

# CS224w

**CH1: Machine Learning for Graphs**  
**CH2: Traditional Methods for ML on Graphs**

# 1. Machine Learning with Graphs

- 실생활에서 얻을 수 있는 여러 종류의 그래프 데이터가 있다



Image credit: Medium

- Q) 이들 같은 네트워크(그래프)로부터 관계 구조 정보를 얻으면 예측에 도움이 될까?

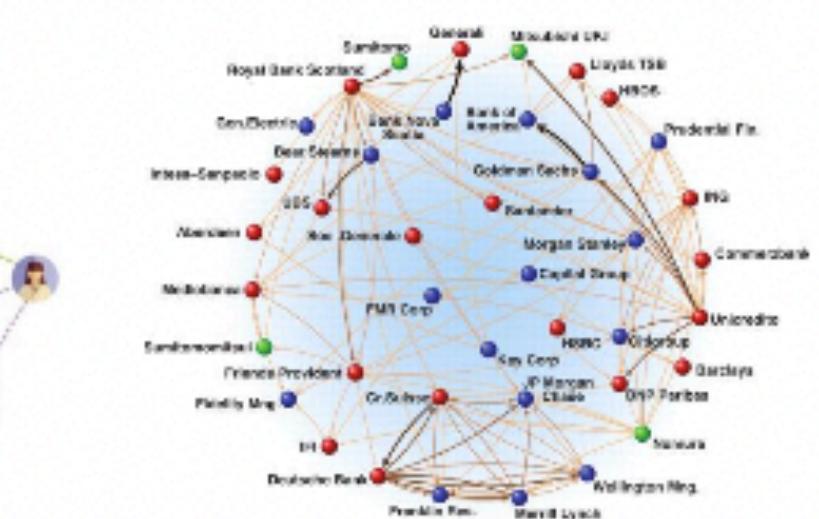


Image credit: Science

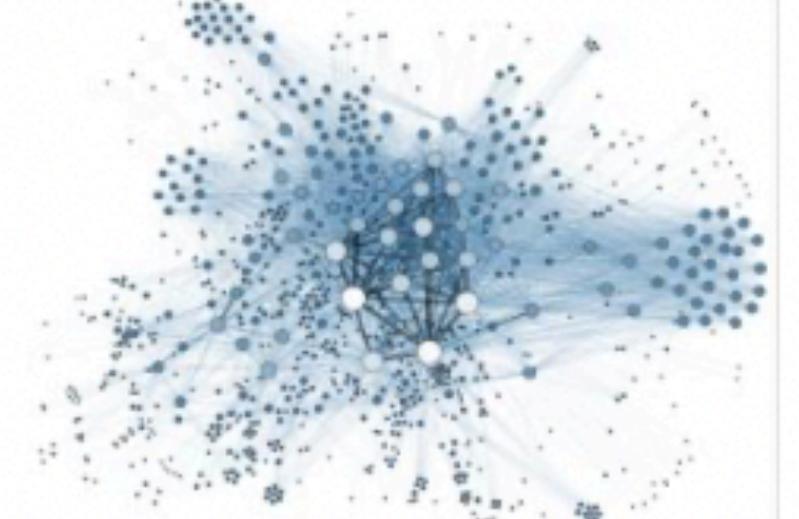


Image credit: Lumen Learning

**Social Networks**

**Economic Networks**

**Communication Networks**



Image credit: Missoula Current News

- Ans) Complex domains have a rich relational structure, which can be represented as a **relational graph**

**Citation Networks**

**Internet**

**Networks of Neurons**

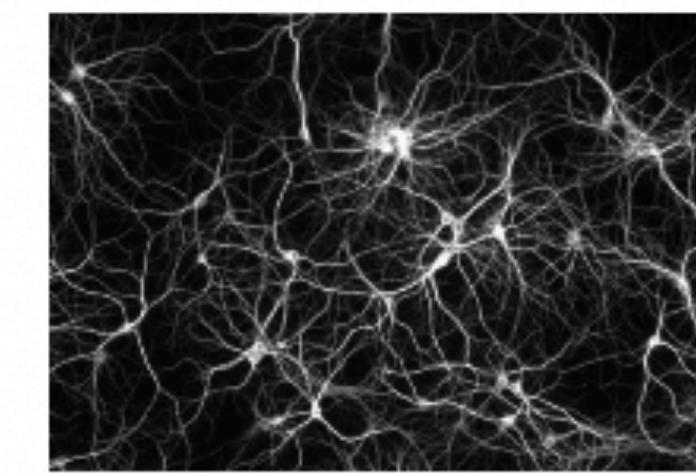
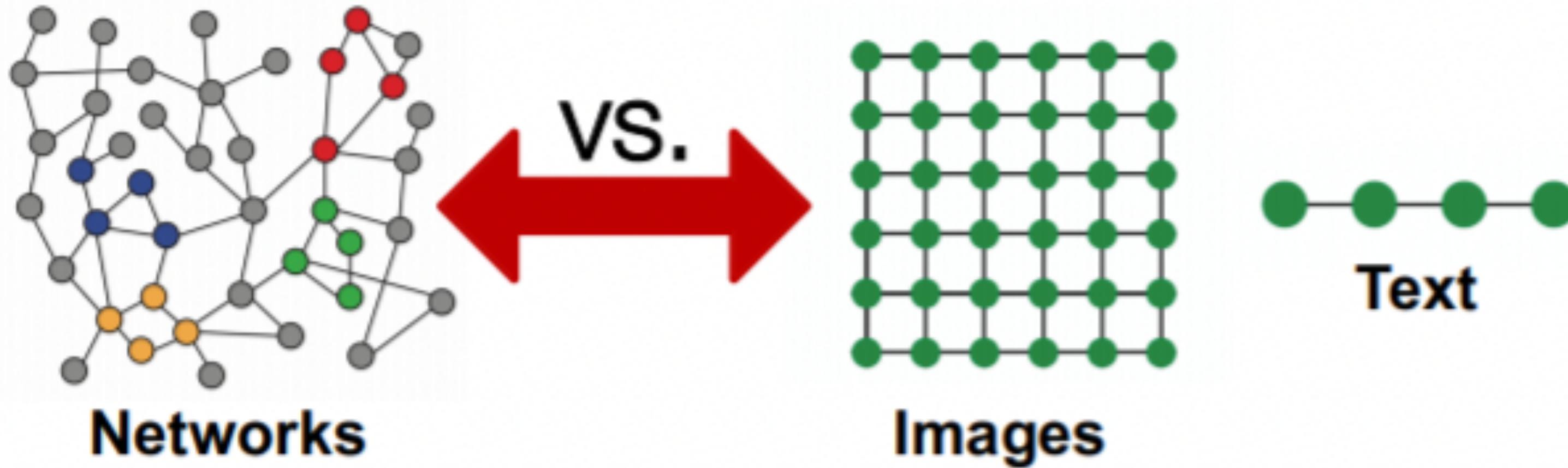


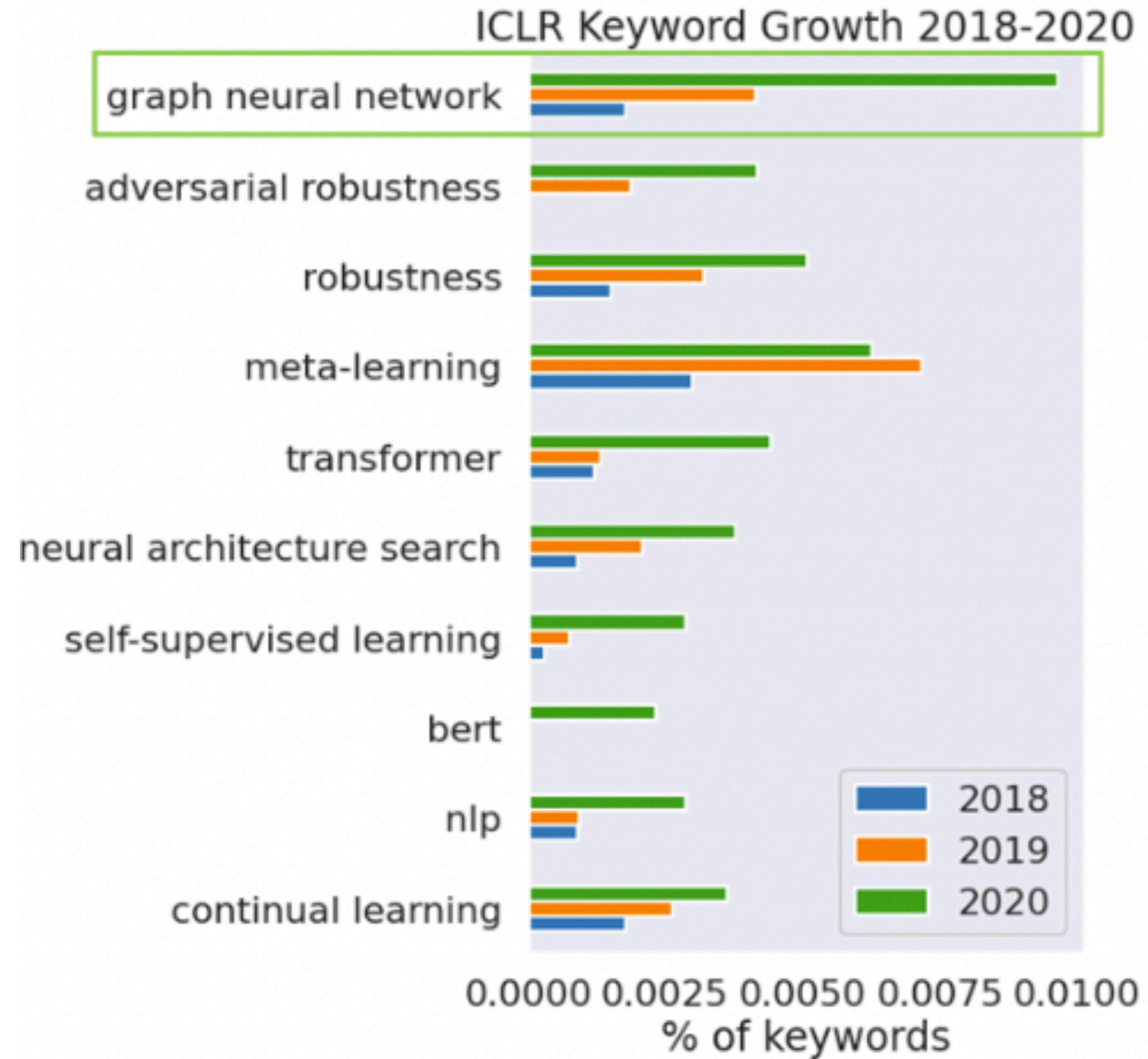
Image credit: The Conversation

# 1. Machine Learning with Graphs



- 그래프 데이터는 기존에 ML/DL에서 사용해오던 데이터와 다르다!
- Image, Text, Audio 등은 모두 격자 위에서 표현되어지는 (유클리드 공간 위에서 표현되어지는) 종류의 데이터다.
- 하지만 그래프는 그렇지 않다. 이것은 격자 위에서, 유클리드 공간 위에서 표현되어지는 그런 종류의 데이터가 아니다. 즉, 비유클리드 공간 위에서 표현되어질 수 있다.

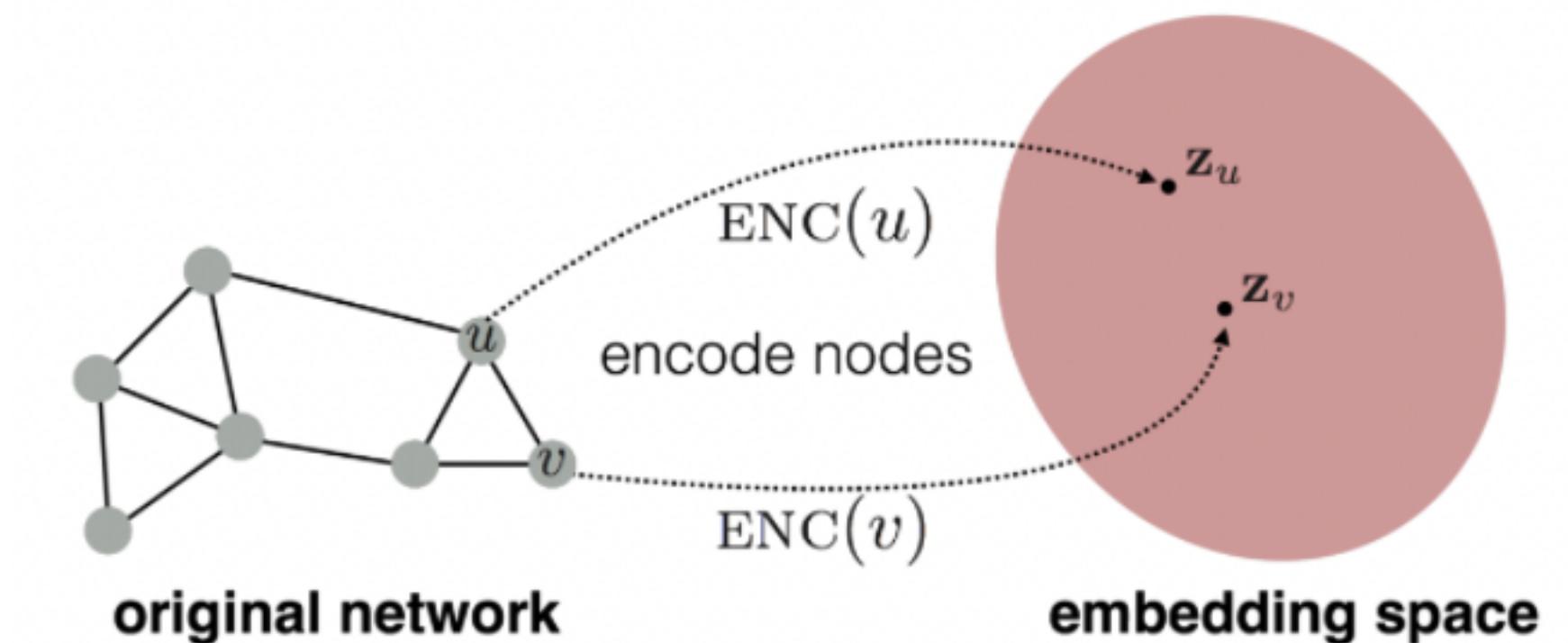
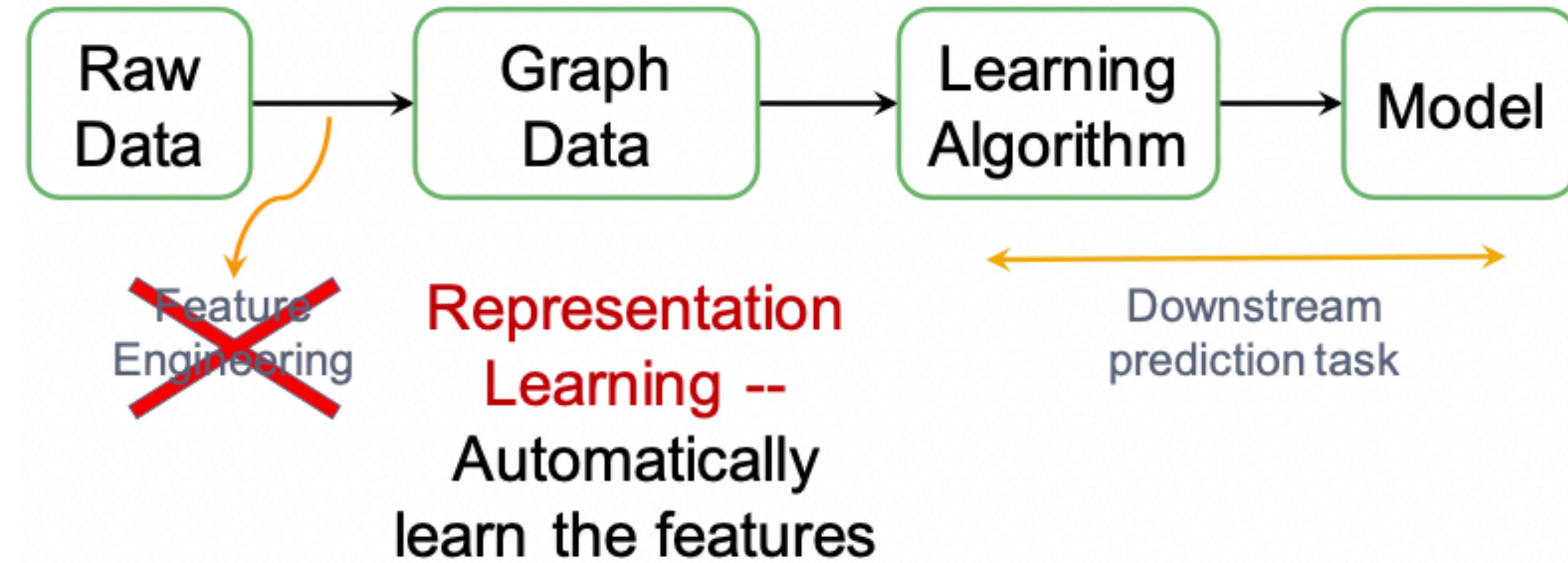
# 1. Machine Learning with Graphs



- GNN은 학계에서도 Trendy 하다 !

