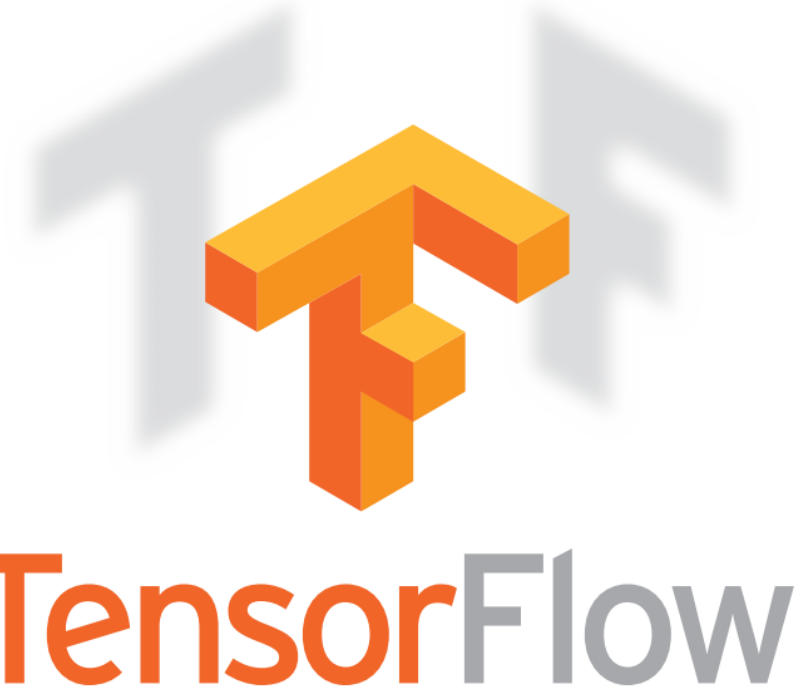
딥러닝 구현을 위한 텐서플로우 개발

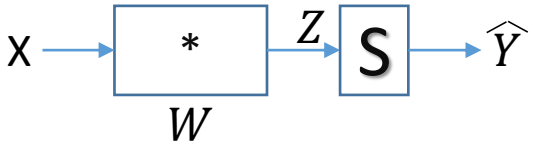


김성균

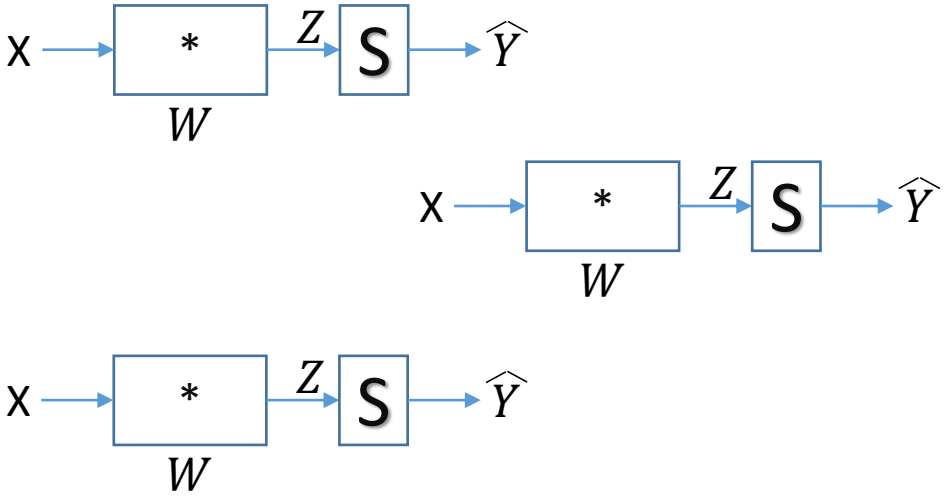
7강
Neural Network

01.XOR

- One logistic regression unit cannot separate XOR



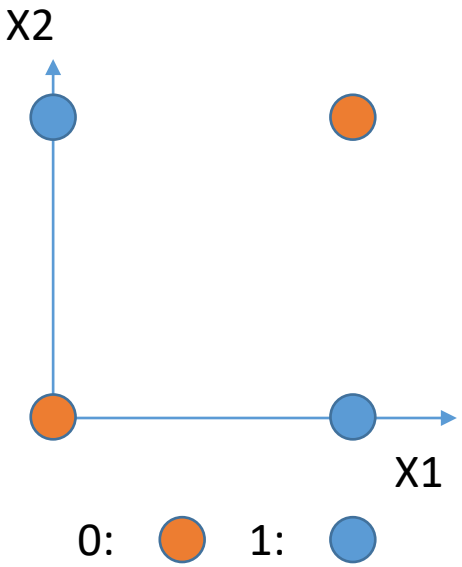
- Multiple logistic regression units



01.XOR

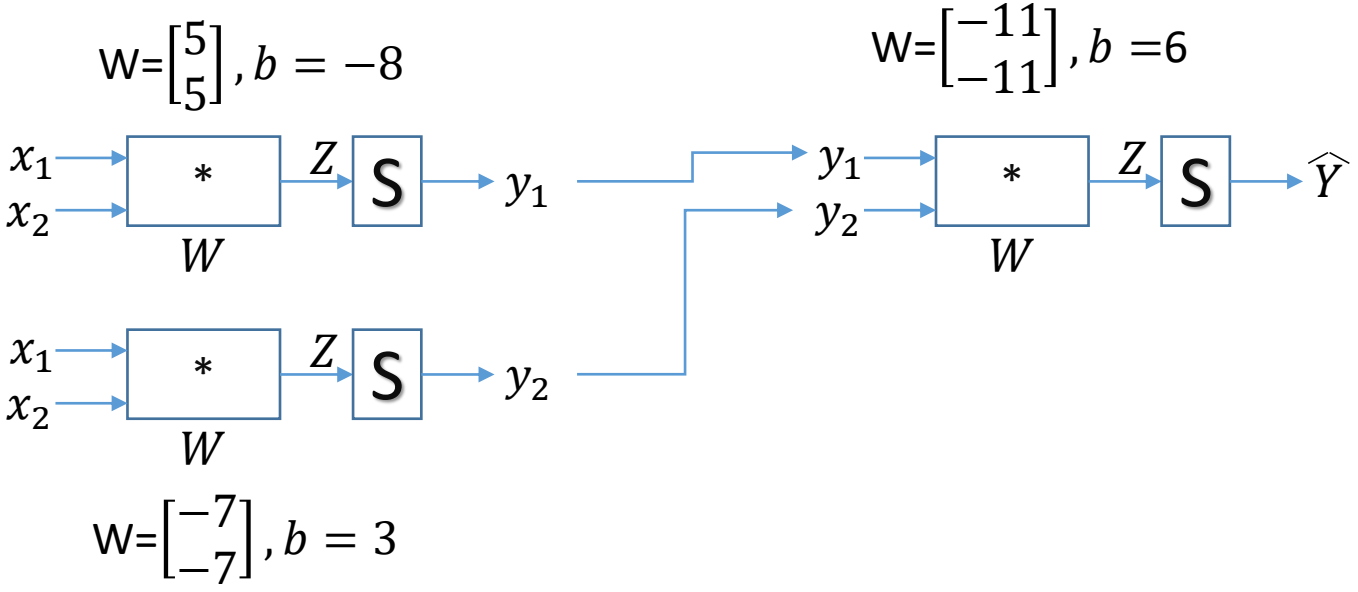
- No one on earth had found a viable way to train

X1	X2	XOR
0	0	0
0	1	1
1	0	1
1	1	0



```
import tensorflow as tf
import numpy as np

tf.set_random_seed(777)  # for reproducibility
learning_rate = 0.1

x_data = [[0, 0], [0, 1], [1, 0], [1, 1]]
y_data = [[0], [1], [1], [0]]
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])
Y = tf.placeholder(tf.float32, [None, 1])

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

W2 = tf.Variable(tf.random_normal([2, 1]), name='weight2')
b2 = tf.Variable(tf.random_normal([1]), name='bias2')
hypothesis = tf.sigmoid(tf.matmul(layer1, W2) + b2)

# cost/loss function
cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) *
                       tf.log(1 - hypothesis))

train =
tf.train.GradientDescentOptimizer(learning_rate=learning_rate).mini
mize(cost)

# Accuracy computation
# True if hypothesis>0.5 else False
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, Y),
dtype=tf.float32))

# Launch graph
with tf.Session() as sess:
    # Initialize TensorFlow variables
    sess.run(tf.global_variables_initializer())

    for step in range(10001):
        sess.run(train, feed_dict={X: x_data, Y: y_data})
        if step % 100 == 0:
            print(step, sess.run(cost, feed_dict={
                  X: x_data, Y: y_data}), sess.run([W1, W2]))

    # Accuracy report
    h, c, a = sess.run([hypothesis, predicted, accuracy],
                       feed_dict={X: x_data, Y: y_data})
    print("\nHypothesis: ", h, "\nCorrect: ", c, "\nAccuracy: ", a)
```

02. XOR with Neural Net



X1	X2	Y1	Y2	$\hat{Y}$	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

```
import tensorflow as tf
import numpy as np

tf.set_random_seed(777)  # for reproducibility
learning_rate = 0.1

x_data = [[0, 0],
          [0, 1],
          [1, 0],
          [1, 1]]
y_data = [[0],
          [1],
          [1],
          [0]]
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])
Y = tf.placeholder(tf.float32, [None, 1])

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

W2 = tf.Variable(tf.random_normal([2, 1]), name='weight2')
b2 = tf.Variable(tf.random_normal([1]), name='bias2')
hypothesis = tf.sigmoid(tf.matmul(layer1, W2) + b2)

# cost/loss function
cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) *
                       tf.log(1 - hypothesis))

train =
tf.train.GradientDescentOptimizer(learning_rate=learning_rate).m
inimize(cost)

# Accuracy computation
# True if hypothesis>0.5 else False
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, Y),
dtype=tf.float32))

# Launch graph
with tf.Session() as sess:
    # Initialize TensorFlow variables
    sess.run(tf.global_variables_initializer())

    for step in range(10001):
        sess.run(train, feed_dict={X: x_data, Y: y_data})
        if step % 100 == 0:
            print(step, sess.run(cost, feed_dict={
                  X: x_data, Y: y_data}), sess.run([W1, W2]))

    # Accuracy report
    h, c, a = sess.run([hypothesis, predicted, accuracy],
                       feed_dict={X: x_data, Y: y_data})
    print("\nHypothesis: ", h, "\nCorrect: ", c, "\nAccuracy: ",
a)
```

02. XOR with Neural Net

- 다항분류(multinomial classification)

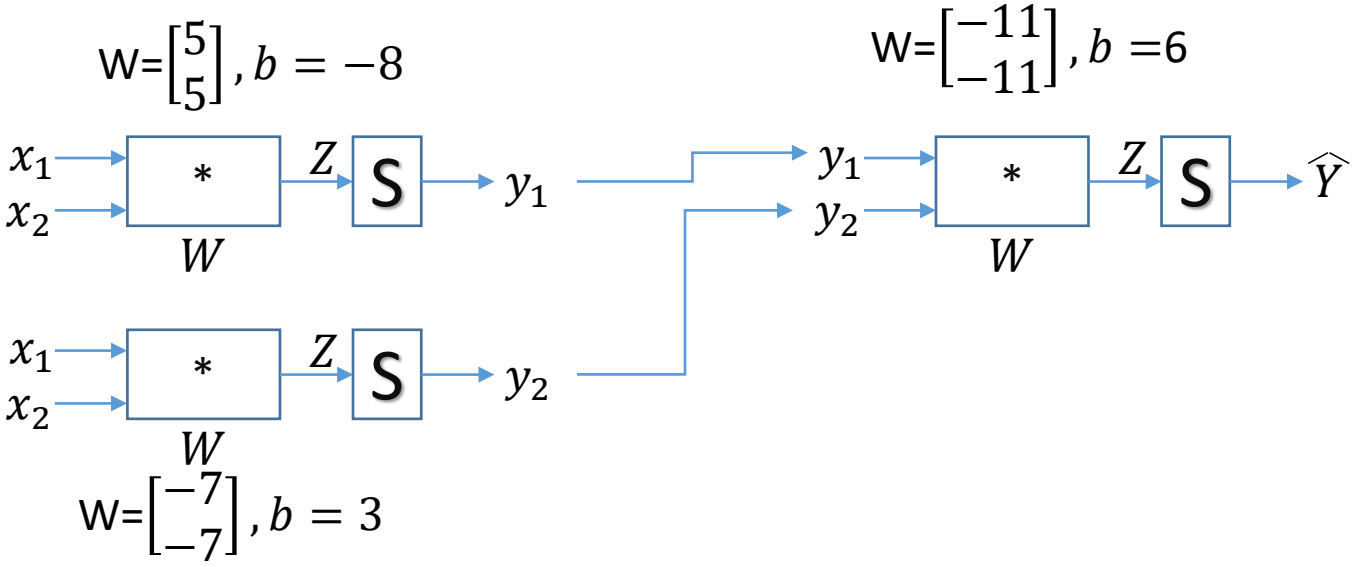
$$(w_{A1} \quad w_{A2} \quad w_{A3}) \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = (w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3)$$

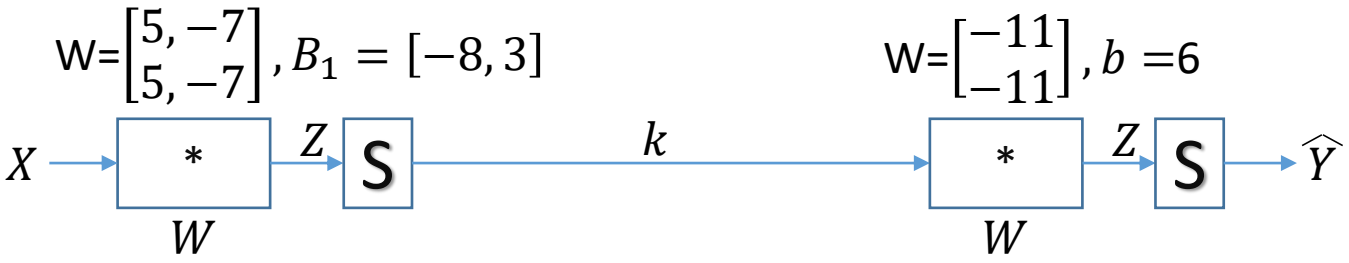
$$(w_{B1} \quad w_{B2} \quad w_{B3}) \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = (w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3)$$

$$(w_{C1} \quad w_{C2} \quad w_{C3}) \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = (w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3)$$

$$\begin{pmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A2}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B2}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C2}x_3 \end{pmatrix} = \begin{pmatrix} \widehat{Y_A} \\ \widehat{Y_B} \\ \widehat{Y_C} \end{pmatrix}$$

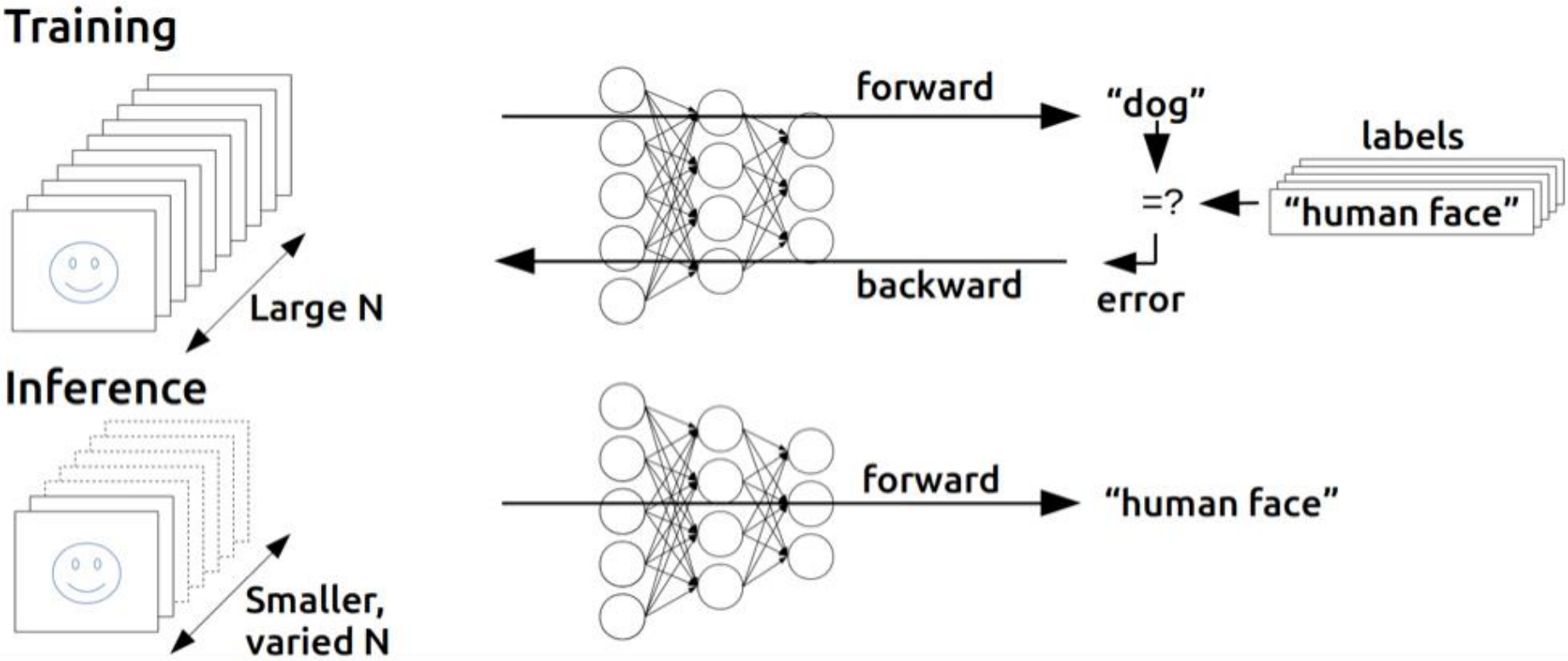
02. XOR with Neural Net





$$K(X) = sigmoid(XW_1 + B_1)$$
$$\hat{Y} = H(X) = sigmoid(K(X)W_2 + b_2)$$

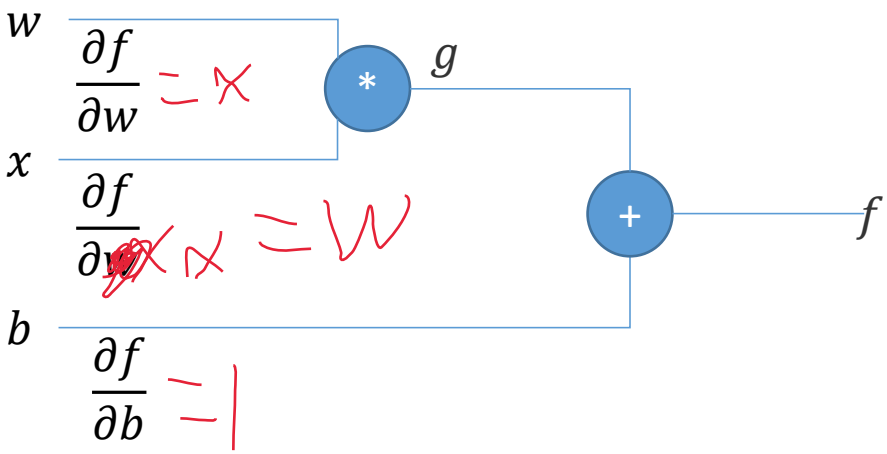
03.Back propagation



https://devblogs.nvidia.com/inference-next-step-gpu-accelerated-deep-learning/

03.Back propagation

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

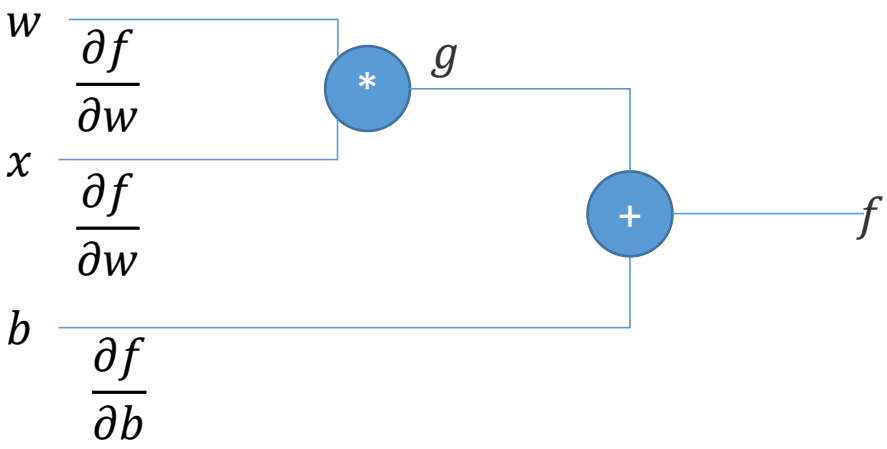


$w = -2, x = 5 \quad --> \quad g = -2 * 5 = -10$
$w = -1, x = 5 \quad --> \quad g = -1 * 5 = -5$ # w가 1 변할 때 $x(5)$만큼 변화
$w = -2, x = 6 \quad --> \quad g = -2 * 6 = -12$ # x가 1 변할 때 $w(-2)$만큼 변화

g를 x로 미분하면 w가 되고, w로 미분하면 x가 된다.

03.Back propagation

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



$g = -10, b = 3 \ \text{--->} \ f = -10 + 3 = -7$
$g = -9, b = 3 \ \text{--->} \ f = -9 + 3 = -6$ # g가 1 변할 때, 1만큼 변화
$g = -10, b = 4 \ \text{--->} \ f = -10 + 4 = -6$ # b가 1 변할 때, 1만큼 변화

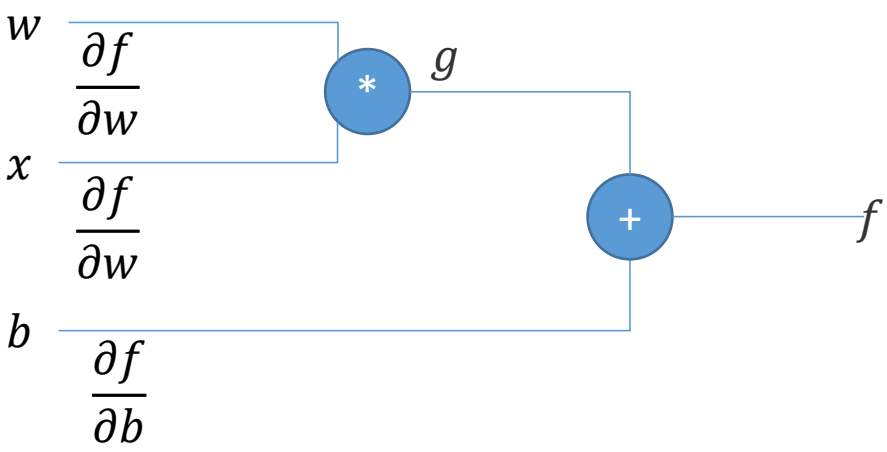
f를 g나 b로 미분하면 1이 된다.

03.Back propagation

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

$$\frac{\partial f}{\partial g} = 1, \frac{\partial f}{\partial b} = 1$$

$$\frac{\partial g}{\partial w} = x, \frac{\partial g}{\partial x} = w$$



Forward

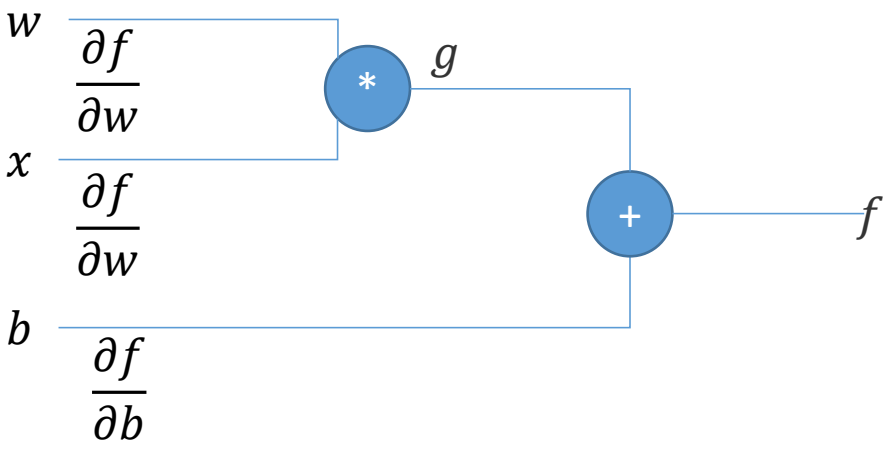
$w = -2, x = 5, b = 3 \ --> \ f = -2 * 5 + 3 = -7$

$w = -1, x = 5, b = 3 \ --> \ f = -1 * 5 + 3 = -2$ # w가 1 변할 때 5만큼 변화

$$\frac{\partial f}{\partial w}(f\text{를 } w\text{로 미분}) = \frac{\partial f}{\partial g}(f\text{를 } g\text{로 미분}) * \frac{\partial g}{\partial w}(g\text{를 } w\text{로 미분}) = 1 * x = x(5)$$

03.Back propagation

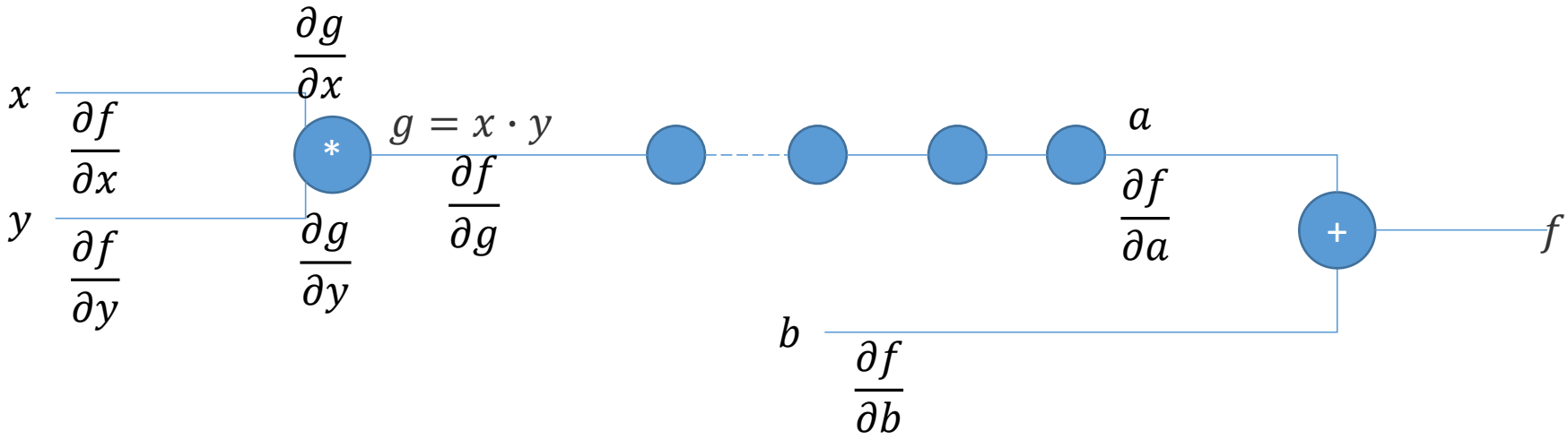
$$f = wx + b,\ g = wx,\ f = g + b$$



$w = -2, x = 5, b = 3 \text{ --> } f = -2 * 5 + 3 = -7$
$w = -2, x = 6, b = 3 \text{ --> } f = -2 * 6 + 3 = -9$ # x가 1 변할 때 -2만큼 변화

$$\frac{\partial f}{\partial w}(f\text{를 } w\text{로 미분}) = \frac{\partial f}{\partial g}(f\text{를 } g\text{로 미분}) * \frac{\partial g}{\partial w}(g\text{를 } w\text{로 미분}) = 1 * x = x(-2)$$

03.Back propagation(chain rule)



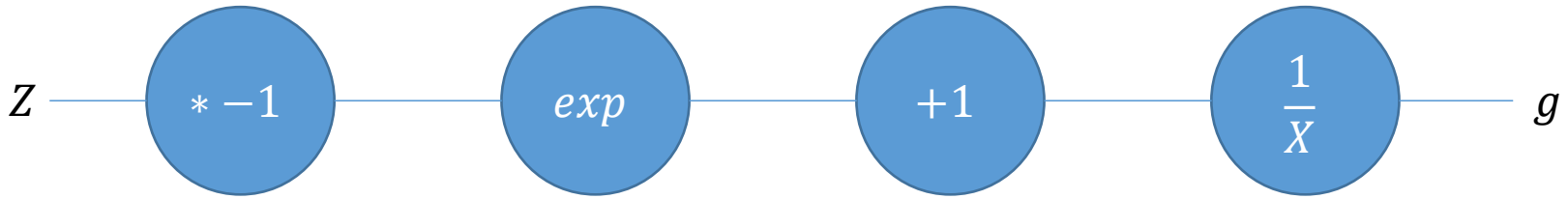
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

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

04. Sigmoid

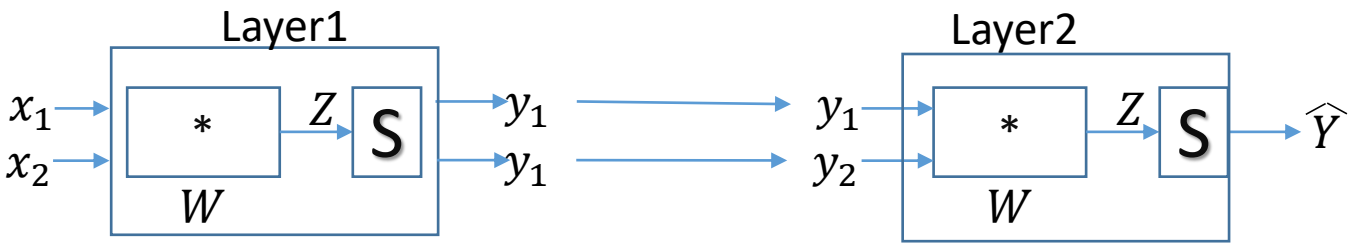
복잡한 수식도 기본미분값을 이용하여 풀 수 있음

$$g(z) = \frac{1}{1 + e^{-z}}$$



1. z에 -1을 곱해서 음수로 만든다.
2. exp(지수)를 계산한다.
3. 1을 더한다.
4. 앞의 결과를 분모로 취한다. (1/x)

04. XOR with Wide Neural Net



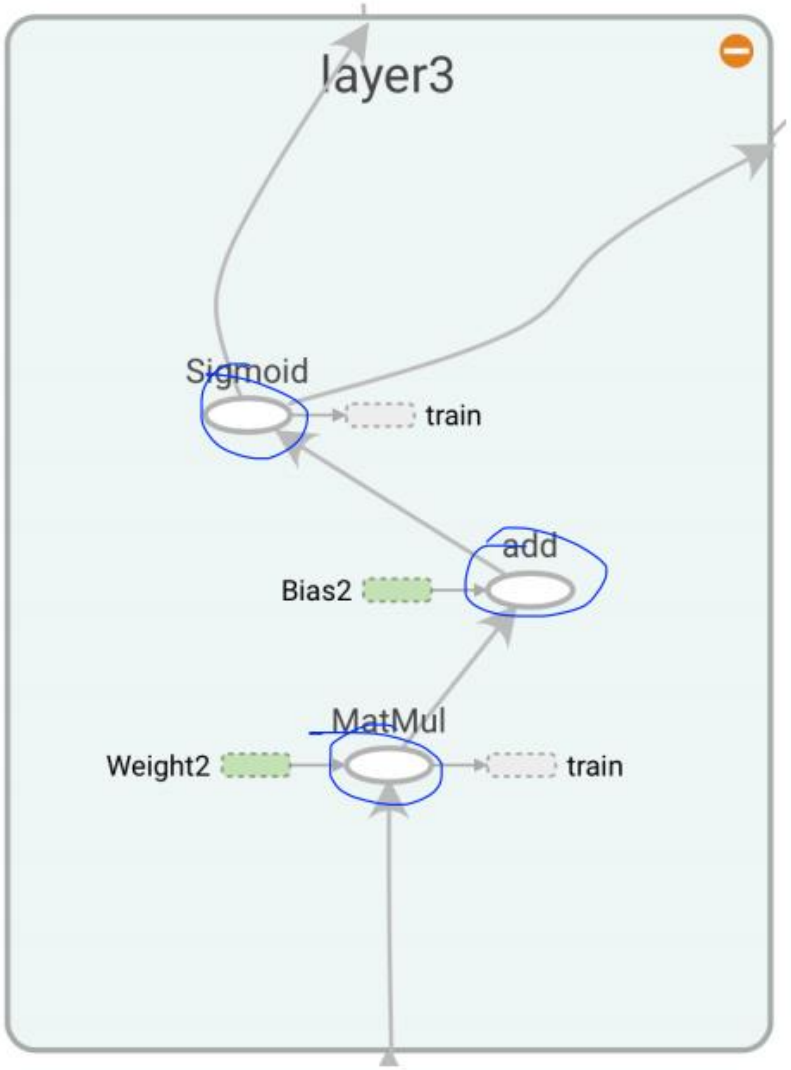
```
import tensorflow as tf
import numpy as np

tf.set_random_seed(777)  # for reproducibility
learning_rate = 0.1

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

# cost/loss function
cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) *
                       tf.log(1 - hypothesis))

train =
tf.train.GradientDescentOptimizer(learning_rate=learning_rate).m
inimize(cost)
```

```
# Accuracy computation
# True if hypothesis>0.5 else False
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, Y),
dtype=tf.float32))

# Launch graph
with tf.Session() as sess:
    # Initialize TensorFlow variables
    sess.run(tf.global_variables_initializer())

    for step in range(10001):
        sess.run(train, feed_dict={X: x_data, Y: y_data})
        if step % 100 == 0:
            print(step, sess.run(cost, feed_dict={
                  X: x_data, Y: y_data}), sess.run([W1, W2]))

    # Accuracy report
    h, c, a = sess.run([hypothesis, predicted, accuracy],
                       feed_dict={X: x_data, Y: y_data})
    print("\nHypothesis: ", h, "\nCorrect: ", c, "\nAccuracy: ",
a)
```

05. XOR with Wide Neural Net

```
import tensorflow as tf
import numpy as np

tf.set_random_seed(777)  # for reproducibility
learning_rate = 0.1

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

# cost/loss function
cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) *
                       tf.log(1 - hypothesis))

train =
tf.train.GradientDescentOptimizer(learning_rate=learning_rate).mini
mize(cost)

# Accuracy computation
# True if hypothesis>0.5 else False
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, Y),
dtype=tf.float32))

# Launch graph
with tf.Session() as sess:
    # Initialize TensorFlow variables
    sess.run(tf.global_variables_initializer())

    for step in range(10001):
        sess.run(train, feed_dict={X: x_data, Y: y_data})
        if step % 100 == 0:
            print(step, sess.run(cost, feed_dict={
                  X: x_data, Y: y_data}), sess.run([W1, W2]))

    # Accuracy report
    h, c, a = sess.run([hypothesis, predicted, accuracy],
                       feed_dict={X: x_data, Y: y_data})
    print("\nHypothesis: ", h, "\nCorrect: ", c, "\nAccuracy: ", a)
```

