SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

by
Thomas N. Kipf, Max Welling

Presented by Sungwon Kim

Contents

Background

Graph Neural Network (GNN) ?

1. Recurrent Graph Neural Network
2. Spatial Convolutional Network
3. Spectral Convolutional Network

Background

Spectral-based GNNs

Spectral CNN (2014)

- 최초의 NN기반 Spectral graph-based 방법론
- Filter를 학습 가능한 parameter로 대체

ChebyNet (2016)

- Chebyshev expansion을 이용하여 filter approximation
- Laplacian Matrix의 eigen-decomposition 생략

GCN (2017)

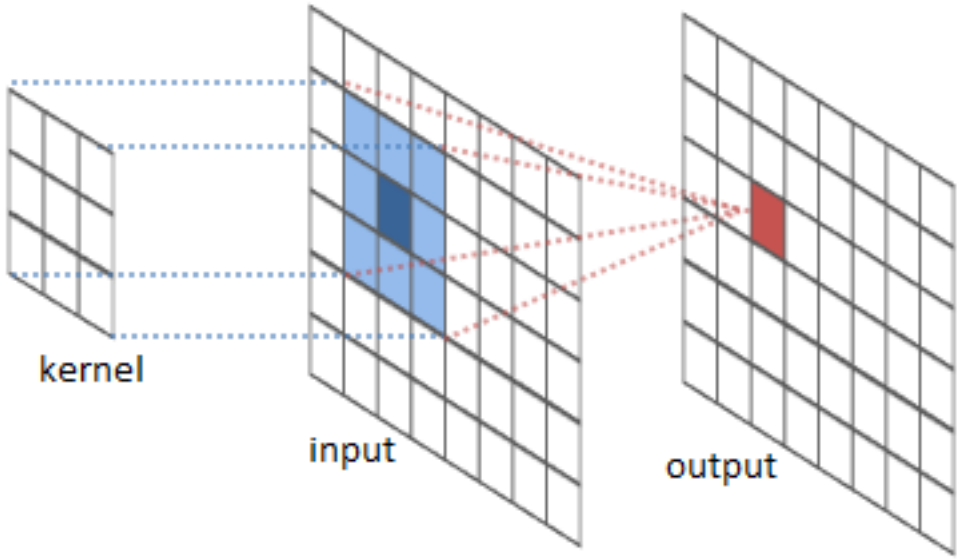
- ChebyNet을 더욱 간소화시킨 GNN 모델 제안
- Self-loop추가하여 Renormalization trick을 제안

SGC(2019)

- GCN을 더 간소화 시킨 모델 제안
- Adjacency의 제곱승을 통해 Message Passing을 정의
- Nonlinearity를 제거하여 학습시간을 단축

Background

What is Convolution?



1. Learn Local Feature
2. Fixed size filter
3. Weight Sharing
   - Reduce the number of parameters

Convolution on Graph?

Background

Graph Convolutional Network

General CNN
- Fixed Filter size

하지만 대부분의 그래프는 그리드 형태가 아니라
이웃간 노드의 크기가 제각각으로 달라짐.

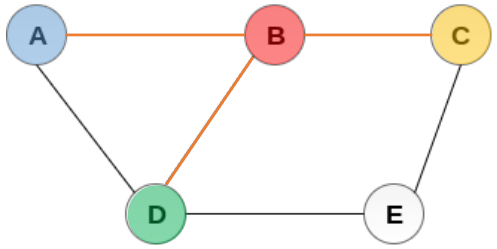
필터의 크기는 유지하면서 달라지는
이웃 노드 수를 반영할 수 있는 방법?

Adjacency matrix !

Background

Graph Convolutional Network – Adjacency Matrix

Update hidden states (= node features)



Undirected Graph

	A	B	C	D	E
A	0	1	0	1	0
B	1	0	1	1	0
C	0	1	0	0	1
D	1	1	0	0	1
E	0	0	1	1	0

Adjacency Matrix

$$H_B^{(l+1)} = \sigma(H_B^{(l)} W^{(l)} + H_A^{(l)} W^{(l)} + H_C^{(l)} W^{(l)} + H_D^{(l)} W^{(l)} + b^l)$$

자기자신과 이웃 노드의 hidden state를 반영하여
Hidden state(=node feature)를 업데이트

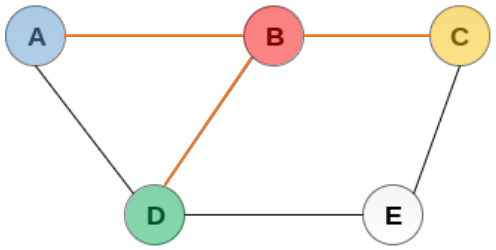
➡ **but too expensive!**

Let's use matrix product!