# 1. Machine Learning with Graphs

(Supervised) Machine Learning Lifecycle:  
This feature, that feature. **Every single time!**



- Graph 데이터로 Represenation Learning을 하려면 어떻게 Raw-graph data를 전처리해야 할까 ?\_?
- 즉, 어떻게 node, edge로 구성된 그래프를 function을 통해 embedding space로 사상시킬 수 있을까 ?

# 1. Machine Learning with Graphs

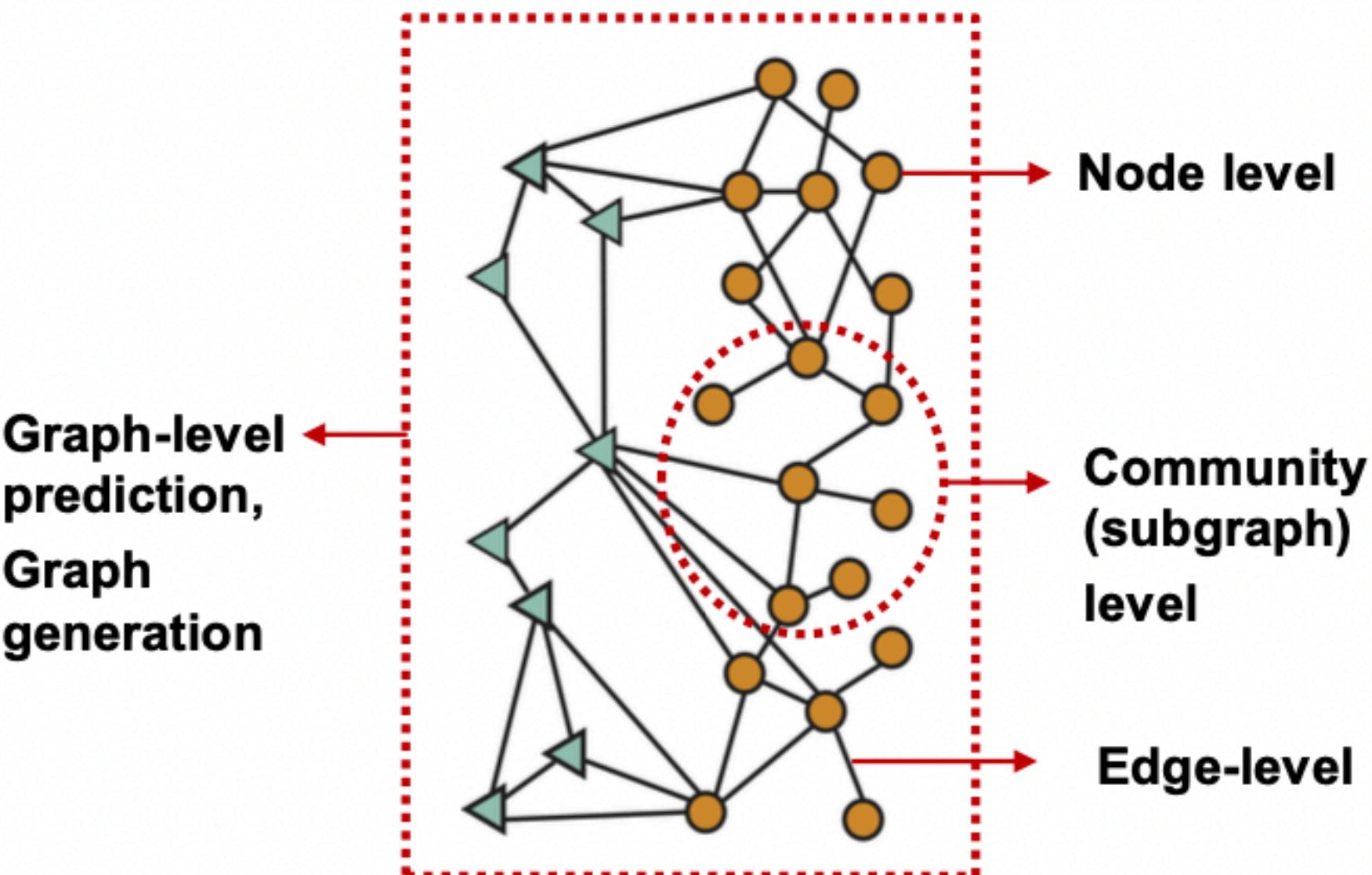
---

- Traditional methods: Graphlets, Graph Kernels
  - Methods for node embeddings: DeepWalk, Node2Vec
  - Graph Neural Networks: GCN, GraphSAGE, GAT,  
Theory of GNNs
  - Knowledge graphs and reasoning: TransE, BetaE
  - Deep generative models for graphs: GraphRNN
  - Applications to Biomedicine, Science, Industry
- 우리는 cs224w에서 위의 주제를 다룬다. 이 강좌에서는 크게 Graph를 다루는 전통 방법, GNN, GNN의 응용을 다룬다.

# 1. Machine Learning with Graphs

- **Node classification:** Predict a property of a node
  - Example: Categorize online users / items
- **Link prediction:** Predict whether there are missing links between two nodes
  - Example: Knowledge graph completion
- **Graph classification:** Categorize different graphs
  - Example: Molecule property prediction
- **Clustering:** Detect if nodes form a community
  - Example: Social circle detection
- **Other tasks:**
  - **Graph generation:** Drug discovery
  - **Graph evolution:** Physical simulation

Machine learning tasks on graphs can be categorized into four levels:

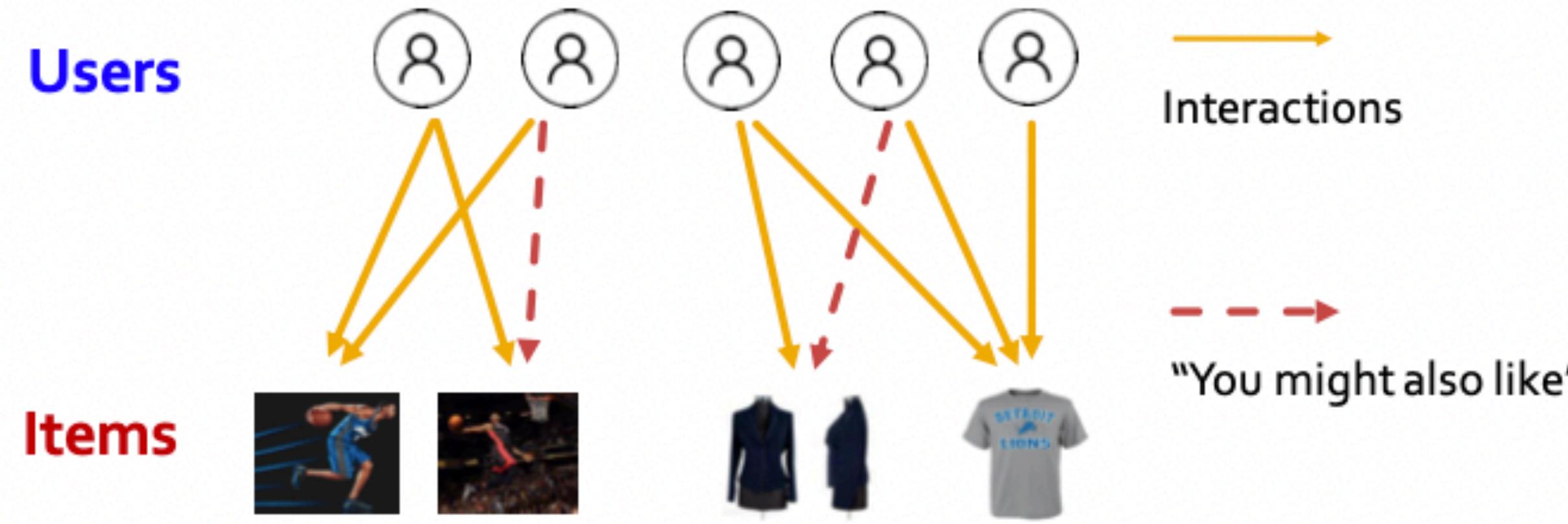


- GNN을 이용하여 수행할 수 있는 작업은 여러가지다. 이 분야도 기존 ML, DL과 마찬가지로 분류/예측/클러스터링/생성 등을 수행한다.

# 1. Machine Learning with Graphs

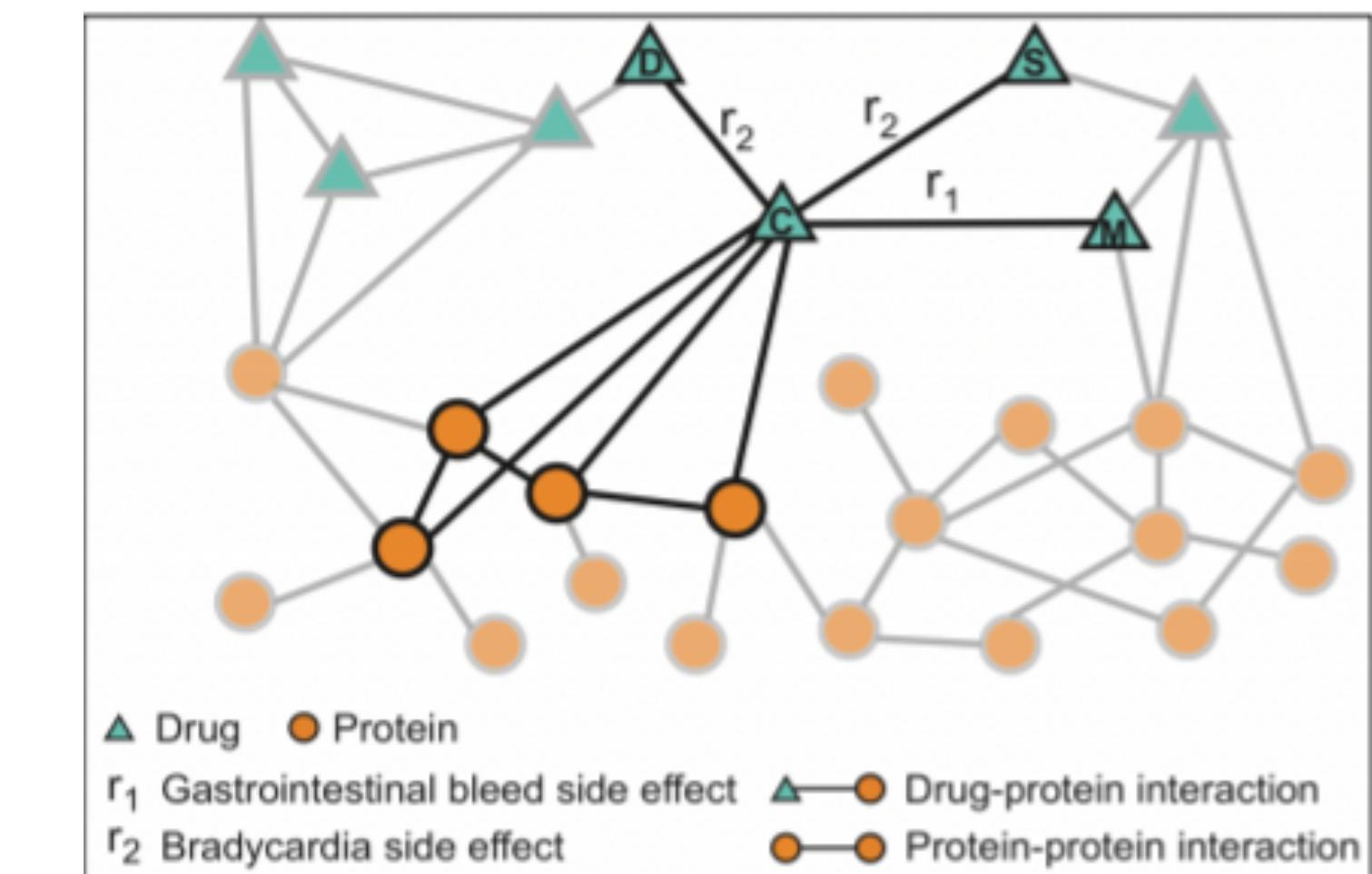
- Edge Prediction: 사용자에게 추천할 Item이 무엇인지 예측

■ **Goal: Recommend items users might like**



- Edge Prediction: 약의 부작용 예측

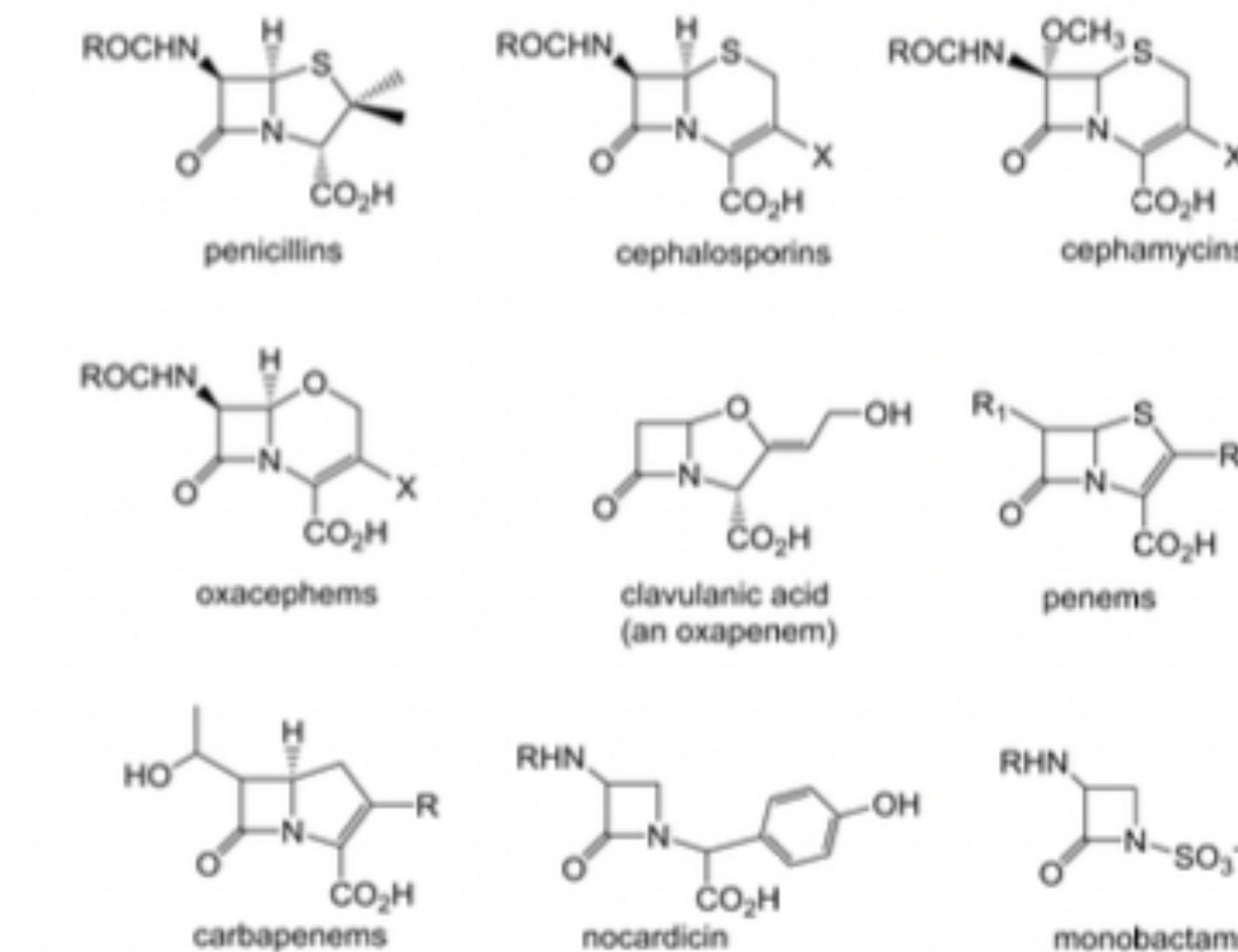
- 증증, 고령 환자는 한 번에 많은 양의 약을 복용한다. 하지만 이렇게 다양한 약을 한 번에 복용하는 것은 예상치 못한 부작용을 일으킬 수 있다. 하지만 이 많은 양의 약을 여러 case에 대해 부작용이 있는지 여부를 manually 실험하기는 어렵다.
- 어떤 약 간의 조합이 부작용을 일으키는지를 edge 예측으로 조사



# 1. Machine Learning with Graphs

- Graph Generation: 신약 개발

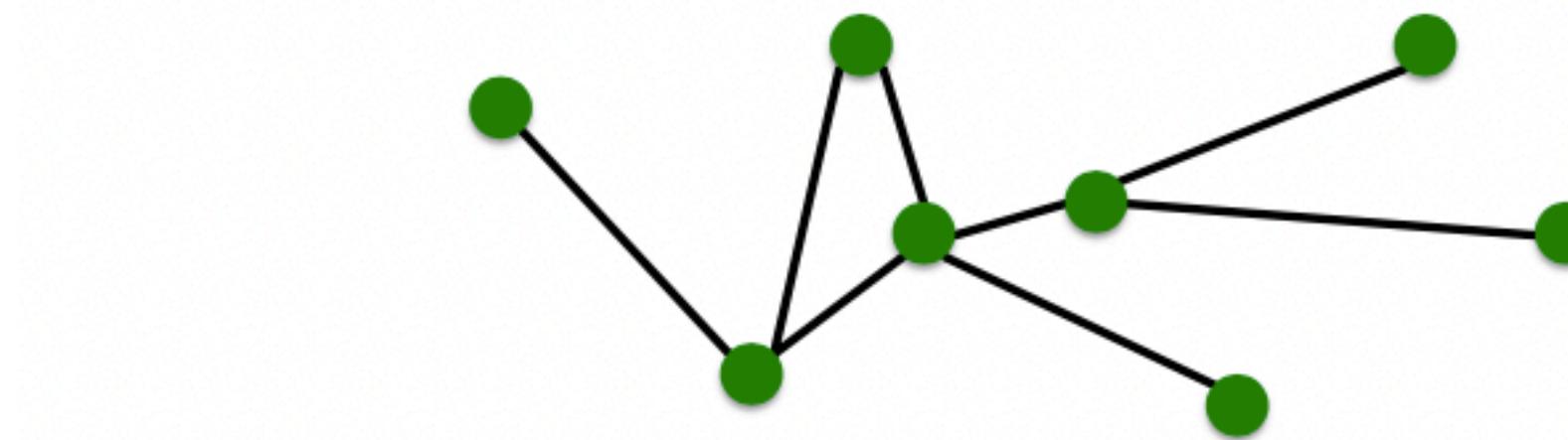
- 분자는 원자 간 결합으로 이루어져 있는데, 이것이 그래프 형태
- 원하는 효과를 가진 약, 현재 약의 보강을 위해 연구되고 있음



Konaklieva, Monika I. "Molecular targets of  $\beta$ -lactam-based antimicrobials: beyond the usual suspects." *Antibiotics* 3.2 (2014): 128-142.

# 1. Machine Learning with Graphs

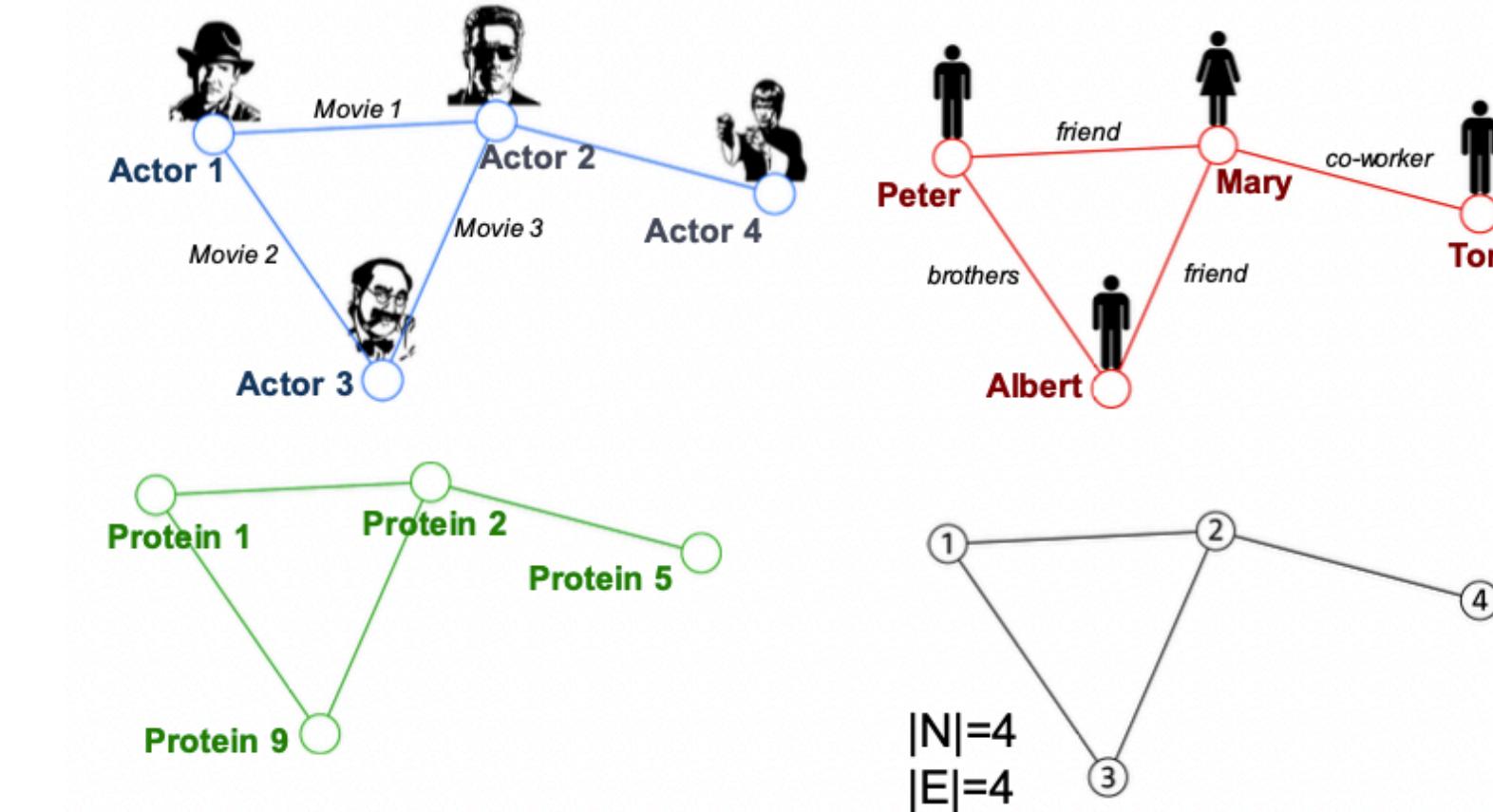
- 일반적으로 그래프는 이와 같이 node, link로 구성된다.



- **Objects:** nodes, vertices
- **Interactions:** links, edges
- **System:** network, graph

$N$   
 $E$   
 $G(N,E)$

- 실세계 그래프 데이터는 이 같이 이론적인 그래프로 표현되어질 수 있다.

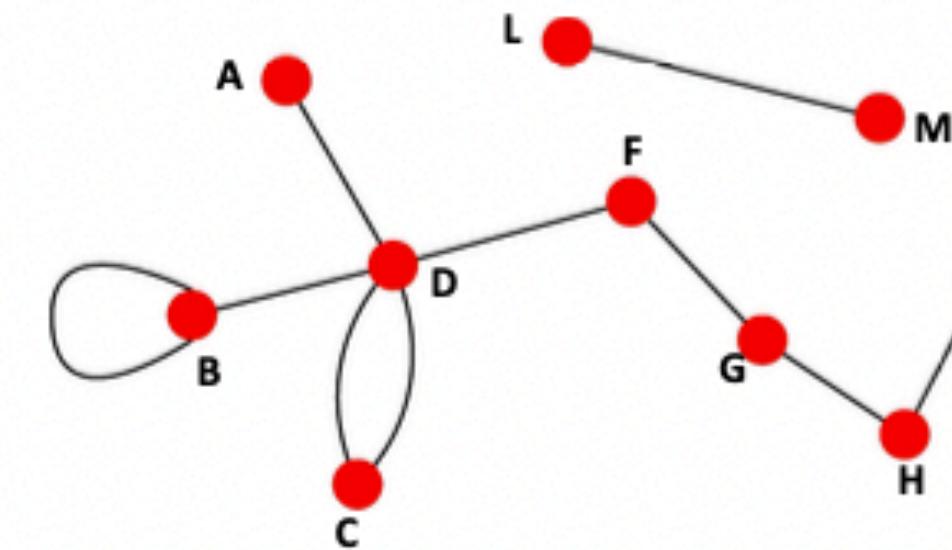


# 1. Machine Learning with Graphs

- 그래프는 여러 종류가 있다.
- 가장 단순한 형태는 무향 undirected 그래프이다. 엣지에 방향성이 없다.
- 반면에, 유향 directed 그래프도 있다. 이것은 엣지에 방향성이 있다.

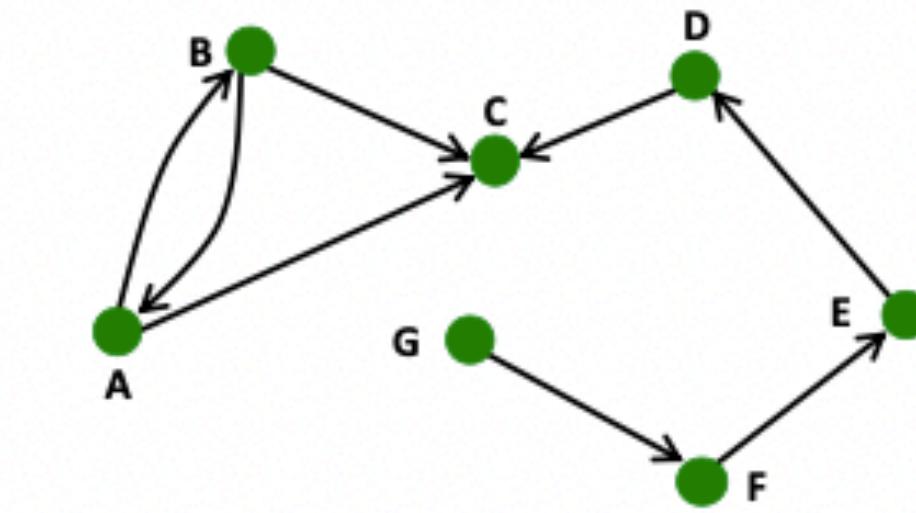
## Undirected

- Links: undirected  
(symmetrical, reciprocal)



## Directed

- Links: directed  
(arcs)



## Examples:

- Collaborations
- Friendship on Facebook

## Examples:

- Phone calls
- Following on Twitter

# 1. Machine Learning with Graphs

---

- **Heterogeneous Graph**

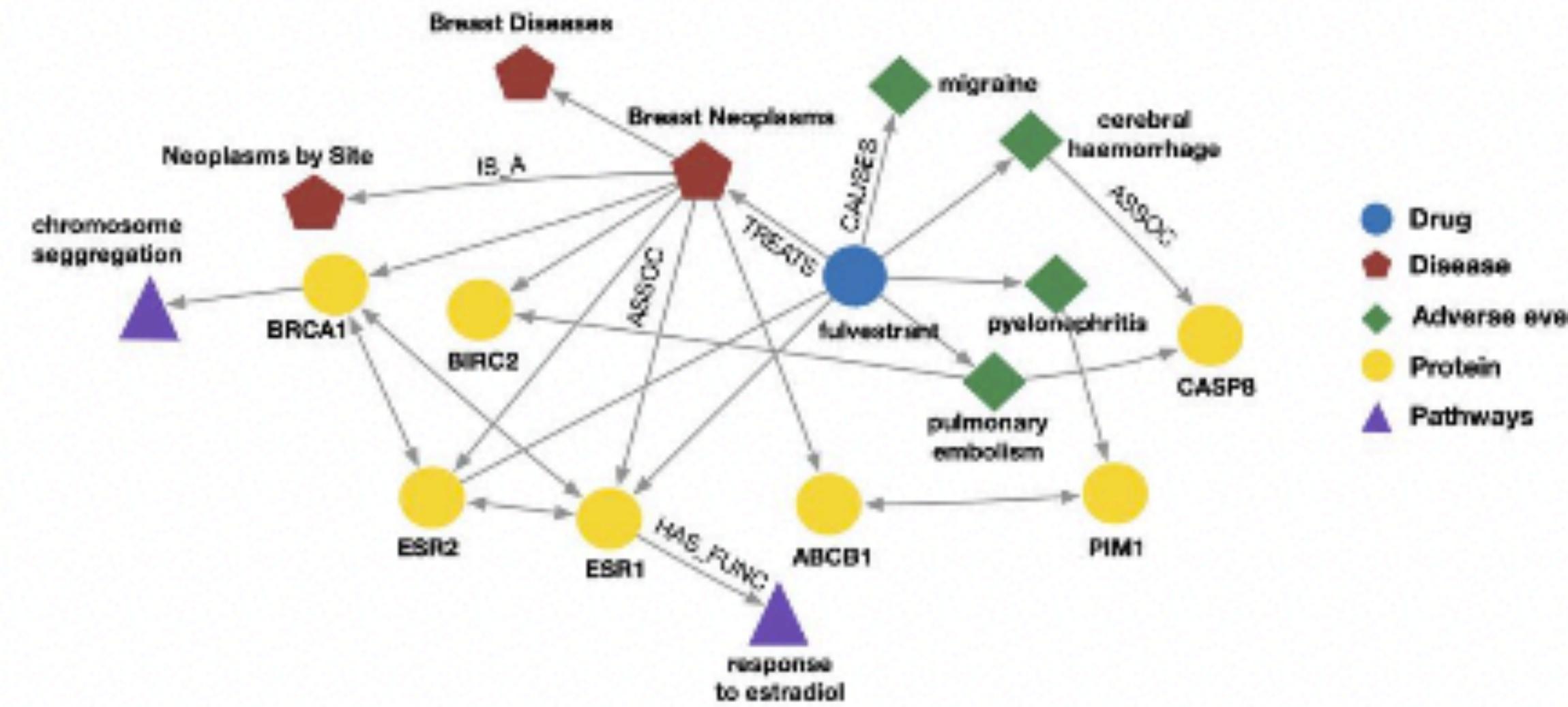
- **A heterogeneous graph is defined as**

$$G = (V, E, R, T)$$

- **Nodes with node types**  $v_i \in V$
- **Edges with relation types**  $(v_i, r, v_j) \in E$
- **Node type**  $T(v_i)$
- **Relation type**  $r \in R$

# 1. Machine Learning with Graphs

- **Heterogeneous Graph**



## Biomedical Knowledge Graphs

**Example node:** Migraine

**Example edge:** (fulvestrant, Treats, Breast Neoplasms)

**Example node type:** Protein

**Example edge type (relation):** Causes

## Academic Graphs

**Example node:** ICML

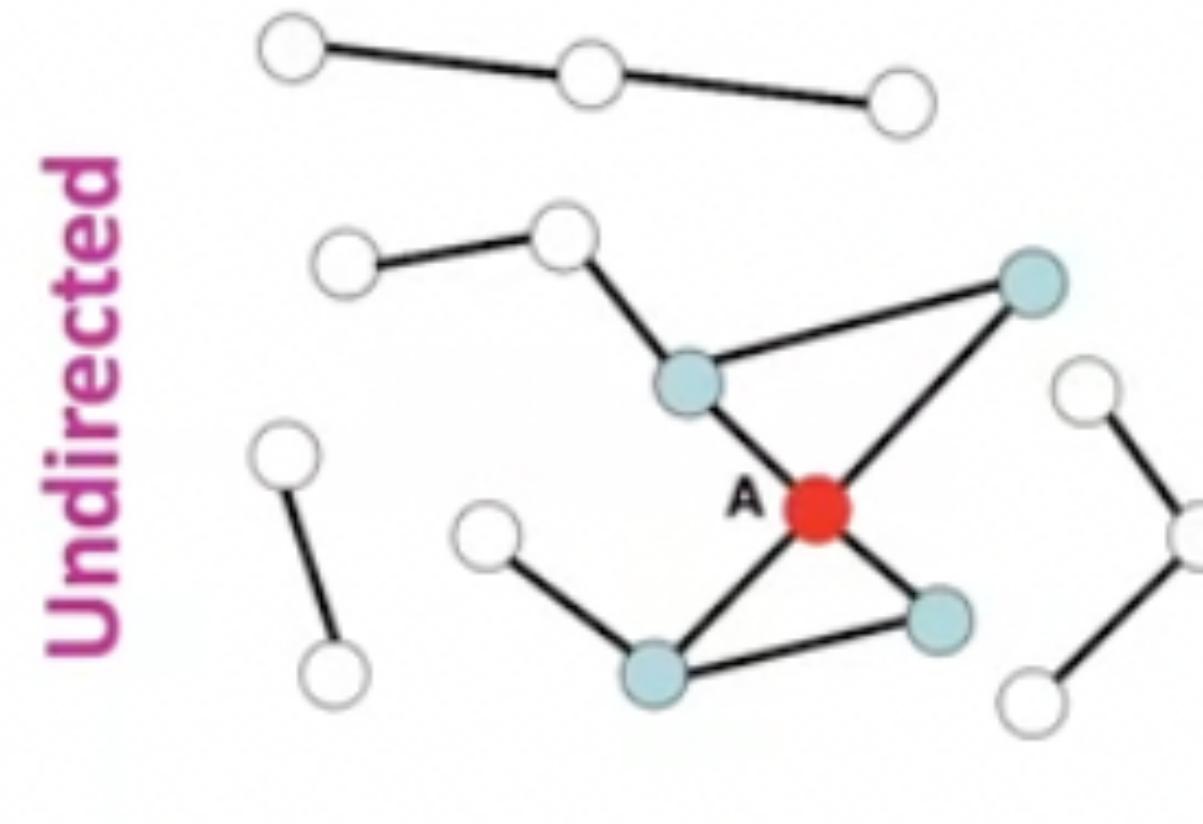
**Example edge:** (GraphSAGE, NeurIPS)

**Example node type:** Author

**Example edge type (relation):** pubYear

# 1. Machine Learning with Graphs

- **Node Degree**



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

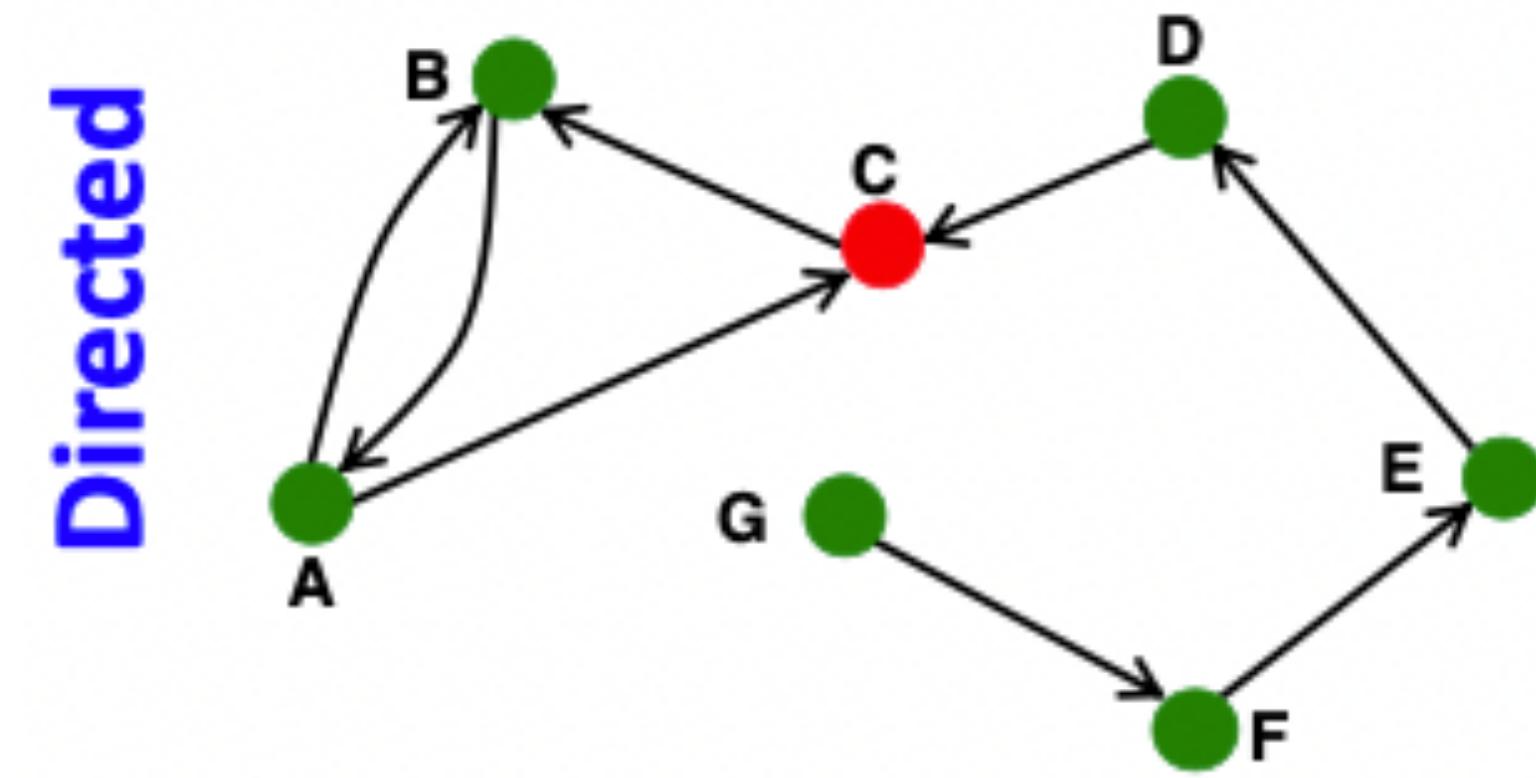
$$k_A = 4$$

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

- 노드 차수 평균 =  $2 * \# \text{엣지} / \# \text{노드}$

# 1. Machine Learning with Graphs

- Node Degree



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

$$\bar{k} = \frac{E}{N}$$

$$\bar{k}^{in} = \bar{k}^{out}$$

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

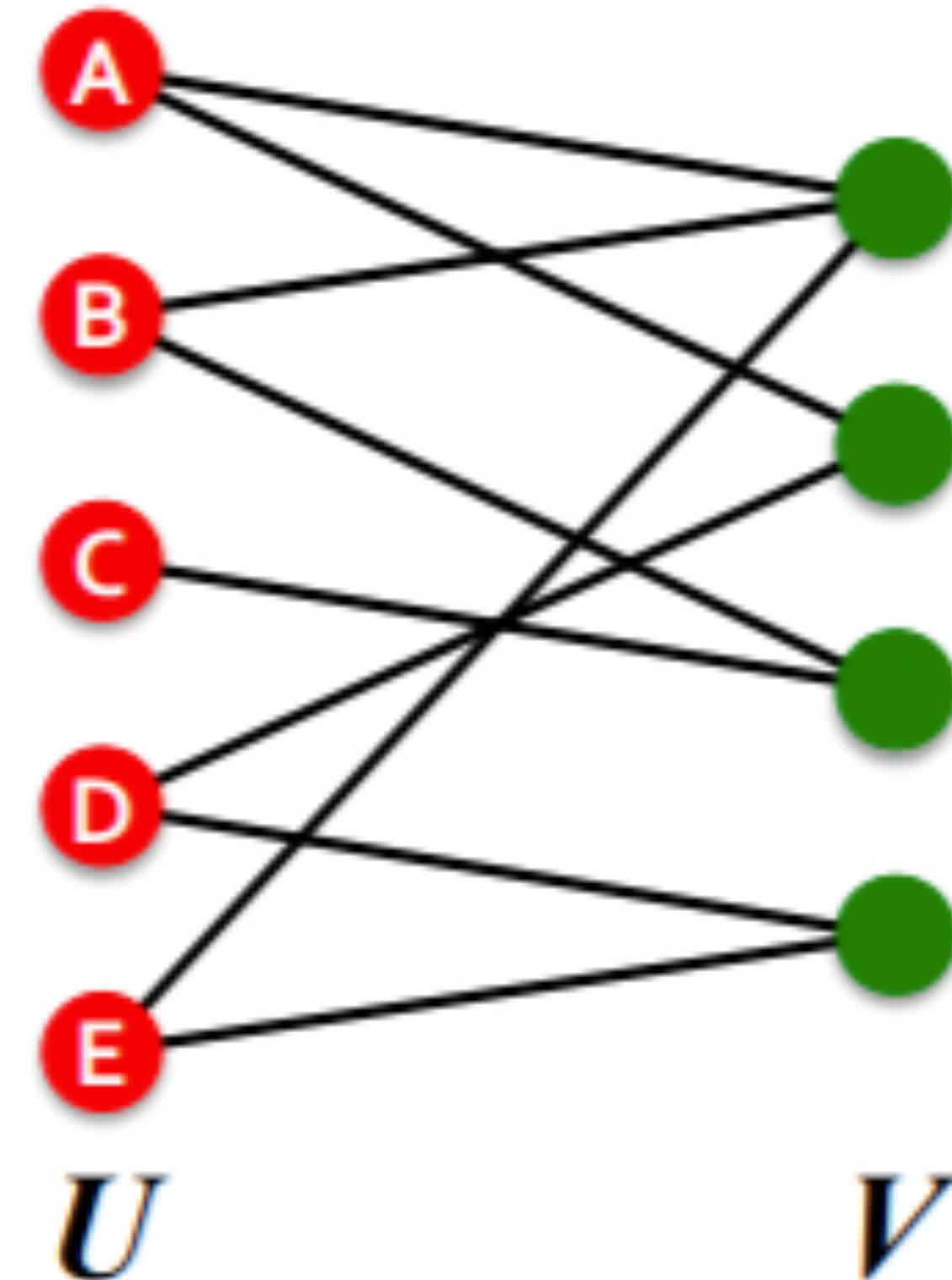
**Sink:** Node with  $k^{out} = 0$

# 1. Machine Learning with Graphs

## • Bipartite Graph

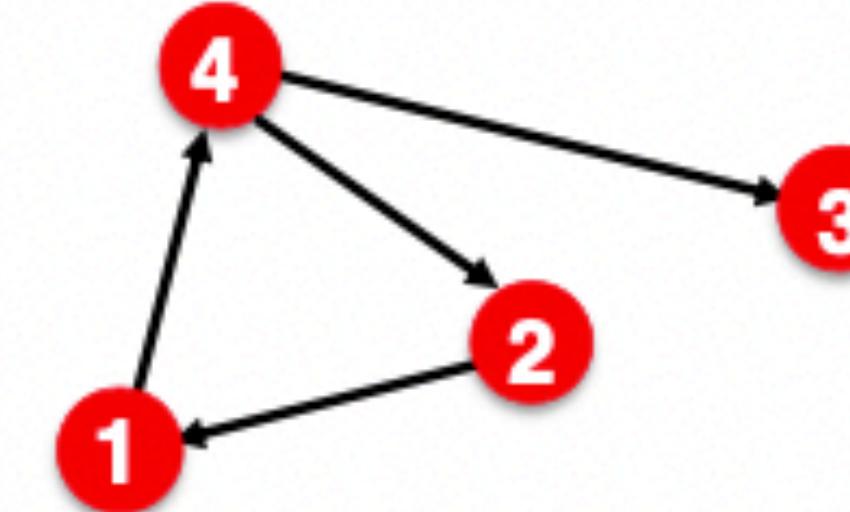
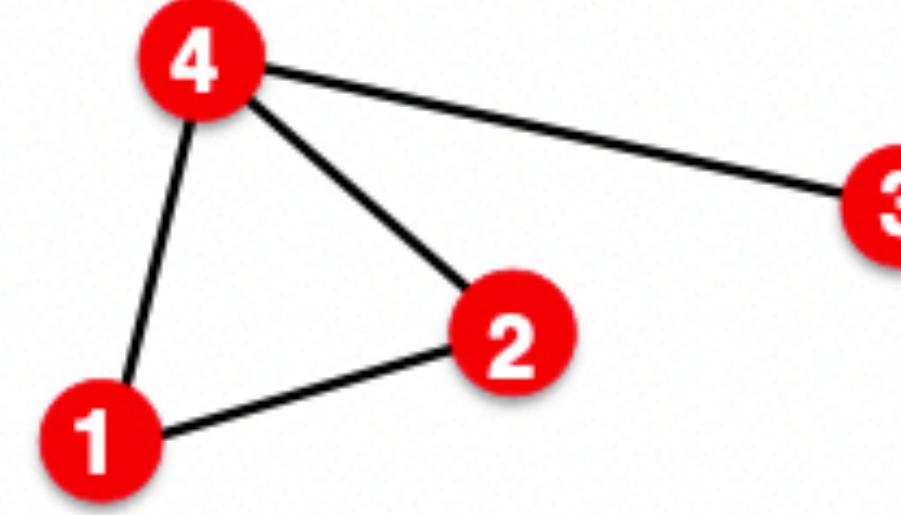
Bipartite Graph는 두 노드 집합으로 나눠질 수 있는 그래프  
두 노드 집합을  $U$ 와  $V$ 라고 할 때,  $U$ 와  $V$ 는 서로 독립적

- 저자 - 논문
- 배우 - 영화
- 사용자 - 영화 (리뷰)
- 레시피 - 재료



# 1. Machine Learning with Graphs

- **Adjacency Matrix**



$A_{ij} = 1$  if there is a link from node  $i$  to node  $j$   
 $A_{ij} = 0$  otherwise

- 무향 그래프:  $A_{ij} = A_{ji}$   
  >> symmetric

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

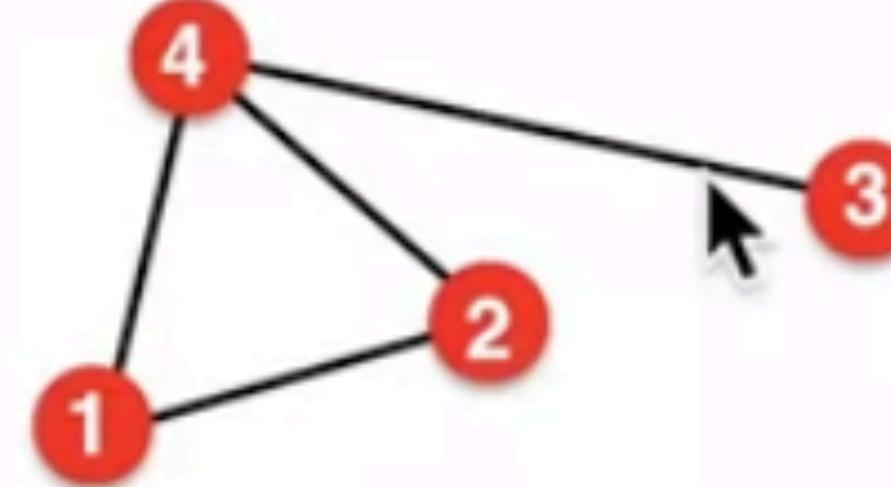
$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

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

# 1. Machine Learning with Graphs

## • Adjacency Matrix

Undirected



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

$$A_{ij} = A_{ji}$$

$$A_{ii} = 0$$

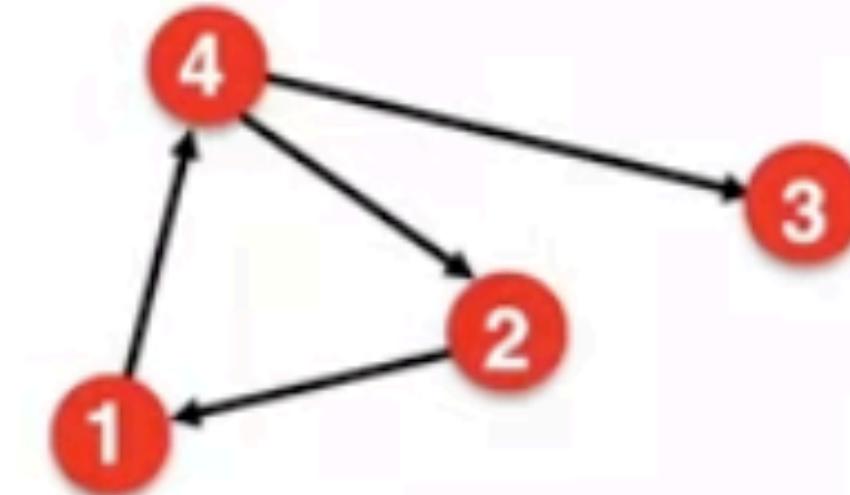
$$k_i = \sum_{j=1}^N A_{ij}$$

$$k_j = \sum_{i=1}^N A_{ij}$$

- Node Degree

$$L = \frac{1}{2} \sum_{i=1}^N k_i = \frac{1}{2} \sum_{i,j} A_{ij}$$

Directed



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

$$A_{ij} \neq A_{ji}$$

$$A_{ii} = 0$$

$$k_i^{out} = \sum_{j=1}^N A_{ij}$$

$$k_j^{in} = \sum_{i=1}^N A_{ij}$$

- Node Degree

- 열을 기준 >> out
- 행을 기준 >> in

$$L = \sum_{i=1}^N k_i^{in} = \sum_{j=1}^N k_j^{out} = \sum_{i,j} A_{ij}$$

# CH2: Traditional Methods for ML on Graphs

## 2. Traditional Methods for ML on Graphs

---

- **Node-Level Features**

- Node degree
- Node Cetrality
- Clustering Coefficient
- Graphlets

- **Link-Level Features**

- Distance-based Feature
- Local/Global Neighborhood Overlap

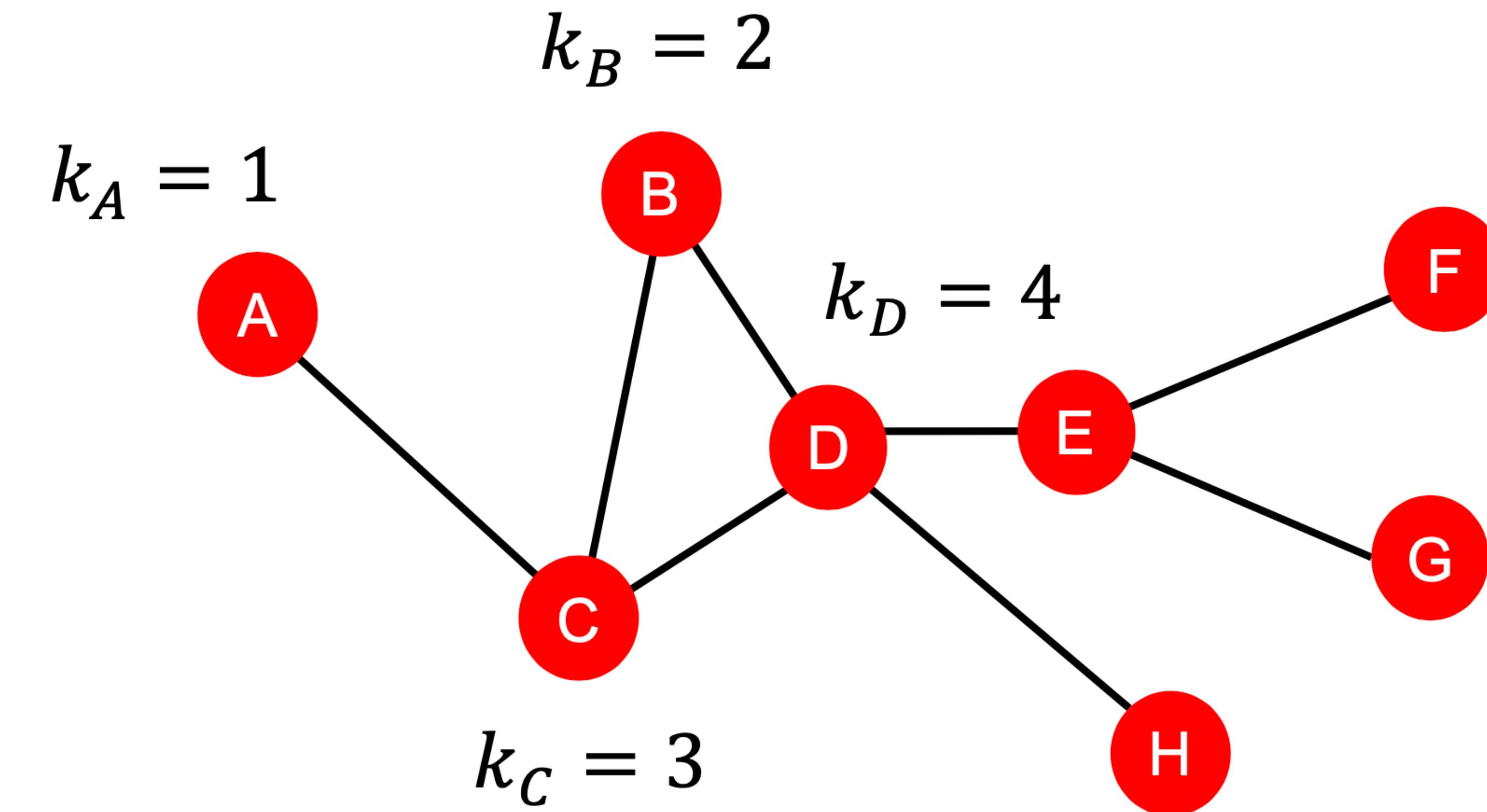
- **Graph-Level Features**

- Graphlet Kernel
- WL Kernel

## 2. Traditional Methods for ML on Graphs

---

- **Node-Level Features**
  - Node degree



## 2. Traditional Methods for Machine Learning in Graphs

- **Node Degree** 한계: 개별 노드의 중요성 정보를 포착할 수 없다. 이 방법은 모든 노드의 중요도는 동일함을 가정하고, 노드 간 차이는 오직 차수에 의해, 즉 몇 개의 엣지와 연결되었는지에만 의존할 뿐이다.

- **Node Centrality**

### (1) Eigenvector Centrality

**Idea:** 중요한 노드한 노드와 더 많이 연결된 노드가 중요하다. (유명인사와 많이 알고 있는 사람이 중요한 사람)

For a given graph  $G := (V, E)$  with  $|V|$  vertices let  $A = (a_{v,t})$  be the [adjacency matrix](#), i.e.  $a_{v,t} = 1$  if vertex  $v$  is linked to vertex  $t$ , and  $a_{v,t} = 0$  otherwise. The relative centrality score,  $x_v$ , of vertex  $v$  can be defined as:

$$x_v = \frac{1}{\lambda} \sum_{t \in M(v)} x_t = \frac{1}{\lambda} \sum_{t \in G} a_{v,t} x_t$$

where  $M(v)$  is a set of the neighbors of  $v$  and  $\lambda$  is a constant. With a small rearrangement this can be rewritten in vector notation as the [eigenvector equation](#)

- 여기서  $M(v)$  는 이웃 노드의 집합

#### Analysis:

- 노드  $v, t$ 가 연결된 경우,  $a_{v,t} = 1$ . O.W.  $= 0$
- Goal:  $x_v$  구하기, 즉 노드  $v$ 의 고유벡터 중심성 구하기.
  - $Ax = \lambda x$  에 대하여,  $A$ 는  $n, n$  정방행렬인 인접행렬이고,  $x$ 는  $A$ 의 고유 벡터,  $\lambda$ 는 고유치일 때,  $x_v = \frac{1}{\lambda} \sum_{t \in G} a_{v,t} x_t$  이다.
  - 여기서,  $x_t$ 는 다른 노드  $v_t$ 에 대한 고유 벡터이다.

## 2. Traditional Methods for Machine Learning in Graphs

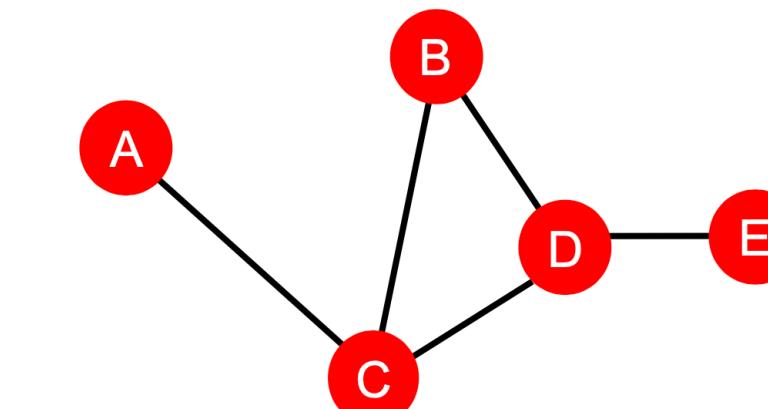
### (2) Betweenness Centrality

- 노드 간 최단 경로를 가지고 계산된다.
- **Intuition:** A도시 중요성을 구할 때, A를 제외한 도시에 사는 사람들이 다른 도시로 이동할 때 얼마나 A를 거쳐가는가? → A 노드의 중요성은 그것이 아닌 X,Y 노드에 대해 X-Y의 최단 경로에 A가 포함되는 비율로 볼 수 있다.

해당 노드가 다른 두 노드 사이의 최단 경로에 있는 경우의 수

$$c_u = \sum_{u \neq v_1 \neq v_2} \frac{v_1 \text{과 } v_2 \text{의 최단 경로에 } u \text{가 포함되는 횟수}}{v_1 \text{과 } v_2 \text{의 최단 경로의 수}}$$

#### ■ Example:



$$c_A = c_B = c_E = 0$$

$$c_C = 3$$

(A-C-B, A-C-D, A-C-D-E)

$$c_D = 3$$

(A-C-D-E, B-D-E, C-D-E)

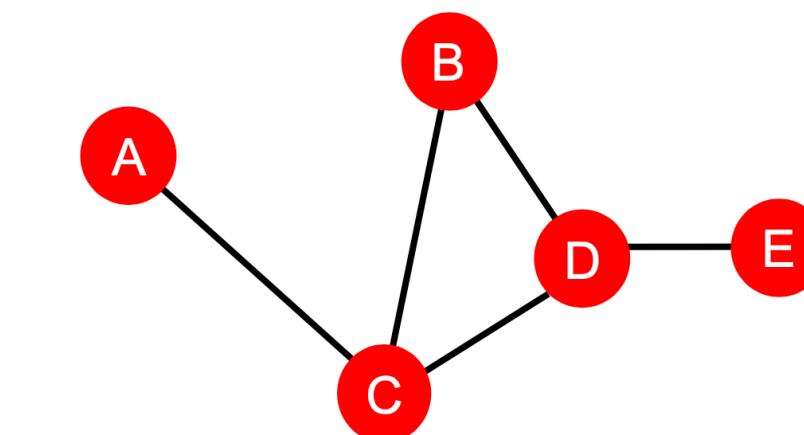
### (3) Closeness Centrality

- 중요한 노드일수록 다른 노드까지 도달하는 경로가 짧을 것이라는 가정

해당 노드와 다른 노드 사이의 최단 경로의 평균값

$$c_u = \frac{1}{\sum_{v \neq u} u \text{와 } v \text{ 사이의 최단 경로 길이}}$$

#### ■ Example:



$$c_A = 1/(2 + 1 + 2 + 3) = 1/8$$

(A-C-B, A-C, A-C-D, A-C-D-E)

$$c_D = 1/(2 + 1 + 1 + 1) = 1/5$$

(D-C-A, D-B, D-C, D-E)

## 2. Traditional Methods for Machine Learning in Graphs

### (3) Clustering Coefficient (결집계수)

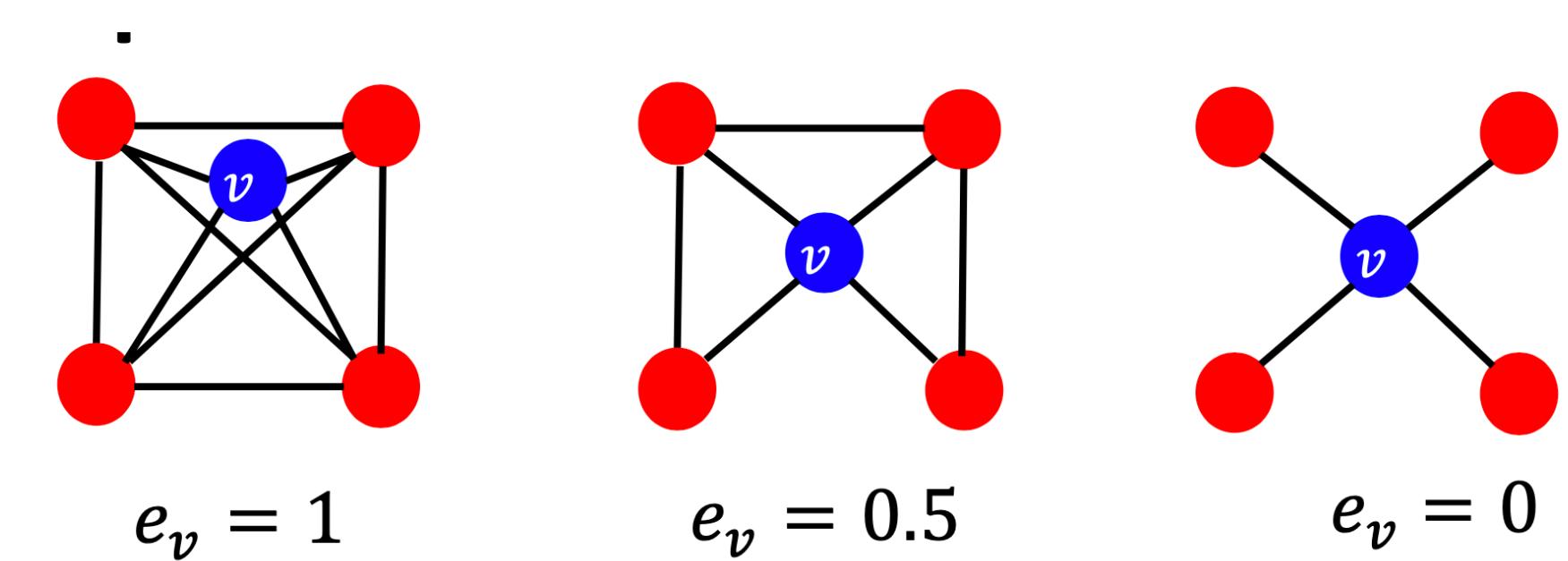
- 결집 계수 산식

한 노드의 이웃 노드들이 얼마나 긴밀하게 연결되어 있는가를 나타낼 수 있는 척도

$$C_i = \frac{2e_i}{k_i(k_i - 1)}$$

$k_i$ : as the number of vertices

where  $e_i$  is the number of edges between the neighbors of node  $i$



### (4) Graphlet

Graphlets can be defined as small, connected, non-isomorphic, induced subgraphs of a large network.

