SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N.Kipf & Max Welling (ICLR 2017)

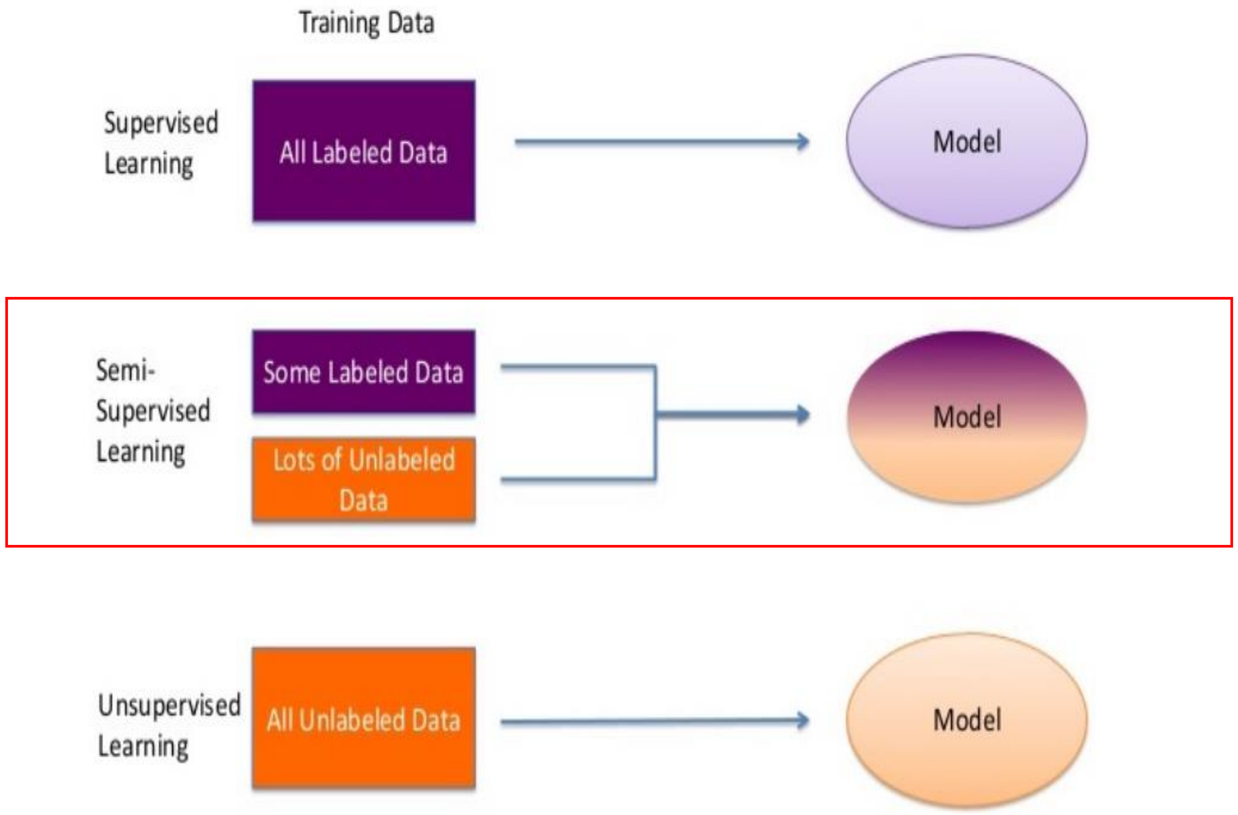
Noh Heewoong

2022.06.28

Contents

1. Background
2. Introduction
3. Method
4. Experiments
5. Conclusion

Semi-supervised Classification, GCN



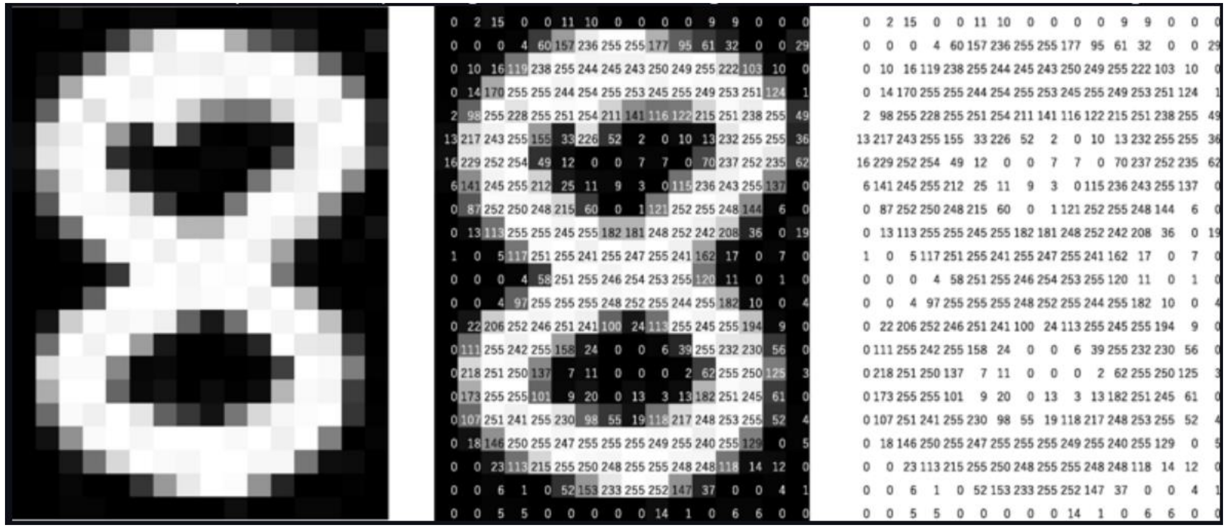
GNN + CNN = GCN

GCN is one of GNNs

Graph와 관련된 모든 Neural Network

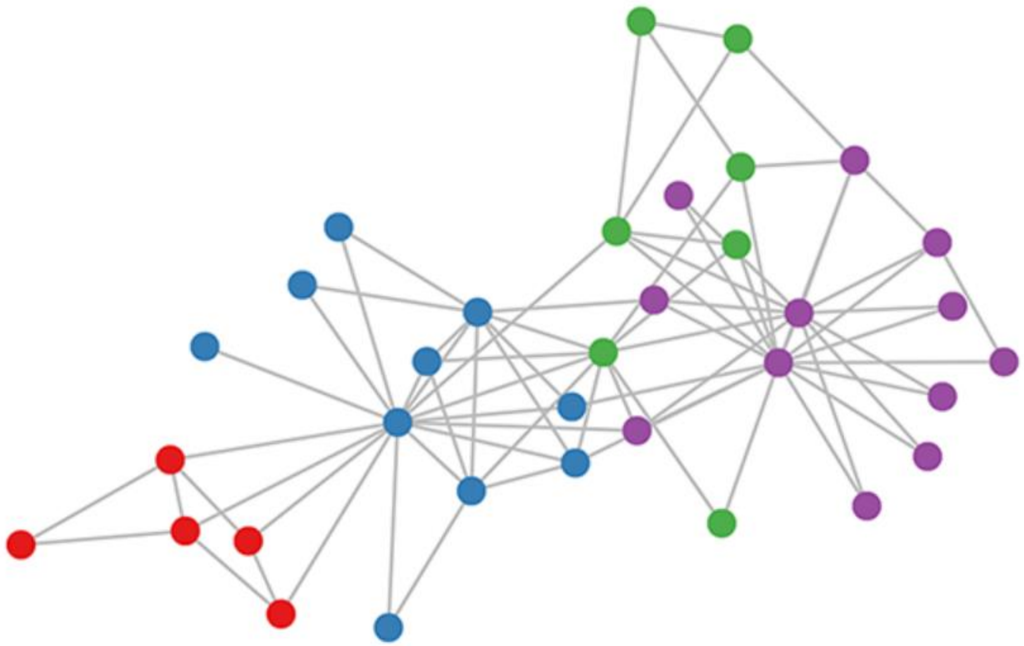
Background Image vs Graph

Image : Data on Grid



mnist image from: https://github.com/NehaCkumari

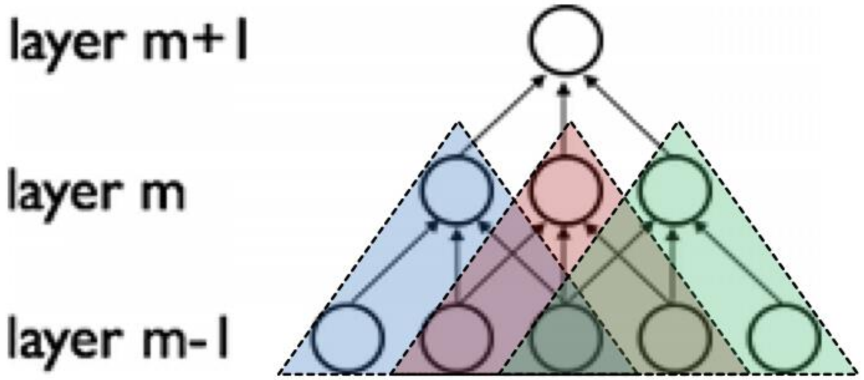
Graph : Data and Relation



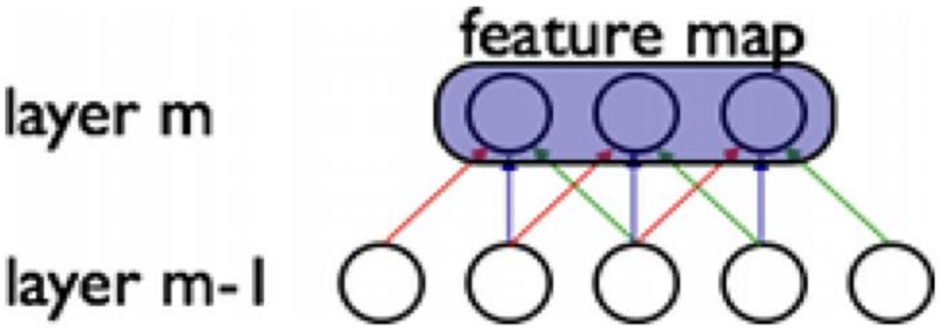
