Signal Processing on Graphs

ACMS 80770: Deep Learning with Graphs

Instructor: Navid Shervani-Tabar

Department of Applied and Comp Math and Stats

UNIVERSITY OF NOTRE DAME

Intro

- So far, we have looked at graph neural network layers that work based on the principle of **neural message passing**.

Intro

- So far, we have looked at graph neural network layers that work based on the principle of **neural message passing**.
  - This learning rule propagates information in a graph by **aggregating** the information in the neighborhood of a node.

$$m^{(k)}_{N_i \rightarrow i} = \texttt{aggregate}(\{h^{(k)}_j | v_j \in N(v_i)\})$$

Intro

- So far, we have looked at graph neural network layers that work based on the principle of **neural message passing**.
  - This learning rule propagates information in a graph by **aggregating** the information in the neighborhood of a node.

$$m_{N_i \to i}^{(k)} = \texttt{aggregate}(\{h_j^{(k)} | v_j \in N(v_i)\})$$

  - This information is then used to construct a message that **updates** the state of the node.

$$h_i^{(k+1)} = \texttt{update}(h_i^{(k)}, m_{N_i \to i}^{(k)})$$

Intro

- So far, we have looked at graph neural network layers that work based on the principle of **neural message passing**.
  - This learning rule propagates information in a graph by **aggregating** the information in the neighborhood of a node.

$$m_{N_i \to i}^{(k)} = \mathtt{aggregate}(\{h_j^{(k)} | v_j \in N(v_i)\})$$

  - This information is then used to construct a message that **updates** the state of the node.

$$h_i^{(k+1)} = \mathtt{update}(h_i^{(k)}, m_{N_i \to i}^{(k)})$$

- These approaches are **motivated** by message passing schemes in **probabilistic graphical models**.

Intro

- So far, we have looked at graph neural network layers that work based on the principle of **neural message passing**.
- These approaches are **motivated** by message passing schemes in probabilistic graphical models.
- One can motivate and develop GNN models from different perspectives:

Intro

- So far, we have looked at graph neural network layers that work based on the principle of **neural message passing**.
- These approaches are **motivated** by message passing schemes in probabilistic graphical models.
- One can motivate and develop GNN models from different perspectives:
  - Probabilistic graph models

Intro

- So far, we have looked at graph neural network layers that work based on the principle of **neural message passing**.
- These approaches are **motivated** by message passing schemes in probabilistic graphical models.
- One can motivate and develop GNN models from different perspectives:
  - Probabilistic graph models
  - Graph isomorphism test

Intro

- So far, we have looked at graph neural network layers that work based on the principle of **neural message passing**.
- These approaches are **motivated** by message passing schemes in probabilistic graphical models.
- One can motivate and develop GNN models from different perspectives:
  - Probabilistic graph models
  - Graph isomorphism test
  - Graph signal processing

Graph Signal Processing

- Signal processing on graphs deals with defining a set of tools to **analyze the data** $\mathbf{f}$ residing on nodes $v_i \in \mathrm{V}$ of the graph.

Graph Signal Processing

- Signal processing on graphs deals with defining a set of tools to **analyze the data** $\mathbf{f}$ residing on nodes $v_i \in \mathrm{V}$ of the graph.

- Application:

Graph Signal Processing

- Signal processing on graphs deals with defining a set of tools to **analyze the data** $\mathbf{f}$ residing on nodes $v_i \in \mathbb{V}$ of the graph.

- Application:
  - **Learning graph** from data:

Graph Signal Processing

- Signal processing on graphs deals with defining a set of tools to **analyze the data** $\mathbf{f}$ residing on nodes $v_i \in \mathbb{V}$ of the graph.

- Application:
  - **Learning graph** from data:
    - E.g. Smoothness.

Graph Signal Processing

- Signal processing on graphs deals with defining a set of tools to **analyze the data** $\mathbf{f}$ residing on nodes $v_i \in \mathrm{V}$ of the graph.

- Application:
  - **Learning graph** from data:
    - E.g. Smoothness.
  - **Filtering** data on graphs:

Graph Signal Processing

- Signal processing on graphs deals with defining a set of tools to **analyze the data** $\mathbf{f}$ residing on nodes $v_i \in \mathrm{V}$ of the graph.

- Application:
  - **Learning graph** from data:
    - E.g. Smoothness.
  - **Filtering** data on graphs:
    - E.g. denoising, inpainting, coarsening.

Graph Signal Processing

- Signal processing on graphs deals with defining a set of tools to **analyze the data** $\mathbf{f}$ residing on nodes $v_i \in \mathrm{V}$ of the graph.

- Application:
  - **Learning graph** from data:
    - E.g. Smoothness.
  - **Filtering** data on graphs:
    - E.g. denoising, inpainting, coarsening.

- The concept of signal processing on graphs can define **convolutions** on graphs from a **spectral perspective**.

Convolution

- Classical **convolution** operation on the continuous Euclidean space is defined as

$$(f * h)(t) = \int_{\mathbb{R}} f(\tau)h(t-\tau)d\tau$$

where f and h are two functions.

Convolution

- Classical **convolution** operation on the continuous Euclidean space is defined as

$$(f * h)(t) = \int_{\mathbb{R}} f(\tau)h(t-\tau)d\tau$$

  where f and h are two functions.

- An alternative way is to perform convolution **in the Fourier** domain using Hadamard product.

Convolution

- Classical **convolution** operation on the continuous Euclidean space is defined as

$$(f * h)(t) = \int_{\mathbb{R}} f(\tau)h(t-\tau)d\tau$$

  where f and h are two functions.

- An alternative way is to perform convolution **in the Fourier** domain using Hadamard product.

- To perform this, one

Convolution

- Classical **convolution** operation on the continuous Euclidean space is defined as

$$(f * h)(t) = \int_{\mathbb{R}} f(\tau)h(t-\tau)d\tau$$

where f and h are two functions.

- An alternative way is to perform convolution **in the Fourier** domain using Hadamard product.
- To perform this, one
  - **Maps** the filter and the signal to the Fourier domain

Convolution

- Classical **convolution** operation on the continuous Euclidean space is defined as

$$(f * h)(t) = \int_{\mathbb{R}} f(\tau)h(t-\tau)d\tau$$

  where f and h are two functions.

- An alternative way is to perform convolution **in the Fourier** domain using Hadamard product.