Background

Graph Convolutional Network – Adjacency Matrix

Update hidden states (= node features)

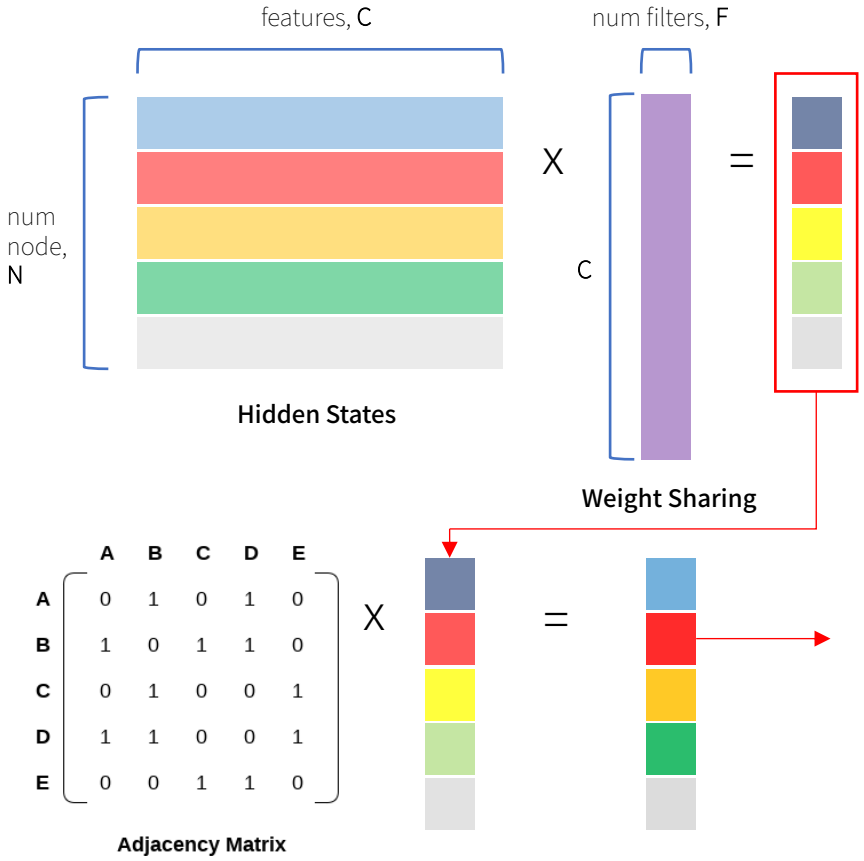


Undirected Graph

	A	B	C	D	E
A	0	1	0	1	0
B	1	0	1	1	0
C	0	1	0	0	1
D	1	1	0	0	1
E	0	0	1	1	0

Adjacency Matrix

$$H_B^{(l+1)} = \sigma(H_B^{(l)} W^{(l)} + H_A^{(l)} W^{(l)} + H_C^{(l)} W^{(l)} + H_D^{(l)} W^{(l)} + b^l)$$



$$H_B^{(l+1)} = \sigma(H_A^{(l)} W^{(l)} + H_C^{(l)} W^{(l)} + H_D^{(l)} W^{(l)})$$

Background

Graph Convolutional Network – Adjacency Matrix

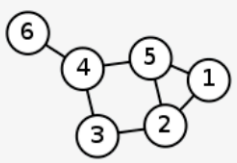
	A	B	C	D	E
A	0	1	0	1	0
B	1	0	1	1	0
C	0	1	0	0	1
D	1	1	0	0	1
E	0	0	1	1	0

Adjacency Matrix

General

- 자기 자신의 정보를 반영하지 못함
- 연결이 많이 되어 있는 노드는 exploding gradient
- 적게 되어 있는 노드는 vanishing gradient

Laplacian Matrix

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

$$L = D - A$$ (Laplacian for Simple Graphs)

$$L^{\text{sym}} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

(Symmetric Normalized Laplacian)

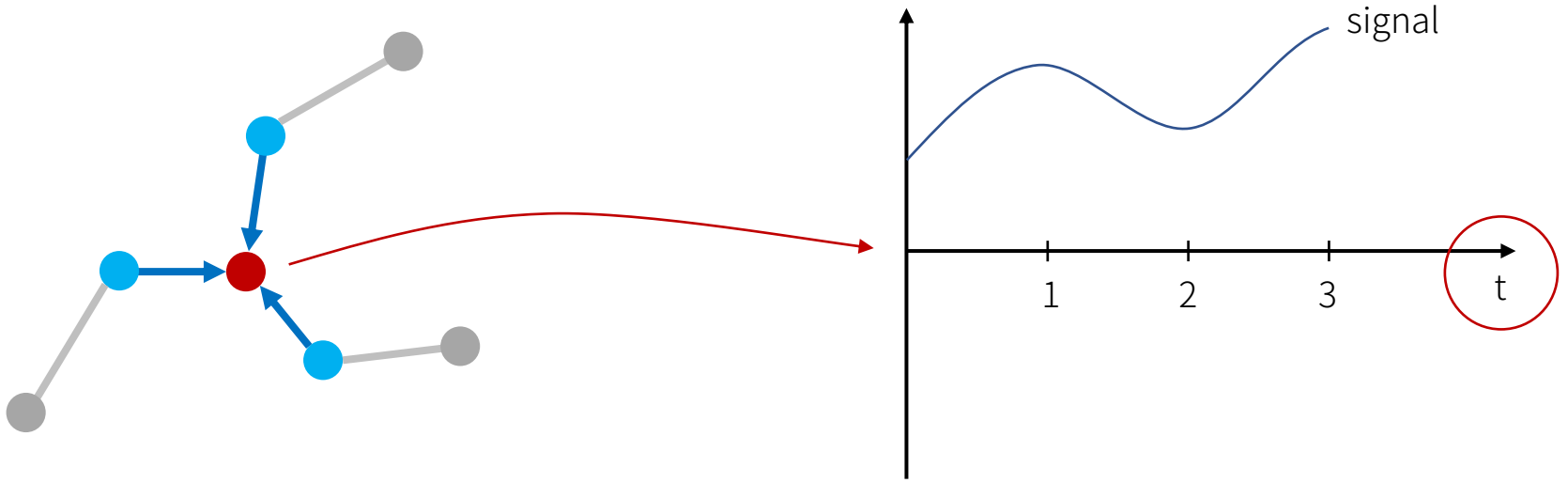
- Weight Sharing on Graph ✓
- Able to use the Fixed-size filter on Graph ✓

How can we make it fast and efficient?

