20230831

Machine Learning with Graphs

서수원

Business Intelligence Lab.

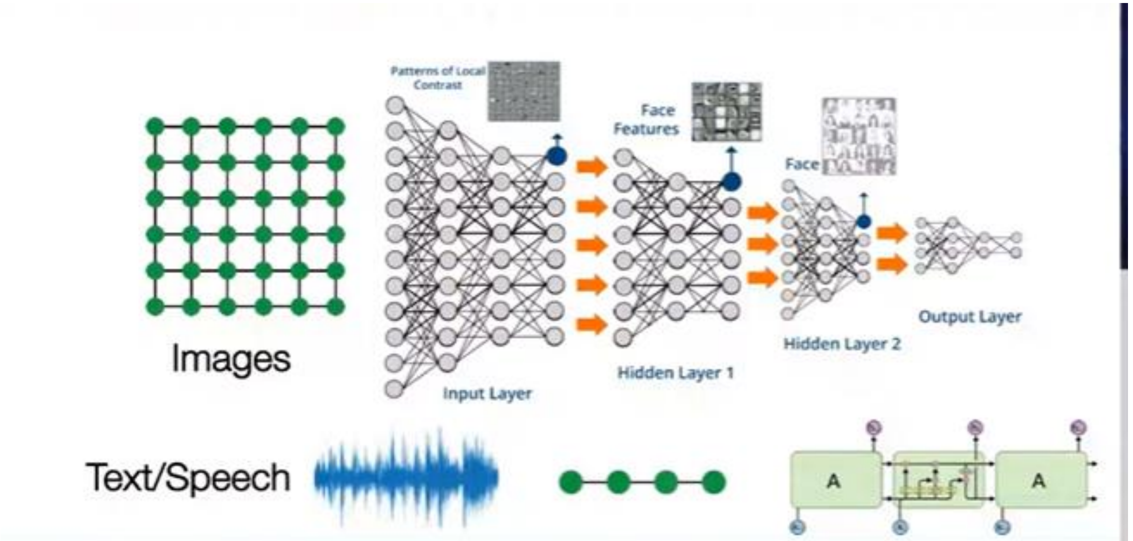
산업경영공학과, 명지대학교

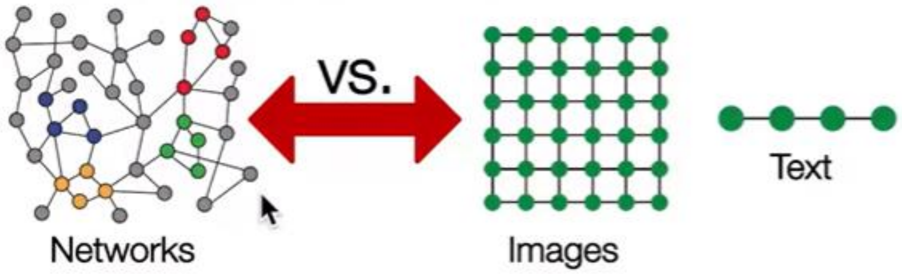
01

Basic of deep learning

Modern Deep Learning

- Modern Deep Learning
  - Sequences or Fixed Grids에 사용이 된다.
    - ✓ 그래프는 연속적이지도 않고 고정된 크기도 아니기 때문에 딥러닝의 일반화된 모델이 필요하다.
    - ✓ 네트워크는 공간지역성이 없다. 즉 기준점이 없다는 의미이다.





Basics of deep learning

- Machine Learning as Optimization
  - 지도학습
    - ✓ Given input X, goal is to predict label Y.
      - ❖ X can be Sequences, vectors, numbers, matrices, Graphs
    - ✓ Y를 잘 예측하는 것을 최적화 문제로 공식화 할 수 있다.

Basics of deep learning

- Machine Learning as Optimization
  - Optimization Problem
    - ✓ 세타는 우리가 최적화 하고싶은 매개변수이다.
    - ✓ Loss Function은 실제값과 예측값의 차이를 비교하는 함수이다. 얼마나 잘못 예측하는지를 확인하는 함수로 보면 된다.

- **Formulate the task as an optimization problem:**

$$\min_{\Theta} \boxed{\mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}))}$$

Objective function

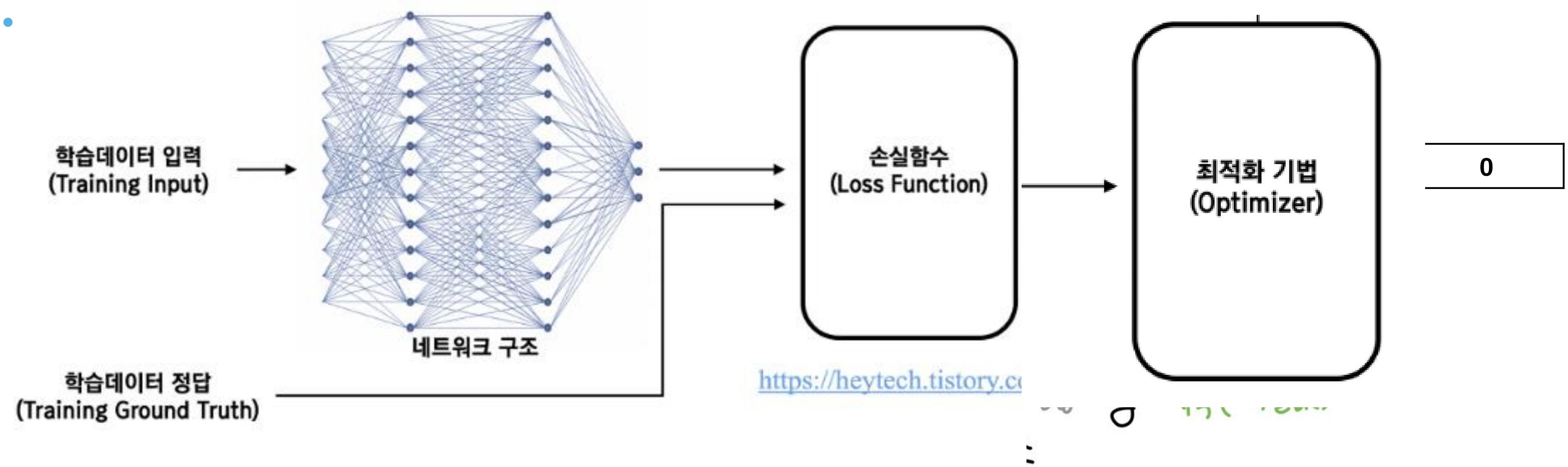
- Θ: a set of **parameters** we optimize
  - Could contain one or more scalars, vectors, matrices ...
  - E.g. $\Theta = \{Z\}$ in the shallow encoder (the embedding lookup)

- $\mathcal{L}$: **loss function**. Example: L2 loss

$$\mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x})) = \|y - f(x)\|_2$$

  - Other common loss functions:
    - L1 loss, huber loss, max margin (hinge loss), cross entropy ...
    - See https://pytorch.org/docs/stable/nn.html#loss-functions

Basics of deep learning



Basics of deep learning

- How to optimize the objective function
  - 학습을 통해서 한다.
  - What is Gradient?

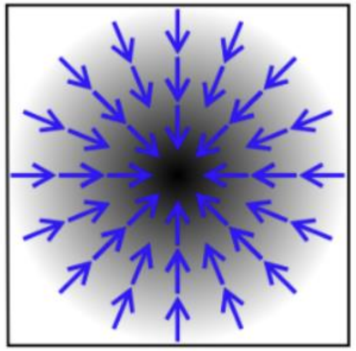
$$\frac{\partial f(x,y)}{\partial x} = \frac{\partial}{\partial x}(x^2+3xy+y^2) = 2x+3y$$

$$f(x,y) = x^2+3xy+y^2$$

$$\frac{\partial f(x,y)}{\partial y} = \frac{\partial}{\partial y}(x^2+3xy+y^2) = 3x+2y$$

  - 다변수 함수가 갖는 축마다 편미분 하는 것을 의미한다.

$$gradient(f) = \nabla f(x) = [\frac{\partial f(x_0)}{\partial x_0}, \frac{\partial f(x_1)}{\partial f(x_1)}, \ldots, \frac{\partial f(x_{N-1})}{\partial x_{N-1}}]^T$$

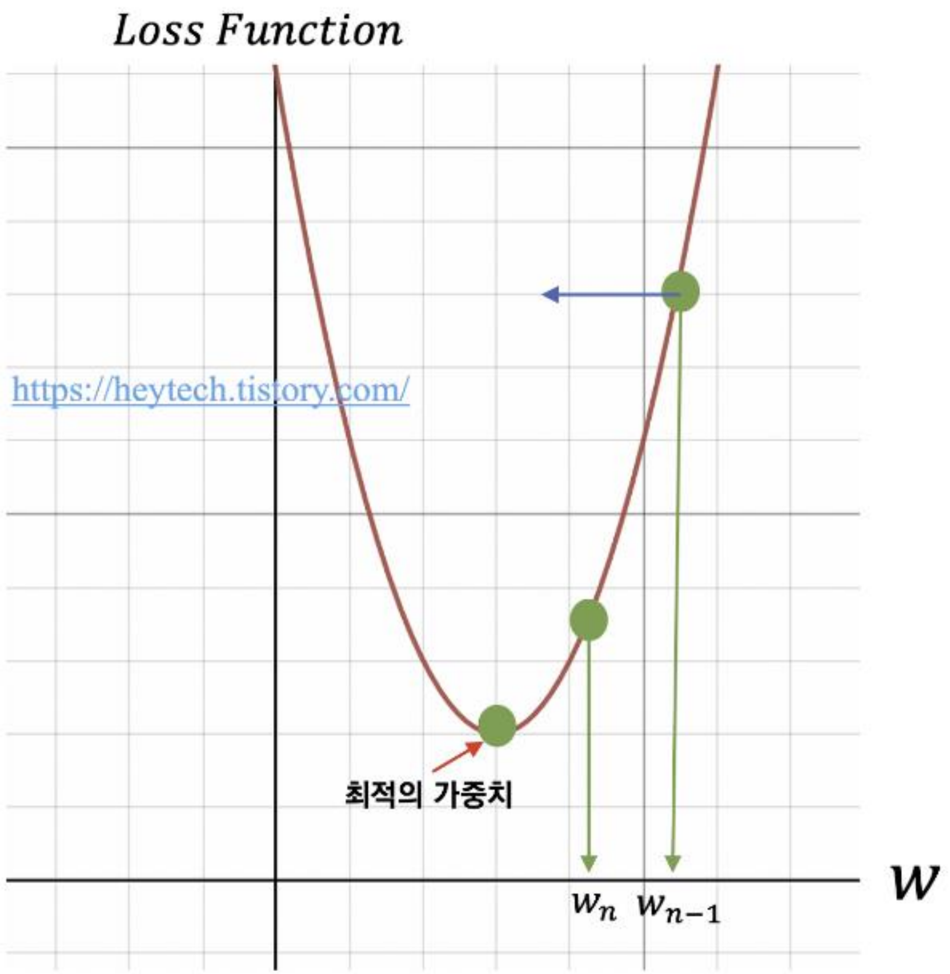


Basics of deep learning

- Gradient Descent
  - Iterative Algorithm

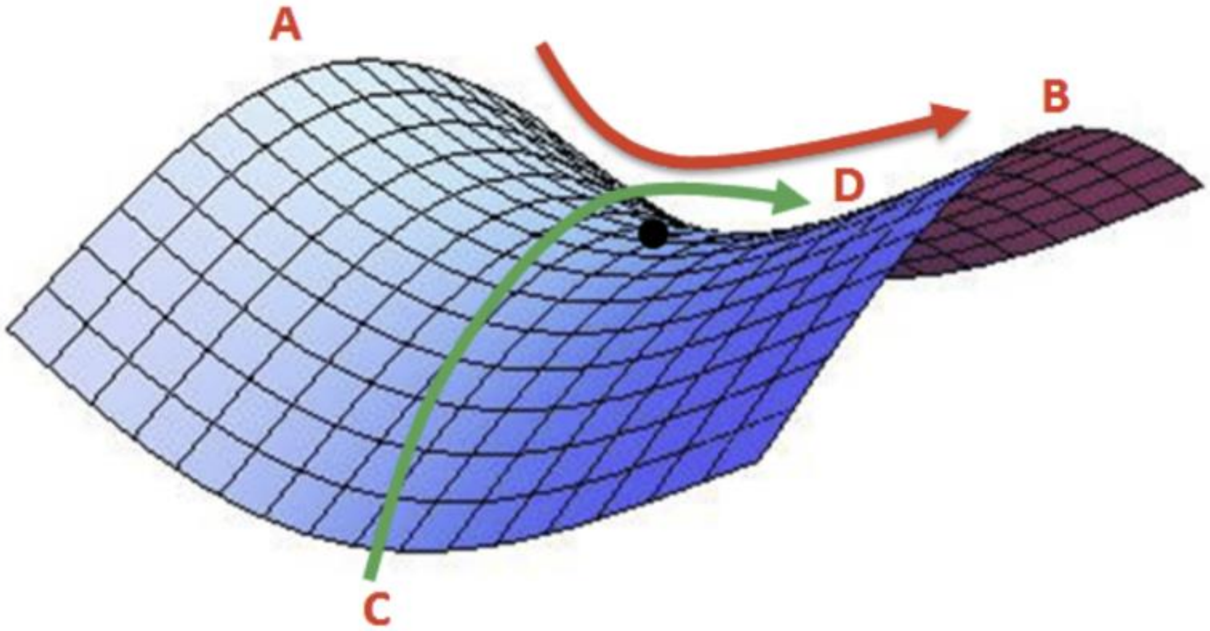
$$w_n = w_{n-1} - \alpha \nabla f(w_{n-1})$$

$$\Theta \leftarrow \Theta - \eta \frac{\partial \mathcal{L}}{\partial \Theta}$$



Basics of deep learning

- Gradient Descent
  - Problem
    - ✓ 완벽히 최적화된 값을 구하려면 모든 값에 대해 계산을 해야 한다.
    - ✓ 모든 단계에서 값을 구하는 것은 현실적으로 불가능 하다.



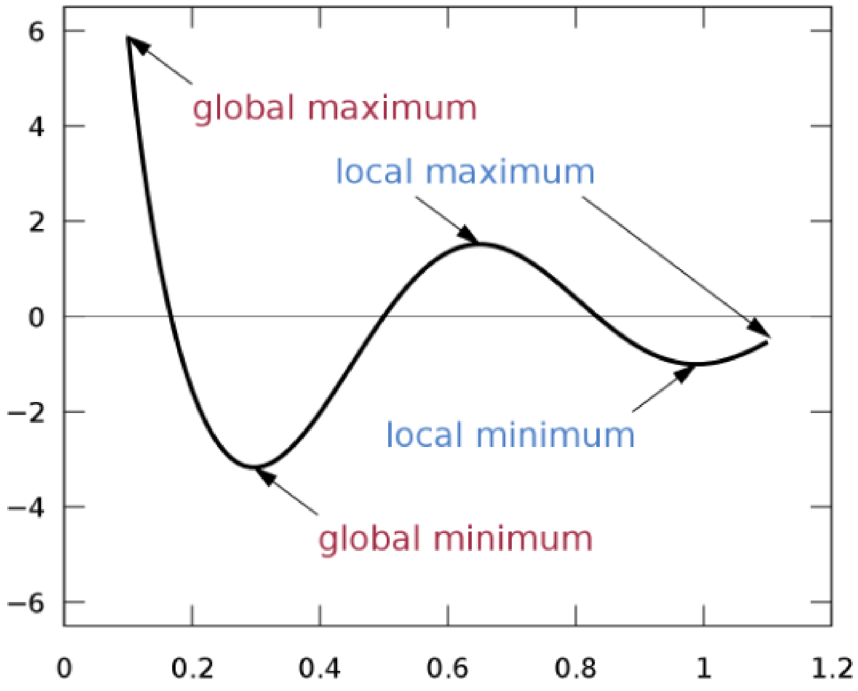
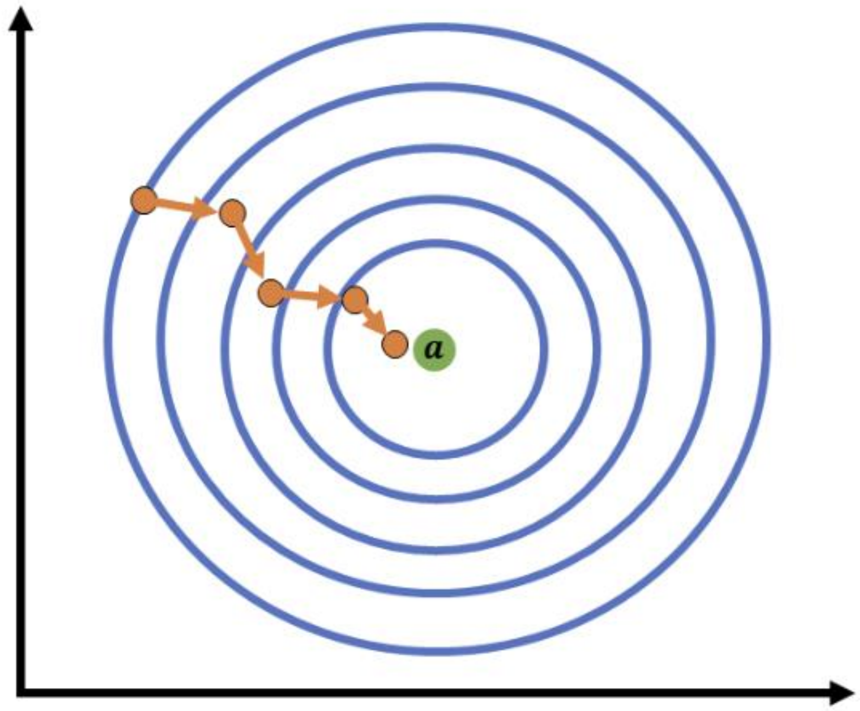


그림 5. Global minimum과 Local minimum

Basics of deep learning

- Gradient Descent
  - Solution : Stochastic Gradient descent
    - ✓ 전제 데이터셋에서 임의의 집합을 추출한다
      - ❖ 해당 집합에 대해서 기울기를 계산한다.
      - ❖ 학습속도가 빠르다.



Basics of deep learning

- Gradient Descent
  - Solution : Stochastic Gradient descent
    - ✓ Concepts
      - ❖ Batch size : 미니배치 안에 들어갈 데이터 수를 의미한다.
      - ❖ Iteration : 1미니배치 단위 학습을 의미한다. 1단계의 학습을 의미한다.
      - ❖ Epoch : 모든 미니배치에 대해 학습을 한 것을 의미한다.
    - ✓ 학습데이터가 1000개, Batch size = 100, 10개의 미니배치 생성이 된다.

Basics of deep learning

- Back - Propagation
  - Using Chain rule

$$f(x) = W_2(W_1 x), \quad \frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

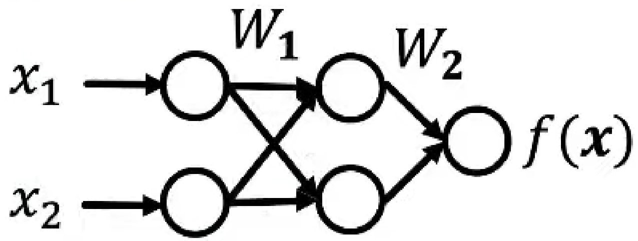
$$\text{E.g. } \nabla_x f = \frac{\partial f}{\partial (W_1 x)} \cdot \frac{\partial (W_1 x)}{\partial x}$$

Basics of deep learning

- Back - Propagation
  - Start from loss, compute the gradient

$$f(x) = g(h(x)) = W_2(W_1 x)$$

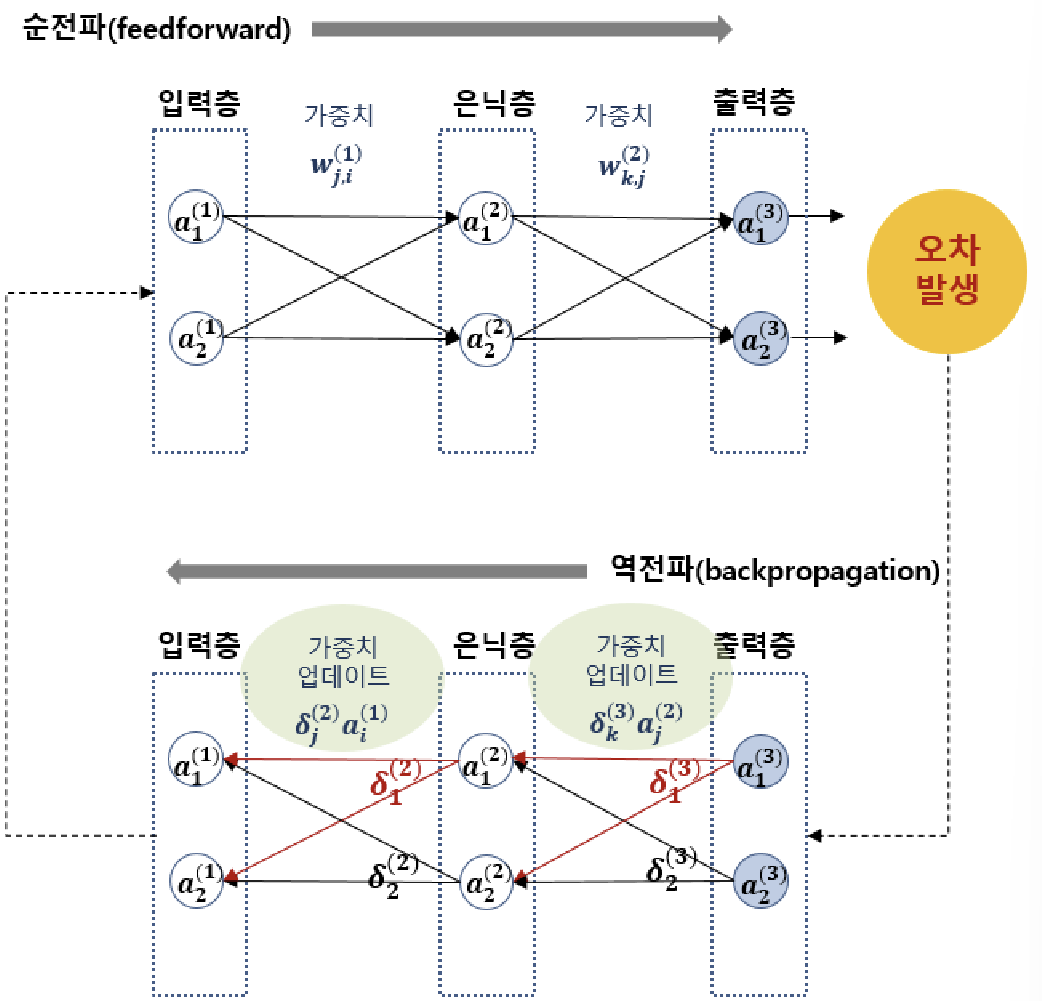
$$x \xrightarrow[\text{Multiply } W_1]{} h \xrightarrow[\text{Multiply } W_2]{} g \xrightarrow[\text{Loss}]{} \mathcal{L}$$



$$\frac{\partial \mathcal{L}}{\partial W_2} = \underbrace{\frac{\partial \mathcal{L}}{\partial f} \cdot \frac{\partial f}{\partial W_2}}_{\text{Compute backwards}}, \qquad \frac{\partial \mathcal{L}}{\partial W_1} = \underbrace{\frac{\partial \mathcal{L}}{\partial f} \cdot \frac{\partial f}{\partial W_2} \cdot \frac{\partial W_2}{\partial W_1}}_{\text{Compute backwards}}$$

Basics of deep learning

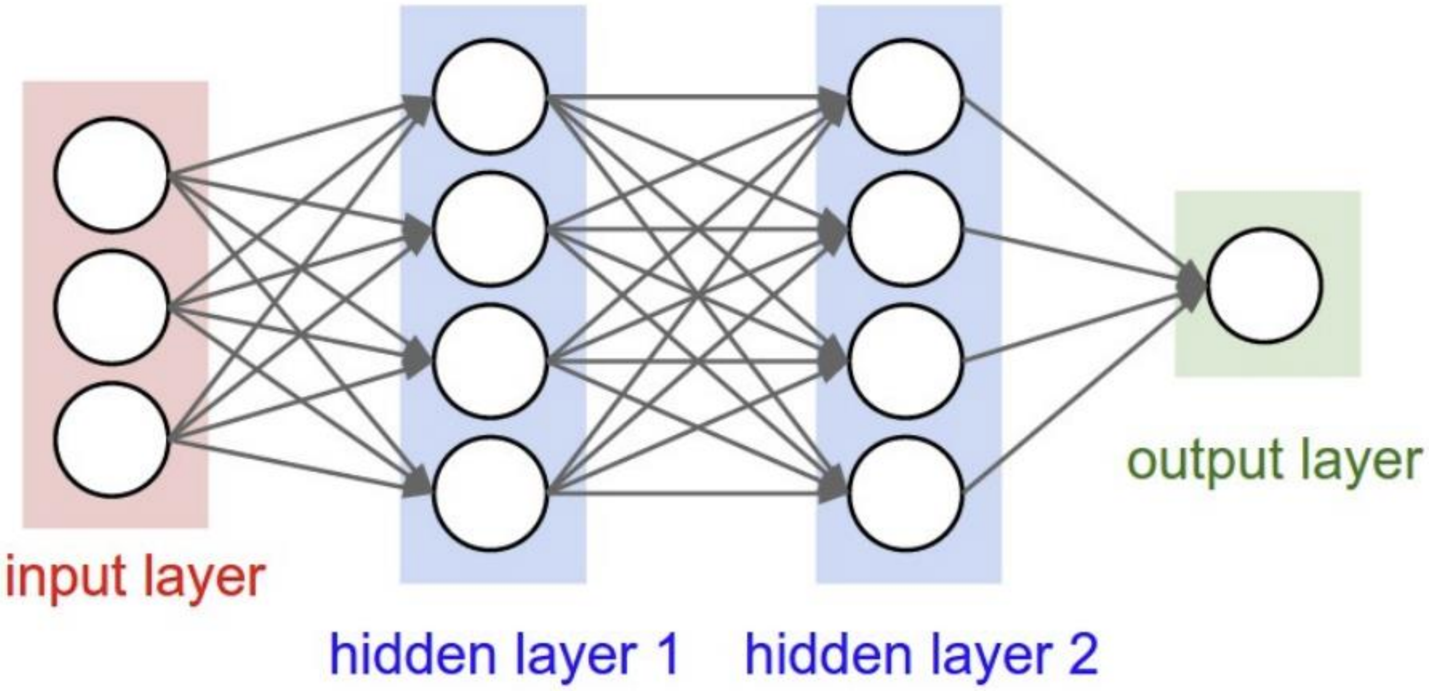
- Back - Propagation
  - Start from loss, compute the gradient

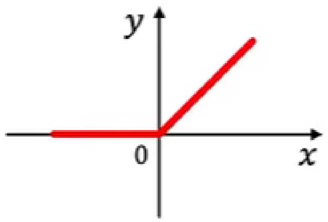


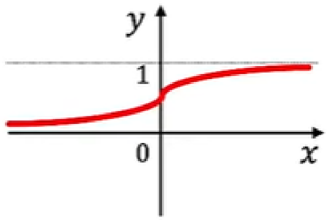
Basics of deep learning

- Multi – layer Perceptron
  - 계산에 비선형 활성함수(Relu, Sigmoid)가 들어간다.

$$x^{(l+1)} = \sigma(W_l x^{(l)} + b^l)$$





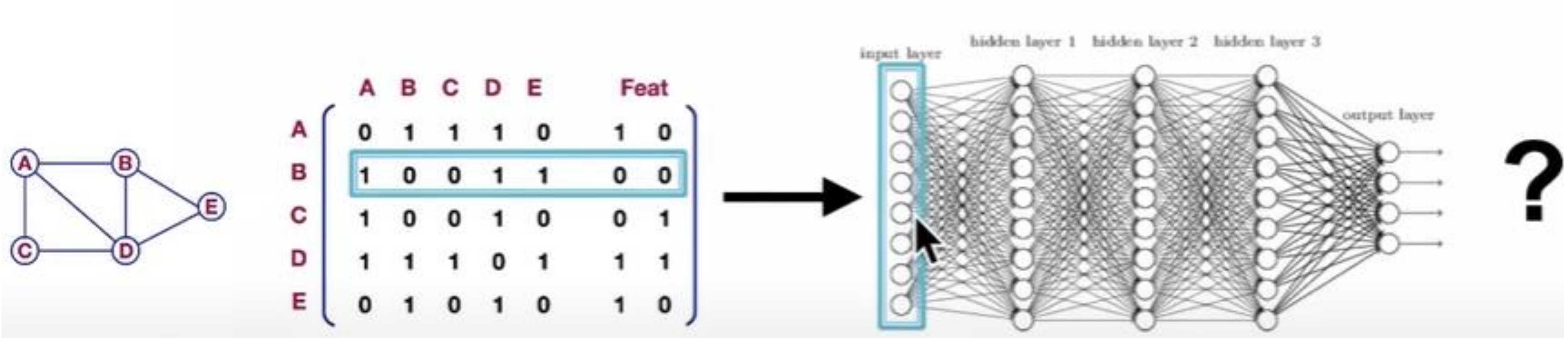


02

Deep Learning for Graphs

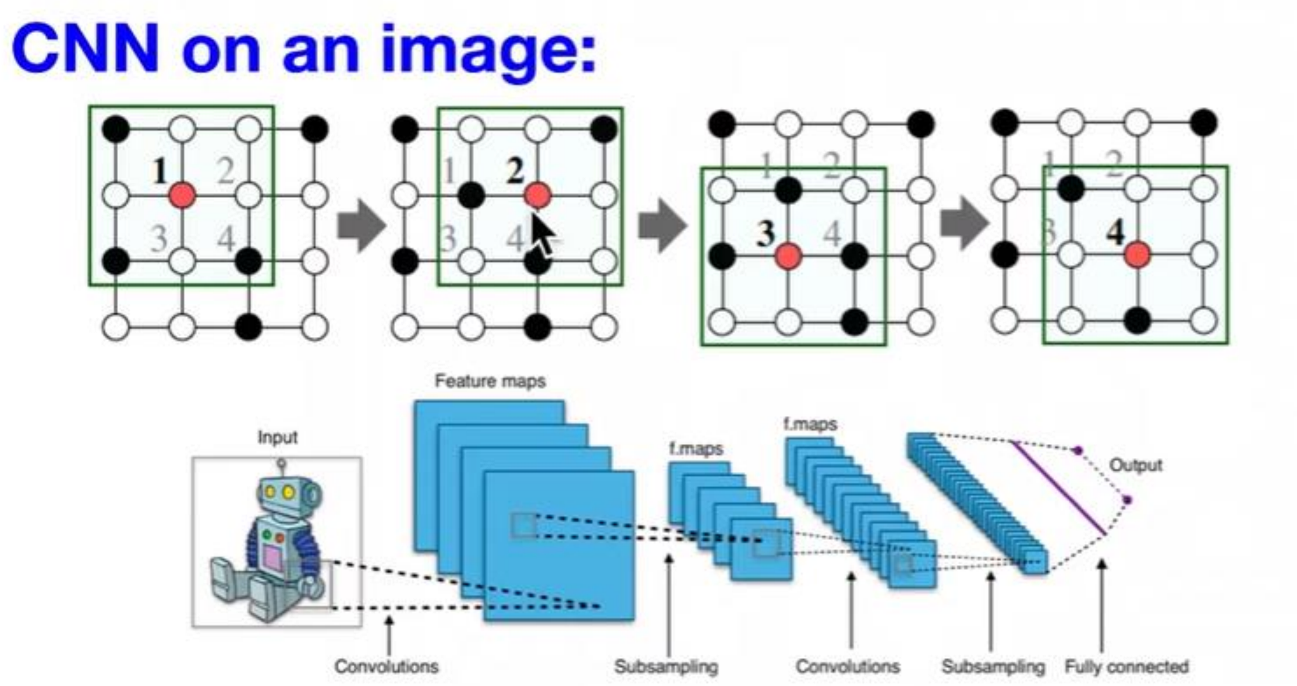
A Naiive Approach

- 인접행렬을 사용해서 네트워크를 표현하지 않는 이유
  - 훈련 수보다 많은 파라미터를 갖게 된다.
    - ✓ 과적합이 쉽게 된다.
  - 다양한 그래프에 적용이 안된다.
  - 노드순서에 따라 행렬이 바뀐다.



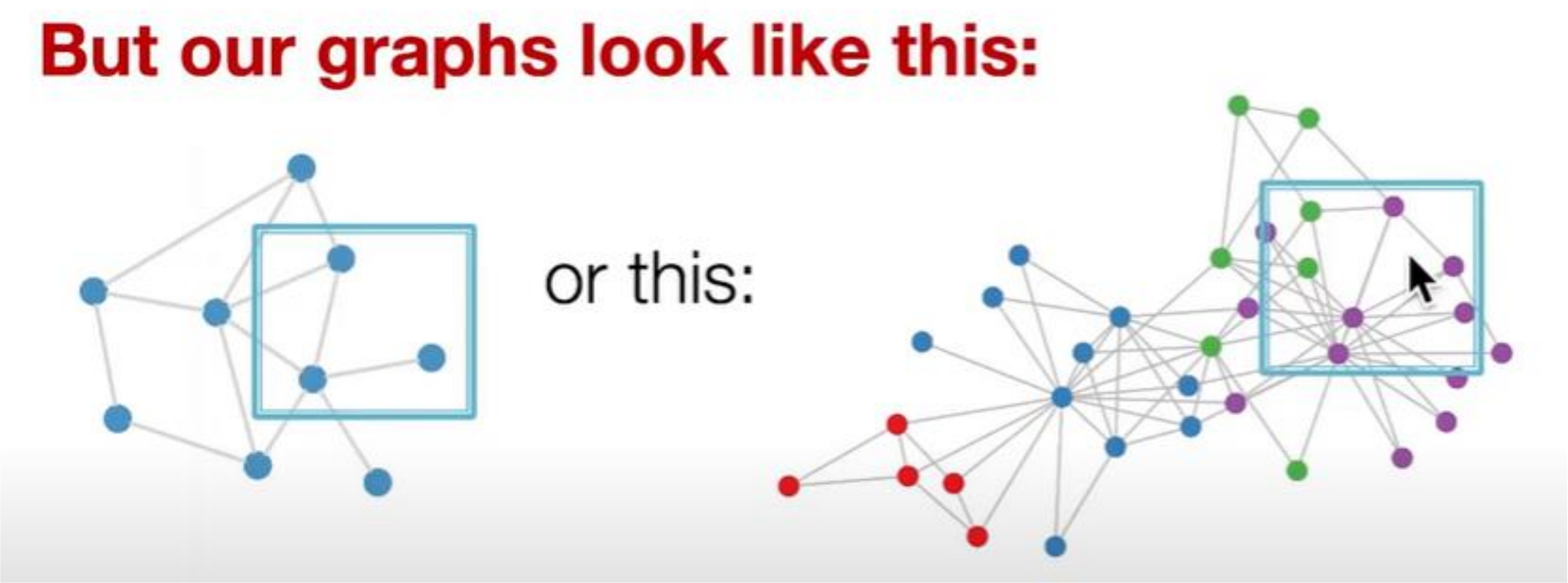
Idea : CNN

- CNN on an image



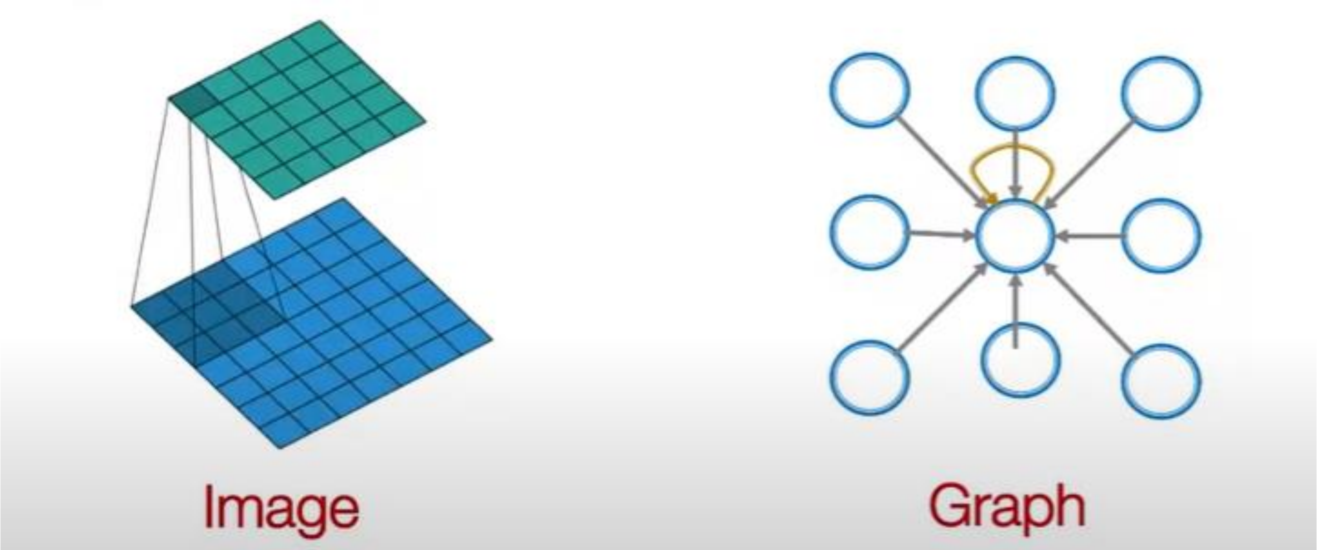
Idea : CNN

- CNN on an image



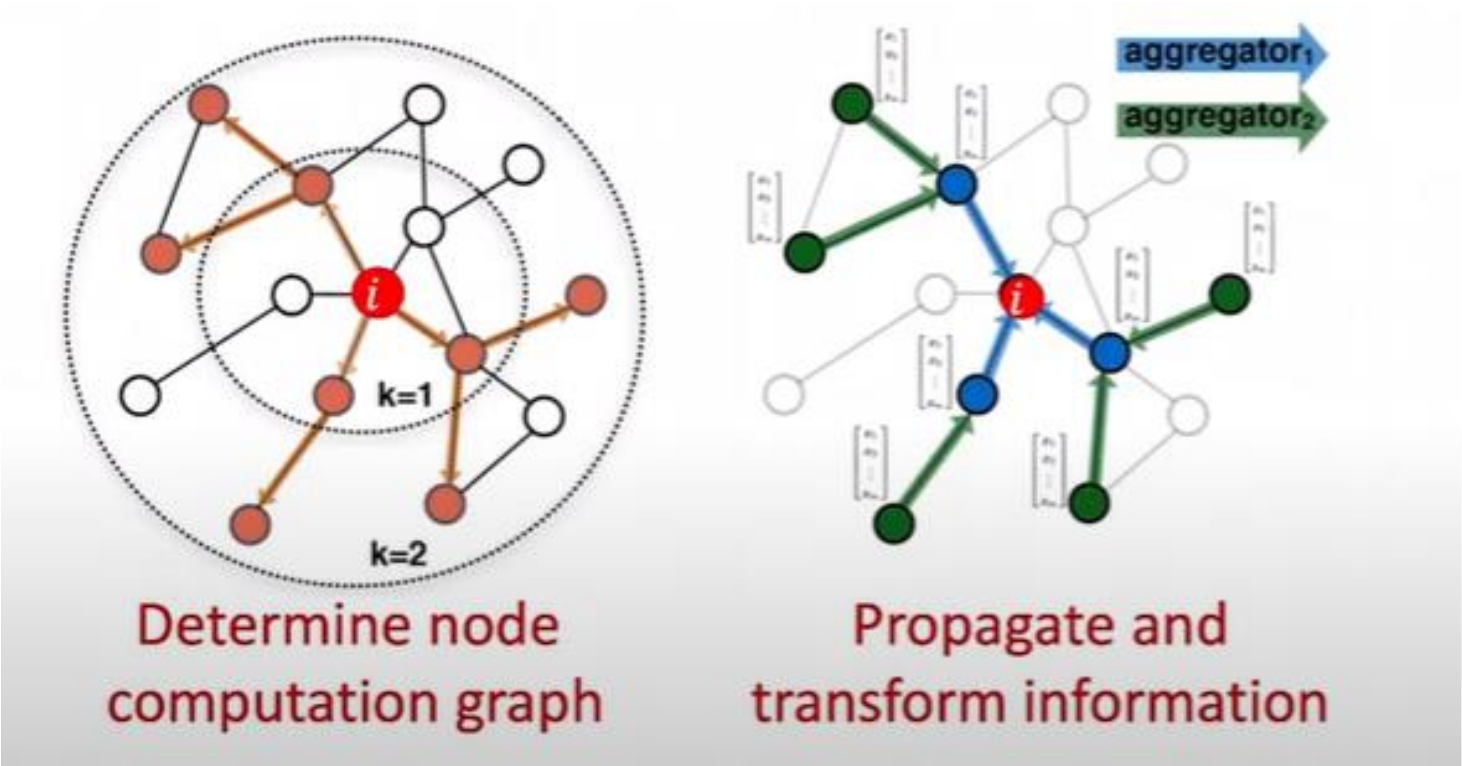
A Naiive Approach

- From Image to Graphs



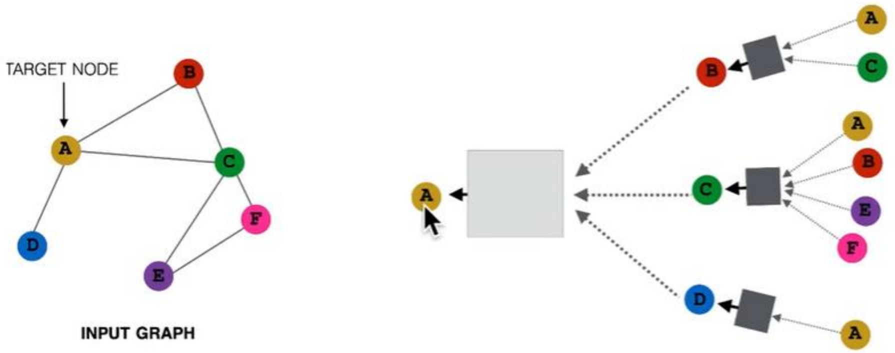
Graph Convolutional Networks

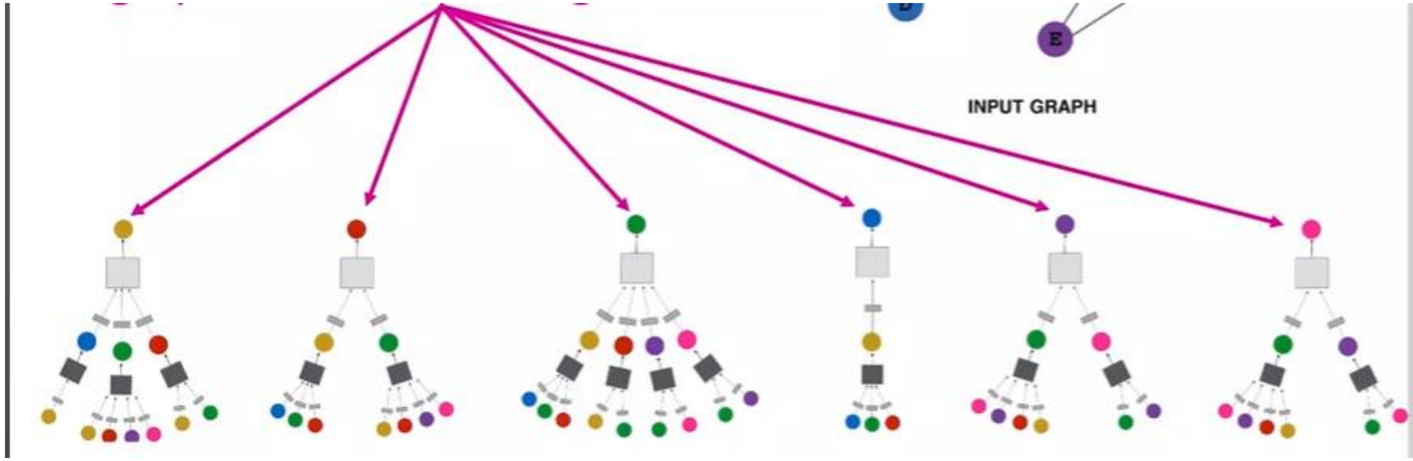
- Intuition
  - 노드의 정보는 이웃을 통해 알 수 있다.



Graph Convolutional Networks

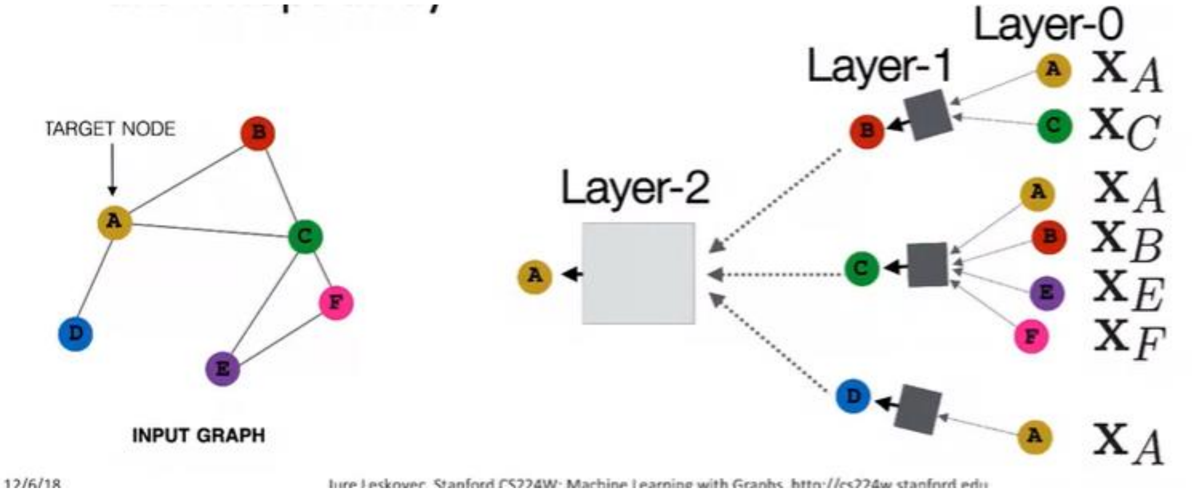
- Generate node embeddings based on local network neighborhoods
  - 노드의 정보는 이웃을 통해 알 수 있다.





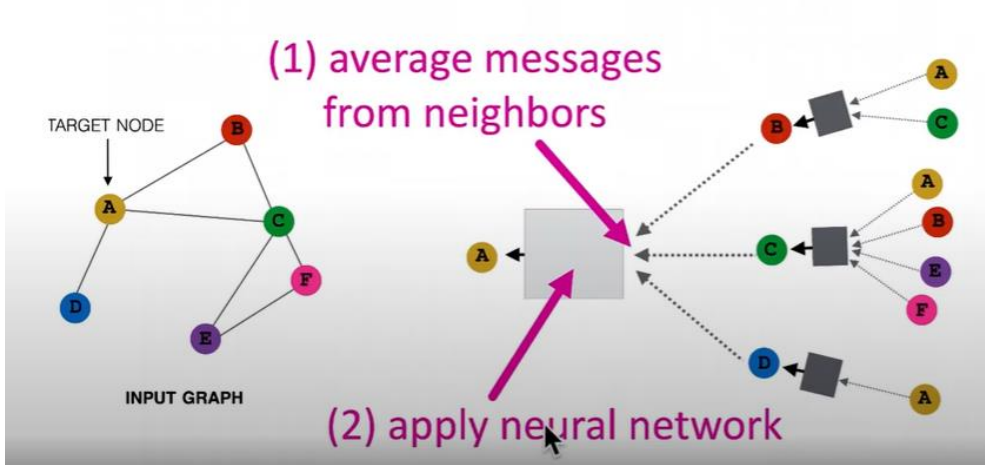
Graph Convolutional Networks

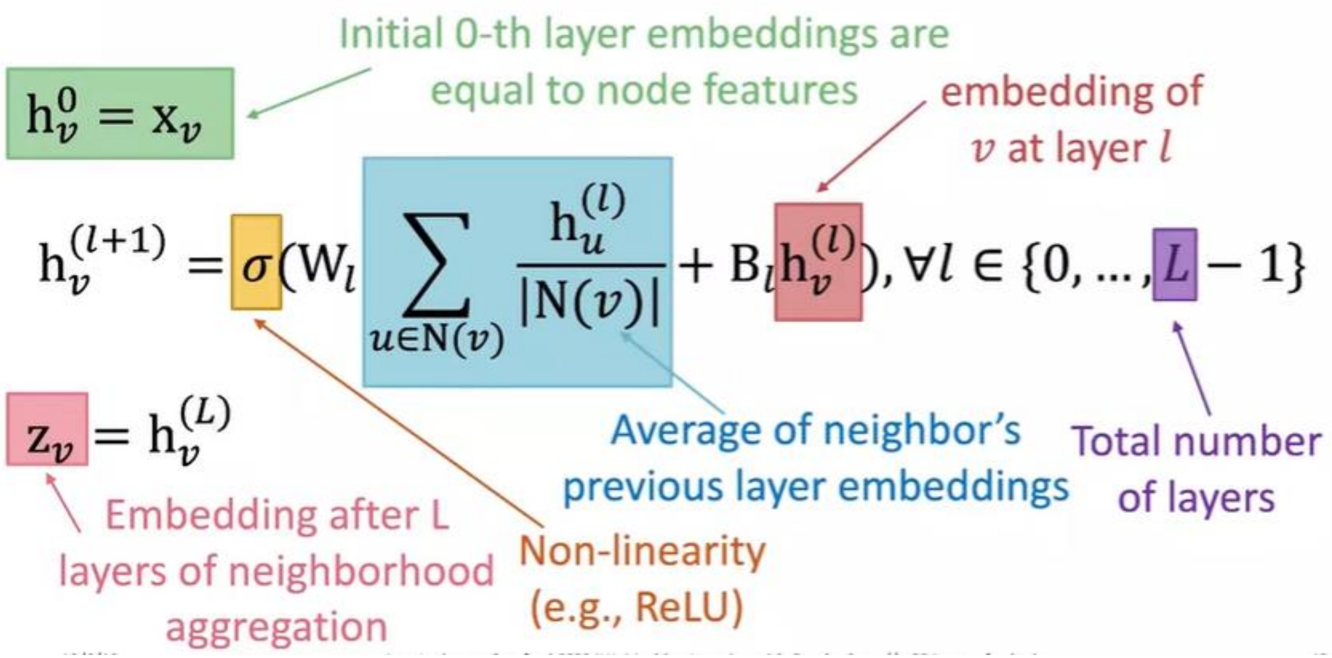
- Many Layers
  - 모델의 깊이는 임의적이다.
    - ✓ 노드 순서가 임의적이기 때문이다.
  - Xa+Xc = Layer-1
  - Layer-1 + B +… = Layer -2
  - A + C or C + A 둘 다 같은 값이 나와야 한다.
    - ✓ 순서에 영향이 없어야 한다.



Graph Convolutional Networks

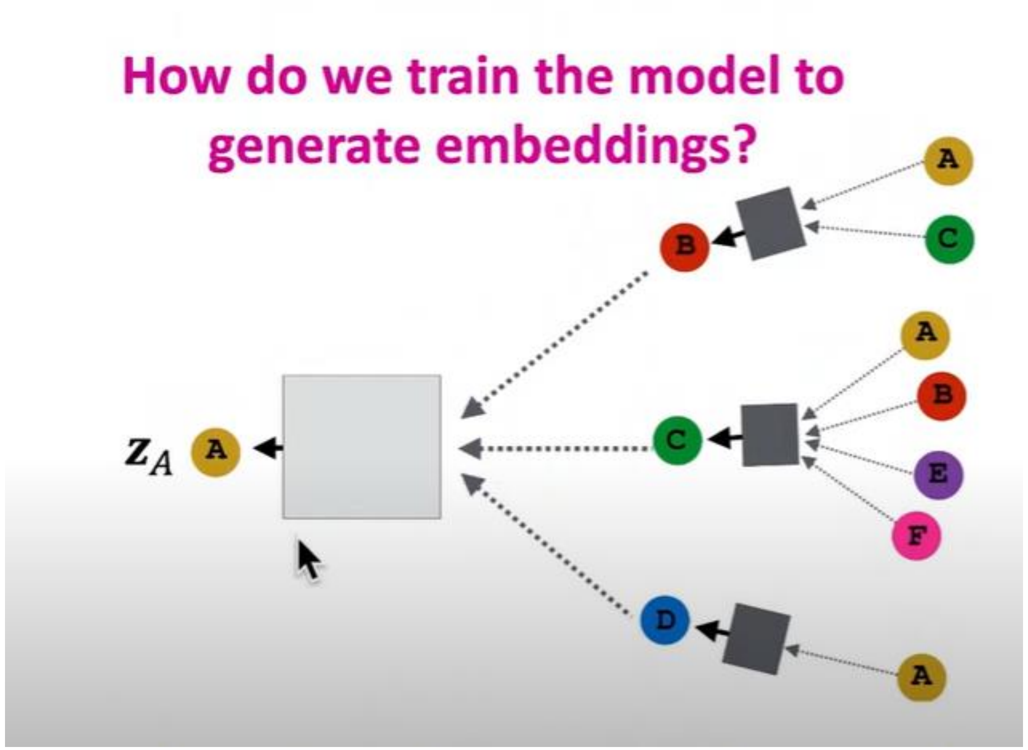
- How to calculate
  - Basic approach
    - ✓ 이웃의 정보를 평균을 낸다.
    - ✓ 그후 신경망을 적용한다.
      - ❖ 선형 비선형을 적용한다.





Graph Convolutional Networks

- How do we train the model
  - Parameter
    - ✓ W,B
      - ❖ 밑 첨자는 레이어를 의미한다. 즉 레이어 마다 다른 값을 갖는다.
  - 모든 손실함수 사용이 가능하다.
  - SGD를 파라미터 조정에 활용한다.



Trainable weight matrices (i.e., what we learn)

$$h_v^{(0)} = x_v$$

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

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

Final node embedding

Graph Convolutional Networks

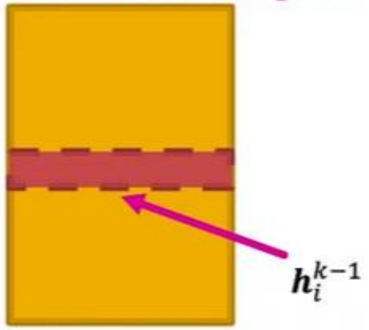
- How do we train the model
  - Matrix Formulation
    - ✓ 3개의 행렬의 결합으로 나타낼 수 있다.

- Let $H^{(l)} = [h_1^{(l)} \dots h_{|V|}^{(l)}]^T$
- Then: $\sum_{u \in N_v} h_u^{(l)} = A_{v,:} H^{(l)}$
- Let D be diagonal matrix where $D_{v,v} = \mathrm{Deg}(v) = |N(v)|$
  - The inverse of D: D^{-1} is also diagonal: $D_{v,v}^{-1} = 1/|N(v)|$
- **Therefore,**

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

$$diag(1,2,3,4) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}$$

Matrix of hidden embeddings H^{k-1}



Trainable weight matrices (i.e., what we learn)

$$h_v^{(0)} = x_v$$

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

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

Final node embedding

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

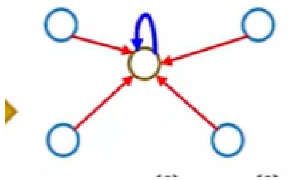
where $\tilde{A} = D^{-1} A$

Graph Convolutional Networks

- How do we train the model
  - Matrix Formulation
    - ✓ 3개의 행렬의 결합으로 나타낼 수 있다.
    - ✓ 빨간부분은 이웃노드의 합을 의미한다.
    - ✓ 파란부분은 자신이 행렬로 변하는 것을 의미한다.

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

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



Graph Convolutional Networks

- How to train GNN
  - 지도학습
    - ✓ 손실함수를 최소화 하는 것을 목표로 한다.
  - 비지도 학습
    - ✓ 노드 라벨을 모를 때, 그래프 구조를 활용한다.
    - ✓ 비슷한 노드는 비슷한 임베딩을 갖는다.

$$\min_{\Theta} \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{z}_v))$$

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

- Where $y_{u,v} = 1$ when node u and v are **similar**
- CE is the cross entropy (slide 16)
- DEC is the decoder such as inner product (lecture 4)

Graph Convolutional Networks

- How to train GNN
  - 지도학습
    - ✓ 손실함수를 최소화 하는 것을 목표로 한다.
      - ❖ 이진분류에 대한 cross entropy loss이다.
        - » 독성이 있는 확률이 1이면 y는 1의 값을 갖는다.