GRAPH CONVOLUTION NETWORK

2022. 01. 25.

Hyunsoo, Yu

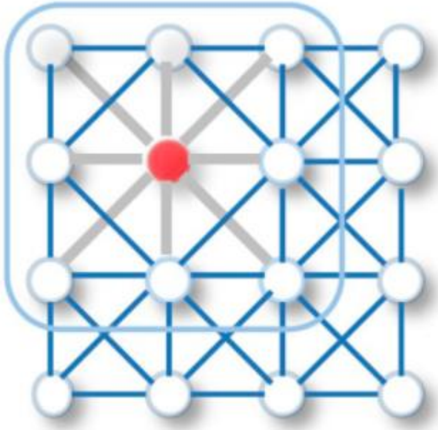
INDEX

1. **Background**
    1. Motivation
    2. Graph Laplacian
    3. Laplacian matrix
    4. Frourier transform
    5. Graph fourier transform
    6. Convolution theorem
2. **Spectral methods**
    1. Spectral CNN
    2. CHEBNET
    3. GCN
3. **Spatial Methods**
    1. NN4G
    2. GCN
    3. PGC
4. **Comparison**

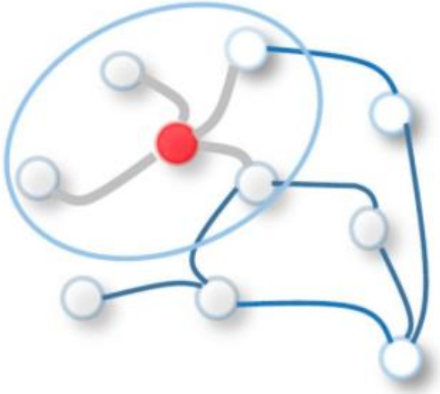
BACKGROUND
- MOTIVATION

- CNN feature
  - Local connectivity (Spatial feature)
  - Shared weights
  - Use of Multi-layer

- GNN feature
  - Local connectivity (Spatial feature)
  - Shared weights
  - Use of Multi-layer
    - Able to extract feature from far nodes which are connected indirectly

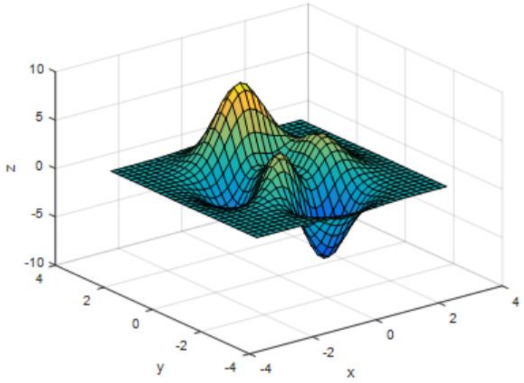


2D-Convolution



Graph Convolution

BACKGROUND FOR SPECTRAL GRAPH CONVOLUTION
- GRAPH LAPLACIAN



- Laplace operator : Differential operator (Divergence)

$$\triangle f = \nabla \cdot \nabla f = \nabla^2 f$$

$$\triangle = \nabla \bullet \nabla = \nabla^2 = \frac{\partial}{\partial x}\left(\frac{\partial}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\partial}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\partial}{\partial z}\right) = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

$$\nabla^2 f = \nabla \bullet (\nabla f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$$

Coordinate system

General

$$\triangle f = \nabla^2 f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial {x_i}^2}$$

- Graph Laplacian [discrete]

$$\triangle f(v_i) = Lf|_{v_i} = \sum_{v_j \; v_i} [f(v_i) - f(v_j)]$$

$x-h$ — x — $x+h$

$$\lim_{h \to 0} \left(\frac{\frac{f(x+h) - f(x)}{h} - \frac{f(x) - f(x-h)}{h}}{h} \right)$$

$$= \lim_{h \to 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

BACKGROUND FOR SPECTRAL GRAPH CONVOLUTION - GRAPH LAPLACIAN

$$\triangle f(v_i) = \sum_{j \in N_i} W_{ij}[f(v_i) - f(v_j)]$$

- Graph Laplacian [discrete]

