

Golbin Tensorflow : chapter 04 ~ 06

Week3. Deep Neural Network (with mnist dataset)

2018.03.30
Made by jb.Park



P A P A T A F A C T O R Y D E S I G N

w w w . p a p a t a f a c t o r y . c o m

S H I N P R O P P T T E M P L A T E

COPYRIGHT © ALL RIGHT RESERVED BY PAPATAFACTORY

I N D E X

01. Mnist dataset

- Data structure
- Load data

03. Save & Load

- Save
- Load

02. DNN

- Input layer
- Hidden layer
- Dropout
- Output layer
- Cost function & Optimizer
- Learning
- Evaluate

04. TensorBoard

- Scalar
- Histogram
- Name scope
- Write file
- Web server

05. Word2Vec

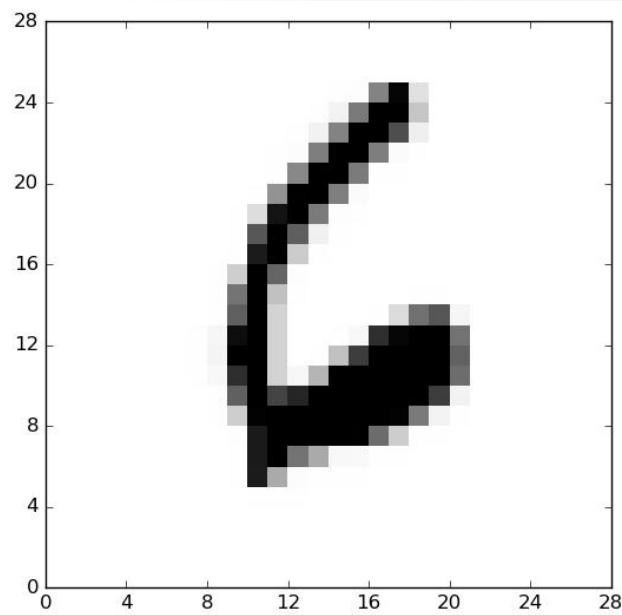
- Text embedding

01. Mnist dataset

- Data structure



MNIST 데이터베이스 (Modified [National Institute of Standards and Technology](#) database)는 손으로 쓴 숫자들로 이루어진
대형 데이터베이스



01. Mnist dataset

- Load data



Tensorflow에 기본적으로 내장되어 있는 mnist 실습을 위한 모듈을 사용

<Load data>

```
from tensorflow.examples.tutorials.mnist import input_data  
# 텐서플로우에 기본 내장된 mnist 모듈을 이용하여 데이터를 로드합니다.  
# 지정한 폴더에 MNIST 데이터가 없는 경우 자동으로 데이터를 다운로드합니다.  
# one_hot 옵션은 레이블을 동물 분류 예제에서 보았던 one_hot 방식의 데이터로 만들어줍니다.  
mnist = input_data.read_data_sets("./mnist/data/", one_hot=True)
```