Karate club graph, colors denote communities obtained via modularity-based clustering

Background Convolution in CNN

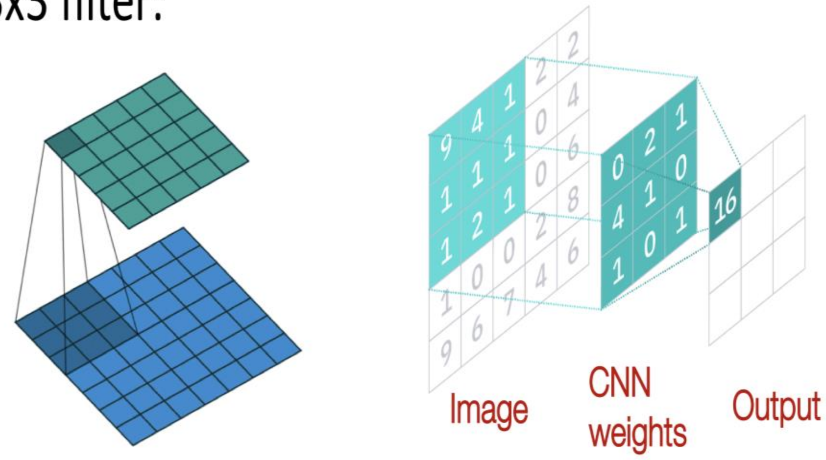
Sparse Connection



Shared Weight



Convolutional neural network (CNN) layer with 3x3 filter:



Spatially-local correlation 을 고려하기 위해 sparse connection을 구성
→인접한 변수만을 이용하여 새로운 feature 생성

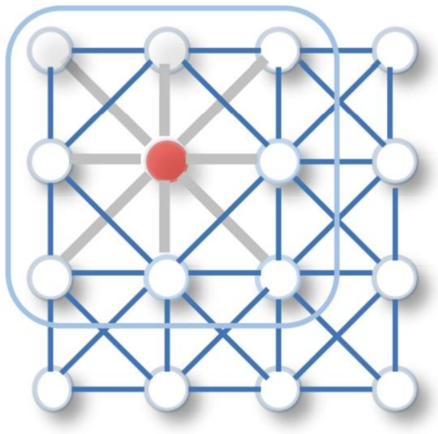
Invariant feature를 추출하기 위해 shared weight개념 이용
→인접한 변수 집합에는 동일한 weight를 적용

위의 둘을 실현시킨 방법이 Convolution in CNN

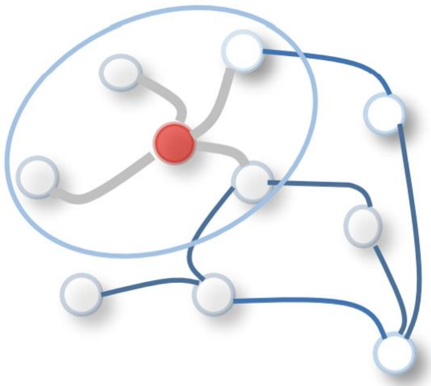
Graph Convolution

하고싶은 Task: Graph에 Convolution filter를 통해 하나의 노드와 이웃한 노드들과의 관계를 계산

CNN의 방식을 그대로 적용할 수 있을까?



