Semi-supervised classification with graph convolutional networks

Kipf, Thomas N., and Max Welling., ICLR 2017.

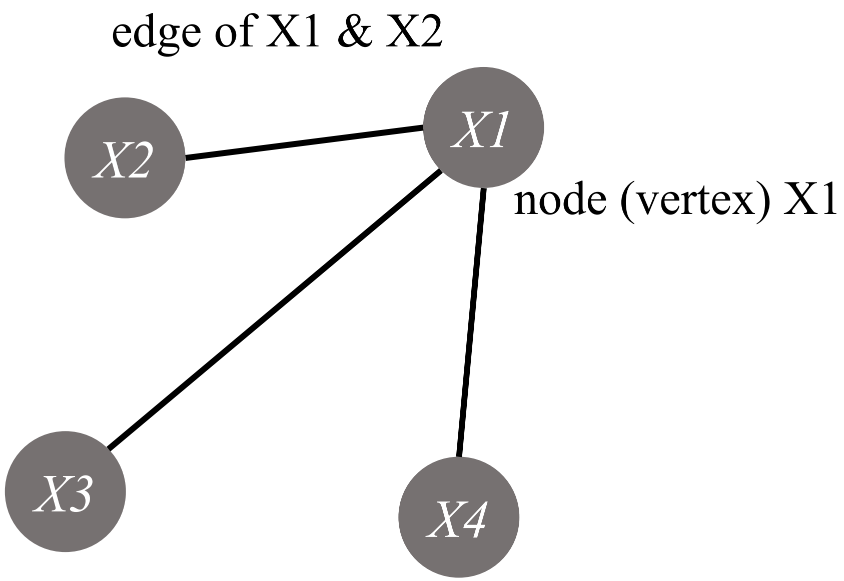
2020-05-14

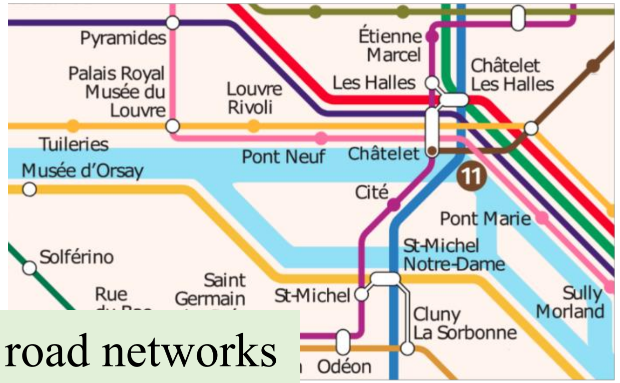
KIM INA (dodary0214@gmail.com)

Contents

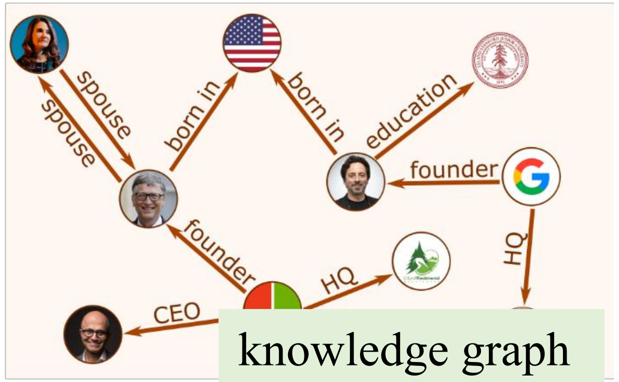
Graph

- Graph $G = (V, E)$
- A set of vertices $V = \{X_1, \ldots, X_n\}$
- A set of edges $E = \{E_{ij}\}, Xi, Xj \in V$

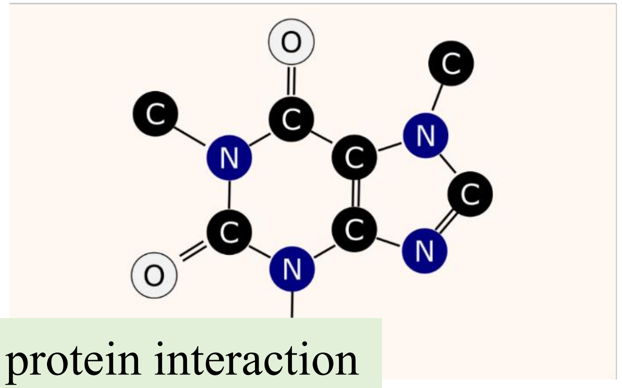




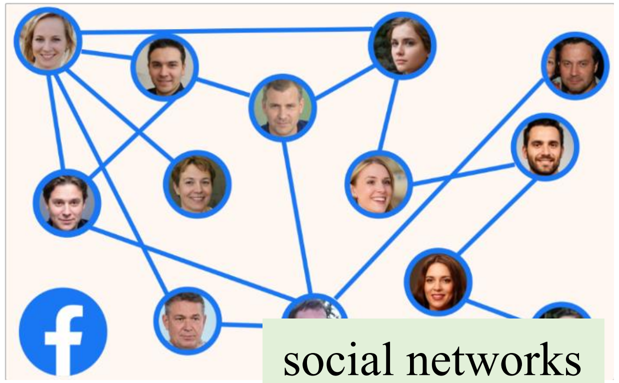
road networks



knowledge graph



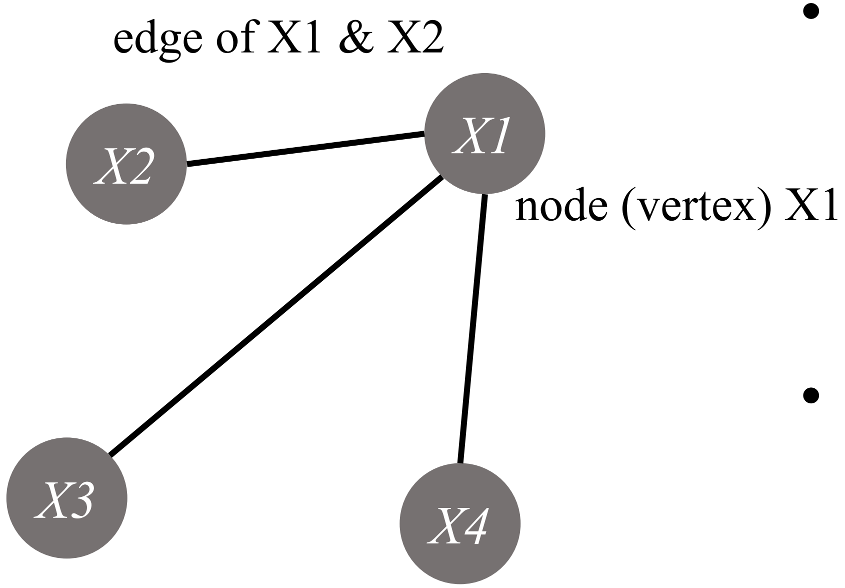
protein interaction


social networks

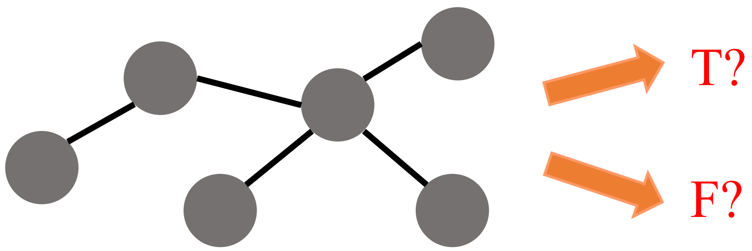
https://towardsdatascience.com/graph-convolutional-networks-deep-99d7fee5706f

Graph Tasks

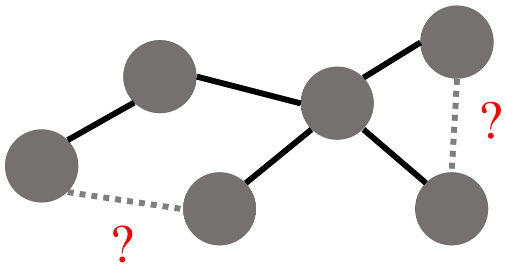
- Graph $G = (V, E)$
- A set of vertices $V = \{X_1, \ldots, X_n\}$
- A set of edges $E = \{E_{ij}\}, Xi, Xj \in V$



