Tf

Tf都是通过session来执行代码

建立Session方法

import tensorflow as tf

############################################session###########################

hello=tf.constant('hello word!')

sess=tf.Session()

print(sess.run(hello))

sess.close()

############################################with使用####################

a=tf.constant(3)

b=tf.constant(4)

with tf.Session() as sess:

print('add:{}+{}={}'.format(sess.run(a),sess.run(b),sess.run(a+b)))

print('multiply:{}\*{}={}'.format(sess.run(a),sess.run(b),sess.run(a\*b)))

############################placeholder##################################

a=tf.placeholder(tf.int16)

b=tf.placeholder(tf.int16)

add=tf.add(a,b)

mul=tf.multiply(a,b)

with tf.Session() as sess:

print('add:{}'.format(sess.run(add,feed\_dict={a:3,b:4})))

print('multiply:{}'.format(sess.run(mul,feed\_dict={a:2,b:7})))

还有sess=tf.InteractiveSession()，交互式Session

#########################模型保存及载入########################

import tensorflow as tf

import numpy as np

import matplotlib.pyplot as plt

train\_X=np.linspace(-1,1,100)

train\_Y=2\*train\_X+np.random.randn(\*train\_X.shape)\*0.3

savepath='g:\learn\data\modelsave\line\line.cpkt'

def moving\_average(a,w=10):

if len(a)<w:

return a[:]

return [val if idx<w else sum(a[idx-w:idx])/w for idx,val in enumerate(a)]

#占位符

X=tf.placeholder("float")

Y=tf.placeholder('float')

#模型参数

W=tf.Variable(tf.random\_normal([1]),name='weight')

b=tf.Variable(tf.zeros([1]),name='bias')

#前向结构

z=tf.multiply(X,W)+b

#反向优化

cost=tf.reduce\_mean(tf.square(Y-z))

learning\_rate=0.01

optimizer=tf.train.GradientDescentOptimizer(learning\_rate).minimize(cost)

#初始化所有变量

init=tf.global\_variables\_initializer()

#定义参数

train\_epoch=30

display\_step=3

saver=tf.train.Saver()

#启动session

with tf.Session() as sess:

sess.run(init)

plotdata={'batchsize':[],'loss':[]}

#向模型输入数据

for epoch in range(train\_epoch):

for x,y in zip(train\_X,train\_Y):

sess.run(optimizer,feed\_dict={X:x,Y:y})

if epoch%display\_step==0:

loss=sess.run(cost,feed\_dict={X:train\_X,Y:train\_Y})

print("Epoch:",epoch+1,"cost=",loss,"W=",sess.run(W),"b=",sess.run(b))

if not (loss=='NA'):

plotdata["batchsize"].append(epoch)

plotdata["loss"].append(loss)

print("finished!")

print("cost=",sess.run(cost,feed\_dict={X:train\_X,Y:train\_Y}),"W=",sess.run(W),"b=",sess.run(b))

saver.save(sess,savepath)

# #图形显示

# plt.plot(train\_X,train\_Y,'ro',label='original data')

# plt.plot(train\_X,sess.run(W)\*train\_X+sess.run(b),label='Fittedline')

# plt.legend()

# plt.show()

#

# plotdata['avgloss']=moving\_average(plotdata['loss'])

# plt.figure(1)

# plt.subplot(211)

# plt.plot(plotdata['batchsize'],plotdata['avgloss'],'b--')

# plt.xlabel('Minibatch number')

# plt.ylabel('Loss')

# plt.title('Minibatch run vs. Training loss')

# plt.show()

with tf.Session() as sess1:

sess1.run(tf.global\_variables\_initializer())

saver.restore(sess1,savepath)

print('x=0.2,z=',sess1.run(z,feed\_dict={X:0.2}))

#############打印保存模型的方法########################3

from tensorflow.python.tools.inspect\_checkpoint import print\_tensors\_in\_checkpoint\_file

print\_tensors\_in\_checkpoint\_file(savepath,None,True)

