INCLUDE 2025 Fall Graph Based AI

그래프 기반 AI 세미나

# 그래프 분류를 위한 표현력 정의

최민재 minjaechoi@kaist.ac.kr

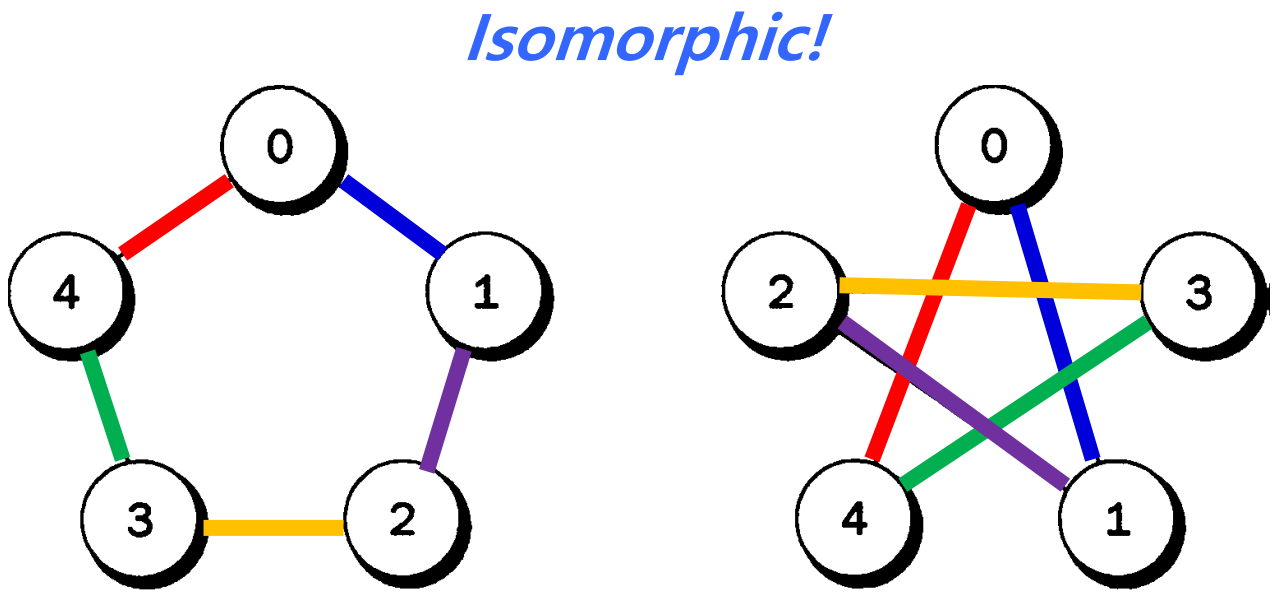
본 자료는 <Hands-on 그래프 신경망> Ch9 ~ Ch11을 토대로 제작되었습니다.

