01_linear_regression_using_tensorflow_homework

September 25, 2020

Import

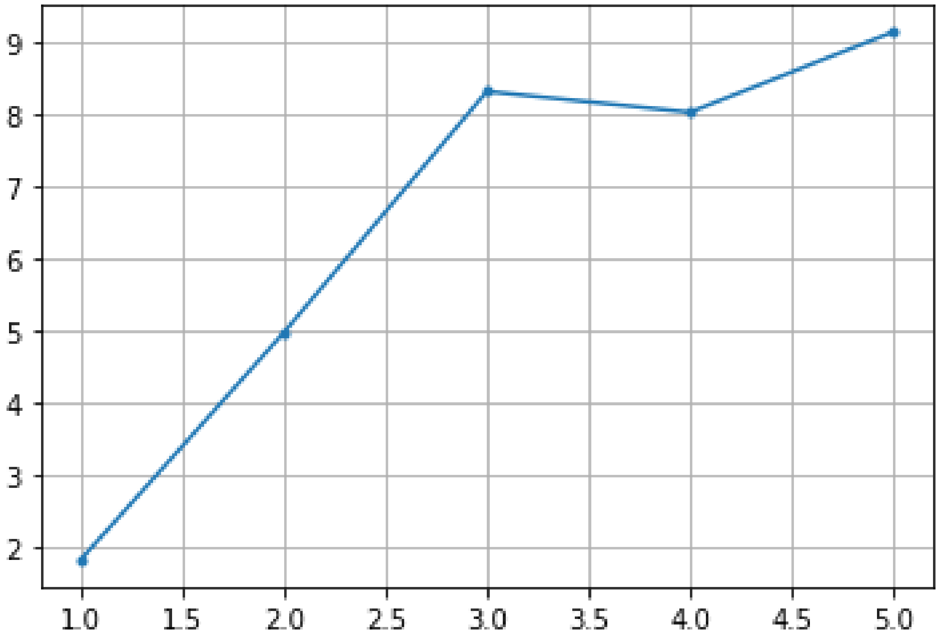
[2]:
```
#import tensorflow as tf
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()

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

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow/python/compat/v2_compat.py:96: disable_resource_variables
(from tensorflow.python.ops.variable_scope) is deprecated and will be removed in
a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

X and Y data

[3]:
```
x_train = [1, 2, 3, 4, 5]
#y_train = [2, 4, 6, 8, 10]
y_train = [3, 5, 7, 9, 11] #y = 2x +1

signal_length = len(x_train)
y_noise = np.random.normal(0, 1, signal_length)

y_train = y_train + y_noise

#x_train = [1, 2, 3]

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

#
#y_train = [2, 4, 6] #  x_train  2
#y_train = [3, 5, 7]
```

[4]:
```
plt.plot(x_train, y_train,'.-')
plt.grid()
```



Initialization

[5]:
```
useRandom = False
```

[6]:
```
if useRandom:
    W = tf.Variable(tf.random_normal([1]), name='weight')
    b = tf.Variable(tf.random_normal([1]), name='bias')
else:
    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

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

[7]:
```
hypothesis = x_train * W + b
```

cost/loss function * loss of one training example :

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

[8]:
```
loss = tf.reduce_mean(tf.square(hypothesis - y_train))
```

Optimizer

[9]:
```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(loss)
```

Launch the graph in a session

[10]:

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

Initializes global variables in the graph.

[11]:

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

[12]:

```
nb_epoch = 1001
vloss = [] #empty list
vb = [] #empty list
vw = [] #empty list
for step in range(nb_epoch):
    sess.run(train)
    loss1 = sess.run(loss)
    vloss.append(loss1)

    if step % 50 == 0: # 5
        w1 = sess.run(W)[0] #
        b1 = sess.run(b)[0] # bias

        print(step,'\t', loss1, '\t', w1, '\t',b1)
```

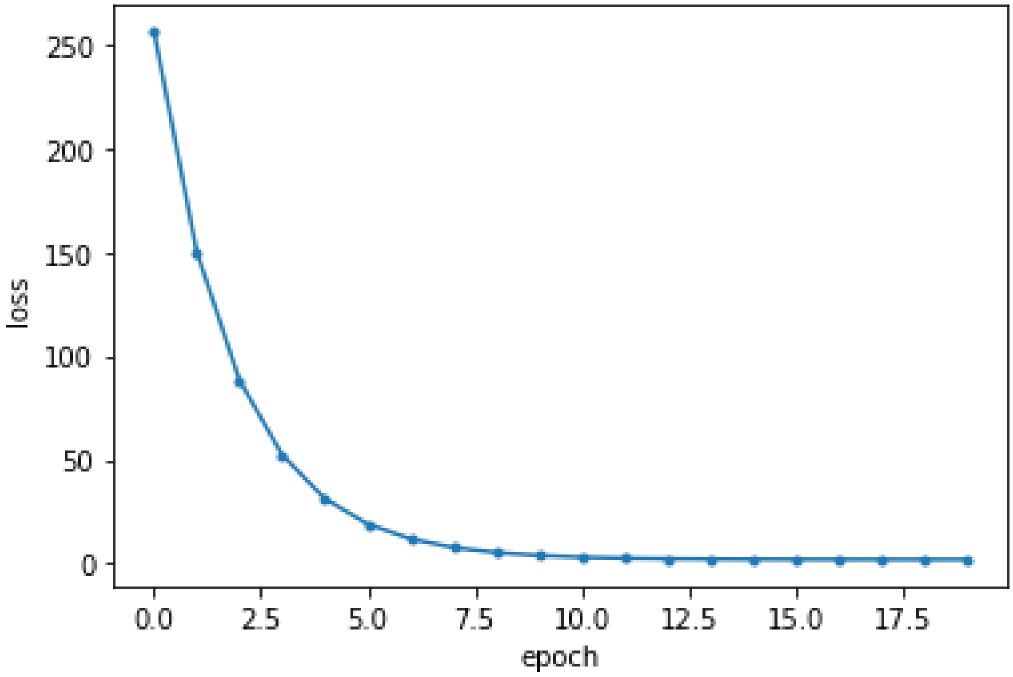
```
0        256.1197         5.61764          4.609017
50       1.7335055        1.2467369        3.0267956
100      1.5506191        1.3274224        2.735472
150      1.4202728        1.3955445        2.4895294
200      1.3273729        1.4530548        2.2818992
250      1.2611611        1.5016066        2.1066117
300      1.2139709        1.5425953        1.9586297
350      1.180338         1.5771987        1.8336997
400      1.1563662        1.6064122        1.7282301
450      1.139282         1.6310749        1.6391898
500      1.127105         1.6518958        1.5640197
550      1.1184269        1.6694733        1.5005594
600      1.1122416        1.6843129        1.4469839
650      1.107833         1.6968408        1.4017541
700      1.1046913        1.7074171        1.3635705
750      1.102452         1.7163459        1.3313347
800      1.1008556        1.7238839        1.3041204
850      1.0997186        1.7302476        1.2811451
900      1.0989077        1.7356201        1.2617486
950      1.0983301        1.7401556        1.245374
1000     1.0979183        1.7439846        1.2315502
```

[13]:

```
plt.plot(vloss[:20],'.-')
plt.xlabel('epoch')
plt.ylabel('loss')
```

[13]:

```
Text(0, 0.5, 'loss')
```



[14]:
```
w1 = sess.run(W)[0] #
b1 = sess.run(b)[0] # bias
```

[15]:
```
print(w1, b1)
```

```
1.7439846 1.2315502
```

[16]:
```
str1 = 'y = ' + str(w1) +'x + ' + str(b1)
print(str1)
```

```
y = 1.7439846x + 1.2315502
```

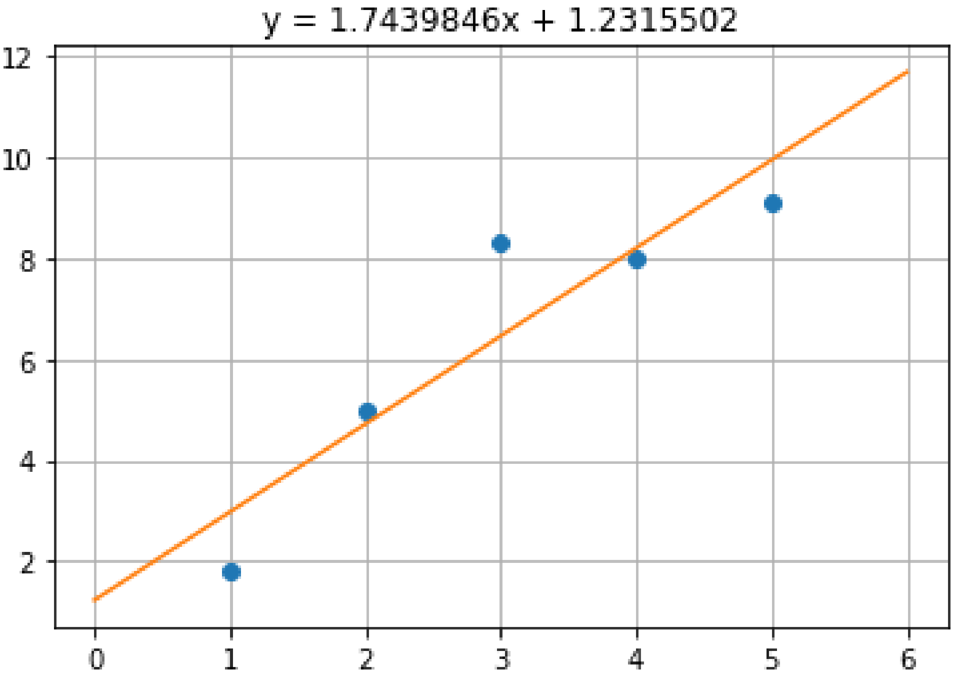
[17]:
```
print(w1, b1)

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

```
1.7439846 1.2315502
y = 1.7439846x + 1.2315502
```

[18]:
```
plt.figure(figsize=(6,4)) # figsize
plt.plot(x_train, y_train,'o') #train data
```

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

plt.grid() #
#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)
```

[18]:
```
Text(0.5, 1.0, 'y = 1.7439846x + 1.2315502')
```

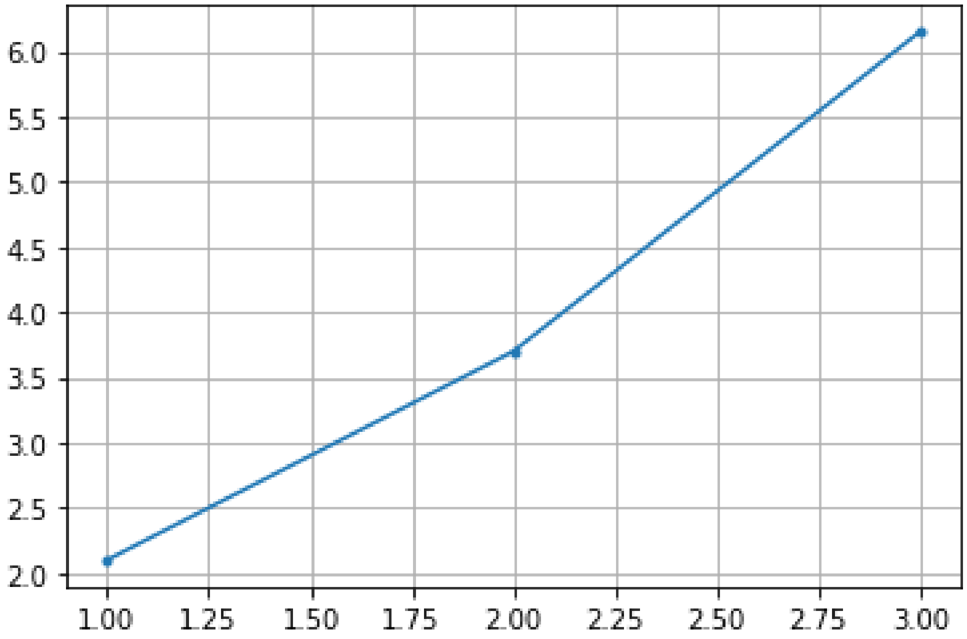


**

**

[19]:
```
x_train = [1, 2, 3]
y_train = [2+0.1, 4-0.3, 6+0.15] #  noise
```

[20]:
```
plt.plot(x_train, y_train,'.-')
plt.grid()
```

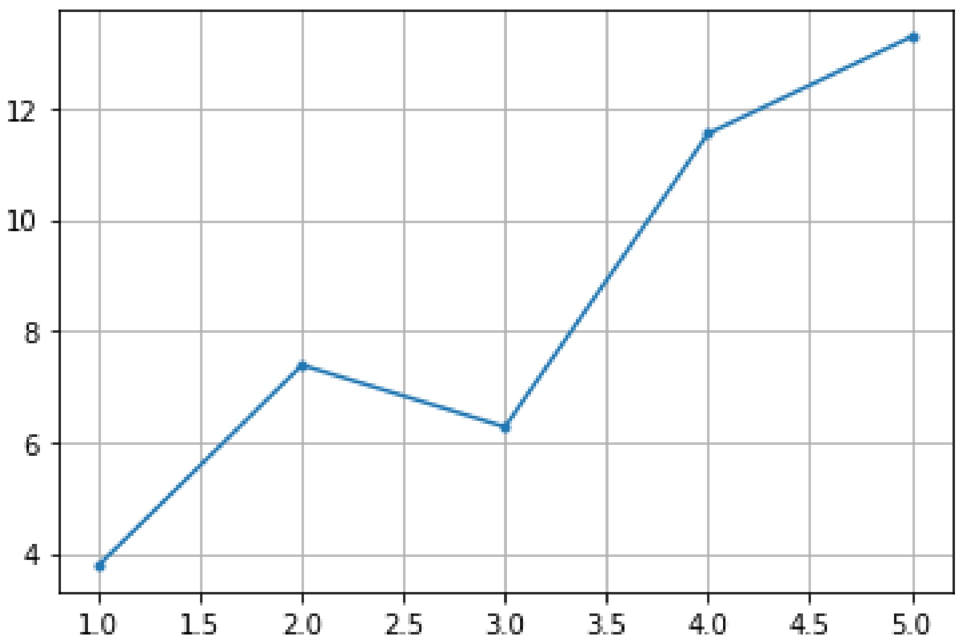


[21]:

```
x_train = [1, 2, 3, 4, 5]
y_train = [2+2, 4+2, 6+2, 8+2, 10+2]#y=2x+2

signal_length = len(x_train)
y_noise = np.random.normal(0, 1, signal_length)

y_train = y_train + y_noise
```

[22]:

```
plt.plot(x_train, y_train,'.-')
plt.grid()
```



[23]:
```
w0 = 15.0;
b0 = 9.0;

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

[24]:
```
hypothesis = x_train * W + b
loss = tf.reduce_mean(tf.square(hypothesis - y_train))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(loss)
sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

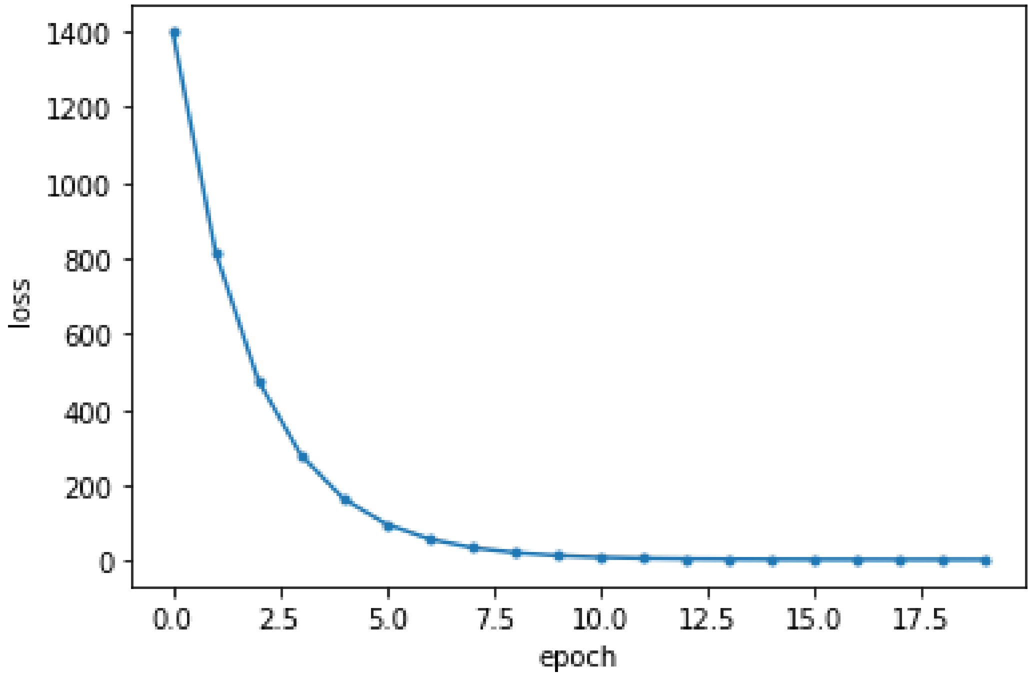
[25]:
```
nb_epoch = 201
vloss =[]
vb = []
vw = []
for step in range(nb_epoch):
    sess.run(train)
    loss1 = sess.run(loss)
    vloss.append(loss1)
    w1 = sess.run(W)[0] #
    vw.append(w1)
    b1 = sess.run(b)[0] # bias
    vb.append(b1)
    if step % 10 == 0: # 200
```

```
        print(step,'\t' ,loss1,'\t' ,w1,'\t', b1)
```

```
0        1398.9922       11.7596245      8.08905
10       9.98214         2.0315993       5.2628326
20       3.5587227       1.4082865       4.9628606
30       3.3899102       1.3958577       4.8363333
40       3.258503        1.4235034       4.725004
50       3.1358213       1.4528939       4.618121
60       3.0211744       1.4814848       4.514847
70       2.9140356       1.5091357       4.4150143
80       2.8139138       1.5358665       4.3185077
90       2.7203498       1.5617073       4.2252145
100      2.6329129       1.5866876       4.1350274
110      2.5512025       1.6108363       4.047844
120      2.4748433       1.6341804       3.963564
130      2.4034848       1.6567472       3.88209
140      2.3367999       1.6785628       3.8033292
150      2.2744815       1.6996518       3.727191
160      2.2162447       1.7200385       3.6535883
170      2.1618223       1.7397466       3.5824366
180      2.1109633       1.7587981       3.5136542
190      2.0634356       1.7772154       3.4471622
200      2.0190206       1.7950191       3.3828845
```

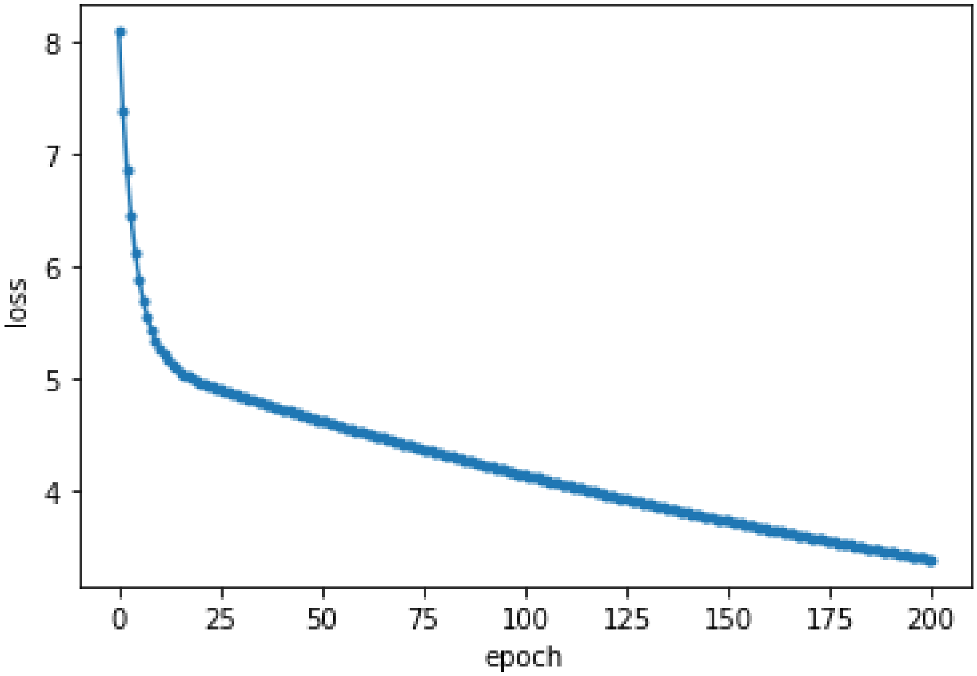
[26]:
```
plt.plot(vloss[:20],'.-')
plt.xlabel('epoch')
plt.ylabel('loss')
```