$$\triangle f(v_i) = Lf|_{v_i} = \sum_{v_j \ v_i}[f(v_i) - f(v_j)]$$

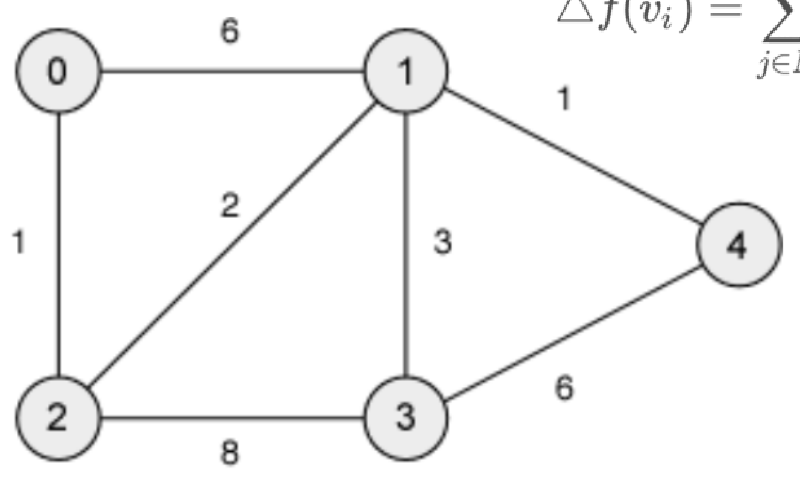
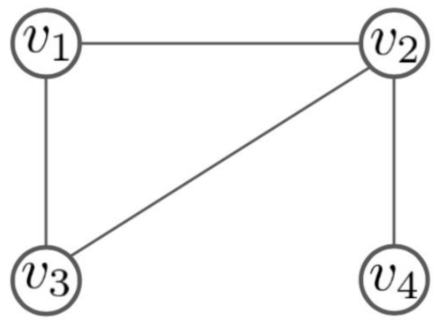


그림 11. weighted undirected graph

- Example



$$\triangle f(v_1) = 2f(v_1) - f(v_2) - f(v_3)$$

$$\triangle f(v_2) = 3f(v_2) - f(v_1) - f(v_3) - f(v_4)$$

$$\triangle f(v_3) = 2f(v_3) - f(v_1) - f(v_2)$$

$$\triangle f(v_4) = f(v_4) - f(v_2)$$

$$M = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} f(v_1) \\ f(v_2) \\ f(v_3) \\ f(v_4) \end{bmatrix}$$

Laplacian matrix

BACKGROUND FOR SPECTRAL GRAPH CONVOLUTION
- LAPLACIAN MATRIX

D A L

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

$$L = D - A$$

$$L_{i,j} = \begin{cases} deg(v_i) & if\ i = j \\ -1 & if\ i \neq j \\ 0 & otherwise \end{cases}$$

$$D_{i,j} = \begin{cases} deg(v_i) & if\ i = j \\ 0 & otherwise \end{cases}$$

- Normalized Laplacian matrix $L = I - D^{-1/2} A D^{-1/2}$
  - All the diagonal term would be 1 $L_{ij(i \neq j)} = -A_{ij} / \sqrt{degree(i)degree(j)}$
  - Able to do eigen-value decomposition
  - $L = U \Lambda U^T \quad (U^T U = I)$

BACKGROUND FOR SPECTRAL GRAPH CONVOLUTION
- FOURIER TRANSFORM

$$\hat{f}(\xi) = \int_{\mathbf{R}^d} f(x) e^{2\pi i x \xi} \, dx$$

- Interpret : how similar are f(x) and $e^{2\pi i x \xi}$

$e^{2\pi i x \xi} = cos(2\pi x \xi) + isin(2\pi x \xi)$: orthogonal basis function

- <Linear algebra>
  - The way to find orthonormal basis : **Eigen-value decomposition** (only real-symmetric matrix)

