01_linear_regression_using_tensorflow_homework

September 27, 2020

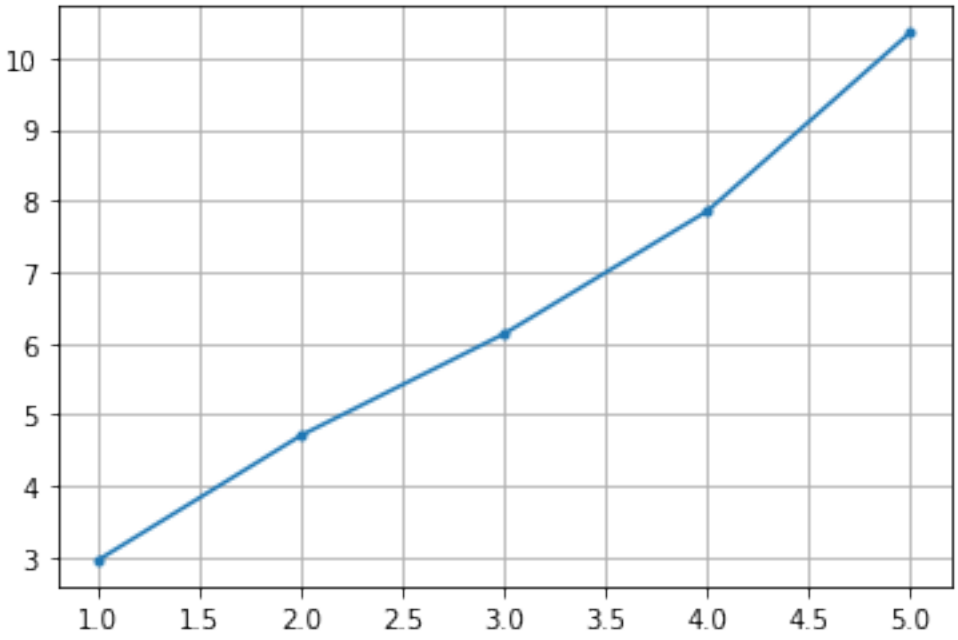
Import

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

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

X and Y data (given)

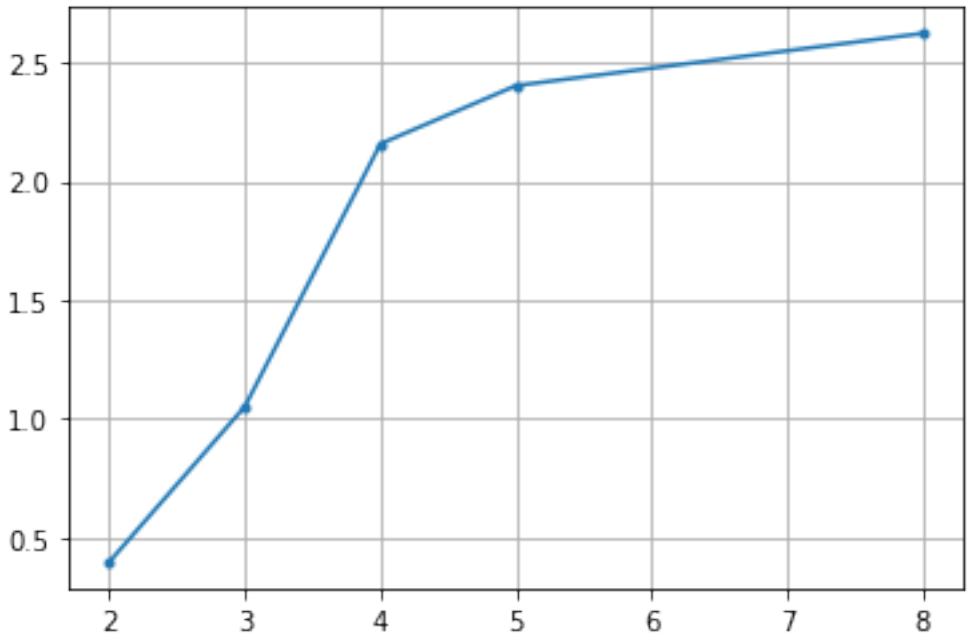
[ ]:
```
x_train = [1, 2, 3, 4, 5]
y_train = [3, 5, 7, 9, 11]

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

y_train = y_train + y_noise
```

