PL / SE / ML

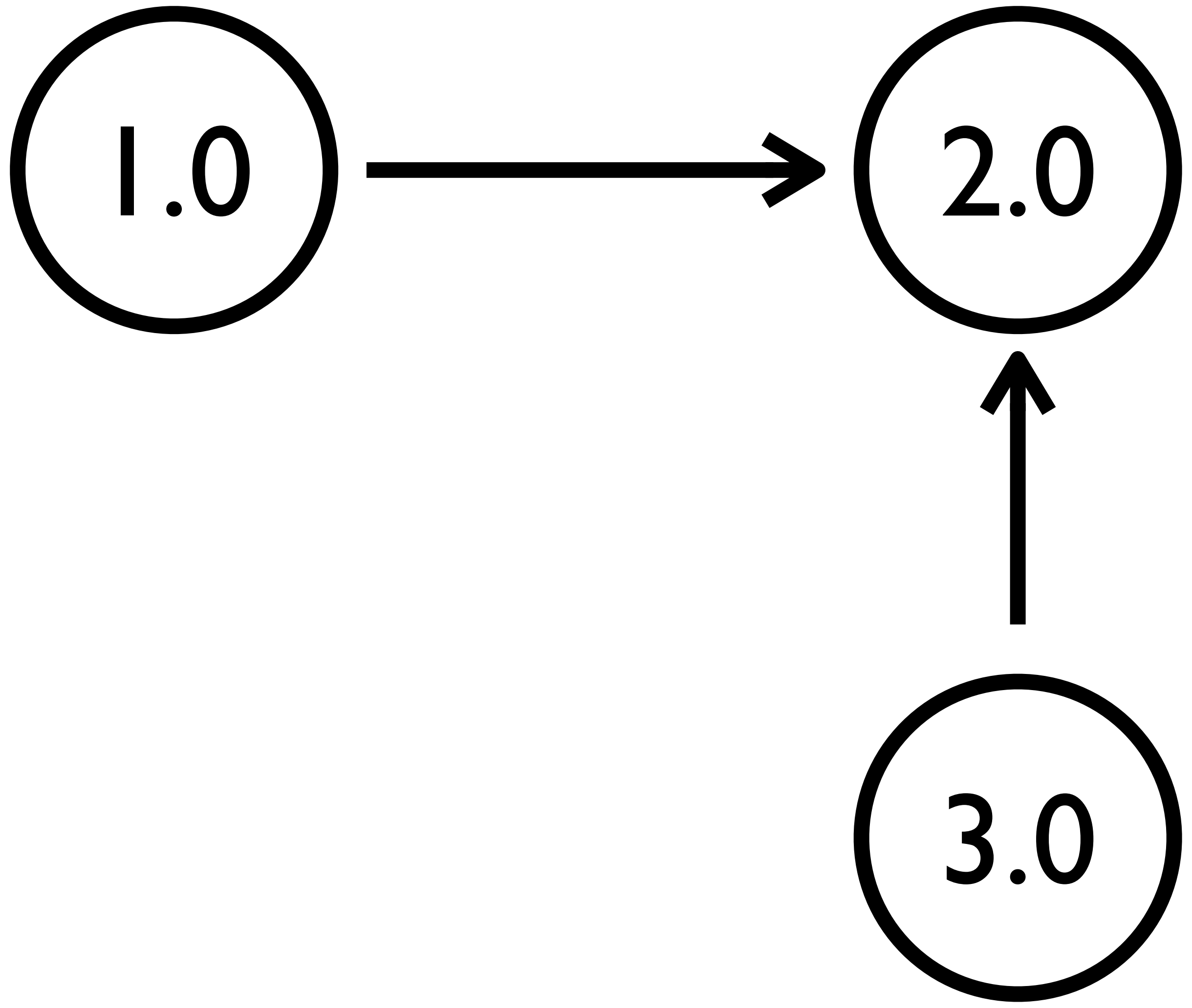
그래프 패턴 언어를 활용하여 다양한 분야의 핵심 문제 접근하기

전민석

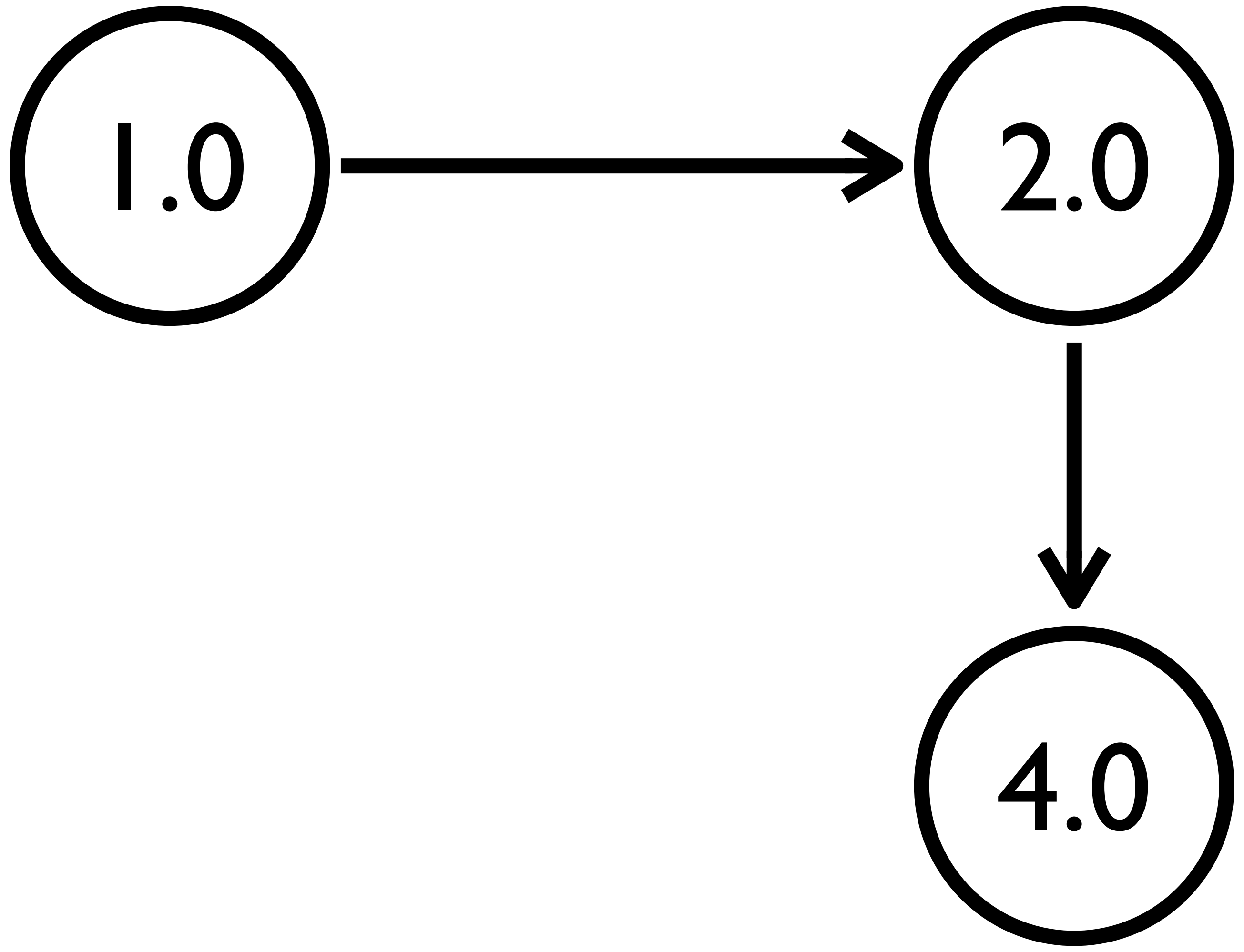
ERC Workshop @ KAIST, Korea

그래프 데이터에서의 패턴

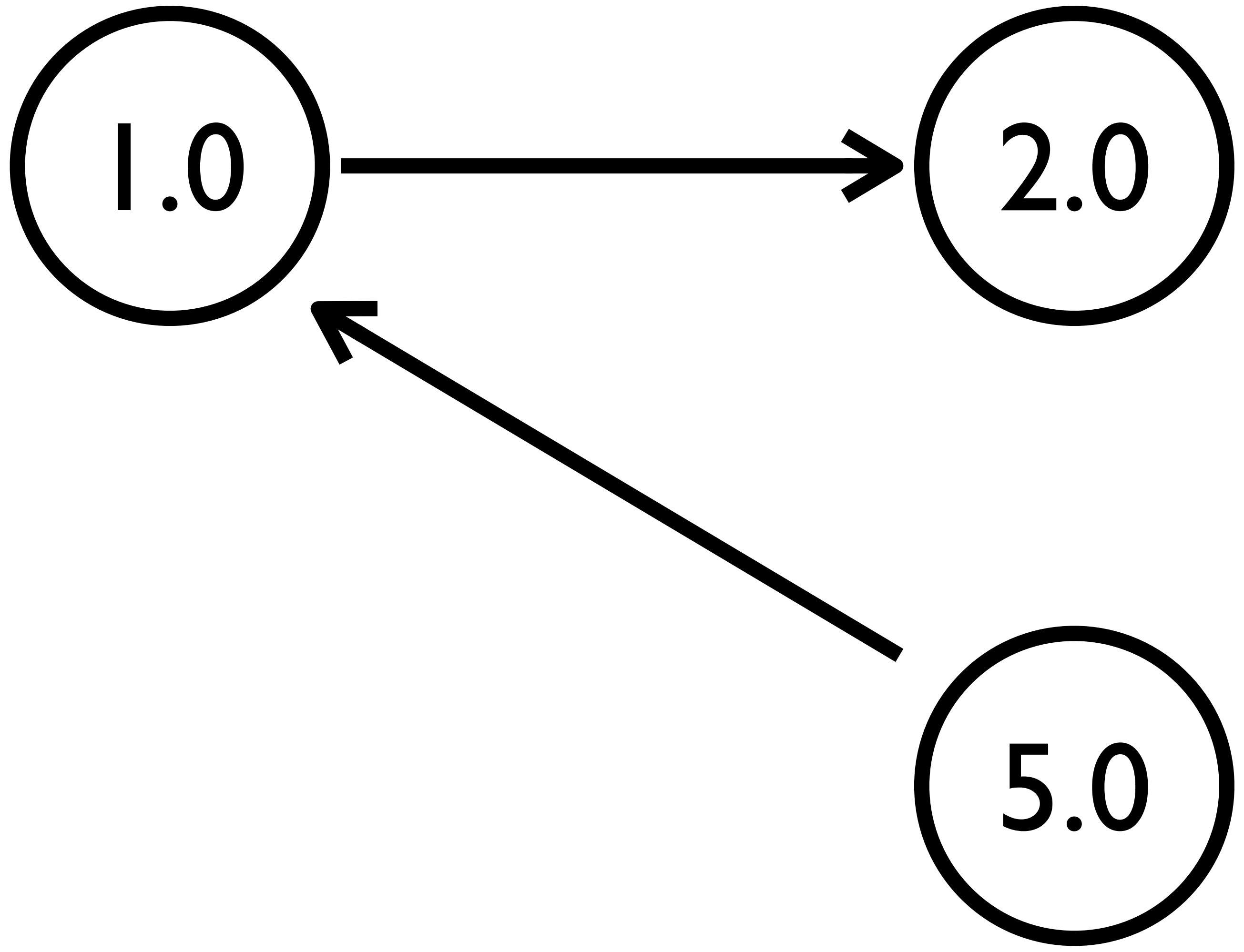
- 세 그래프 데이터의 공통된 패턴은?



그래프 1

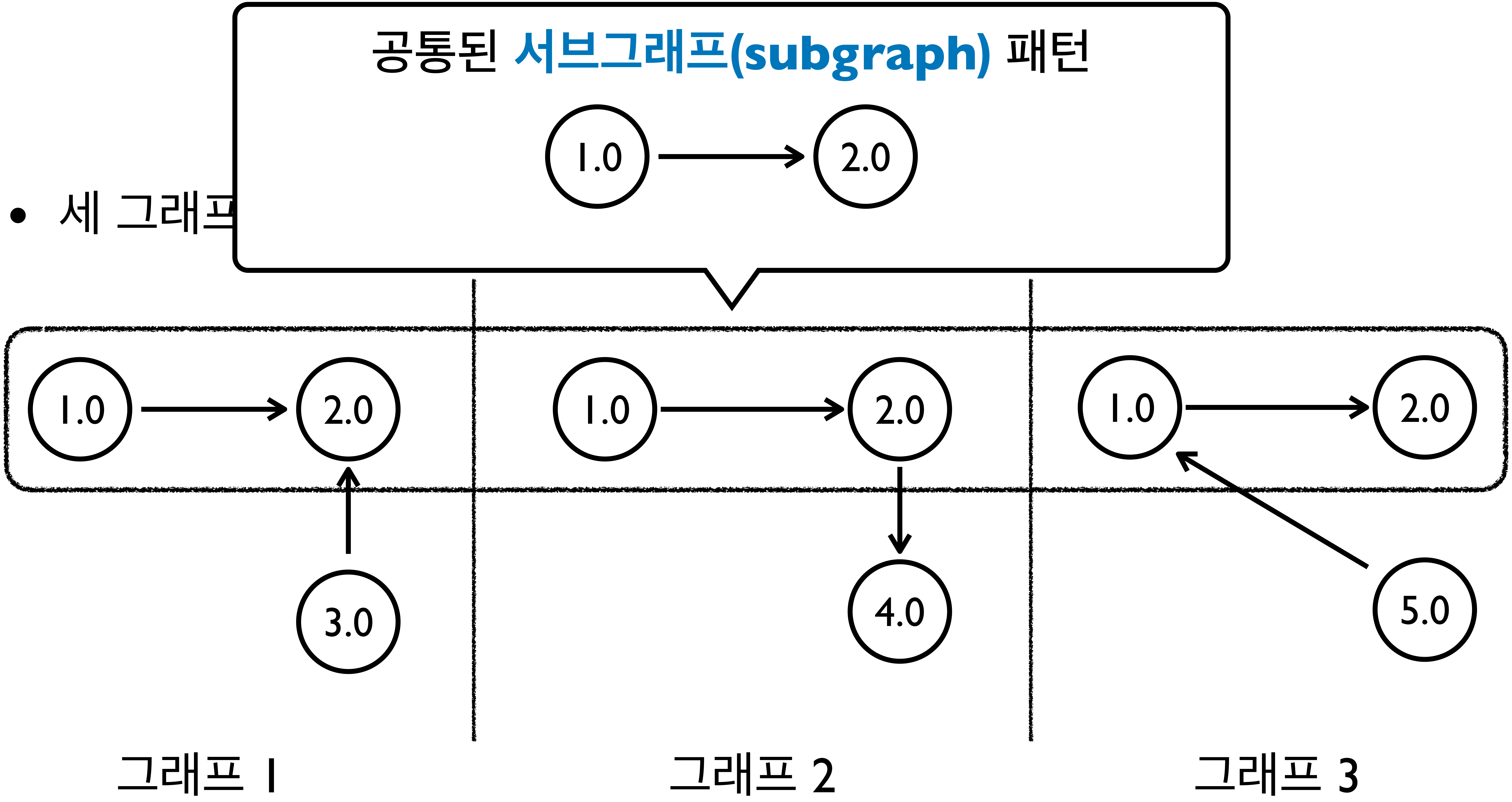


그래프 2



그래프 3

- 세 그래프



공통된 서브그래프(subgraph) 패턴
1.0
2.0
1.0
2.0
3.0
1.0
2.0
4.0
1.0
2.0
5.0
그래프 1
그래프 2
그래프 3

기존의 그래프 패턴 표현 언어: 서브그래프

Application: 그래프 데이터 마이닝

Graph Mining Algorithms

서브그래프 마이닝 알고리즘

Inductive Logic Programming (WARMR, King et al. 2001)

- Graphs are represented by Datalog facts

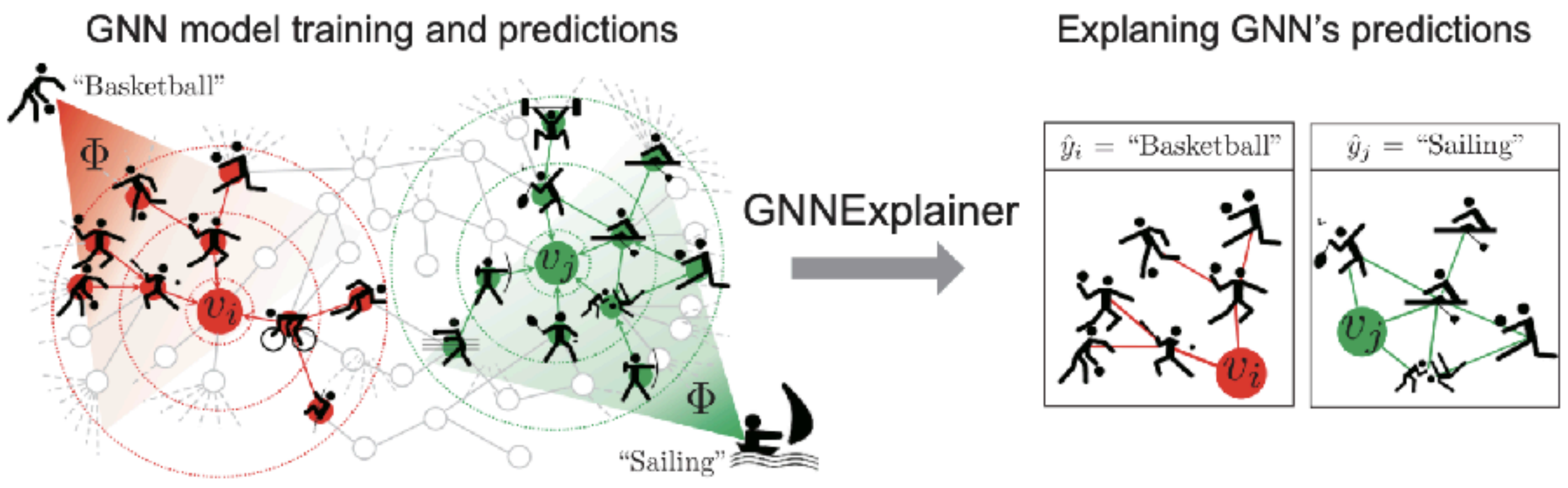
Graph Based Approaches

- Apriori-based approach
  - AGM/AcGM: Inokuchi, et al. (PKDD'00)
  - FSG: Kuramochi and Karypis (ICDM'01)
  - PATH#: Vanetik and Gudes (ICDM'02, ICDM'04)
  - FFSM: Huan, et al. (ICDM'03) and SPIN: Huan et al. (KDD'04)
  - FTOSM: Horvath et al. (KDD'06)
- Pattern growth approach
  - Subdue: Holder et al. (KDD'94)
  - MoFa: Borgelt and Berthold (ICDM'02)
  - gSpan: Yan and Han (ICDM'02)
  - Gaston: Nijssen and Kok (KDD'04)
  - CMTreeMiner: Chi et al. (TKDE'05), LEAP: Yan et al. (SIGMOD'08)

Application: GNN 설명 기술

2. GNNExplainer Overview

Subgraph

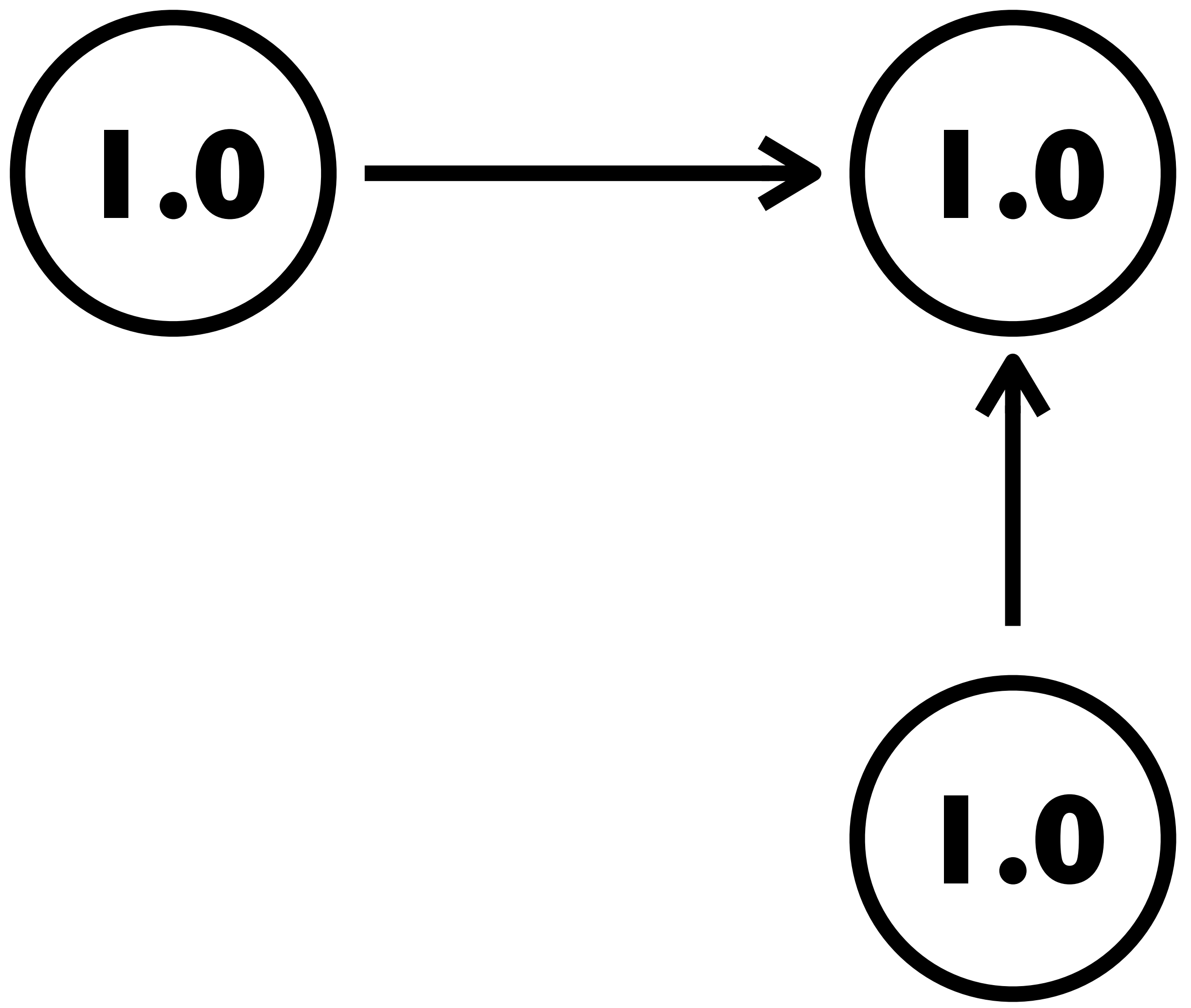


https://ethz.ch/content/dam/ethz/special-interest/bsse/borgwardt-lab/documents/slides/CA10_GraphMining.pdf

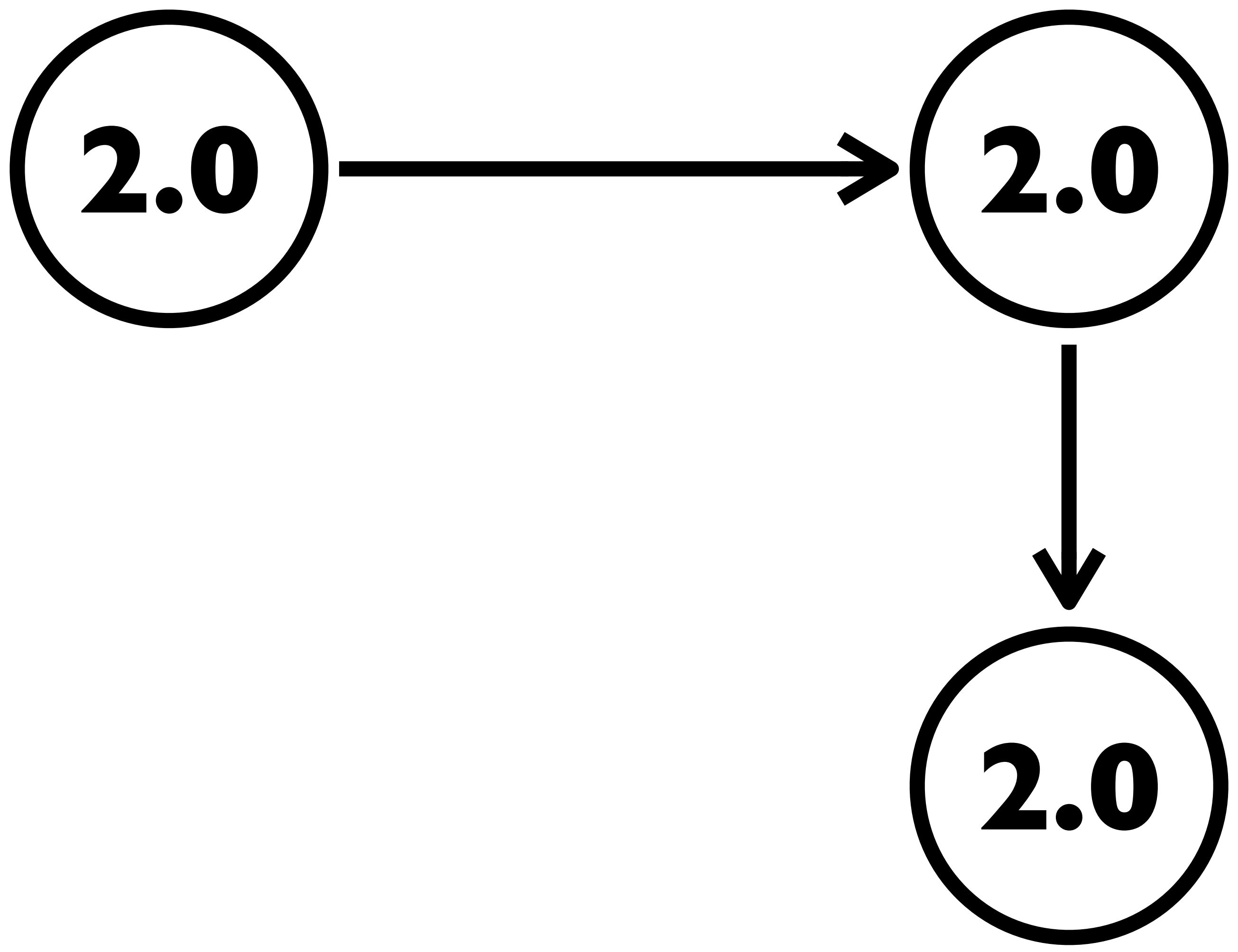
https://velog.io/@tobigs_xai/5%EC%A3%BC%EC%B0%A8-%EB%85%BC%EB%AC%B8%EB%A6%AC%EB%B7%B0-GNNExplainer-Generating-Explanations-for-Graph-Neural-Networks

서브그래프의 표현력 한계

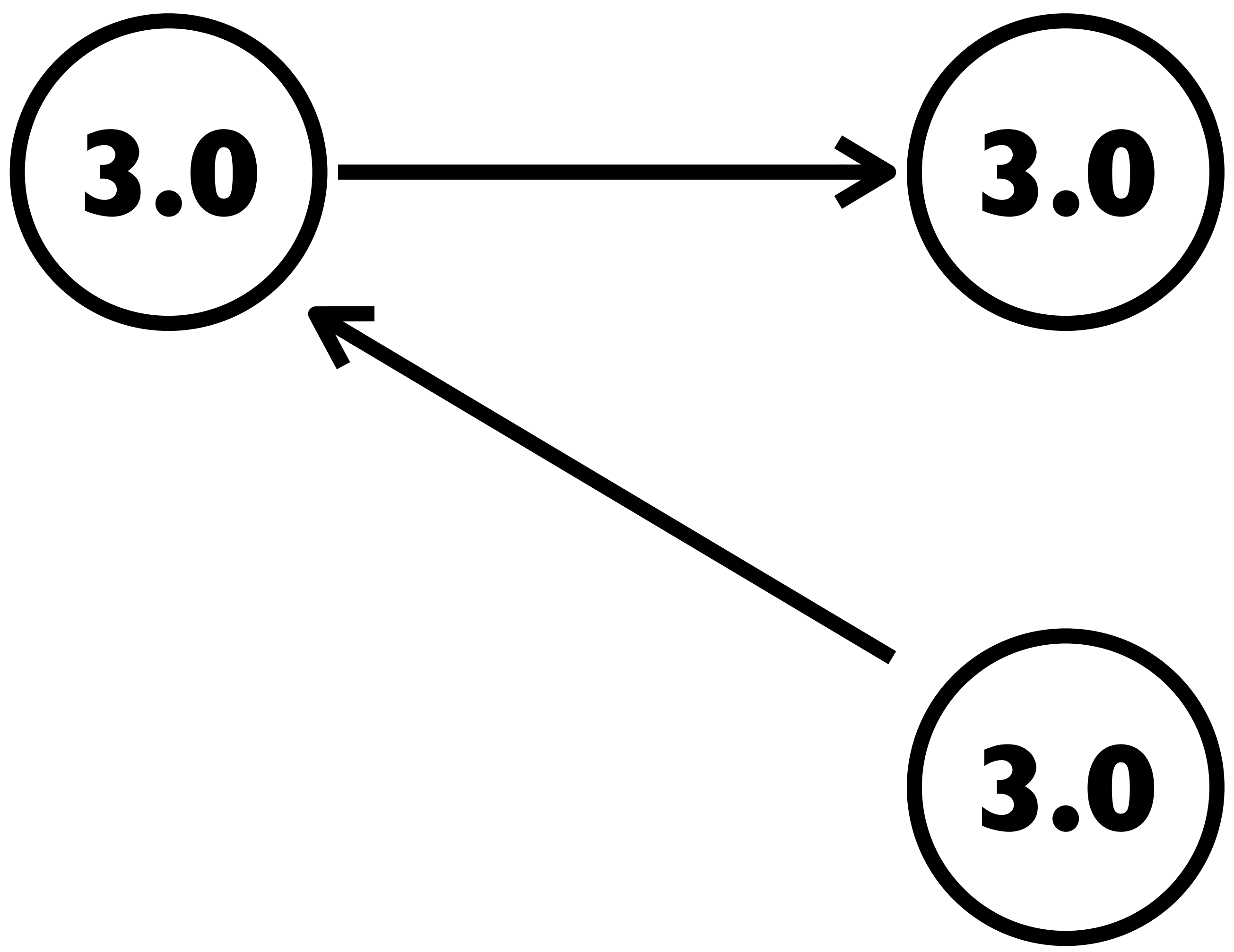
- 세 그래프 데이터의 공통된 패턴은?



그래프 1

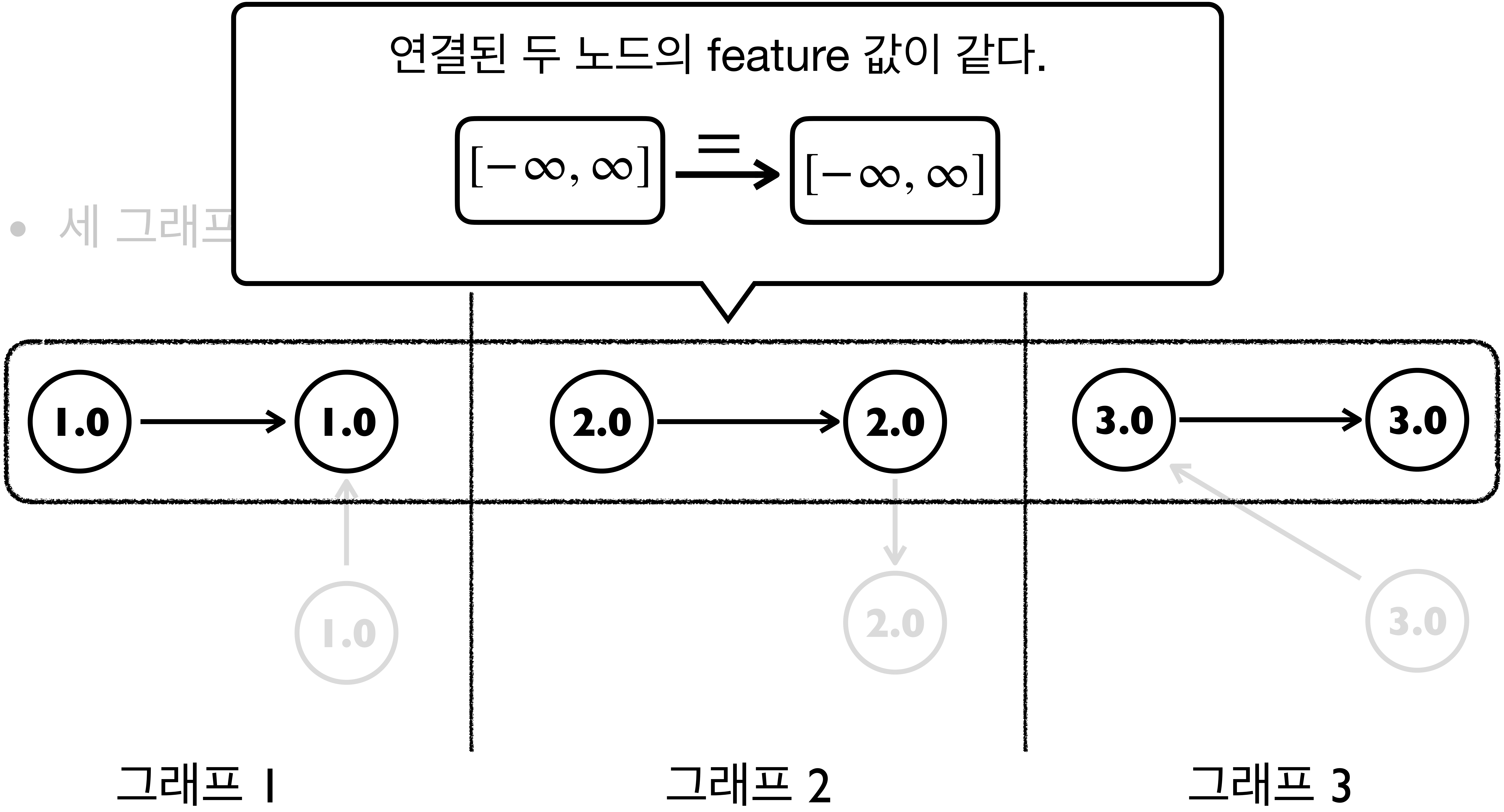


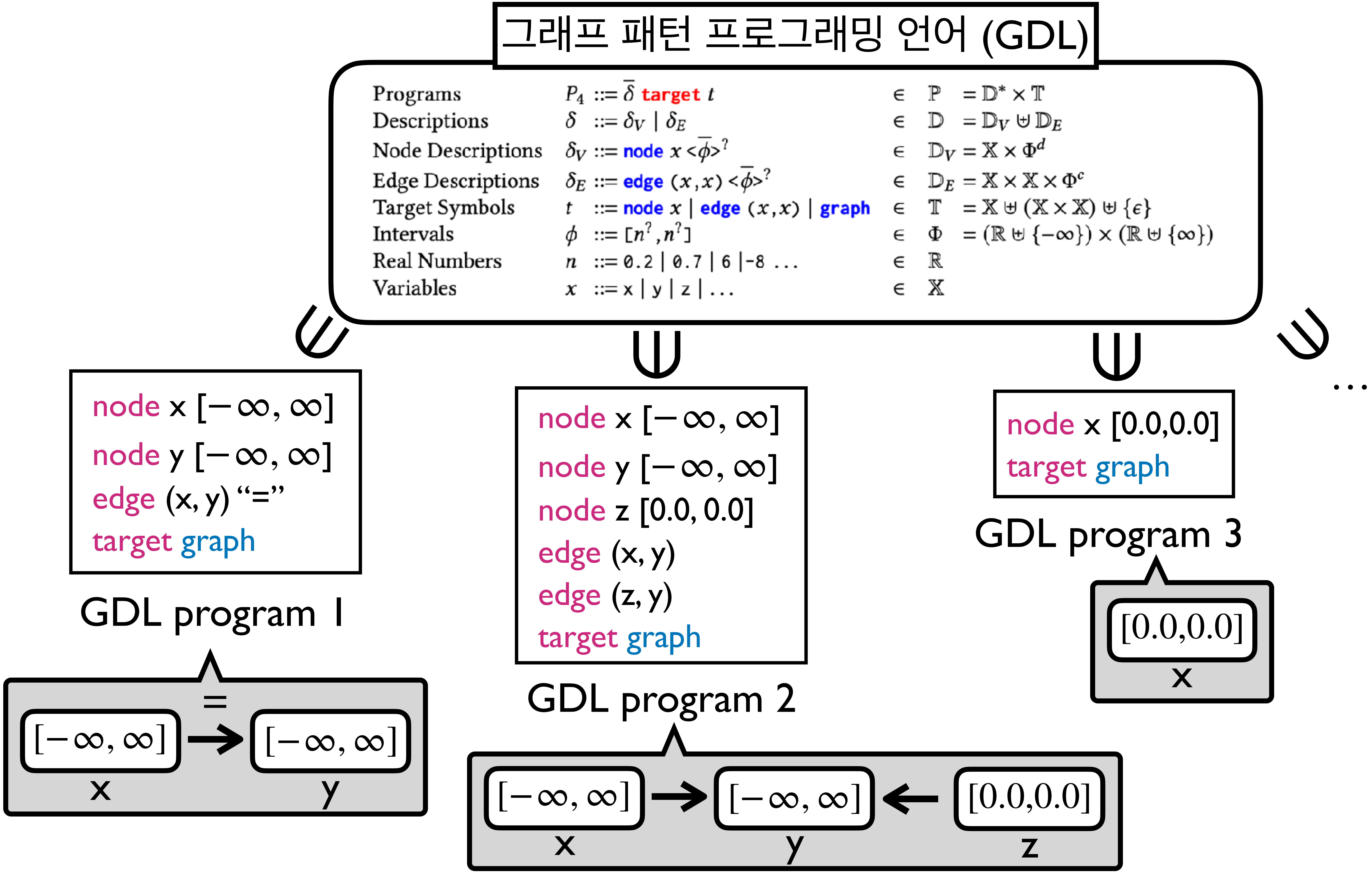
그래프 2



그래프 3

• 세 그래프





그래프 패턴 프로그래밍 언어 (GDL)
Programs $P_4 ::= \overline{\delta}$ target t $\in \mathbb{P} = \mathbb{D}^* \times \mathbb{T}$
Descriptions $\delta ::= \delta_V \mid \delta_E$ $\in \mathbb{D} = \mathbb{D}_V \uplus \mathbb{D}_E$
Node Descriptions $\delta_V ::=$ node x <$\overline{\phi}$>$^?$ $\in \mathbb{D}_V = \mathbb{X} \times \Phi^d$
Edge Descriptions $\delta_E ::=$ edge (x, x) <$\overline{\phi}$>$^?$ $\in \mathbb{D}_E = \mathbb{X} \times \mathbb{X} \times \Phi^c$
Target Symbols $t ::=$ node x | edge (x, x) | graph $\in \mathbb{T} = \mathbb{X} \uplus (\mathbb{X} \times \mathbb{X}) \uplus \{\epsilon\}$
Intervals $\phi ::= [n^?, n^?]$ $\in \Phi = (\mathbb{R} \uplus \{-\infty\}) \times (\mathbb{R} \uplus \{\infty\})$
Real Numbers $n ::=$ 0.2 | 0.7 | 6 | -8 ... $\in \mathbb{R}$
Variables $x ::=$ x | y | z | ... $\in \mathbb{X}$
node x [−∞, ∞]
node y [−∞, ∞]
edge (x, y) “=”
target graph
GDL program 1
=
[−∞, ∞]
x
[−∞, ∞]
y
node x [−∞, ∞]
node y [−∞, ∞]
node z [0.0, 0.0]
edge (x, y)
edge (z, y)
target graph
GDL program 2
[−∞, ∞]
x
[−∞, ∞]
y
[0.0,0.0]
z
node x [0.0,0.0]
target graph
GDL program 3
[0.0,0.0]
x
...

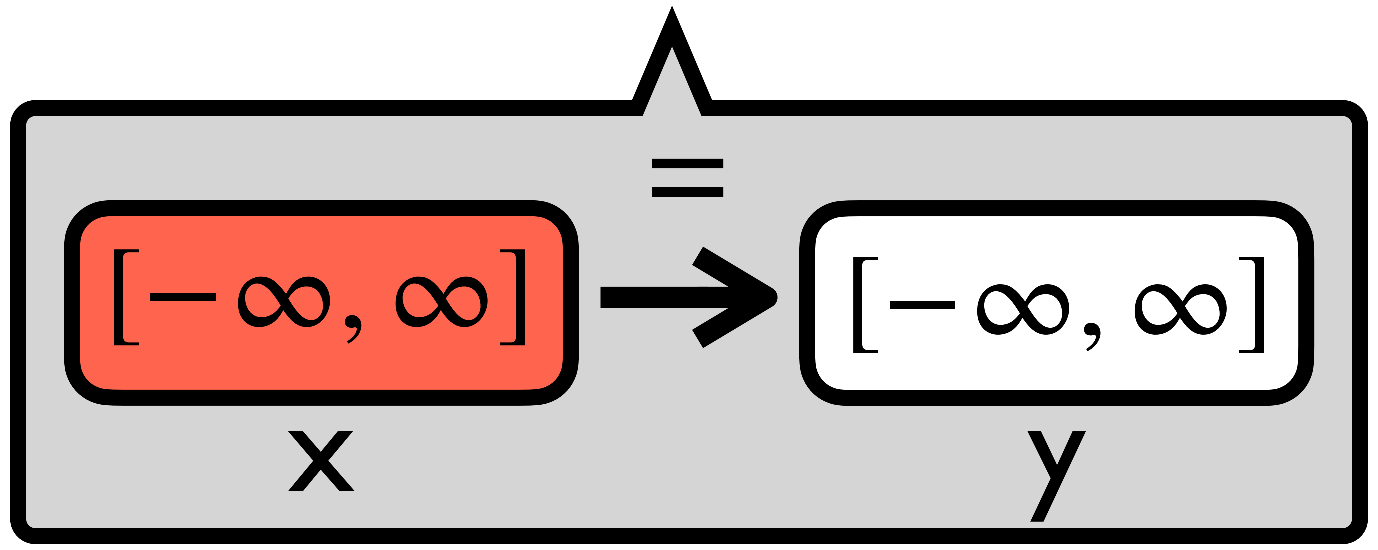
그래프 패턴 프로그래밍 언어 (GDL)

Programs	$P_4 ::= \overline{\delta}$ **target** t	$\in$	$\mathbb{P} = \mathbb{D}^* \times \mathbb{T}$
Descriptions	$\delta ::= \delta_V \mid \delta_E$	$\in$	$\mathbb{D} = \mathbb{D}_V \uplus \mathbb{D}_E$
Node Descriptions	$\delta_V ::=$ **node** x <$\overline{\phi}$>$^?$	$\in$	$\mathbb{D}_V = \mathbb{X} \times \Phi^d$
Edge Descriptions	$\delta_E ::=$ **edge** (x, x) <$\overline{\phi}$>$^?$	$\in$	$\mathbb{D}_E = \mathbb{X} \times \mathbb{X} \times \Phi^c$
Target Symbols	$t ::=$ **node** x | **edge** (x, x) | **graph**	$\in$	$\mathbb{T} = \mathbb{X} \uplus (\mathbb{X} \times \mathbb{X}) \uplus \{\epsilon\}$
Intervals	$\phi ::= [n^?, n^?]$	$\in$	$\Phi = (\mathbb{R} \uplus \{-\infty\}) \times (\mathbb{R} \uplus \{\infty\})$
Real Numbers	$n ::=$ `0.2` | `0.7` | `6` | `-8` ...	$\in$	$\mathbb{R}$
Variables	$x ::=$ x | y | z | ...	$\in$	$\mathbb{X}$

...

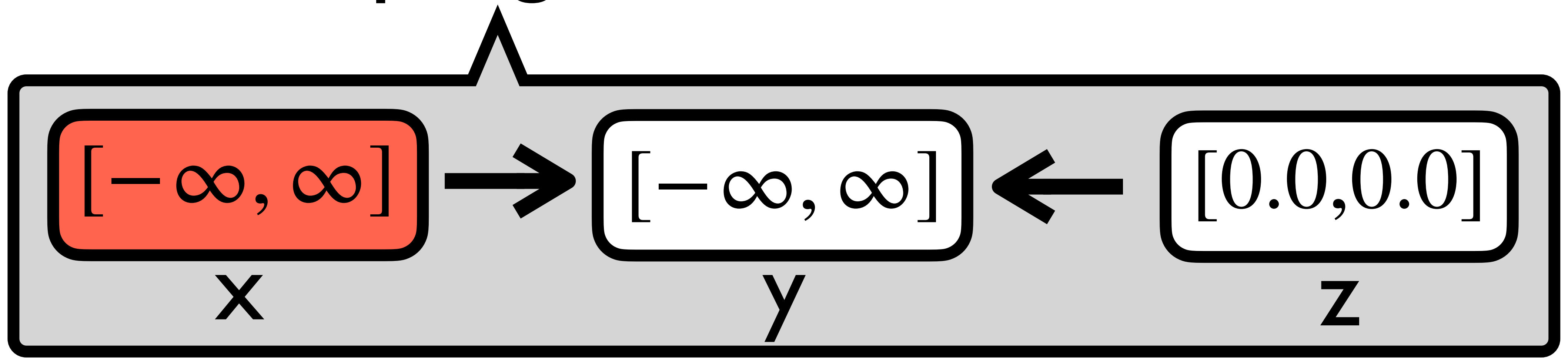
node x $[-\infty, \infty]$
node y $[-\infty, \infty]$
edge (x, y) "="
target node x

GDL program 1



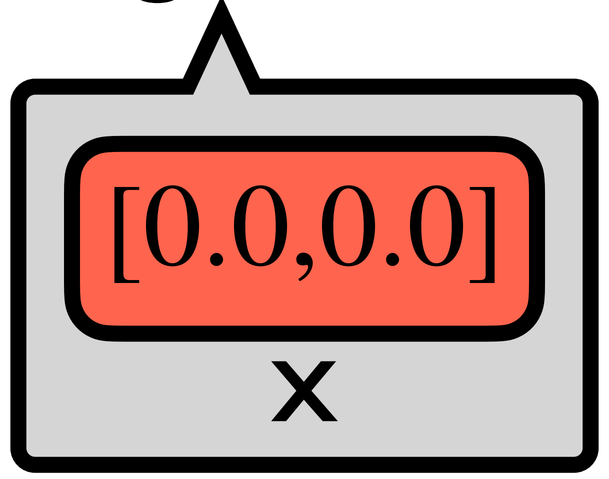
node x $[-\infty, \infty]$
node y $[-\infty, \infty]$
node z [0.0, 0.0]
edge (x, y)
edge (z, y)
target node x

GDL program 2



node x [0.0,0.0]
target node x

GDL program 3



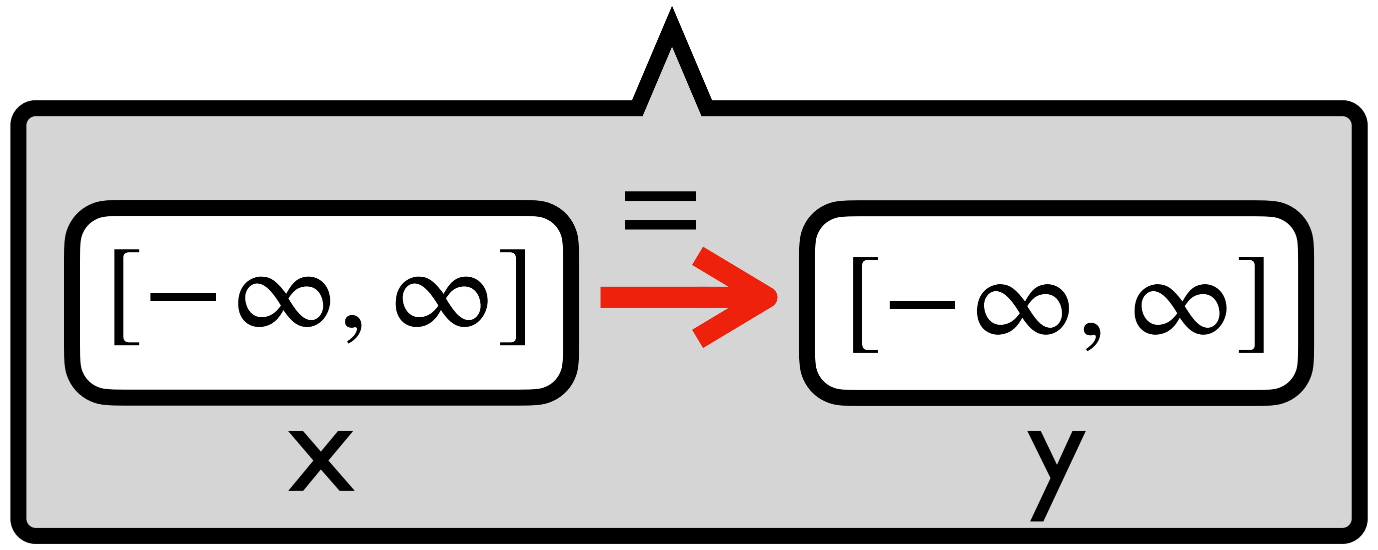
그래프 패턴 프로그래밍 언어 (GDL)

Programs	$P_4 ::= \overline{\delta}$ **target** t	$\in$	$\mathbb{P} = \mathbb{D}^* \times \mathbb{T}$
Descriptions	$\delta ::= \delta_V \mid \delta_E$	$\in$	$\mathbb{D} = \mathbb{D}_V \uplus \mathbb{D}_E$
Node Descriptions	$\delta_V ::=$ **node** x $<\overline{\phi}>^?$	$\in$	$\mathbb{D}_V = \mathbb{X} \times \Phi^d$
Edge Descriptions	$\delta_E ::=$ **edge** (x, x) $<\overline{\phi}>^?$	$\in$	$\mathbb{D}_E = \mathbb{X} \times \mathbb{X} \times \Phi^c$
Target Symbols	$t ::=$ **node** $x \mid$ **edge** $(x, x) \mid$ **graph**	$\in$	$\mathbb{T} = \mathbb{X} \uplus (\mathbb{X} \times \mathbb{X}) \uplus \{\epsilon\}$
Intervals	$\phi ::= [n^?, n^?]$	$\in$	$\Phi = (\mathbb{R} \uplus \{-\infty\}) \times (\mathbb{R} \uplus \{\infty\})$
Real Numbers	$n ::=$ `0.2` $\mid$ `0.7` $\mid$ `6` $\mid$ `-8` $\ldots$	$\in$	$\mathbb{R}$
Variables	$x ::=$ x $\mid$ y $\mid$ z $\mid \ldots$	$\in$	$\mathbb{X}$

$\ldots$

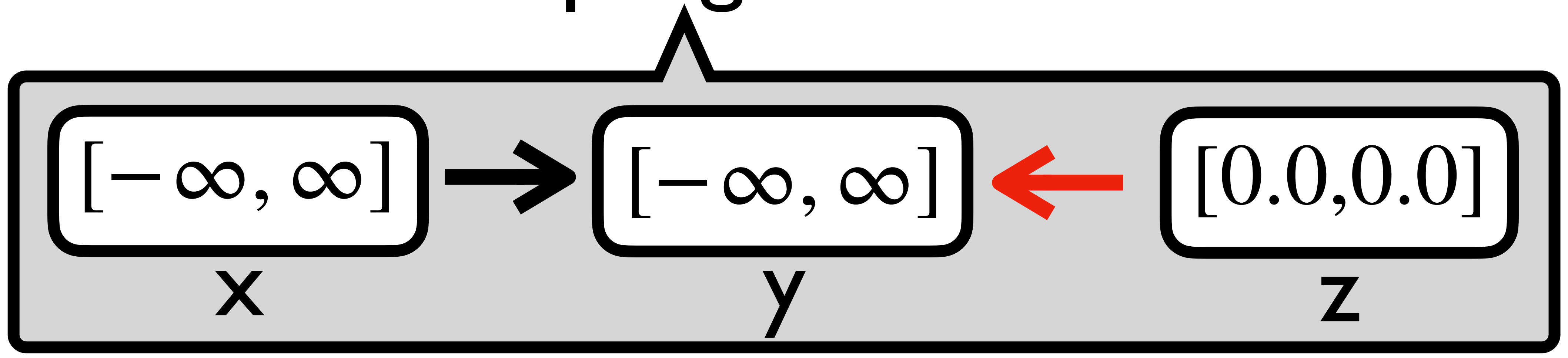
node x $[-\infty, \infty]$
node y $[-\infty, \infty]$
edge (x, y) “=”
target edge (x, y)

GDL program 1



node x $[-\infty, \infty]$
node y $[-\infty, \infty]$
node z [0.0, 0.0]
edge (x, y)
edge (z, y)
target edge (z,y)

GDL program 2



Graph Description Language (GDL) Project

- 목표: 그래프 표현 언어를 확장 및 사용하여 각 분야의 핵심 문제 해결하기

Published OOPSLA' 20
이터 기반 정적 분석을 위한 feature 자동 생성

In progress
결함 위치 추정 (Fault localization)

In progress
그래프 패턴 언어 개선

그래프 패턴 언어 (GDL)

Core language

Programs	P_4	::=	δ **target** t
Descriptions	δ	::=	$\delta_V \mid \delta_E$
Node Descriptions	δ_V	::=	**node** x <$\overline{\phi}$>$^?$
Edge Descriptions	δ_E	::=	**edge** (x, x) <$\overline{\phi}$>$^?$
Target Symbols	t	::=	**node** x | **edge** (x, x) | **graph**
Intervals	ϕ	::=	$[n^?, n^?]$
Real Numbers	n	::=	0.2 | 0.7 | 6 | -8 ...
Variables	x	::=	x | y | z | ...

This talk!

Submitted
설명 가능한 그래프 기계학습 방법

In progress
GNN을 위한 graph feature 자동 생성

...

그래프 패턴 언어를 활용한 설명 가능한 기계학습 방법

전민석

ERC Workshop @ KAIST, Korea

- 기존: 설명 불가능한 기계학습 방법 (Graph Neural Network)

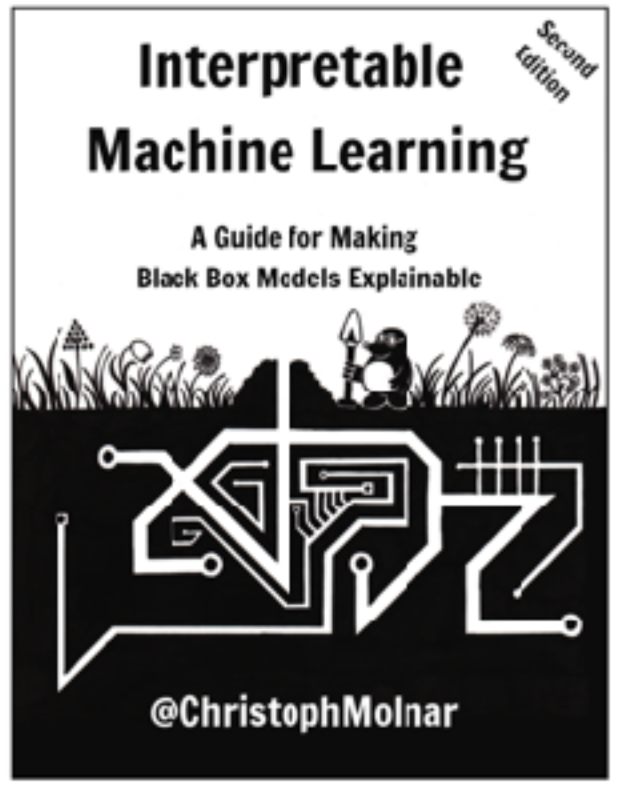
그래프 데이터
(e.g., 프로그램)
→
GNN
(Black box)
→
예측(분류) 결과
(e.g., 버그가 있음)

예측(분류)의 이유를 설명해주지 않음

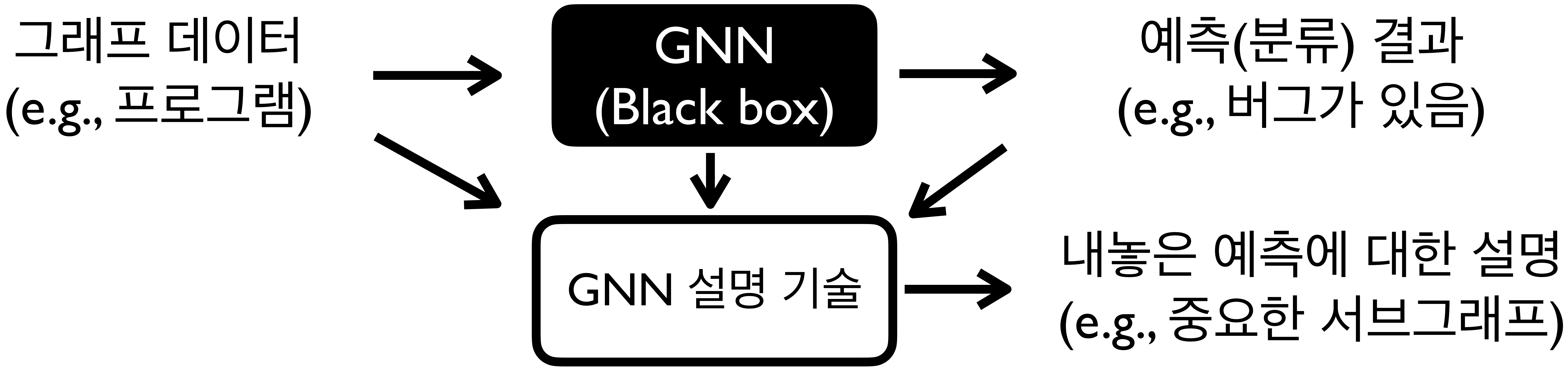
Value of explainability is growing fast

A correct prediction only partially solves your problem. The model must also explain **why.**

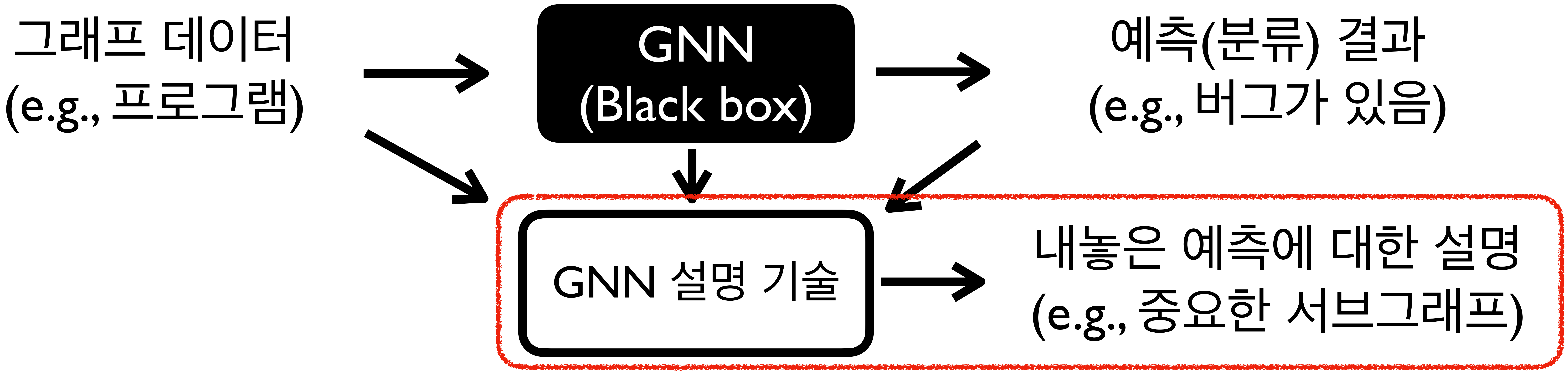
\- Molnar [**2022**]



- 기존: 설명 불가능한 기계학습 방법 (Graph Neural Network)



- 기존: 설명 불가능한 기계학습 방법 (Graph Neural Network)



근본적인 문제 두가지

- 설명을 생성하기 위해 (비싼) 추가비용을 지불해야함
- 제공된 설명이 옳은 설명임을 보장해주지 않음

- PL4XGL: 그래프 패턴 언어(GDL) 기반 설명 가능한 그래프 기계학습 방법

그래프 데이터 → Our model → 예측(분류) 결과 & 결과에 대한 옳은 설명

- 추가적인 설명 비용 필요없음
- 제공된 설명은 옳은 설명임을 보장함


예측 결과
설명
n1
n2
n4
n3
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
([−∞, 0.5] → [−∞, ∞] ,1, 0.8)
([−∞, ∞] → [−∞, 0.5], 2, 0.7)
([−∞, ∞], 1, 0.0)
n1: (1, [−∞, 0.5] → [−∞, ∞])
n2: (2, [−∞, ∞] → [−∞, 0.5])
n3: (1, [−∞, 0.5] → [−∞, ∞])
n4: (2, [−∞, ∞] → [−∞, 0.5])
그래프 데이터
분류 모델
분류 결과


= label 1
= label 2
n1
n2
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
n4
n3
그래프 데이터
([−∞, 0.5] → [−∞, ∞] , 1, 0.8)
([−∞, ∞] → [−∞, 0.5] , 2, 0.7)
([−∞, ∞] , 1, 0.0)
분류 모델
예측 결과
설명
n1: (1, [−∞, 0.5] → [−∞, ∞])
n2: (2, [−∞, ∞] → [−∞, 0.5])
n3: (1, [−∞, 0.5] → [−∞, ∞])
n4: (2, [−∞, ∞] → [−∞, 0.5])
분류 결과

n1
n2
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
n4
n3
그래프 데이터
([−∞, 0.5] → [−∞, ∞] , 1, 0.8)
([−∞, ∞] → [−∞, 0.5] , 2, 0.7)
([−∞, ∞] , 1, 0.0)
분류 모델
예측 결과
설명
n1: (1, [−∞, 0.5] → [−∞, ∞])
n2: (2, [−∞, ∞] → [−∞, 0.5])
n3: (1, [−∞, 0.5] → [−∞, ∞])
n4: (2, [−∞, ∞] → [−∞, 0.5])
분류 결과


node x ⟨[-∞,∞]⟩
node y ⟨[-∞,0.5]⟩
edge (y, x)
target node x
y
x
[−∞, 0.5]
[−∞, ∞]
Target
예측 결과
설명
n1: (1, [−∞, 0.5] [−∞, ∞])
n2: (2, [−∞, ∞] [−∞, 0.5])
⟨1.0⟩
n4
표현하고 있는 노드 패턴:
“선행 노드(predecessor)중 feature값이 0.5 이하인 노드가 존재함”
그래프 데이터
분류 모델
분류 결과

해당 패턴의 노드들은 레이블 1로 분류함
예측 결과
설명
n1
n2
n4
([−∞, 0.5] → [−∞, ∞] ,1, 0.8)
패턴의 점수는 0.8점
표현하고 있는 노드 패턴:
“선행 노드(predecessor)중 feature값이 0.5 이하인 노드가 존재함”
그래프 데이터
분류 모델
분류 결과


표현하고 있는 노드 패턴:
“선행 노드(predecessor)중 feature값이 0.5 이하인 노드가 존재함”
n1
n2
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
n4
n3
([−∞, 0.5] → [−∞, ∞] , 1, 0.8)
([−∞, ∞] → [−∞, 0.5] , 2, 0.7)
([−∞, ∞] , 1, 0.0)
예측 결과
설명
n1: (1, [−∞, 0.5] → [−∞, ∞])
n2: (2, [−∞, ∞] → [−∞, 0.5])
n3: (1, [−∞, 0.5] → [−∞, ∞])
n4: (2, [−∞, ∞] → [−∞, 0.5])
그래프 데이터
분류 모델
분류 결과


표현하고 있는 노드 패턴:
“후속 노드(successor)중 feature값이 0.5 이하인 노드가 존재함”
예측 결과
설명
n1
n2
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
n4
n3
([−∞, 0.5] → [−∞, ∞] , 1, 0.8)
([−∞, ∞] → [−∞, 0.5] , 2, 0.7)
([−∞, ∞] , 1, 0.0)
n1: (1, [−∞, 0.5] → [−∞, ∞])
n2: (2, [−∞, ∞] → [−∞, 0.5])
n3: (1, [−∞, 0.5] → [−∞, ∞])
n4: (2, [−∞, ∞] → [−∞, 0.5])
그래프 데이터
분류 모델
분류 결과

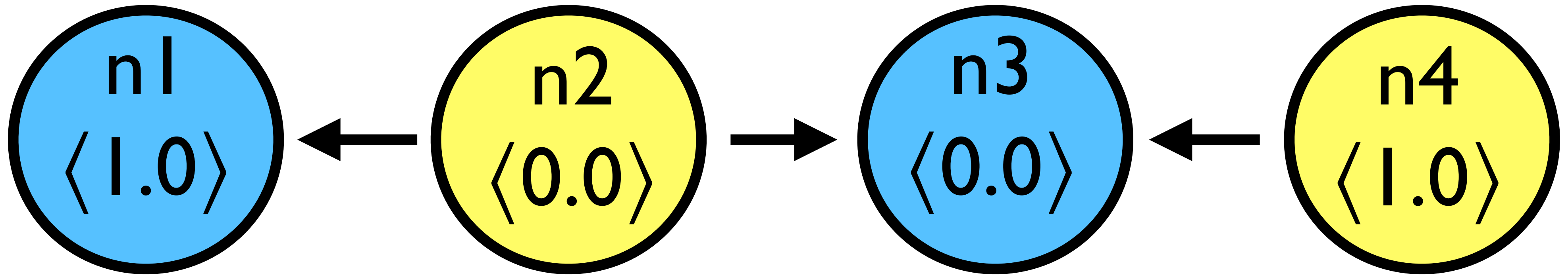

예측 결과
설명
n1
n2
n3
n4
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
([−∞, 0.5] → [−∞, ∞] , 1, 0.8)
([−∞, ∞] → [−∞, 0.5] , 2, 0.7)
([−∞, ∞] , 1, 0.0)
n1: (1, [−∞, 0.5] → [−∞, ∞])
n2: (2, [−∞, ∞] → [−∞, 0.5])
n3: (1, [−∞, 0.5] → [−∞, ∞])
n4: (2, [−∞, ∞] → [−∞, 0.5])
패턴이 겹칠 경우 더 높은 점수의 패턴으로 분류
그래프 데이터
분류 결과

- 추가적인 설명 비용 필요없음
- 제공된 설명은 옳은 설명임을 보장함


학습 데이터
GDL 프로그램
합성 알고리즘
n1
n2
n4
n3
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
([−∞, 0.5] → [−∞, ∞] , 1, 0.8)
([−∞, ∞] → [−∞, 0.5] , 2, 0.7)
([−∞, ∞] , 1, 0.0)
n1: (1, [−∞, 0.5] → [−∞, ∞])
n2: (2, [−∞, ∞] → [−∞, 0.5])
n3: (1, [−∞, 0.5] → [−∞, ∞])
n4: (2, [−∞, ∞] → [−∞, 0.5])
그래프 데이터
분류 모델
분류 결과

하향식(Top-down) GDL 프로그램 합성 알고리즘

학습 데이터(그래프) :



타깃 레이블 = 1 ( )

(1) 가장 일반적인 패턴(간단한 GDL 프로그램)에서 부터 시작

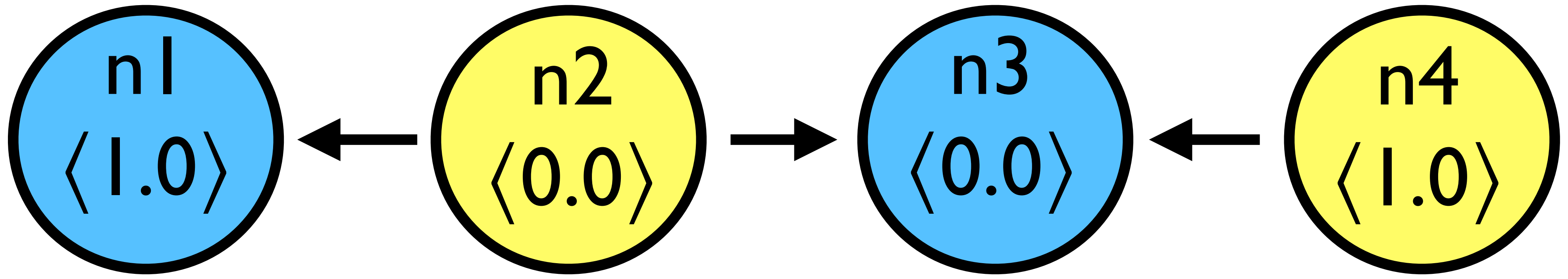
$[-\infty, \infty]$

Score : 0.5

$$\frac{|\{n1,n3\}|}{|\{n1,n2,n3,n4\}|}$$

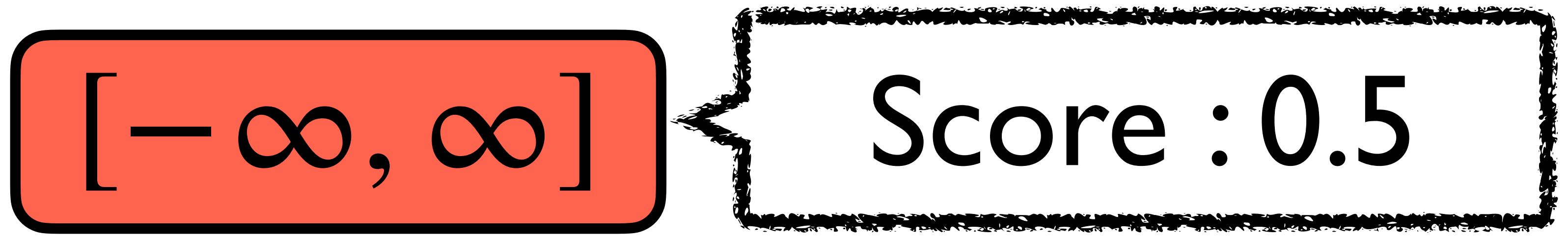
하향식(Top-down) GDL 프로그램 합성 알고리즘

학습 데이터(그래프) :

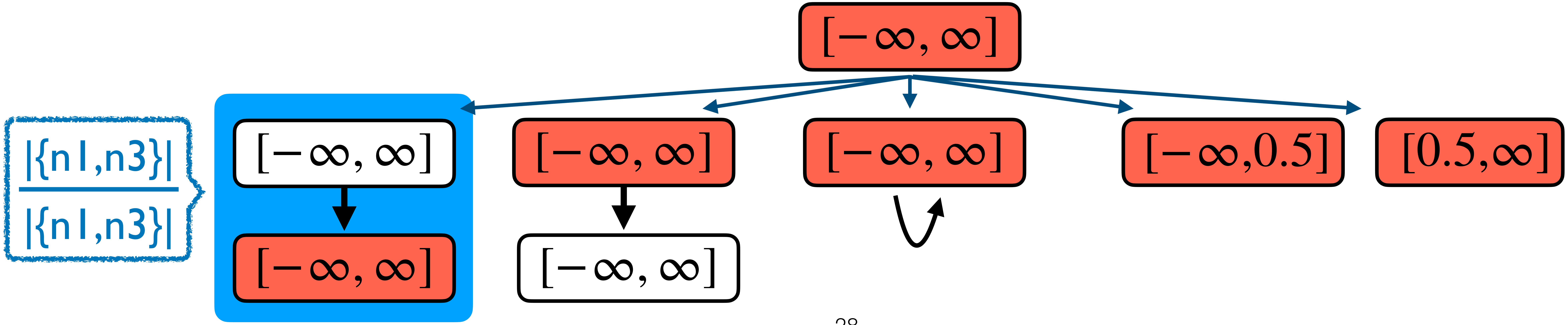


타깃 레이블 = 1 ( )

(1) 가장 일반적인 패턴(간단한 GDL 프로그램)에서 부터 시작

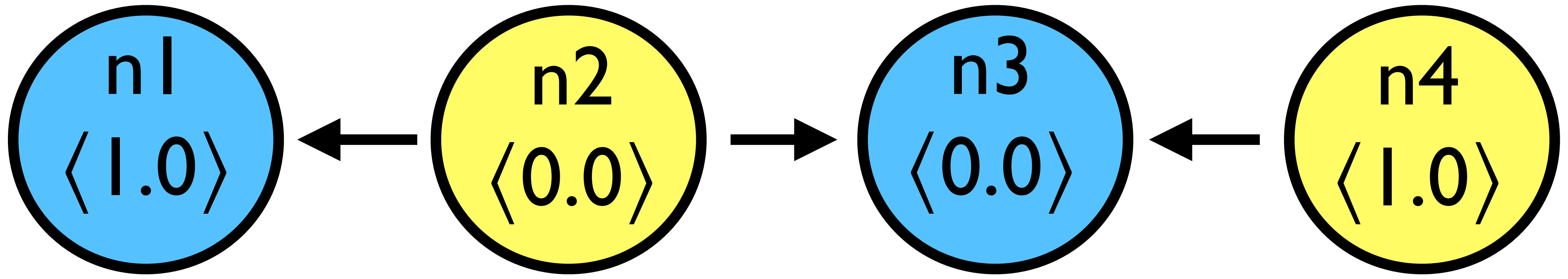


(2) 주어진 패턴을 다양하게 specify하여 나열 후 가장 높은 점수의 패턴을 선택 (나열 탐색)



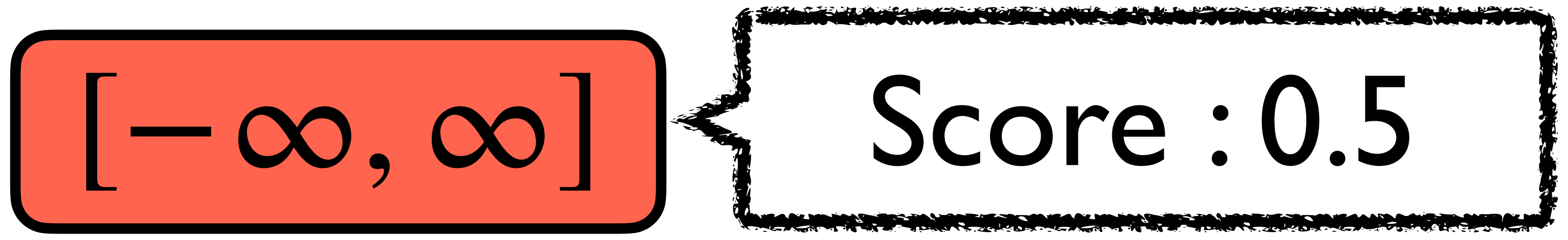
하향식(Top-down) GDL 프로그램 합성 알고리즘

학습 데이터(그래프) :

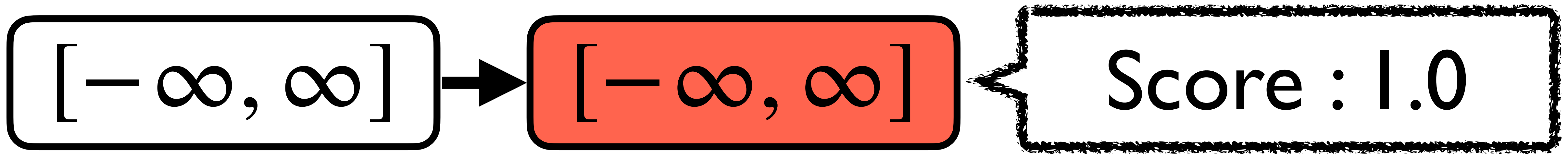


타깃 레이블 = 1 ( )

(1) 가장 일반적인 패턴(간단한 GDL 프로그램)에서 부터 시작



(2) 주어진 패턴을 다양하게 specify하여 나열 후 가장 높은 점수의 패턴을 선택 (나열 탐색)



(3) 모든 나열된 패턴이 현재 패턴보다 같거나 낮은 점수를 가질 때 까지 (2)를 반복

하향식(Top-down) GDL 프로그램 합성 알고리즘

학습 데이터(그래프) : n1 $\langle 1.0 \rangle$ ← n2 $\langle 0.0 \rangle$ → n3 $\langle 0.0 \rangle$ ← n4 $\langle 1.0 \rangle$

타깃 레이블 = 1 (○)

(1) 가장 일반적인 패턴(간단한 GDL 프로그램)에서 부터 시작

$[-\infty, \infty]$ Score : 0.5

(2) 주어진 패턴을 다양하게 specify하여 나열 후 가장 높은 점수의 패턴을 선택 (나열 탐색)

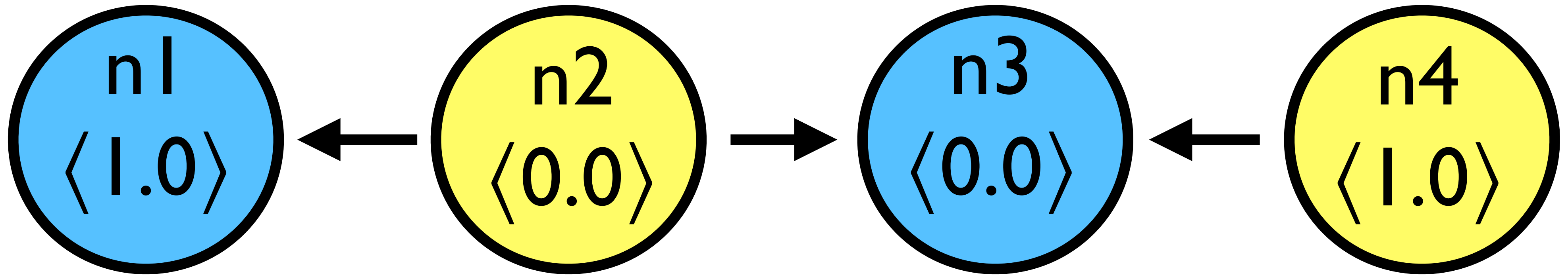
$[-\infty, \infty]$ → $[-\infty, \infty]$ Score : 1.0

(3) 모든 나열된 패턴이 현재 패턴보다 같거나 낮은 점수를 가질 때 까지 (2)를 반복

(4) 현재 패턴을 반환 ($[-\infty, \infty]$ → $[-\infty, \infty]$, label 1, 1.0)

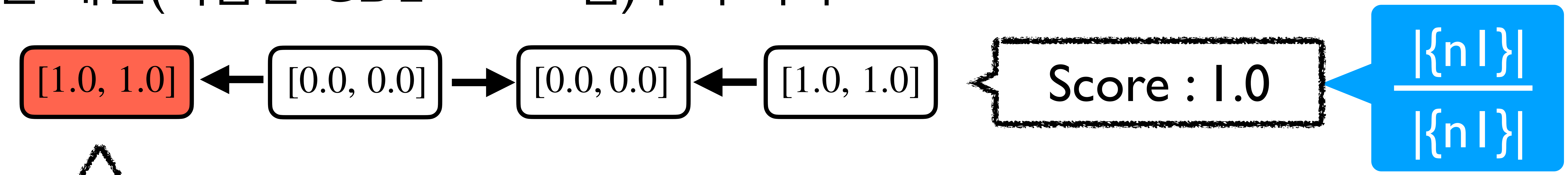
상향식(Bottom-up) GDL 프로그램 합성 알고리즘

학습 데이터(그래프) :



타깃 레이블 = 1 ( )

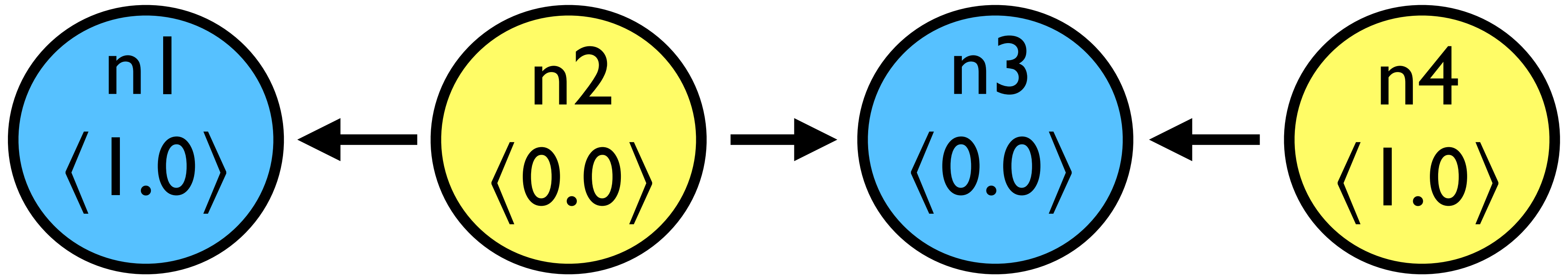
(1) 가장 specific한 패턴(복잡한 GDL 프로그램)부터 시작




```
node v1 ⟨[1.0,1.0]⟩
node v2 ⟨[0.0,0.0]⟩
node v3 ⟨[0.0,0.0]⟩
node v4 ⟨[0.0,0.0]⟩
edge (v2, v1)
edge (v2, v3)
edge (v4, v3)
target node v1
```

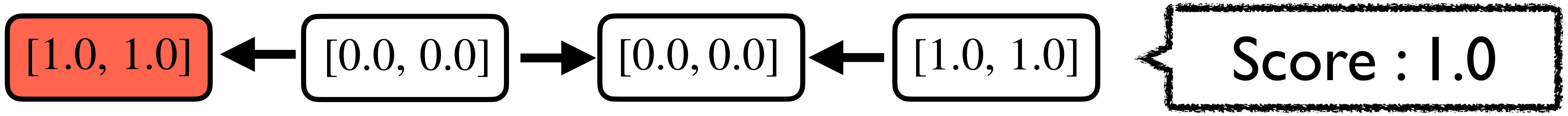
상향식(Bottom-up) GDL 프로그램 합성 알고리즘

학습 데이터(그래프) :

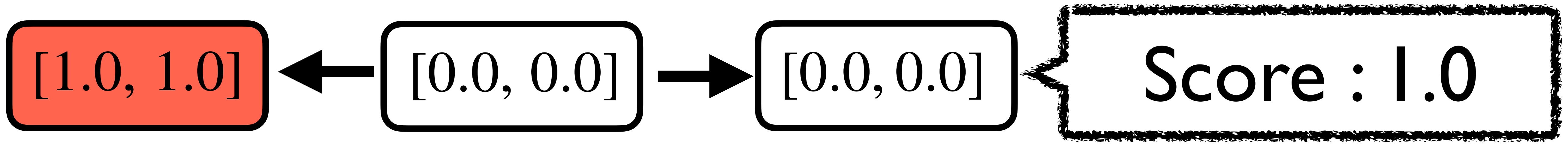


타깃 레이블 = 1 ( )

(1) 가장 specific한 패턴(복잡한 GDL 프로그램)부터 시작

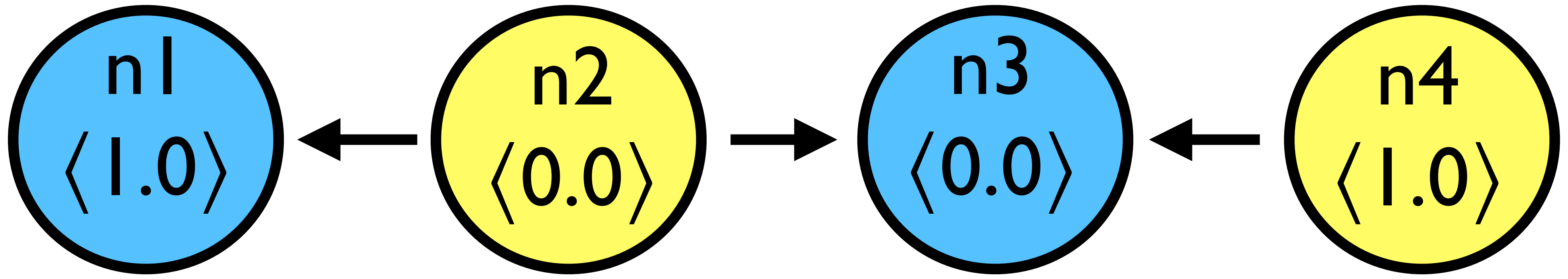


(2) 주어진 패턴을 다양하게 간단화하여 나열 후 가장 높은 점수의 패턴을 선택 (나열 탐색)



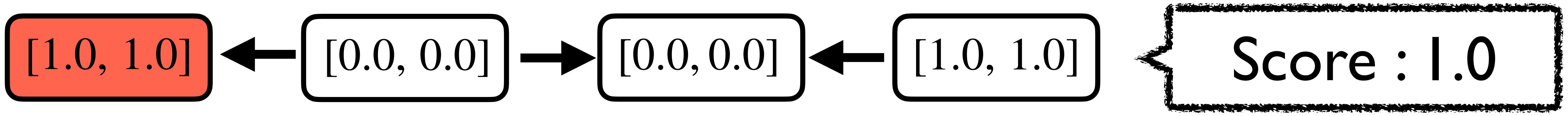
상향식(Bottom-up) GDL 프로그램 합성 알고리즘

학습 데이터(그래프) :

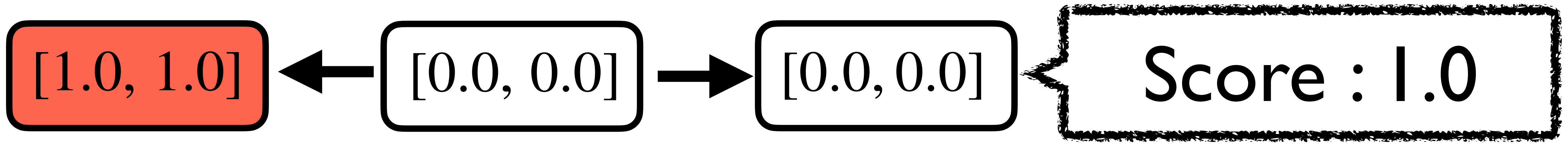


타깃 레이블 = 1 ( )

(1) 가장 specific한 패턴(복잡한 GDL 프로그램)부터 시작



(2) 주어진 패턴을 다양하게 간단화하여 나열 후 가장 높은 점수의 패턴을 선택 (나열 탐색)



(3) 모든 나열된 패턴이 현재 패턴보다 낮은 점수를 가질 때 까지 (2)를 반복

상향식(Bottom-up) GDL 프로그램 합성 알고리즘

학습 데이터(그래프) : n1 ⟨1.0⟩ ← n2 ⟨0.0⟩ → n3 ⟨0.0⟩ ← n4 ⟨1.0⟩ 타깃 레이블 = 1 (◯)

(1) 가장 specific한 패턴(복잡한 GDL 프로그램)부터 시작

[1.0, 1.0] ← [0.0, 0.0] → [0.0, 0.0] ← [1.0, 1.0] Score : 1.0

(2) 주어진 패턴을 다양하게 간단화하여 나열 후 가장 높은 점수의 패턴을 선택 (나열 탐색)

[1.0, 1.0] ← [0.0, 0.0] → [0.0, 0.0] Score : 1.0

(3) 모든 나열된 패턴이 현재 패턴보다 낮은 점수를 가질 때 까지 (2)를 반복

(4) 현재 패턴을 반환 ($[-\infty, \infty]$ ← $[-\infty, \infty]$, 1, 1.0)

GDL 프로그램 합성 알고리즘

정확도 비교

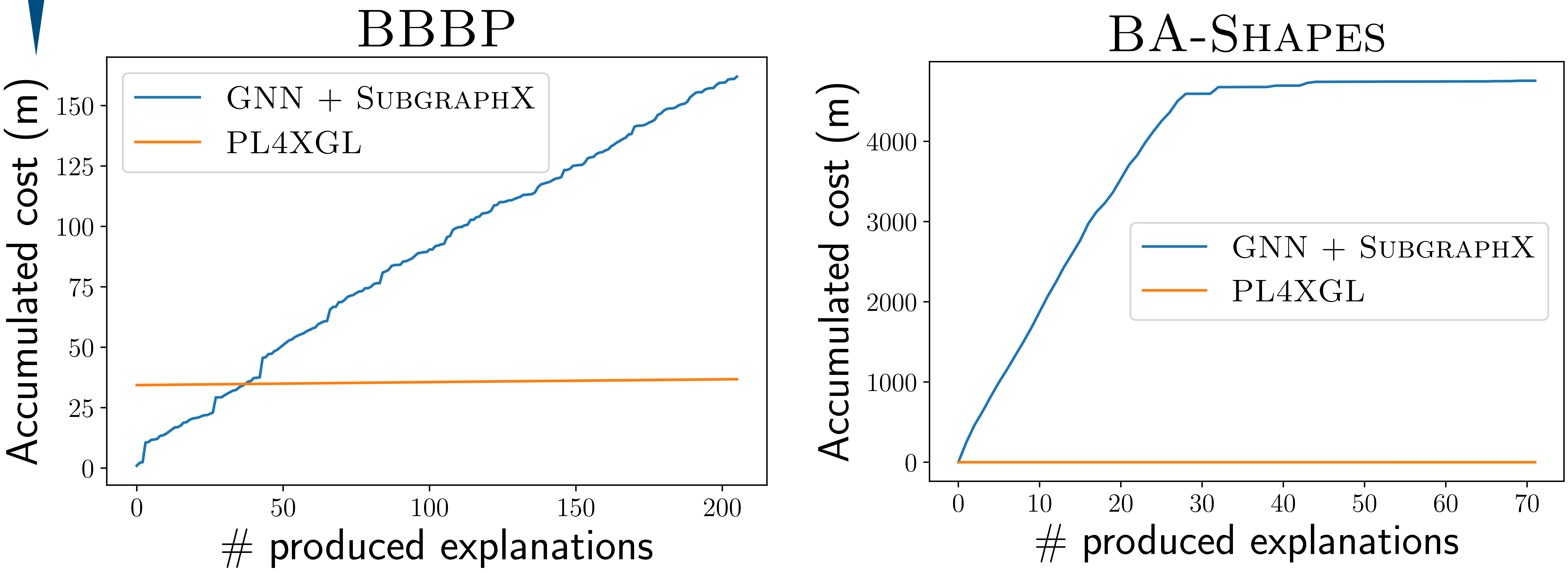
- We split the dataset into 8:1:1 for training, validation, and testing

Ours

	GCN	GAT	ChebyNet	JKNet	GraphSage	GIN	DGCN	PL4XGL
MUTAG	80.0±0.0	89.0±2.2	86.0±4.1	68.0±7.5	78.0±4.4	91.0±5.4	N/A	**100.0±0.0**
BBBP	83.6±1.4	82.3±1.6	84.6±1.0	85.6±1.9	86.6±0.9	86.2±1.4	N/A	**86.8±0.0**
BACE	78.4±2.8	52.4±3.3	78.9±1.4	79.9±1.9	79.8±0.8	**80.9±0.4**	N/A	**80.9±0.0**
BA-Shapes	95.1±0.6	76.8±2.3	**97.1±0.0**	94.3±0.0	**97.1±0.0**	92.0±1.1	95.1±0.7	95.7±0.0
Tree-Cycles	97.7±0.0	90.9±0.0	**100.0±0.0**	98.9±0.0	**100.0±0.0**	93.2±0.0	99.2±0.5	**100.0±0.0**
Wisconsin	64.0±0.0	49.6±3.1	86.4±3.9	64.8±1.5	92.8±2.9	56.0±0.0	**96.0±0.0**	88.0±0.0
Texas	67.7±5.3	50.0±0.0	87.7±2.1	68.8±4.3	86.6±2.6	50.0±0.0	**86.6±2.6**	83.3±0.0
Cornell	58.9±2.6	61.1±0.0	81.0±6.5	61.1±0.0	87.7±2.1	61.1±0.0	86.6±2.6	**88.8±0.0**
Cora	85.6±0.3	86.4±1.8	86.5±5.2	84.9±3.5	86.3±3.2	**86.7±0.0**	83.2±5.9	80.0± 0.0
Citeseer	75.2±0.0	74.3±0.7	**79.1±0.9**	73.7±4.2	75.9±2.3	75.2±0.0	71.3±6.0	63.8± 0.0
PubMed	82.8±1.1	84.7±1.2	**88.7±1.0**	83.2±0.4	88.0±0.4	86.1±0.6	85.1±0.6	81.4±0.0

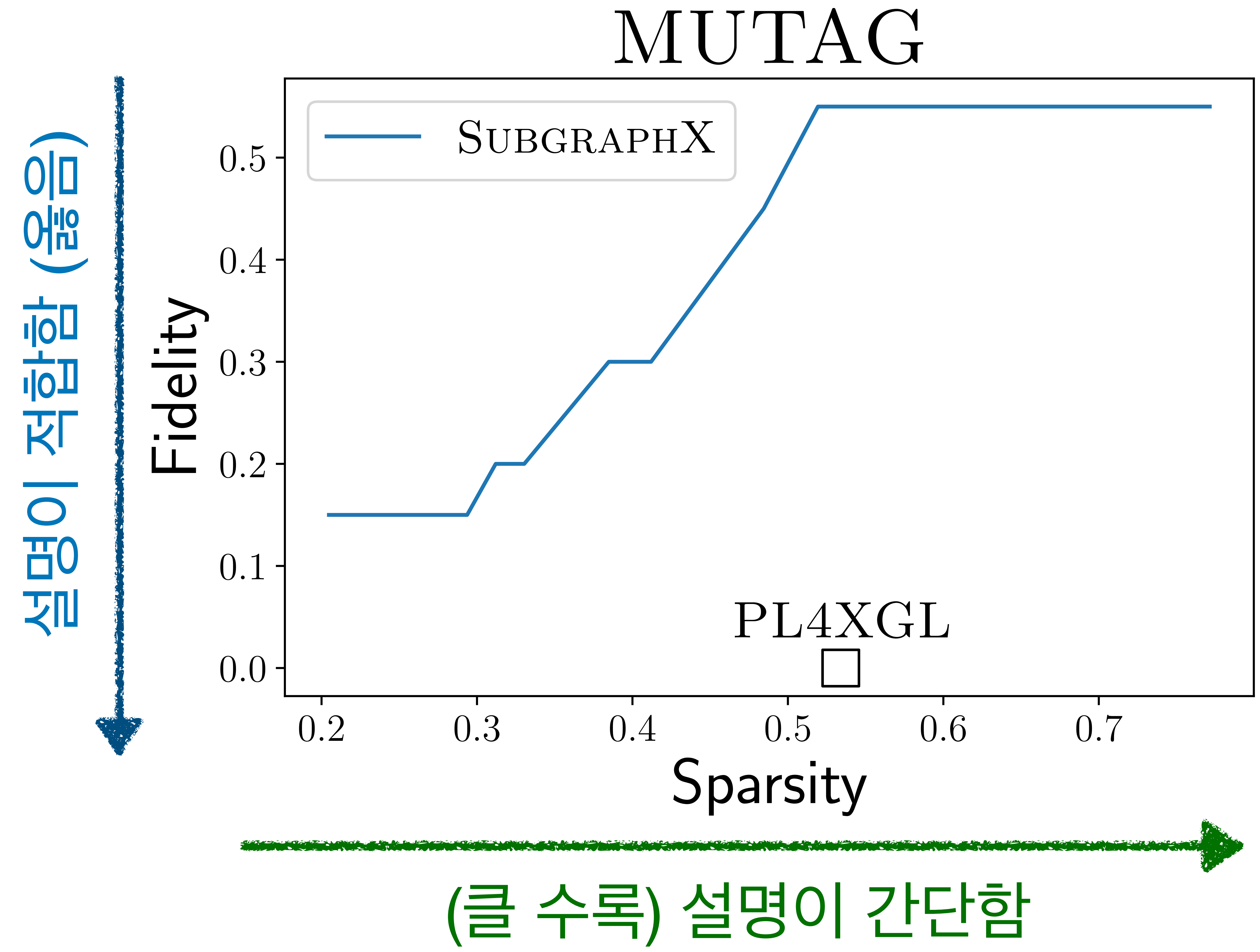
누적(학습+분류+설명)비용 비교

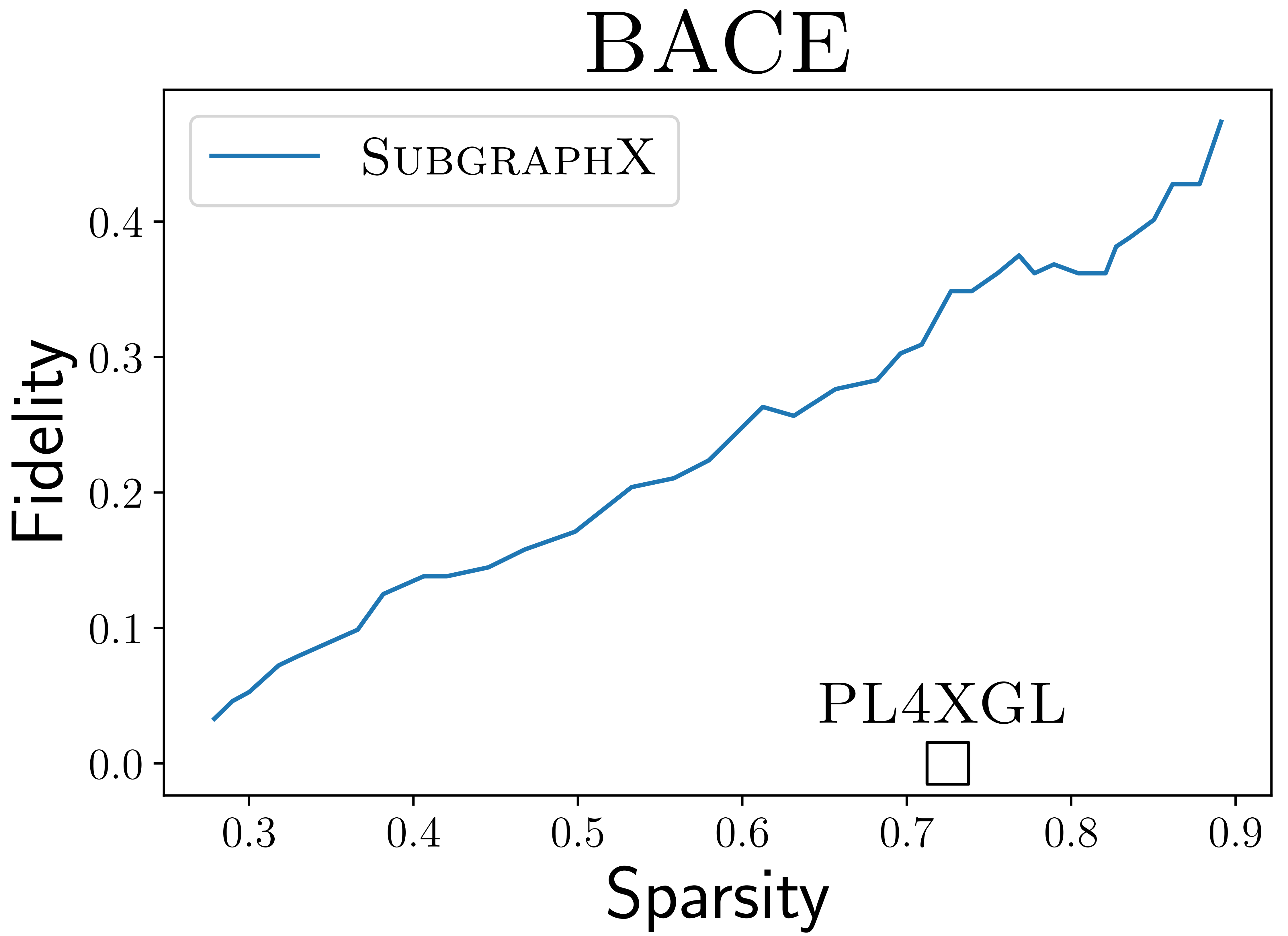
학습 비용 + 분류 비용 + 설명 비용



제공된 설명의 품질 비교

- 설명의 적합성(fidelity), 간단성(Sparsity) 비교





Graph Description Language (GDL) Project

- 목표: 그래프 표현 언어를 확장 및 사용하여 각 분야의 핵심 문제 해결하기

Published
OOPSLA' 20
데이터 기반 정적 분석을 위한 feature 자동 생성

In progress
결함 위치 추정 (Fault localization)

In progress
그래프 패턴 언어 개선

그래프 패턴 언어 (GDL)

Core language

Programs	P_4	::= δ **target** t
Descriptions	δ	::= $\delta_V \mid \delta_E$
Node Descriptions	δ_V	::= **node** x <$\overline{\phi}$>$^?$
Edge Descriptions	δ_E	::= **edge** (x,x) <$\overline{\phi}$>$^?$
Target Symbols	t	::= **node** x | **edge** (x,x) | **graph**
Intervals	ϕ	::= [$n^?$,$n^?$]
Real Numbers	n	::= 0.2 | 0.7 | 6 | -8 ...
Variables	x	::= x | y | z | ...

Submitted
This talk!
설명 가능한 그래프 기계학습 방법

In progress
GNN을 위한 graph feature 자동 생성

...

감사합니다!