- To perform this, one
  - **Maps** the filter and the signal to the Fourier domain
  - **Performs** the convolution in the Fourier domain

Convolution

- Classical **convolution** operation on the continuous Euclidean space is defined as

$$(f * h)(t) = \int_{\mathbb{R}} f(\tau)h(t-\tau)d\tau$$

where f and h are two functions.

- An alternative way is to perform convolution **in the Fourier** domain using Hadamard product.
- To perform this, one
  - **Maps** the filter and the signal to the Fourier domain
  - **Performs** the convolution in the Fourier domain
  - **Transforms** the result **back** to the Euclidean space.

Fourier transform

- Thus, in the first step, we need to define a **map** to the Fourier domain.

Fourier transform

- Thus, in the first step, we need to define a **map** to the Fourier domain.
- **Fourier transform** takes a data residing on the Euclidean space and maps it to the Fourier domain.
- This transform is performed by **expanding** the data in Fourier bases.

Fourier transform

- Thus, in the first step, we need to define a **map** to the Fourier domain.
- **Fourier transform** takes a data residing on the Euclidean space and maps it to the Fourier domain.
- This transform is performed by **expanding** the data in Fourier bases.
- Mathematically put

$$\begin{aligned}\mathcal{F}(f(t)) = \hat{f}(s) &:= \langle f, e^{2\pi ist} \rangle \\ &= \int_{\mathbb{R}} f(t) e^{-2\pi ist} dt\end{aligned}$$

where $e^{2\pi ist}$ represents the **Fourier bases**.

Inverse Fourier transform

- In the last step, we need to **project** the convolved data to the **original space**.
- This can be done using the **inverse Fourier transform**.

Inverse Fourier transform

- In the last step, we need to **project** the convolved data to the **original space**.
- This can be done using the **inverse Fourier transform**.
- Given the Fourier coefficient of a function, one can reconstruct the original data by **projecting** the resulted Fourier coefficients back to the Euclidean domain using inverse Fourier transform.
- **Inverse Fourier transform** is formulated as

$$\mathcal{F}^{-1}(\hat{f}(s)) = f(t) := \int_{\mathbb{R}} \hat{f}(s) e^{2\pi i s t} ds$$

Convolution

- As discussed earlier, we can use this notion to define **convolution**.
- Convolution of the two signals in the Fourier domain is represented by **element-wise** (Hadamard) **product**.

$$\mathcal{F}\left((f * h)(t)\right) = \hat{f}(s) \odot \hat{h}(s)$$

Convolution

- As discussed earlier, we can use this notion to define **convolution**.
- Convolution of the two signals in the Fourier domain is represented by **element-wise** (Hadamard) **product**.

$$\mathcal{F}\left((f * h)(t)\right) = \hat{f}(s) \odot \hat{h}(s)$$

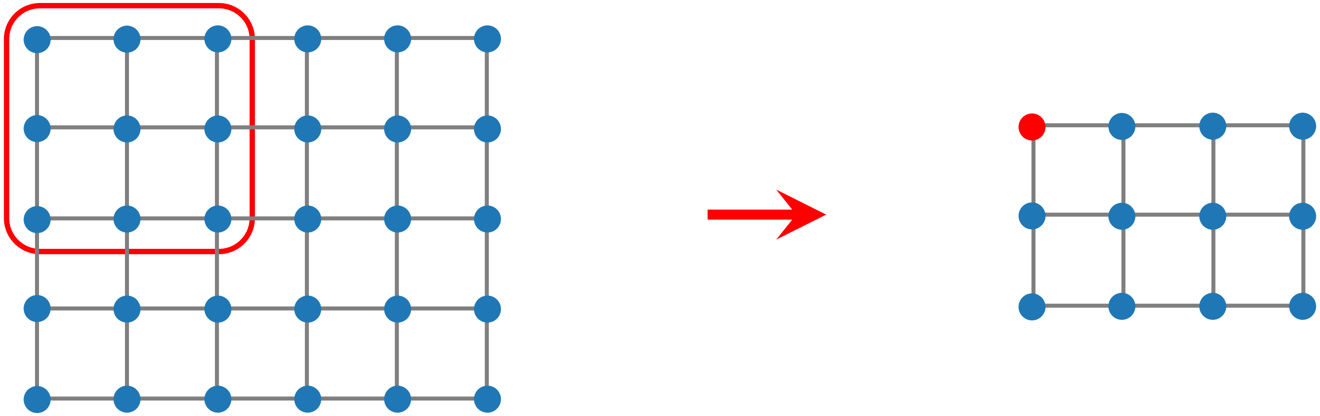
- Using **Fourier transform**, we can define convolution as

$$\begin{aligned}(f * h)(t) &= \mathcal{F}^{-1}\left(\mathcal{F}\left(f(t)\right) \odot \mathcal{F}\left(h(t)\right)\right) \\ &= \mathcal{F}^{-1}\left(\hat{f}(s) \odot \hat{h}(s)\right)\end{aligned}$$

- This operation filters the signal f using the filter h, such that it **amplifies** or **reduces** the contribution of some basis functions.

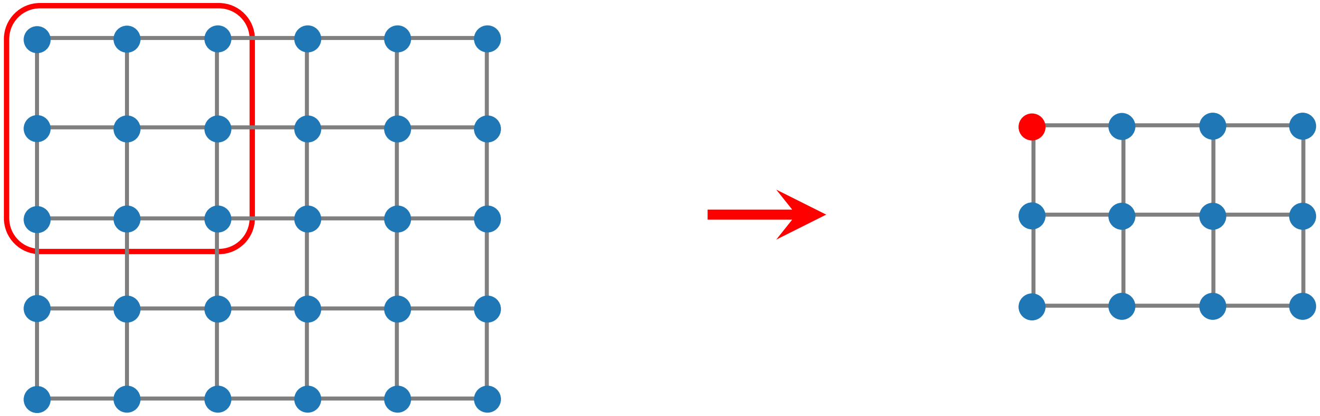
Convolution

- Convolutional neural networks are one of the most successful deep learning models, that use **convolution** operation.



