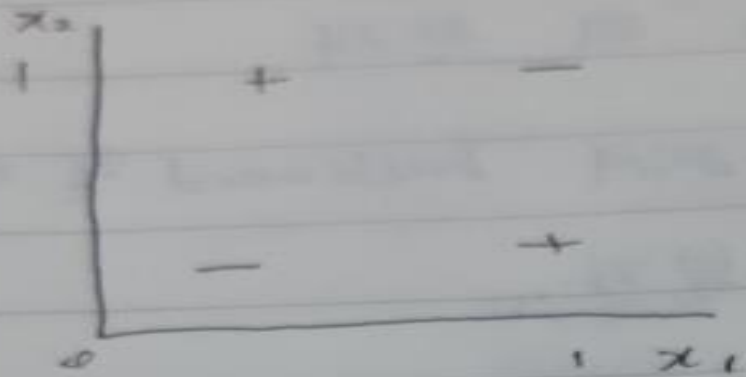
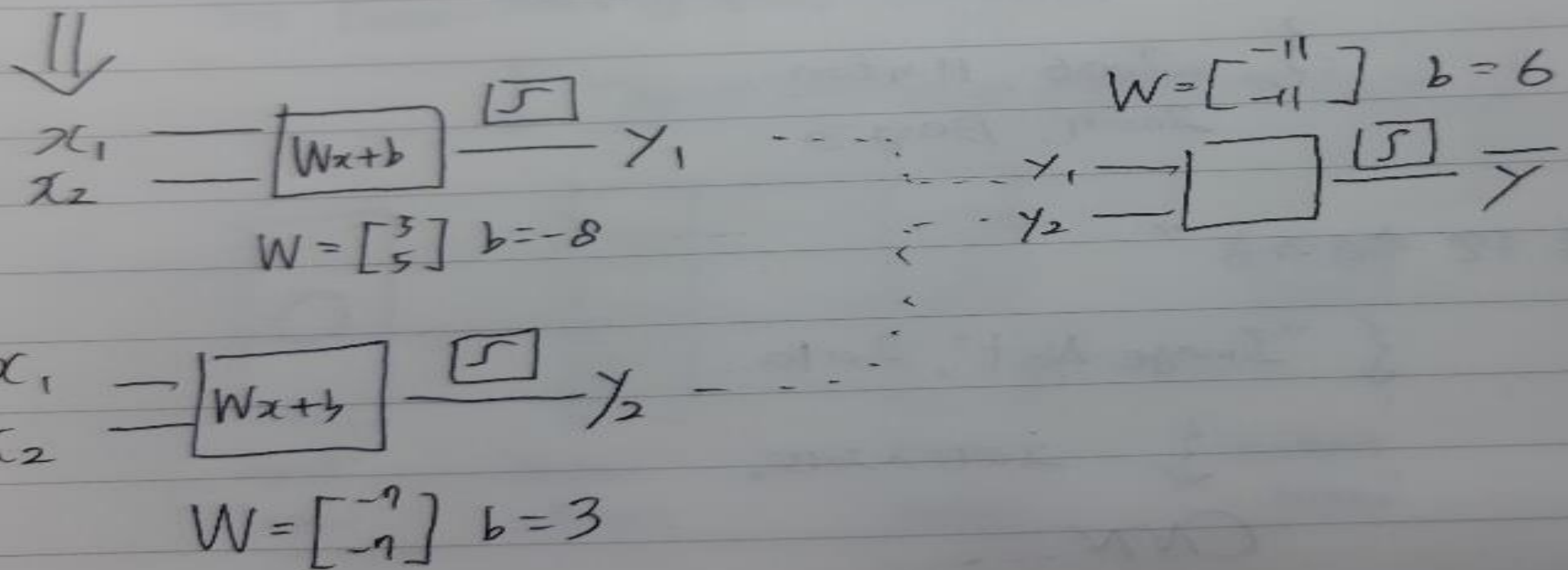
Neural Network1 : XOR문제와 학습방법 Backpropagation

x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0





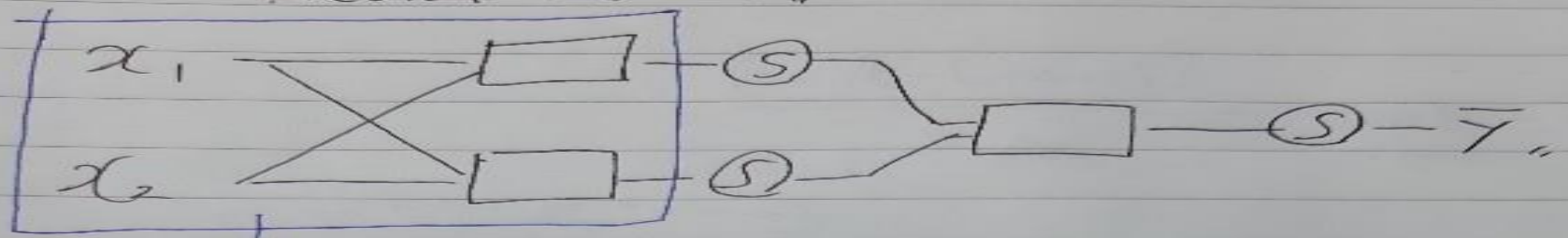
$$[0\ \ 0]\begin{bmatrix}5\\5\end{bmatrix}-8=-8,\ Y_1=S(-8)=0$$

$$[0\ \ 0]\begin{bmatrix}-7\\-7\end{bmatrix}+3=3,\ Y_2=S(3)=1$$

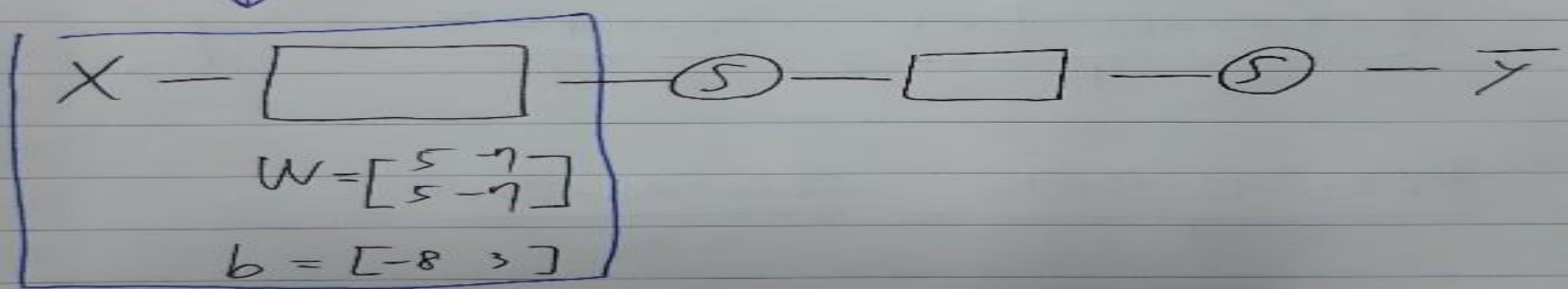
$$[0\ \ 1]\begin{bmatrix}-11\\-11\end{bmatrix}+6=-5,\ \bar{Y}=S(-5)=0$$

x_1	x_2	Y_1	Y_2	$\bar{Y}$	XOR
0	0	0	1	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	0	0	0

Neural Network



W와 b를 다른 값으로 정해 가능.



$K(x) = \text{sigmoid}(XW_1 + b_1)$

$\bar{Y} = H(x) = \text{sigmoid}(K(x)W_2 + b_2)$

미분

$$\frac{d}{dx} f(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

($\Delta x = 0.1$ 이라고 가정)

$f(x) = 3$ $\quad \frac{f(x+0.1) - f(x)}{0.1} = \frac{3-3}{0.1} = 0$

$f(x) = x$ $\quad \frac{f(x+0.01) - f(x)}{0.01} = \frac{x+0.01-x}{0.01} = 1$

$f(x) = 2x$ $\quad \frac{f(x+0.01) - f(x)}{0.01} = \frac{2(x+0.01) - 2x}{0.01} = 2$

편미분 (Partial derivative) : 상수 취급 (Variable)

$f(x, y) = xy$ $\quad \frac{\partial y}{\partial x} = y$

관심변수 (=미분)

$f(x, y) = xy$ $\quad \frac{\partial x}{\partial y} = x$

관심변수 (=미분)

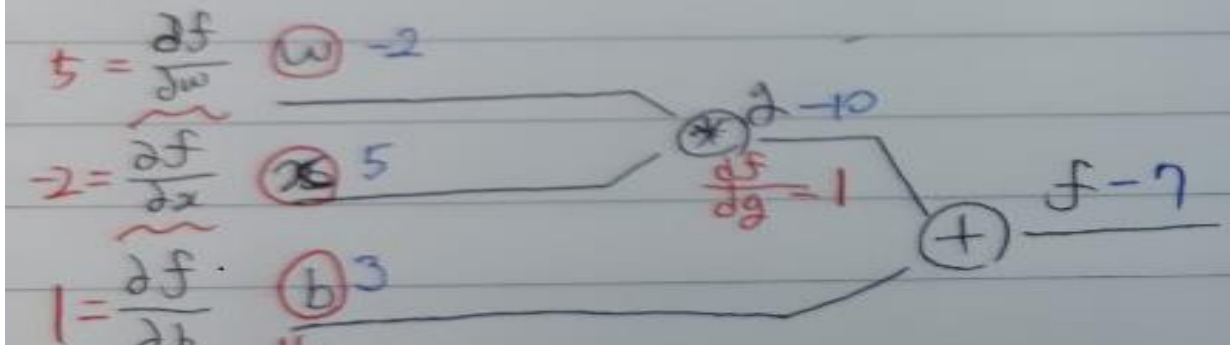
예측값과 실제값과 비교한 loss (=cost)를

되어서 부터 앞으로 작동해 무엇을 계산해야 할지 알아냄,,

$f = wx + b$, $g = wx$, $f = g + b$,,

$\frac{\partial g}{\partial w} = x$ $\quad \frac{\partial f}{\partial g} = 1$

$\frac{\partial g}{\partial x} = w$ (곱셈) $\quad \frac{\partial f}{\partial b} = 1$



⇓

⇒ 각 f에 미치는 영향을 알아내자

① $\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g} * \frac{\partial g}{\partial w} = 5$

