Python实验报告

实验十一

学号：2016326603039 姓名:郭仲

学号：2016326603029 姓名:黄欣悦

学号：2016326603071 姓名:王明凯

学号：2016329600130 姓名:王宇舟

学号：2016326603046 姓名:余泓锷

实验内容：

请你修改本实例代码，纯属函数变为y=3x+1，并改变数据集，尝试输出结果。

In [1]:

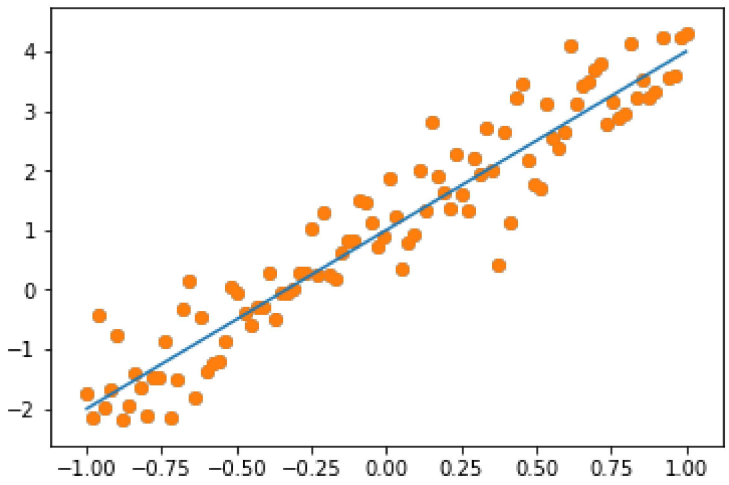
```
#设置inline模式，显示图像
%matplotlib inline
# 载入matplotlib、numpy、Tensorflow
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf

np.random.seed(5)
x_data = np.linspace(-1, 1,100) #生成-1~1之间100个点
y_data = 3*x_data + 1.0 + np.random.randn(*x_data.shape)*0.6   # y = 3x +1 + 噪声，
plt.figure()
plt.scatter(x_data, y_data)          #画出随机生成数据的散点图
plt.scatter(x_data, y_data)
plt.plot (x_data,3 * x_data+1.0) # 画出线性函数 y = 3x +1
```

```
C:\Users\710S\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conve
rsion of the second argument of issubdtype from `float` to `np.floating` is depreca
ted. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
```

Out[1]:

```
[<matplotlib.lines.Line2D at 0x21d3cf38470>]
```



In [2]:

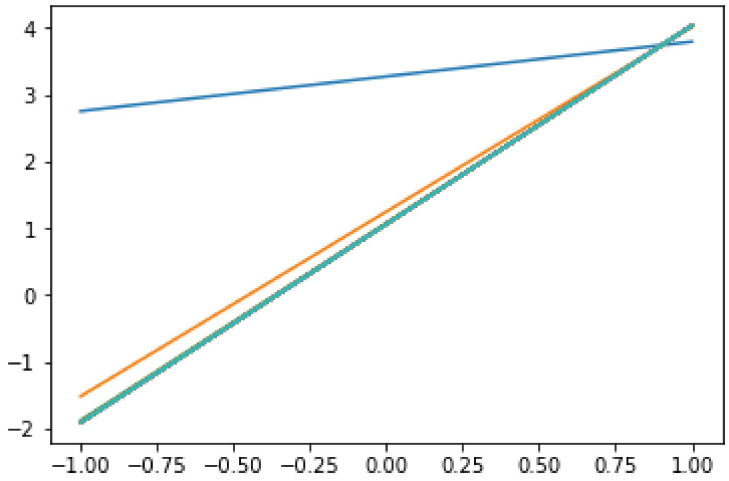
```
#构建模型
x = tf.placeholder("float", name = "x")
y = tf.placeholder("float", name = "y")
def model(x, w, b):
    return tf.multiply(x, w) + b
w = tf.Variable(-1.5, name="w0")  # 斜率
b = tf.Variable(0.0, name="b0")   # 截距
pred = model(x, w, b)             # 预测值
```

In [3]:

```
#训练模型
train_epochs = 30    # 迭代次数
learning_rate = 0.05 #学习率
loss_function = tf.reduce_mean(tf.square(y-pred))  # 采用均方差作为损失函数
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(loss_function)# 梯度下降
sess = tf.Session()                          #声明会话
init = tf.global_variables_initializer() #变量初始化
```

In [4]:

```
#执行训练
sess.run(init)
for epoch in range(train_epochs):
    for xs,ys in zip(x_data, y_data):
        _, loss=sess.run([optimizer,loss_function], feed_dict={x: xs, y: ys})
    b0temp=b.eval(session=sess)
    w0temp=w.eval(session=sess)
    plt.plot (x_data, b0temp + w0temp * x_data )
```



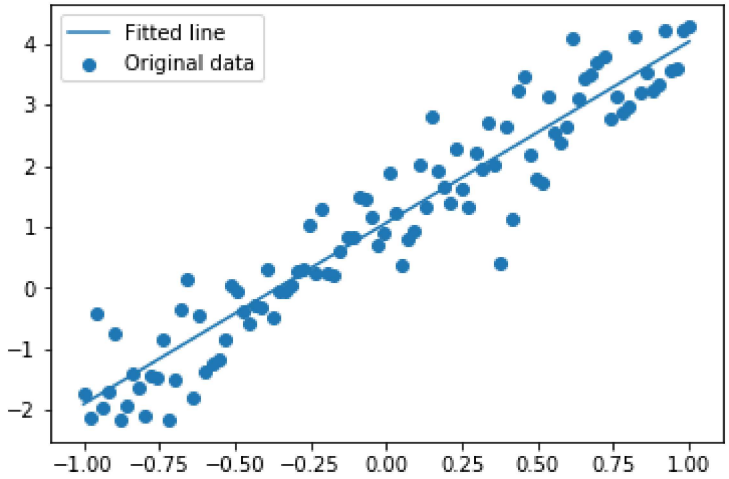
In [5]:

```
#结果认证
print  ("w: ", sess.run(w)) # w应约为3
print  ("b: ", sess.run(b)) # b应约为1
plt.scatter(x_data,y_data,label='Original data')
plt.plot (x_data, x_data * sess.run(w) + sess.run(b),label='Fitted line')
plt.legend(loc=2)# 通过参数loc指定图例位置
```

```
w:  2.9734433
b:  1.0630201
```

Out[5]:

```
<matplotlib.legend.Legend at 0x21d3d39a7f0>
```



In [6]:

```
#进行预测
x_test1 =2.0
x_test2 =10.0
x_test3 =50.0
output1 = round(sess.run(w)  * x_test1 + sess.run(b),5)
output2 = round(sess.run(w)  * x_test2 + sess.run(b),5)
output3 = round(sess.run(w)  * x_test3 + sess.run(b),5)
print("预测值: " , output1 , output2 , output3)
target1 = 3 * x_test1 + 1.0
target2 = 3 * x_test2 + 1.0
target3 = 3 * x_test3 + 1.0
print("目标值: " , target1 , target2 ,target3 )
```

```
预测值:  7.00991 30.79745 149.73518
目标值:  7.0 31.0 151.0
```

实验分析:

通过实验结果，预测值与目标值相差仅0.009907，0.20255，1.26482，均是是较小的值，说明构建的模型是合理的。

其中，需要注意的是：关于学习率的设置 ：学习率控制参数更新的幅度。如果学习率设置过大，可能导致参数在极值附近来回摇摆，无法保证收敛。如果设置过小，虽然能保证收敛，但优化速度会大大降低，我们需要更多迭代次数才能达到较理想的优化效果。