Graph Convolution Network (GCN)

https://youtu.be/Gb5aVZOYmv0

Contents

CryptoCraft LAB

Graph

❖**edge로 연결된 node의 집합**

- node == vertice → 꼭지점
- edge → 모서리

❖edge간의 **방향**이 있는 경우, 없는 경우 모두 표현 가능

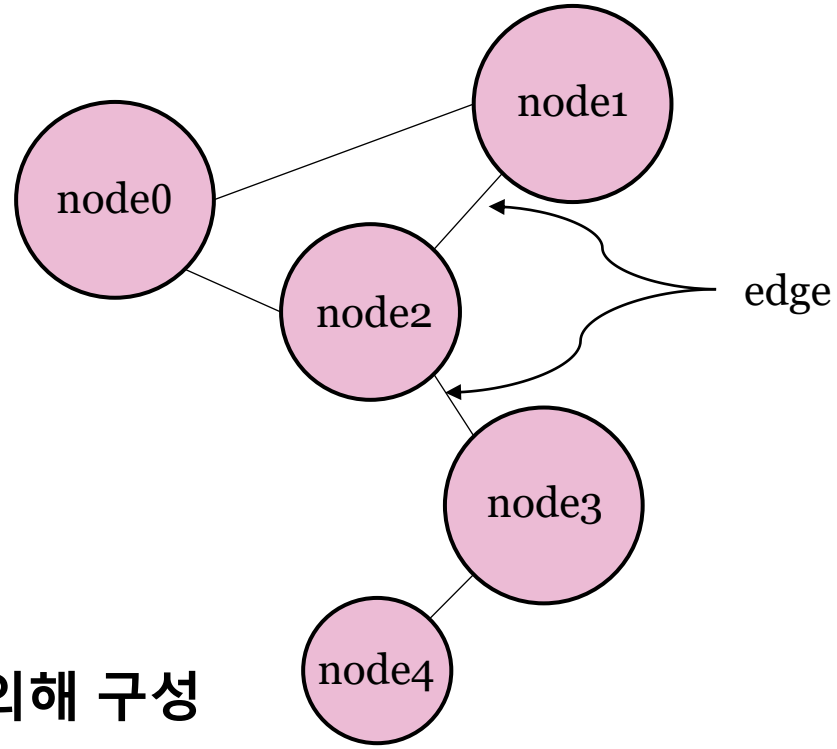
❖edge의 **중요도**(값, vertice간의 거리) 또한 표현 가능

- weight, distance → weighted graph

❖그래프는 유연한 데이터 구조를 가짐

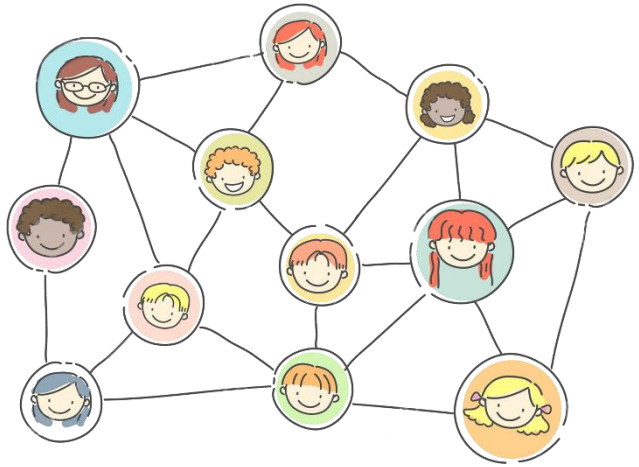
❖node, edge는 **풀고자 하는 문제에 대한 지식이나 직관 등에 의해 구성**

