

GNN

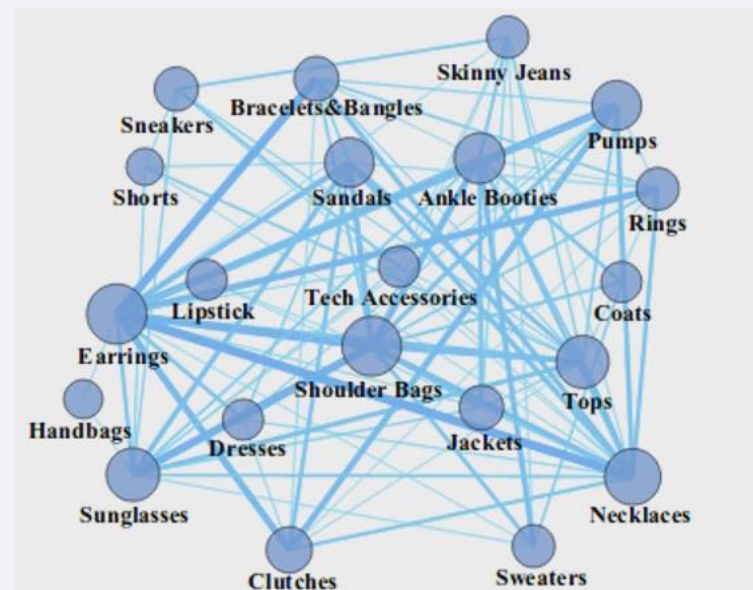
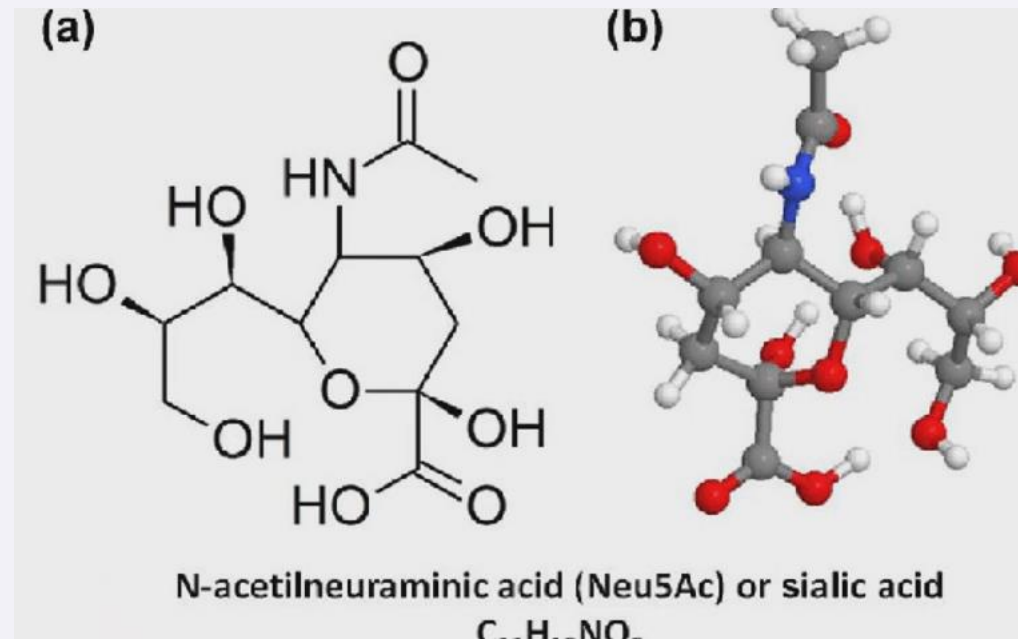
The Graph Neural Network Model

01

The Graph Neural Network Model

Graph Structure

Social Network, 영화 추천, 분자 화합물 등
현실세계의 다양한 것들이 graph 구조로 되어있음

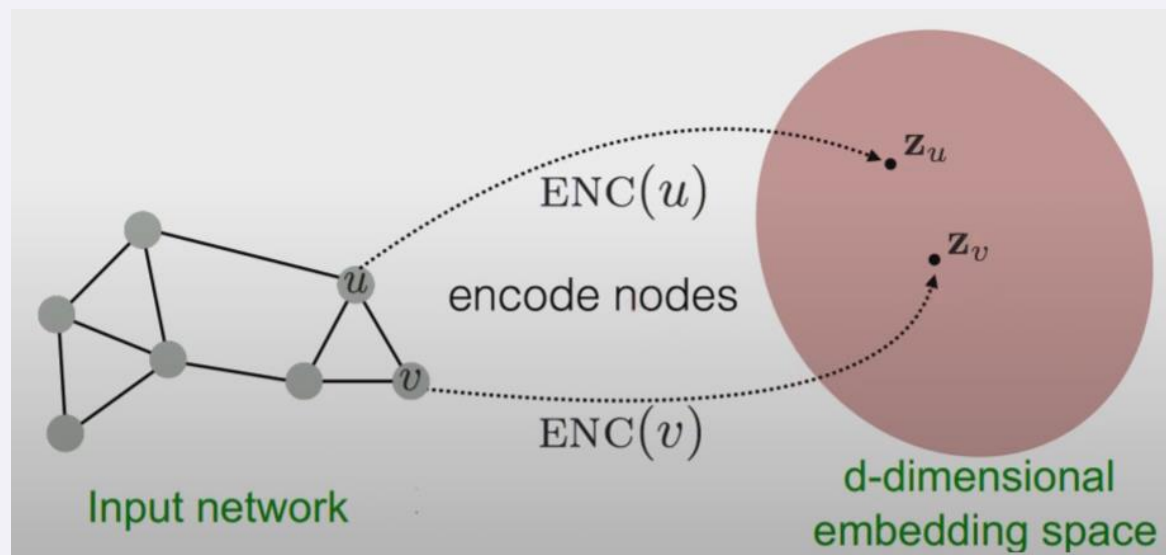


01

The Graph Neural Network Model

Graph in Machine Learning

1. Graph structure는 구조 자체가 learning에 많이 사용
2. Graph structure를 인코딩하여 embedding space에 mapping, 해당 값들을 application에 이용

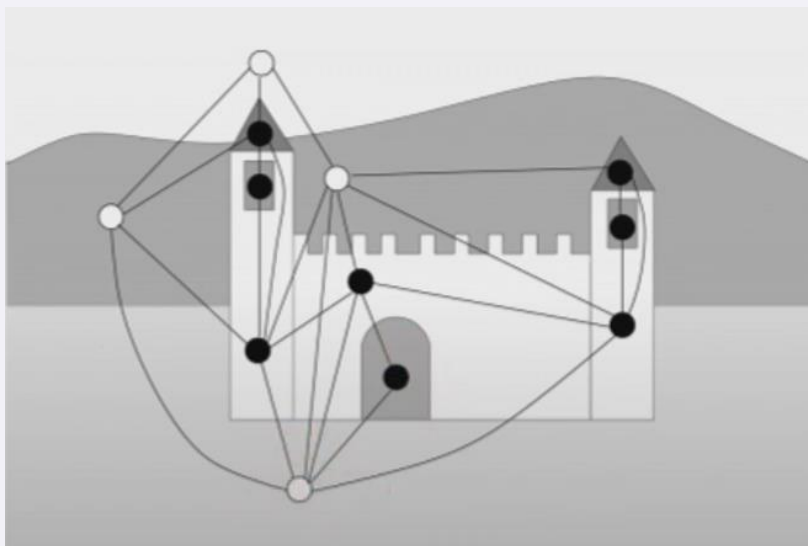


01

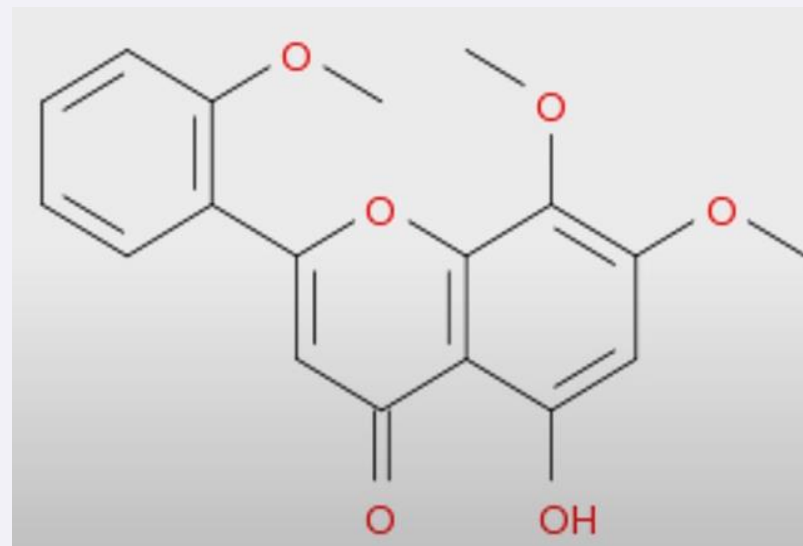
The Graph Neural Network Model

Application class

1) node-focused application



2) graph-focused application



01

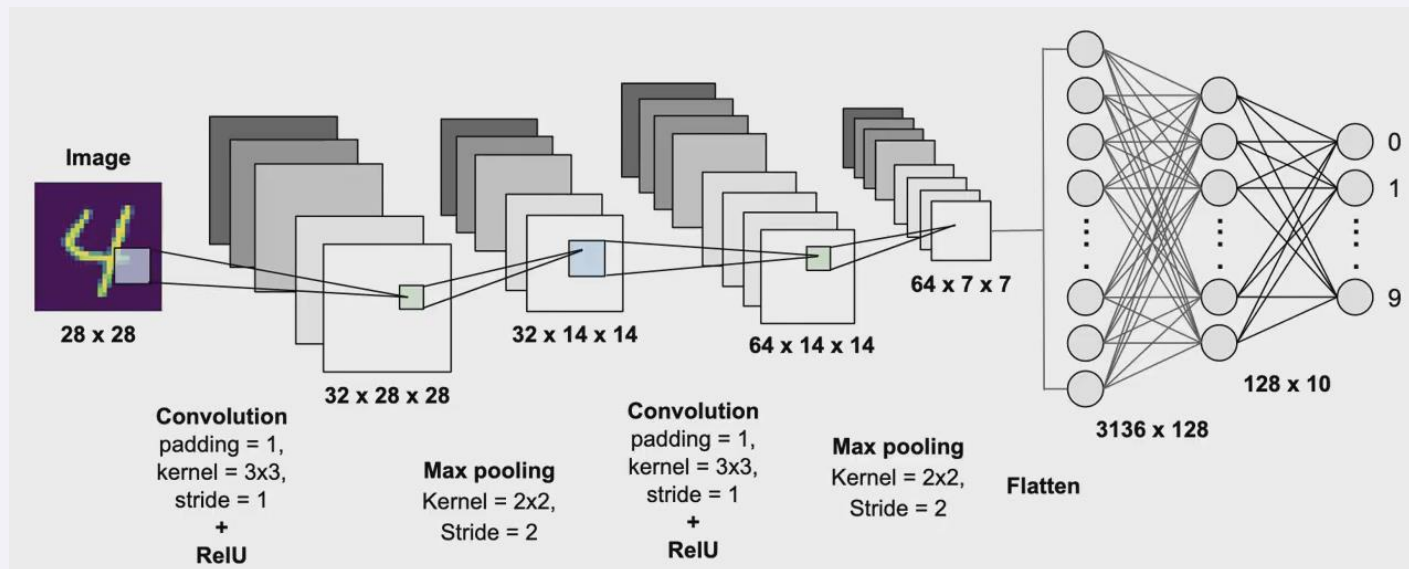
The Graph Neural Network Model

Motivation

MLP로는 image를 1-D vector data로 변환시켜 연산

→ image의 위치정보 손실!

→ Image 자체로 학습하기 위해 Convolutional Neural Network 등장

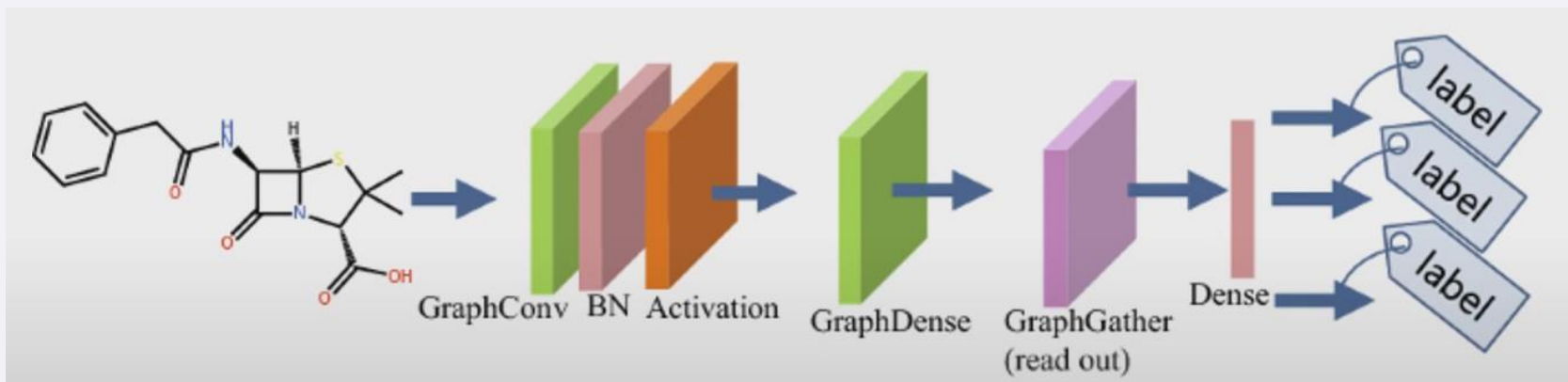


01

The Graph Neural Network Model

Motivation

Using Diffusion + RNN techniques



01

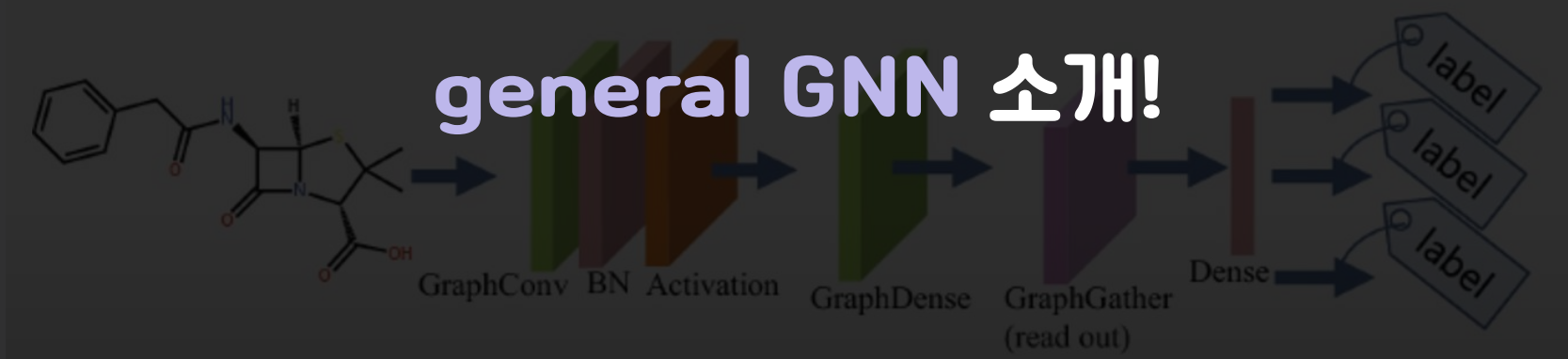
The Graph Neural Network Model

Motivation

Using Diffusion + RNN techniques

다양한 그래프 구조에 적용 가능한

general GNN 소개!



01

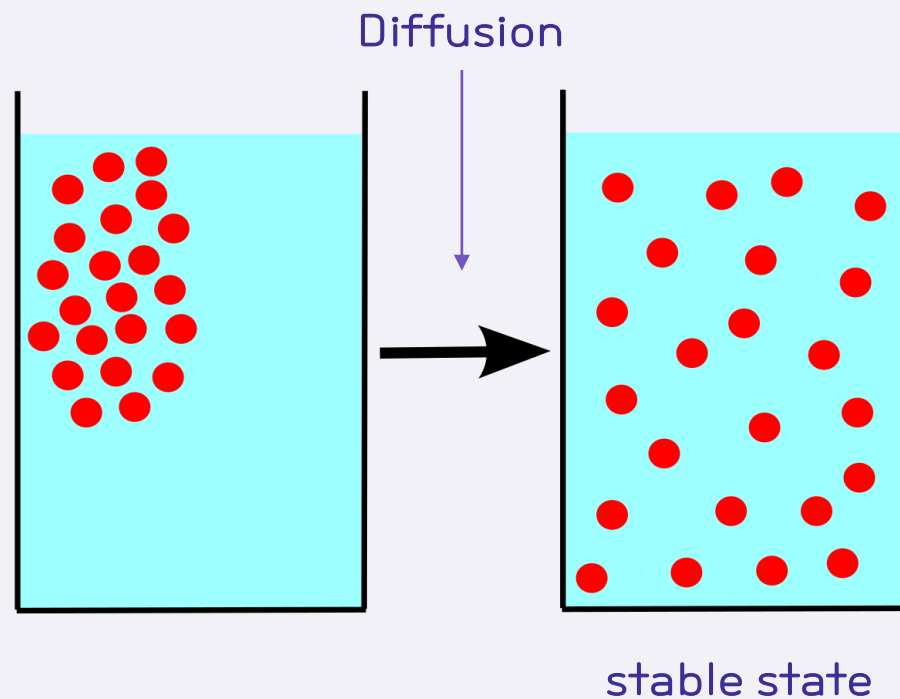
The Graph Neural Network Model

Diffusion & Relaxation Mechanism

CellularNN 과 HopfieldNN에서 이미 사용 중에 있음
state가 stable 해질 때 까지 update

The condition exists

→ 항상 같은 fixed point로 수렴해야한다!



02

The Graph Neural Network Model

Notation

$co([n]$: n 을 중심노드로 하는 edge의 집합

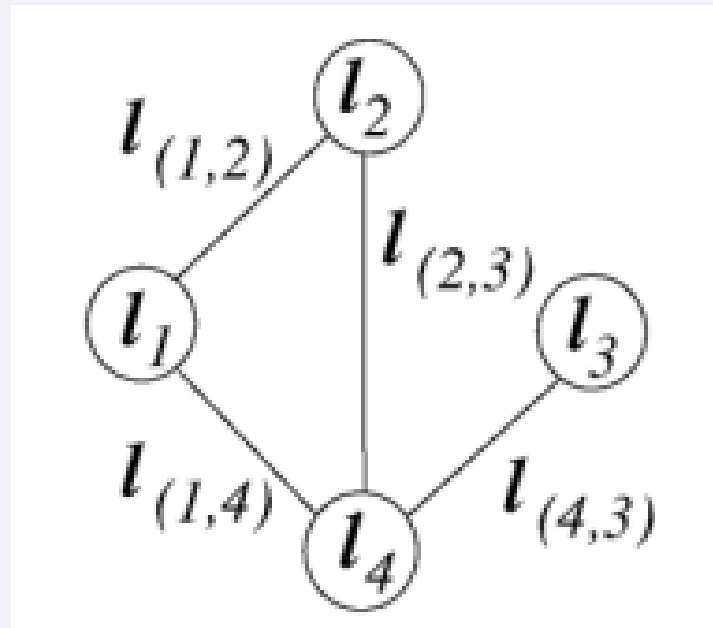
$ne[n]$: n 을 중심으로 연결되어 있는 인접 노드들의 집합

$\mathbf{l}_n \in \mathbb{R}^{l_N}$ 은 graph의 모든 label을

label은 node나 edge의 feature를

encoding 하기 전 input domain은

graph집합과 node들의 집합으로 이루어



$$\mathcal{L} = (G_i, n_{i,j}, t_{i,j}) |, G_i = (N_i, E_i) \in \mathcal{G};$$
$$n_{i,j} \in N_i; t_{i,j} \in \mathbb{R}^m, 1 \leq i \leq p, 1 \leq j \leq q_i$$

Learning set 에 대해 raw data(graph, node)와 node의

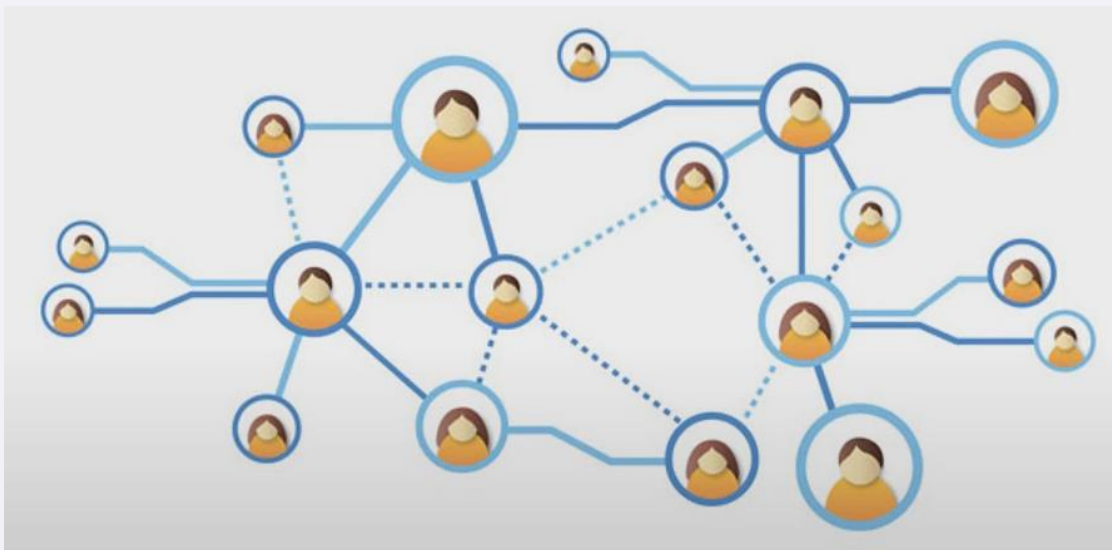
인코딩 target $t_{i,j}$ 사이의 loss를 최소한이 되도록 학습!

02

The Graph Neural Network Model

Graph Representation

Idea: graph에서 node가 객체나 한 개념을 표현하고, edge가 객체(node)간의 관계를 표현하지 않을까??



나와 친분이 있을수록
social network에서 가깝고 큰 관계를 형성한다.



나에 대해 설명하기 위해
나의 특징과 주변 사람을 정보로 사용할 수 있다.

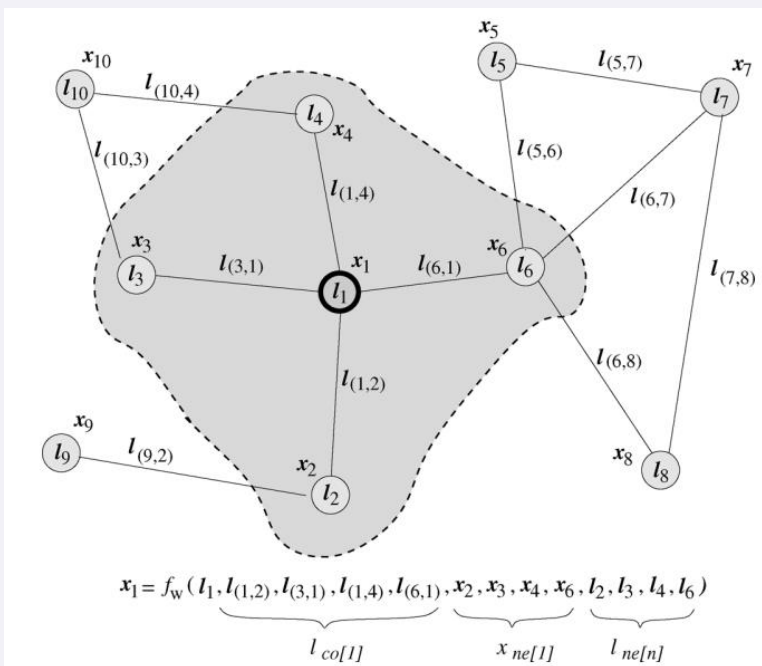
02

The Graph Neural Network Model

Graph Representation

Idea: node를 표현하기 위해 해당 node feature와 주변 정보를 이용한다!

x_n : representation of node n's concept



state transition function: f_w 을 통해

node의 state를 계산한다!



state를 이용해 output 계산!

local transition function:

$$\mathbf{x}_n = f_w(\mathbf{l}_n, \mathbf{l}_{co[n]}, \mathbf{x}_{ne[n]}, \mathbf{l}_{ne[n]})$$

local output function:

$$\mathbf{o}_n = g_w(\mathbf{x}_n, \mathbf{l}_n)$$

02

The Graph Neural Network Model

Graph Representation(example)

인접 노드의 label정보가 node state에 포함된 경우, 생략 가능

노드 간 edge가 한 개만 있는 것이 아니라 다수의 edge 존재 가능!

directed & undirected edge로 이루어진 graph 도한 구성 가능!

여러가지 방식으로 graph를 표현할 수 있다!!

Nonpositional Graph는 어떨까?

$$\mathbf{x}_n = \sum_{u \in \text{ne}[n]} h_{\mathbf{w}}(\mathbf{l}_n, \mathbf{l}_{(n,u)}, \mathbf{x}_u, \mathbf{l}_u), \quad n \in \mathbf{N}$$

로 표현 가능!

각각의 인접노드에 대해 개별적으로 연산 가능

02

The Graph Neural Network Model

Graph Representation(example)

인접 노드의 label정보가 node state에 포함된 경우, 생략 가능

노드 간 edge가 한 개만 있는 것이 아니라 다수의 edge 존재 가능!

directed & undirected edge로 이루어진 graph 도한 구성 가능!

여러가지 방식으로 graph를 표현할 수 있다!!

→ general form of various type of graph representation

k_n type의 다른 node가 있는 경우 f^{k_n} , g^{k_n} 으로 함수를 표현할 수 있고 다음과 같이 표현된다.

$\mathbf{x} = F_{\mathbf{w}}(\mathbf{x}, \mathbf{l})$: Global transition function

$\mathbf{o} = G_{\mathbf{w}}(\mathbf{x}, \mathbf{l}_N)$: Global output function

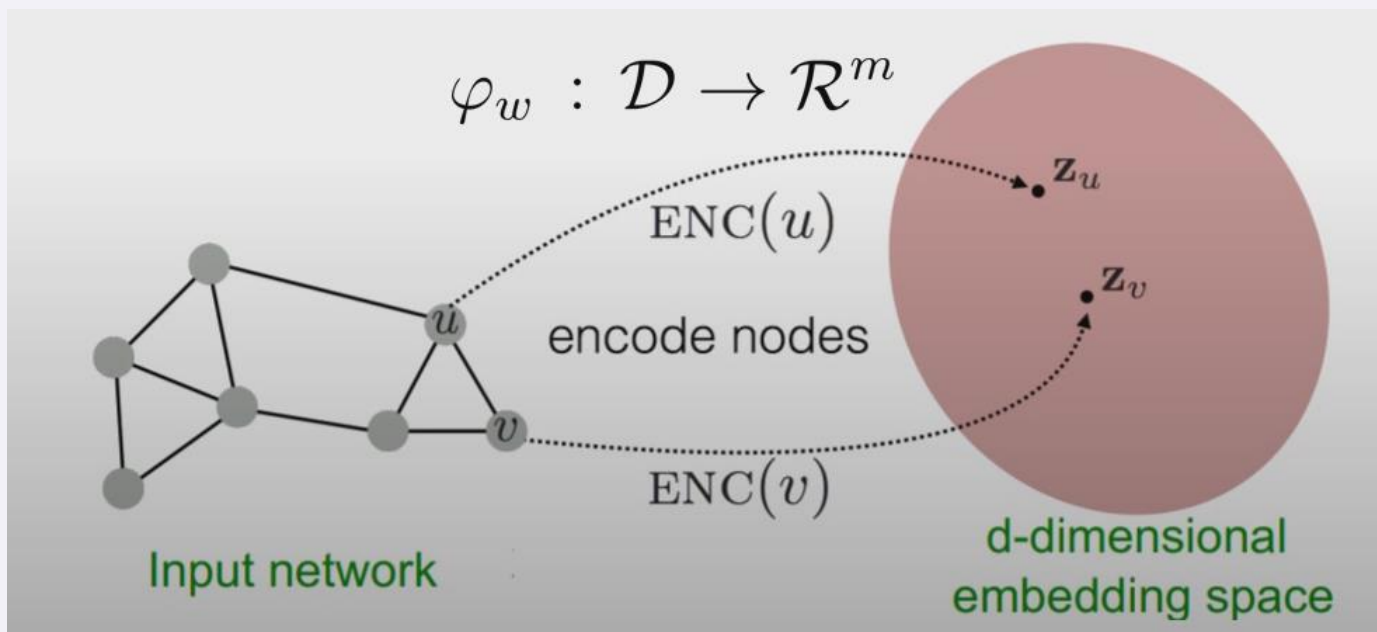
03

The Graph Neural Network Model

Encoding Condition(Banach Theorem)

output 함수는 특정 vector space에 mapping하는 함수 φ_w

전체 모델에 대해 state와 output은 유일한 값을 갖도록 해야한다.



03

The Graph Neural Network Model

Encoding Condition(Banach Theorem)

Banach theorem을 이용해 φ_w 가 contraction map으로 state를 한 점에 수렴하도록 할 때, 다음과 같은 증명이 가능

Transition function이 contraction map으로 다음과 같이 표현이 가능하다

$$x_n^t = F_w(x_n^{t-1}, l)$$

$$\|x_n^t - x_n^{t-1}\| \leq \|x_n^1 - x_n^0\|$$

다음과 같은 부등식을 세울 수 있으며

$$[0,1)에 존재하는 \mu에 대해 \|F_w(x, l) - F_w(y, l)\| \leq \mu \|x - y\|$$

으로 정리가 된다. 결국 transition 함수는 수렴한다는 것을 알 수 있다.

final state x는 stable한 상태가 된다 → 유일한 값을 가짐

One of Diffusion Mechanism

03

The Graph Neural Network Model

Encoding Condition(Banach Theorem)

Step 1. method to solve function

$$\begin{aligned}x_n &= f_w(l_n, l_{co[n]}, x_{ne[n]}, l_{ne[n]}) \\ o_n &= g_w(x_n, l_n)\end{aligned}$$

Step 2. learning algorithm

Step 3. implementation of f_w & g_w

03

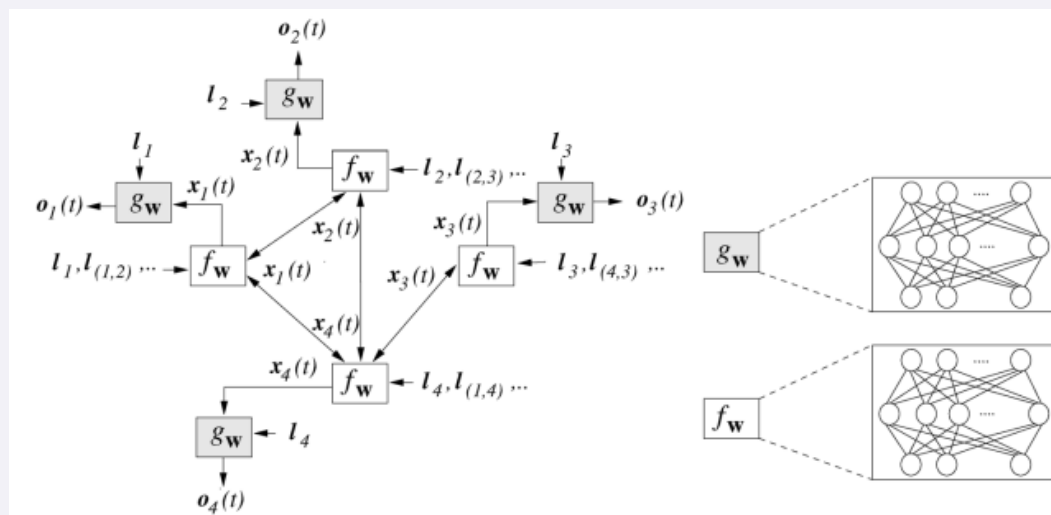
The Graph Neural Network Model

Solve function (1 step)

Banach Theorem에 의해 f_w 의 해가 존재하고, 유일하다는 것을 증명

$x(t+1) = F_w(x(t), l)$ 을 반복 계산하여 state를 얻어낸다.

아래의 연산과정 집합을 하나의 compute unit으로 취급 \rightarrow encoding network



04

The Graph Neural Network Model

Solve function (1 step)

Banach Theorem에 의해 f_w 의 해가 존재하고, 유일하다는 것을 증명

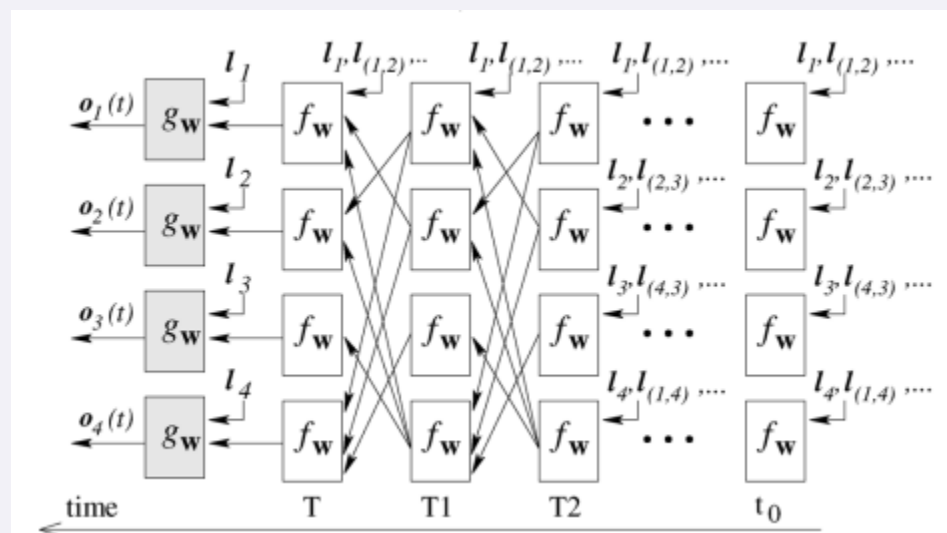
$x(t+1) = F_w(x(t), l)$ 을 반복 계산하여 state를 얻어낸다.

아래의 연산과정 집합을 하나의 compute unit으로 취급 \rightarrow encoding network

f_w & g_w 를 feedforward neural network로 구현

\rightarrow Input label은 external connection,

input state는 internal connection



04

The Graph Neural Network Model

Learning Algorithm(2 step)

estimating the parameter w

함수 φ_w 가 learning data set을 이용해 data를 approximate

Data set

$$\mathcal{L} = (G_i, n_{i,j}, t_{i,j}) |, G_i = (N_i, E_i) \in \mathcal{G}; \\ n_{i,j} \in N_i; t_{i,j} \in \mathbb{R}^m, 1 \leq i \leq p, 1 \leq j \leq q_i$$

Cost function:

$$e_w = \sum_{i=1}^p \sum_{j=1}^{q_i} (t_{i,j} - \varphi_w(G_i, n_{i,j}))^2$$

04

The Graph Neural Network Model

Learning Algorithm(2 step)

Gradient Descent flows

1. state $x_n(t)$ 가 T번 반복 update 되어 fixed point solution 에 근사하게 한다. $x(T) \approx x$
2. $\partial e_w(T) / \partial w$ 계산
3. 계산한 기울기를 이용해 w update

04

The Graph Neural Network Model

Learning Algorithm(2 step)

Limitation

1. RNN에서 사용한 backpropagation through time algorithm을 통해 각 time step에 대해 모든 기울기를 계산
2. 위의 계산을 위해 모든 time step에서의 state를 저장해야함
 - memory exceeds its storage!
 - 진행된 time step이 길수록 더 많은 메모리소요 발생

04

The Graph Neural Network Model

Almeida-Pineda Algorithm

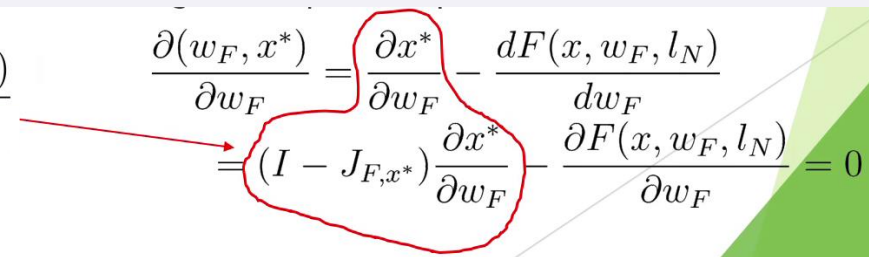
Key point: 수렴하는 마지막 state 값만 저장해서 gradient 계산!

1. Compute $x(t+1) = F(x(t), w_F, l_N)$ $x^* = F(x^*, w_F, l_N)$ $o = G(x^*, w_G, l_N)$

2. Parameter w 에 대해 이전 state와의 차이

$$\Psi(w_F, x^*) = 0 \quad \Psi(w_F, x) = x - F(x, w_F, l_N)$$

3. Continuously differentiable & converge to unique fixed point

$$J_{F, x^*} = \frac{\partial F(x^*, w_F, l_N)}{\partial x} \quad \frac{\partial(w_F, x^*)}{\partial w_F} = \frac{\partial x^*}{\partial w_F} - \frac{dF(x, w_F, l_N)}{dw_F}$$

$$= (I - J_{F, x^*}) \frac{\partial x^*}{\partial w_F} - \frac{\partial F(x, w_F, l_N)}{\partial w_F} = 0$$

04

The Graph Neural Network Model

Almeida-Pineda Algorithm

$(I - J_{F,x^*})^{-1}$ 가 존재!

4. Make invertible

$$(I - J_{F,x^*}) \frac{\partial x^*}{\partial w_F} - \frac{\partial F(x, w_F, l_N)}{\partial w_F} = 0 \longrightarrow \frac{\partial x^*}{\partial w_F} = (I - J_{F,x^*})^{-1} \frac{\partial F(x, w_F, l_N)}{\partial w_F}$$

5. Gradient of output

$$\frac{\partial L}{\partial w_F} = \frac{\partial L}{\partial o} \frac{\partial o}{\partial x^*} (I - J_{F,x^*})^{-1} \frac{\partial F(x^*, w_F, l_N)}{\partial w_F}$$

transition gradient \rightarrow store every states!

$$\frac{\partial L}{\partial w_G} = \frac{\partial L}{\partial o} \frac{\partial G(x^*, w_F, l_N)}{\partial w_G}$$

output gradient

04

The Graph Neural Network Model

Almeida-Pineda Algorithm

Using Auxiliary variable on backpropagation

6. Set auxiliary variable

$$z = (I - J_{F,x^*})^{-1} \left(\frac{\partial L}{\partial o} \frac{\partial o}{\partial x^*} \right)^\top \longrightarrow z = J_{F,x^*}^\top * z + \left(\frac{\partial L}{\partial o} \frac{\partial o}{\partial x^*} \right)^\top$$

7. Compute gradient by following task

- 1: **Initialization:** initial guess z_0 , *e.g.*, draw uniformly from $[0, 1]$, $i = 0$, threshold ϵ
- 2: **repeat**
- 3: $i = i + 1$
- 4: $z_i = J_{F,h^*}^\top z_{i-1} + \left(\frac{\partial L}{\partial y} \frac{\partial y}{\partial h^*} \right)^\top$
- 5: **until** $\|z_i - z_{i-1}\| < \epsilon$
- 6: $\frac{\partial L}{\partial w_F} = z_i^\top \frac{\partial F(x, w_F, h^*)}{\partial w_F}$
- 7: **Return** $\frac{\partial L}{\partial w_F}$

04

The Graph Neural Network Model

Almeida-Pineda Algorithm(proof)

Transition function has unique & fixed-point solution(by Banach Theorem)

Proof $(I - J_{F,x^*})^{-1}$ is invertible

1. Contraction map

$$\|J_{F,x^*}\| \leq \mu < 1$$

2. build inequality equation, $\sigma_i(J_{F,x^*})$ 은 Jacobian의 i번째 singular value이다.

$$|\det(I - J_{F,x^*})| = \prod_i |\sigma_i(I - J_{F,x^*})| \geq [1 - \sigma_{\max}(J_{F,h^*})]^d > 0$$

3. $|\det(I - J_{F,x^*})|$ 이 0이 아니므로 invertible

04

The Graph Neural Network Model

Implementation (3 step)

Output function g_w 는 multi-layer Feed Forward Network로 구현
구성이 자유롭다.

State transition function 구현이 중요하다.

→ f_w 의 해를 결정하기 때문이다

05

The Graph Neural Network Model

Linear (nonpositional) GNN

Transition function $f_w \equiv$ linear function으로 표현

$$h_w(l_n, l_{(n,u)}, x_u, l_u) = A_{n,u}x_u + b_n \quad (A_{n,u} \in R^{s \times s}, b_n \in R^s)$$

transition network: generate $A_{n,u}$

forcing network: generate b_n

Matrix defined by following equation (contraction scale $\mu \in (0,1)$)

$$\begin{aligned} A_{n,u} &= \frac{\mu}{s|\text{ne}[u]|} \cdot \Xi \quad \longrightarrow \quad \Xi = \text{resize}(\phi_w(l_n, l_{(n,u)}, l_u)) \\ b_n &= \rho_w(l_n) \quad \|\phi_w(l_n, l_{(n,u)}, l_u)\|_1 \leq s \\ &\quad \longrightarrow \quad \left\| \frac{\partial F_w}{\partial x} \right\|_1 = \|A\|_1 \leq \mu \end{aligned}$$

05

The Graph Neural Network Model

NonLinear(nonpositional) GNN

non-linear function인 경우, transition function이 $\frac{\partial F_w}{\partial x} \leq \mu$ 를 만족하기 위해

모든 파라미터를 사용하기 힘들다. → regularization을 통해 해결

다음과 같이 Ridge 규제항을 추가

$$e_w = \sum_{i=1}^p \sum_{j=1}^{q_i} (t_{i,j} - \varphi_w(G_i, n_{i,j}))^2 + \beta L(\|\frac{\partial F_w}{\partial x}\|)$$

06

The Graph Neural Network Model

Complexity

3가지 종류의 GNN model에 대한 복잡도 계산

instruction	positional	non – linear	linear	execs.
$\mathbf{z}(t+1) = \mathbf{z}(t) \cdot \mathbf{A} + \mathbf{b}$	$s^2 \mathbf{E} $	$s^2 \mathbf{E} $	$s^2 \mathbf{E} $	it _b
$\mathbf{o} = G_w(\mathbf{x}(t), \mathbf{l}_w)$	$ \mathbf{N} \overrightarrow{\mathcal{C}}_g$	$ \mathbf{N} \overrightarrow{\mathcal{C}}_g$	$ \mathbf{N} \overrightarrow{\mathcal{C}}_g$	1
$\mathbf{x}(t+1) = F_w(\mathbf{x}(t), \mathbf{l})$	$ \mathbf{N} \overrightarrow{\mathcal{C}}_f$	$ \mathbf{E} \overrightarrow{\mathcal{C}}_h$	$s^2 \mathbf{E} $ $ \mathbf{N} \overrightarrow{\mathcal{C}}_\rho + \mathbf{E} \overrightarrow{\mathcal{C}}_\phi$	it _f 1
$\mathbf{A} = \frac{\partial F_w}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{l})$	$s \mathbf{N} \overleftarrow{\mathcal{C}}_f$	$s \mathbf{E} \overleftarrow{\mathcal{C}}_h$	–	1
$\frac{\partial e_w}{\partial \mathbf{o}}$	$ \mathbf{N} $	$ \mathbf{N} $	$ \mathbf{N} $	1
$\frac{\partial p_w}{\partial \mathbf{w}}$	$t_R \cdot \max(s^2 \cdot \text{hi}_f, \overleftarrow{\mathcal{C}}_f)$	$t_R \cdot \max(s^2 \cdot \text{hi}_h, \overleftarrow{\mathcal{C}}_h)$	–	1
$\mathbf{b} = \frac{\partial e_w}{\partial \mathbf{o}} \frac{\partial G_w}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{l}_N)$	$ \mathbf{N} \overleftarrow{\mathcal{C}}_g$	$ \mathbf{N} \overleftarrow{\mathcal{C}}_g$	$ \mathbf{N} \overleftarrow{\mathcal{C}}_g$	1
$\mathbf{c} = \frac{\partial e_w}{\partial \mathbf{o}} \frac{\partial G_w}{\partial \mathbf{w}}(\mathbf{x}, \mathbf{l}_N)$	$ \mathbf{N} \overleftarrow{\mathcal{C}}_g$	$ \mathbf{N} \overleftarrow{\mathcal{C}}_g$	$ \mathbf{N} \overleftarrow{\mathcal{C}}_g$	1
$\mathbf{d} = \mathbf{z}(t) \frac{\partial F_w}{\partial \mathbf{w}}(\mathbf{x}, \mathbf{l})$	$ \mathbf{N} \overleftarrow{\mathcal{C}}_f$	$ \mathbf{E} \overleftarrow{\mathcal{C}}_h$	$ \mathbf{N} \overleftarrow{\mathcal{C}}_\rho + \mathbf{E} \overleftarrow{\mathcal{C}}_\phi$	1

06

The Graph Neural Network Model

Experiment

Application

- Subgraph Matching Problem
- Mutagenesis Problem

06

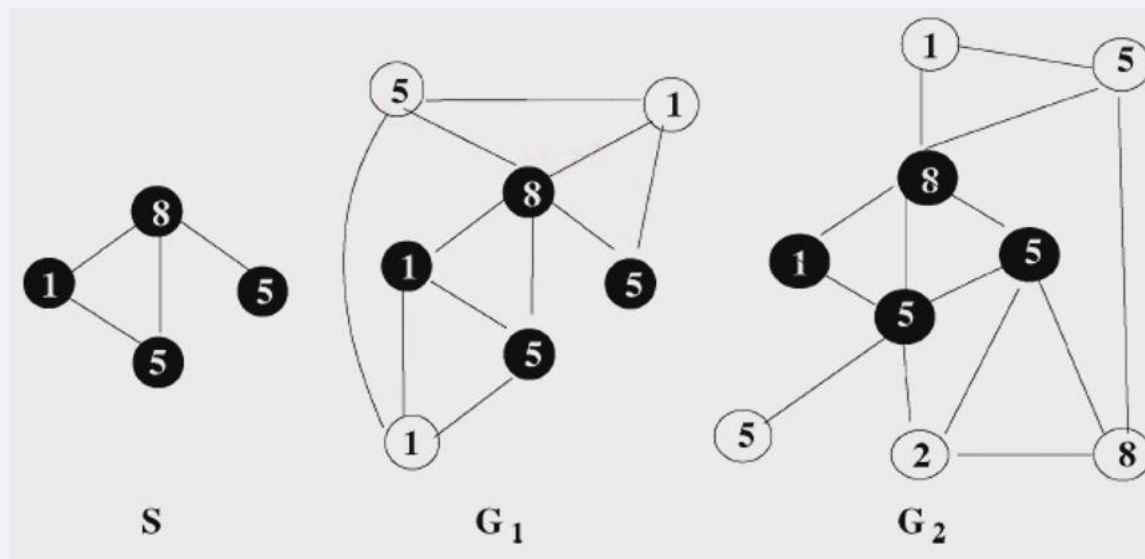
The Graph Neural Network Model

The Subgraph Matching Problem _node focused application

Goal: graph와 노드가 주어졌을 때, 해당 노드가 subgraph에 속하는 노드인지 classification

$\tau(G_i, n_{i,j})$ 이 주어졌을 때, node(i,j)가 subgraph의 노드일 때,

$\tau(G_i, n_{i,j}) = 1$, 아니면 -1을 출력



06

The Graph Neural Network Model

The Subgraph Matching Problem node focused application

Performance: NL > L > FNN

S와 G의 balance가 좋을수록 평가가 어려운 data

Subgraph와 graph간의 크기 차이가 적을수록 성능이 향상된다.

Graph G의 크기가 커질수록 평가가 어렵다

FNN보다 좋은 성능

→ graph topology와 label정보 이용을 적절히 활용할 수 있다.

Subgraph의 크기가 작을수록 GNN이 graph topology를 더 잘 학습한다. (FNN과 성능차)

TABLE III
ACCURACIES ACHIEVED BY NONLINEAR MODEL (NL), LINEAR MODEL (L), AND A FEEDFORWARD NEURAL NETWORK ON SUBGRAPH MATCHING PROBLEM

			No. of nodes in G				
			6	10	14	18	Avg.
No. of nodes in S	3	NL	92.4	90.0	90.0	84.3	89.1
		L	93.3	84.5	86.7	84.7	87.3
		FNN	81.4	78.2	79.6	82.2	80.3
	5	NL	91.3	87.7	84.9	83.3	86.8
		L	90.4	85.8	85.3	80.6	85.5
		FNN	85.2	73.2	65.2	75.5	74.8
	7	NL		89.8	84.6	79.9	84.8
		L		91.3	84.4	79.2	85.0
		FNN		84.2	66.9	64.6	71.9
	9	NL		93.3	84.0	77.8	85.0
		L		92.2	84.0	77.7	84.7
		FNN		91.6	73.7	67.0	77.4
	Avg.	NL	91.8	90.2	85.9	81.3	
		L	91.9	88.5	85.1	80.6	
		FNN	83.3	81.8	71.3	72.3	
	Total avg.	NL	87.3				
		L	86.5				
		FNN	77.2				

06

The Graph Neural Network Model

The Mutagenesis Problem Graph-focused application

10-fold cross validation

Friendly part

→ SOTA에 근접

나머지 모델도 평균적으로 높은 성능을 보임

Method	Features	Reference	Accuracy
non-linear GNN	AB+C+PS		94.3
Neural Networks	C+PS	[13]	89.0%
P-Progol	AB+C	[13]	82.0%
P-Progol	AB+C+FG	[13]	88.0%
MFLOG	AB+C	[84]	95.7%
FOIL	AB	[85]	76%
boosted-FOIL	not available	[86]	88.3%
$1nn(d_m)$	AB	[87]	83
$1nn(d_m)$	AB+C	[87]	91%
RDBC	AB	[88]	84%
RDBC	AB+C	[88]	83%
RSD	AB+C+FG	[89]	92.6%
SINUS	AB+C+FG	[89]	84.5%
RELAGGS	AB+C+FG	[89]	88.0%
RS	AB	[90]	88.9%
RS	AB+FG	[90]	89.9%
RS	AB+C+PS+FG	[90]	95.8%
SVM_P	not available	[91]	91.5

06

The Graph Neural Network Model

The Mutagenesis Problem Graph-focused application

Unfriendly part

→ Best accuracy!!

Unfriendly 한 data set 에서는 다른 모델들은 성능이 떨어짐

GNN만 성능 향상

Method	Knowledge	Reference	Accuracy
non-linear GNN	AB+C+PS		96.0%
$1nn(d_m)$	AB	[87]	72%
$1nn(d_m)$	AB+C	[87]	72%
TILDE	AB	[92]	85%
TILDE	AB+C	[92]	79%
RDBC	AB	[88]	79%
RDBC	AB+C	[88]	79%

06

The Graph Neural Network Model

The Mutagenesis Problem Graph-focused application

Whole training set

→ Still best accuracy!!

GNN을 이용해서 regression이 잘되는지 상관없이

Graph의 특징을 추출하는 것이 가능!

Method	Knowledge	Reference	Accuracy
non-linear GNN	AB+C+PS		90.5%
$1nn(d_m)$	AB	[87]	81%
$1nn(d_m)$	AB+C	[87]	88%
TILDE	AB	[92]	77%
TILDE	AB+C	[92]	82%
RDBC	AB	[88]	83%
RDBC	AB+C	[88]	82%

07

The Graph Neural Network Model

Conclusion

- GNN: model is based on information diffusion & relaxation mechanism
- Can applied to the practical applications!
- Give rise to new topics of research!
→ 패턴과 관계성이 존재하는 데이터를 처리하기 좋다
- Limitation: static domain에서만 다룸

THANK YOU.