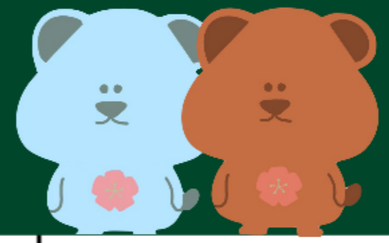
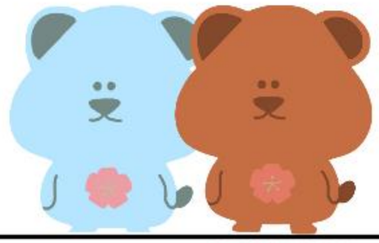


1주차 세션

이다현 최하경

EWHA EURON

목차



EWHA EURON

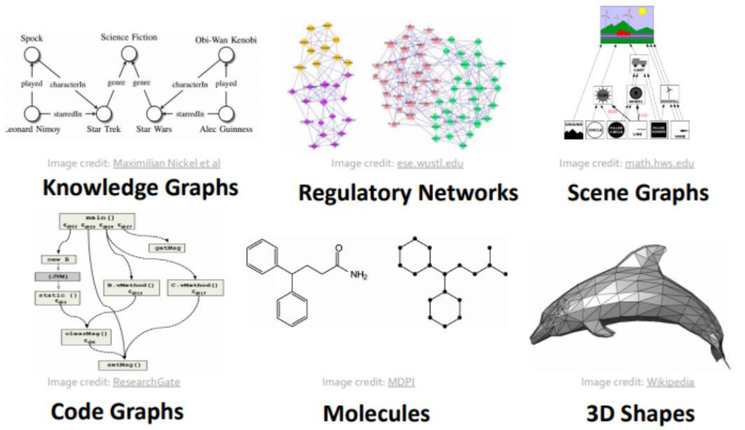
01. Why Graphs?

EWHA EURON

01. Why Graphs?

📌 Graph란?

- **"관계(relation)"와 "상호작용(interaction)"**으로 객체들을 분석하고 설명하는 언어
- Graph로 표현하고 모델링할 수 있는 도메인들이 다양함
- Graphical structure을 갖는 도메인은 두 가지로 표현
  - ① Network (Natural graphs)
  - ② Graph (representation)
  - -> 가끔 네트워크와 그래프 간 경계가 흐려지기도 함



📌 이 수업에서의 main question

- 더 나은 prediction을 위해 relational structure의 이점을 어떻게 사용해야 하는가
- 복잡한 도메인들은 rich relational structure을 갖기 때문에, 명확하게 relational modeling을 하는 것이 중요!!

EWHA EURON

01. Why Graphs?

📌 Modern ML Toolbox

- 간단한 sequence(ex. 텍스트, 음성 데이터)와 grid(ex. 이미지 데이터)를 위해 디자인됨
- graph에 적용하기 힘듦

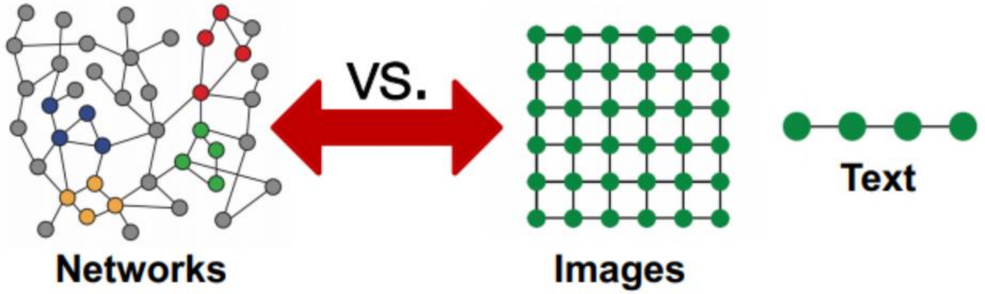
  (because) 네트워크의 복잡성

  ① arbitrary size와 복잡한 topological structure

  : grid와 텍스트 데이터처럼 공간적 기준점이 없음

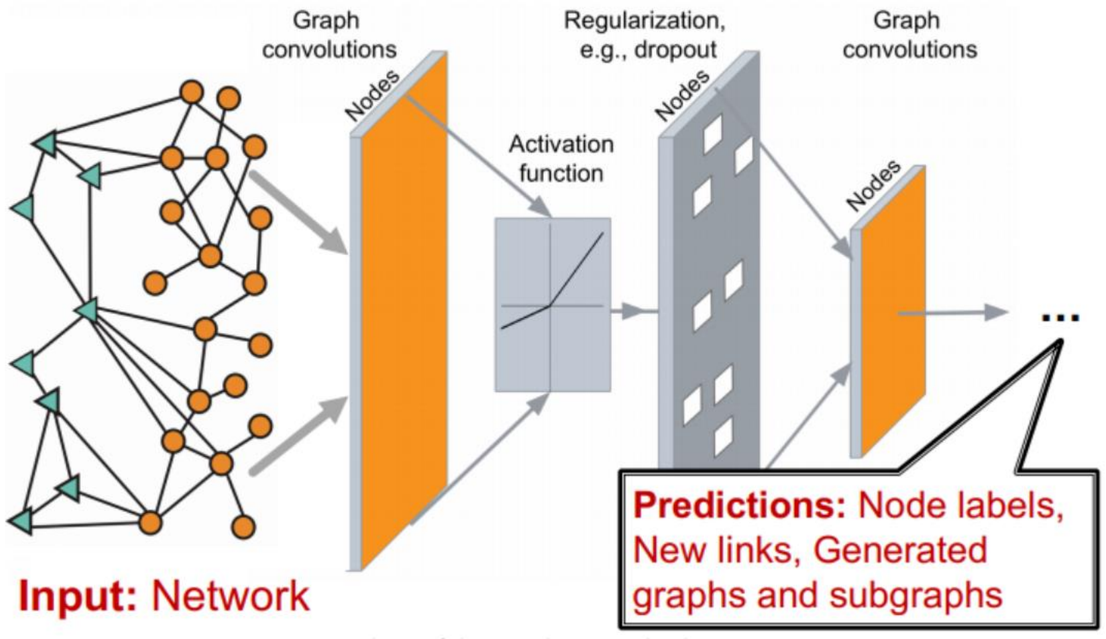
  ② 고정적인 노드의 순서가 없음

  ③ 가끔 dynamic하고 multimodal한 feature들이 존재



01. Why Graphs?

📌 Deep Learning in Graph

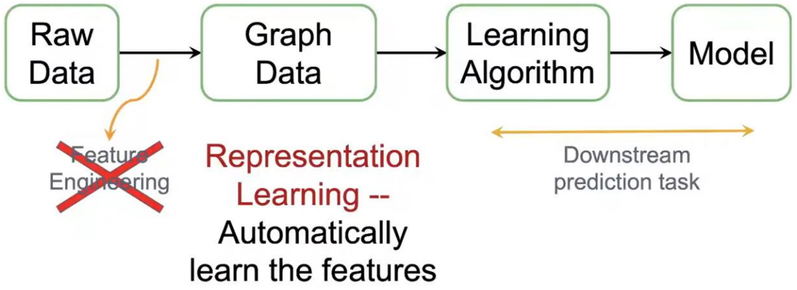


=> Question : 어떻게 이러한 모델을 만들 수 있는가?

