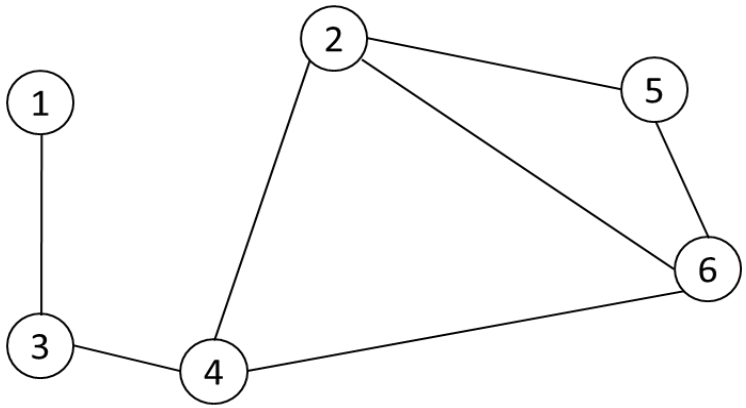
Graph Neural Network

Sang Yup Lee

GNN

- What is GNN?
  - It is a type of neural networks that can be applied to graph data.
- Then what is a graph (a.k.a., network)?
  - Something that is composed of nodes (or vertices) and ties (edges) between nodes.
  - A graph can be denoted as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, \ldots, v_{\mathrm{N}}\}$ is a set of N = $|\mathcal{V}|$ nodes and $\mathcal{E} = \{e_1, \ldots, e_{\mathrm{M}}\}$ is a set of M edges.
  - Example graph

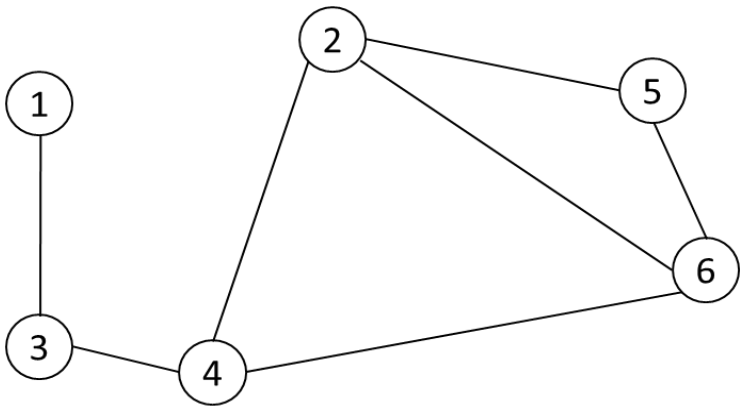


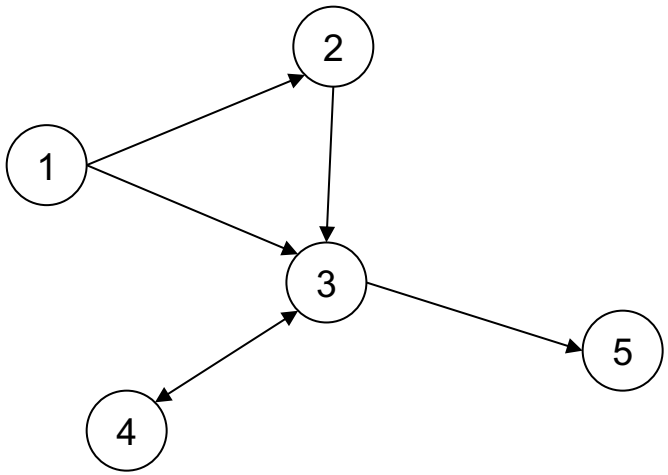
Graph

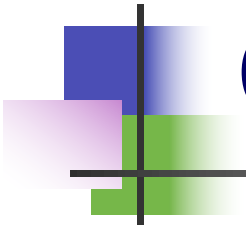
- 그래프의 예
  - 소셜 네트워크, 인터넷, 분자, 텍스트, 지역, 인용 네트워크 등
  - 타이의 종류 두 가지
    - 방향성이 없는 타이 (undirected tie)
      - 대칭적 타이 (symmetric tie) 라고도 함
      - 친구 관계, 연인 관계, 페이스북에서의 친구 관계 등
    - 방향성이 있는 타이 (directed tie)
      - 비대칭 (asymmetric) 타이라고도 함
      - 의사-환자, follower-followee 등

