01_linear_regression_using_tensorflow

September 28, 2020

Linear regression ,   TensorFlow   notebook .

Import

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

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

X and Y data

[ ]:
```
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]
```

Initialization

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

[ ]:
```
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 XW+b

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

cost/loss function * loss of one training example :

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

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

Optimizer

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

Launch the graph in a session

[ ]:
```
sess = tf.Session()
```

Initializes global variables in the graph.

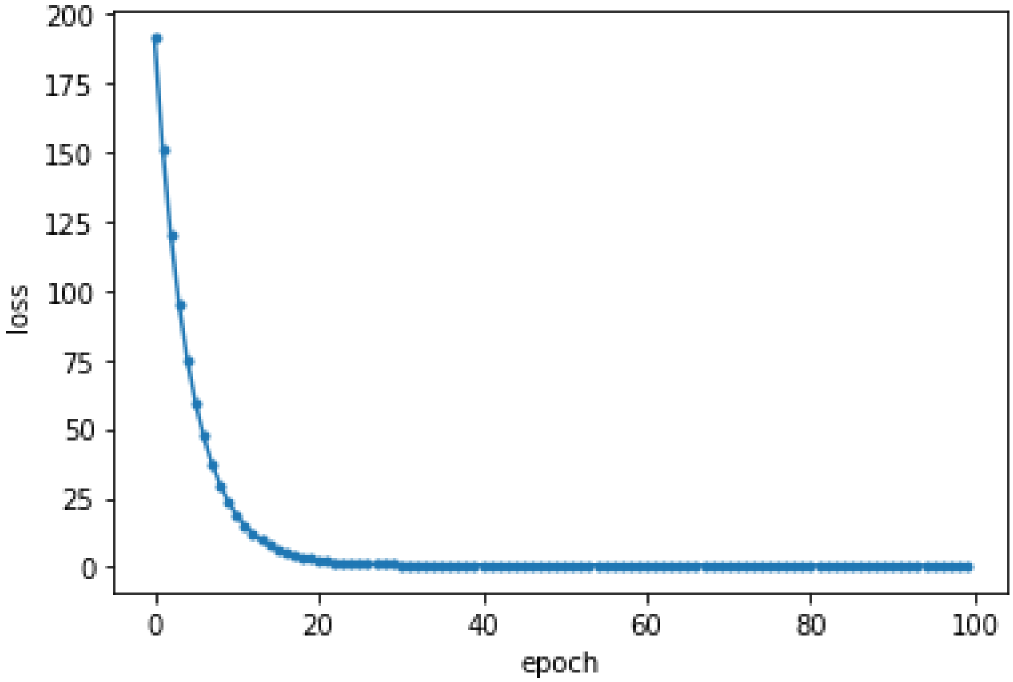
[ ]:
```
sess.run(tf.global_variables_initializer())
```

[ ]:
```
nb_epoch = 2001
vloss = []
vw = []
vb = []

for step in range(nb_epoch):
    sess.run(train)
    loss1 = sess.run(loss)
    vloss.append(loss1)
    vw.append(w1)
    vb.append(b1)

    if step % 200 == 0: # 200
        w1 = sess.run(W)[0] #
        b1 = sess.run(b)[0] # bias

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

```
0        191.49957       6.333   4.6996665
200      0.35716426      1.3710524       1.41991
400      0.16119462      1.6209003       0.8519472
600      0.08636397      1.7752908       0.5009811
800      0.057789873     1.870695        0.2841049
1000     0.046878874     1.9296489       0.15008868
1200     0.042712513     1.966079        0.06727501
1400     0.041121617     1.9885904       0.016101157
1600     0.040514123     2.0025008       -0.015521015
1800     0.040282175     2.011097        -0.035061836
2000     0.0401936       2.016409        -0.047137048
```

