# 第三章 线性模型

线性模型是最简单的统计学模型之一，它在计算机被发明之前就被统计学家提出，不过它在现代的计算数学和统计学中同样有着重要的作用。线性模型能够清晰和直观地呈现输入数据与输出数据之间的关联，以及输入是如何影响输出的。有时候，在特定的回归任务和预测任务中，线性模型能比某些非线性模型更优，尤其是在小样本预测和稀疏样本或者低信噪比的情形中。

同样，线性模型在接收输入数据的时候，可以将输入数据进行变换（transformation），显著提升分类和回归能力，这样的方法称为基函数方法（basis function method）。线性模型假设了最终习得的 函数是关于输入 每个坐标线性组合的形式，表示成矩阵即

其中 且 为偏置项（bias）， 为可以接收的误差。

两种表示模型的形式，什么时候需要加 帽号 和 epsilon

在本章线性模型的讲解中，我们主要介绍线性回归和线性分类模型。同时，我们需要对简单的模型进行拓展，因为很多非线性模型方法在本质上是对线性模型方法的扩展，这能够很好地帮助我们建立一个分析基础从而更快速和轻松地理解之后的模型。

## 线性回归

线性回归假设了：对于一个输入矩阵 ，其中 表示所有样本第 分量组成的向量，回归函数 是线性的。线性回归模型的两个基本的用途是：一利用已经建立的回归模型对一组自变量（independent value）样本（即 X）进行预测，可以估计因变量（即 Y）的值；二利用模型进行自变量和因变量之间相关性的分析，例如检测是否存在线性关系或者检测是否存在X数据冗余。

### 线性回归模型推导

我们主要的任务是利用回归模型来完成数据预测任务，下面我们来推导线性回归模型。

根据线性回归模型的假设：回归函数 是线性的（为什么是条件概率的形式？我们在统计解释中说明），也就是说我们需要习得的 有一般形式

通常我们假设有一组数据集 和 ，同时 与 对应，我们的目标是计算出 和偏置 项。那么由此根据 (3.2) 可以对任意的输入 得对 的值进行预测。

那么如何计算 呢？我们利用已有的数据集 进行学习的过程就是计算 的过程。根据 (3.2) 对于任意 需要满足

并且误差 必须非常小才能让我们接受它。 也就是对于输入集合 中每一组数据，我们所计算出的 必须使我们得到的模型对所有数据都有非常小而可容忍的误差。

最普遍常用的最小化误差的方式是均方误差（least square），这里给出均方损失函数

(3.4) 中均方损失 的大小就决定了误差的大小，因此我们要最小化 。为了数学上的便利，我们令

上 (3.2) 改写成

可以看出来我们只是将偏置项写到了矩阵和向量中，那么经过变换后

目标函数

可以看出 是连续单值函数，因此直接另一阶导数为零

根据 (3.7) 可知 存在二阶导数，再由 (3.8) 可得根据 (3.7) 得到的 使 取到了最小值。其中 (3.7) 涉及到了单值函数对向量求导，具体求导法则参见【附录】。

下面假设矩阵 是列满秩的，因此 可逆，根据 (3.7) 得到

当然很多情况下，矩阵 不是满秩的，因为存在某些冗余数据或者某些行向量存在线性相关性（如 ），那么这样的话，我们计算出来的 就不是唯一的了。这里给出三种解决方案

1. 进行数据预处理，特征过滤
2. 进行正则化处理，比如说岭回归（ridge regression）之后会提及
3. 直接去掉冗余的行向量

当然，很多的库函数已经帮我们解决了这个问题，他们会自动帮我们处理。最终我们习得的

同时，我们可以将我们的线性回归模型进一步推广，模型的任意一个输入样本 可以是来源于

1. 样本 经过函数变换后的值，比如说经过 函数变换后 的值作为新的
2. 也可以是别的样本的组合形式，比如 ，或者 ，那么很明显我们也可以用线性回归模型来做多项式拟合

### 最小二乘法的统计解释

我们的模型训练数据 实质上是从分布 上取下的样本。需要注意的是这里的 与一般的统计模型假设不同： 本身也是随机变量，是从一个分布 上获取的，因此任意一个 实质上是有条件依赖于 的，不过即使 条件独立，最小二乘法仍然成立，因为最小二乘法只是代表了残差，对 本身并无关系。

但是为了从统计角度进行分析，我们需要给出假定：观测变量 与给定 不相关，并且有定方差 ，同时保证 不是随机而是固定的。

根据 (3.1) 对模型的 次观察取样得到

$$y^{(j)}=\beta\_0+X^{(j)} \beta +\epsilon\_j \\
1\leq j \leq N\qquad\text{(undefined)}$$

这里的 是真实的输出值即样本中的值，即有 ，我们需要对模型增加假定：这里的 是随机误差，是无法得知的，因此假设满足正态分布 条件 :

$$E(\epsilon\_j)=0 \\Var(\epsilon\_j)=\sigma^2\\1\leq j\leq N\qquad\text{(undefined)}$$

同时 中的每一个 都独立同分布，这也就说明残差 满足相同的正态分布 ，结合 (3.1) 和 (3.3) 就是我们的模型 。

根据之前的假定我们能够得出 的方差

$$Var(\hat{\beta})=Var( \ (X^TX)^{-1}X^Ty \ ) \\ \qquad\text{(3.11)}$$

$$=||(X^TX)^{-1}X^T||^2Var(y)\\=||(X^TX)^{-1}||^2 \cdot ||X^T||^2\cdot \sigma^2\\=(X^TX)^{-1}\sigma^2\qquad\text{(undefined)}$$

那么 的分布也就是 ，但是我们想要知道 的精确度还需要知道 的值，于是我们需要进一步估计 ，利用我们之前计算的残差有

求和前的系数是为了保证 ，即保证 是 的一个无偏估计。这里的残差 满足 分布，那么也可以另外推出

其中 为自由度为 的卡方分布。于是结合 (3.11) 得到 的分布，接下来通过上面decubitus的估计式，我们需要求出 的置信区间，首先建立一个假设

然后我们构造一个标准化的变量

其中， 为矩阵 对角线上第 个元素，(3.12) 在假设 下满足自由度为 的 分布

置信区间分析 待写

### 实例3.1 拟合函数

我们来运用最小二乘法线性回归算法来拟合一个函数 ，对于这样一个函数，我们首先需要做一个变换令 ，，，转化成一个线性回归问题

这里需要更详细的解释 后补

先给出我们所需要的库

# 引入 基本科学运算库  
import numpy as np  
# python 作图库  
import matplotlib.pyplot as plt  
# 三维作图库  
from mpl\_toolkits.mplot3d import Axes3D as aa  
# 引入 sklearn.linear\_model  
from sklearn import linear\_model as limd   
# 引入 交叉验证函数  
from sklearn.model\_selection import train\_test\_split

**第一步：处理数据**

随机产生100组数据，同时加入随机偏移噪声

''' 模型数据处理  
'''  
scale=2 # 坐标数据尺度  
# 随机产生二维数据  
x1=np.random.rand(100,1)\*scale   
x2=np.random.rand(100,1)\*scale  
X=np.ones([100,3]) # 模型的输入数据  
# 矩阵表示 x1+x2^2+x1\*x2  
X[:,0]=x1[:,0]  
X[:,1]=x2[:,0]\*x2[:,0]  
X[:,2]=np.multiply(x1[:,0],x2[:,0])  
# 产生随机误差  
bias=(np.random.rand(100,1)-0.5 )\*0.4\*scale # 偏移系数 0.4  
Z=X[:,0]+X[:,1]+bias[:,0]+X[:,2]

**第二步：拟合**

调用 sklearn.linear\_model 最小二乘法的线性回归函数进行函数拟合。

这是我们模型的核心，这里简单介绍一下过程，limd 是 linear*model 对象，运用的方法是 LinearRegression()，fit(X,Y) 函数是对数据进行拟合，拟合的结果可以从 lr 对象的属性得知，其中 coef* 表示线性模型中的系数 ，最后输出验证结果。lr.score(X,Y) 是对数据 拟合结果的评分。

我们接着介绍 **LinearRegression()** 方法的参数：

**fit\_intercept** : boolean, 默认 True

模型是否使用截距

**normalize** : boolean, 默认 False

当 fit\_intercept 为 False，该参数无意义，自动忽略。如果被设置为 True ，自动进行正则化 L2

**copy\_X** : boolean, 默认 True

X 是否能被修改，True 表示能；False 表示不能

**n\_jobs** : int, 默认 1

计算任务数，达到一定数会自动开启加速优化。

# 加载最小二乘法回归函数  
lr=limd.LinearRegression()  
print "拟合结果\n",lr.fit(Xt,Zt) # 数据进行拟合  
print "系数\n",lr.coef\_  
print "验证结果\n",lr.score(Xtest,Ztest) # 用验证数据进行模型评分

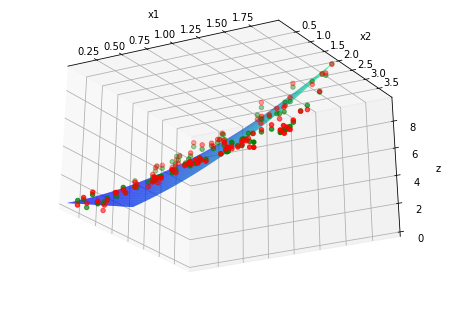
然后给出这段代码的输出结果

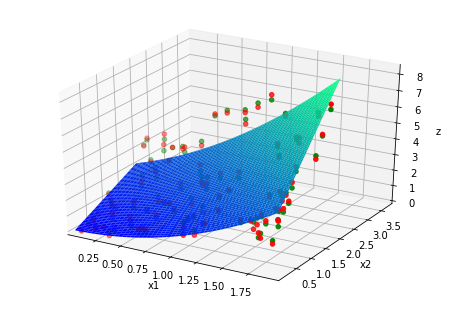
随机曲面拟合  
拟合结果  
LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=1, normalize=False)  
系数  
[ 1.07534516 1.05496164 0.9445371 ]  
验证结果  
0.992572204065

**第三步：作图**

1. 在拟合函数的定义域范围内制作一个二维网格 [XLR,YLR]
2. 然后通过两层循环，计算网格上每一点模型的拟合值 ZLR(i,j)，通过 Axes3D.plot\_surface() 把网格上模型拟合值代表的曲面画出来
3. 分别作出原始数据的散点图（红），以及模型拟合后数据的散点图（绿）
4. 设置图的大小和相关 label 并显示

# pyplot 三维图片加载  
fig=plt.figure()  
# ax 为三维图对象  
ax=aa(fig)  
  
# plot\_surface 部分  
# 在数据范围内生成 meshgrid 网格坐标 100\*100  
Xlr=np.linspace(np.min(x1[:,0]),np.max(x1[:,0]),100)  
Ylr=np.linspace(np.min(x2[:,0]),np.max(x2[:,0]),100)  
# 两个一维向量通过 meshgrid() 生成二维网格  
[XLR,YLR]=np.meshgrid(Xlr,Ylr)  
# 在网格上计算模型的输出值  
i=0  
ZLR=np.ones([100,100])   
for x in Xlr:  
 j=0  
 for y in Ylr:  
 # 对每一个坐标点作为模型的输入求解输出值 ZLR[i,j]  
 ZLR[i,j]=np.dot([x,y\*y,x\*y],lr.coef\_)+lr.intercept\_  
 j=j+1  
 i=i+1  
   
ax.plot\_surface(XLR,YLR,ZLR,cmap='winter') # 绘制模型输出矩阵的曲面  
Zlr=np.dot(X,lr.coef\_)+lr.intercept\_ # 根据矩阵X计算模型输出  
ax.scatter(x1,x2,Z,c='r') # 根据数据的三维散点图  
ax.scatter(x1,x2,Zlr,c='g') # 根据模型的三维散点图  
# 设置视觉范围 仰角45 旋转-60  
ax.view\_init(elev=45,azim=-60)  
# 设置 x y z 取值范围  
ax.set\_xlim([np.min(X[:,0]),np.max(X[:,0])])  
ax.set\_ylim([np.min(X[:,1]),np.max(X[:,1])])  
ax.set\_zlim([np.min(Z),np.max(Z)])  
# 设置坐标 label  
ax.set\_xlabel('x1')  
ax.set\_ylabel('x2')  
ax.set\_zlabel('z')  
# 展示  
plt.show()





图片说明：对于我们拟合的曲面，以及原数据散点（红），拟合数据散点（绿），从图中可以看出，所有数据点都位于平面两侧不太远的地方。可以看出我们的拟合精度还是可以接受的，随机偏移噪声的影响对我们最终的模型结果影响并不是非常大。

### 梯度下降方法

在数学中，梯度代表的是一个函数的最快上升方向，比如给定一个定义在 $\R^2$ 上的单值连续光滑函数 ，想要知道它在某点 的增加量最快的方向就是

$$\nabla {f}=\frac{\part{f}}{\part{x}}\hat{i} + \frac{\part{f}}{\part{y}}\hat{j} + \frac{\part{f}}{\part{z}}\hat{k}\qquad\text{(undefined)}$$

其中 是沿 轴的单位向量，那么梯度向量就是 $(\frac{\part f}{\part x},\frac{\part f}{\part y},\frac{\part f}{\part z})$.

那么我们如何利用梯度向量求解最优化问题呢？显然，我们写出一个目标函数，就是要寻找目标函数在约束条件下的极值点。

先给出梯度下降方法的基本思想：设定一组关于 参数的初始值，以及学习率，然后通过根据梯度和学习率，不断迭代和修改 值，使得最终的目标函数达到极值，这个过程就是梯度下降。例如（3.6）是一个凸函数最小值问题，以此为例，我们想要求解出使得 最小的 值，可以通过不断迭代

$$\beta^{t+1}=\beta^t-\alpha \frac{\part R}{\part \beta} \qquad\text{(3.13)}$$

其中 表示迭代次数， 是学习率表示单次迭代的下降快慢程度，这里 是一个向量。化解成展开形式为

因为有

$$\frac{\part R(\beta)}{\part \beta}=\frac{\part}{\part \beta}\sum\_{i=1}^N\frac{1}{2}{(y^{(i)}-\sum\_{j=1}^pX\_j^{(i)}\beta\_j)^2}\\
=[-\sum\_{i=1}^N{ (y^{(i)}-\sum\_{j=1}^p{X\_j^{(i)}\beta\_j})X\_k^{(i)} }]\_k \qquad\text{(3.15)}$$

最后结果是一个与 适合的向量，$\frac{\part R(\beta)}{\part \beta\_k}$表示分量 的梯度大小。

可以看到基本原理和爬坡的过程非常相似。

这里给出梯度下降算法框架：

1. 确定学习参数

的值应该适中，如果过大就会不收敛或发生震荡现象；如果过小收敛会非常慢，浪费计算资源。

1. 设定初始值

让 从 ，开始迭代。

1. 迭代计算

如果没有达到收敛，迭代将继续下去。

1. 判断是否收敛

当 的值小到一个我们可以接受的误差 的时候，我们就停止计算；否则继续3迭代。

最终求解出来的 就是最终的最优参数值。

代码 图片

### Lasso方法与Ridge方法

Lasso方法和岭回归（Ridge）方法属于收缩方法（shrinkage method），他们的基本思想是相近的：通过在目标函数增加一个关于系数的代价，使得在求解过程中系数也会逐渐收缩趋近于零。使用收缩方法有两个好处：其一能够有效避免模型的过拟合问题，即模型的训练得和数据过于一致以至于出现了泛化程度不高的现象。其二能够避免 不是满秩的问题，前面说过（3.9）需要满足 为列满秩的假设才能够使得 是可逆的，因此即使最小二乘法因为数据原因不能进行下去的情况下，收缩方法有效。

**Ridge方法**

岭回归是这样定义目标函数的

其中正则化参数 表示收缩程度， 越大，收缩程度越大。如果在模型中有很多的线性相关变量，那么模型的系数 的值是不稳定的，即存在多个 满足（3.7），并且 具有很大的方差。不过在岭回归方法中，这种情况得到缓解，因为伴随着惩罚项的收缩，那些相关项的系数可能会形成正负相消，具体说在一个变量上的一个很大的正系数可能被在其相关变量上的类似大小的负系数抵消，从而不对最终求解出的 值产生影响。选取大小合适的 参数值是非常重要的一点，若 过大，导致 过小，欠拟合；若 过小，导致 过大，过拟合，因此 值应该适中。

同时我们还应该注意到，如果想要分析方差，岭回归模型中对输入样本 的要求：对于每一个样本必须是经过标准化的，因为随着 的缩放，最终求解方差的结果会有很大差别。

centering 待写 equivariant

同样将 第一列用 $\bf{1}$ 向量代替， 为 矩阵。用矩阵的形式表示岭回归目标函数

按照之前说的线性回归最小二乘法偏导求解的方程得到

可以注意到 一定是可逆的，因此 可直接求解。这就是岭回归模型的优势之一：即使因为某些原因 不列满秩，（3.15）仍然成立。若 是正交矩阵，那么岭回归和最小二乘法的解相差一个倍数，即 。

梯度下降法公式 岭迹图

方差分析 主成分分析

**Lasso方法**

Lasso方法在很大程度上和岭回归是类似的，都是利用正则化项进行惩罚。只不过Ridge法用范数，Lasso法用范数，在本质和目的上没有什么不同。

Lasso方法的拉格朗日形式为

可以看到（3.16）和（3.13）非常相似，除了最后的惩罚项都是相同的。（3.16）是一个二次规划问题，我们也可以采用软件自带的优化包解决，同时数值优化方法还有坐标下降算法，最小角回归算法，近端梯度下降算法等，原理部分都超出本书的范围，需要读者自行去网络上寻找资源。

gradient decreasing method

Lp范数的讨论

为了能够更好地理解 Lasso 和 Ridge 之间的联系，我们假设他们的变量 都是二维向量。根据 Lasso 的约束条件有 其中 为一常数，再根据 Ridge 约束条件有 其中 ，这就对应了（图）中的情形，同时有目标函数 的等值线，每一条等值线都代表了一个函数取值，其中中心 P 是函数的极小值，我们需要在令目标函数值尽可能得小和满足约束条件之间做一个折中，也就是正好取的是等值线和约束曲线相切的那个点，这就是我们的最优情况。

图片说明 beta

Lasso 方法和 Ridge 方法的不同之处在于正则化项。现在来讨论一般情况，对于一般情形的正则化项 对于不同的 的取值，约束曲线的情况是不同的。我们可以从（图）中看出，当 时，约束曲线是凸函数；当 时，约束曲线是凹的。当 时，约束代表的是 Lasso 与 Ridge 的折中。特殊地，当 时，是令曲线满足凸性质的最小的 的取值。当 时，方差分析不太方便，当 时，便不是一个凸优化的求解问题了，问题将变得更加复杂。因此常用的就是 Lasso 或 Ridge 的正则化项。

### 实例3.2 预测波士顿房价

这节我们会应用之前所学的线性回归方法来对 sklearn 中自带的数据集进行学习并且预测。

**需要用到的库**

import numpy as np
  
   
# 作图
  
import matplotlib.pyplot as plt
  
   
# 线性模型
  
from sklearn import linear\_model as limd
  
   
# 数据集 波士顿房价
  
from sklearn.datasets import load\_boston as lb
  
   
# 数据预处理
  
from sklearn import preprocessing as pp
  
   
# 数据归一化预处理
  
from sklearn.preprocessing import StandardScaler as ss
  
   
# 交叉验证
  
import sklearn.cross\_validation as cv
  
   
# 矩阵计算
  
from sklearn import metrics

**加载数据**

预测房价需要分析哪些变量和房价的高低相关呢？Boston 数据集已经给出了13个特征，我们首先直接加载它。

print "回归预测房价"
  
# 加载 boston 数据
  
boston=lb()
  
# 输出数据特征
  
print boston['DESCR']

我们加载完毕后，分析数据集 boston 的组成和结构:

boston.keys()
  
   
Out[35]: ['data', 'feature\_names', 'DESCR', 'target']

看出 boston 由4个主要键值，data 表示输入特征集；feature\_names 表示特征名称；DESCR 表示数据集的描述，包括数据名称，特征单位以及数据值等；target 表示目标值，即训练的目标输出。

这里简单介绍一下 boston 数据集的描述（DESCR）信息：

Data Set Characteristics:
  
   
 :Number of Instances: 506
  
   
 :Number of Attributes: 13 numeric/categorical predictive
  
   
 :Median Value (attribute 14) is usually the target
  
   
 :Attribute Information (in order):
  
 - CRIM 人均犯罪率
  
 - ZN 超过 25,000 平方英寸的住宅区占地比例
  
 - INDUS 非零售区占地比例
  
 - CHAS 是否靠近河边 （1 靠近；0 不靠近）
  
 - NOX 一氧化氮浓度
  
 - RM 每个居住房屋的平均房间数
  
 - AGE 在1940年之前建成且业主自主的房子的比例
  
 - DIS 距波士顿市中心的距离
  
 - RAD 周边高速大道的可到达指数
  
 - TAX 每 10,000 美元的财产税率
  
 - PTRATIO 师生比例
  
 - B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
  
 - LSTAT % lower status of the population
  
 - MEDV Median value of owner-occupied homes in $1000's

#### Ridge 回归

输出

回归预测房价
  
['data', 'feature\_names', 'DESCR', 'target']
  
模型
  
Ridge(alpha=1.0, copy\_X=True, fit\_intercept=True, max\_iter=None,
  
 normalize=False, random\_state=None, solver='auto', tol=0.001)
  
训练拟合评分
  
 0.774
  
预测均方误差
  
 28.300
  
系数
  
 [-0.10420072 0.01474884 0.01109709 3.29285993 -7.86386651 4.51696735
  
 -0.01665711 -1.11167889 0.29372571 -0.01323551 -0.99963203 0.01010772
  
 -0.49263281]
  
截距
  
27.3570597241

#### 模型优化

多项式回归预测房价
  
模型
  
Ridge(alpha=1.0, copy\_X=True, fit\_intercept=True, max\_iter=None,
  
 normalize=False, random\_state=None, solver='auto', tol=0.001)
  
训练拟合评分
  
 0.937
  
预测均方误差
  
 17.000
  
系数
  
 [ 0.00000000e+00 -3.89936462e-01 -4.87538997e-01 5.08679134e-02
  
 2.87410466e-01 -1.63111424e-02 5.66393955e-01 8.44987615e-01
  
 6.07480708e-01 1.54690940e+00 -2.41539630e-01 -8.14558400e-01
  
 2.86702975e-01 -1.07054133e+00 6.40090955e-05 -4.79791833e-02
  
 4.42453072e-01 2.37672412e+00 -1.26862645e+00 1.36894909e-01
  
 -5.70754016e-03 -4.68481571e-02 4.83421541e-01 -3.67209121e-02
  
 2.64272903e-01 -5.15202891e-04 2.64557986e-02 4.78746061e-04
  
 -3.18185939e-03 -3.86146676e-02 1.00658318e+00 -2.39990812e-02
  
 -2.60477571e-03 -1.95919257e-02 1.14699771e-02 9.89940832e-05
  
 -7.73933333e-03 1.12503167e-03 -6.32630125e-04 1.80752464e-02
  
 -4.94050764e-01 2.45084774e+00 1.43617633e-02 6.07630977e-03
  
 7.68412909e-02 8.40095210e-03 1.33806111e-03 -3.45348451e-02
  
 -4.25898781e-03 -5.54562243e-02 2.87410466e-01 -8.67903967e-01
  
 -1.76821863e+00 4.33367657e-02 9.60370774e-01 -6.40151350e-01
  
 9.92910310e-03 5.99444645e-01 3.12144425e-03 -4.37794699e-02
  
 -2.10536319e-01 -6.86387202e-01 -4.21012303e-01 3.13339659e+00
  
 -8.54920189e-01 -1.05180408e-01 -6.41916953e-01 4.38028044e-02
  
 2.30281944e+00 1.12752900e+00 -3.01039860e-02 3.44783892e-01
  
 -3.86804041e-01 1.06023489e-02 -2.44474066e-01 -8.94430049e-03
  
 -1.43210380e-01 -4.66478461e-05 1.29539329e-02 1.74945130e-02
  
 -5.62617934e-04 -1.00504983e-02 -7.03764883e-04 -5.09043206e-03
  
 4.88657896e-01 -1.95085901e-01 -5.63478638e-04 -1.14003008e-01
  
 -2.17944277e-02 8.99680766e-02 -1.70615053e-01 7.23160193e-03
  
 1.31705767e-01 -1.77171411e-03 -8.33276129e-02 -1.42630946e-05
  
 5.86313426e-03 2.20041205e-04 1.70866655e-03 7.96239919e-02
  
 -5.35703514e-03 5.07871263e-02 -7.50445589e-05 -1.58708090e-03
  
 1.84004046e-02]
  
截距
  
-18.1240331127

## 线性分类