#########模型的保存，载入，chekpoint###########################

import tensorflow as tf

import numpy as np

import matplotlib.pyplot as plt

train\_X=np.linspace(-1,1,100)

train\_Y=2\*train\_X+np.random.randn(\*train\_X.shape)\*0.3

savepath='g:\learn\data\modelsave\line\line.cpkt'

def moving\_average(a,w=10):

if len(a)<w:

return a[:]

return [val if idx<w else sum(a[idx-w:idx])/w for idx,val in enumerate(a)]

#占位符

X=tf.placeholder("float")

Y=tf.placeholder('float')

#模型参数

W=tf.Variable(tf.random\_normal([1]),name='weight')

b=tf.Variable(tf.zeros([1]),name='bias')

#前向结构

z=tf.multiply(X,W)+b

#反向优化

cost=tf.reduce\_mean(tf.square(Y-z))

learning\_rate=0.01

optimizer=tf.train.GradientDescentOptimizer(learning\_rate).minimize(cost)

#初始化所有变量

init=tf.global\_variables\_initializer()

#定义参数

train\_epoch=30

display\_step=3

saver=tf.train.Saver(max\_to\_keep=2)

# #启动session

# with tf.Session() as sess:

# sess.run(init)

# plotdata={'batchsize':[],'loss':[]}

# #向模型输入数据

# for epoch in range(train\_epoch):

# for x,y in zip(train\_X,train\_Y):

# sess.run(optimizer,feed\_dict={X:x,Y:y})

# if epoch%display\_step==0:

# loss=sess.run(cost,feed\_dict={X:train\_X,Y:train\_Y})

# print("Epoch:",epoch+1,"cost=",loss,"W=",sess.run(W),"b=",sess.run(b))

# if not (loss=='NA'):

# plotdata["batchsize"].append(epoch)

# plotdata["loss"].append(loss)

# print("finished!")

# print("cost=",sess.run(cost,feed\_dict={X:train\_X,Y:train\_Y}),"W=",sess.run(W),"b=",sess.run(b))

# saver.save(sess,savepath,global\_step=epoch)

#

# # #图形显示

# # plt.plot(train\_X,train\_Y,'ro',label='original data')

# # plt.plot(train\_X,sess.run(W)\*train\_X+sess.run(b),label='Fittedline')

# # plt.legend()

# # plt.show()

# #

# # plotdata['avgloss']=moving\_average(plotdata['loss'])

# # plt.figure(1)

# # plt.subplot(211)

# # plt.plot(plotdata['batchsize'],plotdata['avgloss'],'b--')

# # plt.xlabel('Minibatch number')

# # plt.ylabel('Loss')

# # plt.title('Minibatch run vs. Training loss')

# # plt.show()

#

# with tf.Session() as sess1:

# sess1.run(tf.global\_variables\_initializer())

# saver.restore(sess1,savepath+'-29')

# print('x=0.2,z=',sess1.run(z,feed\_dict={X:0.2}))

#

# #打印模型保存的参数

# from tensorflow.python.tools.inspect\_checkpoint import print\_tensors\_in\_checkpoint\_file

# print\_tensors\_in\_checkpoint\_file(savepath+'-29',None,True)

#

#检测相应的状态

ckpt\_dir='g:\learn\data\modelsave\line'

with tf.Session() as sess2:

sess2.run(tf.global\_variables\_initializer())

ckpt=tf.train.get\_checkpoint\_state(ckpt\_dir)

if ckpt and ckpt.model\_checkpoint\_path:

saver.restore(sess2,ckpt.model\_checkpoint\_path)

print('x=0.2,z=', sess2.run(z, feed\_dict={X: 0.2}))

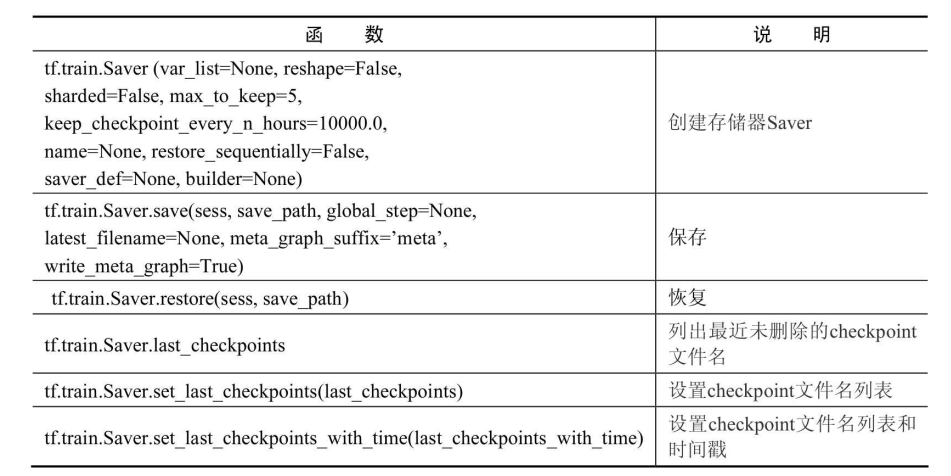
# 或者

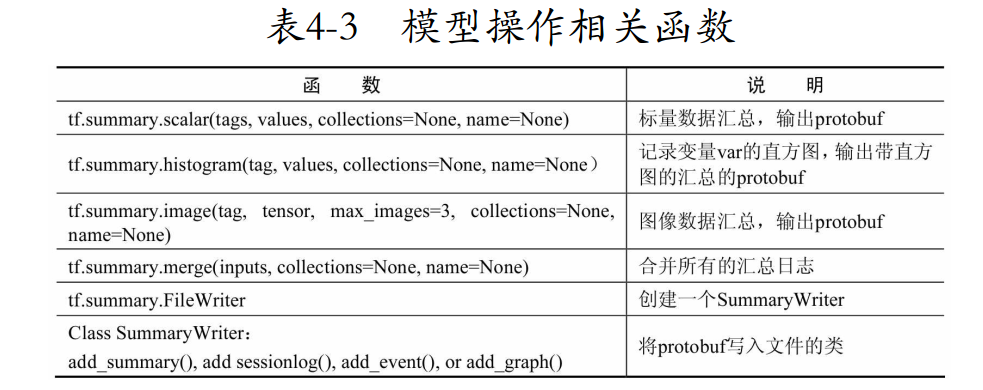
kpt = tf.train.latest\_checkpoint(savepath)

if kpt != None:

saver.restore(sess2)

print('x=0.3,z=', sess2.run(z, feed\_dict={X: 0.3}))





tf.convert\_to\_tensor()将其他类型的数据转换为tf类型

读取csv

reader = tf.TextLineReader()   
key, value = reader.read(filename\_queue)

decode

record\_defaults = [[1], [1], [1]]  
col1, col2, col3 = tf.decode\_csv(value, record\_defaults=record\_defaults)

tf.train.GradientDescentOptimizer(learning\_rate) 剃度下降优化

optimizer = tf.train.MomentumOtimizer(learning\_rate=0.01, momentum=0.5).minimize(loss)

optimizer = tf.train.AdadeltaOptimizer(learning\_rate=0.8, rho=0.95).minimize(loss)

optimizer = tf.train.RMSpropOptimizer(learning\_rate=0.01, decay=0.8, momentum=0.1).minimize(loss)

optimizer = tf.train.AdamOptimizer().minimize(loss)

tf.train.exponential\_decay

global\_step = tf.Variable(0, trainable = false)  
initial\_learning\_rate = 0.2  
learning\_rate = tf.train.exponential\_decay(initial\_learning\_rate, global\_step, decay\_steps=100000, decay\_rate=0.95, staircase=True)