Tensorflow Tutorial

Natural Language Processing & AI Lab.,
Korea University

INDEX

Tensorflow Concepts

What is Tensorflow?

- Data Flow Graph
  - 노드와 노드들을 연결하는 “엣지”로 구성된 트리와 유사한 구조
  - Tensorflow에서 노드는 하나의 “operation” 덧셈, 곱셈과 같은 연산을 의미
  - 엣지는 노드와 노드 간에 주고 받는 다차원 데이터 배열 (Multi-dimensional data arrays)를 의미
  - Tensor가 노드와 노드 사이를 흘러(Flow) 다니면서 학습이 이루어짐

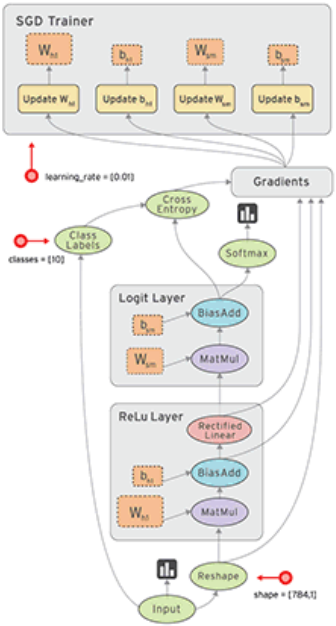
Graphs

- Nodes (노드)
  - constants (상수)
  - tensor operations (텐서 연산)

- Edges (엣지)
  - data flow between operations
  - control dependency between operations

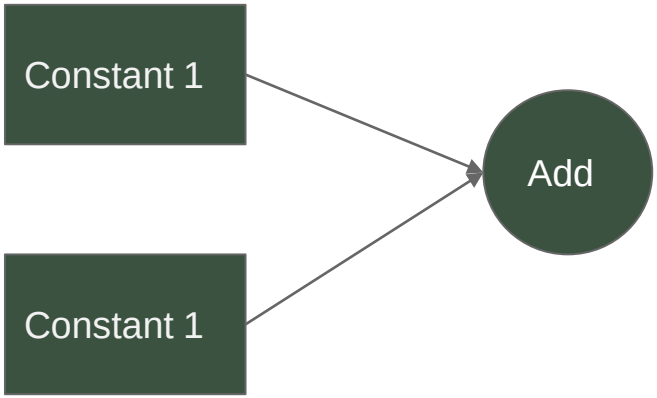
Tensorflow Overview

- 선언부와 실행부가 다름
- Graph를 먼저 정의한 후, Session(세션) 실행 시 실제
- 연산이 이루어짐(Static Graph)



www.tesorflow.org

DataFlow Programming Example



Tensorflow Basic Example

- tf.placeholder 또는 Input Tensor를 정의하여 Input Node를 구성한다.
- Input Node에서부터 Output Node까지 이어지는 관계를 정의하여 Graph를 그린다.
- Session을 이용하여 Input Node (tf.placeholder)에 값을 주입(feeding) 하고, Graph를 Run 시킨다.

Placeholder -> Graph -> Session

DataFlow Programming Example

- 3과 4를 더하는 예제를 수행합니다

```
import tensorflow as tf

node1 = tf.constant(3.0, dtype=tf.float32)
node2 = tf.constant(4.0, dtype=tf.float32)
node3 = tf.add(node1, node2)  # node1 + node2

sess = tf.Session()
print(sess.run(node3))
```

```
c = tf.add(a,b) <-> a + b
c = tf.subtract(a,b) <-> a - b
c = tf.multiply(a,b) <-> a * b
c = tf.div(a,b) <-> a / b
```

DataFlow Programming Example

- Tensor name

```
node1 = tf.constant(3.0, dtype=tf.float32)
node2 = tf.constant(4.0, dtype=tf.float32)
node3 = tf.add(node1, node2)  # node1 + node2

print(node1, node2, node3)
```

Tensor("Const:0", shape=(), dtype=float32) Tensor("Const_1:0", shape=(), dtype=float32)
Tensor("add:0", shape=(), dtype=float32)

모든 텐서는 op name으로 구분 및 접근되어서, 이후 원하는 텐서를 저장(save) / 복원(restore) 또는 재사용(reuse) 할 때에도 name으로 접근할 수 있습니다.