# 표현력

**GNN의 목표:** 비슷한 노드에는 비슷한 임베딩을, 다른 노드에는 다른 임베딩 생성

**표현력**: GNN이 서로 다른 그래프 구조를 얼마나 잘 구별할 수 있는지 나타내는 지표

**동형성 문제:** 두 그래프가 같은 연결 구조를 가질 때, Isomorphic을 가진다고 판단하는 문제



# Weisfeiler – Leman (WL) 테스트

두 그래프가 같은 연결 구조를 가질 때, Isomorphic을 가진다고 판단하는

***그래프 동형성 문제***를 해결하는 알고리즘 (복잡도는 알려지지 않음)

## 작동방식

(1) 각 노드는 처음에는 동일한 색 받음

(2) 각 노드는 자신의 색과 이웃의 색상을 집계함 **Sum Aggregator**

(3) (2)의 결과를 새로운 색상을 생성하는 해시 함수에 넣음

(4) 노드의 색상이 더 이상 변하지 않을 때 까지 반복

**결과로 얻은 색상들은 그래프의 정규 형태를 제공함**

**두 그래프가 동일한 색상을 공유하지 않는다면, 동형이 아님**

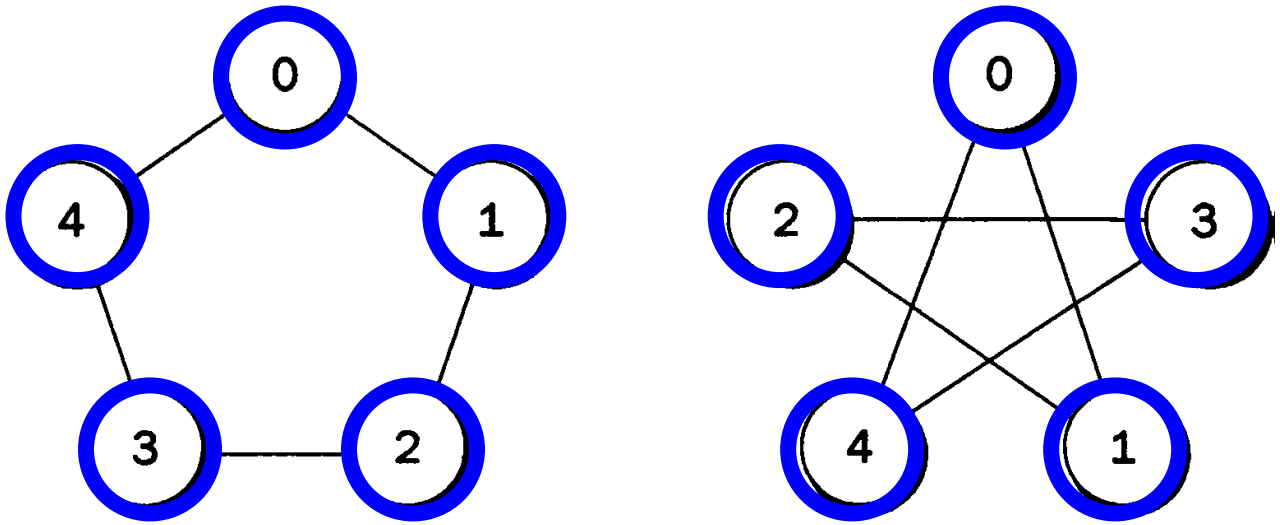
**➔ 해시 함수를 통해 완벽하게 노드 구별함**

**➔ 타 방식보다 더 많은 그래프 구조 구별 가능**

# Weisfeiler – Leman (WL) 테스트

## Step1. 그래프 각 노드는 동일한 색으로 색칠

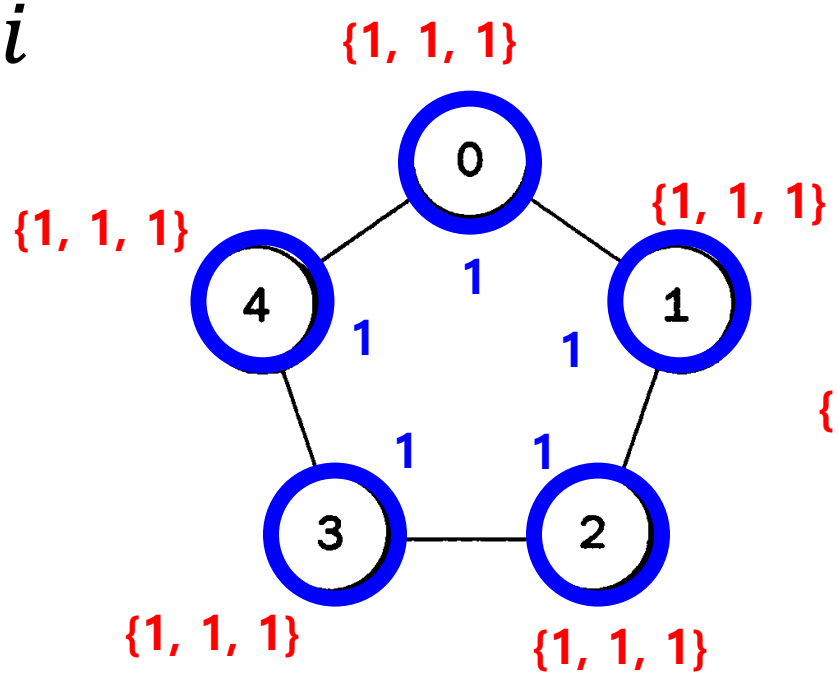
$h_i^1$

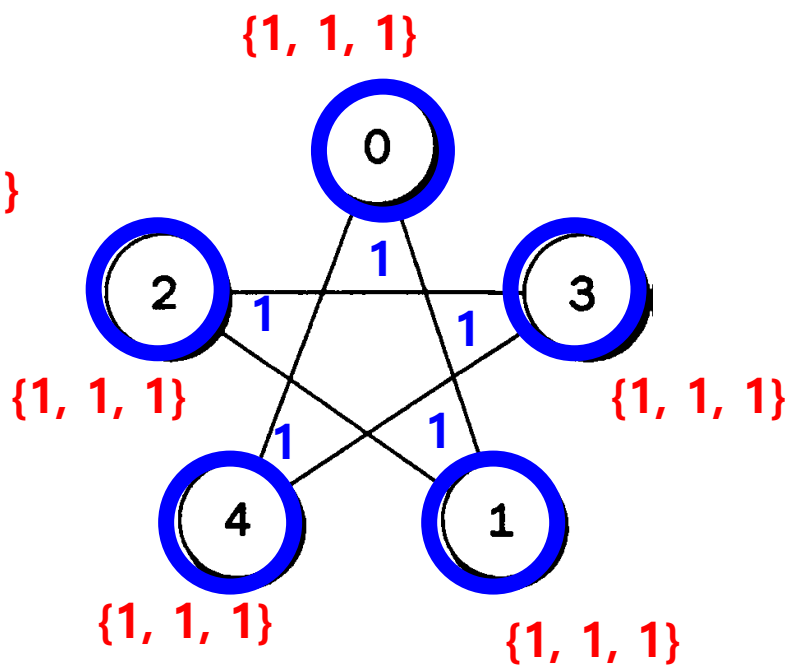


# Weisfeiler – Leman (WL) 테스트

## Step2. 각 노드는 본인 색상과 이웃의 색상을 집계

$h_i^1$

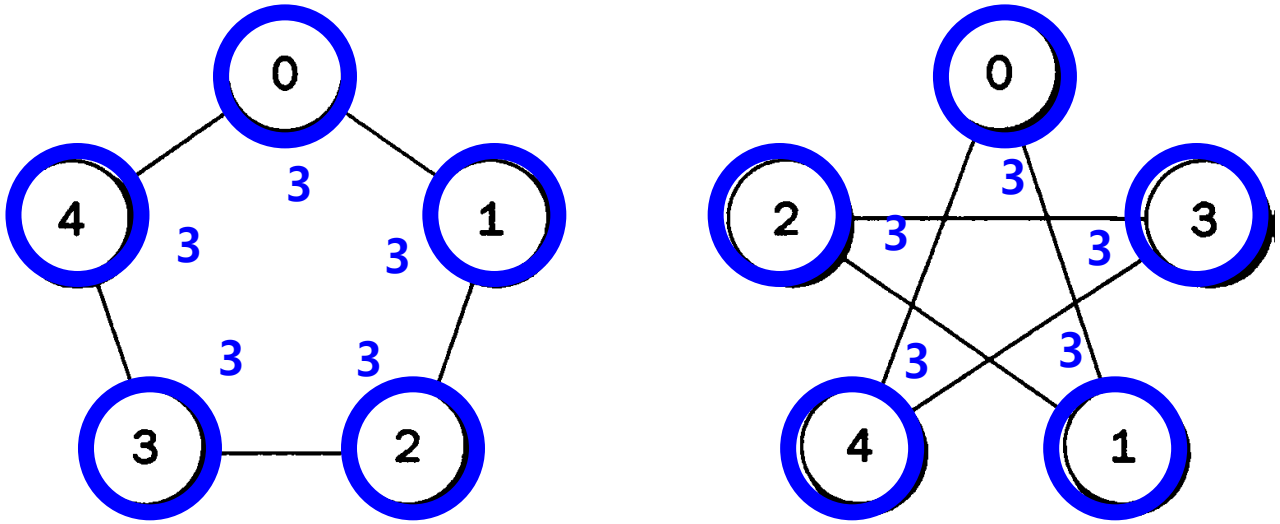


# Weisfeiler – Leman (WL) 테스트

## Step3. 그 결과를 새로운 색상을 생성하는 해시 함수에 넣음

$$h_i^1$$

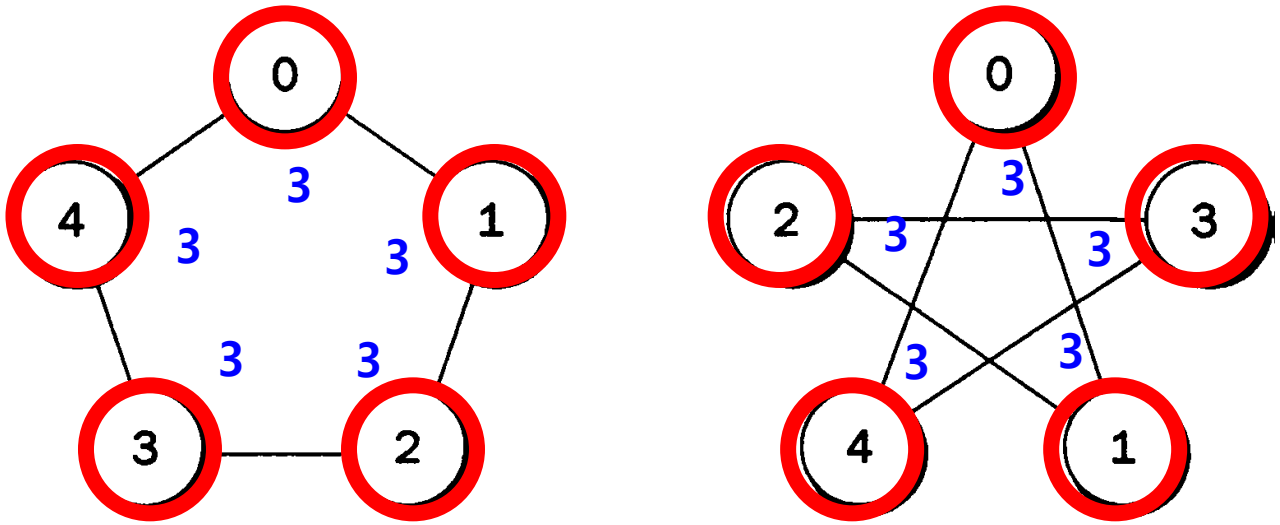



*Assume Hash Function = Identity Function*

# Weisfeiler – Leman (WL) 테스트

## Step4. Step1 ~ 3를 색상 변경이 없을 때 까지 반복

$h_i^2$




*Assume Hash Function = Identity Function*

# Graph Isomorphism Network (GIN)

**WL 테스트기와 동일한 표현력을 가지도록 설계된 GNN 아키텍처**

**집계 과정**

**집계:** 함수 f는 GNN이 다루는 이웃 노드 선택

**결합:** 함수 $\phi$는 선택된 노드로부터 임베딩을 결합하여 타겟노드에 대한 새로운 임베딩 생성

**i 번째 노드 임베딩**

$$h_i' = \phi\left(h_i, f\left(\{h_j : j \in \mathcal{N}_i\}\right)\right)$$

# Graph Isomorphism Network (GIN)

i 번째 노드 임베딩

$$h_i' = \phi(h_i, f(\{h_j : j \in \mathcal{N}_i\}))$$

평균 집계기 i 번째 노드의 모든 이웃 집계

i 번째 노드 임베딩

$$h_i' = \mathrm{MLP}((1+\epsilon) \cdot h_i + \sum_{j \in \mathcal{N}_i} h_j)$$

# Graph Isomorphism Network (GIN)

**k-WL 테스트와 k-GNN**

개별 노드만 고려하는 것 대신 k-tuple을 이용해 먼 거리의 노드들 고려

(더 많은 그래프 구조를 구분)

➔ Graph Classification 문제에서 유용한 신경망 구조

**그래프 분류(Graph Classification): 노드 임베딩을 하나의 그래프 임베딩으로 변환하여 그래프 전체의 특성을 예측**

# GNN을 이용한 그래프 분류

그래프 분류: GNN이 만들어내는 노드 임베딩에 기반함

전역풀링 / 그래프 수준의 리드아웃(Readout)

평균 그래프 풀링

$$h_G = \frac{1}{N}\sum_{i=0}^{N} h_i$$

최댓값 그래프 풀링

$$h_G = \max_{i=0}^{N}(h_i)$$

합산 그래프 풀링

$$h_G = \sum_{i=0}^{N} h_i$$

표현력 가장 강함

$$h_G = \sum_{i=0}^{N} h_i^0 \,||\, \ldots \,||\, \sum_{i=0}^{N} h_i^k$$

GNN의 각 k-layer에서의 노드 임베딩을 합한 후 연결

INCLUDE 2025 Fall Graph Based AI

그래프 기반 AI 세미나

# 그래프 신경망을 이용한 링크 예측

최민재 minjaechoi@kaist.ac.kr

# 전통적 링크 예측 방법

**링크 예측: 두 노드 사이에 링크(Edge) 가 존재할지 여부를 예측하는 문제**

**휴리스틱 기법 : (1) 지역 휴리스틱 (1hop or 2hop)**

**공통 이웃 (Common Neighbors)** 1-hop

$$f(u,v) = |\mathcal{N}(u) \cap \mathcal{N}(v)|$$

두 노드 사이 공통인 이웃 수 측정

**자카드 계수 (Jaccard's Coefficient)** 1-hop

$$f(u,v) = \frac{|\mathcal{N}(u) \cap \mathcal{N}(v)|}{|\mathcal{N}(u) \cup \mathcal{N}(v)|}$$

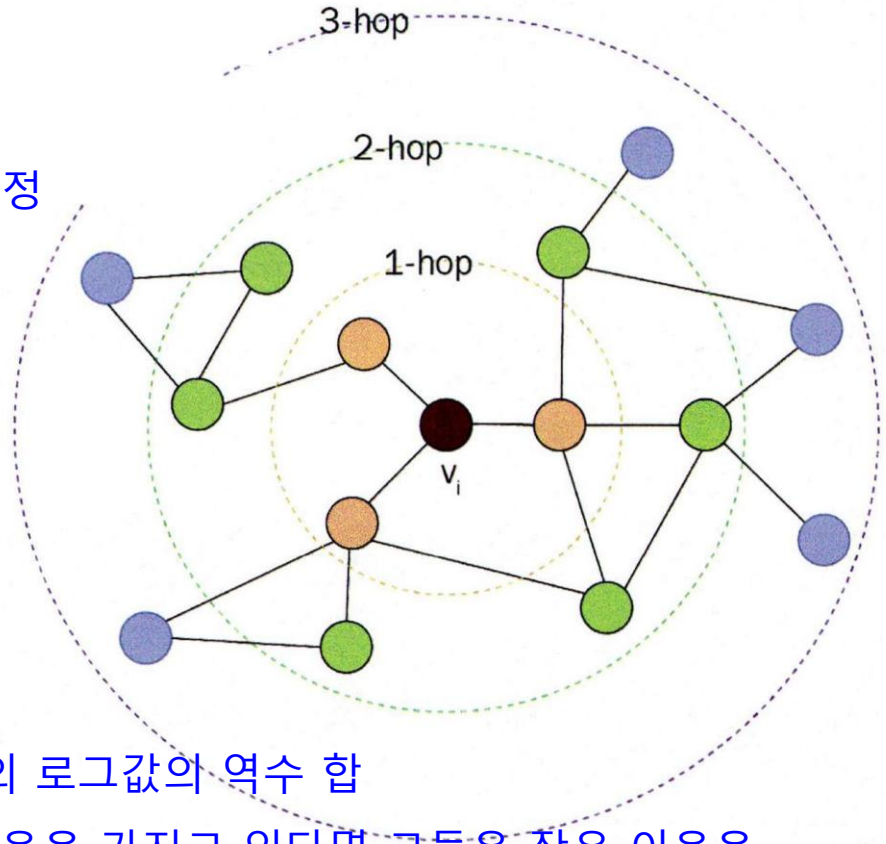
공통 이웃 비율 측정

**아다믹-아다르 인덱스 (Adamic-Adar index)** 2-hop

$$f(u,v) = \sum_{x \in \mathcal{N}(u) \cap \mathcal{N}(v)} \frac{1}{\log|\mathcal{N}(x)|}$$

공통 이웃 노드의 차수의 로그값의 역수 합

Idea: 공통 이웃이 큰 이웃을 가지고 있다면 그들은 작은 이웃을 가진 이웃보다 중요성이 적어짐



# 전통적 링크 예측 방법

링크 예측: 두 노드 사이에 링크(Edge) 가 존재할지 여부를 예측하는 문제

## 휴리스틱 기법 : (2) 전역 휴리스틱

### 카츠 인덱스 (Katz index)

가중치: 더 긴 경로에 패널티 부여 (보통 0.8~0.9)

$$f(u,v) = \sum_{i=1}^{\infty} \beta^i A^i$$

모든 길이 경로는 인접행렬의 거듭제곱으로 계산

두 노드 사이 모든 가능한 경로의 가중 합 계산

Idea: 짧은 경로가 있으면 연결될 가능성이 높음

### 재시작 랜덤 워크 (Random walk with restart)

특정 시작노드(s)에서 출발한 Random Walker가 그래프 간선에 따라 이동. 보행자는 이웃노드로 이동할 확률 외에도 매 단계마다 정해진 재시작 확률($\alpha$)로 시작 노드(s)로 돌아갈 수 있음.

일정 횟수 이후 대상 노드와 가장 방문 횟수 높은 노드 사이 링크 제안

**➔ S로부터 경로가 짧거나 연결성이 강한 노드에 높은 값 부여**

# 전통적 링크 예측 방법

링크 예측: 두 노드 사이에 링크(Edge) 가 존재할지 여부를 예측하는 문제

**행렬 분해 (Matrix Factorization)**

노드 임베딩(Z) 학습하여 인접행렬(A) 전체를 근사

$$A \approx Z^T Z$$

노드들이 유사할 경우 내적 값 최대

노드들이 상이할 경우 내적 값 최소

**목표:**

$$\underset{z}{\text{minimize}} \sum_{i \in V,\, j \in V} (A_{uv} - z_v^T z_u)^2$$

$A_{uv}$: 실측값, $z_v^T z_u$: 예측값

노드 임베딩 학습하여 실측값과 예측 값 사이 Norm을 최소화

# 노드 임베딩 활용한 링크 예측 : Graph Autoencoder

**Encoder**

$$Z = \mathrm{GCN}(X, A)$$

클래식한 두 개 계층으로 구성된 GCN

**Decoder**

$$\hat{A} = \sigma(Z^T Z)$$

행렬 분해와 Sigmoid 함수 사용하여 인접행렬 근사

➔ **인접 행렬 각 요소에 대한 확률 예측**

두 인접 행렬의 요소 사이의 이진 교차 엔트로피 손실함수 사용하여 훈련됨

$$\mathcal{L}_{BCE} = \sum_{i \in V, j \in V} -A_{ij} \log(\hat{A}_{ij}) - (1 - A_{ij}) \log(1 - \hat{A}_{ij})$$

Cf) 인접행렬은 종종 희소하여, 이전 손실함수에서 $A_{ij} = 1$을 선호하는 가중치 추가 또는 훈련 중 0 값을 더 적게 샘플링 하여 레이블을 균형있게 만들 수 있음

# 노드 임베딩 활용한 링크 예측 : VGAE(Variational Graph Autoencoder)

➔ **노드 임베딩을 직접 학습하는 대신, VGAE를 샘플링 하여 임베딩을 생성하는 정규 분포 학습**

(GAE의 확률적 변형)

**Encoder**

임베딩 Z 자체 대신, Z가 따르는 잠재 정규 분포의 평균$\mu$ 및 분산$\sigma^2$의 매개변수 학습

첫 번째 계층을 공유하는 두 GCN으로 구성

**Decoder**

학습된 분포 $\mathcal{N}(\mu, \sigma^2)$에서 재매개변수화 기법을 사용하여 임베딩 $Z_i$ 샘플링.

이후 인접행렬 근사: $\hat{A} = \sigma(Z^T Z)$

➔ 인코더의 출력값이 정규분포를 따라야 하기에, 손실함수에 새로운 항(쿨백 라이블러 발산) 추가 :

두 분포간 차이 측정

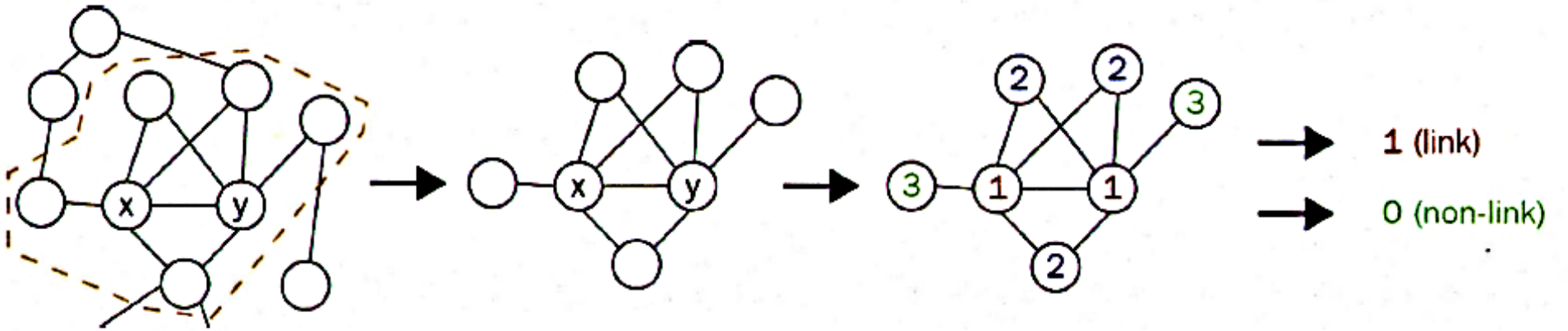
$$\mathcal{L}_{ELBO} = \mathcal{L}_{BCE} - \mathrm{KL}[q(Z|X,A) \,||\, p(Z)]$$

인코더 사전분포

# 노드 임베딩 활용한 링크 예측 : SEAL

**SEAL: Subgraphs, Embeddings, and Attributes for Link prediction**

**전체 그래프 대신 입력으로 사용**
(= 지역 휴리스틱을 자동으로 학습)




**둘러싼 서브그래프 추출**

**둘러싼 서브 그래프 추출**

실제 링크 세트와 가짜링크 세트를 통해 훈련 데이터 형성을 포함함

**노드 레이블링**

**노드 정보 행렬 구축**

노드 레이블, 노드 임베딩, 노드 특성의 세 가지 구성 요소 포함

**그래프 인공신경망**

**GNN 학습**

노드 정보 행렬을 입력받아 링크 가능성 출력

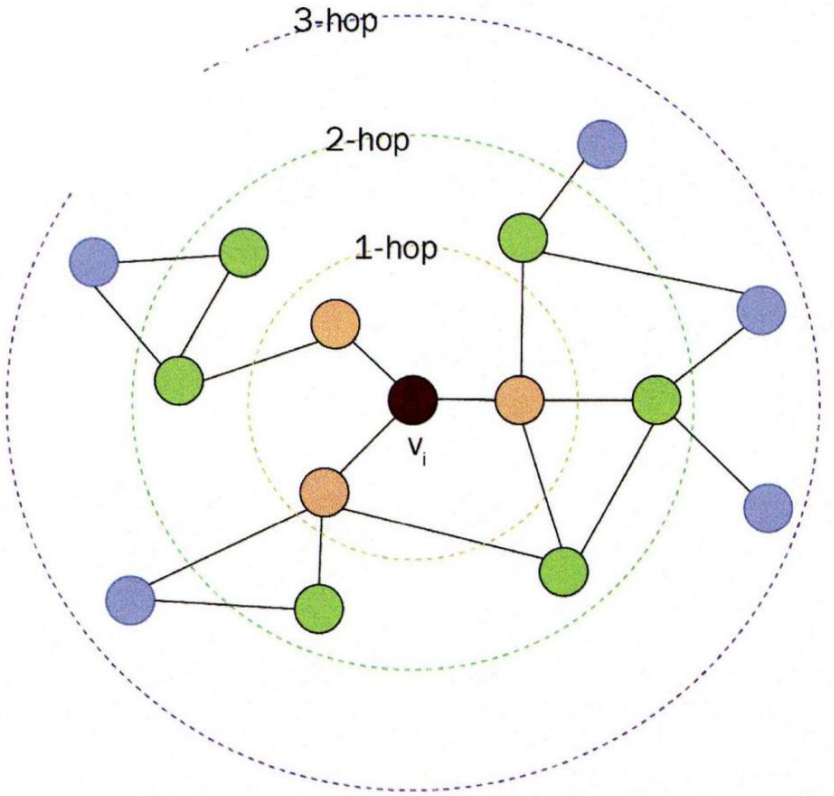
# 노드 임베딩 활용한 링크 예측 : SEAL

**SEAL: Subgraphs, Embeddings, and Attributes for Link prediction**

## 둘러싼 서브 그래프 추출

➔ **대상 노드와 그들의 k-hop 이웃을 나열하여 그들의 간선과 특성 추출**

높은 hop 수 = 휴리스틱 품질 향상하나, 계산 비용 높음



# 노드 임베딩 활용한 링크 예측 : SEAL

**SEAL: Subgraphs, Embeddings, and Attributes for Link prediction**

## 노드 레이블링

**이중 반경**

$(d(i,x), d(i,y))$

GNN이 단순히 이웃이라는 사실을 아는 것 외에, **각 노드가 대상 x, y로 부터 얼마나 멀리 떨어져 있는지를 알 수 있게** 하여 구조적 중요성 설명 가능

**DRNL(Double-Radius Node Labeling) Algorithm**

이중 반경 정보를 **하나의 정수형 레이블로** 변환하는 알고리즘

# 노드 임베딩 활용한 링크 예측 : SEAL

**SEAL: Subgraphs, Embeddings, and Attributes for Link prediction**

## DRNL Algorithm

이중 반경 정보를 **하나의 정수형 레이블로** 변환하는 알고리즘

**목표:** 이중 반경 쌍을 계산한 후, 이를 원핫 인코딩 하여 GNN의 입력 특성으로 사용하기 위한 **레이블** 생성

**두 거리를 결합하여 거리가 가까울수록 더 작은 레이블 부여**

**(1) x, y에 레이블 1 할당**

**(2) 반경 (1,1) 가진 노드에 레이블 1 할당**

**(3) 반경 (1,2) or (2,1) 가진 노드에 레이블 3 할당**

**(4) 반경 (1,3) or (3,1) 가진 노드에 레이블 4 할당**

$$f(i) = 1 + min(d(i,x), d(i,y)) + (d/2)[(d/2) + (d\%2) - 1]1$$

# 노드 임베딩 활용한 링크 예측 : SEAL

**SEAL: Subgraphs, Embeddings, and Attributes for Link prediction**

## GNN 훈련 (Deep Graph Convolutional Neural Network, DGCNN)

(1) 여러 GCN 레이어가 노드 임베딩 계산후 연결됨

(2) 글로벌 정렬 풀링 레이어(Global Sort Pooling)은 임베딩을 일관된 순서로 정렬 후 순열 불변이 아닌 컨볼루션 레이어에 입력

(3) 전통적인 컨볼루션 및 밀집 레이어가 정렬된 그래프 표현에 적용되어 링크 황률 출력

INCLUDE 2025 Fall Graph Based AI

그래프 기반 AI 세미나

# 그래프 신경망을 이용한 그래프 생성

최민재 minjaechoi@kaist.ac.kr

# 그래프 생성의 중요성

## 활용 분야

데이터 증강 / 이상치 감지 / 신약 개발...

## 유형

**사실적 생성**: 주어진 그래프 모방

**목표 지향적 생성:** 특정 지표 최적화

# 기존 그래프 생성 방식

**에르되시-레니 (Erdos – Renyi Model)**

단점: 각 간선이 독립적이라는 가정 때문에 클러스터/커뮤니티가 존재할 경우, 실제 그래프 모방하기 어려움

$G(n, p)$

n: 노드 개수, p: 가능한 모든 두 노드 쌍 사이에 간선이 연결될 확률

➔ 각 간선이 존재할지 여부를 확률 p 로 독립적으로 결정하여 그래프 생성

$G(n, M)$

n: 노드 개수, M: 간선 개수

노드가 n개, 간선이 M개인 모든 가능한 그래프 중 임의로 한 개 선택

# 기존 그래프 생성 방식

**스몰 월드 (Small-World) 모델**

실세계 네트워크 특징인 **짧은 경로와 높은 클러스터링 개수**를 모방

**스몰 월드 모델로 생성된 그래프의 특징**

(1) Short Path Length: 네트워크 내의 임의의 두 노드 사이의 평균거리가 상대적으로 짧음 → 정보가 네트워크 전체에 빠르게 확산되기 쉬움

(2) High Clustering Coefficient: 네트워크의 노드들이 서로 가깝게 연결되어 촘촘한 클러스터(/커뮤니티)를 형성

# 기존 그래프 생성 방식

**스몰 월드 (Small-World) 모델**

실세계 네트워크 특징인 **짧은 경로와 높은 클러스터링 개수**를 모방

**와츠-스트로가츠 모델 (Watts-Strogatz Model)**

**(1) 노드 생성 :** n개 노드 생성

**(2) 규칙적 연결 :** 각 노드는 가장 가까운 k개의 이웃과 연결됨 (k가 홀수일 경우 k-1)

**(3) 무작위 재배선(Rewiring):** 확률 p로 기존 간선 중 일부를 무작위의 다른 노드와 연결되도록 재배선 ➔ 네트워크 전체 경로 길이를 줄임

한계: 노드 수가 고정되어 있어 네트워크가 커질 수 없고, 실제와 같은 차수 분포 만들기 어려움

# Recall : VGAE

**➔ 노드 임베딩을 직접 학습하는 대신, VGAE를 샘플링 하여 임베딩을 생성하는 정규 분포 학습**

(GAE의 확률적 변형)

**Encoder**

임베딩 Z 자체 대신, Z가 따르는 잠재 정규 분포의 평균$\mu$ 및 분산$\sigma^2$의 매개변수 학습

➔ 원본 그래프의 구조적 정보를 확률적 분포로 저장

**Decoder**

학습된 분포 $\mathcal{N}(\mu, \sigma^2)$에서 재매개변수화 기법을 사용하여 임베딩 $Z_i$ 샘플링.

이후 인접행렬 근사: $\hat{A} = \sigma(Z^T Z)$

**새 그래프 확정:** 인접행렬: 노드간 연결 확률 나타내므로, 특정 임계값 설정 후 임계값을 넘는 노드 쌍을 연결하여 이진 인접 행렬(그래프) 생성

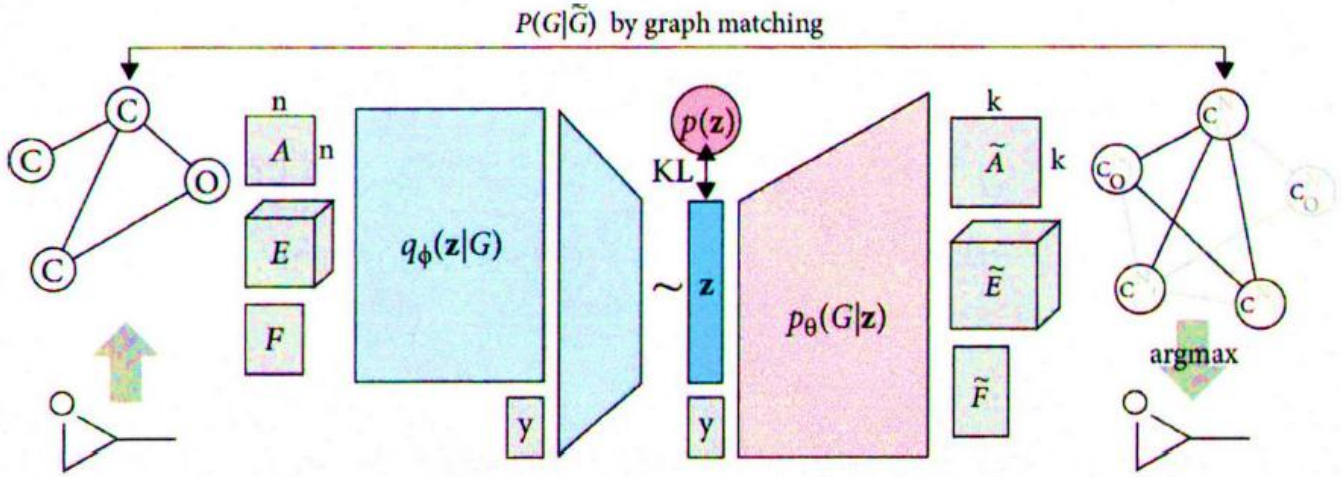
# GNN 기반 그래프 생성 (딥 그래프 생성모델)

**GVAE (Graph Variational Autoencoder)**

인접행렬 뿐만 아니라 엣지 속성 텐서(E)와 노드 특성값 행렬(F)까지 포함하는 더 복잡한 그래프 $G = (A, E, F)$ 학습하여 분자구조 생성

**CGVAE(Constrained Graph Variational Autoencoder)**

제약 조건 검사하거나 라그랑지안 기반으로 정규화를 추가하여 유효한 분자 구조와 같이 조건을 갖는 그래프 생성 가능

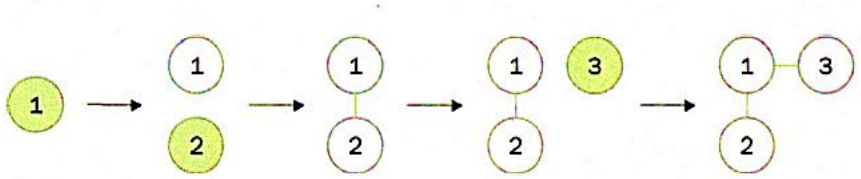


# GNN 기반 그래프 생성 (딥 그래프 생성모델)

**자기회귀 모델**

그래프 생성을 데이터와 과거 의사 결정을 고려하는 순차적 의사결정 과적으로 봄

**원리** 각 단계에서 모델은 새 노드나 새 링크를 만들고, 이렇게 얻은 부분 그래프는 다음 생성 단계의 입력이 됨.



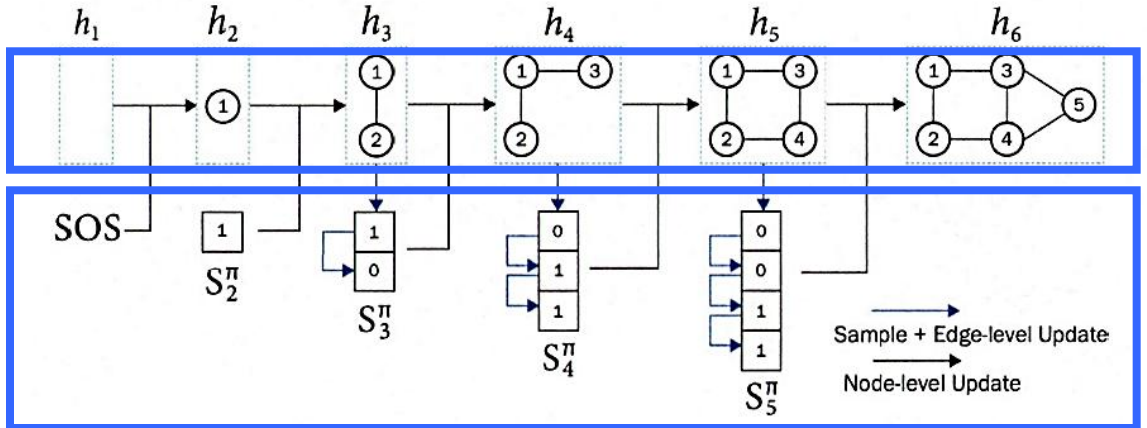
**GraphRNN:** 두 개의 순환인공신경망(RNN)을 사용하여 과정 구현