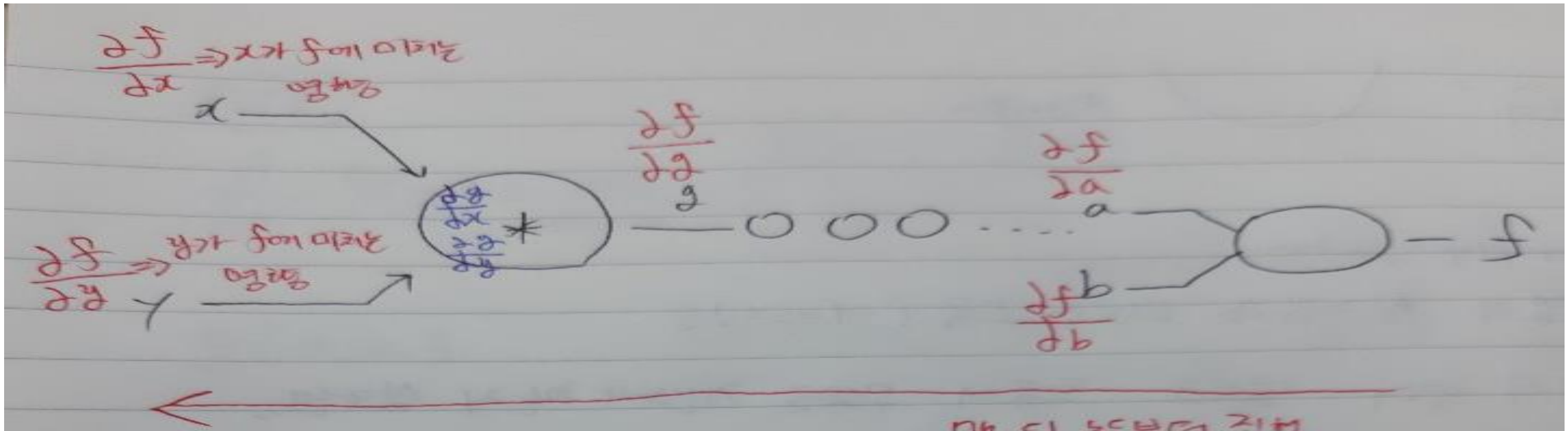
$\frac{\partial f}{\partial g} = 1$ $\quad \frac{\partial g}{\partial w} = x(5)$

② $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} * \frac{\partial g}{\partial x} = -2$

$\frac{\partial f}{\partial g} = 1$ $\quad \frac{\partial g}{\partial x} = w(-2)$

① forward,, ($w=-2, x=5, b=3$) ⇒ $f=-7$.

② Backward,,



"Chain Rule"

① $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}$

② $\frac{\partial f}{\partial y} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial y}$

XOR을 위한 TF Deep Network

```
# instance, 특성수(x1,x2)
X = tf.placeholder(tf.float32, [None, 2])
# instance, 결과(y)
Y = tf.placeholder(tf.float32, [None, 1])

# 특성수, 결과(y)
W = tf.Variable(tf.random_normal([2, 1]), name="weight")
# 결과(y)
b = tf.Variable(tf.random_normal([1]), name="bias")

# Hypothesis using sigmoid: tf.div(1., 1. + tf.exp(tf.matmul(X, W)))
hypothesis = tf.sigmoid(tf.matmul(X, W) + b)
```

```
Hypothesis:  [[0.5]
 [0.5]
 [0.5]
 [0.5]]
Correct:  [[0.]
 [0.]
 [0.]
 [0.]]
Accuracy:  0.5
```

결과가 좋지 않음

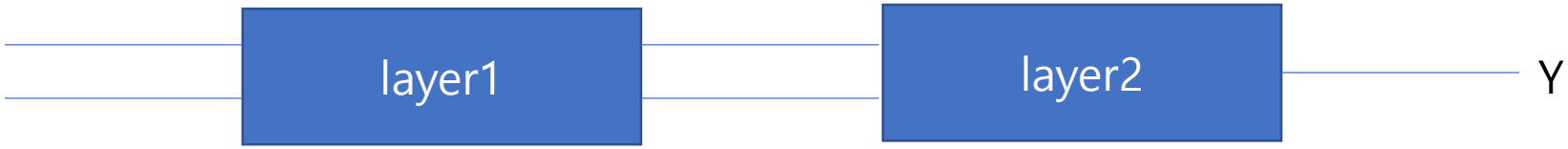
Y

XOR을 위한 TF Deep Network

```python
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)

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

# 첫번째 layer를 구성함(X*W1+b1)
# Weight의 크기 특성수(x1,x2), 결과는 임의로 지정
W1 = tf.Variable(tf.random_normal([2, 2]), name='weight1')
# 나가는 결과와 일치시키기
b1 = tf.Variable(tf.random_normal([2]), name='bias1')
layer1 = tf.sigmoid(tf.matmul(X, W1) + b1)

# 첫번째 layer의 결과값을 두 번째 layer로 넘겨준다.
# W1의 out과 최종 나가는 결과(1)
W2 = tf.Variable(tf.random_normal([2, 1]), name='weight2')
# 최종 나가는 결과(1)
b2 = tf.Variable(tf.random_normal([1]), name='bias2')
hypothesis = tf.sigmoid(tf.matmul(layer1, W2) + b2)
```

```
Hypothesis:
[[0.01859056]
 [0.9855298 ]
 [0.9854667 ]
 [0.0160412 ]]
Predicted:
[[0.]
 [1.]
 [1.]
 [0.]]
Accuracy:
1.0
```

결과가 좋음

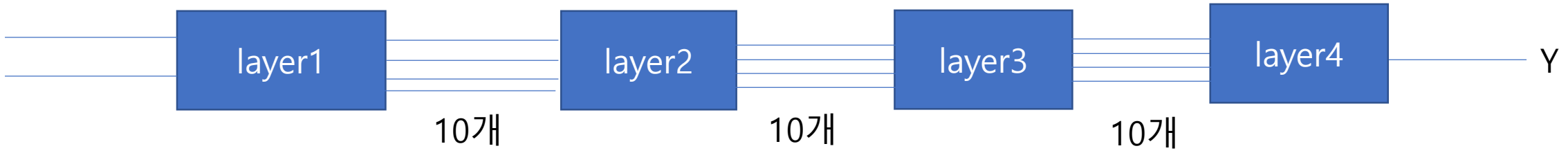


XOR을 위한 TF Deep Network(Wide & Deep)

```
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)

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

W1 = tf.Variable(tf.random_normal([2, 10]), name='weight1')
b1 = tf.Variable(tf.random_normal([10]), name='bias1')
layer1 = tf.sigmoid(tf.matmul(X, W1) + b1)

W2 = tf.Variable(tf.random_normal([10, 10]), name='weight2')
b2 = tf.Variable(tf.random_normal([10]), name='bias2')
layer2 = tf.sigmoid(tf.matmul(layer1, W2) + b2)

W3 = tf.Variable(tf.random_normal([10, 10]), name='weight3')
b3 = tf.Variable(tf.random_normal([10]), name='bias3')
layer3 = tf.sigmoid(tf.matmul(layer2, W3) + b3)

W4 = tf.Variable(tf.random_normal([10, 1]), name='weight4')
b4 = tf.Variable(tf.random_normal([1]), name='bias4')
hypothesis = tf.sigmoid(tf.matmul(layer3, W4) + b4)
```

```
Hypothesis:  [[0.00122806]
 [0.9988813 ]
 [0.99831235]
 [0.00222466]]
Correct:  [[0.]
 [1.]
 [1.]
 [0.]]
Accuracy:  1.0
```

결과가 더 좋음
(hypothesis : 작은 값은
더 작게 큰 값은 더 크게)



텐서보드 사용법

5 steps of using TensorBoard

1. From TF graph, decide which tensors you want to log

```
w2_hist = tf.summary.histogram("weights2", W2)
cost_summ = tf.summary.scalar("cost", cost)
```

2. Merge all summaries

```
summary = tf.summary.merge_all()
```

3. Create writer and add graph

```
# Create summary writer
writer = tf.summary.FileWriter('./logs')
writer.add_graph(sess.graph)
```

4. Run summary merge and add_summary

```
s, _ = sess.run([summary, optimizer], feed_dict=feed_dict)
writer.add_summary(s, global_step=global_step)
```

5. Launch TensorBoard

```
tensorboard --logdir=./logs
```

텐서보드 사용법

- activate tensorflow (Anaconda prompt 실행)
- Tensorboard –-logdir 디렉터리경로
- http://localhost:6006 접속
- 여러 개 그래프 그리기
  - Logs/디렉터리 안에 파일 여러 개 생성하기
- ValueError: Not a TBLoader or TBPlugin subclass:
  - pip uninstall tensorboard-plugin-wit