Convolution

- Convolutional neural networks are one of the most successful deep learning models, that use **convolution** operation.
- CNN models operate on data defined over **discrete grids**.
- Therefore, the transformations above need to be reformulated for the data defined on **discrete** mesh.



Discrete Fourier transform

- For the case of a finite 1D grid $t \in \{0, \ldots N-1\}$, the **Fourier transform** is derived as

$$\hat{\mathbf{f}}_\ell = \frac{1}{\sqrt{N}} \sum_{t=0}^{N-1} \mathbf{f}_t e^{-i\frac{2\pi}{N}kt}$$

where $\hat{\mathbf{f}}_\ell$ are Fourier **coefficients** of the signal.

Discrete Fourier transform

- For the case of a finite 1D grid $t \in \{0, \dots N-1\}$, the **Fourier transform** is derived as

$$\hat{\mathbf{f}}_\ell = \frac{1}{\sqrt{N}} \sum_{t=0}^{N-1} \mathbf{f}_t e^{-i \frac{2\pi}{N} kt}$$

where $\hat{\mathbf{f}}_\ell$ are Fourier **coefficients** of the signal.

- In essence, this decomposition represents the signal $\mathbf{f}$ as **weighted sum** of the Fourier bases.

Discrete Fourier transform

- For the case of a finite 1D grid $t \in \{0, \dots N-1\}$, the **Fourier transform** is derived as

$$\hat{\mathbf{f}}_\ell = \frac{1}{\sqrt{N}} \sum_{t=0}^{N-1} \mathbf{f}_t e^{-i\frac{2\pi}{N}kt}$$

  where $\hat{\mathbf{f}}_\ell$ are Fourier **coefficients** of the signal.

- In essence, this decomposition represents the signal $\mathbf{f}$ as **weighted sum** of the Fourier bases.

- Analogously, the **inverse Fourier transform** is defined as

$$\mathbf{f}_t = \frac{1}{\sqrt{N}} \sum_{\ell=0}^{N-1} \hat{\mathbf{f}}_\ell e^{i\frac{2\pi}{N}\ell t}$$

Shift and Difference

- **Convolution** operation is characterized by its **shift** equivariance and **difference equivariance** properties.

Shift and Difference

- **Convolution** operation is characterized by its **shift** equivariance and **difference equivariance** properties.
- The **shift** operation is defined as

$$\mathcal{T}_a f(t) = f(t - a)$$

Shift and Difference

- **Convolution** operation is characterized by its **shift** equivariance and **difference equivariance** properties.
- The **shift** operation is defined as

$$\mathcal{T}_a f(t) = f(t-a)$$

- **Shift equivariance** implies that

$$f(t+a) * g(t) = f(t) * g(t+a) = (f*g)(t+a)$$

- In other words, as a result of shift equivariance of the convolution operation, convolving a translated signal is equal to translating a convolved signal.

Shift and Difference

- **Convolution** operation is characterized by its **shift** equivariance and **difference equivariance** properties.
- The **difference** operation is defined as

$$\Delta f(t) = f(t+1) - f(t)$$

Shift and Difference

- **Convolution** operation is characterized by its **shift** equivariance and **difference equivariance** properties.
- The **difference** operation is defined as

$$\Delta f(t) = f(t+1) - f(t)$$

- Difference **equivariance** conveys that

$$\Delta f(t) * g(t) = f(t) * \Delta g(t) = \Delta (f * g)(t)$$

Shift and Difference

- **Convolution** operation is characterized by its **shift** equivariance and **difference equivariance** properties.
- The **difference** operation is defined as

$$\Delta f(t) = f(t+1) - f(t)$$

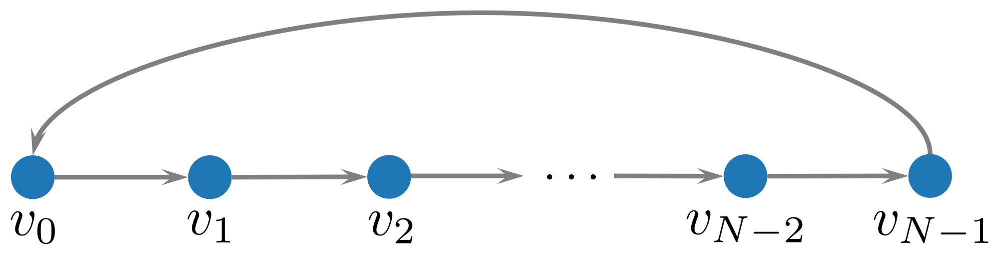
- Difference **equivariance** conveys that

$$\Delta f(t) * g(t) = f(t) * \Delta g(t) = \Delta (f * g)(t)$$

- To motivate these definitions, we rely on the **ring graphs**.

Ring Graph

- A **cycle** graph, **circular** graph, or **ring graph** is a graph consisting of a single cycle.
- All nodes of the graph are connected through this closed cycle.



Ring Graph

- A **cycle** graph, **circular** graph, or **ring graph** is a graph consisting of a single cycle.
- All nodes of the graph are connected through this closed cycle.
- We can represent the **signal f** defined on **discrete time** domain

$$f\left(t_0\right), f\left(t_2\right), \ldots, f\left(t_{N-1}\right)$$

  using ring graphs.

Ring Graph

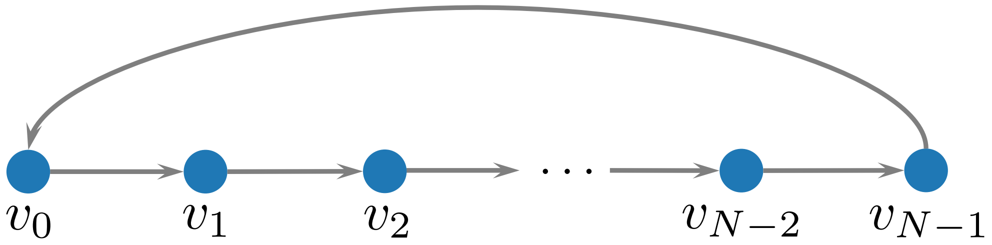
- A **cycle** graph, **circular** graph, or **ring graph** is a graph consisting of a single cycle.
- All nodes of the graph are connected through this closed cycle.
- We can represent the **signal f** defined on **discrete time** domain

$$f\left(t_0\right), f\left(t_2\right), \ldots, f\left(t_{N-1}\right)$$

  using ring graphs.
- In this representation, each **time** t is represented by **node** v_t of the ring graph
- Thus, function f's **value at time** t is represented by signal value $f(t)$ residing on **node** v_t.

Ring Graph

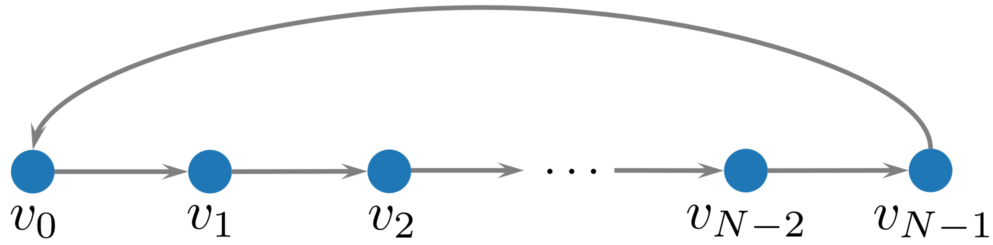
- In other words, $\mathbf{f}$ is represented on the graph, such that $\mathbf{f}_t$ is signal value corresponding to node v_t.



Shift and Difference on Chain Graph

- In other words, $\mathbf{f}$ is represented on the graph, such that $\mathbf{f}_t$ is signal value corresponding to node v_t.
- Using this representation, the shift operator is represented by the **circulant adjacency** matrix

$$[\mathbf{A}_c]_{i,j} = \begin{cases} 1 & \text{if } j = (i+1) \mod N \\ 0 & \text{otherwise} \end{cases}$$



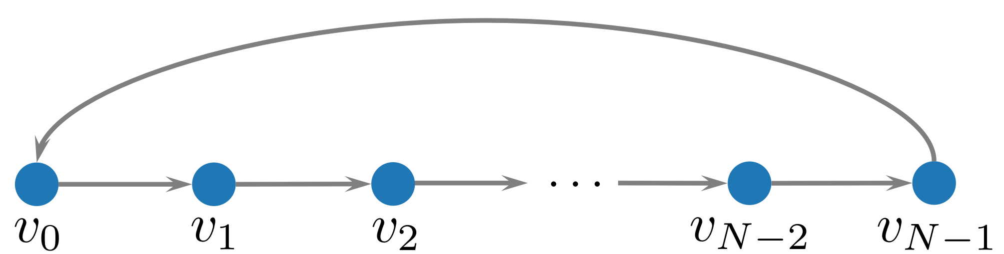
Shift and Difference on Chain Graph

- In other words, $\mathbf{f}$ is represented on the graph, such that $\mathbf{f}_t$ is signal value corresponding to node v_t.
- Using this representation, the shift operator is represented by the **circulant adjacency** matrix

$$[\mathbf{A}_c]_{i,j} = \begin{cases} 1 & \text{if } j = (i+1) \mod N \\ 0 & \text{otherwise} \end{cases}$$

- Then, **time shift** is performed using

$$[\mathbf{A}_c \mathbf{f}]_t = \mathbf{f}_{(t+1) \mod N}$$



Shift and Difference on Chain Graph

- In other words, $\mathbf{f}$ is represented on the graph, such that $\mathbf{f}_t$ is signal value corresponding to node v_t.
- Using this representation, the shift operator is represented by the **circulant adjacency** matrix

$$[\mathbf{A}_c]_{i,j} = \begin{cases} 1 & \text{if } j = (i+1) \mod N \\ 0 & \text{otherwise} \end{cases}$$

- Then, **time shift** is performed using

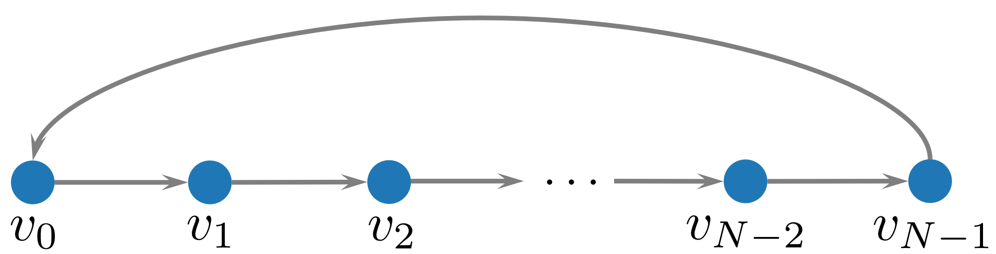
$$[\mathbf{A}_c \mathbf{f}]_t = \mathbf{f}_{(t+1) \mod N}$$

- Intuitively, applying **shift** in signal $\mathbf{f}$ residing on graph G **propagates information** from one node to the other.

Shift and Difference on Chain Graph

- On the other hand, the difference operator is defined using the **circulant Laplace** matrix

$$\mathbf{L}_c = \mathbf{I} - \mathbf{A}_c$$



Shift and Difference on Chain Graph

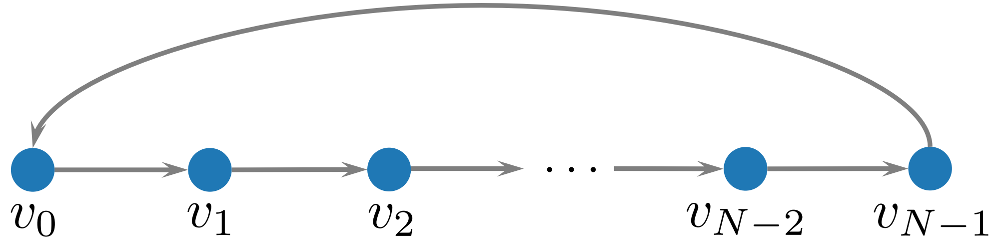
- On the other hand, the difference operator is defined using the **circulant Laplace** matrix

$$\mathbf{L}_c = \mathbf{I} - \mathbf{A}_c$$

- Using $\mathbf{L}_c$, one can apply **difference operation** on signal $\mathbf{f}$ as

$$[\mathbf{L}_c \mathbf{f}]_t = \mathbf{f}_t - \mathbf{f}_{(t+1) \bmod N}$$

- In this equation, applying the difference operator computes the difference between $\mathbf{f}$ at node v_t with its neighbors.