**(1) 그래프 레벨 RNN:** 새로운 노드를 생성하고 초기 상태 설정

**(2) 간선 레벨 RNN:** 새로 생긴 노드가 기존 노드들에 연결될지 여부 예측하여 인접행렬 채워 나감

그래프 레벨 RNN

간선 레벨 RNN




➔ 행/열

➔ 0/1

# GNN 기반 그래프 생성 (딥 그래프 생성모델)

## 생성적 적대 신경망 (GAN)

Generator와 Discriminator라는 두 신경망이 경쟁하며 학습하는 제로섬 게임 프레임워크

**Generator:** 훈련 데이터와 유사한 새로운 그래프 데이터 생성

**Discriminator:** 각 샘플이 진짜인지 또는 생성기가 만든 가짜인지를 구분

# GNN 기반 그래프 생성 (딥 그래프 생성모델)

**WGAN (Wasserstein GAN)**

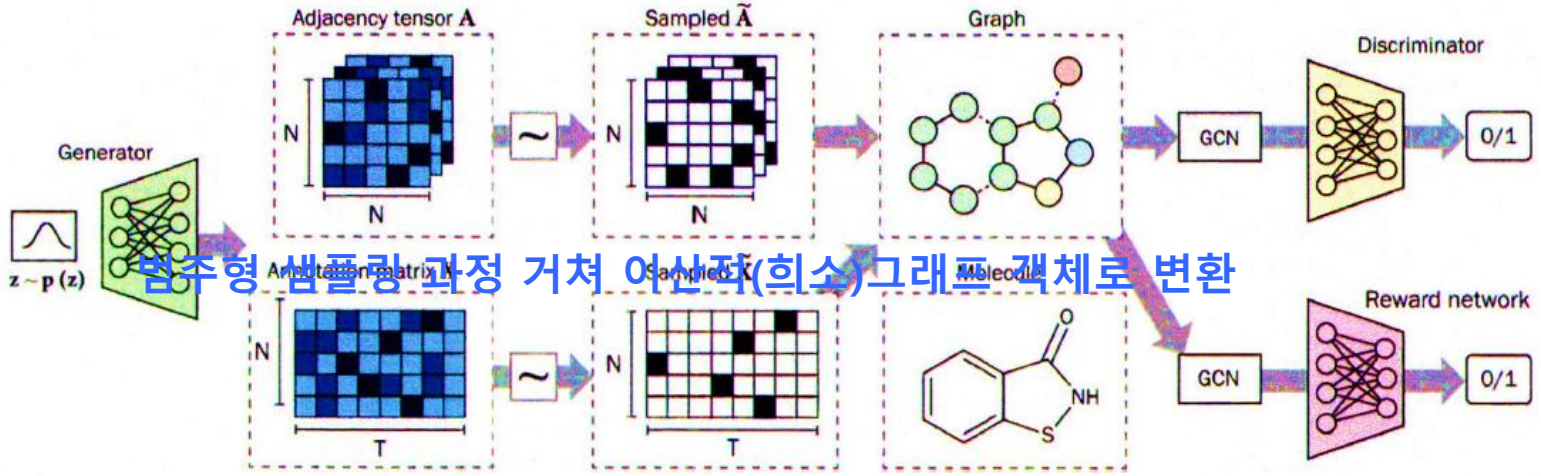
두 확률 분포 사이의 바서슈타인 거리를 최소화하여 학습 안정성 높임.

생성된 데이터분포와 실제 데이터 분포 사이 차이를 측정하는 행렬

**MolGAN(molecular GAN)**

그래프 구조의 데이터를 직접 처리하는 그라디언트 패널티를 쓰는 WGAN과 원하는 화학적 성질을 갖는 분자를 생성하기 위한 강화학습 목표 결합.

그래프 간선 & 결합유형에 대한 정보




범주형 샘플링 과정 거쳐 어산적(희소)그래프 객체로 변환

그래프 노드 유형에 대한 정보

# GNN 기반 그래프 생성 (딥 그래프 생성모델)

**WGAN (Wasserstein GAN)**

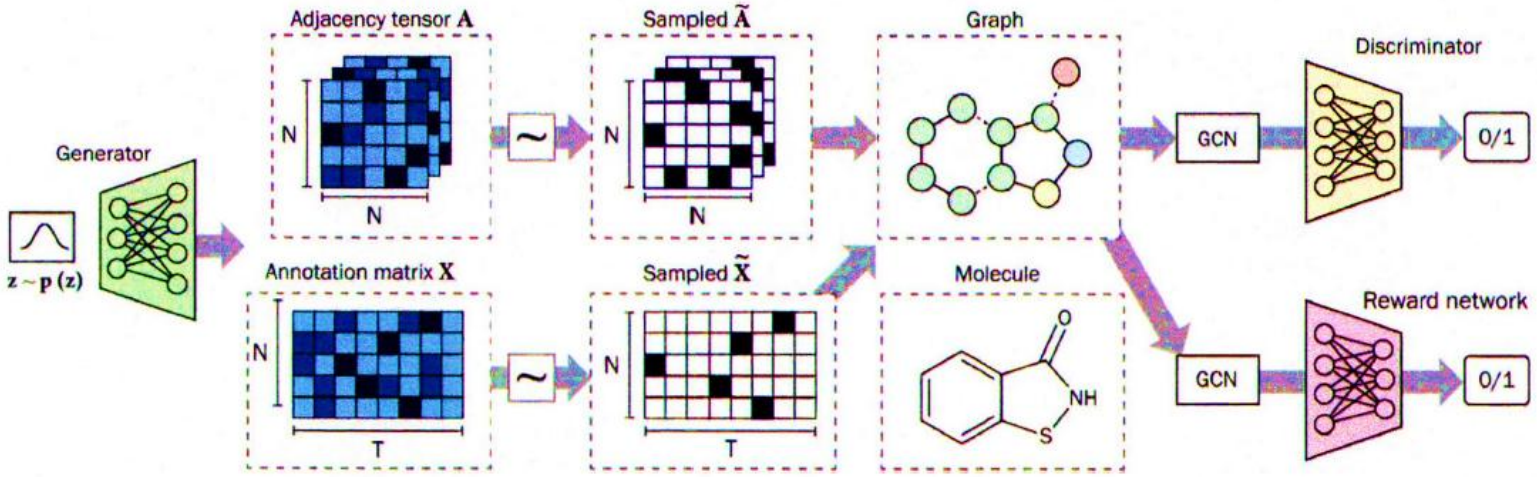
두 확률 분포 사이의 바서슈타인 거리를 최소화하여 학습 안정성 높임.

**생성된 데이터분포와 실제 데이터 분포 사이 차이를 측정하는 행렬**

**MolGAN(molecular GAN)**

그래프 구조의 데이터를 직접 처리하는 그라디언트 패널티를 쓰는 WGAN과 원하는 화학적 성질을 갖는 분자를 생성하기 위한 강화학습 목표 결합.

**범주형 샘플링 과정 거쳐 이산적(희소)그래프 객체로 변환**




**샘플링 하는 이유**

원자 유형(O/C/N...)은 연속적 값이 아닌 정해진 범주에 속하는 이산적 값이기 때문, 대부분의 쌍은 분자에서 실제로 연결되어 있지 않기 때문