We use graphlets to obtain a node-level subgraph metric

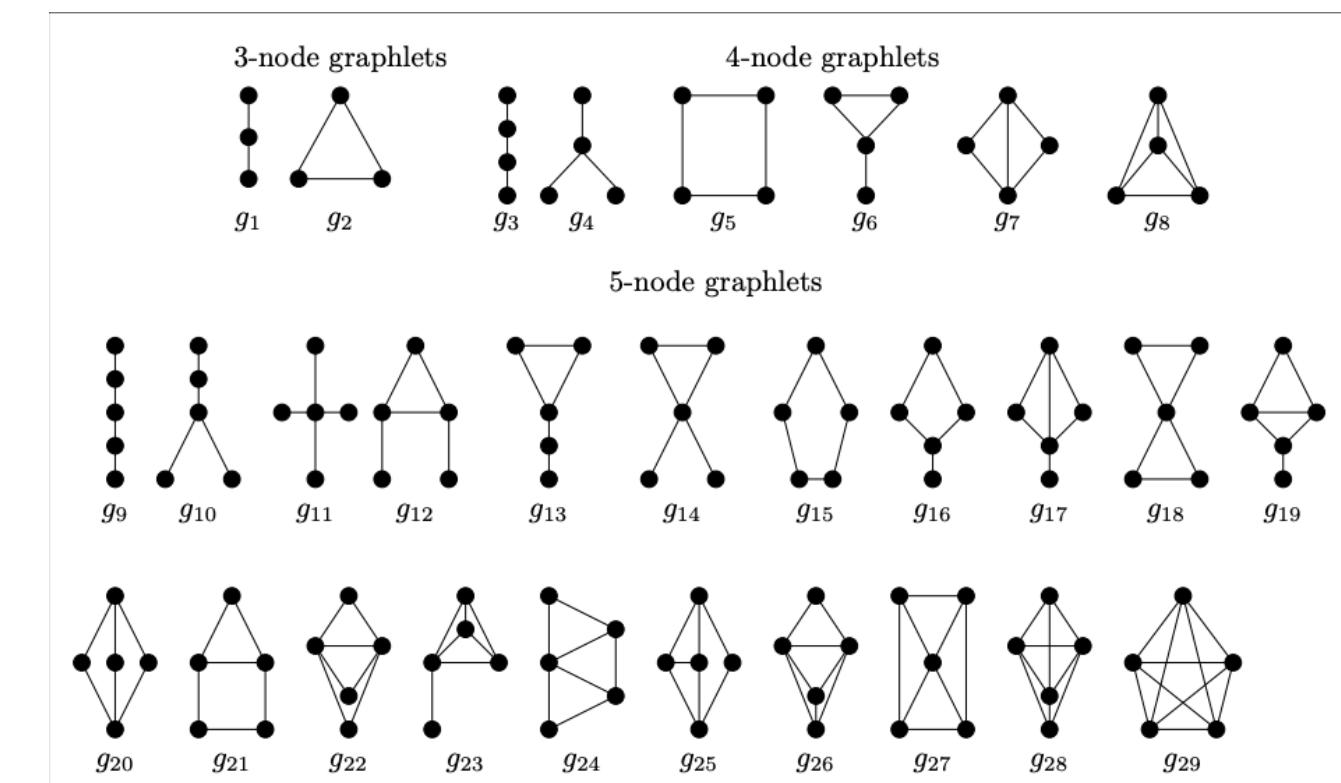


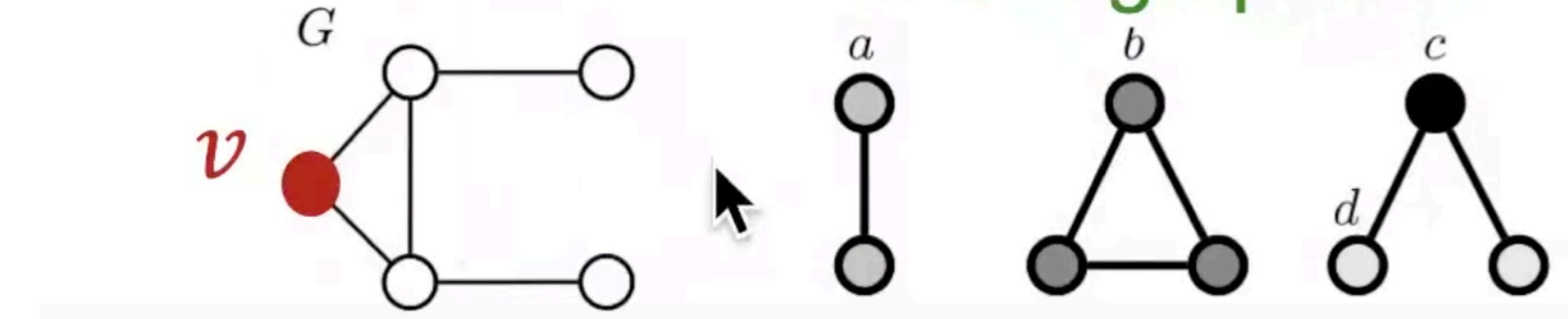
Figure 1.1.: All 3,4,5-node graphlets

## 2. Traditional Methods for Machine Learning in Graphs

### (5) Graplet Degree Vector (GDV)

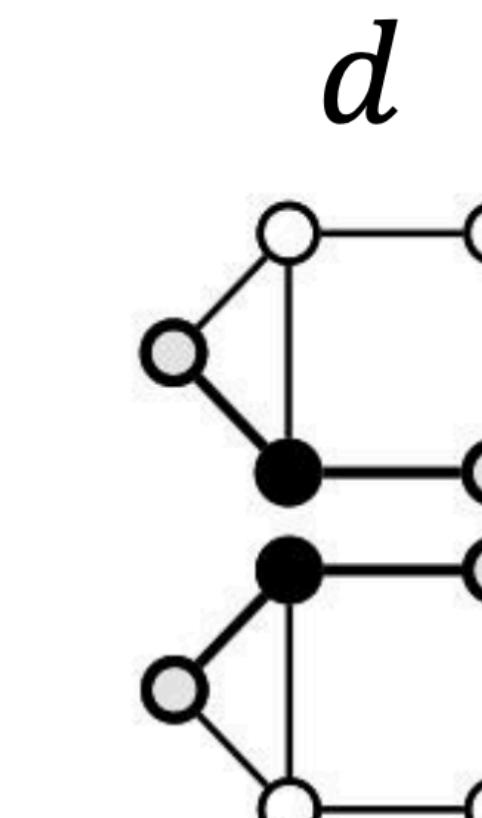
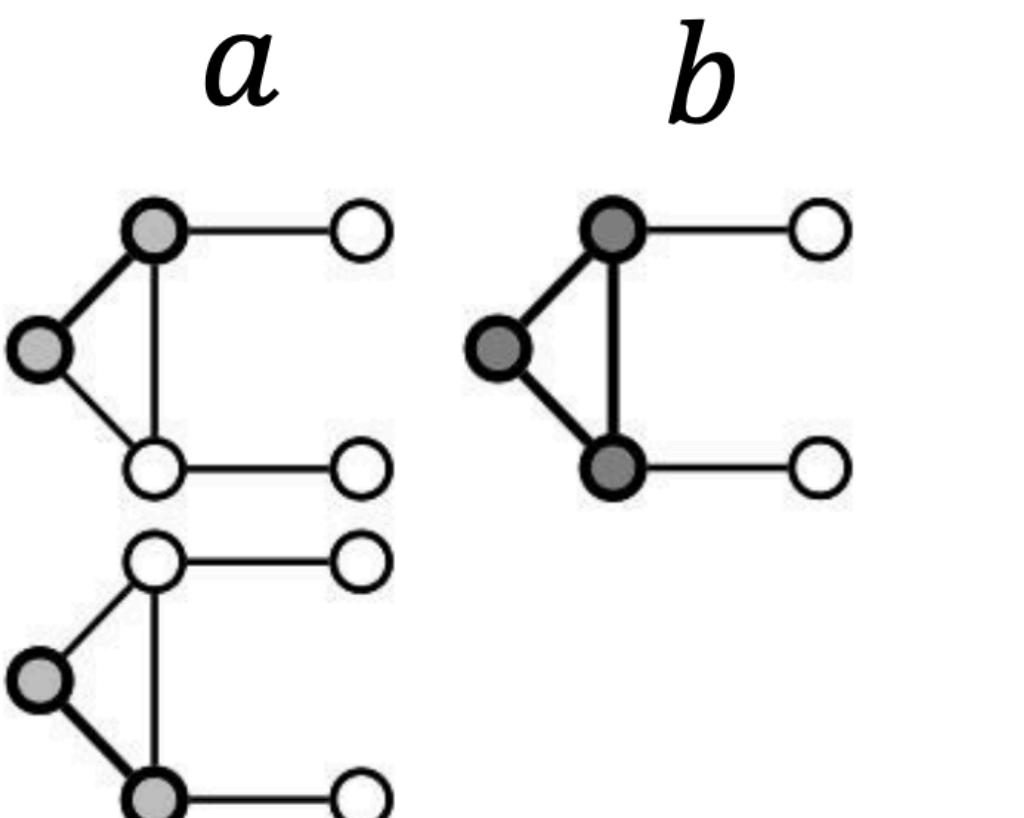
- **GDV** counts #graphlet that a node touch or participates in // unclear
- GDV is a count vector of graphlets rooted at that given node
- 각 orbit position에서 node의 빈도에 대한 벡터를 의미

#### ■ Example:



A set of graphlet on two or three nodes  
➤ 3 가지 type을 고려할 수 있다

#### Graphlet instances of node $u$ :



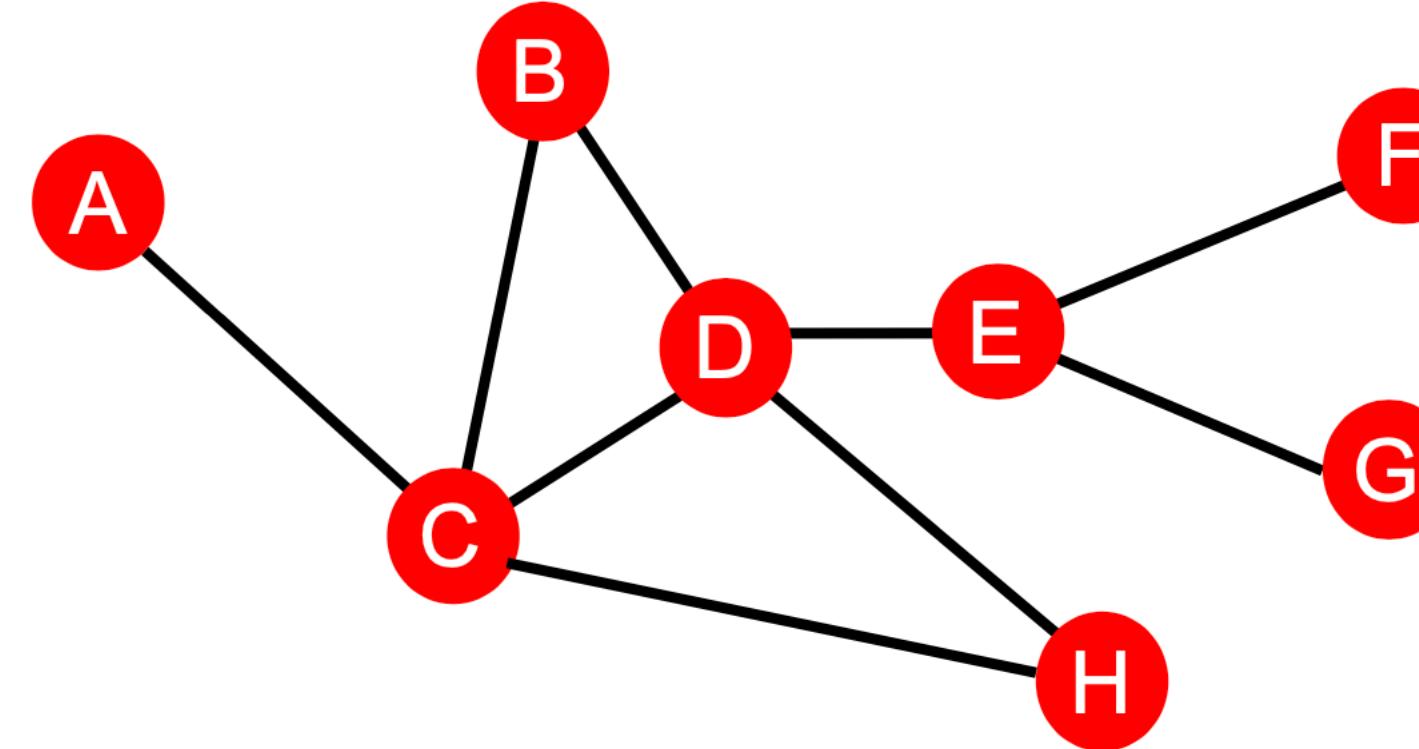
GDV of node  $u$ :  
 $a, b, c, d$   
 $[2, 1, 0, 2]$

## 2. Traditional Methods for Machine Learning in Graphs

### (6) Link-Level Features: Distance based feature

거리 기반 특징 중 하나인 Shortest path 거리를 알아보자. 이것은 가령, BH 간 연결 path 중 shortest 한 2를 특징으로서 추출하는 방법이다.

#### ■ Example:



$$S_{BH} = S_{BE} = S_{AB} = 2$$
$$S_{BG} = S_{BF} = 3$$

Limitation: 하지만 이 방법은 노드 간 연결 강도나 이웃 중첩 정도를 포착하지 못하는 한계가 있다.