<split data for batch>

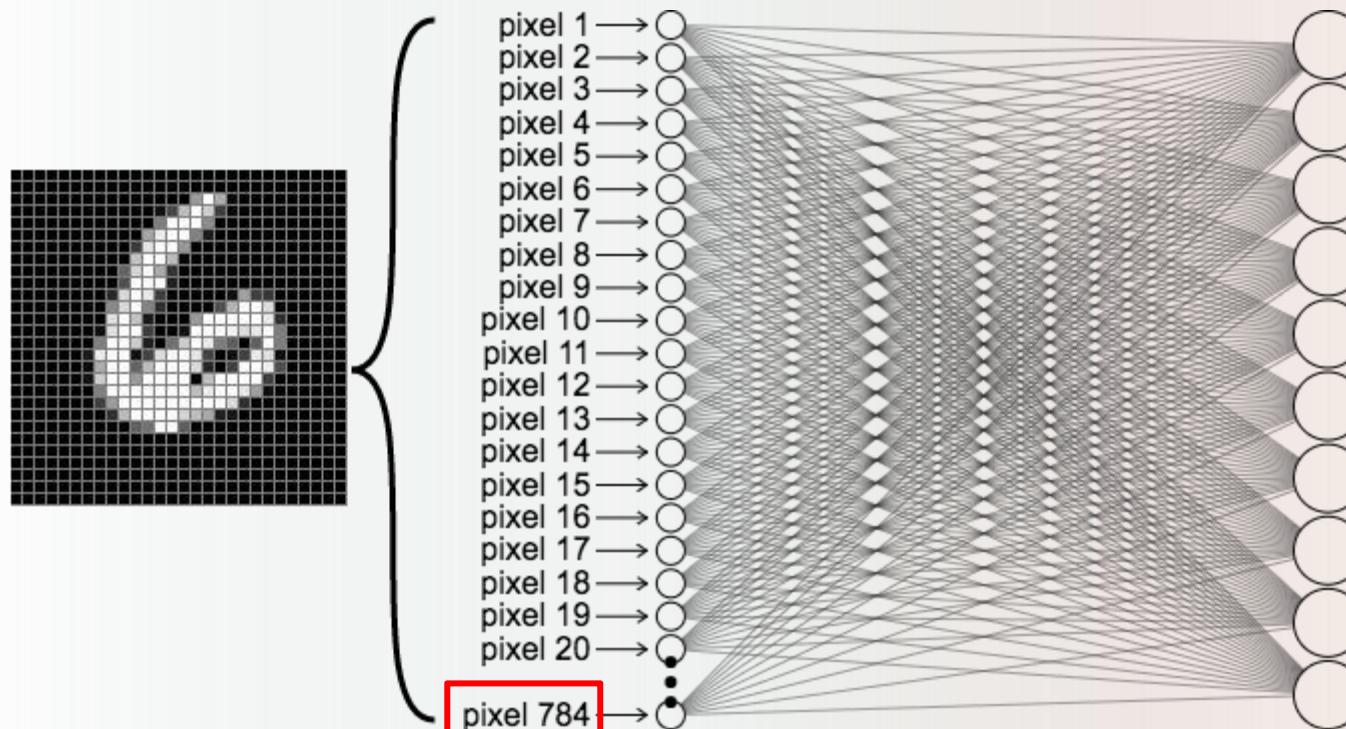
```
# 텐서플로우의 mnist 모델의 next_batch 함수를 이용해  
# 지정한 크기만큼 학습할 데이터를 가져옵니다.  
batch_xs, batch_ys = mnist.train.next_batch(batch_size)
```

02. DNN classifier

- Input Layer

$$1 \text{ image pixels} = 28 * 28 * 1 = 784$$

784개의 픽셀 값이 입력 레이어



02. DNN classifier

- Input Layer



Placeholder 함수를 통해 입력 데이터의 크기를 지정

X -> None: 이미지 개수

784 : 픽셀 개수

Y -> None : 이미지 개수

10 : 레이블 개수 (0~9)

입력 값의 차원은 [배치크기, 특성값] 으로 되어 있습니다.

손글씨 이미지는 28x28 픽셀로 이루어져 있고, 이를 784개의 특성값으로 정합니다.

X = tf.placeholder(tf.float32, [None, 784])

결과는 0~9 의 10 가지 분류를 가집니다.

Y = tf.placeholder(tf.float32, [None, 10])