Placeholders

Placeholder : 선언 당시에 값은 비어 있고, 크기 (shape)와 데이터 타입 (type)만 정의되어 있어 session runtime에 input을 feeding 할 수 있습니다.

```
node1 = tf.placeholder(tf.float32, name='node1_ph')
node2 = tf.placeholder(tf.float32, name='node2_ph')
node3 = tf.add(node1,node2) # node1 + node2

with tf.Session() as sess:
  print(sess.run(node3, feed_dict={node1 : 3, node2 : 4}))
  print(sess.run(node3, feed_dict={node1 : 7, node2 : 2}))
```

Quiz 1
placeholder, session, feed_dict

1. 3x4 행렬에 대한 placeholder a 와 4x6 행렬에 대한 placeholder b를 선언한다.

2. 행렬 a와 b를 곱하여 3x6 행렬 c로 이어지는 그래프를 그린다.

3. numpy.random.randn 함수를 통해 a, b에 대한 랜덤 feed를 만들고,
   session을 이용하여 랜덤 값으로 채운 a, b에 대한 c의 값을 출력한다.

Hint!

```
import numpy as np
a = np.random.randn(3,4)
print(a)
# Matrix multiplication(행렬 곱) : tf.matmul
```

Variables

Variable과 Constant / Placeholder의 차이점

-> 텐서의 값이 변할 수 있느냐 없느냐의 여부

```
W = tf.Variable([.3], dtype=tf.float32)
b = tf.Variable([-.3], dtype=tf.float32)
x = tf.placeholder(tf.float32)
linear_model = W * x + b
init_op = tf.global_variables_initializer()

with tf.Session() as sess:
  sess.run(init_op)
  sess.run(linear_model, feed_dict={x:5.0})
```

Variables

만약 sess.run 하는 op의 그래프에 변수 tf.Variable이 하나라도 포함되어 있다면, 반드시 해당 변수를 초기화하는 tf.global_variables_initializer()를 먼저 실행해 줘야 합니다.

```
W = tf.Variable([.3], dtype=tf.float32)
b = tf.Variable([-.3], dtype=tf.float32)
x = tf.placeholder(tf.float32)
linear_model = W * x + b
init_op = tf.global_variables_initializer()

with tf.Session() as sess:
  sess.run(init_op)
  sess.run(linear_model, feed_dict={x:5.0})
```