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



[ ]:
```
x_train = [2, 3, 4, 5, 8]
y_train = [1.1, 1.6, 2.1, 2.6, 4]

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

y_train = y_train + y_noise
```

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



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 = 10.0;
    b0 = 9.0;

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

Our hypothesis

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

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

cost/loss function

- loss of one training example :

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

[ ]:
```
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())
```

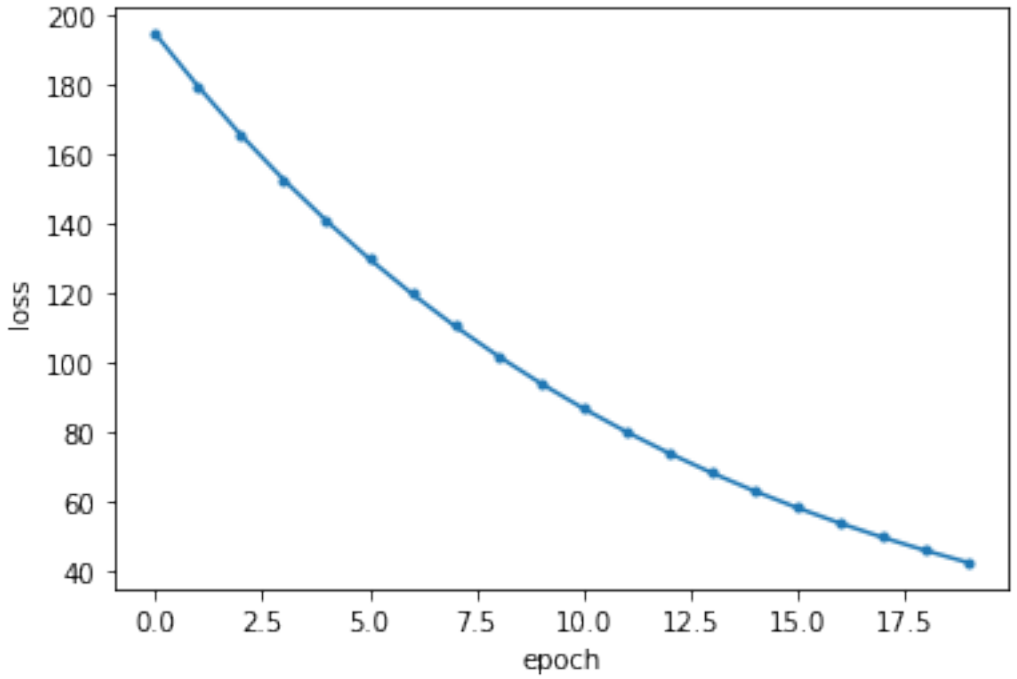
[ ]:
```
nb_epoch = 1001
vloss = []
vb = []
vw = []
for step in range(nb_epoch):
  sess.run(train)
  loss1 = sess.run(loss)
  vloss.append(loss1)

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

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

```
0       194.60141       9.710324        8.710324
50      4.529094        3.6610925       2.661091
100     1.3225114       2.875383        1.8753811
150     1.2684156       2.7733307       1.7733287
200     1.2675029       2.760076        1.7600733
250     1.2674878       2.7583535       1.7583514
300     1.2674873       2.75813         1.7581279
350     1.2674873       2.7581012       1.7580988
400     1.2674873       2.7580993       1.7580953
450     1.2674873       2.7580993       1.7580953
500     1.2674873       2.7580993       1.7580953
550     1.2674873       2.7580993       1.7580953
600     1.2674873       2.7580993       1.7580953
650     1.2674873       2.7580993       1.7580953
```