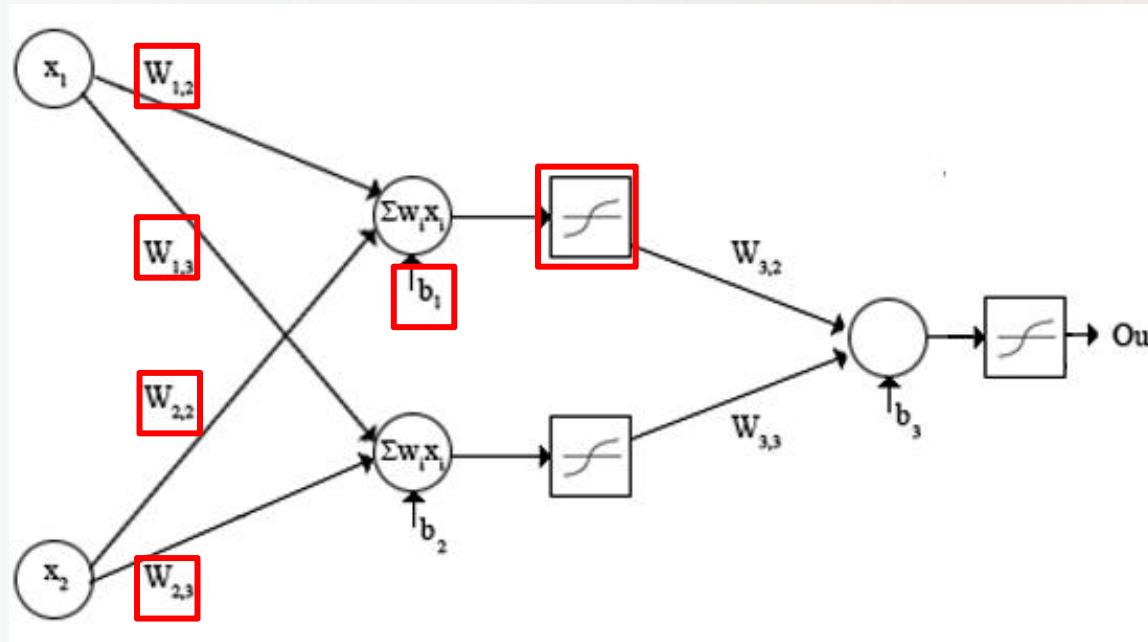
02. DNN classifier

- Hidden Layer

Hidden Layer 는

1. Weight
2. Bias
3. Activation function

으로 구성



02. DNN classifier

- Hidden Layer



Weight & Bias & Activation

```
# 신경망의 레이어는 다음처럼 구성합니다.  
# 784(입력 node 수), 256(출력 node 수)  
W1 = tf.Variable(tf.random_normal([784, 256], stddev=0.01))  
# 편향(바이어스)을 하든 레이어의 node 개수로 설정합니다.  
b1 = tf.Variable(tf.random_normal([256], -1, 1,))  
# 입력값에 가중치를 곱하고 편향을 더한 다음,  
# Activation function(ReLU)을 이용하여 레이어를 만듭니다.  
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)
```

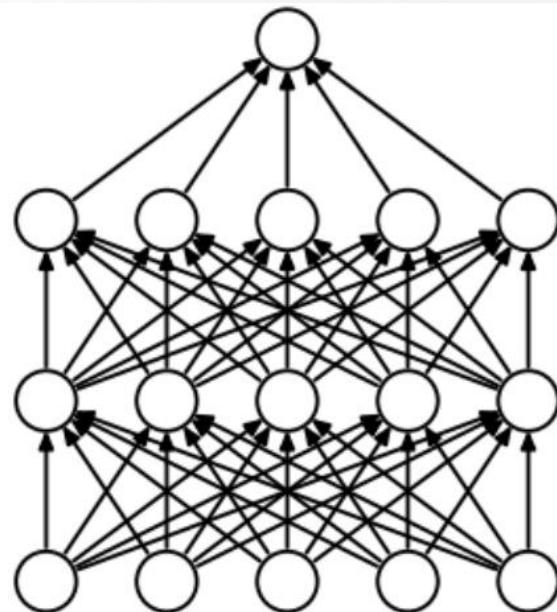
02. DNN classifier

- Dropout

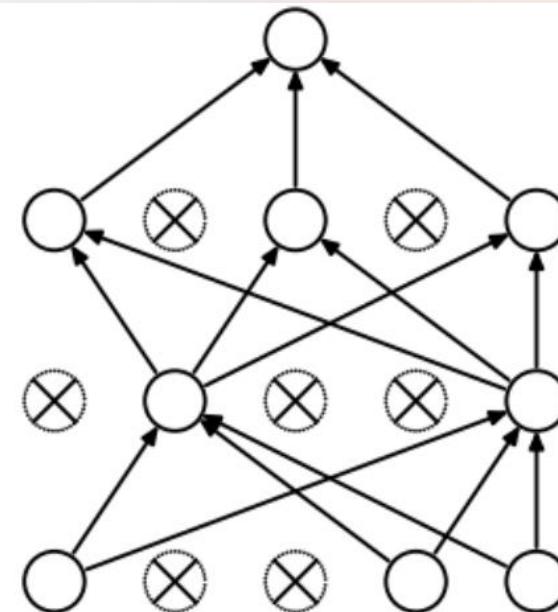


Overfitting을 방지하기 위한 기법

학습할 때, 일부의 노드만 사용한다



(a) Standard Neural Net



(b) After applying dropout.

