# Tensorflow模型的保存和载入

保存模型使用saver = tf.train.Saver()定义一个Saver，在训练时使用以下语句保存模型：

saver.save(sess, savedir + **"linermodel.cpkt"**, global\_step=epoch)

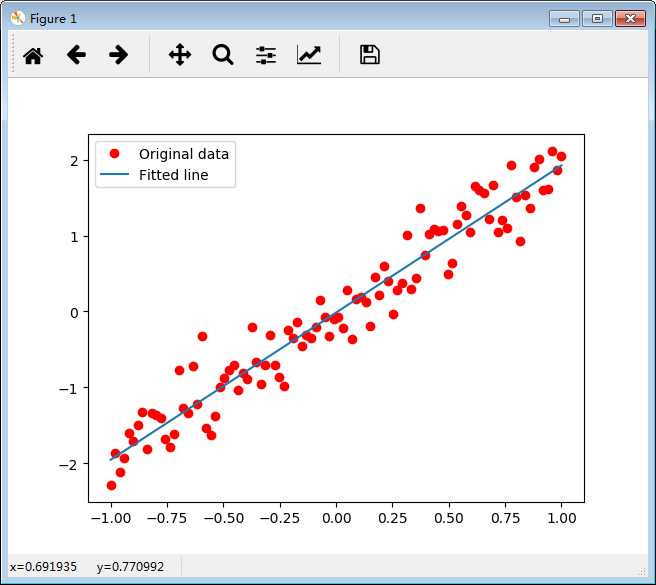
在载入模型时民，使用以下语句：

saver.restore(sess, kpt)

以下演示程序，请参考：

1.训练并保存模型

|  |
| --- |
| **import** tensorflow **as** tf **import** numpy **as** np **import** matplotlib.pyplot **as** plt  trainX = np.linspace(-1,1,100) trainY = 2 \* trainX + np.random.randn(\*trainX.shape) \* 0.3  plt.plot(trainX, trainY, **'ro'**, label = **'Original data'**) plt.legend() plt.show()  tf.reset\_default\_graph() input\_X = tf.placeholder(**"float"**) input\_Y = tf.placeholder(dtype=tf.float32)  W = tf.Variable(tf.random\_normal([1]), name=**"weight"**) b = tf.Variable(tf.zeros([1]), name=**"bias"**) z = tf.multiply(input\_X, W) + b  cost = tf.reduce\_mean(tf.square(input\_Y - z)) learning\_rate = 0.01 optimizer = tf.train.GradientDescentOptimizer(learning\_rate).minimize(cost)  init = tf.global\_variables\_initializer() training\_epochs = 10 display\_step = 2 saver = tf.train.Saver() savedir = **"log/" with** tf.Session() **as** sess:  sess.run(init)  **for** epoch **in** range(training\_epochs):  **for** (x, y) **in** zip(trainX, trainY):  sess.run(optimizer, feed\_dict={input\_X:x, input\_Y:y})  **if** epoch % display\_step == 0:  loss = sess.run(cost, feed\_dict={input\_X:x, input\_Y:y})  print(**"Epoch:"**, epoch + 1, **" cost="**, loss, **" W="**, sess.run(W), **" b="**, sess.run(b))  saver.save(sess, savedir + **"linermodel.cpkt"**, global\_step=epoch)  print(**"Finish Trainning!"**) *#显示结果* plt.plot(trainX, trainY, **'ro'**, label = **'Original data'**)  plt.plot(trainX, sess.run(W) \* trainX + sess.run(b), label = **'Fitted line'**)  plt.legend()  plt.show() |



载入模型：

|  |
| --- |
| **import** tensorflow **as** tf  input\_X = tf.placeholder(**"float"**) W = tf.Variable(tf.random\_normal([1]), name=**"weight"**) b = tf.Variable(tf.zeros([1]), name=**"bias"**) z = tf.multiply(input\_X, W) + b  load\_epoch = 9 savedir = **"log/"** saver = tf.train.Saver()**with** tf.Session() **as** sess:  sess.run(tf.global\_variables\_initializer())  kpt = tf.train.latest\_checkpoint(savedir)  **if** kpt!=**None**:  saver.restore(sess, kpt)  print(**"x = 0.2, z = "**, sess.run(z, feed\_dict={input\_X: 0.2})) |