第三章 线性回归

回归与分类的区别在于目标变量：

- 回归——连续数值型
- 分类——标称型

回归中最常用的是线性回归（linear regression），即目标变量与各输入变量之间成线性关系。假设房价（y）与房屋面积（x）近似为线性：

$$y = ax + b$$

如果有一系列的样本点：

$$D = \{(x_0, y_0), (x_1, y_1), \cdots, (x_{n-1}, y_{n-1})\}$$

利用线性回归可以确定模型中a、b的值，然后利用模型就可以预测任意面积房屋的价格。

实际应用中，可能因变量y与自变量x之间并不成线性关系，但可以转化为线性关系来处理，比如y与x成指数关系：

$$y = ae^{bx}$$

通过取对数可得：

$$\ln(y) = \ln(a) + bx$$

于是$\ln(y)$与x之间成线性关系。再比如y与x成多项式关系：

$$y = a + bx + cx^2 + dx^3$$

定义新的自变量$x_1=x$，$x_2=x^2$，$x_3=x^3$，则有：

$$y = a + bx_1 + cx_2 + dx_3$$

此时y与新的自变量x_1、x_2、x_3都成线性关系。

1. 线性回归的正规方程解法

线性模型：

$$h_w(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_{m-1} x_{m-1} = \sum_{j=0}^{m-1} w_j x_j = \boldsymbol{x}W$$

在上式中，我们对每个样本点***x***都加一个分量x_0, 以便将偏置项W_0也写到加和式中。

假设有n个样本点：$D = \{(x^{(0)}, y^{(0)}), (x^{(1)}, y^{(1)}), \cdots, (x^{(n-1)}, y^{(n-1)})\}$

写成矩阵形式:

$$X = \begin{bmatrix} 1 & x_1^{(0)} & \cdots & x_{m-1}^{(0)} \\ 1 & x_1^{(1)} & \cdots & x_{m-1}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n-1)} & \cdots & x_{m-1}^{(n-1)} \end{bmatrix} \qquad Y = \begin{bmatrix} y^{(0)} \\ y^{(1)} \\ \vdots \\ y^{(n-1)} \end{bmatrix} \qquad W = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{m-1} \end{bmatrix}$$

显然应该有$n \geq m$。如果$n=m$，则X为方阵，易得W，令：

$$h_w\left(x^{(i)}\right) = y^{(i)} \Longrightarrow XW = Y \Longrightarrow W = X^{-1}Y$$

当$n>m$时，可通过最小二乘法求解：

设置最小二乘法损失函数(loss function):

$$J(W) = \frac{1}{2n}\sum_{i=0}^{n-1}\left[h_w\left(x^{(i)}\right) - y^{(i)}\right]^2$$

$$= \frac{1}{2n}(XW - Y)^{\mathrm{T}}(XW - Y)$$

$$= \frac{1}{2n}(W^{\mathrm{T}}X^{\mathrm{T}} - Y^{\mathrm{T}})(XW - Y)$$

$$= \frac{1}{2n}(W^{\mathrm{T}}X^{\mathrm{T}}XW - W^{\mathrm{T}}X^{\mathrm{T}}Y - Y^{\mathrm{T}}XW + Y^{\mathrm{T}}Y)$$

$$J(W)=\frac{1}{2n}(W^{\mathrm{T}}X^{\mathrm{T}}XW-W^{\mathrm{T}}X^{\mathrm{T}}Y-Y^{\mathrm{T}}XW+Y^{\mathrm{T}}Y)$$

由于W^TX^TY是一行一列的矩阵，其转置等于本身：

$$W^{\mathrm{T}}X^{\mathrm{T}}Y=(W^{\mathrm{T}}X^{\mathrm{T}}Y)^{\mathrm{T}}=Y^{\mathrm{T}}XW$$

则：

$$J(W)=\frac{1}{2n}(W^{\mathrm{T}}X^{\mathrm{T}}XW-2W^{\mathrm{T}}X^{\mathrm{T}}Y+Y^{\mathrm{T}}Y)$$

要使$J(W)$取最小值则需要求导，利用矩阵求导公式：

$$\frac{d(X^{\mathrm{T}}AX)}{dX}=(A+A^{\mathrm{T}})X,\quad \frac{d(X^{\mathrm{T}}A)}{dX}=A$$

$$\frac{dJ(W)}{dW}=\frac{1}{2n}(2X^{\mathrm{T}}XW-2X^{\mathrm{T}}Y)=0 \quad \Longrightarrow$$

$X^{\mathrm{T}}XW=X^{\mathrm{T}}Y$ 正规方程（normal equation）

$$X^{\mathrm{T}}XW = X^{\mathrm{T}}Y$$

解得： $W = (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}Y$

实际上$(X^TX)^{-1}X^T$即为X的广义逆矩阵。与m=n时的W=$X^{-1}Y$相比，上式即利用X的广义逆矩阵代替X的逆矩阵。

正规方程解法需要X^TX矩阵可逆。

要衡量目标变量与输入之间线性关系的强弱程度，常使用Pearson相关系数来描述：

- 当相关系数为0时，无相关关系。
- 相关系数在0与1之间，正相关关系，越接近于1表示相关性越强。
- 相关系数在-1与0之间，负相关关系，越接近于-1表示相关性越强。

2. 线性回归正规方程解法的python实现

已经搜集到房屋面积与房价的十个样本，数据如表1所示。

表1 房屋面积与房价数据表

面积(m^2)	52	55	60	75	78	80	84	92	95	98
房价(万元)	160	152	250	280	310	350	320	345	380	350

利用python编程实现线性回归过程，并确定房价预测模型。

所得线性模型为：

y = -58.5981 + 4.529234x

线性相关系数为0.937。

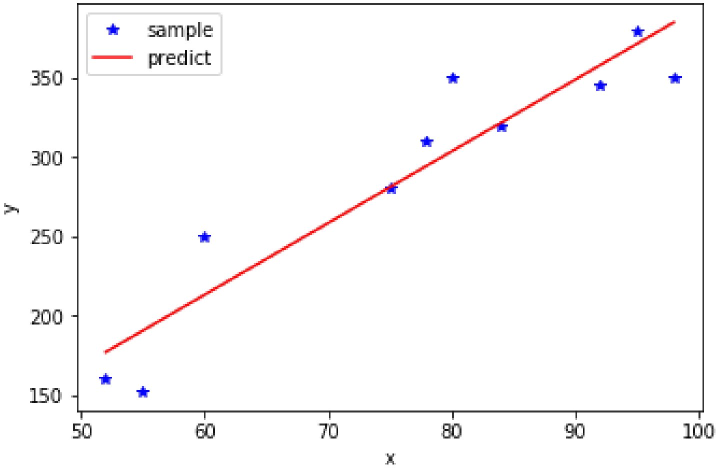

图3-1 房价模型与样本点比较

3. 欠拟合与过拟合

从图3-1看出，其实线性模型的拟合效果一般，那么能否通过多项式拟合来提高模型的准确度呢？

下面采用不同阶多项式进行拟合，为了便于比较，我们将样本集分本训练集（6个样本点）和测试集（4个样本点），分别计算n阶多项式模型在训练集和测试集上的平均误差。

程序输出结果如下：

while n = 1,
Correlation Coefficient: 0.9370
Average train error: 19.09
Average test error: 24.74

while n = 2,
Correlation Coefficient: 0.9636
Average train error: 15.21
Average test error: 21.43

while n = 3,
Correlation Coefficient: 0.9645
Average train error: 15.15
Average test error: 21.76

while n = 4,
Correlation Coefficient: 0.9660
Average train error: 13.37
Average test error: 23.50

while n = 5,
Correlation Coefficient: 0.9685
Average train error: 9.30
Average test error: 25.48

从结果可以看出，2阶多项式比线性模型拟合效果更好，相关系数从0.937提高到0.964，在训练集上的误差及测试集上的误差也都减小。但随多项式次数的进一步增加，模型相关系数略有增大，在训练集上的误差减小，但在测试集上的误差却开始变大。这样的结果说明，2阶或3阶多项式模型拟合效果最好，线性模型欠拟合（underfit），而4次及更高次多项式模型则发生了过拟合（overfit）。

欠拟合是指机器学习模型未能学习到数据背后的规律性，表现为在训练集和测试集上都存在较大的误差。
过拟合是指模型过度学习了样本集中的细节和噪音，表现为在训练集上误差小而在测试集上误差大。

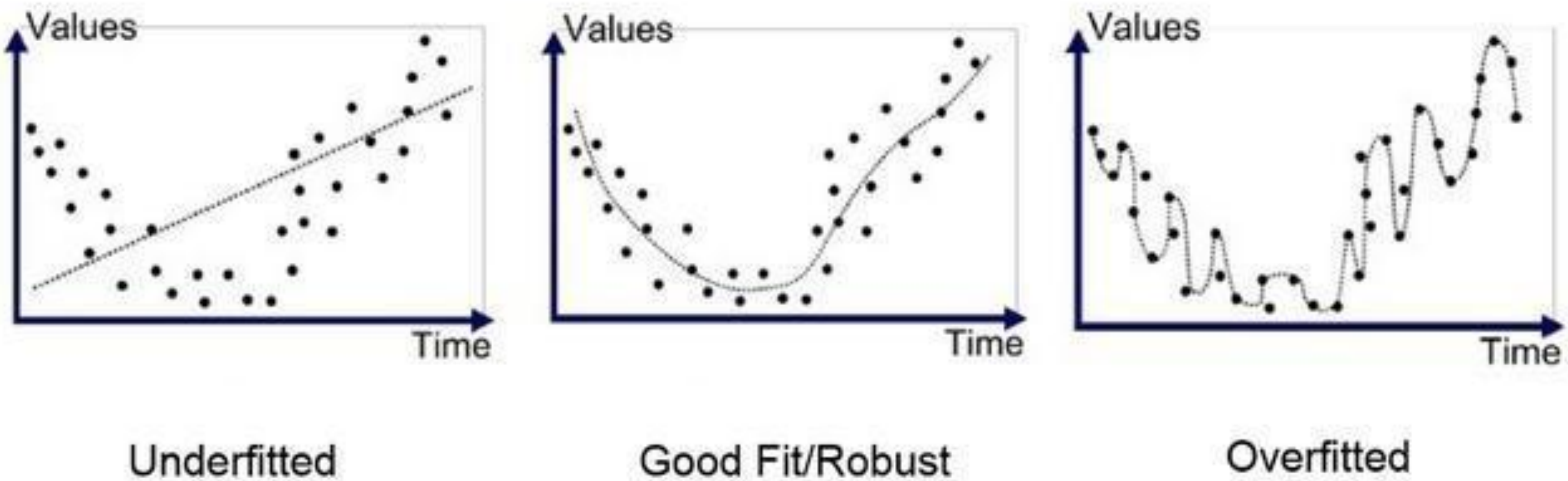

图3-2 欠拟合与过拟合的图示说明

欠拟合常用的解决方法有：

- 增加模型复杂度
- 增加数据特征

过拟合常用解决方法有：

- 增加训练数据量
- 正则化（regularization）

4. 正则化

前面提到，样本数n一定要大于等于数据的特征数m。如果数据的特征比样本点还多怎么办？显然不能采用上面的方法求解，因为X^TX不是满秩矩阵，无法求逆。此时可以在矩阵X^TX上加一个αI从而使矩阵非奇异，进而对$X^TX+\alpha I$求逆。其中α是一个用户定义的数值，此时回归系数的计算公式变成：

$$W=(X^{\mathrm{T}}X+\alpha I)^{-1}X^{\mathrm{T}}Y$$

这相当于在损失函数中增加了一个惩罚项，即：

$$J(W)=\frac{1}{2n}\left\{\sum_{i=0}^{n-1}\left[h_w\left(x^{(i)}\right)-y^{(i)}\right]^2+\alpha\sum_{j=0}^{m-1}w_j^2\right\}$$

$$=\frac{1}{2n}(W^{\mathrm{T}}X^{\mathrm{T}}XW-2W^{\mathrm{T}}X^{\mathrm{T}}Y+Y^{\mathrm{T}}Y+\alpha W^{\mathrm{T}}W)$$

$$J(W)=\frac{1}{2n}(W^{\mathrm{T}}X^{\mathrm{T}}XW-2W^{\mathrm{T}}X^{\mathrm{T}}Y+Y^{\mathrm{T}}Y+\alpha W^{\mathrm{T}}W)$$

为使$J(W)$取最小值，则需要：

$$\frac{dJ(W)}{dW}=\frac{1}{2n}(2X^{\mathrm{T}}XW-2X^{\mathrm{T}}Y+2\alpha W)=0 \quad \Longrightarrow$$

$$(X^{\mathrm{T}}X+\alpha I)W=X^{\mathrm{T}}Y \quad \Longrightarrow \quad W=(X^{\mathrm{T}}X+\alpha I)^{-1}X^{\mathrm{T}}Y$$

通过在损失函数中引入惩罚项$\alpha W^{\mathrm{T}}W$，可以限制w_j的值，让所有w_j向0的方向靠近，从而可以减少一些不重要的参数，这个技术在机器学习中称为正则化。

正则化的影响程度可以通过设置α的值来调整， α=0时与普通的线性模型无异，随α的增大，正则化影响程度增加，当α→∞时，所有w_j都将趋近于0。

上面的惩罚项为w_j的平方和，称为岭回归（也叫L2正则化），岭回归名称的由来是其中用到了αI，单位矩阵I只有对角线上元素为1，其余都为0，很象是在0构成的平面上有一条1组成的岭。常用的还有另一种正则化称为Lasso回归（L1正则化），其惩罚项为w_j的绝对值之和：

$$J(W) = \frac{1}{2n}\left\{\sum_{i=0}^{n-1}\left[h_w\left(x^{(i)}\right) - y^{(i)}\right]^2 + \alpha\sum_{j=0}^{m-1}\left|w_j\right|\right\}$$

正则化通过限制w_j的值可以避免过拟合，Lasso回归通常可以获得稀疏解，即使得部分w_j变为0，这种稀疏解具有更好的可解释性。但它在惩罚项中使用了绝对值，数学处理上更加困难，算法实现上增加了难度。

5. 岭回归的python实现

在文件abaone.txt中存放着鲍鱼的8个特征与年龄（第9列）数据，下面我们利用岭回归来建立模型以预测鲍鱼的年龄。

通过第二章中建立的MinMaxNorm类来对数据进行归一化。

利用holdout_split类将数据集分割为训练集和测试集。

利用不同α值的岭回归在训练集上建立模型，对测试集计算平均预测误差。

程序输出 (由于数据分割的随机性，每次运行结果略有不同):

```
while alpha = 0.000000, average error = 1.645742
while alpha = 0.001000, average error = 1.645700
while alpha = 0.010000, average error = 1.645325
while alpha = 0.100000, average error = 1.642166
while alpha = 1.000000, average error = 1.639832
while alpha = 10.000000, average error = 1.713220
while alpha = 100.000000, average error = 1.837783
```

从结果看，普通线性模型（即α=0）已经能对这个数据模型给出较好的预测，平均误差约为1.646，通过岭回归并没有明显改善预测精度。但还是可以看出，α很小时，与普通线性模型的误差相似，大约在α=0.1-1的范围取得最优结果，α进一步增大会使误差变大。

6. 梯度下降算法

在前面的分析中，线性回归的问题最终转化为求损失函数 $J(W)$的最小值问题:

$$J(W)=\frac{1}{2n}\sum_{i=0}^{n-1}\left[h_w\left(x^{(i)}\right)-y^{(i)}\right]^2=\frac{1}{2n}\sum_{i=0}^{n-1}\left[\sum_{j=0}^{m-1}w_j x_j^{(i)}-y^{(i)}\right]^2$$

$J(W)$是w_j的函数。

求函数极值，除了可通过求导的方法以外，也可以通过优化的方法实现，包括动态规划、梯度下降、共轭梯度等以及现代进化算法如遗传算法、粒子群算法等。

下面我们介绍梯度下降（Gradient Descent）算法在线性回归问题中的应用。

6.1 梯度下降算法原理

我们先以单变量线性回归为例，此时m = 2，只有w_0和w_1两个变量。考虑一个简单的函数：

$$J(w_0, w_1) = 4w_0^2 + w_1^2$$

画出该函数的图形如图3-3所示。

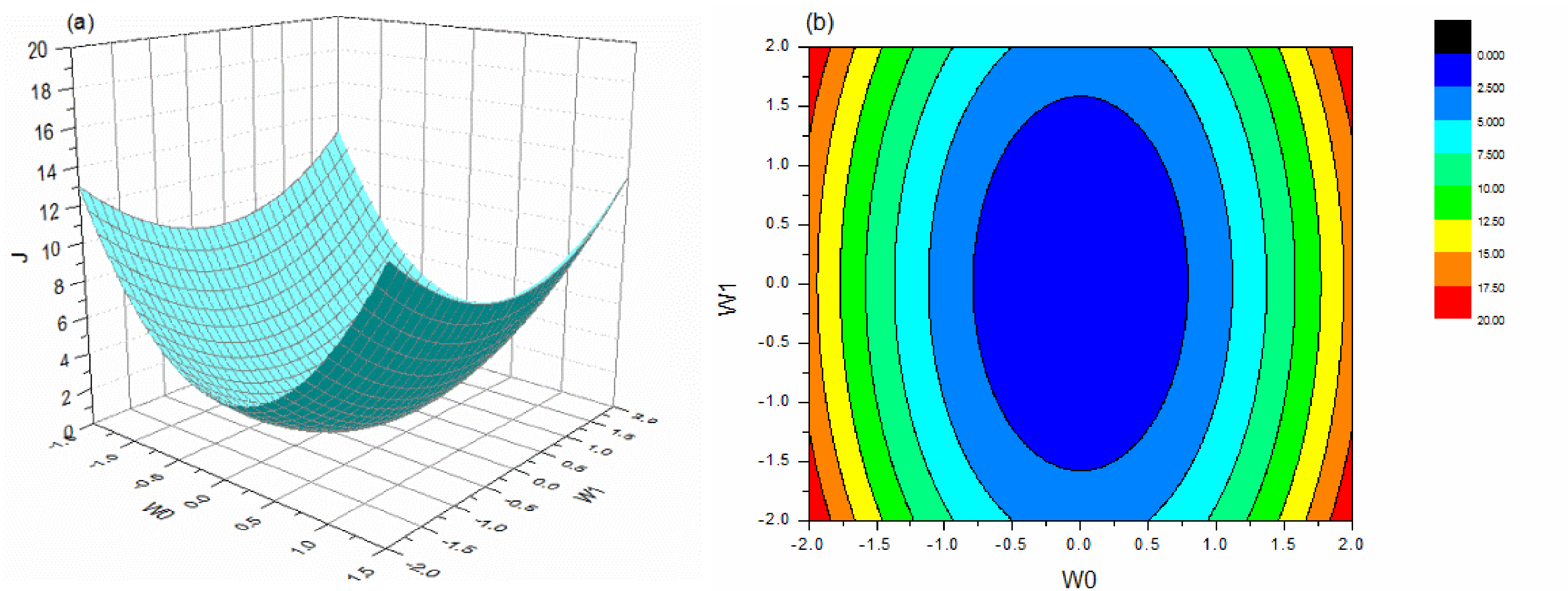

图3-3 函数$J(w_0,w_2)$图像

在梯度下降算法中，先选择一个初值 $W^{(0)} = \left[w_0^{(0)}, w_1^{(0)}\right]^{\mathrm{T}}$，然后沿着函数值下降的方向逐渐找到极小值点，为了加快收敛的速度，希望每一步都沿着最陡的方向即函数值下降最快的方向前进。

根据数学知识，任何一点处函数值增大最快的方向为该点的梯度方向，相应的下降最快的方向就在梯度的反方向，该方向与通过该点的等高线垂直。

还有一个问题是前进多远的距离，通常采用的方法是人为选取一个步长η（learning rate），则:

$$w_0^{(1)} = w_0^{(0)} - \eta \left.\frac{\partial J}{\partial w_0}\right|_{W^{(0)}}$$

$$w_1^{(1)} = w_1^{(0)} - \eta \left.\frac{\partial J}{\partial w_1}\right|_{W^{(0)}}$$

然后在$W^{(1)}$的位置再计算新的梯度并沿梯度反方向继续前进，直到前进的距离（或沿任一分量前进距离的最大值）小于指定精度要求ε，算法收敛。

根据：

$$J(W)=\frac{1}{2n}\sum_{i=0}^{n-1}\left[\sum_{j=0}^{m-1}w_j x_j^{(i)}-y^{(i)}\right]^2$$

得到梯度下降算法迭代公式为：

$$w_j=w_j-\eta\frac{\partial J}{w_j}=w_j-\frac{\eta}{n}\sum_{i=0}^{n-1}\left[h_w\left(x^{(i)}\right)-y^{(i)}\right]x_j^{(i)}$$

6.2 梯度下降算法的变体

- 批量梯度下降（Batch Gradient Descent， BGD）算法，每次参数更新要使用全部样本点，当样本数量较大时，计算较为耗时；
- 随机梯度下降（Stochastic Gradient Descent， SGD）算法，每次参数更新只使用一个样本点，虽然迭代次数会增加，但每次迭代的计算量大幅减小；
- 小批量梯度下降（Mini-Batch Gradient Descent，MBGD）算法，每次参数更新使用用户定义的p个样本点，1<p<n，通常选择p = 10。

6.3 BGD算法的python实现

利用BGD算法建立房价预测模型。

程序输出结果：

```
Iteration number: 1370341
w =
 [[-58.34678637]
 [  4.52609605]]
```

得到的拟合参数w与前面正规方程解法的结果一致。

在梯度下降算法中，步长的选择很重要，步长太大会导致震荡不收敛，步长太小则收敛较慢。而正规方程法是计算解析解，如果样本量不大（<10000），推荐使用正规方程法、岭回归或Lasso。当样本量很大时，可以选择梯度下降法，并通过随机或小批量梯度下降算法提高计算效率。另外，对数据进行归一化处理往往能提高梯度下降算法的收敛效率。

7. 利用scikit-learn进行线性回归

在scikit-learn的linear_model模块，实现了多个线性回归的类：

- LinearRegression类用于通常的最小二乘法回归
- Ridge类用于岭回归（L2正则化）
- Lasso类用于Lasso回归（L1正则化）
- SGDRegressor类用于随机梯度下降法线性回归

7.1 LinearRegression类

LinearRegression(fit_intercept=True, normalize=False, copy_X=True, n_jobs=None)

属性：

- coef_: 回归系数
- intercept_: 截距

方法：

- fit(X, y, sample_weight=None)：对数据集X、y进行拟合。
- predict(X)：利用线性模型对新数据点X进行预测。
- score(X, y, sample_weight=None)：计算并返回模型的评分，即数据集X、y线性回归的相关系数的平方。

利用LinearRegression类对前面的房价数据进行线性回归.

输出结果为：

```
intercept: -58.598107
coefficient: 4.529234
```

7.2 Ridge类

```
Ridge(alpha=1.0, fit_intercept=True, normalize=False,
copy_X=True, max_iter=None, tol=0.001, solver='auto',
random_state=None)
```

利用Ridge类回归房价数据，比较不同正则化强度时的回归系数值。输出结果为：

```
alpha = 0.000000, intercept: -58.5981, coefficient: 4.5292
alpha = 0.100000, intercept: -58.5839, coefficient: 4.5290
alpha = 1.000000, intercept: -58.4561, coefficient: 4.5274
alpha = 10.000000, intercept: -57.1828, coefficient: 4.5108
alpha = 100.000000, intercept: -44.9442, coefficient: 4.3517
alpha = 1000.000000, intercept: 42.3315, coefficient: 3.2168
```

7.3 Lasso类

Lasso(alpha=1.0, fit_intercept=True, normalize=False, precompute=False, copy_X=True, max_iter=1000, tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic')

利用Lasso回归房价问题并比较不同正则化强度的结果，输出：

```
alpha = 0.000000, intercept: -58.5981, coefficient: 4.5292
alpha = 0.100000, intercept: -58.5667, coefficient: 4.5288
alpha = 1.000000, intercept: -58.2843, coefficient: 4.5252
alpha = 10.000000, intercept: -55.4605, coefficient: 4.4884
alpha = 100.000000, intercept: -27.2219, coefficient: 4.1212
alpha = 1000.000000, intercept: 255.1642, coefficient: 0.4491
```

7.4 SGDRegressor类

SGDRegressor(loss='squared_loss', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=None, tol=None, shuffle=True, verbose=0, epsilon=0.1, random_state=None, learning_rate='invscaling', eta0=0.01, power_t=0.25, early_stopping=False, validation_fraction=0.1, n_iter_no_change=5, warm_start=False, average=False, n_iter=None)

7.5 利用scikit-learn线性拟合Boston房价数据

在scikit-learn自带的数据包中，有一个Boston房价数据集，其中收集了506个样本，包含有13个特征，分别是：

1. CRIM（城镇人均犯罪率）

2. ZN（城镇超过25000平方英尺的住宅区域的占地比例）
3. INDUS（城镇非零售用地占地比例）
4. CHAS（是否靠近河边，1为靠近，0为远离）
5. NOX（一氧化氮浓度）
6. RM（每套房产的平均房间个数）
7. AGE（在1940年之前建成且业主自住的房子的比例）
8. DIS（与Boston市中心的距离）
9. RAD（周边高速公路的便利性指数）
10. TAX（每10000美元的财产税率）
11. PTRATIO（小学老师的比例）
12. B（城镇黑人的比例）
13. LSTAT（地位较低的人口比例）。

下面我们分别采用原始数据及2阶和3阶多项式来回归Boston房价数据集，为了产生多项式特征数据，我们可以利用scikit-learn preprocessing模块的PolynomialFeatures类，该类可以方便地对原始数据添加多项式特征，其用法如下：

PolynomialFeatures(degree=2, interaction_only=False, include_bias=True)

参数含义：

- degree: 多项式阶数，假设输入样本包含两个特征[a, b]，则2阶多项式特征包括[1, a, b, a*a, ab, b*b]。
- interaction_only: 当设置为True时，只产生相互作用特征，象a * a, b * b这样的特征将不包括在内。
- include_bias: True表示包括偏差列（截距项），即首列的1，False则不包含偏差列。

主要属性有：

- n_input_features_: 输入特征的数目。
- n_output_features_: 输出特征的数目。

主要方法包括：

- fit(X): 对输入X计算输出特征。
- fit_transform(X): 对X计算输出特征，并返回新的特征矩阵。
- transform(X): 将输入转化为多项式特征。

下面分别采用1阶、2阶、3阶多项式对Boston房价进行回归，先读入原始数据并打印数据结构，然后循环产生多项式特征，将数据集分割为训练集和测试集，调用LinearRegression类对归一化的数据进行回归，最后打印模型在训练集和测试集上的评分。

输出结果（每次运行结果会有不同）：

```
X shape: (506, 13), y shape: (506,)
degree = 1, train score: 0.733635, test score: 0.741874
degree = 2, train score: 0.937639, test score: 0.823159
degree = 3, train score: 1.000000, test score: -441.546723
```

从结果可以看出，如果只使用原始数据的13个特征，拟合效果一般，在训练集和测试集上的评分都在0.7左右，说明发生了欠拟合。

当增加2阶多项式特征后，拟合效果提高了，在训练集上的评分达到0.93，在测试集上的评分达到0.82。

但当进一步增加3阶多项式特征时，训练集上的评分为1，而在测试集上的评分却为负值，明显发生了过拟合现象。