02. DNN classifier

- Dropout



Overfitting을 방지하기 위한 기법

학습할 때, 일부의 노드만 사용한다

```
#hidden layer의 노드들 중 학습 시 유지시킬 변수 선언  
keep_prob = tf.placeholder(tf.float32)  
  
#hidden layer  
W1 = tf.Variable(tf.random_normal([784, 256], stddev=0.01))  
b1 = tf.Variable(tf.random_normal([256], -1, 1, ))  
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)  
  
#Drop out  
L1 = tf.nn.dropout(L1, keep_prob)
```

02. DNN classifier

- Output Layer



마지막 레이어의 노드 수는 데이터셋의 레이블 수와 같아야 한다

마지막 활성함수의 형태가 다름

```
#output layer  
#마지막 레이어의 노드 수는 레이블 수(10)와 같아야 함  
W4 = tf.Variable(tf.random_normal([256, 10], stddev=0.01))  
b4 = tf.Variable(tf.random_normal([10], -1, 1, ))  
L4 = tf.matmul(L3, W4) + b4  
model = tf.nn.softmax(L4)
```

02. DNN classifier

- Cost function & Optimizer



데이터가 모델을 거쳐 나온 결과를 참고해
모델을 개선하기 위한 함수

- Cross entropy 방법

$$H(p, q) = - \sum_x p(x) \log q(x).$$

신경망을 최적화하기 위한 비용 함수를 작성합니다.
각 개별 결과에 대한 합을 구한 뒤 평균을 내는 방식을 사용합니다.
전체 합이 아닌, 개별 결과를 구한 뒤 평균을 내는 방식을 사용하기 위해 axis 옵션을 사용합니다.
axis 옵션이 없으면 -1.09 처럼 총합인 스칼라값으로 출력됩니다.

```
#         Y      model      Y * tf.log(model)  reduce_sum(axis=1)
# 예) [[1 0 0]  [[0.1 0.7 0.2]  -> [[-1.0  0    0]  -> [-1.0, -0.09]
#      [0 1 0]]  [0.2 0.8 0.0]]      [ 0   -0.09 0]]
# 즉, 이것은 예측값과 실제값 사이의 확률 분포의 차이를 비용으로 계산한 것이며,
# 이것을 Cross-Entropy 라고 합니다.
```

```
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(model), axis=1))
```

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train_op = optimizer.minimize(cost)
```

02. DNN classifier

- Cost function & Optimizer



하지만 굳이 softmax를 써서 결과를 보지 않아도 됨
-> 어차피 제일 큰 값을 볼꺼니까

그래서 코드 축약 가능

```
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=model, labels=Y))  
optimizer = tf.train.AdamOptimizer(0.001).minimize(cost)
```

데이터와 모델을 거쳐서 나온 결과(코스트)를 가지고 코스트가 줄어드는 방향으로 학습이 필요

이를 계산 하기 위해서 optimizer 사용

02. DNN classifier

- Learning



Train 데이터를 model에 여러 번 학습시키는 과정

1 Epoch : 전체 training 데이터를 1번 학습.

20 Batch size : training 데이터를 20개씩 쪼개서 학습.

```
#####
# 신경망 모델 학습
#####

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

batch_size = 100
total_batch = int(mnist.train.num_examples / batch_size)
```

```
for epoch in range(30):
    total_cost = 0

    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)

        _, cost_val = sess.run([optimizer, cost],
                              feed_dict={X: batch_xs,
                                         Y: batch_ys,
                                         keep_prob: 0.8})

        total_cost += cost_val

    print('Epoch:', '%04d' % (epoch + 1),
          'Avg. cost =', '{:.3f}'.format(total_cost / total_batch))
```

02. DNN classifier

- Evaluate



학습이 완료된 모델을 test 데이터를 사용해서
정확도 확인

```
#####
# 결과 확인
#####

is_correct = tf.equal(tf.argmax(model, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
print('정확도:', sess.run(accuracy,
                       feed_dict={X: mnist.test.images,
                                  Y: mnist.test.labels,
                                  keep_prob: 1}))
```

```
#####
# 결과 확인 (matplotlib)
#####

labels = sess.run(model,
                  feed_dict={X: mnist.test.images,
                             Y: mnist.test.labels,
                             keep_prob: 1})

fig = plt.figure()
for i in range(10):
    subplot = fig.add_subplot(2, 5, i + 1)
    subplot.set_xticks([])
    subplot.set_yticks([])
    subplot.set_title('%d' % np.argmax(labels[i]))
    subplot.imshow(mnist.test.images[i].reshape((28, 28)),
                   cmap=plt.cm.gray_r)

plt.show()
```

03. Save & Load

-Save



Saver를 만들고,
학습 후에 저장

```
# 학습에 직접적으로 사용하지 않고 학습 횟수에 따라 단순히 증가시킬 변수를 만듭니다.
```

```
global_step = tf.Variable(0, trainable=False, name='global_step')
```

```
# 모델을 저장하고 불러오는 API를 초기화합니다.
```

```
# global_variables 함수를 통해 앞서 정의하였던 변수들을 저장하거나 불러올 변수들로 설정합니다.
```

```
saver = tf.train.Saver(tf.global_variables())
```

```
# 최적화가 끝난 뒤, 변수를 저장합니다.
```

```
saver.save(sess, './model/dnn.ckpt', global_step=global_step)
```

03. Save & Load

- Load



학습 진행 전에,

저장된 웨이트 값을 불러올 수 있다

```
ckpt = tf.train.get_checkpoint_state('./model')
if ckpt and tf.train.checkpoint_exists(ckpt.model_checkpoint_path):
    saver.restore(sess, ckpt.model_checkpoint_path)
else:
    sess.run(tf.global_variables_initializer())
```

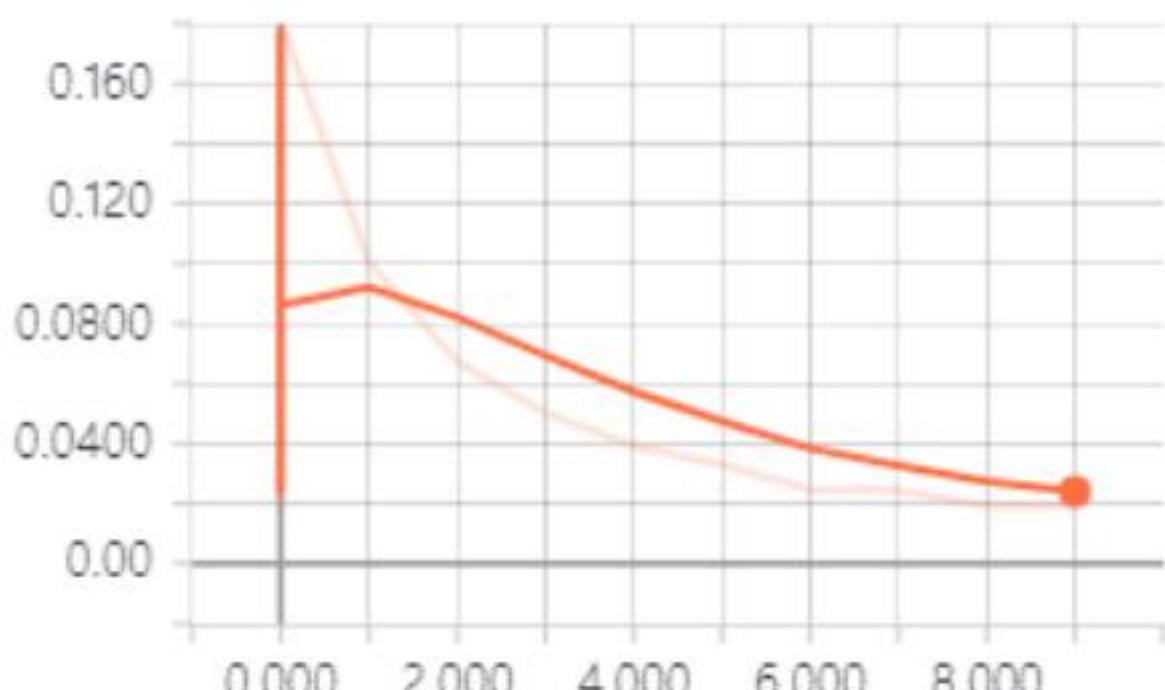
04. TensorBoard

-Scalar



코스트 값을 꺾은선 그래프로 표현 가능