2D convolution



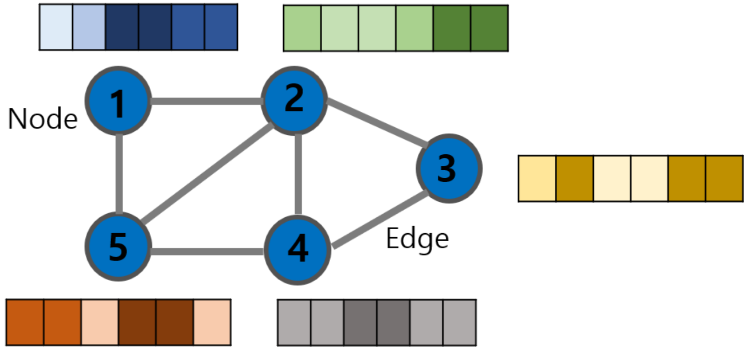
Graph convolution

기존 CNN의 convolution filter를 사용할 수 없음.

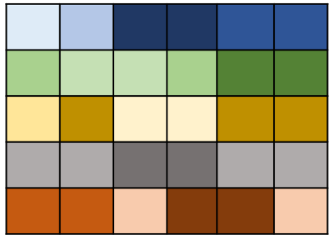
- 기본적으로 사각형의 grid 데이터에 적용됨.
- Filter의 크기가 노드들 마다 달라질 수 있음.

Image from: 'A Comprehensive Survey on Graph Neural Networks'

Background Review Graph



Node Feature Matrix
$X \in R^{N\times F}$

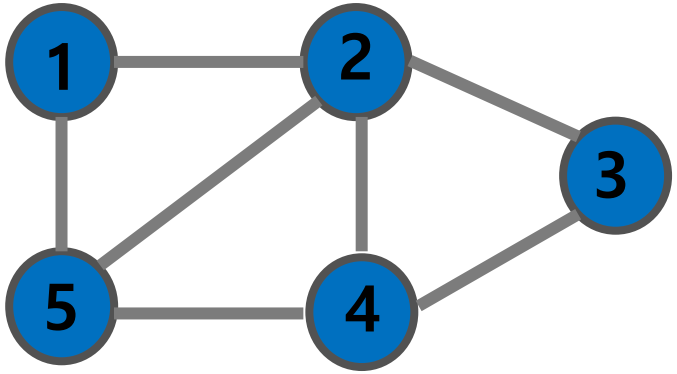


Node Class Matrix
$X \in R^{N\times C}$

0	1
0	1
0	1
1	0
1	0

Node: Data (feature matrix, class matrix)

Edge: Relation b/w nodes (weight, scalar)



Adjacency Matrix

노드간의 관계를 표현한 행렬

0	1	0	0	1
1	0	1	1	1
0	1	0	1	0
0	1	1	0	1
1	1	0	1	0

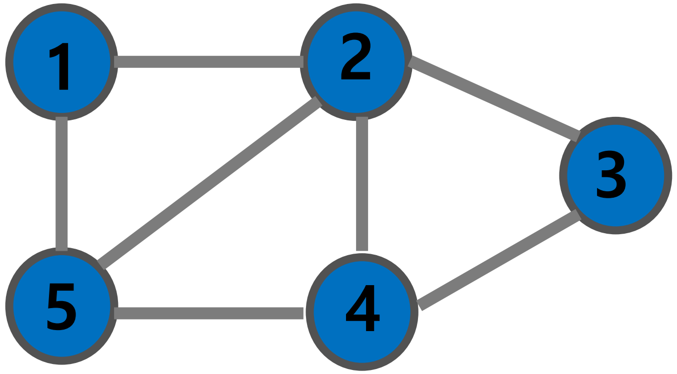
Degree Matrix (Diagonal)

Degree는 각 노드와 연결된 edge의 수

2	0	0	0	0
0	4	0	0	0
0	0	2	0	0
0	0	0	3	0
0	0	0	0	3

$$D \in R^{n\times n}, D_{ii} = \sum_j A_{ij}$$

Background Laplacian Matrix



$$Laplacian\ Matrix: L \in R^{N \times N}, D - A$$
$$Degree\ matrix\ - Adjacency\ matrix$$

2	0	0	0	0
0	4	0	0	0
0	0	2	0	0
0	0	0	3	0
0	0	0	0	3

−

0	1	0	0	1
1	0	1	1	1
0	1	0	1	0
0	1	1	0	1
1	1	0	1	0

=

L

2	-1	0	0	-1
-1	4	-1	-1	-1
0	-1	2	-1	0
0	-1	-1	3	-1
-1	-1	0	-1	3

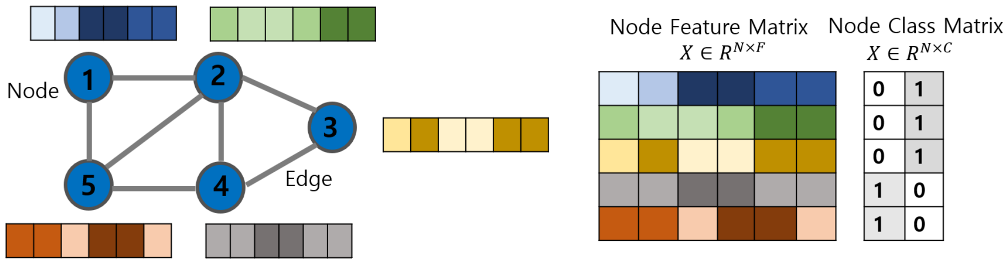
Undirected graph

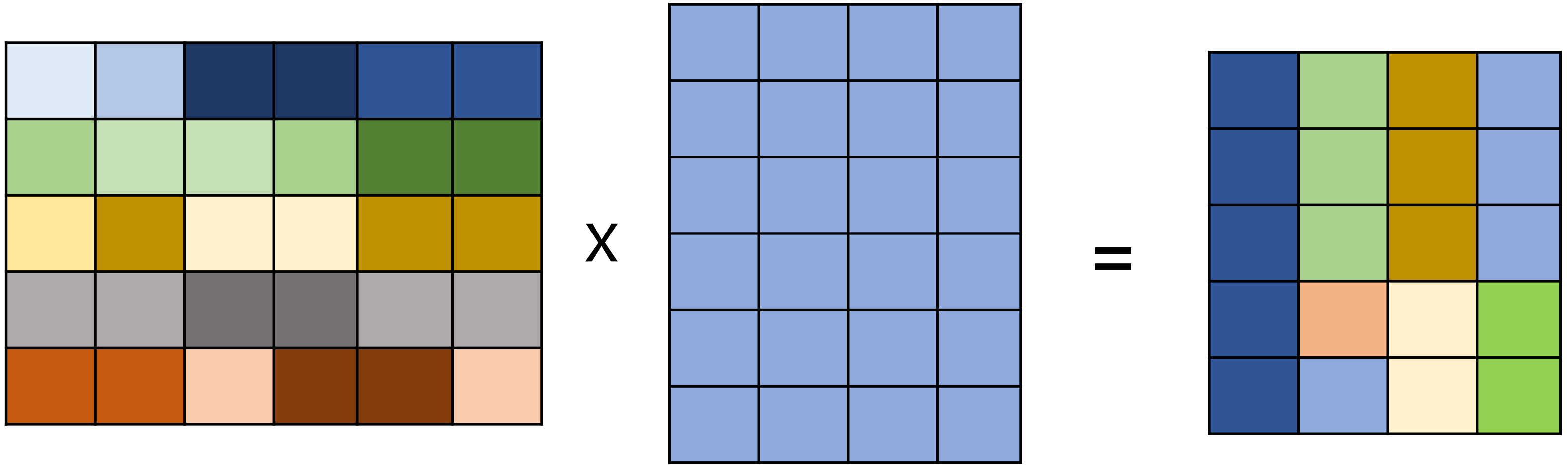
L: Real symmetric matrix
Positive Semi-Definite matrix
때문에, non-negative real eigenvalue 가짐.

Degree matrix로 L을 정규화하면:

$Normalized\ Laplacian$:
$= I - D^{-1/2} A D^{-1/2}$

Background Update hidden state in GCN





N x F

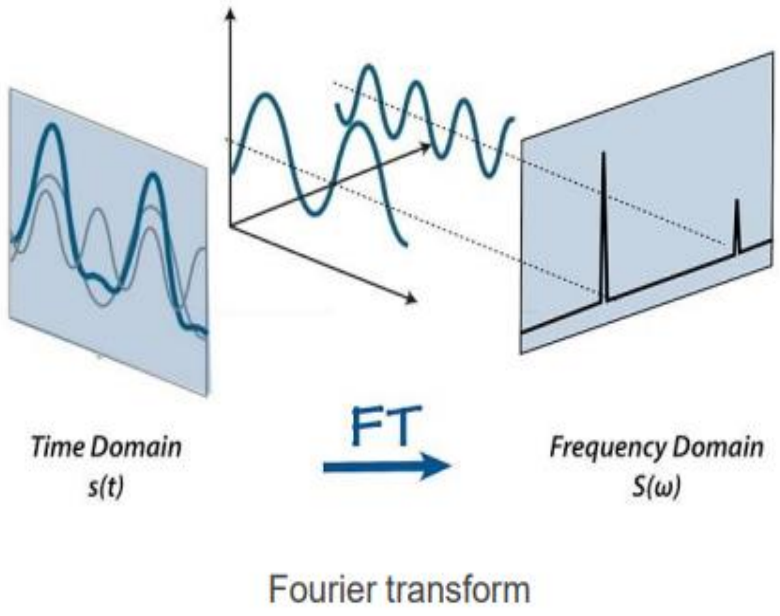
F x K (K: no of filters)

Feature map: N x K

Adjacency matrix와 곱해주면
(N x N) X (N x K)

(N x K)로 Node Feature Matrix(hidden state) update

Graph Fourier Transform



Signal Processing에서 spectral analysis 라는 것은 이미지/음성/그래프 신호를 time/spatial domain에서 frequency domain으로 변환하여 분석을 진행하는 것을 의미함.

대표적인 방법: Fourier Transform
→임의의 입력신호를 다양한 주파수를 갖는 주기함수들의 합으로 분해하여 표현하는 것을 말함.

Graph Fourier Transform

$$\hat{f}(\xi) = \int_{\mathbf{R}^d} f(x) e^{-2\pi i x \xi} \, dx$$ (1)Fourier Transform

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

$$e^{ix} = cost + isinx$$ (3)Euler's Formula

$$e^{2\pi i x \xi} = cos(2\pi x \xi) + isin(2\pi x \xi)$$ (4) (3)을 이용하여 (1)식의 $e^{2\pi i x \xi}$ 부분을 cos요소와 sin 요소의 합으로 표현

(1)에서 임의의 주파수 $f(x)$에 대하여 $\hat{f}(\xi)$는 $f(x)$와 $e^{-2\pi i x \xi}$의 내적 → 두 함수의 유사도

(4)를 통해서 주파수 $f(x)$에 대해 cosine에서 유사한 정도와 sine과 유사한 정도의 합이 푸리에변환이 됨.

$x \in R^d$ 에서 d차원의 orthonormal basis를 찾을 수 있다면, $vector\ x$를 orthonormal basis의 linear combination으로 표현 가능함.

Eigen decomposition을 통해서 orthonormal basis 찾기 가능함.

Real-symmetric matrix의 eigenvectors는 orthogonal함.

삼각함수의 직교성으로 인하여, sine-sine, sine-cosine, cosine-cosine 내적 모두 0

Graph Fourier Transform

<정리>
Signal: Node feature
Feature: Central node와 Neighbor node간의 차이(Laplacian Matrix)

어떤 특정 graph signal에 관한 행렬이 존재하고, real-symmetric matrix이며, eigenvectors를 구할 수 있다면, eigenvector의 선형결합이 graph signal의 푸리에 변환임을 의미하는 것.

Graph Fourier Transform
Laplacian Matrix를 eigen-decomposition하는 것

By Minimization of Laplacian Quadratic Form, Graph의 feature가 GFT를 통해 높은 Frequency 값이 나온다면, Feature간의 차이가 크다. →두 노드 간의 차이가 크다

결국 GFT를 통해서 graph signal을 spectral domain에서 표현 후 다시 spatial domain으로 복원할 때 사용되는 filter를 학습함으로써 graph signal의 noise를 제거하고, 필요성분으로만 node embedding을 하기 위해서 이다.

Graph Fourier Transform

$Eigen\ decomposition\ of\ normalized\ laplacian\ matrix$

$$\boldsymbol{L = U\Lambda U^T}$$

Orthonormal eigenvectors ordered by eigenvalues

$$U^T U = I$$

Diagonal matrix of eigenvalues

$$\Lambda_{ii} = \lambda_i$$

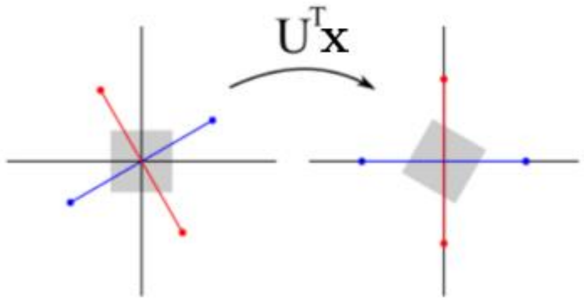
$U \rightarrow L_S$의 eigenvector로 이루어진 Fourier basis

Graph Fourier Transform

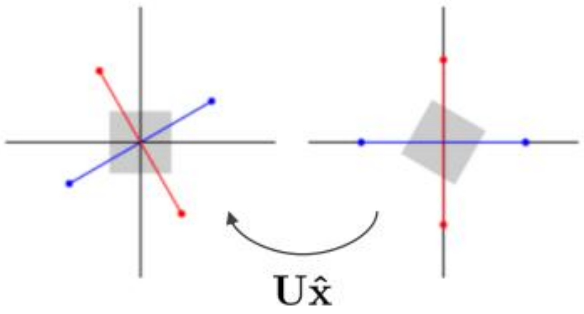
$$\mathcal{F}(x) = U^T x$$

$$\mathcal{F}^{-1}(\hat{x}) = U\hat{x}, \qquad (\hat{x} = U^T x)$$

GFT는 x를 L의 orthonormal (eigenvector) space로 projection 시킨 것.



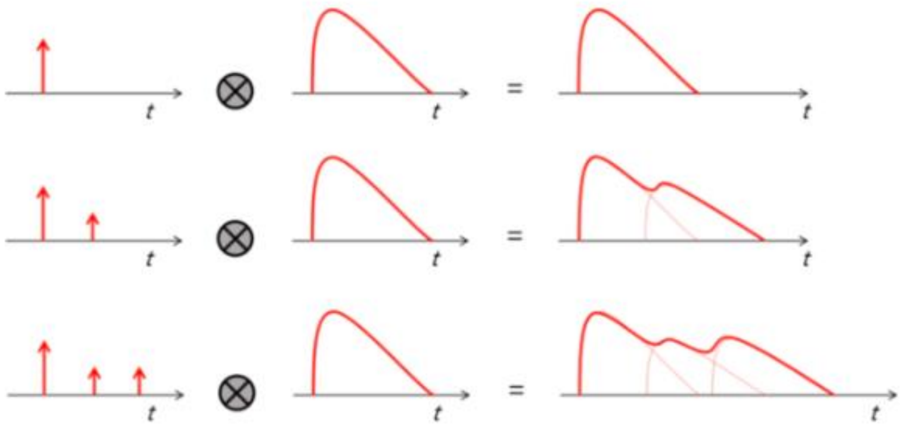
Fourier transform

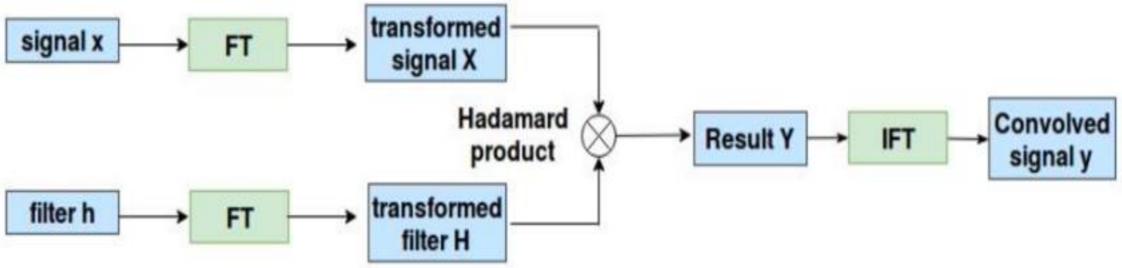


Inverse Fourier transform

Image from Jonghyun Choi, http://dsba.korea.ac.kr/seminar/?mod=document&uid=1329

Convolution Theorem





입/출력 시스템에서 출력값은 입력과 이전의 입력값에도 영향을 받음.
Convolution: 이전의 값까지의 영향을 고려하여 출력 계산하기 위한 연산

Convolution in spatial/time domain is equivalent to multiplication in Fourier domain
→Fourier Transform을 통해 convolution이 가능해짐.

Graph domain에서도 graph signal(node)를 FT하여 frequency domain 으로 변환하면 계산이 편리해짐.
또한 가까운 노드와 멀리 있는 노드에서 오는 신호 모두 고려하여 graph signal 특징을 추출할 수 있음.