01. Why Graphs?

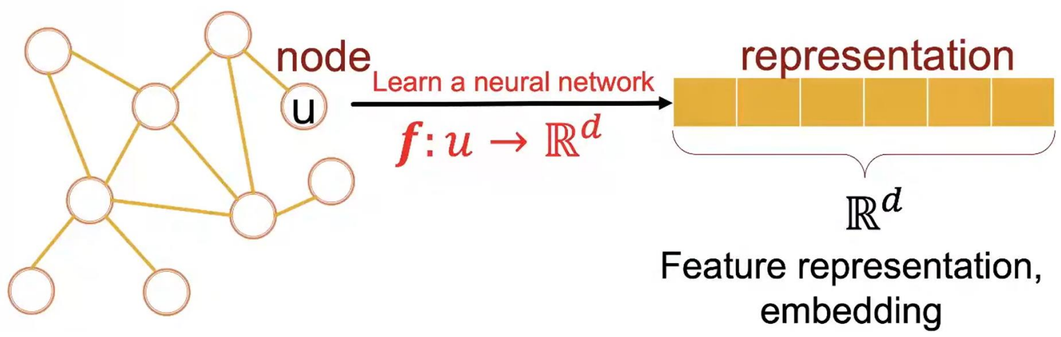
📌 Representative Learning (표현 학습)

- feature engineering 단계가 없어 그래프에서 특징을 자동으로 추출하거나 학습하도록 함



- 그래프의 노드를 d차원의 벡터로 임베딩하는 방법을 이용

  -> 네트워크에서 비슷한 노드들은 임베딩 공간에 가깝게 mapping됨



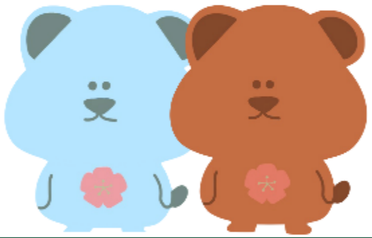
➡ 목표 : mapping 함수 f를 학습하는 것

01. Why Graphs?

📌 이 수업에서 배울 내용

- graph structured data를 이용한 machine learning & representative learning
  - ① 그래프를 나타내는 전통적인 방법 : Graphlets, Graph Kernels
  - ② node embedding 방법들 : DeepWalk, Node2Vec
  - ③ Graph Neural Networks : GCN, GraphSAGE, GAT, Theory of GNNs
  - ④ Knowledge graphs and reasoning : TransE, BetaE
  - ⑤ Deep generative models for Graphs
  - ⑥ Biomedicine과 과학, 산업에서의 응용