```
tf.summary.scalar('cost', cost)
```



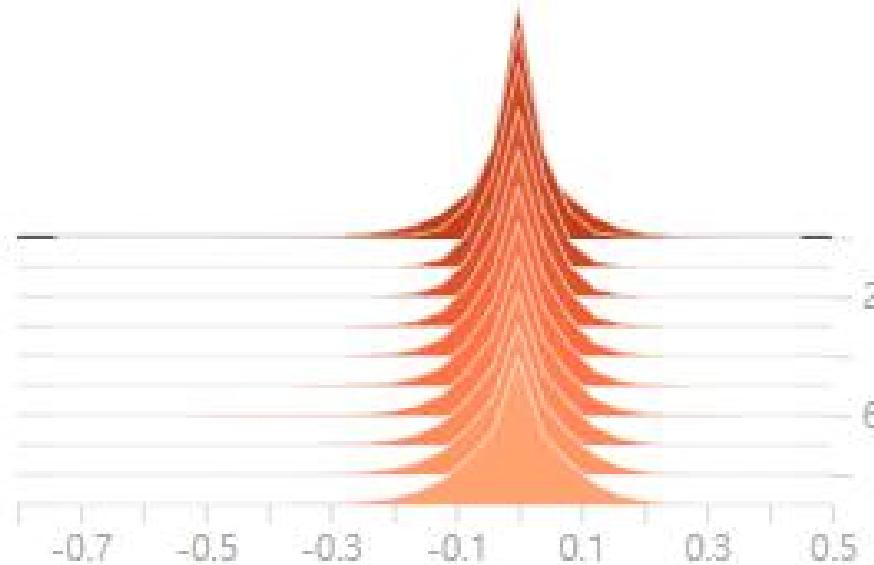
04. TensorBoard

- Histogram

웨이트 값을 히스토그램으로 표현

```
tf.summary.histogram("Weights", w3)  
tf.summary.histogram("Model", model)
```

layer1/Weights

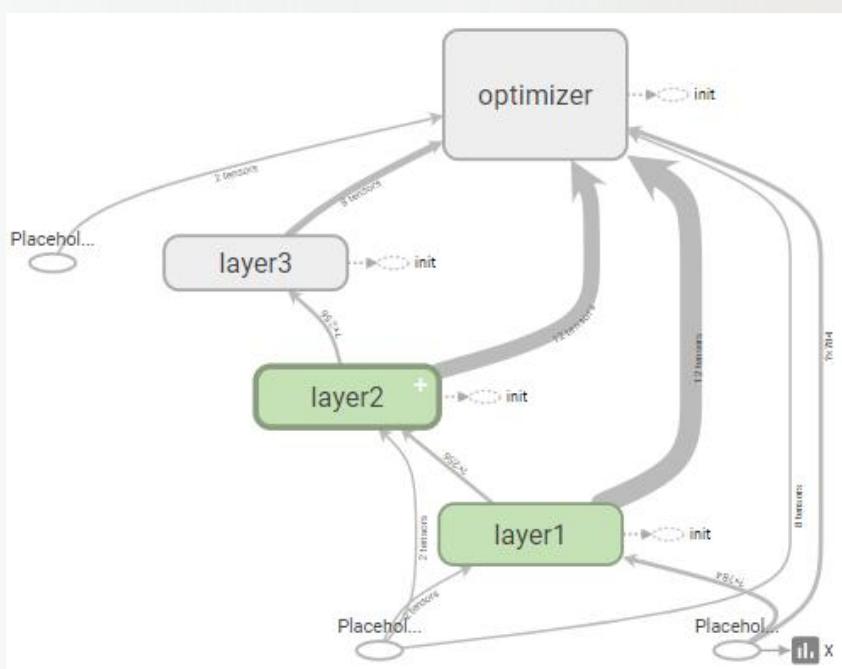


04. TensorBoard

- Name scope

여러 Tensor를 묶어서 그래프로 표현