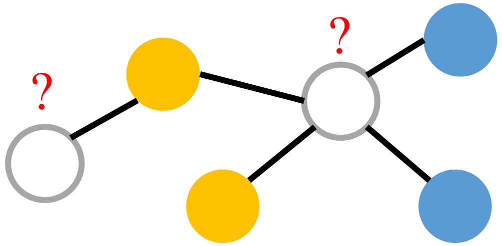
- Graph-level: graph classification



- Edge-level: edge classification, link prediction



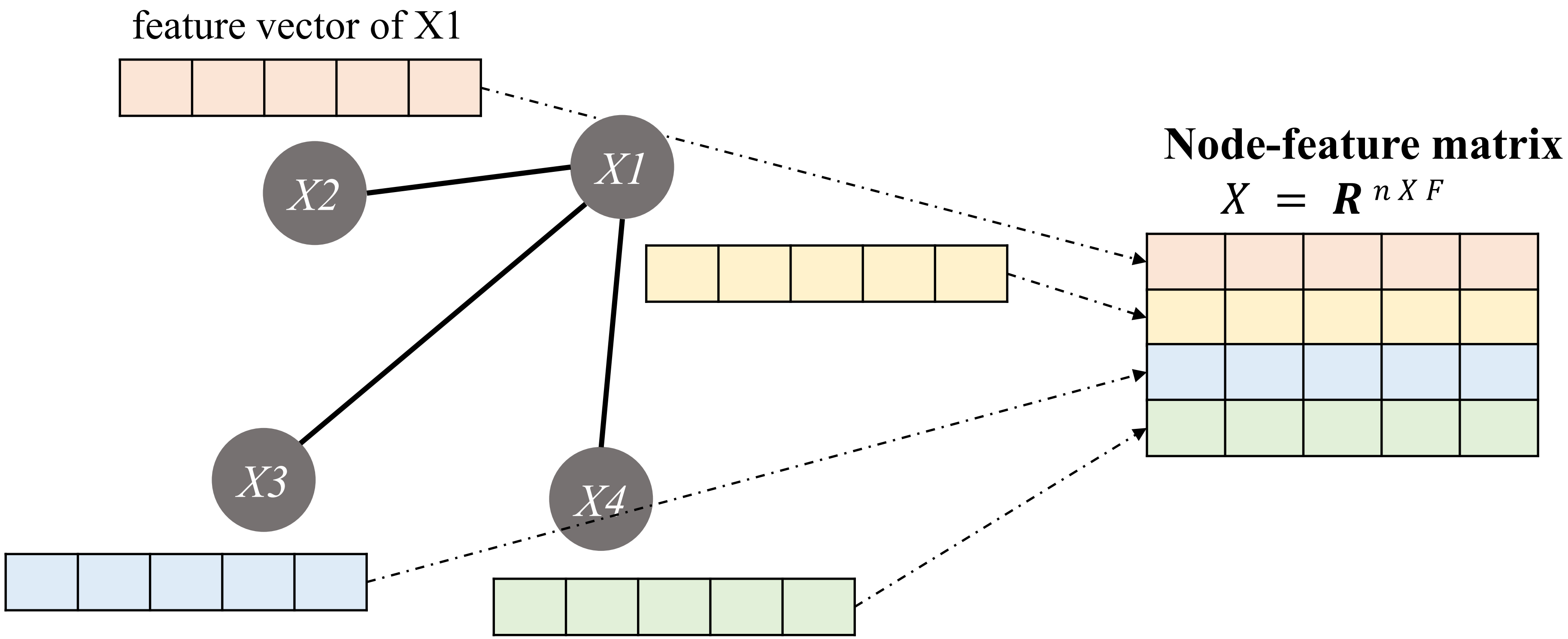
- Node-level: node classification



https://towardsdatascience.com/graph-convolutional-networks-deep-99d7fee5706f

Graph Representation – Node-Feature Matrix

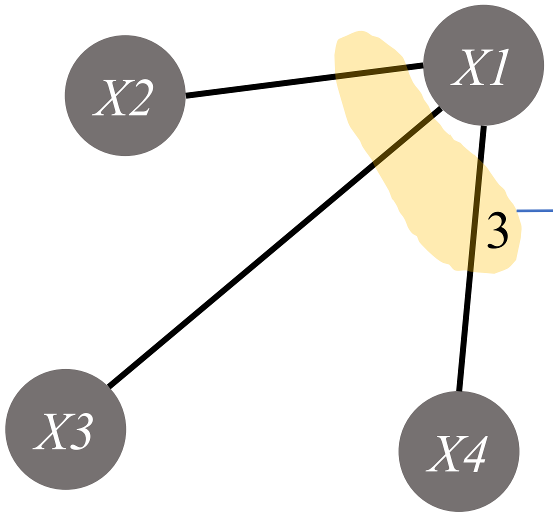
Representation structures : **Node-feature matrix**, Adjacency matrix, Degree matrix, Laplacian matrix



Graph Representation – Graph Structure Representation

Representation structures : Node-feature matrix, **Adjacency matrix, Degree matrix, Laplacian matrix**

X1 → X2, X3, X4



Adjacency matrix *A*

	X1	X2	X3	X4
X1	0	1	1	1
X2	1	0	0	0
X3	1	0	0	0
X4	1	0	0	0

Degree matrix *D*

	X1	X2	X3	X4
X1	3	0	0	0
X2	0	1	0	0
X3	0	0	1	0
X4	0	0	0	1

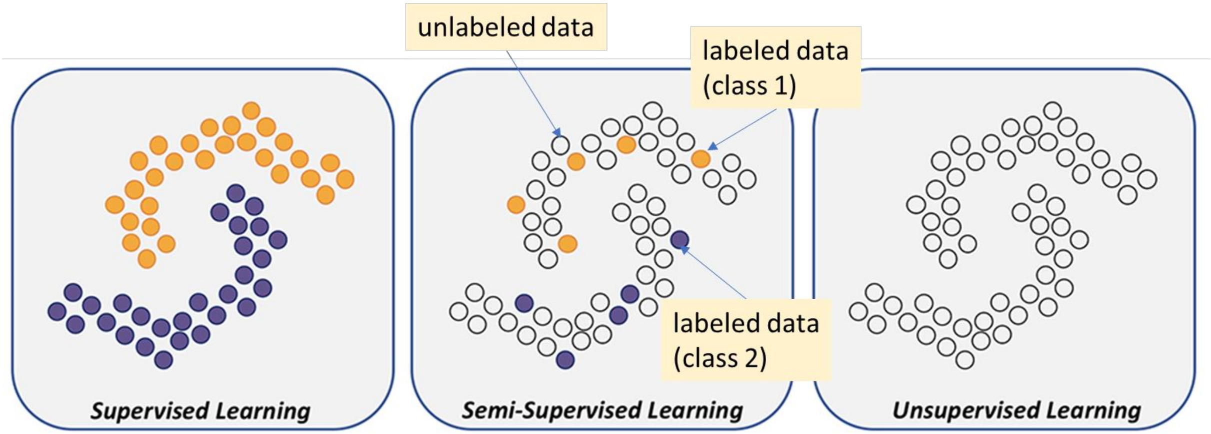
degree info.

Laplacian matrix $L = D - A =$

	X1	X2	X3	X4
X1	3	-1	-1	-1
X2	-1	1	0	0
X3	-1	0	1	0
X4	-1	0	0	1

Semi-Supervised Classification with Graph Convolutional Networks

Semi-Supervised Classification



https://blog.est.ai/2020/11/ssl/

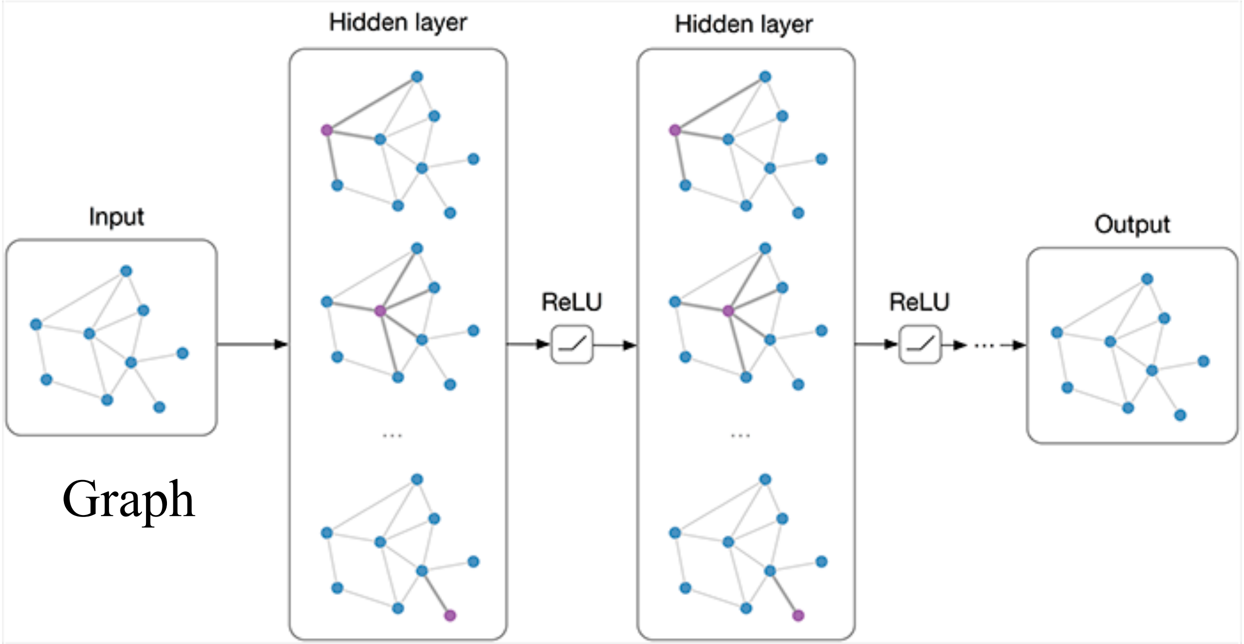
Graph Convolutional Network

GNN + CNN = GCN

Graph Neural Network (GNN)

GNN

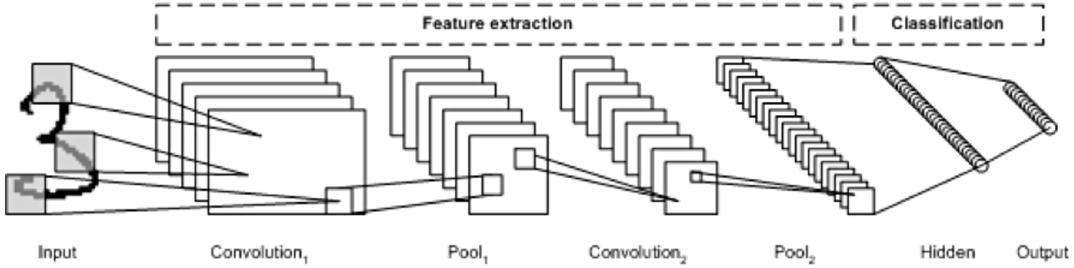
- 그래프 구조에서 사용하는 Neural Network로, Graph를 입력으로 받음
- Graph와 관련된 모든 Neural Network를 GNN으로 칭함



https://tkipf.github.io/graph-convolutional-networks/

Convolution

- CNN
  - 이미지에 대하여 Filter를 사용해 정보를 Aggregate
  - Convolution values (i.e., weight, filter)를 학습

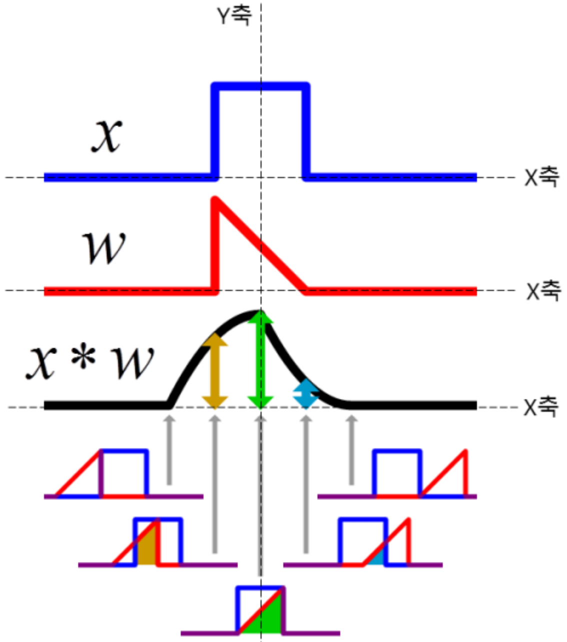


http://taewan.kim/post/cnn/

- Convolution (합성곱)
  - f, g 가운데 하나의 함수를 반전, 전이 시킨 후 다른 하나의 함수와 곱한 결과를 적분
  - 현재 합성곱의 값은 이전 시간의 결과를 포함

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

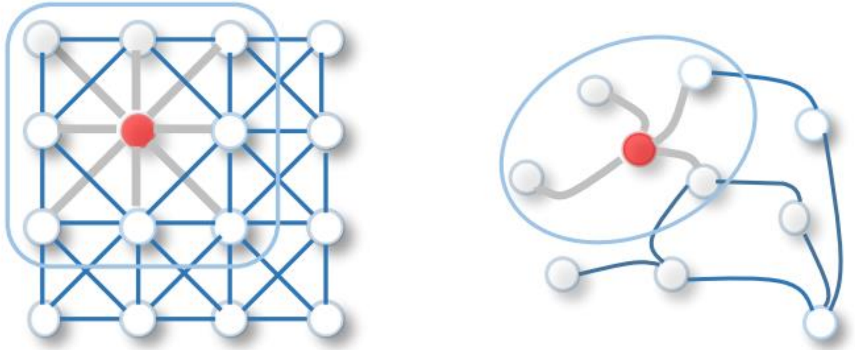
* convolution symbol



https://en.wikipedia.org/wiki/Convolution

Convolution on Graph

- Graph convolution
  - Convolution filter를 사용해서, 그래프 노드와 인접 노드간의 관계 계산 목적
  - 전체 데이터에서 Local feature 추출 목적
  - Filter들이 Spatial location에 따라 변하지 않음



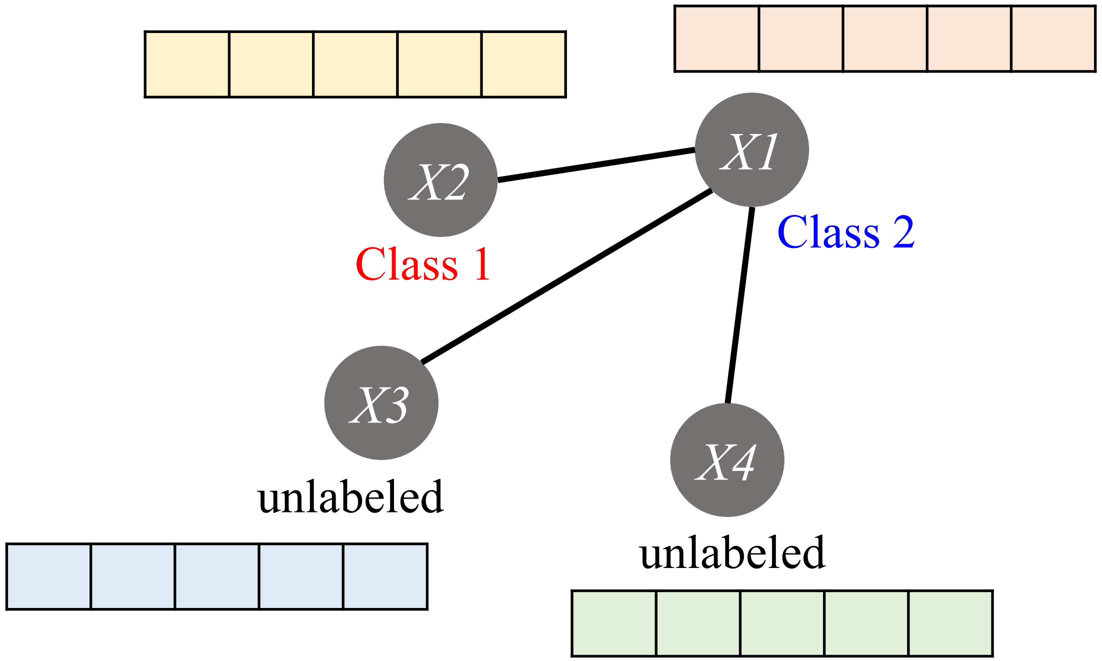
- Convolution theorem
  - 한 domain 의 convolution은 다른 domain의 point-wise multiplication과 같음 → Graph domain의 convolution은 Fourier domain의 point-wise multiplication과 같음
  - Convolution의 laplace 변환은 point-wise multiplication으로 변함

$$\mathcal{F}(f \star g) = \mathcal{F}\{f\} \odot \mathcal{F}\{g\} \qquad x *_G g = \mathcal{F}^{-1}(\mathcal{F}(x) \odot \mathcal{F}(g))$$