Convolution on Chain Graph

- Given a convolution matrix $\boldsymbol{Q}$ of filter function h, we can perform **convolution** on signals on **graph** using the matrix product

$$\begin{aligned}(f \star h)(\mathbf{t}) &= \sum_{\tau=0}^{N-1} f(\tau)h(\tau - t) \\ &= \mathbf{Q}_h \mathbf{f}\end{aligned}$$

Convolution on Chain Graph

- Given a convolution matrix $\boldsymbol{Q}$ of filter function h, we can perform **convolution** on signals on **graph** using the matrix product

$$\begin{aligned}(f \star h)(\mathbf{t}) &= \sum_{\tau=0}^{N-1} f(\tau)h(\tau - t) \\ &= \mathbf{Q}_h \mathbf{f}\end{aligned}$$

- The **convolution matrix** $\boldsymbol{Q}_h$ though, has to follow shift and difference equivariance.

Convolution on Chain Graph

- Given a convolution matrix $\boldsymbol{Q}$ of filter function h, we can perform **convolution** on signals on **graph** using the matrix product

$$(f \star h)(\mathbf{t}) = \sum_{\tau=0}^{N-1} f(\tau)h(\tau - t)$$
$$= \mathbf{Q}_h \mathbf{f}$$

- The **convolution matrix $\boldsymbol{Q}_h$** though, has to follow shift and difference equivariance.

- In the matrix form, we can represent that as

  - **Shift equivariance**

$$\mathbf{A}_c \mathbf{Q}_h = \mathbf{Q}_h \mathbf{A}_c$$

  - **Difference equivariance**

$$\mathbf{L}_c \mathbf{Q}_h = \mathbf{Q}_h \mathbf{L}_c$$

Convolution on Chain Graph

- In other words, $\boldsymbol{Q}_h$ should be **commutative** with respect to $\mathbf{A}_c$ and $\mathbf{L}_c$.
- One can show that defining the matrix $\boldsymbol{Q}_h$ as a **polynomial function** of the **circular adjacency** $\mathbf{A}_c$ matrix would satisfy shift and difference equivariance.

Convolution on Chain Graph

- In other words, $\boldsymbol{Q}_h$ should be **commutative** with respect to $\mathbf{A}_c$ and $\mathbf{L}_c$.
- One can show that defining the matrix $\boldsymbol{Q}_h$ as a **polynomial function** of the **circular adjacency** $\mathbf{A}_c$ matrix would satisfy shift and difference equivariance.
- In other words,

$$\mathbf{Q}_h = p_N\left(\mathbf{A}_c\right) = \sum_{i=0}^{N-1} \alpha_i \mathbf{A}_c^i$$

Convolution on Chain Graph

- In other words, $\boldsymbol{Q}_h$ should be **commutative** with respect to $\mathbf{A}_c$ and $\mathbf{L}_c$.
- One can show that defining the matrix $\boldsymbol{Q}_h$ as a **polynomial function** of the **circular adjacency** $\mathbf{A}_c$ matrix would satisfy shift and difference equivariance.
- In other words,

$$\mathbf{Q}_h = p_N\left(\mathbf{A}_c\right) = \sum_{i=0}^{N-1} \alpha_i \mathbf{A}_c^i$$

- One can generalize the idea defined on cycle graph to perform **convolution** on **arbitrary graphs**.

Spatial Graph Convolution

- In other words, we define **convolution filter matrix** $\boldsymbol{Q}_h$ for a general **graph** such that it **commutes** with respect to adjacency or Laplacian.

Spatial Graph Convolution

- In other words, we define **convolution filter matrix** $\boldsymbol{Q}_h$ for a general **graph** such that it **commutes** with respect to adjacency or Laplacian.
- Using the adjacency $\mathbf{A}$, we can define convolution filter matrices as

$$\mathbf{Q}_h = \alpha_0 \mathbf{I} + \alpha_1 \mathbf{A} + \alpha_2 \mathbf{A}^2 + \ldots + \alpha_N \mathbf{A}^N$$

Spatial Graph Convolution

- In other words, we define **convolution filter matrix** $\boldsymbol{Q}_h$ for a general **graph** such that it **commutes** with respect to adjacency or Laplacian.
- Using the adjacency $\boldsymbol{A}$, we can define convolution filter matrices as

$$\mathbf{Q}_h = \alpha_0 \mathbf{I} + \alpha_1 \mathbf{A} + \alpha_2 \mathbf{A}^2 + \ldots + \alpha_N \mathbf{A}^N$$

- We recall that $\boldsymbol{A}^n$ represents neighbors of node v_i that are within $\boldsymbol{n}$ **−hop** distance from the node.

Spatial Graph Convolution

- In other words, we define **convolution filter matrix** $\boldsymbol{Q}_h$ for a general **graph** such that it **commutes** with respect to adjacency or Laplacian.
- Using the adjacency $\mathbf{A}$, we can define convolution filter matrices as

$$\mathbf{Q}_h = \alpha_0 \mathbf{I} + \alpha_1 \mathbf{A} + \alpha_2 \mathbf{A}^2 + \ldots + \alpha_N \mathbf{A}^N$$

- We recall that $\mathbf{A}^n$ represents neighbors of node v_i that are within $\boldsymbol{n}$ **−hop** distance from the node.
- Therefore, performing convolution $\boldsymbol{Q}_h$ on signal $f \in \mathbb{R}^{|V|}$ in the **nodal domain** will collect information from an N −hop neighborhood of each node $v_i \in V$.

Spatial Graph Convolution

- Thus, $\boldsymbol{Q}_h\mathbf{f}$ contains information from different $\boldsymbol{n}$ **−hop** neighborhoods,

$$\mathbf{Q}_h\mathbf{f} = \alpha_0\mathbf{I}\mathbf{f} + \alpha_1\mathbf{A}\mathbf{f} + \alpha_2\mathbf{A}^2\mathbf{f} + \ldots + \alpha_N\mathbf{A}^N\mathbf{f}$$

where parameters α_n regulates the **influence** if different n −hop neighbors.

Spatial Graph Convolution

- Thus, $\boldsymbol{Q}_h\mathbf{f}$ contains information from different $\boldsymbol{n}$ **−hop** neighborhoods,

$$\mathbf{Q}_h\mathbf{f} = \alpha_0\mathbf{I}\mathbf{f} + \alpha_1\mathbf{A}\mathbf{f} + \alpha_2\mathbf{A}^2\mathbf{f} + \ldots + \alpha_N\mathbf{A}^N\mathbf{f}$$

  where parameters α_n regulates the **influence** if different n −hop neighbors.