Background

Spatial - Spectral (Fourier Transform)

Spatial Graph Convolution – Time dependancy

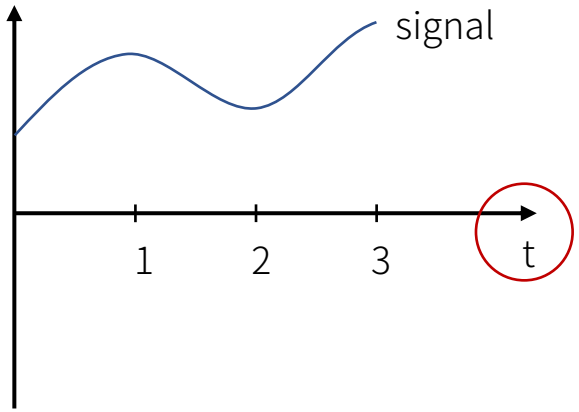


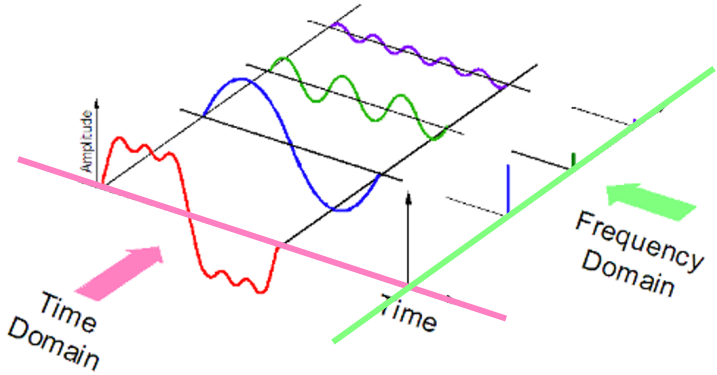
- 각노드와 가깝게 연결된 이웃 노드들에 한해서 Convolution 연산 수행
- 한 노드의 정보는 시간에 따라 여러 노드들의 정보의 혼합으로 표현됨
- 시간에 따라 멀리 연결되어 있는 노드의 정보를 시간이 흐르면서 밀려 들어올 수 있음 (message passing)

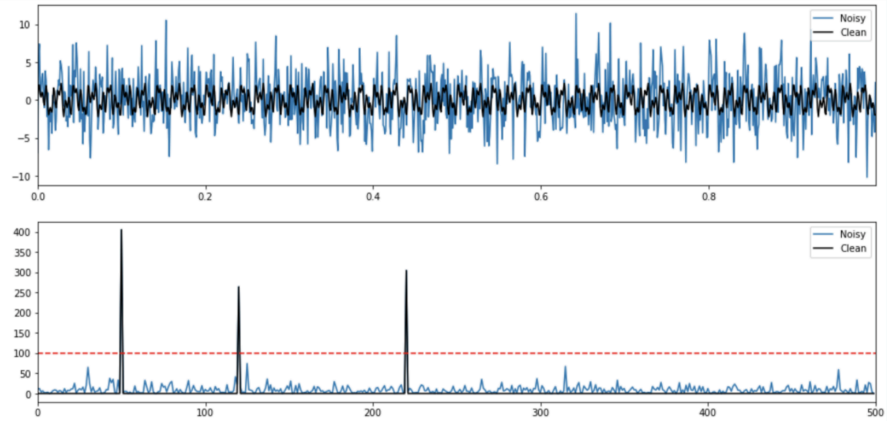
Background

Spatial - Spectral (Fourier Transform)

Fourier Transform







Time domain → **Frequency domain**

임의의 입력 신호를 주기함수들의 합으로 분해

Signal → Sine + Cosine → Orthonormal basis

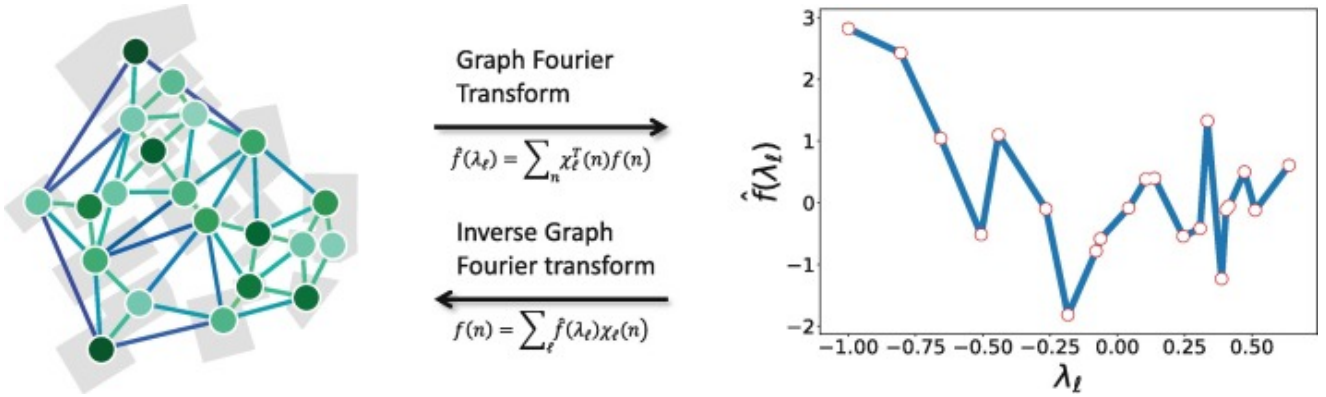
Signal ➡ **Frequency**

Fourier Transform

Background

Spatial - Spectral (Fourier Transform)

Graph Fourier Transform – Spectral Graph



Graph domain → **Frequency domain**

Graph domain의 Orthonormal basis가 되는 벡터를 찾자!

Background

Spatial - Spectral (Fourier Transform)

Graph Fourier Transform – What is orthonormal basis?

Orthonormal Basis

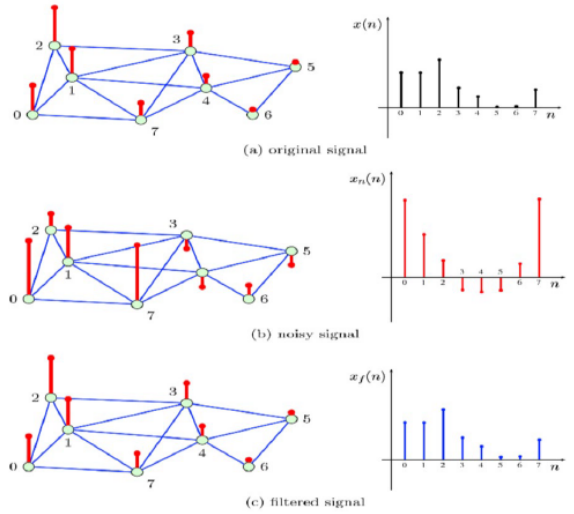
- 서로 직교 && 길이가 1인 벡터
- Real-symmetric matrix에 대하여 구한 eigenvector는 orthgonal

On graph?

- 이웃 노드와의 차이를 나타내는 Laplacian Matrix
- Laplcian Matrix는 Real Symmetric Matrix이므로 Eigenvector들은 Orthonormal Basis

따라서 Graph Signal을 Laplacian Eigenvector 행렬에 사영시키면,
Graph signal을 Frequency(=이웃 노드간 차이)로 변환

Graph Fourier Transform



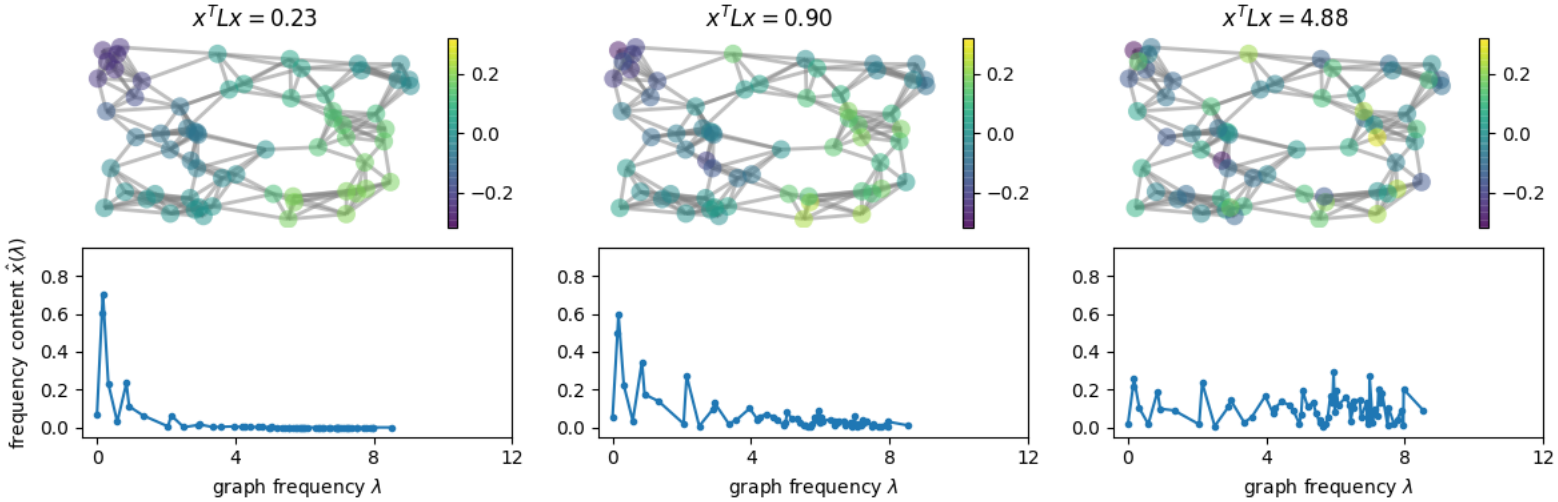
Graph Signal (Node Label) → Frequency (Difference on Central, neighbor node)

Eigenvalue of Laplacian : Frequency
Eigenvectors of Laplacian : Fourier basis

Graph의 Laplacian Matrix를 Eigen-decomposition하는 것이 Graph Signal의 푸리에 변환!