Spectral Graph Convolution

Convolution filter: $g \in R^n$

$$\mathbf{x} *_G g = \mathscr{F}^{-1}(\mathscr{F}(\mathbf{x}) \odot \mathscr{F}(g))$$
$$= U(U^T\mathbf{x} \odot U^T g)$$

Convolution filter $g_\theta = diag(U^T g)$

$$\mathbf{x} *_G g_\theta = U g_\theta U^T \mathbf{x}$$

Spectral Graph Convolutions

Normalized Laplacian:

$Ls = D^{-1/2}LD^{-1/2} = I - D^{-1/2}AD^{-1/2}$

$L\ matrix$는 이미 Positive Semi-definite
normalized 이후 $Ls\ matrix$ 또한 Positive Semi-definite
→non-negative real eigenvalues 가짐.
0=$\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n \le 2$

$graph\ signal\ x \in R^n$ and $filter\ g_\theta = diag(\theta)$ Spectral Convolution $g_\theta * x = Ug_\theta U^T x = Lx$

$U \Rightarrow L_s(normalized\ graph\ Laplacian), I - S$ 의 $eigenvectors$로 이루어진 $Fourier\ basis$

$g_\theta: function\ of\ the\ eigenvalues\ of\ L$
but use all eigenvalues ,not localized

$$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U}\underbrace{\mathbf{g}_\theta}\mathbf{U}^T\mathbf{x}$$

$\Rightarrow$ $matrix\ multiplcation\ O(N^2)$

$$= \mathbf{U}\begin{bmatrix} \hat{\mathbf{g}}(\lambda_1) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \hat{\mathbf{g}}(\lambda_N) \end{bmatrix}\mathbf{U}^\top\mathbf{x}$$

$Eigen - decomposition\ for\ L:\ O(N^3)$

Spectral Graph Convolutions

Using "Chebyshev Expansion" to approximate $g_\theta(\Lambda)$

<Chebyshev Polynomial>

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$$

$$T_0(x) = 1 \qquad T_1(x) = x$$

$$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U}\underline{\mathbf{g}_\theta}\mathbf{U}^T\mathbf{x}$$

$$= \mathbf{U} \begin{bmatrix} \hat{\mathbf{g}}(\lambda_1) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \hat{\mathbf{g}}(\lambda_N) \end{bmatrix} \mathbf{U}^\top \mathbf{x}$$

$$g_\theta \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda})$$

$$\tilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - I_n, (-1 \le \tilde{\Lambda} \le 1)$$

Spectral Graph Convolutions

$$g_{\theta^*} x \approx U \left(\sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda}) \right) U^T x$$

$$T_k(\tilde{L}) = U T_k(\tilde{\Lambda}) U^T, \qquad \tilde{L} = \frac{2L}{\lambda_{max}} - I_n$$

$$\boxed{\tilde{L}} = \boxed{U} \overbrace{\boxed{T_k(\tilde{\Lambda})}}^{K: localized} \boxed{U^T}$$

Chebyshev expansion을 사용한
Laplacian Matrix

Spectral Graph Convolutions-ChebyNet

$$g_{\theta^*}x = U\left(\sum_{k=0}^{K}\theta_k T_k(\widetilde{\Lambda})\right)U^T x$$

$$T_k(\tilde{L}) = UT_k(\widetilde{\Lambda})U^T, \tilde{L} = \frac{2L}{\lambda_{max}} - I_n$$

$$g_{\theta^*}x = \sum_{k=0}^{K}\theta_k T_k(\tilde{L})\, x$$

ChebNet
Convolution

Introduction

Problem of classifying nodes in a graph (where labels are only available for a small subset of nodes)

Graph-based semi-supervised learning

Explicit graph-based regularization: a graph Laplacian regularization term in the loss function

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{\text{reg}} , \quad \text{with} \quad \mathcal{L}_{\text{reg}} = \sum_{i,j} A_{ij} \| f(X_i) - f(X_j) \|^2 = f(X)^\top \Delta f(X) .$$

$L_0: Supervised\ loss\ with\ label\ data$
$f(\cdot): differentiable\ function$
$\lambda: weighing\ factor, X: feature\ vector\ matrix$
$\Delta: D - A\ , unnormalized\ graph\ laplacian\ (undirected)$

Assumption

- Connected nodes in the graph are likely to share the same label
- Modeling capacity가 제한됨. Edge가 similarity 이외의 추가적인 정보를 가질 수 있음.

Method – Fast approximate convolution on graphs

Multi-layer Graph Convolutional Network (GCN)

Layer-wise propagation rule: **Convolution→** $H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\ \tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$

$\tilde{A} = A + I_N\ adjacency\ matrix\ of\ undirected\ graph\ with\ added\ self\ connections\ (I_N)$

$\tilde{D}_{ii} = \sum_j \tilde{A}_{iJ}\ and\ W^{(l)} : layer\ specific\ trainable\ weight\ matrix\ /\sigma: activation\ function, ReLU$

$H^{(l)} \in R^{N\times D}$:matrix of activation in the l^{th} layer $H^{(0)} = X$

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

→First-order approximation of localized spectral filters on graph

Graph Convolutional Network (GCN)

First-order approximation of ChebNet
K=1, λ_{max}=2로 설정하면 아래와 같이 유도됨.

$$\tilde{L} = \frac{2L}{\lambda_{max}} - I_n, \qquad Ls = I - D^{-1/2}AD^{-1/2}$$

<Chebyshev Polynomial>

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$$

$$T_0(x) = 1 \qquad T_1(x) = x$$

$$\begin{aligned} g_{\theta^*}x &= \sum_{k=0}^{K} \theta_k T_k(\tilde{L})\, x \\ &= \theta_0 x + \theta_1 T_1\left(\tilde{L}\right)x \\ &= \theta_0 x + \theta_1 (L - I_n)x \\ &= \theta_0 x - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x \end{aligned}$$

Graph Convolutional Network (GCN)

First-order approximation of ChebNet
K=1, λ_{max}=2로 설정하면 아래와 같이 유도됨.

$$\tilde{L} = \frac{2L}{\lambda_{max}} - I_n, \qquad Ls = I - D^{-1/2}AD^{-1/2}$$

<Chebyshev Polynomial>

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$$
$$T_0(x) = 1 \qquad T_1(x) = x$$

$$\begin{aligned} g_{\theta^*}x &= \sum_{k=0}^{K} \theta_k T_k(\tilde{L})\, x \\ &= \theta_0 x + \theta_1 T_1(\tilde{L})x \\ &= \theta_0 x + \theta_1 (L - I_n)x \\ &= \theta_0 x - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x \end{aligned}$$

Set parameter $\theta = \theta_0' = -\theta_1'$

$$g_{\theta^*}x = \theta(I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})x$$

Graph Convolutional Network (GCN)

Renormalization Trick

$$I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \widetilde{D}^{-\frac{1}{2}} \tilde{A} \widetilde{D}^{-\frac{1}{2}},\ \tilde{A} = \mathrm{A} + I_n \text{ and } \widetilde{D}_{ii} = \textstyle\sum_j \tilde{A}_{ij}$$

$$g_{\theta^*} x = \theta(I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) x$$

$$g_{\theta^*} x = \theta \widetilde{D}^{-\frac{1}{2}} \tilde{A} \widetilde{D}^{-\frac{1}{2}} x$$

이러한 layer를 여러 개
쌓아 deep model을
만들면 불안정 학습

Convolved signal matrix

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

Semi-supervised node classification

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

Z: filtering의 결과
Θ: 그래프의 모든 노드들에 동일하게 사용
→CNN의 filter(kernel)처럼 weight-sharing 관점에서 비슷함.