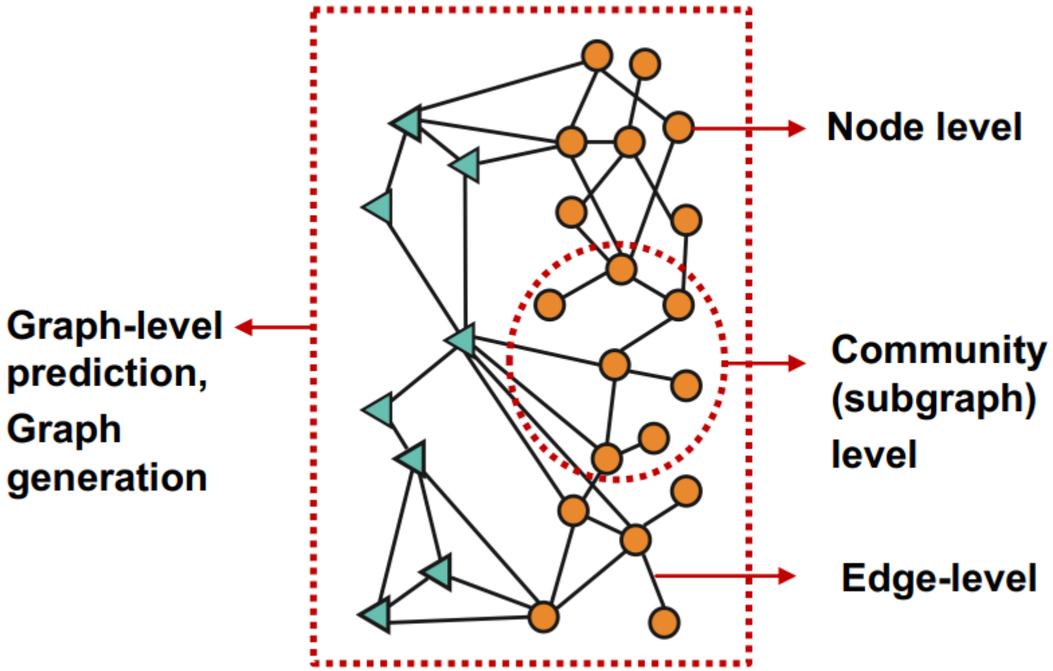
02. Applications of Graph ML



EWHA EURON

02. Applications of Graph ML

1 GNN TASK 종류



Node Edge Graph

① Node classification

② Line prediction

③ Graph classification

④ Clustering

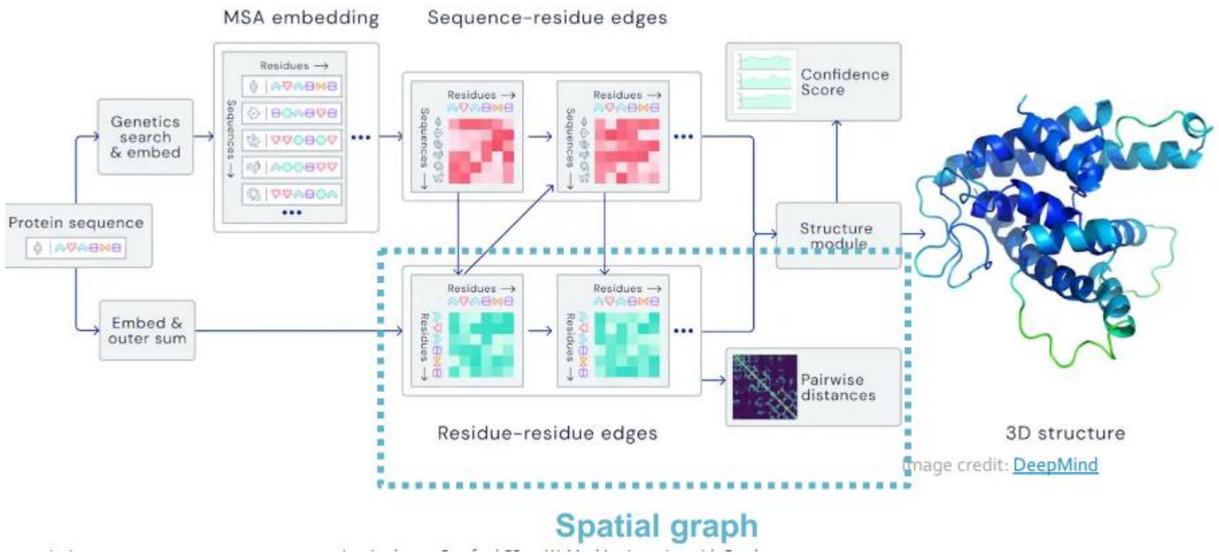
⑤ Graph generation, Graph evolution

High – impact applications

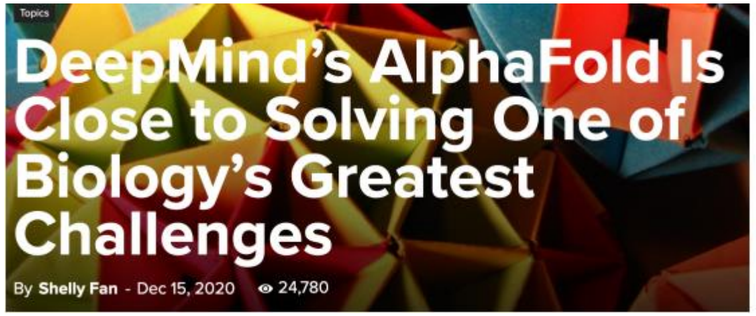
02. Applications of Graph ML

2 Node Level

(1) 단백질 생성 (Protein Folding)



- **Node** : Amino Acids in protein sequence
- **Edges** : Proximity between amino acids (residues)



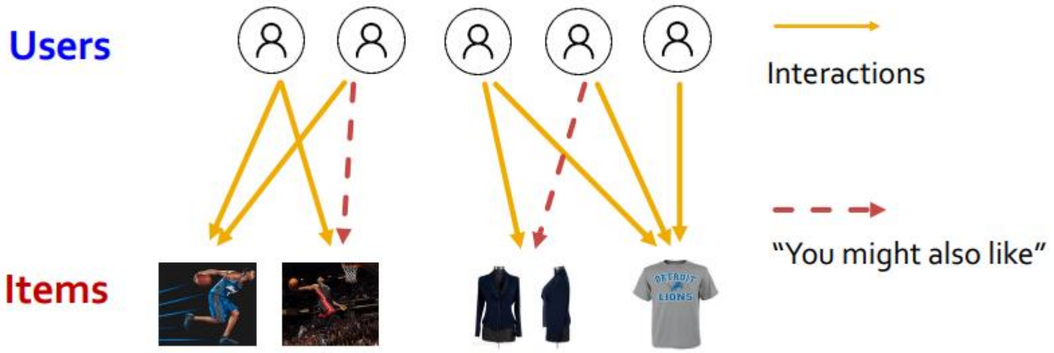
단백질 구조 예측대회 : 단백질의 3D 구조를 AI가 예측한다.

https://velog.io/@hewas1230/AlphaFold

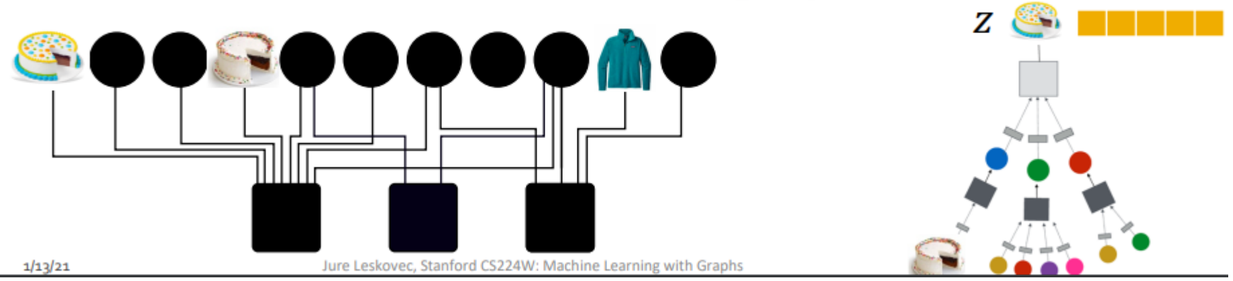
02. Applications of Graph ML

3 Edge Level

(1) 추천시스템



Predict whether two nodes in a graph are related



- **Node** : Users and items
- **Edges** : User-item interactions

  유저가 좋아할 법한 상품을 추천

Task: Learn node embeddings z_i such that

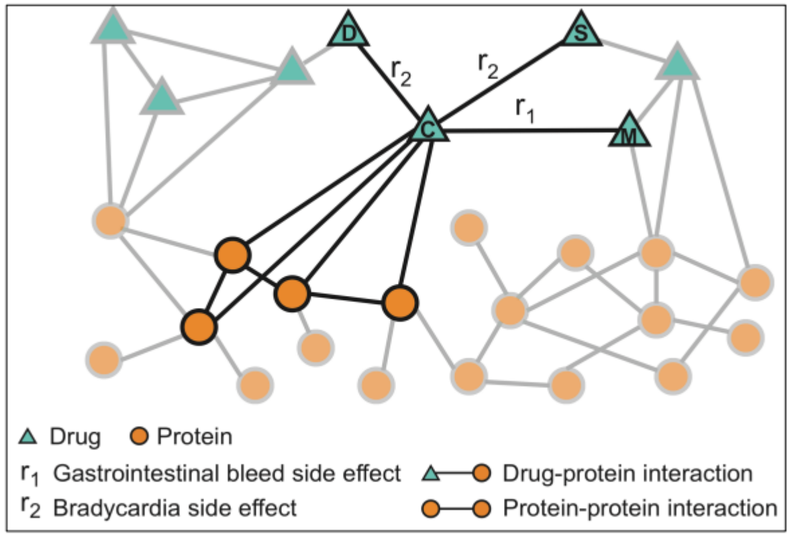
$$d(z_{cake1}, z_{cake2}) < d(z_{cake1}, z_{sweater})$$