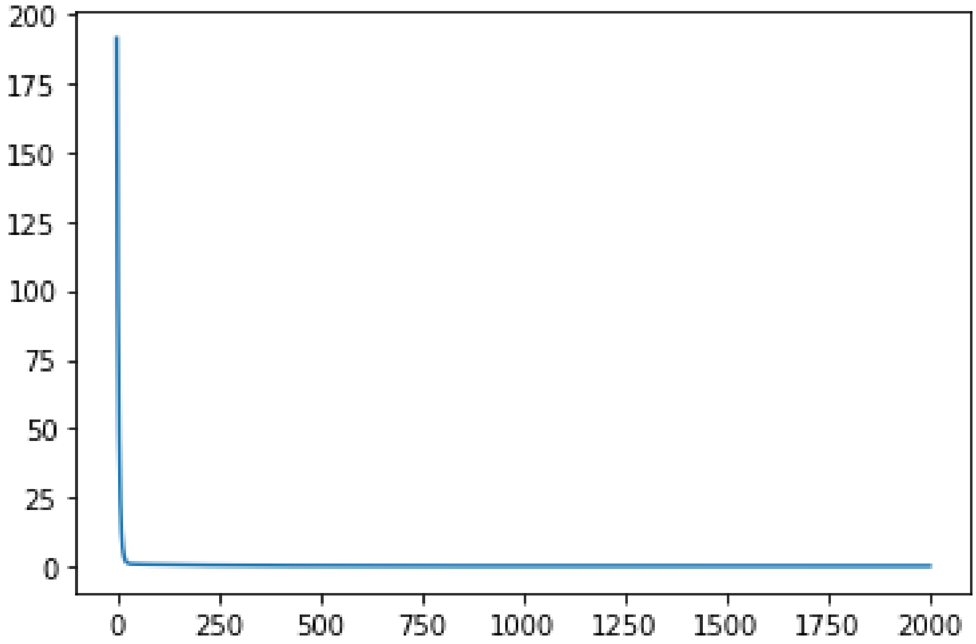
[ ]:
```
plt.plot(vloss[:100],'.-')
plt.xlabel('epoch')
plt.ylabel('loss')
```

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



[ ]:
```
plt.plot(vloss)
```

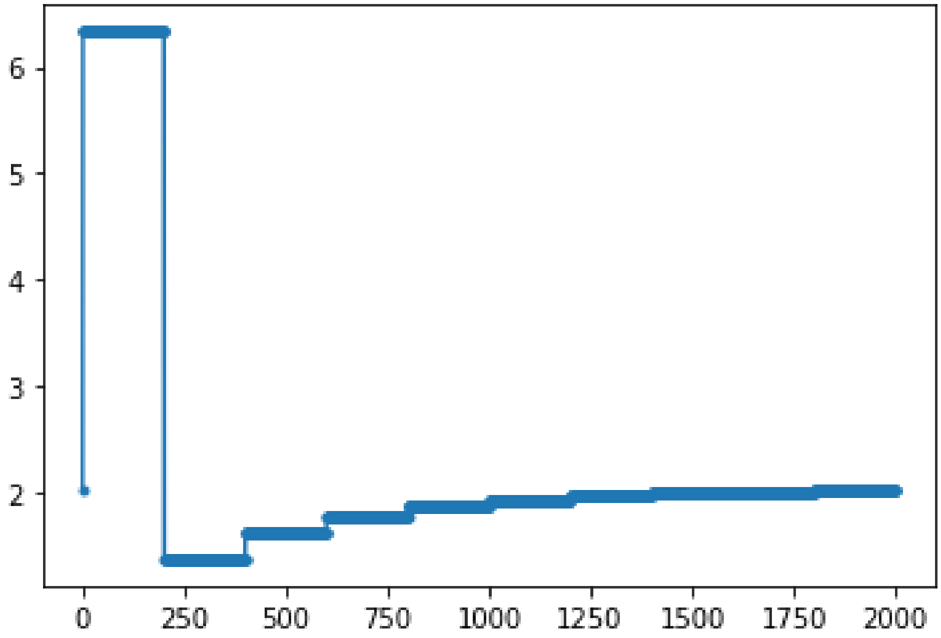
[ ]:
```
[<matplotlib.lines.Line2D at 0x7ff7a9ba1978>]
```