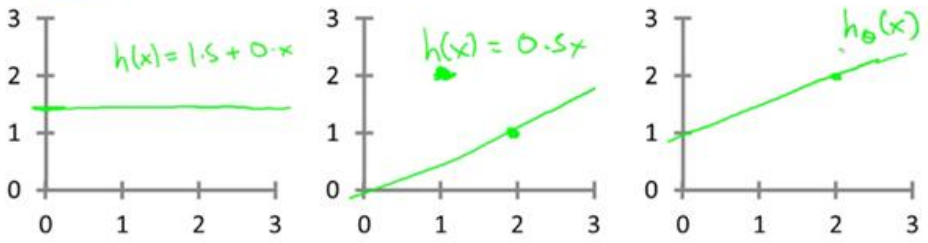
Tensorflow Application

Linear Regression

Linear Regression

$$h_\theta = \theta_0 + \theta_1 x$$

$h_\theta(x) = \theta_0 + \theta_1 x$



→ $\theta_0 = 1.5$
→ $\theta_1 = 0$

→ $\theta_0 = 0$
→ $\theta_1 = 0.5$

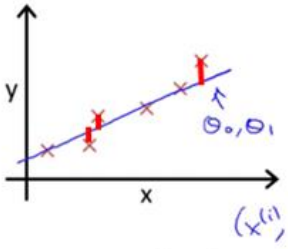
→ $\theta_0 = 1$
→ $\theta_1 = 0.5$

https://ko.coursera.org/learn/machine-learning, lectured by Andrew Ng

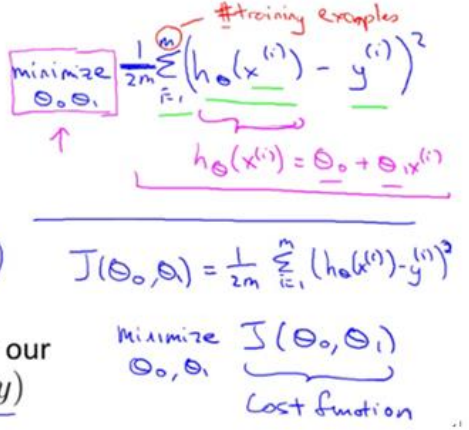
Linear Regression (Cost Function)

$$y = ax + b$$

$$\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$



Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

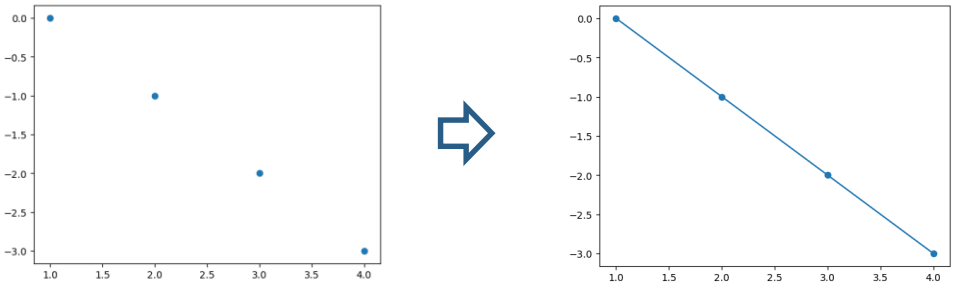


https://ko.coursera.org/learn/machine-learning, lectured by Andrew Ng

Linear Regression

데이터 정의하기

```
from matplotlib import pyplot as plt

# training data
x_train = [1, 2, 3, 4]
y_train = [0, -1, -2, -3]

plt.scatter(x_train,y_train)
plt.plot(x_train, y_train)
plt.show()
```



Linear Regression

Placeholder & Variables 정의하기

```
# Model input and output
x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)

# Model parameters
W = tf.Variable([0.3], tf.float32)
b = tf.Variable([-0.3], tf.float32)
```

Linear Regression

모델 그래프 그리기

```
hypothesis = x * W + b

# cost/loss function
cost = tf.reduce_sum(tf.square(hypothesis - y))  # sum of the squares

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

Cost function -> $\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

Linear Regression

Linear Regression 모델 학습

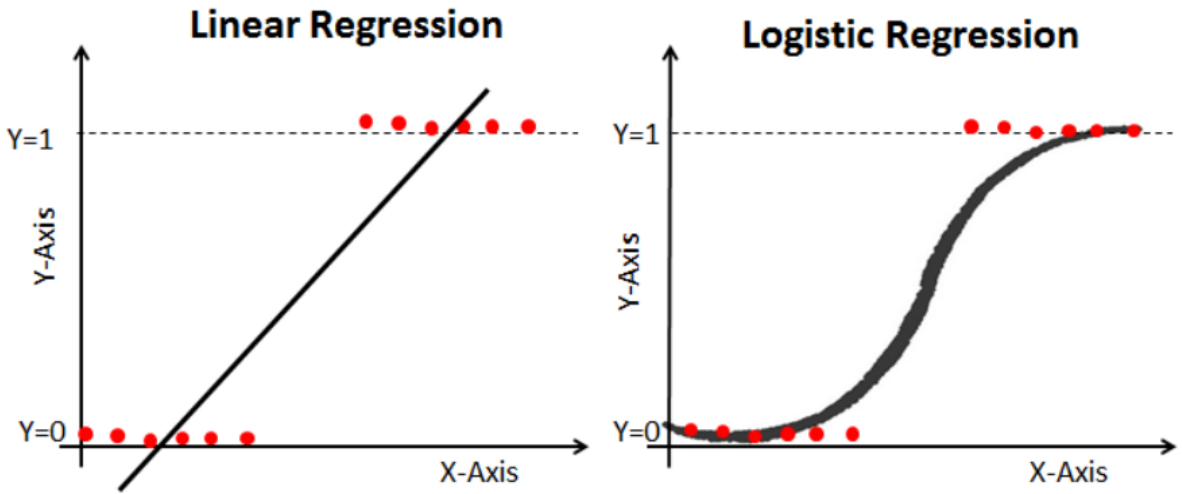
```
# training
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    for step in range(1000):
        sess.run(train, {x: x_train, y: y_train})