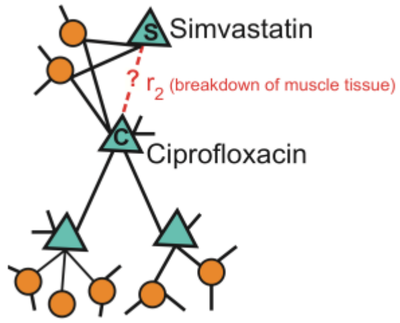
Distance

02. Applications of Graph ML

3 Edge Level

(2) 약물 복합 섭취 시 부작용 예측





- **Node** : Drugs & Protein
- **Edges** : Interactions

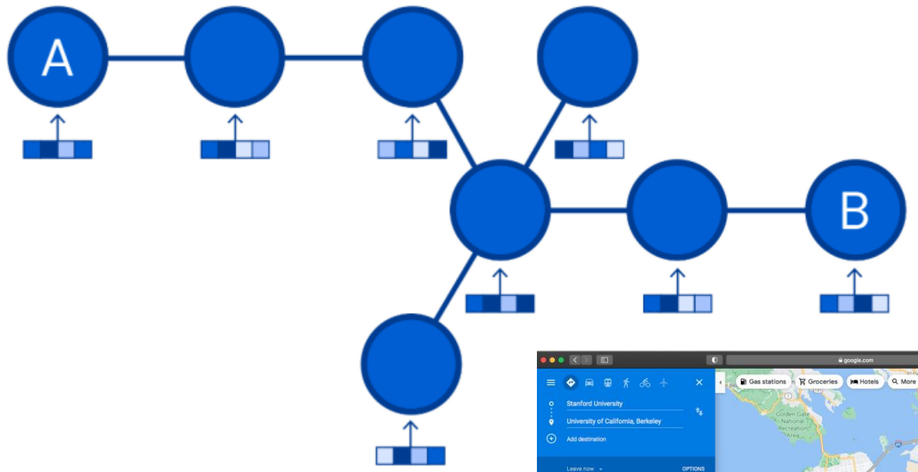
A 약과 B 약을 함께 복용했을 때,

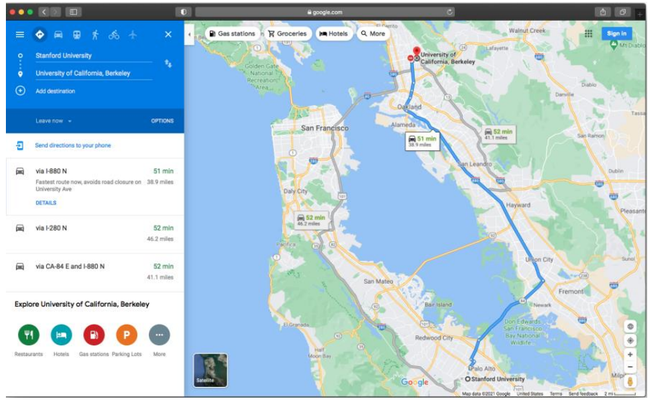
근육조직을 파괴하는가

02. Applications of Graph ML

4 Subgraph Level

(1) 지도 경로 예측 (구글맵)



