeed, this optimization problem has a closed-form solution. However, this solution involves computing the inverse of a matrix of size $MN \times MN$... Not a good idea

Reconstruction of Time-varying Graph Signals

Solving the Optimization Problem

- ▶ I'm gonna give you only the intuitions on how to solve the optimization problem
- ▶ Indeed, this optimization problem has a closed-form solution. However, this solution involves computing the inverse of a matrix of size $MN \times MN$... Not a good idea
- ▶ We can solve this problem with some **gradient descent** method

Reconstruction of Time-varying Graph Signals

Solving the Optimization Problem

- ▶ I'm gonna give you only the intuitions on how to solve the optimization problem
- ▶ Indeed, this optimization problem has a closed-form solution. However, this solution involves computing the inverse of a matrix of size $MN \times MN$... Not a good idea
- ▶ We can solve this problem with some **gradient descent** method
- ▶ Let $f_u(\tilde{\mathbf{X}}) = \frac{1}{2} \|\mathbf{J} \circ \tilde{\mathbf{X}} - \mathbf{Y}\|_F^2 + \frac{v}{2} \text{tr} \left((\tilde{\mathbf{X}} \mathbf{D}_h)^\top \mathbf{L} \tilde{\mathbf{X}} \mathbf{D}_h \right)$ then its gradient is given by: $\nabla_{\tilde{\mathbf{X}}} f_u(\tilde{\mathbf{X}}) = \mathbf{J} \circ \tilde{\mathbf{X}} - \mathbf{Y} + v \mathbf{L} \tilde{\mathbf{X}} \mathbf{D}_h \mathbf{D}_h^\top$

Reconstruction of Time-varying Graph Signals

Solving the Optimization Problem

- ▶ I'm gonna give you only the intuitions on how to solve the optimization problem
- ▶ Indeed, this optimization problem has a closed-form solution. However, this solution involves computing the inverse of a matrix of size $MN \times MN$... Not a good idea
- ▶ We can solve this problem with some **gradient descent** method
- ▶ Let $f_u(\tilde{\mathbf{X}}) = \frac{1}{2} \|\mathbf{J} \circ \tilde{\mathbf{X}} - \mathbf{Y}\|_F^2 + \frac{v}{2} \text{tr} \left((\tilde{\mathbf{X}} \mathbf{D}_h)^\top \mathbf{L} \tilde{\mathbf{X}} \mathbf{D}_h \right)$ then its gradient is given by: $\nabla_{\tilde{\mathbf{X}}} f_u(\tilde{\mathbf{X}}) = \mathbf{J} \circ \tilde{\mathbf{X}} - \mathbf{Y} + v \mathbf{L} \tilde{\mathbf{X}} \mathbf{D}_h \mathbf{D}_h^\top$
- ▶ We can then use gradient descent or some of its derivations. In the original paper, the authors proposed to use a conjugate gradient descent method

Reconstruction of Time-varying Graph Signals

Making the Problem Less Ill-posed

- In 2022, we showed [Giraldo et al.(2022)] that the solution by Qiu *et al.* (2017) was somehow ill-posed

Reconstruction of Time-varying Graph Signals

Making the Problem Less Ill-posed

- ▶ In 2022, we showed [Giraldo et al.(2022)] that the solution by Qiu *et al.* (2017) was somehow ill-posed
- ▶ Therefore, the problem we saw before has a slow convergence

Reconstruction of Time-varying Graph Signals

Making the Problem Less Ill-posed

- ▶ In 2022, we showed [Giraldo et al.(2022)] that the solution by Qiu *et al.* (2017) was somehow ill-posed
- ▶ Therefore, the problem we saw before has a slow convergence
- ▶ Intuitively, better condition numbers are associated with faster convergence rates in gradient descent methods

Reconstruction of Time-varying Graph Signals

Making the Problem Less Ill-posed

- The condition number $\kappa(\mathbf{L})$ associated with the square matrix $\mathbf{L}$ is a measure of how well- or ill-conditioned the inversion of $\mathbf{L}$ is

Reconstruction of Time-varying Graph Signals

Making the Problem Less Ill-posed

- ▶ The condition number $\kappa(\mathbf{L})$ associated with the square matrix $\mathbf{L}$ is a measure of how well- or ill-conditioned the inversion of $\mathbf{L}$ is
- ▶ When relying on the Laplacian matrix, we have that
$$\kappa(\mathbf{L}) = \frac{|\lambda_{\max}(\mathbf{L})|}{|\lambda_{\min}(\mathbf{L})|} \rightarrow \infty \text{ since } \lambda_{\min}(\mathbf{L}) = 0$$

Reconstruction of Time-varying Graph Signals

Making the Problem Less Ill-posed

- ▶ The condition number $\kappa(\mathbf{L})$ associated with the square matrix $\mathbf{L}$ is a measure of how well- or ill-conditioned the inversion of $\mathbf{L}$ is
- ▶ When relying on the Laplacian matrix, we have that
$$\kappa(\mathbf{L}) = \frac{|\lambda_{\max}(\mathbf{L})|}{|\lambda_{\min}(\mathbf{L})|} \rightarrow \infty \text{ since } \lambda_{\min}(\mathbf{L}) = 0$$
- ▶ We then proposed to “regularize” the regularization as follows
$$\min_{\tilde{\mathbf{X}}} \frac{1}{2} \|\mathbf{J} \circ \tilde{\mathbf{X}} - \mathbf{Y}\|_F^2 + \frac{\nu}{2} \text{tr} \left((\tilde{\mathbf{X}} \mathbf{D}_h)^\top (\mathbf{L} + \epsilon \mathbf{I})^\beta \tilde{\mathbf{X}} \mathbf{D}_h \right),$$
 where we add a positive definite matrix $\epsilon \mathbf{I}$ to the Laplacian
- ▶ This new regularization was called **Sobolev smoothness** because of its relationship with the Sobolev norm in static graph signals

Reconstruction of Time-varying Graph Signals

Making the Problem Less Ill-posed

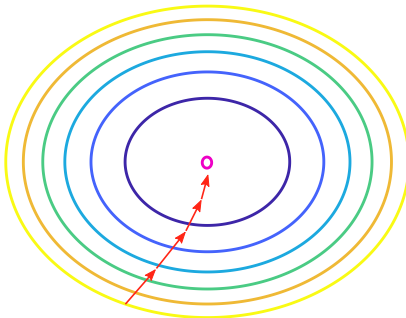
- We're not going into the mathematical details, but we showed this optimization problem converges faster than the previous method and has robust performance

Reconstruction of Time-varying Graph Signals

Making the Problem Less Ill-posed

- We're not going into the mathematical details, but we showed this optimization problem converges faster than the previous method and has robust performance

Well-conditioned Problem

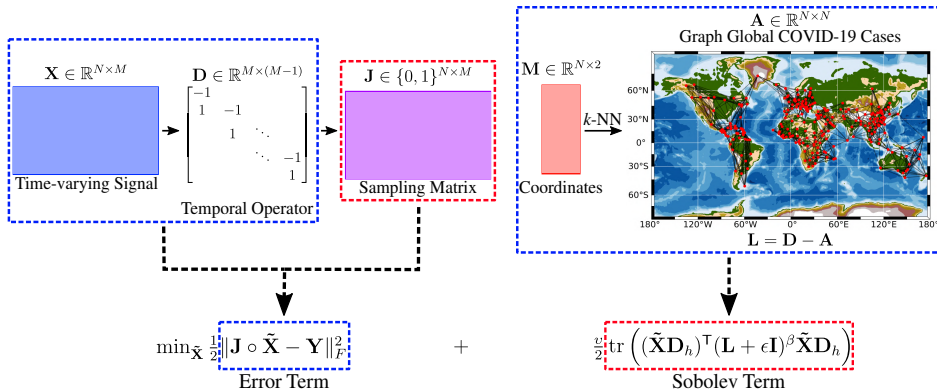


Ill-conditioned Problem



Reconstruction of Time-varying Graph Signals

Making the Problem Less Ill-posed

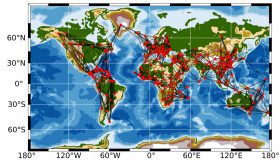


- The only change in the optimization algorithm is the calculation of the gradient as $\nabla_{\tilde{\mathbf{X}}} f_u(\tilde{\mathbf{X}}) = \mathbf{J} \circ \tilde{\mathbf{X}} - \mathbf{Y} + v(\mathbf{L} + \epsilon \mathbf{I})^\beta \tilde{\mathbf{X}} \mathbf{D}_h \mathbf{D}_h^\top$

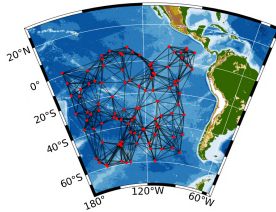
Reconstruction of Time-varying Graph Signals

Some Applications

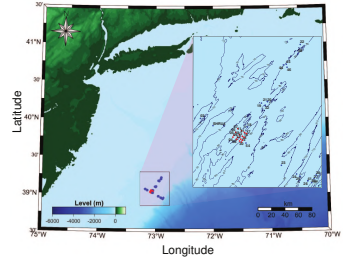
Graph Global COVID-19 Cases



Sea Surface Temperature



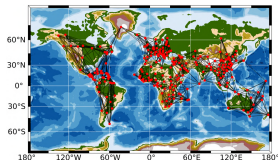
SW06 Experiment



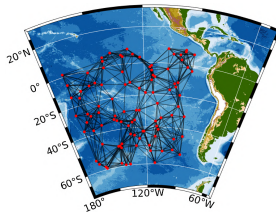
Reconstruction of Time-varying Graph Signals

Some Applications

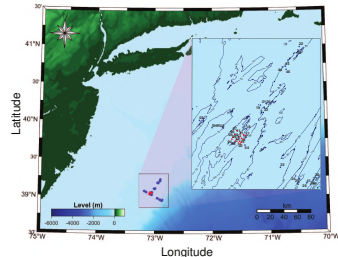
Graph Global COVID-19 Cases



Sea Surface Temperature



SW06 Experiment



- People have used these methodologies for pollution datasets (PM 2.5 concentration)

Reconstruction of Time-varying Graph Signals

Summary of Reconstructing the Optimization Way

- Reconstruction is improved by enforcing smoothness in the graph signal using the Laplacian quadratic form

Reconstruction of Time-varying Graph Signals

Summary of Reconstructing the Optimization Way

- ▶ Reconstruction is improved by enforcing smoothness in the graph signal using the Laplacian quadratic form
- ▶ The problem is formulated as an optimization task that minimizes a loss function combining a reconstruction error term and a smoothness regularization term

Reconstruction of Time-varying Graph Signals

Summary of Reconstructing the Optimization Way

- ▶ Reconstruction is improved by enforcing smoothness in the graph signal using the Laplacian quadratic form
- ▶ The problem is formulated as an optimization task that minimizes a loss function combining a reconstruction error term and a smoothness regularization term
- ▶ We can solve the problem using gradient descent techniques

Reconstruction of Time-varying Graph Signals

Summary of Reconstructing the Optimization Way

- ▶ Reconstruction is improved by enforcing smoothness in the graph signal using the Laplacian quadratic form
- ▶ The problem is formulated as an optimization task that minimizes a loss function combining a reconstruction error term and a smoothness regularization term
- ▶ We can solve the problem using gradient descent techniques
- ▶ Adding a small positive definite matrix ($\epsilon \mathbf{I}$) to the Laplacian matrix improves the problem's conditioning, accelerating convergence

Solving Reconstruction, the Neural Network Way

Reconstruction of Time-varying Graph Signals

Solving Reconstruction, the Neural Network Way (Roadmap)

- Now, we're gonna solve the problem of reconstruction with GNNs

Reconstruction of Time-varying Graph Signals

Solving Reconstruction, the Neural Network Way (Roadmap)

- ▶ Now, we're gonna solve the problem of reconstruction with GNNs
- ▶ What we do in this case is to define a GNN where the inputs are the Laplacian matrix and the signal with missing values

Reconstruction of Time-varying Graph Signals

Solving Reconstruction, the Neural Network Way (Roadmap)

- ▶ Now, we're gonna solve the problem of reconstruction with GNNs
- ▶ What we do in this case is to define a GNN where the inputs are the Laplacian matrix and the signal with missing values
- ▶ The GNN has an **autoencoder-type** architecture

Reconstruction of Time-varying Graph Signals

Solving Reconstruction, the Neural Network Way (Roadmap)

- ▶ Now, we're gonna solve the problem of reconstruction with GNNs
- ▶ What we do in this case is to define a GNN where the inputs are the Laplacian matrix and the signal with missing values
- ▶ The GNN has an **autoencoder-type** architecture
- ▶ The loss function resembles practically the same as in [Giraldo et al.(2022)], so you're already understand the intuitions

Reconstruction of Time-varying Graph Signals

Solving Reconstruction, the Neural Network Way (Roadmap)

- ▶ Now, we're gonna solve the problem of reconstruction with GNNs
- ▶ What we do in this case is to define a GNN where the inputs are the Laplacian matrix and the signal with missing values
- ▶ The GNN has an **autoencoder-type** architecture
- ▶ The loss function resembles practically the same as in [Giraldo et al.(2022)], so you're already understand the intuitions
- ▶ We don't need to apply complex optimization methods, but the regular SGD or Adam

Reconstruction of Time-varying Graph Signals

Solving Reconstruction, the Neural Network Way (Roadmap)

- ▶ Now, we're gonna solve the problem of reconstruction with GNNs
- ▶ What we do in this case is to define a GNN where the inputs are the Laplacian matrix and the signal with missing values
- ▶ The GNN has an **autoencoder-type** architecture
- ▶ The loss function resembles practically the same as in [Giraldo et al.(2022)], so you're already understand the intuitions
- ▶ We don't need to apply complex optimization methods, but the regular SGD or Adam
- ▶ However, optimizing the hyperparameters in this case is more complicated

Reconstruction of Time-varying Graph Signals

Solving Reconstruction, the Neural Network Way (Roadmap)

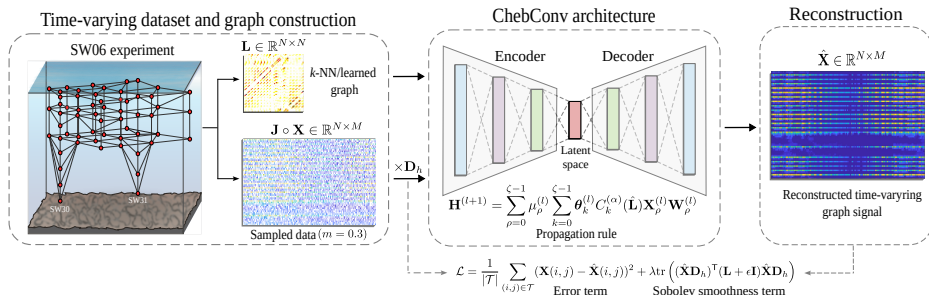
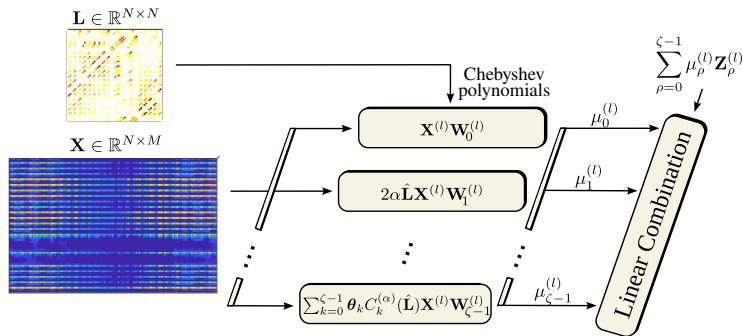


Figure: Source: [Castro-Correa et al.(2024)]

- Reconstruction of time-varying graph signals using a similar loss function as before, but here using a GNN (following an auto-encoder structure)

Reconstruction of Time-varying Graph Signals

The GNN layer

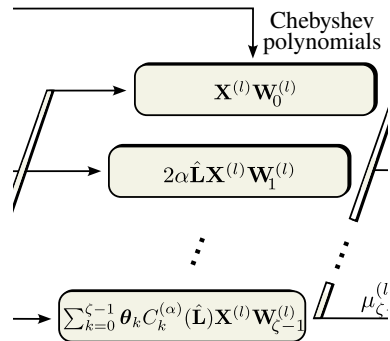


- Each layer of the GNN consists of a cascade of Chebyshev-based convolutions of increasing order $\rho = 0, \dots, \zeta - 1$. The outputs from all ζ convolutions are then merged using a linear combination layer, which includes learnable parameters μ_ρ

Reconstruction of Time-varying Graph Signals

The GNN layer

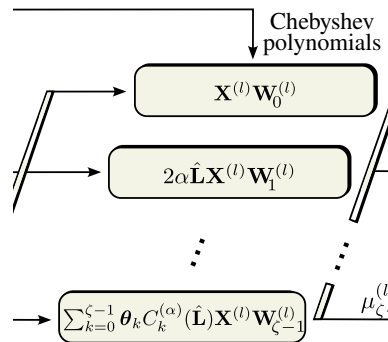
- For the first layer we have that $\mathbf{X}^{(0)} \in \mathbb{R}^{N \times M}$



Reconstruction of Time-varying Graph Signals

The GNN layer

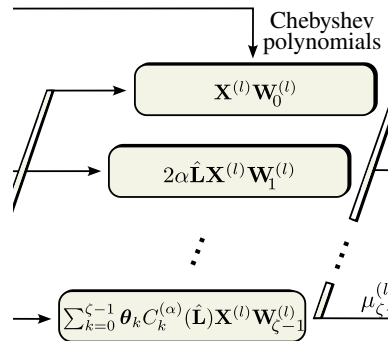
- ▶ For the first layer we have that $\mathbf{X}^{(0)} \in \mathbb{R}^{N \times M}$
- ▶ Each branch in the layer is a Chebyshev convolution of degree k , with $0 \leq k \leq \zeta - 1$



Reconstruction of Time-varying Graph Signals

The GNN layer

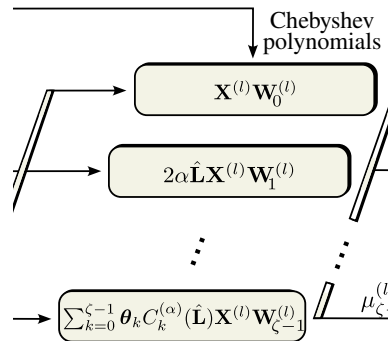
- ▶ For the first layer we have that $\mathbf{X}^{(0)} \in \mathbb{R}^{N \times M}$
- ▶ Each branch in the layer is a Chebyshev convolution of degree k , with $0 \leq k \leq \zeta - 1$
- ▶ The dimensions of all $\mathbf{W}_{\rho}^{(0)} \in \mathbb{R}^{M \times F_0} \quad \forall 0 \leq \rho \leq \zeta - 1$, where F_0 is the hidden dimension of the first layer



Reconstruction of Time-varying Graph Signals

The GNN layer

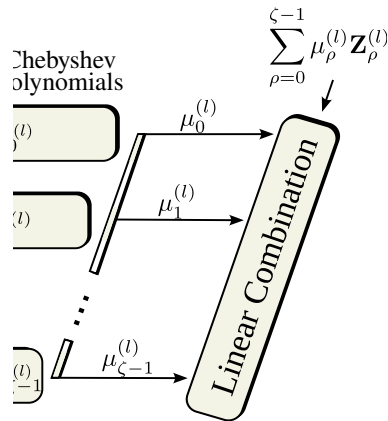
- ▶ For the first layer we have that $\mathbf{X}^{(0)} \in \mathbb{R}^{N \times M}$
- ▶ Each branch in the layer is a Chebyshev convolution of degree k , with $0 \leq k \leq \zeta - 1$
- ▶ The dimensions of all $\mathbf{W}_\rho^{(0)} \in \mathbb{R}^{M \times F_0} \forall 0 \leq \rho \leq \zeta - 1$, where F_0 is the hidden dimension of the first layer
- ▶ The number of branches ζ the layers is a hyperparameter



Reconstruction of Time-varying Graph Signals

The GNN layer

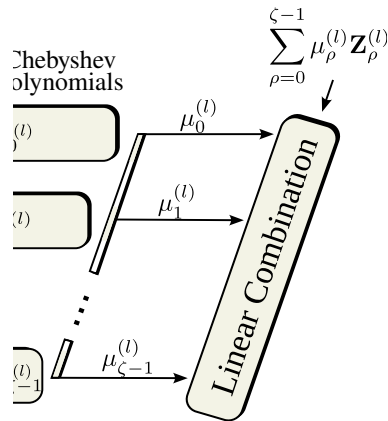
- The output of each branch ρ is a matrix $\mathbf{Z}_\rho^{(l)} \in \mathbb{R}^{N \times F_l}$



Reconstruction of Time-varying Graph Signals

The GNN layer

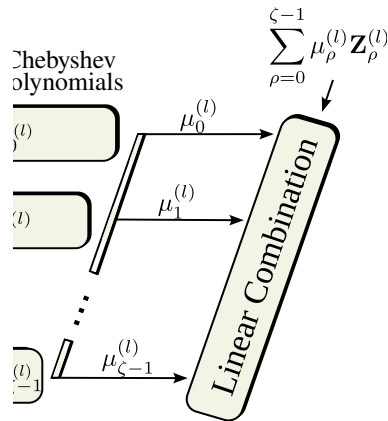
- ▶ The output of each branch ρ is a matrix $\mathbf{Z}_\rho^{(l)} \in \mathbb{R}^{N \times F_l}$
- ▶ We need to combine these outputs somehow, e.g. concatenate all of them, and apply some linear transformation



Reconstruction of Time-varying Graph Signals

The GNN layer

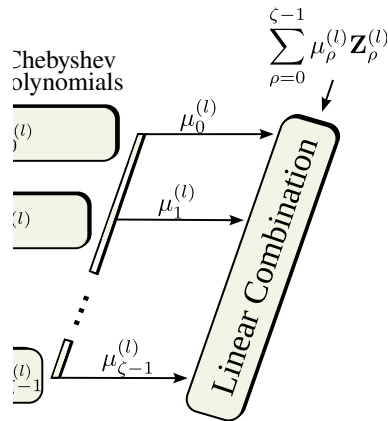
- ▶ The output of each branch ρ is a matrix $\mathbf{Z}_\rho^{(l)} \in \mathbb{R}^{N \times F_l}$
- ▶ We need to combine these outputs somehow, e.g. concatenate all of them, and apply some linear transformation
- ▶ In this case, we'll combine these outputs as $\mathbf{X}^{(l+1)} = \sum_{\rho=0}^{\zeta-1} \mu_\rho^{(l)} \mathbf{Z}_\rho^{(l)} \in \mathbb{R}^{N \times F_l}$



Reconstruction of Time-varying Graph Signals

The GNN layer

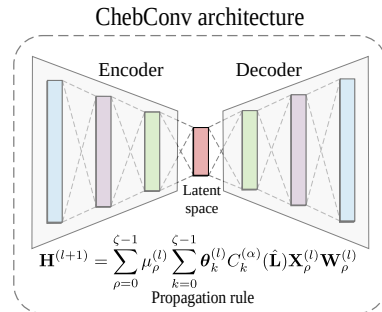
- ▶ The output of each branch ρ is a matrix $\mathbf{Z}_\rho^{(l)} \in \mathbb{R}^{N \times F_l}$
- ▶ We need to combine these outputs somehow, e.g. concatenate all of them, and apply some linear transformation
- ▶ In this case, we'll combine these outputs as $\mathbf{X}^{(l+1)} = \sum_{\rho=0}^{\zeta-1} \mu_\rho^{(l)} \mathbf{Z}_\rho^{(l)} \in \mathbb{R}^{N \times F_l}$
- ▶ $\mu_\rho^{(l)}$ are parameters in the GNN



Reconstruction of Time-varying Graph Signals

The Auto-encoder Architecture

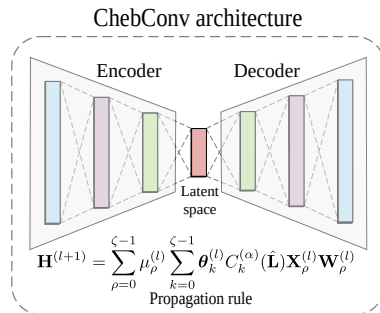
- The auto-encoder here means that for each node we reduce the hidden dimensions progressively



Reconstruction of Time-varying Graph Signals

The Auto-encoder Architecture

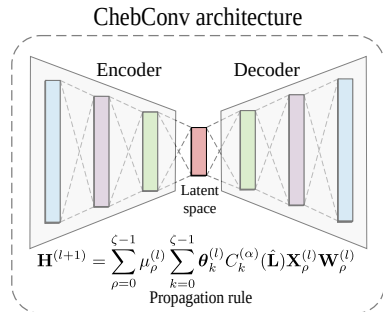
- ▶ The auto-encoder here means that for each node we reduce the hidden dimensions progressively
- ▶ For example, in $\mathbf{W}_\rho^{(0)} \in \mathbb{R}^{M \times F_0}$, $F_0 < M$



Reconstruction of Time-varying Graph Signals

The Auto-encoder Architecture

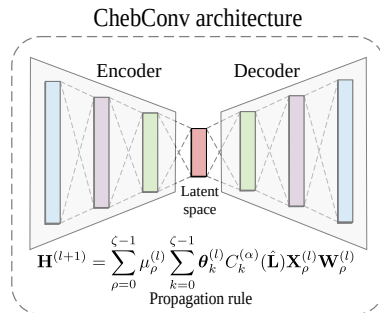
- ▶ The auto-encoder here means that for each node we reduce the hidden dimensions progressively
- ▶ For example, in $\mathbf{W}_\rho^{(0)} \in \mathbb{R}^{M \times F_0}$, $F_0 < M$
- ▶ In the middle of the GNN, we start increasing the hidden dimensions progressively



Reconstruction of Time-varying Graph Signals

The Auto-encoder Architecture

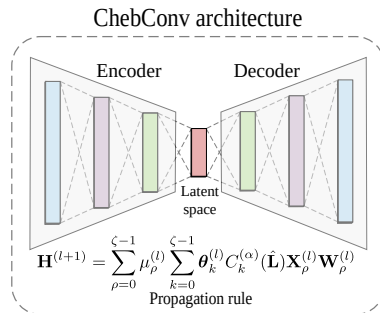
- ▶ The auto-encoder here means that for each node we reduce the hidden dimensions progressively
- ▶ For example, in $\mathbf{W}_\rho^{(0)} \in \mathbb{R}^{M \times F_0}$, $F_0 < M$
- ▶ In the middle of the GNN, we start increasing the hidden dimensions progressively
- ▶ The last layer $\mathbf{W}_\rho^{(l)} \in \mathbb{R}^{F_l \times M}$



Reconstruction of Time-varying Graph Signals

The Auto-encoder Architecture

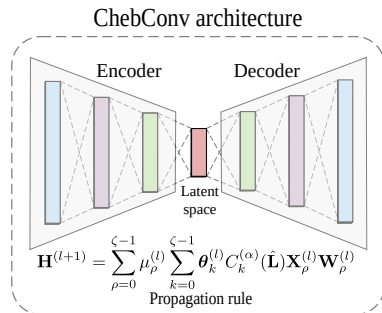
- ▶ The auto-encoder here means that for each node we reduce the hidden dimensions progressively
- ▶ For example, in $\mathbf{W}_\rho^{(0)} \in \mathbb{R}^{M \times F_0}$, $F_0 < M$
- ▶ In the middle of the GNN, we start increasing the hidden dimensions progressively
- ▶ The last layer $\mathbf{W}_\rho^{(l)} \in \mathbb{R}^{F_l \times M}$
- ▶ In practice, these GNNs are shallow, why?



Reconstruction of Time-varying Graph Signals

The Auto-encoder Architecture

- ▶ The auto-encoder here means that for each node we reduce the hidden dimensions progressively
- ▶ For example, in $\mathbf{W}_\rho^{(0)} \in \mathbb{R}^{M \times F_0}$, $F_0 < M$
- ▶ In the middle of the GNN, we start increasing the hidden dimensions progressively
- ▶ The last layer $\mathbf{W}_\rho^{(l)} \in \mathbb{R}^{F_l \times M}$
- ▶ In practice, these GNNs are shallow, why?
- ▶ **Over-smoothing**



- The loss function in this model is given by:

$$\mathcal{L} = \frac{1}{|\mathcal{T}|} \sum_{(i,j) \in \mathcal{T}} (\mathbf{X}_{i,j} - \tilde{\mathbf{X}}_{i,j})^2 + \lambda \operatorname{tr} \left((\tilde{\mathbf{X}} \mathbf{D}_h)^\top (\mathbf{L} + \epsilon \mathbf{I}) \tilde{\mathbf{X}} \mathbf{D}_h \right), \quad (4)$$

Reconstruction of Time-varying Graph Signals

The Loss Function

- The loss function in this model is given by:

$$\mathcal{L} = \frac{1}{|\mathcal{T}|} \sum_{(i,j) \in \mathcal{T}} (\mathbf{X}_{i,j} - \tilde{\mathbf{X}}_{i,j})^2 + \lambda \text{tr} \left((\tilde{\mathbf{X}} \mathbf{D}_h)^T (\mathbf{L} + \epsilon \mathbf{I}) \tilde{\mathbf{X}} \mathbf{D}_h \right), \quad (4)$$

- $\tilde{\mathbf{X}}$ is the reconstructed graph signal, $\mathcal{T}$ is the training set, with $\mathcal{T}$ a subset of the spatiotemporal sampled indexes given by the sampling matrix $\mathbf{J}$, and $\epsilon \in \mathbb{R}^+$ is a hyperparameter

Reconstruction of Time-varying Graph Signals

The Loss Function

- The loss function in this model is given by:

$$\mathcal{L} = \frac{1}{|\mathcal{T}|} \sum_{(i,j) \in \mathcal{T}} (\mathbf{X}_{i,j} - \tilde{\mathbf{X}}_{i,j})^2 + \lambda \operatorname{tr} \left((\tilde{\mathbf{X}} \mathbf{D}_h)^\top (\mathbf{L} + \epsilon \mathbf{I}) \tilde{\mathbf{X}} \mathbf{D}_h \right), \quad (4)$$

- $\tilde{\mathbf{X}}$ is the reconstructed graph signal, $\mathcal{T}$ is the training set, with $\mathcal{T}$ a subset of the spatiotemporal sampled indexes given by the sampling matrix $\mathbf{J}$, and $\epsilon \in \mathbb{R}^+$ is a hyperparameter
- This is just an MSE loss plus the Sobolev function we introduced before

Reconstruction of Time-varying Graph Signals

Summary of Reconstructing the Neural Network Way

- Reconstruction is performed using a GNN with an autoencoder-type structure, where the inputs are the Laplacian matrix and the signal with missing values

Reconstruction of Time-varying Graph Signals

Summary of Reconstructing the Neural Network Way

- ▶ Reconstruction is performed using a GNN with an autoencoder-type structure, where the inputs are the Laplacian matrix and the signal with missing values
- ▶ The model uses a cascade of graph Chebyshev convolutions in each layer

Reconstruction of Time-varying Graph Signals

Summary of Reconstructing the Neural Network Way

- ▶ Reconstruction is performed using a GNN with an autoencoder-type structure, where the inputs are the Laplacian matrix and the signal with missing values
- ▶ The model uses a cascade of graph Chebyshev convolutions in each layer
- ▶ The loss function consists of an MSE term for reconstruction accuracy and a Sobolev smoothness regularization term

Reconstruction of Time-varying Graph Signals

Summary of Reconstructing the Neural Network Way

- ▶ Reconstruction is performed using a GNN with an autoencoder-type structure, where the inputs are the Laplacian matrix and the signal with missing values
- ▶ The model uses a cascade of graph Chebyshev convolutions in each layer
- ▶ The loss function consists of an MSE term for reconstruction accuracy and a Sobolev smoothness regularization term
- ▶ We train the models with standard Adam or SGD algorithms

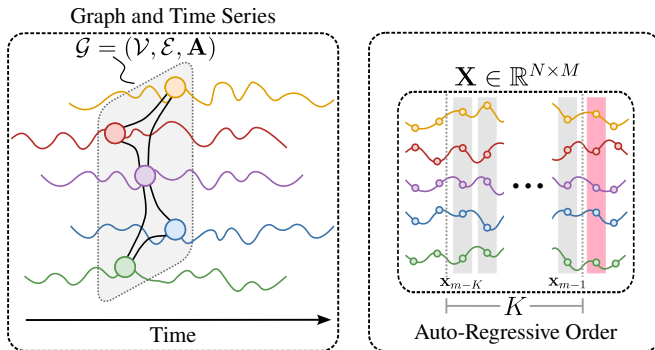
Forecasting of Time-varying Graph Signals

Problem Definition in Forecasting

Given a graph $\mathcal{G}$, a time-varying graph signal $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M] \in \mathbb{R}^{N \times M}$, and a temporal window $\mathbf{X}_K = [\mathbf{x}_{m-K}, \dots, \mathbf{x}_{m-1}] \in \mathbb{R}^{N \times K}$ with $K < M$, the objective of forecasting is to predict the future time-step(s) $\mathbf{x}_m$. We call K the auto-regressive order

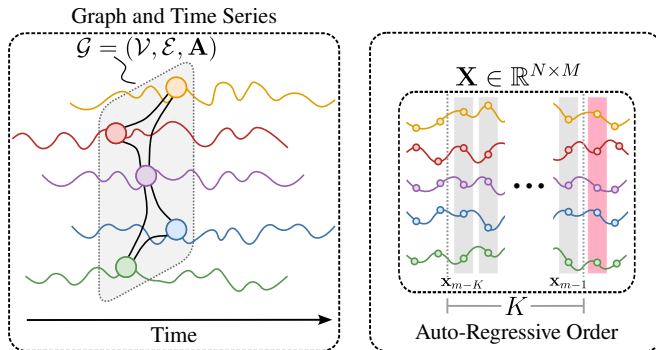
Forecasting of Time-varying Graph Signals

Formal Problem Definition for Forecasting



Forecasting of Time-varying Graph Signals

Formal Problem Definition for Forecasting



- We can also try to forecast more than one future value, in such case, we say we forecast multiple horizons

Forecasting of Time-varying Graph Signals

Auto-regressive Modeling

- ▶ The **auto-regressive** (AR) model is a fundamental time series model where future values are predicted based on past values

Forecasting of Time-varying Graph Signals

Auto-regressive Modeling

- ▶ The **auto-regressive** (AR) model is a fundamental time series model where future values are predicted based on past values
- ▶ It assumes a linear dependency between a variable and its previous values

Forecasting of Time-varying Graph Signals

Auto-regressive Modeling

- ▶ The **auto-regressive** (AR) model is a fundamental time series model where future values are predicted based on past values
- ▶ It assumes a linear dependency between a variable and its previous values
- ▶ Assuming a time series (no graph structure yet), the general $AR(K)$ model (where K is the order) is formulated as:

$$x_m = \sum_{i=1}^K \phi_i x_{m-i} + \epsilon_m, \quad (5)$$

where x_m is the current value of the time series, ϕ_i are the autoregressive coefficients, x_{m-i} are past values (**lags**), and ϵ_m is a white noise error term

Forecasting of Time-varying Graph Signals

Intuitions of Auto-regressive Modeling

- ▶ The value of a time series at the current time step is a weighted sum of its past values, e.g. today's temperature is influenced by the temperatures of previous days

Forecasting of Time-varying Graph Signals

Intuitions of Auto-regressive Modeling

- ▶ The value of a time series at the current time step is a weighted sum of its past values, e.g. today's temperature is influenced by the temperatures of previous days
- ▶ The order K (number of lags) defines how much past information is retained

Forecasting of Time-varying Graph Signals

Intuitions of Auto-regressive Modeling

- ▶ The value of a time series at the current time step is a weighted sum of its past values, e.g. today's temperature is influenced by the temperatures of previous days
- ▶ The order K (number of lags) defines how much past information is retained
- ▶ A higher-order AR model captures longer dependencies, while a lower order captures short-term patterns

Forecasting of Time-varying Graph Signals

Intuitions of Auto-regressive Modeling

- ▶ The value of a time series at the current time step is a weighted sum of its past values, e.g. today's temperature is influenced by the temperatures of previous days
- ▶ The order K (number of lags) defines how much past information is retained
- ▶ A higher-order AR model captures longer dependencies, while a lower order captures short-term patterns
- ▶ The error term ϵ_m represents **random noise** or external influences, like sudden events

Forecasting of Time-varying Graph Signals

Intuitions of Auto-regressive Modeling

- ▶ The value of a time series at the current time step is a weighted sum of its past values, e.g. today's temperature is influenced by the temperatures of previous days
- ▶ The order K (number of lags) defines how much past information is retained
- ▶ A higher-order AR model captures longer dependencies, while a lower order captures short-term patterns
- ▶ The error term ϵ_m represents **random noise** or external influences, like sudden events
- ▶ The autoregressive coefficients weigh the importance of past values, typically the weight of past values gradually diminishes as time progresses

- We can easily extend the idea of AR modeling to graphs as follows:

$$\mathbf{x}_m = \sum_{i=1}^K \Phi_i \mathbf{x}_{m-i} + \epsilon_m \quad (6)$$

Forecasting of Time-varying Graph Signals

Auto-regressive Modeling on Graphs

- ▶ We can easily extend the idea of AR modeling to graphs as follows:

$$\mathbf{x}_m = \sum_{i=1}^K \Phi_i \mathbf{x}_{m-i} + \epsilon_m \quad (6)$$

- ▶ In this case, the matrices $\Phi_i \in \mathbb{R}^{N \times N}$ could represent graph filters whose coefficients should be optimized somehow or could be GNNs

Forecasting of Time-varying Graph Signals

Auto-regressive Modeling on Graphs

- ▶ We can easily extend the idea of AR modeling to graphs as follows:

$$\mathbf{x}_m = \sum_{i=1}^K \Phi_i \mathbf{x}_{m-i} + \epsilon_m \quad (6)$$

- ▶ In this case, the matrices $\Phi_i \in \mathbb{R}^{N \times N}$ could represent graph filters whose coefficients should be optimized somehow or could be GNNs
- ▶ Intuitively, the value of a node at time m depends on its own past values at previous time steps, e.g. the traffic flow at a given intersection is influenced by traffic patterns in the past few hours

Forecasting of Time-varying Graph Signals

Auto-regressive Modeling on Graphs

- ▶ Unlike traditional time series, in spatiotemporal graphs, dependencies exist not only over time but also across connected nodes

Forecasting of Time-varying Graph Signals

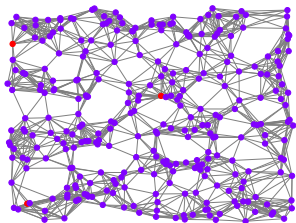
Auto-regressive Modeling on Graphs

- ▶ Unlike traditional time series, in spatiotemporal graphs, dependencies exist not only over time but also across connected nodes
- ▶ This means that both historical values and neighboring nodes contribute to the forecast

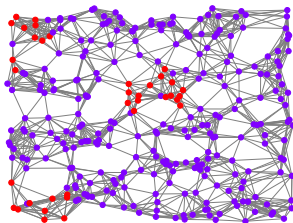
Forecasting of Time-varying Graph Signals

Auto-regressive Modeling on Graphs

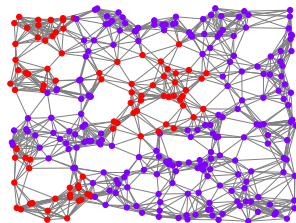
- ▶ Unlike traditional time series, in spatiotemporal graphs, dependencies exist not only over time but also across connected nodes
- ▶ This means that both historical values and neighboring nodes contribute to the forecast



$$\mathbf{x}^{(0)} = \mathbf{x} = \mathbf{S}^0 \mathbf{x}$$



$$\mathbf{x}^{(1)} = \mathbf{S}\mathbf{x}^{(0)} = \mathbf{S}^1 \mathbf{x}$$



$$\mathbf{x}^{(2)} = \mathbf{S}\mathbf{x}^{(1)} = \mathbf{S}^2 \mathbf{x}$$

Remember the diffusion sequence?

Forecasting of Time-varying Graph Signals

Auto-regressive Modeling on Graphs

- In a very simple case we'll have:

$$\mathbf{x}_m = \sum_{i=1}^K (h_{0,i} \mathbf{I} + h_{1,i} \mathbf{S}) \mathbf{x}_{m-i} + \epsilon_m, \quad (7)$$

where the value of some node at time m depends on its past values and the past values of its direct neighbors

Forecasting of Time-varying Graph Signals

Auto-regressive Modeling on Graphs

- In a very simple case we'll have:

$$\mathbf{x}_m = \sum_{i=1}^K (h_{0,i} \mathbf{I} + h_{1,i} \mathbf{S}) \mathbf{x}_{m-i} + \epsilon_m, \quad (7)$$

where the value of some node at time m depends on its past values and the past values of its direct neighbors

- In general, we have:

$$\mathbf{x}_m = \sum_{i=1}^K \sum_{j=0}^p h_{j,i} \mathbf{S}^j \mathbf{x}_{m-i} + \epsilon_m, \quad (8)$$

where the value of some node at time m depends on its past values and the past values of its p -hop neighborhood

- We can now define a GNN to forecast $\mathbf{x}_m$ as follows:

$$\tilde{\mathbf{x}}_m = \sum_{i=1}^K \alpha_i \sigma \left(\sum_{j=0}^p h_{j,i} \mathbf{S}^j \mathbf{x}_{m-i} \right), \quad (9)$$

where α_i is some weighting parameters and $\sigma(\cdot)$ is some non-linearity

- We can now define a GNN to forecast $\mathbf{x}_m$ as follows:

$$\tilde{\mathbf{x}}_m = \sum_{i=1}^K \alpha_i \sigma \left(\sum_{j=0}^p h_{j,i} \mathbf{S}^j \mathbf{x}_{m-i} \right), \quad (9)$$

where α_i is some weighting parameters and $\sigma(\cdot)$ is some non-linearity

- The intuition is similar to that of the graph filter, with the difference that now we have non-linearities and the parameters α_i

Forecasting of Time-varying Graph Signals

Loss Function

- ▶ We're only missing the loss function to train our GNN

- ▶ We're only missing the loss function to train our GNN
- ▶ The typical choice is the MSE loss function:

$$\mathcal{L}_{\text{MSE}}(\mathbf{x}_m, \tilde{\mathbf{x}}_m) = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_m(i) - \tilde{\mathbf{x}}_m(i))^2 \quad (10)$$

Forecasting of Time-varying Graph Signals

Loss Function

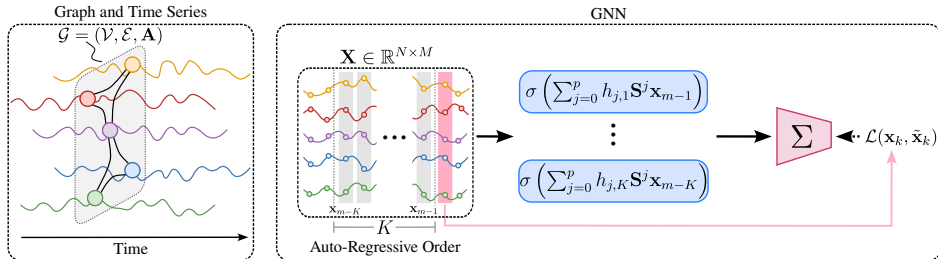
- ▶ We're only missing the loss function to train our GNN
- ▶ The typical choice is the MSE loss function:

$$\mathcal{L}_{\text{MSE}}(\mathbf{x}_m, \tilde{\mathbf{x}}_m) = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_m(i) - \tilde{\mathbf{x}}_m(i))^2 \quad (10)$$

- ▶ Now we just need to define our optimizer and train our model

Forecasting of Time-varying Graph Signals

The GNN Model Graphically



Forecasting of time-varying graph signals using an MSE loss function using a GNN architecture

Forecasting of Time-varying Graph Signals

Forecasting Multiple Horizons (MLP Head)

- ▶ If we want to forecast multiple horizons, the easiest way is to use an **MLP head** at the end of our model

Forecasting of Time-varying Graph Signals

Forecasting Multiple Horizons (MLP Head)

- ▶ If we want to forecast multiple horizons, the easiest way is to use an **MLP head** at the end of our model
- ▶ Let H be the number of horizons we want to forecast. Our GNN architecture is now given as follows:

$$\tilde{\mathbf{X}}_m = \sigma \left(\left[\sum_{i=1}^K \alpha_i \sigma \left(\sum_{j=0}^p h_{j,i} \mathbf{S}^j \mathbf{x}_{m-i} \right) \right] \mathbf{w}_h^\top \right), \quad (11)$$

where $\tilde{\mathbf{X}}_m \in \mathbb{R}^{N \times H}$, and $\mathbf{w}_h \in \mathbb{R}^{H \times 1}$ is a vector of learnable parameters

Forecasting of Time-varying Graph Signals

Forecasting Multiple Horizons (MLP Head)

- ▶ If we want to forecast multiple horizons, the easiest way is to use an **MLP head** at the end of our model
- ▶ Let H be the number of horizons we want to forecast. Our GNN architecture is now given as follows:

$$\tilde{\mathbf{X}}_m = \sigma \left(\left[\sum_{i=1}^K \alpha_i \sigma \left(\sum_{j=0}^p h_{j,i} \mathbf{S}^j \mathbf{x}_{m-i} \right) \right] \mathbf{w}_h^\top \right), \quad (11)$$

where $\tilde{\mathbf{X}}_m \in \mathbb{R}^{N \times H}$, and $\mathbf{w}_h \in \mathbb{R}^{H \times 1}$ is a vector of learnable parameters

- ▶ Now we have to modify the loss as the MSE function between the matrices $\tilde{\mathbf{X}}_m$ and $\mathbf{X}_m$

Forecasting of Time-varying Graph Signals

Forecasting Multiple Horizons (Recursive Formulation)

- Let $\text{GNN}_{\mathcal{G},\theta}(\cdot)$ be the GNN we define before.

Forecasting of Time-varying Graph Signals

Forecasting Multiple Horizons (Recursive Formulation)

- ▶ Let $\text{GNN}_{\mathcal{G},\theta}(\cdot)$ be the GNN we define before.
- ▶ For the first time horizon we have $\tilde{\mathbf{x}}_m = \text{GNN}_{\mathcal{G},\theta}(\mathbf{x}_{m-1}, \dots, \mathbf{x}_{m-K})$

Forecasting of Time-varying Graph Signals

Forecasting Multiple Horizons (Recursive Formulation)

- ▶ Let $\text{GNN}_{\mathcal{G},\theta}(\cdot)$ be the GNN we define before.
- ▶ For the first time horizon we have $\tilde{\mathbf{x}}_m = \text{GNN}_{\mathcal{G},\theta}(\mathbf{x}_{m-1}, \dots, \mathbf{x}_{m-K})$
- ▶ For the second time horizon we have
 $\tilde{\mathbf{x}}_{m+1} = \text{GNN}_{\mathcal{G},\theta}(\tilde{\mathbf{x}}_m, \mathbf{x}_{m-1}, \dots, \mathbf{x}_{m-K+1})$, *i.e.*, we're using the outputs of the previous forward pass of the GNN as a new input

Forecasting of Time-varying Graph Signals

Forecasting Multiple Horizons (Recursive Formulation)

$$\tilde{\mathbf{x}}_{m+1} = \text{GNN}_{\mathcal{G}, \theta}(\tilde{\mathbf{x}}_m, \mathbf{x}_{m-1}, \dots, \mathbf{x}_{m-K+1})$$

- Please notice two things:

Forecasting of Time-varying Graph Signals

Forecasting Multiple Horizons (Recursive Formulation)

$$\tilde{\mathbf{x}}_{m+1} = \text{GNN}_{\mathcal{G}, \theta}(\tilde{\mathbf{x}}_m, \mathbf{x}_{m-1}, \dots, \mathbf{x}_{m-K+1})$$

- Please notice two things:
 1. The window has been shifted

Forecasting of Time-varying Graph Signals

Forecasting Multiple Horizons (Recursive Formulation)

$$\tilde{\mathbf{x}}_{m+1} = \text{GNN}_{\mathcal{G}, \theta}(\tilde{\mathbf{x}}_m, \mathbf{x}_{m-1}, \dots, \mathbf{x}_{m-K+1})$$

- Please notice two things:
 1. The window has been shifted
 2. The GNN has shared parameters θ

Forecasting of Time-varying Graph Signals

Forecasting Multiple Horizons (Recursive Formulation)

$$\tilde{\mathbf{x}}_{m+1} = \text{GNN}_{\mathcal{G}, \theta}(\tilde{\mathbf{x}}_m, \mathbf{x}_{m-1}, \dots, \mathbf{x}_{m-K+1})$$

- ▶ Please notice two things:
 1. The window has been shifted
 2. The GNN has shared parameters θ
- ▶ We apply this recursion $H - 1$ times to get a matrix $\tilde{\mathbf{X}}_m \in \mathbb{R}^{N \times H}$ and apply MSE

Forecasting of Time-varying Graph Signals

Forecasting Multiple Horizons (Recursive Formulation)

$$\tilde{\mathbf{x}}_{m+1} = \text{GNN}_{\mathcal{G}, \theta}(\tilde{\mathbf{x}}_m, \mathbf{x}_{m-1}, \dots, \mathbf{x}_{m-K+1})$$

- ▶ Please notice two things:
 1. The window has been shifted
 2. The GNN has shared parameters θ
- ▶ We apply this recursion $H - 1$ times to get a matrix $\tilde{\mathbf{X}}_m \in \mathbb{R}^{N \times H}$ and apply MSE
- ▶ This formulation works better in practice

Forecasting of Time-varying Graph Signals

GraphCast

- ▶ **GraphCast** from DeepMind is one of the most promising machine-learning based spatiotemporal forecasting methods currently

Forecasting of Time-varying Graph Signals

GraphCast

- ▶ **GraphCast** from DeepMind is one of the most promising machine-learning based spatiotemporal forecasting methods currently
- ▶ GraphCast is a GNN-based model for medium-range weather forecasting (up to 10 days)

Forecasting of Time-varying Graph Signals

GraphCast

- ▶ **GraphCast** from DeepMind is one of the most promising machine-learning based spatiotemporal forecasting methods currently
- ▶ GraphCast is a GNN-based model for medium-range weather forecasting (up to 10 days)
- ▶ It learns directly from historical weather data instead of relying on traditional numerical weather prediction

Forecasting of Time-varying Graph Signals

GraphCast

- ▶ **GraphCast** from DeepMind is one of the most promising machine-learning based spatiotemporal forecasting methods currently
- ▶ GraphCast is a GNN-based model for medium-range weather forecasting (up to 10 days)
- ▶ It learns directly from historical weather data instead of relying on traditional numerical weather prediction
- ▶ Weather data is represented as a **global multi-mesh graph**, rather than a regular latitude-longitude grid

Forecasting of Time-varying Graph Signals

GraphCast

- ▶ **GraphCast** from DeepMind is one of the most promising machine-learning based spatiotemporal forecasting methods currently
- ▶ GraphCast is a GNN-based model for medium-range weather forecasting (up to 10 days)
- ▶ It learns directly from historical weather data instead of relying on traditional numerical weather prediction
- ▶ Weather data is represented as a **global multi-mesh graph**, rather than a regular latitude-longitude grid
- ▶ The GNN architecture uses message-passing across hierarchical icosahedral meshes, enabling local and global information propagation

Forecasting of Time-varying Graph Signals

GraphCast

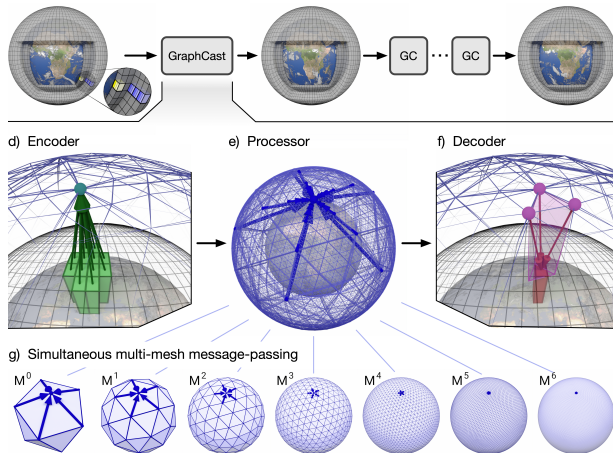


Figure: GraphCast
Source: [Lam et al.(2023)]

Forecasting of Time-varying Graph Signals

GraphCast

- ▶ GraphCast follows an AR approach, similar to what we saw before

Forecasting of Time-varying Graph Signals

GraphCast

- ▶ GraphCast follows an AR approach, similar to what we saw before
- ▶ GraphCast has many other elements in its architecture that I'm not discussing here

Forecasting of Time-varying Graph Signals

GraphCast

- ▶ GraphCast follows an AR approach, similar to what we saw before
- ▶ GraphCast has many other elements in its architecture that I'm not discussing here
- ▶ The key message you should retain is that we compute each node output based on the previous time steps and the message-passing at different resolutions

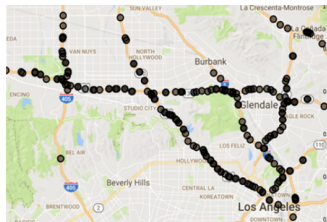
Forecasting of Time-varying Graph Signals

GraphCast

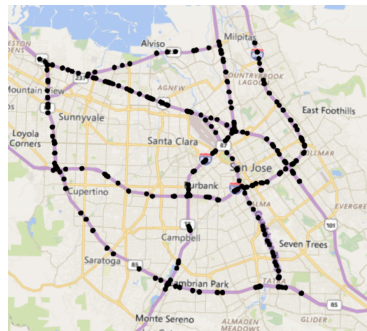
- ▶ GraphCast follows an AR approach, similar to what we saw before
- ▶ GraphCast has many other elements in its architecture that I'm not discussing here
- ▶ The key message you should retain is that we compute each node output based on the previous time steps and the message-passing at different resolutions
- ▶ GraphCast is faster than traditional numerical methods, producing a 10-day forecast in under one minute on a single TPU, compared to the around one hour computation required for traditional methods

Forecasting of Time-varying Graph Signals

Other Applications



MetrLA



PemsBay

- ▶ **MetrLA** consists of recorded traffic data for four months on 207 highways in Los Angeles County with 5-minute resolutions
- ▶ **Pems-Bay** contains 5-minute traffic load data for six months associated with 325 stations in the Bay Area

Forecasting of Time-varying Graph Signals

Summary

- ▶ One of the most common approaches for forecasting in spatiotemporal graphs is auto-regressive modeling, where future values are predicted based on past values of the time series

Forecasting of Time-varying Graph Signals

Summary

- ▶ One of the most common approaches for forecasting in spatiotemporal graphs is auto-regressive modeling, where future values are predicted based on past values of the time series
- ▶ In the graph domain, AR models incorporate graph filters or GNNs to capture both temporal dependencies and spatial relationships among connected nodes

Forecasting of Time-varying Graph Signals

Summary

- ▶ One of the most common approaches for forecasting in spatiotemporal graphs is auto-regressive modeling, where future values are predicted based on past values of the time series
- ▶ In the graph domain, AR models incorporate graph filters or GNNs to capture both temporal dependencies and spatial relationships among connected nodes
- ▶ The MSE loss is used as loss function, and models are optimized using standard methods like SGD or Adam

Forecasting of Time-varying Graph Signals

Summary

- ▶ One of the most common approaches for forecasting in spatiotemporal graphs is auto-regressive modeling, where future values are predicted based on past values of the time series
- ▶ In the graph domain, AR models incorporate graph filters or GNNs to capture both temporal dependencies and spatial relationships among connected nodes
- ▶ The MSE loss is used as loss function, and models are optimized using standard methods like SGD or Adam
- ▶ Forecasting can predict multiple future time steps using either an MLP head or a recursive formulation, where previous predictions are fed back as inputs to extend the forecasting horizon

Wooclap Championship!

Multiple Answers Might be Correct, Time is Considered for the Score.

Prizes: bonus +2 in the TP for the first and +1 for the second and third!

- ▶ The GFT is a tool to capture spectral information from graph signals

Main Takeaways

- ▶ The GFT is a tool to capture spectral information from graph signals
- ▶ Spectral GNNs use the spectral domain to compute convolutions

Main Takeaways

- ▶ The GFT is a tool to capture spectral information from graph signals
- ▶ Spectral GNNs use the spectral domain to compute convolutions
- ▶ ChebConv approximates spectral GNNs, making spectral methods more practical for large graphs

Main Takeaways

- ▶ The GFT is a tool to capture spectral information from graph signals
- ▶ Spectral GNNs use the spectral domain to compute convolutions
- ▶ ChebConv approximates spectral GNNs, making spectral methods more practical for large graphs
- ▶ Missing data in spatiotemporal graphs is a common issue, and reconstruction techniques using optimization or neural networks help recover missing values

- ▶ The GFT is a tool to capture spectral information from graph signals
- ▶ Spectral GNNs use the spectral domain to compute convolutions
- ▶ ChebConv approximates spectral GNNs, making spectral methods more practical for large graphs
- ▶ Missing data in spatiotemporal graphs is a common issue, and reconstruction techniques using optimization or neural networks help recover missing values
- ▶ By minimizing an error term with Laplacian smoothness regularization, missing graph signals can be reconstructed using gradient descent techniques

Main Takeaways

- ▶ A GNN autoencoder can be trained using an MSE loss function with Sobolev smoothness regularization, making reconstruction more flexible but requiring hyperparameter tuning

Main Takeaways

- ▶ A GNN autoencoder can be trained using an MSE loss function with Sobolev smoothness regularization, making reconstruction more flexible but requiring hyperparameter tuning
- ▶ Future values in a time-varying graph signal can be predicted using graph AR models, where past time steps influence current predictions

Main Takeaways

- ▶ A GNN autoencoder can be trained using an MSE loss function with Sobolev smoothness regularization, making reconstruction more flexible but requiring hyperparameter tuning
- ▶ Future values in a time-varying graph signal can be predicted using graph AR models, where past time steps influence current predictions
- ▶ Graph-based forecasting leverages message passing and graph convolutions, capturing both temporal and spatial dependencies

- ▶ A GNN autoencoder can be trained using an MSE loss function with Sobolev smoothness regularization, making reconstruction more flexible but requiring hyperparameter tuning
- ▶ Future values in a time-varying graph signal can be predicted using graph AR models, where past time steps influence current predictions
- ▶ Graph-based forecasting leverages message passing and graph convolutions, capturing both temporal and spatial dependencies
- ▶ Methods like GraphCast demonstrate how GNN-based forecasting can outperform traditional numerical approaches in areas like climate modeling

Thank you!

Jhony H. Giraldo: jhony.giraldo@telecom-paris.fr
<https://jhonygiraldo.github.io/>



Jhon A Castro-Correa et al.

Gegenbauer graph neural networks for time-varying signal reconstruction.
IEEE Transactions on Neural Networks and Learning Systems, 2024.



M. Defferrard et al.

Convolutional neural networks on graphs with fast localized spectral filtering.
In Advances in Neural Information Processing Systems, 2016.



Jhony H Giraldo et al.

Reconstruction of time-varying graph signals via sobolev smoothness.
IEEE Transactions on Signal and Information Processing over Networks, 8:201–214, 2022.



Remi Lam et al.

Learning skillful medium-range global weather forecasting.

Science, 382(6677):1416–1421, 2023.



Antonio Ortega.

Introduction to graph signal processing.

Cambridge University Press, 2022.



Kai Qiu et al.

Time-varying graph signal reconstruction.

IEEE Journal of Selected Topics in Signal Processing, 11(6):870–883, 2017.



Xiyuan Wang and Muhan Zhang.

How powerful are spectral graph neural networks.

In International Conference on Machine Learning, 2022.

References III