- While $\boldsymbol{Q}_h$ defined for chain graphs are commutative with both adjacency and Laplacian matrices, this **does not** always **extend** to general graphs.

Spatial Graph Convolution

- Thus, $\boldsymbol{Q}_h\mathbf{f}$ contains information from different $\boldsymbol{n}$ **−hop** neighborhoods,

$$\mathbf{Q}_h\mathbf{f} = \alpha_0\mathbf{I}\mathbf{f} + \alpha_1\mathbf{A}\mathbf{f} + \alpha_2\mathbf{A}^2\mathbf{f} + \ldots + \alpha_N\mathbf{A}^N\mathbf{f}$$

  where parameters α_n regulates the **influence** if different n −hop neighbors.

- While $\boldsymbol{Q}_h$ defined for chain graphs are commutative with both adjacency and Laplacian matrices, this **does not** always **extend** to general graphs.

- In addition to the Laplace and adjacency matrices, one can use **normalized variants** of them to define the convolution matrix,

$$\mathbf{L}_{\text{sym}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} \qquad \mathbf{A}_{\text{sym}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$$

Spatial Graph Convolution

- Choosing **which matri**x to use is a matter of **trade-off** between the pros and cons of each method.

Spatial Graph Convolution

- Choosing **which matri**x to use is a matter of **trade-off** between the pros and cons of each method.
  - We can define **bounds** for the spectrum of the normalized Laplacian and Adjacency matrixes, which results in numerical stability.

Spatial Graph Convolution

- Choosing **which matrix** to use is a matter of **trade-off** between the pros and cons of each method.
  - We can define **bounds** for the spectrum of the normalized Laplacian and Adjacency matrixes, which results in numerical stability.
  - Another plausible property of the normalized variants is that **commutativity** with respect to one will indicate commutativity with respect to the other one.

Graph Fourier Transform

- So far, we have relied on the notion of **convolution** in the **nodal domain**.
- One can leverage tools from graph signal processing to define convolution in the **graph spectral domain**.

Graph Fourier Transform

- So far, we have relied on the notion of **convolution** in the **nodal domain**.
- One can leverage tools from graph signal processing to define convolution in the **graph spectral domain**.
- To do so, we need to redefine the **building block** of signal processing tools, namely Fourier transform, on graph domain.

Graph Fourier Transform

- So far, we have relied on the notion of **convolution** in the **nodal domain**.
- One can leverage tools from graph signal processing to define convolution in the **graph spectral domain**.
- To do so, we need to redefine the **building block** of signal processing tools, namely Fourier transform, on graph domain.
- In the earlier slides, we presented **Fourier transform** as

$$\mathcal{F}(f(t)) = \int_{\mathbb{R}} f(t) e^{-2\pi i s t} dt$$

where we expand the signal in Fourier **bases** $e^{2\pi i s t}$.

Graph Fourier Transform

- We recall that the **Laplace operator** is defined as

$$\Delta f(t) = \nabla^2 f(t) = \frac{\partial^2 f}{\partial t^2}$$

Graph Fourier Transform

- We recall that the **Laplace operator** is defined as

$$\Delta f(t) = \nabla^2 f(t) = \frac{\partial^2 f}{\partial t^2}$$

- We can show that **bases** of the Fourier domain are **eigenfunctions** of the Laplace operator

$$-\Delta\left(e^{2\pi i s t}\right) = -\frac{\partial^2}{\partial t^2}\left(e^{2\pi i s t}\right) = (2\pi s)^2 e^{2\pi i s t}$$

Graph Fourier Transform

- We recall that the **Laplace operator** is defined as

$$\Delta f(t) = \nabla^2 f(t) = \frac{\partial^2 f}{\partial t^2}$$

- We can show that **bases** of the Fourier domain are **eigenfunctions** of the Laplace operator

$$-\Delta \left(e^{2\pi i s t}\right) = -\frac{\partial^2}{\partial t^2}\left(e^{2\pi i s t}\right) = (2\pi s)^2 e^{2\pi i s t}$$

- The corresponding eigenvalues $\{(2\pi s)^2\}_{s\in\mathbb{R}}$ contain a notion of **frequency**, where:

Graph Fourier Transform

- We recall that the **Laplace operator** is defined as

$$\Delta f(t) = \nabla^2 f(t) = \frac{\partial^2 f}{\partial t^2}$$

- We can show that **bases** of the Fourier domain are **eigenfunctions** of the Laplace operator

$$-\Delta\left(e^{2\pi i s t}\right) = -\frac{\partial^2}{\partial t^2}\left(e^{2\pi i s t}\right) = (2\pi s)^2 e^{2\pi i s t}$$

- The corresponding eigenvalues $\{(2\pi s)^2\}_{s\in\mathbb{R}}$ contain a notion of **frequency**, where:
  - For s close to zero, the corresponding basis component **oscillates slowly** (smooth).

Graph Fourier Transform

- We recall that the **Laplace operator** is defined as

$$\Delta f(t) = \nabla^2 f(t) = \frac{\partial^2 f}{\partial t^2}$$

- We can show that **bases** of the Fourier domain are **eigenfunctions** of the Laplace operator

$$-\Delta\left(e^{2\pi i s t}\right) = -\frac{\partial^2}{\partial t^2}\left(e^{2\pi i s t}\right) = (2\pi s)^2 e^{2\pi i s t}$$

- The corresponding eigenvalues $\{(2\pi s)^2\}_{s\in\mathbb{R}}$ contain a notion of **frequency**, where:
  - For s close to zero, the corresponding basis component **oscillates slowly** (smooth).
  - For s far larger than zero, the associated exponential component **oscillates more rapidly**.

Graph Fourier Transform

- Recall from earlier lecture that **Laplacian matrix** is analogue of the multivariate **Laplace operator** in the graph domain.
- One can exploit this analogy and define the **Fourier bases** for the **graph domain**.

Graph Fourier Transform

- Recall from earlier lecture that **Laplacian matrix** is analogue of the multivariate **Laplace operator** in the graph domain.
- One can exploit this analogy and define the **Fourier bases** for the **graph domain**.
- Since the Laplace matrix is a positive semi-definite matrix, it has a set of orthonormal **eigenvectors** $\{u_\ell\}_{\ell=0,\ldots,N-1}$ and associated real non-negative **eigenvalues** $\{\lambda_\ell\}_{\ell=0,\ldots,N-1}$ that satisfy

$$\mathbf{L}u_\ell = \lambda_\ell u_\ell$$

for $\ell = 0, \ldots, N-1$.

