```python
# -*- coding: utf-8 -*-
"""
Created on Tue Oct  9 10:27:10 2018

@author: lxr
"""

#导入相关模块
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import math

'''
python的math与np.math中的sinx输入与输出不一样，np的是矩阵数组运算
'''

#====================================
#定义常量
seed=12355
BATCH_SIZE=8
count=32
#====================================


#生成训练集随机数sinx
rmd=np.random.RandomState(seed)

#生成数据集第一维
X=rmd.rand(count,1)*2*np.pi
X_s=np.sort(X,axis=0)

print(X)
print(X_s)

#生成数据集第二维
X2=np.hstack((X,pow(X,2)))
print(X2)

#生成数据集第三维
X3=np.hstack((X2,pow(X,3)))
print(X3)


#标准值参杂噪声
Y_=[[math.sin(x)+rmd.rand()/2-0.25] for x in X]
print(Y_)
#生成初始标准点集
plot=np.hstack((X,Y_))
plot=np.vstack(plot).reshape(-1,2)
#打印矩阵plot
print(plot)



'''
#排序后的点集
index=np.argsort(plot[:,0])
plot_=plot[index]
'''
```

```python
#生成正弦函数
sin_x=np.linspace(0,2*np.pi,num=50)
sin_y=np.sin(sin_x)



#显示需要的模拟函数图像
plt.scatter(plot[:,0],plot[:,1])
plt.plot(sin_x,sin_y,linestyle='--')


'''
plt.plot(plot_[:,0],plot_[:,1])
'''

#标注
plt.Annotation(r'$y=ax+b$',xy=(sin_x[25],sin_y[25]),xycoords='data',xytext=(+30,-30),te

#设置横纵坐标轴取值的范围
plt.xlim((0,2*np.pi))
plt.ylim((-1,1))
#设置横纵坐标轴的刻度和标识以及样式
plt.xticks([0,np.pi/2,np.pi,3/2*np.pi,2*np.pi],['0','1/2π','π','3/2π','2π'])
#设置横纵坐标轴的位置
ax=plt.gca()
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
ax.spines['bottom'].set_position(('data',0.00))


plt.show()
#==================================

#定义一次函数计算模型
def model(x,w,b):
    y=tf.matmul(x,w)+b
    return y
 


#定义返回参数
def gw(shape):
    w=tf.Variable(tf.random_normal(shape,mean=0,stddev=1,seed=1))
    return w


def get_bias(shape):
    b=tf.Variable(tf.random_normal(shape,mean=0,stddev=1,seed=2))
    return b
#==================================

#建立前向传播
#一次函数
x1=tf.placeholder(tf.float32,shape=(None,1))
#二次函数
x2=tf.placeholder(tf.float32,shape=(None,2))
#三次函数
```

```
x3=tf.placeholder(tf.float32,shape=(None,3))
#标准值占位
y_=tf.placeholder(tf.float32,shape=(None,1))

#一次函数
w1=gw([1,1])
b1=get_bias([1,1])
y1=model(x1,w1,b1)

#二次函数
w2=gw([2,1])
b2=get_bias([1,1])
y2=model(x2,w2,b2)

#二次函数
w3=gw([3,1])
b3=get_bias([1,1])
y3=model(x3,w3,b3)
#=================================

#建立反向传播
loss1=tf.reduce_mean(tf.square(y1-y_))
train_step1=tf.train.GradientDescentOptimizer(0.001).minimize(loss1)


loss2=tf.reduce_mean(tf.square(y2-y_))
train_step2=tf.train.GradientDescentOptimizer(0.001).minimize(loss2)

loss3=tf.reduce_mean(tf.square(y3-y_))
train_step3=tf.train.GradientDescentOptimizer(0.0001).minimize(loss3)


#===================================
#一次函数
#建立会话
with tf.Session() as sess:
    init_op=tf.global_variables_initializer()
    sess.run(init_op)
    
    #建立训练模型
    STEPS=3000
    for i in range(STEPS):
        start=i*BATCH_SIZE % count
        end=i*BATCH_SIZE %count + BATCH_SIZE
        sess.run(train_step1,feed_dict={x1:X[start:end],y_:Y_[start:end]})
        if i % 500==0:
            print('after',i,' training is:\n')
            print('w1:\n',sess.run(w1))
            print('b1:\n',sess.run(b1))
    line1_x=np.linspace(0,2*np.pi,num=50)
    print(line1_x)
    line1_y=sess.run(w1)*line1_x+sess.run(b1)
    print(line1_y)
    

#设置横纵坐标轴取值的范围
plt.xlim((0,2*np.pi))
plt.ylim((-1,1))
#设置横纵坐标轴的刻度和标识以及样式
plt.xticks([0,np.pi/2,np.pi,3/2*np.pi,2*np.pi],['0','1/2π','π','3/2π','2π'])
```

```
#设置横纵坐标轴的位置
ax=plt.gca()
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
ax.spines['bottom'].set_position(('data',0.00))

#画出图像
plt.scatter(plot[:,0],plot[:,1])
plt.plot(line1_x,line1_y[0,:])
plt.plot(sin_x,sin_y,linestyle='--')


plt.show()

#==================================



#二次函数
#建立会话
with tf.Session() as sess:
    init_op=tf.global_variables_initializer()
    sess.run(init_op)

    #建立训练模型
    STEPS=10000
    for i in range(STEPS):
        start=i*BATCH_SIZE % count
        end=i*BATCH_SIZE %count + BATCH_SIZE
        sess.run(train_step2,feed_dict={x2:X2[start:end],y_:Y_[start:end]})
        if i % 500==0:
            print('after',i,' training is:\n')
            print('w1:\n',sess.run(w2))
            print('b1:\n',sess.run(b2))
    print('最后的w2,b2的结果为：',sess.run(w2),sess.run(b2))



    #测试集
    line2_x=np.linspace(0,2*np.pi,num=50)
    print(line2_x)
    line2_y=pow(line2_x,2)*sess.run(w2)[1,0]+line2_x*sess.run(w2)[0,0]+sess.run(b2)
    print(line2_y)


#设置横纵坐标轴取值的范围
plt.xlim((0,2*np.pi))
plt.ylim((-1,1))
#设置横纵坐标轴的刻度和标识以及样式
plt.xticks([0,np.pi/2,np.pi,3/2*np.pi,2*np.pi],['0','1/2π','π','3/2π','2π'])
#设置横纵坐标轴的位置
ax=plt.gca()
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
ax.spines['bottom'].set_position(('data',0.00))

```

```
#画出图像
plt.scatter(plot[:,0],plot[:,1])
plt.plot(line2_x,line2_y[0,:])
plt.plot(sin_x,sin_y,linestyle='--')


plt.show()


#==========================================


#三次函数
#建立会话
with tf.Session() as sess:
    init_op=tf.global_variables_initializer()
    sess.run(init_op)

    #建立训练模型
    STEPS=600000
    for i in range(STEPS):
        start=i*BATCH_SIZE % count
        end=i*BATCH_SIZE %count + BATCH_SIZE
        sess.run(train_step3,feed_dict={x3:X3[start:end],y_:Y_[start:end]})
        if i % 500==0:
            print('after',i,' training is:\n')
            print('w1:\n',sess.run(w3))
            print('b1:\n',sess.run(b3))
    print('最后结果w1和b1的结果为: ',sess.run(w3),sess.run(b3))
    line3_x=np.linspace(0,2*np.pi,num=50)
    print(line3_x)
    line3_y=pow(line3_x,3)*sess.run(w3)[2,0]+pow(line3_x,2)*sess.run(w3)[1,0]+line3_x*s
    print(line3_y)


#设置横纵坐标轴取值的范围
plt.xlim((0,2*np.pi))
plt.ylim((-1,1))
#设置横纵坐标轴的刻度和标识以及样式
plt.xticks([0,np.pi/2,np.pi,3/2*np.pi,2*np.pi],['0','1/2π','π','3/2π','2π'])
#设置横纵坐标轴的位置
ax=plt.gca()
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
ax.spines['bottom'].set_position(('data',0.00))

#画出图像
plt.scatter(plot[:,0],plot[:,1])
plt.plot(line3_x,line3_y[0,:])
plt.plot(sin_x,sin_y,linestyle='--')


plt.show()

#=================================================================
#打印总图
#设置横纵坐标轴取值的范围
print('\n')
print('一次函数，二次函数，三次函数，模拟情况总图如下:\n')
plt.xlim((0,2*np.pi))
plt.ylim((-1,1))
#设置横纵坐标轴的刻度和标识以及样式
plt.xticks([0,np.pi/2,np.pi,3/2*np.pi,2*np.pi],['0','1/2π','π','3/2π','2π'])
#设置横纵坐标轴的位置
ax=plt.gca()
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
ax.spines['bottom'].set_position(('data',0.00))

#曲线标注
plt.Annotation(r'$y=ax+b$',xy=(line1_x[25],line1_y[0,25]),xycoords='data',xytext=(+30,-
plt.Annotation(r'$y=ax+b$',xy=(line2_x[25],line2_y[0,25]),xycoords='data',xytext=(+30,-
plt.Annotation(r'$y=ax+b$',xy=(line3_x[25],line3_y[0,25]),xycoords='data',xytext=(+30,-

plt.scatter(plot[:,0],plot[:,1])
plt.plot(line1_x,line1_y[0,:])
plt.plot(line2_x,line2_y[0,:])
plt.plot(line3_x,line3_y[0,:])
plt.plot(sin_x,sin_y,linestyle='--')


plt.show()
```