Graph Attention Networks

이은경

목차

1. Abstract

1) 개요

1. Abstract

Graph convolutions

approximations

+ Masked self-attention layer

Graph Attention Networ(GAT)

장점

1) Costly Matrix Computation (inverse) 필요 없음

2) 그래프의 전체 구조를 미리 알 필요 없음

3) inductive 문제, transductive 문제 모두에서 쉽게 적용(데이터 구조에 따른 영향 적음)

성능

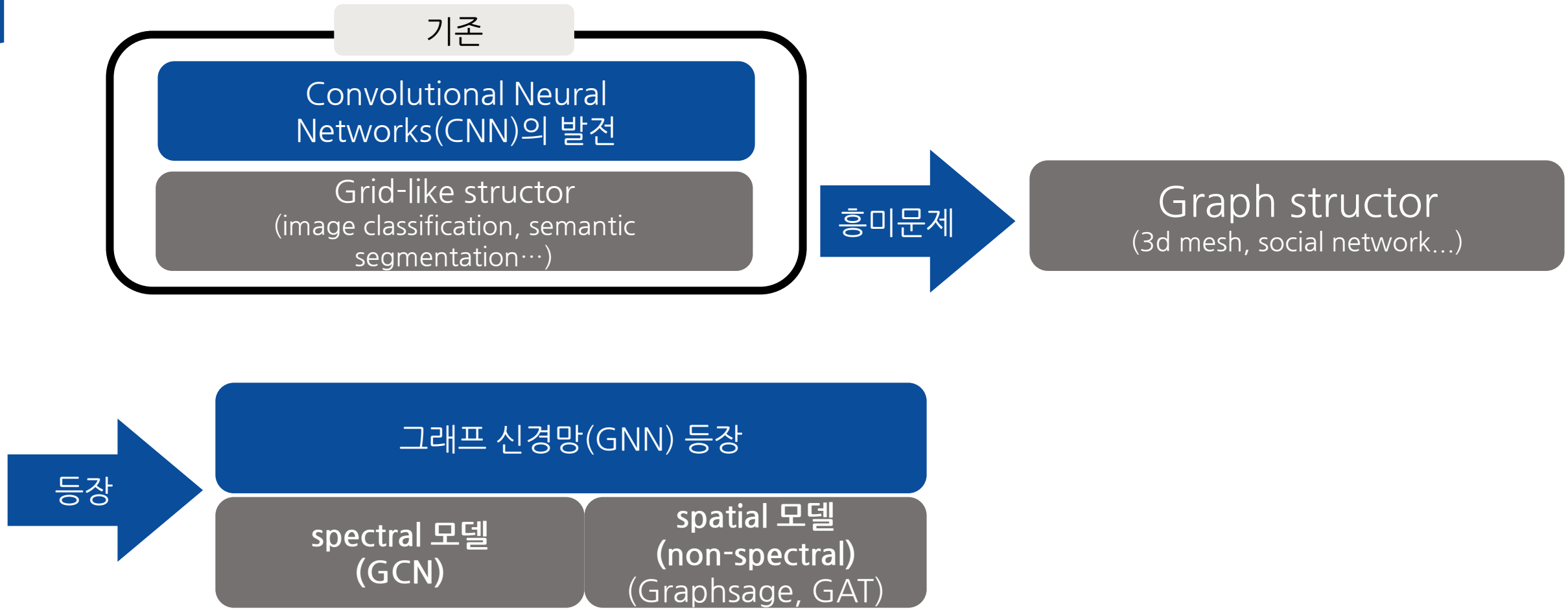
- inductive 문제 : Cora, Citeseer, Pubmed 데이터셋
- transductive 문제 : protein-protein interaction (PPI) dataset

> 그래프 벤치마크에서 SOTA 결과를 달성/일치.

2. Introduction

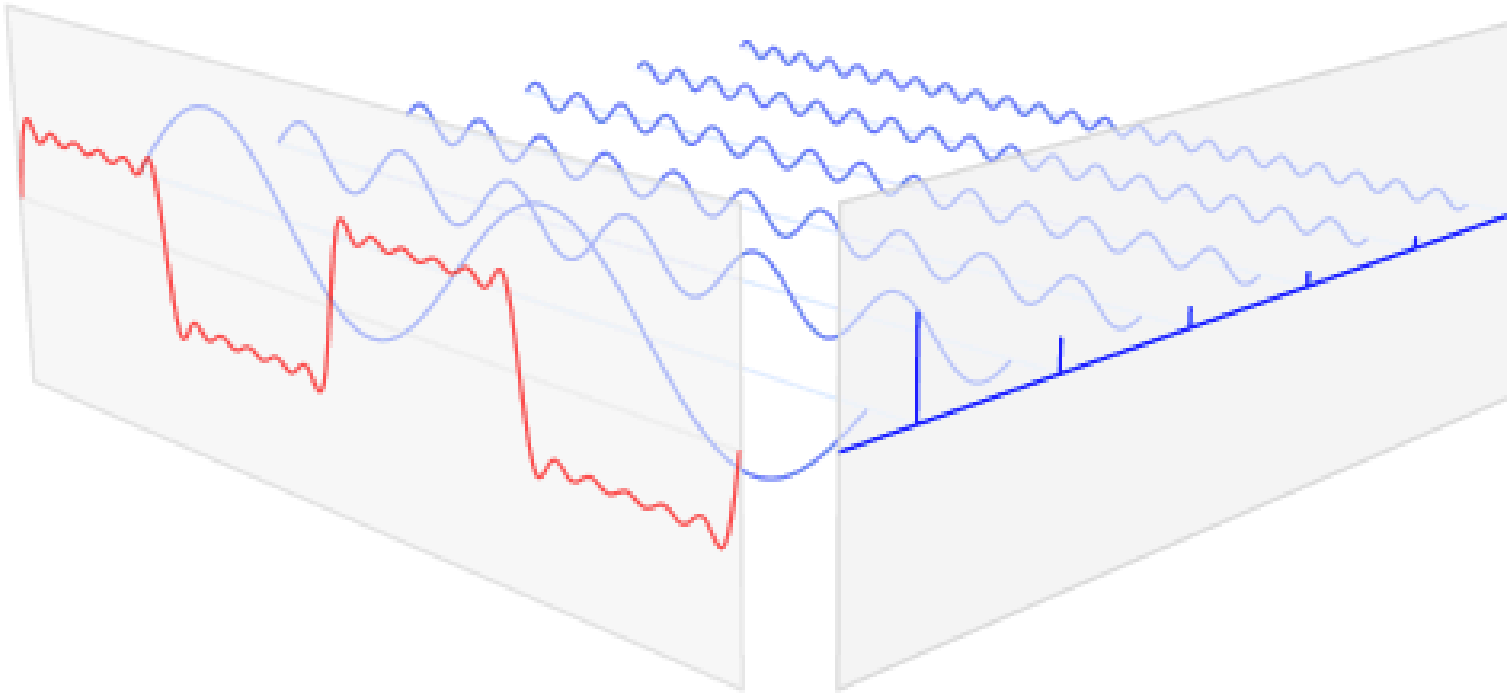
2. Introduction

1) 개요



2) spectral

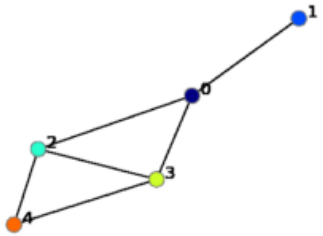
- 정의 : 신호/오디오/이미지/그래프를 time/spatial domain 이 아니라 frequcency domain으로 변환해 분석. 특정 신호를 단순한 요소(wavelets, graphlets)의 (조)합으로 **분해** 하는 것 > 푸리에 변환

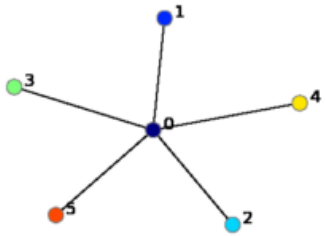


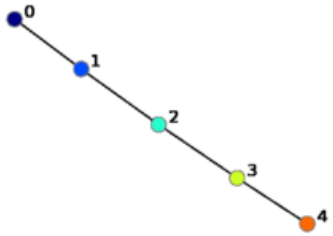
➢ **그래프 Laplacian** *L* 의 *고유 분해eigen-decomposition* 하는 것.
- 그래프 라플라시안 L을 특별한 방법으로 정규화된 인접 행렬 A로 생각할 수 있으며 고유 분해는 그래프를 구성하는 elementary orthogonal components을 찾는 방법.

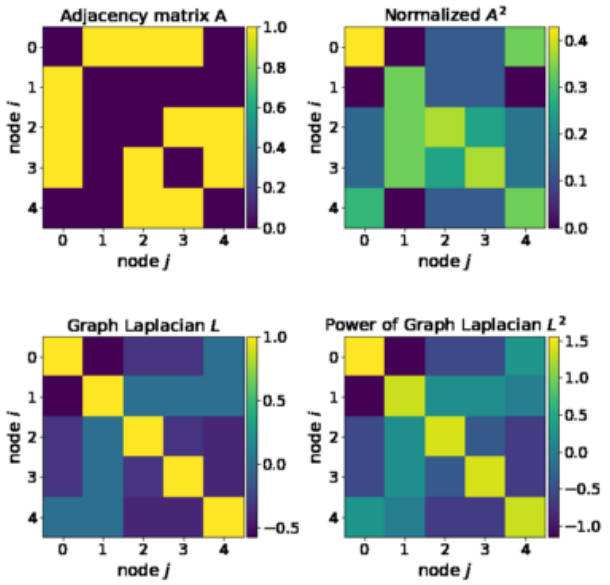
2) spectral

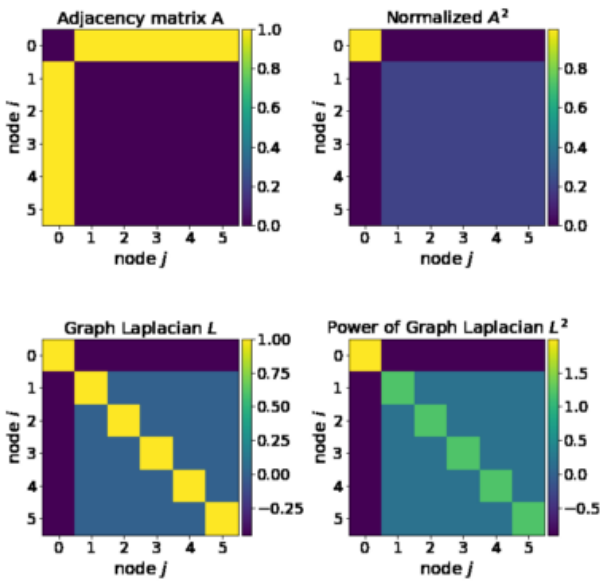
+ 그래프 Laplacian은 노드 *i* 에 힘을 가하면 그래프에서 "에너지"가 어떤 방향으로 얼마나 부드럽게 확산 되는지 보여줌
- 인접 행렬 A 는 A^n 가 노드 간의 n 홉 연결을 의미

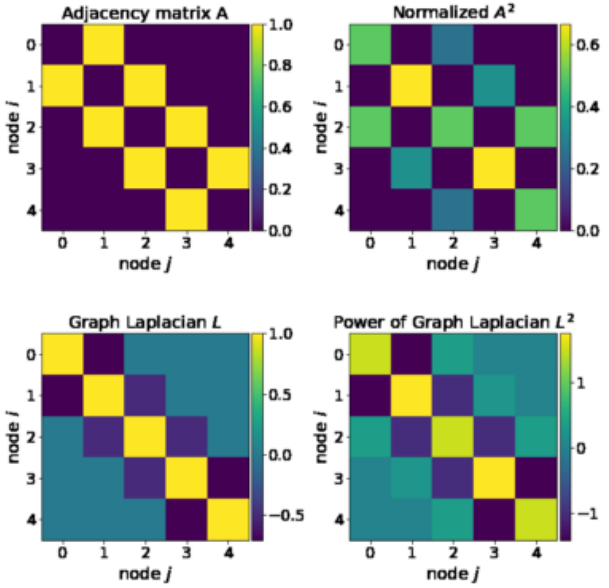












2) spectral

가정

1. Symmetric normalized Laplacian 가정
2. A 는 $A = A^{\mathrm{T}}$ 대칭행렬.
3. 그래프는 방향이 없음.

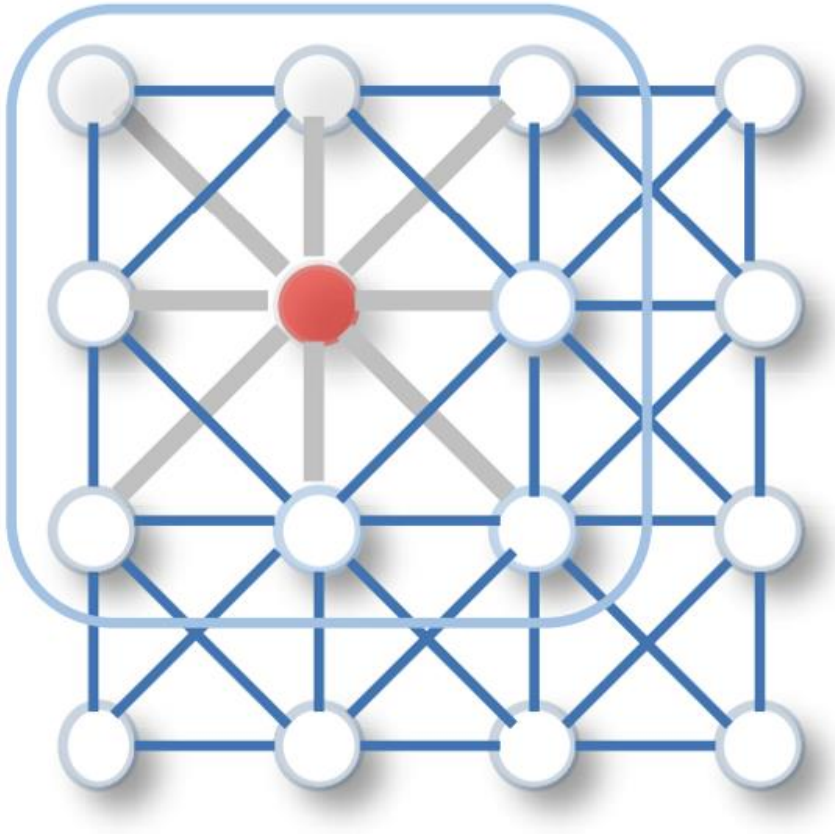
➢ 2,3 조건이 만족되지 않으면 노드 차수가 정의되지 않으며 Laplacian을 계산하기 위해 몇 가지 가정 필요

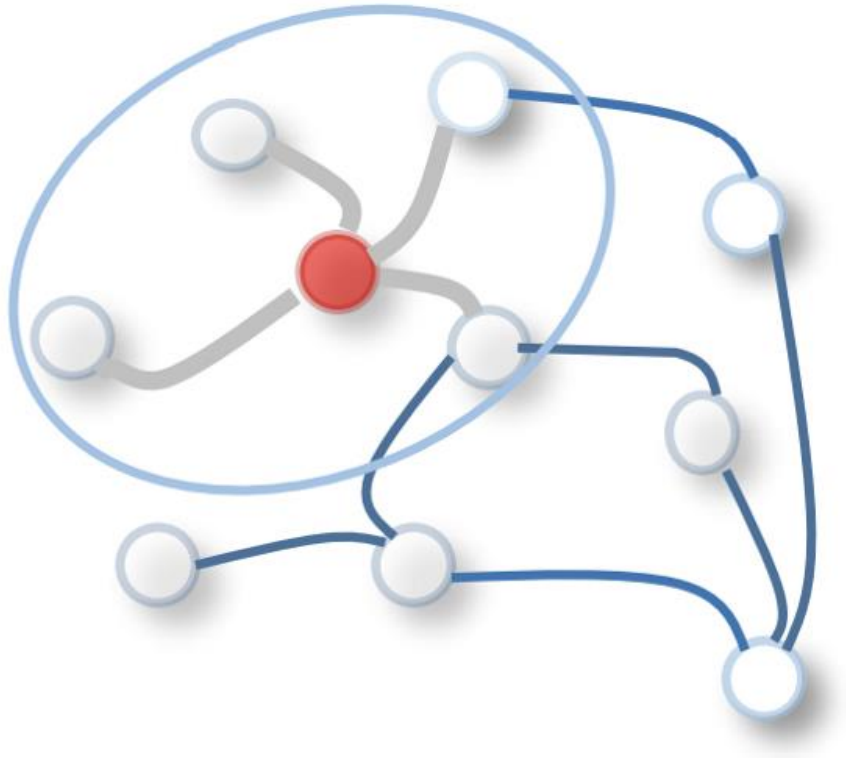
eigen-decomposition of the graph Laplacian.

- 고유 분해에 의해 찾을 수 있는 그래프 Laplacian L 의 고유 벡터 V 인 보다 일반적인 basis를 사용 > $L = V\Lambda V^{\mathrm{T}}$, 여기서 Λ은 L의/ 고유값
- spectral graph convolution : 가장 작은 고유 값에 해당하는 몇 개의 고유 벡터를 사용

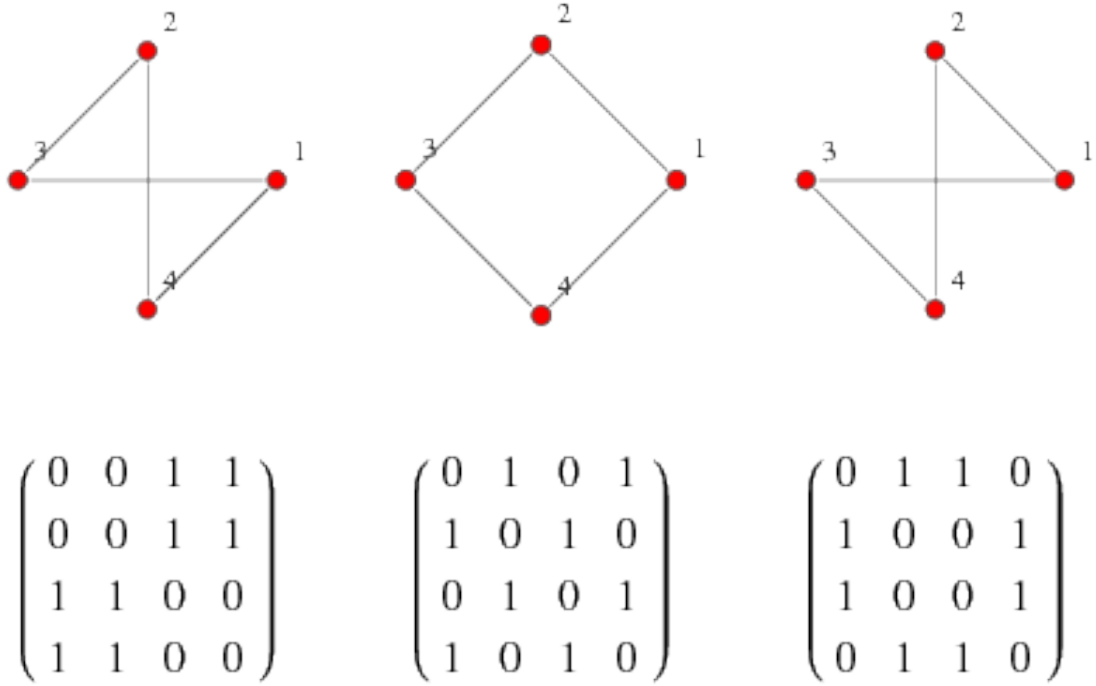
3) spatial

- Spatial 컨볼루션은 노드에 대해 공간적으로 컨볼루션하는 방식에서 일반 컨볼루션과 유사
- 각 노드와 가깝게 연결된 **고정된 이웃 노드에서만** convolution 연산을 수행해 노드를 업데이트.





4) Notation



$$\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Table 1. Frequently used notations.

V/E	The set of graph nodes/edges
$\mathcal{U}/\mathcal{I}$	The set of users/items
$\mathcal{N}_i$	The neighborhood set of graph node i
$\mathbf{h}_i^l$	The embedding of graph node i in the l-th propagation layer
$\mathbf{A}$	The adjacency matrix of graph
$\mathbf{W}^l$	Learnable transformation matrix in the l-th propagation layer
$\delta(\cdot)$	Nonlinear activation function
$\|$	Concatenation operation
$\odot$	Hadamard product

출처 : Graph Laplacian - Graph ML review (harryjo97.github.io)

5) Inductive vs Transductive

- Inductive : 전체가 아니라 자기 주변의 이웃만 보면 됨 > 단점 : 새로운 하위 그래프가 주어졌을 때 노드 임베딩시 “aligning” 필요
- Transductive : 전체를 다 봐야됨

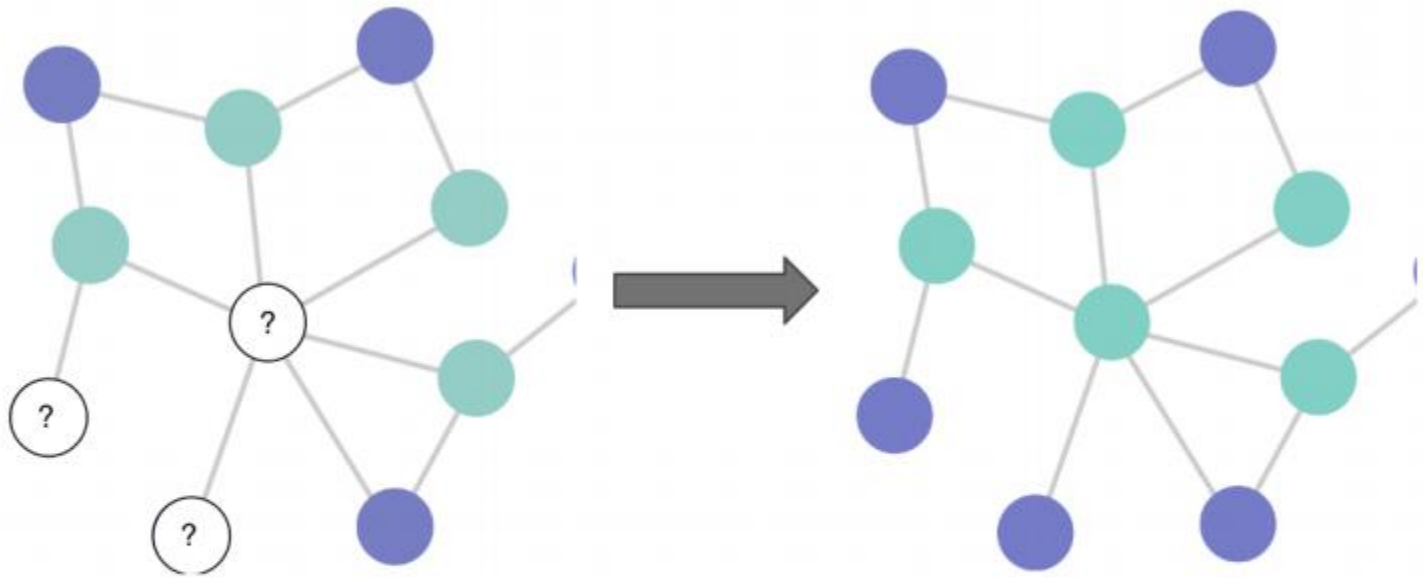


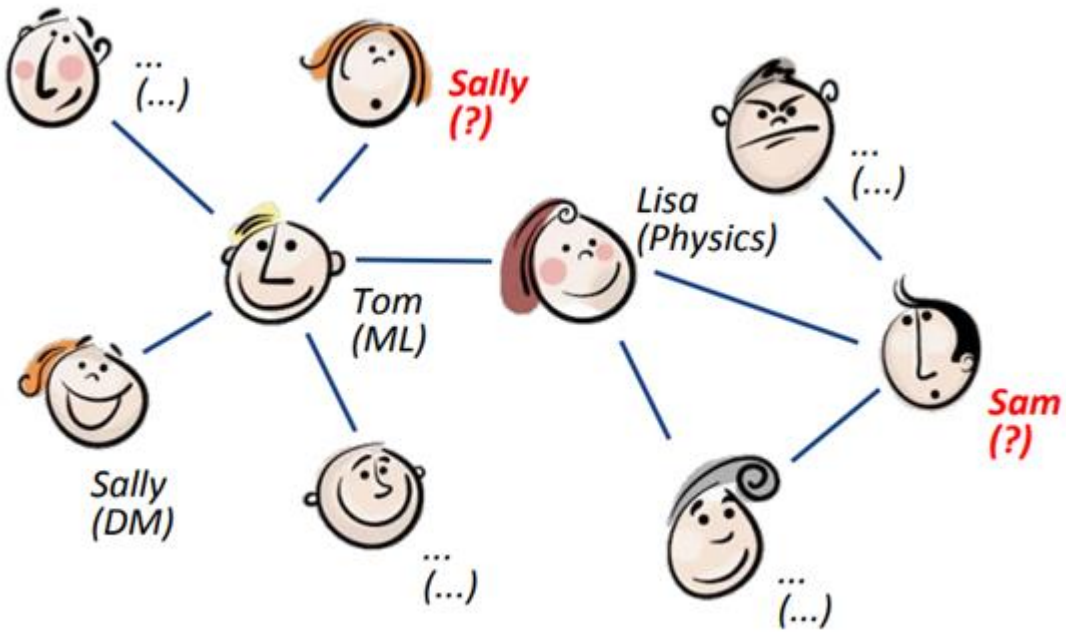
Figure 1. Node classification in transductive setting. At training time, the learning algorithm has access to all the nodes and edges including nodes for which labels are to be predicted.

출처 : inductive learning vs. transductive learning | Kyu's Blog (wordpress.com)

3. GCN

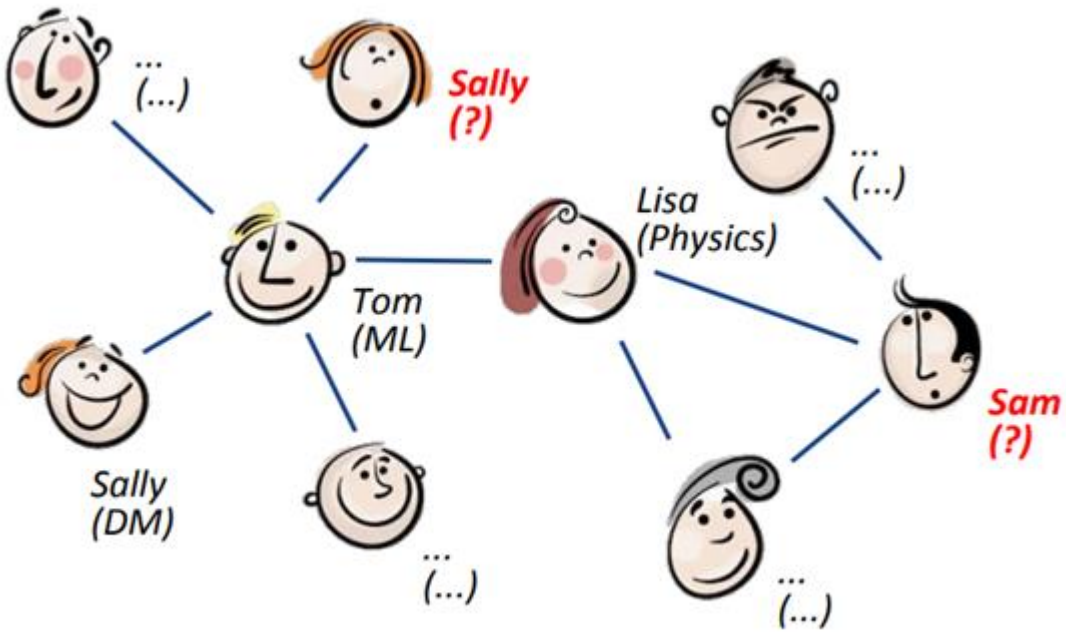
1) Introduction

- $\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{reg}, with \mathcal{L}_{reg} = \sum_{i,j} A_{ij} \| f(X_i) - f(X_j) \|^2 = f(X)^T \Delta f(X)$
  - $\mathcal{L}_0$: 라벨이 있는 부분에 사용하는 supervised loss
  - $f(.)$: 신경망에서 미분가능한 함수처럼 사용 가능한 함수.
  - λ : weight factor
  - X : 노드의 feature vectors matrix
  - Δ : D-A로 방향이 없는 그래프에서 라플라시안 비표준화를 나타냄.
  - N : 노드 $v_i \in \mathcal{V}$
  - 엣지 : $(v_i, v_j) \in \mathcal{E}$
  - adjacency matrix : $A \in \mathbb{R}^{N \times N}$
  - degree matrix $D_{ii} = \sum_j A_{ij}$

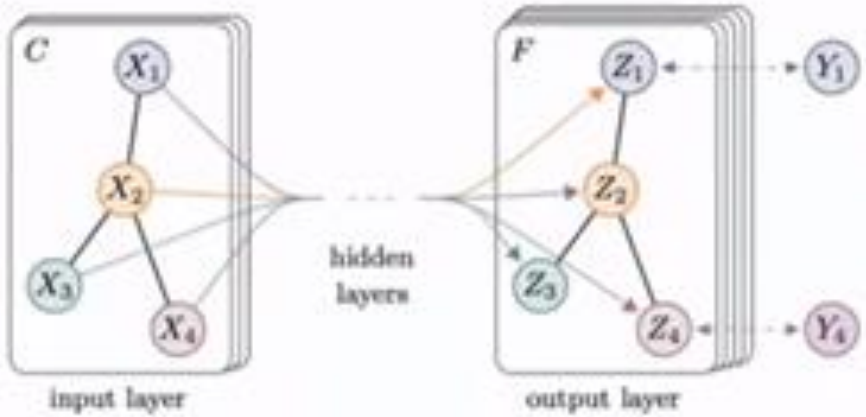


2) Graph Convolutions

- $H^{l+1} = \delta\left(\tilde{D}^{\frac{1}{2}}\tilde{A}\tilde{D}^{\frac{1}{2}}H^{l}W^{l}\right)$
  - $H^{l} \in R^{|V|\times D}$는 l 번째 convolution layer 의 graph nodes에 대한 embedding matrix.
  - D는 embedding dimension .
  - $\tilde{A} \in R^{|V|\times|V|}$ 는 self-loop하는 graph의 adjacency matrix. $\tilde{A} = \mathrm{A} + \mathrm{I_N}$
  - node i 가 j와 연결, i=j 일 때 각 entry $\tilde{A}_{ij} = 1$. 반대는 $\tilde{A}_{ii} = 0$, $\widetilde{D_{ii}} = \sum_j \tilde{A}_{ij}$
  - W^{l} : layer-specific trainable weight matrix
  - $\sigma(\cdot)$: activation function
  - $ReLU(\cdot) = \max(0,\cdot)$



GCN Layers (Model)



(a) Graph Convolutional Network

(b) Hidden layer activations

$$\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$$

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

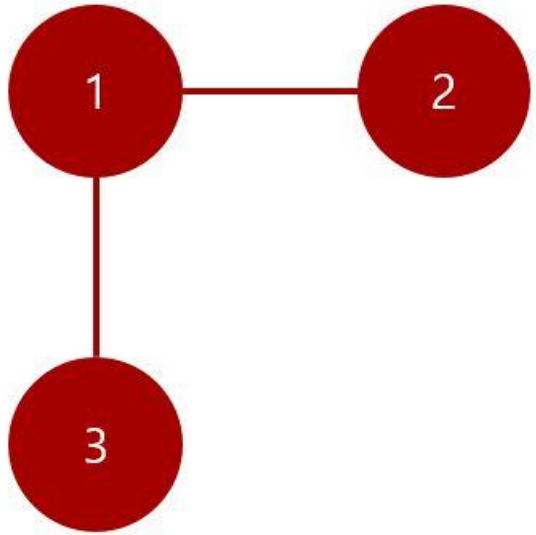
$$L = - \sum_{l \in Y_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

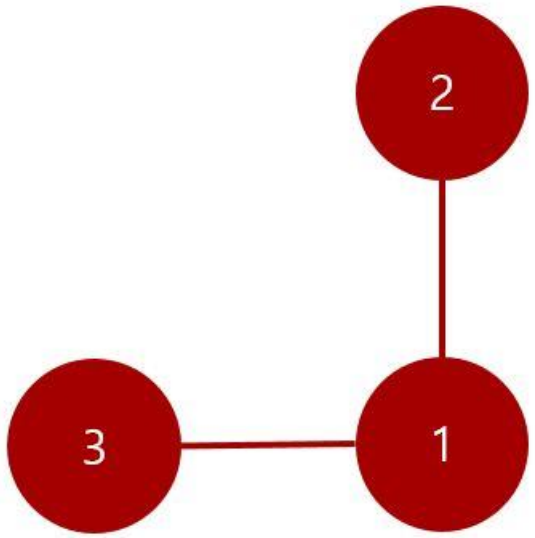
Y: 실제 라벨 정답, Z: 예측 값 에 대한 cross-entropy

4. GraphSAGE

1) introduction

4. GraphSAGE

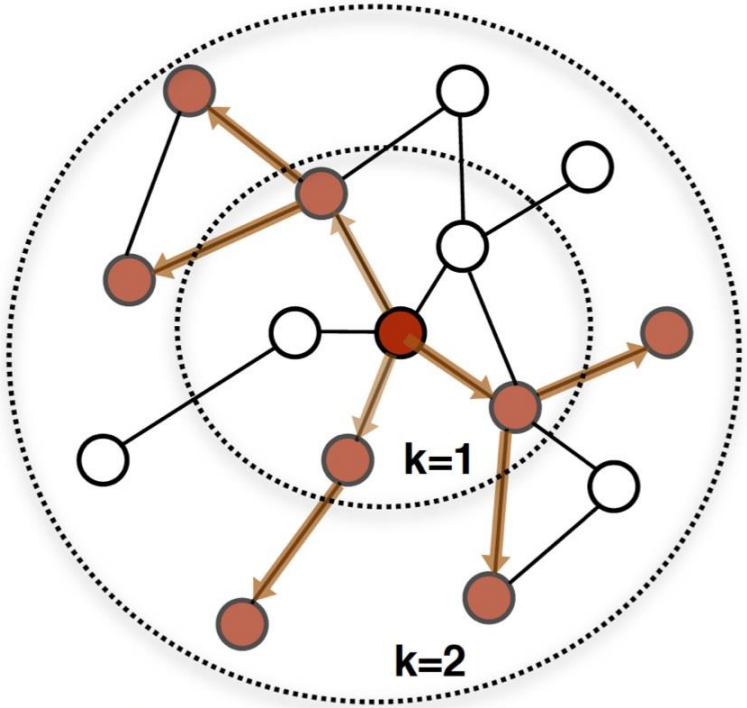




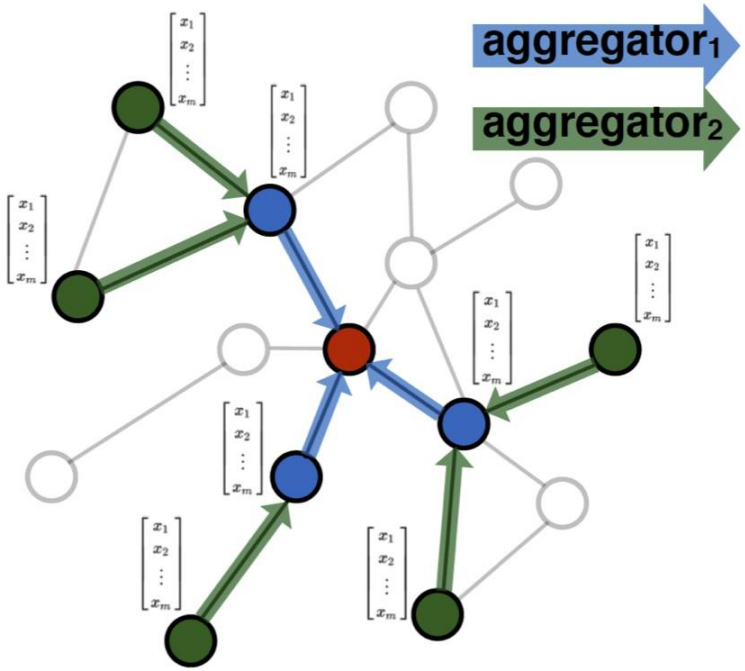
$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

2) GraphSAGE

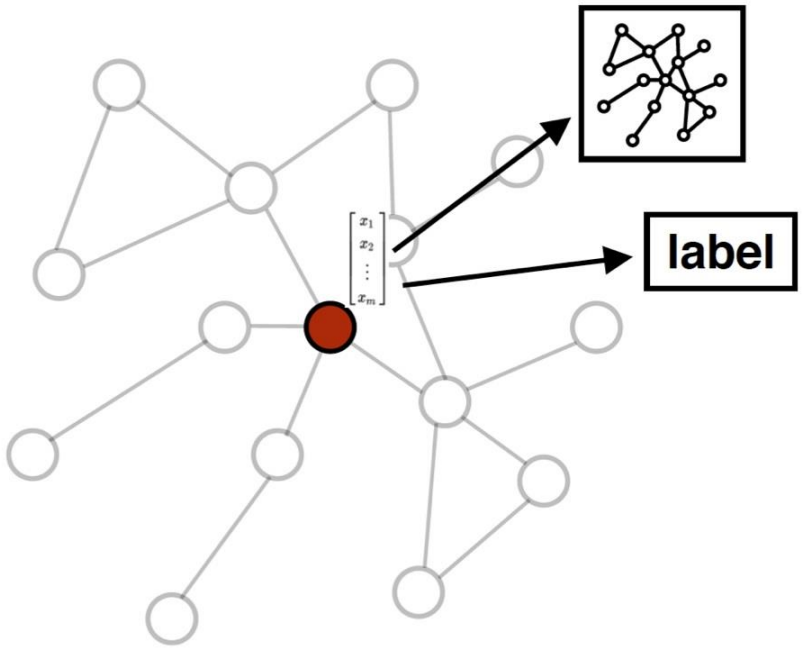
4. GraphSAGE



1. Sample neighborhood



2. Aggregate feature information from neighbors



3. Predict graph context and label using aggregated information

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output: Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

5. GAT

1) Graph Attentional Layer

- Bahdanau Attention 활용

Layer 입력

- 노드 feature 집합$h = \{\overrightarrow{h_1}, \overrightarrow{h_2}, \overrightarrow{h_3}, \ldots, \overrightarrow{h_N}\}, \overrightarrow{h_i} \in \mathbb{R}^F$ / N : 노드번호 / F : 각 노드의 features 수
- layer는 potentially 다른 cardinality F' 를 가지므로
- 새 노드 features 집합을 생성. $h' = \{\overrightarrow{h_1}', \overrightarrow{h_2}', \overrightarrow{h_3}', \ldots, \overrightarrow{h_N}'\}, \overrightarrow{h_i}' \in \mathbb{R}^{F'}$
- input features을 higher-level features로 변환하기 위해 최소 하나 이상의 선형 변환 필요.

- 가중치 행렬 $W \in \mathbb{R}^{F' \times F}$에 의해 매개 변수화된 **공유** 선형 변환이 모든 노드에 적용.
- 노드에서 self-attention 수행 $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$ attention coefficients를 계산
- $e_{ij} = a(W\overrightarrow{h_i}, W\overrightarrow{h_j})$ (1) 노드 i에 대한 노드 j의 features의 중요도
- Masked : 모든 노드가 다른 모든 노드에 영향을 줄 수 있는 구조이지만 마스킹된 attention를 수행하여 모든 노드가 아닌 일부의 노드만으로 제한. $\mathcal{N}_i$ 그래프에서 노드 i의 일부 neighborhood
- 논문에서는 I (i를 포함한)의 1차 neighborhood으로 제한.
- 서로 다른 노드 간에 쉽게 비교가능하도록 소프트맥스 함수를 사용하여 j에서 정규화.
- $\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$ (2)

1) Graph Attentional Layer

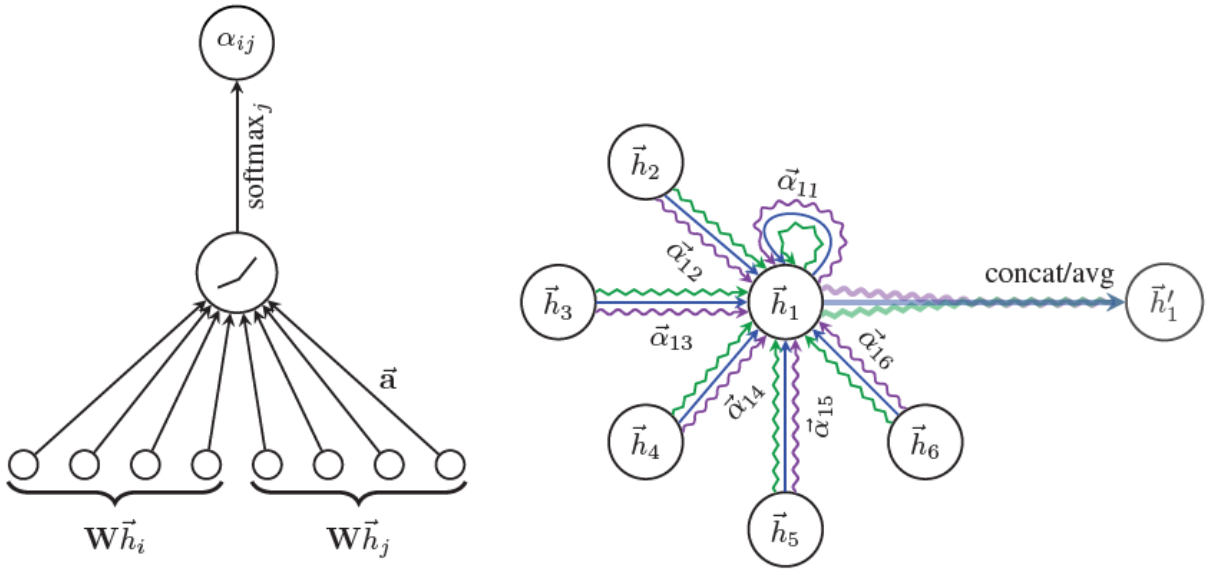


Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

- attention 메커니즘 α 는 single-layer feedforward neural network이며, weight vector $\vec{a} \in \mathbb{R}^{2F'}$에 의해 매개 변수화되고 LeakyReLu를 적용.
- 비선형성(음의 입력 기울기 α = 0.2) Fully expanded되면, attention 메커니즘에 의해 계산된 coefficients 는 $\alpha_{ij} = \frac{\exp(LeakyReLU(\vec{a}^T[W\overrightarrow{h_i}||W\overrightarrow{h_j}))}{\sum_{k\in\mathcal{N}_i}\exp(LeakyReLU(\vec{a}^T[W\overrightarrow{h_i}||W\overrightarrow{h_k}))}$ / ||는 concatenation 연산.

1) Graph Attentional Layer

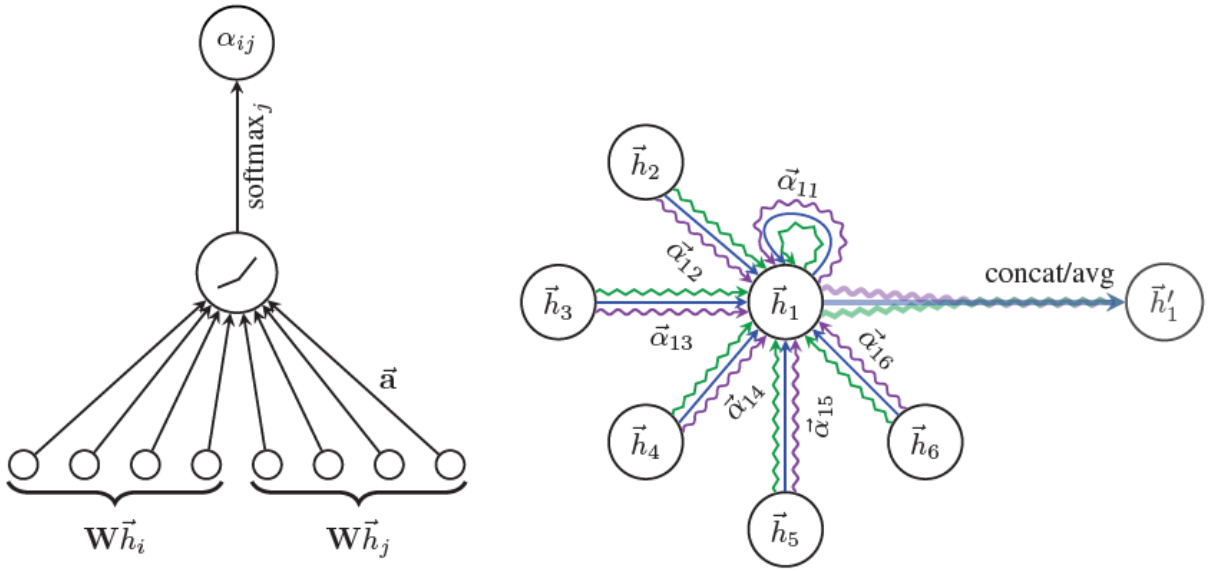


Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

- normalized attention coefficients는 (잠재적으로 nonlinearity σ (concat/avg)를 적용한 후) 모든 노드에 대한 출력 features의 linear combination을 계산하기 위해 사용됨.
- $\overrightarrow{h_i}' = \sigma(\frac{1}{K} \sum_{\{k=1\}}^{\{K\}} \sum_{\{j \in \mathcal{N}_i\}} \alpha_{\{ij\}}^{k} \, W^k \overrightarrow{h_j})$(6)
- multi-head graph attentional layer의 aggregation process는 그림 1(오른쪽)에 의해 설명됨.

2) Comparisons to related work

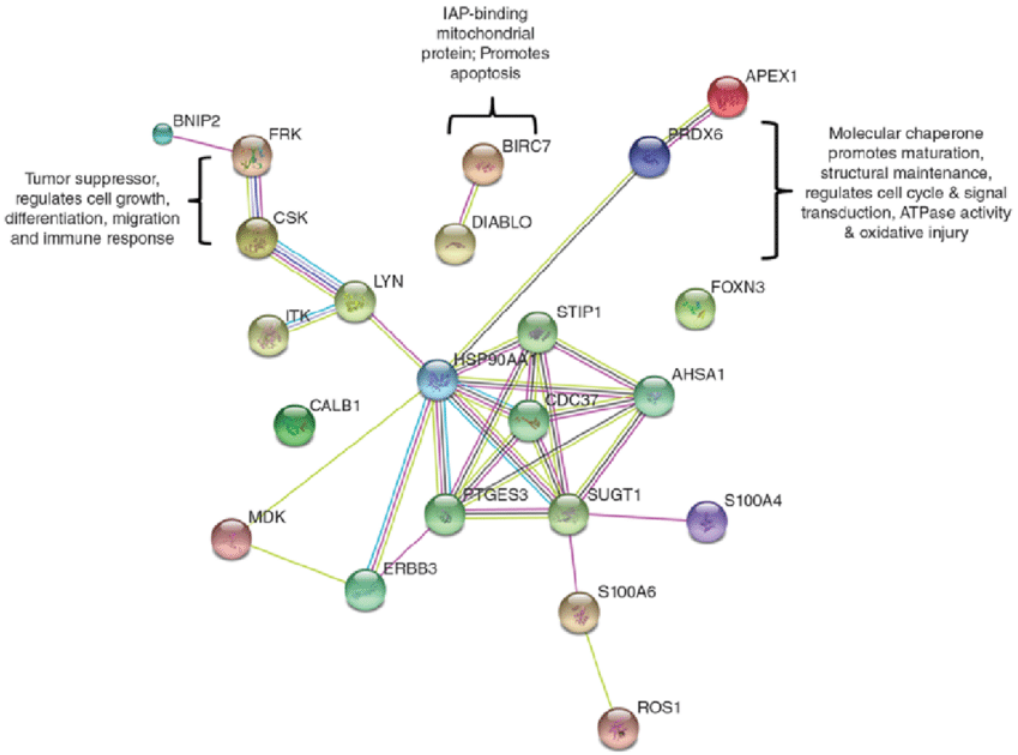
1. 계산효율적 : self-attention layer는 모든 edges 에서 병렬화 가능.
- eigen decompositions등과 같은 복잡도 높은 행렬 연산은 필요하지 않음
- F' features의 시간 복잡도는 $O(|V|FF' + |E|F')$ / GCN(spectral)의 시간 복잡도는 $O(|E|D + |V|D^2)$
  - F는 입력 features의 수이고 $|V|$와 $|E|$는 각각 그래프에서 node와 edges
- 시간복잡도는 같으나, multi-head attention를 적용하면 완전히 독립적으로 병렬화 가능.

2. 공간효율적
- 동일한 neighborhood의 다른 중요도를 할당하여 모델 용량 감소
- 그래프의 모든 edge에서 공유되므로 global graph structure에 대한 upfront 액세스 또는 all edges(많은 prior techniques의 limitation)에 의존하지 않음.
- Neighborhood 크기를 지정하여 샘플링
- Sparse matrix 에서 노드 수와 에지 수를 선형으로 줄이면 효율적으로 사용 가능

3) datasets

Table 1: Summary of the datasets used in our experiments.

	Cora	Citeseer	Pubmed	PPI
Task	Transductive	Transductive	Transductive	Inductive
# Nodes	2708 (1 graph)	3327 (1 graph)	19717 (1 graph)	56944 (24 graphs
# Edges	5429	4732	44338	818716
# Features/Node	1433	3703	500	50
# Classes	7	6	3	121 (multilabel)
# Training Nodes	140	120	60	44906 (20 graphs
# Validation Nodes	500	500	500	6514 (2 graphs)
# Test Nodes	1000	1000	1000	5524 (2 graphs)

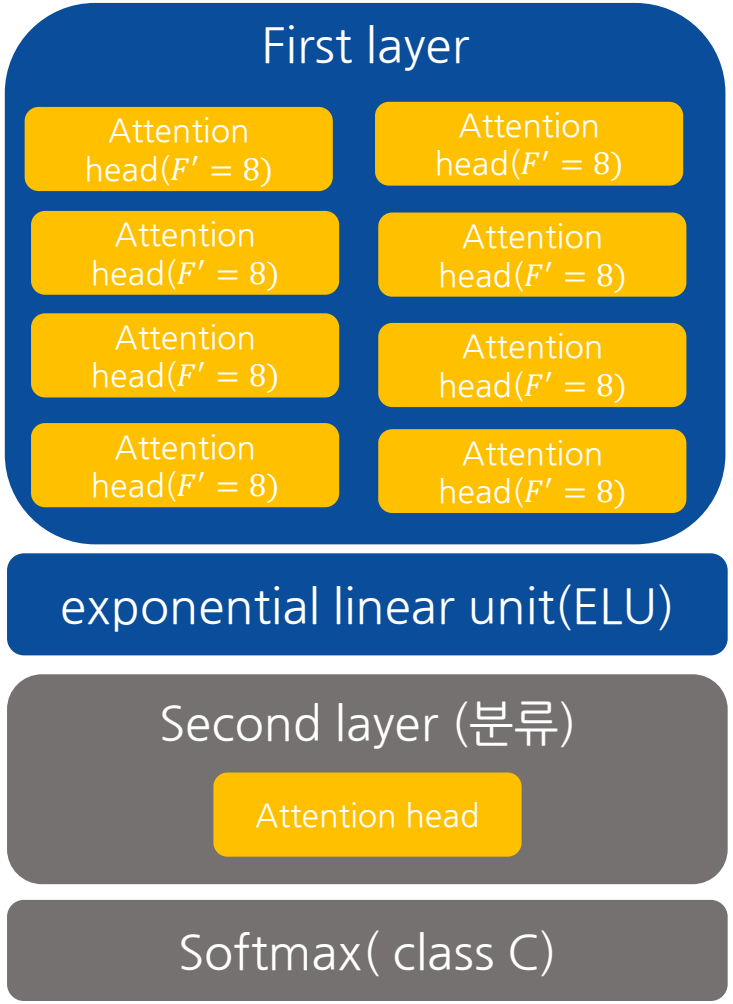


- Transductive learning : node는 documents에 해당하고 edge는 (undirected) citations에 해당. node features는 documents의 bag-of-words representation
- Inductive learning : 다양한 인체 tissues에 해당하는 그래프로 구성된 protein-protein interaction

4) EXPERIMENTAL SETUP

Transductive learning

- 2-layer GAT 모델
- p = 0.6인 dropout



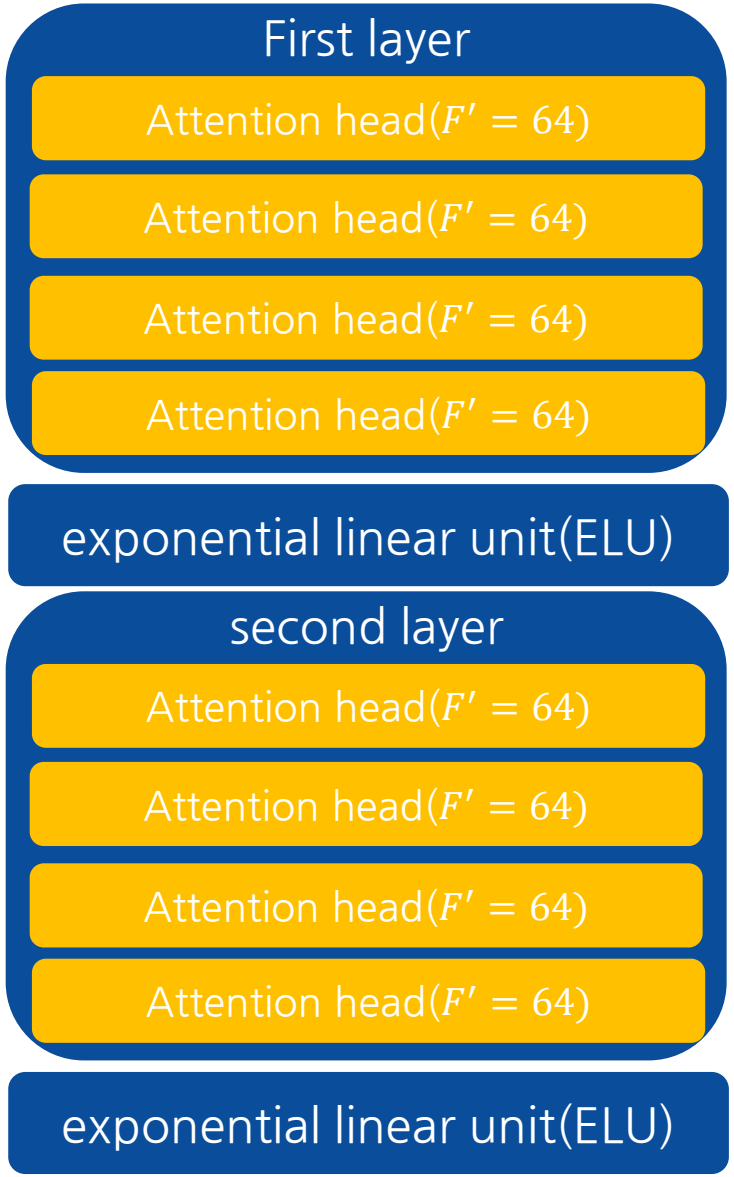
3) EXPERIMENTAL SETUP

inductive learning

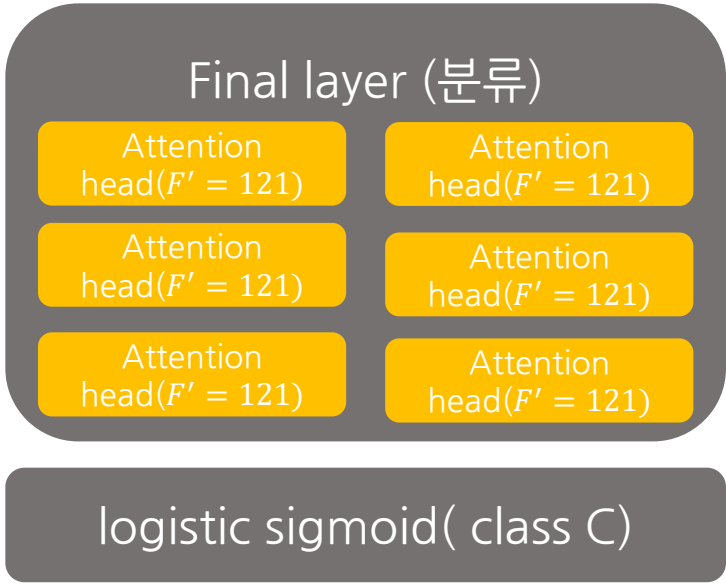
- L2 정규화 또는 드롭아웃을 적용할 필요가 없음. (training dataset 큼)
- 중간 attention layer에 걸쳐 skip connections
- 모든 neighborhood에 동일한 가중치를 할당

공통

- Glorot initialization
- Adam SGD optimizer
- Pubmed에 대해 0.01이고 다른 모든 dataset에 대해서는 0.005.
- 100 Epoch의 validation nodes의 cross-entropy loss 또는 accuracy (transductive), micro-F1 (inductive) score 에 대해 early stopping strategy



5. GAT



5) Result

Transductive

Method	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	81.7 ± 0.5%	—	78.8 ± 0.3%
GCN-64*	81.4 ± 0.5%	70.9 ± 0.5%	**79.0** ± 0.3%
GAT (ours)	**83.0** ± 0.7%	**72.5** ± 0.7%	**79.0** ± 0.3%

Inductive

Method	PPI
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	**0.973** ± 0.002

5) Result

- Explainable
  - Cora dataset의 GAT의 첫 번째 layer에 의해 추출된 t-SNE 변환 feature representation 시각화

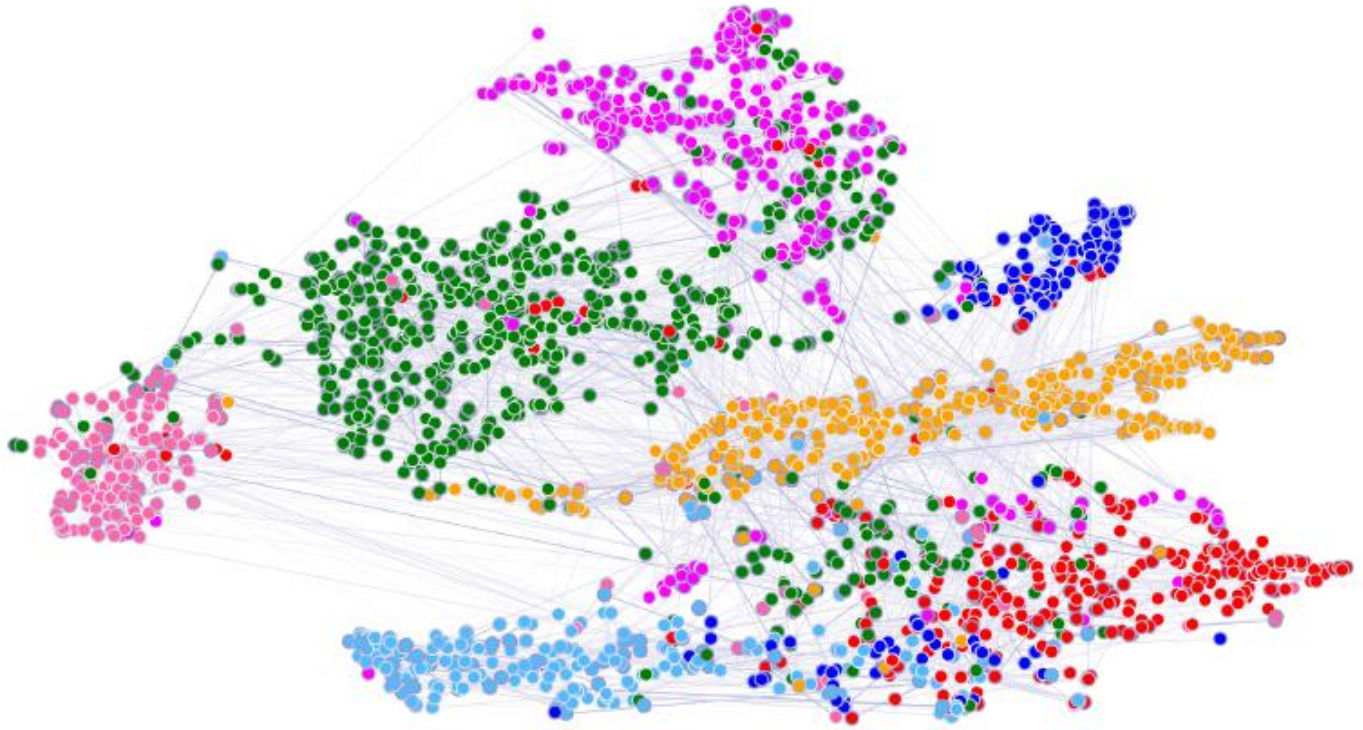


Figure 2: A t-SNE plot of the computed feature representations of a pre-trained GAT model's first hidden layer on the Cora dataset. Node colors denote classes. Edge thickness indicates aggregated normalized attention coefficients between nodes i and j, across all eight attention heads $(\sum_{k=1}^{K} \alpha_{ij}^{k} + \alpha_{ji}^{k})$.

6) conclusion

- masked self-attention layer를 활용하여 그래프 데이터에서 작동하는graph attention network(GAT)를 제시.
- 그래프 attention layer은 계산적으로 효율적, 서로 다른 크기의 neighborhood을 다루면서 neighborhood 내의 서로 다른 노드에 서로 다른 중요성을 할당할 수 있음(implicitly).
- 이전의 spectral 기반 접근방식으로 많은 이론적 문제를 해결함.
- 모델은 4개의 노드 분류 벤치마크에서 성공적으로 SOTA 성능을 달성하거나 일치시킴.
- 특히 흥미로운 연구 방향은 모델 해석 가능성에 대한 철저한 분석을 수행하기 위해 attention 메커니즘을 활용하는 것.
- 모델을 확장하여 edge features(노드 간 관계를 나타낼 수 있음)을 통합하면 더 다양한 문제 해결 가능.

References

https://towardsdatascience.com/tutorial-on-graph-neural-networks-for-computer-vision-and-beyond-part-2-be6d71d70f49

https://towardsdatascience.com/an-introduction-to-graph-neural-networks-e23dc7bdfba5

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

http://outobox.cs.umn.edu/PCA_on_a_Graph.pdf

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

https://arxiv.org/abs/1611.08097

Q & A

감사합니다