Graph Fourier Transform

- In the matrix form

$$\mathbf{L} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$$

where $\mathbf{U}$ is the **eigenvector matrix** and $\boldsymbol{\Lambda}$ is a **diagonal matrix** with eigenvalues λ_ℓ of $\mathbf{L}$ as its diagonal elements.

- The set of eigenvalues

$$\sigma(\mathbf{L}) = \{\lambda_0, \lambda_1, \ldots, \lambda_{N-1}\}$$

is the **spectrum** of graph G, where we assume

$$0 = \lambda_0 < \lambda_1 \leq \cdots \leq \lambda_{N-1} \leq \lambda_{\max}$$

Graph Fourier Transform

- Using the eigenvectors U as Fourier bases, we can define the **Fourier transform** of a signal $\mathbf{f} \in \mathbb{R}^N$ in the **graph domain** as

$$\hat{f}(\lambda_\ell) := \langle f, u_\ell \rangle = \sum_{i=1}^{N} f(i) u_\ell(i)$$

- In the matrix form

$$\hat{\mathbf{f}} = \mathbf{U}^\top \mathbf{f}$$

Graph Fourier Transform

- Using the eigenvectors U as Fourier bases, we can define the **Fourier transform** of a signal $\mathbf{f} \in \mathbb{R}^N$ in the **graph domain** as

$$\hat{f}\left(\lambda_\ell\right) := \langle f, u_\ell \rangle = \sum_{i=1}^{N} f(i) u_\ell(i)$$

- In the matrix form

$$\hat{\mathbf{f}} = \mathbf{U}^\top \mathbf{f}$$

- Analogously, one can define the **inverse graph Fourier transform** as

$$f(i) = \sum_{\ell=0}^{N-1} \hat{f}\left(\lambda_\ell\right) u_\ell(i)$$

- In the matrix notation

$$\mathbf{f} = \mathbf{U}\hat{\mathbf{f}}$$

Graph Fourier Transform

- In this analogy, eigenvalues λ_ℓ present a notion of **frequency** for graphs, where

Graph Fourier Transform

- In this analogy, eigenvalues λ_ℓ present a notion of **frequency** for graphs, where
  - The eigenvector u_0 associated with $\lambda_\ell = 0$ is constant throughout the graph.

$$\boldsymbol{u}_0 = \frac{1}{N}\vec{\mathbf{1}}$$

Graph Fourier Transform

- In this analogy, eigenvalues λ_ℓ present a notion of **frequency** for graphs, where
  - The eigenvector u_0 associated with $\lambda_\ell = 0$ is constant throughout the graph.

$$\boldsymbol{u}_0 = \frac{1}{N}\overrightarrow{\mathbf{1}}$$

  - The eigenvector u_0 associated with smaller eigenvalues λ_ℓ change slowly across the graph.
    - E.g. Interests of friends.

Graph Fourier Transform

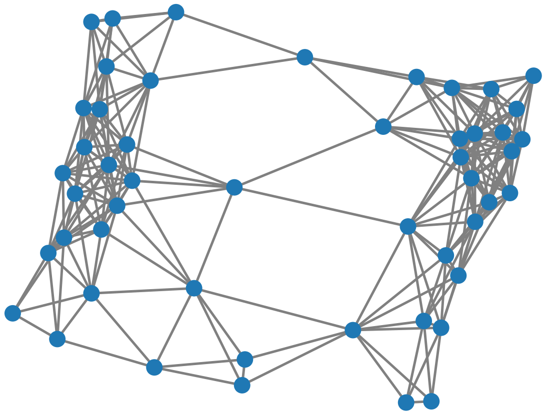
- In this analogy, eigenvalues λ_ℓ present a notion of **frequency** for graphs, where
  - The eigenvector u_0 associated with $\lambda_\ell = 0$ is constant throughout the graph.

$$\boldsymbol{u}_0 = \frac{1}{N}\overrightarrow{\mathbf{1}}$$

  - The eigenvector u_0 associated with smaller eigenvalues λ_ℓ change slowly across the graph.
    - E.g. Interests of friends.
  - The eigenvector u_0 associated with larger eigenvalues λ_ℓ change rapidly across the graph.

Graph Fourier Transform

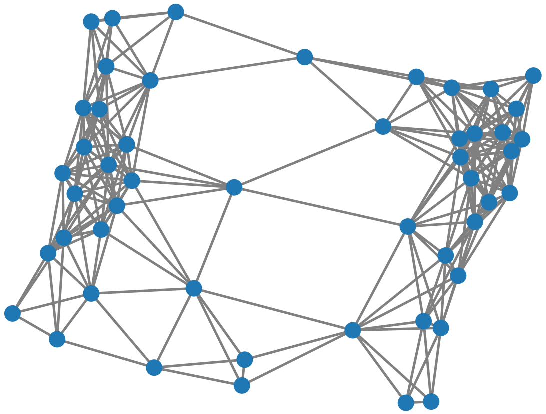
- Consider the graph G below:



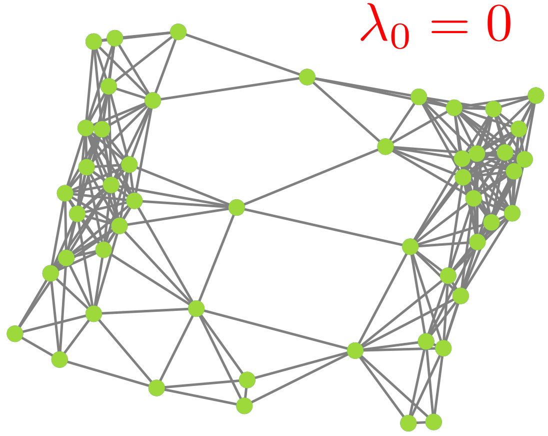
- We can visualize the **graph Fourier bases** for G as:

Graph Fourier Transform

- Consider the graph G below:

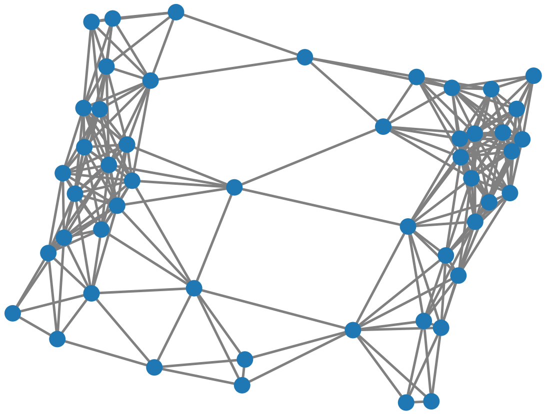


- We can visualize the **graph Fourier bases** for G as:

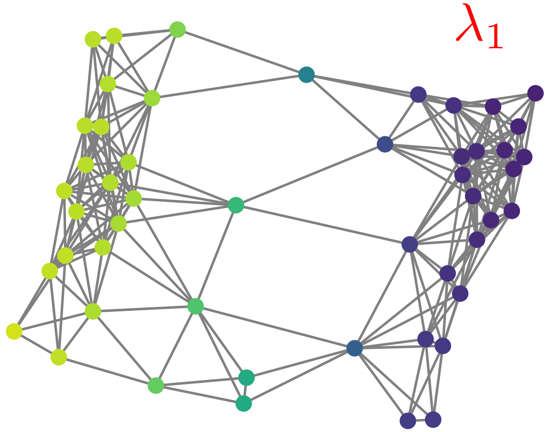


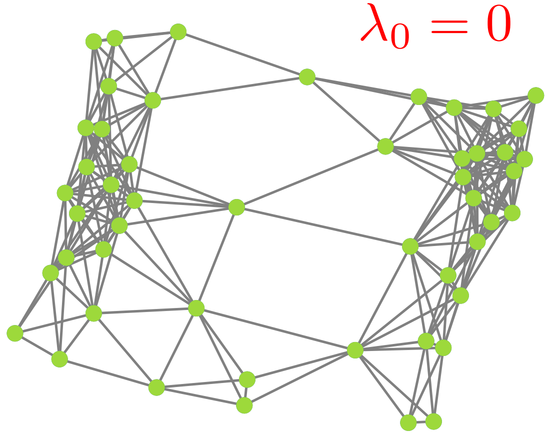
Graph Fourier Transform

- Consider the graph G below:



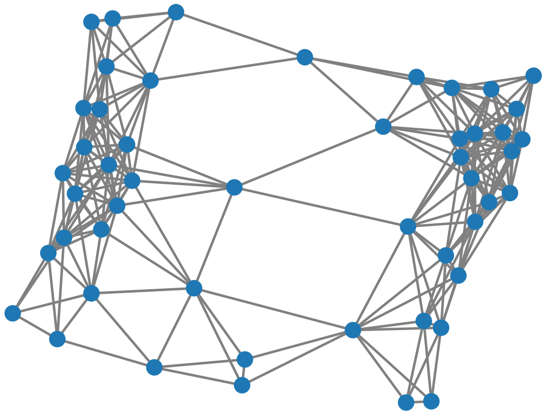
- We can visualize the **graph Fourier bases** for G as:



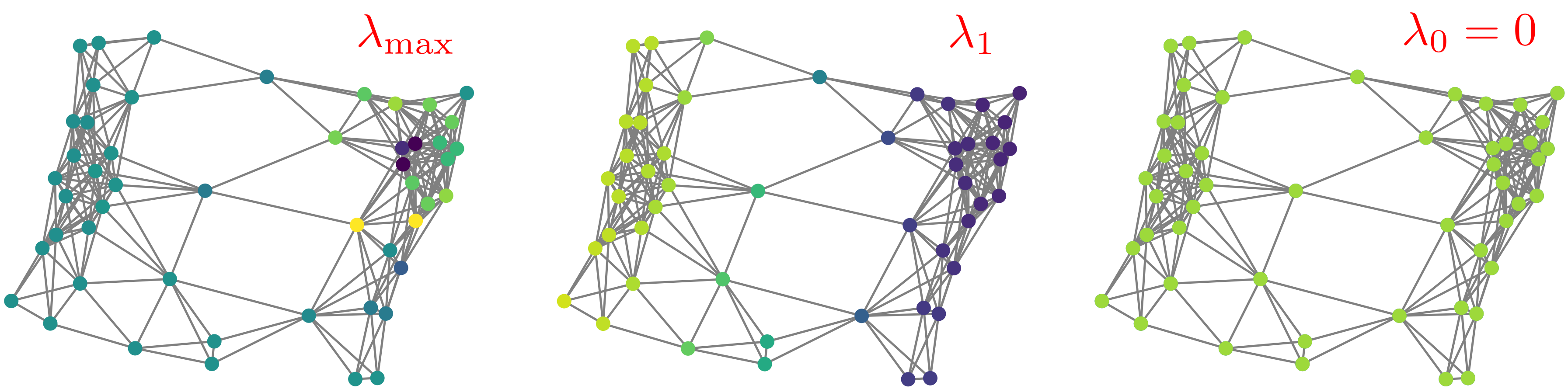


Graph Fourier Transform

- Consider the graph G below:



- We can visualize the **graph Fourier bases** for G as:



Graph Fourier Transform

- So far, we have used **graph Laplacian matrix** to construct graph Fourier bases.
- While this is the most popular approach, **other matrices** have been adapted to define the notion of **graph Fourier domain**.

Graph Fourier Transform

- So far, we have used **graph Laplacian matrix** to construct graph Fourier bases.
- While this is the most popular approach, **other matrices** have been adapted to define the notion of **graph Fourier domain**.
- One alternative is **normalized** graph Laplacian $\tilde{\mathbf{L}}$, in which elements $\mathbf{A}_{ij}$ are normalized by $\frac{1}{\sqrt{d_i d_j}}$.
- However, the eigenvalues associated with the smallest eigenvalue $\tilde{\lambda}_0$ does **not** have **constant elements** anymore.

Graph Fourier Transform

- So far, we have used **graph Laplacian matrix** to construct graph Fourier bases.
- While this is the most popular approach, **other matrices** have been adapted to define the notion of **graph Fourier domain**.
- One alternative is **normalized** graph Laplacian $\tilde{\mathbf{L}}$, in which elements $\mathbf{A}_{ij}$ are normalized by $\frac{1}{\sqrt{d_i d_j}}$.
- However, the eigenvalues associated with the smallest eigenvalue $\tilde{\lambda}_0$ does **not** have **constant elements** anymore.
- Another approach is to use the **stochastic matrix**

$$P := \mathrm{D}^{-1}\mathbf{A}$$

where each element $\boldsymbol{P}_{ij}$ is the probability of going from v_i to v_j in one step.

Summary

- Signal Processing on Graphs
- Convolution
- Fourier transform
- Discrete Fourier transform
- Shift and Difference
- Chain Graph
- Shift and Difference on Chain Graph
- Convolution on Chain Graph
- Graph Convolution
- Graph Fourier Transform