Linear Regression Cost 함수 최소화

$H(x) = Wx + b$.

$$\frac{(H(x_1)-y(1))^2 + (H(x_2)-y(2))^2 + (H(x_3)-y(3))^2}{3}$$

예측 값 ↓ 정답 labeled 값 ↓

$$\Rightarrow \text{일반화} \quad Cost = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)})-y^{(i)})^2, \quad H(x)=Wx+b$$

$$Cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)})-y^{(i)})^2$$

Goal: minimize Cost(W, b)

"W와 b를 최소화"

$H(x) = Wx$

$$Cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$

Min

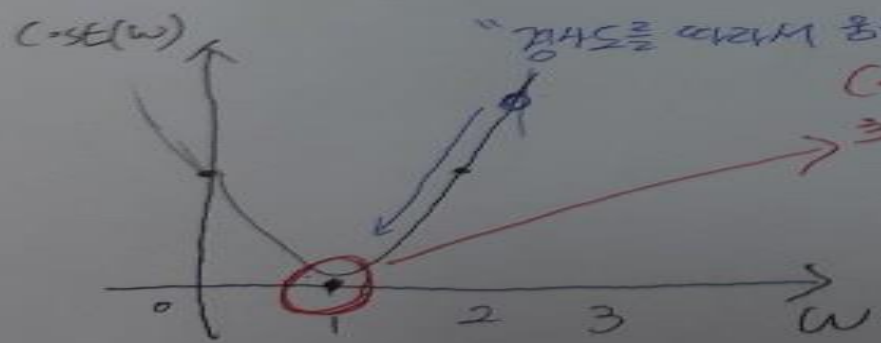
x	y
1	1
2	2
3	3

i) $W=1$, $cost(W)=?$

$$\frac{(1\times1-1)^2+(1\times2-2)^2+(1\times3-3)^2}{3}=0$$

ii) $W=0$, $Cost(W)=?$

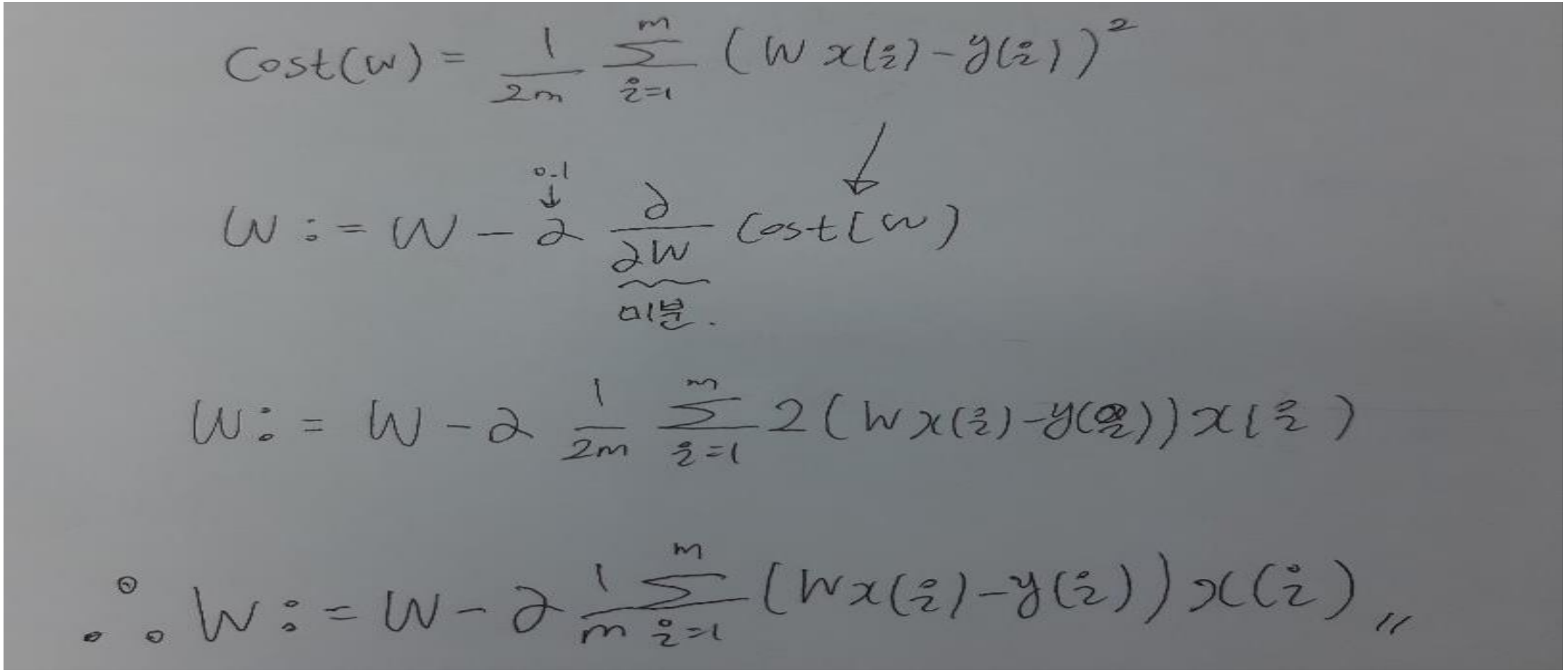
$$\frac{(0\times1-1)^2+(0\times2-2)^2+(0\times3-3)^2}{3}=\frac{14}{3}=4.67$$



Cost(W) 최소화하는 점.. ⇒ Gradient descent algorithm.

- 아무 점에서 시작
- W, b를 조금 바꿈

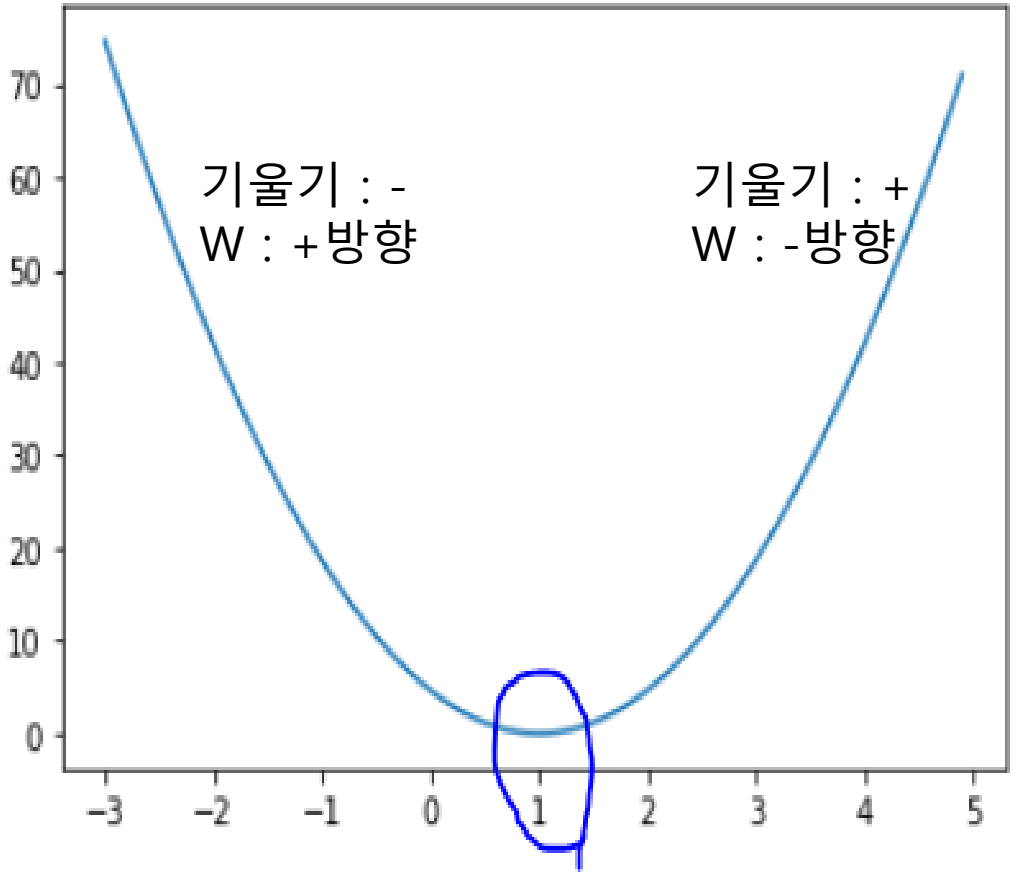
Gradient descent algorithm



Convex function

- Cost function이 convex function이 되어야 한다.
  - 어느 점에서 시작하더라도 항상 도착 지점이 원하는 지점이다.
  - 항상 답을 찾음을 보장함.

Tensorflow로 linear regression cost 최소화 구현



미분한 cost function

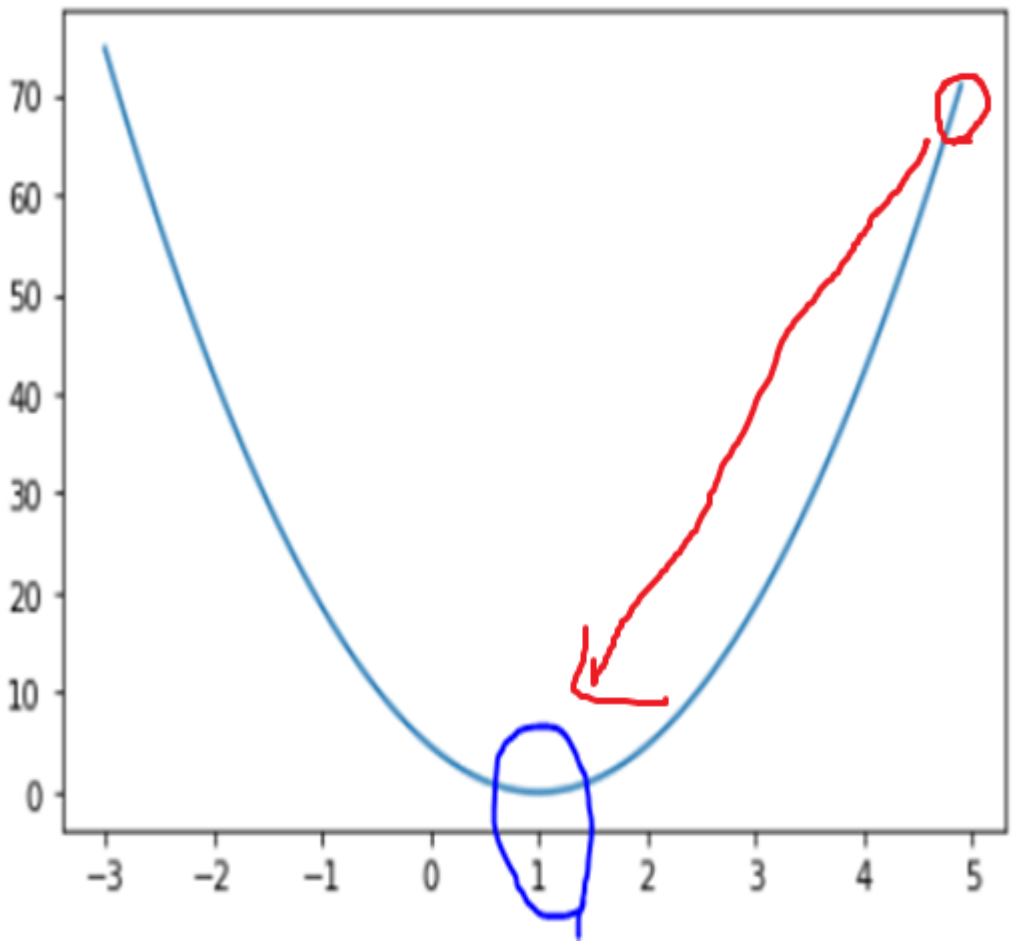
$$\sum_{i=1}^{m} (Wx(i) - y(i))\, x(i)$$

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

```
# cost(loss) function
cost = tf.reduce_sum(tf.square(hypothesis - Y))

# gradient descent
learning_rate = 0.1
gradient = tf.reduce_mean((W*X - Y) * X)
descent = W- learning_rate * gradient
update = W.assign(descent)
```

Tensorflow로 linear regression cost 최소화 구현2



$$cost(w) = \frac{1}{m}\sum_{i=1}^{m}\left(Wx^{(i)} - y^{(i)}\right)^2$$

```python
# 같은 표현

# Part1 - Build graph using TF operations
X = [1,2,3]
Y = [1,2,3]

W = tf.Variable(5.0)

# H(x) = Wx+b
hypothesis = W * X

# cost(loss) function
cost = tf.reduce_mean(tf.square(hypothesis - Y))

# gradient descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate = 0.1)
train = optimizer.minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for step in range(100):
    print(step, sess.run(W))
    sess.run(train)
```