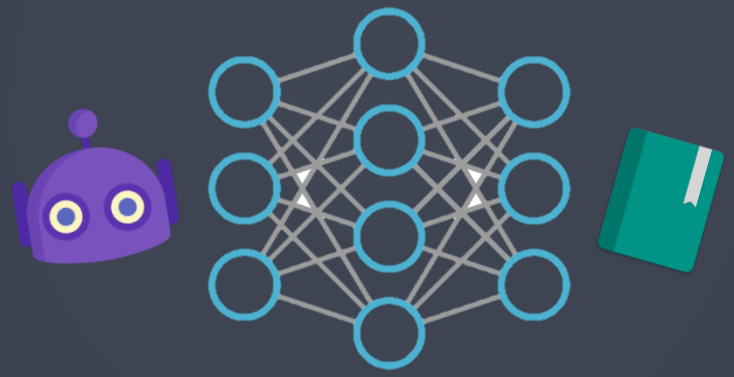
Deep Learning

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



START

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.

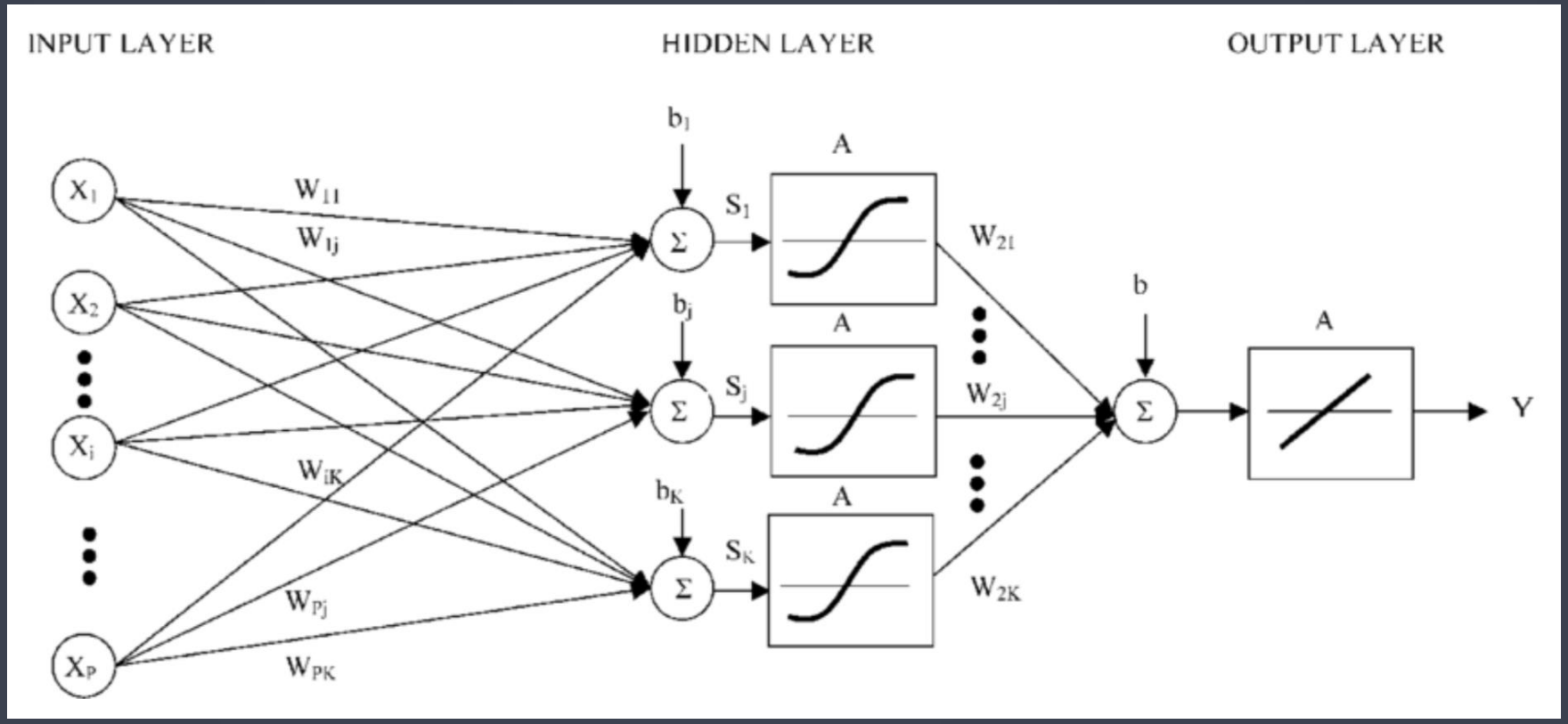
활성화 함수(Activation Function)

활성화 함수(Activation Function)

- 신경망은 선형회귀와 달리 한 계층의 신호를 다음 계층으로 그대로 전달하지 않고 활성화 함수를 거친 후에 전달함
- 사람의 신경망 속 뉴런들도 모든 자극을 다 다음 뉴런으로 전달하는 것은 아니고 **역치 이상의 자극만 전달**하게 됨
- 활성화 함수는 이런 부분까지 사람과 유사하게 구현하여 **사람처럼 사고하고 행동하는 인공지능 기술을 실현**하기 위해 도입됨
- 또한 선형모델을 기반으로 하는 딥러닝 신경망에서 **분류 문제를 해결**하기 위해서 비선형 활성화 함수가 필요함

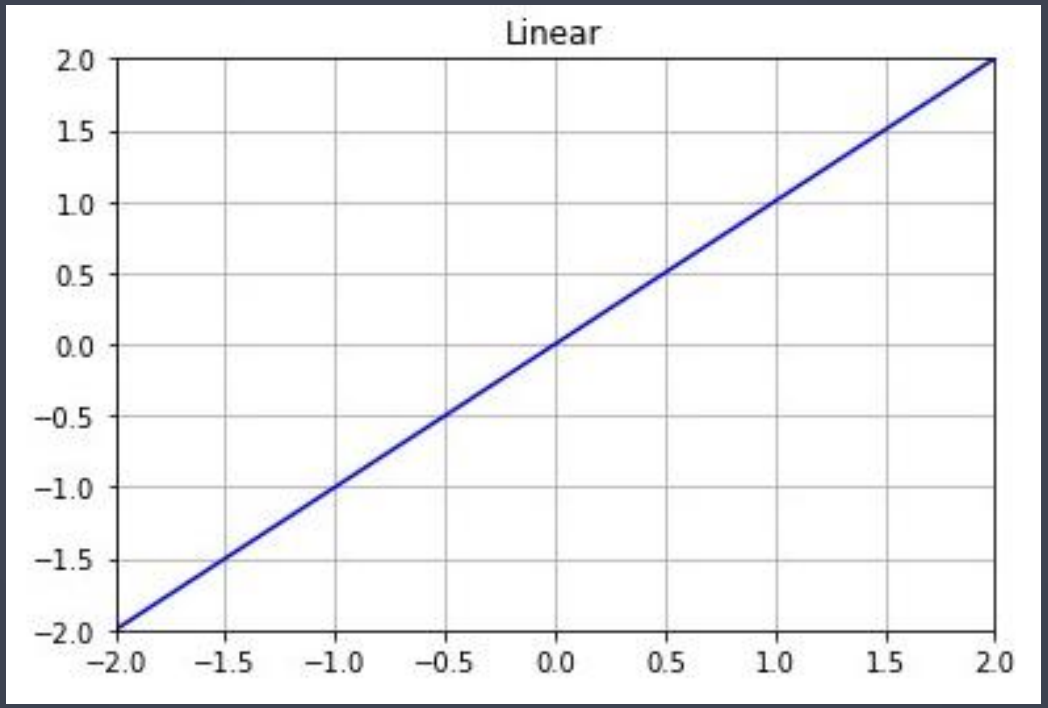
활성화 함수(Activation Function)

층에 따라 예측 결과값에 따라
다른 활성화 함수를 사용 할 수 있다.



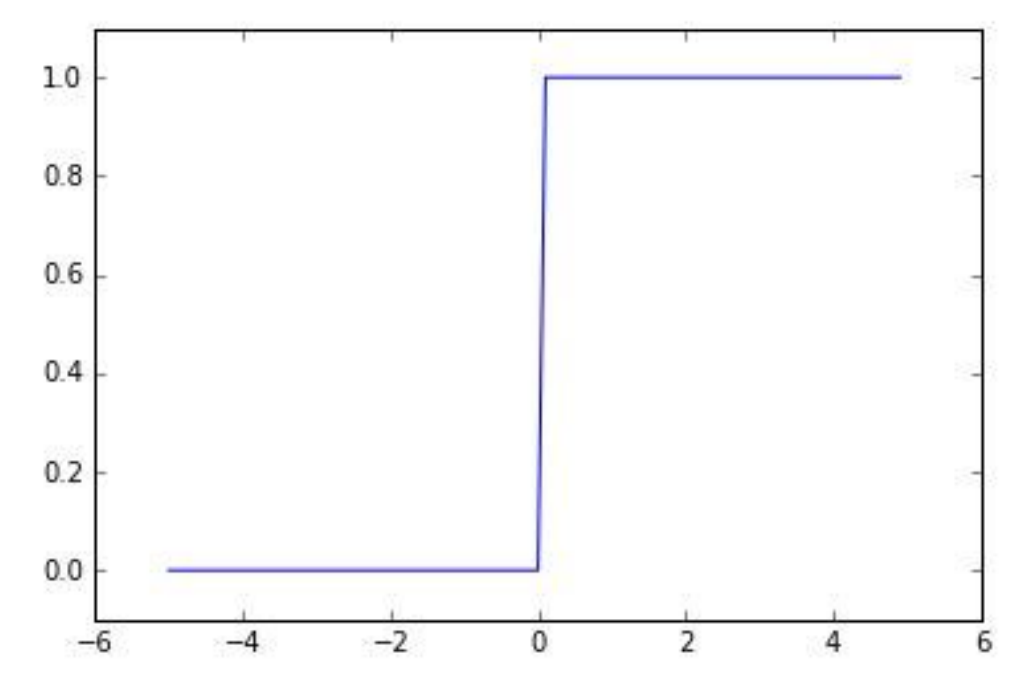
활성화 함수(Activation Function)

Linear function(선형함수=항등함수) → 회귀



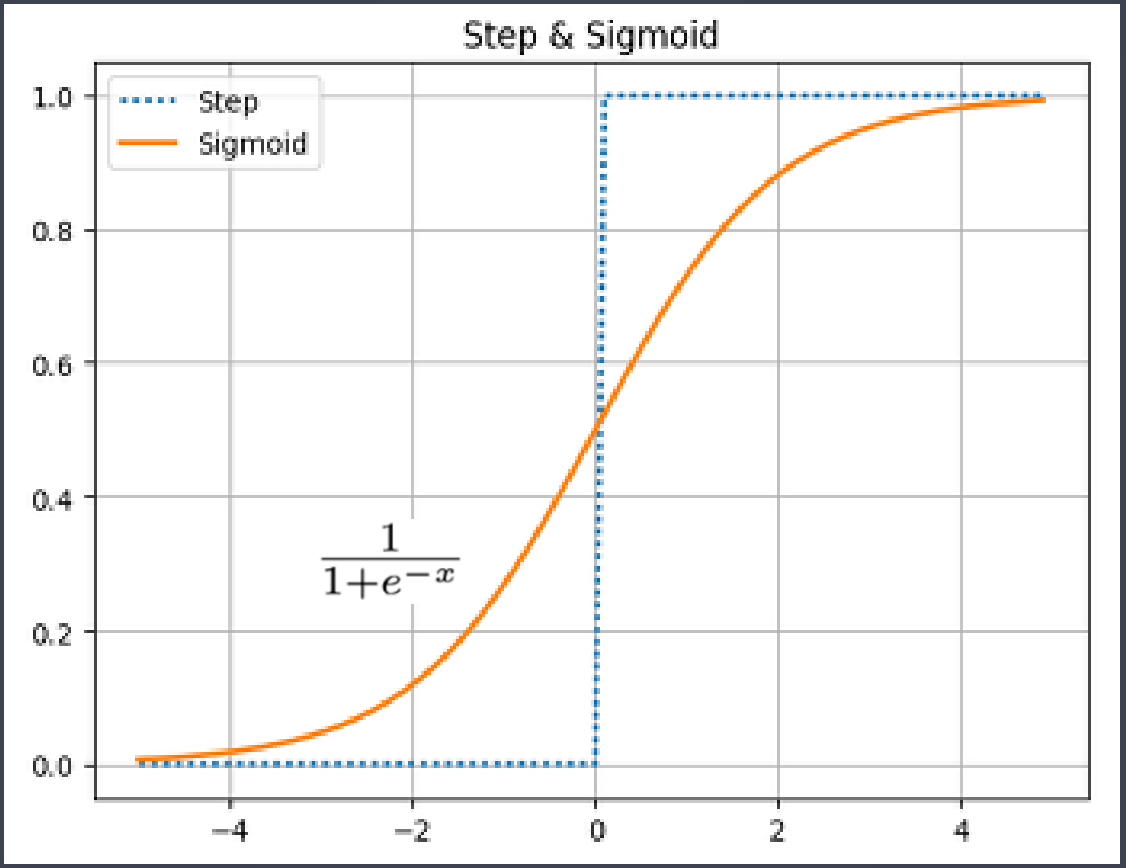
활성화 함수(Activation Function)

Step function(계단 함수) → 분류의 초기 활성화 함수



활성화 함수(Activation Function)

Sigmoid 함수 → 이진분류



활성화 함수(Activation Function) 정리

1. 중간층에 활성화 함수로 비선형 함수를 사용하는 이유

- 계단 함수(step)와 시그모이드 함수(sigmoid)는 비선형 함수이다.
- 중간층 활성화 함수로 선형함수(linear)를 사용하면 다층 구조의 효과를 살릴 수 없다.

활성화 함수(Activation Function) 정리

- 딥러닝 신경망에서 **다중분류** 문제를 해결하는 프로세스는 **각 클래스**에 대한 확률 값을 토대로 **가장 높은 확률 값**을 가지는 클래스로 최종분류를 진행함

- 각 레이블의 확률들을 알기 위해 **출력층 퍼셉트론 개수**를 **클래스 개수**와 맞춰야 함**(하나의 퍼셉트론이 하나의 클래스에 대한 확률 값을 출력)**

- 또한 다중 분류 문제를 풀 경우 정답 데이터를 **원 핫 인코딩** 해야 함

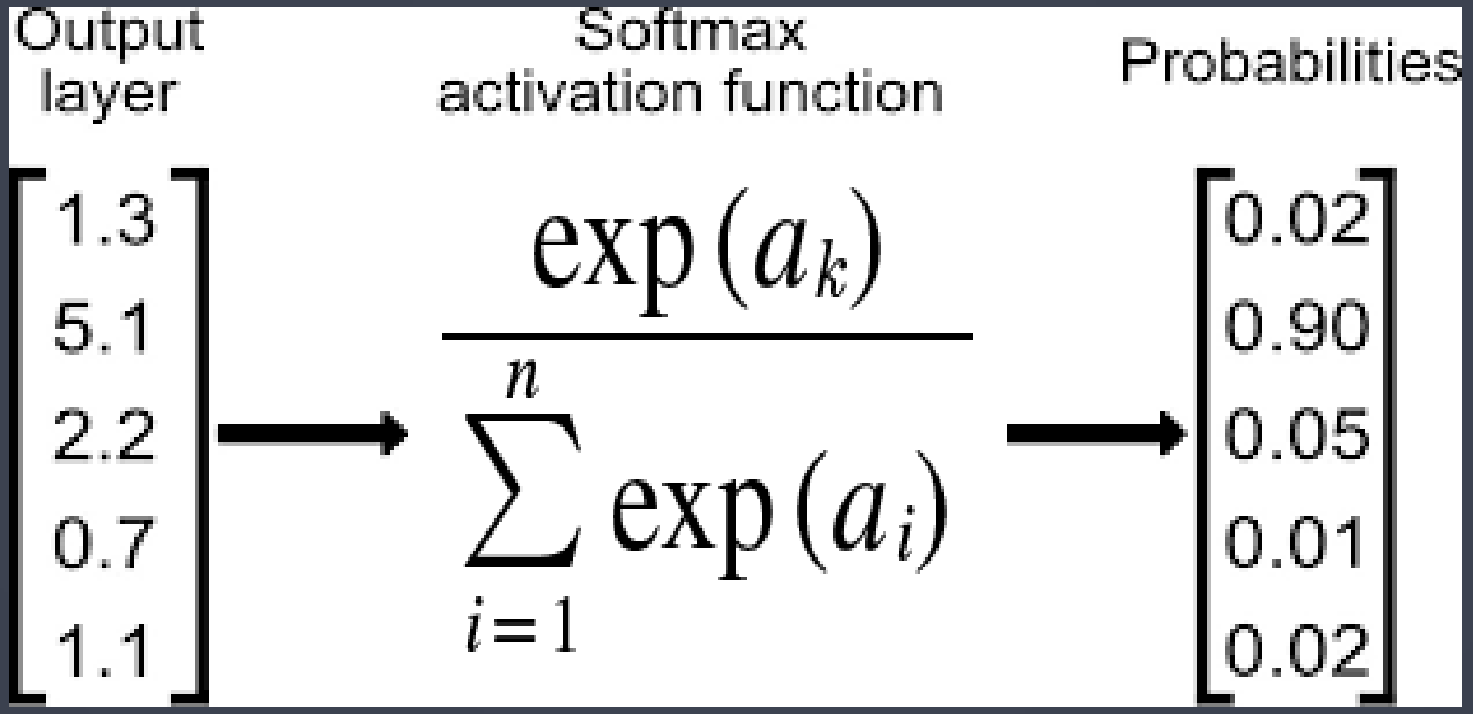
- 신경망 학습을 위해서는 원 핫 인코딩 된 정보(0,1)와 출력층의 각 퍼셉트론이 예측한 확률(0~1)과의 **오차**를 바탕으로 신경망이 학습하게 됨

활성화 함수(Activation Function) 정리

소프트맥스(softmax) 함수 → 다중분류

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

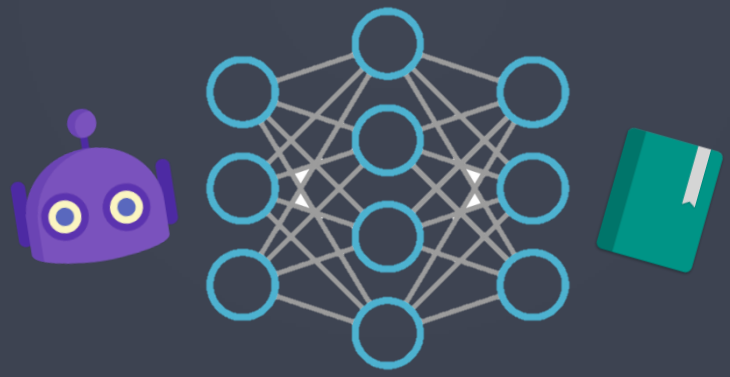
sigmoid에 비해 **예측 오차의 평균을 줄여주는 효과**



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

오차역전파

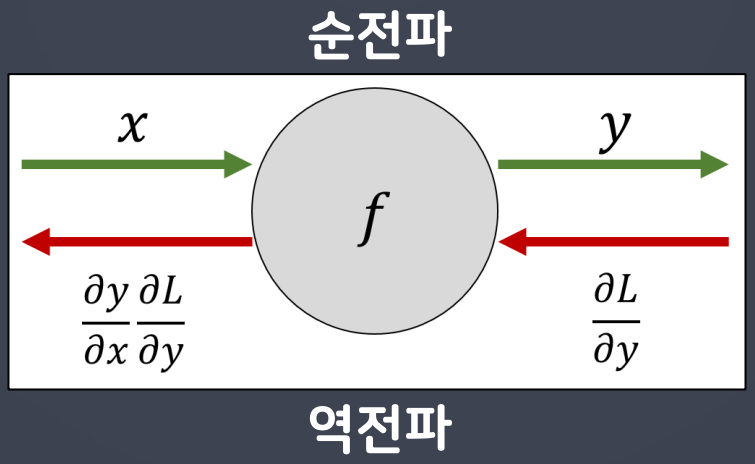


오차 역전파(Back Propagation)

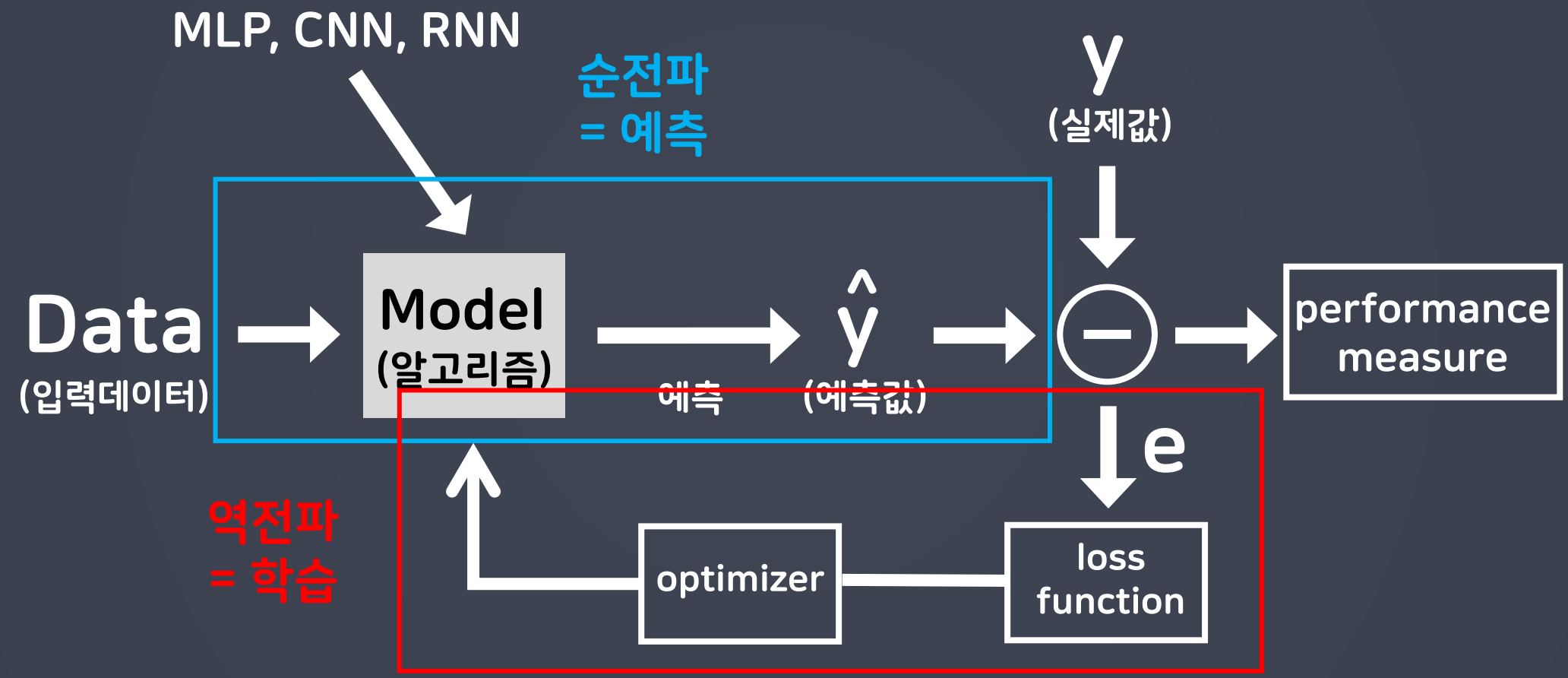
오차 역전파(Back Propagation)

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

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



오차 역전파(Back Propagation)



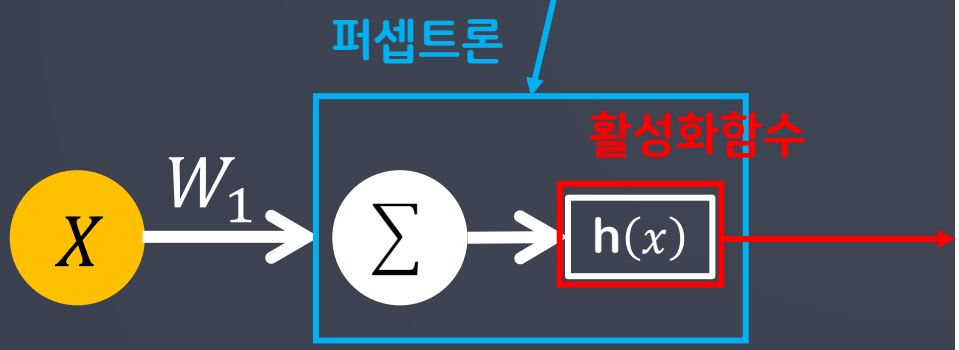
오차 역전파(Back Propagation)

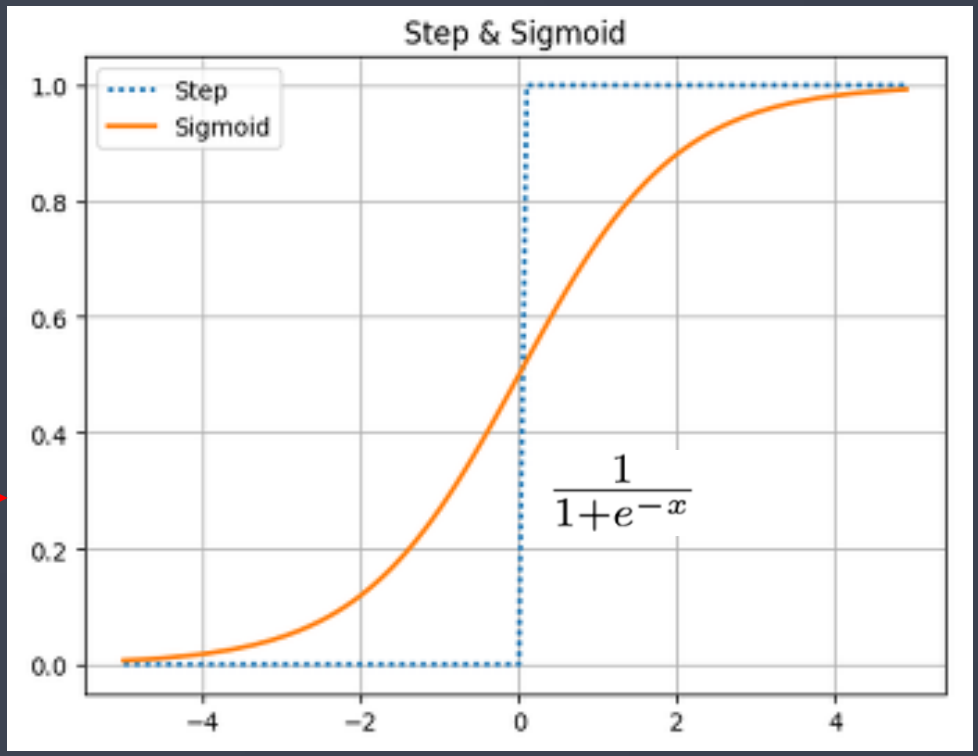
손실함수 및 Sigmoid 함수의 미분

- 신경망이 학습하기 위해서는 경사하강법(loss 함수를 미분)을 사용.

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

예측값

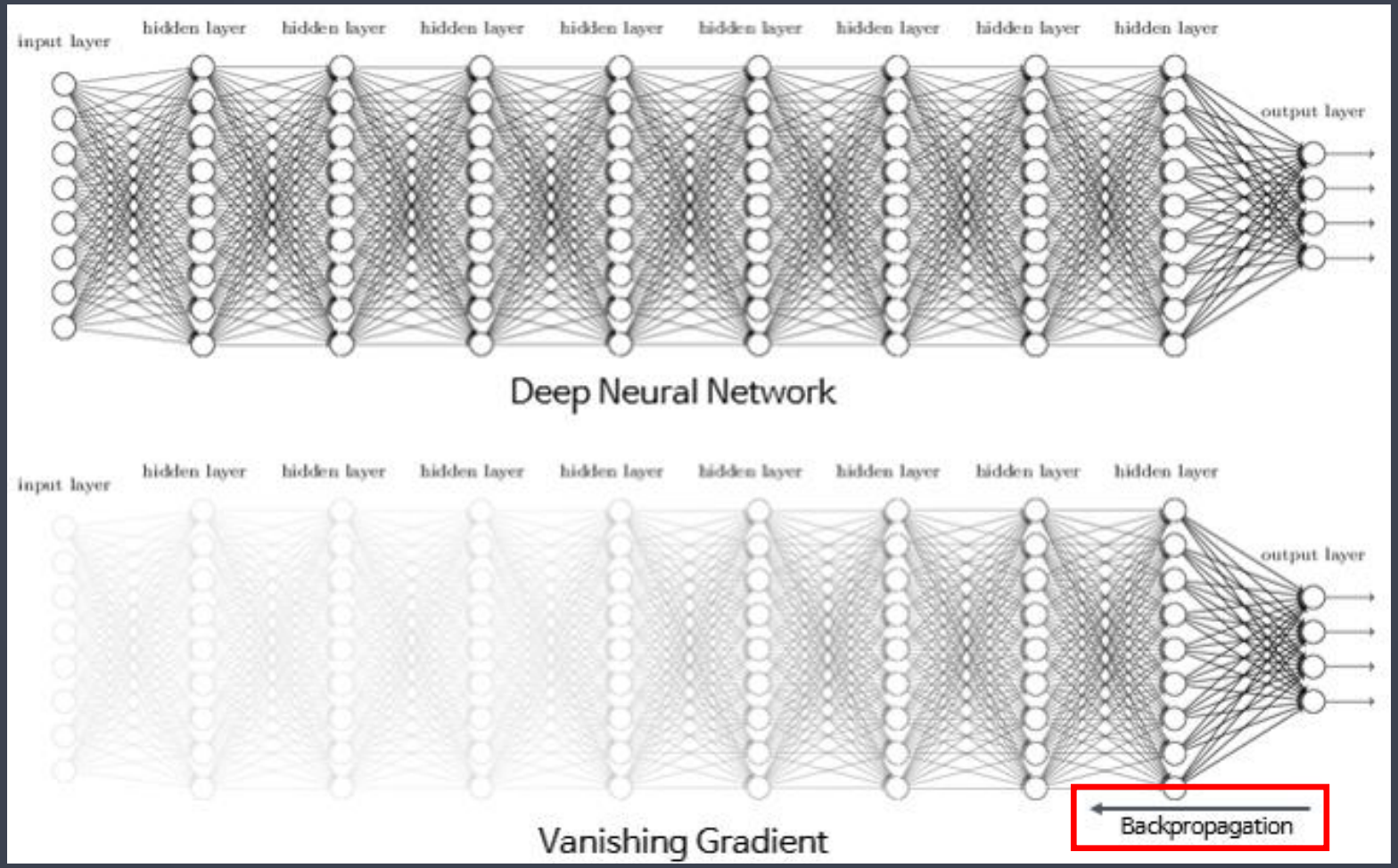




오차 역전파(Back Propagation)

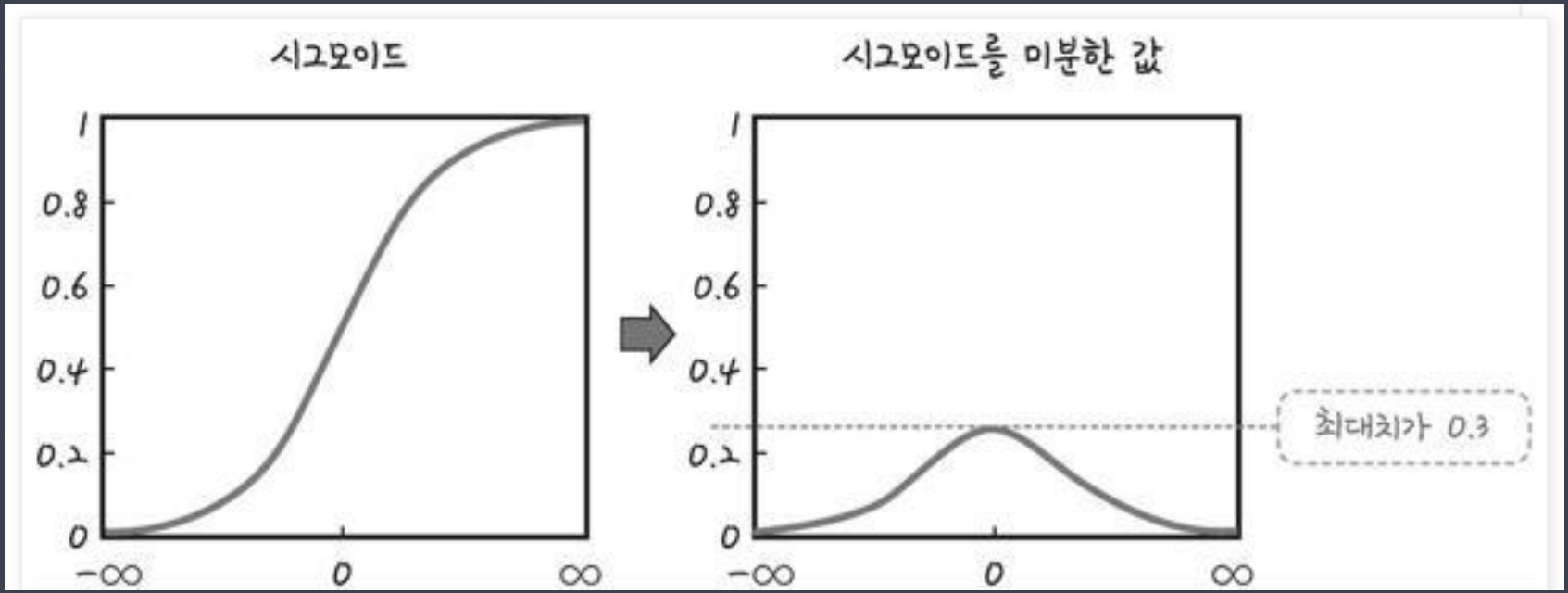
Sigmoid 함수의 문제점

- 기울기 소실 문제(Vanishing Gradient)

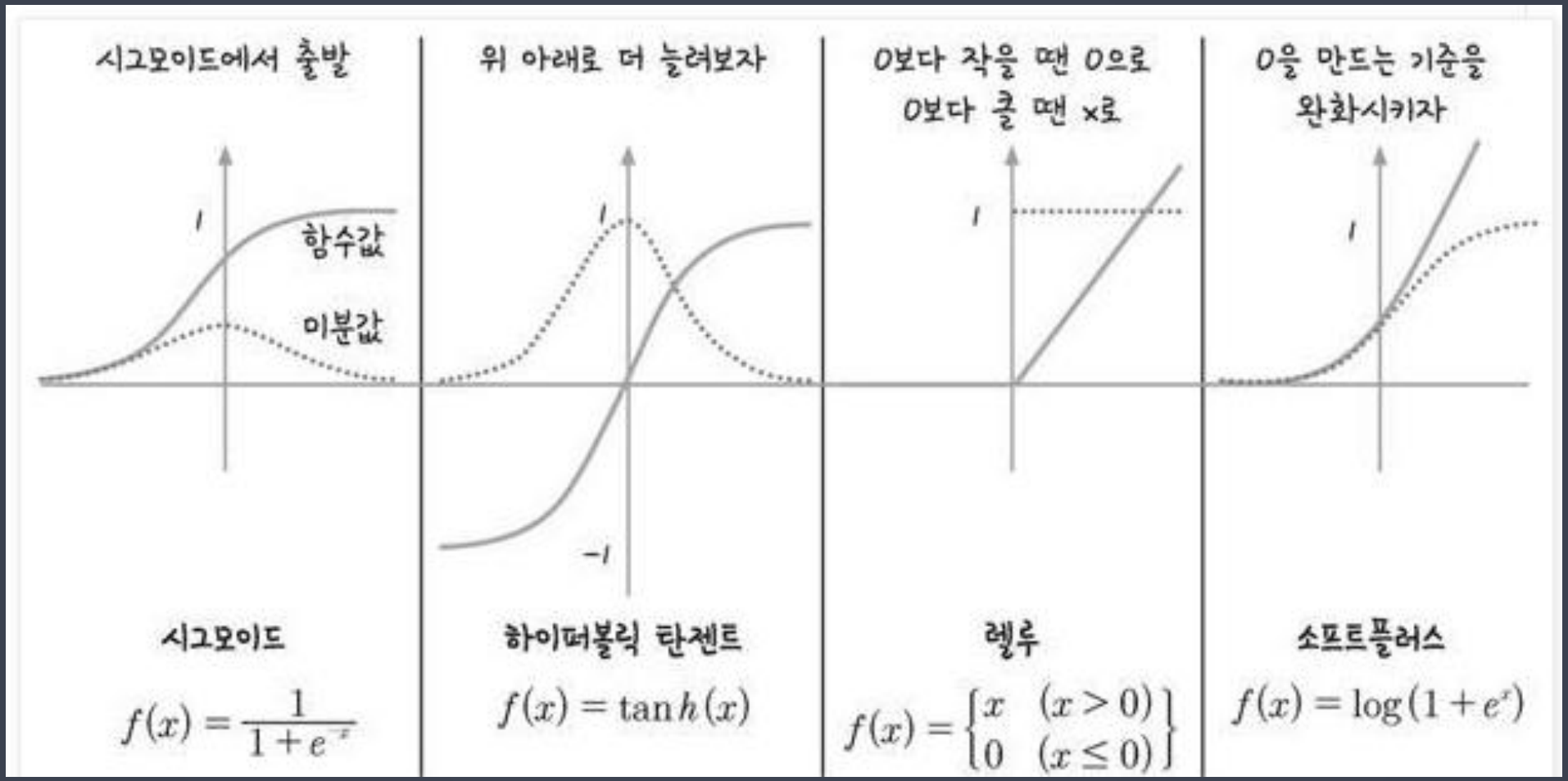


오차 역전파(Back Propagation)

Sigmoid 함수의 문제점



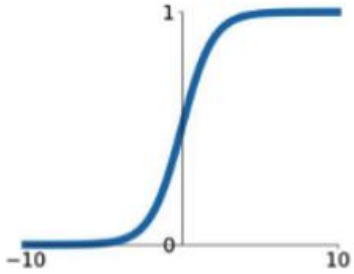
활성화 함수(Activation)의 종류



활성화 함수(Activation)의 종류

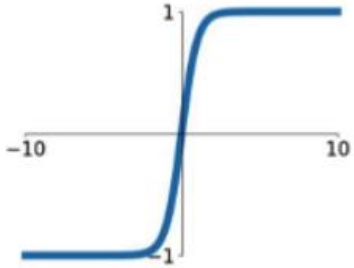
Sigmoid

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



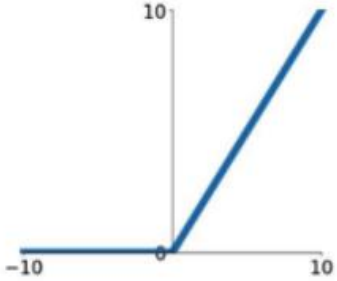
tanh

$$\tanh(x)$$



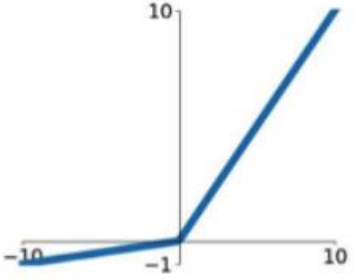
ReLU

$$\max(0, x)$$



Leaky ReLU

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

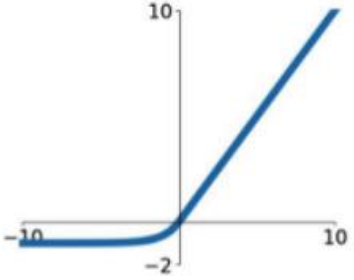


Maxout

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

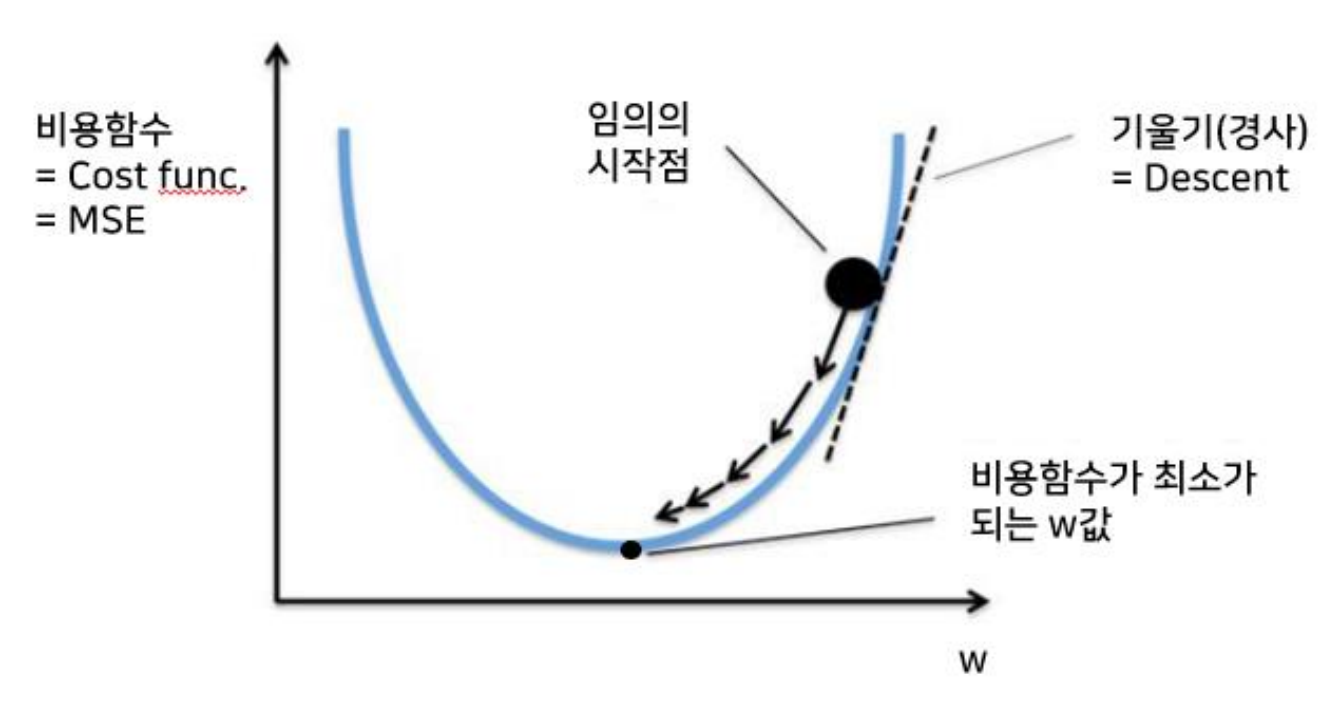
ELU

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

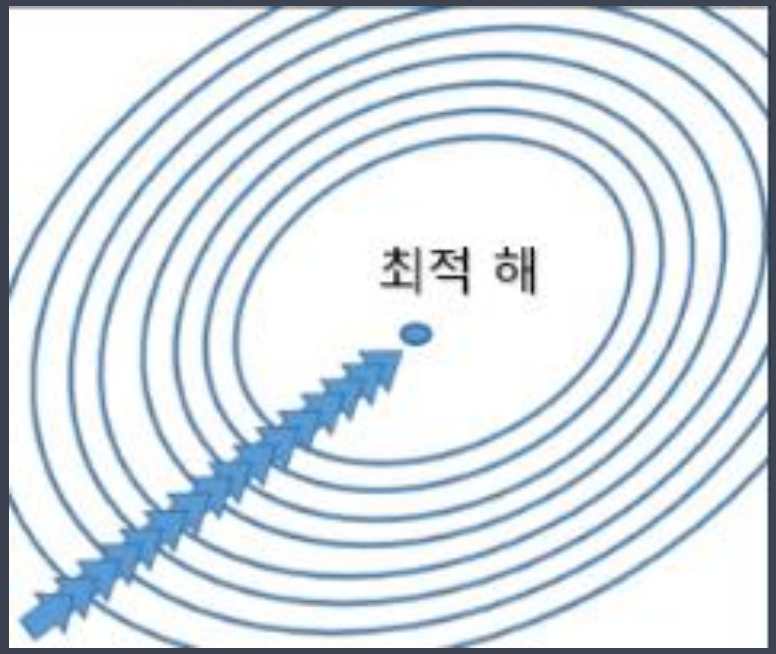


최적화함수(Optimizer)의 종류

경사하강법(Gradient Descent Algorithm)

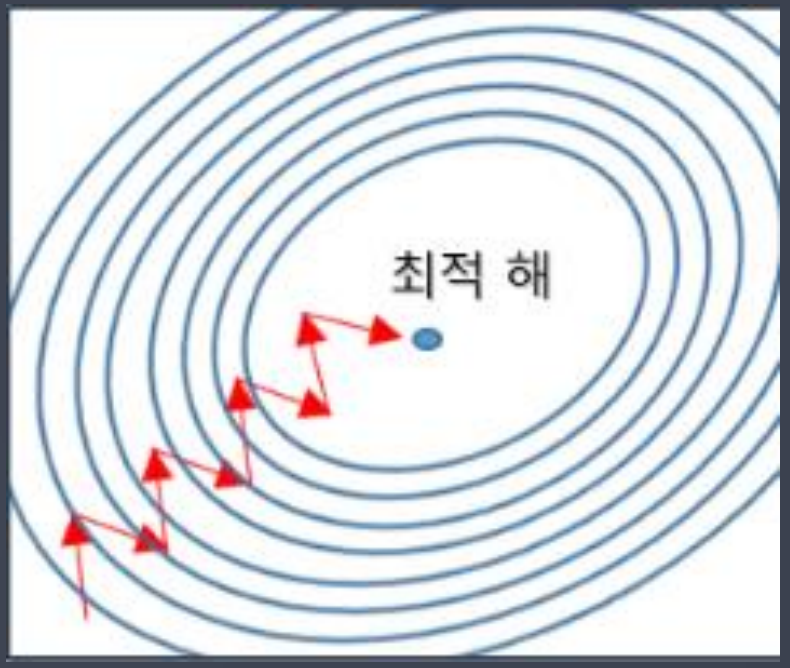


최적화함수(Optimizer)의 종류



경사하강법
(Gradient Descent)

전체 데이터를 이용해 업데이트



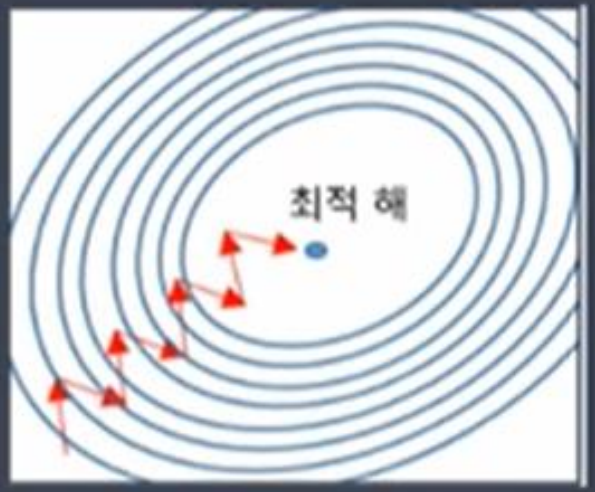
확률적경사하강법
(Stochastic Gradient Descent)

확률적으로 선택된 일부 데이터를 이용해 업데이트

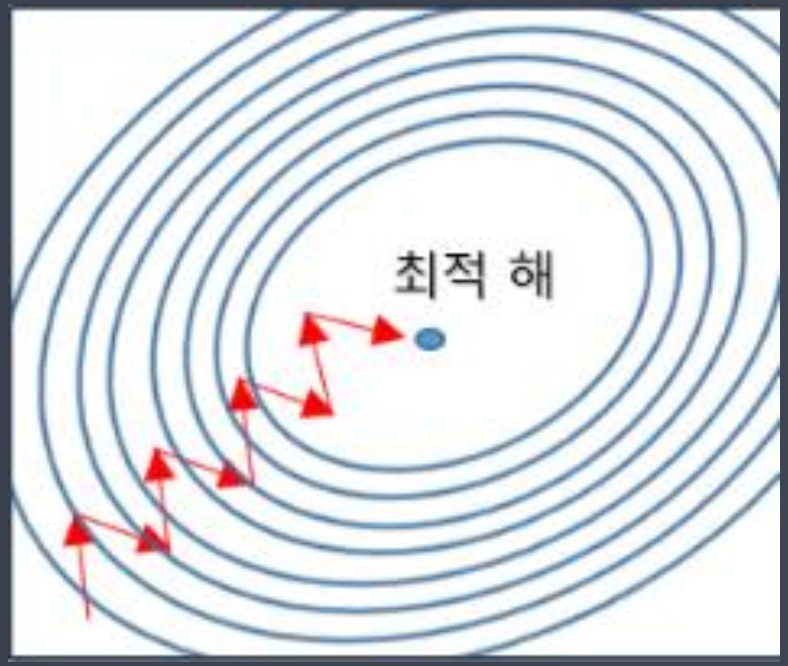
경사하강법(Gradient Descent Algorithm)

장·단점 (SGD)

- 배치 GD보다 더 빨리, 더 자주 업데이트를 한다.
- 지역 최저점을 빠져 나갈 수 있다.
- 탐색 경로가 비효율적이다. (진폭이 크고 불안정)

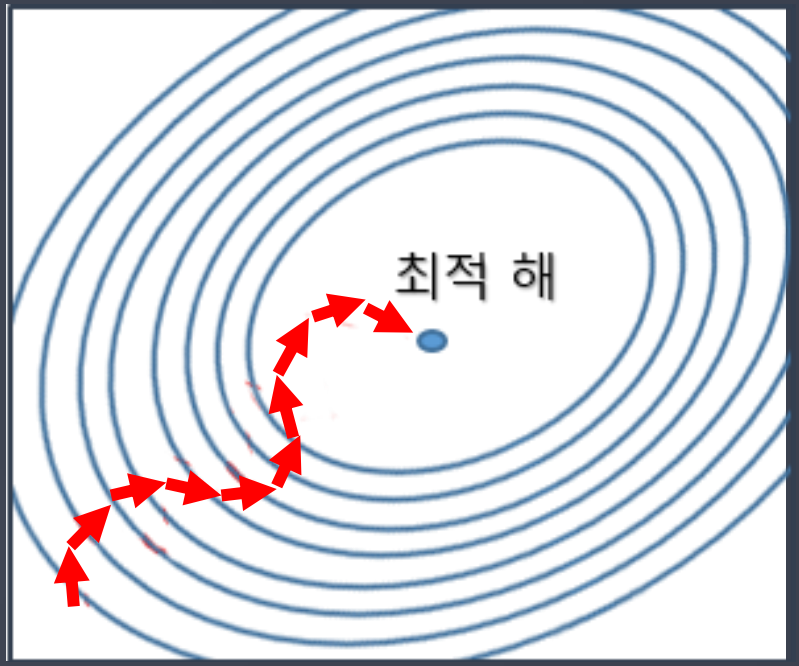


최적화함수(Optimizer)의 종류



확률적경사하강법
(Stochastic Gradient Descent)

확률적으로 선택된 일부 데이터를 이용해 업데이트



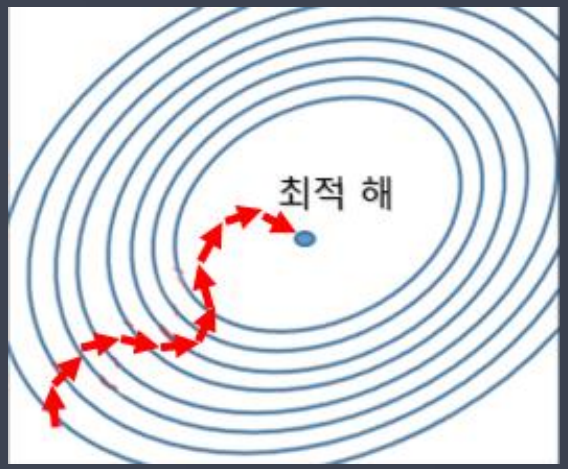
모멘텀
(Momentum)

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

최적화함수(Optimizer)의 종류

특징 (Momentum)

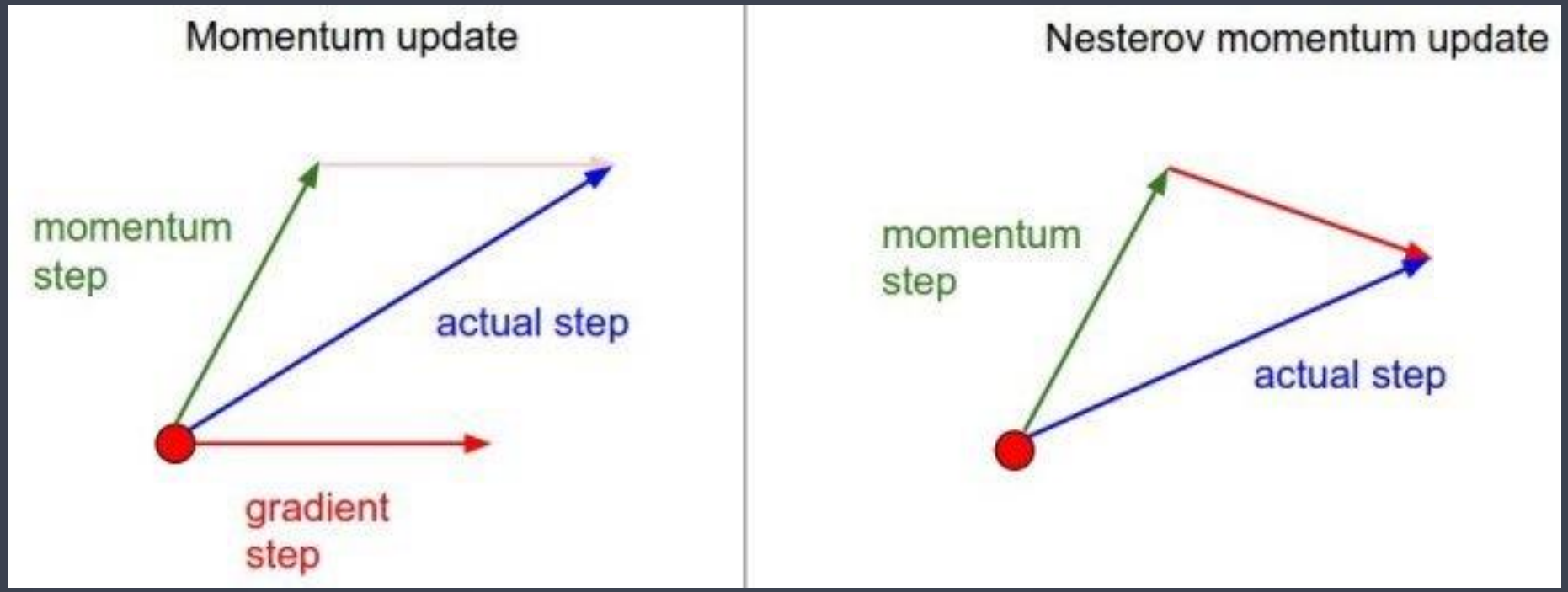
- 가중치를 수정하기 전 **이전 방향을 참고**하여 업데이트
- 지그재그 형태로 이동하는 현상이 줄어든다
- α는 Learning Rate, m은 momentum 계수 (보통 0.9)



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

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

최적화함수(Optimizer)의 종류



네스테로프 모멘텀

(Nesterov Accelrated Gradient)

개선된 모멘텀 방식

최적화함수(Optimizer)의 종류

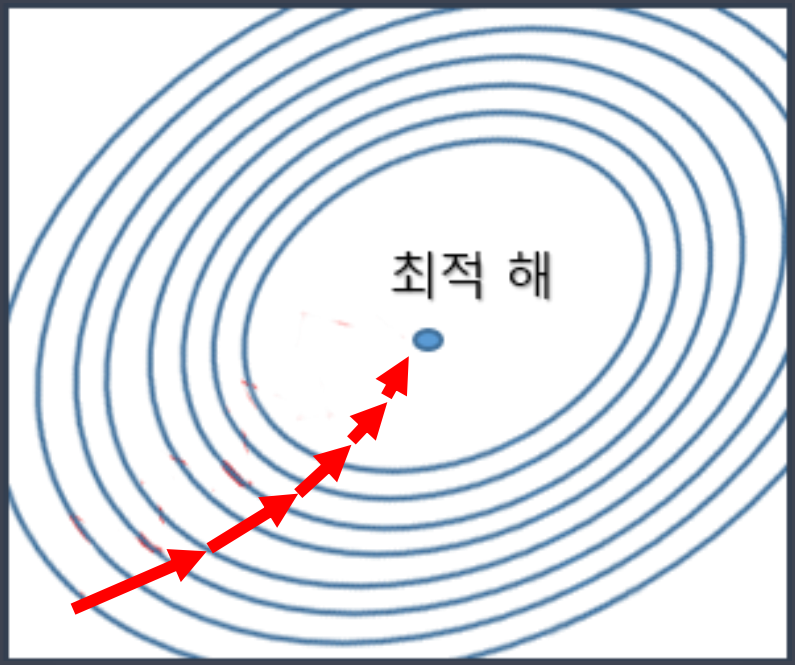
특징 (NAG)

- w, b값 업데이트 시 모멘텀 방식으로 먼저 더한 다음 계산
- 미리 해당 방향으로 이동한다고 **가정**하고 기울기를 계산해본 뒤 실제 업데이트 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

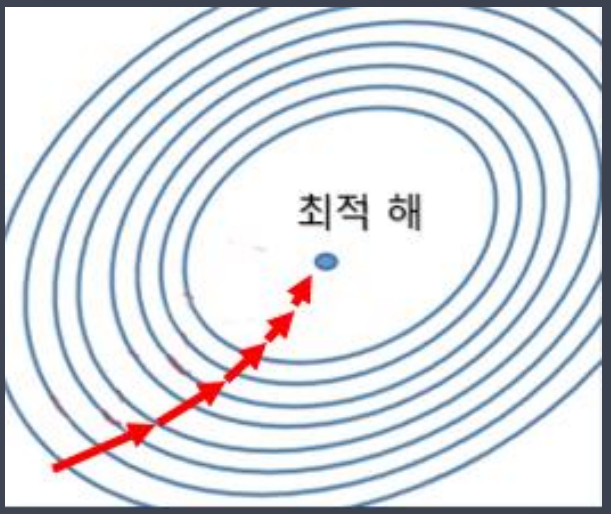
(Adaptive Gradient)

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

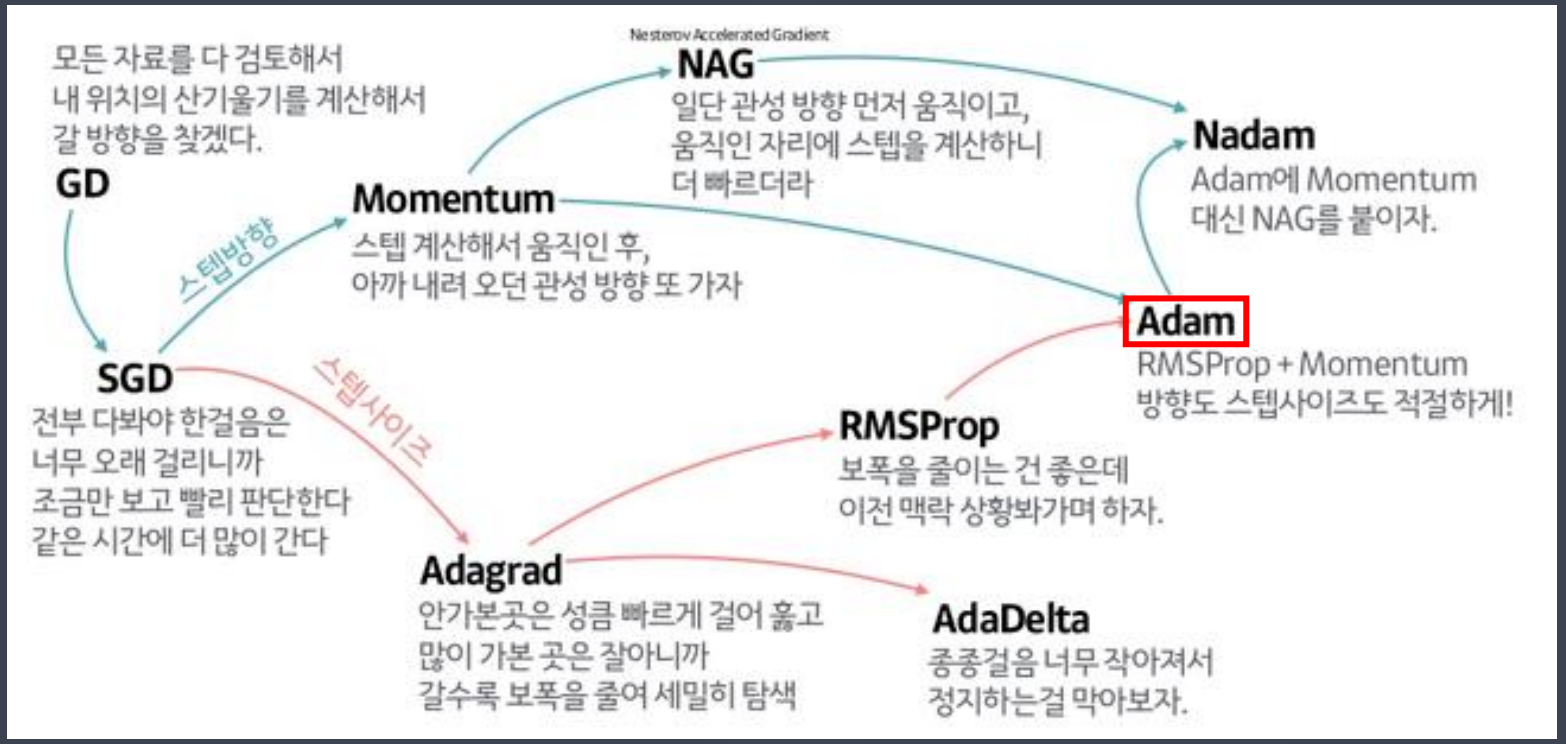


$$G(t) = G(t-1) + \left(\frac{\partial}{\partial w(t)} Cost(w(t))\right)^2$$

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

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

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