深度学习方法与实践第二次作业

姓名：杨玉雷
学号：18023040

1.基础作业

假设有函数 y = cos(ax + b)，其中 a 为学号前两位，b 为学号最后两位。首先从此函数中以相同步长（点与点之间在 x 轴上距离相同），在 0<(ax+b)<2pi 范围内，采样出 2000 个点，然后利用采样的 2000 个点作为特征点进行三次函数拟合(三次函数形式为 y = w1 * x + w2 * x^2 + w3 * x^3 + b，其中 wi 为可训练的权值，b 为可训练的偏置值，x 和 y 为输入的训练数据 ）。要求使用 TensorFlow 实现三次函数拟合的全部流程。拟合完成后，分别使用 ckpt 模式和 PB 模式保存拟合的模型。然后，针对两种模型存储方式分别编写恢复模型的程序。两个模型恢复程序分别使用 ckpt 模式和 PB 模式恢复模型，并将恢复的模型参数（wi 和 b）打印在屏幕上，同时绘制图像（图像包括所有的 2000 个采样点及拟合的函数曲线）。

请提交文档，内容包括模型参数配置、程序运行截图、拟合的三次函数以及绘制的图像，同时提交三个 python 脚本文件：1. 函数拟合及两种模式的模型保存程序 2. ckpt 模型恢复程序，3. PB 模型恢复程序。

实验过程和结果

1.1 函数拟合三次曲线

本实验按如下流程进行实验，并附上关键代码。

(1) 定义 Tensorflow 输入节点

```
a=18 #学号前两位
b=40 #学号后两位
step=((2*math.pi-b)/a+b/a)/2000 #计算步长
x=np.arange(2000.)
y=np.arange(2000.)
for i in range(0,2000):
    x[i]=-b/a+i*step
    y[i]=math.cos(a*x[i]+b)
```

(2) 定义“学习参数”的变量

```
w1= tf.Variable(tf.random_uniform([1]),name='w1')
w2= tf.Variable(tf.random_uniform([1]),name='w2')
w3= tf.Variable(tf.random_uniform([1]),name='w3')
b= tf.Variable(tf.random_uniform([1]),name='b')
```

(3) 定义“运算”

```
# 梯度下降法优化参数(学习率为 lr)
```

```
lr = 0.01
# 定义 loss:均方差
loss = tf.reduce_mean(tf.square(y - y_))
```

(4) 优化函数，优化目标

```
optimizer = tf.train.GradientDescentOptimizer(lr)
# 训练的过程就是最小化这个误差值
train = optimizer.minimize(loss)
```

(5) 初始化所有变量

```
#初始化全局变量
init = tf.initialize_all_variables()
# 通过 session 执行上述操作
sess = tf.Session()
sess.run(init)
```

(6) 迭代更新参数到最优解

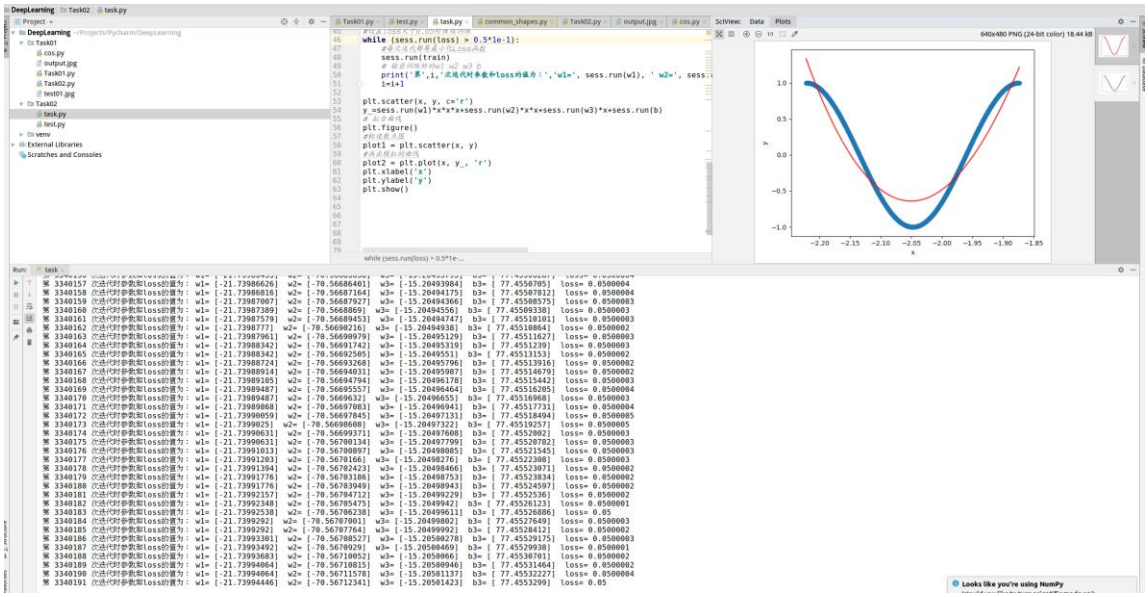
```
i=0
#设置 loss 大于 0.05 时继续训练
while (sess.run(loss) > 0.5*1e-1):
    #每次迭代都要最小化 Loss 函数
    sess.run(train)

    # 输出训练好的 w1 w2 w3 b
    print('第',i,'次迭代时参数和 loss 的值为：','w1=', sess.run(w1), ' w2=',
sess.run(w2), ' w3=', sess.run(w3), ' b3=', sess.run(b), '
loss=',sess.run(loss))
    i=i+1
```