Background

Spectral Convolution (Graph Fourier Transform)



Signal : Node feature
Frequency : Central Node와 Neighbor Node간의 차이 (by Laplacian)

Frequency가 작다 ➡ 이웃 노드 간 차이 작다
Frequency가 크다 ➡ 이웃 노드 간 차이 크다

Lapalacian matrix의 Eigenvalue (λ) = Frequency가 되며
Eigenvalue (λ)가 작은 Eigenvector로 Fourier Transform

➡ **Central Node와 차이가 적은 노드들 간의 관계만 반영!**

Interim Check

Sum up

1. Adjacency Matrix를 이용하면 Graph에 Convolution의 특징을 반영할 수 있다.
   - Laplacian Matrix는 D와 A를 이용하여 노드 정보를 정규화시킨 Matrix이다.
2. Laplacian Matrix의 Eigen-decomposition을 이용하면 Graph를 Graph domain에서 Frequency domain으로 변환할 수 있다. (Fourier transform)

➡ **Applying Laplacian Matrix means Graph Convolution !**

Introduction

Goal

Given

1. **각 노드의 Feature를 담고 있는 matrix, X**
2. **Adjacency Matrix, A**

Semi-Supervised Classification With Graph Convolutional Networks

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

Introduction

Spectral Convolutions on Graphs

$$g_\theta \star x = U g_\theta U^\top x$$

➡ Laplacian – Eigen Decomposition

signal $x \in \mathbb{R}^N$

filter $g_\theta = \text{diag}(\theta)$ parameterized by $\theta \in \mathbb{R}^N$

➡ function of the eigenvalues of L, i.e. $g_\theta(\Lambda)$

$Laplacian\ L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^T$

U = matrix of eigenvectors of the Laplacian L

x ➡ Signal

⬇

$U^\top x$ ➡ Graph Fourier Transform

⬇

$g_\theta U^\top x$ ➡ Filtering ($g_\theta^* = g_\theta^*(\Lambda)$)

⬇

$U g_\theta U^\top x$ ➡ Inverse Graph Fourier Transform

➡ **Computationally Expensive !**

Introduction

Chebyshev Polynomials

$$g_\theta \star x = U g_\theta U^\top x$$

$g(\theta) =$ function of the eigenvalues of L, i.e. $g_\theta(\Lambda)$

➡ 전체 Eigen-value 사용으로 Localized 되지 않음 !

⬇ $g_\theta(\Lambda)$: well-approximated by Chebyshev polynomials

$\tilde{\Lambda} = \frac{2}{\lambda_{\max}}\Lambda - I_N$

$\lambda_{\max}$ denotes the largest eigenvalue of L

θ': Chebyshev coefficients

Recursively defined as
$T_k(x) = 2xT_{k-1} - \mathrm{T_{k-2}}(x)$
$T_0(x) = 1, T_1 = x$

$$g_{\theta'}(\Lambda) \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{\Lambda}) = \theta'_0 T_0(\tilde{\Lambda}) + \theta'_1 T_1(\tilde{\Lambda}) + \theta'_2 T_2(\tilde{\Lambda}) \cdots + \theta'_K T_K(\tilde{\Lambda})$$

Introduction

Approximation

$$g_{\theta'} \star x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L}) x$$

➡ K^{th}-order polynomial in the Laplacian

Now K-localized!

⬇ K=1 (1st order neighborhood)

$$g_{\theta'} \star x \approx \theta'_0 T_0(\tilde{L}) x + \theta'_1 T_1(\tilde{L}) x \quad ➡ \tilde{L} = \frac{2}{\lambda_{max}} L - I_N,\ let\ \lambda_{max} = 2$$

$$\approx \theta'_0 x + \theta'_1 (L - I_N) x \quad ➡ L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

$$\approx \theta'_0 x - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x \quad ➡ \theta = \theta'_0 = -\theta'_1$$

$$\approx \theta \left(\mathrm{I_N} + \mathrm{D}^{-\frac{1}{2}} \mathrm{A} \mathrm{D}^{-\frac{1}{2}} \right) x$$

Renomarlization Trick !

$$\mathrm{I_N} + \mathrm{D}^{-\frac{1}{2}} \mathrm{A} \mathrm{D}^{-\frac{1}{2}} \quad ➡ \quad \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$$

➡ $\tilde{A} = A + I_N \quad and \quad \tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta$$

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right)$$

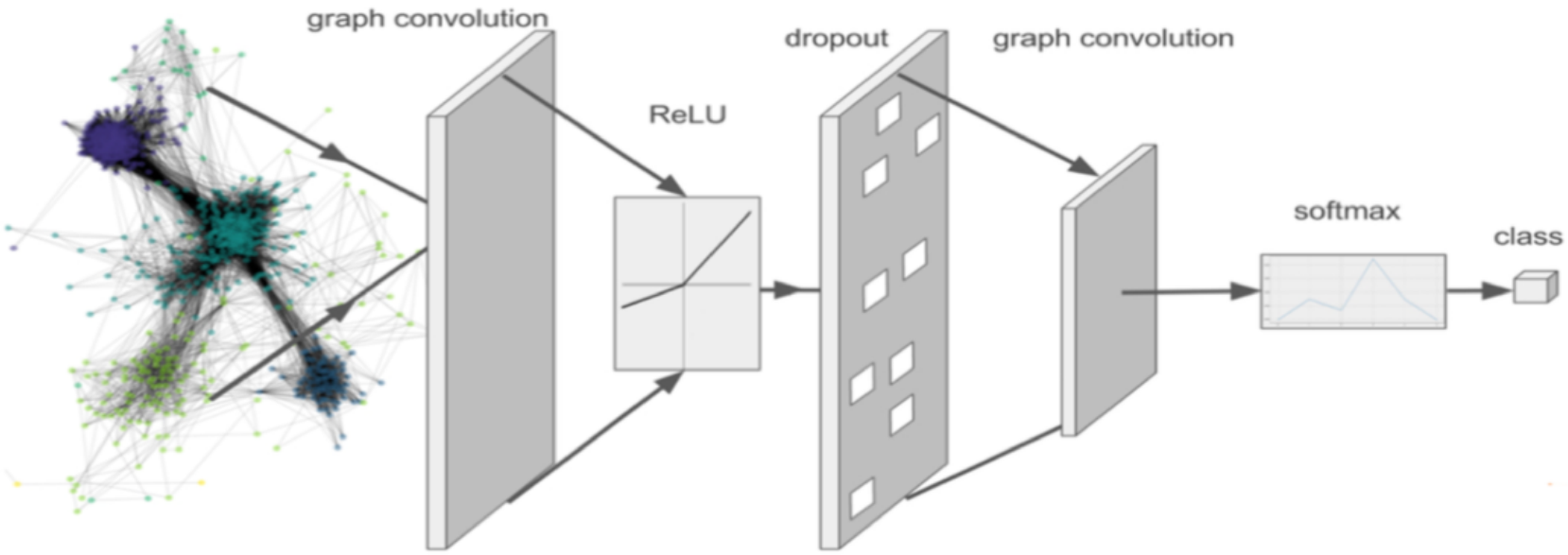
Introduction

Finally

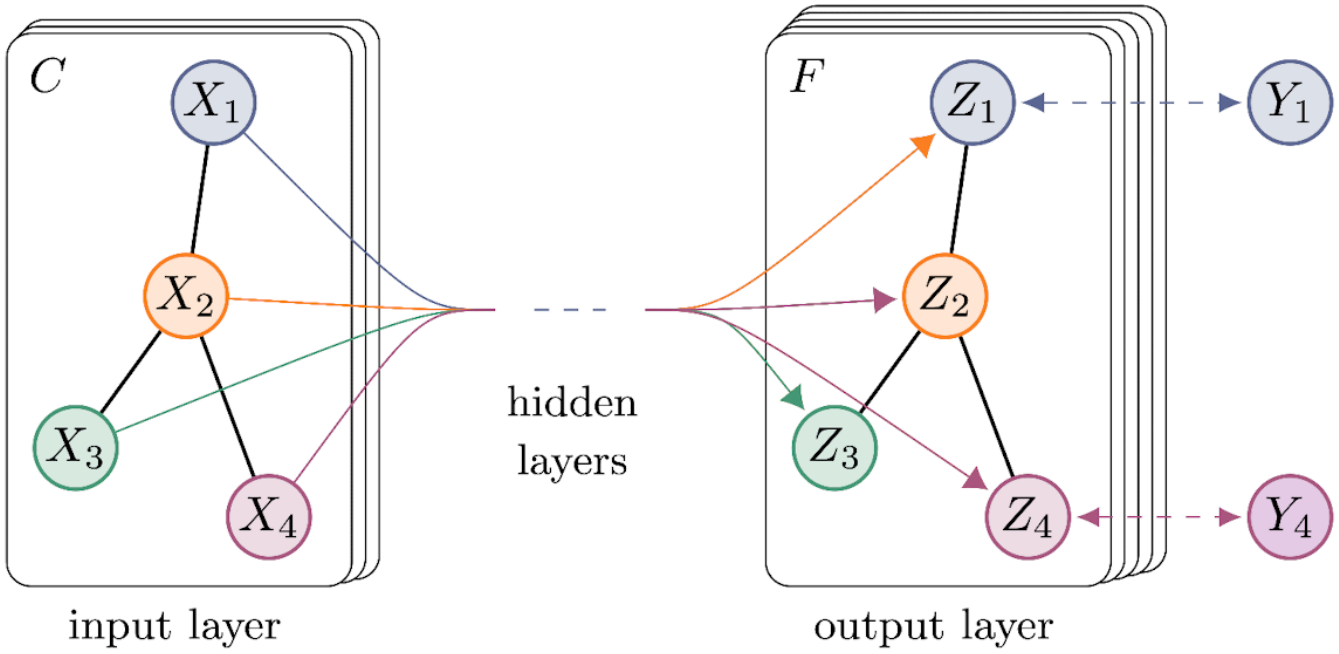
$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\ W^{(1)}\Big)$$

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{\mathrm{reg}}$$

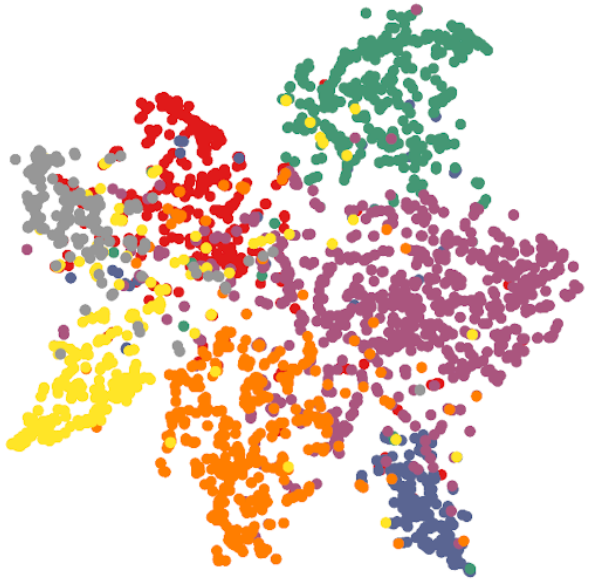
$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$



Introduction



(a) Graph Convolutional Network



(b) Hidden layer activations

EXPERIMENTS

Dataset	Type	Nodes	Edges	Classes	Features	Label rate
Citeseer	Citation network	3,327	4,732	6	3,703	0.036
Cora	Citation network	2,708	5,429	7	1,433	0.052
Pubmed	Citation network	19,717	44,338	3	500	0.003
NELL	Knowledge graph	65,755	266,144	210	5,414	0.001

Table 2: Summary of results in terms of classification accuracy (in percent).

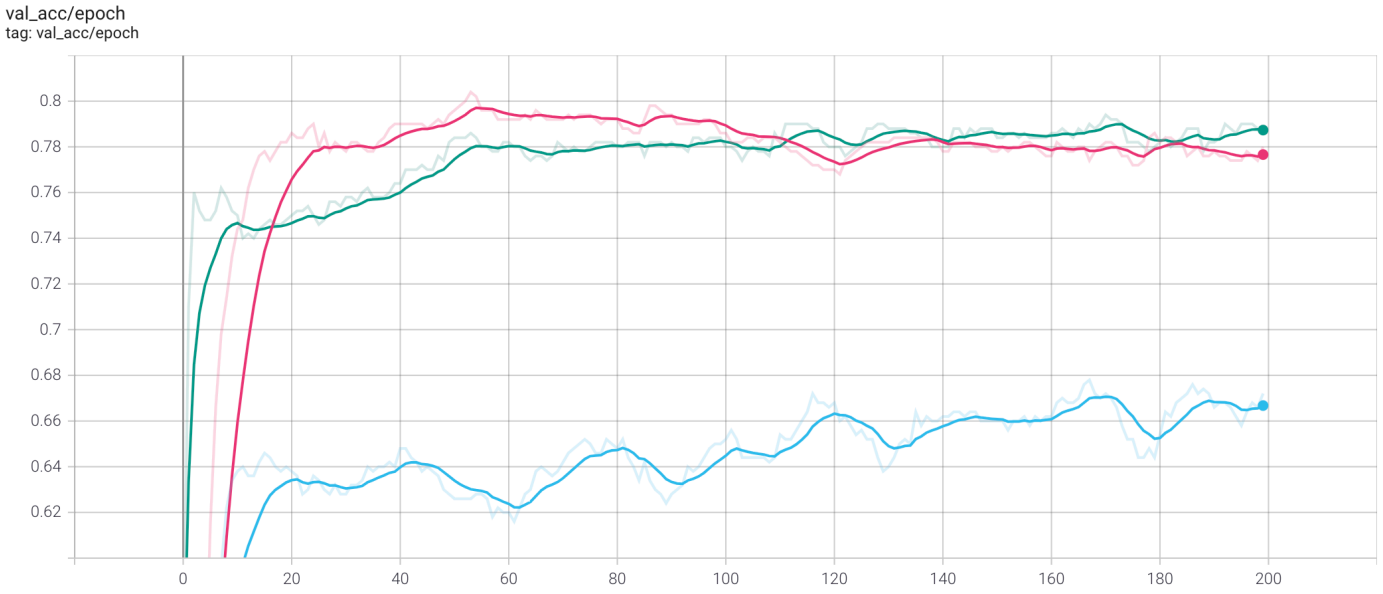
Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7

EXPERIMENTS

Table 3: Comparison of propagation models.

Description		**Propagation model**	**Citeseer**	**Cora**	**Pubmed**
Chebyshev filter (Eq. 5)	$K = 3$	$\sum_{k=0}^{K} T_k(\tilde{L})X\Theta_k$	69.8	79.5	74.4
	$K = 2$		69.6	81.2	73.8
1^{st}-order model (Eq. 6)		$X\Theta_0 + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta_1$	68.3	80.0	77.5
Single parameter (Eq. 7)		$(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})X\Theta$	69.3	79.2	77.4
Renormalization trick (Eq. 8)		$\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta$	**70.3**	**81.5**	**79.0**
1^{st}-order term only		$D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta$	68.7	80.5	77.8
Multi-layer perceptron		$X\Theta$	46.5	55.1	71.4

