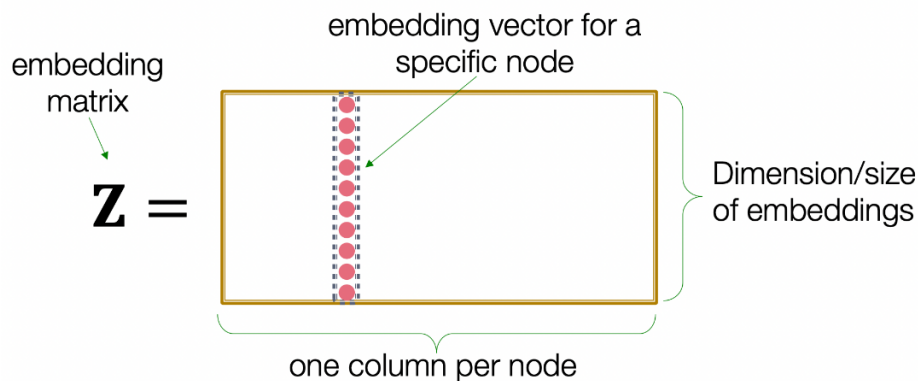


Graph Neural Networks 1

Recap

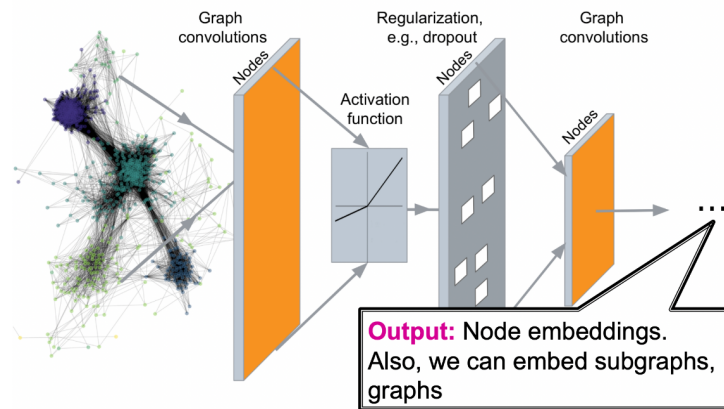
- 노드 임베딩
 - 그래프 상에서 유사한 노드들이 함수 f 를 거쳐 d 차원으로 임베딩 되었을 때에 가깝게 위치하도록 만드는 것
 - 저차원으로 mapping하는 **encoder** & original network에서의 유사도와 embedding space에서의 내적이 유사하도록 만드는 **similarity function**



- 가장 간단한 인코딩 : 임베딩 벡터를 각 column에 담는 **lookup** 방식
 - 한계점 존재
 - $O(|V|)$ 의 파라미터 필요
 - 훈련 과정에서 보지 못한 노드는 임베딩 생성 불가능
 - 노드의 feature을 포함하지 않음

Graph Neural Networks

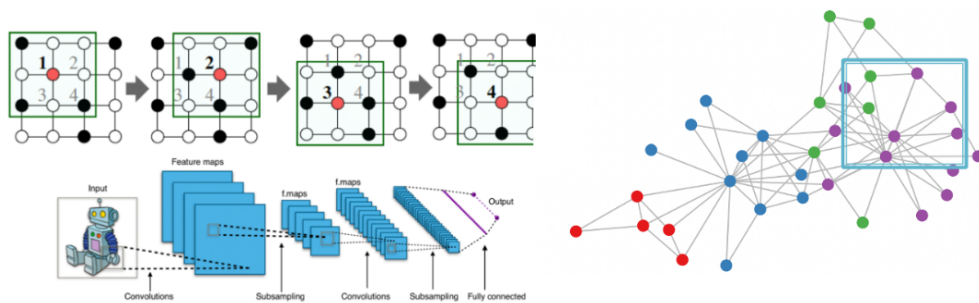
- GNN에서는 multiple layer로 구성된 encoder 활용
 - node classification, link prediction, community detection, network similarity 등의 task 수행
- GNN 구조



Deep Learning for Graphs

• Convolutional Networks

- 그래프에 convolution을 적용할 때 locality나 sliding window에 고정된 개념이 없어 어려움
 - 하나의 필터에 들어오는 노드의 수가 슬라이딩을 하며 바뀜
- CNN 필터는 중심 픽셀을 기준으로 이웃 픽셀들과의 관계를 통해 값을 내주는 역할
 - 그래프에서도 이웃 노드들의 메시지를 통해 결과를 내는 개념으로 적용 가능

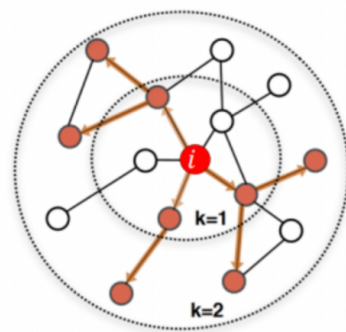


• Graph Convolutional Networks

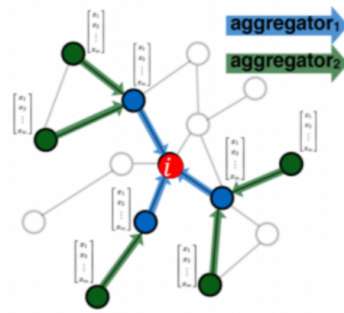
- CNN은 고정된 구조에서 여러 데이터를 받아 순전파, 손실함수 계산, 역전파 과정을 거침
- 하지만 GCN은 고정된 구조가 아님!

- GCN의 과정

- 각 노드별 계산 그래프 생성
- 각 노드별 계산 그래프에 따라 순전파
- 손실함수 계산 및 역전파



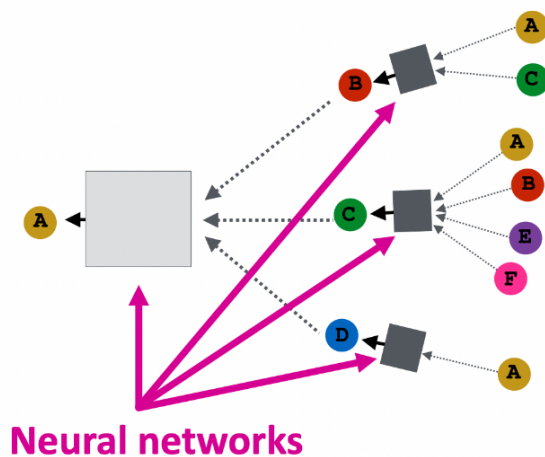
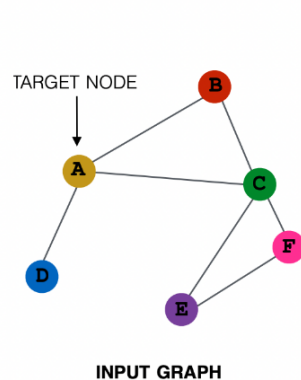
Determine node computation graph



Propagate and transform information

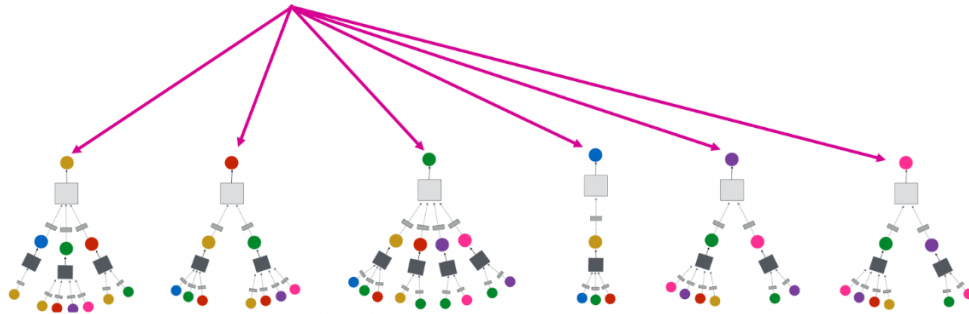
- Aggregate Neighbors

- local network neighborhood를 통해 노드 임베딩을 만들 수 있음
- 2개의 layer를 거치도록 구성하면 target node A는 이웃 노드인 B, C, D로부터 정보를 받고 이 이웃 노드들은 또 다른 각자의 이웃 노드들로부터 정보를 받음



Neural networks

- Layer 1 에 있는 B는 A와 C로부터 받은 정보와 자기 자신의 정보가 합쳐져 있어, Layer 0 에 있는 B와 다른 벡터로 구성되어 있음
- 모든 노드들에 대한 local한 계산 그래프



- 각각의 계산 그래프는 서로 다른 구조를 갖게 됨
- Layer의 수는 임의로 정할 수 있으나 노드의 개수만큼 계산 그래프가 생기기 때문에 너무 깊게 쌓는 데는 한계가 있음
- 공식

Initial 0-th layer embeddings are equal to node features

$$h_v^0 = x_v$$

$$h_v^{(l+1)} = \sigma \left(W_l \sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|} + B_l h_v^{(l)} \right), \forall l \in \{0, \dots, L-1\}$$

embedding of v at layer l

Average of neighbor's previous layer embeddings

Non-linearity (e.g., ReLU)

Embedding after L layers of neighborhood aggregation

Total number of layers

$$z_v = h_v^{(L)}$$

- 이웃 노드 정보의 전달은 input의 위치와 관계 없는 평균값을 활용한 이후 neural network를 거침
- 첫번째 layer
 - input input으로 노드의 feature 그대로 들어감

- 두번째 layer

input 이웃 노드들의 이전 layer에서의 임베딩 평균값과 각 이웃 노드들의 임베딩

output $h_v^{(l+1)}$

- WI과 BI은 학습되는 파라미터

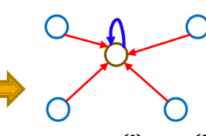
- **Matrix Formulation**

- 공식

$$\sum_{u \in N(v)} \frac{h_u^{(l-1)}}{|N(v)|} \longrightarrow H^{(l+1)} = D^{-1}AH^{(l)}$$

$$H^{(l+1)} = \sigma(\tilde{A}H^{(l)}W_l^T + H^{(l)}B_l^T)$$

where $\tilde{A} = D^{-1}A$



$H^{(l)} = [h_1^{(l)} \dots h_{|V|}^{(l)}]^T$

- Red: neighborhood aggregation
- Blue: self transformation

- $H^{(l)}$ l번째 layer의 모든 노드에 대한 vector를 concat한 행렬
- A 인접 행렬 $\rightarrow AH^{(l)}$ 을 통해 v 노드의 모든 이웃 노드의 벡터 합을 구할 수 있음
- D v 노드의 이웃노드 수가 담긴 대각행렬 $\rightarrow D^{-1}AH^{(l)}$ 은 v 노드의 이웃노드 수의 역수
- $H^{(l+1)} = D^{-1}AH^{(l)}$ 연산 가능
- 빨간 색은 이웃 노드의 정보를 모으는 부분 / 파란 색은 본인 노드의 정보를 변형

- **Training**

- 공식

Unsupervised

$$\mathcal{L} = \sum_{z_u, z_v} \text{CE}(y_{u,v}, \text{DEC}(z_u, z_v))$$

Supervised

$$\mathcal{L} = \sum_{v \in V} y_v \log(\sigma(z_v^T \theta)) + (1 - y_v) \log(1 - \sigma(z_v^T \theta))$$

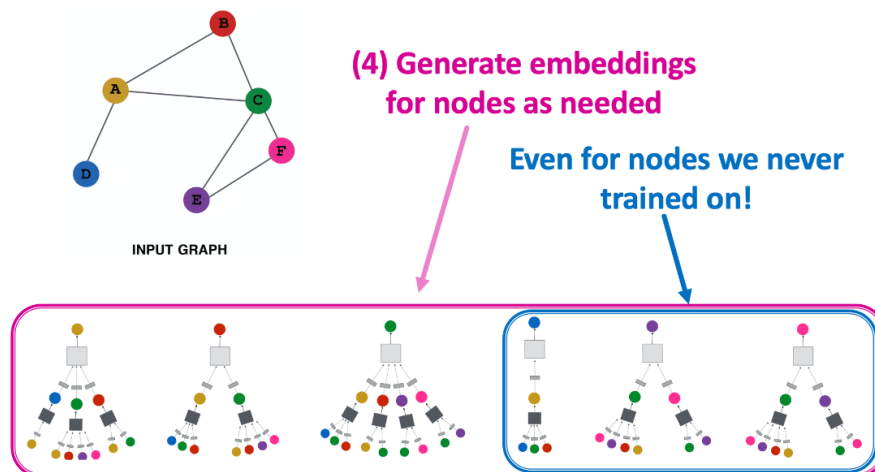
Encoder output: node embedding

Classification weights

Node class label

Safe or toxic drug?

- Unsupervised
 - 그래프의 구조를 supervision으로 사용해 유사한 노드가 embedding space에서도 유사해지도록 학습
- Supervised
 - 노드 예측 시 에러가 최소가 되도록 학습
- 여러 개의 계산 그래프를 배치 단위로 학습
 - 모델의 파라미터들이 공유되어 보지 않은 노드 혹은 그래프에 대해 일반화 능력을 가질 수 있음



GraphSAGE

- Aggregation : 가중평균을 이용하는 대신 미분 가능한 함수를 사용할 수 있음