(7) 绘制图像

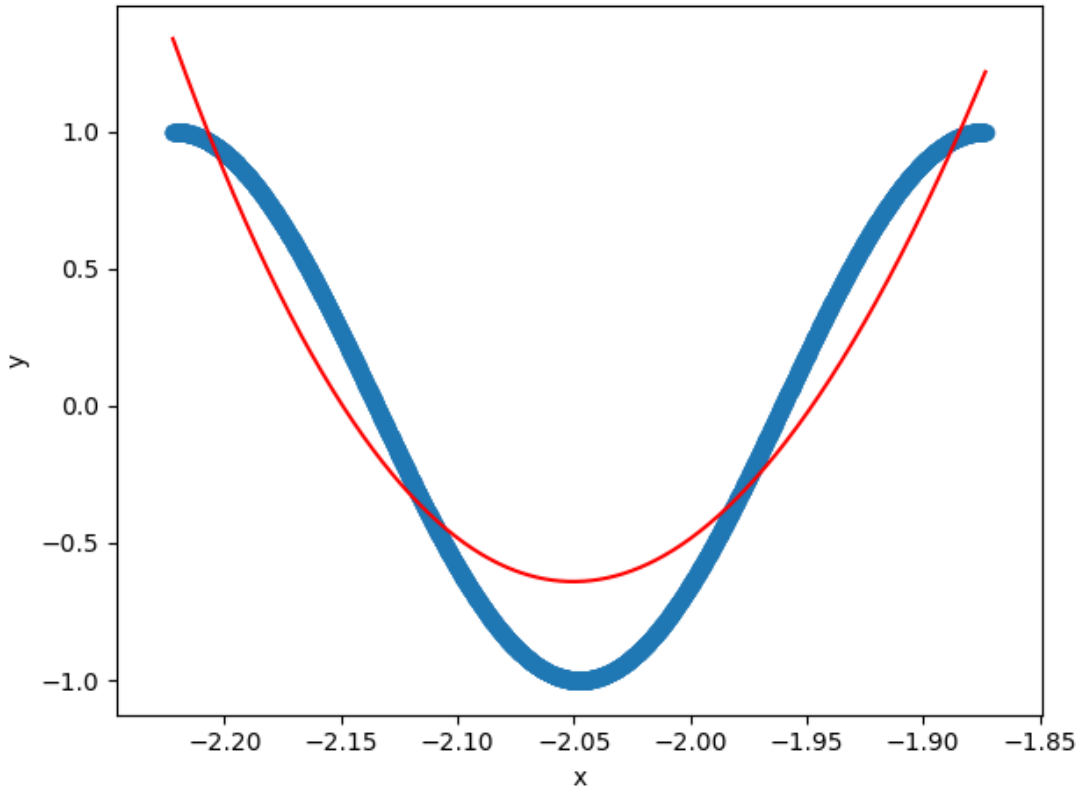
```
plt.scatter(x, y, c='r')
y_=sess.run(w1)*x*x*x+sess.run(w2)*x*x+sess.run(w3)*x+sess.run(b)
# 拟合曲线
plt.figure()
#构造散点图
plot1 = plt.scatter(x, y)
#画出模拟的曲线，红色部分
plot2 = plt.plot(x, y_, 'r')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

由于参数（学习率）和迭代终止条件的设置不同，实验结果也不同。在学习率为 0.01，迭代终止条件为 loss 小于等于 0.05 时，迭代了 3340191 次后实验

结束。图像如下：

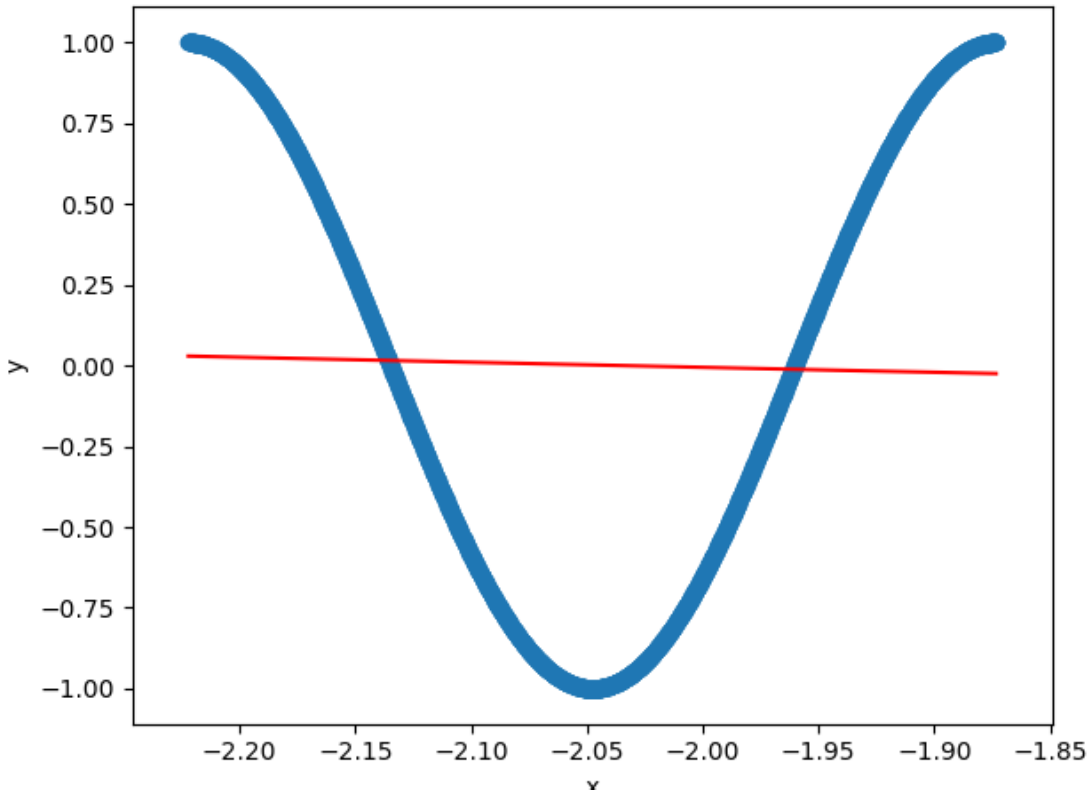