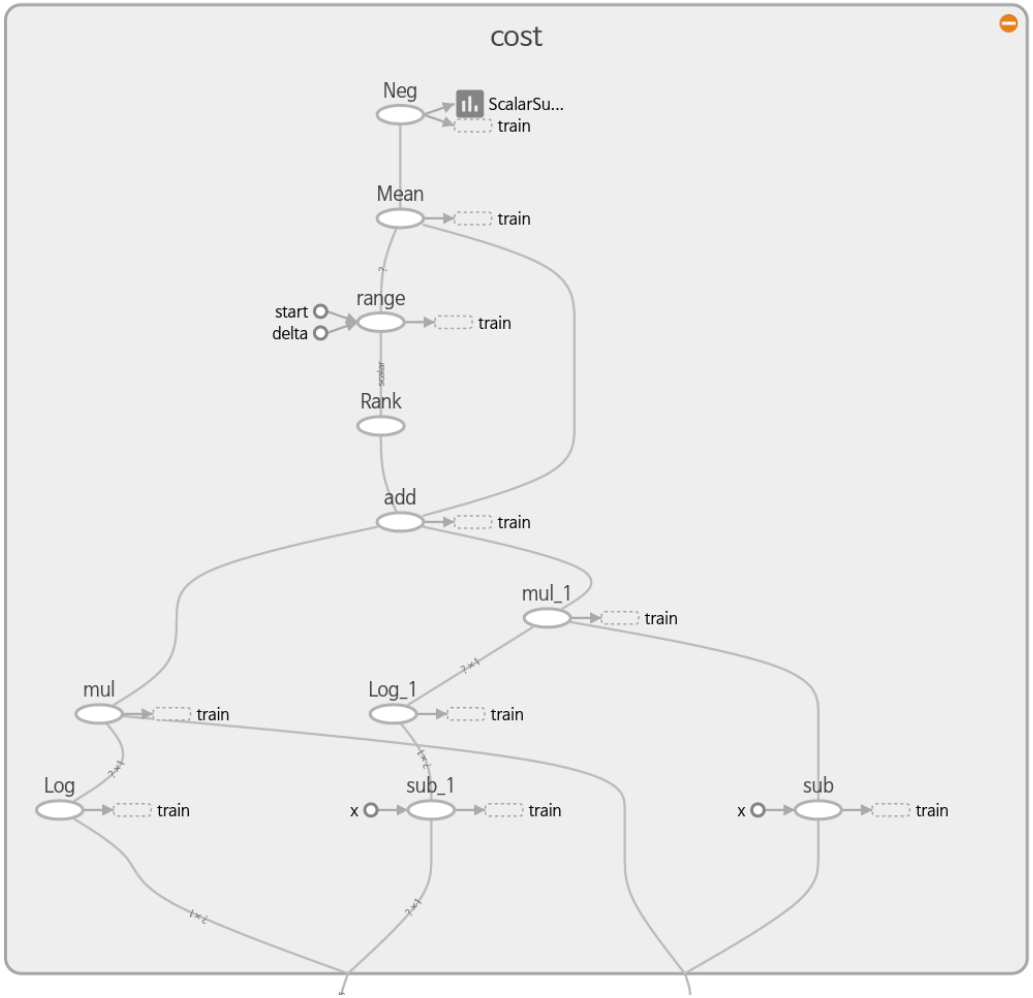
06. Back propagation in TensorFlow




```
hypothesis = tf.sigmoid(tf.matmul(L2, W2) + b2)
```

07. Back propagation in TensorFlow

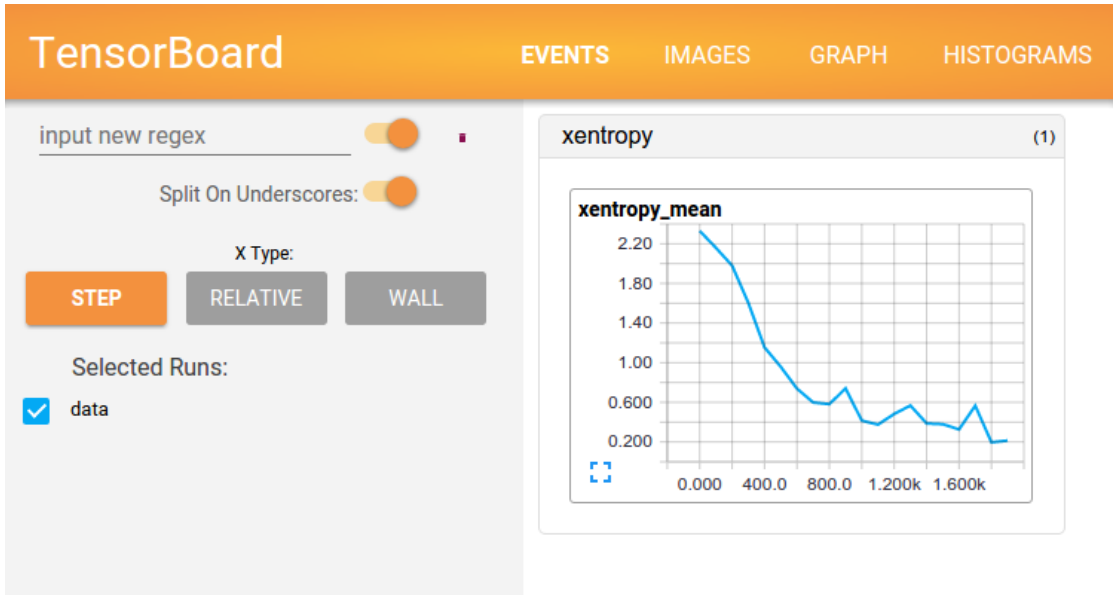
```
cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))
```



