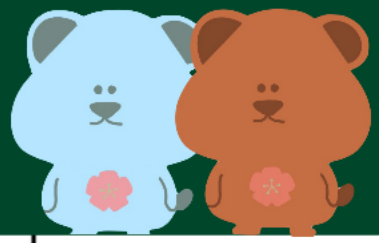
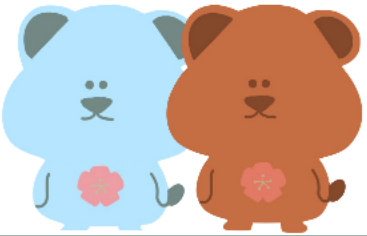


21주차 세션

DL팀 이다현

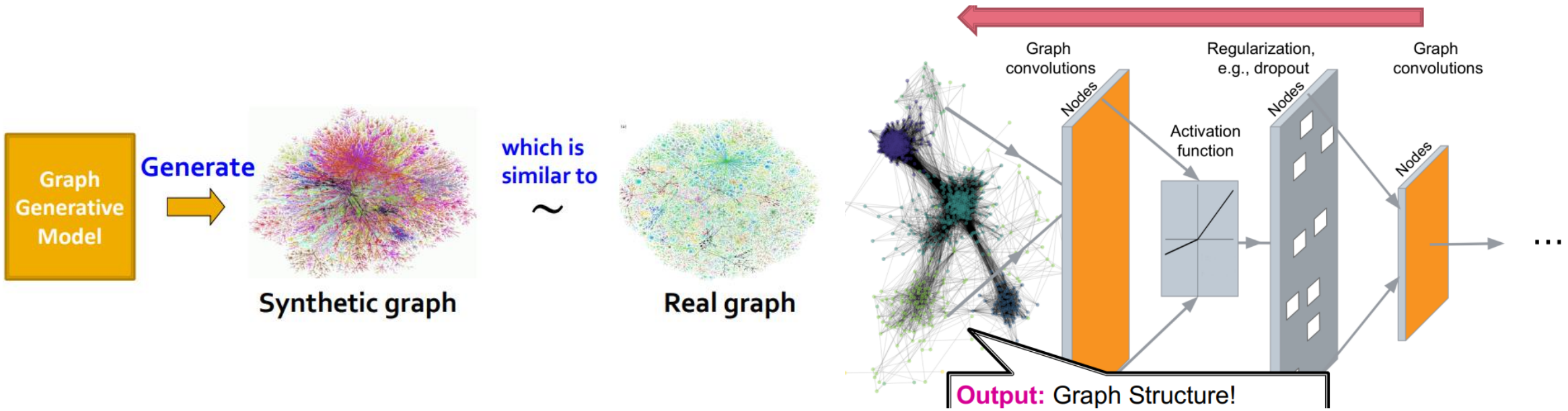
EWHA EURON

ML for graph generation



EWHA EURON

Deep Graph generation



- Learn graph formation process from the data by **deep learning model**
- 그래프 생성의 목적에는 Similar 와 optimize 가 있는데, 이번 강의에선 similar 에 주목!

Deep Graph generation

① **Density estimation** : Pmodel(x) 를 Pdata(x) 에 근접하게 추정

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{\theta})$$

MLE 를 사용!

② **Sampling** : Pmodel(x) 로부터 샘플링을 진행

표준정규분포로 노이즈 zi 를 샘플링 $\boldsymbol{z}_i \sim N(0,1)$

이때 f 는 DNN

$\boldsymbol{x}_i = f(\boldsymbol{z}_i; \theta)$

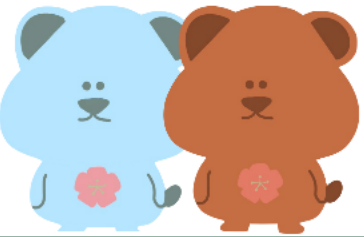
f 를 통해 complex dist 를 생성

- 목적 : 주어진 그래프와 유사한 그래프를 생성하는 것

👉 주어진 그래프 데이터로부터 분포를 학습하여 모델을 생성하고 샘플링을 진행

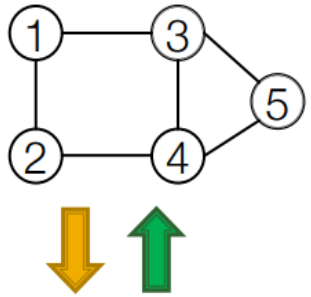
EWHA EURON

GraphRNN

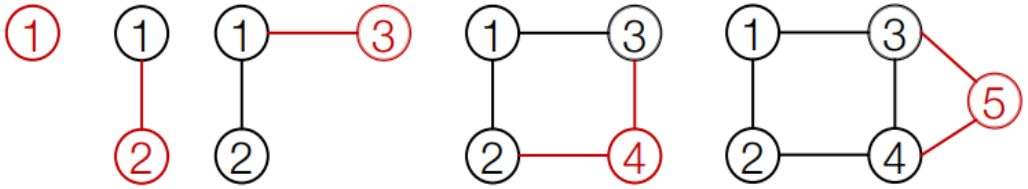


GraphRNN

Graph G with node ordering π:

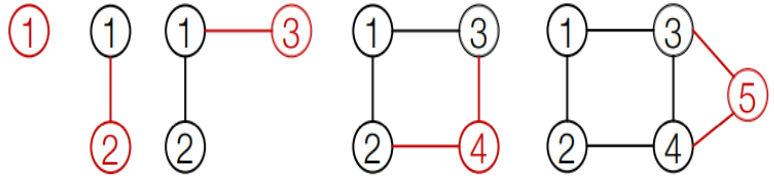


Sequence S^{π}:



$$S^{\pi} = (S_1^{\pi} , S_2^{\pi} , S_3^{\pi} , S_4^{\pi} , S_5^{\pi})$$

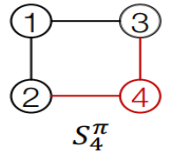
- **Node-level**: At each step, a new node is added



$$S^{\pi} = (S_1^{\pi} , S_2^{\pi} , S_3^{\pi} , S_4^{\pi} , S_5^{\pi})$$

"Add node 1" ⋯ "Add node 5"

- **Edge-level**: At each step, add a new edge



S_4^{π}

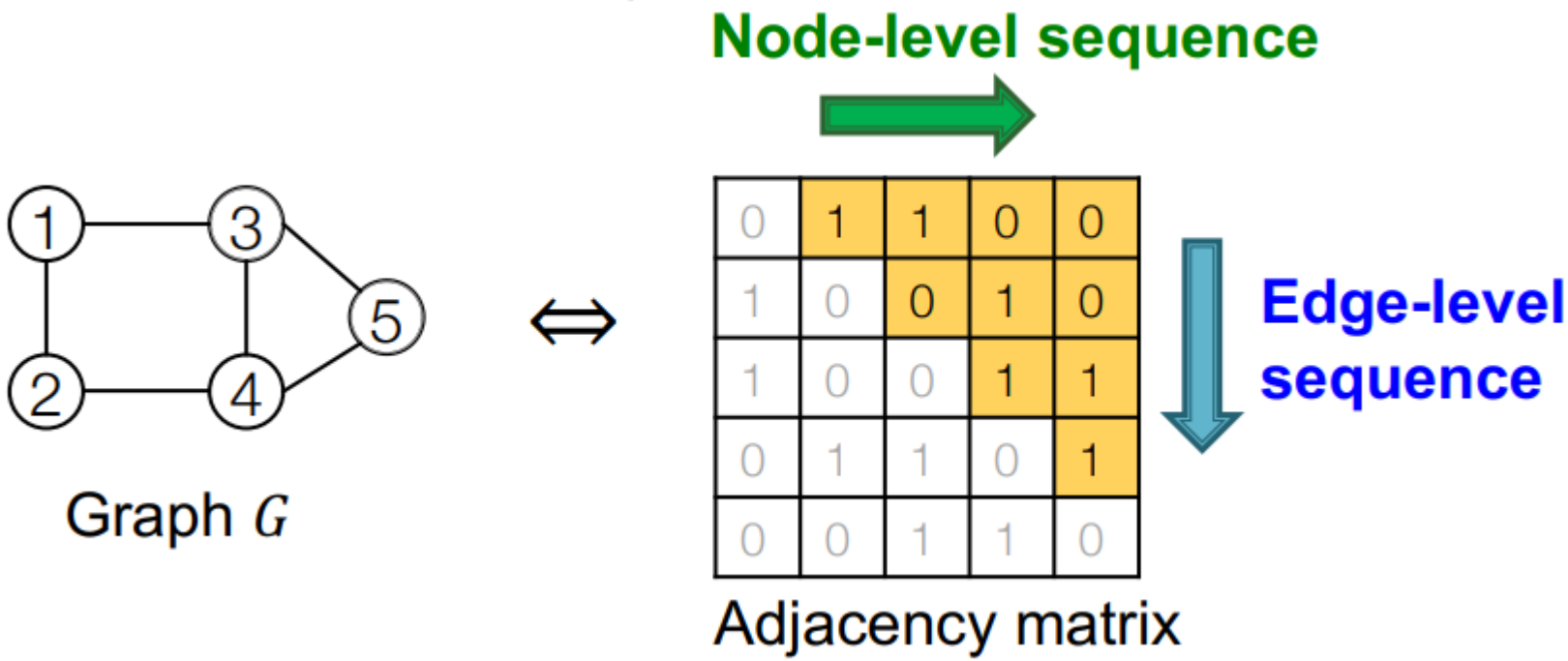
$$S_4^{\pi} = (S_{4,1}^{\pi} , S_{4,2}^{\pi} , S_{4,3}^{\pi})$$

"Not connect 4, 1" "Connect 4, 2" "Connect 4, 3"

0 1 1

- Graph generation 문제를 sequence generation 문제로 바꾼다.
- 노드를 추가하는 Node level , 엣지를 추가하는 Edge level step 이 존재한다.

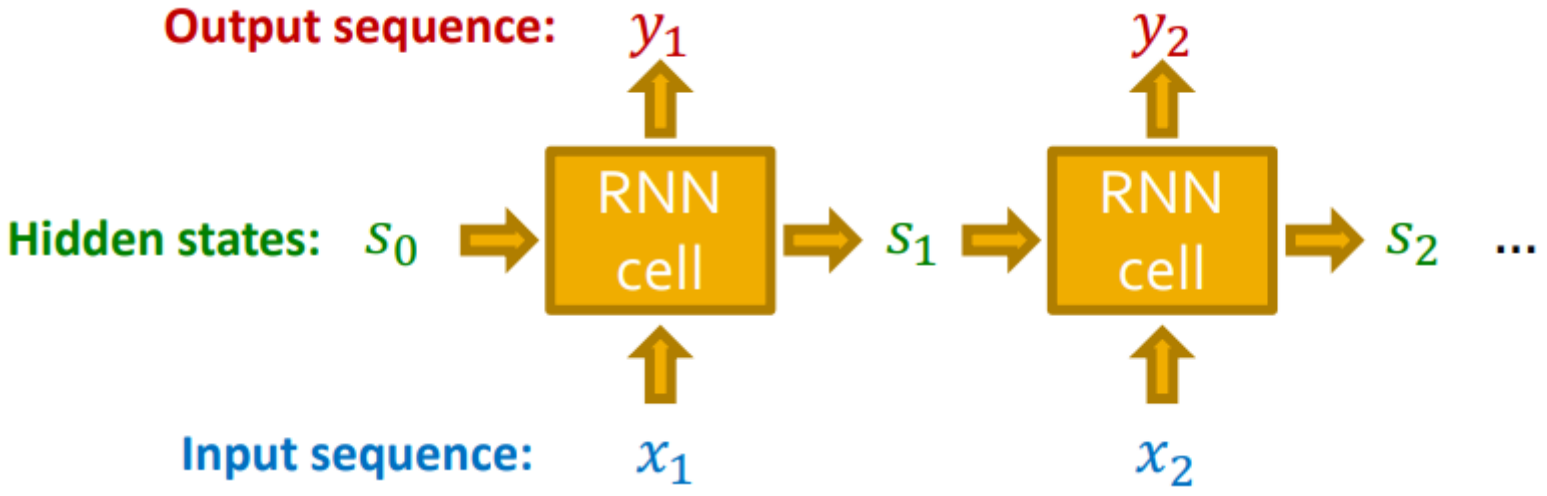
GraphRNN

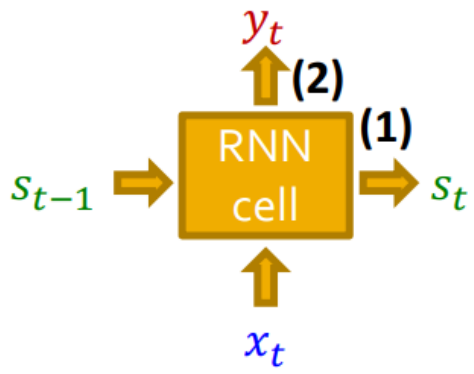


- 새로운 노드를 생성하고, 그 상태에서 새로운 엣지를 추가하는 과정을 반복

  👉 sequence of sequences → RNN 모델을 활용해서 학습을 진행시켜보자

GraphRNN





The RNN cell:

(1) Update hidden state:

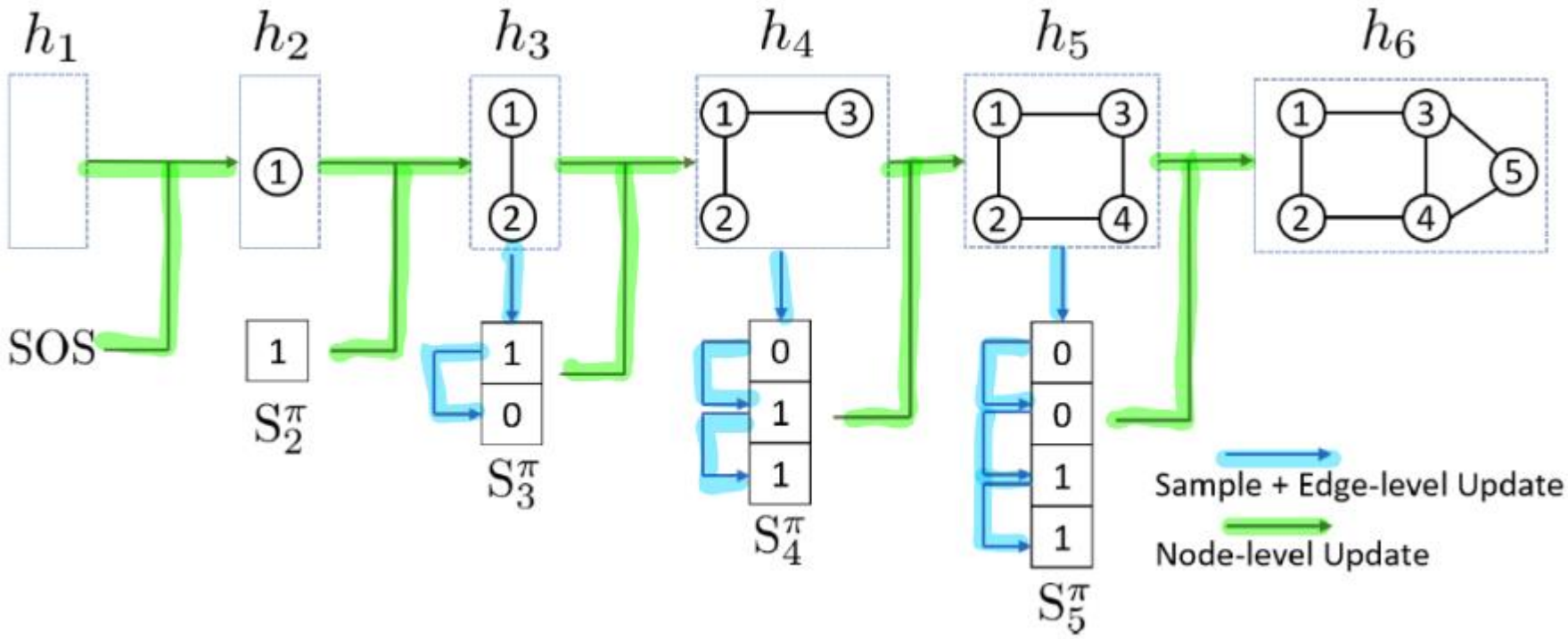
$$s_t = \sigma(W \cdot x_t + U \cdot s_{t-1})$$

(2) Output prediction:

$$y_t = V \cdot s_t$$

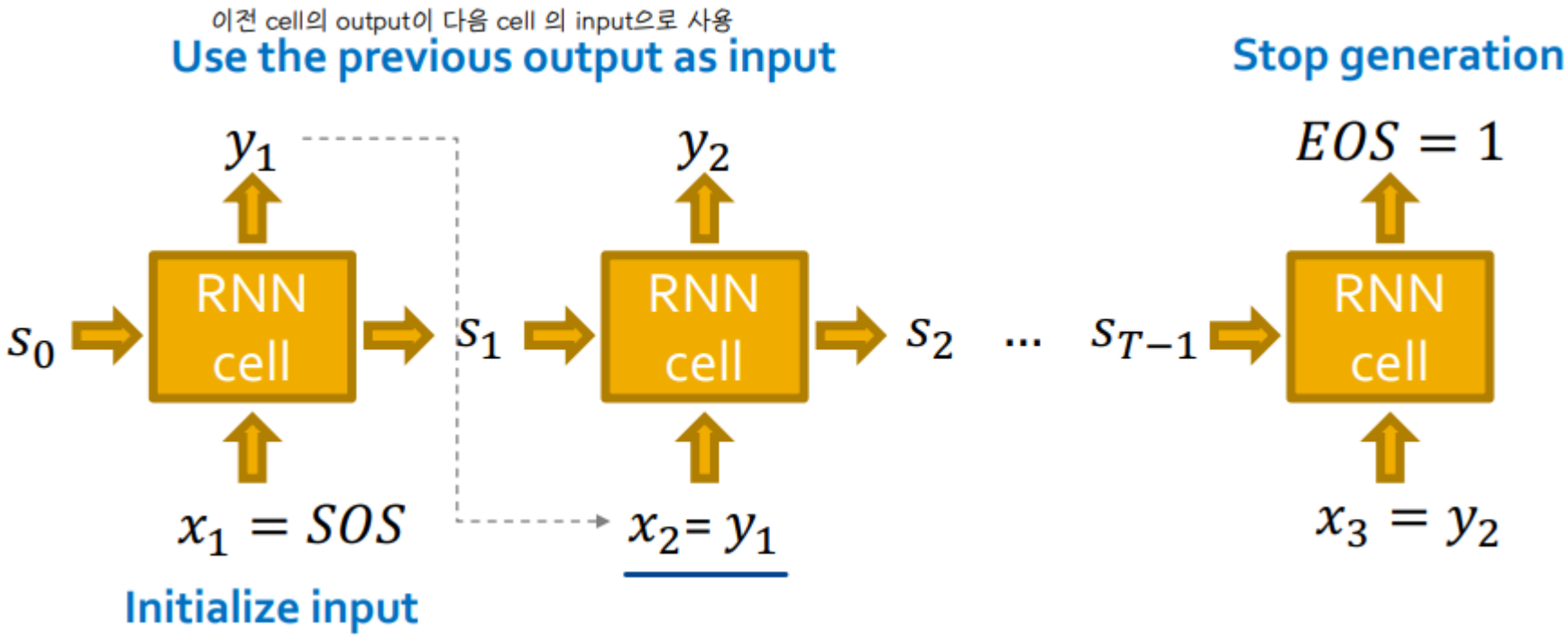
- RNN : sequential data 를 받아 입력 정보를 담고 있는 hidden state 를 update
- s(t) : state of RNN , x(t) : input to RNN, y(t) : output of RNN at step t
- W,U,V : trainable parameters

GraphRNN



- Node level RNN : edge level RNN 을 위한 초기 상태를 생성
- Edge level RNN : 이전 노드와 새로운 노드의 연결상태를 순차적으로 예측

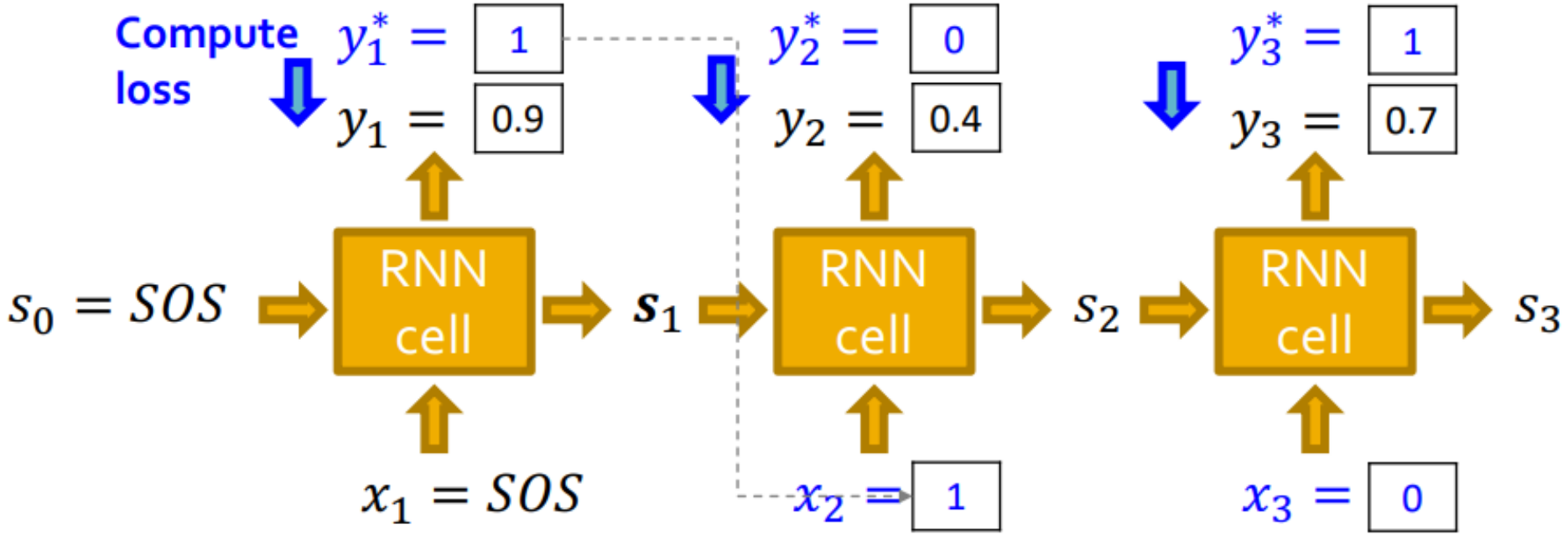
GraphRNN



- 이전 cell 의 output 을 다음 cell 의 input 으로 사용 → sequence
- start token 의 SOS vector 를 사용 (모든 값이 00이거나 1) → initialize
- End token 의 EOS 를 사용 (값이 10이면 stop) → Stop generation

똑같은 그래프만 생성해내는 문제점 **(deterministic)**

GraphRNN



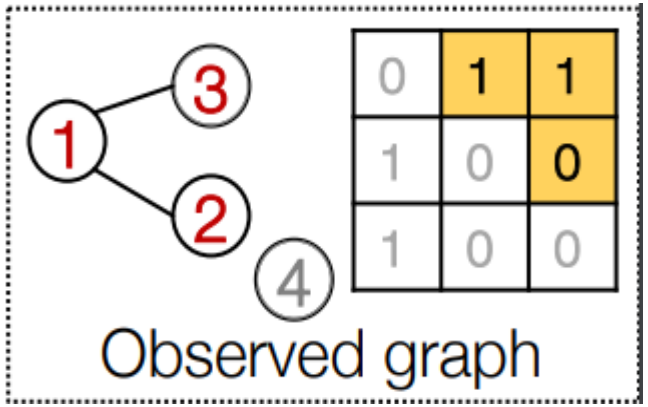
Output 을 엣지 연결 여부 자체가 아닌 확률 정보를 출력하도록 만듦 + teacher Forcing 방법

(Teacher Forcing : 정답 레이블을 디코더의 다음 입력으로 넣는 방법)

GraphRNN

- Training

For each prediction, we **get supervision from the ground truth**



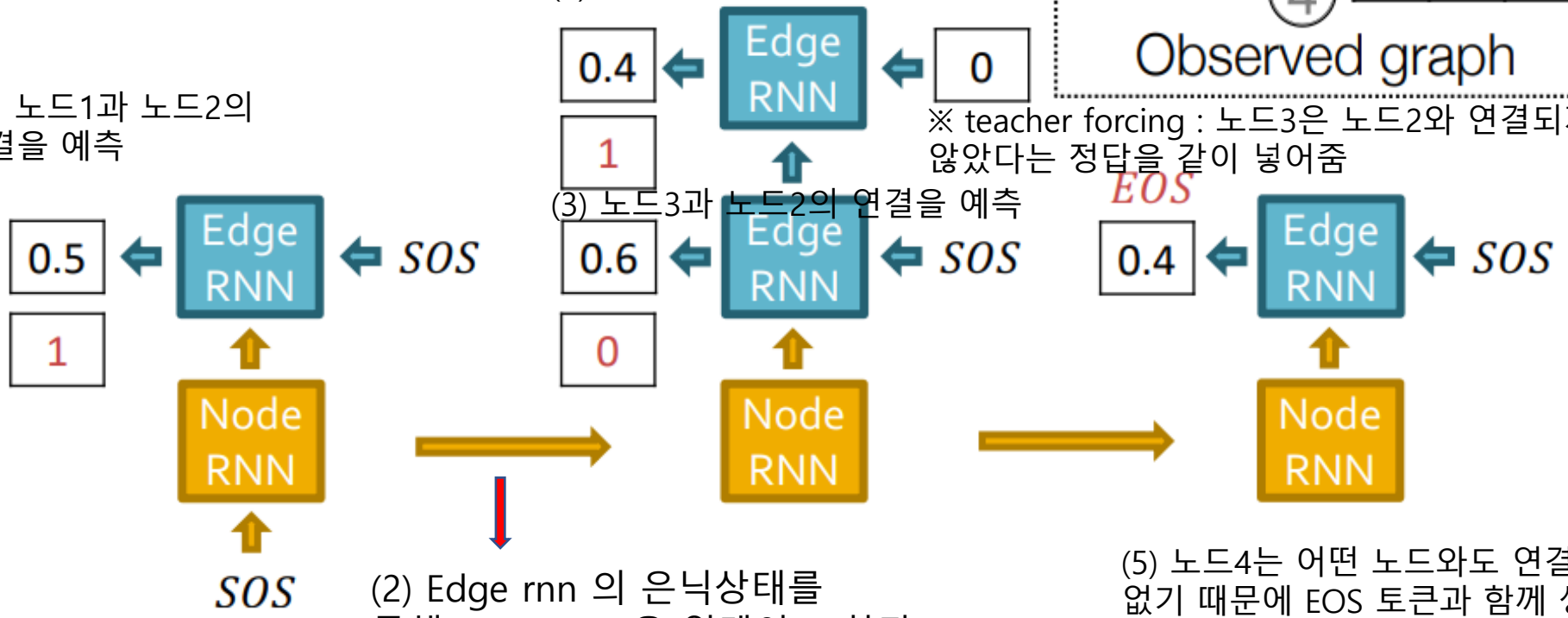
(6) Backprop : 역전파, 파라미터 업데이트

(4) 노드3과 노드1의 연결을 예측

(1) 노드1과 노드2의 연결을 예측

※ teacher forcing : 노드3은 노드2와 연결되지 않았다는 정답을 같이 넣어줌

(3) 노드3과 노드2의 연결을 예측

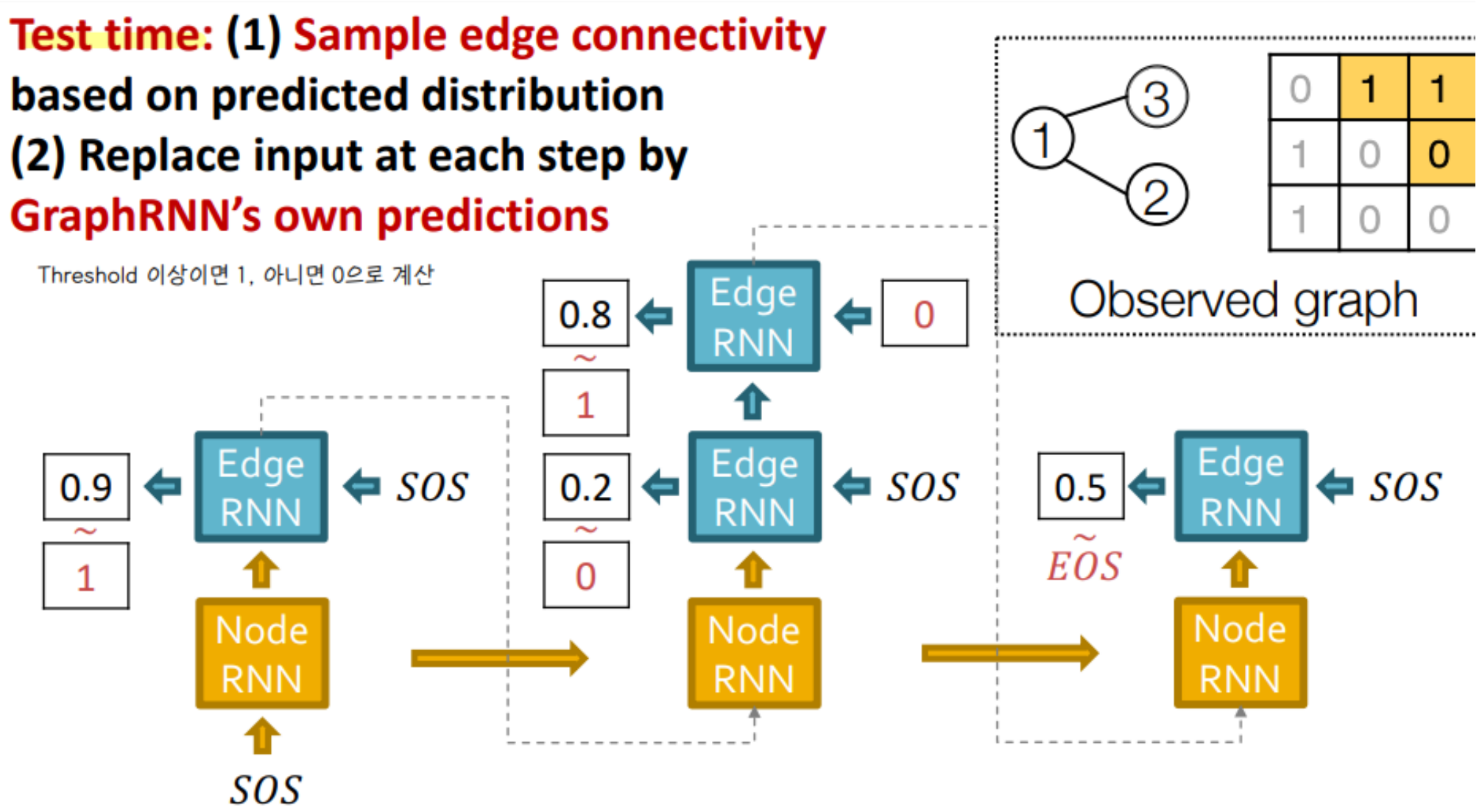


(2) Edge rnn 의 은닉상태를 통해 Node rnn 을 업데이트 한다.

(5) 노드4는 어떤 노드와도 연결이 없기 때문에 EOS 토큰과 함께 생성을 멈추게 된다.

GraphRNN

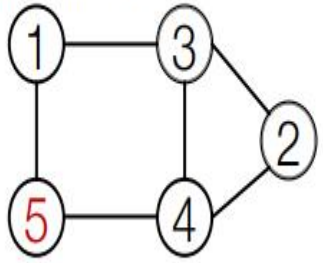
- Test



GraphRNN

- Complex too-long edge dependencies

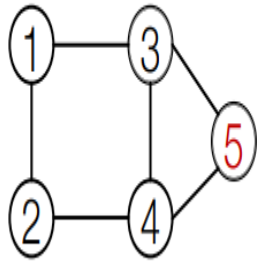
노드2가 추가되면 1과의 엣지확인,
노드3이 추가되면 1,2와의 엣지확인 …



"Recipe" to generate the left graph:
- Add node 1
- Add node 2
- Add node 3
- Connect 3 with 2 and 1
- Add node 4
- …

Random node ordering:
Node 5 may connect to any/all previous nodes

- Breadth-First Search node ordering



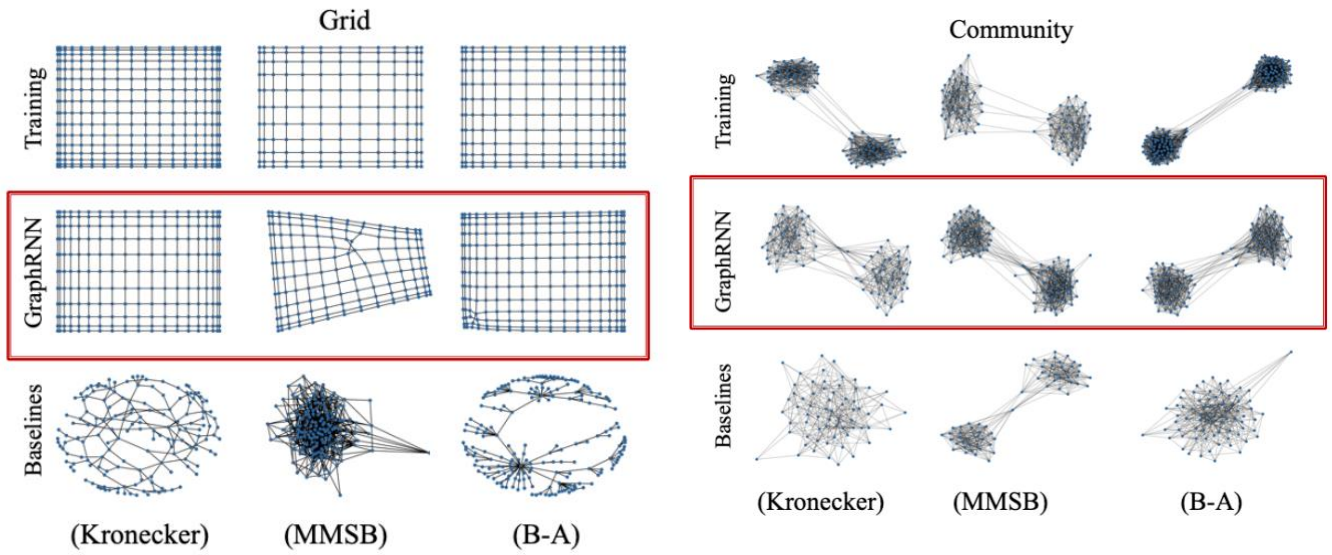
BFS ordering

BFS node ordering: Node 5 will never connect to node 1 (only need memory of 2 "steps" rather than $n-1$ steps)

- 문제점 : Tractability) 엣지 생성을 위한 계산 과정이 복잡함
- 해결책 : BFS방식을 도입해, 엣지 생성에 있어 탐색해야 하는 개수를 줄임

GraphRNN

(1) Visual Similarity



외관상 얼마나 비슷해 보이는지

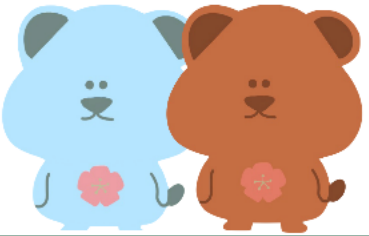
(2) Graph statistics

- Typical Graph Statistics: 그래프 통계치를 비교
  - Degree distribution (Deg.)
  - Clustering coefficient distribution (Clus.)
  - Orbit count statistics (Orbit)