```
with tf.name_scope('output'):
    W3 = tf.Variable(tf.random_uniform([20, 3], -1., 1.), name='W3')
    model = tf.matmul(L2, W3)
```



04. TensorBoard

- Write file



Summary 한 것들을 합치고(merge)
File에 write 하는 과정이 필요

```
merged = tf.summary.merge_all()  
writer = tf.summary.FileWriter('./logs', sess.graph)  
  
for step in range(100):  
    sess.run(train_op, feed_dict={X: x_data, Y: y_data})  
  
    print('Step: %d, ' % sess.run(global_step),  
          'Cost: %.3f' % sess.run(cost, feed_dict={X: x_data, Y: y_data}))  
  
    summary = sess.run(merged, feed_dict={X: x_data, Y: y_data})  
    writer.add_summary(summary, global_step=sess.run(global_step))
```

04. TensorBoard

- Web server



웹페이지를 통해 history 확인 가능

텐서보드에서 표시해주기 위한 텐서들을 수집합니다.

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

저장할 그래프와 텐서값들을 저장할 딕렉토리를 설정합니다.

```
writer = tf.summary.FileWriter('./logs', sess.graph)
```

이렇게 저장한 로그는, 학습 후 다음의 명령어를 이용해 웹서버를 실행시킨 뒤

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

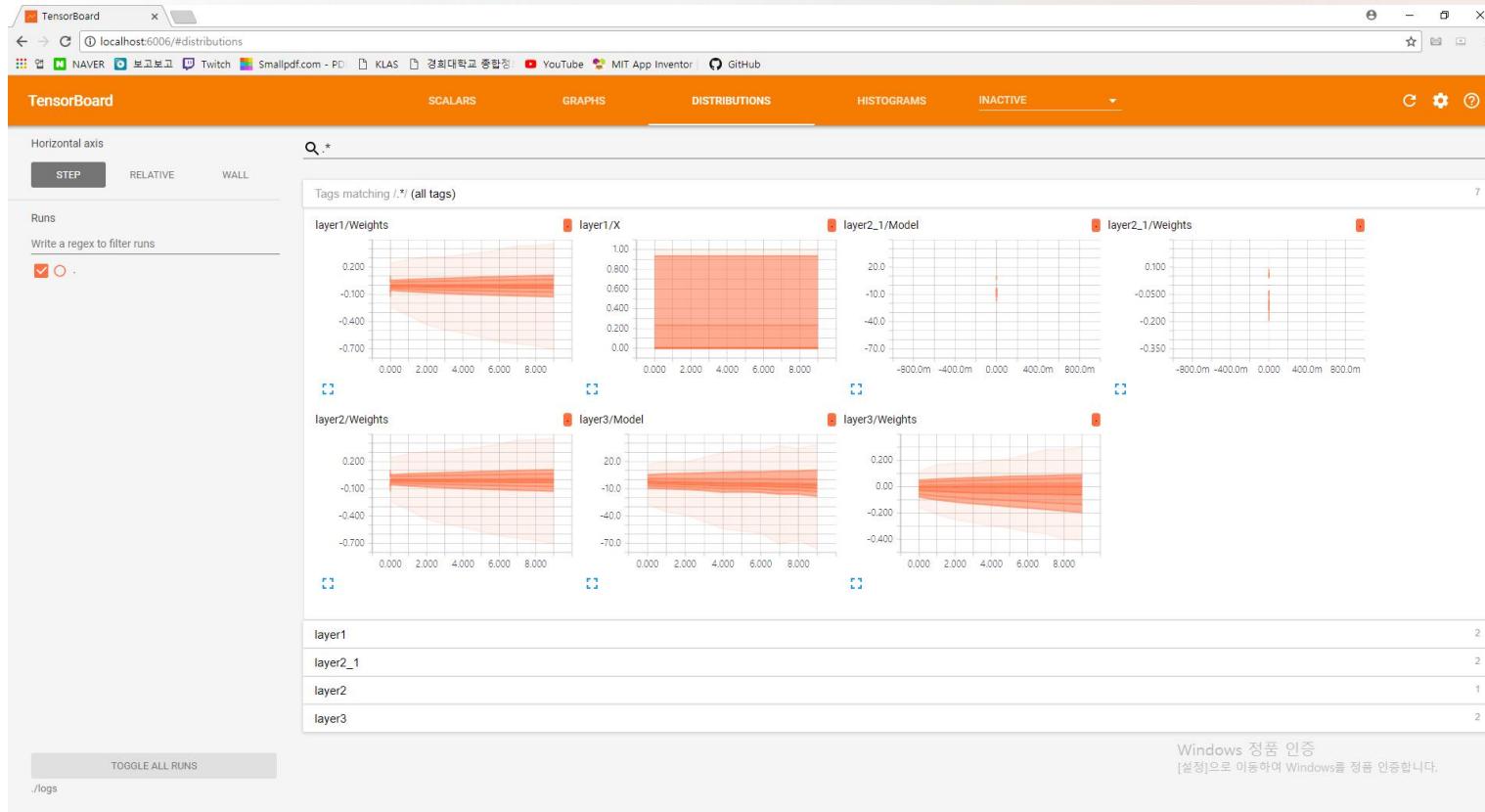
다음 주소와 웹브라우저를 이용해 텐서보드에서 확인할 수 있습니다.

```
# http://localhost:6006
```

04. TensorBoard

- Web server

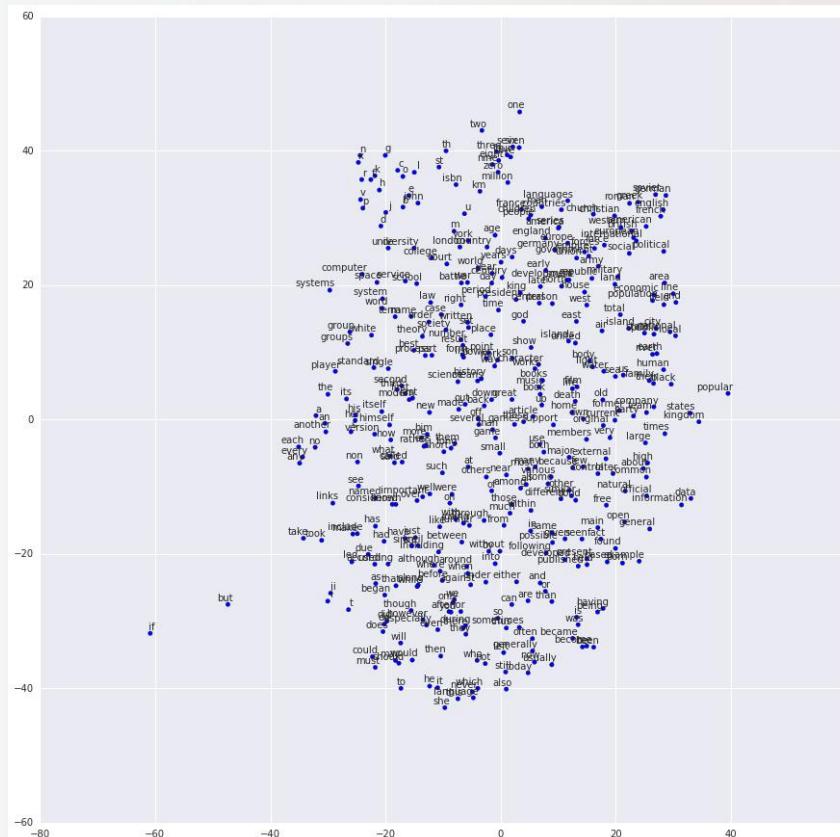
<http://localhost:6006/>



05. Word2Vec

- Text embedding

단어를 벡터로 표현하는 것
Ex) Apple -> [0, 0.3, 0.1, 1.2, 0, 3]





Thank you

SHINPRO DESIGN

COPYRIGHT © ALL RIGHT RESERVED BY PAPATAFACTORY