“Linear Regression (Hypothesis)!”

“Let’s figure out the relationship between x and y”

```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()

### 1.Build graph using TF operations

# X & Y data
x_train = [1, 2, 3]
y_train = [1, 2, 3]

W = tf.Variable(tf.random_normal([1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')
```

→ ”Variable”: something what tensorflow use, not user
→ ”Variable”: trainable variable
→ While operating tensorflow, tensorflow will change its value while training itself

```
# Our hypothesis H(x) = Wx + b
hypothesis = x_train * W + b

# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - y_train))

# GradientDescent →Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)

### (2. & 3.) Run/update graph and get results

# Launch the graph in a session.
sess = tf.Session()

# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())
```

→Before running this code using tensorflow variable
→We should initialize “gloabal_variables_initializer()”

```
# Fit the line
for step in range(2001):
   sess.run(train)
   if step % 20 == 0:
      print(step, sess.run(cost), sess.run(W), sess.run(b))
```

<If you want to use "Placeholder"….>

```
# Using placeholders for a tensor that will be always fed using
feed_dict

X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

...
...
...

for step in range(2001):
   cost_val, W_val, b_val, _ = sess.run([cost, W, b,
train],
       feed_dict={X: [1, 2, 3, 4, 5],
            Y: [2.1, 3.1, 4.1, 5.1, 6.1]})
   if step % 20 == 0:
       print(step, cost_val, W_val, b_val)
```

→Different example 1

: H(x) = 1x + 1.1

→[~~,train]

: X & Y values are used

2nd way

```
for step in range(2001):
   cost_val, W_val, b_val, _ = \
       sess.run([cost, W, b, train],
            feed_dict={X: [1, 2, 3,4], Y: [5,7,9,11]})
   if step % 20 == 0:
       print(step, cost_val, W_val, b_val)
```

→Different example 2

: H(x) = 2x + 3

```
# Testing Model
print(sess.run(hypothesis, feed_dict={X: [5]}))
print(sess.run(hypothesis, feed_dict={X: [2.5]}))
print(sess.run(hypothesis,
      feed_dict={X: [1.5, 3.5]}))
```

→Testing Result 1:

1. [6.1000001~~~]

2. [3.5999998~~~~]

3. [2.60001~~~ 4.600000~]

→→ We can estimate that

H(x) = 1x + 1.1

→Testing Result 2:

1. [13.000012~~~]

2. [7.9999988~~~~]

3. [5.00001~~~ 10.000023~]

→→ We can estimate that

H(x) = 2x + 3

<Materials by>

-Sung Kim (Youtuber)

Code: https://github.com/hunkim/DeepLearningZeroToAll/