BACKGROUND FOR SPECTRAL GRAPH CONVOLUTION
- GRAPH FOURIER TRANSFORM

- Eigen-value decomposition of Laplacian matrix

$$\mathbf{L} = \mathbf{U}^{\intercal}\mathbf{\Lambda}\mathbf{U}$$

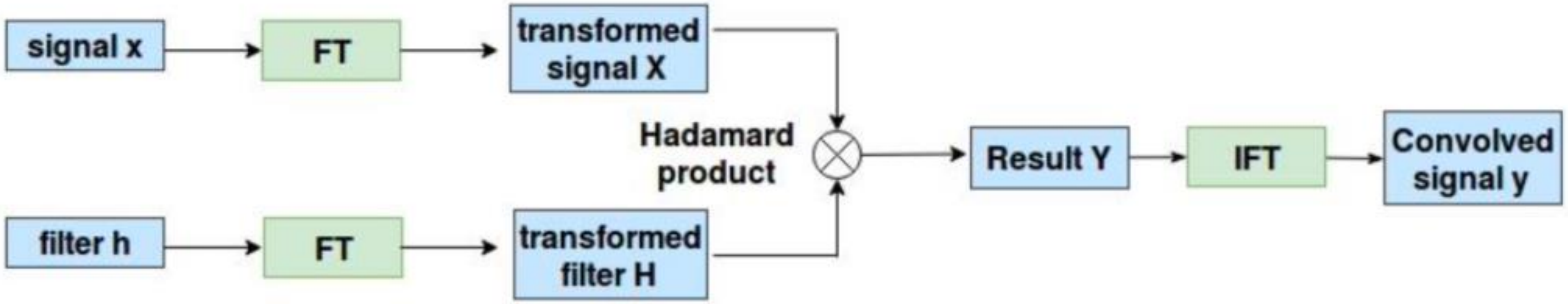
- Graph fourier transform

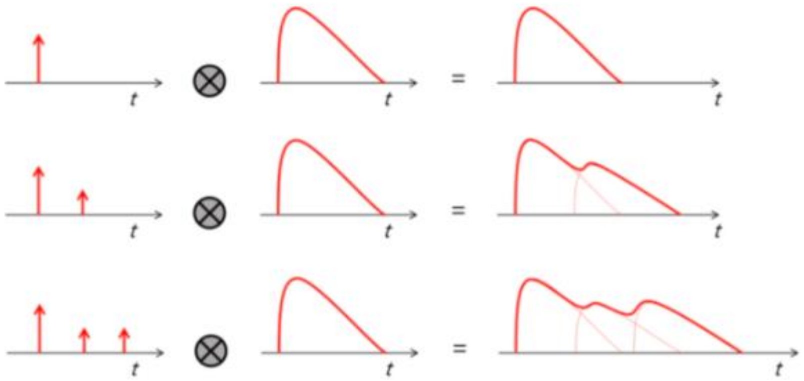
$$\mathcal{F}(\mathbf{x}) = \mathbf{U}^{\intercal}\mathbf{x}$$

- Inverse graph fourier transform

$$\mathcal{F}^{-1}(\hat{\mathbf{x}}) = \mathbf{U}\hat{\mathbf{x}}$$

BACKGROUND FOR SPECTRAL GRAPH CONVOLUTION
- CONVOLUTION THEOREM





$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

- Convolution theorem
  - Convolutional operation between signal **x(t)** and filter **h(t)** is the same as the product after converting to the frequency domain
  - **Convolution in spatial/time domain = multiplication in fourier domain**

$$\mathbf{x} * \mathbf{g} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}) \odot \mathcal{F}(\mathbf{g}))$$

1. SPECTRAL GRAPH CONVOLUTION

: a spectral representation of the graphs

- Filter learns with Laplacian eigenbasis depending on graph structure.

  : A model trained on a specific structure can not be applied a graph with a different structure

$$\mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}) \odot \mathcal{F}(\mathbf{g})) = \mathbf{U}(\mathbf{U}^{\intercal}\mathbf{x} \odot \mathbf{U}^{\intercal}\mathbf{g})$$

- If filter has diagonal factors only, ($\mathbf{g}_{\theta} = diag(\mathbf{U}^{\intercal}\mathbf{g})$)

$$\mathbf{x} * \mathbf{g}_{\theta} = \mathbf{U}\mathbf{g}_{\theta}\mathbf{U}^{\intercal}\mathbf{x}$$

1. SPECTRAL GRAPH CONVOLUTION
1.1 SPECTRAL CONVOLUTIONAL NEURAL NETWORK (SPECTRAL CNN)

- *Convolution filter :* $g_\theta = \Theta_{i,j}^{(k)}$ (diagonal matrix)
- *Graph Convolutional Layer :* $H_{i,j}^{(k)} = \sigma(\sum_{i=1}^{f_{k-1}} U\Theta_{i,j}^{(k)} U^T H_{:,i}^{(k-1)})(j = 1, 2, ..f_k)$
- *k : index of Layer H*
- σ : activation function

- Limitation
  - When the graph get small change, eigen vector is changed
  - Each filters are domain-dependent. When graph structure is changed, it is not applicable
  - Computation complexity is high on eigen-decomposition

1. SPECTRAL GRAPH CONVOLUTION
1.2 CHEBYSHEV SPECTRAL CONVOLUTIONAL NEURAL NETWORK (CHEBNET)

- Filter : $g_\theta = \sum_{i=0}^{K} \theta_i T_i(\tilde{\Lambda})$
- $\tilde{\Lambda} = 2\Lambda/\lambda_{max} - I$
  [-1, 1]
- If $\tilde{L} = 2L/\lambda_{max} - I$

  => $T_i(\tilde{L}) = UT_i(\tilde{\Lambda})U^T$

  (by mathematical induction)

- Update

$$\begin{aligned} \mathrm{x} *_G g_\theta &= Ug_\theta U^T \mathrm{x} \\ &= U(\sum_{i=0}^{K} \theta_i T_i(\tilde{\Lambda}))U^T \mathrm{x} \\ &= \sum_{i=0}^{K} \theta_i T_i(\tilde{L})\mathrm{x} \end{aligned}$$

- Chebyshev polynomial *T(x)*

$$T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x)$$
$$T_0(x) = 1, T_1(x) = x$$

- Improvement
  - Filter is defined by polynomial form
  - => filters can extract local feature independently
- Limitation
  - Polynomials have never used in Deep learning
  - => Use Linear convolution filter : GCN

1. SPECTRAL GRAPH CONVOLUTION
1.3 GRAPH CONVOLUTION NETWORK (GCN)

- ChebNet $= \sum_{i=0}^{K} \theta_i T_i(\tilde{L})\mathrm{x}$
- Apply K=1, λ_{max} = 2 ChebNet

$$\mathrm{x} *_G g_\theta = U g_\theta U^T \mathrm{x}$$
$$= \theta_0 \mathrm{x} - \theta_1 D^{-1/2} A D^{-1/2} \mathrm{x}$$

- To prevent over-fitting, set $\theta_0 = -\theta_1 = \theta$

$$\mathrm{x} *_G g_\theta = \theta(I + D^{-1/2} A D^{-1/2})\mathrm{x}$$

- Layer : $H = X *_G g_\Theta = f(\bar{A} X \Theta)$
  - $\bar{A} = I + D^{-1/2} A D^{-1/2}$
  - f: activation function
- To make learning stable, set

$$\bar{A} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$$
$$(\tilde{A} = A + I, \tilde{D}_{i,i} = \sum_j \tilde{A}_{i,j})$$

- Significance : GCN is base of Spatial methods

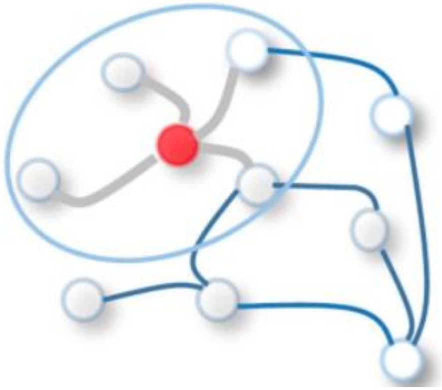
GRAPH CONVOLUTION
2. SPATIAL METHODS

: define convolutions on the graphs operating on neighbors

- major challenge 1) defining the conv operation with different sized neighborhoods

  2) maintaining the local invariance of CNNs

- Spatial methods
  - Convolutionally operate on graph directly
    - With only neighborhood nodes
  - Rearrange the node and neighborhood nodes in grid form



2. SPATIAL GRAPH CONVOLUTION
2.1 NEURAL NETWORK FOR GRAPH (NN4G)

- First spatial-based ConvGNN
- Layer
  - $$h_v^{(k)} = f({W^{(k)}}^T \mathbf{x}_v + \sum_{i=1}^{k-1} \sum_{u \in N(v)} {\Theta^{(k)}}^T h_u^{(k-1)}), h_v^{(0)} = 0$$
  - Matrix form $$H^{(k)} = f(XW^{(k)} + \sum_{i=1}^{k-1} AH^{(k-1)}\Theta^{(k)})$$
  - F : activation function
  - W : edge parameter
  - Θ : Layer parameter
  - A : adjacency matrix

2. SPATIAL GRAPH CONVOLUTION
2.2 GRAPH CONVOLUTIONAL NETWORK (SPATIAL-BASED)

- Spectral-based form

$$H = X *_G g_\Theta = f(\bar{A}X\Theta)$$

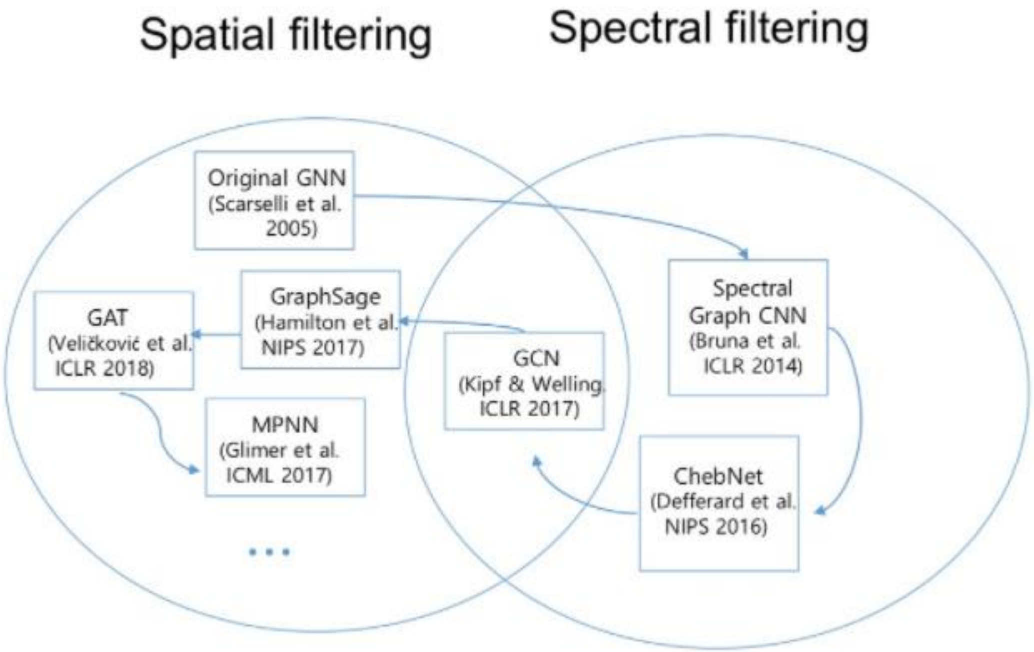
- Spatial-based form

$$h_v = f(\Theta^T(\sum_{u \in N(v) \cup v} \bar{A}_{vu}X_u)), \forall v \in V$$

  - Matrix form

$$H^{(k)} = f(\sum_{i=1}^{k-1} \bar{A}H^{(k-1)}\Theta^{(k)})$$

- Improvement
  - Use $\bar{\text{A}}$ instead of just A. $\bar{A} = I + D^{-1/2}AD^{-1/2}$
    - HIgher performance



2. SPATIAL GRAPH CONVOLUTION
2.3 PARTITION GRAPH CONVOLUTION (PGC)

- Improvement
  - Reduce the amount of computation with reducing **the size of adjacency matrix(A)**
  - By **divide neighbor nodes** into Q partitions

- Layer
  - $$H^{(k)} = \sum_{j=1}^{Q} \bar{A}^{(j)} H^{(k-1)} W^{(j,k)}$$

COMPARISON

- Spectral methods vs Spatial methods
  - Computational efficiency
    - Spectral : require eigen decomposition
    - Spatial : Compute only neighbor
  - Flexibility about change of graph structure
    - Spectral : can be used in only fixed graph
    - Spatial : flexible
  - Acceptability about various graph types
    - Spectral : only for undirected graph
    - Spatial : able to be applied to various graph including directed graph

REFERENCE

- https://thejb.ai/comprehensive-gnns-4/
- https://ralasun.github.io/deep%20learning/2021/02/15/gcn/
- https://ahjeong.tistory.com/14