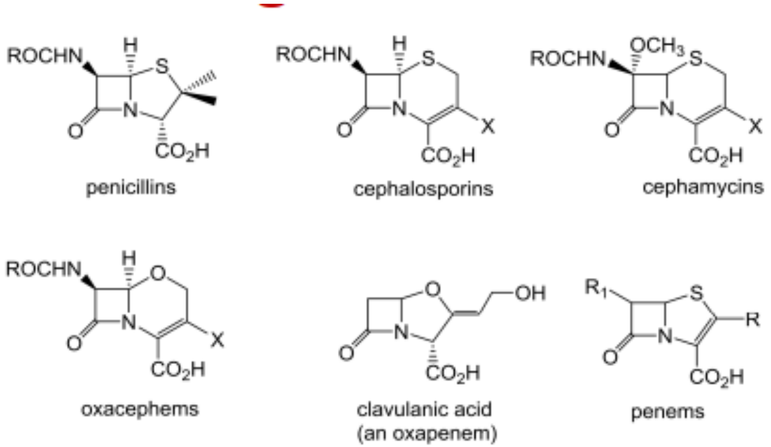


- **Node** : Road segments
- **Edges** : Connectivity between road segments

02. Applications of Graph ML

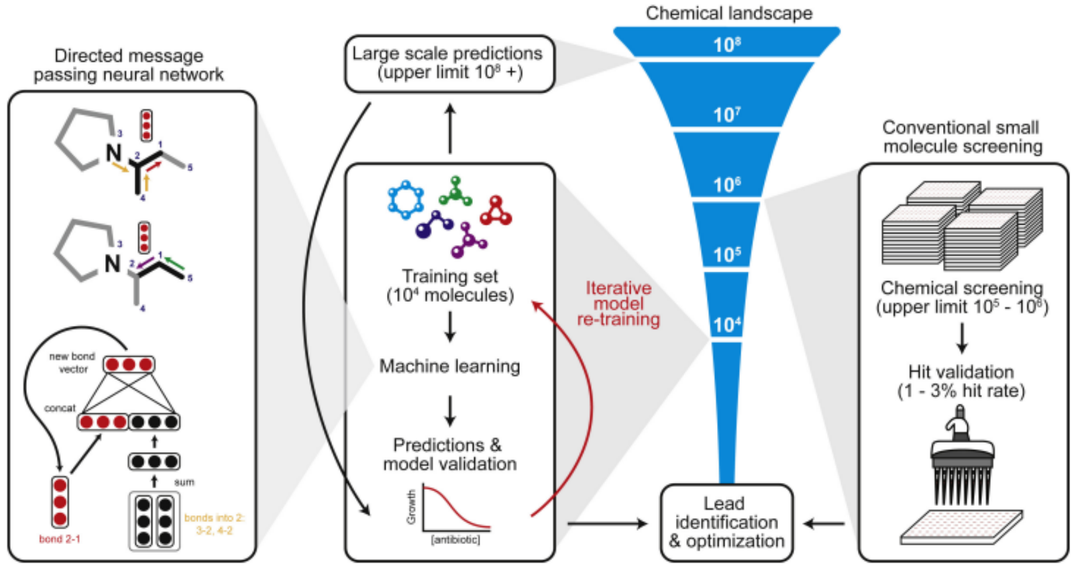
5 Graph Level

(1) 신약 개발



- **Node** : Atoms
- **Edges** : Chemical bonds

(2) 항생제 개발

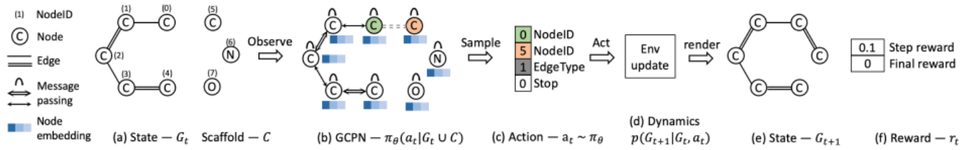


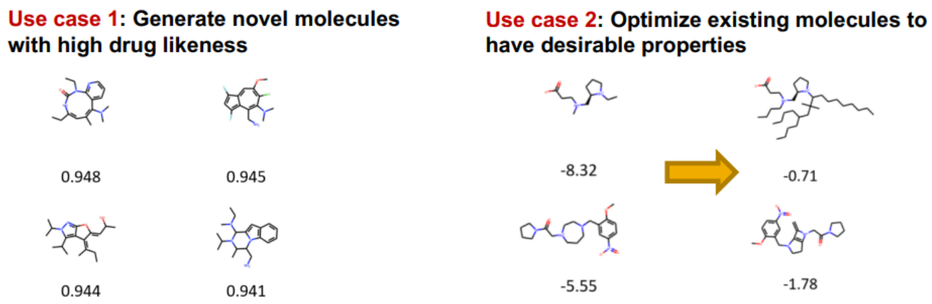
- Graph classification model

02. Applications of Graph ML

5 Graph Level

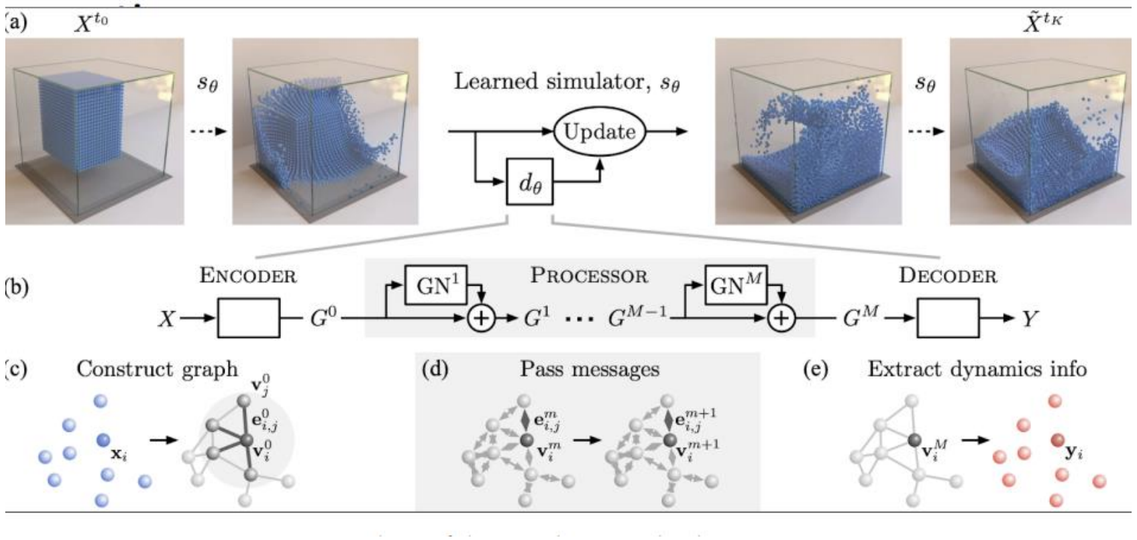
(3) 분자 생성/최적화





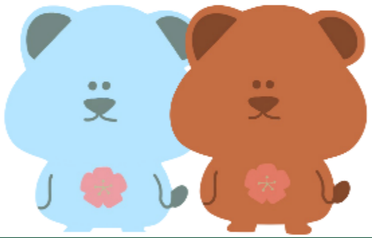
- Graph generation

(4) 시뮬레이션 (물리학)



- Graph evolution
- Nodes : 입자
- Edges : 입자 간 상호작용

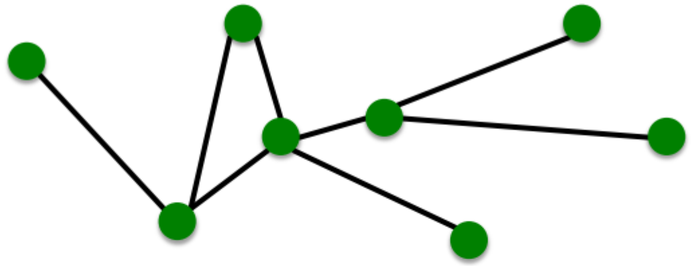
03. Choice of Graph Representation



EWHA EURON

03. Choice of Graph Representation

그래프 구조를 어떻게 **수학적으로** 표현할까

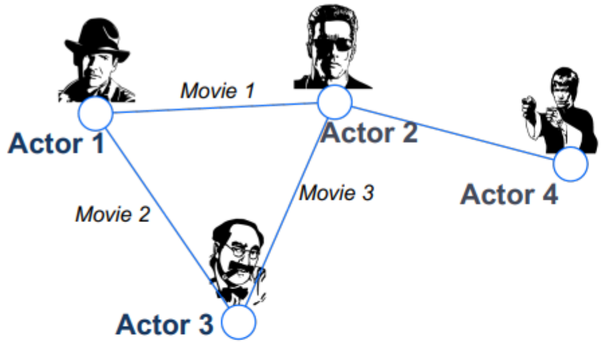


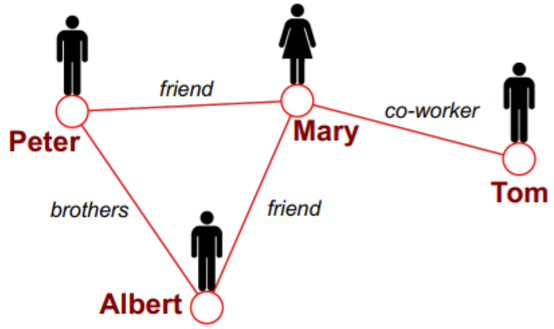
- **Objects:** nodes, vertices — N
- **Interactions:** links, edges — E
- **System:** network, graph — $G(N,E)$

