Linear regression 을 학습하며, 기계학습의 원리 및 TensorFlow 를 익히는 notebook 입니다.

라이브러리 Import 하기

In [1]:

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
```

X and Y data

In [2]:

```
x_train = [1, 2, 3]

y_train = [2+0.1, 4-0.3, 6+0.15] # 약간의 noise 추가

# 다음의 것들도 해보시오
#y_train = [2, 4, 6] # 그냥 x_train 에 2배 곱해서 생성
#y_train = [3, 5, 7]
```

Initialization

In [3]:

```
#W = tf.Variable(tf.random_normal([1]), name='weight')
#b = tf.Variable(tf.random_normal([1]), name='bias')
w0 = 7.0;
b0 = 5.0;

W = tf.Variable(w0*tf.ones([1]), name='weight')
b = tf.Variable(b0*tf.ones([1]), name='bias')
```

Our hypothesis XW+b

In [4]:

```
hypothesis = x_train * W + b
```

cost/loss function 정의하기

- loss of one training example :

$$loss = \mathcal{L}(\hat{y}, y) = (\hat{y}^{(i)} - y^{(i)})^2 \tag{1}$$

In [5]:

```
loss = tf.reduce_mean(tf.square(hypothesis - y_train))
```

Optimizer

In [6]:

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

Launch the graph in a session

In [7]:

```
sess = tf.Session()
```

Initializes global variables in the graph.

In [8]:

```
sess.run(tf.global_variables_initializer())
```

In [9]:

```
nb_epoch = 2001

for step in range(nb_epoch):
    sess.run(train)

    if step % 100 == 0: # 100번마다
      w1 = sess.run(W)[0] # 기울기
      b1 = sess.run(b)[0] # bias
      print(step, sess.run(loss), w1, b1)
```

```
0 191.49959 6.333 4.6996665
100 0.5531757 1.1931438 1.8244518
200 0.3571643 1.3710526 1.4199096
300 0.23604111 1.5109378 1.1019175
400 0.1611947 1.6209002 0.85194725
500 0.11494404 1.7073406 0.6554478
600 0.086363904 1.7752907 0.50098115
700 0.06870318 1.8287058 0.37955633
800 0.057789847 1.870695 0.28410515
900 0.051046133 1.9037024 0.20907182
1000 0.046878885 1.9296489 0.15008885
1100 0.044303846 1.9500453 0.10372282
1200 0.042712536 1.9660789 0.06727502
1300 0.041729197 1.9786826 0.038623624
1400 0.041121576 1.9885905 0.016101124
1500 0.040746186 1.9963787 -0.0016036468
1600 0.040514156 2.002501 -0.015521128
1700 0.040370733 2.007314 -0.02646201
1800 0.040282138 2.0110972 -0.035062194
1900 0.040227447 2.0140712 -0.04182288
2000 0.040193584 2.016409 -0.047137175
```

학습완료

In [10]:

```
w1 = sess.run(W)[0] # 기울기
b1 = sess.run(b)[0] # bias
```

출력해보기

In [11]:

```
print(w1, b1)

str1 = 'y = ' + str(w1) +'x + ' + str(b1)
print(str1)
```

```
2.016409 -0.047137175
y = 2.016409x + -0.047137175
```

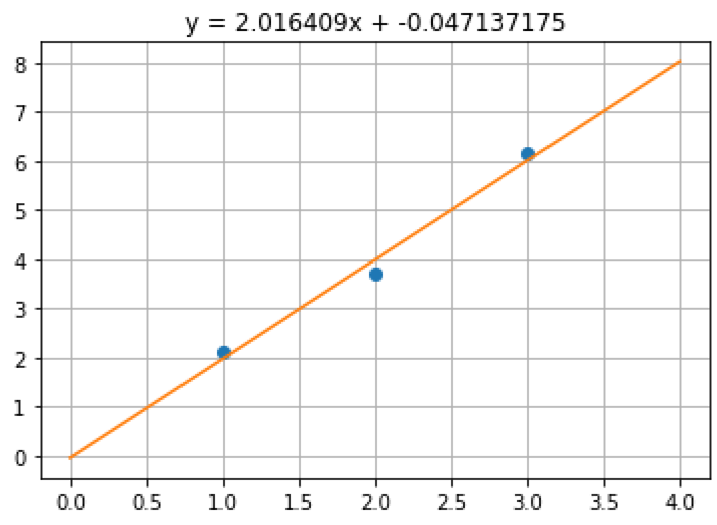
In [12]:

```
plt.figure(1)
plt.plot(x_train, y_train,'o')

# 그래프의 x좌표를 일정 간격으로 설정함
x1 = np.linspace(np.min(x_train)-1, np.max(x_train)+1)
y1 = w1*x1 + b1
plt.plot(x1, y1)
plt.grid() # 격자
#plt.axis((np.min(x_train) - 1, np.max(x_train) + 1, np.min(y_train) - 1, np.max(y_train) + 1))
plt.title(str1)
```

Out[12]:

```
Text(0.5,1,'y = 2.016409x + -0.047137175')
```



스스로 해보기

아래 부분을 수정해서 처음부터 다시 진행해보기 바랍니다.

```
x_train = [1, 2, 3]
y_train = [2+0.1, 4-0.3, 6+0.15] # 약간의 noise 추가
```

In [ ]:

```
```