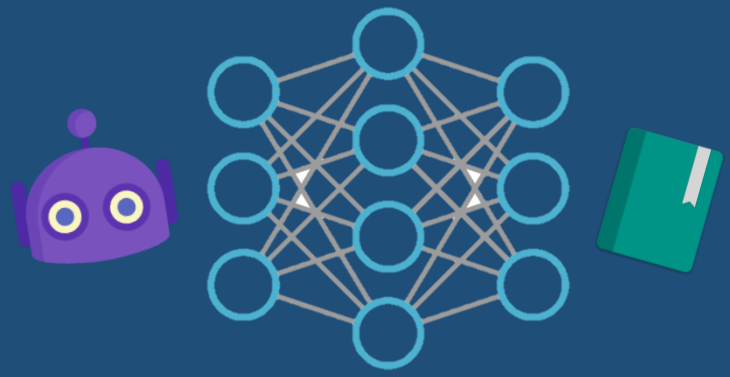
Deep Learning

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



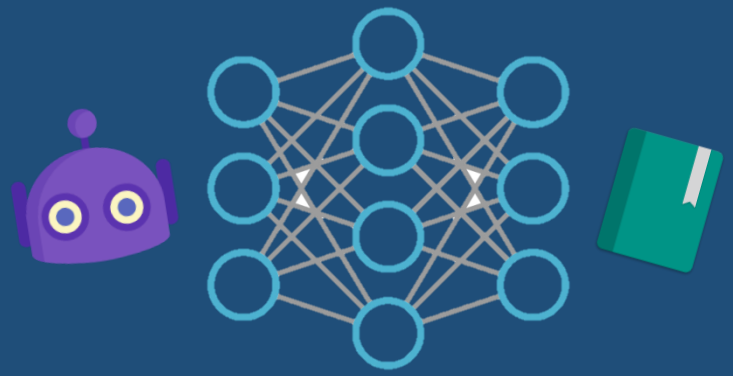
START

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.

활성화함수(Activation)



START

활성화 함수(Activation Function)

활성화 함수(Activation Function)

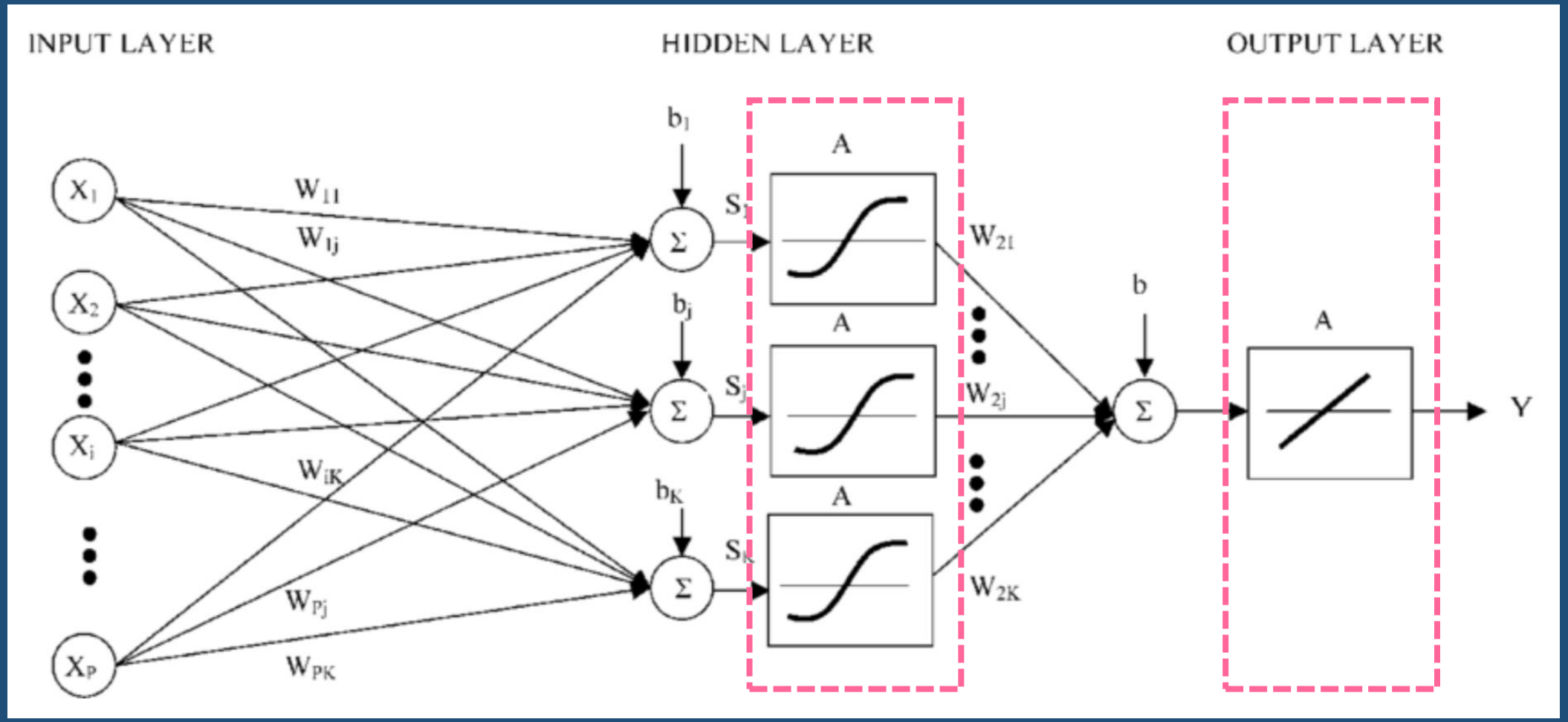
- 신경망은 선형회귀와 달리 한 계층의 신호를 다음 계층으로 그대로 전달하지 않고 비 선형적인 활성화 함수를 거친 후에 전달한다.

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

- 실제로 비선형의 활성화 함수를 도입한 신경망이 잘 동작하고 있다.

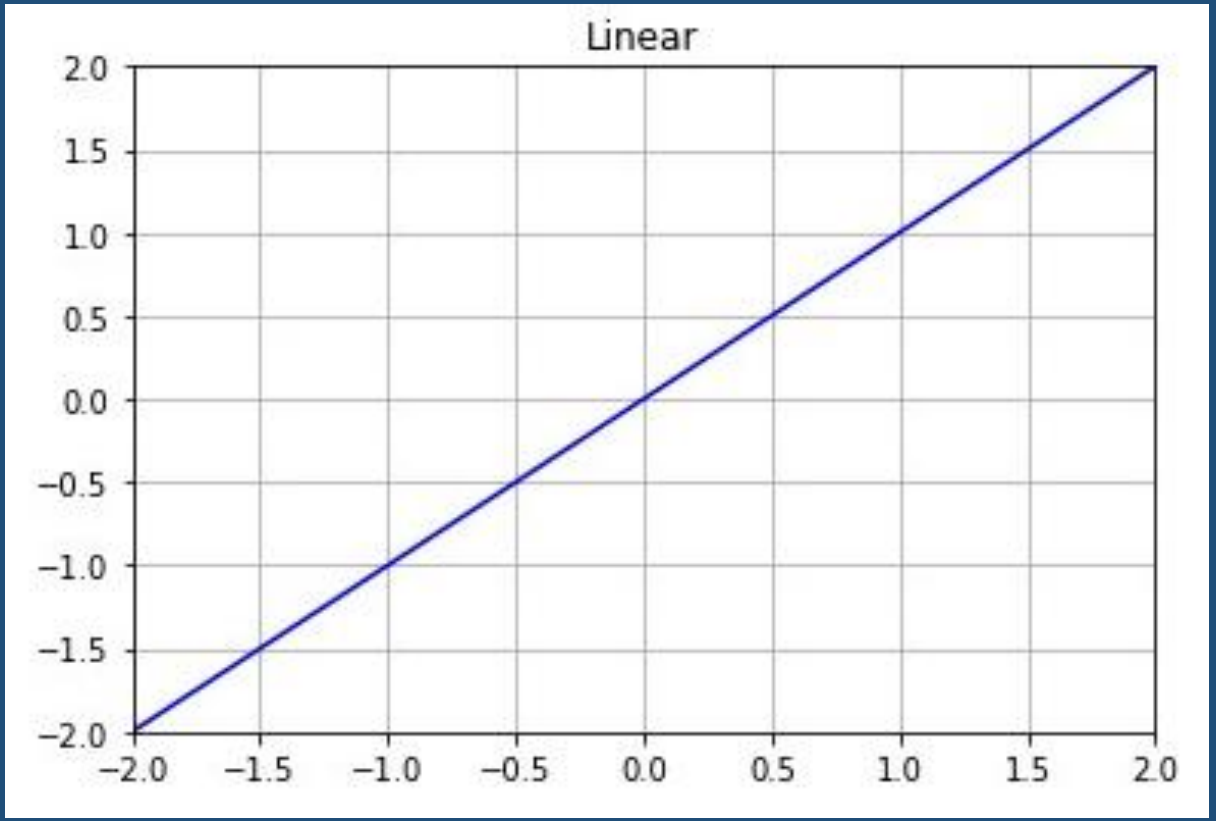
활성화 함수(Activation Function)

층에 따라 다른 활성화 함수 사용 가능



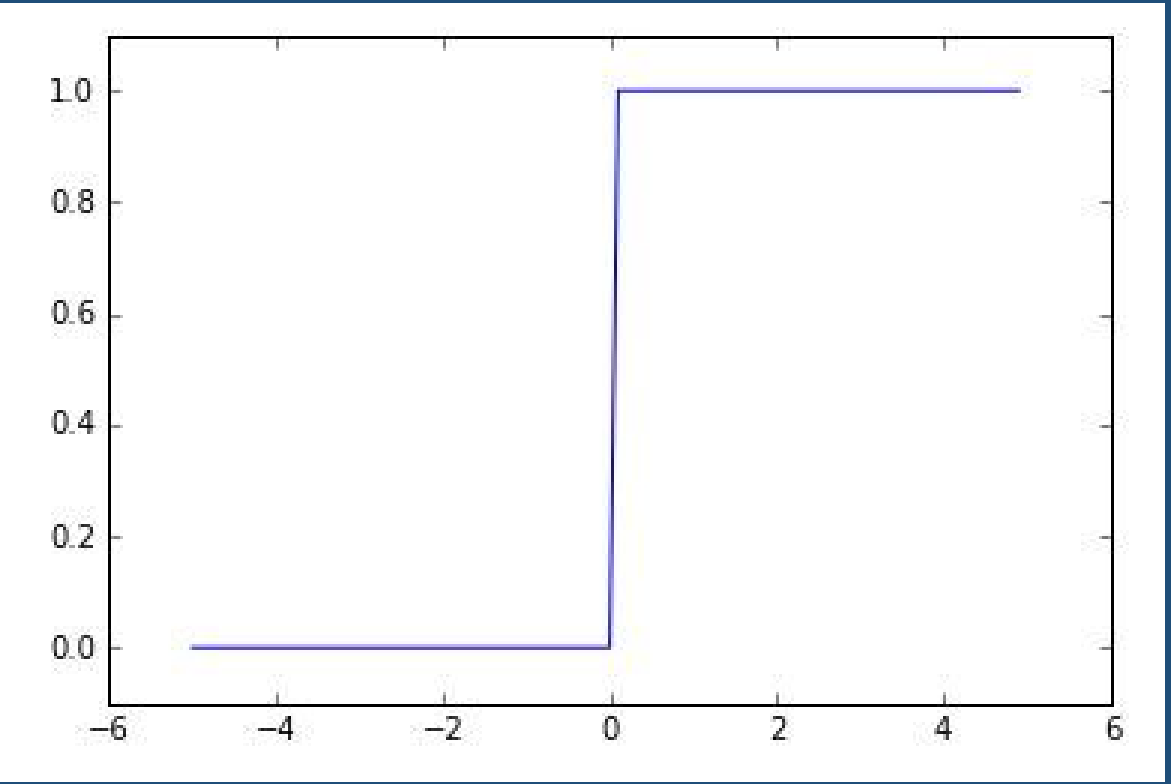
활성화 함수(Activation Function)

Linear function(항등 함수=선형 함수) → 회귀



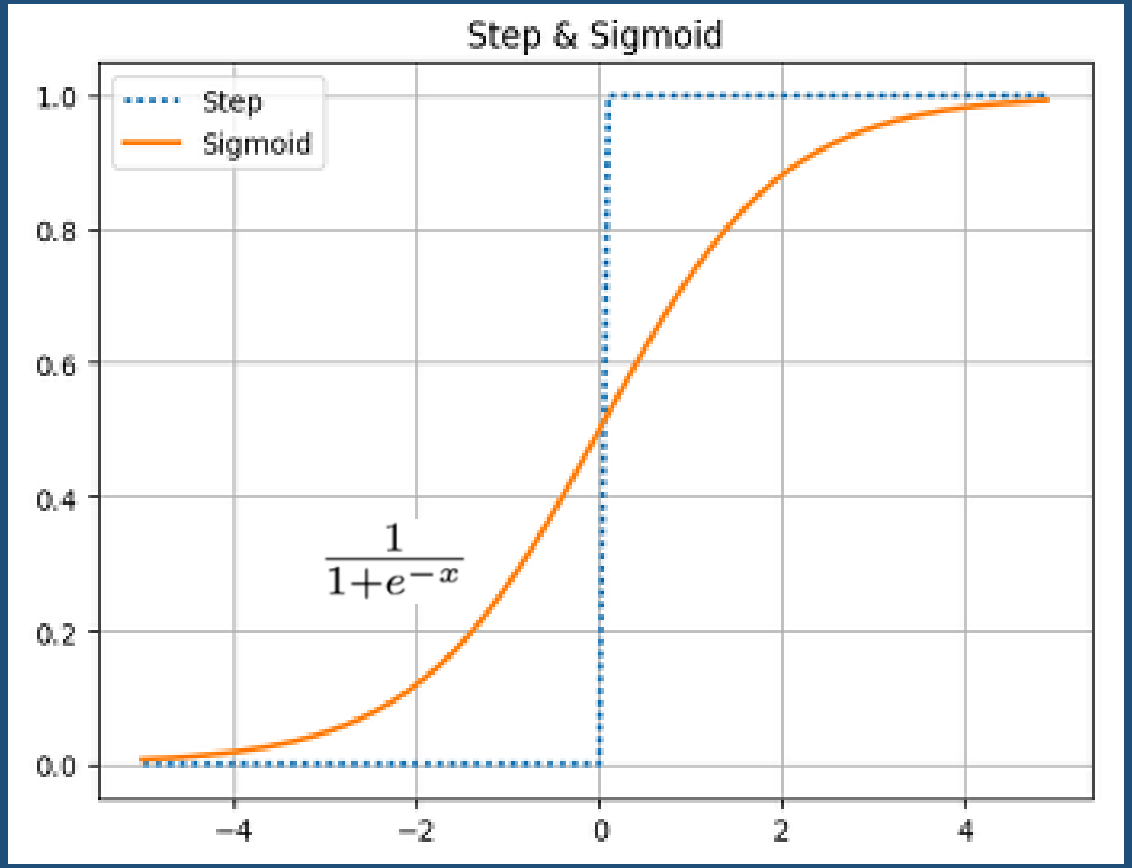
활성화 함수(Activation Function)

Step function(계단 함수) → 분류의 초기 활성화 함수



활성화 함수(Activation Function)

Sigmoid 함수 → 이진분류



활성화 함수(Activation Function) 정리

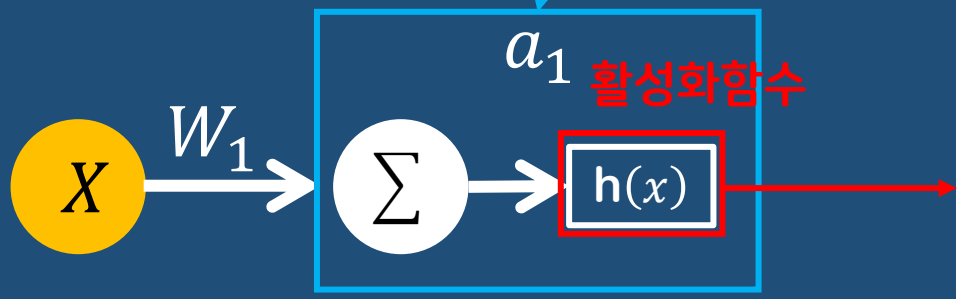
1. Step function과 Sigmoid의 차이

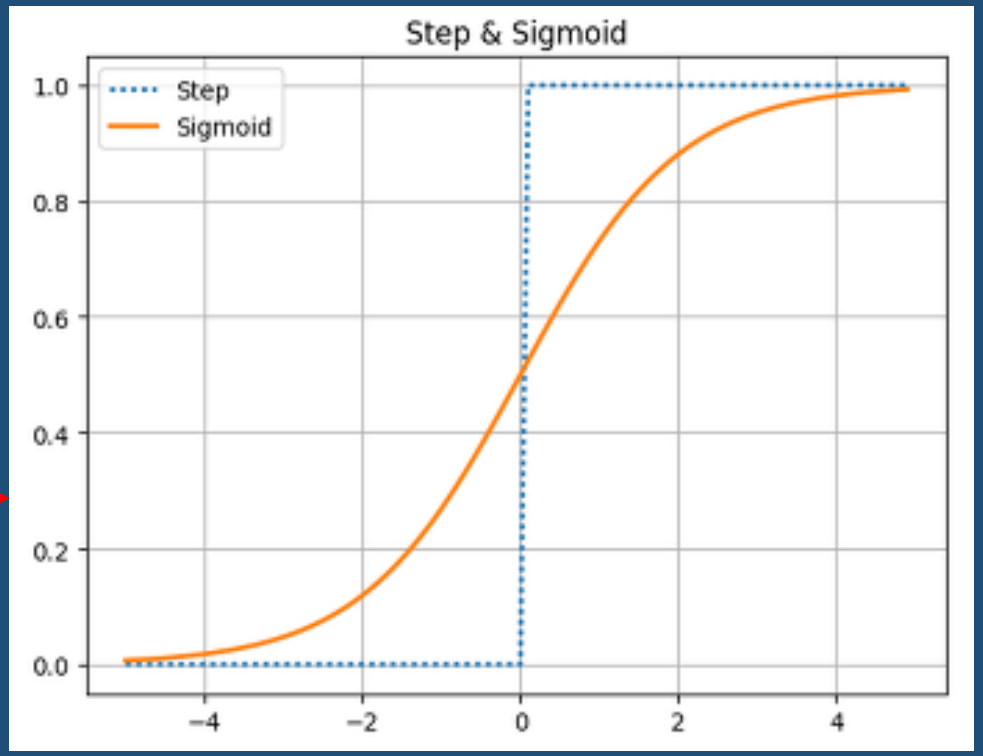
- 선형 모델이 학습하기 위해서 경사하강법(cost 함수를 미분)을 사용

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

예측값

퍼셉트론





활성화 함수(Activation Function) 정리

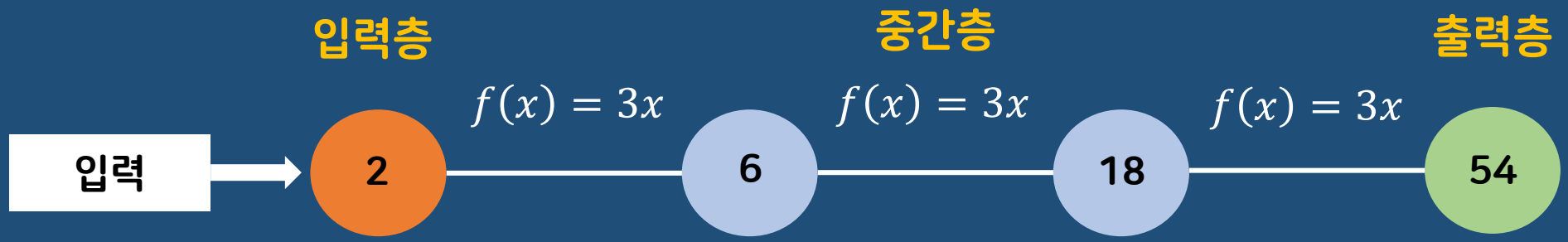
2. 중간층에 활성화 함수로 비선형 함수를 사용하는 이유

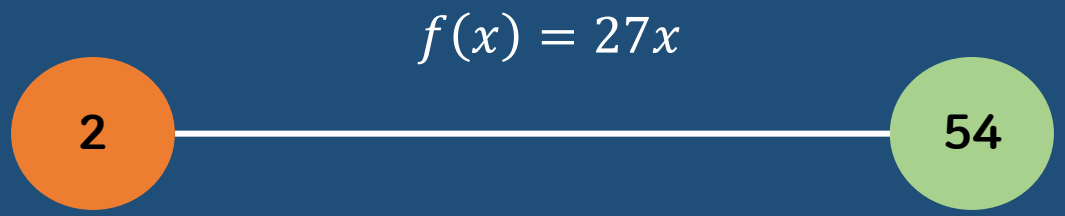
- 계단 함수(step)와 시그모이드 함수(sigmoid)는 비선형 함수
- 활성화 함수로 선형함수(linear) ex) h(x) = cx 를 사용하면 중간층(은닉층)을 여러 개 구성한 효과를 살릴 수 없음

활성화 함수(Activation Function) 정리

3. 중간층에 활성화 함수로 선형 함수를 사용하게 된다면

선형함수를 h(x) = 3x 라고 가정 해보자!





중간층을 여러 개 쌓을 필요가 없음!

활성화 함수(Activation Function) 정리

4. 다중분류 문제일 경우

- 다중분류 문제의 경우 정답 데이터를 원 핫 인코딩 해줘야 함
- 딥러닝 신경망에서는 각 레이블 값에 대한 확률 정보를 토대로 최종 분류를 진행함
- 각 레이블의 확률들을 알기 위해 출력층 퍼셉트론 개수를 레이블 개수와 맞춰야 함
- 원 핫 인코딩 된 정보와 출력층의 각 퍼셉트론이 예측한 확률과의 오차를 바탕으로 신경망이 학습하게 됨

활성화 함수(Activation Function) 정리

소프트맥스(softmax) 함수 → 다중분류

다중분류에서 레이블 값에 대한 각 퍼셉트론의 예측 확률의 합을 1로 설정

sigmoid에 비해 예측 오차의 평균을 줄여주는 효과

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

활성화 함수(Activation Function) 정리

소프트맥스(softmax) 함수 코드 구현

```
1 import numpy as np
2
3 def softmax(x):
4     e_x = np.exp(x-x.max())
5     return e_x/e_x.sum()
```

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

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

```
1 y = softmax(x)
2 y
```

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

```
1 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
이진 분류	sigmoid(로지스틱 함수)	binary_crossentropy
다중 분류	softmax(소프트맥스 함수)	categorical_crossentropy

코드 실습

폐암 데이터 신경망으로 풀기 (이진 분류)

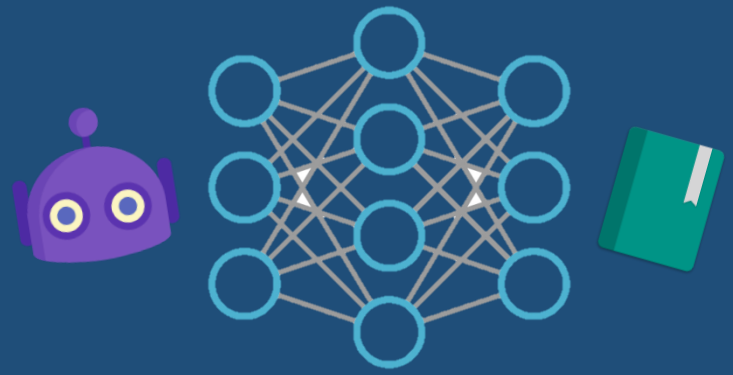
코드 실습

유방암 데이터 신경망으로 풀기 (이진 분류)

코드 실습

iris 데이터 신경망으로 풀기 (다중 분류)

오차 역전파 (Back Propagation)



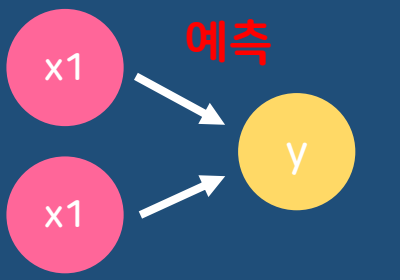
START

오차 역전파(Back Propagation)

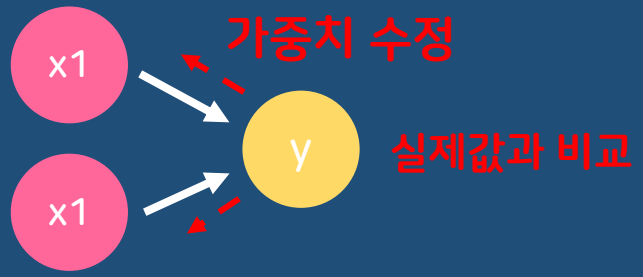
오차 역전파(Back Propagation)

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

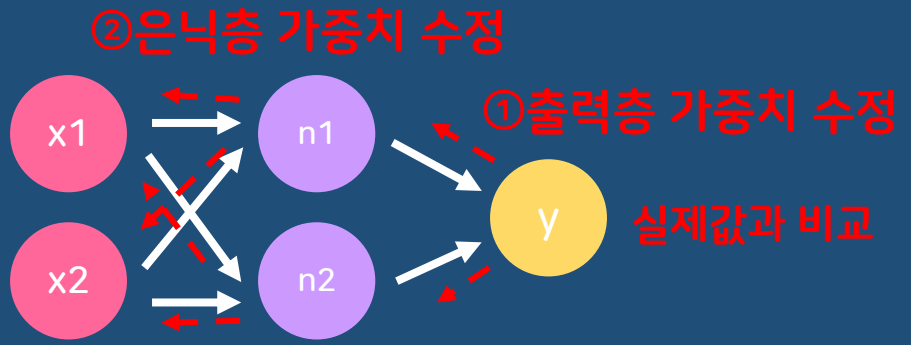
- **역전파** : 출력층에서 발생한 에러를 입력층 쪽으로 전파시키면서 최적의 결과를 **학습**해 나가는 과정



단일 퍼셉트론 예측

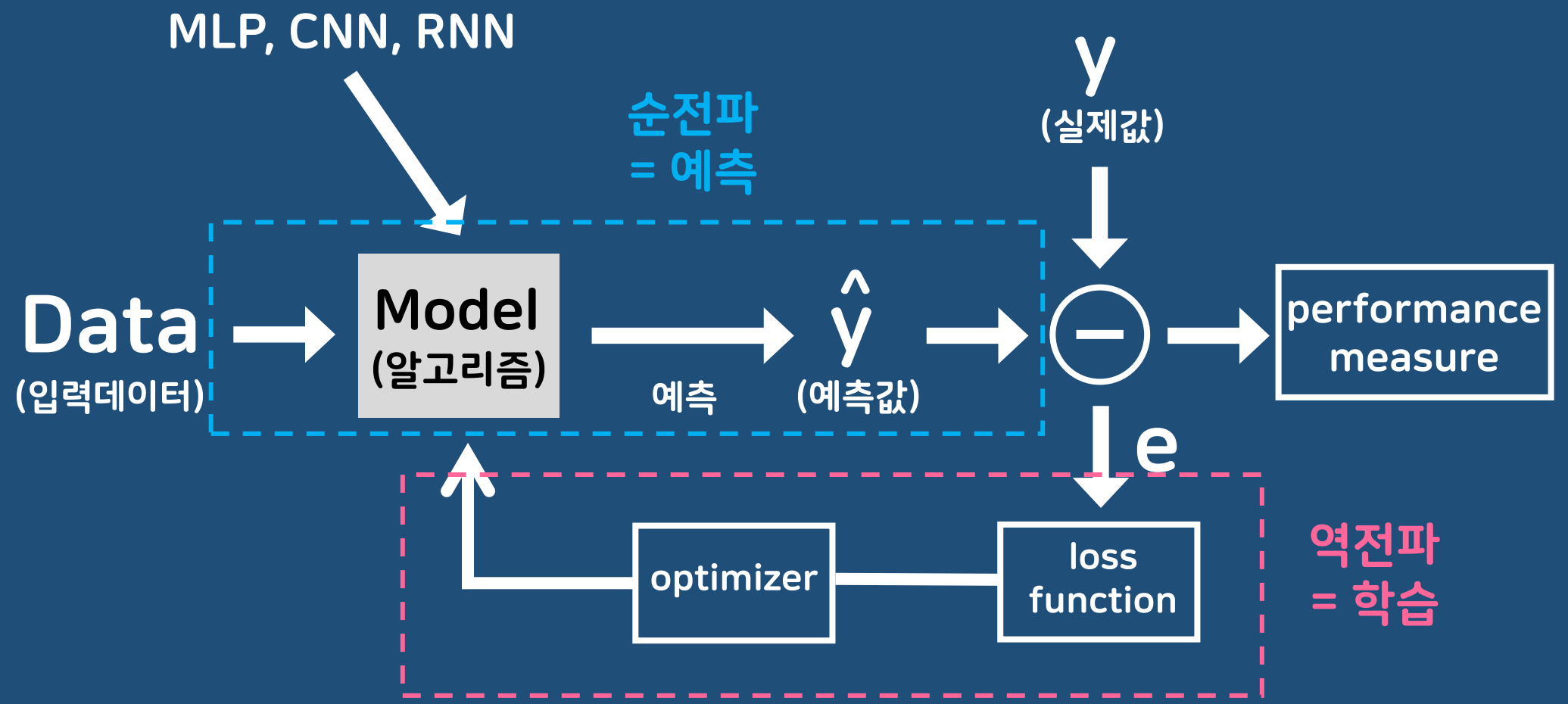


단일 퍼셉트론 오차 수정

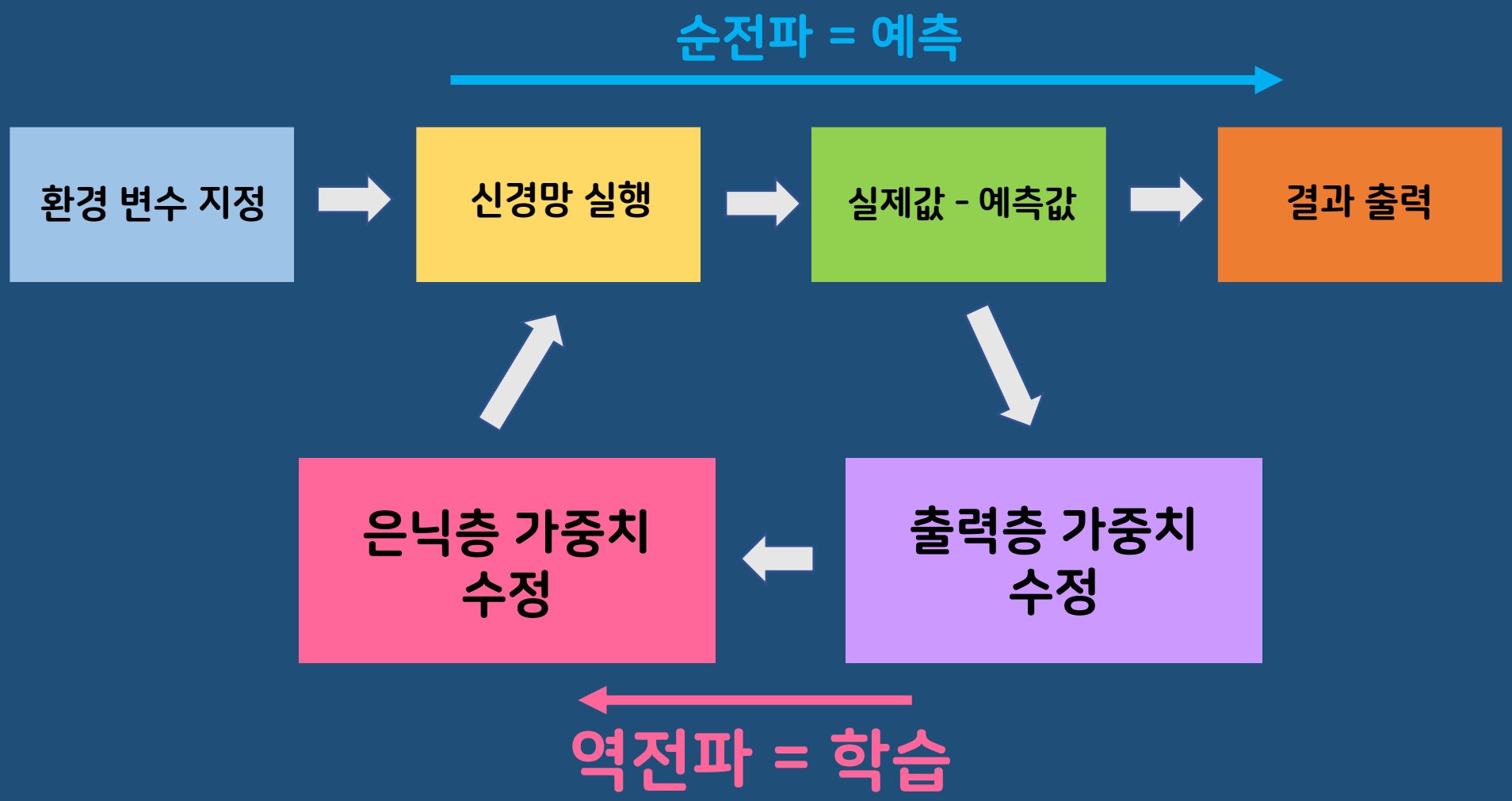


다층 퍼셉트론 오차 수정

오차 역전파(Back Propagation)

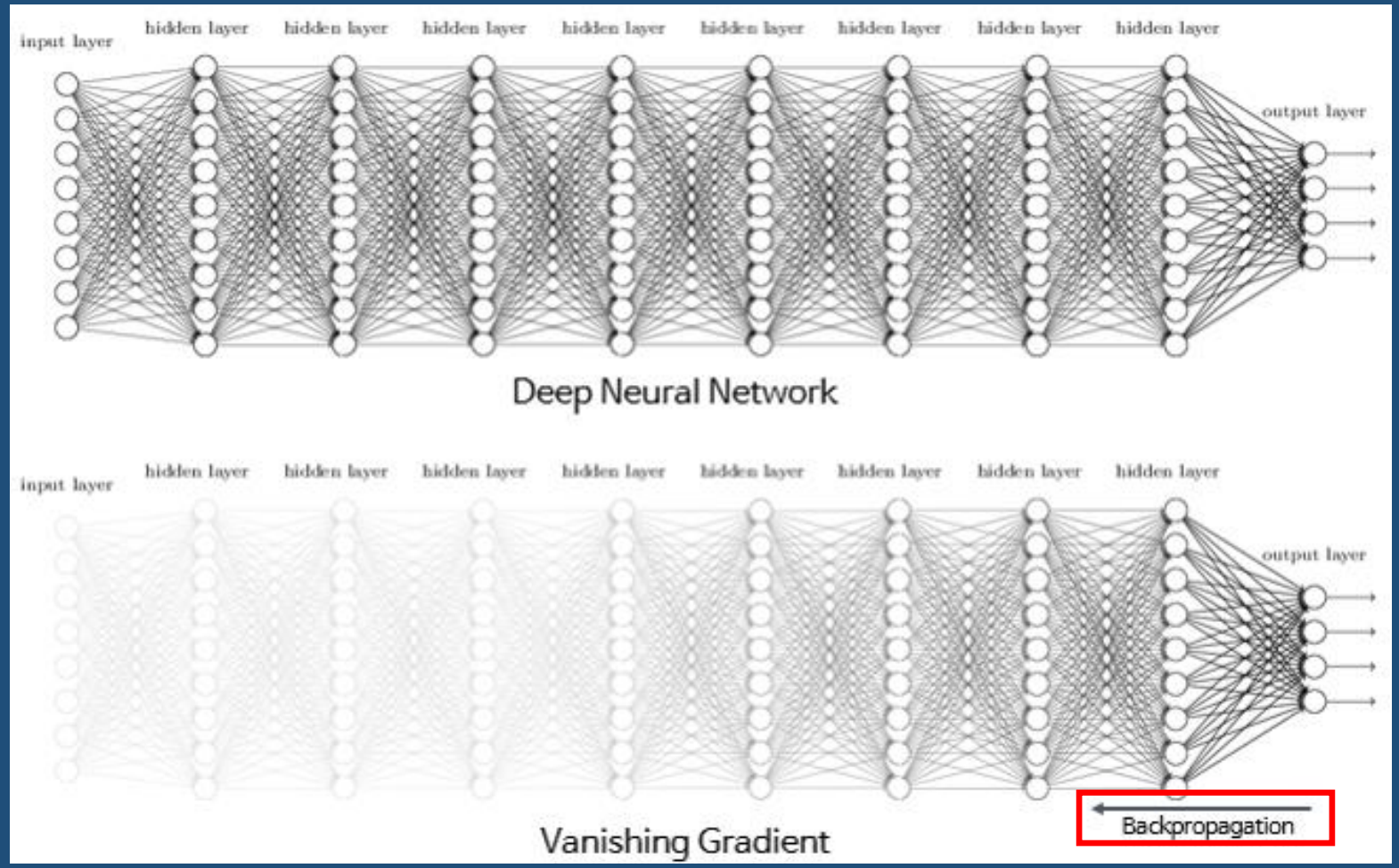


오차 역전파(Back Propagation)



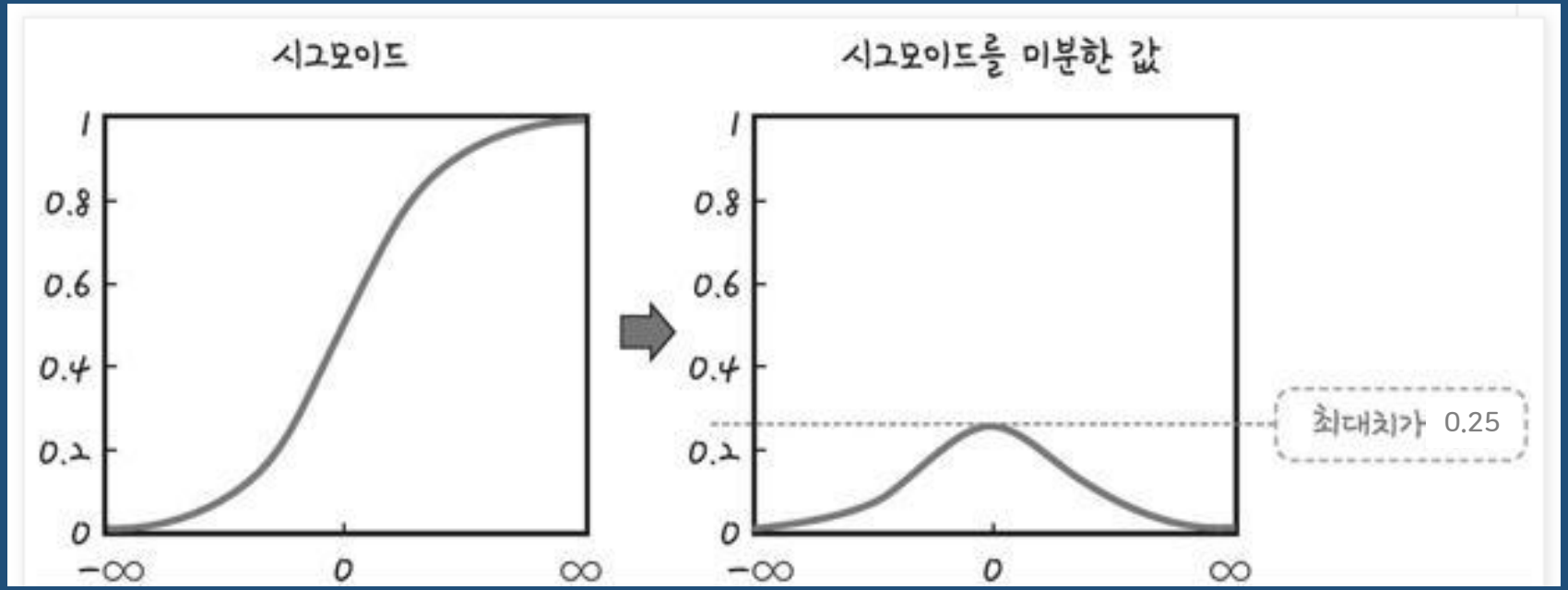
오차 역전파(Back Propagation) 기울기 소실 문제(Vanishing Gradient)

중간층 Sigmoid 함수의 문제점

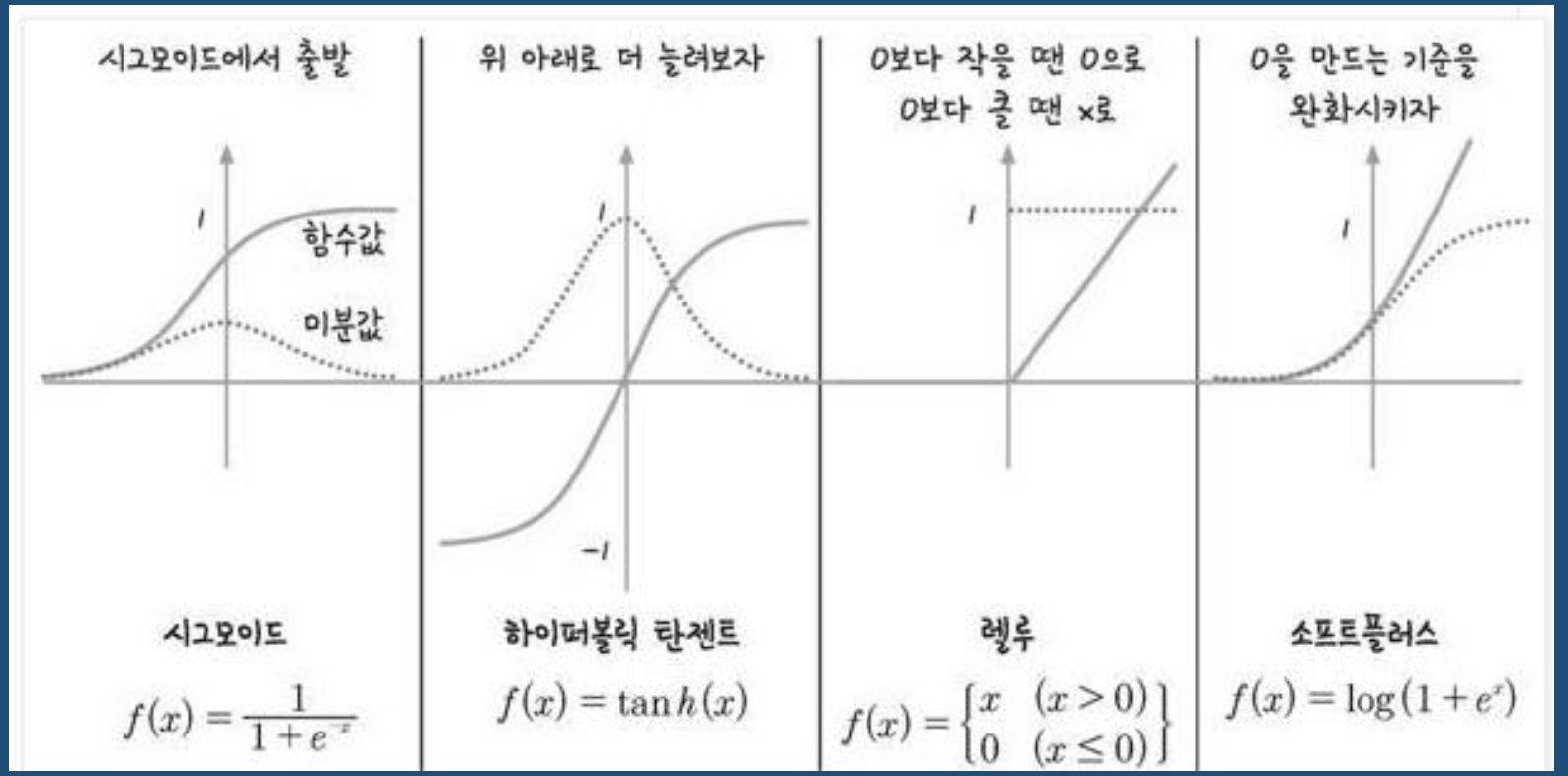


오차 역전파(Back Propagation) 기울기 소실 문제(Vanishing Gradient)

중간층 Sigmoid 함수의 문제점



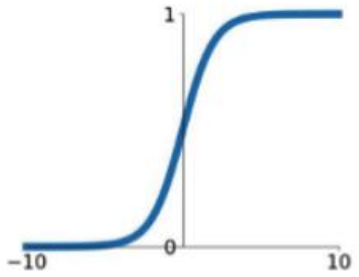
활성화 함수(Activation)의 종류



활성화 함수(Activation)의 종류

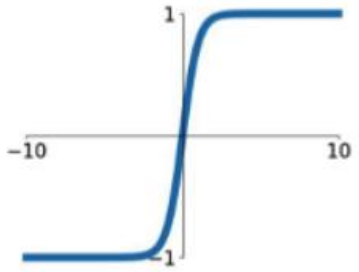
Sigmoid

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



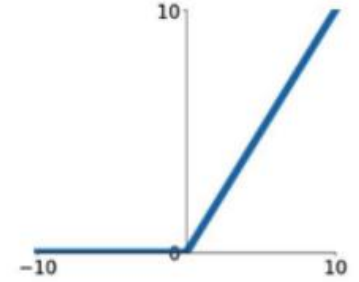
tanh

$$\tanh(x)$$



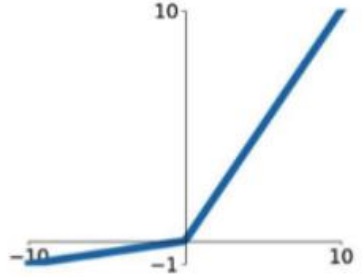
ReLU

$$\max(0, x)$$



Leaky ReLU

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

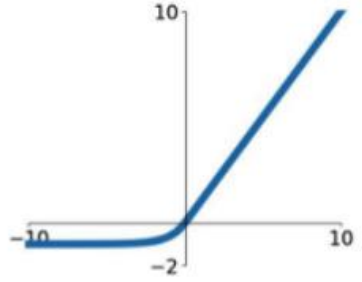


Maxout

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

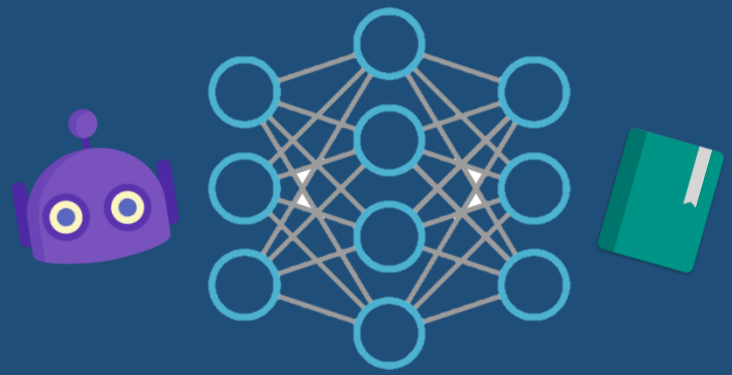
ELU

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



최적화함수(Optimizer)의 종류

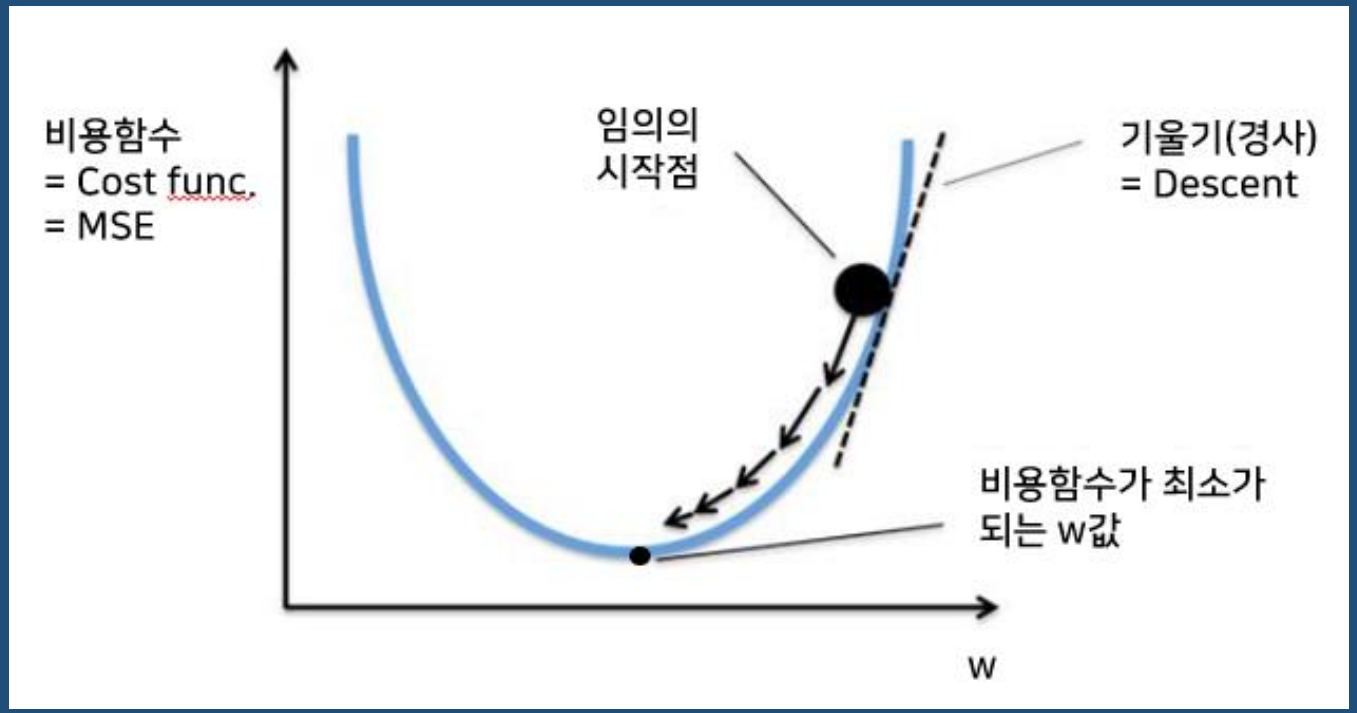
최적화 함수(Optimizer)



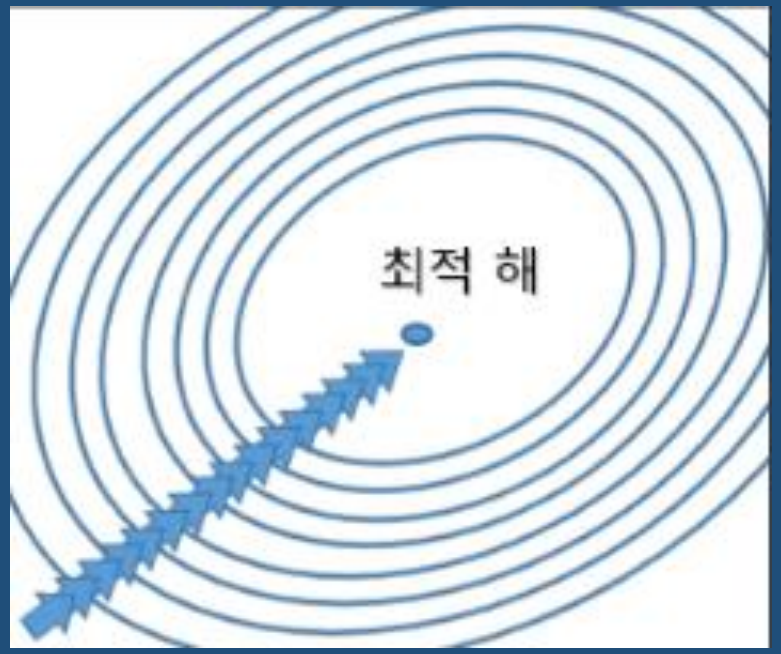
START

최적화함수(Optimizer)의 종류

경사하강법(Gradient Descent Algorithm)

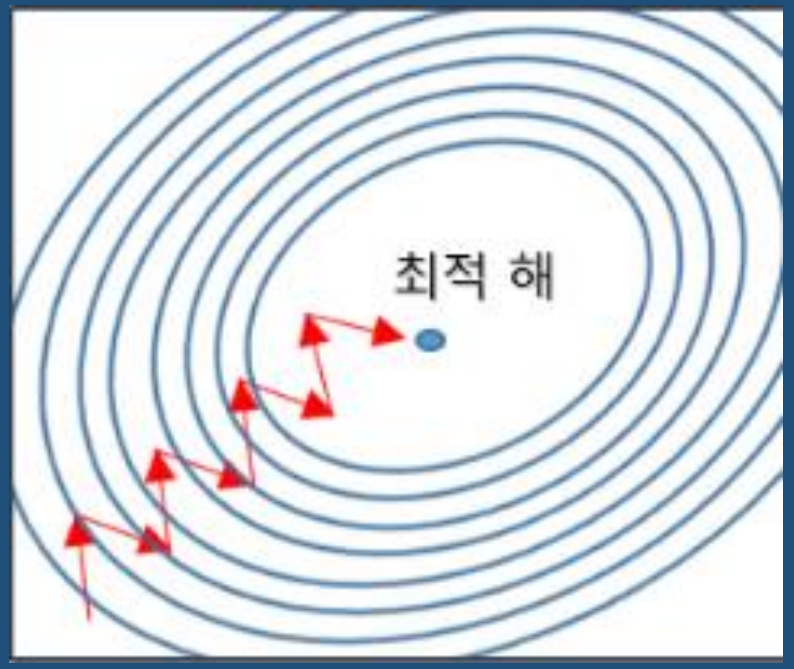


최적화함수(Optimizer)의 종류



경사하강법
(Gradient Descent)

전체 데이터를 이용해 업데이트



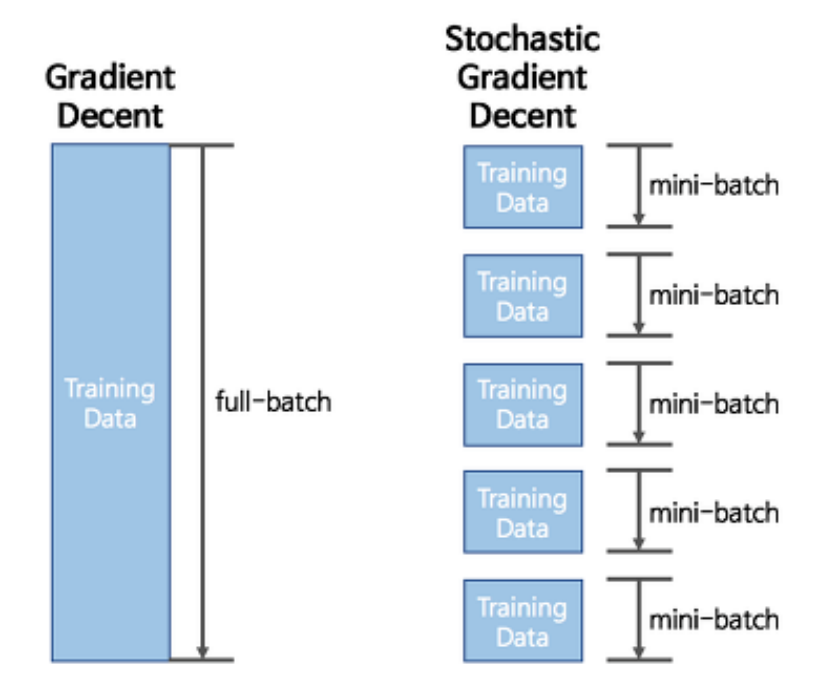
확률적경사하강법
(Stochastic Gradient Descent)

확률적으로 선택된 일부 데이터를 이용해 업데이트

최적화함수(Optimizer)의 종류

Batch_size

일반적으로 PC 메모리의 한계 및 속도 저하 때문에 대부분의 경우에는 한번의 epoch에 모든 데이터를 한꺼번에 집어넣기가 힘듦

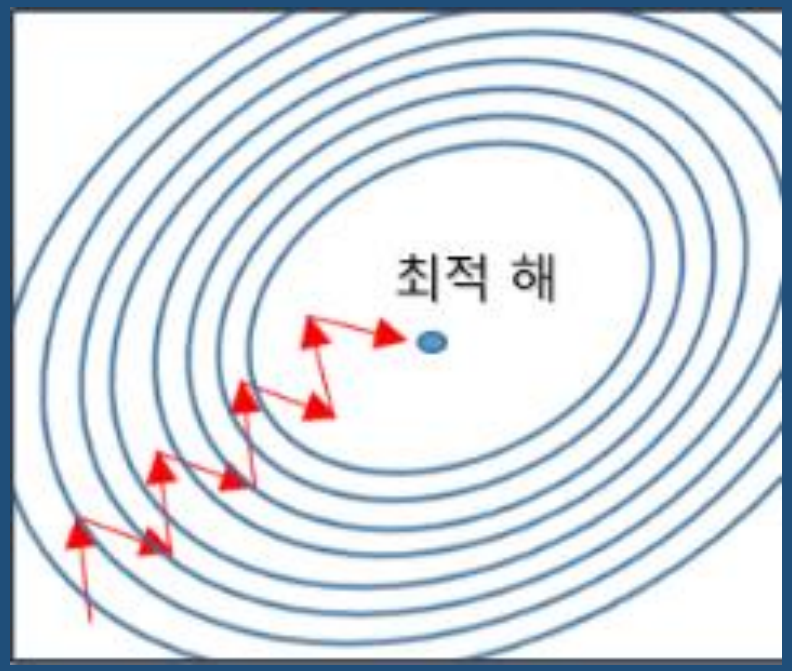


- batch_size를 **줄임**
  - 메모리 소모가 **적음**(pc성능이 안 좋을 때)

- batch_size를 **높임**
  - 메모리 소모가 **큼**, 학습 속도가 **빠름**

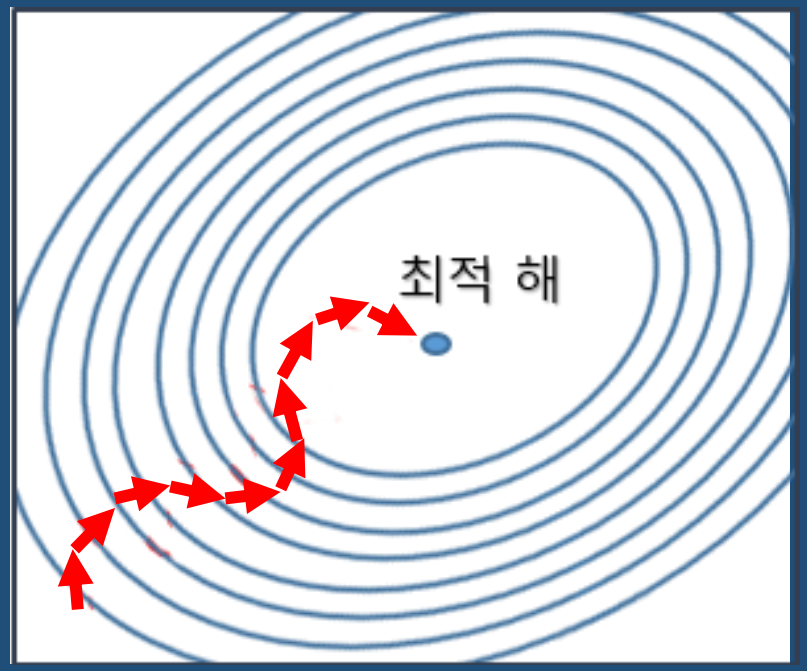
→ batch_size의 디폴트 값은 32이며 일반적으로 32, 64가 많이 사용됨

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

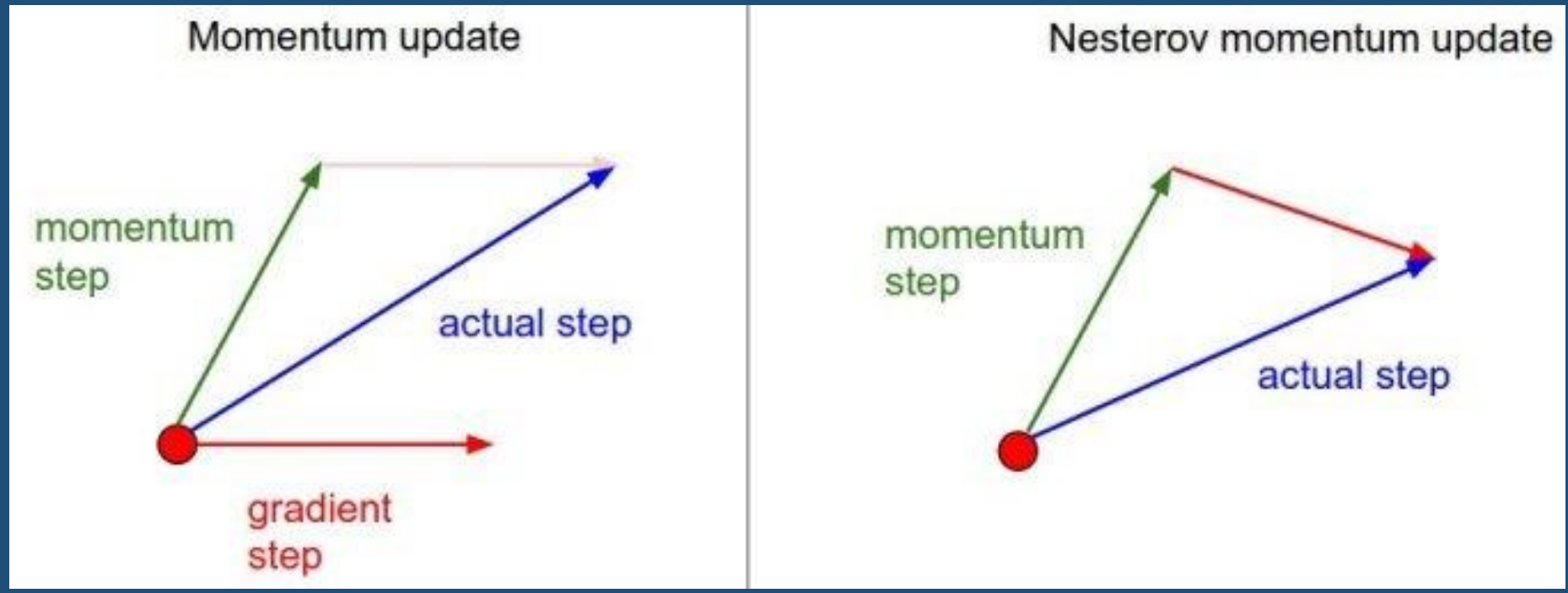
특징 (Momentum)

- 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

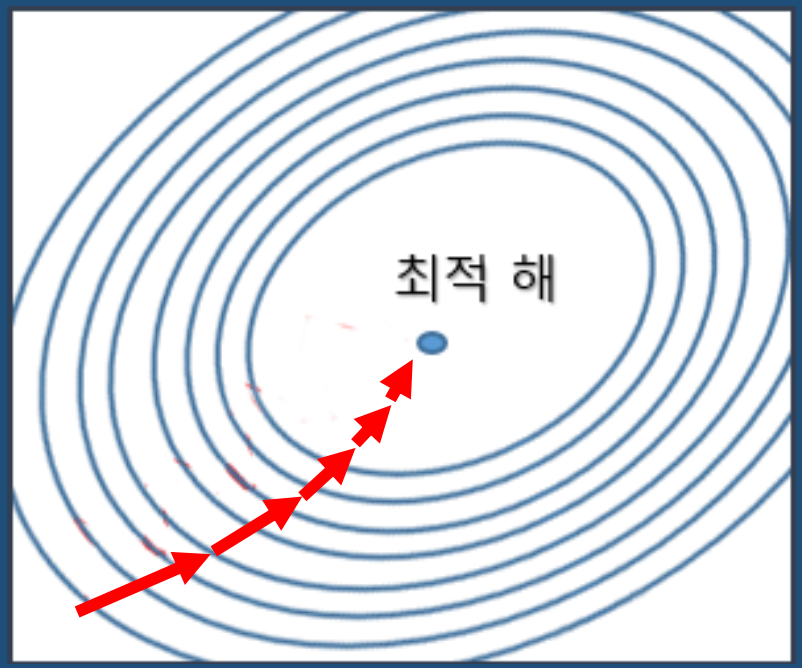
특징 (NAG)

- 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

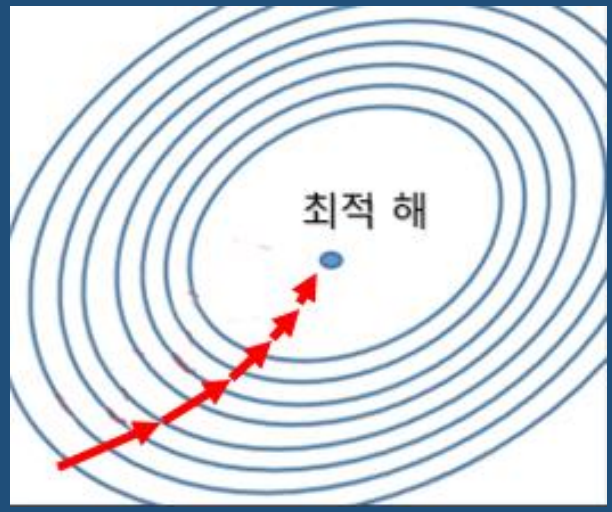
(Adaptive Gradient)

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

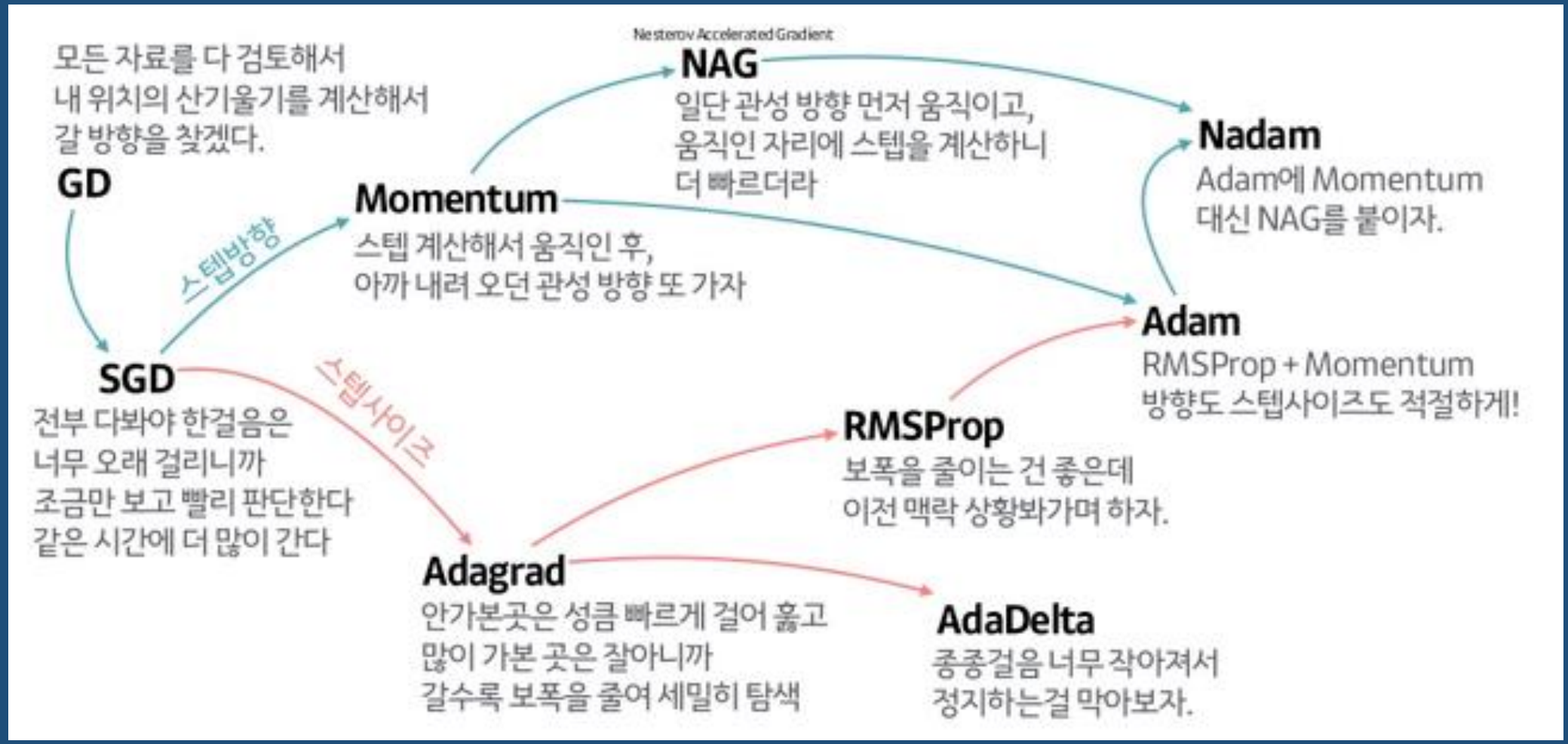


$$G(t) = G(t-1) + \left(\frac{\partial}{\partial w(t)} Cost(w(t))\right)^2$$

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

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

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 동물 이미지 데이터를 분류 모델을 만들어보자

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   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   38  48  48  22  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   62  97  198 243 254 254 212 27  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   67  172 254 254 225 218 218 237 248 40  0   21  164 187 0   0
0   0   0   0   0   0   0   0   0   0   0   89  219 254 97  67  14  0   0   92  231 122 23  203 236 59  0   0
0   0   0   0   0   0   0   0   0   0   25  217 242 92  4   0   0   0   0   4   147 253 240 232 92  0   0   0
0   0   0   0   0   0   0   0   0   0   101 255 92  0   0   0   0   0   0   105 254 254 177 11  0   0   0   0
0   0   0   0   0   0   0   0   0   0   167 244 41  0   0   0   7   76  199 238 239 94  10  0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   192 121 0   0   2   63  180 254 233 126 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   190 196 14  2   97  254 252 146 52  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   130 225 71  180 232 181 60  0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   130 254 254 230 46  0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   6   77  244 254 162 4   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   110 254 218 254 116 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   131 254 154 28  213 86  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   66  209 153 19  19  233 60  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   142 254 165 0   14  216 167 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   90  254 175 0   18  229 92  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   26  229 249 176 222 244 44  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   73  193 197 134 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
```

코드 실습

Keras로 패션 이미지 데이터 분류 모델을 만들어보자