```
700     1.2674873       2.7580993       1.7580953
750     1.2674873       2.7580993       1.7580953
800     1.2674873       2.7580993       1.7580953
850     1.2674873       2.7580993       1.7580953
900     1.2674873       2.7580993       1.7580953
950     1.2674873       2.7580993       1.7580953
1000    1.2674873       2.7580993       1.7580953
```

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

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



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

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

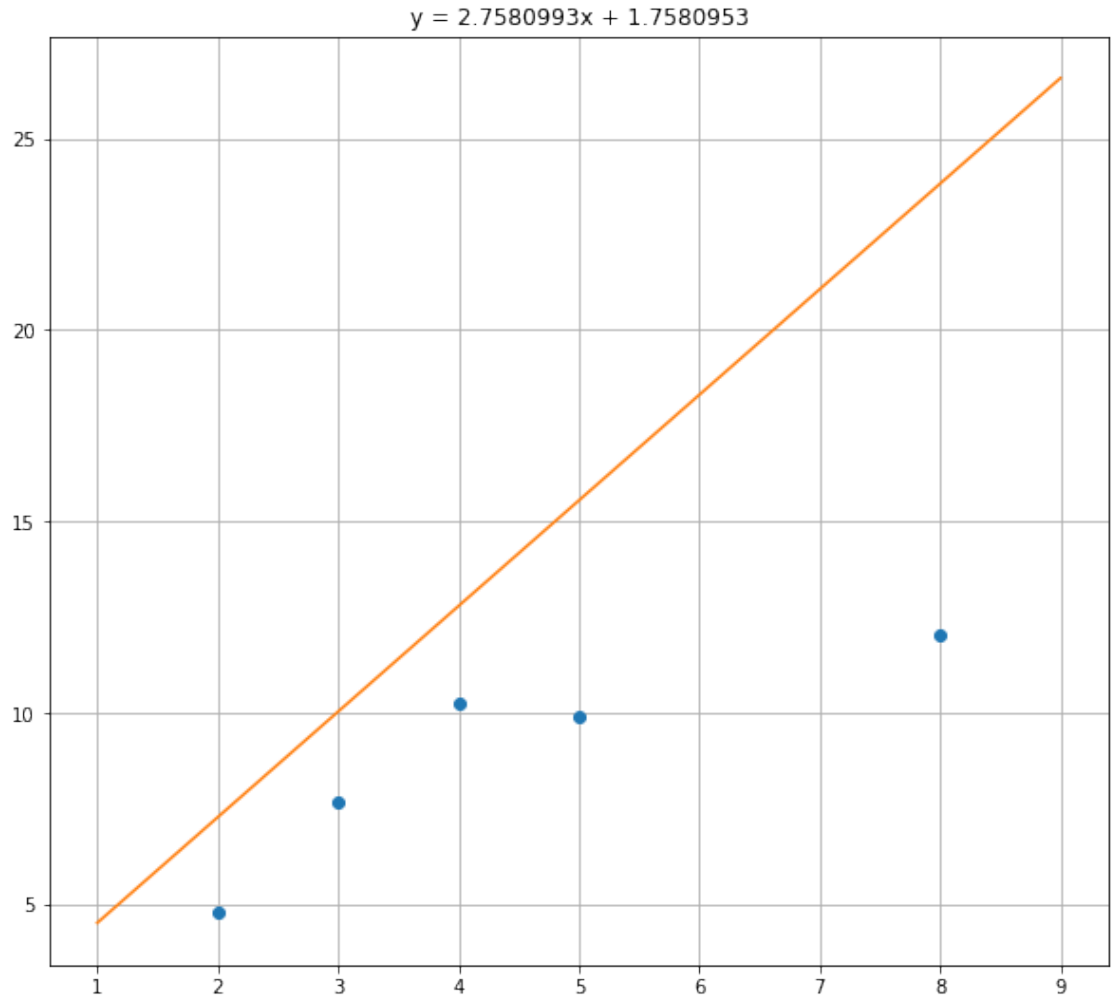
```
2.7580993 1.7580953
```

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

```
y = 2.7580993x + 1.7580953
```

[ ]:
```
plt.figure(figsize=(10,9))
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 = 2.7580993x + 1.7580953')
```



[ ]: