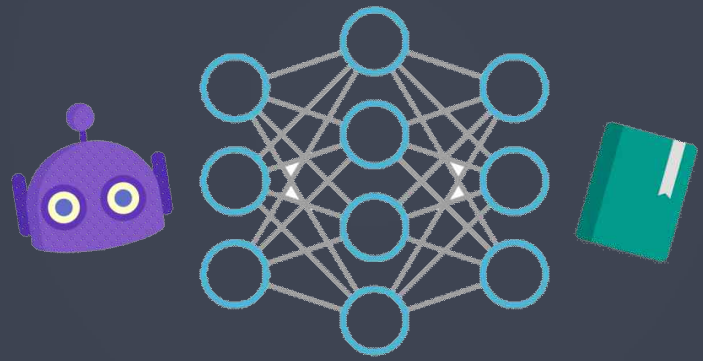
Deep Learning

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



START

스마트인재개발원
Smart Human Resources Development

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.

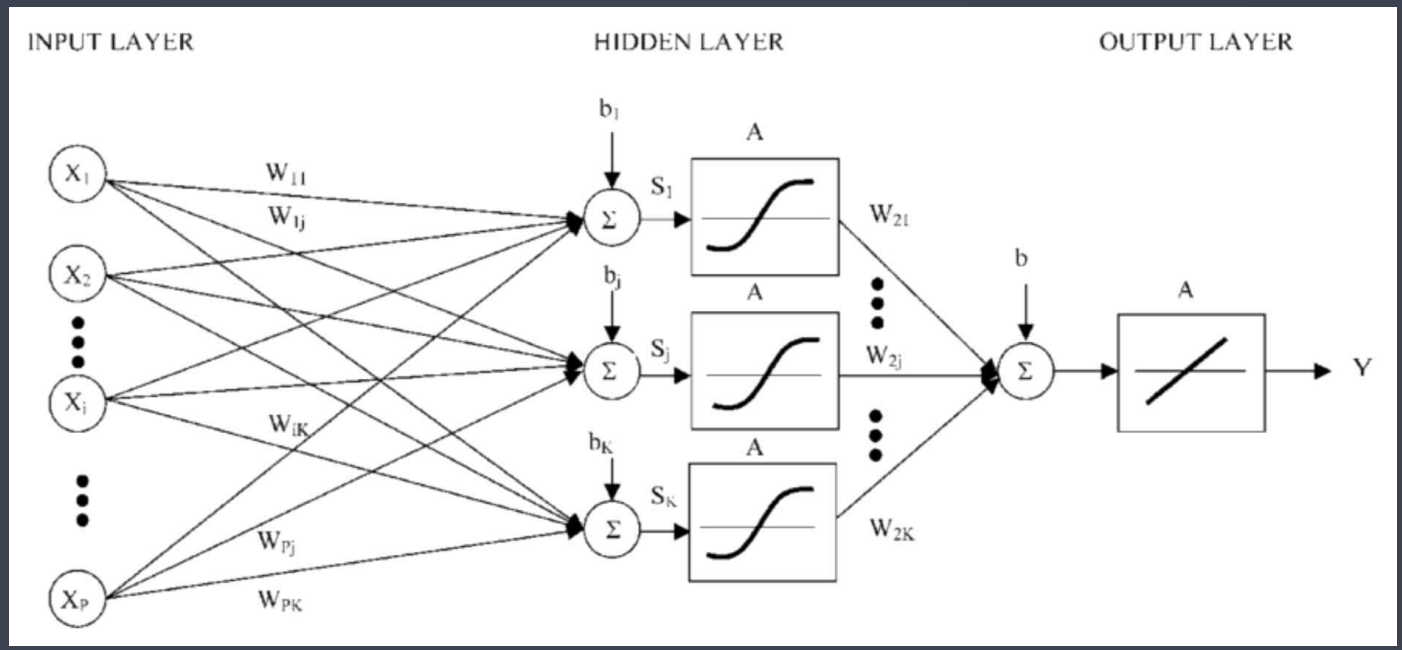
활성화 함수(Activation Function)

활성화 함수(Activation Function)

- 신경망은 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 전달하게 됨
- 활성화 함수는 이런 부분까지 사람과 유사하게 구현하여 사람처럼 사고하고 행동하는 인공지능 기술을 실현하기 위해 도입됨
- 또한 선형모델을 기반으로 하는 딥러닝 신경망에서 분류 문제를 해결하기 위해서 비선형 활성화 함수가 필요함

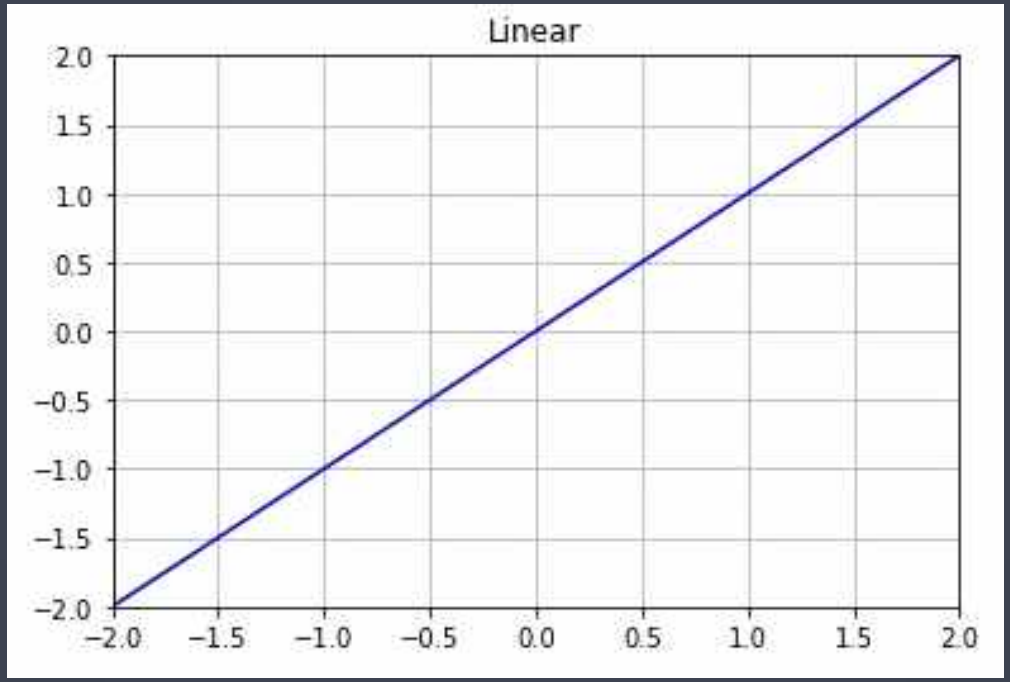
활성화 함수(Activation Function)

층에 따라 다른 활성화 함수를 사용 할 수 있다.



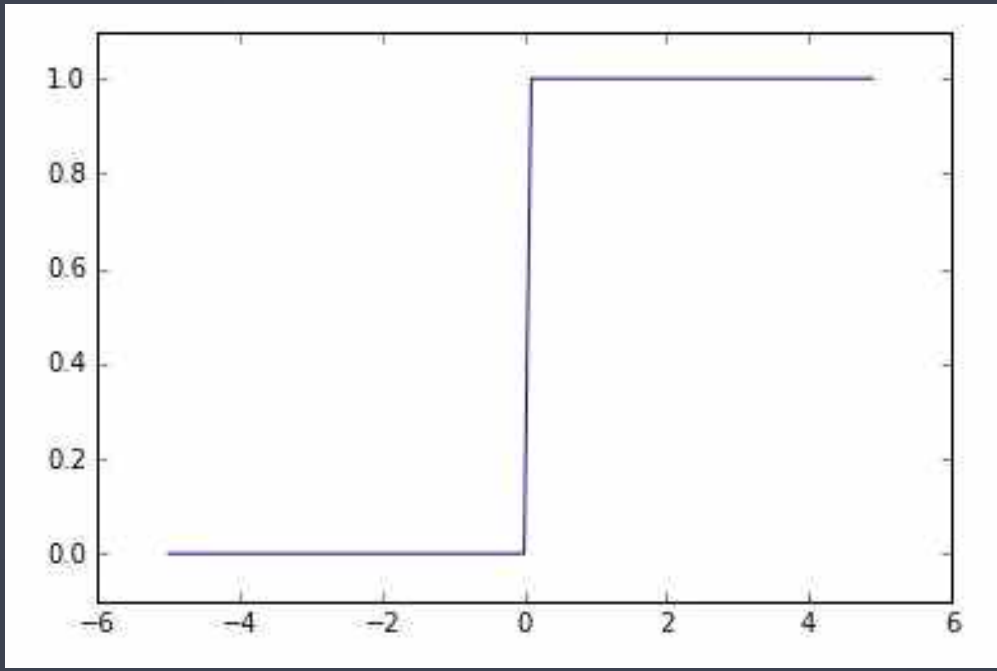
활성화 함수(Activation Function)

Linear function(선형함수=항등함수) → 회귀



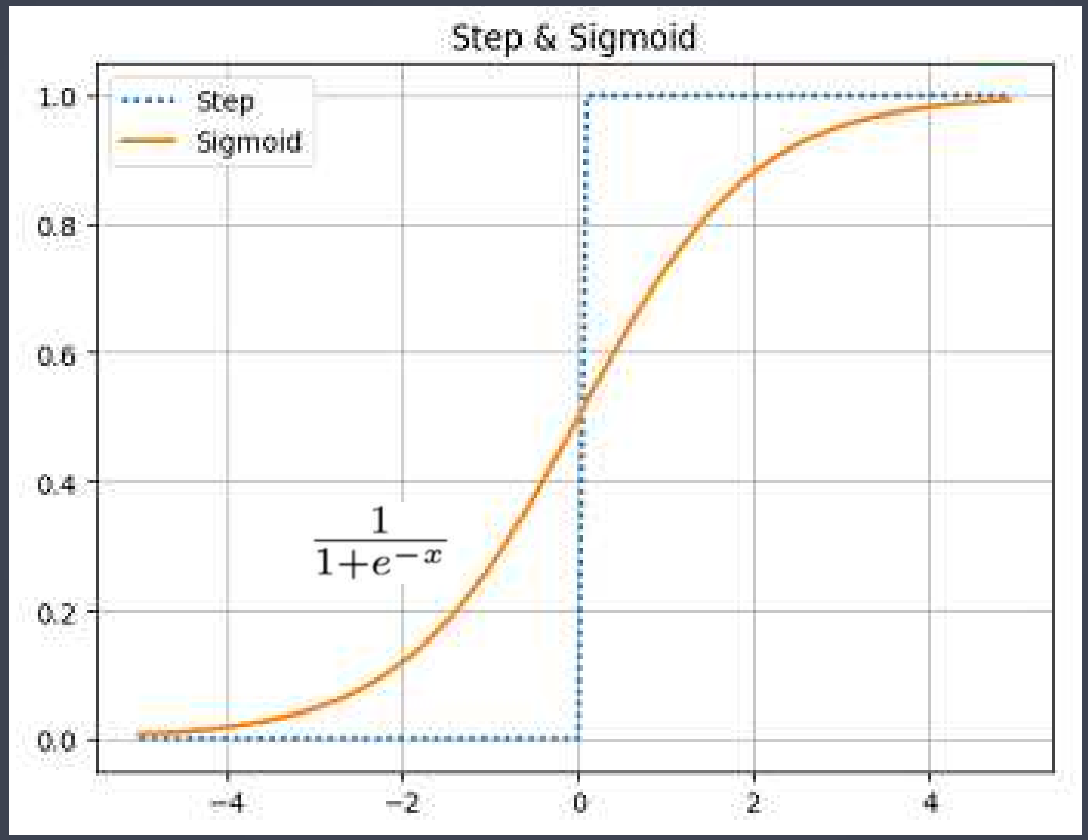
활성화 함수(Activation Function)

Step function(계단 함수) → 분류의 초기 활성화 함수



활성화 함수(Activation Function)

Sigmoid 함수 → 이진분류



활성화 함수(Activation Function) 정리

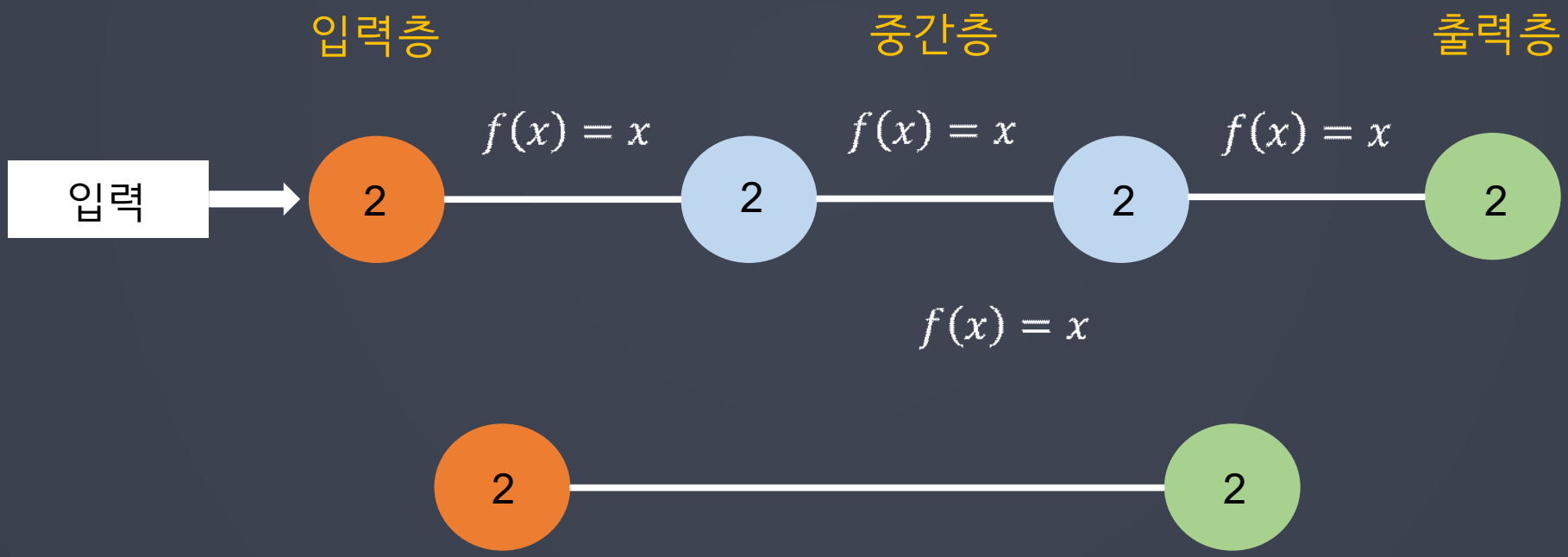
1. 중간층에 활성화 함수로 비선형 함수를 사용하는 이유

- 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

2. 중간층에 활성화 함수로 선형 함수를 사용하게 된다면

선형함수(linear) 수식은 h(x) = x



활성화 함수(Activation Function) 정리

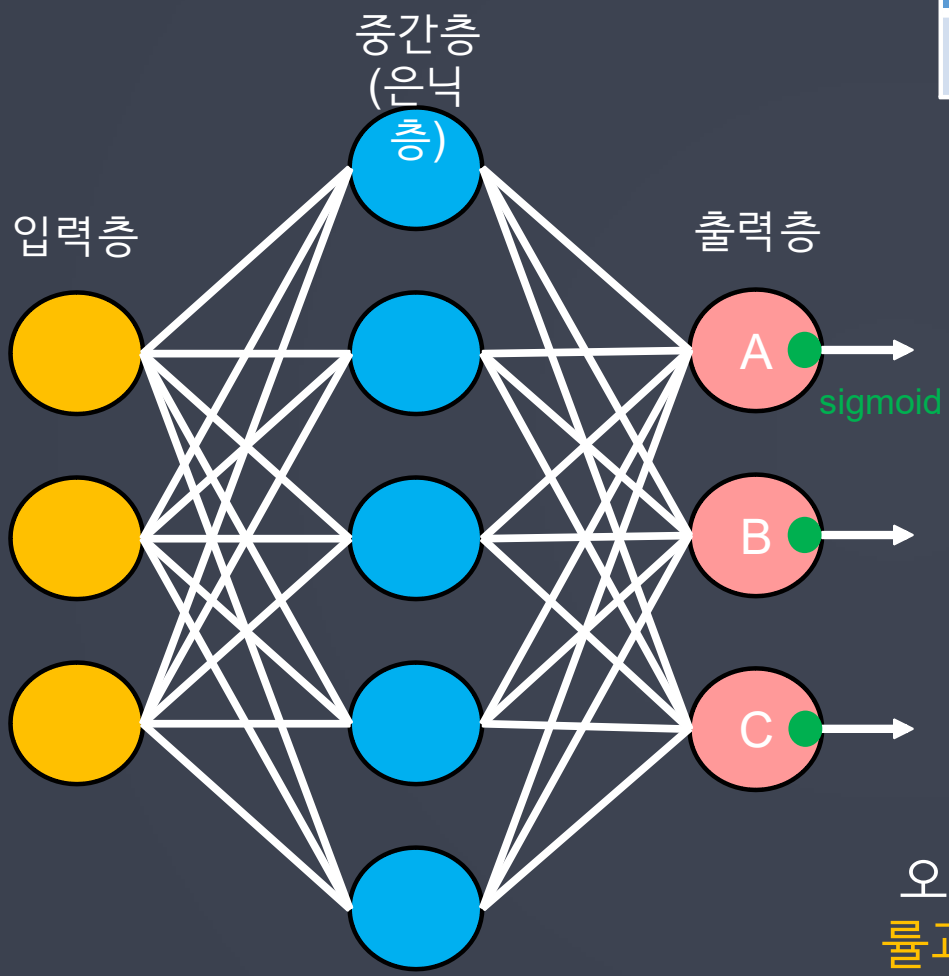
· 딥러닝 신경망에서 다중분류 문제를 해결하는 프로세스는 각 클래스에
대한 확률 값을 토대로 가장 높은 확률 값을 가지는 클래스로 최종 분류
를 진행함

· 각 레이블의 확률들을 알기 위해 출력층 퍼셉트론 개수를 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 퍼셉

활성화 함수(Activation Function) 정리



A	B	C
1	0	0

원핫인코딩 된 정답데이터
(A가 정답이라고 가정)

	예측 확률 값	오차
A	0.9	0.1
B	0.7	0.7
C	0.4	0.4

오차를 제대로 파악하기 위해서는 예측확률과 실제 정답의 범위(0~1)가 같아야 비교 가능

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

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

입력값들의 지수함수의 합

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
2진 분류	sigmoid(로지스틱 함수)	binary_crossentropy
다중 분류	softmax(소프트맥스 함수)	categorical_crossentropy

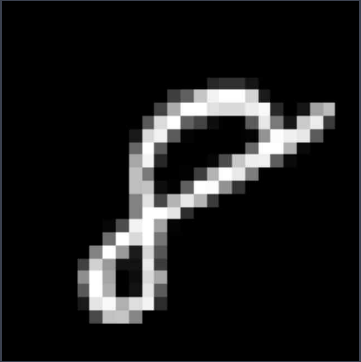
코드 실습

iris 데이터 신경망으로 풀기 (다중 분류)

코드 실습

Keras로 MNIST 손글씨 이미지 데이터 분류 모델을 만들어보자

코드 실습

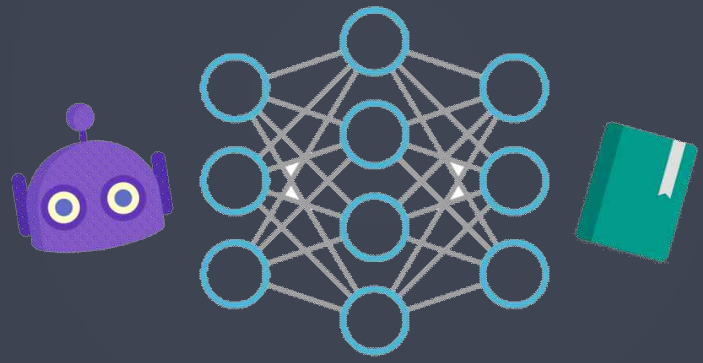


0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	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

오차 역전파
(Back Propagation)



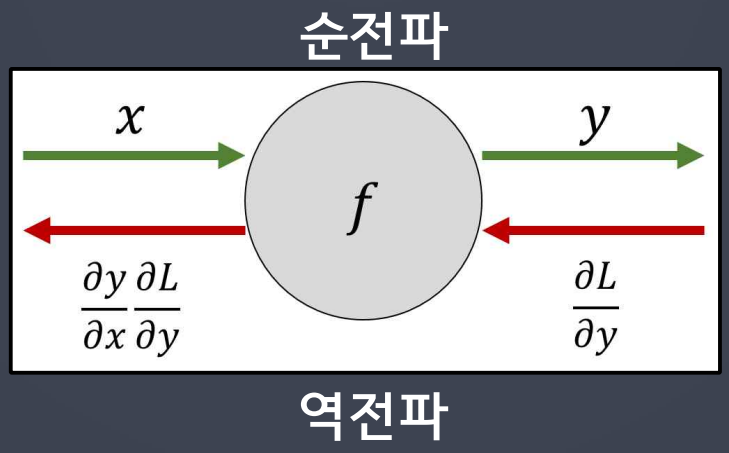
START

스마트인재개발원
Smart Human Resources Development

오차 역전파(Back Propagation)

오차 역전파(Back Propagation)

- 순전파 : 입력 데이터를 입력층에서부터 출력층까지 정방향으로 이동시키며 출력 값을 예측해 나가는 과정
- 역전파 : 출력층에서 발생한 에러를 입력층 쪽으로 전파시키면서 최적의 결과를 학습해 나가는 과정

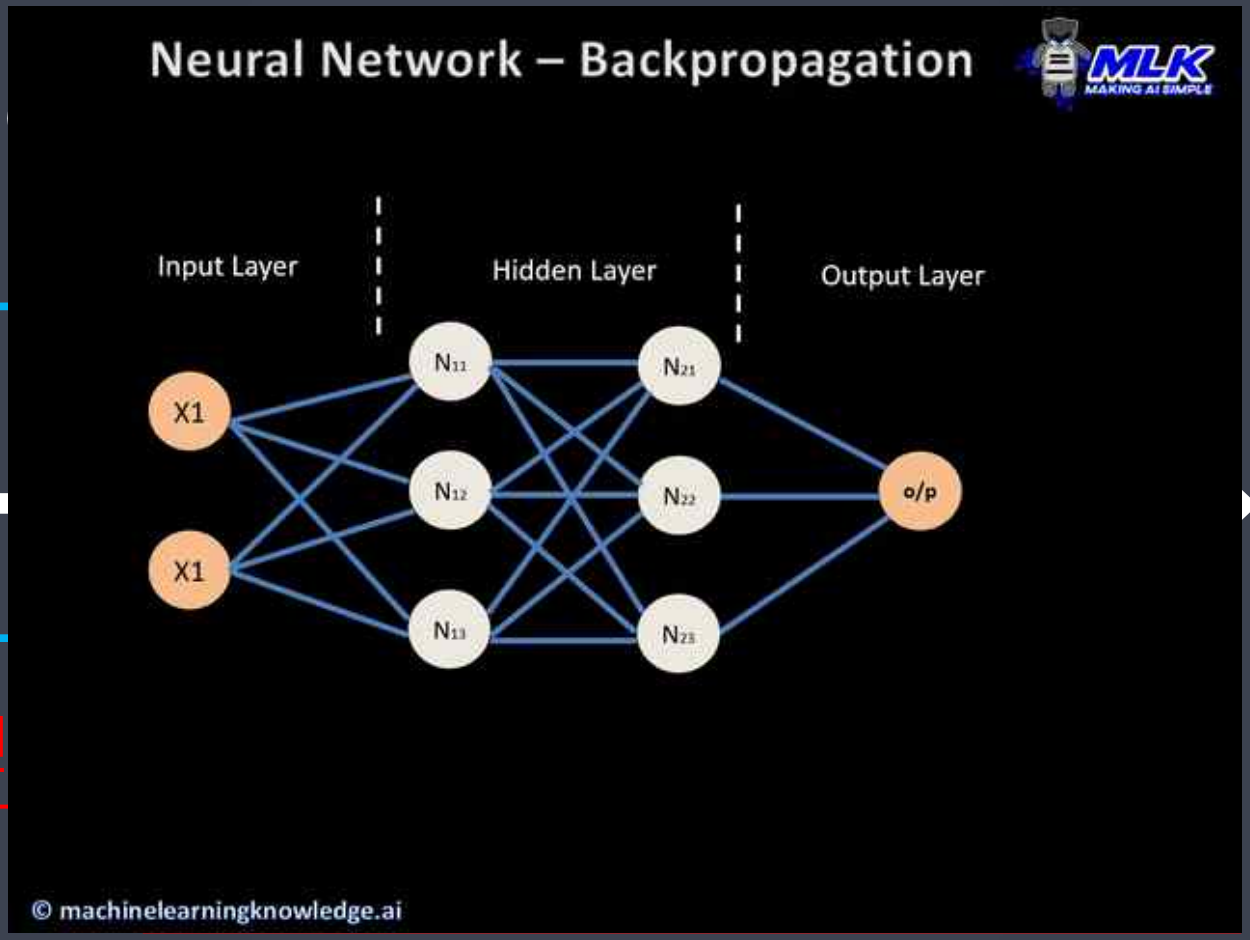


오차 역전파(Back Propagation)

MLP,

Data
(입력데이터)

역전
= 학



performance measure

출처:https://machinelearningknowledge.ai/animated-explanation-of-feed-forward-neural-network-architecture/

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

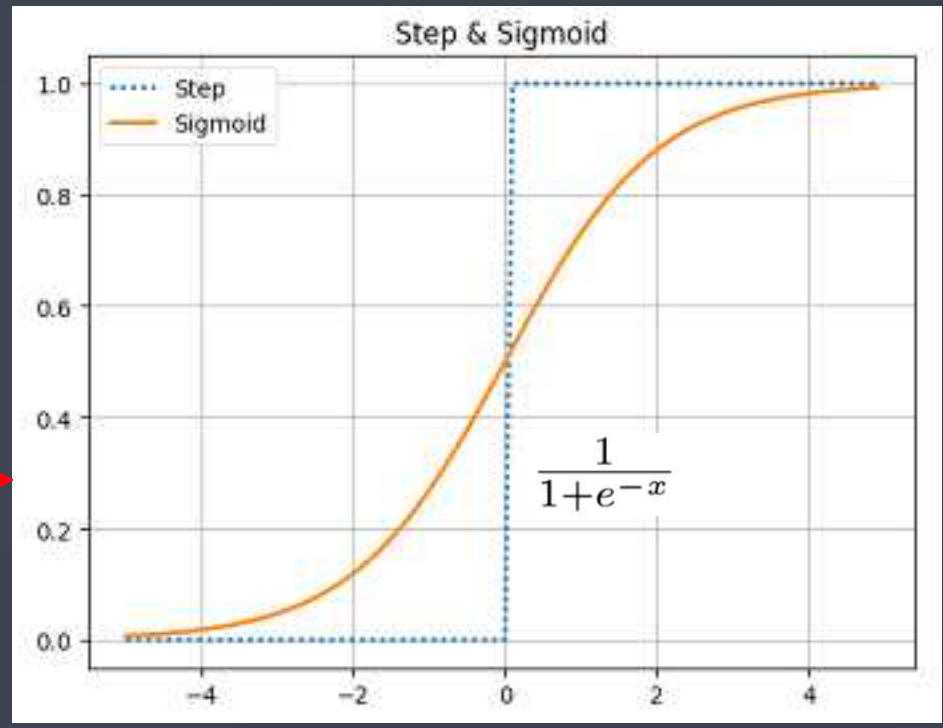
퍼셉트론

활성화함수

W_1

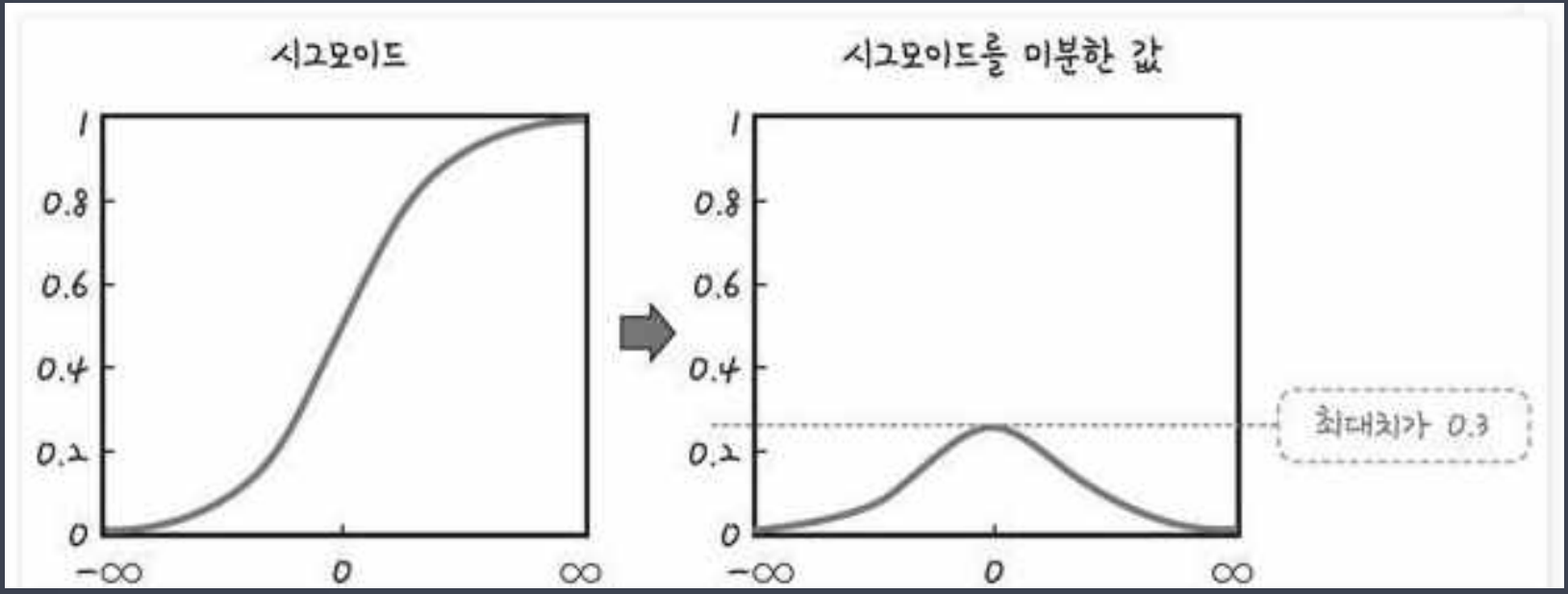
X

$H(x)$



오차 역전파(Back Propagation)

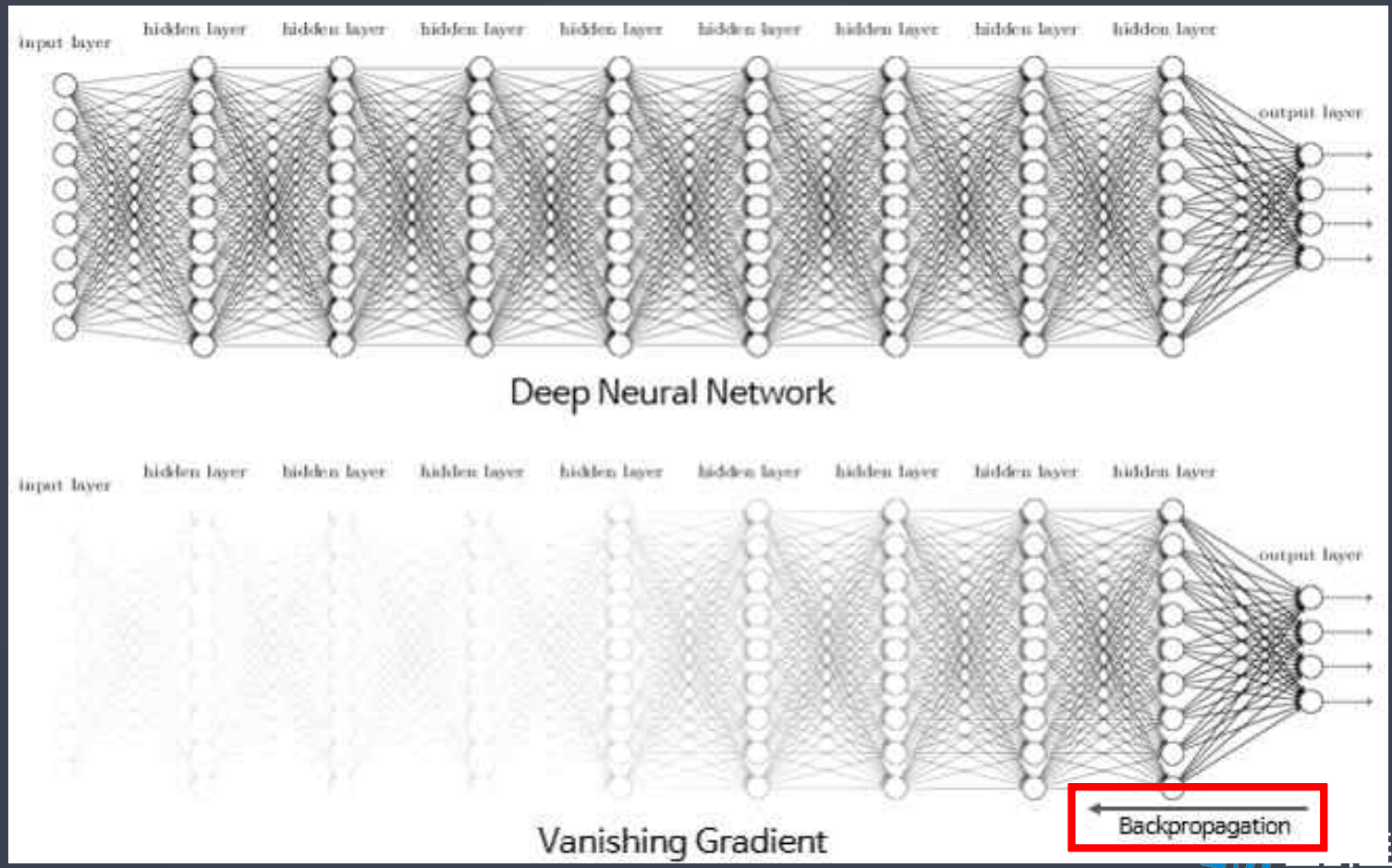
Sigmoid 함수의 문제점



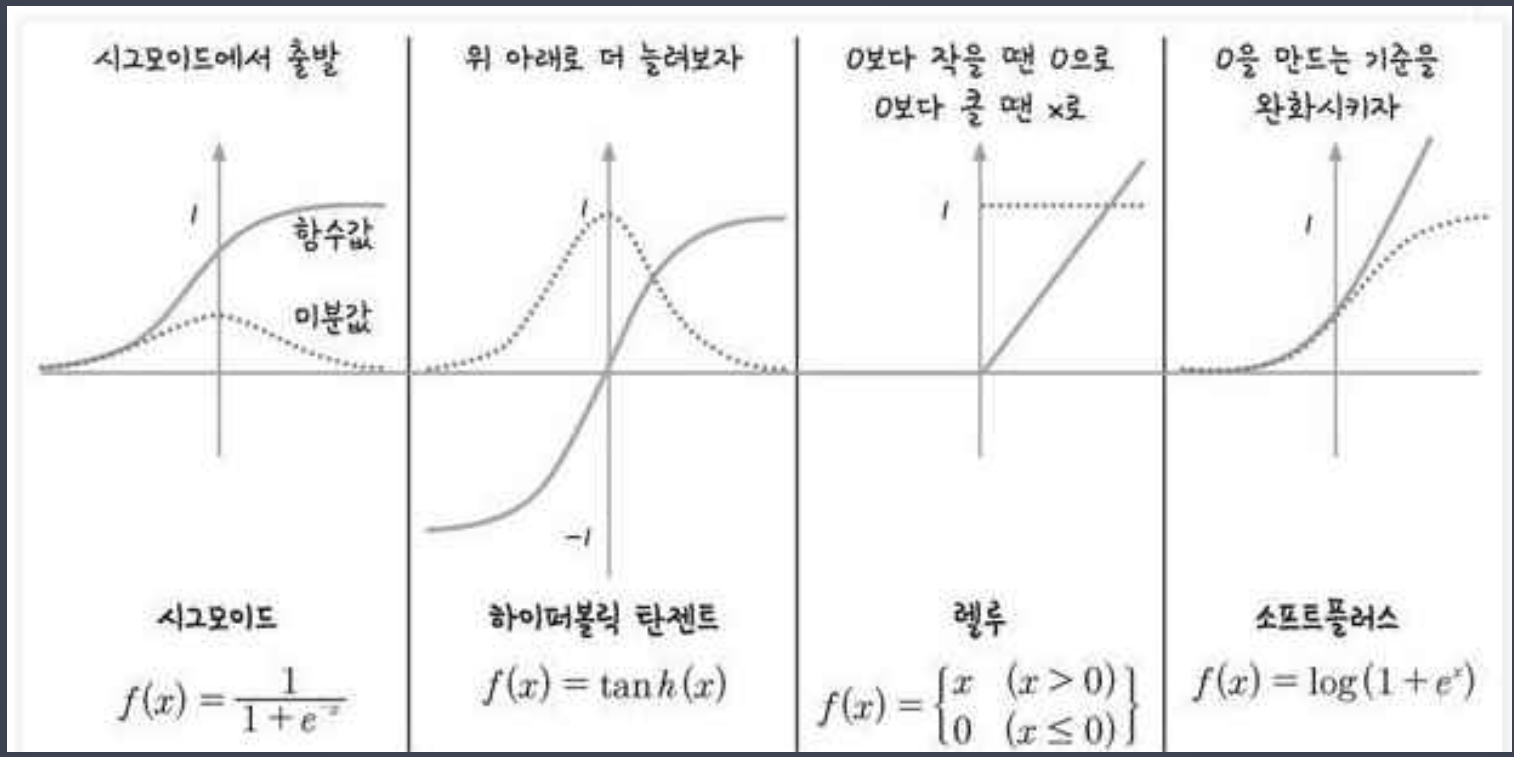
오차 역전파(Back Propagation)

Sigmoid 함수의 문제점

- 기울기 소실 문제(Vanishing Gradient)



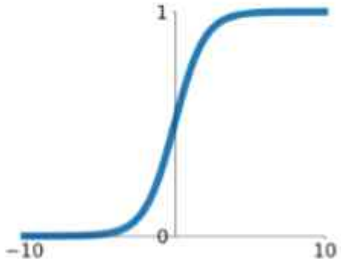
활성화 함수(Activation)의 종류



활성화 함수(Activation)의 종류

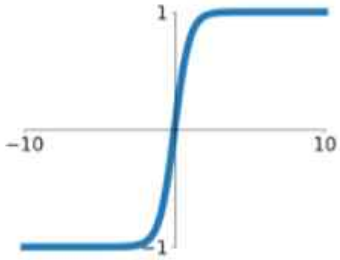
Sigmoid

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



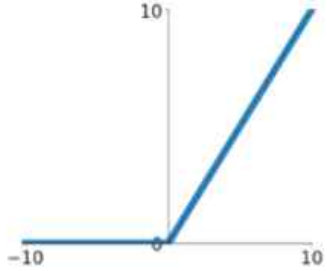
tanh

$$\tanh(x)$$



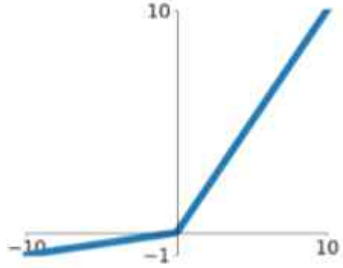
ReLU

$$\max(0, x)$$



Leaky ReLU

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

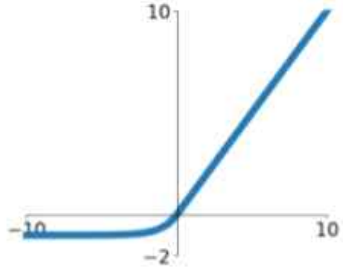


Maxout

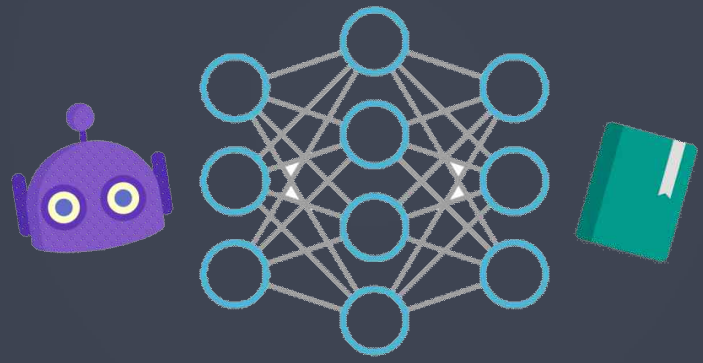
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

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



최적화 함수
(Optimizer)

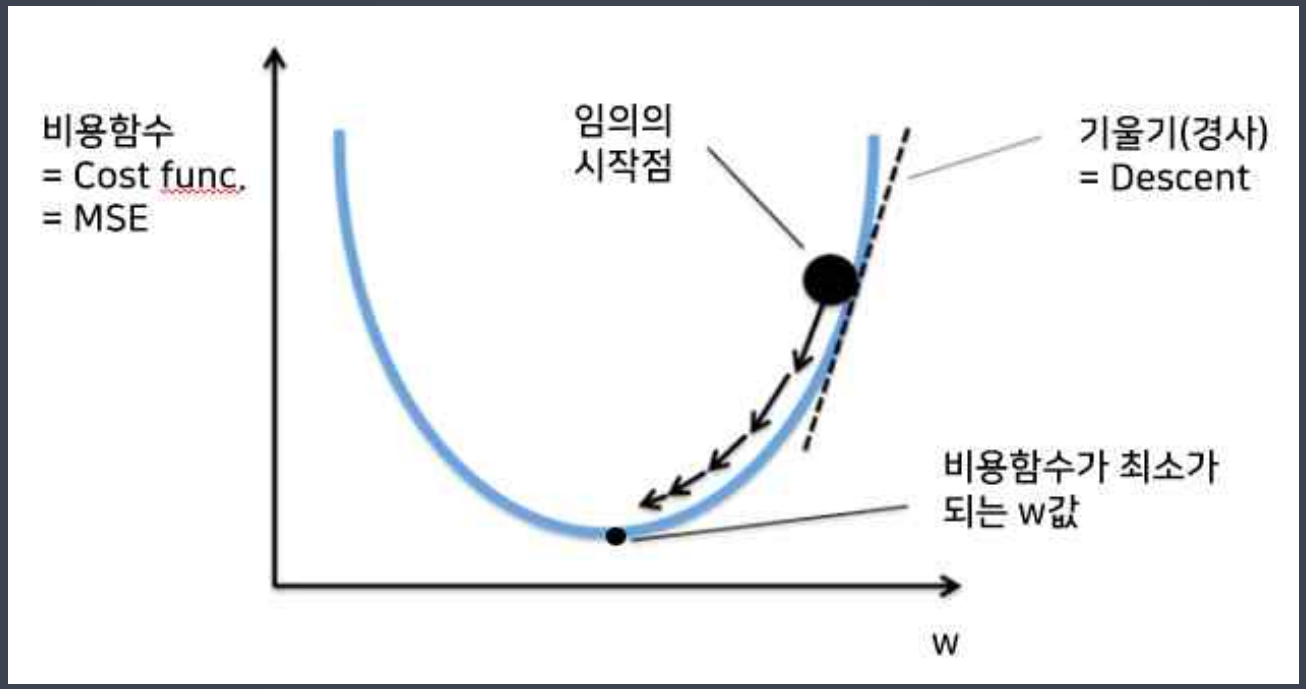


START

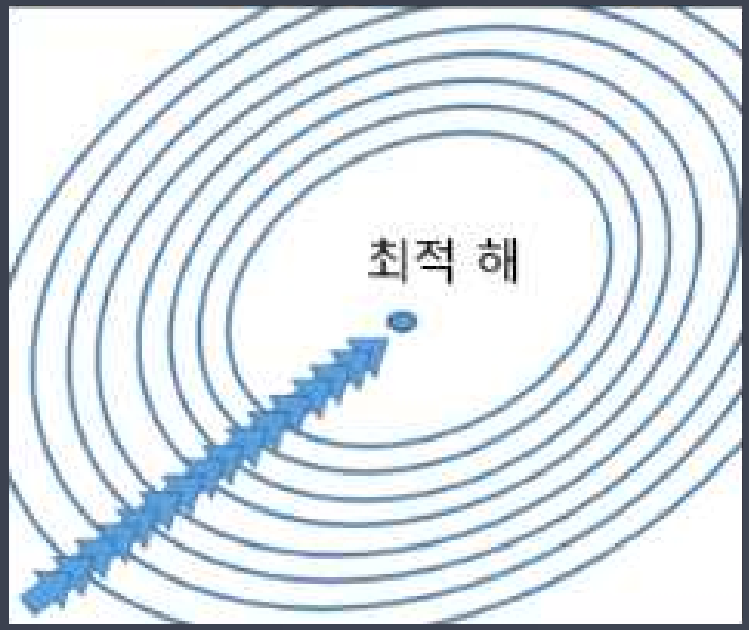
스마트인재개발원
Smart Human Resources Development

최적화함수(Optimizer)의 종류

경사하강법(Gradient Descent Algorithm)

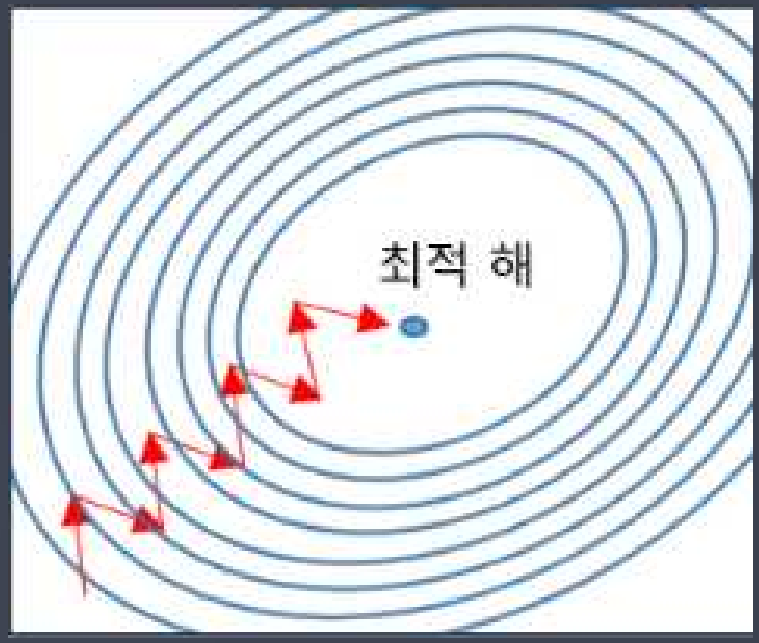


최적화함수(Optimizer)의 종류



경사하강법
(Gradient Descent)

전체 데이터를 이용해 업데이트



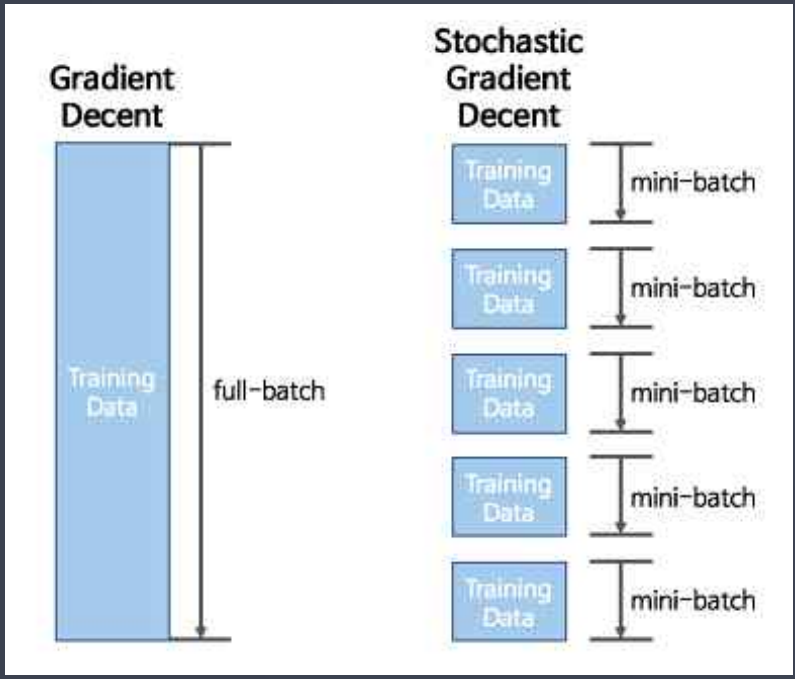
확률적경사하강법
(Stochastic Gradient Descent)

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

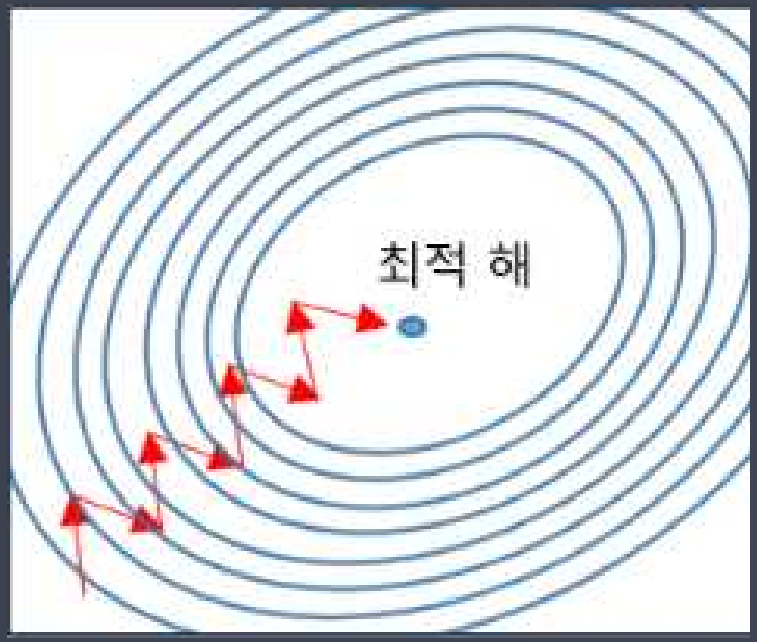


- batch_size를 줄임
  - 메모리 소모가 적음(저 사양일 경우), 학습 속도가 느림, 정확도↑

- batch_size를 높임
  - 메모리 소모가 큼, 학습 속도가 빠름, 정확도↓

→ 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)

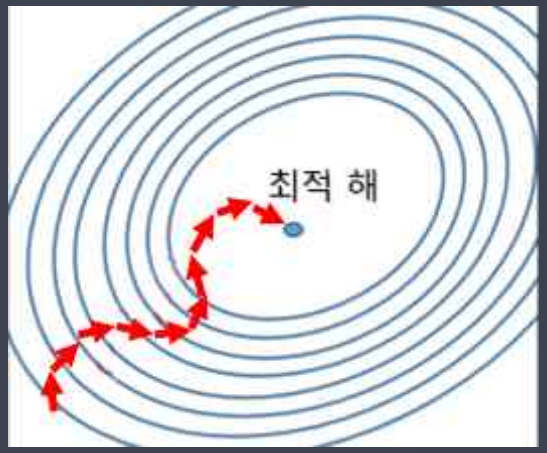
경사 하강법에 관성을 적용해 업데이트

현재 batch뿐만 아니라 이전 batch 데이터의 학습 결과도 반영

최적화함수(Optimizer)의 종류

특징 (Momentum)

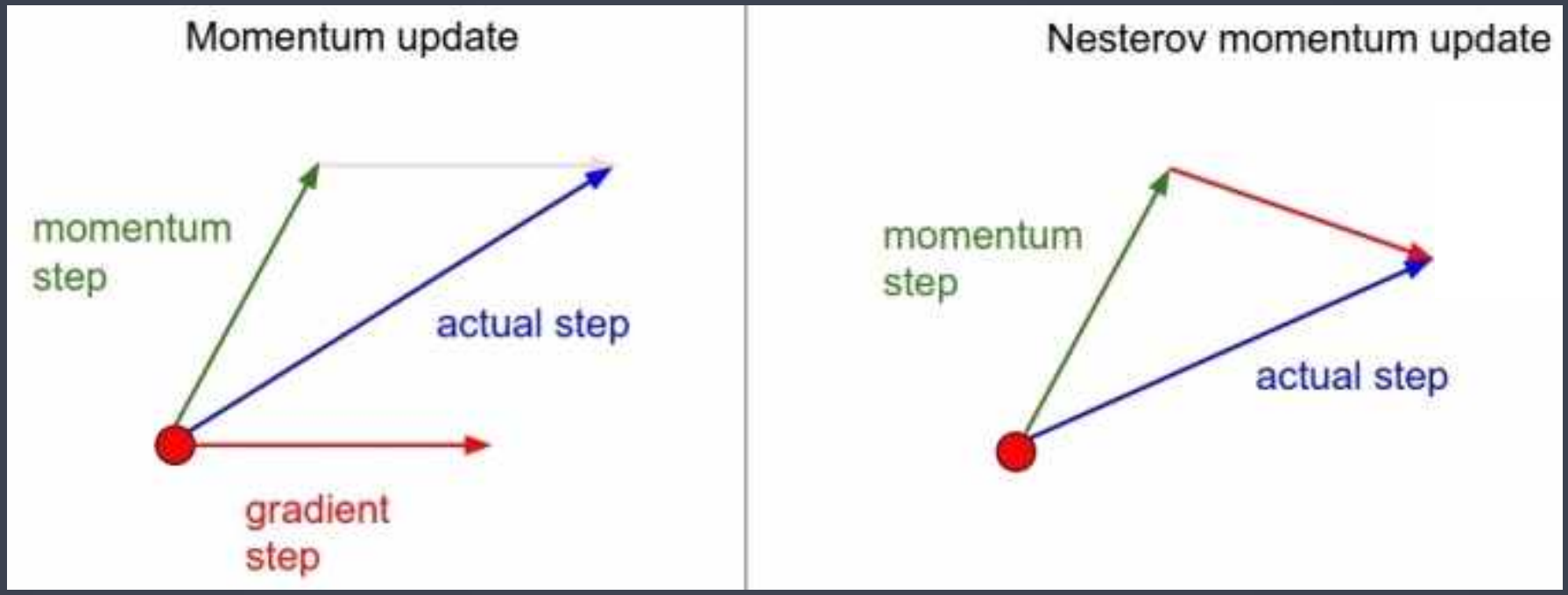
- 가중치를 수정하기 전 **이전 방향을 참고**하여 업데이트
- 지그재그 형태로 이동하는 현상이 줄어든다
- **α는 Learning Rate, m은 momentum 계수 (보통 0.9)**



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

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

최적화함수(Optimizer)의 종류



네스테로프 모멘텀

(Nesterov Accelrated Gradient)

개선된 모멘텀 방식

최적화함수(Optimizer)의 종류

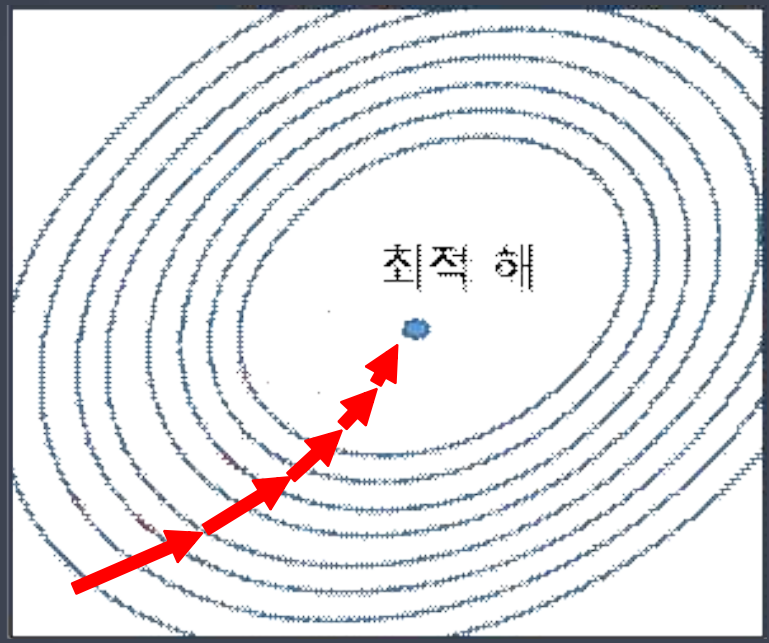
특징 (NAG)

- w, b값 업데이트 시 모멘텀 방식으로 먼저 더한 다음 계산
- 미리 해당 방향으로 이동한다고 가정하고 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

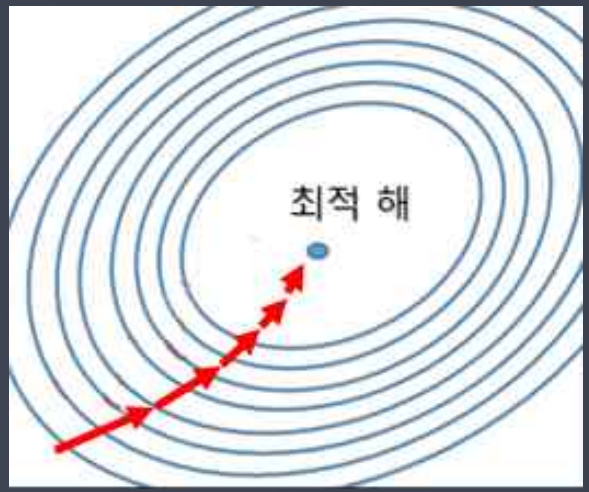
(Adaptive Gradient)

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

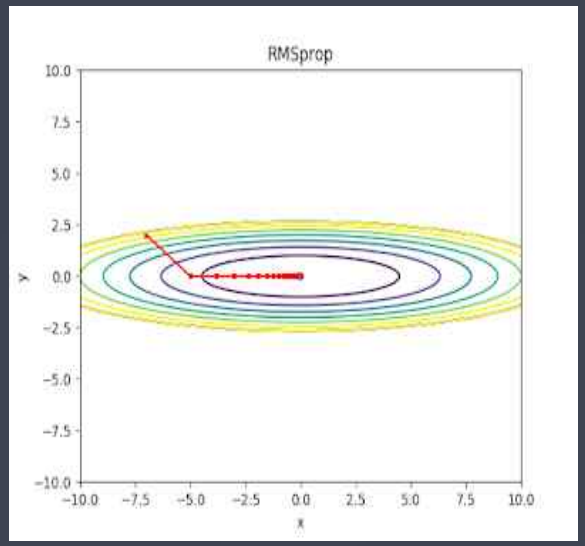


$$G(t) = G(t-1) + \left(\frac{\partial}{\partial w(t)} Cost(w(t))\right)^2$$

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

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

최적화함수(Optimizer)의 종류

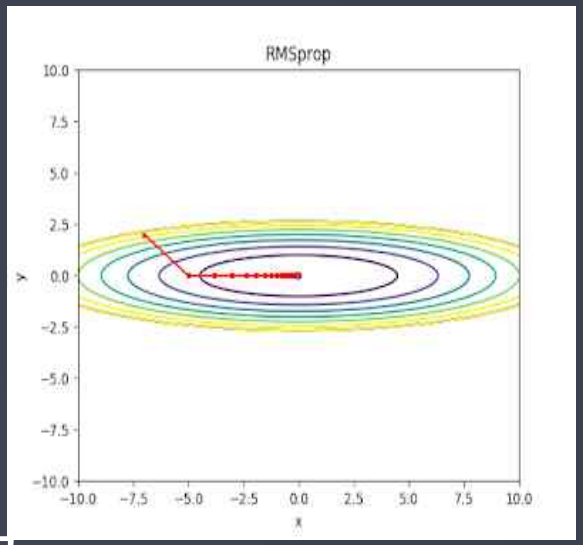


RMSProp

Adagrad의 단점을 해결한 최적화함수

최적화함수(Optimizer)의 종류

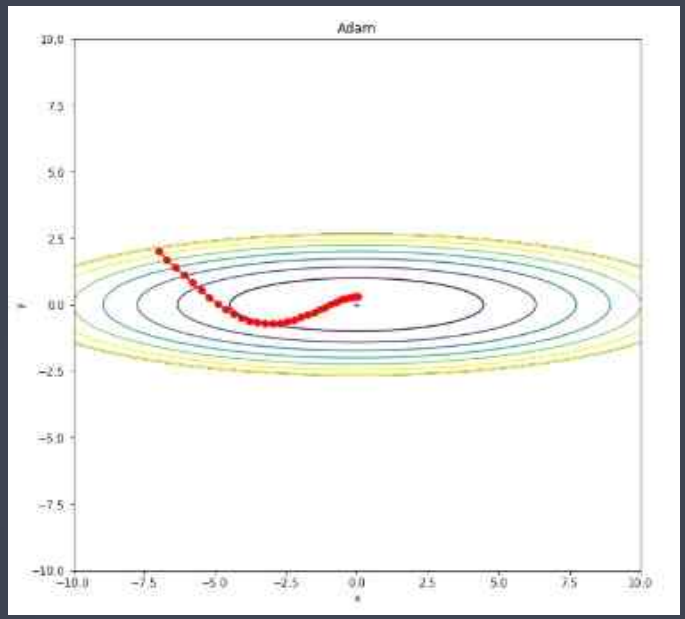
특징 (RMSProp)



- **Adagrad와 동일하게 학습을 진행하면서**
  학습률을 점차 줄여가는 방법
- **최소값을 찾기전 학습이 멈추는 Adagrad의 단점을**
  지수이동 평균을 도입해서 해결
- **지수 이동 평균 : 최근 학습한 수치의 영향력은 높이고**
  과거 학습한 수치의 영향력은 낮추는 방식.

$$h_i \leftarrow \rho h_{i-1} + (1-\rho)\frac{\partial L_i}{\partial W} \odot \frac{\partial L_i}{\partial W}$$

최적화함수(Optimizer)의 종류

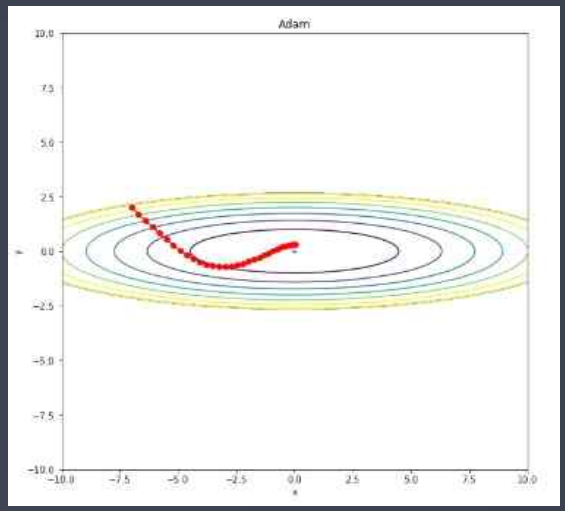


Adam

Momentum과 RMSProp의 장점만을
취한 최적화 함수

최적화함수(Optimizer)의 종류

특징 (Adam)



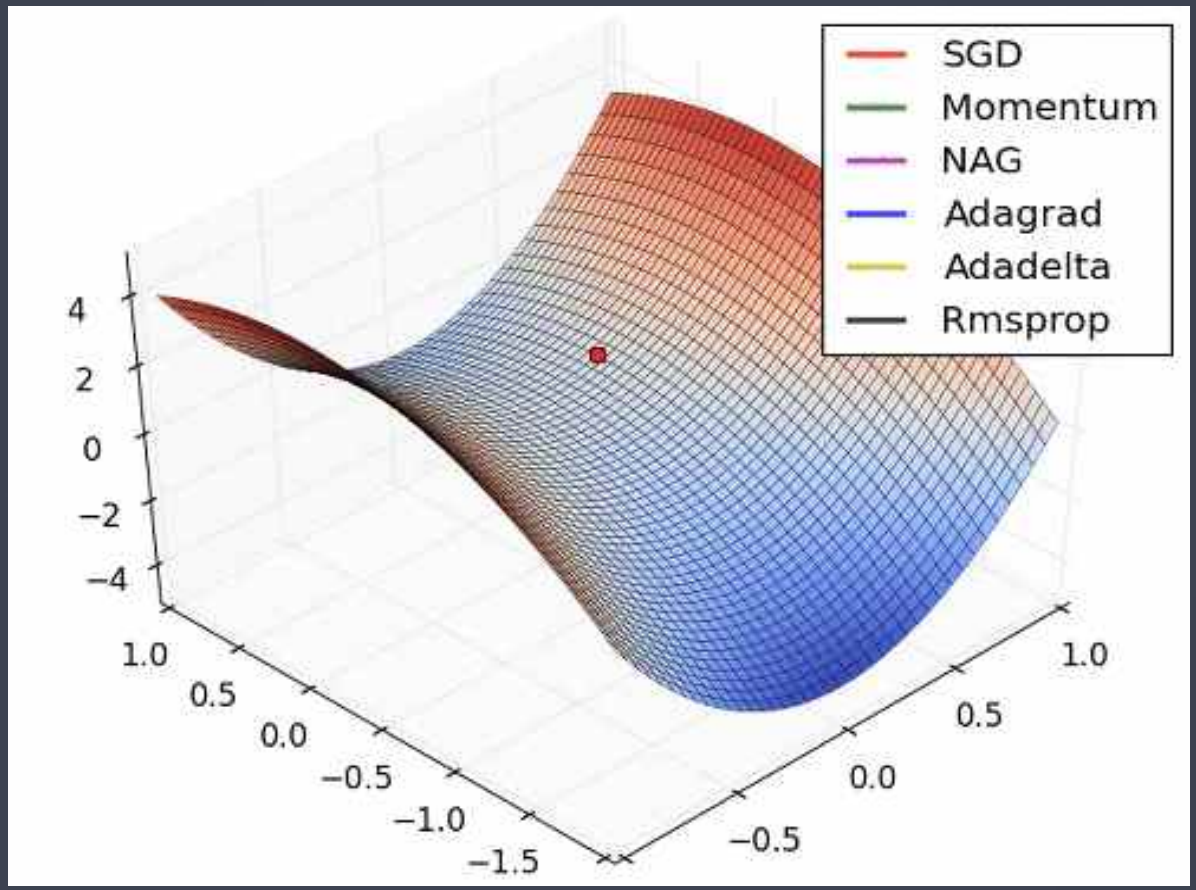
- 관성 방향으로 움직이는 **Momentum**과 보폭을 조절하며 움직이는 **RMSProp**의 장점을 하나로 합친 최적화 함수.
- 현재 보편적으로 사용하는 최적화 함수이며, 성능적인 측면에서 가장 나은 최적화 함수라 할 수 있다.

$$m_1 \leftarrow \beta_1 m_0 + (1-\beta_1) g_1$$

$$\widehat{m_1} \leftarrow \frac{m_1}{1-\beta_1^1} = \frac{\beta_1 m_0}{1-\beta_1^1} + \frac{(1-\beta_1) g_1}{1-\beta_1^1}$$

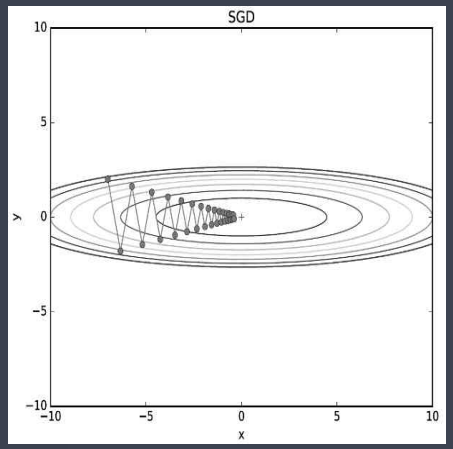
$$= 0 + g_1 (\because m_0 = 0)$$

움직임으로 보는 최적화 함수

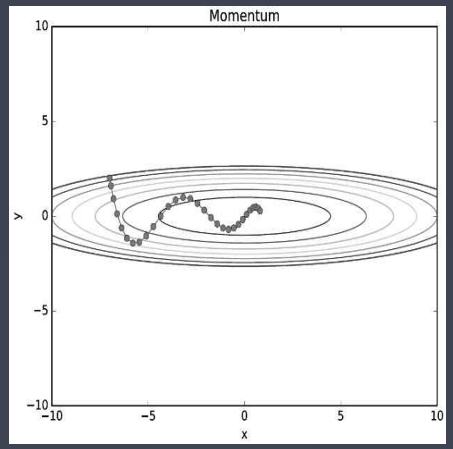


출처 : https://www.slideshare.net/yongho/ss-79607172

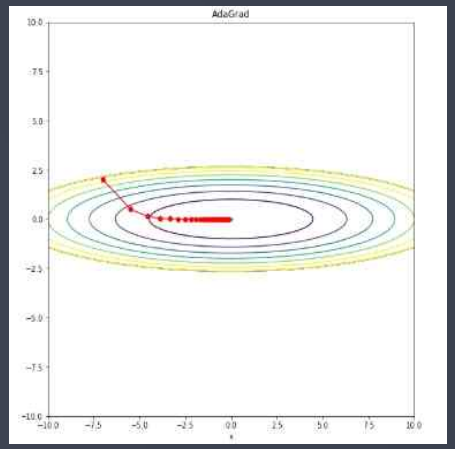
최적화함수(Optimizer) 선택론



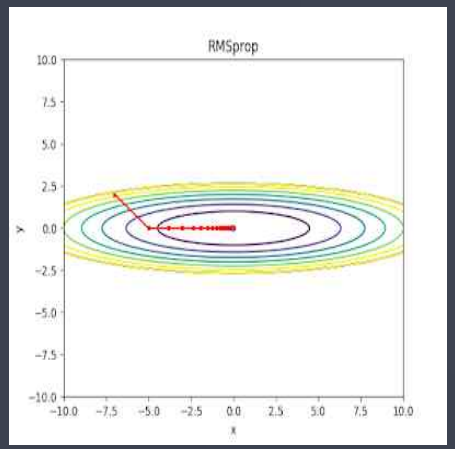
SGD



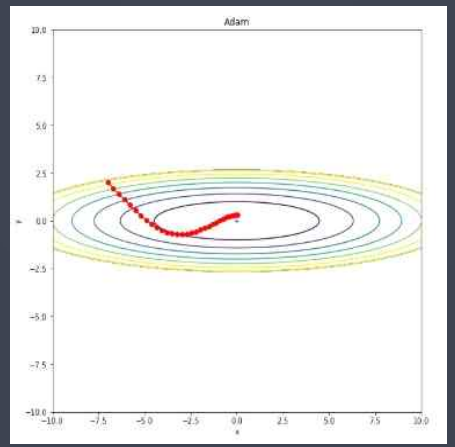
Momentum



Adagrad



RMSProp



Adam

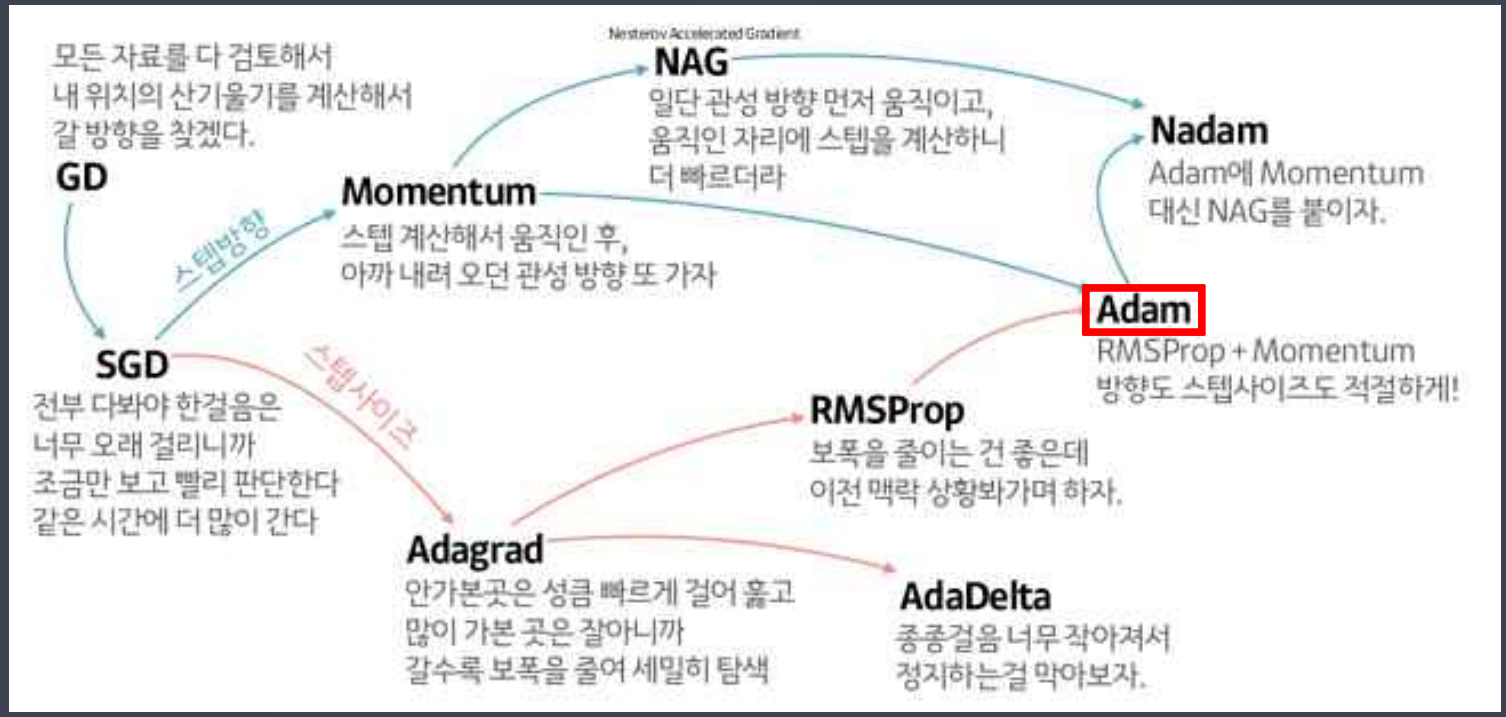
모든 문제에서 뛰어난 기법은 없다.
하지만 많은 사람들이 Adam에
만족하며 사용하는 것 같다.

But! 각자의 상황을 고려해보고,
여러가지를 시도하는 것이
가장 좋은 방법론이라고 말할 수 있다.

하지만 모르겠으면 Adam을 쓰는 것도 하나의 방법론!

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

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

활성화함수, 최적화함수를 바꿔가며
패션 이미지 데이터 분류 모델을 만들어
비교해보자