❖컴퓨터에서는 보통 **matrix로 표현**

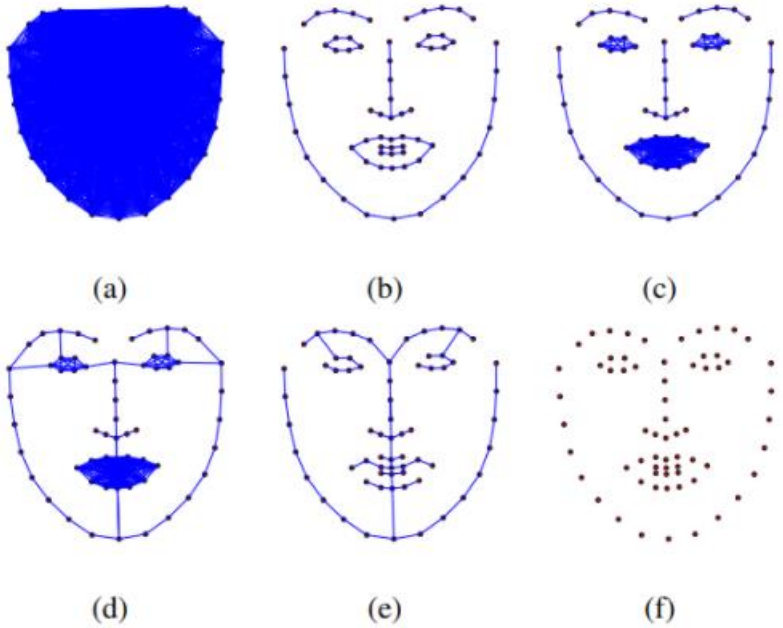


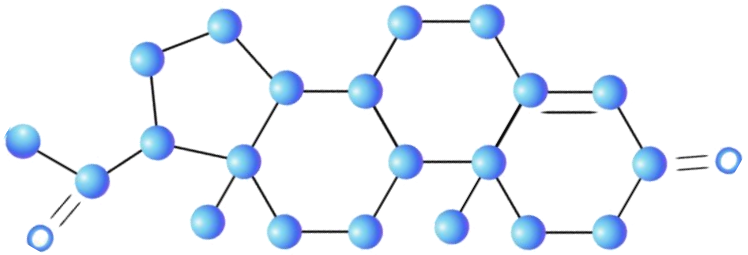
Graph

Social Graph

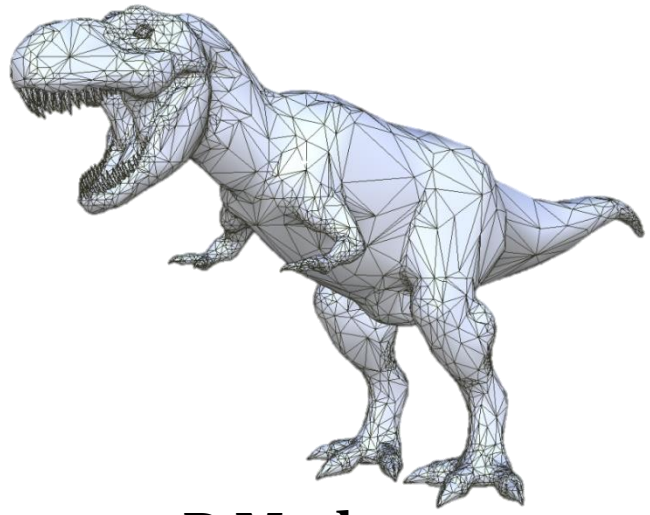


- **node** : 사람들의 정보
  (이름, 나이 등의 특징 가짐)
- **edge** : 친밀도, 관계 등





Molecular Structure



3D Mesh

- **node** : 원자
  (종류 등의 특성)
- **edge** : 원자 간 결합
  (공유 결합, 이온 결합 등)

Graph Neural Network

❖**Neural Network (NN)**

- 데이터의 특징, 의미 등을 추출하는 역할

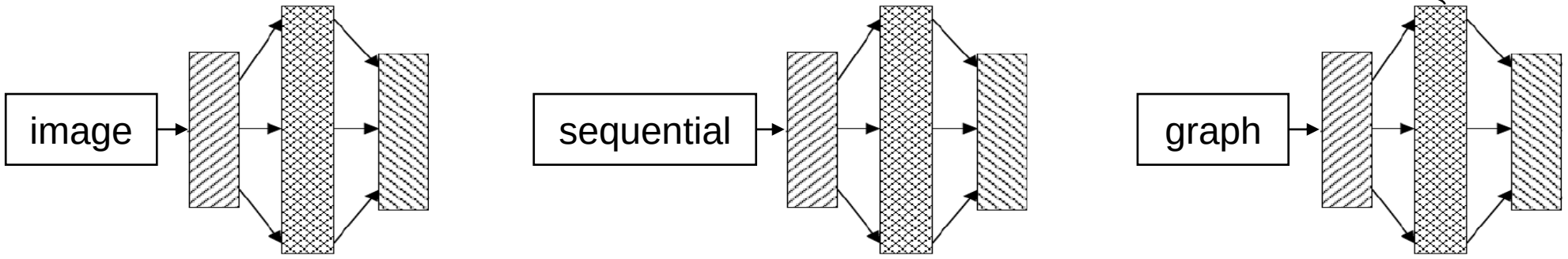
❖**다뤄야 할 데이터에 맞는 NN 구조 필요**

- image data : CNN
- sequential data : RNN, LSTM
- **graph : GNN (Graph Neural Network)**

***graph 구조에 대해 학습하는 신경망**

node 간의 관계성, node의 정보를 학습

→ graph의 feature를 추출하여 학습



Convolutional NN　　Recurrent NN　　Graph NN

Convolutional Neural Network

❖ **convolution layer : feature 추출**

❖ **maxpooling layer : feature 강화**

❖ **지역적 특징** 학습

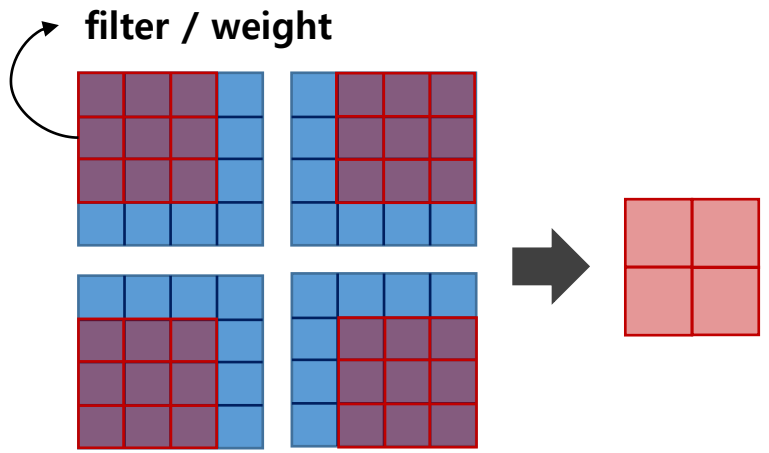
- **filter** 영역씩 연산을 수행 → translation invariance

❖ **weight sharing**

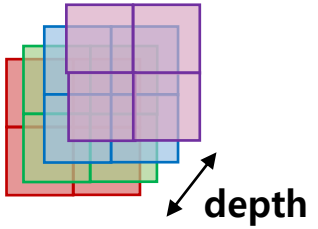
- **동일한 가중치를 공유** == **동일한 필터**가 입력 이미지를 순회
  → 동일한 가중치를 적용하여 convolution 연산 수행
  → 학습할 파라미터가 적어 효율적이며 과적합 감소

❖ **output tensor의 depth → filter 수**

- 각 filter 마다 convolution 연산 수행



$input\ data\ *\ filter$



연산 후의 상태 (activation map)

필터가 여러 개면 depth가 늘어남

❖ **이러한 CNN의 장점을 GNN에도 적용 → Graph Convolution Network**

Graph Convolution Network (GCN)

❖각 레이어를 통과하며 그래프의 상태나 정보를 학습

- graph의 상태, 정보를 결정하는 요소

  node에 담긴 정보 (특징, 값)

- 즉, 각 레이어를 거칠 때마다 그래프의 정보 (노드의 값)이 갱신되며 학습

  → hidden state update

❖idea

1. weight sharing

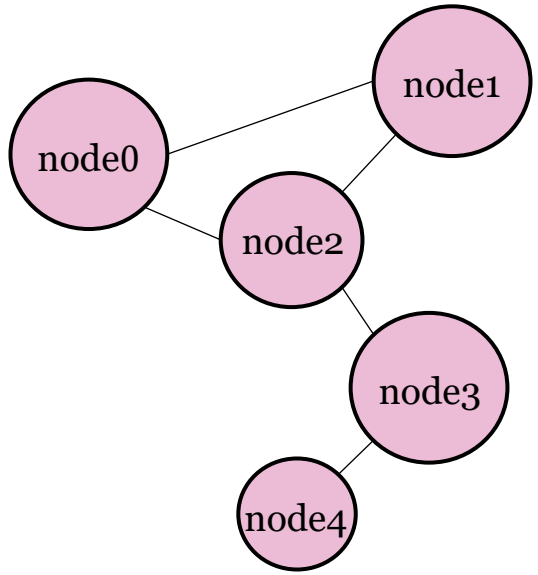
   동일한 가중치를 적용하여 학습

2. local value (학습할 노드의 근처 노드들의 정보)

   1개의 뉴런이 local한 정보들만을 받아 학습

Matrix in GCN

GCN에서의 graph 표현



	f		
node 0	255	0	17
node 1	8	144	76
node 2	127	32	241
node 3	56	11	23
node 4	216	98	102

n

feature

Node Feature Matrix

	n				
node 0	1	1	1	0	0
node 1	1	1	1	0	0
node 2	1	1	1	1	0
node 3	0	0	1	1	1
node 4	0	0	0	1	1

n

node 0 ~ node4

Adjacency Matrix

node에 담긴 정보(feature)를 나타내는 행렬

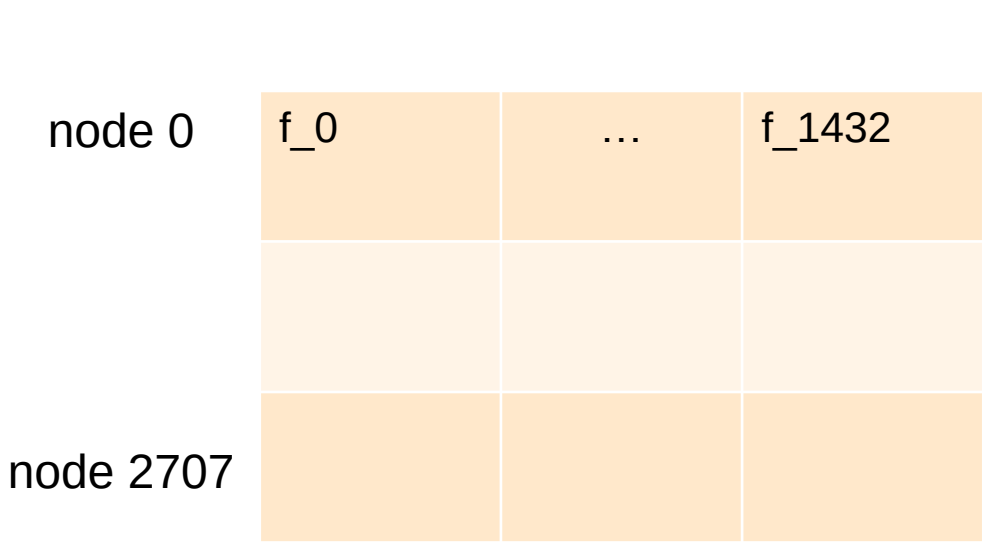
- node의 수 (n) , feature의 수 (f) → $n \times f$ 행렬
- X_{ij} → i번째 node의 j번째 feature

node간의 연결을 나타내는 행렬

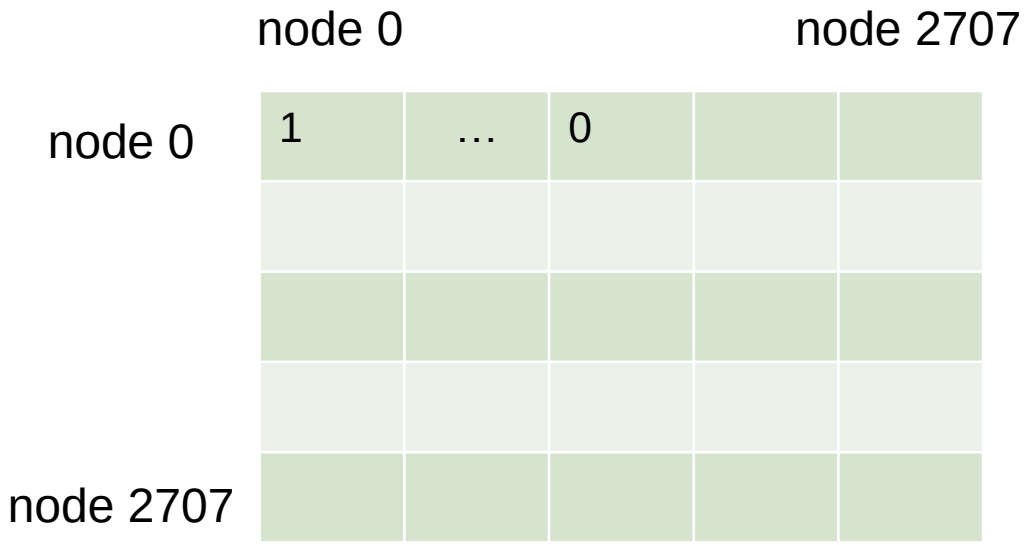
- node의 수 (n) → $n \times n$ 행렬
- A_{ij} → i번째 node와 j번째 node의 연결 관계

graph :: input data

```
X dim=(2708, 1433)
A dim=(2708, 2708)
y dim=(2708, 7)
```

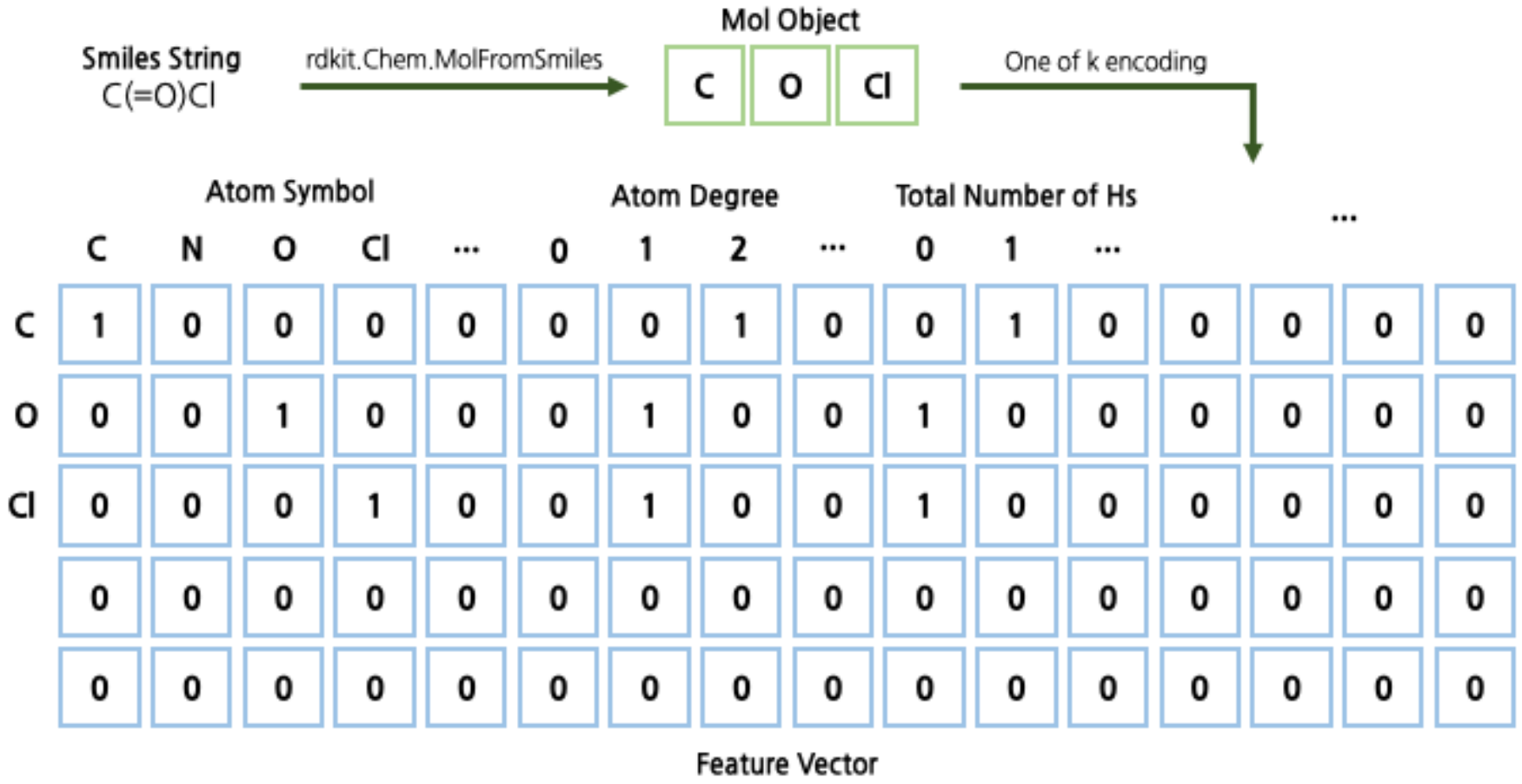


node feature matrix



adjacency matrix

graph :: input data



hidden state update

❖ hidden state

- 각 layer를 거친 후의 node feature matrix
- $H_1{}^l$: l번째 layer를 거친 후의 1번 노드의 node feature matrix

❖ 각 노드의 정보를 업데이트 == node feature matrix를 업데이트

❖ 하나의 노드와 연결된 노드만 반영하여 업데이트 (local)

- 해당 노드와 관련있는 정보만을 지역적으로 학습

$$H_1^{l+1} = \sigma\left(H_0{}^l W^l + H_1{}^l W^l + H_2{}^l W^l + b^l\right)$$

hidden state update

Node Feature Matrix

node 0	255	0	17
node 1	8	144	76
node 2	127	32	241
node 3	56	11	23
node 4	216	98	102

n

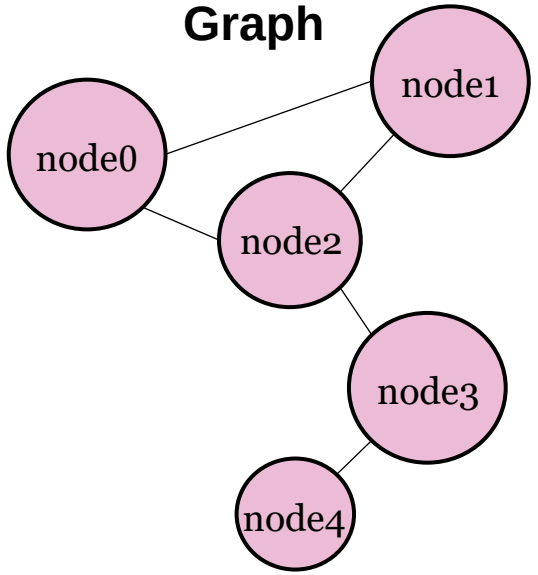
f (feature)

H_0 (node0의 feature matrix)

H_1 (node1의 feature matrix)

H_4 (node4의 feature matrix)

Graph



activation function

$$H_1^{l+1} = \sigma(H_0{}^l W^l + H_1{}^l W^l + H_2{}^l W^l + b^l)$$

node1이 l번째 layer에서
하나의 layer를 통과한 후의
node feature matrix

node1과 연결된 node들의
hidden state

동일한 가중치
(동일 필터)

hidden state update

- 각 layer의 hidden state 위해 연결 관계를 각 레이어마다 파악해야하는가?

255	0	17	➡ H_0
8	144	76	
127	32	241	
56	11	23	
216	98	102	➡ H_4

node feature matrix

5개의 노드에 대해 각 3개씩의 feature

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

adjacency matrix

연결관계를 담고있는 행렬

node1,2,3만 관련있으므로 H_0, H_4는 사용 x

$$H_1^{l+1} = \sigma\left(H_1{}^l W^l + H_2{}^l W^l + H_3{}^l W^l + b^l\right)$$

$$H^l W^l + b^l$$

$$\mathrm{H}^{l+1} = \sigma(AH^l W^l + b^l)$$

H의 전체 행을 모두 사용 시, 관련없는 노드의 정보도 학습

→ **관련 노드의 정보만 반영하기 위해 A행렬을 곱함**

(연결x인 node의 정보는 0이 곱해져서 반영x)

hidden state update

자신의 정보는 반영해야하므로
연결되어 있지 않아도
$i == j$인 경우 1

CNN의 depth

filter 수

251	25	177
10	72	1
82	181	164
17	6	26
21	9	112

=

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

•

255	0	17
8	144	76
127	32	241
56	11	23
216	98	102

•

W_0	W_1	⋯	W_{64}
W_0	W_1	⋯	W_{64}
W_0	W_1	⋯	W_{64}

각기 다른 필터를 곱해 나온 고차원적인 정보

adjacency matrix

node feature matrix

weight (filter)

동일한 filter는 동일한 weight 적용

- 노드 수 : 레이어마다 동일
- node feature matrix의 열 : 이전 레이어의 filter 수에 따라 결정
- weight 행렬의 행 : node feature의 feature 수
- weight 행렬의 열 : 해당 레이어의 filter 수

layers

```
self.conv1 = GCNConv(dataset.num_node_features, 16)
self.conv2 = GCNConv(16, dataset.num_classes)
```

```
X, edge_index = data.x, data.edge_index
H = self.conv1(X, edge_index)
H = F.relu(H)
Y = self.conv2(H, edge_index)
Y = F.log_softmax(Y, dim=1)
```

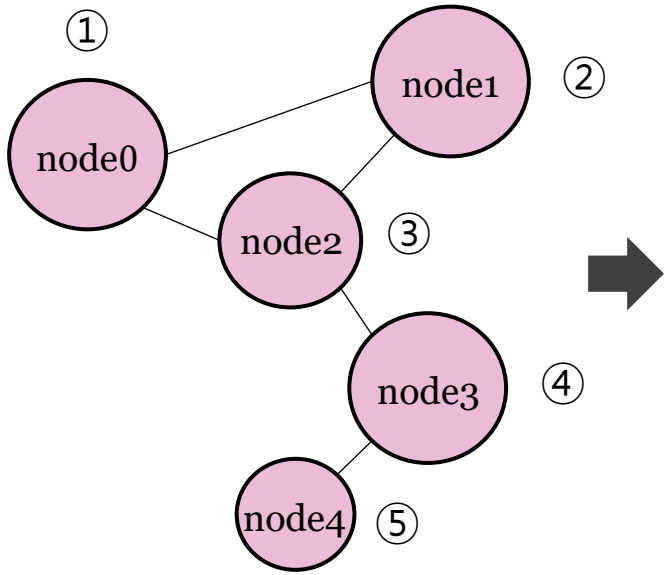
2개의 input

- X : node feature matrix (노드 정보)
- edge_index : adjacency matrix (연결 정보)
- H : hidden state
- Y : output

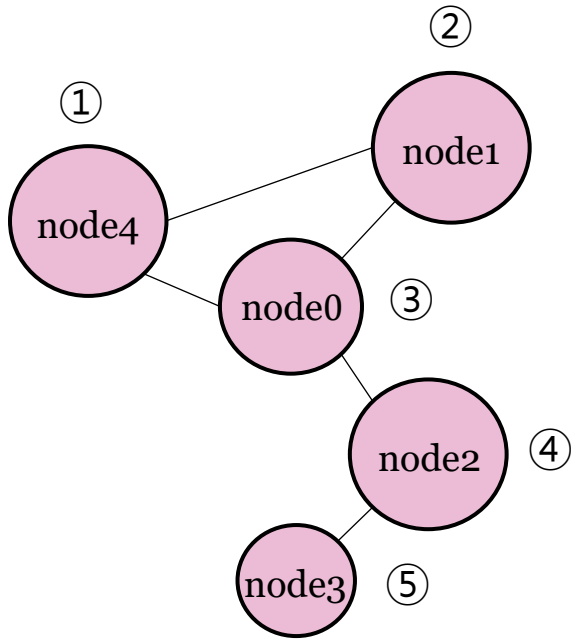
read out

❖ 노드 순서(배치)만 변경되는 경우, 동일 그래프지만 matrix는 변함

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



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



노드가 바뀌어서 변경되는 모든 행렬들 → **permutation set**

permutation에 상관없게(permutation invariance) 학습되도록 하기 위한 layer

*1차원 벡터 형태로 변환 후 활성화 함수 → linear layer

prediction

❖output

- node, graph
- 2개의 input (matrix)로부터 1개의 output 생성

❖loss에 따라 회귀, 분류 문제 모두 가능

- 회귀 : Mean Squared Error
- 분류 : Cross Entropy Error

prediction

❖prediction

- 노드간의 관계 학습 후 예측

  ex) 추천 시스템, 행동 예측, 분자구조로 향기 예측

❖graph classification

- 그래프 자체를 분류

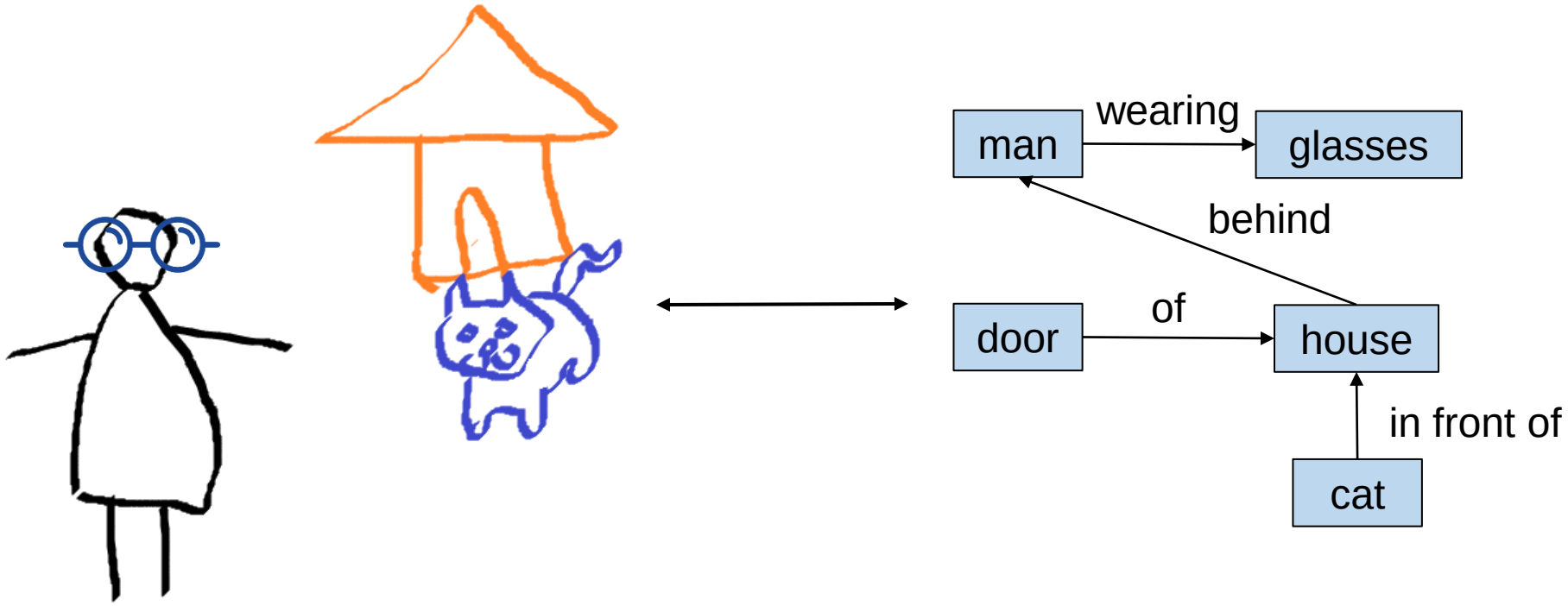
  ex) 분자 구조 분류, 악성 댓글 분류, 논문 주제 분류

❖node classification

- 그래프를 구성하는 노드 레벨의 분류

image generation from graph

- 탐지된 물체들을 graph로 만들어 탐지뿐만 아니라 이미지에서의 관계까지 파악 가능
- 반대로 graph를 사용하여 이미지 생성 또한 가능



summary

❖input data

- graph (node feature matrix, adjacency matrix)

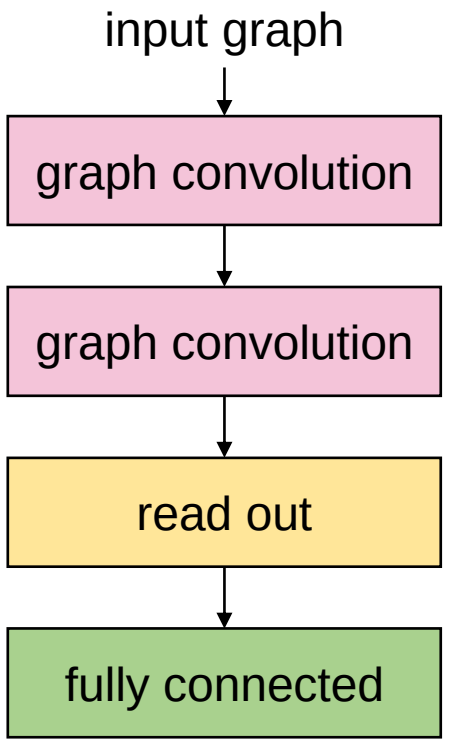
❖training

- graph convolution : 여러 개 거쳐 고도화된 정보 학습
- read out : 1차원 벡터형태로 변형 → 노드의 배치에 상관 없도록 함

❖predict

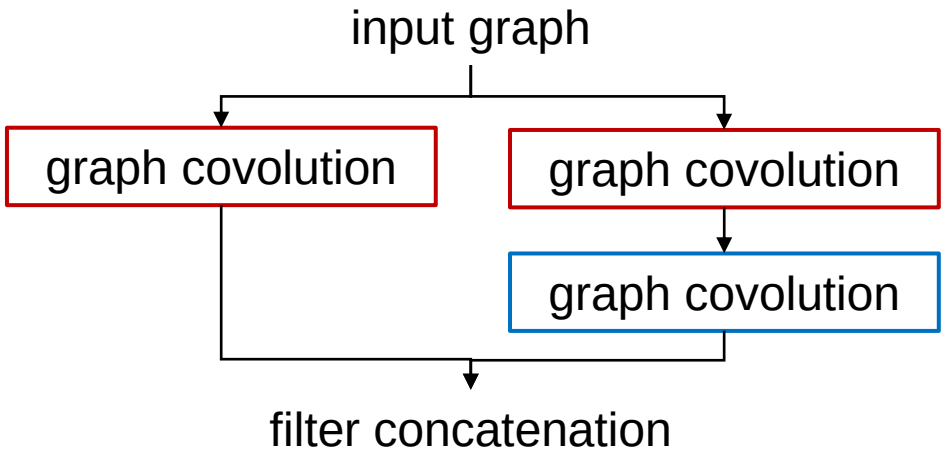
- fully connected layer → 회귀, 분류 가능 (node level, graph level)

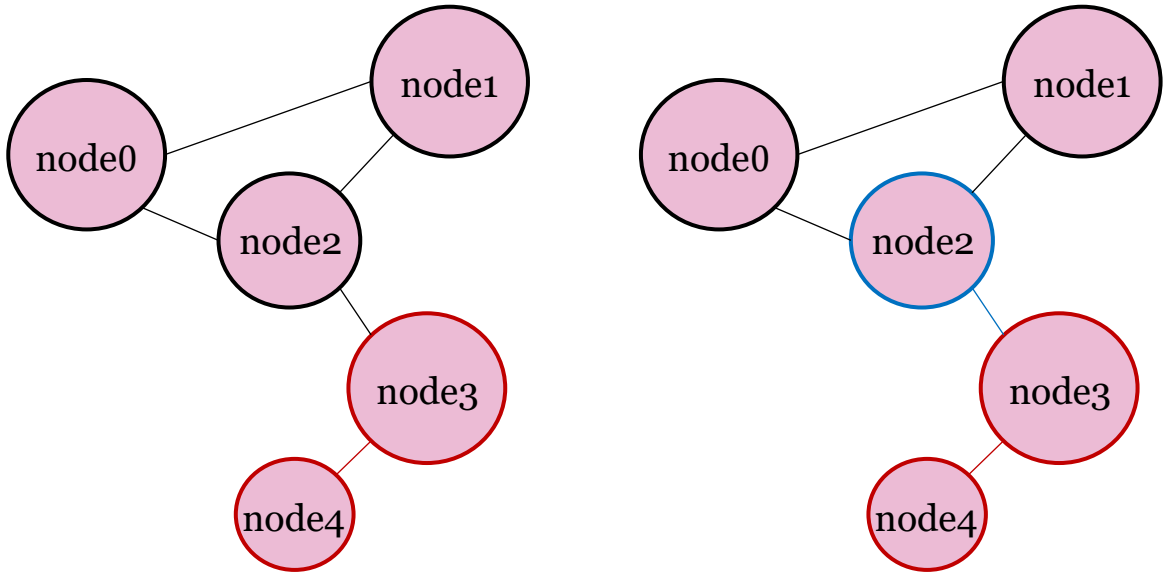
❖layer마다 node feature matrix, weight가 갱신되며 훈련



inception

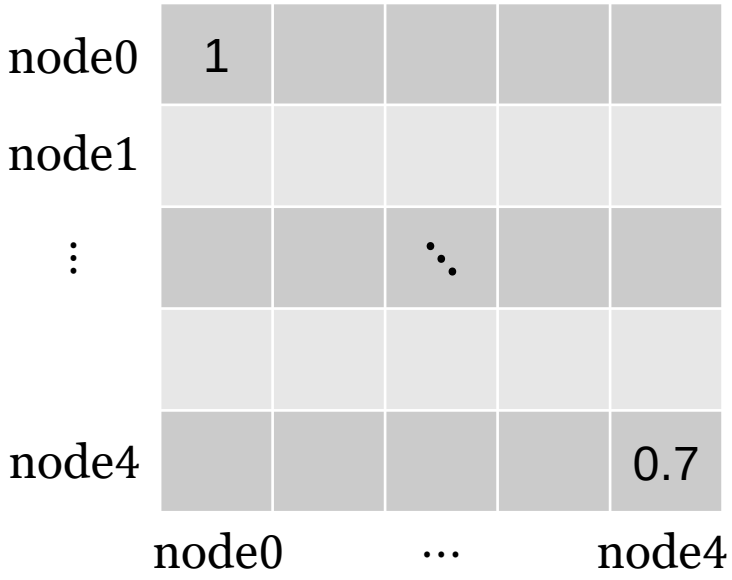
- inception module
  - 다양한 크기의 필터를 사용하고, concatenate하여 다음 레이어로 입력
  - 필터가 많을수록 더 큰 범위의 값을 학습 가능 → 거리가 얼마인 노드까지 포함할 것인지
- graph convolution 횟수 == 거리
  - 1번 → 거리가 1인 노드들
  - 2번 → 거리가 2인 노드들의 정보까지 반영

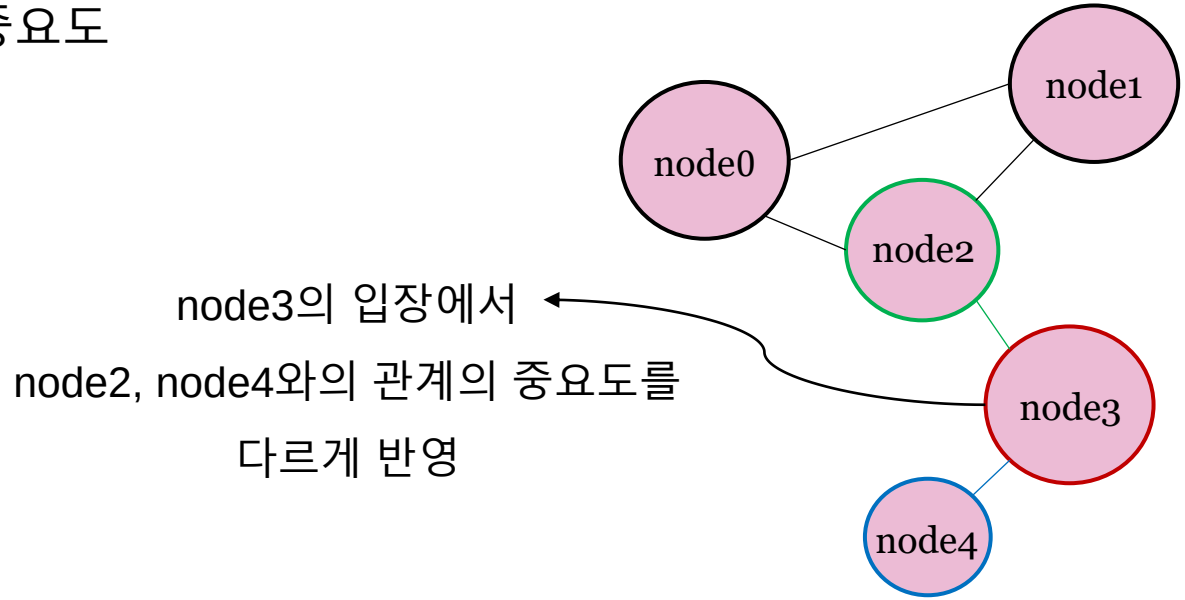




attention

- 하나의 노드의 갱신할 때 관련 노드들은 모두 같은 비율로 반영 → Graph Attention Networks(GAT)
- 적절한 비율을 정하여 노드마다 다르게 적용
  - 해당 비율은 행렬(α) 형태로 곱함 (행렬 곱x)
  - α_{ij} → i번째, j번째 노드 간의 상관관계의 중요도





Q & A