Linear Regression

Lecture 02

Hypothesis and cost function

$$H(x) = Wx + b$$

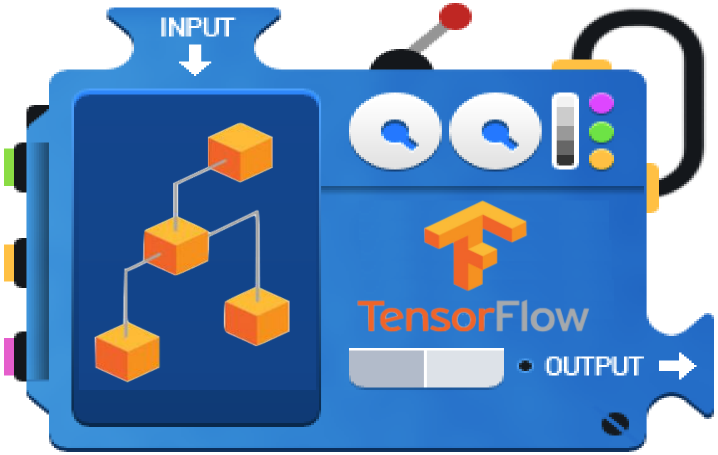
$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$

Hypothesis

True value

TensorFlow Mechanics

2 feed data and run graph (operation)
sess.run (op, feed_dict={x: x_data})



1 Build graph using TensorFlow operations

W b

3 **update variables** in the graph
(and return values)

WWW.MATHWAREHOUSE.COM

1 Build graph using TF operations

$$H(x) = Wx + b$$

```
# X and 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')
# Our hypothesis XW+b
hypothesis = x_train * W + b
```

Tranable value

shape : 1-D

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$

```
t = [1., 2. ,3. ,4.]
tf.reduce_mean(t) ==> 2.5
```

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

1 Build graph using TF operations

GradientDescent

Gradient minimize

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

2 3 Run/update graph and get results

```python
# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_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))
```

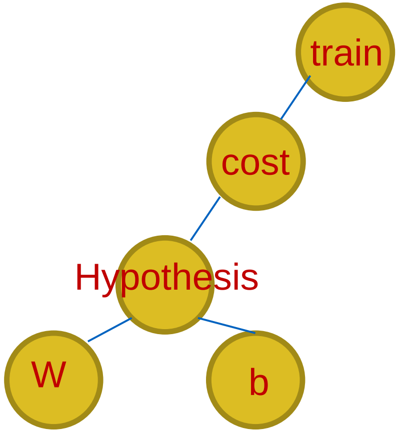
ex02_1.ipynb

Full code

```python
import tensorflow as tf

# X and 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')

# Our hypothesis XW+b
hypothesis = x_train * W + b

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

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

# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

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

$$\boxed{H(x) = \underbrace{W}_{1}x + \underbrace{b}_{0}}$$

$$\boxed{cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}\left(H(x^{(i)}) - y^{(i)}\right)^2}$$



sess.run(train)

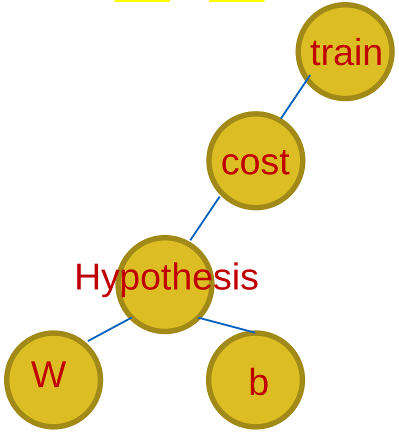
Result

```
step =  0       cost = 2.356354         W = [0.5918387]        b= [-0.68210673]
step =  20      cost = 0.044234347      W = [1.1200231]        b= [-0.42613932]
step =  40      cost = 0.021171913      W = [1.1628493]        b= [-0.38479176]
step =  60      cost = 0.019056372      W = [1.1598109]        b= [-0.3646778].
.
.
step =  1660    cost = 8.613274e-06     W = [1.0034087]        b= [-0.00774858]
step =  1680    cost = 7.822621e-06     W = [1.0032485]        b= [-0.00738446]
step =  1700    cost = 7.1048366e-06    W = [1.0030957]        b= [-0.00703742]
step =  1720    cost = 6.4528454e-06    W = [1.0029503]        b= [-0.00670672]
step =  1740    cost = 5.860434e-06     W = [1.0028117]        b= [-0.00639155]
step =  1760    cost = 5.3227263e-06    W = [1.0026796]        b= [-0.00609118]
step =  1780    cost = 4.8342026e-06    W = [1.0025536]        b= [-0.00580497]
step =  1800    cost = 4.3903005e-06    W = [1.0024337]        b= [-0.00553218]
step =  1820    cost = 3.987675e-06     W = [1.0023193]        b= [-0.00527222]
step =  1840    cost = 3.6217832e-06    W = [1.0022103]        b= [-0.00502447]
step =  1860    cost = 3.2892756e-06    W = [1.0021064]        b= [-0.00478839]
step =  1880    cost = 2.987611e-06     W = [1.0020075]        b= [-0.00456344]
step =  1900    cost = 2.7134327e-06    W = [1.0019132]        b= [-0.00434902]
step =  1920    cost = 2.4645017e-06    W = [1.0018234]        b= [-0.00414473]
step =  1940    cost = 2.238519e-06     W = [1.0017377]        b= [-0.00394999]
step =  1960    cost = 2.0327607e-06    W = [1.0016559]        b= [-0.00376439]
step =  1980    cost = 1.8464402e-06    W = [1.0015783]        b= [-0.00358754]
step =  2000    cost = 1.6770867e-06    W = [1.0015041]        b= [-0.00341899]
```

1 0

$$\boxed{H(x) = \boxed{W}x + \boxed{b}}$$

1 0



Placeholders

In [2]:

```
import tensorflow as tf
a=tf.placeholder(tf.float32)
b=tf.placeholder(tf.float32)
adder_node = a+b

sess = tf.Session()

print(sess.run(adder_node, feed_dict={a:3, b:4.5}))
print(sess.run(adder_node, feed_dict={a:[1,3], b:[2,4]}))
```

```
7.5
[3. 7.]
```

Placeholders

```
# Now we can use X and Y in place of x_data and y_data
# placeholders for a tensor that will be always fed using feed_dict

X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)
...
# Fit the line
# Fit the line
for step in range(2001):
    cost_val, W_val, b_val, _ = \
        sess.run([cost, W, b, train], feed_dict={X: [1, 2, 3], Y: [1, 2, 3]})
    if step % 20 == 0:
        print("step = ", step, "\t cost =", cost_val, "\t W =" , W_val, "\t b=" ,b_val)
```

Full code with placeholders

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

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

# Our hypothesis XW+b
hypothesis = X * W + b
# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)

# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

# Fit the line
for step in range(2001):
    cost_val, W_val, b_val, _ = sess.run([cost, W, b, train], feed_dict={X: [1, 2, 3], Y: [1, 2, 3]})
    if step % 20 == 0:
        print("step = ", step, "\t cost =", cost_val, "\t W =" , W_val, "\t b=" ,b_val)
```

$$H(x) = \boxed{W}x + \boxed{b}$$

1 0

$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

Result

```
step =  0     cost = 1.4190131         W = [0.51419586]        b= [-0.01082231]
step =  20    cost = 0.016611034       W = [0.8895984]         b= [0.14478312]
step =  40    cost = 0.0035521083      W = [0.92835855]        b= [0.15274705]
step =  60    cost = 0.0031215001      W = [0.9349221]         b= [0.14697465]
step =  80    cost = 0.0028340549      W = [0.9382849]         b= [0.14020129]
step =  100   cost = 0.0025739267      W = [0.9412142]         b= [0.13362509]
step =  120   cost = 0.0023376767      W = [0.9439797]         b= [0.12734643]
.
.
step =  1720  cost = 1.0566646e-06     W = [0.9988089]         b= [0.00270756]
step =  1740  cost = 9.596978e-07      W = [0.9988648]         b= [0.00258033]
step =  1760  cost = 8.7179546e-07     W = [0.99891824]        b= [0.00245911]
step =  1780  cost = 7.9172463e-07     W = [0.9989691]         b= [0.00234354]
step =  1800  cost = 7.1902133e-07     W = [0.9990175]         b= [0.00223342]
step =  1820  cost = 6.5307046e-07     W = [0.9990636]         b= [0.00212849]
step =  1840  cost = 5.931062e-07      W = [0.9991076]         b= [0.00202847]
step =  1860  cost = 5.3867456e-07     W = [0.99914956]        b= [0.00193317]
step =  1880  cost = 4.893247e-07      W = [0.9991895]         b= [0.00184235]
step =  1900  cost = 4.4433477e-07     W = [0.9992276]         b= [0.00175579]
step =  1920  cost = 4.0366527e-07     W = [0.9992639]         b= [0.00167331]
step =  1940  cost = 3.6649055e-07     W = [0.99929845]        b= [0.00159468]
step =  1960  cost = 3.3290038e-07     W = [0.9993314]         b= [0.00151976]
step =  1980  cost = 3.0241787e-07     W = [0.9993628]         b= [0.00144836]
step =  2000  cost = 2.7462912e-07     W = [0.9993928]         b= [0.00138031]
```

1

0

Full code with placeholders

```
import tensorflow as tf
W = tf.Variable(tf.random_normal([1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')
X = tf.placeholder(tf.float32, shape=[None])
Y = tf.placeholder(tf.float32, shape=[None])
# Our hypothesis XW+b
hypothesis = X * W + b
# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)

# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

# Fit the line with new training data
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 = ", step, "\t cost =", cost_val, "\t W =" , W_val, "\t b= " ,b_val)
```

$$H(x) = \boxed{W}x + \boxed{b}$$

1 1.1

$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

Result

```
step =  0       cost = 11.023929        W = [0.489554]      b= [0.19391438]
step =  20      cost = 0.08094631       W = [1.1806402]     b= [0.43541503]
step =  40      cost = 0.07049624       W = [1.1717805]     b= [0.479762]
step =  60      cost = 0.061564885      W = [1.1605438]     b= [0.52038556]
step =  80      cost = 0.053765047      W = [1.1500298]     b= [0.558345]
step =  100     cost = 0.046953432      W = [1.1402042]     b= [0.5938184]
step =  120     cost = 0.041004788      W = [1.1310221]     b= [0.6269687]
.
.
step =  1720    cost = 8.060732e-07     W = [1.000581]      b= [1.0979024]
step =  1740    cost = 7.039638e-07     W = [1.000543]      b= [1.0980397]
step =  1760    cost = 6.149433e-07     W = [1.0005075]     b= [1.098168]
step =  1780    cost = 5.3725876e-07    W = [1.0004741]     b= [1.0982879]
step =  1800    cost = 4.6897708e-07    W = [1.0004431]     b= [1.0984]
step =  1820    cost = 4.0989312e-07    W = [1.0004143]     b= [1.0985045]
step =  1840    cost = 3.5799212e-07    W = [1.0003873]     b= [1.0986024]
step =  1860    cost = 3.125985e-07     W = [1.0003618]     b= [1.0986938]
step =  1880    cost = 2.7307448e-07    W = [1.0003381]     b= [1.0987792]
step =  1900    cost = 2.3841932e-07    W = [1.0003161]     b= [1.0988591]
step =  1920    cost = 2.0840157e-07    W = [1.0002953]     b= [1.0989337]
step =  1940    cost = 1.8199853e-07    W = [1.0002761]     b= [1.0990033]
step =  1960    cost = 1.5897385e-07    W = [1.000258]      b= [1.0990686]
step =  1980    cost = 1.3886101e-07    W = [1.0002412]     b= [1.0991294]
step =  2000    cost = 1.2121755e-07    W = [1.0002254]     b= [1.0991864]
```

1 1.1

$$\boxed{H(x) = \boxed{W}x + \boxed{b}}$$

1 1.1

ex02_4.ipynb (2/2)

Full code with placeholders

```
# 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]}))


[6.1003137]
[3.59975]
[2.5995245 4.5999756]
```

$$H(x) = \underbrace{\boxed{W}}_{1}x + \underbrace{\boxed{b}}_{1.1}$$

TensorFlow Mechanics

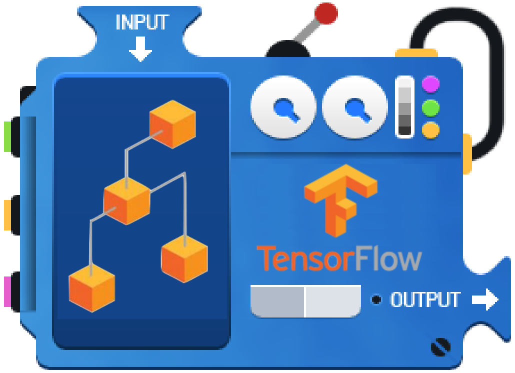
2 feed data and run graph (operation)
sess.run (op, feed_dict={x: x_data})

```
feed_dict={X: [1, 2, 3, 4, 5],
 Y: [2.1, 3.1, 4.1, 5.1, 6.1]})
```

1 Build graph using TensorFlow operations

$$H(x) = Wx + b$$

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$



WWW.MATHWAREHOUSE.COM

W b

3 **update** variables in the graph (and return values)

Linear Regression (summary)

```
# X 와 Y 의 상관관계를 분석하는 기초적인 선형 회귀 모델을 만들고 실행해봅니다.
import tensorflow as tf

x_data = [1, 2, 3]
y_data = [1, 2, 3]

# random_uniform : 범위 내에서 균등한 확률로 랜덤 값이 나오게 함
# random_normal : 범위 내에서 종모양의 정규분포에 따라 값이 나오게 함
W = tf.[redacted](tf.random_uniform([1], -1.0, 1.0))
b = tf.[redacted](tf.random_uniform([1], -1.0, 1.0))

# name: 나중에 텐서보드등으로 값의 변화를 추적하거나 살펴보기 쉽게 하기 위해 이름을 붙여줍니다.
X = tf.[redacted](tf.float32, name="X")
Y = tf.[redacted](tf.float32, name="Y")
print(X)
print(Y)

# X 와 Y 의 상관 관계를 분석하기 위한 가설 수식을 작성합니다.
# y = W * x + b
# W 와 X 가 행렬이 아니므로 tf.matmul 이 아니라 기본 곱셈 기호를 사용했습니다.
hypothesis = W * X + b
```

Linear Regression (summary)

$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

```
# 손실 함수를 작성합니다.
# mean(h - Y)^2 : 예측값과 실제값의 거리를 비용(손실) 함수로 정합니다.
cost =             (tf.square(hypothesis -   ))
# 텐서플로우에 기본적으로 포함되어 있는 함수를 이용해 경사 기울기법을 최적화 수행합니다.
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1)
# 비용을 최소화 하는 것이 최종 목표
train_op = optimizer.minimize(cost)
```

```
# with 기능은 세션을 생성하고 초기화한 후 세션 종료를 자동으로 처리합니다.
with tf.Session() as sess:
    sess.run(                         )

    # 최적화를 100번 수행합니다.
    for step in range(100):
        # sess.run 을 통해 train_op 와 cost 그래프를 계산합니다.
        # 이 때, 가설 수식에 넣어야 할 실제값을 feed_dict 을 통해 전달합니다.
        _, cost_val = sess.run([train_op,     ], feed_dict={  : x_data,   : y_data})

        print("step = ", step, "\t cost_val =%10.8f" %cost_val, "\t W = [%10.8f]" %sess.run(W), "\t b = [%10.8f]" %sess.run(b))
```

```
# 최적화가 완료된 모델에 테스트 값을 넣고 결과가 잘 나오는지 확인해봅니다.
    print("\n=== Test ===")
    print("X: 5, Y:", sess.run(hypothesis, feed_dict={X: 5}))
    print("X: 2.5, Y:", sess.run(hypothesis, feed_dict={X: 2.5}))
```

Result

```
Tensor("X_1:0", dtype=float32)
Tensor("Y_1:0", dtype=float32)
step =  0     cost_val =14.43337250     W = [1.20410192]          b = [-0.02613848]
step =  1     cost_val =0.17374559      W = [1.02406228]          b = [-0.10255154]
step =  2     cost_val =0.00334829      W = [1.04262471]          b = [-0.09166615]
step =  3     cost_val =0.00125242      W = [1.03950810]          b = [-0.09038281]
step =  4     cost_val =0.00116979      W = [1.03878701]          b = [-0.08810949]
.
.
.
step =  90    cost_val =0.00001780      W = [1.00478196]          b = [-0.01087058]
step =  91    cost_val =0.00001695      W = [1.00466704]          b = [-0.01060924]
step =  92    cost_val =0.00001615      W = [1.00455487]          b = [-0.01035422]
step =  93    cost_val =0.00001538      W = [1.00444531]          b = [-0.01010533]
step =  94    cost_val =0.00001465      W = [1.00433850]          b = [-0.00986238]
step =  95    cost_val =0.00001395      W = [1.00423419]          b = [-0.00962530]
step =  96    cost_val =0.00001329      W = [1.00413239]          b = [-0.00939392]
step =  97    cost_val =0.00001266      W = [1.00403309]          b = [-0.00916809]
step =  98    cost_val =0.00001206      W = [1.00393617]          b = [-0.00894770]
step =  99    cost_val =0.00001149      W = [1.00384152]          b = [-0.00873263]

=== Test ===
X: 5, Y: [5.0104747]
X: 2.5, Y: [2.5008712]
```