IMPLEMENTATION



Name	Smoothed	Value	Step	Time	Relative
GCN_dataset(citeseer)_lr(0.01)_epoch(200)_hidden(16)	0.6668	0.672	199	Sun Jul 4, 15:24:27	0s
GCN_dataset(cora)_lr(0.01)_epoch(200)_hidden(16)	0.7766	0.78	199	Sun Jul 4, 15:24:53	0s
GCN_dataset(pubmed)_lr(0.01)_epoch(200)_hidden(16)	0.7873	0.786	199	Sun Jul 4, 15:25:06	0s

```python
class GCNNet(torch.nn.Module):
    def __init__(self, dataset,num_hidden):
        super(GCNNet, self).__init__()
        self.conv1 = GCN(dataset.num_node_features, num_hidden)
        self.conv2 = GCN(num_hidden, dataset.num_classes)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index

        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = F.dropout(x, training=self.training)
        x = self.conv2(x, edge_index)

        return F.log_softmax(x, dim=1)
```

time: 0.0043s/epoch (cora)

LIMITATINS AND FUTURE WORK

Memory requirement :
Full-batch gradient descent → Mini-batch gradient descent

Directed edges and edge features :
Undirected graphs → Directed graphs (i.e. NELL)

Limiting assumptions :
Equal importance of self-connections vs. edges to neighboring nodes → $\tilde{A} = A + \lambda I_N$

CONCLUSION

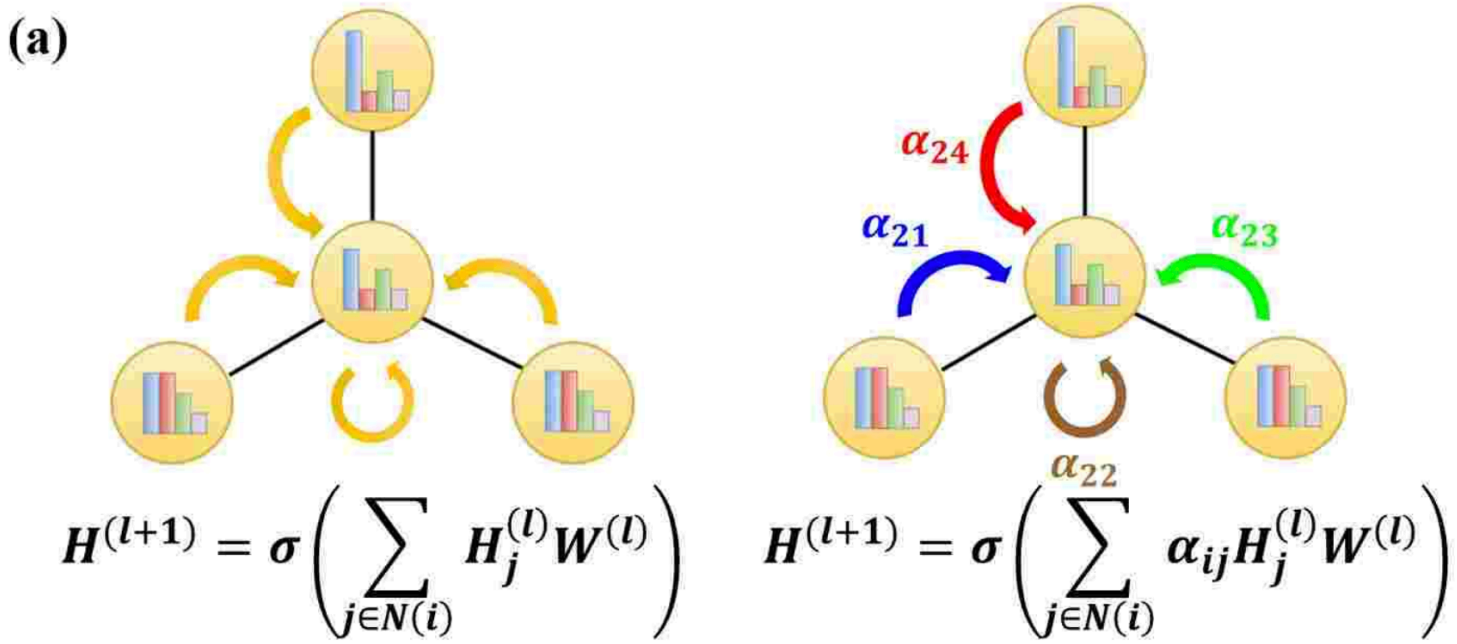
1
Applying Laplacian Matrix means Graph Convolution

2
Polynomial한 Approximation을 통하여 First-order approximation으로
Locality Feature 반영 + 계산복잡도도 낮춤

3
GCN은 Spectral Convolutions on Graph 문제를 First-order approximation을
통해더 빠르고, 정확하게 Semi-supervised classification를 할 수 있다.

ADVANCE

GCN vs GAT



Ryu, Seongok, Jaechang Lim, Seung Hwan Hong and Woo Youn Kim
"Deeply learning molecular structure-property relationships using attention- and gate augmentation graph convolutional network" arXiv preprint arXiv:1805.10988 (2018)

Reference

- Basic of Graph Convolution Network - 딥러닝 홀로서기
- http://cd4761.blogspot.com/2016/03/cnnconvolution-neural-network.html
- https://baekyeongmin.github.io/paper-review/gcn-review/
- The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains
- https://pygsp.readthedocs.io/en/latest/examples/fourier_transform.html
- https://hannah37.github.io/graph%20convolutional%20network/2020/08/06/Convolutional-Neural-Networks-on-Graphs-with-Fast-Localized-Spectral-Filtering/
- [Seminar] Semi-supervised classification with graph convolutional networks – presenter Sukwon Yun
- [Seminar] Spectral-based_GCN – 최종현 (http://dsba.korea.ac.kr/seminar/?mod=document&uid=1329)