## 2. Traditional Methods for Machine Learning in Graphs

### (7) Link-Level Features: Local neighborhood overlap

한쌍의 노드 간 공통 이웃의 개수는 몇 개인가? → local neighborhood overlap 방식으로 특징 추출

- **Common neighbors:**  $|N(v_1) \cap N(v_2)|$  가장 단순한 방법

- Example:  $|N(A) \cap N(B)| = |\{C\}| = 1$

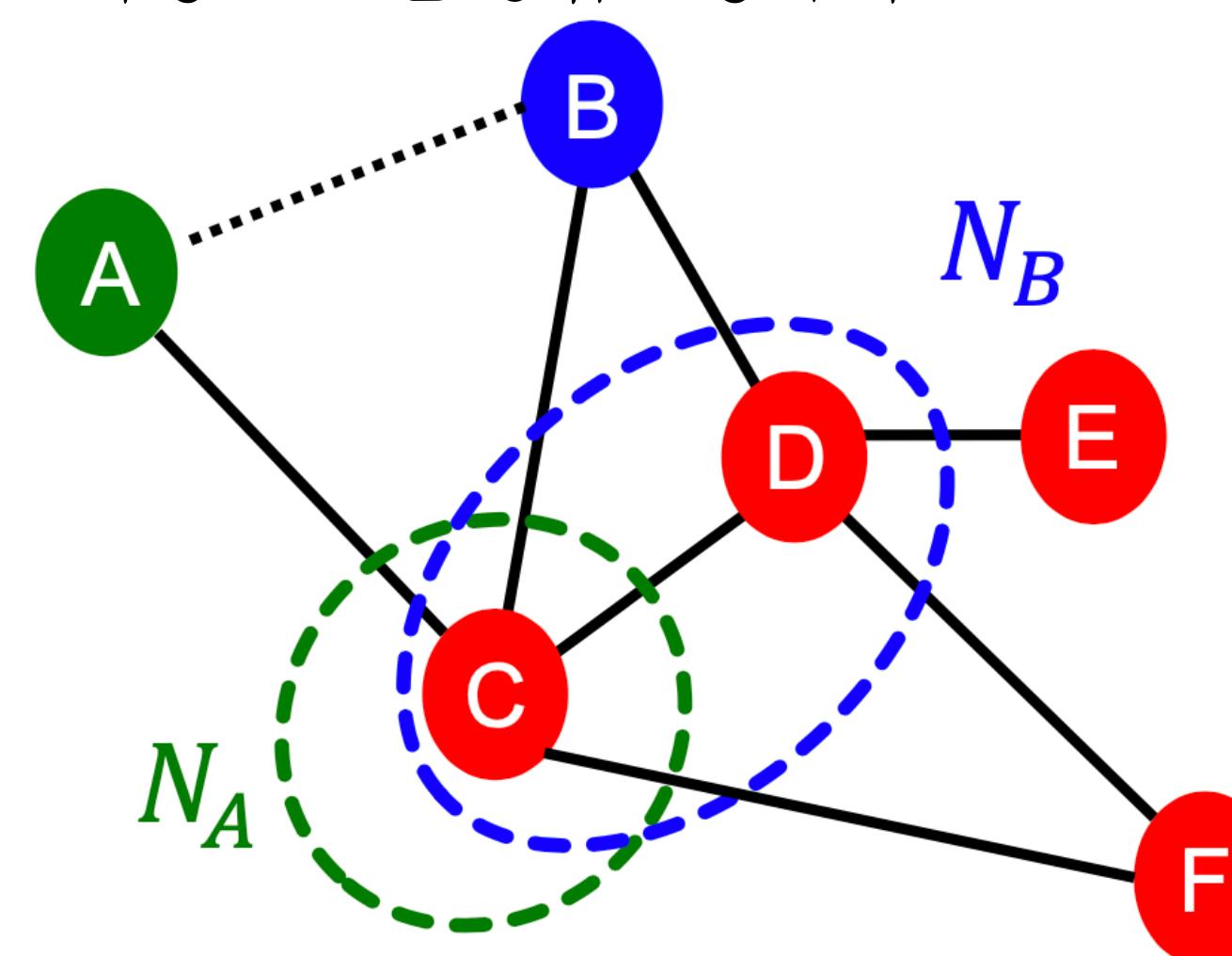
- **Jaccard's coefficient:**  $\frac{|N(v_1) \cap N(v_2)|}{|N(v_1) \cup N(v_2)|}$  가장 단순한 방법 + 정규화

- Example:  $\frac{|N(A) \cap N(B)|}{|N(A) \cup N(B)|} = \frac{|\{C\}|}{|\{A, B, C, D\}|} = \frac{1}{4}$

- **Adamic-Adar index:**

$$\sum_{u \in N(v_1) \cap N(v_2)} \frac{1}{\log(k_u)}$$

- Example:  $\frac{1}{\log(k_C)} = \frac{1}{\log 4}$



## 2. Traditional Methods for Machine Learning in Graphs

### (8) Link-Level Features: Adamic-Adar index in Local neighborhood overlap

- Adamic-Adar index 방식은 실무에서 괜찮은 방법이고, 이것은 공유 이웃을 기반으로 노드의 근접성을 계산하는 방법이다.

$$\sum_{u \in N(v_1) \cap N(v_2)} \frac{1}{\log(k_u)}$$

- Example:  $\frac{1}{\log(k_C)} = \frac{1}{\log 4}$

- Analysis:

- (1) 임의의 두 노드  $v_1, v_2$ 에 대하여, 그것의 이웃 집합  $N(v_1), N(v_2)$ 가 있을 때,  $u \in N(v_1) \cap N(v_2)$  는 이 조건에 포함되는 노드  $u$ 이다. (여러 개 있을 수 있음)
- (2) 위 식의  $k_u$ 는 위 (1) 조건에 만족하는  $u$  노드의 인접 노드(이웃)의 개수이다.
- (3) 두 노드  $v_1, v_2$ 의 공통 이웃 개수  $\uparrow \rightarrow$  Summation  $\uparrow \Rightarrow$  유사도 증가
- (4)  $u$ 의 이웃 수  $\downarrow \rightarrow$  분모 크기  $\downarrow \Rightarrow$  유사도 증가

- Limitation

- 두 노드의 공통 이웃이 없는 경우, 이 측정 지표는 항상 0일 것. 하지만 이것은 unacceptable. 두 노드는 미래 시점에 연결될 수도 있음

## 2. Traditional Methods for Machine Learning in Graphs

### (9) Link-Level Features: Global neighborhood overlap

- **Motivation:** 앞에서 보았던 Local Neighborhood overlap 방법의 한계는 그래프 내 nodes 간 one-hop 만을 탐색하기 때문. 그러면, 임의의 노드의 two-hops 이상 거리에 있는 이웃 노드를 탐색하면 어떨까?

#### Katz Index

- 주어진 a pair of nodes 사이의 모든 가능한 path의 길이의 개수를 count ↪ 주어진 그래프 내의 edge를 모두 활용

$$S_{v_1 v_2} = \sum_{l=1}^{\infty} \boxed{\beta^l} \boxed{A_{v_1 v_2}^l}$$

#paths of length  $l$   
between  $v_1$  and  $v_2$

$0 < \beta < 1$ : discount factor

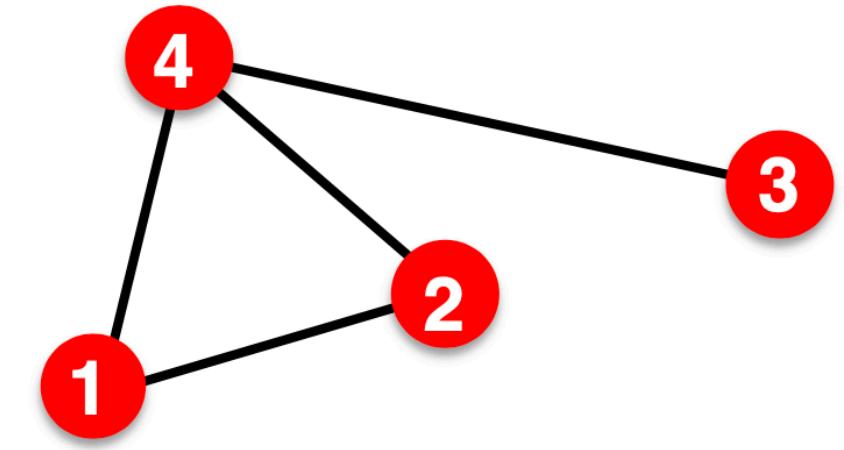
## 2. Traditional Methods for Machine Learning in Graphs

### (9) Link-Level Features: Global neighborhood overlap

#### Katz Index

- 주어진 a pair of nodes 사이의 모든 가능한 path의 길이의 개수를 count ↪ 주어진 그래프 내의 edge를 모두 활용
- 그런데, 이 **Katz Index**를 어떻게 계산할 수 있을까? **Goal: Show you  $P^1 = A^1$**
- Adjacency Matrix** 방법을 상기해보자. 임의의 두 노드  $u, v$ 에 대하여,  $u \in N(v)$  이면,  $A_{uv} = 1$  이었다.  
동일하게, 임의의 두 노드  $u, v$  사이의 path 길이를  $K$ 라고 할 때, 그  $K$  길이를 갖는 path의 개수를  $P_{uv}^{(K)}$ 라고 해보자.

오른쪽 예시와 같이, 모든 노드 간 path에서 길이( $K$ )가 1 일 때, path에 대한 adj mat는 그 아래와 같이 표현될 수 있다.  
이것은 이전에 보았던 adj mat와 동일한 것이다:  $(P^k = A^k) = (P^1 = A^1)$



$$P_{12}^{(1)} = A_{12}$$

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

## 2. Traditional Methods for Machine Learning in Graphs

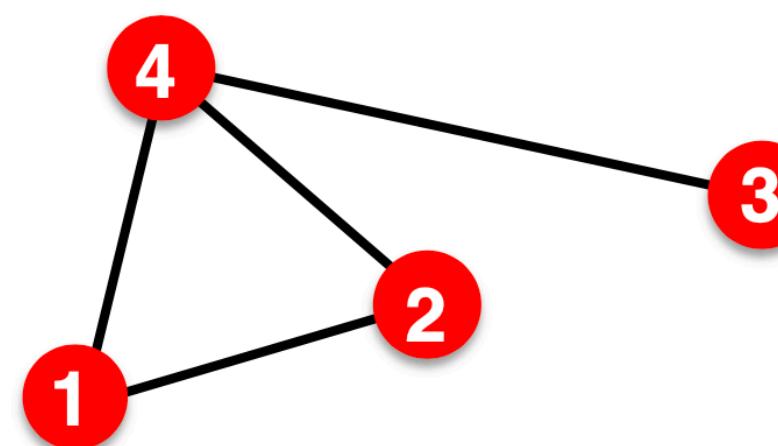
### (9) Link-Level Features: Global neighborhood overlap

#### Katz Index

- 그러면,  $K=2$  인 경우는 어떻게 구할까? (Goal: Show you  $P^2 = A^2$ )

step1)  $K=1$ 인 경우의 경로 행렬( $K$ )을 구한다. 그런데  $K=1$ 인 경우의 경로 행렬은  $K=1$ 인 경우의 인접 행렬과 동일하다.  $K^1 = A^1$

step2)  $K=2$ 인 경우를 구하기 위해,  $A_{uv}^2 = A_{ui}^1 * A_{iv}^1$  를 계산한다.



$$\begin{aligned} P_{ui}^1 * P_{iv}^1 &= A_{ui} * P_{iv}^1 \\ A_{ui} * P_{iv}^1 &= \sum_i A_{ui} * P_{iv}^1 \\ \sum_i A_{ui} * P_{iv}^1 &= \sum_i A_{ui} * A_{iv} \\ \sum_i A_{ui} * A_{iv} &= A_{uv}^2 \end{aligned}$$

Node 1's neighbors      #walks of length 1 between  
 Node 1's neighbors and Node 2       $P_{12}^{(2)} = A_{12}^2$

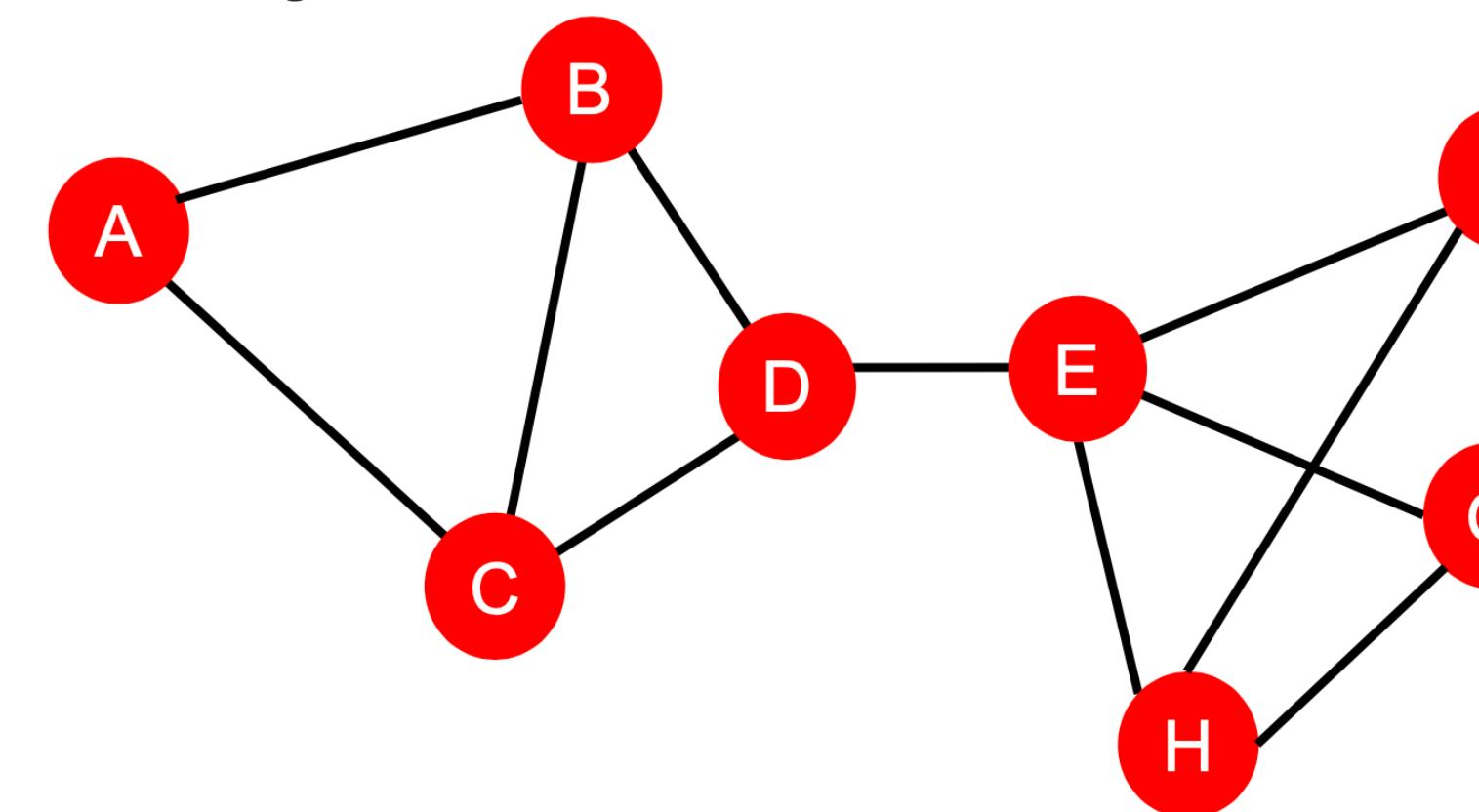
$$A^2 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 3 \end{pmatrix}$$

**Power of adjacency**

## 2. Traditional Methods for Machine Learning in Graphs

### (10) Graph-Level Features and Graph Kernels

- **Goal:** We want features that characterize the structure of an entire graph.
- **For example:**



- 전체 그래프의 구조의 특징을 얻는 방법을 알아보자. 아래는 1 개의 엣지로 이어진 2 개의 서브 그래프다.