[26]:
```
Text(0, 0.5, 'loss')
```



[33]:
```
plt.plot(vb,'.-')
plt.xlabel('epoch')
plt.ylabel('loss')
```

[33]:
```
Text(0, 0.5, 'loss')
```

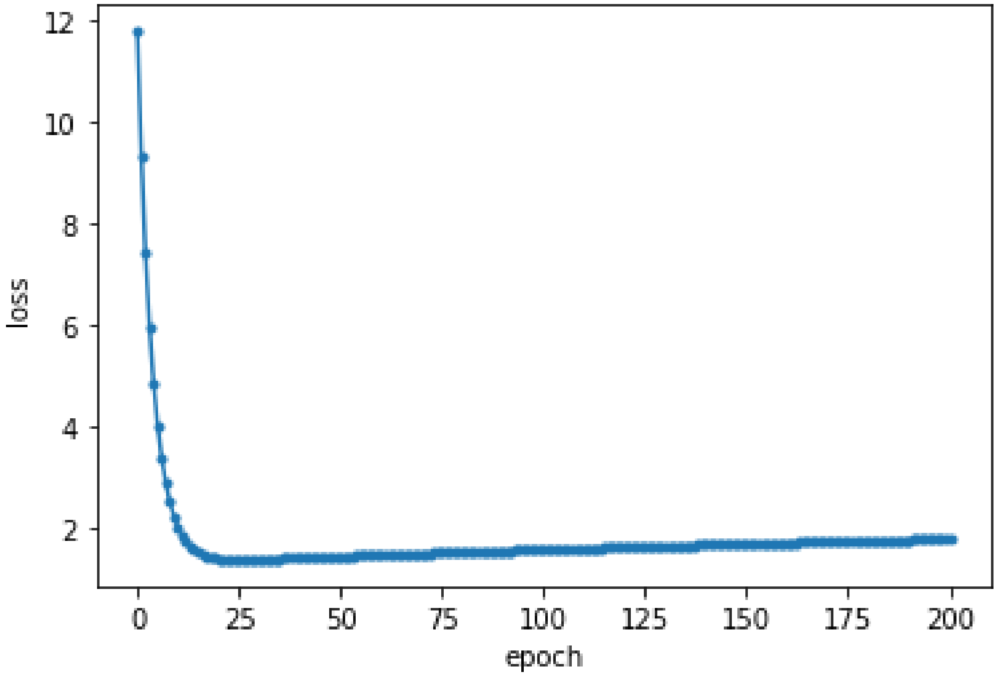


[34]:

```
plt.plot(vw,'.-')
plt.xlabel('epoch')
plt.ylabel('loss')
```

[34]:

```
Text(0, 0.5, 'loss')
```



[29]:

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

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

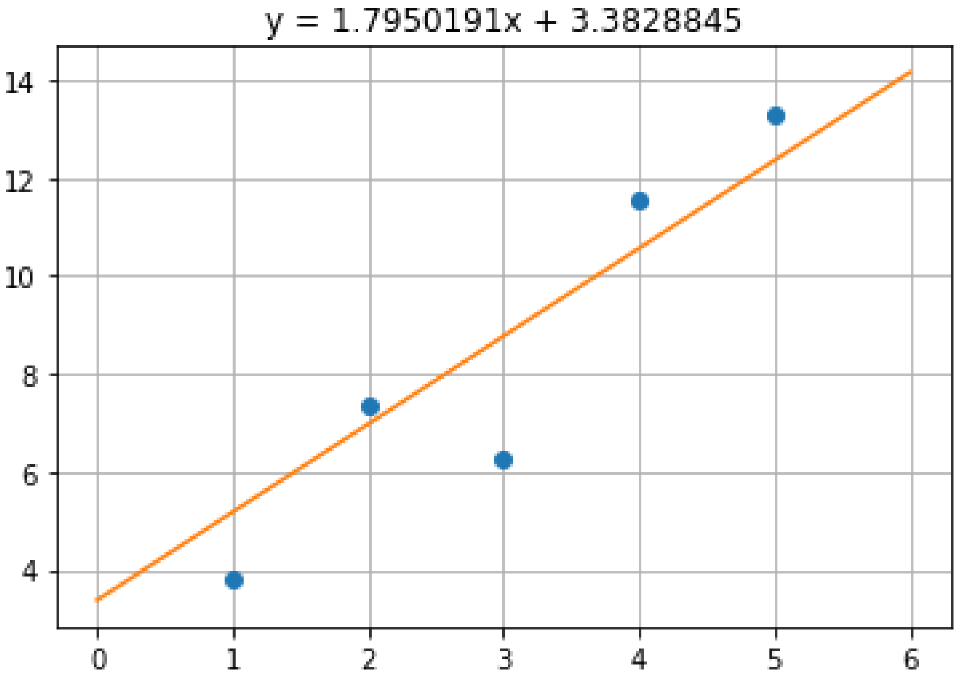
```
1.7950191 3.3828845
y = 1.7950191x + 3.3828845
```

[30]:

```
plt.figure(figsize=(6,4)) # figsize
plt.plot(x_train, y_train,'o') #train data

#
#  x
x1 = np.linspace(np.min(x_train)-1, np.max(x_train)+1)
y1 = w1*x1 + b1
```

```
plt.plot(x1, y1)

plt.grid() #
#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)
```

[30]:
```
Text(0.5, 1.0, 'y = 1.7950191x + 3.3828845')
```



[30]: