AI 프로그래밍 13

융합학과 권오영

oykwon@koreatech.ac.kr

LINEAR REGRESSION : MACHINE LEARNING ALGORITHM

Regression Analysis(회귀분석) 이란?

❖ Regression Analysis (회귀분석)

- 관찰된 연속형 변수들에 대해 두 변수 사이의 모형을 구한 뒤 적합도를 특정해 내는 분석 방법

$$y = h(x_1, x_2, x_3, \ldots, x_k; \beta_1, \beta_2, \beta_3, \ldots, \beta_k) + \epsilon$$

❖ 기본 동작 원리

- 데이터들의 특성을 파악
- 경향성(Tendency) 및 의존성(Dependency)을 수식으로 작성
- 앞으로 발생할 일을 예측(Prediction)

❖ Regression Analysis 데이터 특징

- 분석을 통해 나온 예측값과 실제 데이터 오차는 모든 데이터값(독립변수)에 대하여 동일한 분산을 가지고 있음
- 데이터의 확률 분포는 정규분포를 이룸
- 독립변수 상호간에는 상관 관계가 없음 (선형적으로 독립)
- 독립변수와 종속변수 사이에는 상관 관계가 존재(선형관계)

❑ **상관관계** : 두 변수 a, b 가 있을 때 a값이 증가하거나 감소할때 b의 값도 a 값의 영향으로 증가하거나 감소하게 됨
❑ **독립변수** : 다른 변수에 영향을 받지 않는 변수
❑ **종속변수** : 독립변수에 영향을 받아서 변화하는 변수

예) 시험공부를 한 시간의 크기와 시험 결과의 상관관계를 분석
독립변수 : 시험공부를 한 시간
종속변수 : 시험의 결과

KOREATECH 한국기술교육대학교

Regression Analysis 데이터 모델 분류

❖ 데이터의 특성에 따른 분류

- Linear Regression Analysis Model(선형 회귀분석 모델)
- NonLinear Regression Analysis Model(비선형 회귀분석 모델)

❖ 독립변수 개수에 따른 분류

- Simple Regression Analysis Model(단순 회귀분석 모델) : 독립변수 1개
- Multiple Regression Analysis Model(다중 회귀분석 모델) : 독립변수 2개 이상

❖ 종속변수 개수에 따른 분류

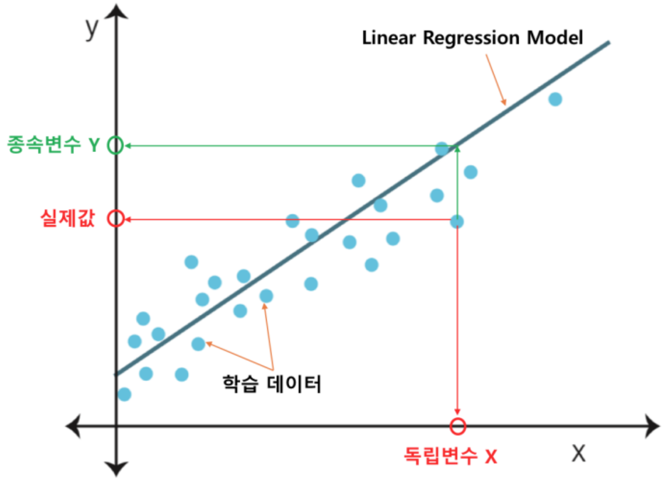
- Univariate Regression Analysis Model(단변량 회귀분석 모델) : 종속 변수 1개
- Multivariate Regression Analysis Model(다변량 회귀분석 모델) : 종속 변수 2개 이상

Single Variable Linear Regression

❖ Linear Regression Analysis 목표
종속변수 Y(결과값)와 독립변수 X(입력값)의 선형적 특성을 가지는 상관관계 모델을 생성하여 새로운 독립변수 X에 대한 결과를 예측

❖ Linear Regression Model

- 학습 데이터 : 그래프에 표시된 점
- 학습 데이터의 특성을 대표하는 모델 : 그래프의 직선
- 독립변수 X(입력값) : 학습데이터의 x 축 값
- 실제값 : 학습데이터의 y 축 값
- 종속변수 Y(결과값 or 예측값 or 가설값) : 학습 모델(직선) 수식에 대입한 독립변수 X의 값



- Linear Regression Model 수식
  - $Y = W \times X + b$
  - 가설 수식이라고도 함
  - W(Weight) 가중치라고 하며, b(bias)편향이라고 함

KOREATECH 한국기술교육대학교

Single Variable Linear Regression

- ❖ 최적화 함수 선언
  - 목적 : 비용함수의 수식이 최소가 되는 W(Weight), b(Bias) 의 값을 찾는 최적화 함수 선언
  - 최적화 알고리즘 경사하강법(Gradient descent) 사용
- ❖ 최적화 함수
  - 미분을 이용하여 스스로 최저 비용(오차)을 찾아가게 됨
  - 최적화 함수를 통하여 W, b의 변수를 변화시키게 됨
  - 오차가 최소가 되는 W, b 의 값을 찾아내는 과정을 통하여 최소 비용(오차)를 가지는 모델을 만듦
- ❖ Gradient descent 알고리즘의 θ의 변화식

$$\theta = \theta - \eta \nabla_\theta J(\theta)$$

θ : Weight, bias 변수
$J(\theta)$: 최적화 시킬 함수(비용 함수)
$\nabla_\theta J(\theta)$: 최적화 시킬 함수의 기울기

  - 최적화 시킬 함수(비용 함수)의 최소값은 함수의 기울기가 최소가 되는 부분
  - 기울기가 0에 가까워 지는 θ의 값을 찾게 됨 (1차 미분계수를 이용해 함수의 최소값을 찾아가는 iterative한 방법을 사용)

KOREATECH 한국기술교육대학교

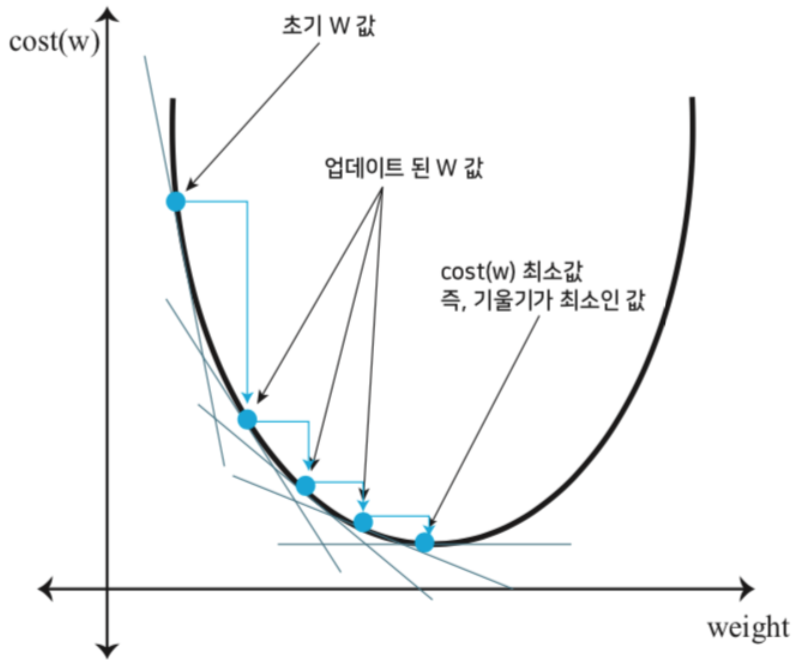
Single Variable Linear Regression

❖ Linear Regression에서 Gradient descent 알고리즘

- 최적화 시킬 비용함수 : 가설 함수의 최소 비용(오차) 함수

$$H(x) = \mathrm{W} * x + \mathrm{b}$$

$$cost(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(H_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$



- ϴ 변화식에 최적화 시킬 함수를 대입하여 정리하여 경사를 구하는 수식으로 정리
(ϴ 는 W, b 변수이지만 식을 간략하게 표현하기 위하여 영향이 적은 b 는 생략함)

$$\mathrm{W} := \mathrm{W} - \alpha\frac{\partial}{\partial \mathrm{w}}cost(w)$$

α : 학습률

$$\mathrm{W} := \mathrm{W} - \alpha\frac{1}{m}\left(Wx^{(i)} - y^{(i)}\right)x^{(i)}$$

- 계산된 ϴ(W, b 변수)의 값을 업데이트 하여 최소 비용(오차)을 구하는 과정을 '머신러닝 모델학습' 이라고 함

Multi Variable Linear Regression

❖ Multi Variable Linear Regression

- 목표 : 독립변수 2개, 종속 변수 1개를 가지는 Linear Regression 모델 학습
- 직접 생성한 학습데이터를 이용하여 모델 학습

```
hypothesis = W1 * X1 + W2 * X2 + b
```

