Chapter03.
Linear Regression

작성자 : 김진성

목차

1. Tensor 상수
2. 차원 축소 함수
3. Random
4. Linear Regression 개요
5. Tensorflow formula
6. Linear Regression model
7. 경사하강법 알고리즘

1. Tensor 상수 함수

tf.zeros(shape, dtype) : 모양과 타입으로 모든 원소가 0으로 초기화된 텐서 상수 생성

tf.ones(shape, dtype) : 모양과 타입으로 모든 원소가 1로 초기화된 텐서 상수 생성

tf.identity(input) : 내용과 모양이 동일한 텐서 상수 생성

tf.fill(dims, value) : 주어진 scalar값으로 초기화된 텐서 상수 생성

tf.constant(value, dtype, shape) : 지정한 값(value)으로 상수 텐서 상수 생성

tf.linspace(start, stop, num) : start~stop 범위에서 num수 만큼 텐서 상수 생성

tf.range(start, limit, delta) : 시작점, 종료점, 차이 이용 텐서 상수 생성

tf.tuple(tensor) : 여러 개의 tensor list로 묶기

2. 차원 축소 함수

tf.reduce_sum(input_tensor, reduction_indices) : 지정한 차원 대상 원소들 덧셈

tf.reduce_mean(input_tensor, reduction_indices) : 지정한 차원 대상 원소들 평균

tf.reduce_prod(input_tensor, reduction_indices) : 지정한 차원 대상 원소들 곱셈

tf.reduce_min(input_tensor, reduction_indices) : 지정한 차원 대상 최솟값 계산

tf.reduce_max(input_tensor, reduction_indices) : 지정한 차원 대상 최댓값 계산

tf.reduce_all(input_tensor) : tensor 원소가 전부 True -> True 반환

tf.reduce_any(input_tensro) : tensor 원소가 하나라도 True -> True 반환

3. random 함수

tf.random.normal(shape, mean, stddev) : 평균,표준편차 적용 정규분포 난수 생성

tf.truncated.normal(shape, mean, stddev) : 표준편차 2배 수보다 큰 값은 제거

tf.random.uniform(shape, minval, maxval) : 균등분포를 따르는 난수 생성

tf.random.shuffle(value) : 첫 번째 차원을 기준으로 텐서의 원소 섞기

tf.random.set_seed(seed) : 난수 seed값 설정

4. Linear Regression 개요

- 회귀분석(Regression Analysis)
  - 특정 변수(독립변수)가 다른 변수(종속변수)에 어떠한 영향을 미치는가 (인과관계 분석)
  - 예) 가격은 제품 만족도에 영향을 미치는가?
  - 한 변수의 값으로 다른 변수의 값 예언

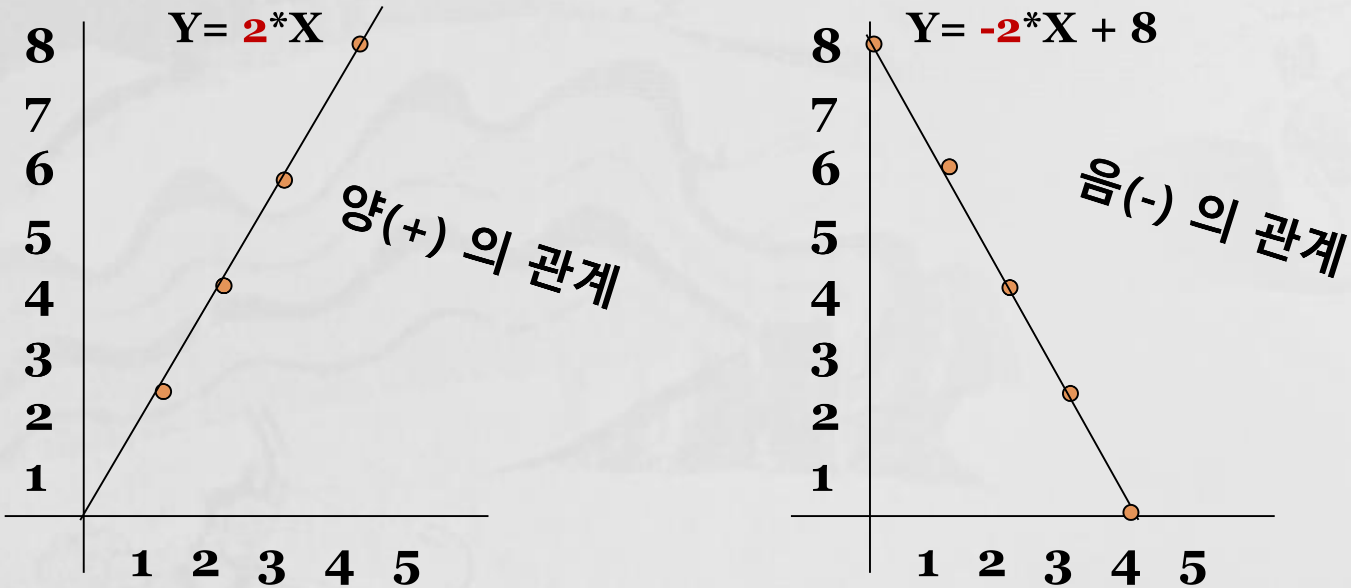
❖ 상관관계분석 : 변수 간의 관련성 분석

❖ 회귀분석 : 변수 간의 인과관계 분석

【회귀분석 중요사항】

- ‘통계분석의 꽃’ → 가장 강력하고, 많이 이용
- 종속변수에 영향을 미치는 변수를 규명(변수 선형 관계 분석)
- 독립변수와 종속변수의 관련성 강도
- 독립변수의 변화에 따른 종속변수 변화 예측
- **회귀 방정식**($Y=a+\beta X$ → Y:종속변수, a:상수, β:회귀계수, X:독립변수)을 도출하여 회귀선 추정
- 독립변수와 종속변수가 모두 <u>등간척도 또는 비율척도</u> 구성

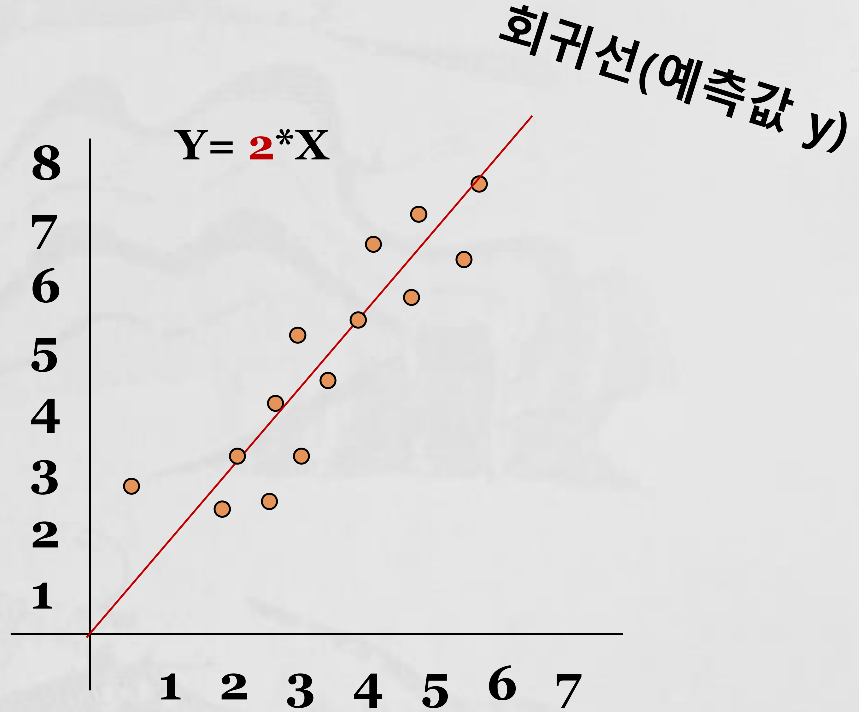
- 회귀 방정식 (1차 함수) -> 회귀선 추정
  - ✓ Y=a+βX : Y:종속변수, a:상수, β:회귀계수, X:독립변수
- 회귀계수(β) : 단위시간에 따라 변하는 양(기울기)이며, 회귀선을 추정함에 있어 최소자승법 이용



회귀방정식에 의해서 x가 10일 때 y는 20 예측 -> 회귀분석은 **수치 예측**

최소자승법 적용 회귀선

회귀방정식에 의해서 그려진 y의 추세선
산포도 각 점의 위치를 기준으로 정중앙 통과하는 회귀선 추정 방법



5. Tensorflow formula

- 회귀방정식 & 오차

```
X = tf.placeholder(tf.float32) # 입력
Y = tf.placeholder(tf.float32) # 정답
a = tf.Variable(0.5) # 기울기
b = tf.Variable(1.5) # 절편

model = x * a + b # 회귀방정식(Y 예측치)
err = tf.abs(model - Y) # model 오차
```

행렬곱(Matrix Multiply)

- tf.matmul() : 두 행렬의 곱 반환

X =

1	2
3	4

*

Y =

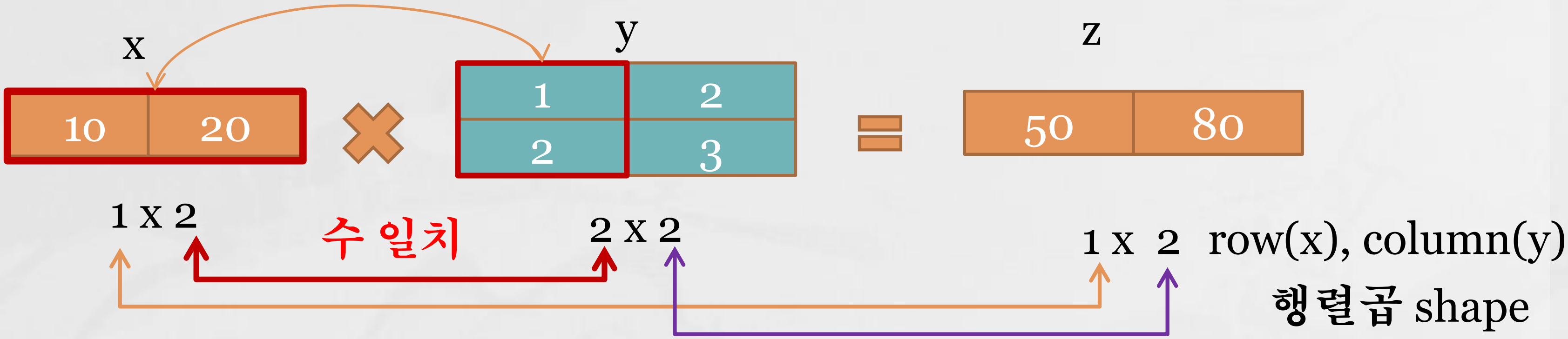
1	2
3	4

X * Y 행렬곱 =

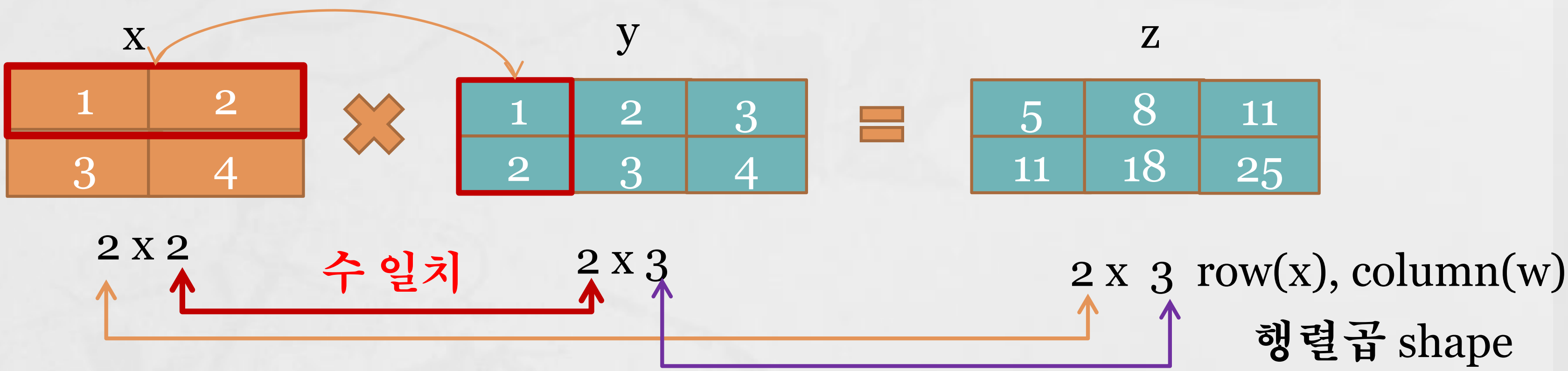
7	10
15	22

행렬곱 shape

1. x(1, 2) * y(2, 2) = z(1, 2) -> row(x), column(y)



2. x(2, 2) * y(2, 3) = z(2, 3) -> row(x), column(y)



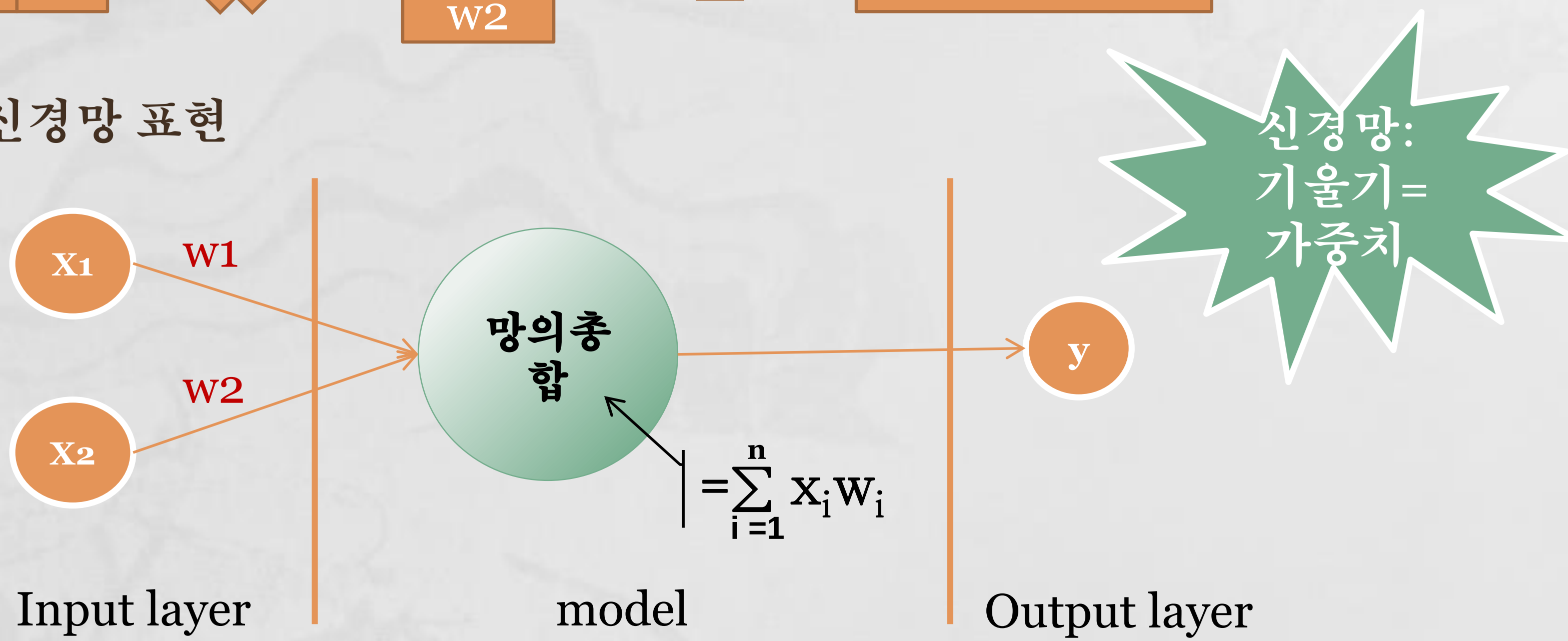
다중회귀방정식에서 행렬곱 함수 : 입력(X)와 기울기(a) 계산

- tf.matmul(X, w) : 두 텐서 행렬곱 연산(입력 : 2개, 기울기 : 2개)

X(1,2) w(2,1) model(1,1)

$$\begin{bmatrix} x1 & x2 \end{bmatrix} \times \begin{bmatrix} w1 \\ w2 \end{bmatrix} = \begin{bmatrix} X1*w1+X2*w2 \end{bmatrix}$$

- 신경망 표현



● 손실함수(loss function)

➢ 예측치와 실제값 사이의 차이(loss)를 계산하는 함수=MSE

```
Y = tf.placeholder(tf.float32) # Y 실제값

# 1. 회귀방정식

model = (X *w) + b # y 예측치(a:기울기, b : 절편)

# 2. 비용함수 = 차(실제값-예측치)의 제곱에 대한 평균값

loss = tf.reduce_mean(tf.square( model - Y )) # 평균제곱오차(MSE)

    #  MSE = 평균(실제값 -예측치)^2
```

제곱 적용
부호양수, Penalty 반영

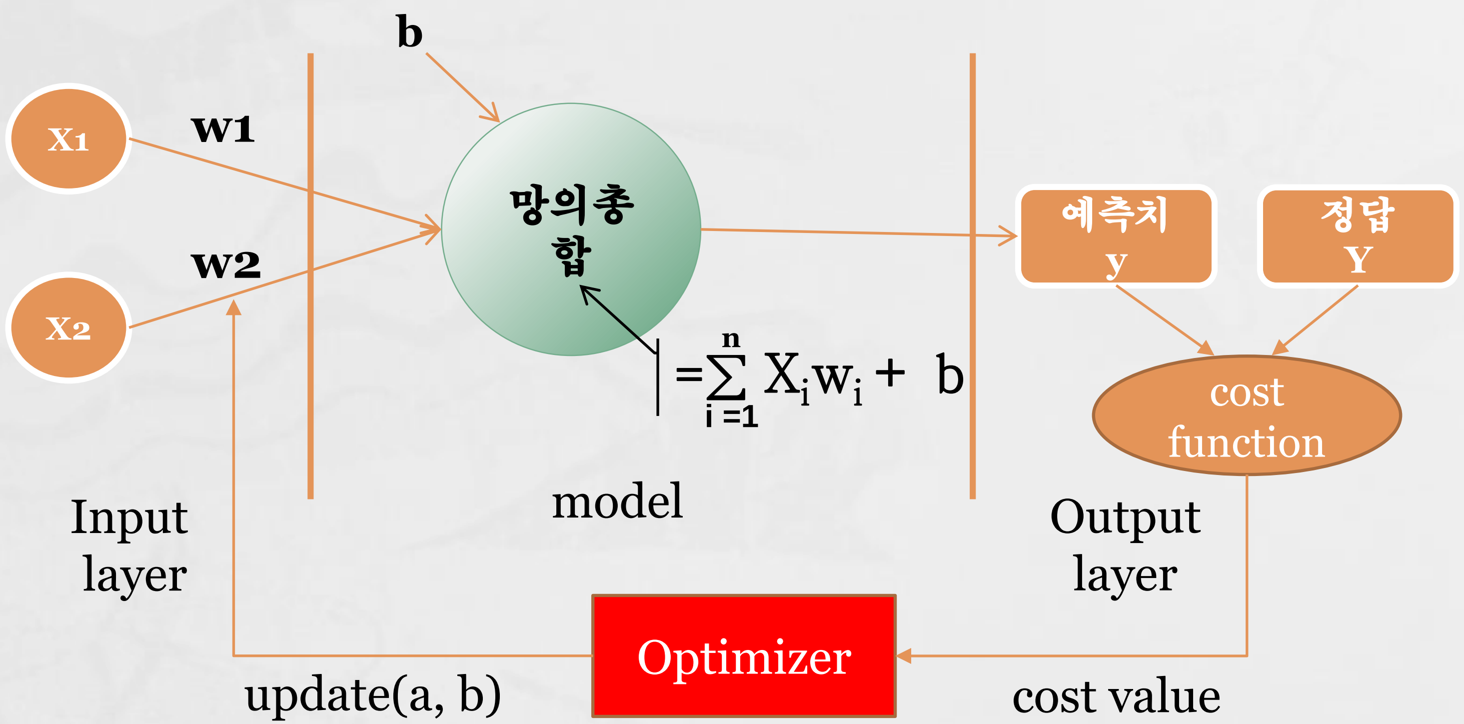
6. Linear Regression model

```
Y = tf.placeholder(tf.float32) # Y 실제값
# 1. 회귀방정식
model = (X * w) + b # Y 예측치(a:기울기, b : 절편)
# 2. 손실함수 = 차(실제값-예측치)의 제곱에 대한 평균값
loss = tf.reduce_mean(tf.square( model - Y )) # 평균제곱오차(MSE)
3. 경사하강법 : 비용함수 최소화 [최적의 a(기울기), b(절편) 수렴]
optimizer=tf.train.GradientDescentOptimizer(0.1) # 알고리즘 객체
train = optimizer.minimize(loss) # 학습에 의한 오차 최소화
```

❖ 경사하강법 : 학습 과정에서 최적의 기울기와 절편을 찾는다.

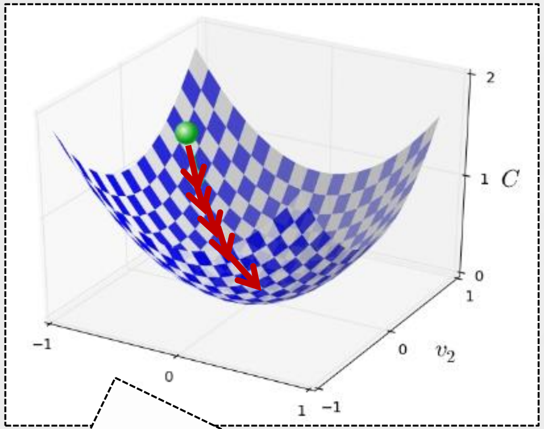
Linear Regression 신경망

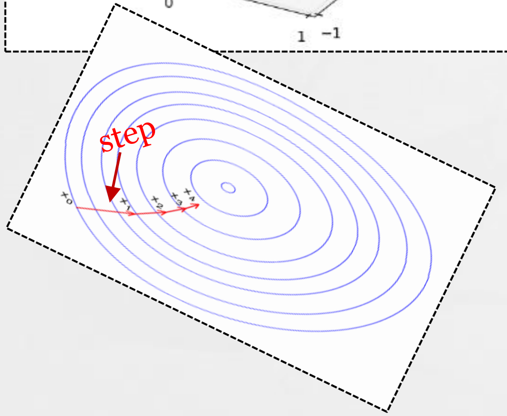
- ✓ Hidden layer 없음



7. 경사하강 알고리즘(GradientDesent algo)

➢ 비용함수를 최소화하는 최적화 알고리즘





model = w * X + b # y 예측치

loss = tf.reduce_mean(tf.square(model-Y))

optimizer=tf.train.GradientDescentOptimizer(0.5)

train = optimizer.minimize(loss)

➢ 예측치(회귀선)와 실제값 간의 <u>차이가 최소가 되도록 w와 b를 </u>찾는 알고리즘(w = *weight*, b = *bias*)

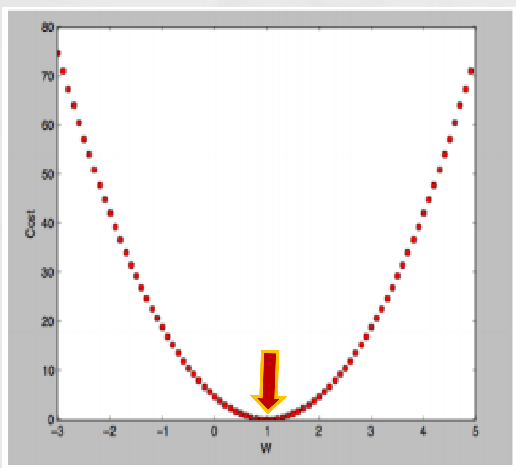
➢ 역전파 알고리즘의 표준(딥러닝 프레임워크 필수 라이브러리 포함)

➢ 경사를 따라서 step(learning rage) 단위 이동 w 조정으로 loss 최소화

● 경사하강법(Gradient Descent algorithm)

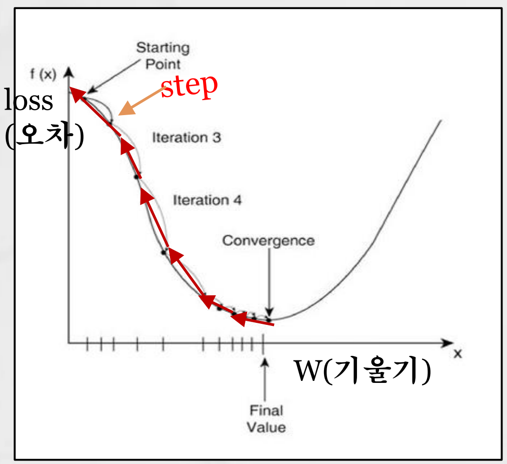
- ➢ 역전파 알고리즘의 표준(딥러닝 프레임워크 필수 라이브러리)
- ➢ 일정한 step 단위로 경사(기울기) 따라 하강하면서 w 조정 → loss 최소화
- ➢ 손실(loss) 최소화를 위해서 최적의 w(가중치)를 찾는 알고리즘
  - ✓ loss : 예측값와 실제값 간의 차이
  - ✓ loss를 w로 편미분 → 접선의 기울기(Gradient) 계산
  - ✓ 기울기(Gradient) → W 업데이트

w=1인 경우 loss=0(최소)



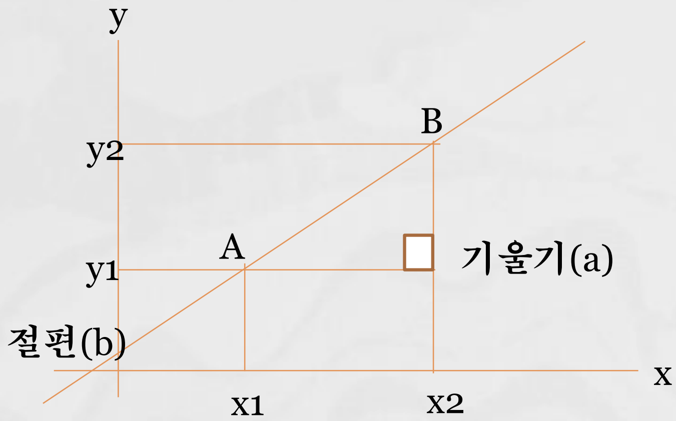
$y=\chi^2$

step 단위 이동 loss 최소화



U자 곡선 : 실제값
화살표 : 편미분 한 기울기
최소점 : 편미분 반대방향

좌표 평면에서 두 점 A(x1, y1)와 B(x2, y2)를 지나는 직선의 방정식 계산



1. 두 점 A와 B를 지나는 직선의 기울기(a)

$$a = \frac{y2 - y1}{x2 - x1}$$

(단, x1≠ x2)
x1=x2 : y축에 평행한 직선
(기울기 없음)

2. 직선의 방정식

$$(y-y1) = \frac{y2 - y1}{x2 - x1}(x-x1) \quad \text{식1}$$

=

$$(y-y2) = \frac{y2 - y1}{x2 - x1}(x-x2) \quad \text{식2}$$

문) 다음 두 점 A와 B를 지나는 직선의 방정식 계산하기

(4, 2), (5, 8)

$$a = \frac{8 - 2}{5 - 4} = 6$$

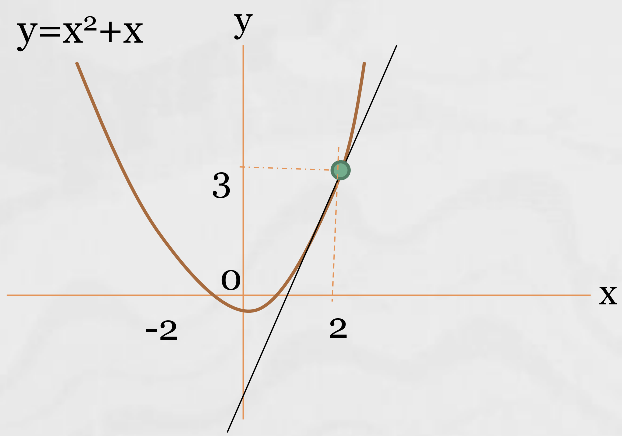
식1

y-2 = 6(x – 4)
y-2 = 6x – 24
y = 6x - 22

식2

y-8 = 6(x – 5)
y-8 = 6x – 30
y = 6x - 22

곡선 위의 한 점 A(x, y)에서 미분계수를 이용한 접선의 방정식 계산



<미분계수 연산>

Y = x -> 1 (1차항 x 소거 상수 1)

Y = 2x -> 2 (1차항 x소거)

Y = 3 -> 0 (0차항 상수 0)

Y = x^2 -> 2x (2차항 x앞으로 내려옴)

1. 한 점 A(2, 3)를 지나는 접선의 기울기

f(x) = x^2+ x # 함수 식

f'(x) = 2x + 1 # 미분계수

f'(2) = 2.2 + 1= 5 # 접선의 기울기

2. 접선의 방정식

(y-y1) = 접선의 기울기 (x-x1)

(y-3) = 5(x – 2)

y-3= 5x – 10

Y = 5x - 7

- **Cost를 W로 편미분하는 과정**
  - U자 곡선에서 접선의 기울기를 통해서 최소값 탐색 과정
  - 빨간점 위치 A, 파란점 위치 B 일때
  - A의 위치에서의 경사의 기울기는 음수(반비례) : W의 값 증가 시킴
  - B의 위치에서의 경사의 기울기는 양수(비례) : W의 값을 감소 시킴