    # evaluate training accuracy
    W_val, b_val, cost_val = sess.run([W, b, cost],
                                      feed_dict={x: x_train, y: y_train})
    print("W: {W_val} b: {b_val} cost: {cost_val}")

"""
W: [-0.9999969] b: [0.9999908] cost: 5.699973826267524e-11
"""
```

F(x) = W * x + b = (−1) * x + 1

Tensorflow Application

Classification
- Logistic Regression
- Softmax Regression(MNIST)

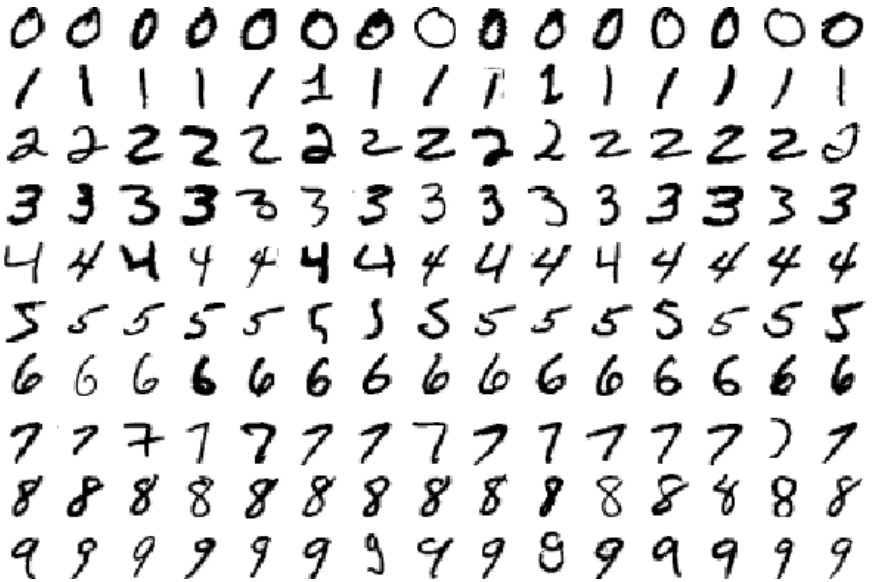
Logistic Regression



MNIST

Image Classification Dataset

0 ~ 9 까지의 손 글씨 이미지를 알맞은 label로 분류하는 Task



MNIST -Softmax Regression

1. 모델의 입력 및 출력 정의

2. 모델 구성하기 (Logistic Regression)

3. 모델 학습

MNIST -Softmax Regression

모델의 입력 및 출력 정의

- Input : 28 * 28 이미지 = 784차원 벡터 [0, 221, 255, …]
- Label : [0.0, 1.0, 0.0, 0.0, …] -> 10차원 One-hot 벡터
- Predictions : 이미지가 각 클래스에 속할 확률 예측 값을 나타내는 10차원 벡터 [0.12, 0.31, …]

⇒ 모델의 예측 값이 정답 데이터 (Label 또는 Ground-truth)와 최대한 비슷해지도록 모델 weight parameter를 학습 시키고 싶다.

⇒ Label과 Prediction의 차이를 최소화하는 방향으로 학습 진행

MNIST -Softmax Regression

MNIST 데이터 불러오기

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("./data", one_hot=True)
```

데이터를 Batch 단위로 불러오기

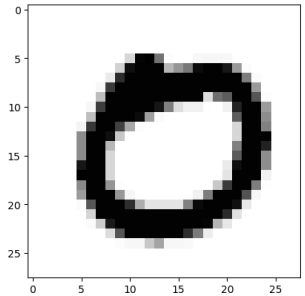
```
for _ in range(10000):
  batch_images, batch_labels = mnist.train.next_batch(100)
  batch_images_val, batch_labels_val = mnist.validation.next_batch(100)
  print (batch_images.shape) # [100, 784]
  print (batch_labels.shape) # [100, 10]
```

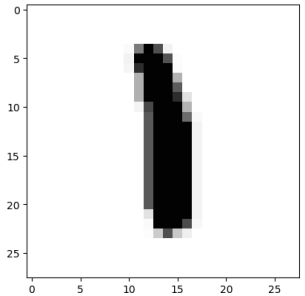
1 Epoch : 전체 데이터에 대해서 모델이 1회 완료
1 Batch : 전체 데이터에서 batch_size 만큼 잘라 모델을 학습
=> 위의 경우, 전체 데이터를 batch_size =100으로 하여
10000 batch만큼 학습 진행

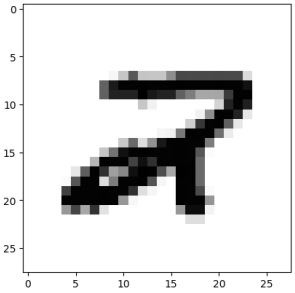
MNIST -Softmax Regression

MNIST 데이터 예시

```
r = random.randint(0, mnist.train.num_examples - 1)
plt.imshow(mnist.train.images[r:r+1].reshape(28, 28),
           cmap='Greys', interpolation='nearest')
plt.show()
```







MNIST -Softmax Regression

모델 구성하기

- 모델의 입력을 placeholder로 구성
- Batch 단위 학습을 위해 shape을 None으로 주어 임의의 batch_size에 대해 모델이 처리할 수 있도록 합니다.

```
# define model input: image and ground-truth label
model_inputs = tf.placeholder(dtype=tf.float32, shape=[None, 784])
labels = tf.placeholder(dtype=tf.float32, shape=[None, 10])
```

- Logistic Regression Model Parameter정의

```
# define parameters for Logistic Regression model
w = tf.Variable(tf.random_normal(shape=[784, 10]))
b = tf.Variable(tf.random_normal(shape=[10]))
```

MNIST -Softmax Regression

모델 구성하기 – 그래프 그리기

```
logits = tf.matmul(model_inputs, w) + b
predictions = tf.nn.softmax(logits)

# LOSS FUNCTION
loss = tf.reduce_mean(-tf.reduce_sum(labels*tf.log(predictions),
                      reduction_indices=1))
```

모델 예측 vs. 정답 비교

```
# PREDICTION
compare_pred = tf.equal(tf.argmax(logits, 1), tf.argmax(labels, 1))
# ACCURACY
accuracy = tf.reduce_mean(tf.cast(compare_pred, "float"))
```

MNIST -Softmax Regression

모델 구성하기 – Optimizer 정의

모델이 loss (predictions과 labels 사이의 차이)를 최소화 하는 방향으로 weight parameter 업데이트를 진행

```
train_op = tf.train.GradientDescentOptimizer(learning_rate=0.001)
            .minimize(loss, global_step=global_step)
```

MNIST -Softmax Regression

모델 학습 진행

- Session을 통해 모든 tensor variable들을 초기화 하고, 데이터를 각 batch만큼 가져와서 모델 학습을 진행 (sess.run(train_op)를 통해 weight update 진행)

```
with tf.Session() as sess:
  sess.run(tf.global_variables_initializer())
  train_epoch = 50
  for epoch in range(train_epoch):
    avg_loss = 0.
    for step in range(10000):
      batch_images, batch_labels = mnist.train.next_batch(100)
      feeds_train = {model_inputs: batch_images, labels: batch_labels}
      _, loss_val, global_step_val = sess.run([train_op, loss, global_step],
                                              feed_dict=feeds_train)
      avg_loss += loss_val
      if (step+1) % 1000 == 0:
        print ("step {}| loss : {}".format(step+1, avg_loss/(step+1)))
      writer.add_summary(summary, global_step=global_step_val)
```

MNIST -Softmax Regression

모델 학습 진행

-매 Epoch이 끝난 후, test set을 통해 모델의 성능을 평가(Evaluation) 합니다.

```
###### Evaluation for every epoch #####
feeds_test = {model_inputs: mnist.test.images, labels: mnist.test.labels}
train_acc = sess.run(accuracy, feed_dict=feeds_train)
test_acc = sess.run(accuracy, feed_dict=feeds_test)
print("Epoch: %03d/%03d cost: %.3f train_acc: %.3f test_acc: %.3f"
      % (epoch+1, train_epoch, avg_loss/(step+1), train_acc, test_acc))
```

MNIST -Softmax Regression

모델 학습 진행

-매 Epoch이 끝난 후, test set을 통해 모델의 성능을 검증(Evaluation) 합니다.

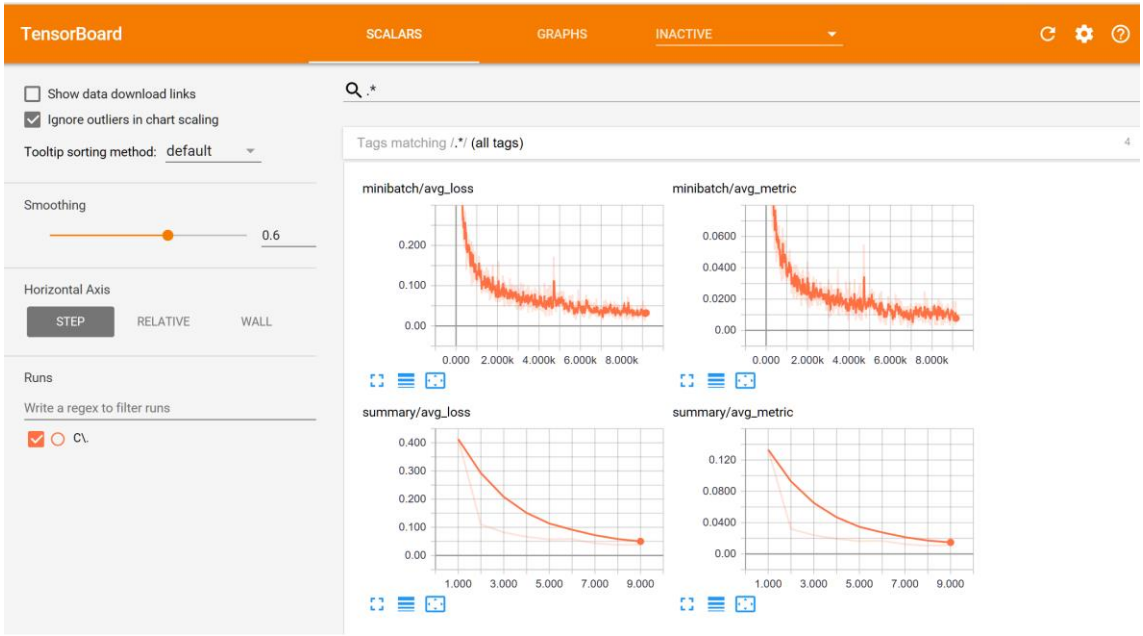
```
###### Evaluation for every epoch #####
feeds_test = {model_inputs: mnist.test.images, labels: mnist.test.labels}
train_acc = sess.run(accuracy, feed_dict=feeds_train)
test_acc = sess.run(accuracy, feed_dict=feeds_test)
print("Epoch: %03d/%03d cost: %.3f train_acc: %.3f test_acc: %.3f"
      % (epoch+1, train_epoch, avg_loss/(step+1), train_acc, test_acc))
```

MNIST -Softmax Regression

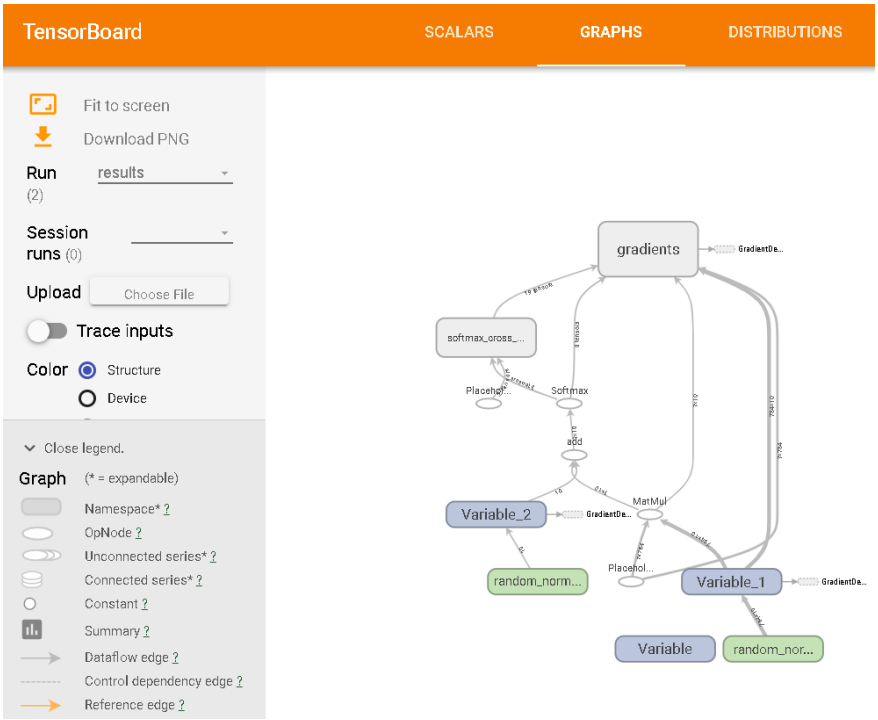
모델 학습 진행 (결과의 예시)

```
Epoch: 000/050 cost: 1.176559254 train_acc: 0.870 test_acc: 0.852
Epoch: 005/050 cost: 0.440937506 train_acc: 0.930 test_acc: 0.895
Epoch: 010/050 cost: 0.383336526 train_acc: 0.900 test_acc: 0.904
Epoch: 015/050 cost: 0.357268913 train_acc: 0.880 test_acc: 0.909
Epoch: 020/050 cost: 0.341493352 train_acc: 0.970 test_acc: 0.912
Epoch: 025/050 cost: 0.330508839 train_acc: 0.890 test_acc: 0.914
Epoch: 030/050 cost: 0.322364672 train_acc: 0.880 test_acc: 0.916
Epoch: 035/050 cost: 0.315942195 train_acc: 0.960 test_acc: 0.917
Epoch: 040/050 cost: 0.310731307 train_acc: 0.910 test_acc: 0.918
Epoch: 045/050 cost: 0.306349064 train_acc: 0.970 test_acc: 0.919
```

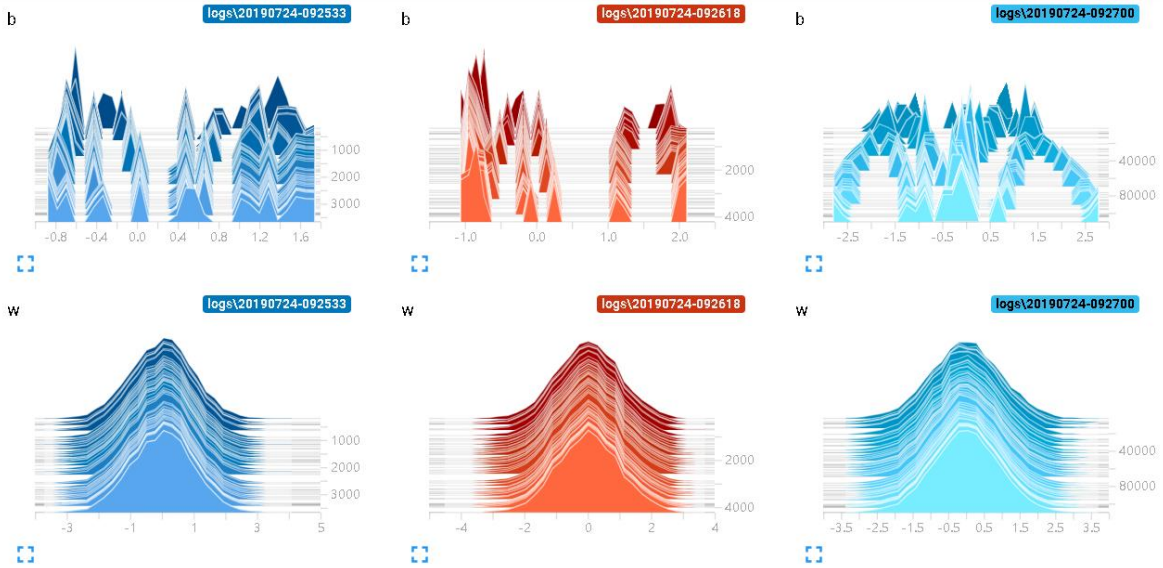
Tensorboard (Scalar)



Tensorboard (Graph)



Tensorboard (Histogram)



Tensorboard (summary)

loss와 같은 scalar 값을 가지는 tensor는 scalar summary로,

parameter tesnsor는 histogram summary로 선언

```
# In the graph

tf.summary.scalar('loss', loss)
tf.summary.histogram('w', w)
tf.summary.histogram('b', b)

# After building graph

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

Tensorboard (summary)

tf.summary.FileWriter 객체를 선언하고, session으로 merged를 실행함

```
# After building graph

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

summary=sess.run(merged, feed_dict={X:x_data, Y:y_data}
writer.add_summary(summary, global_step=sess.run(global_step))
```

$ tensorboard --logdir="./logs" --port=6006 입력

Model Save

tf.train.Saver 객체를 이용해서 변수 저장을 진행합니다.

```
w1 = tf.Variable(tf.random_normal(shape=[2]))
w2 = tf.Variable(tf.random_normal(shape=[5]))

saver = tf.train.Saver()
sess = tf.Session()
sess.run(tf.global_variables_initializer())
saver.save(sess, 'my_test_model', global_step=1000)
```

Saver 객체를 생성합니다. Saver 객체 내에 아무런 파라미터가 없다면 기본으로 Saver 객체는 {key="Variable name", value=variable tensor} 쌍의 dictionary를 내부적으로 가지게 됩니다.

Ex) {"Variable_0:0":w1, "Variable_1:0":w2}

Model Save

tf.train.Saver 객체를 이용해서 모델을 불러옵니다.

```
w1 = tf.Variable(tf.random_normal(shape=[2]), name='w1')
w2 = tf.Variable(tf.random_normal(shape=[5]), name='w2')

saver = tf.train.Saver()
sess = tf.Session()
sess.run(tf.global_variables_initializer())
saver.save(sess, 'my_test_model', global_step=1000)
saver.restore(sess, tf.train.latest_checkpoint('./logs'))
```

- `model.ckpt.data-00000-of-00001`
- `model.ckpt.index`

Model Save

tf.train.Saver 객체를 이용해서 모델을 불러옵니다.

```
with tf.Session() as sess:
  saver = tf.train.Saver()
  saver.restore(sess, "./logs/%s/model.ckpt-%d" % ("20190724-094521",90000))
  feeds_test = {model_inputs: mnist.test.images, labels: mnist.test.labels}
  test_acc = sess.run(accuracy, feed_dict=feeds_test)
  print("test accuracy : %.3f %%" % (test_acc * 100))
```

- 결과 => test accuracy : 89.150 %

Specifying devices using with blocks

모델을 CPU 또는 GPU를 통해 학습을 진행합니다.

```
with tf.device("/cpu:0"):
  W = tf.Variable(...)
  V = tf.Variable(...)

with tf.device("/gpu:0"):
  W = tf.Variable(...)
  V = tf.Variable(...)
```

THANKS!

Any questions?