- 생성한 그래프를 평가하는 방법 : 두 그래프를 비교

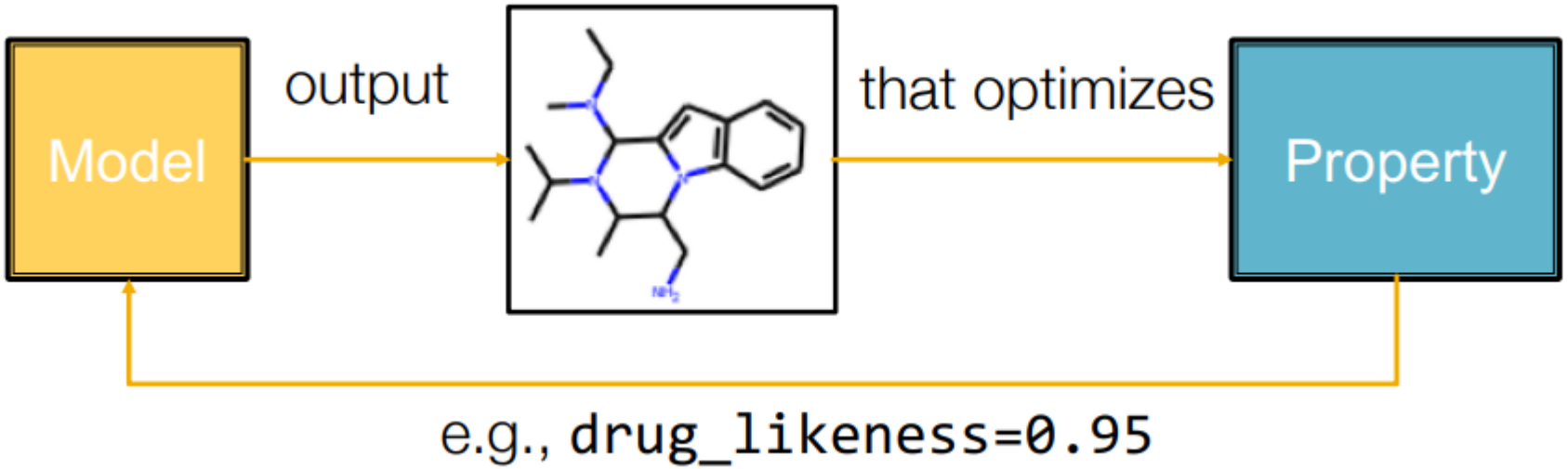
👉 “similarity” 를 측정하기 위해 Visual 방법과 Graph statistics 방법을 도입

Application of Deep GM



EWHA EURON

Application : Drug Discovery

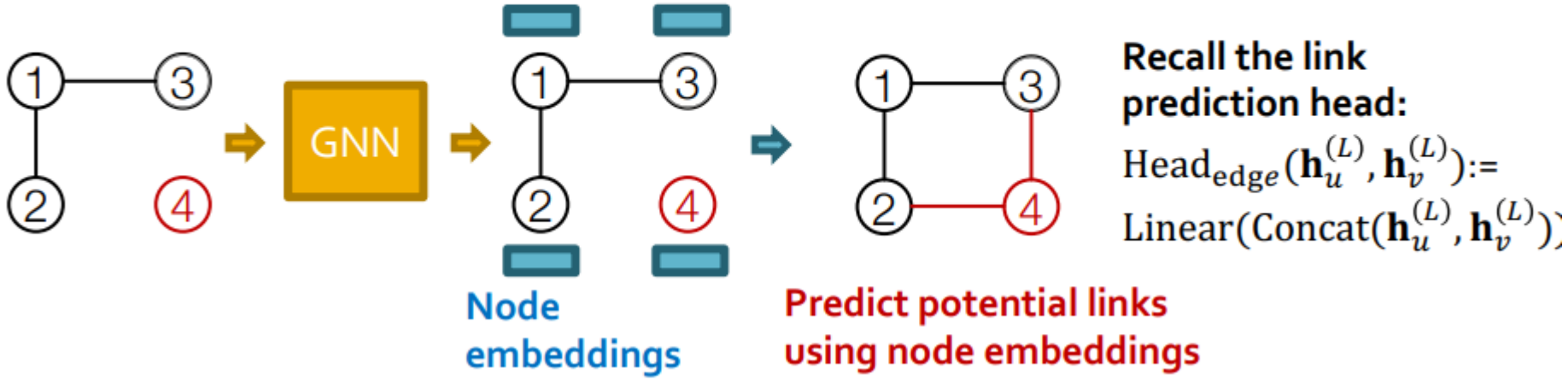


- 약품개발과 같이 graph optimize 가 필요한 task 는 Valid, Realistic, Optimize 특성이 존재

👉 강화학습을 결합해볼까 … → GCPN

GCPN

- **GCPN:** predict action based on **GNN node embeddings**



- **GCPN** : graph representation + Reinforcement learning

👉 RNN 에 비해 표현력은 높아지지만, 계산 시간이 더 오래 걸린다.