AI : Linear Regression

Date	@2021/09/08

문제 이해

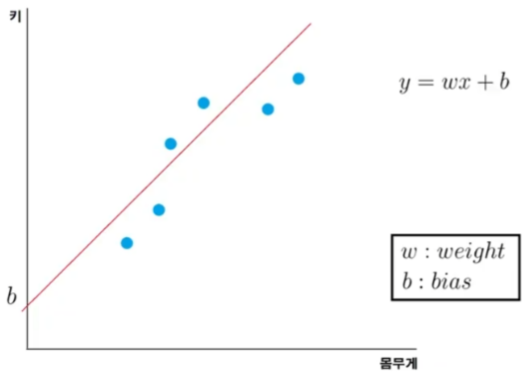
Train Data set (Feature x: 몸무게, Traget t: 키)

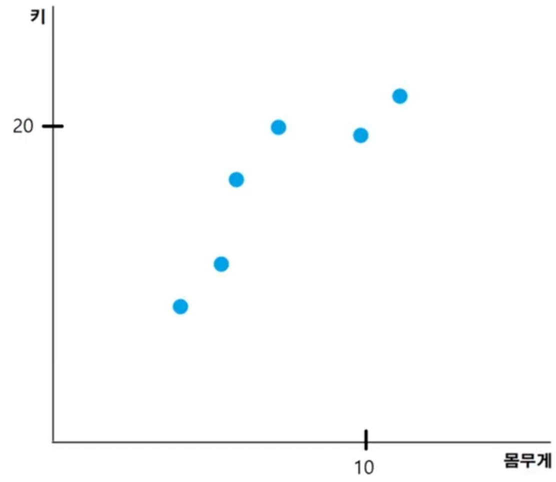
스머프	몸무게	키
파파	5.0	13.0
투덜이	6.0	15.5
욕심이	10.0	22.5
요리사	7.0	17.0
우주인	8.0	20.0
농부	12.0	26.5

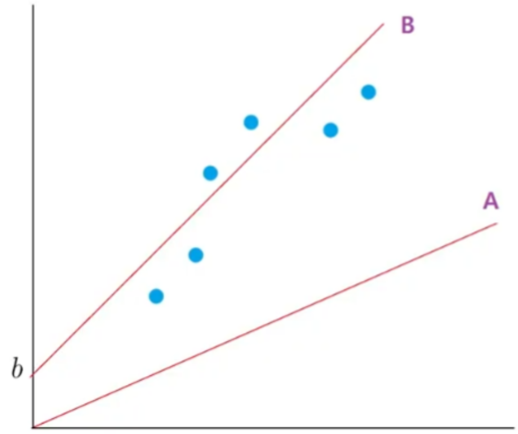
Test Data set (y (predict): 키)

스머프	몸무게	키
똘똘이	9.0	?

선형함수

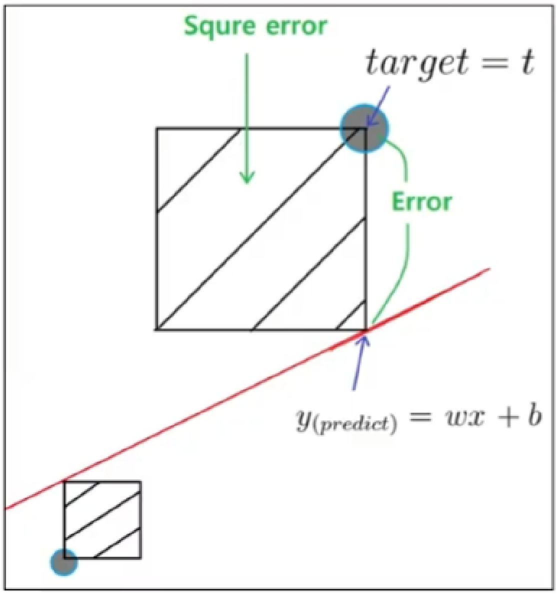






- A보다 B가 더 좋은 예측 함수이다. 실제 데이터와의 오차범위가 더 적기 때문이다.
- 즉, 예측하기 좋은 함수는 오차범위가 제일 적은 함수이다.

MSE(Mean Square Error)



Mean Square Error(MSE)

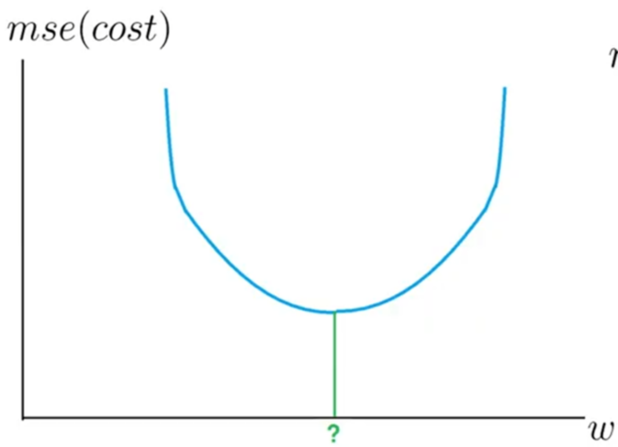
$$y_{i(predict)} = wx_i + b$$

$$error_i = e_i = wx_i + b - t_i$$

$$mse = \frac{1}{n}\sum_{i=1}^{n}(wx_i + b - t_i)^2$$

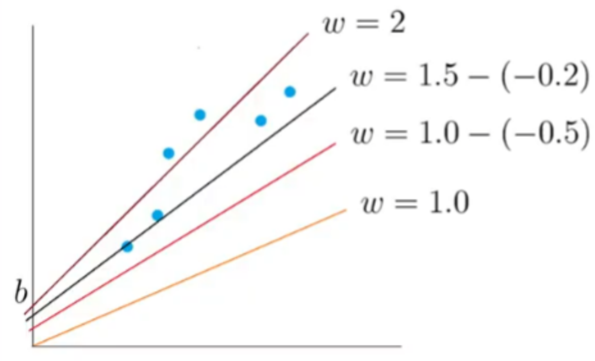
- MAE, RMSE 등의 다양한 방법도 있다.

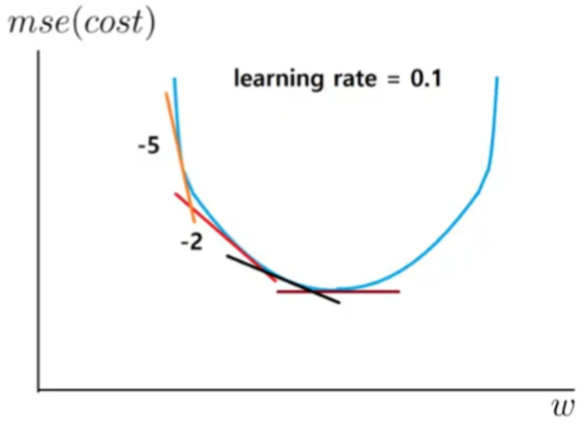
경사하강법



$$mse = \frac{1}{n}\sum_{i=1}^{n}(wx_i + b - t_i)^2$$

Gradient Descent





$$\frac{\partial}{\partial w}\frac{1}{n}\sum_{i=1}^{n}(wx_i + b - t_i)^2$$

$$\frac{\partial}{\partial w}\frac{1}{n}\sum_{i=1}^{n} w^2x_i^2 + 2wx_ib + b^2 - 2t_i(wx_i + b) + t_i^2$$

$$\frac{1}{n}\sum_{i=1}^{n} 2wx_i^2 + 2x_ib - 2t_ix_i \qquad \frac{2}{n}\sum_{i=1}^{n}(wx_i + b - t_i)x_i = \boxed{\frac{2}{n}\sum_{i=1}^{n} e_ix_i}$$

$$\frac{\partial}{\partial b}\frac{1}{n}\sum_{i=1}^{n}(wx_i+b-t_i)^2$$

$$\frac{\partial}{\partial b}\frac{1}{n}\sum_{i=1}^{n} w^2x_i^2+2wx_ib+b^2-2t_i(wx_i+b)+t_i^2$$

$$\frac{1}{n}\sum_{i=1}^{n}2wx_i+2b-2t_i$$

$$\frac{2}{n}\sum_{i=1}^{n}wx_i+b-t_i=\boxed{\frac{2}{n}\sum_{i=1}^{n}e_i}$$

MSE 편미분

$$y_{i(predict)} = wx_i + b$$

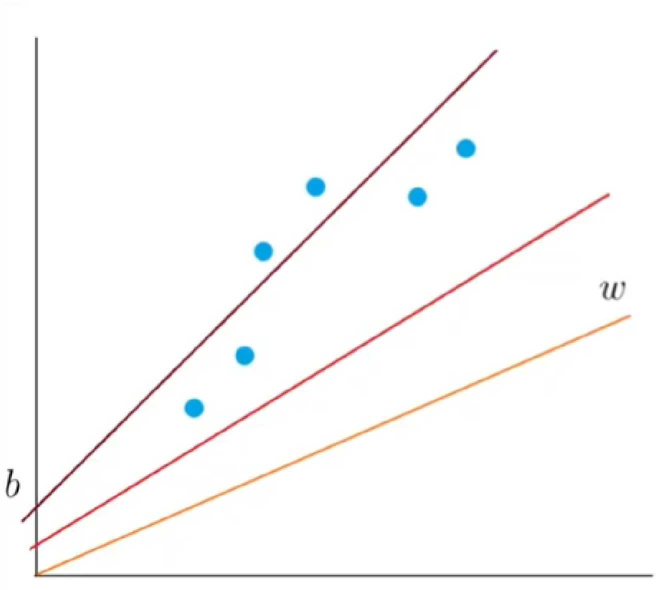
$$error_i = e_i = wx_i + b - t_i$$

$$\frac{\partial mse}{\partial w} = \frac{2}{n}\sum_{i=1}^{n} e_i x_i \approx \boxed{\sum_{i=1}^{n} e_i x_i}$$

$$\frac{\partial mse}{\partial b} = \frac{2}{n}\sum_{i=1}^{n} e_i \approx \boxed{\sum_{i=1}^{n} e_i}$$

2/n은 생략해보 무관하다.

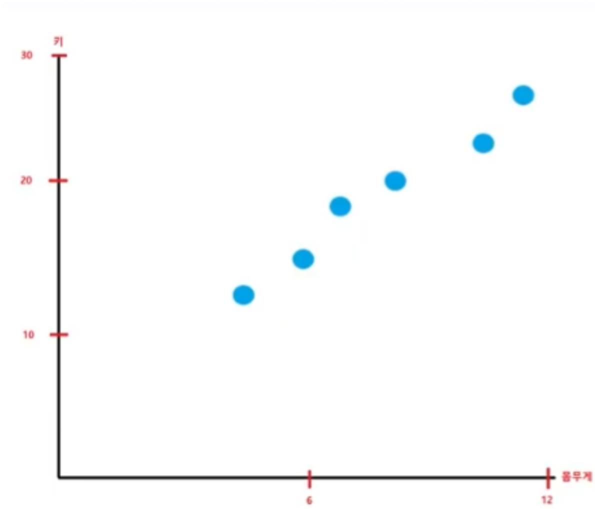
경사하강법 최종 수식

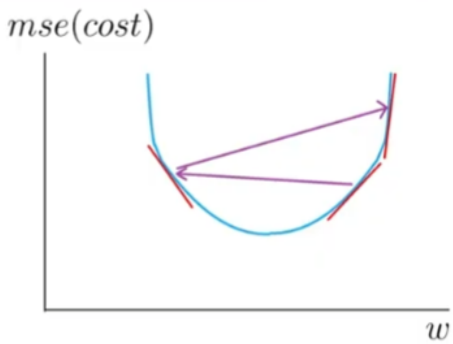


$$w_{(new)} = w - \gamma \sum_{i=1}^{n} e_i x_i$$

$$b_{(new)} = b - \gamma \sum_{i=1}^{n} e_i$$

Exploading Gradient





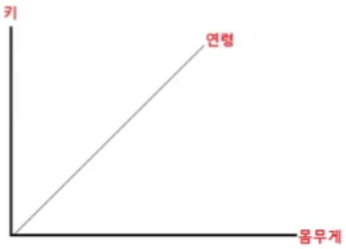
$$w_{(new)} = w - \gamma \sum_{i=1}^{n} e_i x_i$$

N-Dimension

스머프	몸무게	연령	키
파파	5.0	50.0	13.0
투덜이	6.0	20.0	15.5
욕심이	10.0	30.0	22.5
요리사	7.0	40.0	17.0
우주인	8.0	20.0	20.0
농부	12.0	60.0	26.5

스머프	몸무게	연령	키
똘똘이	9.0	40.0	?

$$y = w_1x_1 + w_2x_2 + b$$




```
x = np.array([[5.0, 50], [6.0, 20], [10.0, 30], [7.0, 40], [8.0, 20], [12.0, 60]])
target = np.array([[13.0], [15.5], [22.5], [17.0], [20.0], [26.5]])

weight, bias = train(x, target, learning_rate = 1e-4, epochs = 100000)

print('weight  : ', weight)
print('bias   : ', bias)

test_x = np.array([[9.0, 40]])

y = test(test_x, weight, bias)

print('test x : ', test_x)
print('predict y : ', y)
```

```
weight  :  [[ 1.9314564 ][-0.02063594]]
bias  :  [4.38369446]
test x :  [[ 9. 40.]]
predict y :  [[20.94136462]]
```

```
def test(x, weight, bias):

    return np.dot(x, weight) + bias
```

$$y_{i(predict)} = wx_i + b$$

```
y = test(x, weight, bias)

error = y - target
```

$$error_i = e_i = wx_i + b - t_i$$

```python
def train(x, target, learning_rate, epochs):

    inputNodes = x.shape[-1]
    outputNodes = target.shape[-1]
    weight = np.zeros((inputNodes, outputNodes))
    bias = np.zeros(outputNodes)

    for i in range(epochs):
        y = test(x, weight, bias)

        error = y - target

        mse = np.average(error**2)

        print_summary(i, mse)

        weight_delta, bias_delta = gradient(x, error)

        weight -= (learning_rate * weight_delta)
        bias -= (learning_rate * bias_delta)

    return weight, bias
```

```python
def gradient(x, error):

    weight_delta = np.dot(x.T, error)
    bias_delta = np.sum(error, axis=0)

    return weight_delta, bias_delta
```

$$w_{(new)} = w - \gamma \boxed{\sum_{i=1}^{n} e_i x_i}$$

$$b_{(new)} = b - \gamma \boxed{\sum_{i=1}^{n} e_i}$$

```python
weight_delta, bias_delta = gradient(x, error)

weight -= (learning_rate * weight_delta)
bias -= (learning_rate * bias_delta)
```

$$w_{(new)} = w - \gamma \sum_{i=1}^{n} e_i x_i$$

$$b_{(new)} = b - \gamma \sum_{i=1}^{n} e_i$$