```
第 3340157 次迭代时参数和loss的值为： w1= [-21.73986626]  w2= [-70.56686401]  w3= [-15.20493984]  b3= [ 77.4550705]  loss= 0.0500004
第 3340158 次迭代时参数和loss的值为： w1= [-21.73986816]  w2= [-70.56687164]  w3= [-15.20494175]  b3= [ 77.45507812]  loss= 0.0500004
第 3340159 次迭代时参数和loss的值为： w1= [-21.73987007]  w2= [-70.56687927]  w3= [-15.20494366]  b3= [ 77.45508575]  loss= 0.0500003
第 3340160 次迭代时参数和loss的值为： w1= [-21.73987389]  w2= [-70.5668869]  w3= [-15.20494556]  b3= [ 77.45509338]  loss= 0.0500003
第 3340161 次迭代时参数和loss的值为： w1= [-21.73987579]  w2= [-70.56689453]  w3= [-15.20494747]  b3= [ 77.45510101]  loss= 0.0500003
第 3340162 次迭代时参数和loss的值为： w1= [-21.7398777]  w2= [-70.56690216]  w3= [-15.20494938]  b3= [ 77.45510864]  loss= 0.0500002
第 3340163 次迭代时参数和loss的值为： w1= [-21.73987961]  w2= [-70.56690979]  w3= [-15.20495129]  b3= [ 77.45511627]  loss= 0.0500003
第 3340164 次迭代时参数和loss的值为： w1= [-21.73988342]  w2= [-70.56691742]  w3= [-15.20495319]  b3= [ 77.4551239]  loss= 0.0500003
第 3340165 次迭代时参数和loss的值为： w1= [-21.73988342]  w2= [-70.56692505]  w3= [-15.2049551]  b3= [ 77.45513153]  loss= 0.0500002
第 3340166 次迭代时参数和loss的值为： w1= [-21.73988724]  w2= [-70.56693268]  w3= [-15.20495796]  b3= [ 77.45513916]  loss= 0.0500002
第 3340167 次迭代时参数和loss的值为： w1= [-21.73988914]  w2= [-70.56694031]  w3= [-15.20495987]  b3= [ 77.45514679]  loss= 0.0500002
第 3340168 次迭代时参数和loss的值为： w1= [-21.73989105]  w2= [-70.56694794]  w3= [-15.20496178]  b3= [ 77.45515442]  loss= 0.0500003
第 3340169 次迭代时参数和loss的值为： w1= [-21.73989487]  w2= [-70.56695557]  w3= [-15.20496464]  b3= [ 77.45516205]  loss= 0.0500004
第 3340170 次迭代时参数和loss的值为： w1= [-21.73989487]  w2= [-70.5669632]  w3= [-15.20496655]  b3= [ 77.45516968]  loss= 0.0500003
第 3340171 次迭代时参数和loss的值为： w1= [-21.73989868]  w2= [-70.56697083]  w3= [-15.20496941]  b3= [ 77.45517731]  loss= 0.0500004
第 3340172 次迭代时参数和loss的值为： w1= [-21.73990059]  w2= [-70.56697845]  w3= [-15.20497131]  b3= [ 77.45518494]  loss= 0.0500005
第 3340173 次迭代时参数和loss的值为： w1= [-21.7399025]  w2= [-70.56698608]  w3= [-15.20497322]  b3= [ 77.45519257]  loss= 0.0500005
第 3340174 次迭代时参数和loss的值为： w1= [-21.73990631]  w2= [-70.56699371]  w3= [-15.20497608]  b3= [ 77.4552002]  loss= 0.0500003
第 3340175 次迭代时参数和loss的值为： w1= [-21.73990631]  w2= [-70.56700134]  w3= [-15.20497799]  b3= [ 77.45520782]  loss= 0.0500003
第 3340176 次迭代时参数和loss的值为： w1= [-21.73991013]  w2= [-70.56700897]  w3= [-15.20498085]  b3= [ 77.45521545]  loss= 0.0500003
第 3340177 次迭代时参数和loss的值为： w1= [-21.73991203]  w2= [-70.5670166]  w3= [-15.20498276]  b3= [ 77.45522308]  loss= 0.0500003
第 3340178 次迭代时参数和loss的值为： w1= [-21.73991394]  w2= [-70.56702423]  w3= [-15.20498466]  b3= [ 77.45523071]  loss= 0.0500002
第 3340179 次迭代时参数和loss的值为： w1= [-21.73991776]  w2= [-70.56703186]  w3= [-15.20498753]  b3= [ 77.45523834]  loss= 0.0500002
第 3340180 次迭代时参数和loss的值为： w1= [-21.73991776]  w2= [-70.56703949]  w3= [-15.20498943]  b3= [ 77.45524597]  loss= 0.0500002
第 3340181 次迭代时参数和loss的值为： w1= [-21.73992157]  w2= [-70.56704712]  w3= [-15.20499229]  b3= [ 77.4552536]  loss= 0.0500002
第 3340182 次迭代时参数和loss的值为： w1= [-21.73992348]  w2= [-70.56705475]  w3= [-15.2049942]  b3= [ 77.45526123]  loss= 0.0500001
第 3340183 次迭代时参数和loss的值为： w1= [-21.73992538]  w2= [-70.56706238]  w3= [-15.20499611]  b3= [ 77.45526886]  loss= 0.05
第 3340184 次迭代时参数和loss的值为： w1= [-21.7399292]  w2= [-70.56707001]  w3= [-15.20499802]  b3= [ 77.45527649]  loss= 0.0500003
第 3340185 次迭代时参数和loss的值为： w1= [-21.7399292]  w2= [-70.56707764]  w3= [-15.20499992]  b3= [ 77.45528412]  loss= 0.0500002
第 3340186 次迭代时参数和loss的值为： w1= [-21.73993301]  w2= [-70.56708527]  w3= [-15.20500278]  b3= [ 77.45529175]  loss= 0.0500003
第 3340187 次迭代时参数和loss的值为： w1= [-21.73993492]  w2= [-70.5670929]  w3= [-15.20500469]  b3= [ 77.45529938]  loss= 0.0500001
第 3340188 次迭代时参数和loss的值为： w1= [-21.73993683]  w2= [-70.56710052]  w3= [-15.2050066]  b3= [ 77.45530701]  loss= 0.0500002
第 3340189 次迭代时参数和loss的值为： w1= [-21.73994064]  w2= [-70.56710815]  w3= [-15.20500946]  b3= [ 77.45531464]  loss= 0.0500002
第 3340190 次迭代时参数和loss的值为： w1= [-21.73994064]  w2= [-70.56711578]  w3= [-15.20501137]  b3= [ 77.45532227]  loss= 0.0500004
第 3340191 次迭代时参数和loss的值为： w1= [-21.73994446]  w2= [-70.56712341]  w3= [-15.20501423]  b3= [ 77.4553299]  loss= 0.05
```



1.1 学习率为 0.01 时的拟合图像

若学习率设置不当或迭代停止条件中的 loss 过大则会出现模拟得到的曲线近似直线的情况：（此时学习率为 0.001，终止条件为：loss 小于等于 0.5）



1.2 模拟曲线为直线的情况

1.2 ckpt 模式和 bp 模式的保存

(1) ckpt 模式

```
saver = tf.train.Saver()
#ckpt 模式保存
save_path = saver.save(sess,'save_ckpt/save.ckpt')
```

（2）bp 模式

首先，将要保存的变量命名

```
w1= tf.Variable(tf.random_uniform([1]),name='w1')
w2= tf.Variable(tf.random_uniform([1]),name='w2')
w3= tf.Variable(tf.random_uniform([1]),name='w3')
b= tf.Variable(tf.random_uniform([1]),name='b')
```

然后，根据变量名进行保存

```
constant_graph = graph_util.convert_variables_to_constants(sess,
sess.graph_def,["w1","w2","w3","b"])
with tf.gfile.FastGFile('save_pb.pb', mode='wb') as f:
    f.write(constant_graph.SerializeToString())
```

保存文件末尾加入如下语句

```
#清除默认图的堆栈，并设置全局图为默认图
#解决多次运行之后的错误：NotFoundError (see above for traceback): Key
Variable_1 not found in checkpoint
tf.reset_default_graph()
```

1.3 ckpt 和 bp 模式的恢复

（1）ckpt 模式恢复

ckpt 多次恢复时出现的一个错误如下：

```
2019-05-15 10:25:00.038946: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.1 i
2019-05-15 10:25:00.038958: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.2 i
2019-05-15 10:25:00.038961: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX inst
2019-05-15 10:25:00.038964: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX2 ins
2019-05-15 10:25:00.038967: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use FMA inst
2019-05-15 10:25:00.055361: W tensorflow/core/framework/op_kernel.cc:1192] Not found: Key Variable not found in checkpoint
2019-05-15 10:25:00.055369: W tensorflow/core/framework/op_kernel.cc:1192] Not found: Key Variable_1 not found in checkpoint
2019-05-15 10:25:00.055362: W tensorflow/core/framework/op_kernel.cc:1192] Not found: Key Variable_3 not found in checkpoint
2019-05-15 10:25:00.055378: W tensorflow/core/framework/op_kernel.cc:1192] Not found: Key Variable_2 not found in checkpoint
Traceback (most recent call last):
  File "/home/yyl/anaconda3/envs/tensorflow/lib/python3.6/site-packages/tensorflow/python/client/session.py", line 1327, in _do_call
    return fn(*args)
  File "/home/yyl/anaconda3/envs/tensorflow/lib/python3.6/site-packages/tensorflow/python/client/session.py", line 1306, in _run_fn
    status, run_metadata)
  File "/home/yyl/anaconda3/envs/tensorflow/lib/python3.6/contextlib.py", line 88, in __exit__
    next(self.gen)
  File "/home/yyl/anaconda3/envs/tensorflow/lib/python3.6/site-packages/tensorflow/python/framework/errors_impl.py", line 466, in rais
    pywrap_tensorflow.TF_GetCode(status))
tensorflow.python.framework.errors_impl.NotFoundError: Key Variable_not found in checkpoint
     [[Node: save/RestoreV2 = RestoreV2[dtypes=[DT_FLOAT], _device="/job:localhost/replica:0/task:0/cpu:0"](_arg_save/Const_0_0, save/
```

1.3 ckpt 恢复多次时出现的错误

网上查询资料原因：出现这样的原因是因为 ckpt 模式下第一次加载完成后，已经有名字为“w1”这样的变量，再次加载时因为名字重复，名字变为“w1_1”，故保存的变量名已经改变了，因此出现 NotFoundError: Key Variable not found in checkpoint 类似的错。

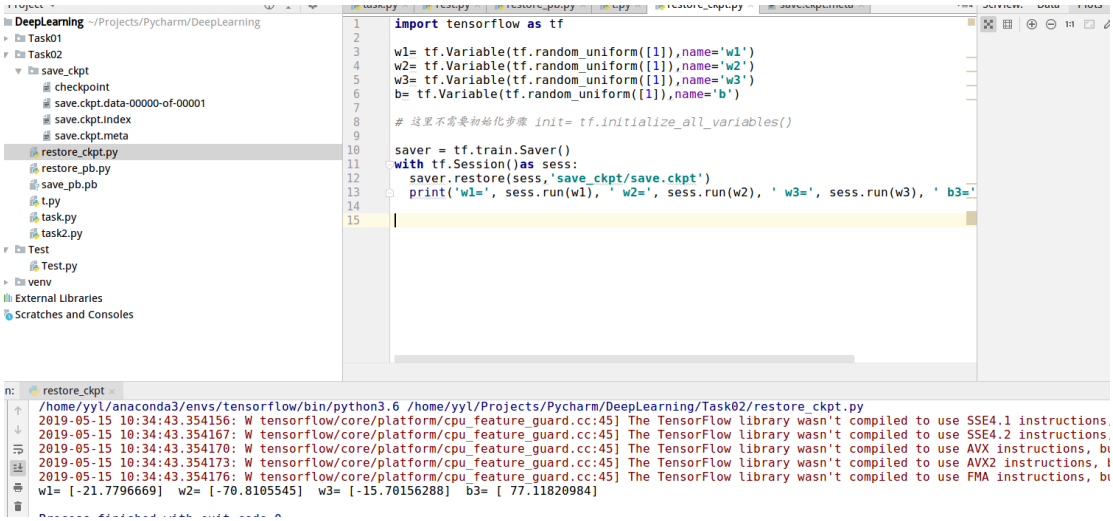
解决方法：在恢复时，给变量命名并在保存模型时加上如下语句：

```
#清除默认图的堆栈，并设置全局图为默认图
tf.reset_default_graph()
```

恢复模型参数代码：

```
import tensorflow as tf

w1= tf.Variable(tf.random_uniform([1]),name='w1')
w2= tf.Variable(tf.random_uniform([1]),name='w2')
w3= tf.Variable(tf.random_uniform([1]),name='w3')
b= tf.Variable(tf.random_uniform([1]),name='b')

# 这里不需要初始化步骤 init= tf.initialize_all_variables()

saver = tf.train.Saver()
with tf.Session()as sess:
  saver.restore(sess,'save_ckpt/save.ckpt')
  print('w1=', sess.run(w1), ' w2=', sess.run(w2), ' w3=', sess.run(w3), '
b3=', sess.run(b))
```

此后，多次恢复模型也不会报错，且输出结果如下图所示：

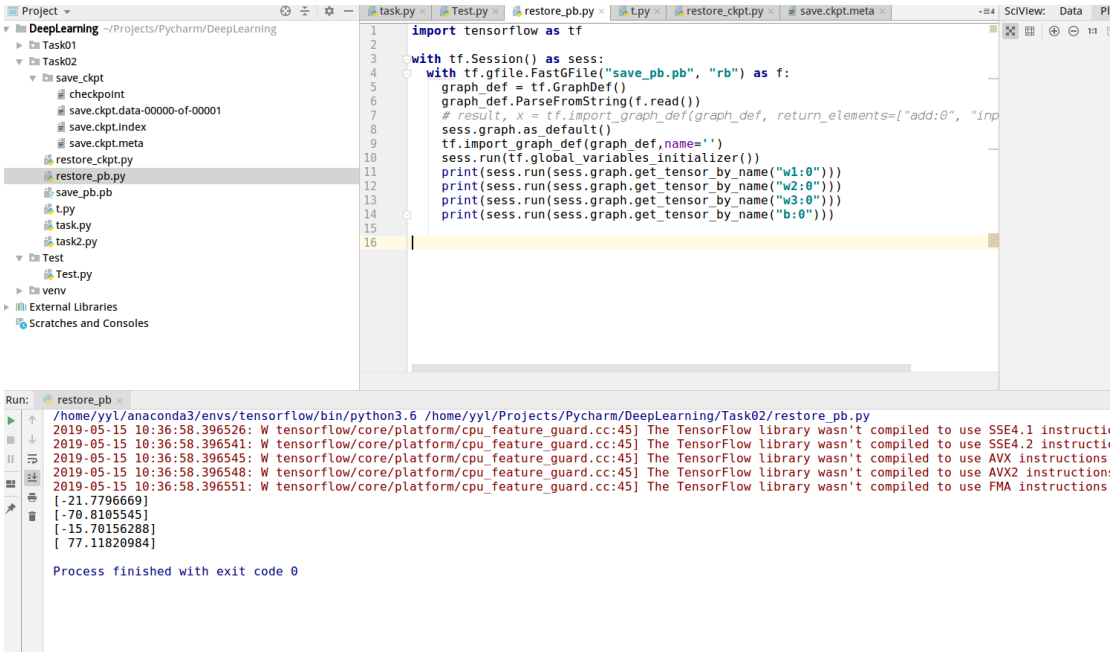


1.4 ckpt 恢复模型参数的结果

（2）bp 模式恢复

```
import tensorflow as tf

with tf.Session() as sess:
  with tf.gfile.FastGFile("save_pb.pb", "rb") as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
    sess.graph.as_default()
    tf.import_graph_def(graph_def,name='')
    sess.run(tf.global_variables_initializer())
    print(sess.run(sess.graph.get_tensor_by_name("w1:0")))
    print(sess.run(sess.graph.get_tensor_by_name("w2:0")))
    print(sess.run(sess.graph.get_tensor_by_name("w3:0")))
    print(sess.run(sess.graph.get_tensor_by_name("b:0")))
```

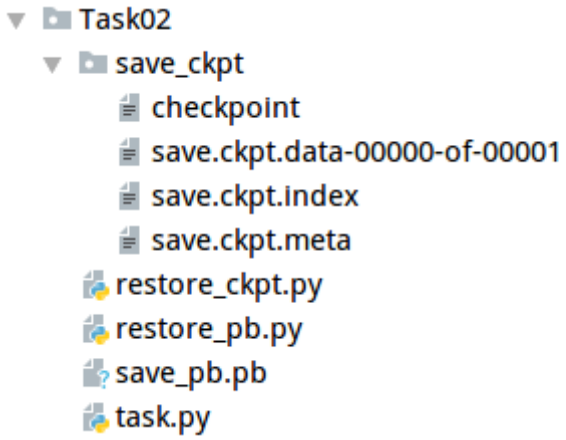
输出结果如下：



1.5 bp 模式恢复模型参数的结果

可见，两种模型恢复时，参数结果一样，模型保存和恢复成功。

文件目录如下：



1.6 分别使用 ckpt 模式和 PB 模式保存拟合的模型后的文件结构

2. **进阶作业**

使用 TensorFlow 定义函数 y = 1 - sin(x)/x, 并求解 3y 达到最小值时对应的 x

步骤提示：

(1) 使用 TensorFlow 给出函数定义

(2) 利用 tf.train.GradientDescentOptimizer 定义优化器

(3) 启动会话，用梯度下降算法训练一定的迭代次数，在每次迭代之后输出当前的 x 和 y 值

实验代码：

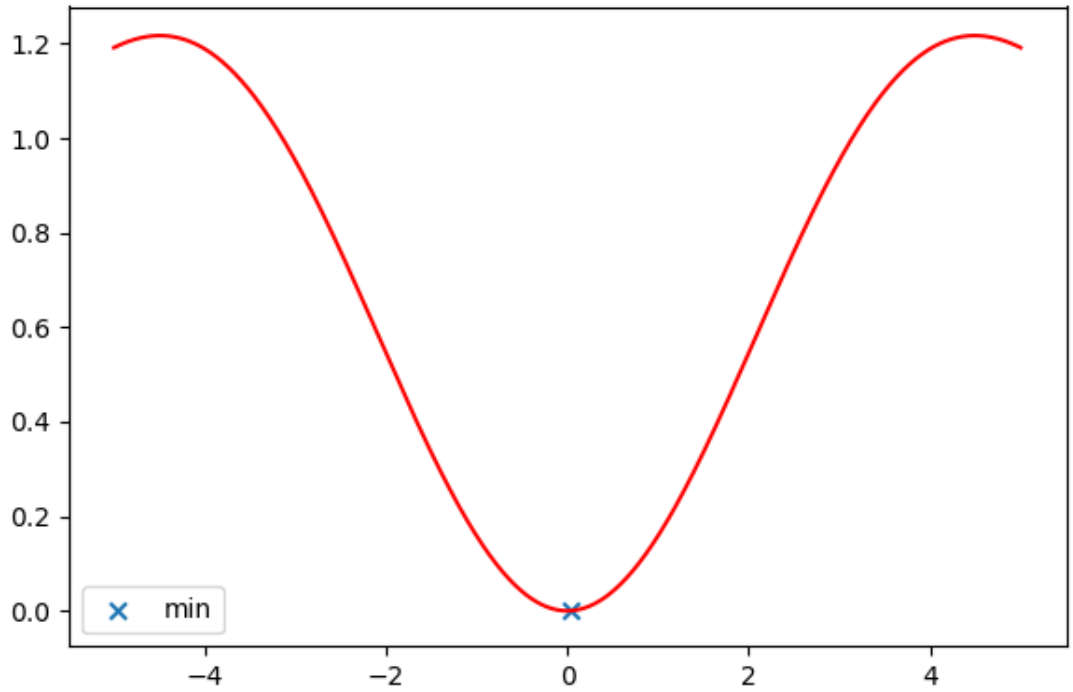
```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np

x = tf.Variable(1,dtype = tf.float32)  # 定义一个可以优化的 x 值
#定义 y 值：y = 1 - sin(x)/x
y = tf.subtract(1.,tf.divide(tf.sin(x),x))
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(y)

init = tf.global_variables_initializer()
# 生成会话，训练 1000 轮
i=0
with tf.Session() as sess:
    sess.run(init)
    for i in range(1000):
        sess.run(train_step)
        x_val = sess.run(x)
        y_val = sess.run(y)
        print("epoch",i,":",sess.run(x)," ",sess.run(y))
    i=i+1
```

```
plt.figure(figsize=(6, 4))  # 设置图片大小
x_data=np.linspace(-5,5,10000)  #这个表示在-5 到 5 之间生成 10000 个 x 值
plt.plot(x_data,1-(np.sin(x_data)/x_data),color='red',)
plt.scatter(x_val, y_val, marker = 'x', s = 40 ,label = 'min') #画出最优值
plt.legend() # 显示图例
plt.show()
```

输出：

```
epoch 962 : 0.0422763   0.000297844
epoch 963 : 0.0421354   0.000295937
epoch 964 : 0.041995   0.000293911
epoch 965 : 0.0418551   0.000291944
epoch 966 : 0.0417156   0.000289977
epoch 967 : 0.0415765   0.000288069
epoch 968 : 0.041438   0.000286162
epoch 969 : 0.0412999   0.000284195
epoch 970 : 0.0411622   0.000282347
epoch 971 : 0.041025   0.000280499
epoch 972 : 0.0408883   0.000278592
epoch 973 : 0.040752   0.000276804
epoch 974 : 0.0406162   0.000274897
epoch 975 : 0.0404808   0.000273108
epoch 976 : 0.0403459   0.000271261
epoch 977 : 0.0402114   0.000269473
epoch 978 : 0.0400774   0.000267684
epoch 979 : 0.0399438   0.000265896
epoch 980 : 0.0398107   0.000264168
epoch 981 : 0.039678   0.000262439
epoch 982 : 0.0395458   0.000260651
epoch 983 : 0.039414   0.000258863
epoch 984 : 0.0392826   0.000257194
epoch 985 : 0.0391517   0.000255466
epoch 986 : 0.0390213   0.000253737
epoch 987 : 0.0388912   0.000252128
epoch 988 : 0.0387616   0.00025034
epoch 989 : 0.0386324   0.000248671
epoch 990 : 0.0385036   0.000247121
epoch 991 : 0.0383753   0.000245392
epoch 992 : 0.0382474   0.000243783
epoch 993 : 0.0381199   0.000242174
epoch 994 : 0.0379929   0.000240505
epoch 995 : 0.0378663   0.000238955
epoch 996 : 0.03774   0.000237405
epoch 997 : 0.0376142   0.000235796
epoch 998 : 0.0374889   0.000234187
epoch 999 : 0.0373639   0.000232697
```

2.1 实验输出结果



2.2 实验图像

上图红色部分绘制了函数 y = 1 - sin(x)/x 的图像，其中蓝色点 x 即为使用梯度下降求得的最优值（0.0373639,0.000232697）。