Propagation Rule

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

2-layers Model

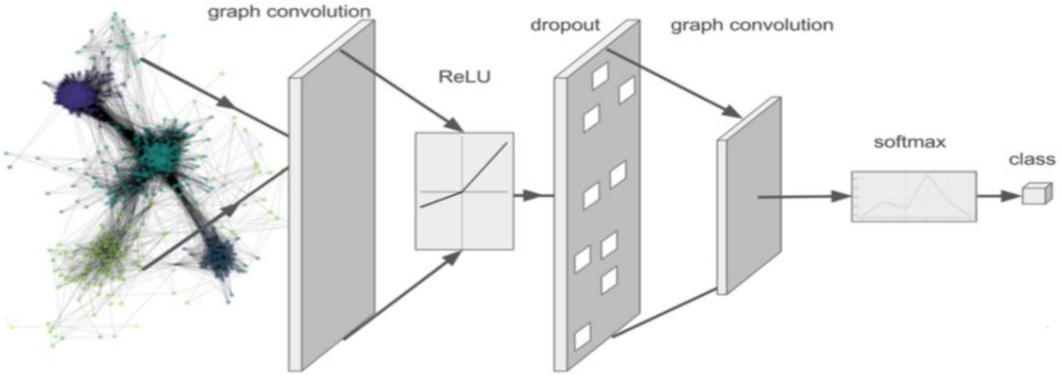
$$Z = softmax(\hat{A}ReLU(\hat{A}XW^{(0)})W^{(1)})$$

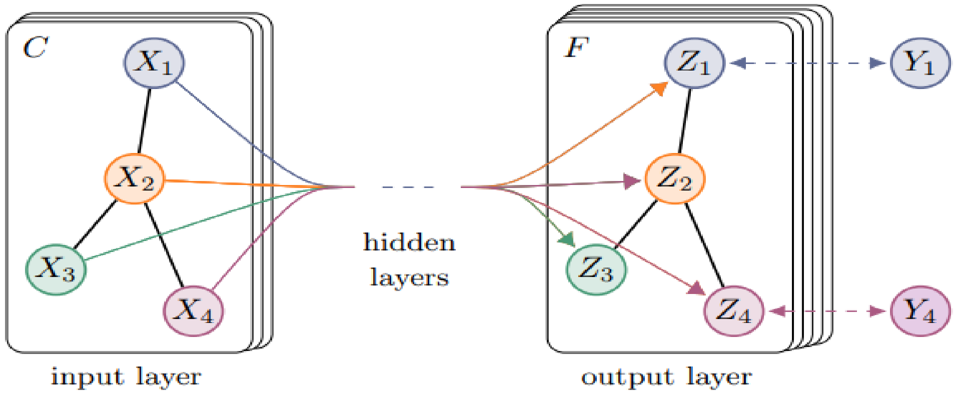
$$L = L_0 + \lambda L_{reg} \Rightarrow L = \sum_{l\in Y_L}\sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

- $\hat{A} = \widetilde{D}^{-\frac{1}{2}}\tilde{A}\ \widetilde{D}^{-\frac{1}{2}}$ 전처리 단계에서 계산
- 마지막 output layer에서 softmax를 각 행렬에 적용해줌
- Loss function의 변화: Label이 있는 node들에 대해서만 cross-entropy 계산, gradient descent 통해 $W^{(0)}$, $W^{(1)}$update

Layer-Wise Linear Model

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





(a) Graph Convolutional Network

```python
class GCN(nn.Module):
    def __init__(self, nfeat, nhid, nclass, dropout):
        super(GCN, self).__init__()

        self.gc1 = GraphConvolution(nfeat, nhid)
        self.gc2 = GraphConvolution(nhid, nclass)
        self.dropout = dropout

    def forward(self, x, adj):
        x = F.relu(self.gc1(x, adj))
        x = F.dropout(x, self.dropout, training=self.training)
        x = self.gc2(x, adj)
        return F.log_softmax(x, dim=1)
```

Experiments & Results

Datasets

Table 1: Dataset statistics, as reported in Yang et al. (2016).