TODO : w,b epoch

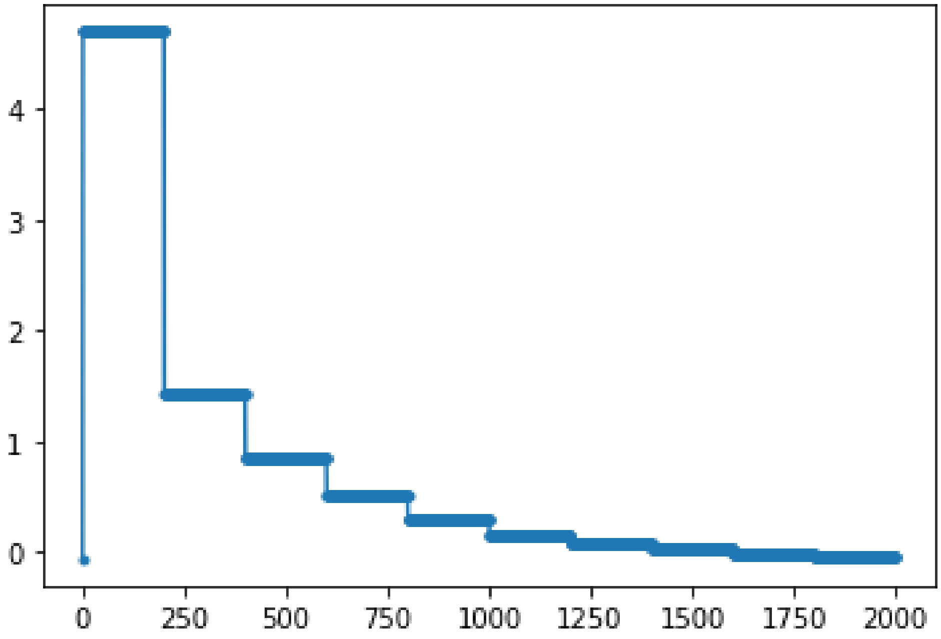
```
[ ]: plt.plot(vw,'.-')
```

```
[ ]: [<matplotlib.lines.Line2D at 0x7ff7a9b905f8>]
```



```
[ ]: plt.plot(vb,'.-')
```

```
[ ]: [<matplotlib.lines.Line2D at 0x7ff7a9aeec88>]
```



[ ]:

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

[ ]:

```
print(w1, b1)

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

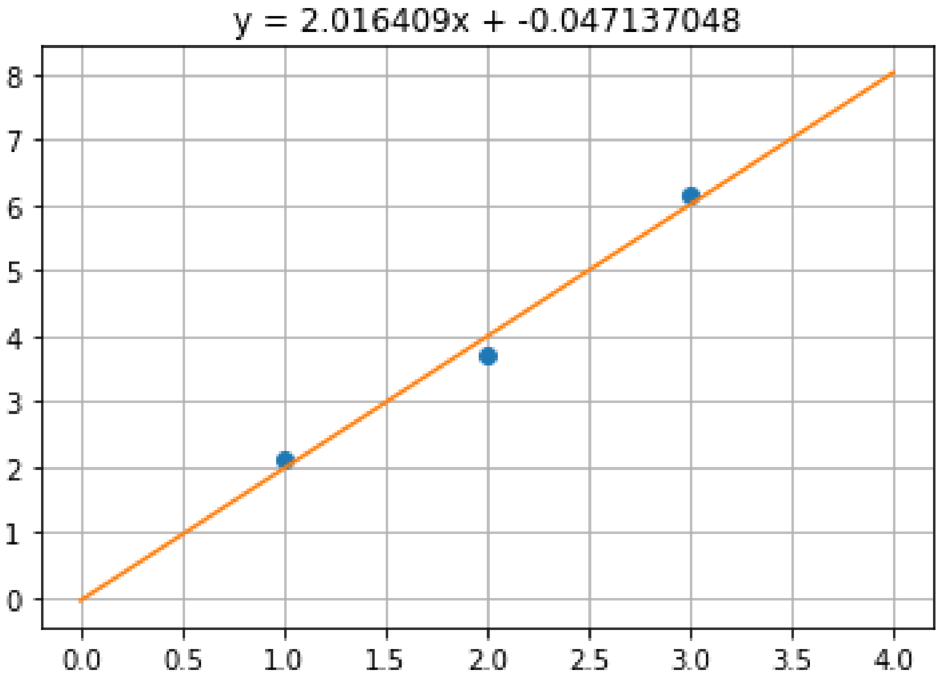
```
2.016409 -0.047137048
y = 2.016409x + -0.047137048
```

[ ]:

```
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)
```

[ ]:

```
Text(0.5, 1.0, 'y = 2.016409x + -0.047137048')
```



0.0.1

~~~
[ ]:       .


* 1)
```
x_train = [1, 2, 3]
y_train = [2+0.1, 4-0.3, 6+0.15] #  noise
```
* 2)           .

* 3)    y=2x+0    , .
```
y=3x-5
y=1.2x + 3
```

* 4)  w0, b0   .
```
w0 = 7.0;
b0 = 5.0;
```
~~~

[ ]:

```
x_train = [1, 3, 5, 7, 9]

y_train = [3+0.15, 9-0.35, 15+0.45, 21+0.15, 27+0.55] #  noise
```

[ ]:

```
#W = tf.Variable(tf.random_normal([1]), name='weight')
#b = tf.Variable(tf.random_normal([1]), name='bias')
w0 = 9.0;
b0 = 3.0;

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

[ ]:

```
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())
```

[ ]:

```
nb_epoch = 2001
vloss = []
vw = []
vb = []

for step in range(nb_epoch):
    sess.run(train)
    loss1 = sess.run(loss)
    vloss.append(loss1)
    vw.append(w1)
    vb.append(b1)

    if step % 200 == 0: # 200
        w1 = sess.run(W)[0] #
        b1 = sess.run(b)[0] # bias

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

```
0        144.32205       4.7694          2.3438
200      0.23507233      2.9370356       0.70350015
400      0.090095565     3.0155125       0.18927136
600      0.06841292      3.0458617       -0.009595188
800      0.06517051      3.0575986       -0.08650231
1000     0.06468534      3.0621376       -0.11624444
1200     0.064612985     3.0638928       -0.12774654
1400     0.06460214      3.0645719       -0.13219467
1600     0.064600244     3.0648344       -0.13391475
```

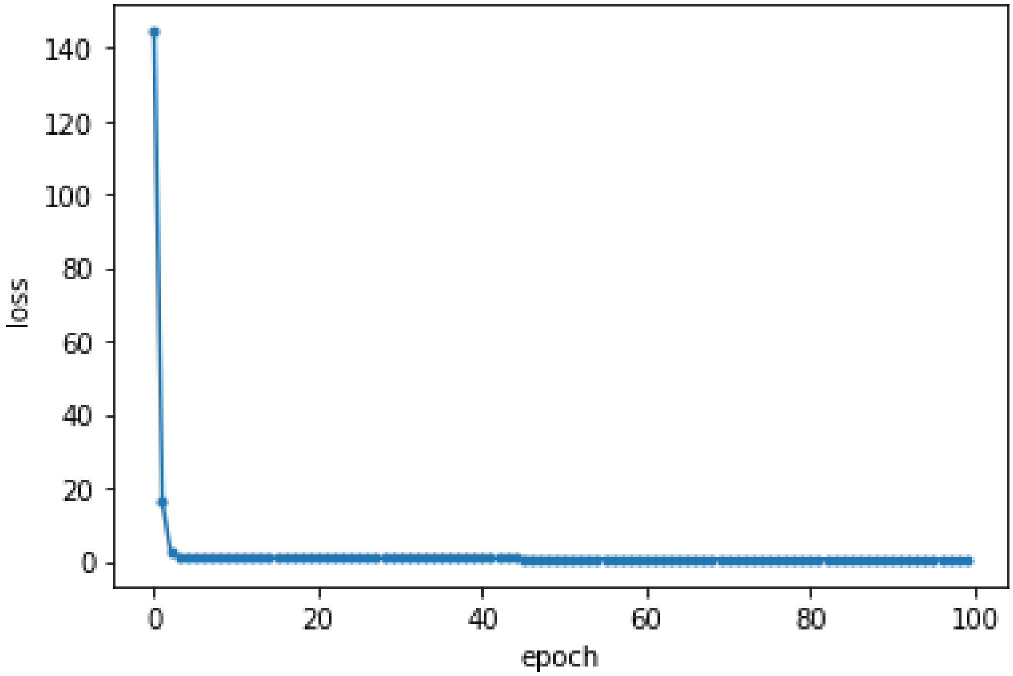
```
1800     0.06460018     3.064936       -0.13458003
2000     0.06460004     3.064975       -0.13483758
```

[ ]:

```
plt.plot(vloss[:100],'.-')
plt.xlabel('epoch')
plt.ylabel('loss')
```

[ ]:

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

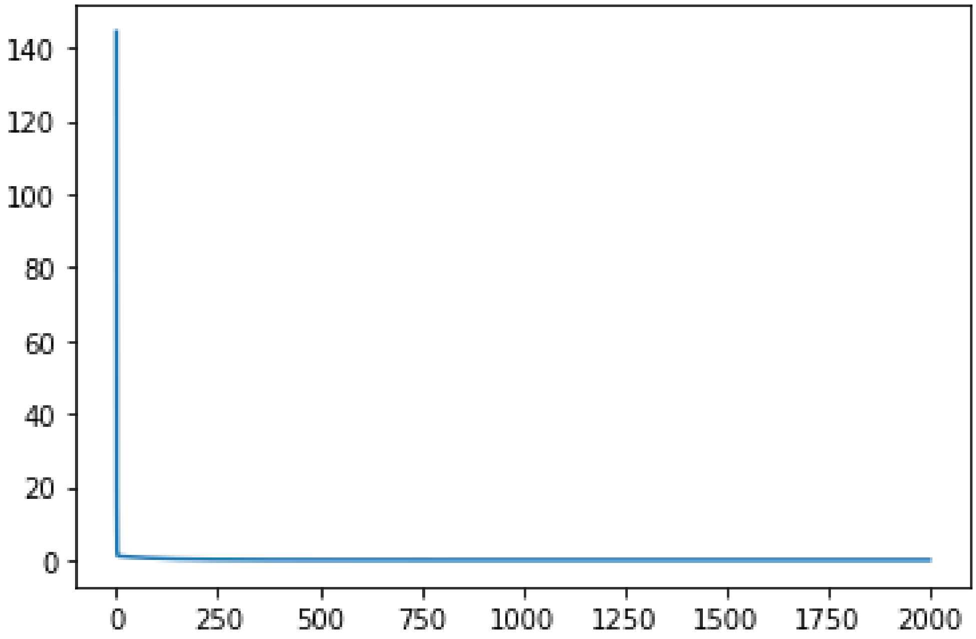


[ ]:

```
plt.plot(vloss)
```

[ ]:

```
[<matplotlib.lines.Line2D at 0x7ff7aa1f1b70>]
```



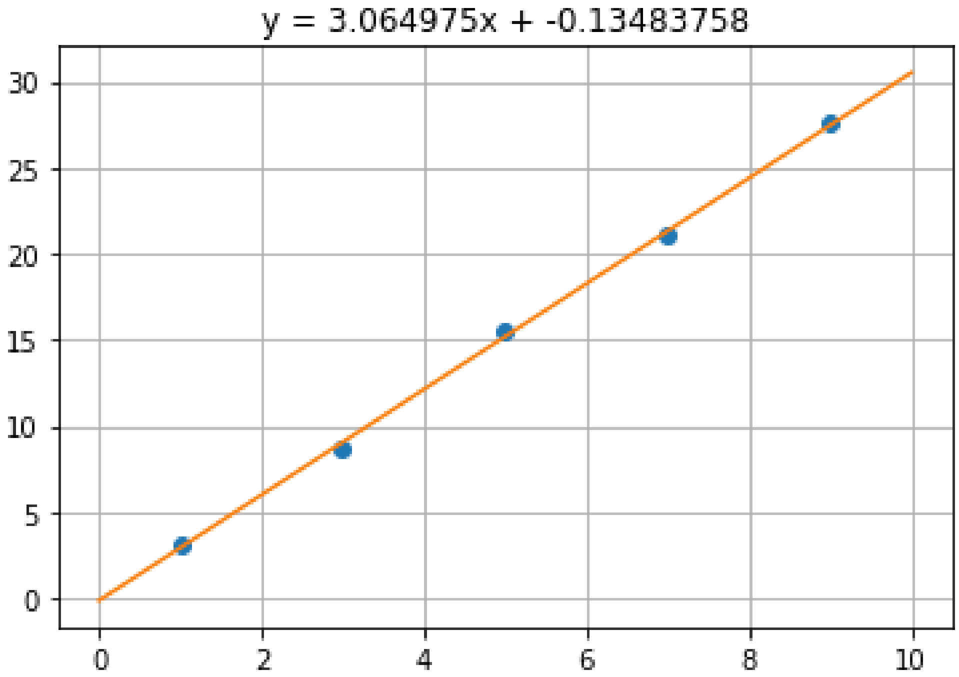
[ ]:
```
w1 = sess.run(W)[0] #
b1 = sess.run(b)[0] # bias
```

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

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

```
3.064975 -0.13483758
y = 3.064975x + -0.13483758
```

[ ]:
```
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)
```

[ ]:
```
Text(0.5, 1.0, 'y = 3.064975x + -0.13483758')
```


y = 3.064975x + -0.13483758
30
25
20
15
10
5
0
0
2
4
6
8
10

[ ]: