


Neural Network Intro

송예은



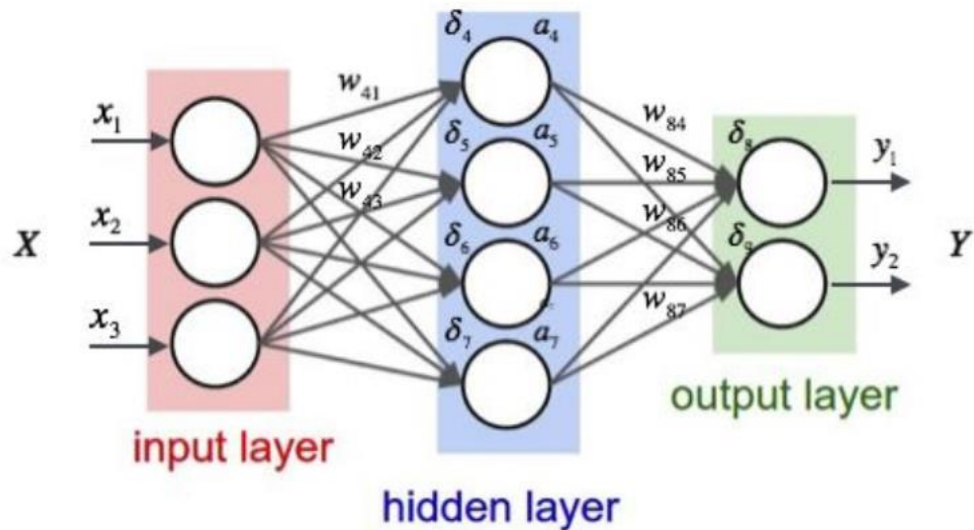
What is Deep Learning?

Deep Learning

- Deep Learning \subset Machine Learning
- Machine Learning: 인공 지능의 한 분야로, 컴퓨터가 학습할 수 있도록 하는 알고리즘과 기술을 개발하는 분야
- Deep Learning: Machine Learning의 분야 중 Deep Neural Network를 이용한 기법

Deep Learning

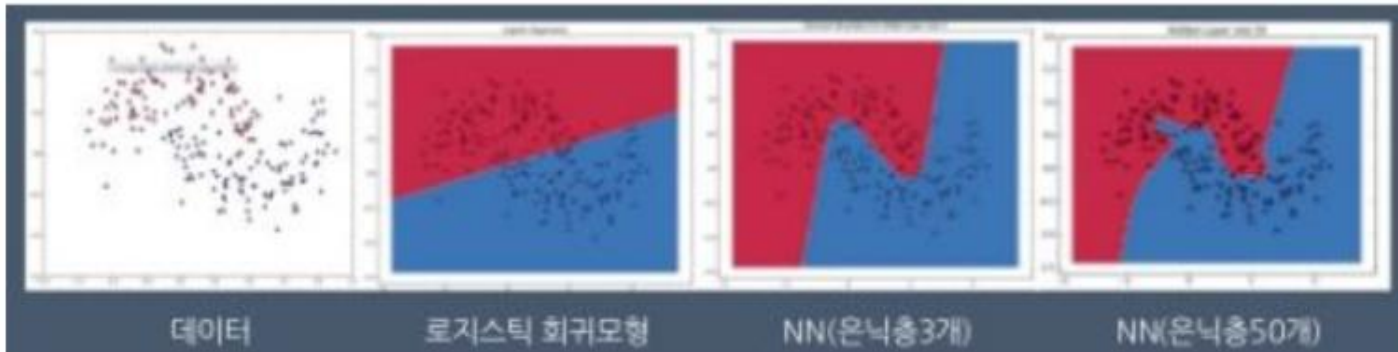
- Deep Learning: input layer, hidden layer, output layer에서 hidden layer가 2개 이상
→ DEEP!



https://miro.medium.com/max/479/1*QVlyc5HnGDWTNX3m-nIm9w.png

Deep Learning

- Hidden layer를 여러 번 결합시킨다.
- Hidden layer 개수 \uparrow \rightarrow 정확한 classification
- What about overfitting?

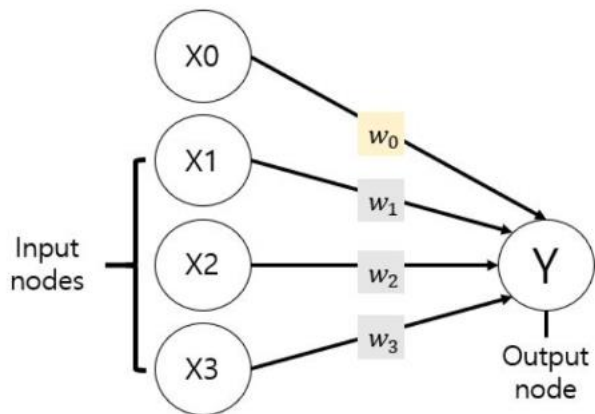


<https://www.slideshare.net/HeeWonPark11/ss-80653977>

Perceptron

Perceptron

- 신경망 구성의 기본 단위
- Input node의 선형 결합으로 output node 도출



Perceptron Model

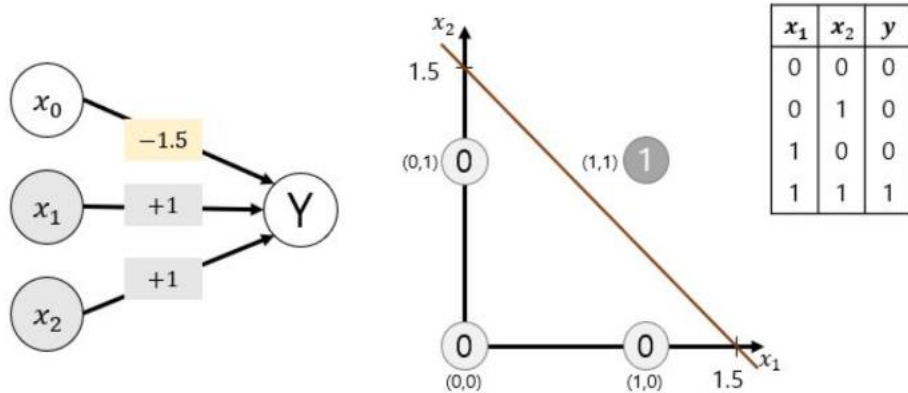
$$Y = I(\sum_j w_j X_j - w_0)$$

$$Y = \text{sign}(\sum_j w_j X_j - w_0)$$

$$Y = w_0 + \sum_j w_j X_j$$

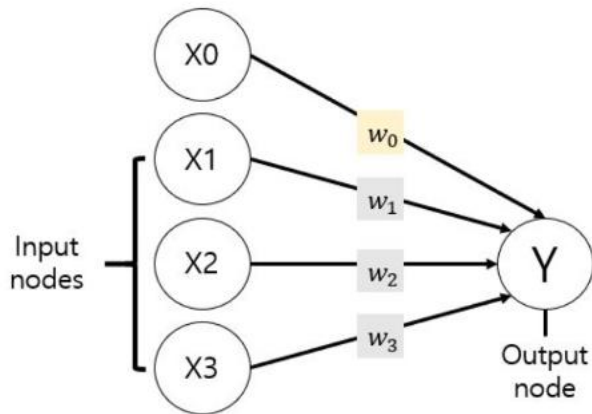
Perceptron

- 그렇다면 Perceptron으로 Data를 어떻게 분류하는가?



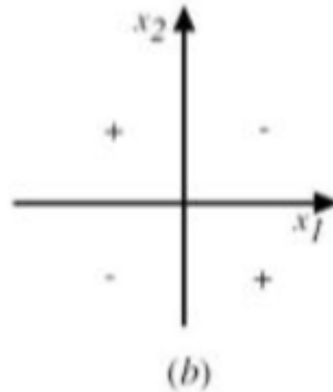
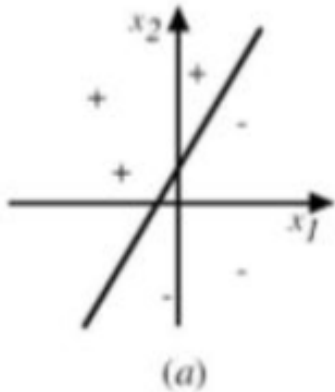
Perceptron

- Perceptron Model 최적화
- 최적화된 weight와 bias를 구해야 한다
- 예측값과 실제값의 차이를 최대한 줄이기
→ 오차 최소화 → 가중치 최적화



Multilayer Perceptron

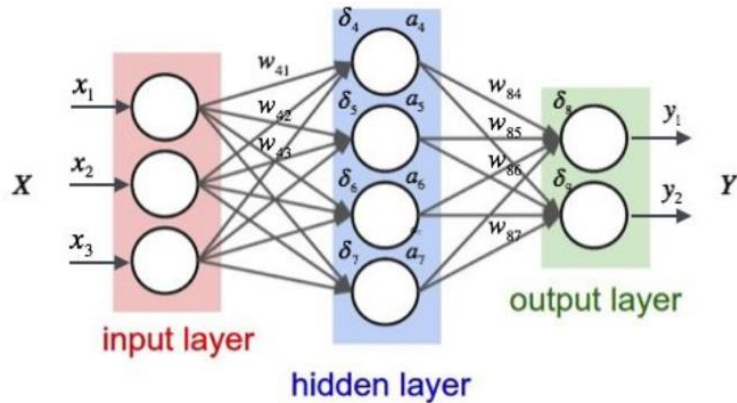
Multilayer Perceptron



- 직선 하나로는 분류 불가능! → Multilayer Perceptron!

Multilayer Perceptron

- 직선 하나로는 분류 불가능! → Multilayer Perceptron!
- “Hidden Layer”
- 여러 개의 decision boundary

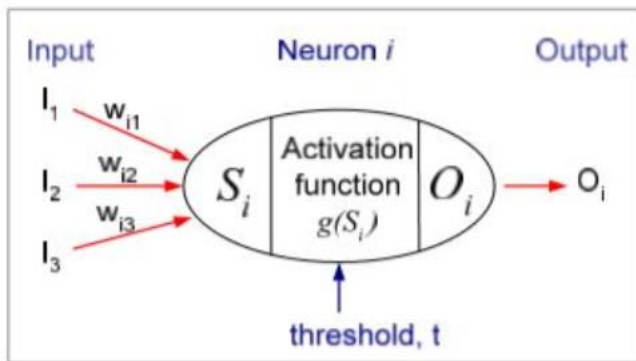


https://miro.medium.com/max/479/1*QVlyc5HnGDWTNX3m-nlm9w.png

Activation Function

Activation Function

- 한 가지 의문: 우리의 Decision Boundary는 선형이었는데 다른 형태는 불가능한가?!
- “Activation Function”: 활성화 함수 연결

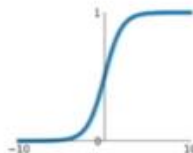


Activation Function

- Activation Function Types

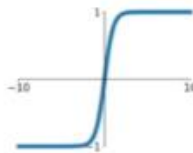
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



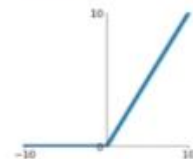
tanh

$$\tanh(x)$$



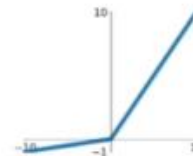
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

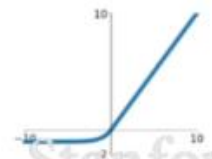


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

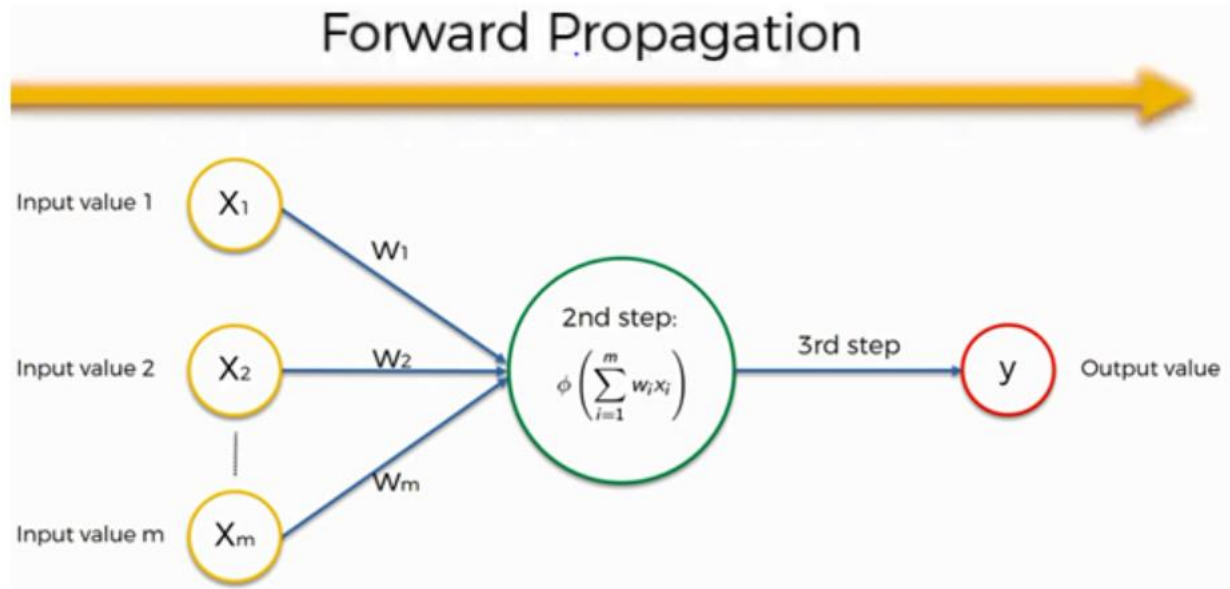
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



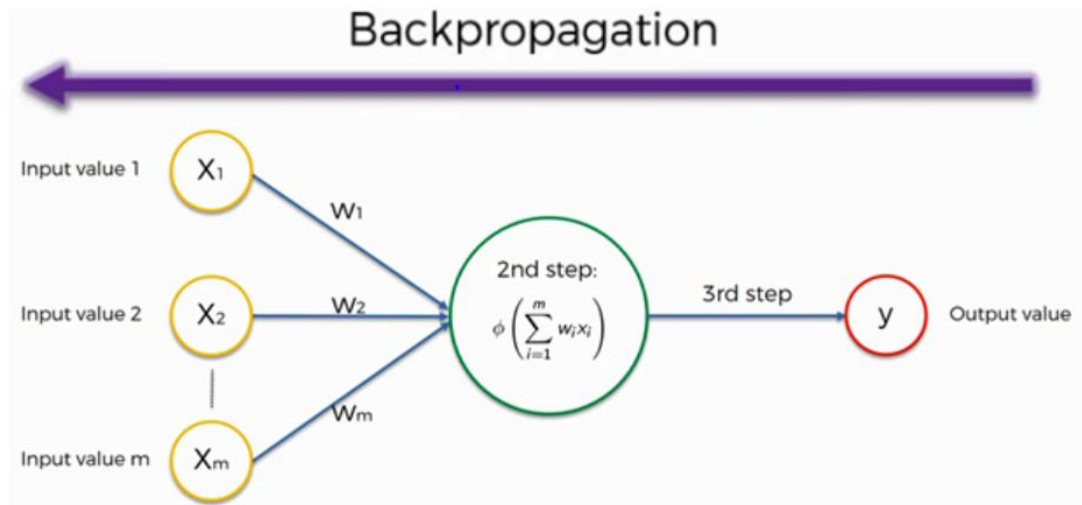
Training Neural Network

Forward Propagation



Backward Propagation

- 최적화된 weight, bias 찾기
- 오차의 기울기를 output layer에서 input layer로 전달 → “backward”



Backward Propagation

- 최적화된 weight, bias 찾기

$$\frac{\partial \ell}{\partial \hat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2} (\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^m -2(x_i - \mu_{\mathcal{B}})}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$


$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^m \frac{\partial \ell}{\partial y_i} \cdot \hat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^m \frac{\partial \ell}{\partial y_i}$$

tqdm

- Progress bars in Python
 - You get a reliable estimate of how long the progress will take.
 - You can see immediately if the progress is stuck.

```
In [*]: from tqdm import trange, tqdm_notebook
        from time import sleep
        for i in trange(4, desc='1st loop'):
            for j in trange(100, desc='2nd loop'):
                sleep(0.01)
```



```
1st loop: 50% 2/4 [00:02<00:02, 1.28s/it]
2nd loop: 100% 100/100 [00:01<00:00, 88.89it/s]
2nd loop: 100% 100/100 [00:01<00:00, 79.11it/s]
2nd loop: 4% 4/100 [00:00<00:02, 36.70it/s]
```

```
In [3]: from tqdm import trange, tqdm_notebook
```

tqdm in action



Thank You