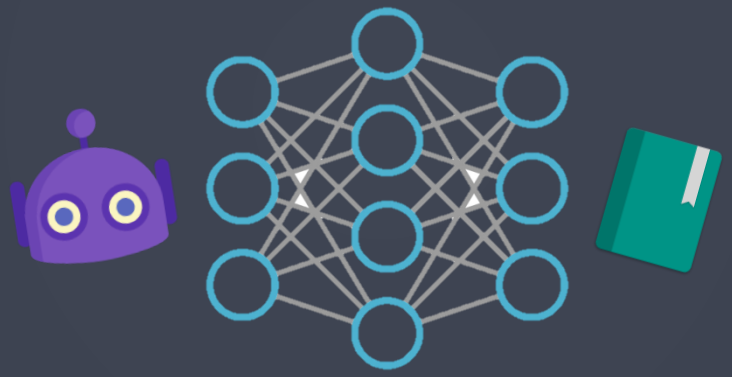
Deep Learning

Chapter 3 활성화 함수,오차 역전파, 경사하강법 (Activation Function, Back Propagation, Gradient Descent Algorithm)



START

Smart Media
스마트미디어인재개발원

학습목표

- 활성화 함수의 개념을 이해 하고 종류를 알 수 있다.
- 오차역전파의 개념을 이해 할 수 있다.
- 다양한 경사하강법 종류를 알 수 있다.
- Keras를 활용해 다양한 경사하강법을 적용 할 수 있다.

활성화 함수(Activation Function)

활성화 함수(Activation Function)

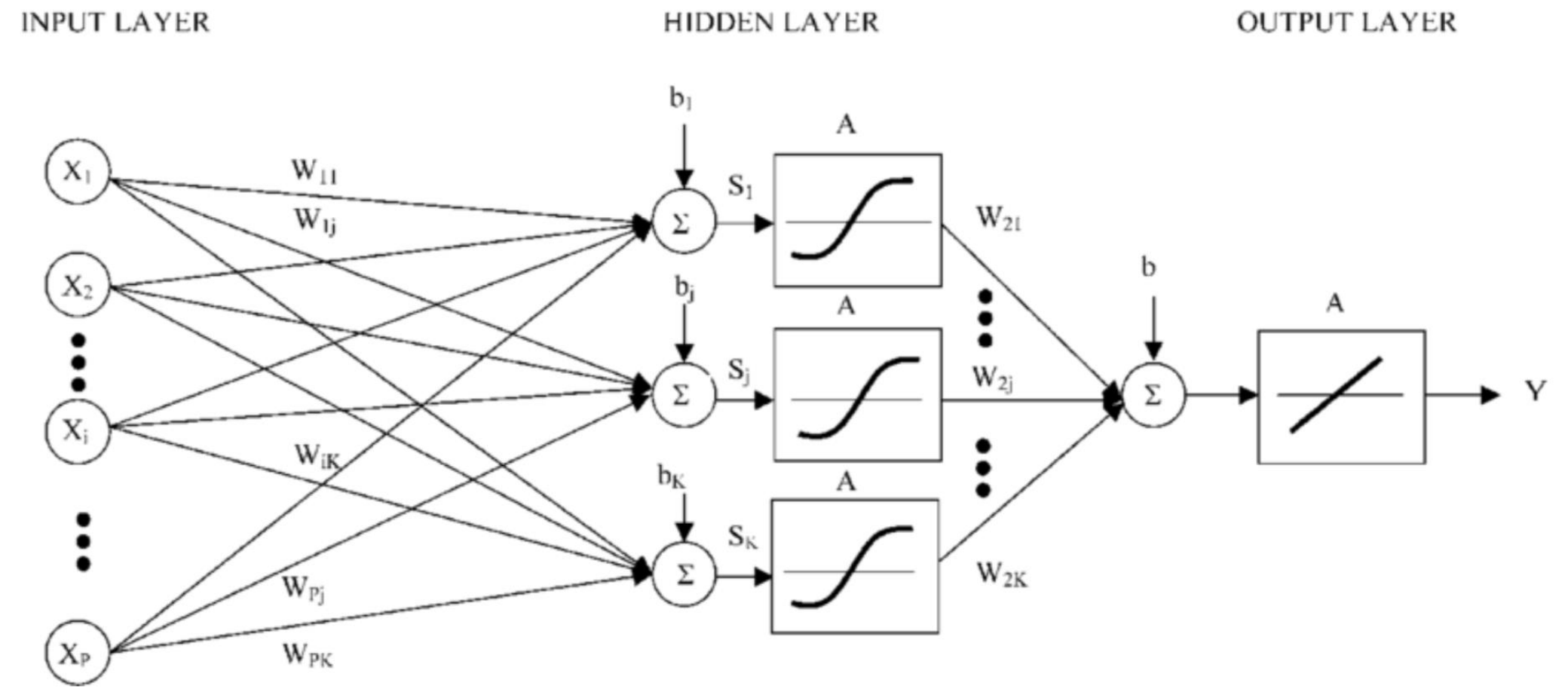
- 신경망은 선형회귀와 달리 한 계층의 신호를 다음 계층으로 그대로 전달하지 않고 비선형적인 활성화 함수를 거친 후에 전달한다.

- 이렇게 하는 이유는 신경망을 모방하여 사람처럼 사고하는 인공지능 기술을 구현하기 위함이다.

- 실제로 비선형의 활성화 함수를 도입한 신경망이 잘 동작하고 있다.

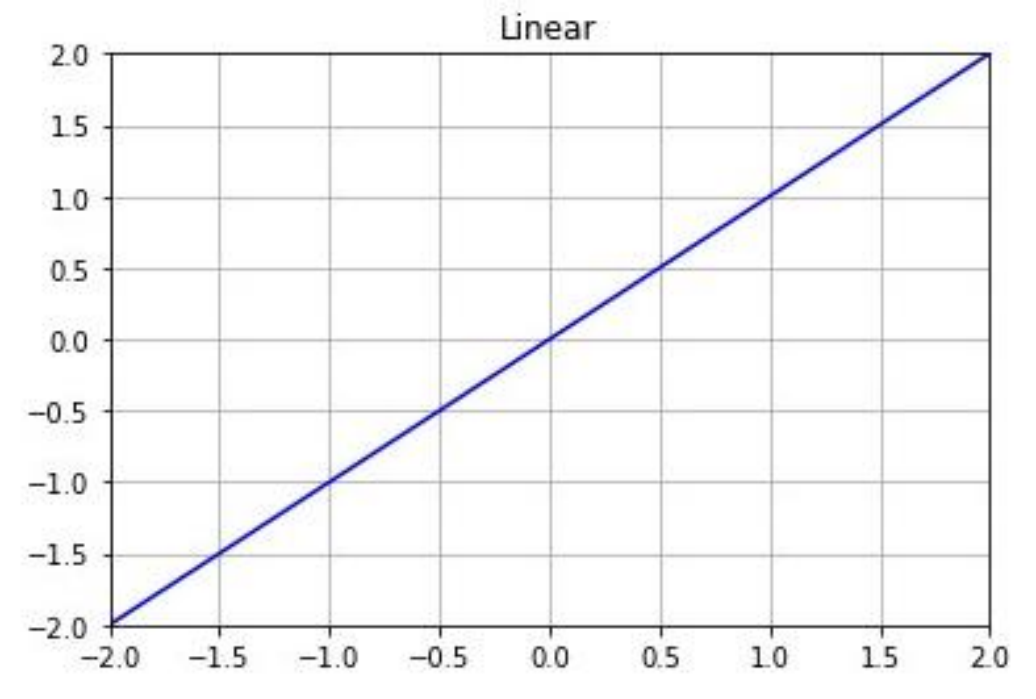
활성화 함수(Activation Function)

층에 따라 다른 활성화 함수를 사용 할 수 있다.



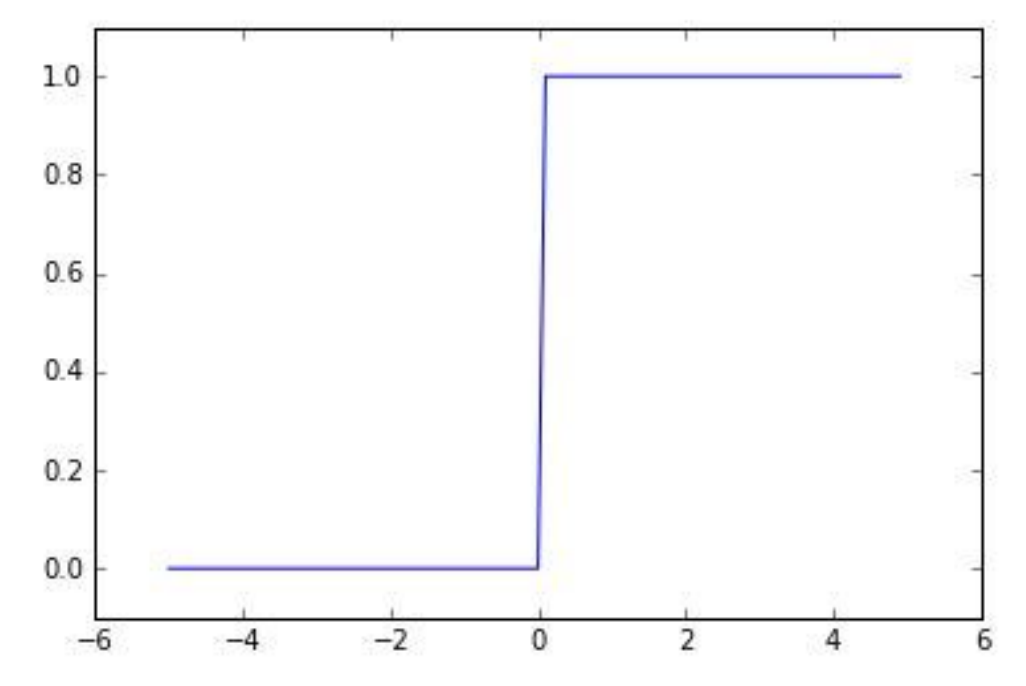
활성화 함수(Activation Function)

Linear function(항등 함수=선형 함수)



활성화 함수(Activation Function)

Step function(계단 함수)

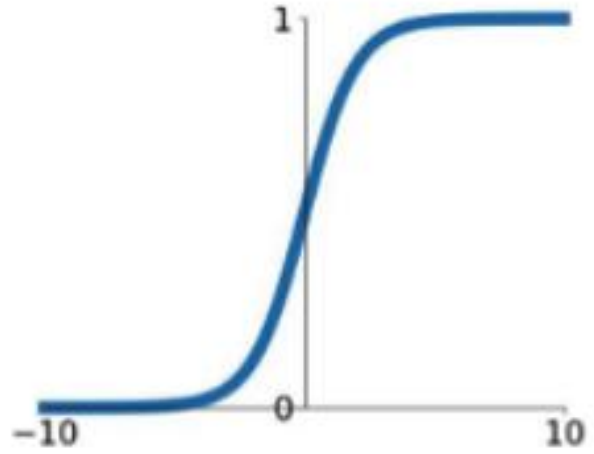


활성화 함수(Activation Function)

Sigmoid 함수

Sigmoid

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



활성화 함수(Activation Function) 정리

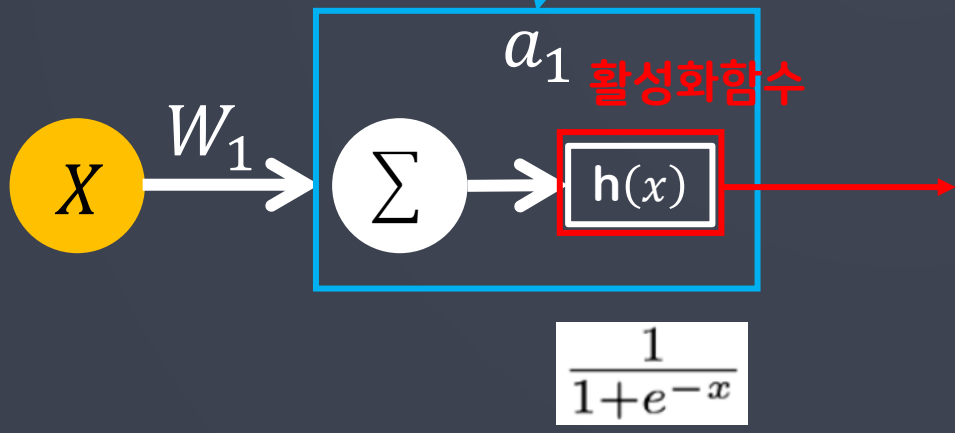
1. Step와 Sigmoid의 차이

- 선형 모델이 학습하기 위해서 경사하강법(cost 함수를 미분)을 사용.

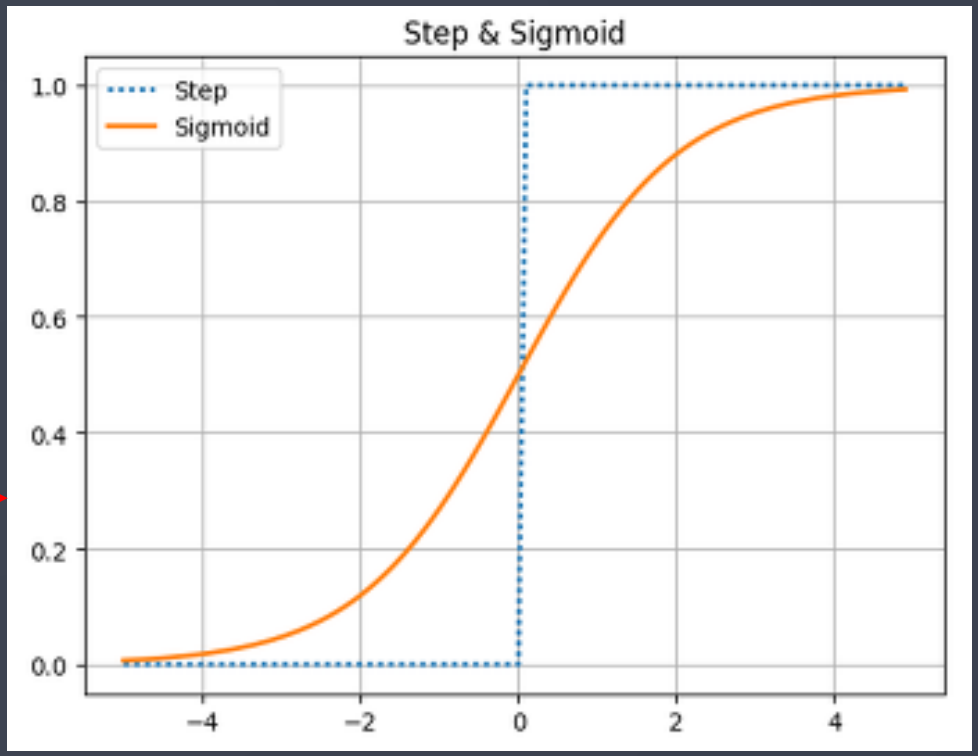
$$cost = \frac{1}{m}\sum_{i=1}^{m}(H(x_i) - y_i)^2$$

예측값

퍼셉트론



$$\frac{1}{1+e^{-x}}$$



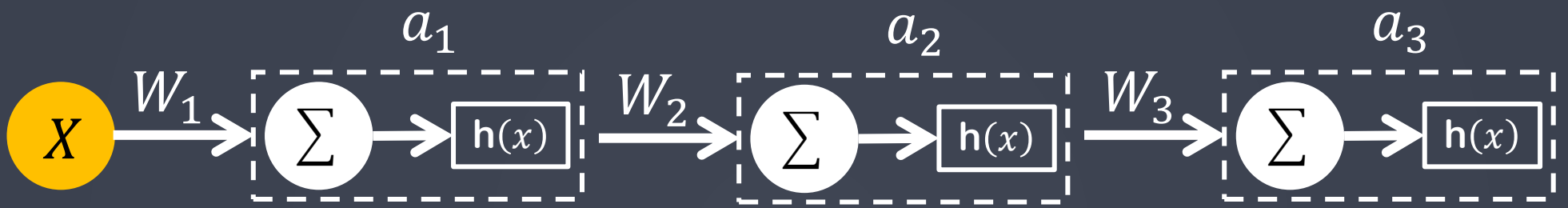
활성화 함수(Activation Function) 정리

2. 중간층에 활성화 함수로 비선형 함수를 사용하는 이유

- 계단 함수(step)와 시그모이드 함수(sigmoid)는 비선형 함수이다.
- 활성화 함수로 선형함수(linear) ex) h(x) = cx 를 사용하면 중간층(은닉층)을 여러 개 구성한 효과를 살릴 수 없다.

활성화 함수(Activation Function) 정리

3. 중간층에 활성화 함수로 선형 함수를 사용하게 된다면



✓ a1 = W1 * X * c ✓ a2 = W2 * a1 * c ✓ a3 = W3 * a2 * c

- a3 = W3 * (W2 * (W1 * X * c) * c) * c
  - (W1 * X * c) → a1
  - (W2 * a1 * c) → a2
- a3 = c * X ≒ a1

→ 신경망을 통과해도 최종 결과는 첫번째 결과와 다르지 않음

문제 유형에 따른 활성화 함수와 손실함수의 종류

소프트맥스(softmax) 함수

다중분류에서 레이블 값에 대한 **예측 확률의 합을 1**로 설정

sigmoid에 비해 예측 오차의 평균을 줄여주는 효과

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

문제 유형에 따른 활성화 함수와 손실함수의 종류

소프트맥스(softmax) 함수 코드 구현

```python
import numpy as np

def softmax(x):
    e_x = np.exp(x-x.max())
    return e_x/e_x.sum()
```

```python
x = np.array([1.0,1.0,2.0])
x
```

```
array([1., 1., 2.])
```

```python
y = softmax(x)
y
```

```
array([0.21194156, 0.21194156, 0.57611688])
```

```python
y.sum()
```

```
1.0
```

문제 유형에 따른 활성화 함수와 손실함수의 종류

유 형	출력층 활성화 함수 (activation)	손실함수(=비용함수) (loss)
회귀	linear(항등 함수)	mse
2진 분류	sigmoid(로지스틱 함수)	binary_crossentropy
다중 분류	softmax(소프트맥스 함수)	categorical_crossentropy

코드 실습

유방암 데이터 신경망으로 풀기 (2진 분류)

코드 실습

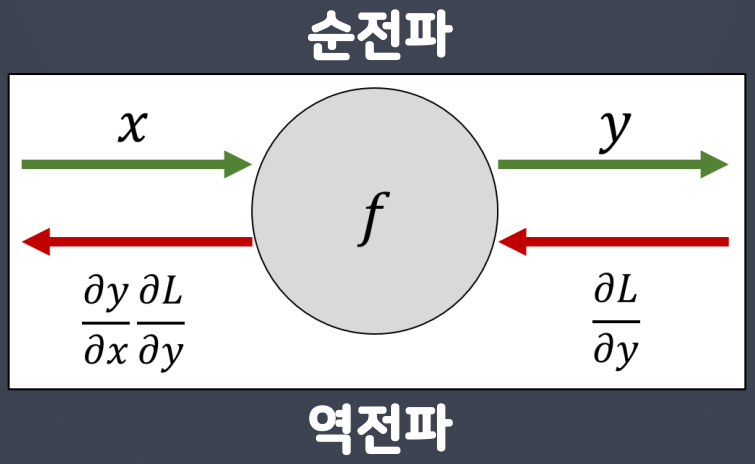
iris 데이터 신경망으로 풀기 (다중 분류)

오차 역전파(Back Propagation)

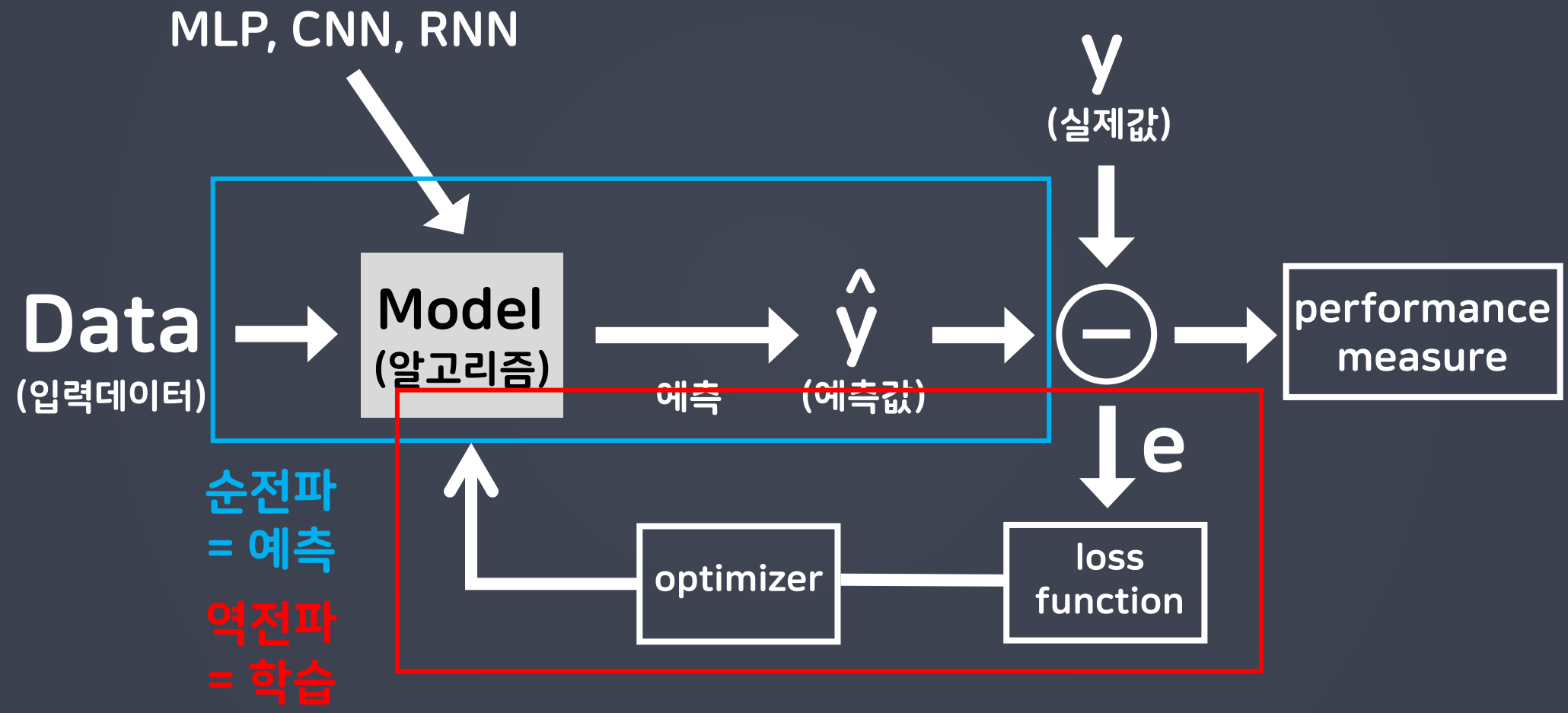
오차 역전파(Back Propagation)

- **순전파** : 입력 데이터를 입력층에서부터 출력층까지 정방향으로 이동시키며 출력 값을 **추론**해 나가는 과정

- **역전파** : 출력층에서 발생한 에러를 입력층 쪽으로 전파시키면서 최적의 결과를 **학습**해 나가는 과정

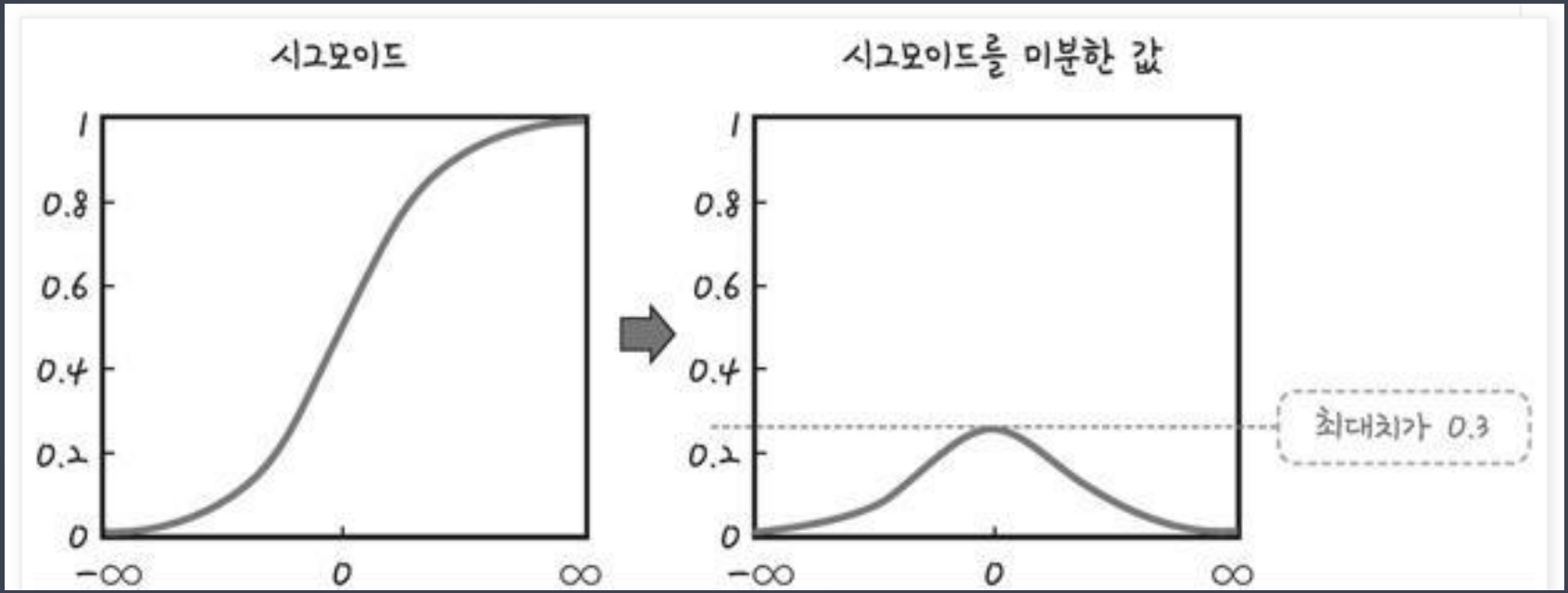


오차 역전파(Back Propagation)



오차 역전파(Back Propagation)

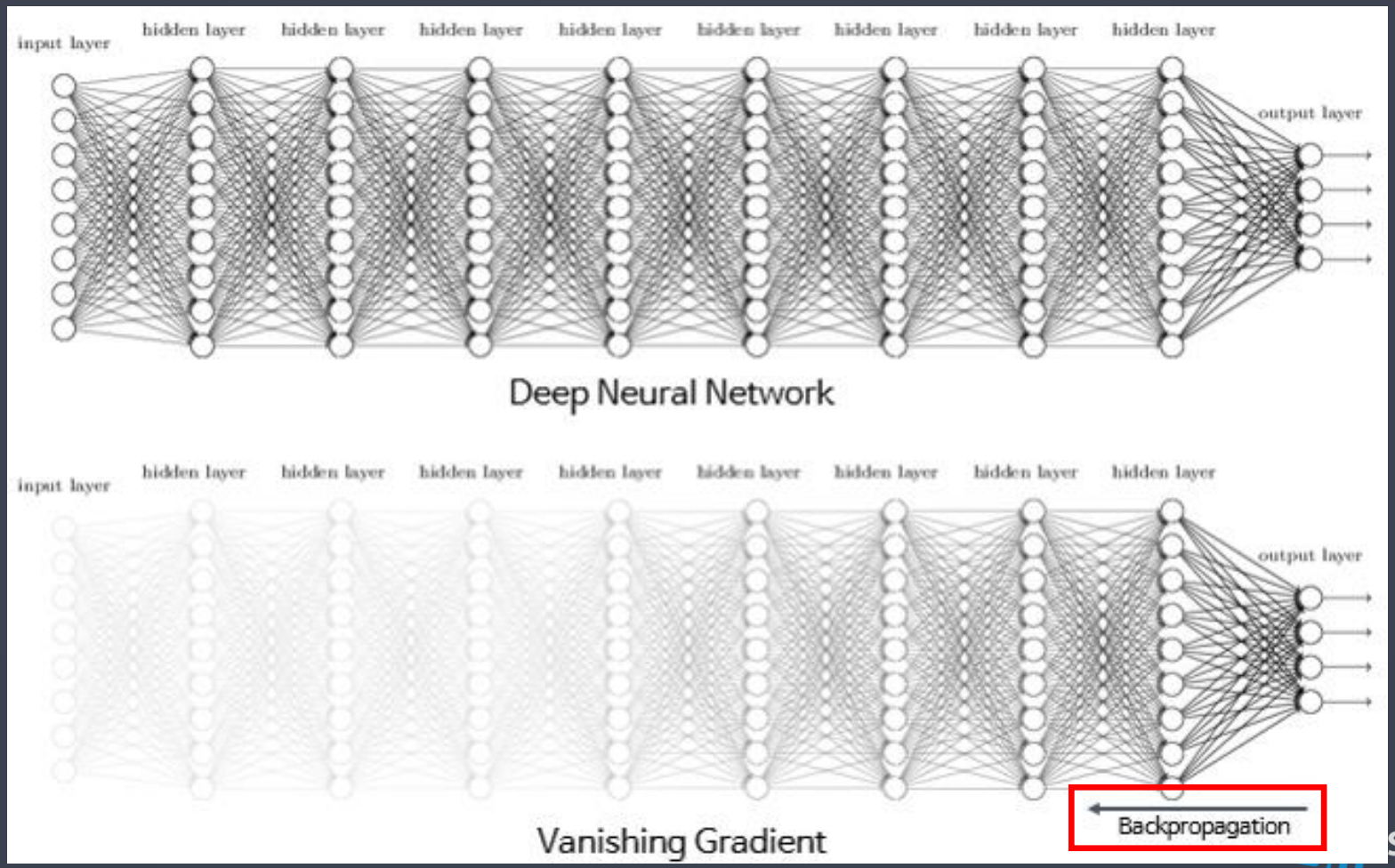
Sigmoid 함수의 문제점



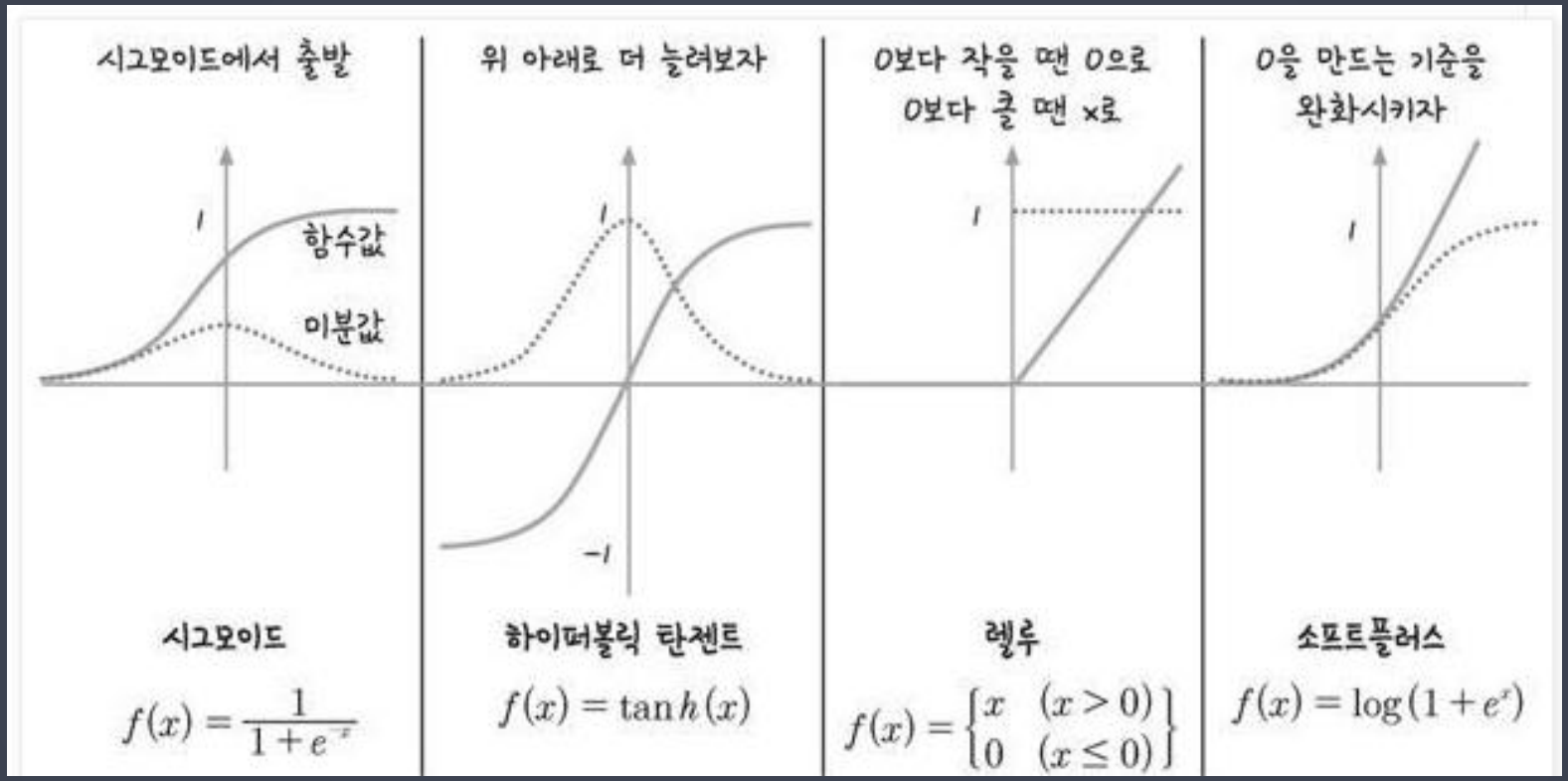
오차 역전파(Back Propagation)

Sigmoid 함수의 문제점

- 기울기 소실 문제(Vanishing Gradient)



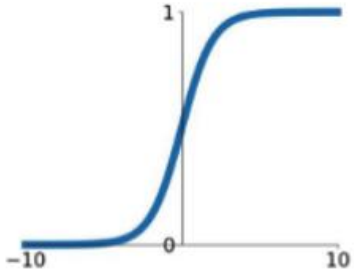
활성화 함수(Activation)의 종류



활성화 함수(Activation)의 종류

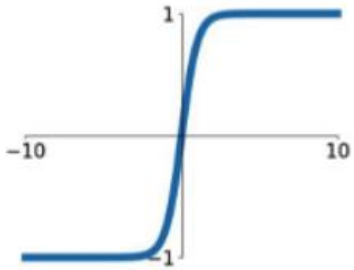
Sigmoid

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



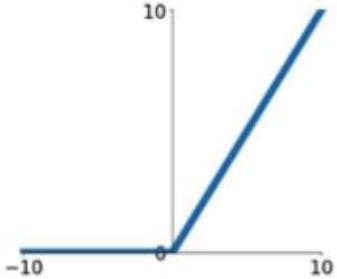
tanh

$$\tanh(x)$$



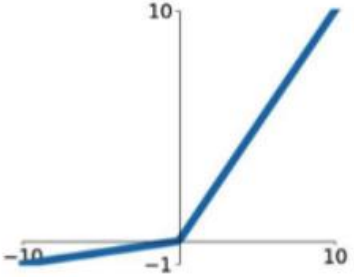
ReLU

$$\max(0, x)$$



Leaky ReLU

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

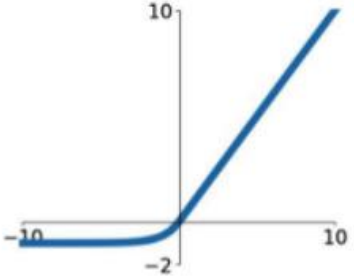


Maxout

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

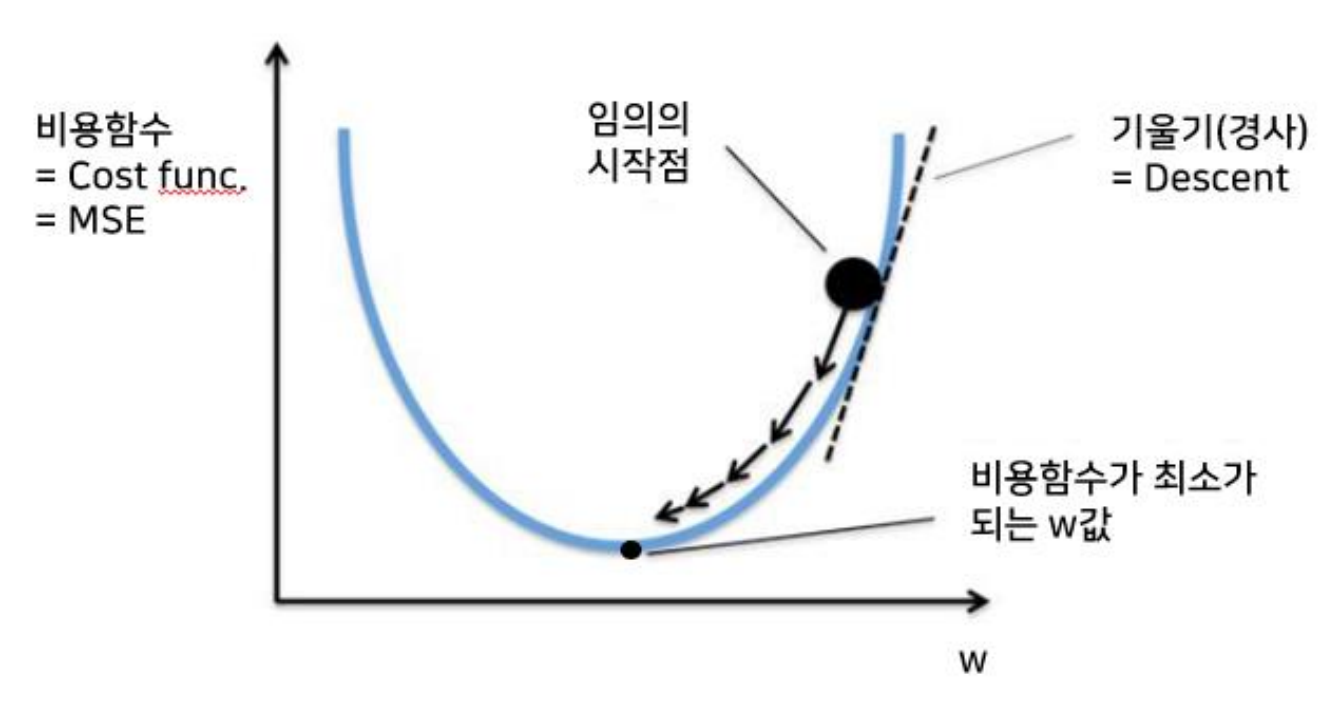
ELU

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

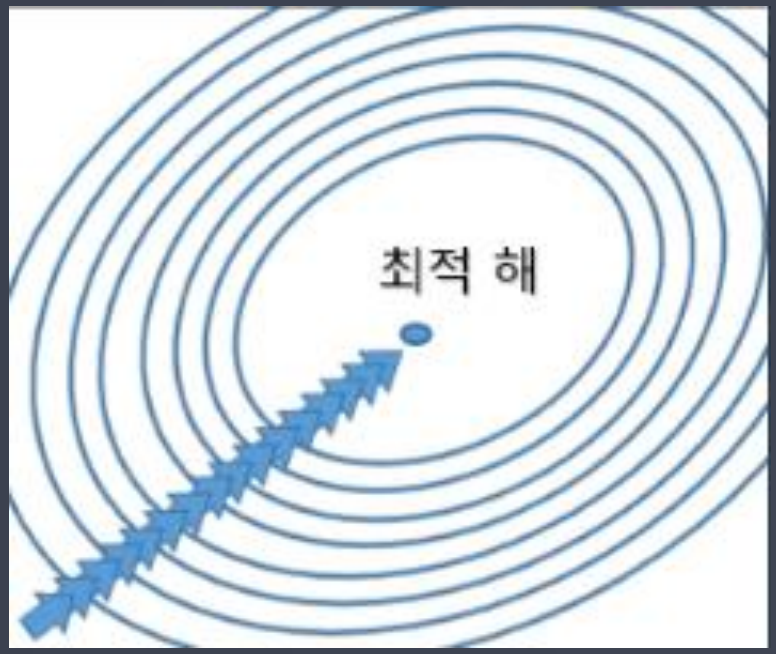


최적화함수(Optimizer)의 종류

경사하강법(Gradient Descent Algorithm)

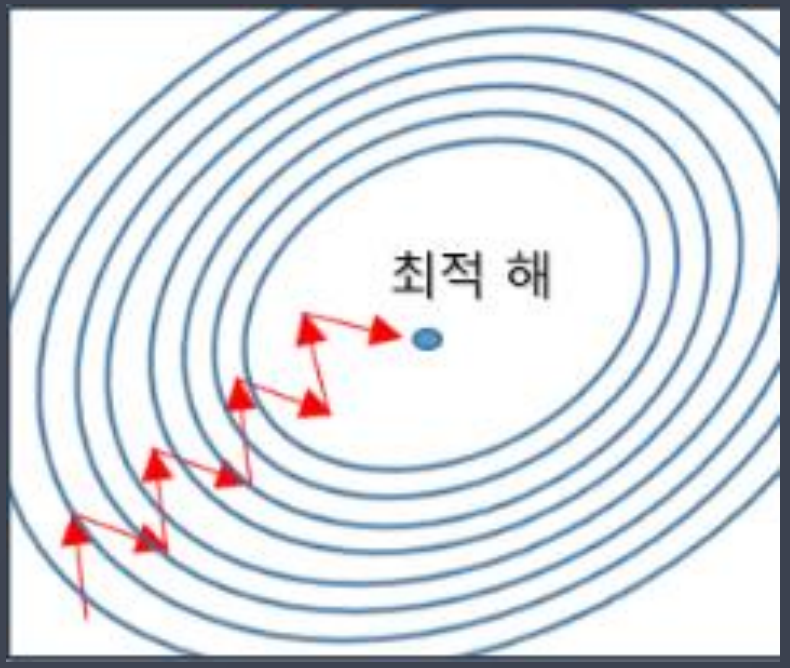


최적화함수(Optimizer)의 종류



경사하강법
(Gradient Descent)

전체 데이터를 이용해 업데이트

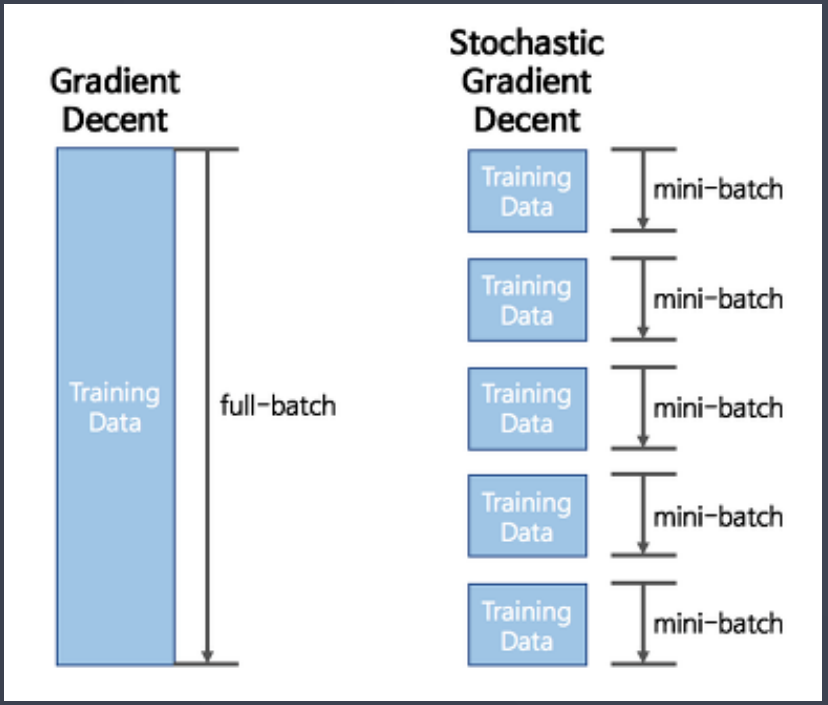


확률적경사하강법
(Stochastic Gradient Descent)

확률적으로 선택된 일부 데이터를 이용해 업데이트

최적화함수(Optimizer)의 종류

Batch_size

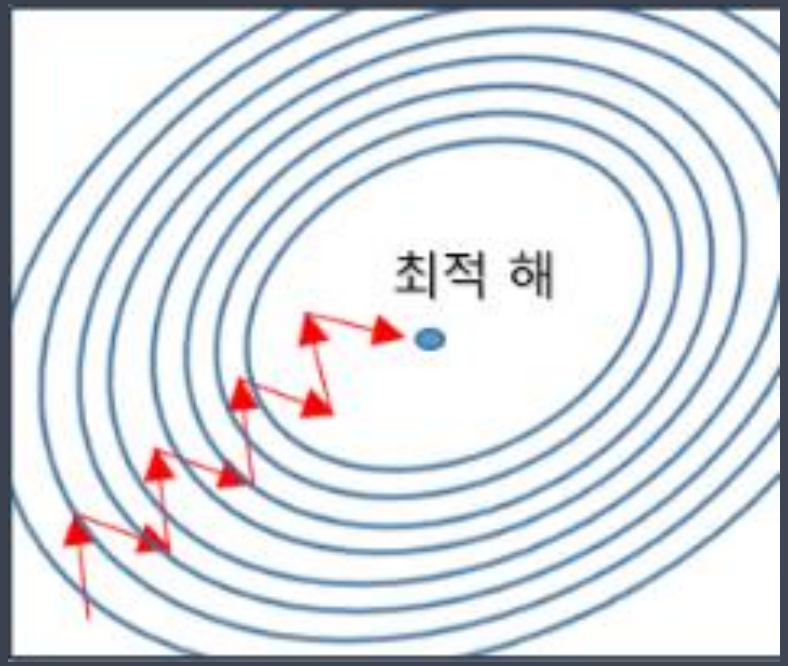


- batch_size를 **줄임**
  - **메모리** 소모가 적음(pc성능이 안 좋을 때)
- batch_size를 **높임**
  - 메모리 소모가 큼, **안정적**으로 학습

→ batch_size의 디폴트 값은 32이며 일반적으로 32, 64가 많이 사용됨

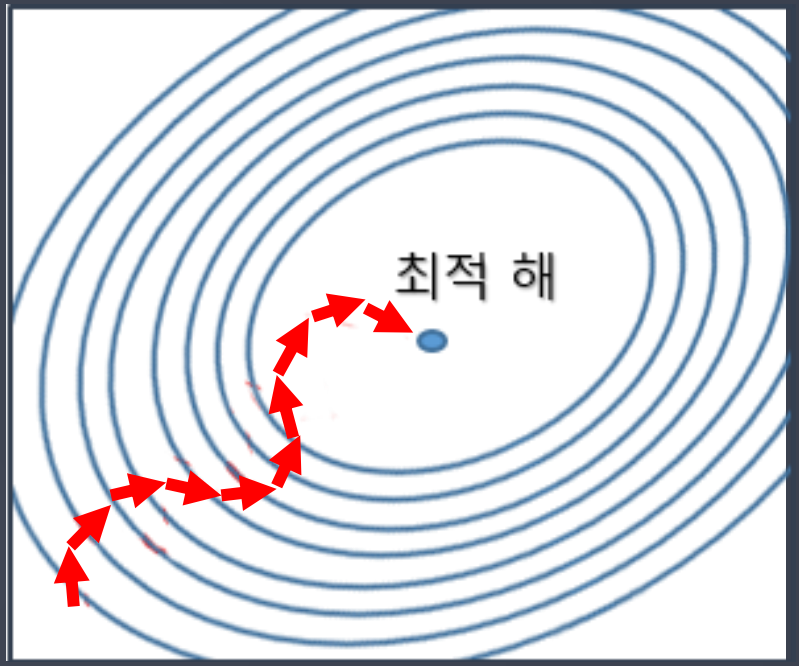
→ batch_size가 커질수록 과대적합의 위험이 높아짐

최적화함수(Optimizer)의 종류



확률적경사하강법
(Stochastic Gradient Descent)

확률적으로 선택된 일부 데이터를 이용해 업데이트



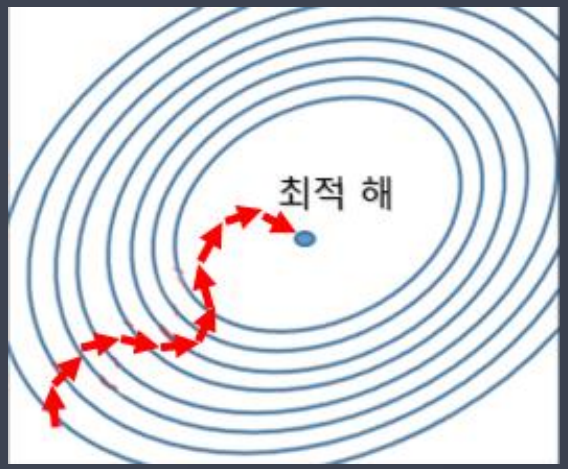
모멘텀
(Momentum)

경사 하강법에 관성을 적용해 업데이트
현재 batch뿐만 아니라 이전 batch 데이터의 학습 결과도 반영

최적화함수(Optimizer)의 종류

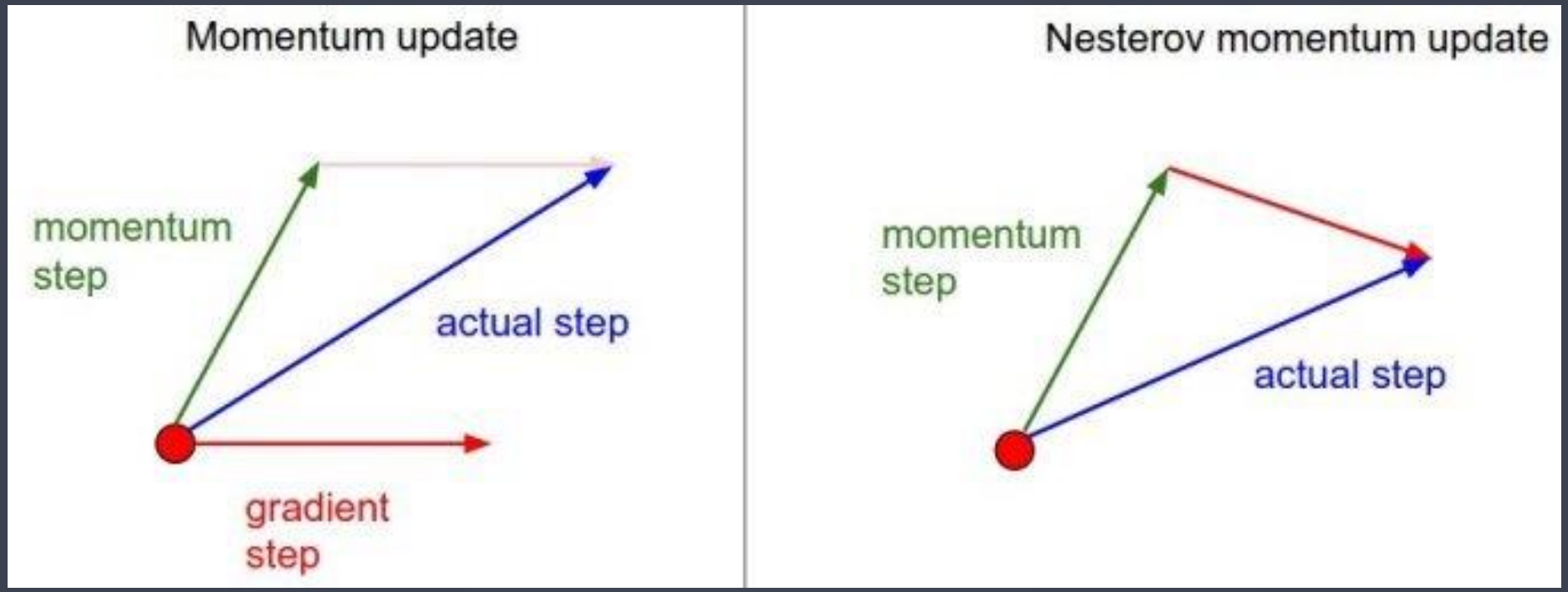
특징 (Momentum)

- 가중치를 수정하기 전 이전 방향을 참고하여 업데이트
- 지그재그 형태로 이동하는 현상이 줄어든다
- α는 Learning Rate, m은 momentum 계수 (보통 0.9)



$$V(t) = m * V(t-1) - \alpha \frac{\partial}{\partial w} Cost(w)$$
$$W(t+1) = W(t) + V(t)$$

최적화함수(Optimizer)의 종류



네스테로프 모멘텀

(Nesterov Accelrated Gradient)

개선된 모멘텀 방식

최적화함수(Optimizer)의 종류

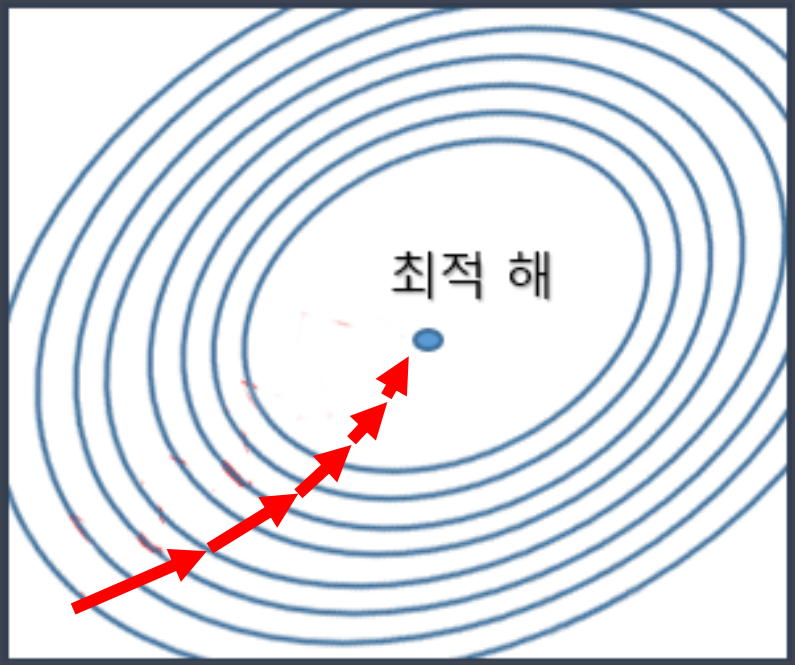
특징 (NAG)

- 업데이트 시 모멘텀 방식으로 먼저 더한 다음 계산
- 미리 해당 방향으로 이동한다고 가정하고 기울기를 계산해본 뒤 실제 계산에 반영
- 불필요한 이동을 줄일 수 있다

$$V(t) = m * V(t-1) - \alpha \frac{\partial}{\partial(w + m * V(t-1))} Cost(w)$$

$$W(t+1) = W(t) + V(t)$$

최적화함수(Optimizer)의 종류



에이다그래드

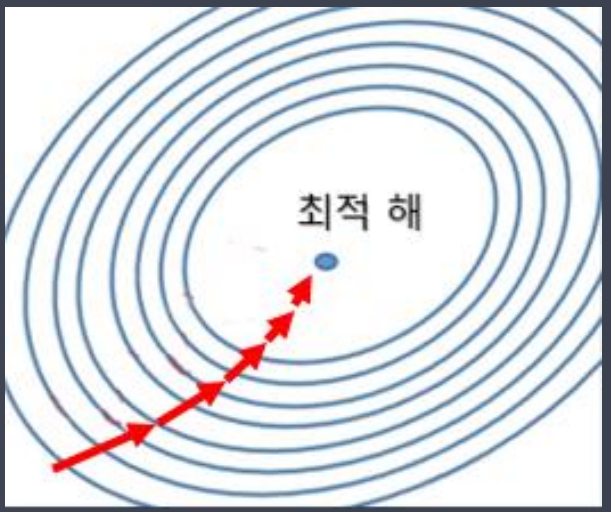
(Adaptive Gradient)

학습률 감소 방법을 적용해 업데이트

최적화함수(Optimizer)의 종류

특징 (Adagrad)

- 학습을 진행하면서 학습률을 점차 줄여가는 방법
- 처음에는 크게 학습하다가 조금씩 작게 학습한다
- 학습을 빠르고 정확하게 할 수 있다

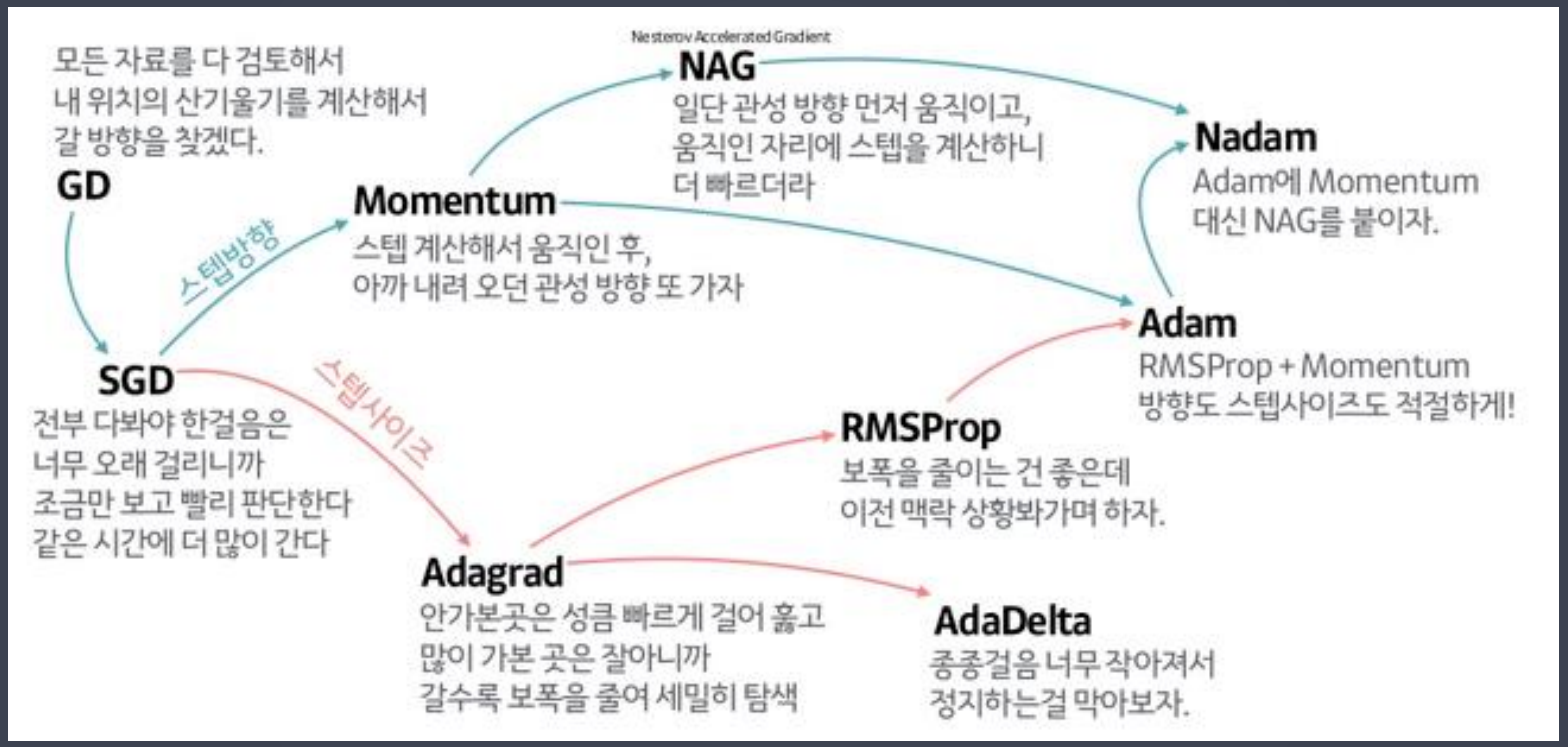


$$G(t) = G(t-1) + \left(\frac{\partial}{\partial w(t)} Cost(w(t))\right)^2$$

$$= \sum_{i=0}^{t} \left(\frac{\partial}{\partial w(i)} Cost(w(i))\right)^2$$

$$W(t+1) = W(t) - \alpha * \frac{1}{\sqrt{G(t)+\epsilon}} * \frac{\partial}{\partial w(i)} Cost(w(i))$$

최적화함수(Optimizer)의 종류



출처 : https://www.slideshare.net/yongho/ss-79607172

최적화함수(Optimizer) 코드

Keras

```
from tensorflow.keras import optimizers

opti = optimizers.SGD(learning_rate=0.01, momentum=0.9)

model.compile(loss='mse', optimizer=opti, metrics=['acc'])
```

Momentum

```
from tensorflow.keras import optimizers

opti = optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=True)

model.compile(loss='mse', optimizer=opti, metrics=['acc'])
```

NAG

```
model.compile(loss ="mse", optimizer="Adam", metrics=["acc"])
```

Adam

Adagrad, RMSprop, Adam 등은 이름으로 지정 가능

코드 실습

Keras로 MNIST 손글씨 이미지 데이터 분류 모델을 만들어보자

코드 실습

```
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   3   18  18  18  126 136 175 26  166 255 247 127 0   0   0   0
0   0   0   0   0   0   0   0   30  36  94  154 170 253 253 253 253 253 225 172 253 242 195 64  0   0   0   0
0   0   0   0   0   0   0   49  238 253 253 253 253 253 253 253 253 251 93  82  82  56  39  0   0   0   0   0
0   0   0   0   0   0   0   18  219 253 253 253 253 253 198 182 247 241 0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   80  156 107 253 253 205 11  0   43  154 0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   14  1   154 253 90  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   139 253 190 2   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   11  190 253 70  0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   35  241 225 160 108 1   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   81  240 253 253 119 25  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   45  186 253 253 150 27  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   16  93  252 253 187 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   249 253 249 64  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   46  130 183 253 253 207 2   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   39  148 229 253 253 253 250 182 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   24  114 221 253 253 253 253 201 78  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   23  66  213 253 253 253 253 198 81  2   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   18  171 219 253 253 253 253 195 80  9   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   55  172 226 253 253 253 253 244 133 11  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   136 253 253 253 212 135 132 16  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

코드 실습

Keras로 패션 이미지 데이터 분류 모델을 만들어보자