Training and Test datasets

```
X = tf.placeholder("float", [None, 3])
Y = tf.placeholder("float", [None, 3])
W = tf.Variable(tf.random_normal([3, 3]))
b = tf.Variable(tf.random_normal([3]))
```

data type

⇒ 데이터 형태

형태

placeholder ⇒ 값을 입력받을 공간을 만들어줌.

선언

Variable ⇒ 변수생성

```
hypothesis = tf.nn.softmax(tf.matmul(X, W)+b)
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)

# Correct prediction Test model
prediction = tf.arg_max(hypothesis, 1)
is_correct = tf.equal(prediction, tf.arg_max(Y, 1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
```

예측

각각의 행에서 가장 큰 값을 찾아 인덱스를 반환

이전과 동일

```
# Launch graph
with tf.Session() as sess:
   # Initialize TensorFlow variables
   sess.run(tf.global_variables_initializer())
   for step in range(201):
       cost_val, W_val, _ = sess.run([cost, W, optimizer],
                       feed_dict={X: x_data, Y: y_data})
       print(step, cost_val, W_val)
```

session () ⇒ 흐름이 이루어지도록.

⇒ 텐서플로의 연산 → 데이터플로우 그래프로 구성
텐서플로우에서 변수형은 그래프를 실행하기 전에
초기화 해줘야 함 → 그래야 그 값이 변수에 지정

출력

학습과정

→ 플레이스홀더 X에 X_data, y_data를 하나씩 피드함

training data만 사용

```
   # predict
   print("Prediction:", sess.run(prediction, feed_dict={X: x_test}))
   # Calculate the accuracy
   print("Accuracy: ", sess.run(accuracy, feed_dict={X: x_test, Y: y_test}))
```

test data로 Test

```
199 0.672261 [[-1.15377033  0.28146935
1.13632679]
[ 0.37484586  0.18958236  0.33544877]
[-0.35609841 -0.43973011 -1.25604188]]
200 0.670909 [[-1.15885413  0.28058422
1.14229572]
[ 0.37609792  0.19073224  0.33304682]
[-0.35536593 -0.44033223 -1.2561723 ]]
Prediction: [2 2 2]
Accuracy:  1.0
```

→ [0. 0, 1, 0, 0]
[0, 0, 1, 0, 0]
[0, 0, 1, 0, 0]

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-1-learning_rate_and_evaluation.py