- 대상 : nodes (N)
- 상호작용 : edges (E)
- 체계 : graph G(N,E)

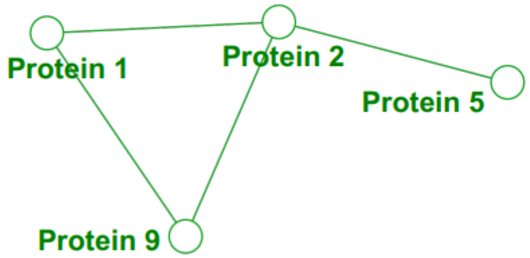
03. Choice of Graph Representation

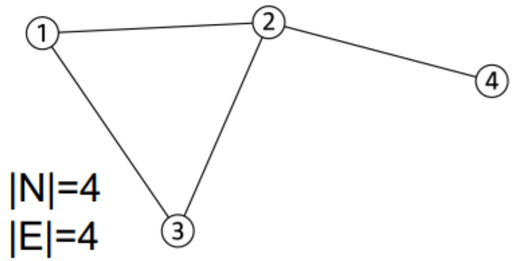
그래프는 일반적인 상황들을 표현할 수 있는 수단이다.





- 배우 – 작품
- 사람 – 관계 (지인, 친구)
- 단백질 – 연결관계





→ 모두 동일한 그래프 구조로 표현 가능

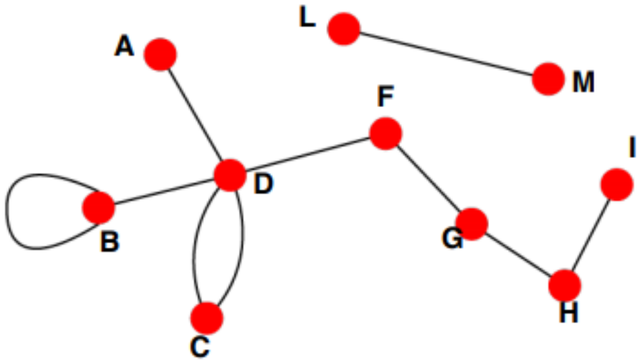
도메인, 정의한 문제에 따라 적절한 그래프 표현법을 선택해야 한다.

03. Choice of Graph Representation

① Direction of Edges

Undirected

- **Links:** undirected (symmetrical, reciprocal)

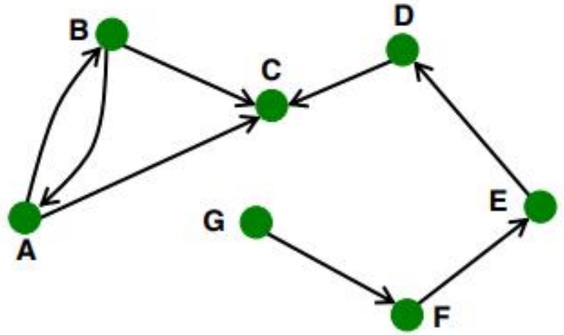


- 연결이 양방향인 그래프

- collaboration, frendship

Directed

- **Links:** directed (arcs)



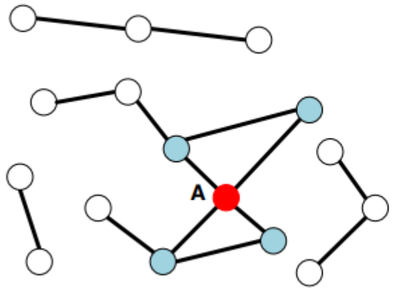
- 연결이 특정 방향으로 생성되는 그래프

- citation/인용, 인스타, DM, 팔로잉

03. Choice of Graph Representation

② Node Degrees

Undirected



Node degree, k_i: the number of edges adjacent to node ***i***

$k_A = 4$ → node A 에 연결된 링크는 4개

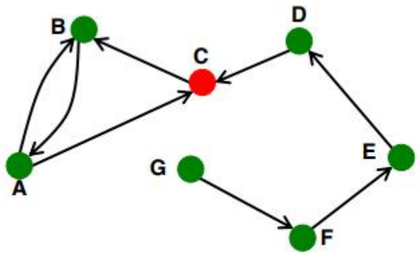
Avg. degree: $\bar{k} = \langle k \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i = \frac{2E}{N}$

- Degree

: 노드에 부속된 edge 의 개수

→ Undirected graph 평균 차수

Directed



Source: Node with $k^{in} = 0$
Sink: Node with $k^{out} = 0$

In directed networks we define an **in-degree** and **out-degree.**
The (total) degree of a node is the sum of in- and out-degrees.

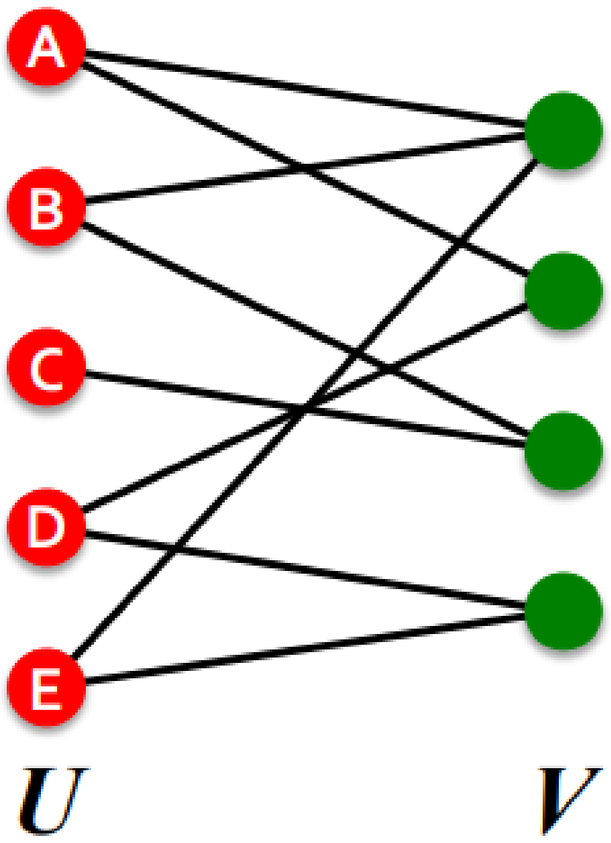
$k_C^{in} = 2 \quad k_C^{out} = 1 \quad k_C = 3$

$\bar{k} = \frac{E}{N}$ $\quad \overline{k^{in}} = \overline{k^{out}}$

→ Directed 의 경우 in-degree 와 out-degree 가 구분된다.

03. Choice of Graph Representation

③ Bipartite Graph



• Bipartite graph

: 서로 다른 종류의 독립된 노드들로 구성된 그래프

• EX. **U 에 속하는 노드는**

: V 에 속한 노드에게만 연결되고 U 끼리는 독립

(V 도 반대로 마찬가지)

• 이는 실제 도메인에서 많이 나타나는 구조이며 특히 추천시스템에서 많이 사용되는 개념임

03. Choice of Graph Representation

④ Graph 표현 방법 – (1) 인접행렬

$A_{ij} = 1$ if there is a link from node *i* to node *j*

$A_{ij} = 0$ otherwise

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \qquad A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Note that for a directed graph (right) the matrix is not symmetric.

- Adjacency Matrix

: 그래프 연결 유무를 1 과 0으로 나눠 행렬을 표현한 것

→ Undirected 의 경우 행렬이 대칭으로 나타나는데, Undirected 는 단방향도 존재하므로 대칭이 아닐 수도 있다.

👎 노드가 많아질수록 (행렬의 차원이 커질수록) 행렬이 Sparse 해지는 단점

Most real-world networks are sparse

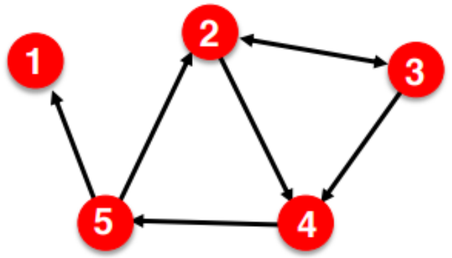
$E << E_{max}$ (or $k << N-1$)

EWHA EURON

03. Choice of Graph Representation

④ Graph 표현 방법 – (2) Edge list

- (2, 3)
- (2, 4)
- (3, 2)
- (3, 4)
- (4, 5)
- (5, 2)
- (5, 1)



- Edge list

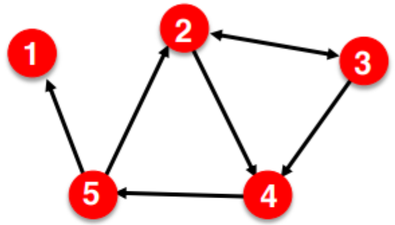
: Edge 를 연결된 노드 쌍으로 표현한 것

03. Choice of Graph Representation

④ Graph 표현 방법 – (3) 인접 list

Adjacency list:

- Easier to work with if network is
  - Large
  - Sparse
- Allows us to quickly retrieve all neighbors of a given node
  - 1:
  - 2: 3, 4
  - 3: 2, 4
  - 4: 5
  - 5: 1, 2



- Adjacency list

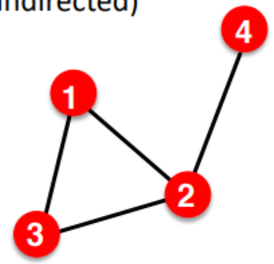
: 출발 방향의 노드를 Key 값으로 도착 노드를 Value 값으로 가지는 Dictionary 형태

: 단방향이거나 거대한 그래프에서 효율이 좋다.

03. Choice of Graph Representation

⑤ Edge 특성

- **Unweighted**
(undirected)



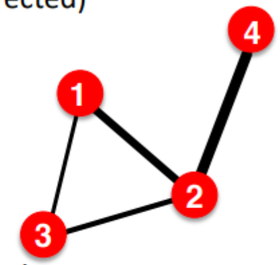
$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

$$E = \frac{1}{2}\sum_{i,j=1}^{N} A_{ij} \qquad \bar{k} = \frac{2E}{N}$$

Examples: Friendship, Hyperlink

- **Weighted**
(undirected)



$$A_{ij} = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

$$E = \frac{1}{2}\sum_{i,j=1}^{N} nonzero(A_{ij}) \qquad \bar{k} = \frac{2E}{N}$$

Examples: Collaboration, Internet, Roads

Possible options:

- Weight (*e.g.*, frequency of communication)
- Ranking (best friend, second best friend…)
- Type (friend, relative, co-worker)
- Sign: Friend vs. Foe, Trust vs. Distrust
- Properties depending on the structure of the rest of the graph: Number of common friends

- **Weight**

: edge 에 weight 를 줄 수 있음

- **Ranking**

: 연결 관계에 순위를 부여 (짱친 > 절친 > …)

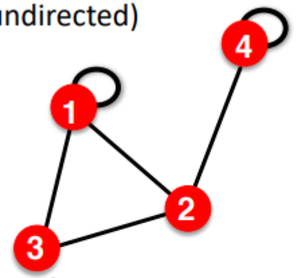
- **Type**

: 연결 관계의 유형을 부여 (친척, 직장동료)

03. Choice of Graph Representation

⑥ 그래프 유형 +

- **Self-edges (self-loops)**
(undirected)



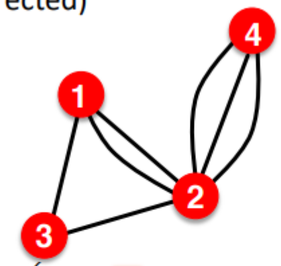
$$A_{ij} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

$$A_{ii} \neq 0 \qquad A_{ij} = A_{ji}$$

$$E = \frac{1}{2}\sum_{i,j=1,i\neq j}^{N} A_{ij} + \sum_{i=1}^{N} A_{ii}$$

Examples: Proteins, Hyperlinks

- **Multigraph**
(undirected)



$$A_{ij} = \begin{pmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 3 \\ 1 & 1 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{pmatrix}$$

$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

$$E = \frac{1}{2}\sum_{i,j=1}^{N} nonzero(A_{ij}) \qquad \bar{k} = \frac{2E}{N}$$

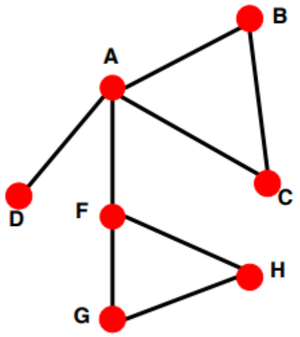
Examples: Communication, Collaboration

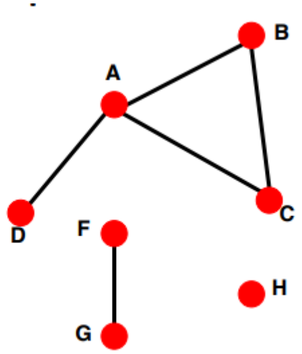
- 자기 자신과 연결된 형태
- 다중 연결

03. Choice of Graph Representation

⑦ Connectivity of Undirected/Directed Graphs

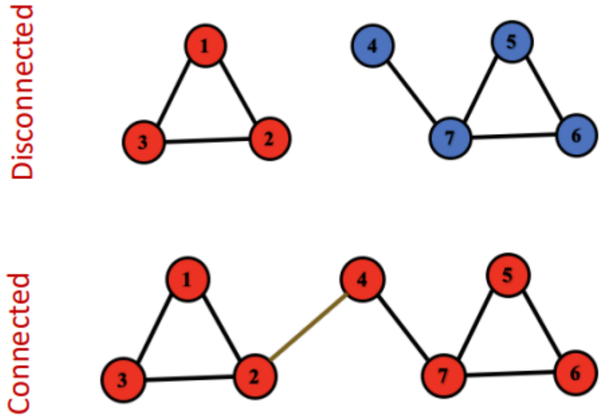
- Connected Graph (undirected) : 어떤 노드에서 출발하든지 다른 모든 노드로 도착할 수 있음
- Disconnected Graph : 최소 2개 이상의 connected graph 로 구성됨
- Bridge edge : 삭제되면 connected 에서 disconnected 로 바꿀 수 있는 Edge
- Articulation node : 삭제되면 connected 에서 disconnected 로 바꿀 수 있는 Node



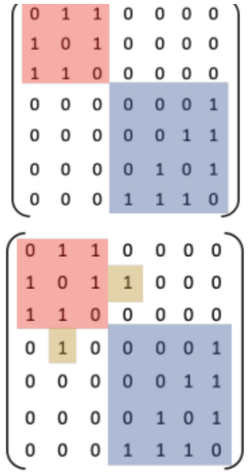


Largest Component:
Giant Component

Isolated node (node H)



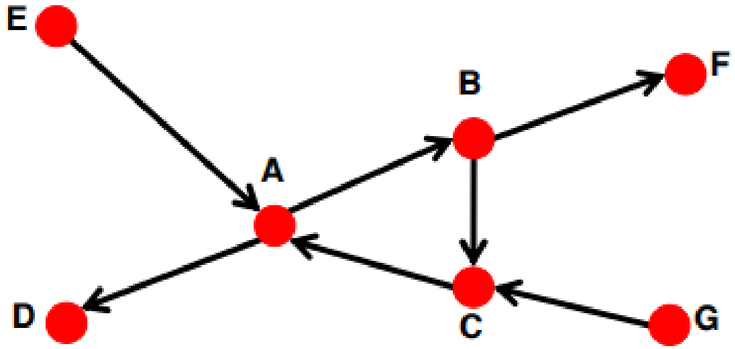
Block diagonal

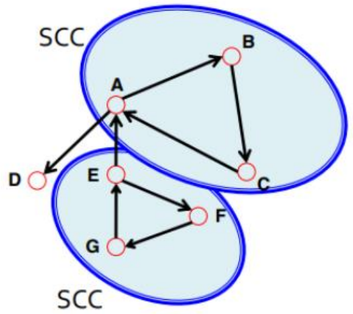


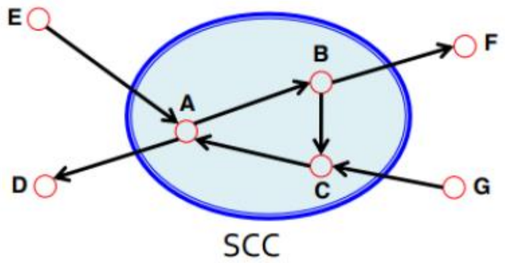
03. Choice of Graph Representation

⑦ Connectivity of Undirected/Directed Graphs

- Connected Directed Graph







✓ **Strong** Connected Directed Graph

: 어떤 노드에서 출발하던지 edge 방향을 지키면서 다른 모든 노드로 도착 가능

✓ **Weakly** Connected Directed Graph

: 어떤 노드에서 출발하던지 edge 방향을 무시한다면 다른 모든 노드로 도착 가능

✓ **SCCs (Strongly)**

: 그래프에서 부분적으로 나타나는 connected subgraphs

SUMMARY

(1) Machine learning with Graphs

- 그래프 구조란
- 응용사례

(2) Different types of tasks

- Node level
- Edge level
- Graph level

(3) Choice of a graph representation

- Directed , Undirected, Bipartite, Weighted, Adjacency matrix

(4) 복습과제 안내

예제를 통해 알아보는 pytorch Geometric

https://baeseongsu.github.io/posts/pytorch-geometric-introduction/

→ 필사하며 기본 메서드, 라이브러리 익히기

SUMMARY

GNN 라이브러리

(1) PyG : GNN 을 위한 파이썬 라이브러리 (https://github.com/pyg-team/pytorch_geometric)

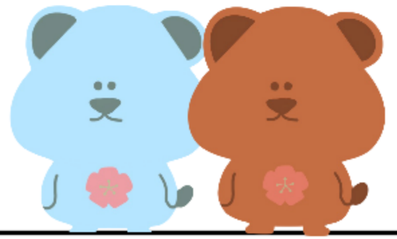
(2) GraphGym (https://github.com/snap-stanford/GraphGym)

(3) Deepsnap (https://github.com/snap-stanford/deepsnap/tree/master/examples)

(4) Tensorflow GNN (https://github.com/tensorflow/gnn)

+) 읽기자료 (GNN 을 활용한 요기요의 추천시스템 2022.07)

THANK YOU



EWHA EURON