Linear Regression

OBJECTIVE: Understand and practice linear regression.

- Very important !

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

X and Y data

```
x_train = [1, 2, 3, 4, 5]
y_train = [2+0.1+3, 4-0.3+3, 6+0.15+3, 8+0.2+3, 10-0.2+3] #  Add some noise
```

Initialization

```
#W = tf.Variable(tf.random_normal([1]), name='weight')
#b = tf.Variable(tf.random_normal([1]), name='bias')
w0 = 4000.0;
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

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

cost/loss function

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

Optimizer

```
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()) # = tf.Session().run(tf.global_variables_initializer())
```

```
vw = [] # vector weight
vb = [] # vector bias

for step in range(4001):
    sess.run(train)
    w1 = sess.run(W)[0] # slope
    b1 = sess.run(b)[0] # bias
    vw.append(w1)
    vb.append(b1)

    if step % 100 == 0:
        print(step, sess.run(cost), w1, b1)
```

```
0 102568500.0 3120.319 -234.9202
100 96793.09 203.9755 -726.21173
200 49167.79 145.9488 -516.7169
300 24975.648 104.59207 -367.406
400 12686.833 75.11636 -260.98938
500 6444.5195 54.10851 -185.14435
600 3273.627 39.135822 -131.08821
700 1662.9153 28.464508 -92.56137
800 844.7262 20.858862 -65.10258
900 429.11313 15.438184 -45.53221
1000 217.99532 11.574768 -31.584053
1100 110.75426 8.821239 -21.642939
1200 56.27932 6.858749 -14.557728
1300 28.607834 5.460046 -9.507966
1400 14.551626 4.4631653 -5.9089117
1500 7.411537 3.7526705 -3.3438
1600 3.7845962 3.2462876 -1.5155963
1700 1.9422306 2.885379 -0.2126022
1800 1.0063696 2.628153 0.71606517
1900 0.53098255 2.444823 1.3779436
2000 0.28950173 2.3141608 1.8496761
2100 0.16683686 2.2210355 2.1858885
2200 0.1045274 2.154663 2.4255137
2300 0.07287623 2.1073585 2.5962985
2400 0.0567985 2.073644 2.7180195
2500 0.04863139 2.0496144 2.8047726
2600 0.04448301 2.0324886 2.8666031
2700 0.042375635 2.0202823 2.910672
2800 0.041305166 2.0115829 2.9420795
2900 0.040761326 2.005382 2.9644651
3000 0.04048509 2.000963 2.9804199
3100 0.040344816 1.9978137 2.9917898
3200 0.040273584 1.9955691 2.999894
3300 0.040237427 1.9939693 3.0056696
3400 0.04021903 1.9928291 3.009786
3500 0.040209614 1.9920164 3.0127199
3600 0.040204912 1.9914373 3.014811
3700 0.040202416 1.9910245 3.0163016
3800 0.040201165 1.9907302 3.0173638
3900 0.040200673 1.9905206 3.0181205
4000 0.040200375 1.9903712 3.0186594
```

Complete training

```
w1 = sess.run(W)[0] # slope
```

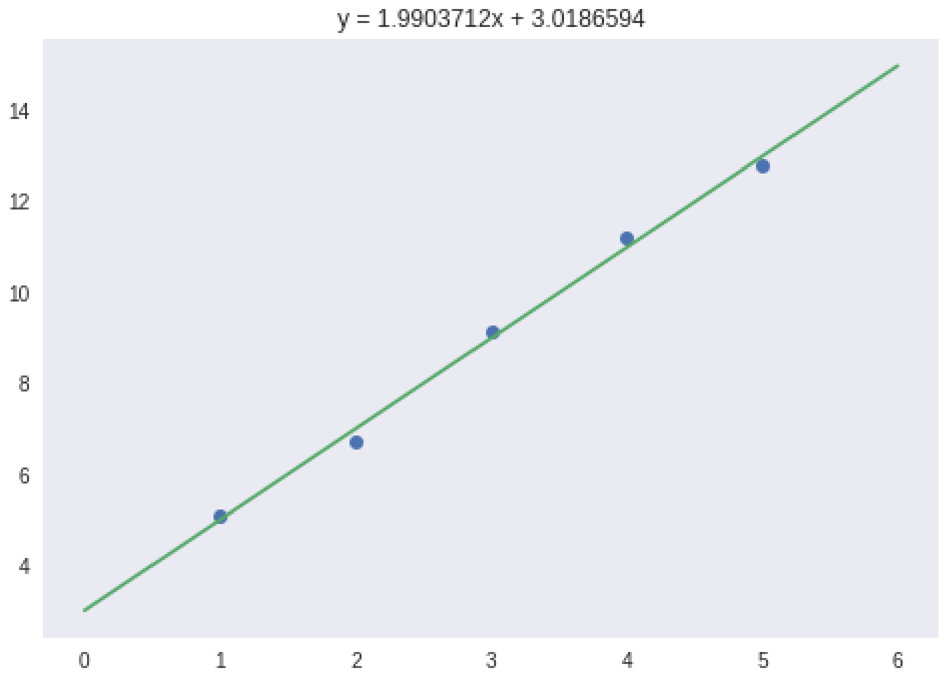
```
b1 = sess.run(b)[0] # bias
str1 = 'y = ' + str(w1) +'x + ' + str(b1)
print(w1, b1)
print(str1)
```

```
1.9903712 3.0186594
y = 1.9903712x + 3.0186594
```

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

x1 = np.linspace(np.min(x_train)-1, np.max(x_train)+1)
y1 = w1*x1 + b1
plt.plot(x1, y1)
plt.grid()
plt.title(str1)
```

```
Text(0.5, 1.0, 'y = 1.9903712x + 3.0186594')
```



위에서 bias에 3을 추가해준 결과 3을 찾아낸 모습이다.

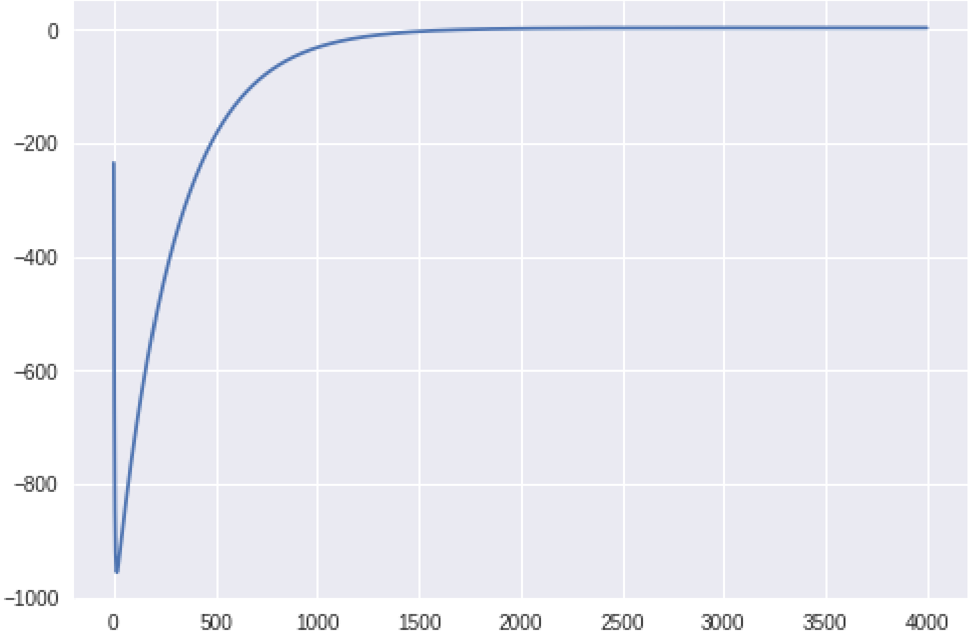
```
plt.plot(vw)
```

```
[<matplotlib.lines.Line2D at 0x7f9da8454048>]
```

3000

```
plt.plot(vb)
```

```
[<matplotlib.lines.Line2D at 0x7f9da842fa58>]
```


0
-200
-400
-600
-800
-1000
0
500
1000
1500
2000
2500
3000
3500
4000