Chapter_3_Section_1_Linear_Regression

February 2, 2019

1 Ch 03: Concept 01

1.1 Linear regression

Import TensorFlow for the learning algorithm. We'll need NumPy to set up the initial data. And we'll use matplotlib to visualize our data.

```
In [1]: %matplotlib inline
        import tensorflow as tf
        import numpy as np
        import matplotlib.pyplot as plt
        np.random.seed(1234)
```

Define some constants used by the learning algorithm. There are called hyper-parameters.

```
In [2]: learning_rate = 0.01
        training_epochs = 100
```

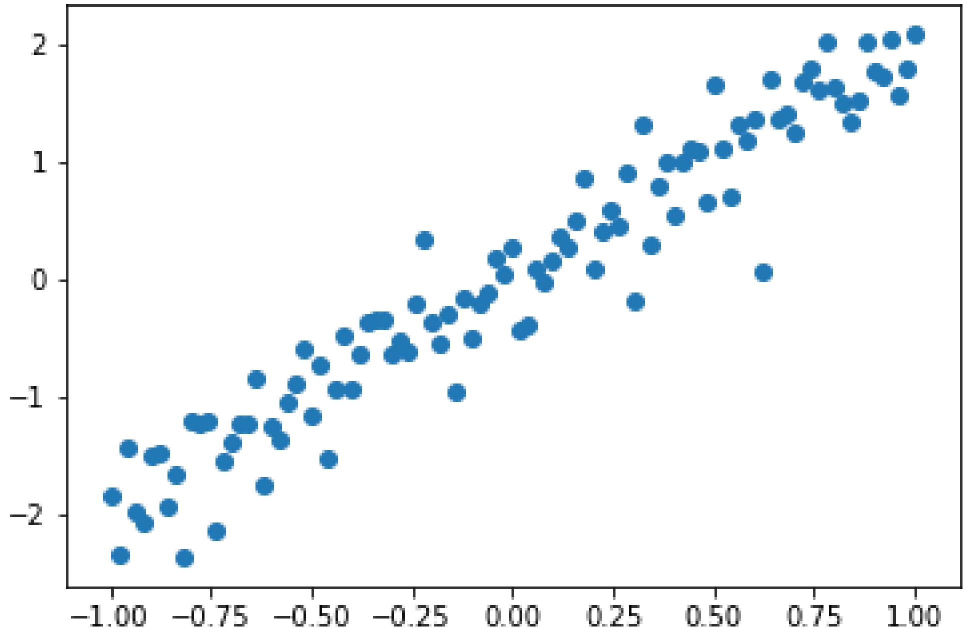
Set up fake data that we will use to to find a best fit line

```
In [3]: x_train = np.linspace(-1, 1, 101)
        y_train = 2 * x_train + np.random.randn(*x_train.shape) * 0.33
```

Plot the raw data

```
In [4]: plt.scatter(x_train, y_train)
```

```
Out[4]: <matplotlib.collections.PathCollection at 0x7f2095a59358>
```



Set up the input and output nodes as placeholders since the value will be injected by `x_train` and `y_train`.

```
In [5]: X = tf.placeholder("float")
        Y = tf.placeholder("float")
```

Define the model as `y = w'*x`

```
In [6]: def model(X, w):
            return tf.multiply(X, w)
```

Set up the weights variable

```
In [7]: w = tf.Variable(0.0, name="weights")
```

Define the cost function as the mean squared error

```
In [8]: y_model = model(X, w)
        cost = tf.reduce_mean(tf.square(Y-y_model))
```

Define the operation that will be called on each iteration of the learning algorithm

```
In [9]: train_op = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
```

Initialize all variables

```
In [10]: sess = tf.Session()
         init = tf.global_variables_initializer()
         sess.run(init)
```

Train on each (x, y) pair multiple times

```
In [11]: for epoch in range(training_epochs):
             for (x, y) in zip(x_train, y_train):
                 sess.run(train_op, feed_dict={X: x, Y: y})
```

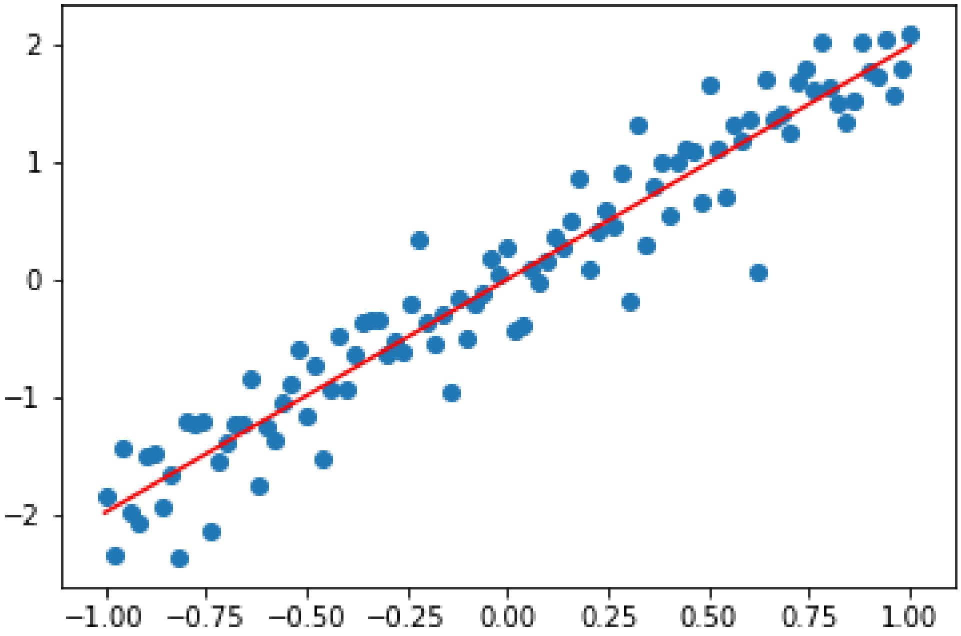
Fetch the value of the learned parameter

```
In [12]: w_val = sess.run(w)
```

```
In [13]: sess.close()
```

Visualize the best fit curve

```
In [14]: plt.scatter(x_train, y_train)
         y_learned = x_train*w_val
         plt.plot(x_train, y_learned, 'r')
         plt.show()
```


2
1
0
−1
−2
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00