版本2：特征数据归一化

特征数据归一化

```
# 对特征数据 【0到11】列 做（0-1）归一化

for i in range(12):
    df[:,i]=(df[:,i]-df[:,i].min())/(df[:,i].max()-df[:,i].min())
```

```
# x_data 为 归一化后的前12列特征数据

x_data = df[:,:12]

# y_data 为最后1列标签数据

y_data = df[:,12]
```

其他代码保持不变

数据归一化后模型运行结果

```
epoch= 1 loss= 44.3609942016 b= 3.60617 w= [[-0.60727906]
 [ 1.37151349]
 [-0.79043895]
 [ 0.51065689]
 [ 2.50340939]
 [ 7.16300583]
 [-0.04436842]
 [ 0.79660386]
 [ 0.38123909]
 [ 0.33448994]
 [ 2.32227421]
 [-4.38565731]]
epoch= 2 loss= 32.054614775 b= 3.99515 w= [[ -1.157341  ]
 [  1.95650125]
 [ -1.52039194]
 [  0.859348  ]
 [  2.8789475 ]
 [ 10.60353088]
 [ -0.81390387]
 [  0.35741901]
 [  0.62936795]
 [ -0.25641721]
 [  1.15861583]
 [ -8.10381222]]
```

Oh，Yeah！

```
epoch= 49 loss= 19.662282074 b= 11.9975 w= [[ -9.93251896]
 [  1.82018924]
 [ -1.10002387]
 [  0.07413481]
 [ -1.97596598]
 [ 22.86484528]
 [ -0.7712326 ]
 [ -7.2605238 ]
 [  6.09851217]
 [ -5.73373365]
 [ -3.09019566]
 [-20.35456848]]
epoch= 50 loss= 19.6482381622 b= 12.149 w= [[ -9.98610115]
 [  1.82591212]
 [ -1.08572245]
 [  0.07217827]
 [ -2.05136132]
 [ 22.85953331]
 [ -0.7745437 ]
 [ -7.31256819]
 [  6.12230444]
 [ -5.76376295]
 [ -3.10729456]
 [-20.33439827]]
```

模型应用

模型应用

模型一般应该用来预测新的样本的值

本例506条数据都用来训练了，暂时没有新的数据

```
n=348   # 指定一条来看看效果
x_test = x_data[n]

x_test = x_test.reshape(1,12)
predict = sess.run(pred, feed_dict={x: x_test})
print("预测值：%f" % predict)

target = y_data[n]
print("标签值：%f" % target)
```

```
预测值：23.972572
标签值：24.500000
```

```
n = np.random.randint(506) # 随机确定一条来看看效果
print(n)
x_test = x_data[n]

x_test = x_test.reshape(1,12)
predict = sess.run(pred, feed_dict={x: x_test})
print("预测值：%f" % predict)

target = y_data[n]
print("标签值：%f" % target)
```

```
361
预测值：17.297903
标签值：19.900000
```

模型应用

思考：该不该全部数据都参与训练？

版本3：可视化训练过程中的损失值

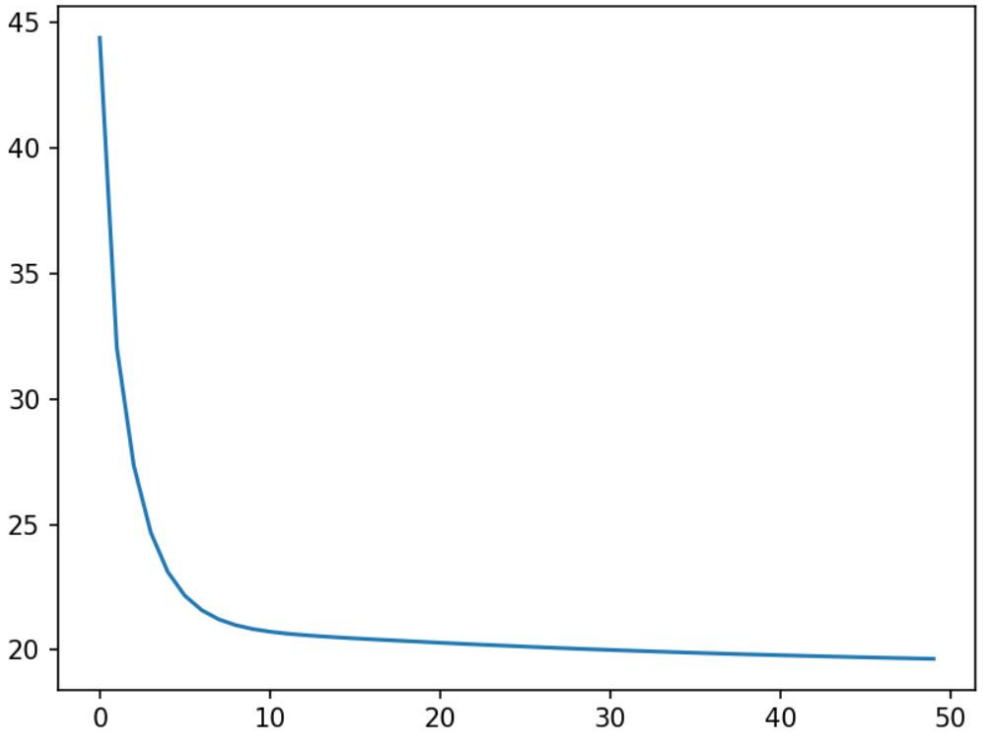
修改训练过程代码

```
loss_list = [] # 用于保存loss值的列表

for epoch in range (train_epochs):
    loss_sum = 0.0
    for xs, ys in zip(x_data, y_data):

        xs = xs.reshape(1,12)
        ys = ys.reshape(1,1)

        _, loss = sess.run([optimizer,loss_function], feed_dict={x: xs, y: ys})

        loss_sum = loss_sum + loss

    # 打乱数据顺序
    x_data, y_data = shuffle(x_data, y_data)

    b0temp=b.eval(session=sess)
    w0temp=w.eval(session=sess)
    loss_average = loss_sum/len(y_data)

    loss_list.append(loss_average) # 每轮添加一次

    print("epoch=", epoch+1,"loss=", loss_average,"b=", b0temp,"w=", w0temp )
```

只是每轮训练后添加一个这一轮的Loss平均值

可视化损失值

```
plt.plot(loss_list)
```



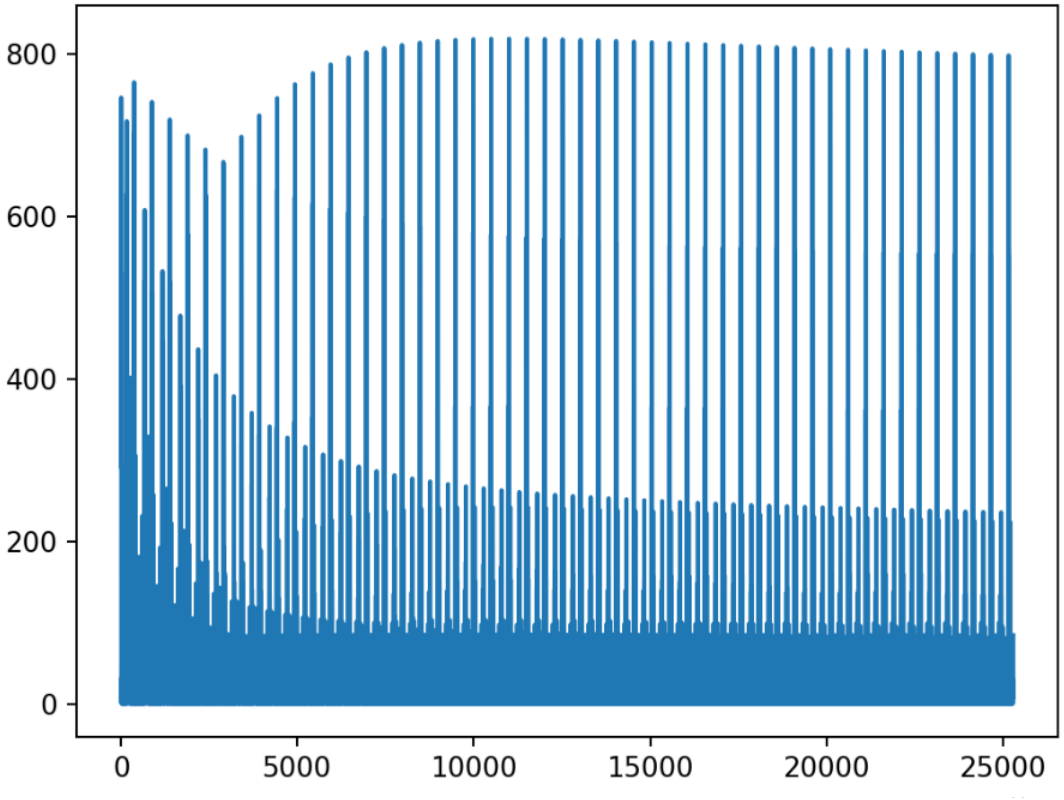
修改训练过程代码

```
loss_list = [] # 用于保存loss值的列表

for epoch in range (train_epochs):
    loss_sum = 0.0
    for xs, ys in zip(x_data, y_data):

        xs = xs.reshape(1,12)
        ys = ys.reshape(1,1)

        _, loss = sess.run([optimizer,loss_function], feed_dict={x: xs, y: ys})

        loss_sum = loss_sum + loss

        loss_list.append(loss) # 每步添加一次

    # 打乱数据顺序
    x_data, y_data = shuffle(x_data, y_data)

    b0temp=b.eval(session=sess)
    w0temp=w.eval(session=sess)
    loss_average = loss_sum/len(y_data)

    print("epoch=", epoch+1,"loss=", loss_average,"b=", b0temp,"w=", w0temp )
```

每步（单个样本）训练后添加这个Loss值

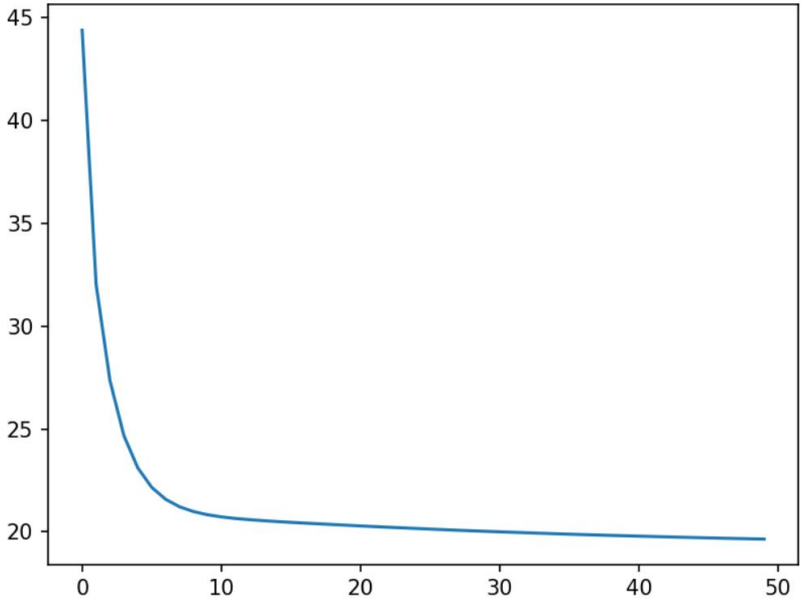
可视化损失值

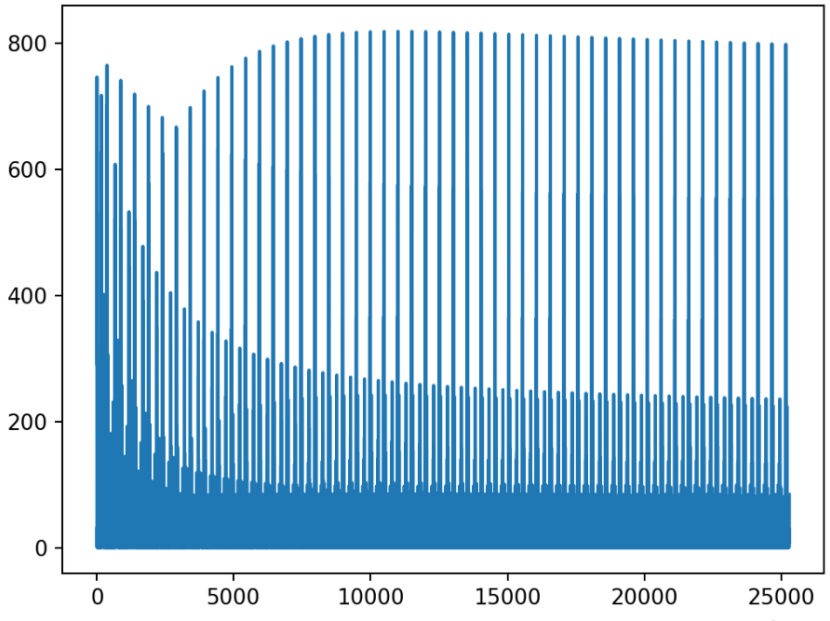
```
plt.plot(loss_list)
```



思考

思考：哪一种显示损失的方案更好？





版本4：加上TensorBoard可视化代码

修改代码

声明会话

```
sess = tf.Session()

# 定义初始化变量的操作
init = tf.global_variables_initializer()
```

为TensorBoard可视化准备数据

```
# 设置日志存储目录
logdir='d:/log'
```

```
# 创建一个操作，用于记录损失值loss，后面在TensorBoard中SCALARS栏可见
sum_loss_op = tf.summary.scalar("loss", loss_function)

# 把所有需要记录摘要日志文件的合并，方便一次性写入
merged = tf.summary.merge_all()
```

tf.summary.scalar("loss", loss_function)

tf.summary.merge_all()

修改代码

启动会话

```
sess.run(init)
```

创建摘要的文件写入器（FileWriter）

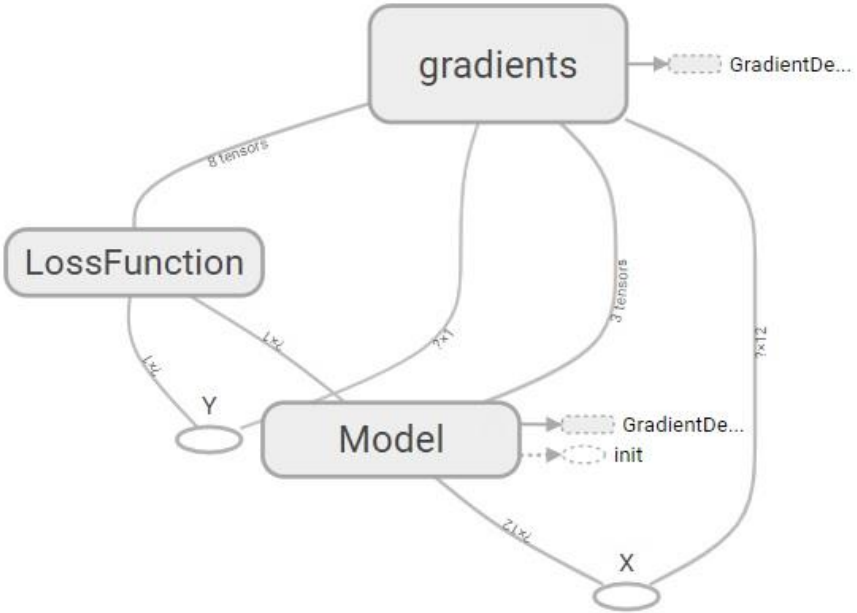
```
# 创建摘要writer ，将计算图写入摘要文件，后面在TensorBoard中GRAPHS栏可见
writer = tf.summary.FileWriter(logdir, sess.graph)
```

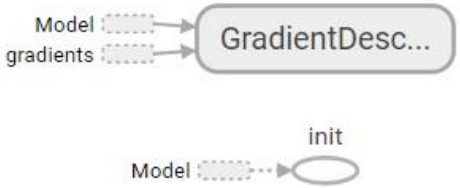
修改代码


```
for epoch in range (train_epochs):
    loss_sum = 0.0
    for xs, ys in zip(x_data, y_data):

        xs = xs.reshape(1,12)
        ys = ys.reshape(1,1)

        _, summary_str, loss = sess.run([optimizer,sum_loss_op,loss_function], feed_dict={x: xs, y: ys})

        writer.add_summary(summary_str, epoch)
        loss_sum = loss_sum + loss

    # 打乱数据顺序
    x_data, y_data = shuffle(x_data, y_data)

    b0temp=b.eval(session=sess)
    w0temp=w.eval(session=sess)
    loss_average = loss_sum/len(y_data)

    print("epoch=", epoch+1,"loss=", loss_average,"b=", b0temp,"w=", w0temp )
```

writer.add_summary(summary_str, epoch)

TensorBoard查看计算图





TensorBoard查看loss