Graph

- 그래프의 종류
  - Undirected graph
    - Undirected ties로 구성된 그래프
    - 예) 친구 관계 네트워크
  - Directed graph
    - Directed ties로 구성된 그래프
    - 예) 인용 네트워크





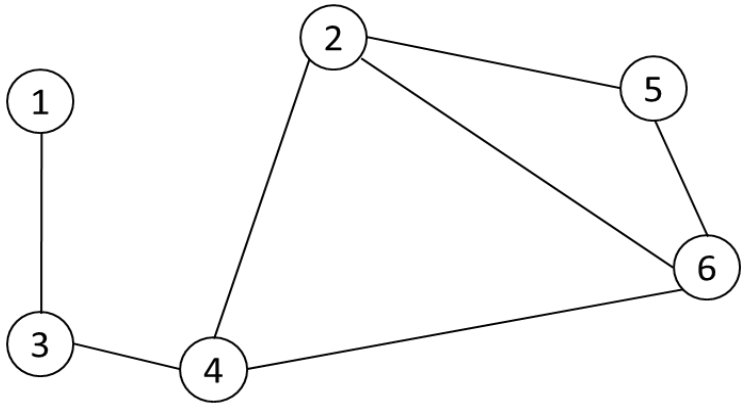


GNN

- GNN 관련 주요 작업의 종류
  - Node level tasks
    - Node classification
      - 예) 소셜 네트워크에서 사람들의 정치 성향
    - Nodes clustering
    - Identifying influential nodes
  - Graph level task
    - Graph classification
      - 예) 문서 분류, 분자 분류 등
  - Edge level task
    - Edge prediction (link prediction)

Graph

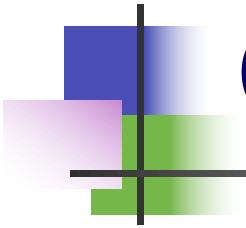
- 파이썬을 이용한 그래프 분석 (network analysis 라고도 함)
  - NetworkX 모듈 사용
    - https://networkx.org/
  - 예제 그래프



Graph

- 파이썬을 이용한 그래프 분석
  - 예제 코드: graph_basics.ipynb
- Main procedure of graph analysis using NetworkX
  - 1) Prepare graph data
    - that is information about nodes and their ties
  - 2) Construct a graph representing the graph data using NetworkX
    - create an empty graph
    - add nodes to the graph
    - add edges to the graph
  - 3) Analysis
    - Do graph analysis using NetworkX
      - We usually do EDA with NetworkX
    - Or export network data to other programs and do some analysis

Graph Neural Networks



GNN

- GNN 작업의 주요 과정
  - 분석하고자 하는 그래프 데이터 준비
    - 무엇을 노드로 하고, 노드들 간의 연결을 어떻게 정의할 것인지에 대해서 생각해 보는 것이 필요
  - 분석 목적에 적합한 GNN 모형 적용, 학습
  - 예측 / Inference

GNN

- GNN 작업의 주요 과정의 예: 문서 분류
  - 주요 과정은 아래와 같음
    - 텍스트 데이터 수집 및 전처리
    - 분석 목적 (즉, 문서 분류)을 위해 텍스트 데이터를 어떠한 형태의 그래프로 / 어떻게 표현할 것인지를 결정하는 것 필요 (선행연구들을 참고할 수 있음)
      - 예) 하나의 문서를 하나의 그래프로 표현 => 단어가 노드가 되고 단어들 간의 연결 관계가 타이 정보가 됨 => 그래프 분류 문제
      - 전체 코퍼스를 하나의 그래프로 표현 => 단어와 문서가 각각의 노드가 됨 => 노드 분류 문제
    - 데이터를 학습 데이터와 평가 데이터로 구분
    - 분석 목적과 데이터 특성에 적합한 GNN 알고리즘 선택 및 적용
      - GCN, GraphSAGE, GAT 등 많은 알고리즘 존재
    - 예측

GNN

- GNN 학습의 주요 목적
  - 학습을 통해 노드 또는 그래프의 특성을 잘 나타내는 embedding 벡터를 생성하는 것 ⇒ representation learning으로 간주 가능
  - Node level의 작업을 위해서는 작업과 관련된 node의 특성을 잘 반영하는 embedding vector를 생성하는 것이 중요
  - Graph level의 작업을 위해서는 그래프의 특성을 잘 반영하는 embedding vector를 생성하는 것이 중요
    - 그래프의 임베딩 벡터는 일반적으로 노드의 특성을 나타내는 임베딩 벡터 정보를 활용해서 만들어 짐 (예, 평균, LSTM 등)
  - 이를 위해서 일반적으로 GNN 모형은 크게 filtering 부분과 pooling 부분으로 구성

GNN

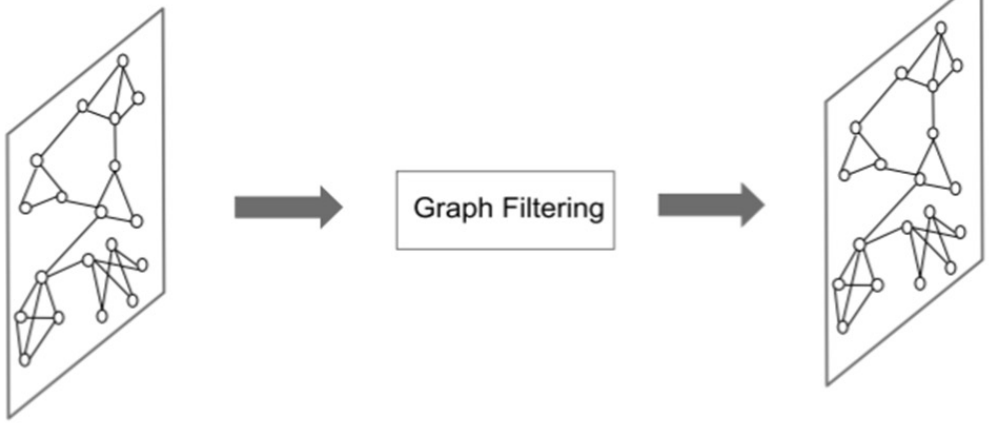
- Filtering
  - 노드의 임베딩 정보를 계산하기 위해서는 일반적으로 두 가지 정보를 사용
    - 노드의 특성 정보 (i.e., features)
      - 노드의 속성 정보라고도 함
    - 그래프의 구조 정보
      - 이는 일반적으로 노드들 간의 연결 정보를 의미
  - 노드의 특성 정보와 그래프의 구조 정보를 이용해서 노드의 임베딩 벡터를 계산하는 과정은 아래와 같이 표현

$$\mathbf{F}^{(\mathrm{of})} = h\left(\mathbf{A}, \mathbf{F}^{(\mathrm{if})}\right)$$

- $\mathbf{A} \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ denotes the adjacency matrix of the graph with N nodes ⇒ graph structure 의미
- $\mathbf{F}^{(\mathrm{if})} \in \mathbb{R}^{\mathrm{N}\times d_{\mathrm{if}}}$ and $\mathbf{F}^{(\mathrm{of})} \in \mathbb{R}^{\mathrm{N}\times d_{\mathrm{of}}}$ denote the input and output feature matrices where d_{if} and d_{of} are their dimensions, respectively

GNN

- Filtering (cont'd)
  - 이렇게 노드의 특성 정보와 구조 정보 (즉, 인접행렬)를 이용해서 특성 정보를 계산 또는 업데이트하는 과정을 graph filtering이라고 함
  - $h()$ 가 그래프 필터가 되며, 그래프 필터에는 여러 가지 종류가 존재
  - 이러한 필터링 과정을 여러번 반복 (여러 개의 신경망 층을 적용한다고 생각할 수 있음)



$\mathbf{A} \in \{0,1\}^{N \times N}, \mathbf{F}^{(\text{if})} \in \mathbb{R}^{N \times d_{\text{if}}}$

$\mathbf{A} \in \{0,1\}^{N \times N}, \mathbf{F}^{(\text{of})} \in \mathbb{R}^{N \times d_{\text{of}}}$

- Node level 작업의 경우, 이러한 filtering 과정을 거쳐, 노드의 특성을 잘 나타내는 임베딩 벡터를 얻는 것으로 충분
- 하지만, graph level 작업의 경우는 추가적인 과정이 더 필요 ⇒ 풀링 과정

GNN

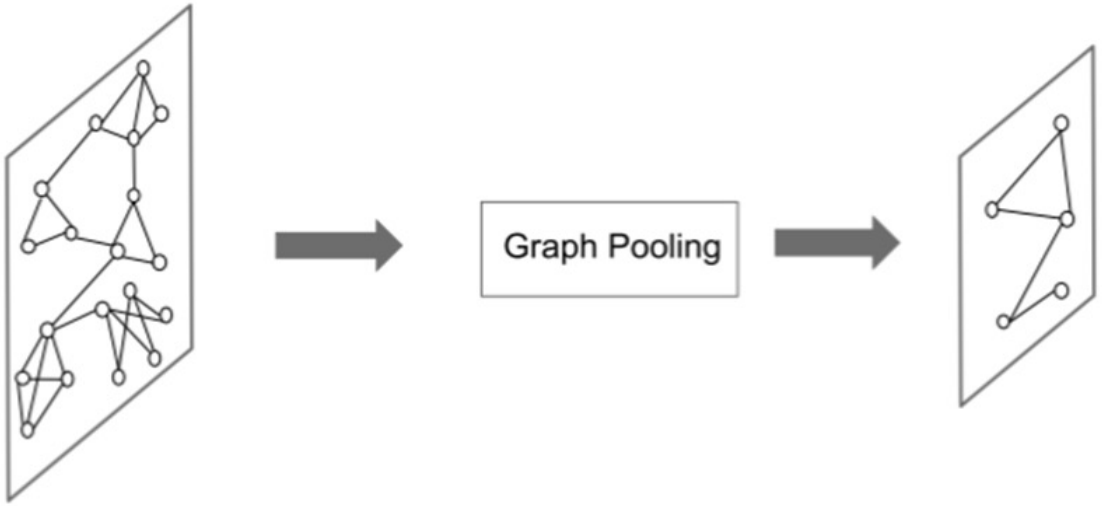
- Pooling
  - Graph level 작업에서는 filtering을 거쳐 업데이트된 노드의 임베딩 정보를 통합해서 그래프의 특성을 나타내는 임베딩 정보를 계산하는 것이 필요 ⇒ pooling 과정이 필요
  - Pooling 과정에서는 하나의 그래프에 대한 정보 (즉, 인접행렬과 노드의 피쳐 혹은 임베딩 정보)를 입력 받아, 노드의 수가 적은 그래프를 결과물로 출력 (이러한 그래프를 coarsened graph라고 표현)
  - Pooling 과정의 결과물 ⇒ coarsened graph의 인접행렬과 노드 임베딩 벡터. 이는 아래와 같이 표현

$$\mathbf{A}^{(\mathrm{op})}, \mathbf{F}^{(\mathrm{op})} = \mathrm{pool}(\mathbf{A}^{(\mathrm{ip})}, \mathbf{F}^{(\mathrm{ip})})$$

where $\mathbf{A}^{(\mathrm{ip})}, \mathbf{F}^{(\mathrm{ip})}, \mathbf{A}^{(\mathrm{op})}, \mathbf{F}^{(\mathrm{op})}$ are the adjacency matrices and feature matrices before and after the pooling operation, respectively.