08. Tensorboard를 활용한 TensorFlow 시각화

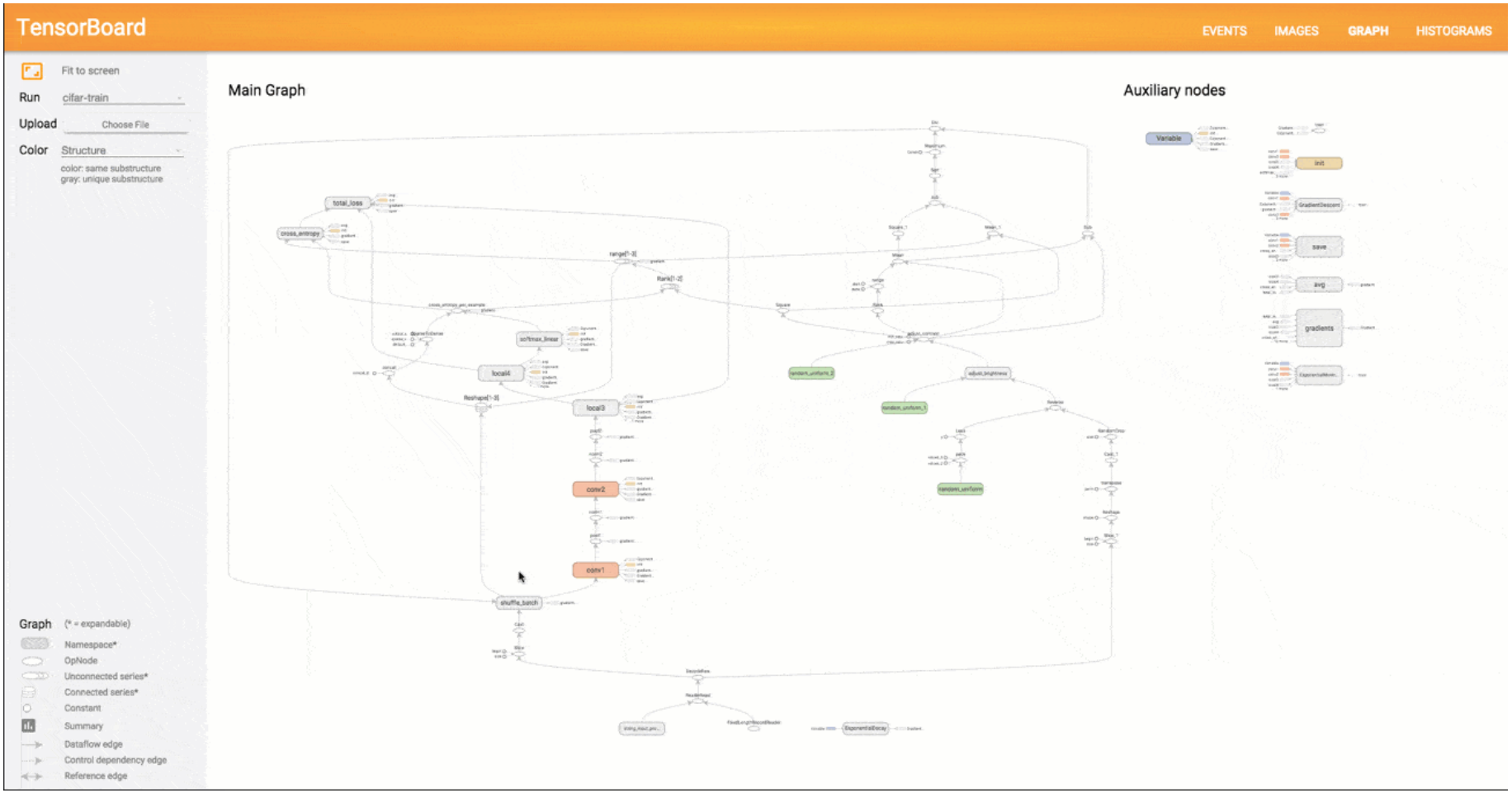
- TensorFlow에서 발생한 로그를 표시하거나 디버깅(debugging)을 하기 위한 도구
- 그래프를 그려서 통계를 시각화하는 것이 주요 기능

```
2000 [0.69364417, array([[ 0.50981331,  0.50592244],
       [ 0.37054271,  0.37088916],
       [ 0.6810087 ,  0.38607275],
       [ 0.54717511,  0.26581794]], dtype=float32), array([[ 0.50861073],
       [ 0.51602864],
       [ 0.4826754 ],
       [ 0.49036184]], dtype=float32), array([[ 0.71915275, -0.48754135],
       [-0.56914777, -0.55209494]], dtype=float32), array([[-0.44138899],
       [ 0.23536676]], dtype=float32), array([ 0.03925836,  0.02369077], dtype=float32), array([ 0.14039496], dtype=float32)]
4000 [0.69332385, array([[ 0.52235132,  0.50927138],
       [ 0.38598102,  0.37814924],
       [ 0.69650716,  0.39592981],
       [ 0.56881481,  0.27748841]], dtype=float32), array([[ 0.50748861],
       [ 0.51554251],
       [ 0.48338425],
       [ 0.49113813]], dtype=float32), array([[ 0.74125487, -0.45954311],
       [-0.55370271, -0.53450096]], dtype=float32), array([[-0.42565805],
       [ 0.19686614]], dtype=float32), array([ 0.08946501,  0.03708982], dtype=float32), array([ 0.15284136], dtype=float32)]
6000 [0.69306737, array([[ 0.53439337,  0.51197231],
       [ 0.39961013,  0.38383543],
       [ 0.71191686,  0.40380618],
       [ 0.58899951,  0.2868301 ]], dtype=float32), array([[ 0.50660294],
       [ 0.51538038],
```



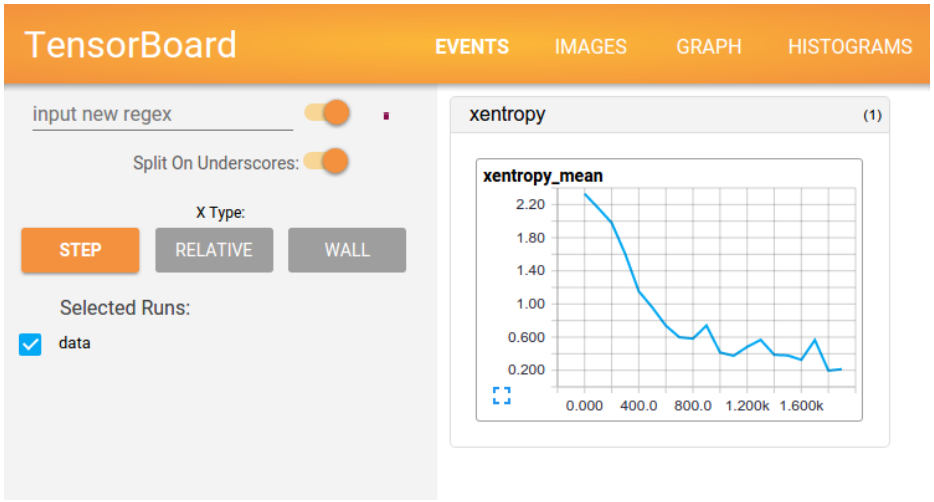
09. Tensorboard를 활용한 TensorFlow 시각화

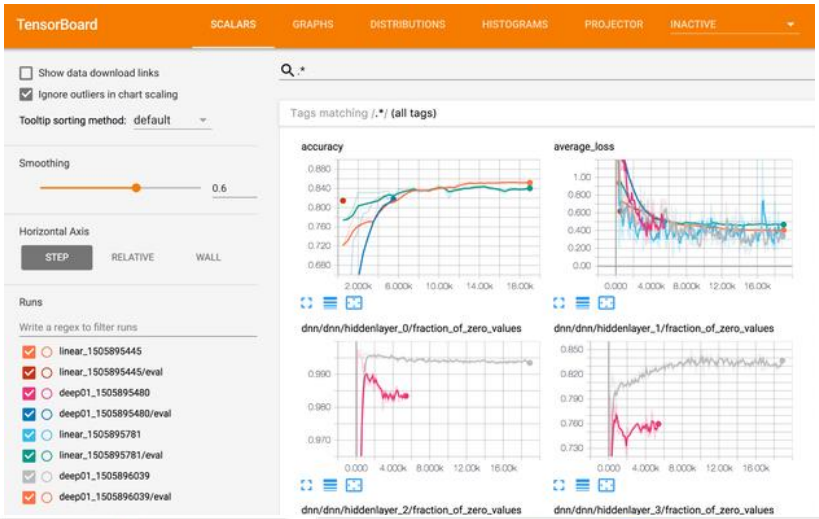
- TensorBoard: 그래프 시각화

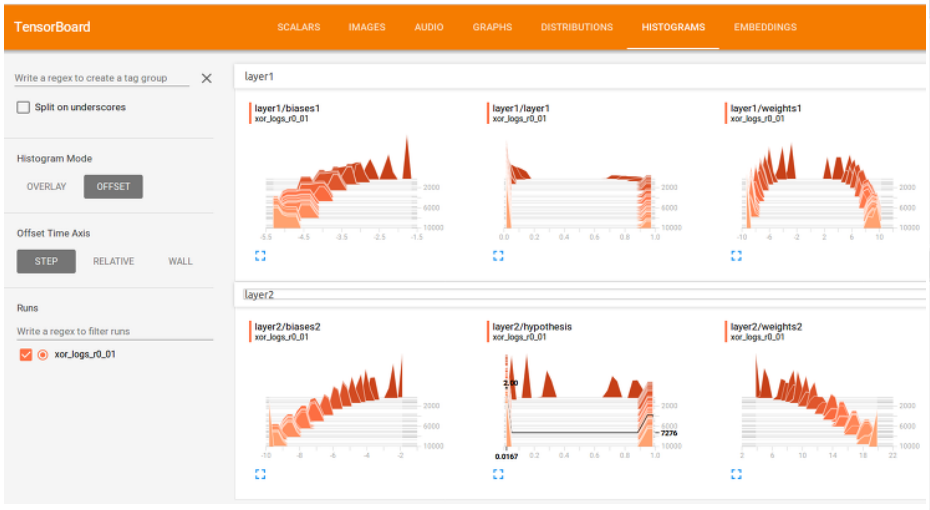


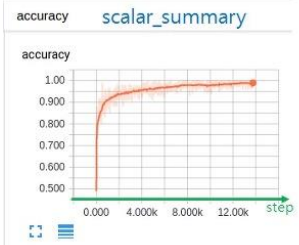
09. Tensorboard를 활용한 TensorFlow 시각화

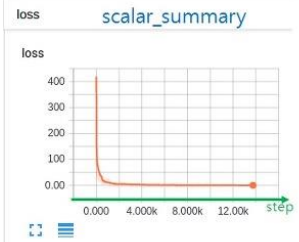
- TensorBoard: 학습 시각화

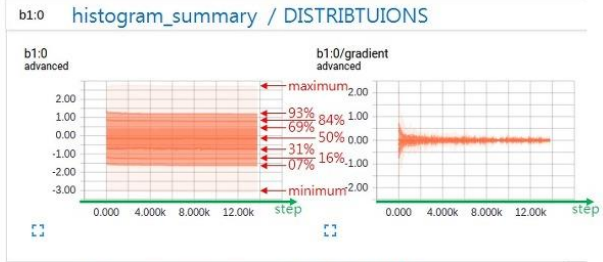


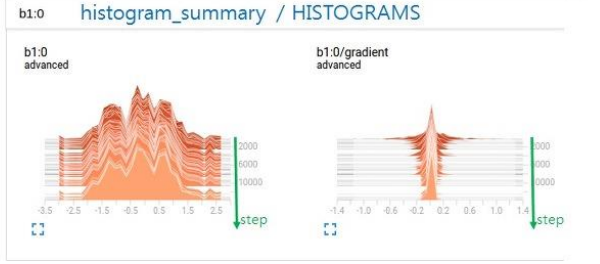












10. TensorBoard

- tensorboard --logdir=/tmp/sample
- 루트(/) 폴더 밑의 tmp 폴더 밑의 sample 폴더에 기록된 로그를 보겠다,라는 명령.
- logdir 뒤에는 로그가 기록된 폴더를 명시.

- 기록된 폴더는 소스 코드를 구동시킬 때 명시.
  - 소스 코드를 구동하지 않으면 로그 없음.
  - 소스 코드에 직접 관련 코드를 삽입

10. TensorBoard

- 명령의 tensorboard는파이썬 설치 폴더 아래의 tensorflow 폴더 밑에 있는 tensorboard.py 파일을 가리킨다.

  …/tensorflow/tensorboard/tensorboard.py

10. TensorBoard

- 포트 번호
  - tensorboard 기본 포트번호는 6006.
  - 필요에 따라 포트 번호를 바꿀 수 있음
    - tensorboard --logdir=/tmp/sample --port=8008

- tensorboard 명령을 실행할 때 port 옵션을 사용해서 포트 번호를 지정가능
- 동시에 여러 개의 로그를 보고 싶을 때 사용.

10. TensorBoard

- 로그 위치
  - 소스 코드에서 로그를 기록하기 위한 코드.
  - writer = tf.train.SummaryWriter("/tmp/test_logs", session.graph)

- 소스 코드를 구동하면 "/tmp/test_logs" 폴더에 확장자가 local인 파일이 생성.

10. TensorBoard

- 웹 브라우저
  - 웹 브라우저를 열고 콘솔에서 명령을 실행 후, 출력된 메시지의 ip 주소를 열면 그래프를 볼 수 있음
  - 입력 주소에는 두 가지가 있다.
    - 0.0.0.0:6006 또는 localhost:6006
- tensorboard 종료
  - 웹 브라우저는 단순히 로그를 시각화하는 역할
    웹 브라우저를 종료한다고 해서 tensorboard가 종료되는 것이 아님.
  - tensorboard 명령을 입력하면 해당 콘솔은 블록(대기) 상태
  - 종료하려면 ctrl+c를 입력

11. TensorBoard 사용 5단계

1. 선택

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

2. 머지

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

3. 기록

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

4. 실행

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

5. Launch TensorBoard

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

12. TensorBoard with mnist

```
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

import tensorflow as tf

with tf.name_scope("input") as scope:
    x = tf.placeholder(tf.float32, [None, 784])

with tf.name_scope("weight") as scope:
    W = tf.Variable(tf.zeros([784, 10]))

with tf.name_scope("bias") as scope:
    b = tf.Variable(tf.zeros([10]))

with tf.name_scope("layer1") as scope:
    y = tf.nn.softmax(tf.matmul(x, W) + b)

w_hist = tf.summary.histogram("weight", W)

b_hist = tf.summary.histogram("bias", b)

y_hist = tf.summary.histogram("y", y)

with tf.name_scope("y_") as scope:
    y_ = tf.placeholder(tf.float32, [None, 10])

with tf.name_scope("cost") as scope:
    cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y),
reduction_indices=[1]))

    cost_sum = tf.summary.scalar("cost", cross_entropy)
```

```
with tf.name_scope("train") as scope:
    train_step =
tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)

with tf.Session() as sess:
    merged = tf.summary.merge_all()

    writer = tf.summary.FileWriter("./board/mnist", sess.graph)

init = tf.global_variables_initializer()

sess.run(init)

for i in range(1000):

    batch_xs, batch_ys = mnist.train.next_batch(100)

    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

    if i % 10 == 0:
        summary = sess.run(merged, feed_dict={x: batch_xs, y_:
batch_ys})

        writer.add_summary(summary, i)

correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))

accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_:
mnist.test.labels}))
```