# GNN 기반 그래프 생성 (딥 그래프 생성모델)

**WGAN (Wasserstein GAN)**

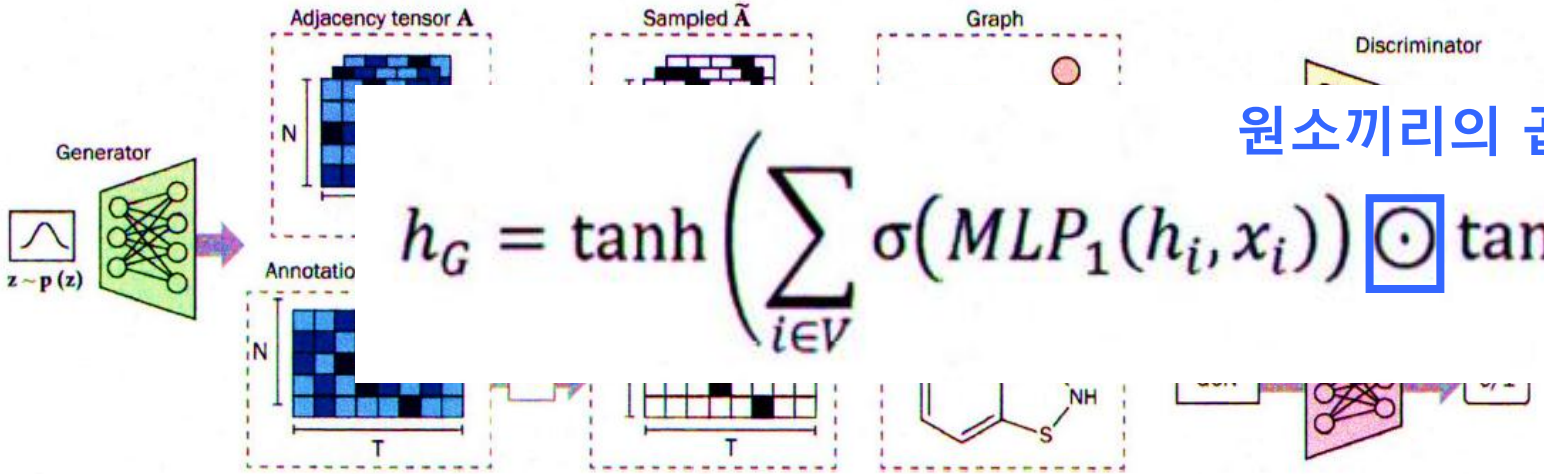
두 확률 분포 사이의 바서슈타인 거리를 최소화하여 학습 안정성 높임.

생성된 데이터분포와 실제 데이터 분포 사이 차이를 측정하는 행렬

**MolGAN(molecular GAN)**

그래프 구조의 데이터를 직접 처리하는 그라디언트 패널티를 쓰는 WGAN과 원하는 화학적 성질을 갖는 분자를 생성하기 위한 강화학습 목표 결합.

범주형 샘플링 과정 거쳐 이산적(희소)그래프 객체로 변환




$$h_G = \tanh\left(\sum_{i \in V} \sigma\left(MLP_1(h_i, x_i)\right) \odot \tanh\left(MLP_2(h_i, x_i)\right)\right)$$

원소끼리의 곱

샘플링 하는 이유

값이 아닌
값이기 때문,
로 연결되어

# GNN 기반 그래프 생성 (딥 그래프 생성모델)

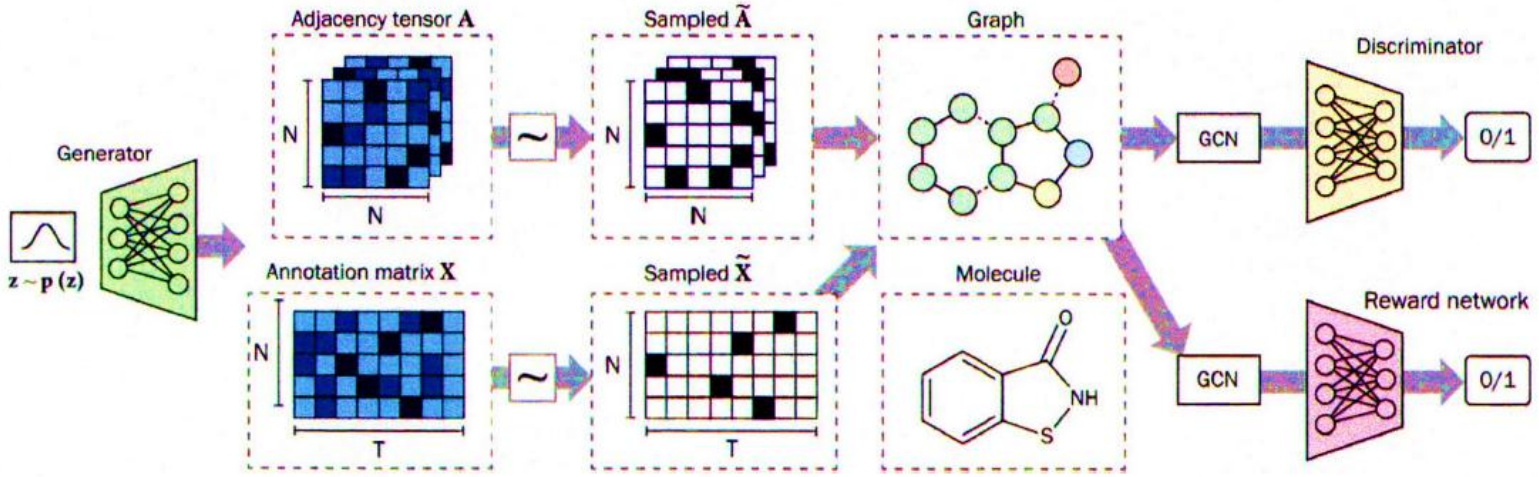
**WGAN (Wasserstein GAN)**

두 확률 분포 사이의 바서슈타인 거리를 최소화하여 학습 안정성 높임.

**생성된 데이터분포와 실제 데이터 분포 사이 차이를 측정하는 행렬**

**MolGAN(molecular GAN)**

그래프 구조의 데이터를 직접 처리하는 그라디언트 패널티를 쓰는 WGAN과 원하는 화학적 성질을 갖는 분자를 생성하기 위한 강화학습 목표 결합.




그래프 임베딩을 MLP 통해 –inf ~ +inf 사이 스칼라값으로 출력(진짜 데이터셋의 그래프와 얼마나 유사한지 평가)

그래프 임베딩을 mlp를 통해 0과 1 사이 스칼라 값으로 출력 (외부시스템에서 제공하는 점수 근사하여 생성된 분자가 원하는 화학적 특성을 얼마나 잘 충족하는지에 대한 보상 제공)

# GNN 기반 그래프 생성 (딥 그래프 생성모델)

**MolGAN**

Generator와 Discriminator라는 두 신경망이 경쟁하며 학습하는 제로섬 게임 프레임워크

**Generator:** 훈련 데이터와 유사한 새로운 그래프 데이터 생성

WGAN과 강화학습 손실함수의 선형 조합을 사용하여 학습됨.

원자 유형을 포함하는 노드 행렬 X와 간선과 결합 유형을 모두 포함하는 인접 행렬 A를 출력하는 MLP

**Discriminator:** 각 샘플이 진짜인지 또는 생성기가 만든 가짜인지를 구분

WGAN 손실함수만을 사용하여 학습

**Reward Network:** 각 그래프에 점수를 매김

외부 시스템(RDKit)에서 제공하는 실제 점수를 기반으로 MSE 손실함수를 사용하여 학습함.

# GNN 기반 그래프 생성 (딥 그래프 생성모델)

**MolGAN**

**Discriminator:** 각 샘플이 진짜인지 또는 생성기가 만든 가짜인지를 구분

**Reward Network:** 각 그래프에 점수를 매김

### Relational-GCN 사용

여러 유형의 간선을 지원하도록 변형됨 (화학 분자 간 다양한 결합 표현 위해 사용)

### 노드 임베딩 집계

여러 층의 Graph Convolution을 거쳐 노드별 임베딩 계산

노드의 임베딩 집계 함수를 통해 하나의 그래프 레벨 벡터로 변환

# 감사합니다


## 최민재

minjaechoi@kaist.ac.kr