## 2. Traditional Methods for Machine Learning in Graphs

### (10) Graph-Level Features and Graph Kernels : Graph Kernel

- **Graph Kernel?** Measure similarity btn two graphs

Kernel methods refer to machine learning algorithms that learn by comparing pairs of data points using particular similarity measures

- **Graph Kernel의 속성**

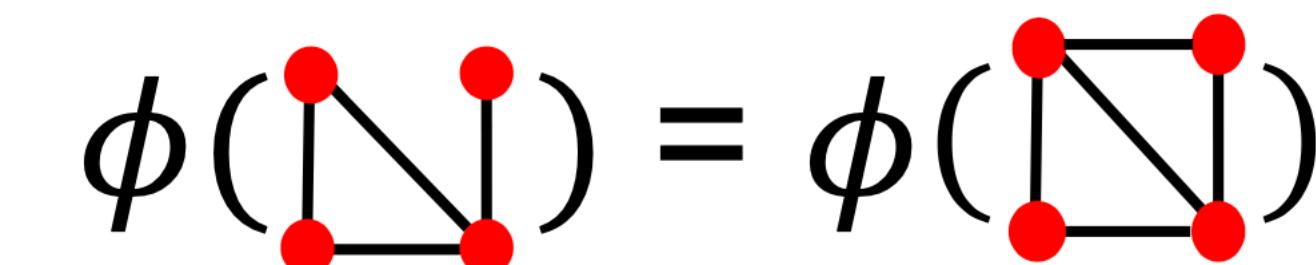
- 커널 행렬  $K = (k(G, G'))_{G, G'}$  은 항상 양의 고유값을 갖는다. (semidefinite) → 대칭행렬 // K는 커널 행렬, k는 실수인 커널 값
- $\exists \phi() s.t K(G, G') = \phi(G)^T \phi(G')$

- $\phi()$  is a feature representation
- first graph, second graph 의 dot product

- $\phi()??$

어떻게 그래프 특징 벡터  $\phi(G)$ 를 만드는가

- **Bag-Of-\* for a graph:** NLP에서의 BoW 방법은 단순히 문서 내 words의 특징을 세는 기법을 사용했다.
- 그래프 맥락에서 \* 부분을 채워 넣음으로써, 다양한 방식으로 그래프 커널 함수를 만들 수 있다.
- Ex) Bag-Of-Nodes 인 경우, 그래프의 Node 개수를 통해 커널 함수를 정의할 수 있다

$$\phi\left(\begin{array}{c} \text{graph 1} \end{array}\right) = \phi\left(\begin{array}{c} \text{graph 2} \end{array}\right)$$


## 2. Traditional Methods for Machine Learning in Graphs

### (10) Graph-Level Features and Graph Kernels : Graph Kernel

- $\phi()$ ??

Degree를 고려하지 않은 경우

$$\phi(\text{graph}) = \phi(\text{graph})$$

Degree를 고려한 경우

Deg1: ● Deg2: ● Deg3: ●

$$\phi(\text{graph}) = \text{count}(\text{graph}) = [1, 2, 1]$$

Obtains different features  
for different graphs!

$$\phi(\text{graph}) = \text{count}(\text{graph}) = [0, 2, 2]$$

## 2. Traditional Methods for Machine Learning in Graphs

### (11) Graph-Level Features and Graph Kernels : Graplet Kernel

- Count the number of different graphlets in a graph.
- 하지만 Graph-Level에서의 Graphlet 개념은 Node-Level 특징 추출 기법에서 본 그것과 다르다.

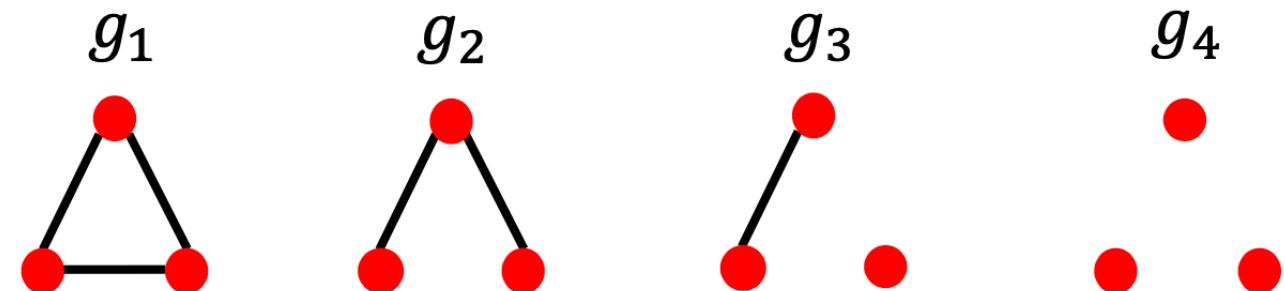
- 차이점 (1)

Nodes in graphlets here do **not need to be connected** (allows for isolated nodes)

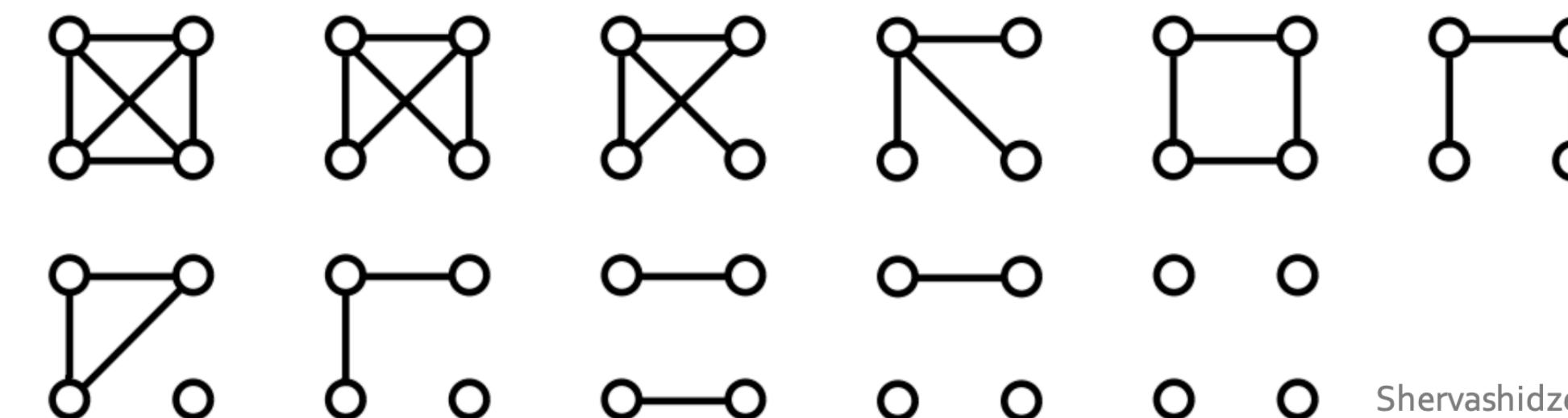
- 차이점 (2)

The graphlets here are not rooted.

- For  $k = 3$ , there are 4 graphlets.



- For  $k = 4$ , there are 11 graphlets.



## 2. Traditional Methods for Machine Learning in Graphs

### (11) Graph-Level Features and Graph Kernels : Graplet Kernel

- Count the number of different graphlets in a graph.
- 하지만 Graph-Level에서의 Graphlet 개념은 Node-Level 특징 추출 기법에서 본 그것과 다르다.

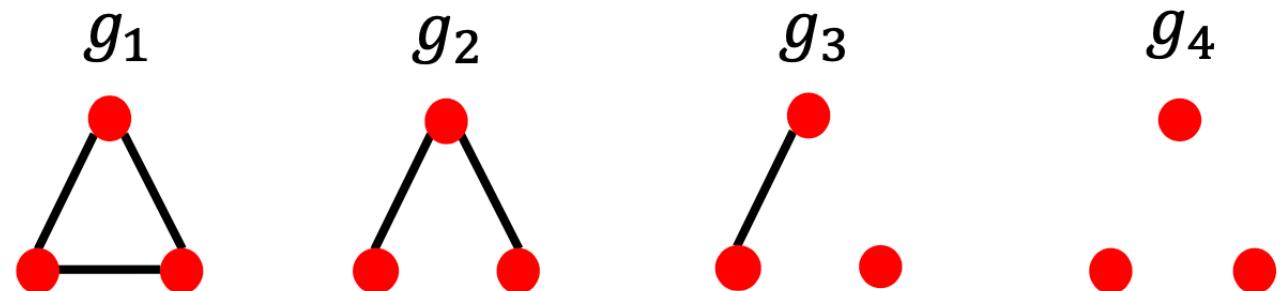
- 차이점 (1)

Nodes in graphlets here do **not need to be connected** (allows for isolated nodes)

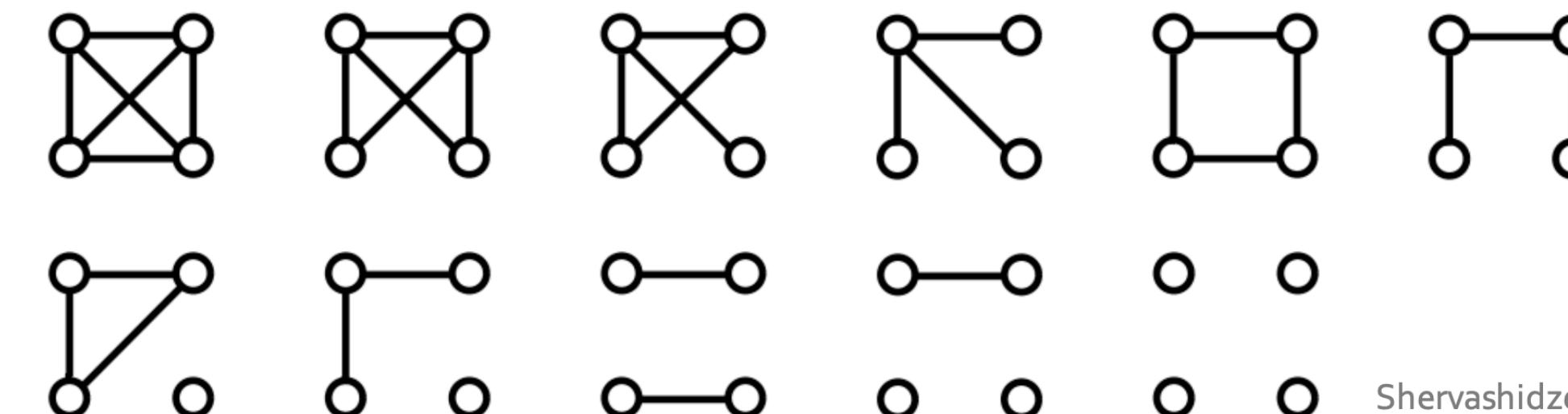
- 차이점 (2)

The graphlets here are not rooted.

- For  $k = 3$ , there are 4 graphlets.



- For  $k = 4$ , there are 11 graphlets.



## 2. Traditional Methods for Machine Learning in Graphs

### (11) Graph-Level Features and Graph Kernels : Graplet Kernel

- **Graphlet Features**

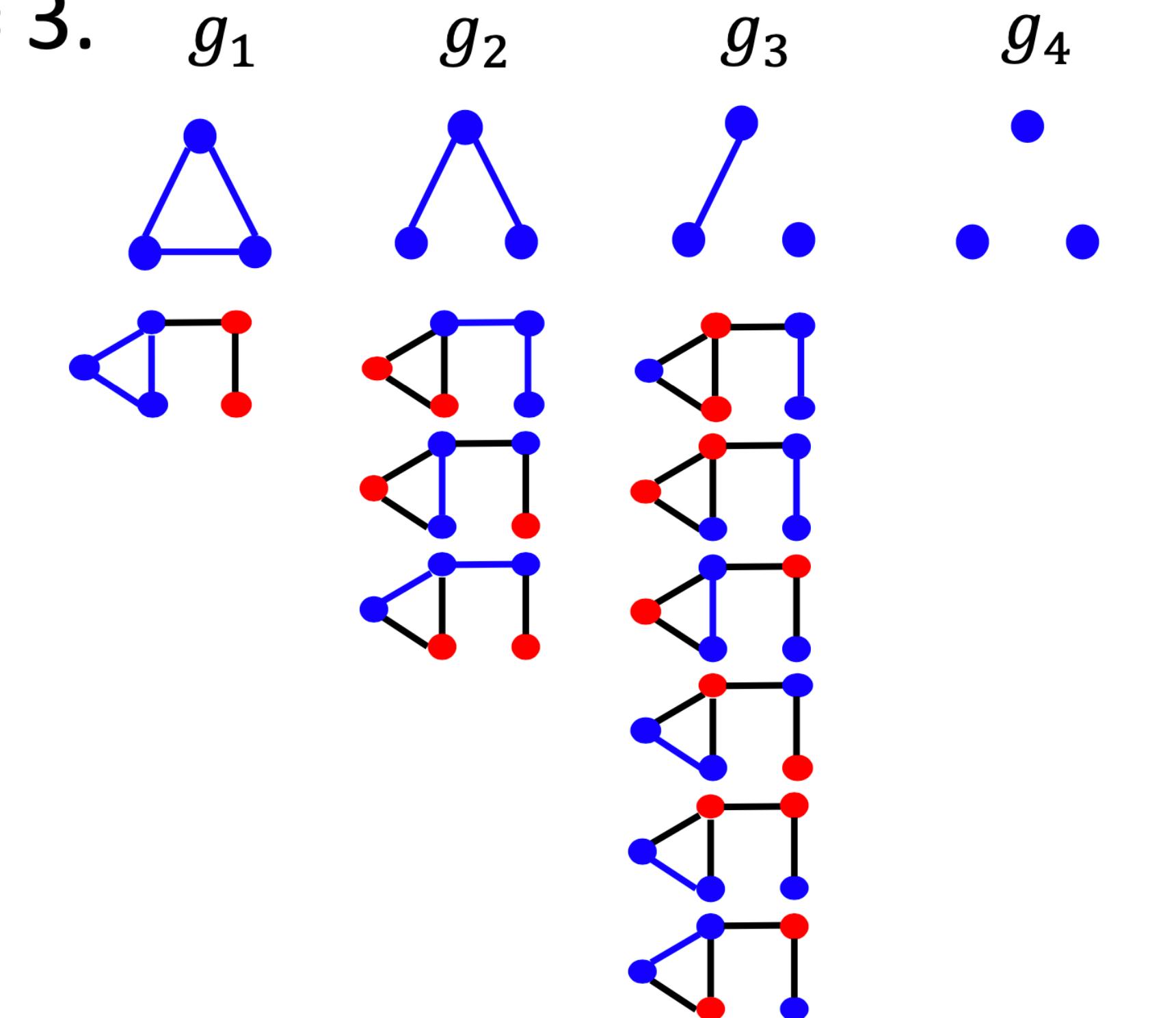
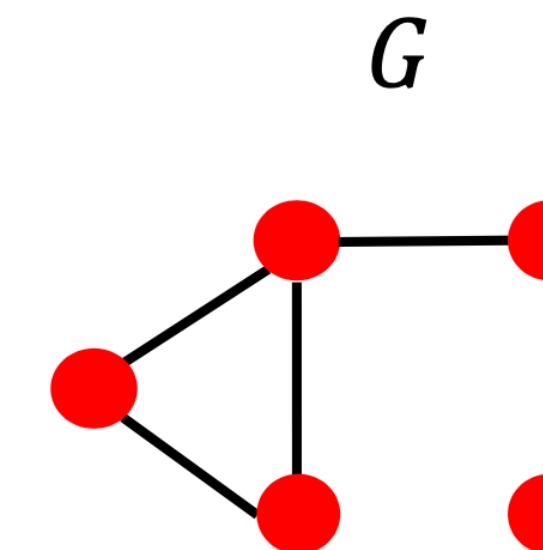
- Given graph  $G$

- **Graphlet List**  $G_k = (g_1, g_2 \dots g_{n_k})$

- **Graphlet Count Vector**  $f_G \in R^{n_k}$

- $f_G = \#g_i$  in  $\mathbf{G}$  (Graphlet 개수)

- Example for  $k = 3$ .



$$f_G = (1, 3, 6, 0)^T$$

**Limitation:** Counting graphlets is too expensive!