딥러닝기초

유튜브:

https://www.youtube.com/watch?v=TuZNCNJ535I

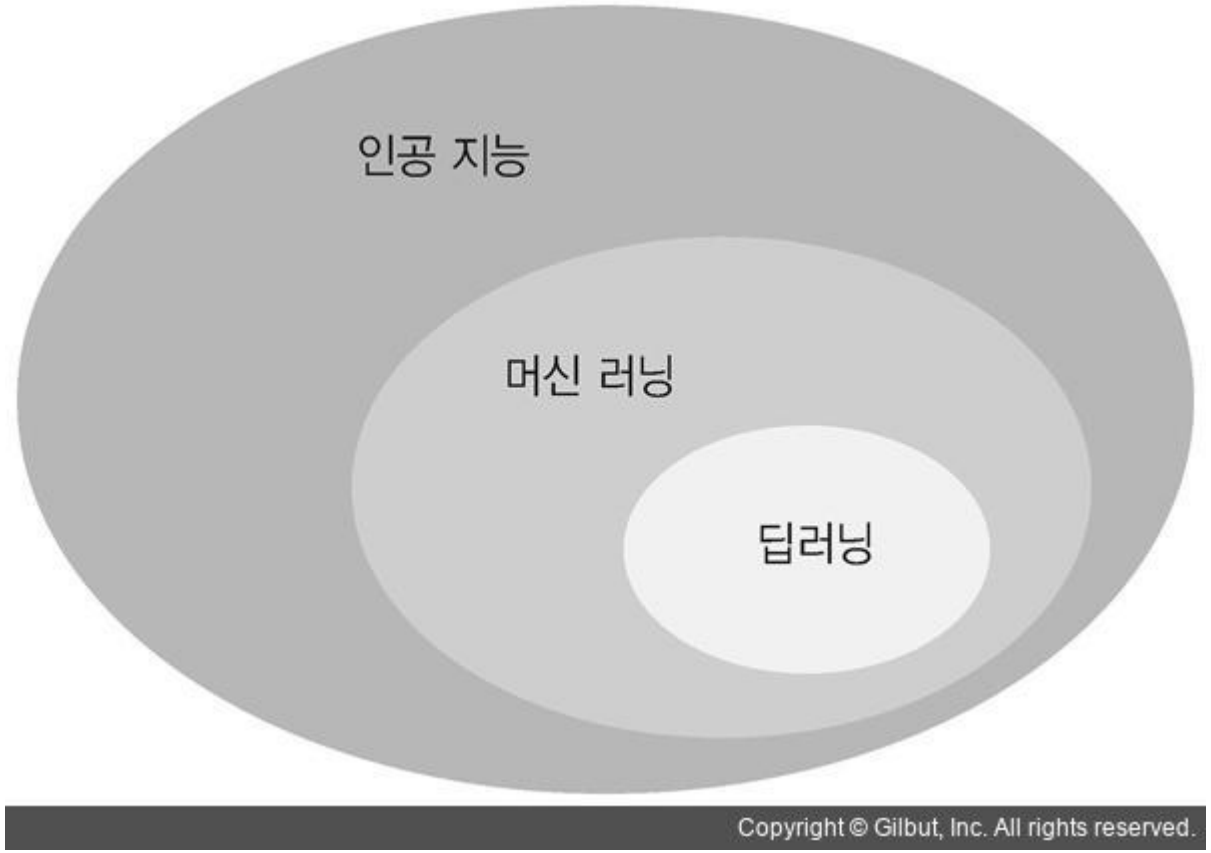
딥러닝

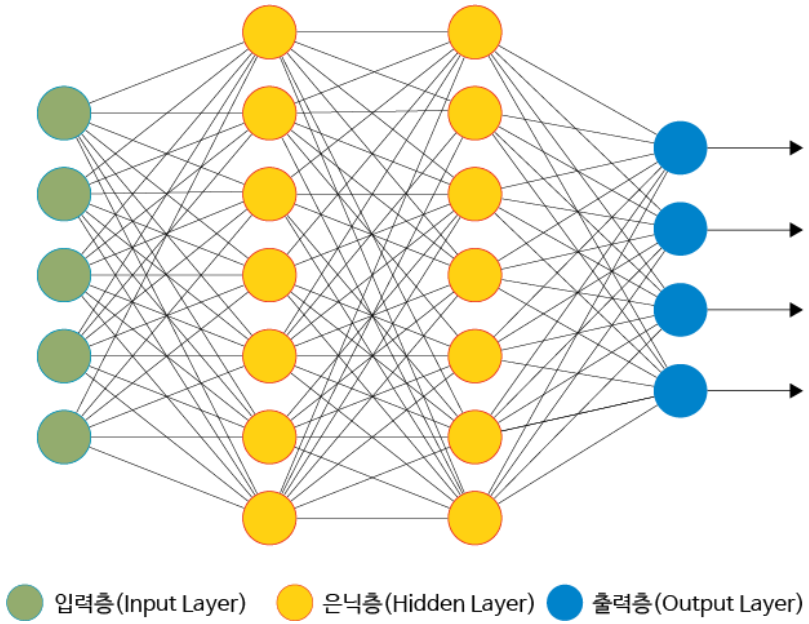
퍼셉트론

활성화 함수

손실함수

딥러닝

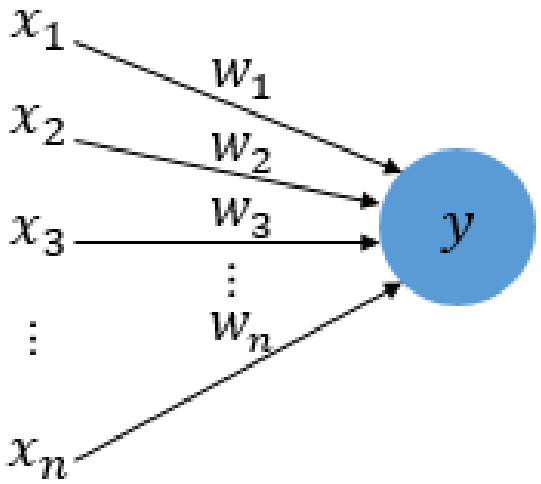




딥러닝: 여러 층을 가진 인공 신경망을 사용하여 머신러닝 학습을 수행.

퍼셉트론

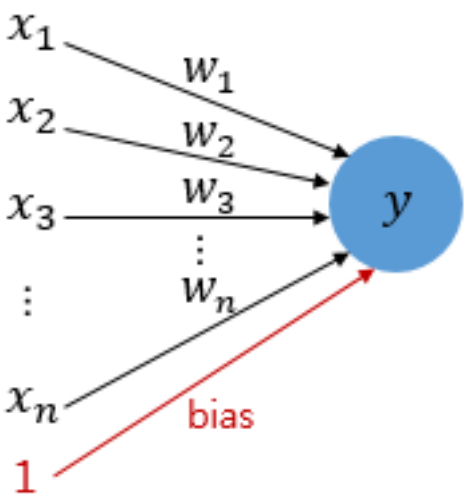
- 신경망의 기원이 되는 알고리즘
- 다수의 신호를 받아 하나의 신호를 출력



$$\sum_i^n w_i x_i \geq \theta \quad \rightarrow y = 1$$

$$\sum_i^n w_i x_i < \theta \quad \rightarrow y = 0$$

단층 퍼셉트론

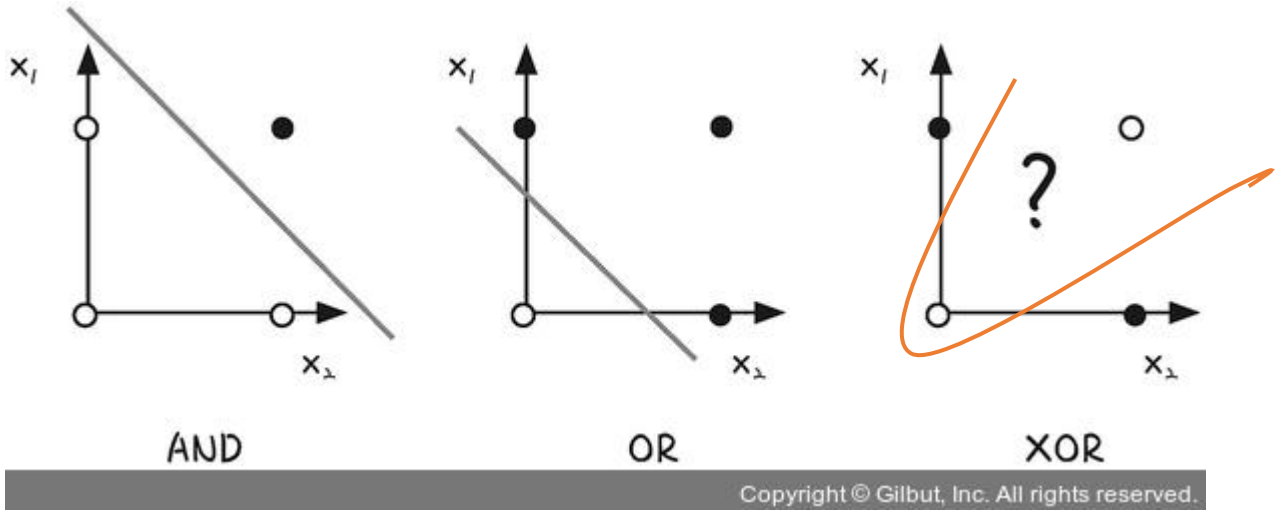


$$\sum_i^n w_i x_i + b \geq 0 \quad \rightarrow y = 1$$

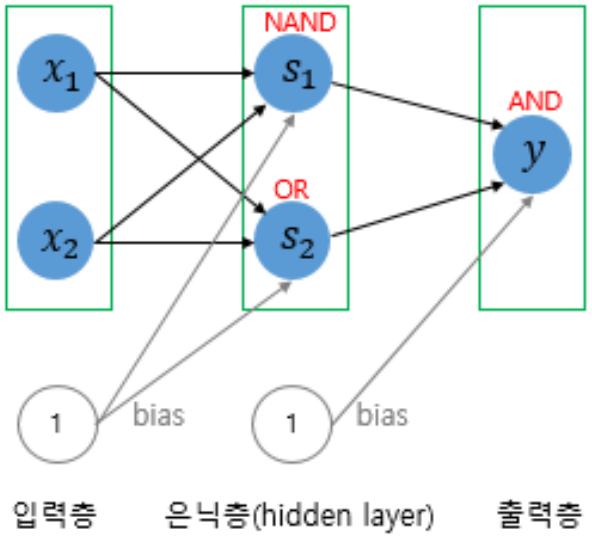
$$\sum_i^n w_i x_i + b < 0 \quad \rightarrow y = 0$$

퍼셉트론 한계

- XOR 게이트 구성 불가. (비선형 구조 표현 불가)



다층 퍼셉트론으로 표현 가능



활성화 함수

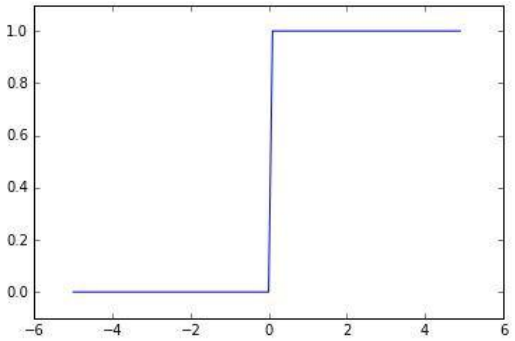
- 활성화 함수:

  입력 신호의 총합을 출력신호로 바꾸어주는 함수

$$\sum_i^n w_i x_i + b \geq 0 \rightarrow y = 1$$
$$\sum_i^n w_i x_i + b < 0 \rightarrow y = 0$$

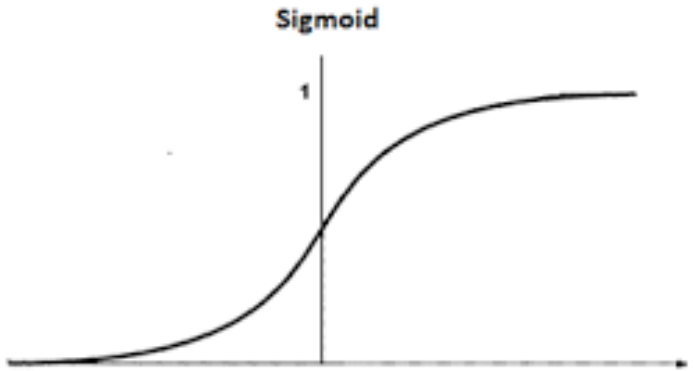
$$y = h(\sum_i^n w_i x_i + b)$$
$$h(x) = \begin{cases} 0\ (x \leq 0) \\ 1\ (x > 0) \end{cases}$$

계단함수



- 시그모이드 함수

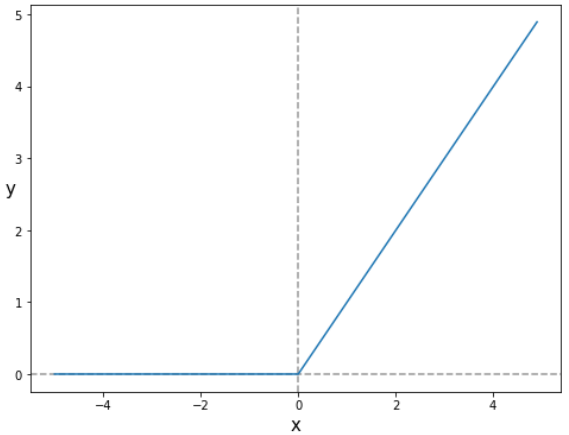
$$h(x) = \frac{1}{1+\exp(-x)}$$



- ReLU 함수

$$h(x) = \begin{cases} x\ (x > 0) \\ 0\ (x \leq 0) \end{cases}$$

ReLU



신경망 활성화 함수 = 비선형함수

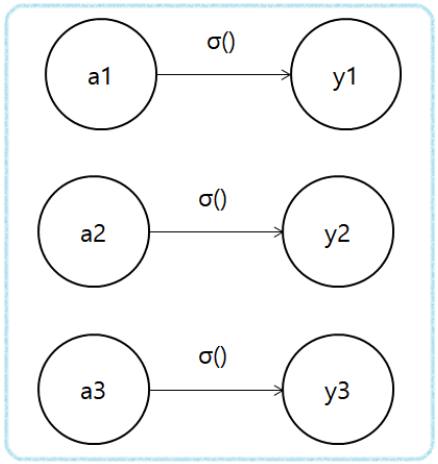
(선형함수로 하면 신경망의 층을 깊게하는 의미가 없다.=은닉층이 없는 네트워크 와 같은 기능)

출력층에서의 활성화 함수

•분류: 데이터가 어느 클래스에 속하느냐. - 소프트맥스 함수

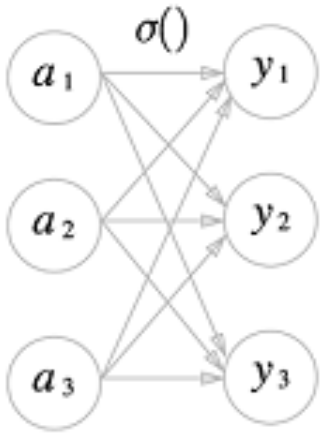
•회귀: 데이터에서 (연속적인) 수치를 예측. - 항등함수

•항등 함수: 입력 값이 출력 그대로



- 소프트맥스 함수: $y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n}\exp(a_i)} = \frac{C\exp(a_k)}{C\sum_{i=1}^{n}\exp(a_i)}$$
$$= \frac{\exp(a_k + \log C)}{\sum_{i=1}^{n}\exp(a_i + \log C)}$$
$$= \frac{\exp(a_k + C')}{\sum_{i=1}^{n}\exp(a_i + C')}$$



소프트맥스 함수 출력은 0에서 1.0 사이 실수.

출력의 총합= 1

학습에서는 출력층에 소프트맥스 함수 사용, 추론에서는 생략.

손실함수

- 신경망 학습에서 사용하는 지표. (신경망 성능의 '나쁨을' 나타냄)
- 신경망이 훈련 데이터를 얼마나 잘 처리하지 '못' 하느냐를 나타냄

- 평균 제곱 오차

$$E = \frac{1}{2}\sum_k (y_k - t_k)^2$$

```
import numpy as np

# 정답은 2
t = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]

# 2일 확률이 60%
y = [0.1, 0.05, 0.6, 0.0, 0.05, 0.1, 0.0, 0.1, 0.0, 0.0]

# 7일 확률이 60%
y2 = [0.1, 0.05, 0.1, 0.0, 0.05, 0.1, 0.0, 0.6, 0.0, 0.0]

def mean_squared_error(y, t):
    return 0.5 * np.sum((y-t)**2)

mean_squared_error(np.array(y), np.array(t))
```

원 핫 인코딩

```
0.09750000000000003
```

```
mean_squared_error(np.array(y2), np.array(t))
```

```
0.5975
```

- 교차 엔트로피

$$E = -\sum_k t_k log y_k$$

```
def cross_entropy_error(y, t):
    delta = 1e-7
    return -np.sum(t * np.log(y + delta))

# 정답은 2
t = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
# 2일 확률이 60%
y = [0.1, 0.05, 0.6, 0.0, 0.05, 0.1, 0.0, 0.1, 0.0, 0.0]
# 7일 확률이 60%
y2 = [0.1, 0.05, 0.1, 0.0, 0.05, 0.1, 0.0, 0.6, 0.0, 0.0]

cross_entropy_error(np.array(y), np.array(t))
```

```
0.510825457099338
```

```
cross_entropy_error(np.array(y2), np.array(t))
```

```
2.302584092994546
```

손실함수

- 딥러닝 목표 = 높은 정확도 도출
- But '정확도'가 아닌 손실함수를 지표로 삼는 이유?

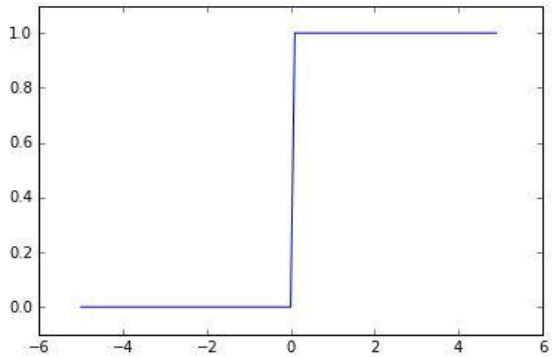
최적의 매개변수를 찾는 것 = 손실함수를 값을 가능한 작게 하는 매개변수

=> 매개변수의 미분을 계산, 미분 값을 단서로 매개변수 값 갱신 반복

미분 =가중치 매개변수의 값을 아주 조금 변화시켰을 때 손실함수가 어떻게 변하나

정확도를 지표로 하면 매개변수의 미분이 대부분 장소에서 0이됨

ex) 정확도가 개선 되도 32-> 33 등 불 연속적인 띄엄띄엄한 값으로 바뀜 , 손실함수는 0.9254 -> 0.9343 처럼 연속적 변화.



계단함수를 활성화 함수로
사용하지 않는 이유

Q & A