중요 : XOR 학습하기

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt


'''
# XOR_GATE
x_data = [[0, 0],
          [0, 1],
          [1, 0],
          [1, 1]]

y_data = [[0],
          [1],
          [1],
          [0]]
'''
# AND_GATE
x_data = [[0, 0],
          [0, 1],
          [1, 0],
          [1, 1]]

y_data = [[0],
          [0],
          [0],
          [1]]
```

x_data는 train set을 의미하며 y_data는 정답 레이블을 의미한다.

XOR_GATE

- x_data = [0,0], [0,1], [1,0], [1,1]
- y_data = [0], [1], [0], [1]

AND_GATE

- x_data = [0,0], [0,1], [1,0], [1,1]
- y_data = [0], [0], [0], [1]

```
x_data = np.array(x_data, dtype=np.float32)
y_data = np.array(y_data, dtype=np.float32)


X = tf.placeholder(tf.float32, [None, 2]) # 2 input
Y = tf.placeholder(tf.float32, [None, 1]) # 1 output

# 1개의 Layer
W1 = tf.Variable(tf.random_normal([2, 1]), name='weight1') # Actually output
b1 = tf.Variable(tf.random_normal([1]), name='bias1') # Actually output
hypothesis = tf.sigmoid(tf.matmul(X, W1) + b1) # matmul은 Matrix Muliply로 행렬곱이다.


cost = tf.reduce_mean(tf.square(hypothesis - Y)) # hypothesis - Y 를 제곱해서 평균값을 낸다. :: RM
train = tf.train.GradientDescentOptimizer(learning_rate=learning_rate).minimize(cost) # cost : 가
# train은 최적화를 시켜줘서 오차값을 줄여주게 된다.
# GradientDescentOptimizer는 최소값을 찾아가는데 최소화의 대상이 cost(오차)이므로...


# Launch graph
```

```
sess = tf.Session()


# TensorFlow 변수들(variables) 초기화 (Initialization)
sess.run(tf.global_variables_initializer())


for i in range(10001):
    sess.run(train, feed_dict={X: x_data, Y: y_data})

    if i % 1000 ==0:
        c1 = sess.run(cost, feed_dict={X: x_data, Y: y_data})
        print('step={} / cost={}'.format(i, c1))
```

결과 확인하기

```
for i in range(4):
    x1 = x_data[[i], :]

    l1 = tf.sigmoid(tf.matmul(x1, W1) + b1)


    print( i, sess.run(l1))
    #print( i, sess.run(l2), sess.run(l2cast), y_data[[i], :])
```

- HW : 위의 코드를 변형하여 XOR 학습시 얻어진 Cost 그래프를 그리시오. Hint : List 사용

참고 : Sigmoid

```
y1 = 1.0
y2 = sess.run(tf.sigmoid(y1))
print('{} --> {}'.format(y1, y2))
```

Sigmoid를 그려볼까요?

```
x1 = np.arange(-10,10, 0.5)
print(x1)
```

```
for i in range(len(x1)):
    y1 = x1[i]
    y2 = sess.run(tf.sigmoid(y1))
    print('{} --> {}'.format(y1, y2))
```