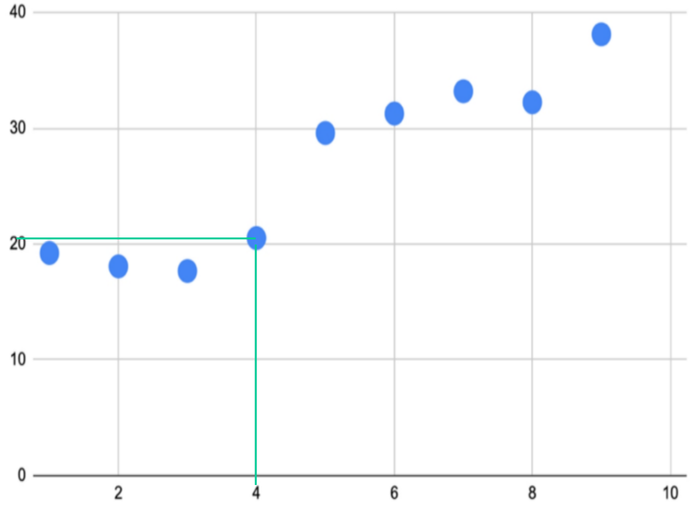
LINEAR REGRESSION

- ❖ 가장 기본적인 알고리즘. 여러 개의 독립변수와 한 개의 종속변수 간의 상관관계를 모델링하는 기법
- ❖ A straight line to describe linear relationships
- ❖ Simple linear regression
  - With one independent variable
- ❖ Multiple regression
  - with multiple independent variables.

독립변수 → Season (x)

종속변수 → Viewers (y)

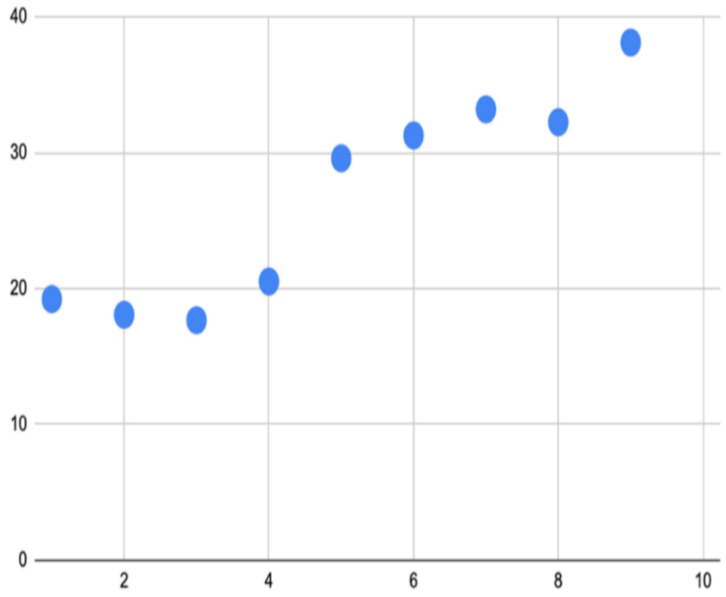
Season (x)	Viewers (y)
1	19.22
2	18.07
3	17.67
4	20.52
5	29.59
6	31.27
7	33.19
8	32.24
9	38.11

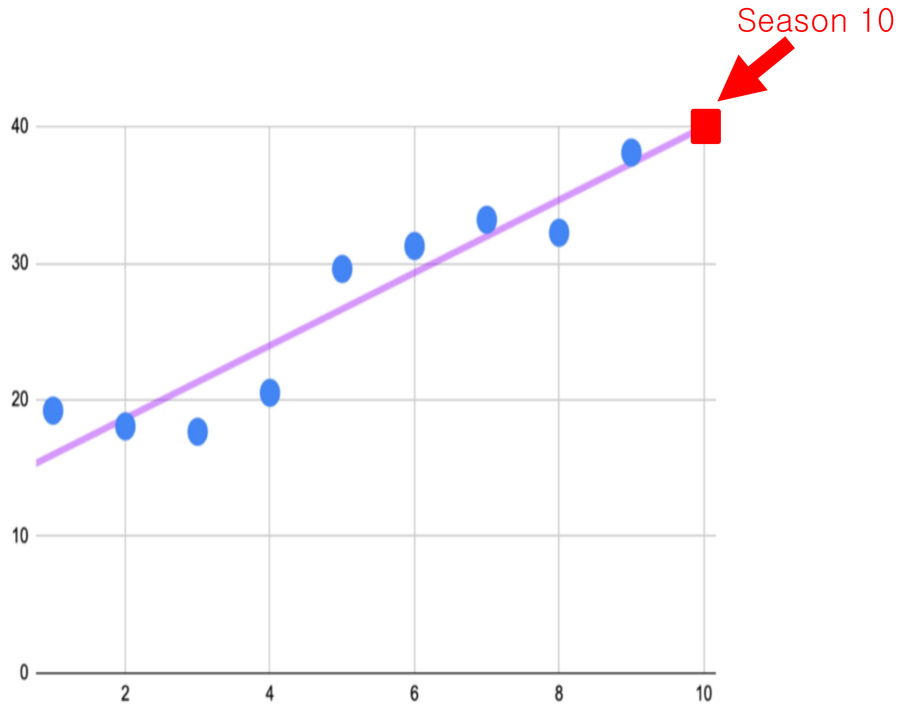


KOREATECH 한국기술교육대학교

LINEAR REGRESSION 예측

Season (x)	Viewers (y)
1	19.22
2	18.07
3	17.67
4	20.52
5	29.59
6	31.27
7	33.19
8	32.24
9	38.11
10	??

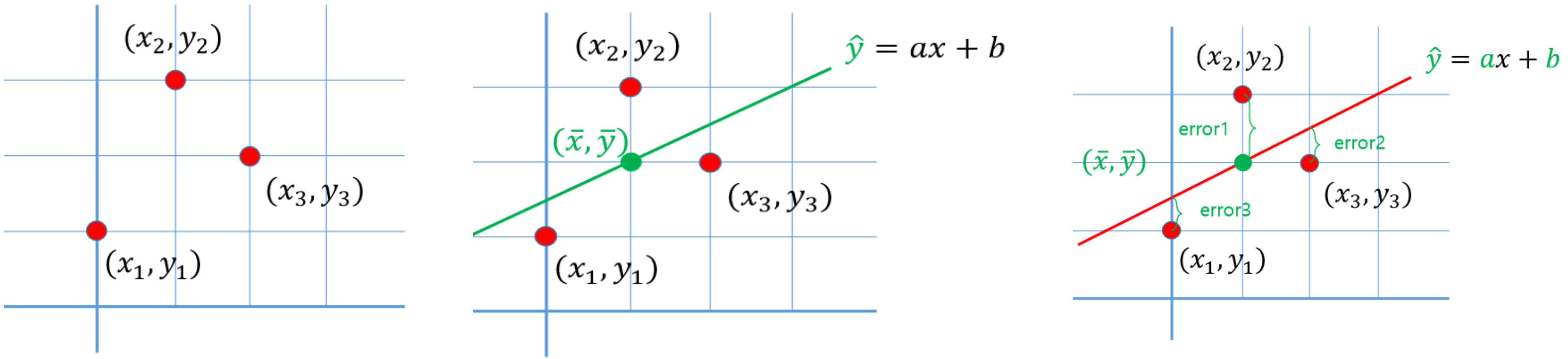




LINEAR REGRESSION

❖ 데이터들과 **오차가 가장 적은** 회귀선 y = ax + b 생성

- 주어진 x, y 값을 이용해서 a, b 를 구함.
  - ✓ 임의의 직선을 그어 이에 대한 평균 제곱 오차를 구하고, 이 값을 가장 작게 만들어 주는 a와 b 값을 구함. (회귀직선은 표본평균을 지난다)
  - ✓ 주어진 데이터의 평균을 지나는 직선과의 y(종속변수) 값 차이(error)가 전체적으로 최소가 되도록 하는 직선을 구한다. 즉, a, b를 구한다. => 최적화 함수(최적화 알고리즘 적용)
- 학습오차 계산으로 추정의 정확성을 파악할 수 있음

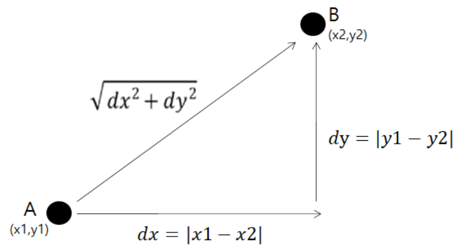


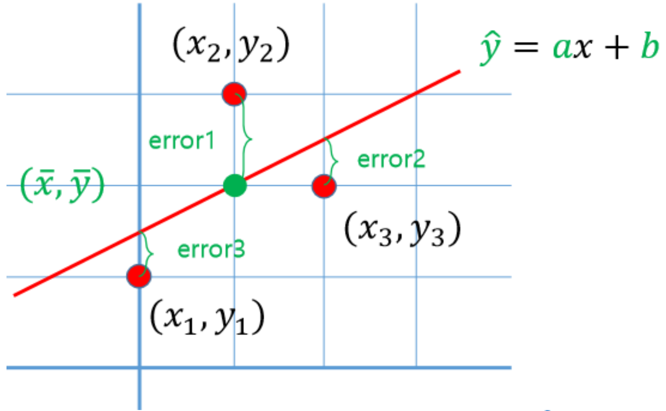
KOREA TECH 한국기술교육대학교

오차 계산

❖ 오차에 대한 수학적 정의

- 평균 절대값 오차 MAE(Mean Absolute Error) $MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - h_i|$
- 평균 제곱 오차 MSE(Mean Square Error) $MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - h_i)^2$
- 평균 제곱근 오차 RMSE(Root Mean Square Error) $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - h_i)^2}$





$$오차의\ 합 = \sum_{i}^{n} (\hat{y}_i - y_i)^2 \quad 평균\ 제곱\ 오차(MSE) = \frac{1}{n}\sum (\hat{y}_i - y_i)^2$$

$$E = \frac{1}{2}\sum_{i}^{3} (y_i - \hat{y}_i)^2$$

$$a = \frac{(\Sigma y)(\Sigma x^2) - (\Sigma x)(\Sigma xy)}{n(\Sigma x^2) - (\Sigma x)^2}$$

$$b = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{n(\Sigma x^2) - (\Sigma x)^2}$$

$$E = \frac{1}{2}((ax_1 + b - y_1)^2 + (ax_2 + b - y_2)^2 + (ax_3 + b - y_3)^2)$$

KOREATECH 한국기술교육대학교

Simple linear regression

❖ 데이터들과 오차가 가장 적은 회귀선 y = ax + b 생성

- 주어진 x, y 값을 이용해서 a, b 를 구함.

	(X)	(Y)	XY	X²
1	1	3	3	1
2	2	4	8	4
3	1	2	2	1
4	4	7	28	16
5	3	5	15	9
Σ (Total)	**11**	**21**	**56**	**31**

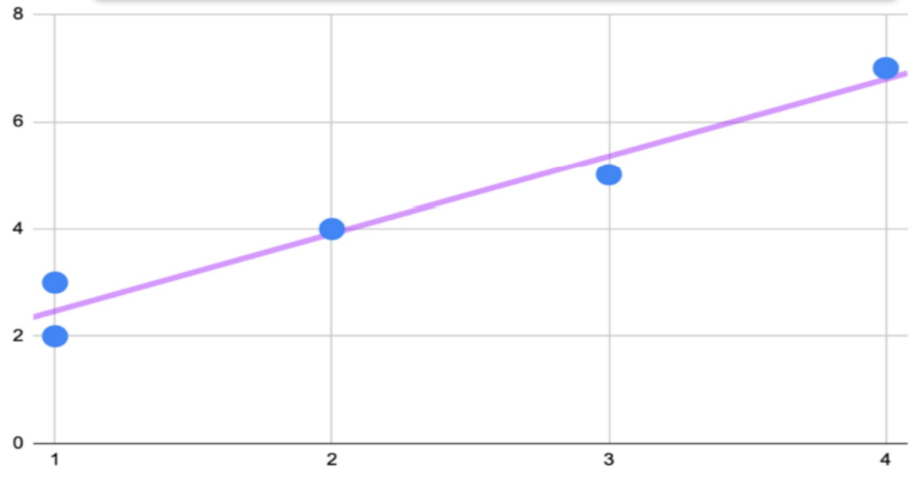
$$a = \frac{(\Sigma y)(\Sigma x^2) - (\Sigma x)(\Sigma xy)}{n(\Sigma x^2) - (\Sigma x)^2}$$

$$b = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{n(\Sigma x^2) - (\Sigma x)^2}$$

a = ((21 x 31) – (11 x 56)) / (5(31) – 11^2)
(651 – 616) / (155 – 121)
35 / 34 = 1.029

b = (5(56) – (11 x 21)) / (5(31) – 11^2)
(280 – 231) / (155 – 121)
49 / 34 = 1.441

$$y = 1.441x + 1.029$$



y = 1.441x + 1.029 plotted on the scatterplot

Σ = Total sum
Σx = Total sum of all x values (1 + 2 + 1 + 4 + 3 = 11)
Σy = Total sum of all y values (3 + 4 + 2 + 7 + 5 = 21)
Σxy = Total sum of x*y for each row (3 + 8 + 2 + 28 + 15 = 56)
Σx^2 = Total sum of x*x for each row (1 + 4 + 1 + 16 + 9 = 31)
n = Total number of rows. In the case of this example, n is equal to 5.

간단한 기계 학습의 예

❖ 선형 회귀 문제

- [그림 1-4]: 식 (1.2)의 직선 모델을 사용하므로 두 개의 매개변수 $\Theta = (w, b)^{\mathrm{T}}$

$$y = wx + b \tag{1.2}$$

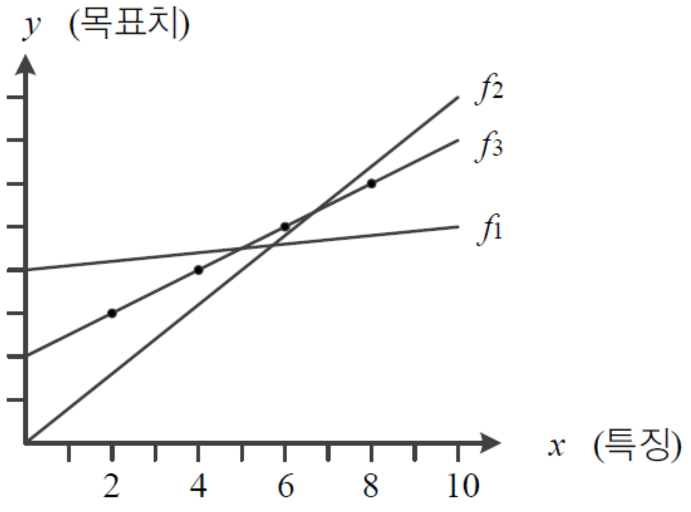


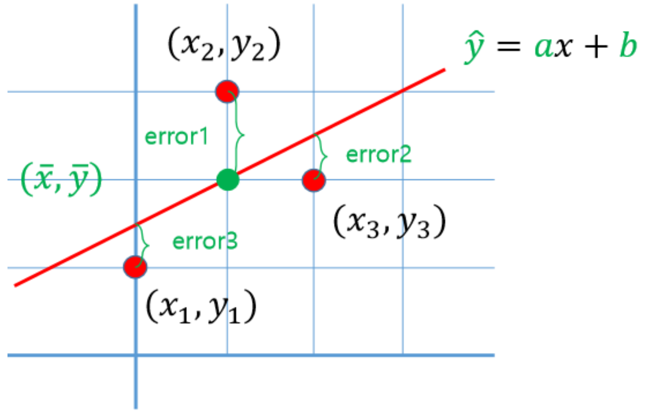
그림 1-4 간단한 기계 학습 예제

간단한 기계 학습의 예

❖ 목적 함수objective function (또는 비용 함수cost function)

- 식 (1.8)은 선형 회귀를 위한 목적 함수
  - ✓ $f_\Theta(\mathbf{x}_i)$는 예측함수의 출력, y_i는 예측함수가 맞추어야 하는 목표값이므로 $f_\Theta(\mathbf{x}_i)-y_i$는 오차
  - ✓ 평균제곱오차MSE(mean squared error)를 사용

$$J(\Theta) = \frac{1}{n}\sum_{i=1}^{n}(f_\Theta(\mathbf{x}_i) - y_i)^2$$



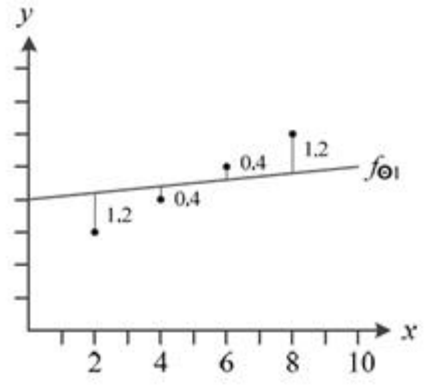
- 처음에는 최적 매개변수 값을 알 수 없으므로 난수로 $\Theta_1 = (w_1, b_1)^\mathrm{T}$ 설정 → $\Theta_2 = (w_2, b_2)^\mathrm{T}$로 개선 → $\Theta_3 = (w_3, b_3)^\mathrm{T}$로 개선 → Θ_3는 최적해 $\hat{\Theta}$
  - ✓ 이때 $J(\Theta_1) > J(\Theta_2) > J(\Theta_3)$

간단한 기계 학습의 예

✓ 훈련집합

$\mathbb{X} = \{x_1 = (2.0), x_2 = (4.0),\ x_3 = (6.0),\ x_4 = (8.0)\},$

$\mathbb{Y} = \{y_1 = 3.0,\ y_2 = 4.0,\ y_3 = 5.0,\ y_4 = 6.0\}$

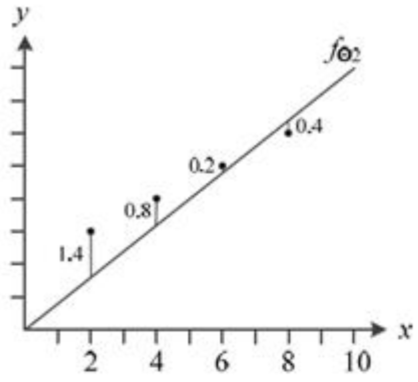


(a) 초기 매개변수 Θ_1

✓ 초기 직선의 매개변수 $\Theta_1 = (0.1, 4.0)^{\mathrm{T}}$라 가정

$$\mathbf{x}_1, \mathrm{y}_1 \rightarrow \left(f_{\Theta_1}(2.0) - 3.0\right)^2 = \left((0.1 * 2.0 + 4.0) - 3.0\right)^2 = 1.44$$
$$\mathbf{x}_2, \mathrm{y}_2 \rightarrow \left(f_{\Theta_1}(4.0) - 4.0\right)^2 = \left((0.1 * 4.0 + 4.0) - 4.0\right)^2 = 0.16$$
$$\mathbf{x}_3, \mathrm{y}_3 \rightarrow \left(f_{\Theta_1}(6.0) - 5.0\right)^2 = \left((0.1 * 6.0 + 4.0) - 5.0\right)^2 = 0.16$$
$$\mathbf{x}_4, \mathrm{y}_4 \rightarrow \left(f_{\Theta_1}(8.0) - 6.0\right)^2 = \left((0.1 * 8.0 + 4.0) - 6.0\right)^2 = 1.44$$

$\longrightarrow$ $J(\Theta_1) = 0.8$

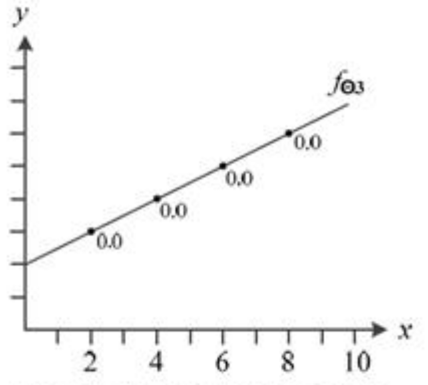


(b) Θ_1을 개선하여 Θ_2가 됨

✓ Θ_1을 개선하여 $\Theta_2 = (0.8, 0.0)^{\mathrm{T}}$가 되었다고 가정

$$\mathbf{x}_1, \mathrm{y}_1 \rightarrow \left(f_{\Theta_2}(2.0) - 3.0\right)^2 = \left((0.8 * 2.0 + 0.0) - 3.0\right)^2 = 1.96$$
$$\mathbf{x}_2, \mathrm{y}_2 \rightarrow \left(f_{\Theta_2}(4.0) - 4.0\right)^2 = \left((0.8 * 4.0 + 0.0) - 4.0\right)^2 = 0.64$$
$$\mathbf{x}_3, \mathrm{y}_3 \rightarrow \left(f_{\Theta_2}(6.0) - 5.0\right)^2 = \left((0.8 * 6.0 + 0.0) - 5.0\right)^2 = 0.04$$
$$\mathbf{x}_4, \mathrm{y}_4 \rightarrow \left(f_{\Theta_2}(8.0) - 6.0\right)^2 = \left((0.8 * 8.0 + 0.0) - 6.0\right)^2 = 0.16$$

$\longrightarrow$ $J(\Theta_2) = 0.7$



(c) Θ_2를 개선하여 최적의 Θ_3을 찾음

✓ Θ_2를 개선하여 $\Theta_3 = (0.5, 2.0)^{\mathrm{T}}$가 되었다고 가정

✓ 이때 $J(\Theta_3) = 0.0$이 되어 Θ_3은 최적값 $\hat{\Theta}$ 이 됨

KOREATECH 한국기술교육대학교

간단한 기계 학습의 예

❖ 기계 학습이 할 일을 공식화하면,

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} J(\Theta) \tag{1.9}$$

- 기계 학습은 작은 개선을 반복하여 최적해를 찾아가는 수치적 방법으로 식 (1.9)를 풀어냄

❖ 알고리즘 형식으로 쓰면,

알고리즘 1-1 기계 학습 알고리즘

입력 : 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$
출력 : 최적의 매개변수 $\hat{\Theta}$

```
난수를 생성하여 초기 해 Θ_1을 설정한다.
t=1
while (J(Θ_t)가 0.0에 충분히 가깝지 않음)      // 수렴 여부 검사
    J(Θ_t)가 작아지는 방향 ΔΘ_t를 구한다.      // ΔΘ_t는 주로 미분을 사용하여 구함
    Θ_{t+1} = Θ_t + ΔΘ_t
    t=t+1
Θ̂ = Θ_t
```

간단한 기계 학습의 예

❖ 좀더 현실적인 상황

- 지금까지는 데이터가 선형을 이루는 아주 단순한 상황을 고려함
- 실제 세계는 선형이 아니며 잡음이 섞임 → 비선형 모델이 필요

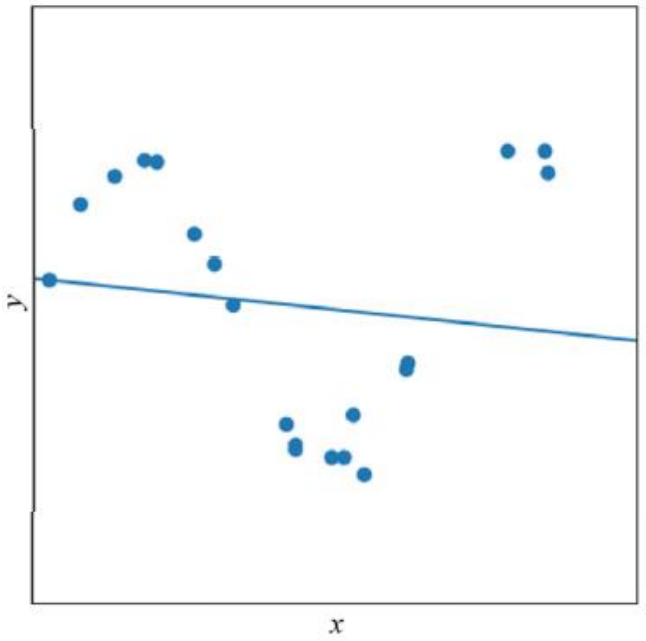


그림 1-12 선형 모델의 한계

실습 : 평균 제곱 오차 MSE(Mean Square Error)

❖ 여러 개의 입력 값을 계산할 때는 임의의 선을 그려, 이 선이 얼마나 잘 그려졌는지를 평가하여 조금씩 수정해 가는 방법을 사용함 =>선의 오차를 평가하는 알고리즘 필요.

▪ 선형회귀: 임의의 직선을 그어 이에 대한 평균 제곱 오차를 구하고, 이 값을 가장 작게 만들어 주는 a와 b 값을 찾아가는 작업. https://raw.githubusercontent.com/angeloyeo/angeloyeo.github.io/master/pics/2020-08-24-linear_regression/pic13.mp4

❖ 적용 : 평균 제곱 오차를 파이썬으로 구현

- 가상의 기울기 a와 y 절편 b 임의로 정함.

```
fake_a_b = [3, 76]
```

- 리스트(변수명: data)에 공부한 [시간]과 [성적] 간의 관계를 지정(저장)
- X 리스트에 [시간]을, y 리스트에 [성적]을 저장.

```
data = [[2, 81], [4, 93], [6, 91], [8, 97]]
x = [i[0] for i in data]
y = [i[1] for i in data]
```

- ** 패키지 임포트[import numpy as np] : 리스트를 배열로 만들어주는 array 메소드 제공

실습 : 평균 제곱 오차

- predict( )라는 함수로 일차 방정식 y = ax + b를 구현.

```
def predict(x):
    return fake_a_b[0]*x + fake_a_b[1]
```

- 평균 제곱근 공식을 그대로 파이썬 함수로 만들기

$$\frac{1}{n}\sum(\hat{y}_i - y_i)^2$$

```
def mse(y_hat, y):
    return ((y_hat-y) ** 2).mean()
```

- mse( ) 함수에 데이터를 대입하여 최종값 계산.

```
def mse_val(predict_result, y):
    return mse(np.array(predict_result), np.array(y))
```

KOREATECH 한국기술교육대학교

실습 : 평균 제곱 오차

- 이제 모든 x 값을 predict( ) 함수에 대입하여 예측 값을 구함
- 이 예측 값과 실제 값을 통해 최종값을 출력.

```
# 예측 값이 들어갈 빈 리스트
predict_result = []

# 모든 x 값을 한 번씩 대입하여
for i in range(len(x)):
    # 그 결과에 해당하는 predict_result 리스트를 완성
    predict_result.append(predict(x[i]))
    print("공부시간=%.f, 실제 점수=%.f, 예측 점수=%.f" % (x[i], y[i],
    predict(x[i])))
```

```python
import numpy as np

# 기울기 a와 y 절편 b
fake_a_b = [3, 76]

# x, y의 데이터 값
data = [[2, 81], [4, 93], [6, 91], [8, 97]]

x = [i[0] for i in data]

y = [i[1] for i in data]

# y = ax + b에 a와 b 값을 대입하여 결과를 출력하는 함수

def predict(x):

    return fake_a_b[0]*x + fake_a_b[1]


# MSE 함수

def mse(y_hat, y):

    return ((y_hat, y) ** 2).mean()


# MSE 함수를 각 y 값에 대입하여 최종 값을 구하는 함수

def mse_val(predict_result,y):

    return mse(np.array(predict_result), np.array(y))
```

```python
# 예측 값이 들어갈 빈 리스트

predict_result = []


# 모든 x 값을 한 번씩 대입하여

for i in range(len(x)):

    # predict_result 리스트를 완성

    predict_result.append(predict(x[i]))

    print("공부한 시간=%.f, 실제 점수=%.f, 예측 점수=%.f" % (x[i], y[i],

    predict(x[i])))


# 최종 MSE 출력

print("mse 최종값: " + str(mse_val(predict_result,y)))
```

```
공부한 시간=2, 실제 점수=81, 예측 점수=82
공부한 시간=4, 실제 점수=93, 예측 점수=88
공부한 시간=6, 실제 점수=91, 예측 점수=94
공부한 시간=8, 실제 점수=97, 예측 점수=100
mse 최종값: 11.0
```

실습 : 평균 제곱 오차 (코드)

```
import numpy as np

#가상의 기울기 a와 y 절편 b
fake_a_b=[3,76]

# x 값과 y값
data = [[2, 81], [4, 93], [6, 91], [8, 97]]
x = [i[0] for i in data]
y = [i[1] for i in data]

# y=ax + b에 a,b 값 대입하여 결과를 출력하는 함수
def predict(x):
    return fake_a_b[0]*x + fake_a_b[1]

# MSE 함수
def mse(y, y_hat):
    return ((y - y_hat) ** 2).mean()

# MSE 함수를 각 y값에 대입하여 최종 값을 구하는 함수
def mse_val(y, predict_result):
    return mse(np.array(y), np.array(predict_result))

# 예측값이 들어갈 빈 리스트
predict_result = []

# 모든 x값을 한 번씩 대입하여 predict_result 리스트완성.
for i in range(len(x)):
    predict_result.append(predict(x[i]))
    print("공부시간=%.f, 실제점수=%.f, 예측점수=%.f" % (x[i], y[i],
predict(x[i])))

# 최종 MSE 출력
print("MSE 최종값: " + str(mse_val(predict_result,y)))
```

KOREATECH 한국기술교육대학교

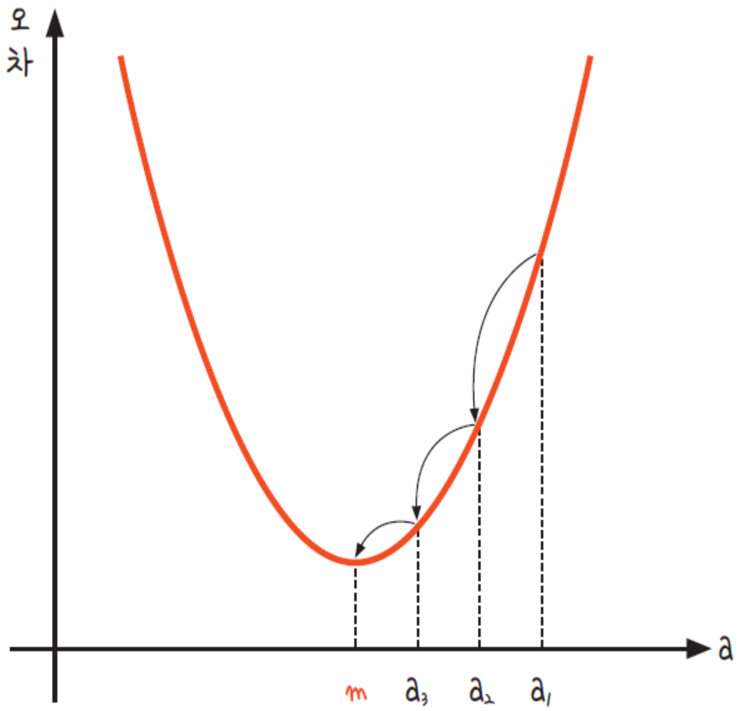
최적화 함수 : 최저 비용(오차) 구하기 –> 머신러닝 모델학습

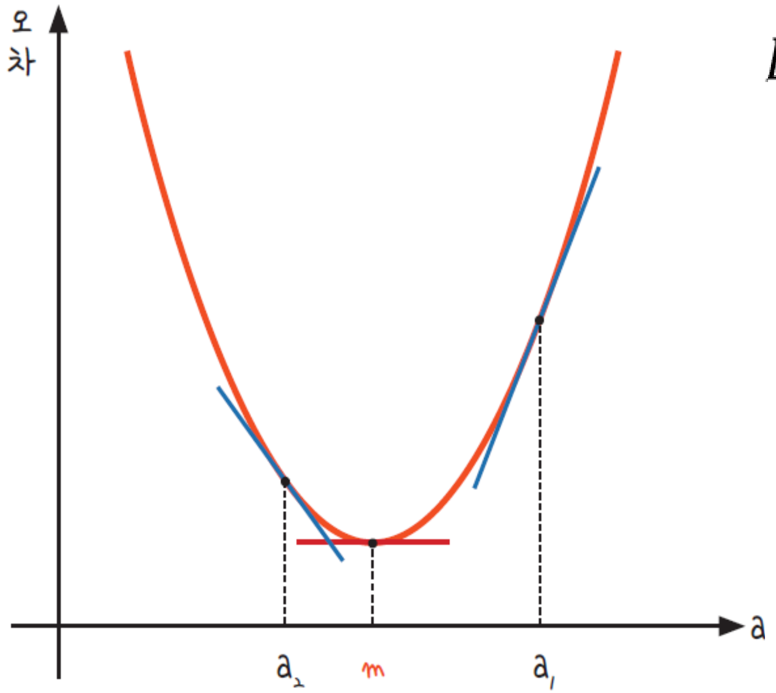
❖ 오차를 나타내는 그래프 : 오차는 a 에 관한 2차식

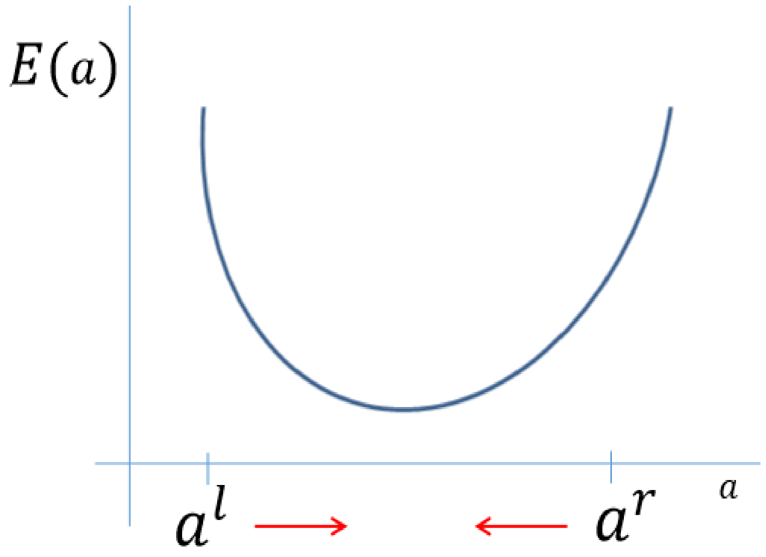
- a를 무한대로 키우면 오차도 무한대로 커지고 a를 무한대로 작게 해도 역시 오차도 무한대로 커짐.

❖ 오차 줄이기

- 기울기와 오차와의 관계: 적절한 기울기를 찾았을 때 오차가 최소화된다.
- 기울기가 0인 점이 곧 우리가 찾는 최소값 m이다. => '미분 값이 0'

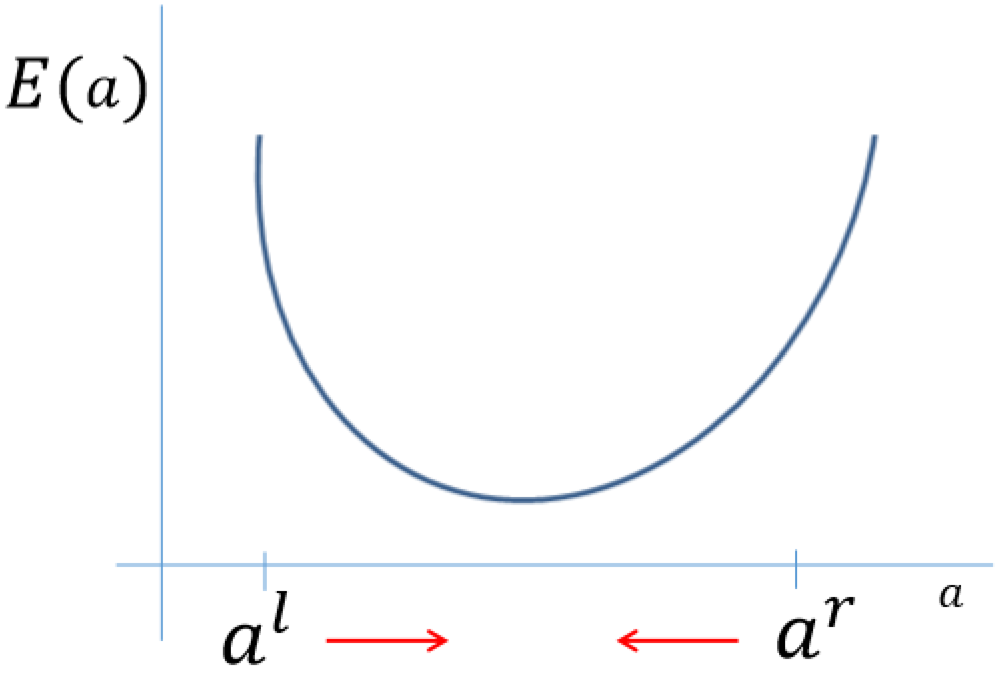






Δa =음수 for a^r
Δa =양수 for a^l
=> Δa = − 기울기

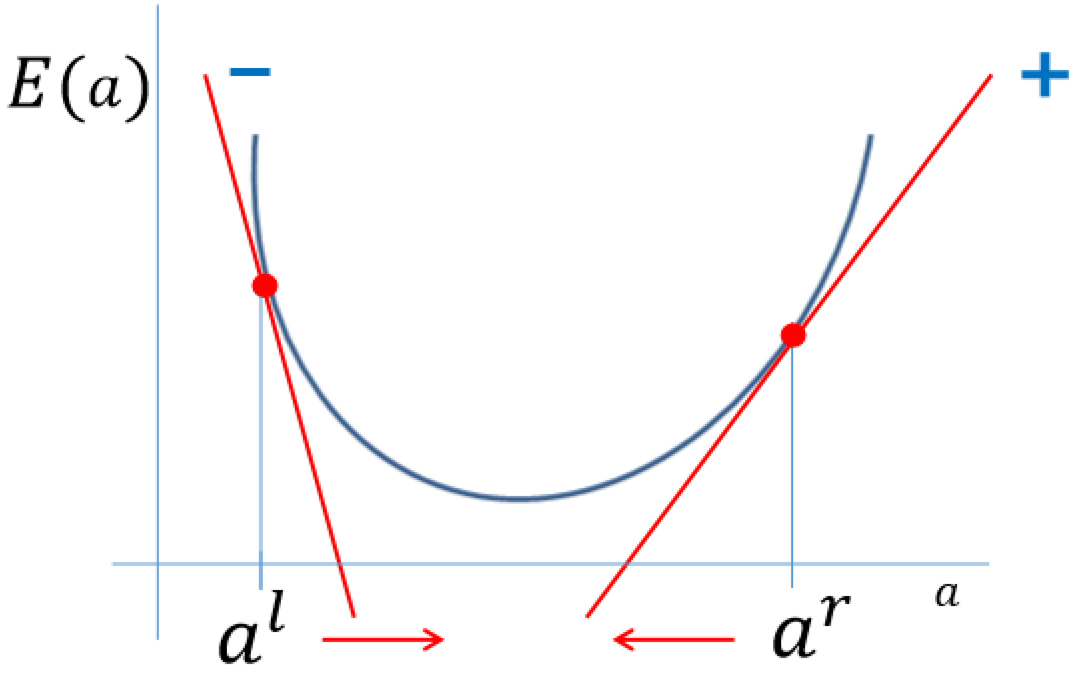
갱신 방향



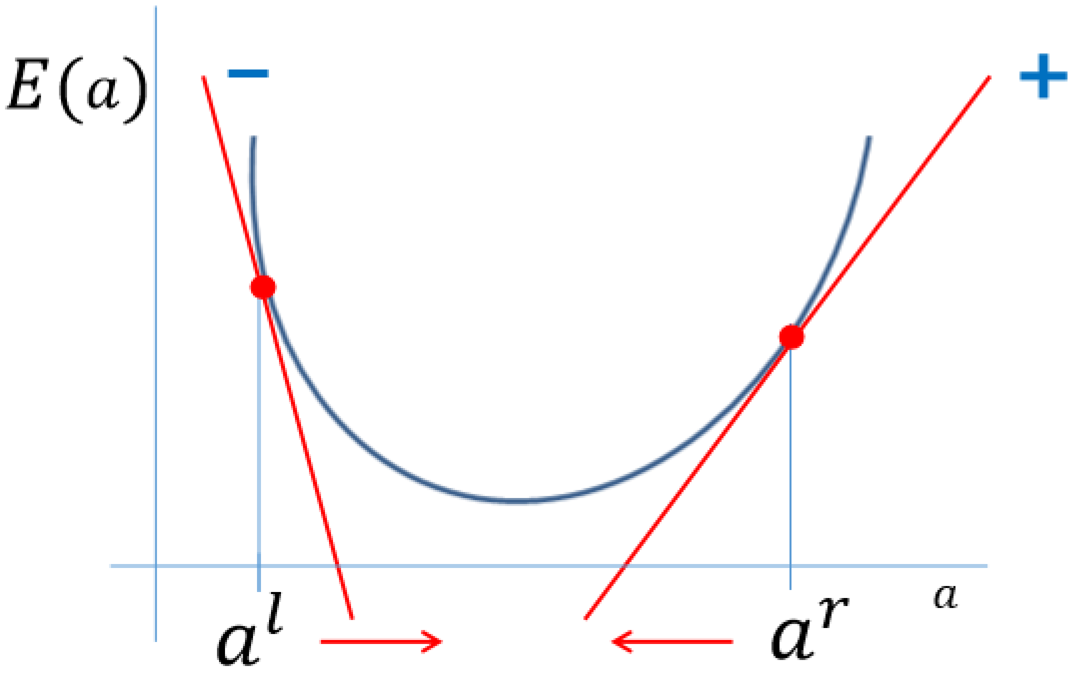
Δa =음수 for a^r

Δa =양수 for a^l

기울기; gradient

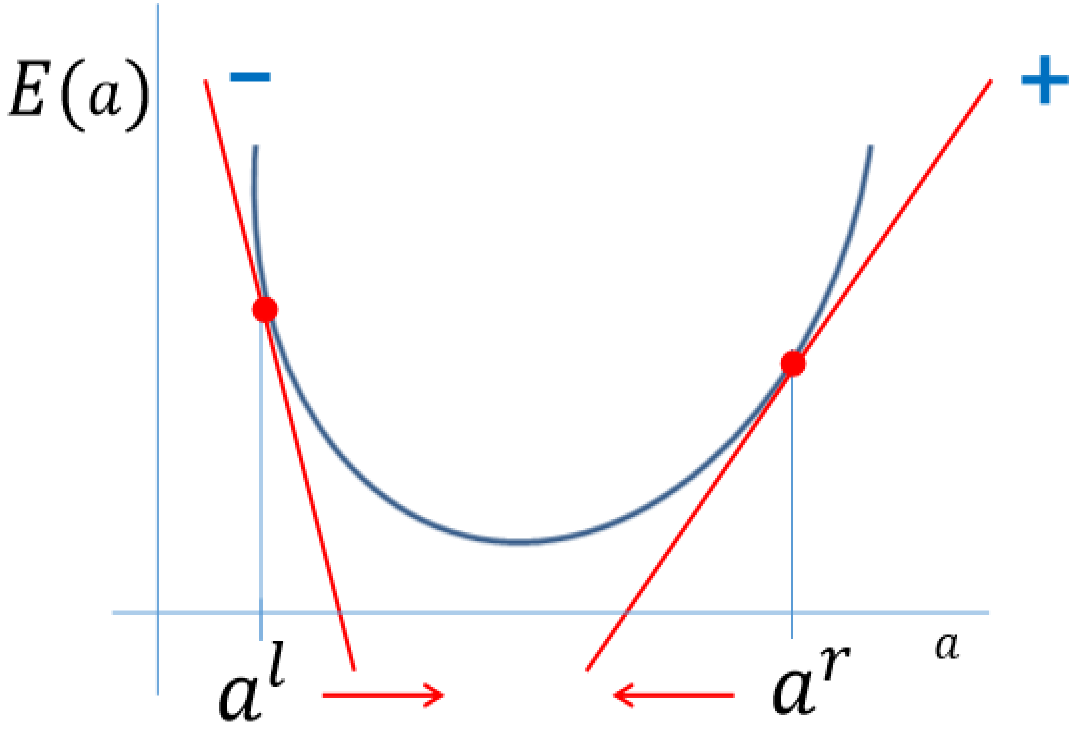


갱신방향 판정



Δa = − 기울기

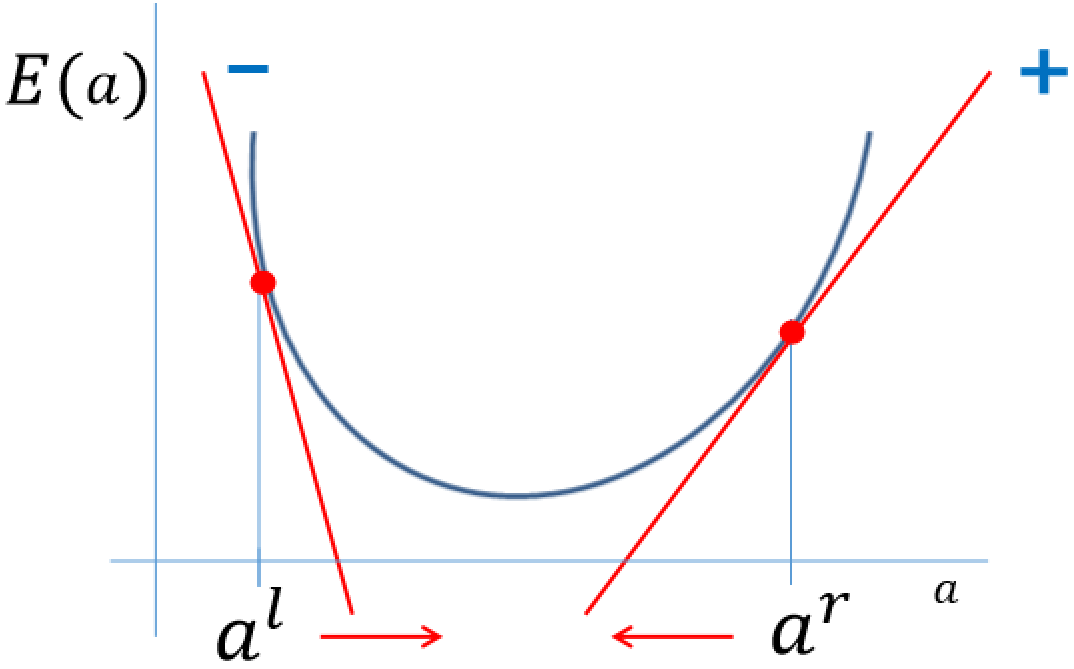
갱신 크기



Δa = − 기울기 × 아주 작은값

경사하강법

경사하강법; gradient descent method



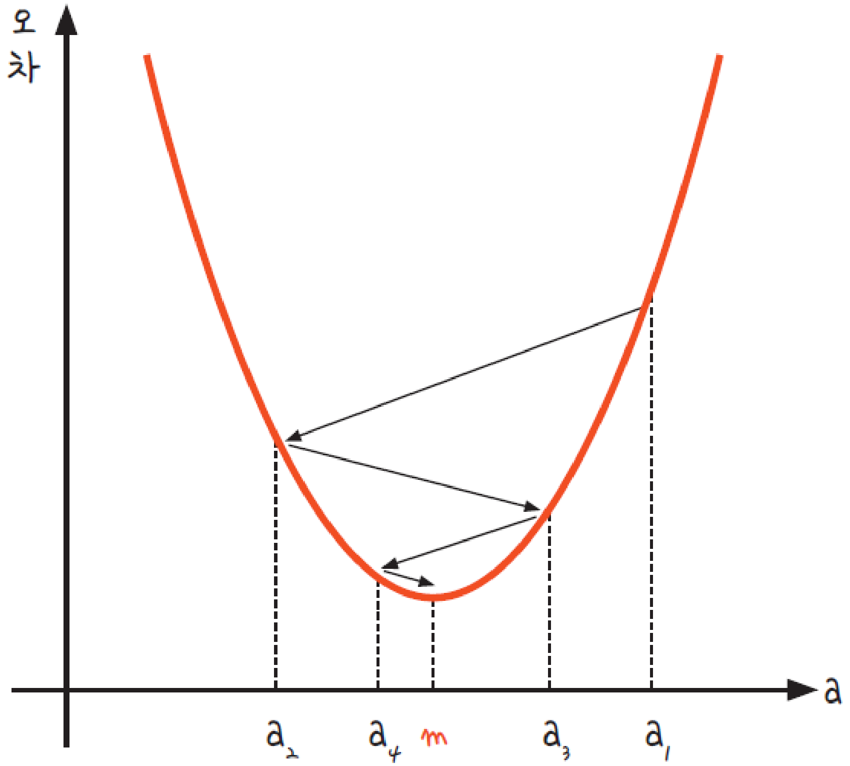
$$\Delta a = -\eta \frac{\partial E}{\partial a}$$

경사 하강법

1 | a_1 에서 미분을 구함

2 | 구해진 기울기의 반대 방향(기울기가 +면 음의 방향, -면 양의 방향)으로 얼마간 이동시킨 a_2 에서 미분을 구함.

3 | 위에서 구한 미분 값이 0이 아니면 위 과정을 반복함



- 학습률 : 어느 만큼 이동시킬지를 결정.
- 경사 하강법 :
  - 반복적으로 기울기 a를 변화시켜서 m의 값을 찾아내는 방법.

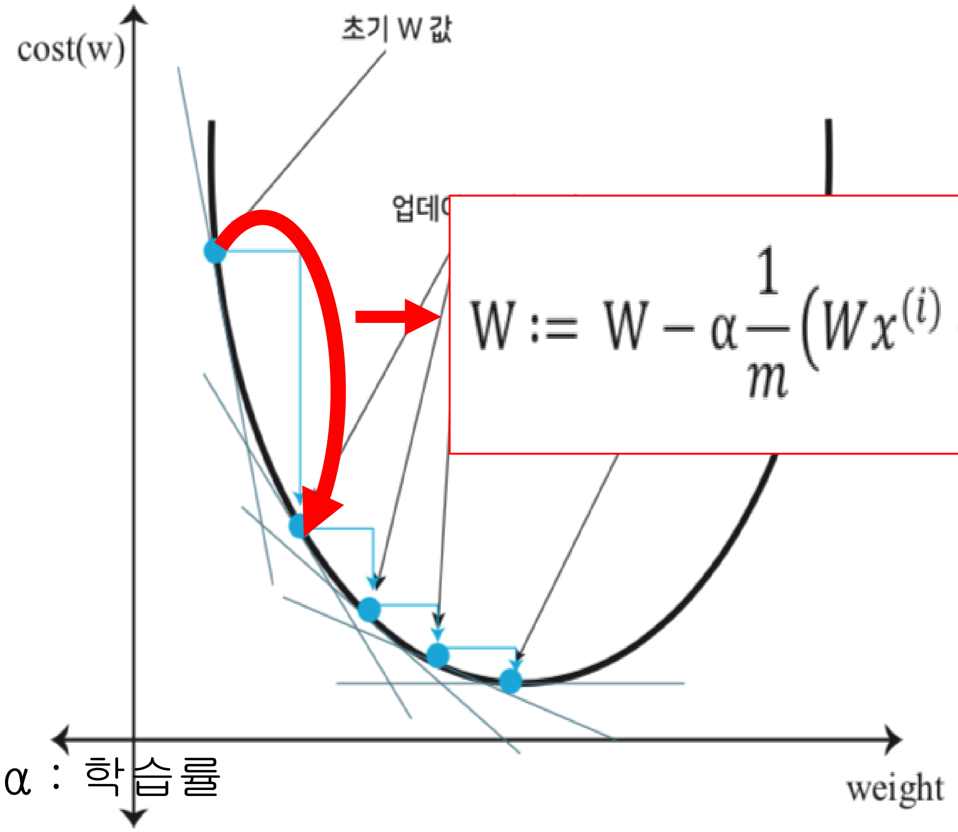
학습률

❖ α가 작은 경우

- 조금씩 이동하면 최저 지점을 찾아 감.
- 이동횟수가 많아져 학습 속도가 느림.

❖ α가 큰 경우

- 최저 지점을 빠르게 찾아 감
- 최저 지점을 지나칠 수 있음



$$W := W - \alpha \frac{1}{m}\left(Wx^{(i)} - y^{(i)}\right)x^{(i)}$$

$$H(x) = W * x + b$$

$$cost(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(H_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$W := W - \alpha \frac{\partial}{\partial w} cost(w)$$

α : 학습률

KOREATECH 한국기술교육대학교

경사 하강법 프로그램

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 공부 시간 X와 성적 Y의 리스트를 만들기
data = [[2, 81], [4, 93], [6, 91], [8, 97]]
x = [i[0] for i in data]
y = [i[1] for i in data]

# 그래프로 나타내기
plt.figure(figsize=(8,5))
plt.scatter(x, y)
plt.show()

# 리스트로 되어 있는 x와 y 값을 넘파이 배열로 바꾸기(인덱스를 주어 하나씩 불
x_data = np.array(x)
y_data = np.array(y)
```

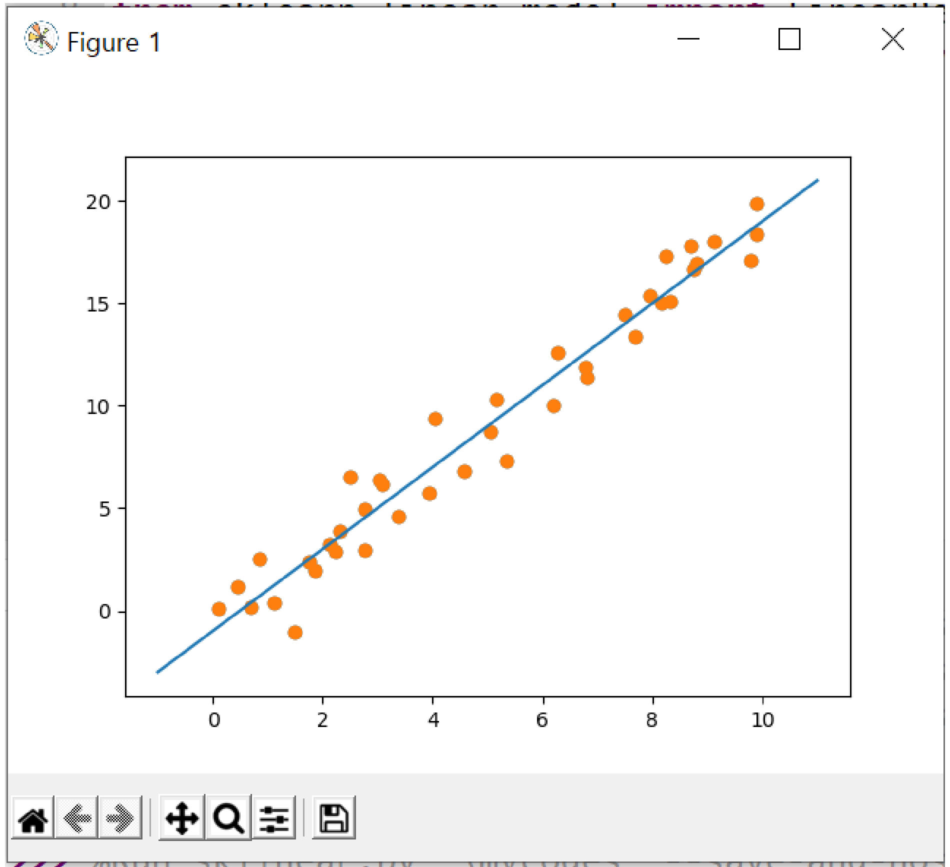
```
lr = 0.05    # 학습률 정하기

a = 0     # 기울기 a와 절편 b의 값 초기화
b = 0
# 몇 번 반복될지 설정(0부터 세므로 원하는 반복 횟수에 +1)
epochs = 2001

# 경사 하강법 시작
for i in range(epochs):        # 에포크 수만큼 반복

    y_pred = a * x_data + b   # y를 구하는 식 세우기

    error = y_data - y_pred   # 오차를 구하는 식

    # 오차 함수를 a로 미분한 값

    a_diff = -(1/len(x_data)) * sum(x_data * (error))

    # 오차 함수를 b로 미분한 값

    b_diff = -(1/len(x_data)) * sum(y_data - y_pred)

    a = a - lr * a_diff   # 학습률을 곱해 기존의 a값 업데이트

    b = b - lr * b_diff   # 학습률을 곱해 기존의 b값 업데이트

    if i % 100 == 0:      # 100번 반복될 때마다 현재의 a값, b값 출력
      print("epoch=%.f, 기울기=%.04f, 절편=%.04f" % (i, a, b))
# 앞서 구한 기울기와 절편을 이용해 그래프를 다시 그리기
y_pred = a * x_data + b

plt.scatter(x, y)

plt.plot([min(x_data), max(x_data)], [min(y_pred), max(y_pred)])

plt.show()
```

K
한

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#공부시간 X와 성적 Y의 리스트를 만듭니다.
data = [[2, 81], [4, 93], [6, 91], [8, 97]]
x = [i[0] for i in data]
y = [i[1] for i in data]

#그래프로 나타내 봅니다.
plt.figure(figsize=(8,5))
plt.scatter(x, y)
plt.show()

#리스트로 되어 있는 x와 y값을 넘파이 배열로 바꿈.
#(인덱스를 주어 하나씩 불러와 계산이 가능해 지도록 하기 위함)
x_data = np.array(x)
y_data = np.array(y)

# 기울기 a와 절편 b의 값을 초기화 합니다.
a = 0
b = 0

#학습률을 정합니다.
lr = 0.03
#몇 번 반복될지를 설정합니다.
epochs = 2001

#경사 하강법을 시작합니다.
for i in range(epochs):        # epoch 수 만큼 반복
    y_hat = a * x_data + b    #y를 구하는 식을 세웁니다
    error = y_data - y_hat    #오차를 구하는 식입니다.
    a_diff = -(2/len(x_data)) * sum(x_data * (error)) # 오차함수를 a로 미분한 값.
    b_diff = -(2/len(x_data)) * sum(error)            # 오차함수를 b로 미분한 값.
    a = a - lr * a_diff                               # 학습률을 곱해 기존의 a값을 업데이트.
    b = b - lr * b_diff                               # 학습률을 곱해 기존의 b값을 업데이트.
    if i % 100 == 0:                                  # 100번 반복될 때마다 현재의 a값, b값을 출력.
        print("epoch=%.f, 기울기=%.04f, 절편=%.04f" % (i, a, b))

# 앞서 구한 기울기와 절편을 이용해 그래프를 그려 봅니다.
y_pred = a * x_data + b
plt.scatter(x, y)
plt.plot([min(x_data), max(x_data)], [min(y_pred), max(y_pred)])
plt.show()
```

Simple linear regression의 scikit learn 적용

❖ simple linear regression




```
LinearRegression()
[1.99839352]
-0.989545945777504
```

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
rng = np.random.RandomState(35)
x = 10*rng.rand(40)
y = 2*x-1+rng.randn(40)
plt.scatter(x,y)
plt.show()
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)
X = x[:, np.newaxis]
print(model.fit(X, y))
print(model.coef_)
print(model.intercept_)
xfit = np.linspace(-1, 11)
Xfit = xfit[:, np.newaxis]
yfit = model.predict(Xfit)
plt.scatter(x, y)
plt.plot(xfit, yfit)
plt.show()
```


KOREATECH 한국기술교육대학교