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

Semi-Supervised Node Classification

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

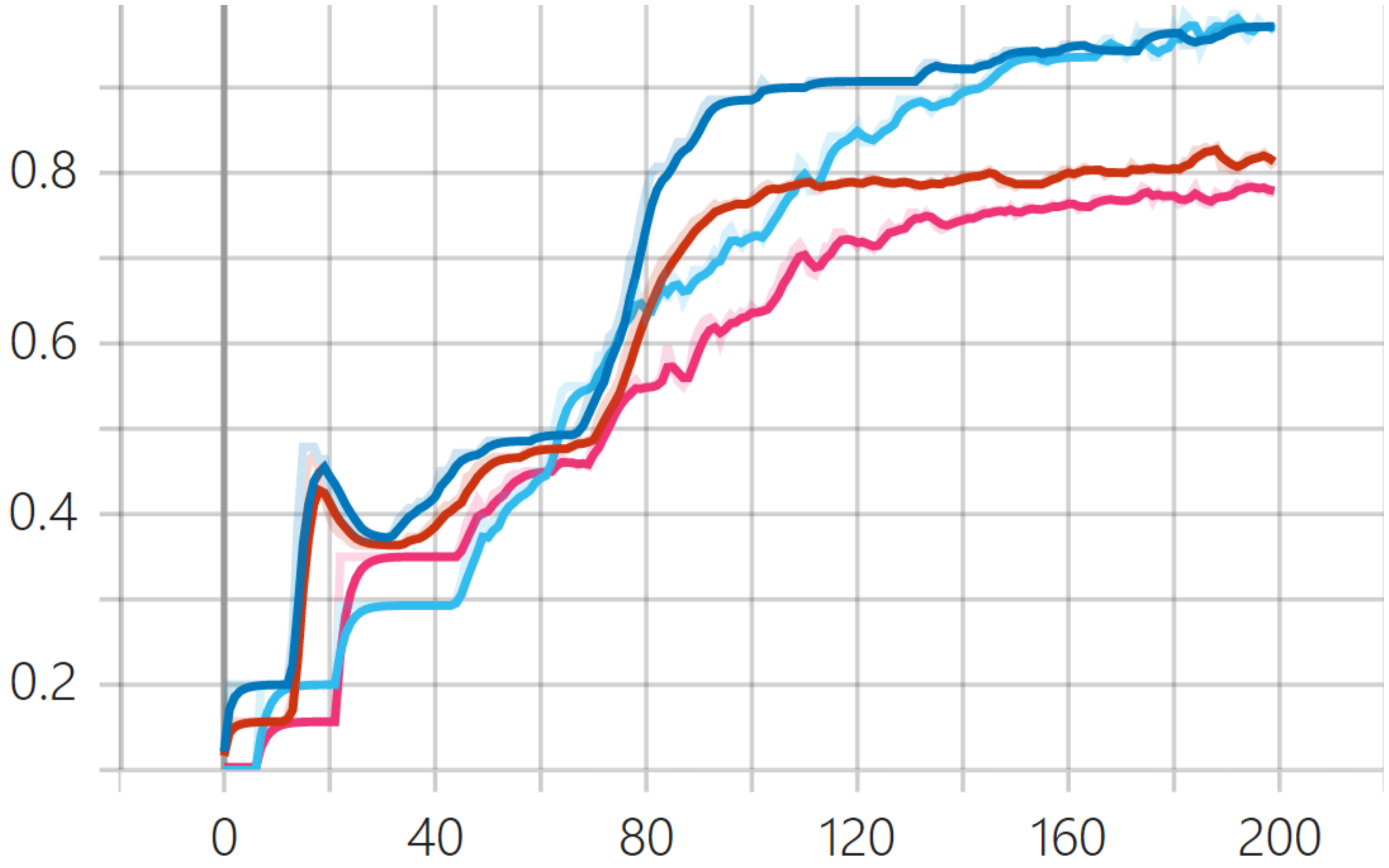
Experiments & Results

Evaluation of Propagation Model

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
1st-order model (Eq. 6)		$X\Theta_0 + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X \Theta_1$	68.3	80.0	77.5
Single parameter (Eq. 7)		$(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) X \Theta$	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**
1st-order term only		$D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X \Theta$	68.7	80.5	77.8
Multi-layer perceptron		$X\Theta$	46.5	55.1	71.4

Implementation



Dataset: Cora
Epoch: 200

Name	Smoothed	Value	Step	Time	Relative
runs\GCN_layer_2_experiment_1\accuracy_acc_train	0.9714	0.9714	199	Mon Jun 27, 19:21:47	1s
runs\GCN_layer_2_experiment_1\accuracy_acc_val	0.8131	0.8067	199	Mon Jun 27, 19:21:47	1s
runs\GCN_layer_3_experiment_1\accuracy_acc_train	0.9685	0.9643	199	Sat Jun 25, 17:24:23	2s
runs\GCN_layer_3_experiment_1\accuracy_acc_val	0.7788	0.7767	199	Sat Jun 25, 17:24:23	2s

Conclusion

Semi-Supervised Model

Proposed renormalized model offers both improved efficiency(fewer parameters and operations) and better predictive performance on a number of dataset compared to other models

Limitations and Future Work

Memory requirement – With full batch gradient descent, memory requirement grows linearly in the size of the dataset. →Use mini-batch

Directed edges and edge features – Model is limited to undirected graphs

Limiting assumptions – assume locality (dependence on the K^{th} order neighborhood for a K layers) and equal importance of self-connections vs. edges to neighboring edges.→ weight on importance

Conclusion

- Laplacian matrix를 적용하는 것은 Graph Convolution을 말한다.
- GCN은 Chebynet을 First-order approximation을 통해 계산 간소화 및 local feature를 반영한다
- GCN은 그래프 구조와 노드 특징을 Encoding할 수 있고, 더욱 정확하고 빠르게 Semi-supervised classification을 할 수 있다.

Appendix

Proof

$For\ x \in R^N, L\ is\ Positive\ semi-definite$

$$x^T L x = x^T D x - x^T W x = \sum_i D_{ii} x_i^2 - \sum_{i,j} x_i W_{ij} x_j$$

$$= \frac{1}{2}\left(2\sum_i D_{ii} x_i^2 - 2\sum_{i,j} W_{ij} x_i x_j\right)$$

$$\overset{(a)}{=} \frac{1}{2}\left(2\sum_i \left\{\sum_j W_{ij}\right\} x_i^2 - 2\sum_{i,j} W_{ij} x_i x_j\right)$$

$$\overset{(b)}{=} \frac{1}{2}\left(\sum_{i,j} W_{ij} x_i^2 + \sum_{i,j} W_{ij} x_j^2 - 2\sum_{i,j} W_{ij} x_i x_j\right)$$

$$= \frac{1}{2}\sum_{i,j} W_{ij}(x_i - x_j)^2 \geq 0$$

$$\mu = sup_{\|x\|=1} I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \leq 2$$

By Courant-Fischer theorem

M = $I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$: real-symmetric matrix

M = 2L-L , L: Positive Semi-definite matrix

References

- Official Pytorch GCN Implementation by Kipf et al., https://github.com/tkipf/pygcn
- GCN readings, https://tkipf.github.io/graph-convolutional-networks/
- Wu, Zonghan, et al. "A comprehensive survey on graph neural networks." IEEE transactions on neural networks and learning systems (2020
- Background contents, https://harryjo97.github.io/categories/
- 고려대 김성범 교수님 강의자료(CNN)
- cs224w, cs231n Slides
- 딥러닝 홀로서기 lecture9 slide, Standalone-DeepLearning/Lec9 at master · heartcored98/Standalone-DeepLearning · GitHub
- PR- Semi-supervised Classification with Graph Convolutional Networks, https://www.youtube.com/watch?v=F-JPKccMP7k&t=1710s
- 푸리에 변환: https://ralasun.github.io/deep%20learning/2021/02/15/gcn/
- Paper review_Jonghun Choi: Spectral-based Graph Convolutional Networks, http://dsba.korea.ac.kr/seminar/?mod=document&uid=1329
- Seminar_Semi-supervised classification with graph convolutional networks – presenter Sukwon Yun, Sungwon Kim

EOD