Section 3. Linear Regression의 cost 함수 최적화

I. Linear Regression의 cost 최소화 알고리즘의 원리

1. cost(W)

$$Cost(w) = \frac{1}{m}\sum_{i=1}^{m}(Wx_i - y_i)^2$$

X	Y
1	1
2	2
3	3

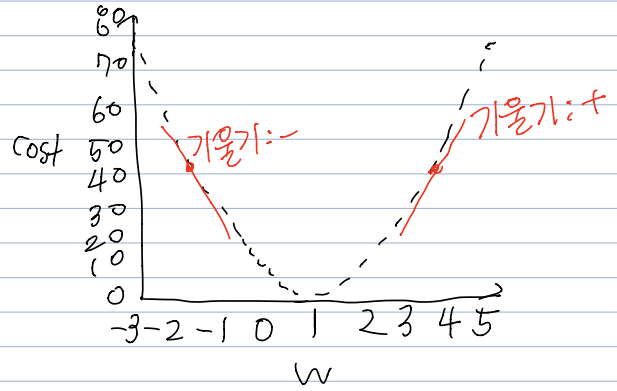
1) W=1 일 때

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

2) W=0 일 때

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

3) W=2 일 때 4.67



2. Gradient descent algorithm

- cost function 최적화에 쓰임.
- 가장 낮은 지점을 찾으려면 경사도가 0인 위치를 찾는다.

$$Cost(w) = \frac{1}{m}\sum_{i=1}^{m}(Wx_i - y_i)^2 \Rightarrow Cost(w) = \frac{1}{2m}\sum_{i=1}^{m}(Wx_i - y_i)^2$$

$$W := W - 2\frac{\partial}{\partial W}Cost(W)$$

$$W := W - 2\frac{1}{2m}\sum_{i=1}^{m}2(Wx_i - y_i)x_i$$

$$W := W - 2\frac{1}{m}\sum_{i=1}^{m}(Wx_i - y_i)x_i$$

II. Linear Regression의 cost 최소화의 Tensorflow 구현

$H(x) = Wx$ $\quad cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx_i - y_i)^2$

```
import tensorflow as tf
import matplotlib.pyplot as plt
X = [1, 2, 3]
Y = [1, 2, 3]
W = tf.placeholder(tf.float32)
hypothesis = X * W
cost = tf.reduce_mean(tf.square(hypothesis - Y))
sess = tf.Session()
sess.run(tf.global_variables_initializer())
W_val = []
cost_val = []
for i in range(-30, 50):
    feed_W = i * 0.1
    curr_cost, curr_W = sess.run([cost, W], feed_dict
                                 ={W: feed_W})
    W_val.append(curr_W)
    cost_val.append(curr_cost)
plt.plot(W_val, cost_val)
plt.show()
```

1. Gradient descent using derivative

$$W := W - 2\frac{1}{m}\sum_{i=1}^{m}(Wx_i - y_i)x_i$$

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

2. compute_gradient and apply_gradient

```
import tensorflow as tf
X = [1,2,3]
Y = [1,2,3]
W = tf.Variable(5.)
hypothesis = X * W
gradient = tf.reduce_mean((W*X-Y)*X)*2
cost = tf.reduce_mean(tf.square(hypothesis - Y))
optimizer = tf.train.GradientDescentOptimizer
            (learning_rate = 0.01)

gvs = optimizer.compute_gradients(cost)
apply_gradients = optimizer.apply_gradients(gvs)
sess = tf.Session()
sess.run(tf.global_variables_initializer())
for step in range(100):
    print(step, sess.run([gradient, W, gvs]))
    sess.run(apply_gradients)
```