https://learnopencv.com/graph-convolutional-networks-model-relations-in-data/

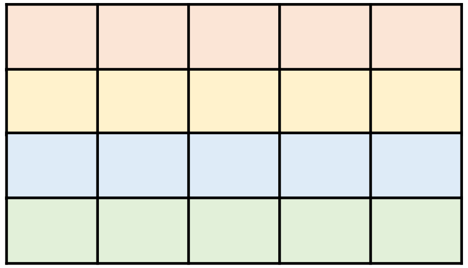
Graph-based Semi-Supervised Learning

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



Node-feature matrix

$X \in \boldsymbol{R}^{n \, X \, F}$



Adjacency matrix

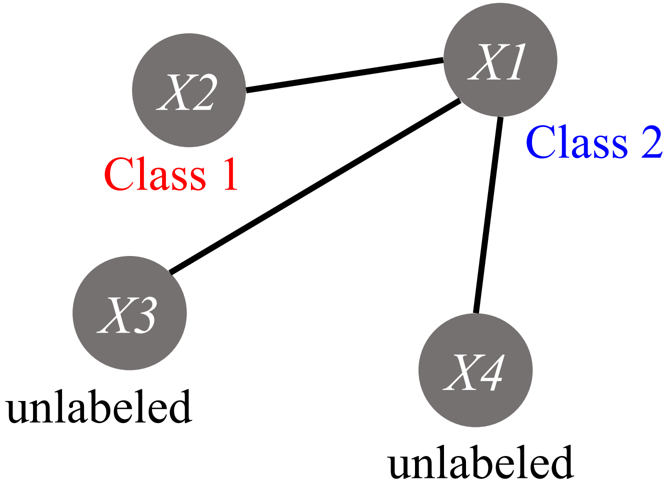
$A \in \boldsymbol{R}^{n \, X \, n}$

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

Graph-based Semi-Supervised Learning

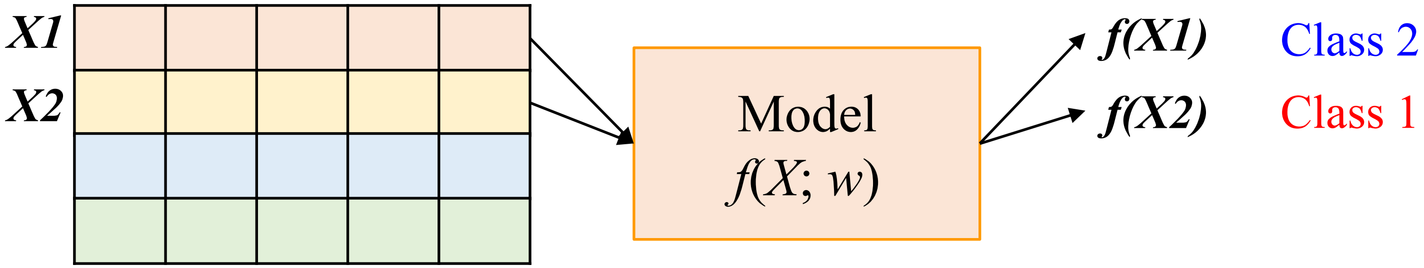
the supervised loss
w.r.t. **the labeled part** of the graph

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



Node-feature matrix

$X \in \boldsymbol{R}^{n \times F}$

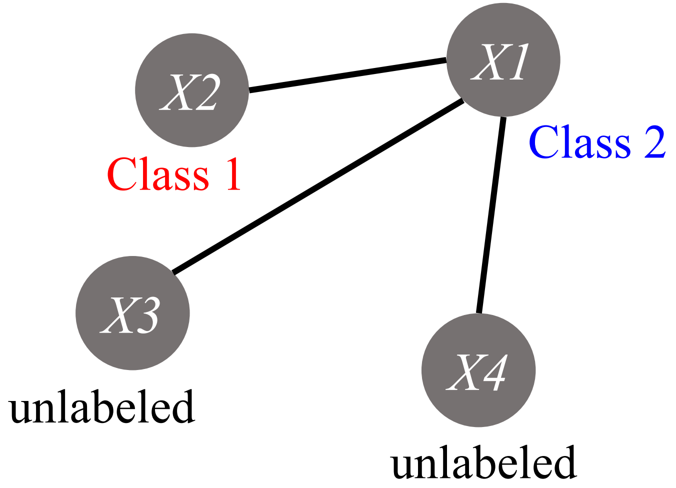


Graph-based Semi-Supervised Learning

graph laplacian regularization term

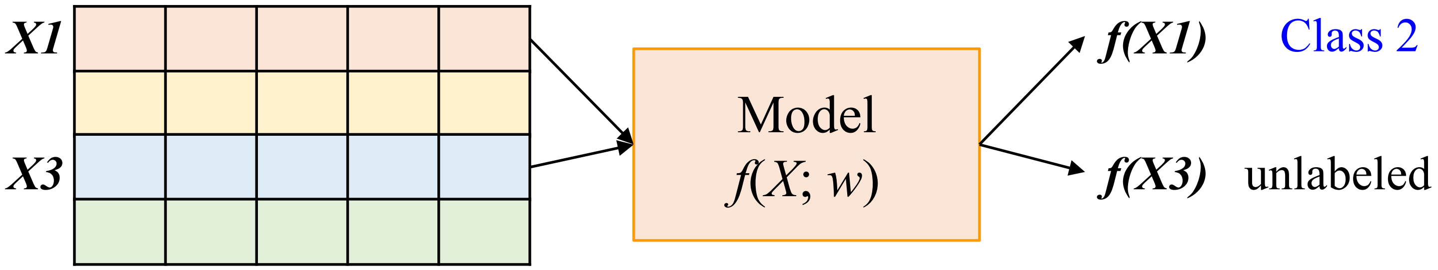
$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{\text{reg}}$$

$$\mathcal{L}_{\text{reg}} = \sum_{i,j} A_{ij} \| \underbrace{f(X_i)}_{f(X1)} - \underbrace{f(X_j)}_{f(X3)} \|^2 = f(X)^\top \Delta f(X)$$



Node-feature matrix

$X \in \boldsymbol{R}^{n \times F}$



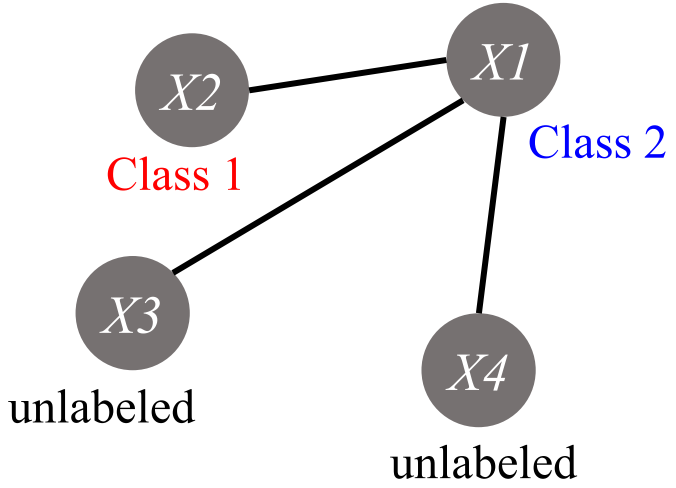
Graph-based Semi-Supervised Learning

graph laplacian regularization term

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

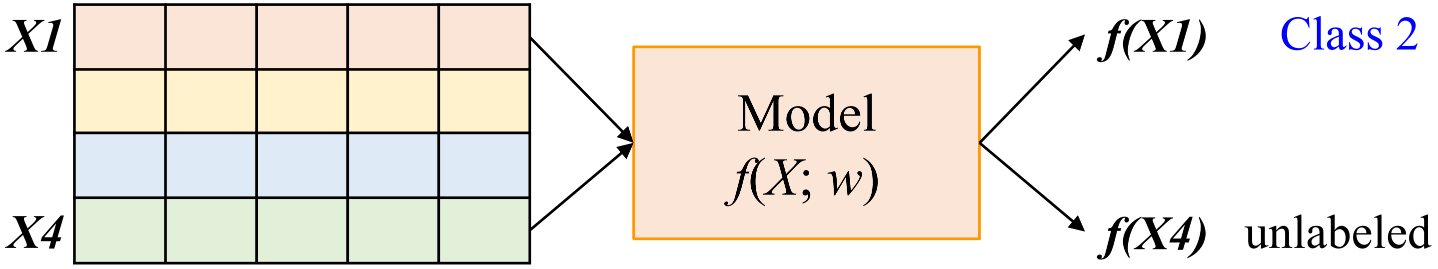
$$\mathcal{L}_{\text{reg}} = \sum_{i,j} A_{ij} \| f(X_i) - f(X_j) \|^2 = f(X)^\top \Delta f(X)$$

f(X1) ***f(X4)***



Node-feature matrix

$X \in \boldsymbol{R}^{n \times F}$



Contributions

- Graph-based Semi-Supervised Learning
  - ✓ **Assumption**: connected nodes in the graph → share same labels
  - ✓ **Limitation**: could not contain addition information of the graph

- This work
  - ✓ Encode the graph structure using a neural network model $\boldsymbol{f}(\boldsymbol{X}, \boldsymbol{A})$ $\boldsymbol{X}$: node-feature matrix, $\boldsymbol{A}$: Adjacency matrix
  - ✓ Can be used for fast and scalable semi-supervised classification of nodes in a graph

Graph Convolutional Networks (GCN)

Graph convolution (Layer-wise propagation rule)

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

- $H^{(l)}$: hidden state of the l-th layer
- $\tilde{A}$: A (adjacency matrix) + I(Identity matrix)
- $\widetilde{D}$: $\widetilde{D}_{ii} = \sum_j \tilde{A}_{ij}$
- $W^{(l)}$: layer-specific trainable weight matrix
- σ: activation function $ReLU(\cdot)$

Multi-layers

Layer 1

$$Z = f(X, A) = \underbrace{\text{softmax}\left(\hat{A}\,\overbrace{\text{ReLU}\left(\hat{A}XW^{(0)}\right)}^{\text{Layer 1}}W^{(1)}\right)}_{\text{Layer 2}}$$

Layer 2

Spectral Graph Convolutions

➢ Graph에 대한 Spectral convolution

- Graph domain → Fourier domain
- singal $x \in R^N$ × filter $g_\theta = diag(\theta) / \theta \in R^N$

Convolution theorem

$$\mathcal{F}(f \star g) = \mathcal{F}\{f\} \odot \mathcal{F}\{g\}$$

singal x × filter g_θ in Fourier domain

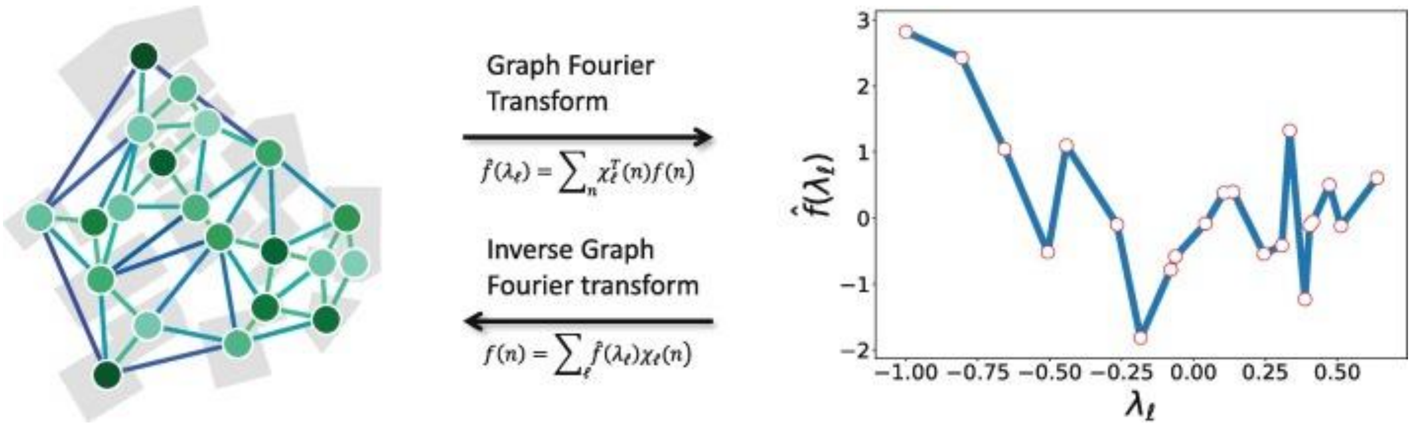
$$g_\theta * x = U g_\theta^* U^T x$$

➢ Graph Fourier Transform

- Graph를 signal / filter의 point-wise multiplication으로 적용하기 위해서, Fourier transform 적용 필요

Graph Fourier Transform

https://www.sciencedirect.com/science/article/pii/S0924271619300437



✓ Graph signal(Graph node features)을 Frequency(Features 간 차이)로 분해

- Fourier Transform
  - 어떤 형태의 주파수가 Signal에 어느 정도로 포함되어 있는지 측정 가능
- Graph Fourier Transform
  - 어떤 형태의 그래프 관계가 Signal에 어느 정도로 포함되어 있는지 측정 가능
- <u>GFT → Filtering → IGFT</u> = Laplacian matrix를 Feature vector와 곱하는 것과 같음
  - Laplacian의 Eigen-vector가 Fourier Basis 역할을 함
  - GFT를 적용하여 Frequency가 낮은 관계들 (i.e., 유사 노드)을 우선적으로 반영

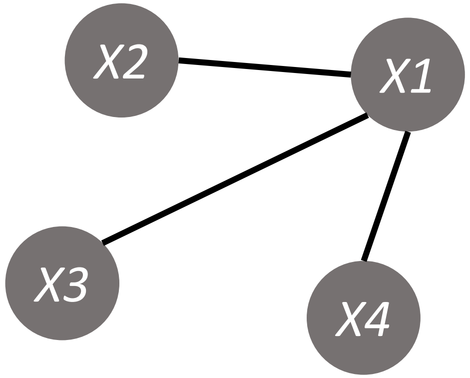
Graph Laplacian (Laplacian Matrix)

- Graph laplacian:
  - 한 노드의 특징 → 해당 노드와 연결된 아웃노드와의 관계 관점에 표현
  - 이웃 노드와의 차이 (변화)를 고려한 것 → 그래프의 유용한 특성 반영

$$(\Delta\phi)(v) = \sum_{w:d(w,v)=1} [\phi(v) - \phi(w)]$$

Central node와 Neighbor node 간 차이

$$\Delta\phi(X1) = 3\phi(X1) - \phi(X2) - \phi(X3) - \phi(X4)$$
$$\Delta\phi(X2) = \phi(X2) - \phi(X1)$$
$$\Delta\phi(X3) = \phi(X3) - \phi(X1)$$
$$\Delta\phi(X4) = \phi(X4) - \phi(X1)$$



$$L = (\Delta\phi)(X) = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \phi(X1) \\ \phi(X2) \\ \phi(X3) \\ \phi(X4) \end{bmatrix} = D - A$$

Graph Laplacian (Laplacian Matrix)

singal x × filter g_θ in Fourier domain

$$g_\theta * x = U g_\theta^* U^T x$$

Eigen decomposition

$$L = U \Lambda U^T$$

Laplacian

$$x \longrightarrow U^T x \longrightarrow g_\theta^* U^T x \longrightarrow \underline{U g_\theta^*} U^T x$$

Signal

Graph fourier transform

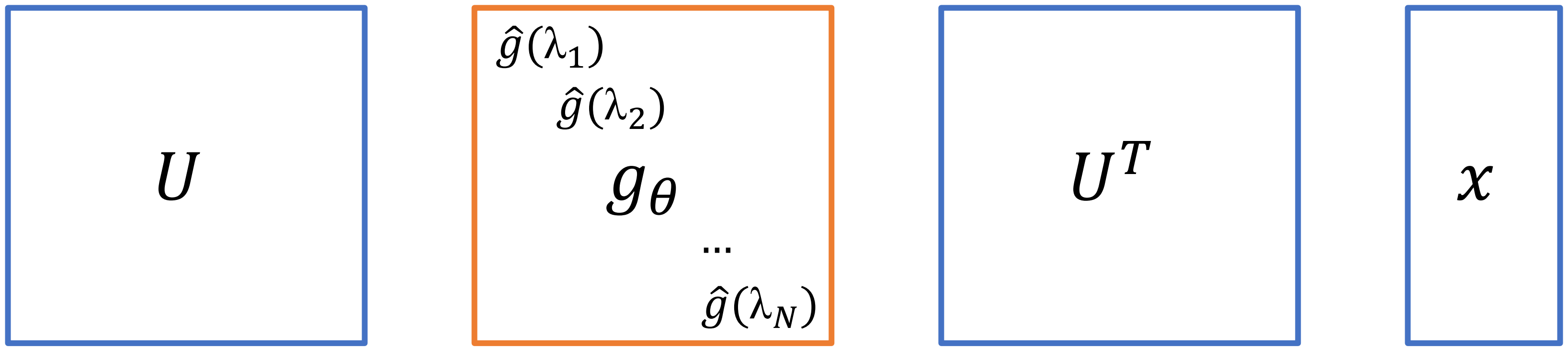
Filtering
$g_\theta^* = g_\theta^*(\Lambda)$

Inverse Graph fourier transform

Graph Laplacian (Laplacian Matrix)

singal $x \times$ filter g_θ in Fourier domain

$$g_\theta * x = U g_\theta^* U^T x$$



→ filter g_θ is non-parametric
High computation for eigen-decomposition

Chebyshev Polynomials

$$\text{filter } g'_\theta(\Lambda) = \sum_{k=0}^{K} \theta'_k \Lambda^k = \theta'_0 \Lambda^0 + \theta'_1 \Lambda^1 + \cdots + \theta'_K \Lambda^K$$

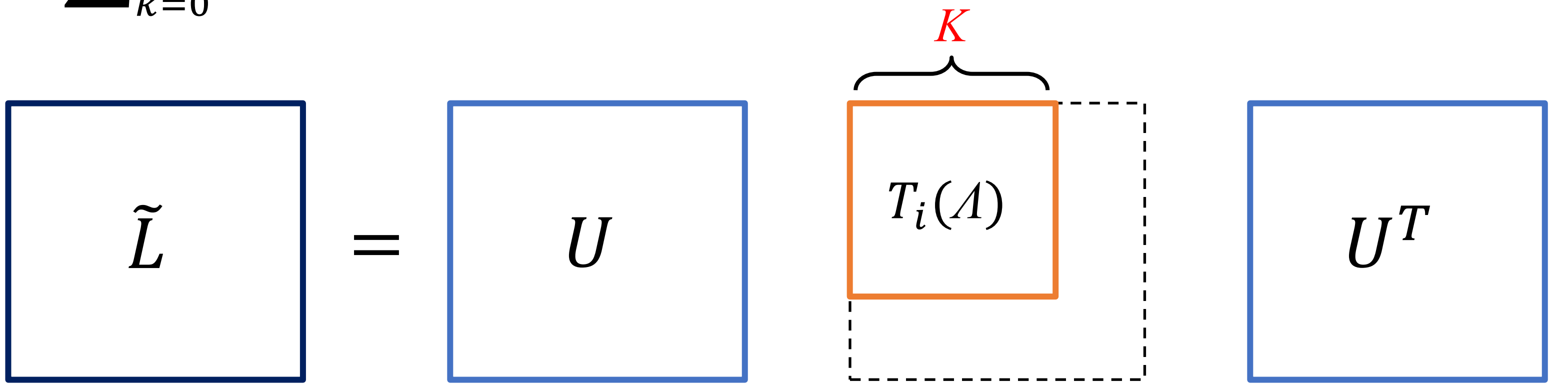
$$\Rightarrow g_\theta * x = U g'_\theta U^T x = U \left(\sum_{k=0}^{K-1} \theta_k \Lambda^k \right) U^T x = \sum_{k=0}^{K-1} \theta_k U \Lambda^k U^T x = \underline{\sum_{k=0}^{K-1} \theta_k L^k x}$$

K-Localized (K: neighbors)

Chebyshev Polynomials $T_k(x)$ 활용하여 근사화

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

$$\Rightarrow g'_\theta * x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L}) x = \theta'_0 T_0(\tilde{L}) x + \theta'_1 T_1(\tilde{L}) x + \theta'_2 T_2(\tilde{L}) x + \cdots + \theta'_k T_k(\tilde{L}) x$$



Layer-wise Linear Model

Chebyshev Polynomials

$$\Rightarrow g'_\theta * x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L})x = \underline{\theta'_0 T_0(\tilde{L})x + \theta'_1 T_1(\tilde{L})x} + \theta'_2 T_2(\tilde{L})x + \cdots + \theta'_k T_k(\tilde{L})x$$

Chebyshev Polynomials (K=1)

$$\begin{aligned}\Rightarrow g'_\theta * x &\approx \theta'_0 T_0(\tilde{L})x + \theta'_1 T_1(\tilde{L})x \\ &\approx \theta'_0 \cdot 1 \cdot x + \theta'_1 (L - I_N)x \\ &\approx \theta'_0 \cdot 1 \cdot x + \theta'_1 D^{-1/2} A D^{-1/2} x \\ &\approx \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) x\end{aligned}$$

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

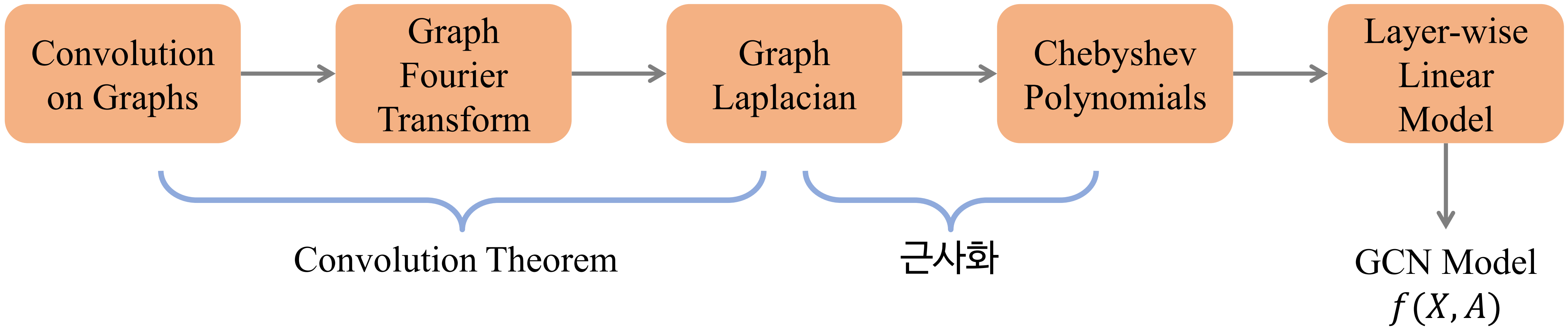
$$L = I_N - D^{-1/2} A D^{-1/2}$$

$$\theta = \theta'_0 = -\theta'_1$$

Renormalization

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

Summary


Convolution on Graphs
Graph Fourier Transform
Graph Laplacian
Chebyshev Polynomials
Layer-wise Linear Model
Convolution Theorem
근사화
GCN Model
$f(X, A)$

Graph Convolutional Networks (GCN)

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

Layer 1

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

Layer 2

→ **Two-layer convolutional neural network**

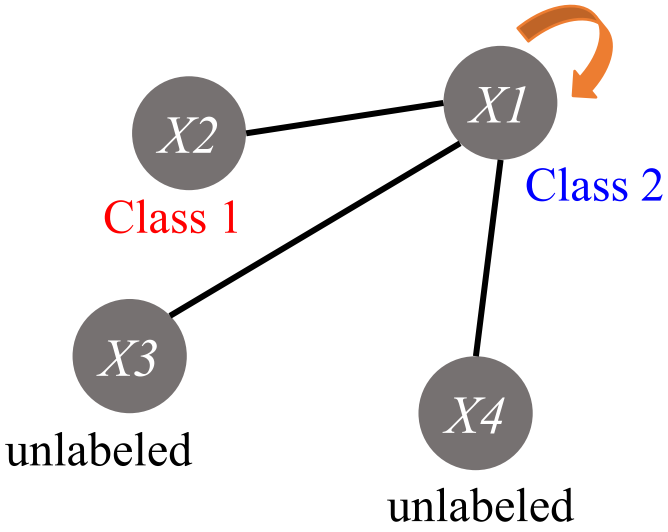
Loss function

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

- Cross-entropy error는 Labeled example에 대하여 진행
- $W^{(0)}, W^{(1)}$은 Gradient descent를 사용하여 훈련됨
- Full batch gradient descent 사용

Graph Convolutional Networks (GCN)

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



$$\underset{A(4\times 4)}{\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}} + \underset{I(4\times 4)}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} = \underset{\tilde{A}(4\times 4)}{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}}$$

self connections

$$\underset{D(4\times 4)}{\begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \rightarrow \underset{\widetilde{D}(4\times 4)}{\begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}}$$

Graph Convolutional Networks (GCN)

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

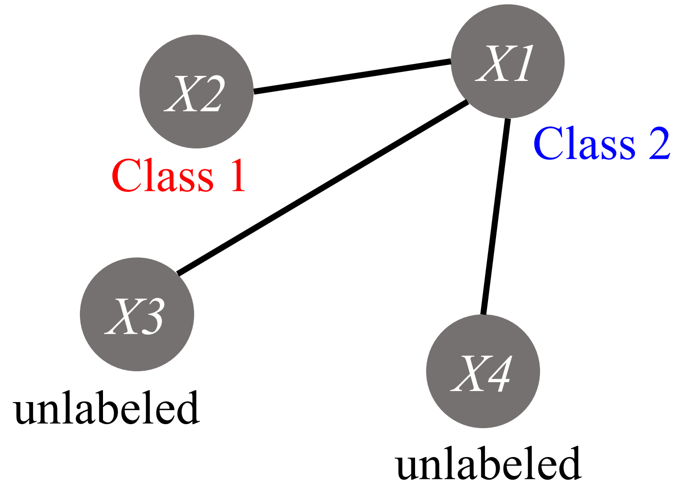
$X2$ has 2 edges

$$X1\text{-}X2 \rightarrow \boxed{\frac{1}{\sqrt{4}}} \times \boxed{\frac{1}{\sqrt{2}}} = \frac{1}{2\sqrt{2}}$$

$X1$ has 4 edges

$$\tilde{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

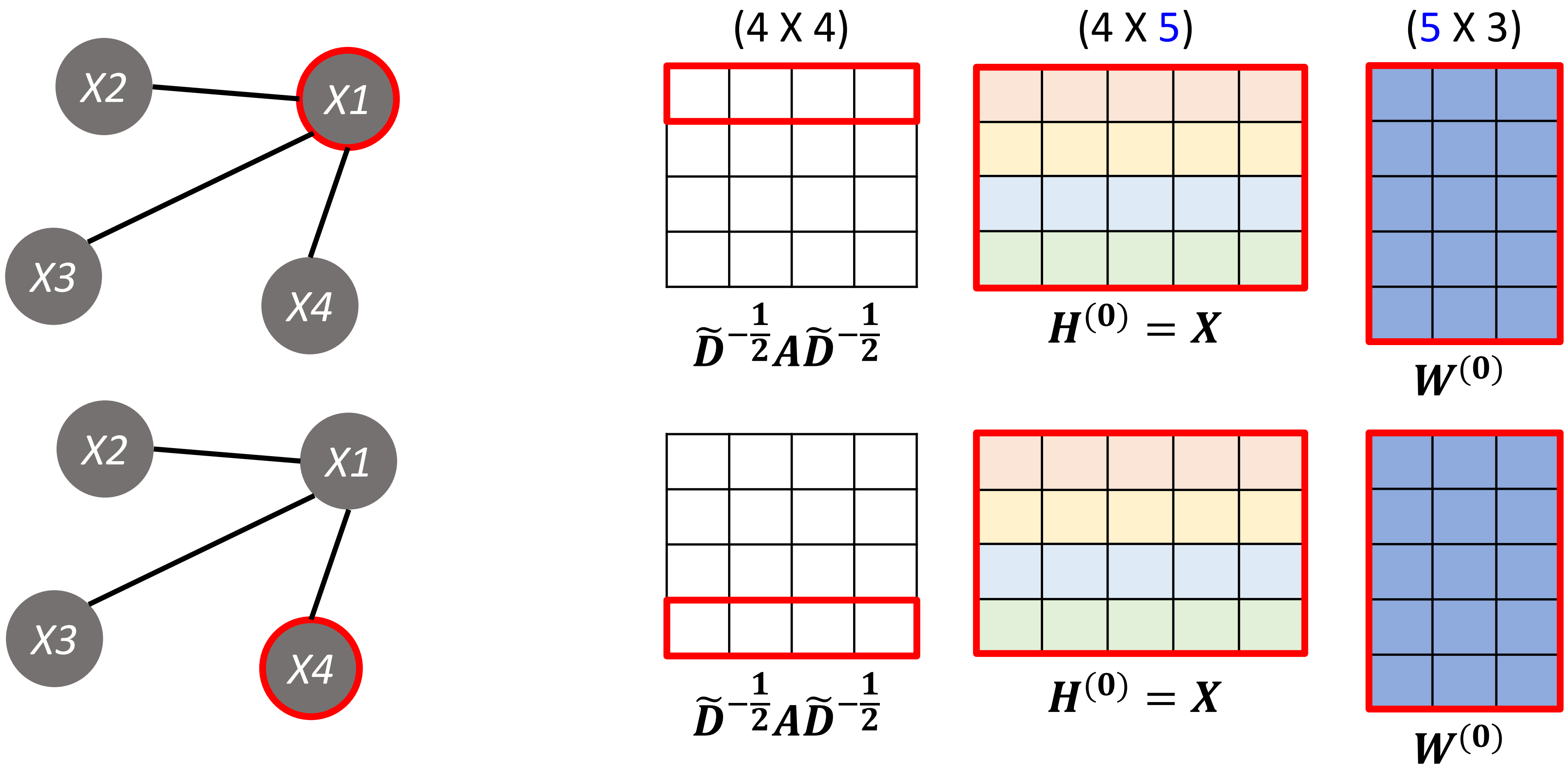
$$= \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{array}{c} \begin{array}{cccc} X1 & X2 & X3 & X4 \end{array} \\ \begin{bmatrix} \frac{1}{4} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\ \frac{1}{2\sqrt{2}} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2\sqrt{2}} & 0 & \frac{1}{2} & 0 \\ \frac{1}{2\sqrt{2}} & 0 & 0 & \frac{1}{2} \end{bmatrix} \end{array} \begin{array}{c} X1 \\ X2 \\ X3 \\ X4 \end{array}$$



$$X3\text{-}X3 \rightarrow \frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}} = \frac{1}{2}$$

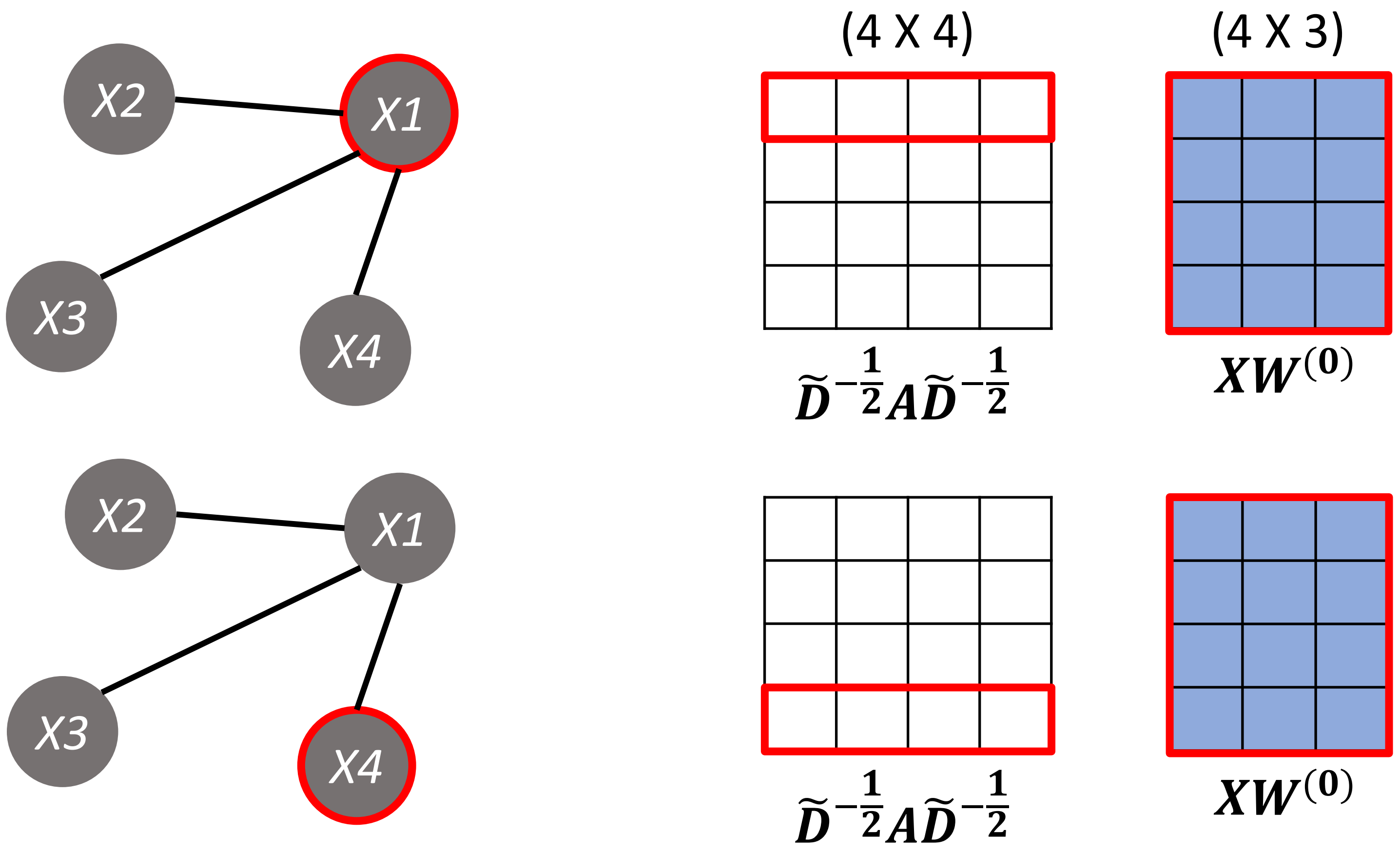
Graph Convolutional Networks (GCN)

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



Graph Convolutional Networks (GCN)

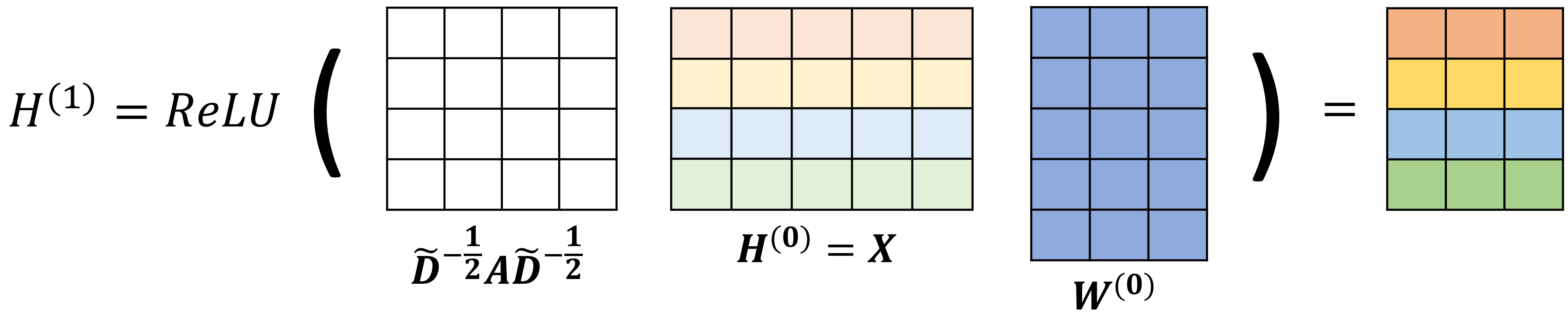
$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} A \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)})$$



Two-layer GCN

$$Z = f(X, A) = \mathrm{softmax}\left(\hat{A}\ \overset{\text{Layer 1}}{\boxed{\mathrm{ReLU}\left(\hat{A}XW^{(0)}\right)}}\ W^{(1)}\right)$$

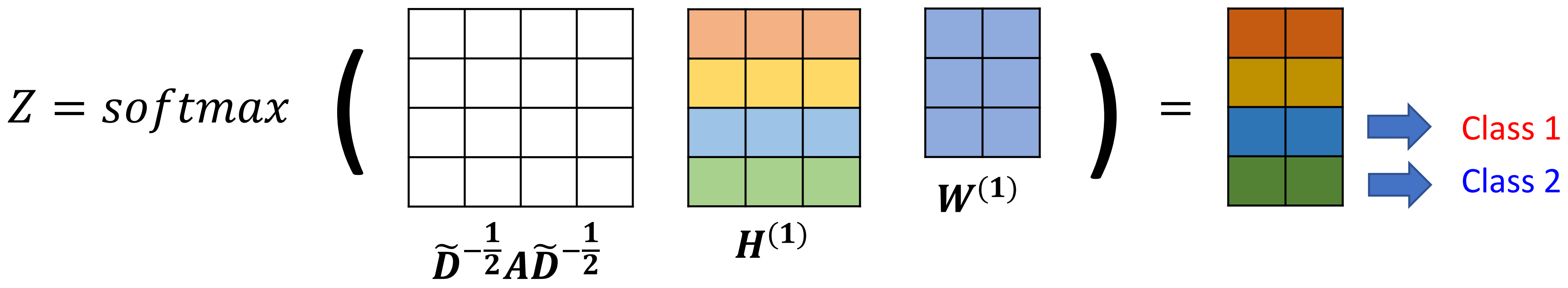
$$\begin{aligned} H^{(1)} &= \sigma(\tilde{D}^{-\frac{1}{2}}A\tilde{D}^{-\frac{1}{2}}H^{(0)}W^{(0)}) \\ &= ReLU\left(\tilde{D}^{-\frac{1}{2}}A\tilde{D}^{-\frac{1}{2}}H^{(0)}W^{(0)}\right) \\ &= ReLU(A^{\wedge}XW^{(0)}) \end{aligned}$$



Two-layer GCN

$$Z = f(X, A) = \overset{\text{Layer 2}}{\boxed{\mathrm{softmax}\left(\hat{A}\ \mathrm{ReLU}\left(\hat{A}XW^{(0)}\right) W^{(1)}\right)}}$$

$$\begin{aligned} Z &= f(X, A) \\ &= softmax(A^{\wedge} ReLU(A^{\wedge} X W^{(0)}) W^{(1)}) \\ &= softmax(A^{\wedge} H^{(1)} W^{(1)}) \end{aligned}$$



Datasets

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

Node: Documents
Edge: Citation Links
Label rate: training nodes / entire node
Node feature: sparse bag of words

Set-up and Baselines

- Set-up
  - ✓ Two-layer GCN (10-layer GCN in appendix)
  - ✓ A test set of 1,000 labeled examples
  - ✓ Train all models for 200 epochs using Adam
  - ✓ A learning rate of 0.01
  - ✓ Initialize weights using the initialization in Glorot & Bengio (2010)

- Baselines
  - ✓ Label propagation(LP)
  - ✓ Semi-supervised embedding (SemiEmb)
  - ✓ Manifold regularization (ManiReg)
  - ✓ Skip-gram based graph embeddings (DeepWalk)
  - ✓ Iterative classification algorithm (ICA)
  - ✓ Planetoid (Planetoid)

Semi-Supervised Node Classification

- Mean accuracy of 100 runs with random weight initialization
- Hyperparameters
  - Citeseer, Cora and Pubmed: 0.5 (dropout rate), 5·10−4 (L2 regularization) and 16 (number of hidden units)
  - NELL: 0.1 (dropout rate), 1·10−5 (L2 regularization) and 64 (number of hidden units)

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

Classification accuracy

Training time until convergence

Evaluation of Propagation Model

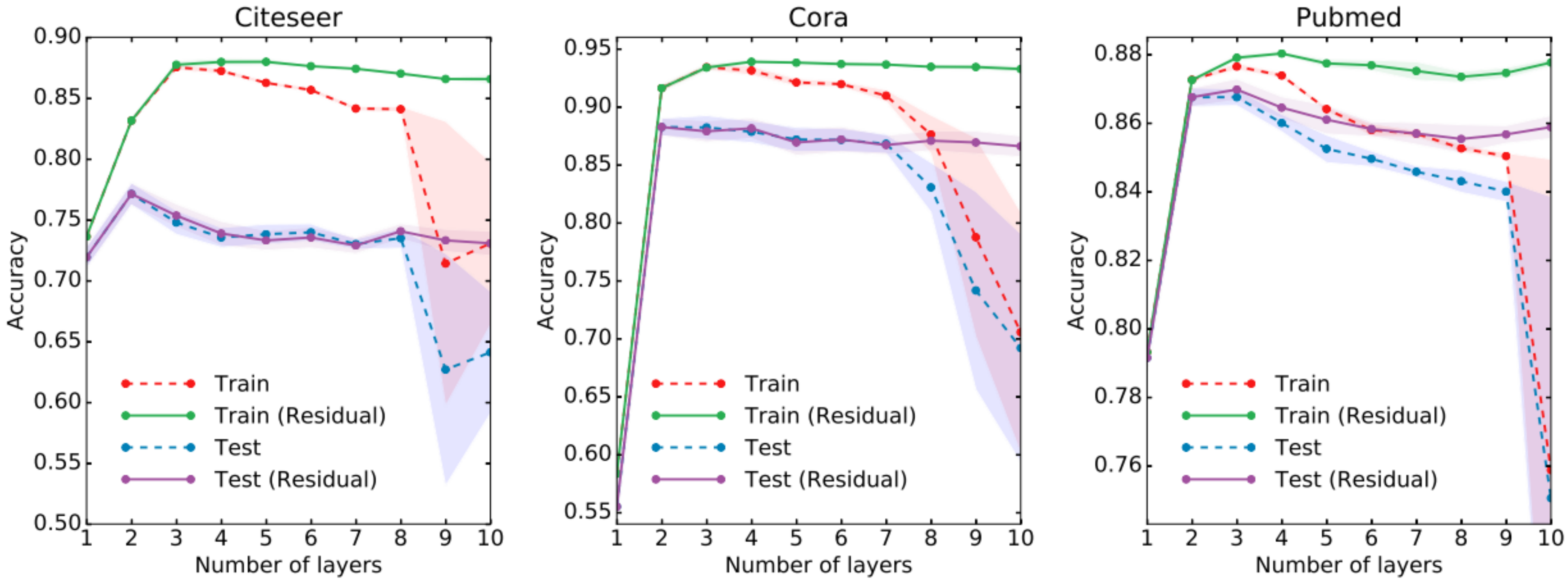
➢ Compare different variants of our proposed per-layer propagation model on the citation network datasets

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)		$X\Theta_0 + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta_1$	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**
1st-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

original model

Experiments on Model Depth

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



Conclusions and Limitations

- GCN model for semi-supervised classification on graph-structured data
  - Spectral graph convolutions
  - Layer-wise linear model

- Memory requirement
  - Full-batch gradient descent → Memory requirement grows
  - Mini-batch stochastic gradient descent

- Directed edges and edge features
  - Limited to undirected graphs (weighted or unweighted)

- Limiting assumptions
  - Equal importance of self-connections vs. edges to neighboring nodes

References

- [발표자료] Semi-supervised Classification with Graph Convolutional Networks, 윤훈상, 고려대학교.
- [발표자료] Spectral-based Graph Convolutional Networks, 최종현, 고려대학교.
- [발표자료] Graph Convolution Networks, 이민정, 고려대학교.
- Convolution, https://en.wikipedia.org/wiki/Convolution
- Semi-supervised learning, https://en.wikipedia.org/wiki/Semi-supervised_learning