GNN

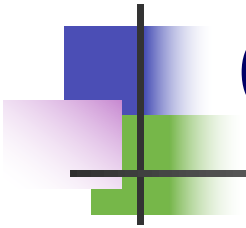
- Pooling (cont'd)
  - 이를 그림으로 표현하면 아래와 같음



$\mathbf{A}^{(\text{ip})} \in \{0,1\}^{N_{\text{ip}} \times N_{\text{ip}}}, \mathbf{F}^{(\text{ip})} \in \mathbb{R}^{N_{\text{ip}} \times d_{\text{ip}}}$

$\mathbf{A}^{(\text{op})} \in \{0,1\}^{N_{\text{op}} \times N_{\text{op}}}, \mathbf{F}^{(\text{op})} \in \mathbb{R}^{N_{\text{op}} \times d_{\text{op}}}$

N_{op} denotes the number of nodes in the coarsened graph and $N_{\text{op}} < N_{\text{ip}}$.

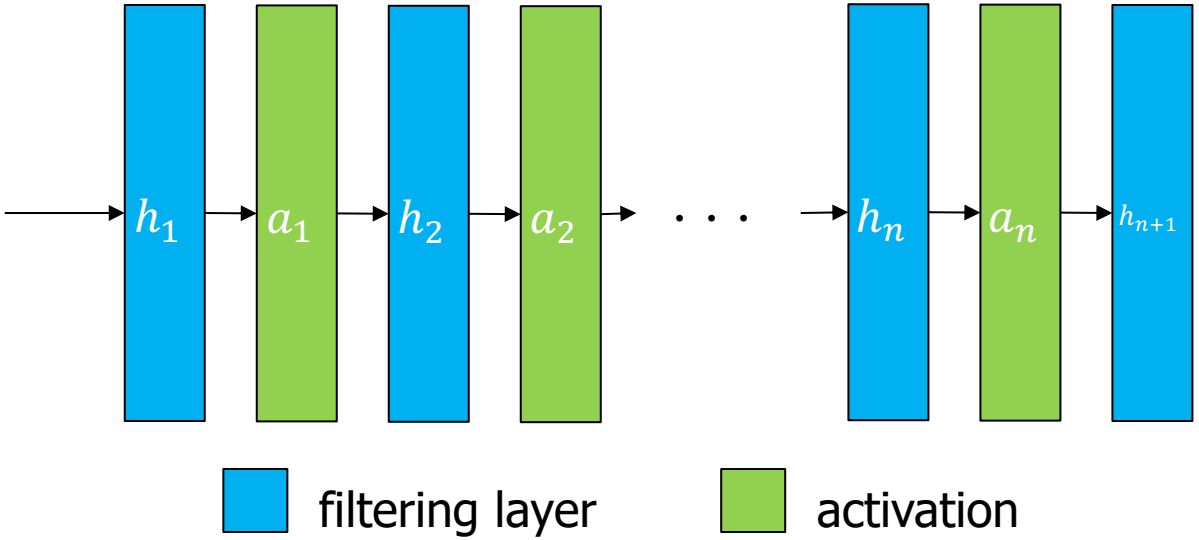


GNN

- 일반적인 GNN 모형의 구조
  - 여러 개의 filtering 부분과 pooling 부분으로 구성
  - Node level task
    - 여러 개의 filtering 사용 (No pooling required)
      - 하나의 filtering이 일반적으로 하나의 신경망 층을 의미하기 때문에 여러 개의 신경망 층을 사용한다는 것을 의미
  - Graph level task
    - Uses both graph filtering and graph pooling operations

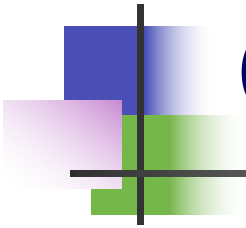
GNN

- Node level 작업의 일반적 구조
  - 일반적으로 그래프 필터링 + 비선형 활성화 함수로 구성
  - downstream task에 적합한 노드 특성 정보 추출 (임베딩 생성)
  - downstream task를 위한 추가적인 층 사용



$$\mathbf{F}^{(i)} = h_i\left(\mathbf{A}, \alpha_{i-1}\left(\mathbf{F}^{(i-1)}\right)\right)$$

- $\mathbf{F}^{(i)}$ denotes the output of the ith graph filtering layer. $\mathbf{F}^{(i)} \in \mathrm{R}^{\mathrm{N}\times\mathrm{di}}$
- $\alpha_{i-1}()$ is the element-wise activation function following the $(i-1)$st graph filtering layer
- The final output $\mathbf{F}^{(L)}$ is leveraged as the input to some specific layers according to the downstream node-focused tasks.

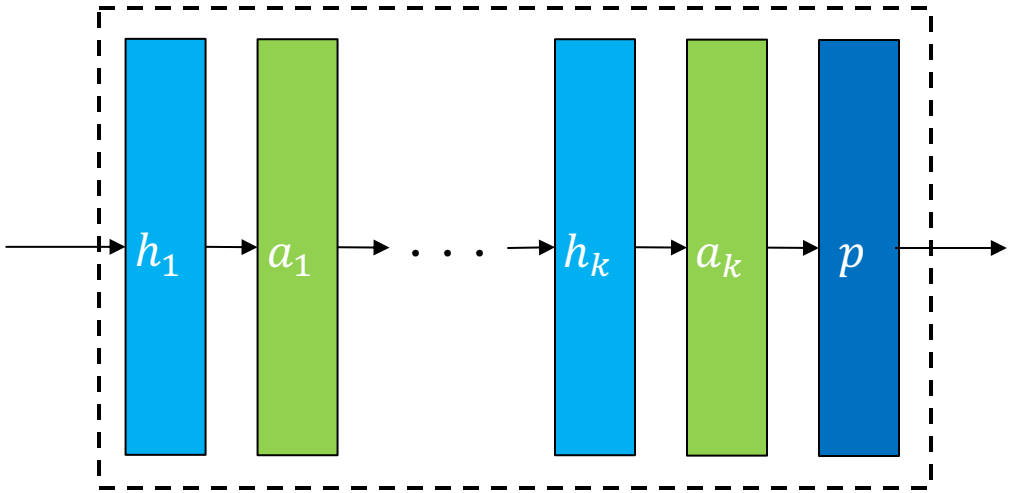


GNN

- Graph level 작업의 일반적 구조
  - 일반적으로 filtering layer + activation + (graph) pooling layer로 구성
  - The graph pooling layer is utilized to summarize the node features and generate higher-level features that can capture the information of the entire graph.
  - Typically, a graph pooling layer follows a series of graph filtering and activation layers.
  - A coarsened graph with more abstract and higher-level node features is generated after the graph pooling layer.

GNN

- Graph level 작업의 일반적 구조 (cont'd)
  - 아래와 같은 block 여러 개 사용

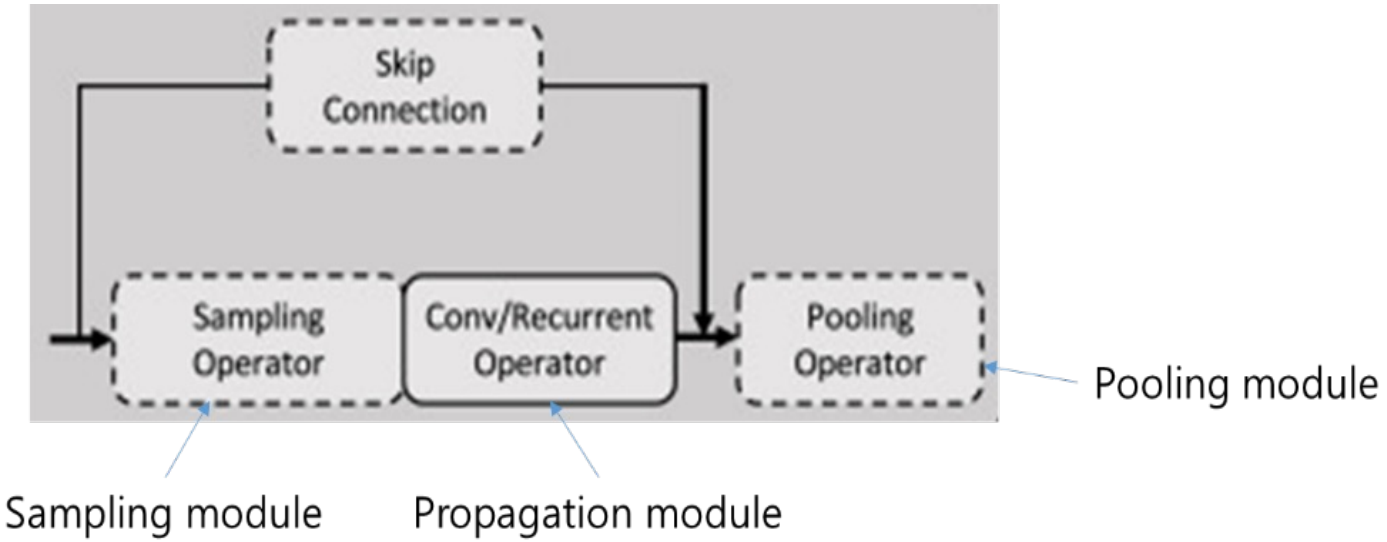


<A block in GNN for graph level tasks>

The input of the block is the adjacency matrix $A^{(ib)}$ and the features $F^{(ib)}$ of a graph $G^{ib} = \{V^{ib}, E^{ib}\}$ and the output are the newly generated adjacency matrix $A^{(ob)}$ and the features $F^{(ob)}$ for the coarsened graph $G^{ob} = \{V^{ob}, E^{ob}\}$

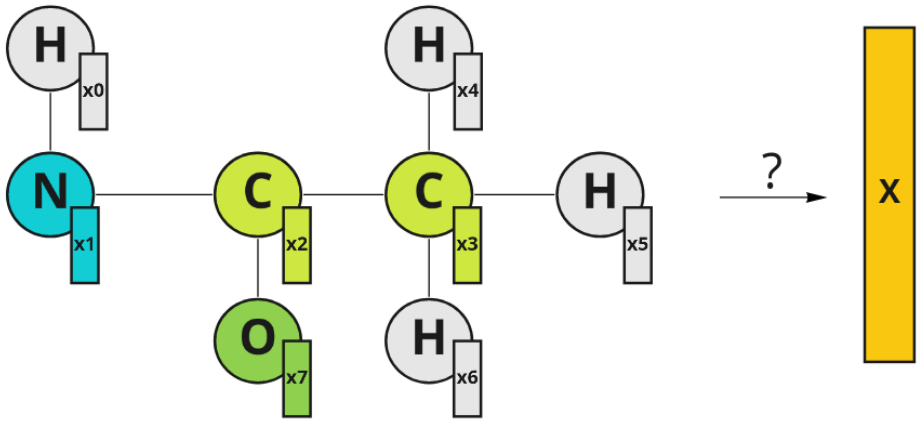
GNN

- GNN block의 또 다른 예시
  - sampling operation과 skip connection이 포함된 경우



GNN

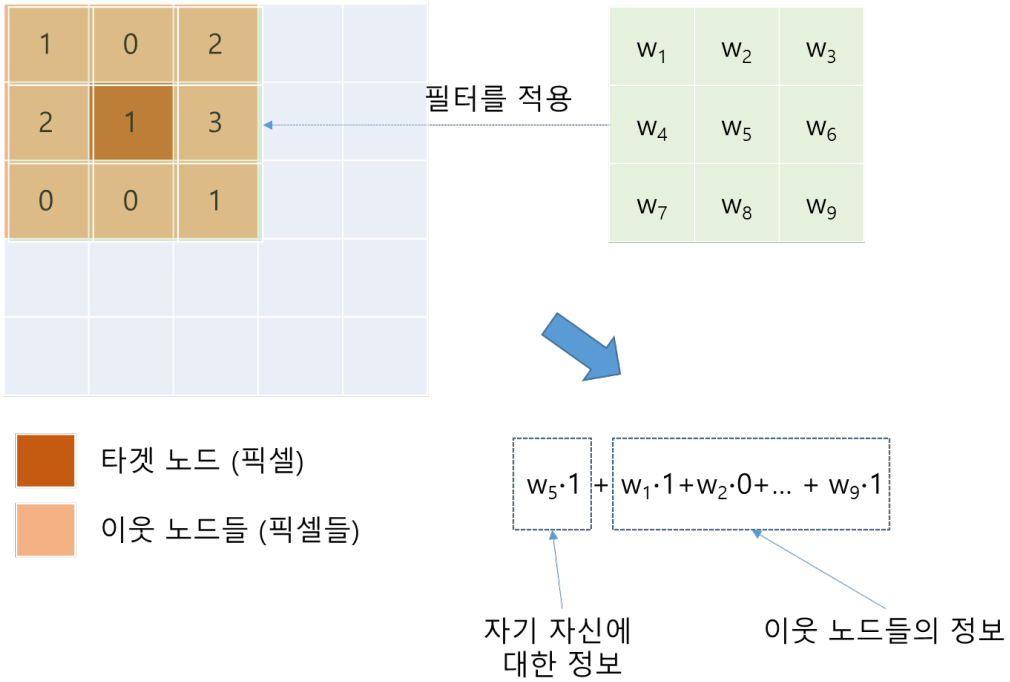
- Filtering의 예
  - Example graph



- Graph-level task $X = \frac{1}{8}\sum_{i=0}^{7} x_i$?

GNN

- Filtering의 예
  - 이미지에서의 convolutional filter



GNN

- Filtering 의 예
  - Graph convolutions (1-hop nodes)

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

A

x

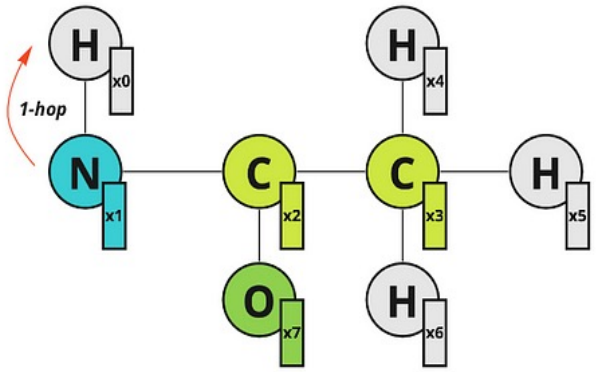
x0
x1
x2
x3
x4
x5
x6
x7

X

=

x1
x0 + x2
x1 + x3 + x7
x2 + x4 + x5 + x6
x3
x3
x3
x2

X'



GNN

- Filtering 의 예
  - Graph convolutions (2-hop nodes)

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

A^2

x

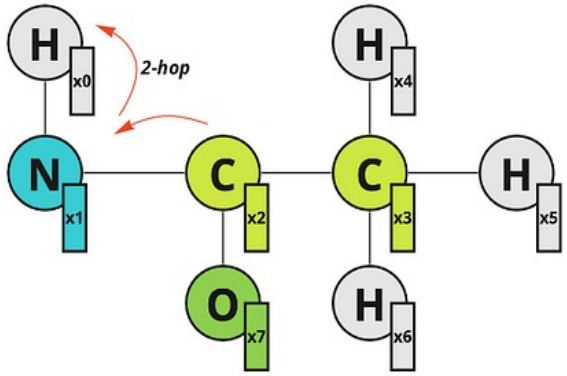
x0
x1
x2
x3
x4
x5
x6
x7

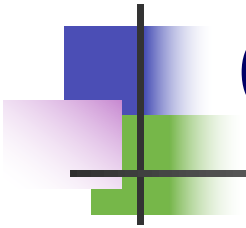
X

=

x0 + x2
2x1 + x3 + x7
x0 + 3x2 + x4 + x5 + x6
x1+ 4x3 + x7
x2 + x4 + x5+ x6
x2 + x4 + x5+ x6
x2 + x4 + x5+ x6
x1 + x3 + x7

X'



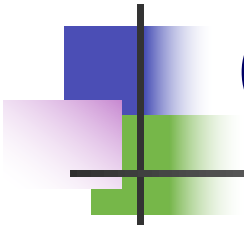


GNN

- Filtering 의 예

$$P = w_0 I + w_1 A + w_2 A^2 + \cdots + w_n A^n$$

$$x' = Px$$



GNN

- Python libraries
  - PyTorch Geometric
  - Deep Graph Library (DGL)
  - Spektral
- Python code
  - Graph_classification_example.ipynb