第5章

回归分析

生活中存在很多相互制约又相互依赖的关系，这些关系主要有确定关系和非确定关系。确定关系指变量之间存在明确的函数关系，如圆的周长(L)与半径(r)之间的关系为$L=2\pi r$。非确定关系指各变量之间虽然有制约/依赖关系，但无法用确定的函数表达式来表示，如人的血压与体重之间存在密切关系，但无法找到一个能准确表达其关系的函数，变量之间存在的这种非确定关系，称为相对关系。

事实上，有一些确定关系，由于测量误差的影响，也经常表现出某种程度的不确定性。对于非确定关系，通过大量观测数值，可以发现其中变量间存在的统计规律。通过回归分析，可以发现自变量和因变量之间的显著关系或多个自变量对一个因变量的影响强度。回归问题在形式上与分类问题十分相似，但是在分类问题中，预测值 y 是一个离散变量，它代表通过特征 x 所预测出来的类别；而在回归问题中，y 是一个连续变量。

视频讲解

5.1 回归分析概述

5.1.1 回归分析的定义与分类

回归分析是一种预测性的建模技术，它研究的是因变量(目标)和自变量(预测器)之间的关系。具体来说，回归分析是指利用数据统计原理，对大量统计数据进行数学处理，并确定因变量与某些自变量的相关关系，建立一个相关性较好的回归方程(函数表达式)，并加以外推，用于预测今后因变量变化的分析。回归分析通常用于预测分析时间序列模型以及发现变量之间的因果关系。目前回归分析的研究范围如图 5-1 所示。

回归分析有许多分类方式，根据因变量和自变量的个数可分为一元回归分析、多元

图 5-1　回归分析的研究范围

回归分析、逻辑回归分析和其他回归分析；根据因变量和自变量的函数表达式可分为线性回归分析和非线性回归分析。线性回归是回归分析中最基本的方法。对于非线性回归，可以借助数学手段将其转化为线性回归，一旦线性回归问题得到解决，非线性回归问题也就迎刃而解。常用的回归分析技术有线性回归、逻辑回归、多项式回归和岭回归等。

5.1.2　回归分析的过程

可以将回归分析简单地理解为数据分析与预测，通过对数据进行分析实现预测，也就是适当扩大已有自变量的取值范围，并承认该回归方程在扩大的定义域内成立。一般来说，回归分析的主要过程如下。

(1) 收集一组包含因变量和自变量的数据；

(2) 根据因变量和自变量之间的关系，初步设定回归模型；

(3) 求解合理的回归系数；

(4) 进行相关性检验，确定相关系数；

(5) 利用模型对因变量作出预测或解释，并计算预测值的置信区间。

5.2　一元线性回归分析

视频讲解

5.2.1　一元线性回归方法

一元线性回归方法是根据自变量 x 和因变量 y 的相关关系，建立 x 与 y 的线性回归方程进行预测的方法。由于市场现象一般是受多种因素的影响，而并不是仅受一个因素

的影响，所以应用一元线性回归方法，必须对影响市场现象的多种因素进行全面分析。只有当诸多的影响因素中确实存在一个对因变量影响作用明显高于其他因素的变量，才能将它作为自变量，应用一元回归分析进行预测。

设 x 和 y 为两个变量，因变量 y 受自变量 x 的影响。将 y 和 x 间的关系表示为

$$y=f(x,\theta)+\varepsilon \tag{5.1}$$

式(5.1)称为一元回归模型。其中，f 为满足一定条件的函数，称为回归函数；θ 为参数，称为回归模型参数；ε 为随机变量，称为误差项或扰动项，它反映了除 x 和 y 之间的线性关系之外的随机因素对 y 的影响，而且，ε 是不能由 x 和 y 之间的线性关系所解释的变异性。

在简单的回归模型中，回归函数是解释变量的线性函数，回归模型则称为一元线性回归模型，表达式为

$$y=\beta_0+\beta_1 x+\varepsilon \tag{5.2}$$

其中，β_0 和 β_1 为回归系数，β_0 为常数项，也称为截距，β_1 为斜率；随机误差 ε 满足期望 $E(\varepsilon)=0$，方差 $D(\varepsilon)=\sigma^2$。

回归模型的设定给出了回归函数的形式，但模型中的回归参数是未知的。要对模型参数进行估计和统计推断，需要从总体样本中抽样获得数据。设从总体中抽取 n 个样本 (x_i,y_i)，$i=1,2,\cdots n$，将回归模型应用于每个样本，得到

$$y_i=\beta_0+\beta_1 x_i+\varepsilon_i,\quad i=1,2,\cdots n \tag{5.3}$$

式(5.3)称为样本回归模型。其中，来自同一总体的不同样本，其回归模型具有不同的误差项 ε_i。

【例 5-1】 分析预测房屋面积(平方英尺)和房价(美元)之间的对应关系。数据如下：

$$y=[6450,\ 7450,\ 8450,\ 94501,\ 11450,\ 15450,\ 18450]$$
$$x=[150,\ 200,\ 250,\ 300,\ 350,\ 400,\ 600]$$

```
In[163]:    import pandas as pd
            import matplotlib.pyplot as plt
            %matplotlib inline
            plt.rcParams['font.sans-serif'] = ['SimHei']
            plt.rcParams['font.size'] = 13
            y = [6450, 7450, 8450, 9450, 11450, 15450, 18450]
            x = [150, 200, 250, 300, 350, 400, 600]
            plt.scatter(x,y)
            plt.xlabel('面积(平方英尺)')
            plt.ylabel('售价(美元)')
            plt.show()
```

输出结果如图 5-2 所示。

如果散点图的趋势大概呈现线性关系，可以建立线性方程；如果不呈线性分布，可以建立其他回归模型。从图 5-2 可以看出，房屋面积和房价之间存在明显的线性关系。获得样本后，要对回归模型进行参数估计和统计推断。

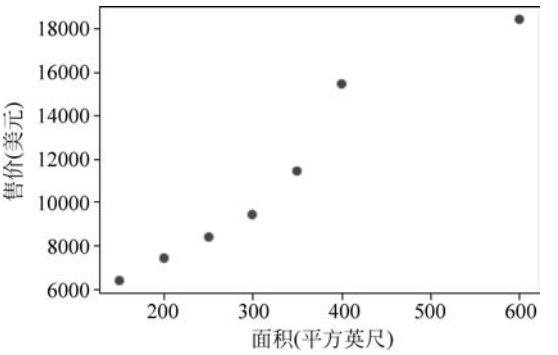

图 5-2 房屋面积与房价散点图

5.2.2 一元线性回归模型的参数估计

一元线性回归模型的参数估计方法有最小二乘法、矩方法和极大似然法，这里仅介绍最小二乘法。最小二乘法(Least Squre Estimation，LSE)又称为最小平方法，它通过最小化误差的平方和寻找数据的最佳函数匹配。

普通最小二乘法是最直观的估计方法，对模型条件要求最少，也就是使散点图上所有的观测值到回归直线距离平方和最小。对任意给定的自变量 x_i，其相应的估计值为 $\hat{y}_i=\hat{\beta}_0+\hat{\beta}_1 x_i$，$i=1,2,\cdots,n$。利用最小二乘法所得的参数估计值 $\hat{\beta}_0$ 和 $\hat{\beta}_1$，将使因变量的观察值 y_i 和估计值 $\hat{y}_i$ 之间的残差平方和 $Q(\beta_0,\beta_1)$ 最小。

残差平方和(Residual Sum of Squares，RSS)函数定义如下，$\Delta y=y_i-\hat{y}_i$ 为残差。

$$Q(\beta_0,\beta_1)=\sum_{i=1}^{n}(y_i-\hat{y}_i)=\sum_{i=1}^{n}(y_i-\hat{\beta}_0-\hat{\beta}_1 x_i)^2 \tag{5.4}$$

最小平方法可提供自变量与因变量关系的最佳近似直线，在计算参数的估计值时，根据微积分求极值原理，通过对 Q 求偏导并置为 0 得到

$$\begin{cases}\dfrac{\partial Q}{\partial \hat{\beta}_0}=-2\sum\limits_{i=1}^{n}(y_i-\hat{\beta}_0-\hat{\beta}_1 x_i)=0\\[2ex] \dfrac{\partial Q}{\partial \hat{\beta}_1}=-2\sum\limits_{i=1}^{n}(y_i-\hat{\beta}_0-\hat{\beta}_1 x_i)x_i=0\end{cases} \tag{5.5}$$

求解方程组得到

$$\begin{cases}\hat{\beta}_0=\bar{y}-\hat{\beta}_1\bar{x}\\[2ex] \hat{\beta}_1=\dfrac{\sum\limits_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum\limits_{i=1}^{n}(x_i-\bar{x})^2}\end{cases} \tag{5.6}$$

其中，$\bar{x}=\dfrac{1}{n}\sum\limits_{i=1}^{n}x_i$，$\bar{y}=\dfrac{1}{n}\sum\limits_{i=1}^{n}y_i$。将求得的 $\hat{\beta}_0$ 和 $\hat{\beta}_1$ 代入方程，即可得到最佳拟合曲线。

5.2.3 一元线性回归模型的误差方差估计

在线性回归方程中，误差项 ε_i 表示因变量 y_i 中不能解释由 x_i 表达的部分，其随机大小代表了 y_i 的随机性。因此，误差项方差 σ^2 十分重要。

残差平方和(误差平方和)为

$$SS_{残}=\sum_{i=1}^{n}e_i^2=\sum_{i=1}^{n}(y_i-\hat{y}_i)^2=\sum_{i=1}^{n}y_i^2-n\bar{y}^2-\hat{\beta}_1S_{xy} \tag{5.7}$$

其中，$\hat{y}_i=\hat{\beta}_0+\hat{\beta}_1x_i$，$\sum_{i=1}^{n}\hat{y}_ie_i=0$，$S_{xy}=\sum_{i=1}^{n}y_i(x_i-\bar{x})$。

由于 $\sum_{i=1}^{n}y_i^2-n\bar{y}^2=\sum_{i=1}^{n}(y_i-\bar{y})^2=SS_{总}$，因此有

$$SS_{残}=SS_{总}-\hat{\beta}_1S_{xy} \tag{5.8}$$

其中，$SS_{总}$ 为响应变量观测值的校正平方和。残差平方和有 $n-2$ 个自由度，因为两个自由度与得到 $\hat{y}_i$ 的估计值 $\hat{\beta}_0$ 与 $\hat{\beta}_1$ 相关。

最后得到 σ^2 的无偏估计量为

$$\sigma^2=\frac{SS_{残}}{n-2}=MS_{残} \tag{5.9}$$

其中，$MS_{残}$ 为残差均方。σ^2 的平方根称为回归标准误差，与响应变量 y 具有相同的单位。

5.2.4 一元回归模型的主要统计检验

回归分析要通过样本所估计的参数代替总体的真实参数，或者说用样本回归线代替总体回归线。尽管从统计性质上已知，如果有足够多的重复抽样，参数的估计值的期望就等于总体的参数真值，但在一次抽样中，估计值不一定就等于真值。那么在一次抽样中，参数的估计值与真值的差异有多大，是否显著，就需要进一步进行统计检验。

一元回归的统计检验主要包括拟合优度检验、变量显著性检验和残差标准差检验。

1. 拟合优度检验

拟合优度检验是用卡方统计量进行统计显著性检验的重要内容之一。它是依据总体分布状况，计算出分类变量中各类别的期望频数，与分布的观察频数进行对比，判断期望频数与观察频数是否有显著差异，从而达到从分类变量进行分析的目的。它是对样本回归直线与样本观测值之间拟合程度的检验。

2. 变量显著性检验(t 检验)

显著性检验就是事先对总体(随机变量)的参数或总体分布形式做出一个假设，然后利用样本信息判断这个假设(备择假设)是否合理，即判断总体的真实情况与原假设是否

有显著性差异。显著性检验是针对我们对总体所做的假设进行检验，其原理就是用“小概率事件实际不可能性原理”接受或否定假设。

5.2.5 一元线性回归的 Python 实现

对鸢尾花数据集中的 petal-length 和 petal-width 两列数据进行回归分析。

首先，导入相关包和数据。

```
In[164]:  import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          from sklearn.datasets import load_iris
          from sklearn.linear_model import LinearRegression
          %matplotlib inline
          iris = load_iris()    #导入数据集
          data = pd.DataFrame(iris.data)
          data.columns = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width']
          data.head()           #显示前5行
```

输出数据如图 5-3 所示。

	sepal-length	sepal-width	petal-length	petal-width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

图 5-3 数据集前 5 行

对数据集中的 petal-length 和 petal-width 列进行回归分析。

```
In[165]:  # 使用 scikit-learn 完成一元线性回归
          x = data['petal-length'].values
          y = data['petal-width'].values
          x = x.reshape(len(x),1)
          y = y.reshape(len(y),1)
          clf = LinearRegression()
          clf.fit(x,y)
          pre = clf.predict(x)
          plt.scatter(x,y,s=50)
          plt.plot(x,pre,'r-',linewidth=2)
          plt.xlabel('petal-length')
          plt.ylabel('petal-width')
          for idx, m in enumerate(x):
              plt.plot([m,m],[y[idx],pre[idx]], 'g-')
          plt.show()
```

输出结果如图 5-4 所示。

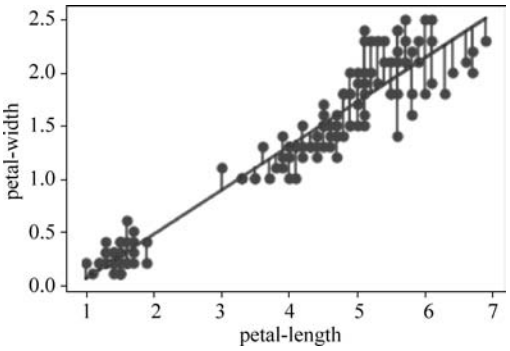

图 5-4　线性回归分析

显示回归线的参数。

```
In[166]:    print(u"系数", clf.coef_  )
            print(u"截距", clf.intercept_  )
            print(np.mean(y - pre) ** 2  )
Out[166]:   系数 [[0.41641913]]
            截距 [ - 0.36651405]
            4.996338194428969e - 32
```

对花萼长度为 3.9 的花，预测其花萼宽度。

```
In[167]:    print(clf.predict([[3.9]]) )
Out[167]:   [[1.25752057]]
```

视频讲解

5.3　多元线性回归

在实际经济问题中，一个变量往往受到多个变量的影响。例如，家庭消费支出除了受家庭可支配收入的影响外，还受诸如家庭所有的财富、物价水平、金融机构存款利息等多种因素的影响。也就是说，一个因变量和多个自变量有依存关系，而且有时几个影响因素的主次难以区分，或者有的因素虽属次要，但也不能忽略。这时采用一元回归分析进行预测难以奏效，需要多元回归分析。

5.3.1　多元线性回归模型

多元回归分析是指通过对两个或两个以上的自变量与一个因变量的相关分析，建立预测模型进行预测的方法。当自变量与因变量之间存在线性关系时称为多元线性回归分析。

假定因变量 y 与 k 个解释变量 $x_1, x_2, \cdots, x_k$ 之间具有线性关系，是解释变量的多元

线性函数，即

$$y=\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_k x_k+\mu \tag{5.10}$$

其中，偏回归系数(Partial Regression Coeficient)$\beta_i(i=1,2,\cdots,k)$为 k 个未知参数，β_0 为常数项；μ 为随机误差项。

对于 n 个观测值，其方程组形式为

$$y_i=\beta_0+\beta_1 x_{1i}+\beta_2 x_{2i}+\cdots+\beta_k x_{ki}+\mu_i \tag{5.11}$$

多元线性回归模型含有多个解释变量，这些解释变量同时对被 y 发生作用，若要考查其中一个解释变量对 y 的影响，就要假设其他解释变量保持不变。因此，多元线性模型中的回归系数 $\beta_i(i=1,2,\cdots,k)$称为偏回归系数，表示在其他自变量保持不变时，x_i 增加或减少一个单位时 y 的平均变化量。

偏回归系数 $\beta_i(i=0,1,2,\cdots,k)$都是未知的，可以利用样本观测值($x_{1i}$，$x_{2i}$，$\cdots$，$x_{ki}$，$y_i$)进行估计。如果计算得到的参数估计值为 $\hat{\beta}_0,\hat{\beta}_1,\cdots,\hat{\beta}_k$，用参数估计值代替总体回归方程中的位置参数 $\beta_0,\beta_1,\cdots,\beta_k$，则多元线性回归方程为

$$\hat{Y}_i=\hat{\beta}_0+\hat{\beta}_1 x_{1i}+\cdots+\hat{\beta}_k x_{ki} \tag{5.12}$$

其中，$\hat{\beta}_i(i=1,2,\cdots,k)$为参数估计值；$\hat{Y}_i(i=1,2,\cdots,k)$为 y_i 的回归值或拟合值。由样本回归方程得到的估计值 $\hat{Y}_i$ 与观测值 Y_i 之间的偏差称为残差 e_i，计算式为

$$e_i=Y_i-\hat{Y}_i=Y_i-(\hat{\beta}_0+\hat{\beta}_1 x_{1i}+,\cdots,\hat{\beta}_k x_{ki}) \tag{5.13}$$

建立多元线性回归模型时，为了保证回归模型具有优良的解释能力和预测效果，应首先注意自变量的选择，其准则如下。

(1) 自变量对因变量必须有显著的影响，并呈密切的线性相关；

(2) 自变量与因变量之间的线性相关必须是真实的，而不是形式上的；

(3) 自变量之间应具有一定的互斥性，即自变量之间的相关程度不应高于自变量与因变量之间的相关程度；

(4) 自变量应具有完整的统计数据，其预测值容易确定。

5.3.2　多元线性回归模型的参数估计

多元线性回归模型的参数估计与一元线性回归相同，也是在要求误差平方和最小的前提下，用最小二乘法求解参数。

以二元线性回归模型为例，求解回归参数的标准方程组为

$$\begin{cases}\sum y=nb_0+b_1\sum x_1+b_2\sum x_2\\ \sum x_1 y=b_0\sum x_1+b_1\sum x_1^2+b_2\sum x_1x_2\\ \sum x_2 y=b_0\sum x_2+b_1\sum x_1x_2+b_2\sum x_2^2\end{cases} \tag{5.14}$$

解此方程可求得 b_0，b_1，b_2 的数值(表示为矩阵形式)如下。

$$\boldsymbol{b}=(\boldsymbol{x}'\boldsymbol{x})^{-1}\cdot(\boldsymbol{x}'\boldsymbol{y}) \tag{5.15}$$

5.3.3 多元线性回归的假设检验及其评价

(1) 将回归方程中所有自变量作为一个整体来检验它们与因变量之间是否具有线性关系(方差分析法、复相关系数);

(2) 对回归方程的预测或解释能力做出综合评价(决定系数);

(3) 在此基础上进一步对各个变量的重要性做出评价(偏回归平方和、t 检验和标准回归系数)。

5.3.4 多元线性回归的 Python 实现

本节利用 scikit-learn 自带的波士顿房价(Boston House Price)数据集。该数据集源于一份美国某经济学杂志上的分析研究,数据集中的每行数据都是对波士顿周边或城镇房价的描述,如表 5-1 所示。

表 5-1 波士顿房价数据集各字段及其含义

字 段 名	含 义
CRIM	城镇人均犯罪率
INDUS	城镇中非住宅用地所占比例
NOX	环保指数
AGE	1940 年以前建成的自住单位的比例
RAD	距离高速公路的便利指数
PRTATIO	城镇中的教师学生比例
LSTAT	地区中有多少房东属于低收入人群
ZN	住宅用地所占比例
CHAS	虚拟变量,用于回归分析
RM	每栋住宅的房间数
DIS	距离 5 个波士顿的就业中心的加权距离
TAX	万美元的不动产税率
B	城镇中的黑人比例
MEDV	自住房屋房价中位数(也就是均价)

首先导入数据集。

```
In[168]:    import numpy as np
            import pandas as pd
            import matplotlib.pyplot as plt
            from sklearn import datasets
            d = datasets.load_boston()
            data = pd.DataFrame(d.data)
            data['price'] = d.target
            data.sample(5)
```

输出数据如图 5-5 所示。

	0	1	2	3	4	5	6	7	8	9	10	11	12	class
459	6.80117	0.0	18.10	0.0	0.713	6.081	84.4	2.7175	24.0	666.0	20.2	396.90	14.70	20.0
40	0.03359	75.0	2.95	0.0	0.428	7.024	15.8	5.4011	3.0	252.0	18.3	395.62	1.98	34.9
192	0.08664	45.0	3.44	0.0	0.437	7.178	26.3	6.4798	5.0	398.0	15.2	390.49	2.87	36.4
185	0.06047	0.0	2.46	0.0	0.488	6.153	68.8	3.2797	3.0	193.0	17.8	387.11	13.15	29.6
429	9.33889	0.0	18.10	0.0	0.679	6.380	95.6	1.9682	24.0	666.0	20.2	60.72	24.08	9.5

图 5-5 波士顿房价数据集

然后进行多元线性回归建模。

```
In[169]:  from sklearn.linear_model import LinearRegression
          #引入多元线性回归算法模块进行相应的训练
          simple2 = LinearRegression()
          from sklearn.model_selection import train_test_split
          x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 666)
          simple2.fit(x_train,y_train)
          print('多元线性回归模型系数: \n',simple2.coef_)
          print('多元线性回归模型常数项: ',simple2.intercept_)
          y_predict = simple2.predict(x_test)
Out[169]: 多元线性回归模型系数:
           [ -1.14235739e-01    3.12783163e-02   -4.30926281e-02
             -9.16425531e-02   -1.09940036e+01    3.49155727e+00
             -1.40778005e-02   -1.06270960e+00    2.45307516e-01
             -1.23179738e-02   -8.80618320e-01
              8.43243544e-03   -3.99667727e-01]
          多元线性回归模型常数项: 32.64566083965332
```

最后进行模型分析。

```
In[170]:  from sklearn.metrics import mean_absolute_error
          from sklearn.metrics import mean_squared_error
          from sklearn.metrics import r2_score
          #直接调用库函数输出 R2
          print('预测值的均方误差: ',
          mean_squared_error(y_test,y_predict))
          print(r2_score(y_test,y_predict))
          print(simple2.score(x_test,y_test))
          print('各特征间的系数矩阵: \n',simple2.coef_)
          print('影响房价的特征排序: \n',np.argsort(simple2.coef_))
          print('影响房价的特征排序: \n',
          d.feature_names[np.argsort(simple2.coef_)])
Out[170]: 预测值的均方误差: 13.012127852260955
          0.8008916199519095
          0.8008916199519095
          各特征间的系数矩阵:
           [ -1.14235739e-01    3.12783163e-02  -4.30926281e-02  -9.16425531e-02
```

```
  -1.09940036e+01   3.49155727e+00  -1.40778005e-02  -1.06270960e+00
   2.45307516e-01  -1.23179738e-02  -8.80618320e-01   8.43243544e-03
  -3.99667727e-01]
影响房价的特征排序:
 [ 4  7 10 12  0  3  2  6  9 11  1  8  5]
影响房价的特征排序:
 ['NOX' 'DIS' 'PTRATIO' 'LSTAT' 'CRIM' 'CHAS' 'INDUS' 'AGE' 'TAX' 'B' 'ZN' 'RAD' 'RM']
```

视频讲解

5.4 逻辑回归

线性回归算法能对连续值的结果进行预测，而逻辑回归模型是机器学习从统计领域借鉴的另一种技术，用于分析二分类或有序的因变量与解释变量之间的关系。逻辑回归算法是一种广义的线性回归分析方法，它仅在线性回归算法的基础上，利用 Sigmoid 函数对事件发生的概率进行预测。也就是说，在线性回归中可以得到一个预测值，然后将该值通过逻辑函数进行转换，将预测值转换为概率值，再根据概率值实现分类。逻辑回归常用于数据挖掘、疾病自动诊断和经济预测等领域。

5.4.1 逻辑回归模型

逻辑回归与线性回归类似，因为二者的目标都是找出每个输入变量的权重值。与线性回归不同的是，输出的预测值需要使用逻辑函数的非线性函数进行变换。逻辑函数即 Sigmoid 函数，能将任意值转换为 0～1 的范围内。Sigmoid 函数定义如下。

$$g(z)=\frac{1}{1+\mathrm{e}^{-z}} \tag{5.16}$$

绘制 Sigmoid 函数图像，代码如下，输出如图 5-6 所示。

```
In[171]:    import matplotlib.pyplot as plt
            import numpy as np
            def sigmoid(x):
                return 1./(1. + np.exp( - x))
            x = np.arange( - 8, 8, 0.2)
            y = sigmoid(x)
            plt.plot(x,y)
            plt.xlabel('$x$',fontsize = 13)
            plt.ylabel('$y$',fontsize = 13,rotation = 0)
            plt.title('Sigmoid')
            plt.show()
```

将 Sigmoid 函数应用到逻辑回归算法中，形式为

$$z=\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}=\theta_0+\theta_1x_1+\cdots+\theta_nx_n=\sum_{i=0}^{n}\theta_ix_i \tag{5.17}$$

结合式(5.16)和式(5.17)可得

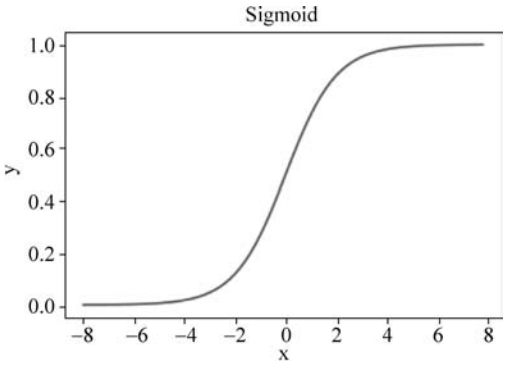

图 5-6 Sigmoid 函数

$$h_\theta(x)=g(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x})=\frac{1}{1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}}} \tag{5.18}$$

其中，$\boldsymbol{\theta}$ 为模型函数；$h_\theta(x)$的输出表示 $y=1$ 的概率。

由于模型特有的学习方式，通过逻辑回归所做的预测也可以用于计算属于类 0 或类 1 的概率。这对于需要给出许多基本原理的问题十分有用。

5.4.2 逻辑回归的 Python 实现

首先导入相关包和数据。

```
In[172]:    from sklearn.datasets import load_iris
            X = load_iris().data
            y = load_iris().target
            print('前 8 条数据:\n',X[:8])
            print('前 8 条数据对应的类型: ',y[:8])
Out[172]:   前 8 条数据:
             [[5.1 3.5 1.4 0.2]
             [4.9 3.  1.4 0.2]
             [4.7 3.2 1.3 0.2]
             [4.6 3.1 1.5 0.2]
             [5.  3.6 1.4 0.2]
             [5.4 3.9 1.7 0.4]
             [4.6 3.4 1.4 0.3]
             [5.  3.4 1.5 0.2]]
            前 8 条数据对应的类型: [0 0 0 0 0 0 0 0]
```

划分训练集和测试集并进行归一化。

```
In[173]:    from sklearn.model_selection import train_test_split
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
            random_state = 0)
            from sklearn.preprocessing import StandardScaler
            sc = StandardScaler()
```

```
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train[:5])
Out[173]: [[  0.01543995  -0.11925475   0.22512685   0.35579762]
 [-0.09984503  -1.04039491   0.11355956  -0.02984109]
 [  1.05300481  -0.11925475   0.95031423   1.12707506]
 [-1.36797986   0.34131533  -1.39259884  -1.31530348]
 [  1.1682898   0.11103029   0.72717965   1.38416753]]
```

训练逻辑回归模型并对测试集进行预测。

```
In[174]:  from sklearn.linear_model import LogisticRegression
          classifier = LogisticRegression(random_state = 0)
          classifier.fit(X_train, y_train)
          y_pred = classifier.predict(X_test)
          #用 LogisticRegression 自带的 score()方法获得模型在测试集上的准确性
          print('Accuracy of LR Classifier: %.3f' % classifier.score(X_test,y_test))
Out[174]: Accuracy of LR Classifier:0.816
```

视频讲解

5.5 其他回归分析

5.5.1 多项式回归

1. 多项式回归原理

线性回归的局限性是只能应用于存在线性关系的数据中，但是在实际生活中，很多数据之间是非线性关系，虽然也可以用线性回归拟合非线性回归，但是效果会变差，这时就需要对线性回归模型进行改进，使之能够拟合非线性数据。多项式回归模型是线性回归模型的一种，此时回归函数关于回归系数是线性的。由于任何函数都可以用多项式逼近，因此多项式回归有着广泛应用。

研究一个因变量与一个或多个自变量间多项式的回归分析方法，称为多项式回归(Polynomial Regression)。自变量只有一个时，称为一元多项式回归；自变量有多个时，称为多元多项式回归。在一元回归分析中，如果因变量 y 与自变量 x 的关系为非线性的，但又找不到适当的函数曲线来拟合，则可以采用一元多项式回归。在这种回归技术中，最佳拟合线不是直线，而是一个用于拟合数据点的曲线。

多项式回归的最大优点是可以通过增加 x 的高次项对观测点进行逼近，直到满意为止。多项式回归在回归分析中占有重要地位，因为任意函数都可以分段用多项式逼近。

2. 多项式回归的 Python 实现

1）准备数据

```
In[175]:  import numpy as np
          import matplotlib.pyplot as plt
          x = np.random.uniform(-3,3, size=100)   # 产生 100 个随机数
          X = x.reshape(-1,1)                     #将 x 变成矩阵,一行一列的形式
          y = 0.5 * x**2 +x +2 + np.random.normal(0,1,size=100)
          #数据中引入噪声
          plt.scatter(x,y)
          plt.show()
```

输出数据如图 5-7 所示。

2）线性回归

```
In[176]:  from sklearn.linear_model import LinearRegression
          #线性回归
          lin_reg = LinearRegression()
          lin_reg.fit(X,y)
          y_predict = lin_reg.predict(X)
          plt.rcParams['font.family'] = ['SimHei']
          plt.rcParams['axes.unicode_minus'] = False
          plt.title('线性回归')
          plt.scatter(x,y)
          plt.plot(x,y_predict,color='r')
          plt.show()
```

输出结果如图 5-8 所示。

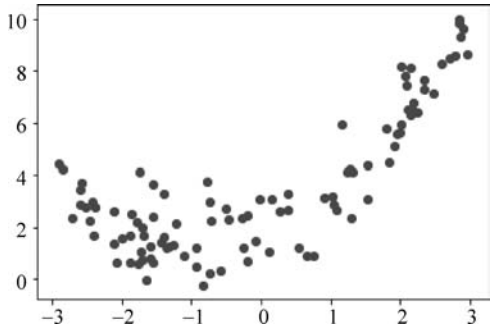

图 5-7 准备数据

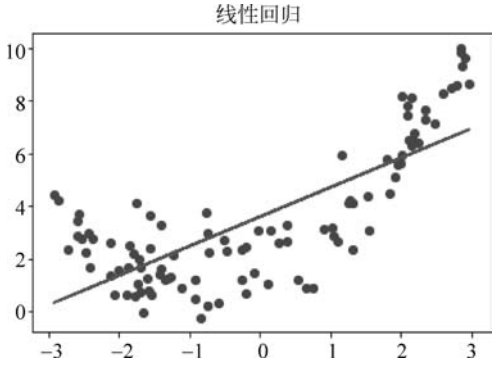

图 5-8 线性回归

3）多项式回归

```
In[177]:  from sklearn.preprocessing import PolynomialFeatures
          poly = PolynomialFeatures(degree=2)
```

```
#设置最多添加几次幂的特征项
poly.fit(X)
x2 = poly.transform(X)
from sklearn.linear_model import LinearRegression
#接下来的代码和线性回归一致
lin_reg2 = LinearRegression()
lin_reg2.fit(x2,y)
y_predict2 = lin_reg2.predict(x2)
plt.scatter(x,y)
plt.plot(np.sort(x),y_predict2[np.argsort(x)],color='r')
plt.title('多项式回归')
```

输出结果如图 5-9 所示。

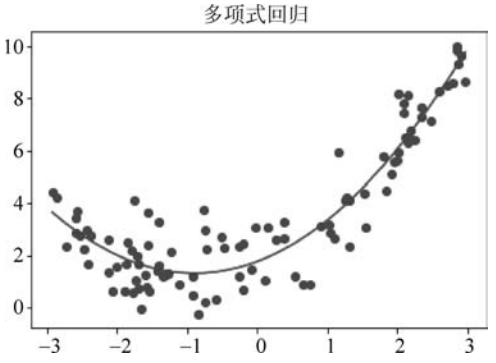

图 5-9　多项式回归

5.5.2　岭回归

1. 岭回归原理

岭回归(Ridge Regression)是一种专用于共线性数据分析的有偏估计回归方法,实质上是一种改良的最小二乘估计法,通过放弃最小二乘法的无偏性,以损失部分信息、降低精度为代价,获得回归系数更符合实际、更可靠的回归方法,对病态数据的耐受性远强于最小二乘法。

岭回归主要适用于过拟合严重或各变量之间存在多重共线性的情况,它可以解决特征数量比样本量多的问题。另外,岭回归作为一种缩减算法,可以判断哪些特征重要或不重要,有点类似于降维,缩减算法可以看作是对一个模型增加偏差的同时减少方差。岭回归方程的 R^2(回归平方和与总离差平方和的比值)会稍低于普通回归分析,但回归系数的显著性往往明显高于普通回归分析,在存在共线性问题和病态数据偏多的研究中有较大的实用价值。

2. 岭回归的 Python 实现

```
In[178]:    import numpy as np
            import matplotlib.pyplot as plt
```

```
from sklearn.linear_model import Ridge,RidgeCV
#岭回归,RidgeCV 带有广义交叉验证的岭回归
data = [
    [0.07,3.12],[0.41,3.82],[0.99,4.55],[0.73,4.25],[0.98,4.56],
    [0.55,3.92],[0.34,3.53],[0.03,3.15],[0.13,3.11],[0.13,3.15],
    [0.31,3.47],[0.65,4.12],[0.73,4.28],[0.23,3.48],[0.96,4.65],
    [0.62,3.95],[0.36,3.51],[0.15,3.12],[0.63,4.09],[0.23,3.46],
    [0.08,3.22],[0.06,3.19],[0.92,4.63],[0.71,4.29],[0.01,3.08],
    [0.34,3.45],[0.04,3.16],[0.21,3.36],[0.61,3.99],[0.54,3.89] ]
#生成 X 和 y 矩阵
dataMat = np.array(data)
X = dataMat[:,0:1]                              # 变量 X
y = dataMat[:,1]                                # 变量 y
# 岭回归
model = Ridge(alpha = 0.5)
model = RidgeCV(alphas = [0.1, 1.0, 10.0])
# RidgeCV 可以设置多个参数,算法使用交叉验证获取最佳参数值
model.fit(X, y)                                 # 线性回归建模
print('系数矩阵:',model.coef_)
print('线性回归模型:\n',model)
# print('交叉验证最佳 alpha 值',model.alpha_)
# 只有在使用 RidgeCV 算法时才有效
# 使用模型预测
predicted = model.predict(X)
# 绘制散点图,参数: X 横轴,y 纵轴
plt.scatter(X, y, marker = 'o')
plt.plot(X, predicted,c = 'r')
# 绘制 x 轴和 y 轴坐标
plt.xlabel('x')
plt.ylabel('y')
# 显示图形
plt.show()
```

```
Out[178]:  系数矩阵: [1.59032686]
           线性回归模型:
            RidgeCV(alphas = array([ 0.1,   1. , 10. ]), cv = None,
           fit_intercept = True,gcv_mode = None, normalize = False, scoring = None,
           store_cv_values = False)
```

输出结果如图 5-10 所示。

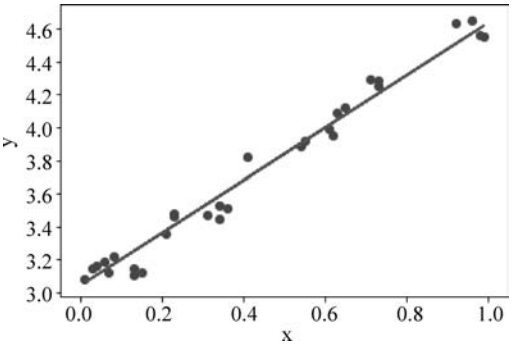

图 5-10　岭回归

5.5.3 Lasso回归

1. Lasso回归原理

岭回归无法剔除变量，而Lasso(Least Absolute Shrinkage and Selection Operator)回归模型将惩罚项由L2范数变为L1范数，可以将一些不重要的回归系数缩减为0，达到剔除变量的目的。

2. Lasso回归的Python实现

```
In[179]: import numpy as np
         import matplotlib.pyplot as plt
         from sklearn.metrics import r2_score
         #def main():
         # 产生一些稀疏数据
         np.random.seed(42)
         n_samples, n_features = 50, 100
         X = np.random.randn(n_samples, n_features)
         # randn()函数产生的是正态分布的数据
         coef = 3 * np.random.randn(n_features)
         # 每个特征对应一个系数
         inds = np.arange(n_features)
         np.random.shuffle(inds)
         coef[inds[10:]] = 0
         # 稀疏化系数,随机地把系数向量其中10个值变为0
         y = np.dot(X, coef)
         # 添加噪声：零均值,标准差为0.01的高斯噪声
         y += 0.01 * np.random.normal(size = n_samples)
         # 把数据划分成训练集和测试集
         n_samples = X.shape[0]
         X_train, y_train = X[:n_samples // 2], y[:n_samples // 2]
         X_test, y_test = X[n_samples // 2:], y[n_samples // 2:]
         # 训练Lasso模型
         from sklearn.linear_model import Lasso
         alpha = 0.1
         lasso = Lasso(alpha = alpha)
         y_pred_lasso = lasso.fit(X_train, y_train).predict(X_test)
         r2_score_lasso = r2_score(y_test, y_pred_lasso)
         print("r^2 on test data : %f" % r2_score_lasso)
         plt.plot(lasso.coef_, color = 'gold', linewidth = 2,label = 'Lasso coefficients')
         plt.title("Lasso R^2: %f" % r2_score_lasso)
         plt.show()
Out[179]: r^2 on test data : 0.518282
```

输出结果如图5-11所示。

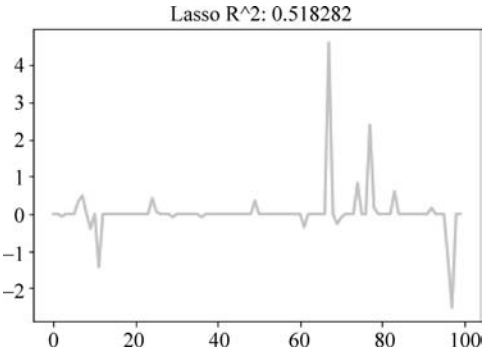

图 5-11　Lasso 回归

5.5.4　逐步回归

在处理多个自变量时，需要使用逐步回归(Stepwise Regression)。逐步回归中，自变量的选择是在一个自动的过程中完成的，其中包括非人为操作。

逐步回归是通过观察统计的值，如 R^2 等指标，来识别重要的变量，并通过同时添加/删除基于指定标准的协变量来拟合模型。常用的逐步回归方法如下。

(1) 标准逐步回归法：该方法做两件事情，即增加和删除每个步骤所需的预测。

(2) 向前选择法：从模型中最显著的预测开始，然后为每一步添加变量。

(3) 向后剔除法：与模型的所有预测同时开始，然后在每一步消除最小显著性的变量。

逐步回归的目的是使用最少的预测变量最大化预测能力，是处理高维数据集的方法之一。

5.6　小结

(1) 回归分析是广泛应用的统计学分析方法，通过建立因变量 y 和影响它的自变量 $x_i(i=1,2,3,\cdots)$ 间的回归模型，衡量自变量 x_i 对因变量的影响能力，进而用来预测因变量的发展趋势。回归分析模型包括线性回归和非线性回归两种。线性回归又分为简单线性回归和多重线性回归。非线性回归需要通过对数转化等方式，将其转化为线性回归的形式进行研究。

(2) 一元线性回归分析是根据自变量 x 和因变量 y 的相关关系，建立 x 与 y 的线性回归方程进行预测的方法。由于市场现象一般受多种因素的影响，而并不是仅受一个因素影响，所以应用一元线性回归分析方法，必须对影响市场现象的多种因素进行全面分析。回归方程是否可靠，估计的误差有多大，都还应经过显著性检验和误差计算。

(3) 包括两个或两个以上自变量的回归称为多元线性回归。多元线性回归的基本原理和基本计算过程与一元线性回归相同。

(4) 逻辑回归是一种广义的线性回归分析模型，常用于数据挖掘、疾病自动诊断、经

济预测等领域。回归的因变量可以是二分类的，也可以是多分类的，但是二分类更常用，也更加容易解释，多分类可以使用 Softmax 方法进行处理。实际中最常用的就是二分类的逻辑回归。

(5) 多项式回归模型是线性回归模型的一种，回归函数是回归变量多项式；岭回归是一种专用于共线性数据分析的有偏估计回归方法，实质上是一种改良的最小二乘估计法，通过放弃最小二乘法的无偏性，以损失部分信息、降低精度为代价获得回归系数，对病态数据的拟合效果要强于最小二乘法；Lasso 回归是一种压缩估计，它通过构造一个惩罚函数得到一个比较精炼的模型，使得压缩一些回归系数，即强制系数绝对值之和小于某个固定值，同时设定一些回归系数为零，因此保留了子集收缩的优点，是一种处理具有复共线性数据的有偏估计；逐步回归的基本思想是将变量逐个引入模型，每引入一个解释变量后都要进行 F 检验，并对已经选入的解释变量逐个进行 t 检验，当原来引入的解释变量由于后面解释变量的引入变得不再显著时，则将其删除，以确保每次引入新的变量之前回归方程中只包含显著性变量。

习题 5

(1) 简述回归分析的含义及常用的回归分析方法。

(2) 简述逻辑回归的含义及主要过程。

(3) 下面是 7 个地区 2000 年的人均国内生产总值(Gross Domestic Product，GDP)与人均消费水平的统计数据。

地　　区	人均 GDP/元	人均消费水平/元
北京	22460	7326
辽宁	11226	4490
上海	34547	11546
江西	4851	2396
河南	5444	2208
贵州	2662	1608
陕西	4549	2035

试求：

① 以人均 GDP 作为自变量，人均消费水平作为因变量，绘制散点图，并说明二者之间的关系；

② 计算两个变量之间的线性相关系数，说明两个变量之间的关系强度；

③ 求出估计的回归方程，并解释回归系数的实际意义；

④ 计算判定系数，并解释其意义；

⑤ 检验回归方程线性关系的显著性($\alpha=0.05$)；

⑥ 如果某地区的人均 GDP 为 5000 元，预测其人均消费水平。