第 4 章 线性回归

4.1 监督学习的回归问题

假设“训练数据”(training data)为$\left\{\mathbf{x}_i, y_i\right\}_{i=1}^n$，其中$\left\{\mathbf{x}_i, y_i\right\}$为第$i$位个体(individual)或观测值(observation)的数据，也称为“样例”(example)或“示例”(instance)，而n为样本容量。

y_i为“响应变量”(response variable)，而$\mathbf{x}_i$为p维“特征向量”(feature vector)：

$$\mathbf{x}_i = \left(x_{i1}\ x_{i2} \cdots x_{ip}\right)' \qquad (4.1)$$

个体i共有p个特征(features)，即$\left(x_{i1}\ x_{i2} \cdots x_{ip}\right)$；而$x_{ik}$表示个体$i$的第$k$个特征($k = 1, \cdots, p$)。

对于监督学习，其基本问题为使用特征向量$\mathbf{x}_i$预测响应变量y_i。

如果响应变量y_i为连续变量，则称为“回归问题”(regression problem)；而如果y_i为离散变量，则称为“分类问题”(classification problem)，本章关注回归问题。

我们试图通过训练数据$\left\{\mathbf{x}_i, y_i\right\}_{i=1}^n$，来学得一个函数$f\left(\mathbf{x}_i\right)$，并以$f\left(\mathbf{x}_i\right)$预测$y_i$。这种预测通常不可能完全准确。对于观测数据的生成过程(Data Generating Process，简记 DGP)，可作很一般的假设：

$$y_i = f\left(\mathbf{x}_i\right) + \varepsilon_i \qquad (4.2)$$

其中，ε_i为“随机扰动项”(stochastic disturbance)或“误差项”(error term)。

一般将 $f(\mathbf{x}_i)$ 视为“信号”(signal)，即来自 $\mathbf{x}_i$ 对于 y_i 的“系统信息”(systematic information)；而把 ε_i 看作随机扰动的“噪音”(noise)。

不失一般性，可对方程(4.2)的扰动项 ε_i 作如下两个假设。

假设 4. 1　扰动项 ε_i 的期望为 0，即 $\mathrm{E}(\varepsilon_i)=0$。

假设 4. 2　扰动项 ε_i 的条件期望为 0，即 $\mathrm{E}(\varepsilon_i \mid \mathbf{x}_i)=0$。

对于假设 4.1，如果 $\mathrm{E}(\varepsilon_i)=c\neq 0$，则总可以把常数 c 归入函数 $f(\mathbf{x}_i)$ 中，故这其实是一个免费的假设。

对于假设 4.2，如果$\mathrm{E}(\varepsilon_i \mid \mathbf{x}_i) = g(\mathbf{x}_i) \neq 0$，则总可以把$g(\mathbf{x}_i)$归入函数$f(\mathbf{x}_i)$中，故这也是一个免费的假设。

事实上，根据迭代期望定律(Law of Iterated Expectation，参见第 3 章)，假设 4.2 意味着假设 4.1：

$$\mathrm{E}(\varepsilon_i) = \mathrm{E}_{\mathbf{x}_i}\underbrace{\left[\mathrm{E}(\varepsilon_i \mid \mathbf{x}_i)\right]}_{=0} = \mathrm{E}_{\mathbf{x}_i}\left[0\right] = 0 \tag{4.3}$$

根据假设 4.2，$\mathrm{E}(\varepsilon_i \mid \mathbf{x}_i) = 0$，这意味着扰动项$\varepsilon_i$均值独立于$\mathbf{x}_i$(mean independence)，故扰动项$\varepsilon_i$与特征向量$\mathbf{x}_i$不相关。直观上，如果$\varepsilon_i$与$\mathbf{x}_i$相关，总可以把$\varepsilon_i$中与$\mathbf{x}_i$相关的部分并入函数$f(\mathbf{x}_i)$，从而使得重新定义之后的$\varepsilon_i$与$\mathbf{x}_i$不相关。

如何学得 $f(\mathbf{x}_i)$？大致可分为两种方法，即“非参数方法”(nonparametric approach)与“参数方法” (parametric approach)。

如果不对 $f(\mathbf{x}_i)$的函数形式(functional form)作任何假设，则为非参数方法；既然不假设函数形式，则自然没有待估计的参数。非参数方法包括K近邻法、决策树、以及基于决策树的装袋法、随机森林与提升法等。

参数方法则对 $f(\mathbf{x}_i)$的函数形式作了具体的假设。模型一般可写为

$$y_i = f(\mathbf{x}_i;\boldsymbol{\beta}) + \varepsilon_i \qquad (4.4)$$

其中，$\boldsymbol{\beta}$为待估计的未知参数向量。函数 $f(\mathbf{x}_i;\boldsymbol{\beta})$的具体形式可以是非常简单的线性函数，也可是复杂的神经网络模型。

4.2 最优预测

如果使用$\mathbf{x}_i$预测y_i，是否存在一个最优的预测函数$g(\mathbf{x}_i)$？

这取决于最优预测的定义。对于回归问题，若使用$g(\mathbf{x}_i)$预测连续的响应变量y_i，则一般使用“均方误差”(mean squared error，简记 MSE)作为对预测优良程度的度量：

$$\text{MSE} = \text{E}\left[y - g(\mathbf{x})\right]^2 \tag{4.5}$$

其中，期望算子$\text{E}(\cdot)$为同时对$(\mathbf{x}, y)$求期望。

下面将证明此最优预测函数为$\text{E}(y \mid \mathbf{x})$，也就是上文的$f(\mathbf{x})$。

命题 4.1 能使均方误差最小化的函数为条件期望函数$\mathrm{E}(y\,|\,\mathbf{x})$。

证明：考虑以下最小化问题

$$\min_{g(\cdot)}\ \mathrm{MSE}=\mathrm{E}(y-g(\mathbf{x}))^2 \tag{4.6}$$

根据迭代期望定律可知：

$$\mathrm{MSE}=\mathrm{E}(y-g(\mathbf{x}))^2=\mathrm{E}_{\mathbf{x}}\,\mathrm{E}_{y}\left[(y-g(\mathbf{x}))^2\Big|\mathbf{x}\right] \tag{4.7}$$

这意味着，只需考虑给定$\mathbf{x}$情况下的最小化问题：

$$\min_{g(\mathbf{x})}\ \mathrm{E}\left[(y-g(\mathbf{x}))^2\Big|\mathbf{x}\right] \tag{4.8}$$

为简化符号，在推导中省略“$\cdot|\mathbf{x}$”，但一直视$\mathbf{x}$为给定。MSE 可分解为

$$
\begin{aligned}
&\text{MSE} \\
&= \mathrm{E}(y - g(\mathbf{x}))^2 \\
&= \mathrm{E}\left[y - \mathrm{E}(y|\mathbf{x}) + \mathrm{E}(y|\mathbf{x}) - g(\mathbf{x})\right]^2 \\
&= \underbrace{\mathrm{E}\left[y - \mathrm{E}(y|\mathbf{x})\right]^2}_{\text{与}g(\mathbf{x})\text{无关}} + \left[\mathrm{E}(y|\mathbf{x}) - g(\mathbf{x})\right]^2 + 2\underbrace{\mathrm{E}\left\{[y - \mathrm{E}(y|\mathbf{x})][\mathrm{E}(y|\mathbf{x}) - g(\mathbf{x})]\right\}}_{=0}
\end{aligned}
\tag{4.9}
$$

其中，第 1 项$\mathrm{E}\left[y - \mathrm{E}(y|\mathbf{x})\right]^2$与$g(\mathbf{x})$无关，而第 3 项等于 0：

$$
\begin{aligned}
&\mathrm{E}\left\{[y-\mathrm{E}(y\,|\,\mathbf{x})][\mathrm{E}(y\,|\,\mathbf{x})-g(\mathbf{x})]\right\} \\
&=[\mathrm{E}(y\,|\,\mathbf{x})-g(\mathbf{x})]\cdot \mathrm{E}[y-\mathrm{E}(y\,|\,\mathbf{x})] \\
&=[\mathrm{E}(y\,|\,\mathbf{x})-g(\mathbf{x})]\cdot [\mathrm{E}(y\,|\,\mathbf{x})-\mathrm{E}(y\,|\,\mathbf{x})] \qquad (4.10) \\
&=[\mathrm{E}(y\,|\,\mathbf{x})-g(\mathbf{x})]\cdot 0 \\
&=0
\end{aligned}
$$

其中，在给定$\mathbf{x}$的情况下，$[\mathrm{E}(y\,|\,\mathbf{x})-g(\mathbf{x})]$可视为常数提出。因此，只须最小化第 2 项即可。为了最小化第 2 项$\left[\mathrm{E}(y\,|\,\mathbf{x})-g(\mathbf{x})\right]^2$，必须让$g(\mathbf{x})=\mathrm{E}(y\,|\,\mathbf{x})$。 ■

进一步，给定$\mathbf{x}_i$，对方程(4.2)两边同时求条件期望可得：

$$\mathrm{E}(y_i \mid \mathbf{x}_i) = f(\mathbf{x}_i) + \underbrace{\mathrm{E}(\varepsilon_i \mid \mathbf{x}_i)}_{=0} = f(\mathbf{x}_i) \qquad (4.11)$$

其中，根据假设 4.2，$\mathrm{E}(\varepsilon_i \mid \mathbf{x}_i) = 0$。

若最小化均方误差，则$f(\mathbf{x}_i) = \mathrm{E}(y_i \mid \mathbf{x}_i)$是对$y_i$的最优预测函数。此结果依赖于所用的“平方损失函数”(squared loss function)。

若使用绝对误差的损失函数(absolute loss function)$\mathrm{E}\left|y-g(\mathbf{x})\right|$，则最优预测函数为条件中位数$median(y_i \mid \mathbf{x}_i)$。

在大多数情况下，对于回归问题，我们都使用平方损失函数。

进一步的问题是，应如何估计条件期望函数$\mathrm{E}(y_i \mid \mathbf{x}_i)$。

4.3 线性回归模型

如果条件期望函数$\mathrm{E}(y_i \mid \mathbf{x}_i)$为$\mathbf{x}_i$的线性函数，则很容易估计。最简单的参数方法假设$f(\mathbf{x}_i;\boldsymbol{\beta})$为如下线性回归模型，也称为“古典线性回归模型”(Classical Linear Regression Model，简记 CLRM)：

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i \quad (i = 1, \cdots, n) \tag{4.12}$$

其中，$\boldsymbol{\beta} = (\beta_1\ \beta_2\ \cdots\ \beta_p)'$为待估计的未知参数，称为“回归系数”(regression coefficients)。

如果方程(4.12)中有常数项(即截距项)，则通常令第 1 个变量恒等于 1，即$x_{i1} \equiv 1,\ \forall i$。

为何使用线性模型？难道世界不是非线性的(nonlinear)吗？总结起来，线性模型有以下优点。

第一，线性模型十分简单，且易于解释(interpretable)。在线性模型的假设下，每个变量x_{ik}对于响应变量y_i的边际效应(marginal effect)均为常数。比如，将方程(4.12)对x_{ik}求偏导数可得：

$$\frac{\partial y_i}{\partial x_{ik}} = \beta_k \tag{4.13}$$

当变量x_{ik}增加 1 单位时，将导致(如果存在因果关系)或伴随着(如果仅存在相关关系)响应变量y_i平均增加β_k单位。

第二，即使 $f(\mathbf{x}_i;\boldsymbol{\beta})$ 为非线性函数，在足够小的局部，一般也可用线性函数来近似。方程(4.12)可视为对函数 $f(\mathbf{x}_i;\boldsymbol{\beta})$ 的一阶泰勒展开(Taylor expansion)。

线性方程依然允许加入高次项(比如 x_{i2}^2)、交互项(比如 $x_{i2}x_{i3}$)或对变量进行非线性变换(比如 $\ln x_{i3}$)，例如

$$y_i = \beta_1 + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i2}^2 + \beta_5 x_{i2}x_{i3} + \beta_6 \ln x_{i3} + \varepsilon_i \quad (4.14)$$

将 x_{i2}^2、$x_{i2}x_{i3}$ 与 $\ln x_{i3}$ 都视为变量，则依然是线性回归模型，因为该方程是参数 $\boldsymbol{\beta}$ 的线性函数(linear in parameters)。

第三，线性模型虽然简单，但可作为复杂模型的组成部分。因此，从线性模型入手，有助于理解机器学习的思想与方法。

第四，有些现象本身就是近似于线性的。下面以两个经典研究为例。

例 Engel(1857)对于食物开支占家庭收入比重(恩格尔系数)的研究。原始数据可从 statsmodels 模块获得。该模块是在 Python 中进行统计分析的主要模块。首先导入 statsmodels 模块，以及用于画图的 seaborn 模块：

```
In [1]: import statsmodels.api as sm
   ...: import statsmodels.formula.api as smf
   ...: import seaborn as sns
```

导入了 statsmodels 的两个不同"应用程序接口"(Application Programming Interface，简记 API)，并分别记为 `sm` 与 `smf`。

其中，第 1 个接口(`statsmodels.api`，简记 `sm`)基于数组(array-based)，而第 2 个接口(`statsmodels.formula.api`，简记 `smf`)则基于公式(formula-based)。

然后，从 statsmodels 的 datasets 子模块载入 Engel(1857)的数据框：

```
In [2]: data = sm.datasets.engel.load_pandas().data
```

考察此数据框的形状与前 5 个观测值：

```
In [3]: data.shape
Out[3]: (235, 2)

In [4]: data.head()
Out[4]:
      income     foodexp
0  420.157651  255.839425
1  541.411707  310.958667
2  901.157457  485.680014
3  639.080229  402.997356
```

结果显示，此数据框包括两个变量，即 income(家庭年收入)与 foodexp(家庭年食物开支)，观测对象为 1857 年的 235 个比利时家庭。

下面以 foodexp 为响应变量，income 作为特征变量，使用 statsmodels 模块基于公式的 smf 接口，进行一元线性回归(即除常数项外，只有一个自变量的线性回归):

```
In [5]: model = smf.ols('foodexp ~ income', data=data)
```

statsmodels 模块也是基于“面向对象编程”(OOP)而设计的。

其中，“ols”是一个 OLS 回归估计量(即最小二乘法)的“类”(class)。其第 1 个参数“'foodexp ~ income'”即所谓“公式”(formula)，表示将 foodexp 对 income 进行回归(默认包含常数项)

参数“data = data”表示使用数据框 data。

此命令的作用是，实例化(instantiate)`ols` 类，得到 `ols` 类的一个实例(instance)，且记为“`model`”。

针对此实例 `model`，可使用其 `fit()`方法进行 OLS 估计，并将输出结果赋值给 `results`：

```
In [6]: results = model.fit()
```

使用回归结果 `results` 的 `params` 属性，即可得到参数估计值：

```
In [7]: results.params
Out[7]:
Intercept    147.475389
```

```
income        0.485178
dtype: float64
```

此线性模型的估计结果为以下“拟合线”(fitted line)：

$$\widehat{foodexp} = 147.475 + 0.485\, income \qquad (4.15)$$

其中，$\widehat{foodexp}$为响应变量 `foodexp` 的拟合值(fitted value)，即预测值(predicted value)。

家庭收入的边际食物支出倾向为 0.4852，即当家庭收入增加 1 比利时法郎，将伴随食物开支平均增加 0.4852 比利时法郎。

使用 seaborn 模块，画 `income` 与 `foodexp` 的散点图，并在图上画出线性拟合方程(4.15)及其置信区间：

```
In [8]: sns.regplot(x='income', y='foodexp', data=data)
```

其中，“`regplot`”表示“regression plot”，结果参见图 4.1。

从图 4.1 可知，在 Engle 的数据集中，食物开支与家庭收入大致呈线性关系。

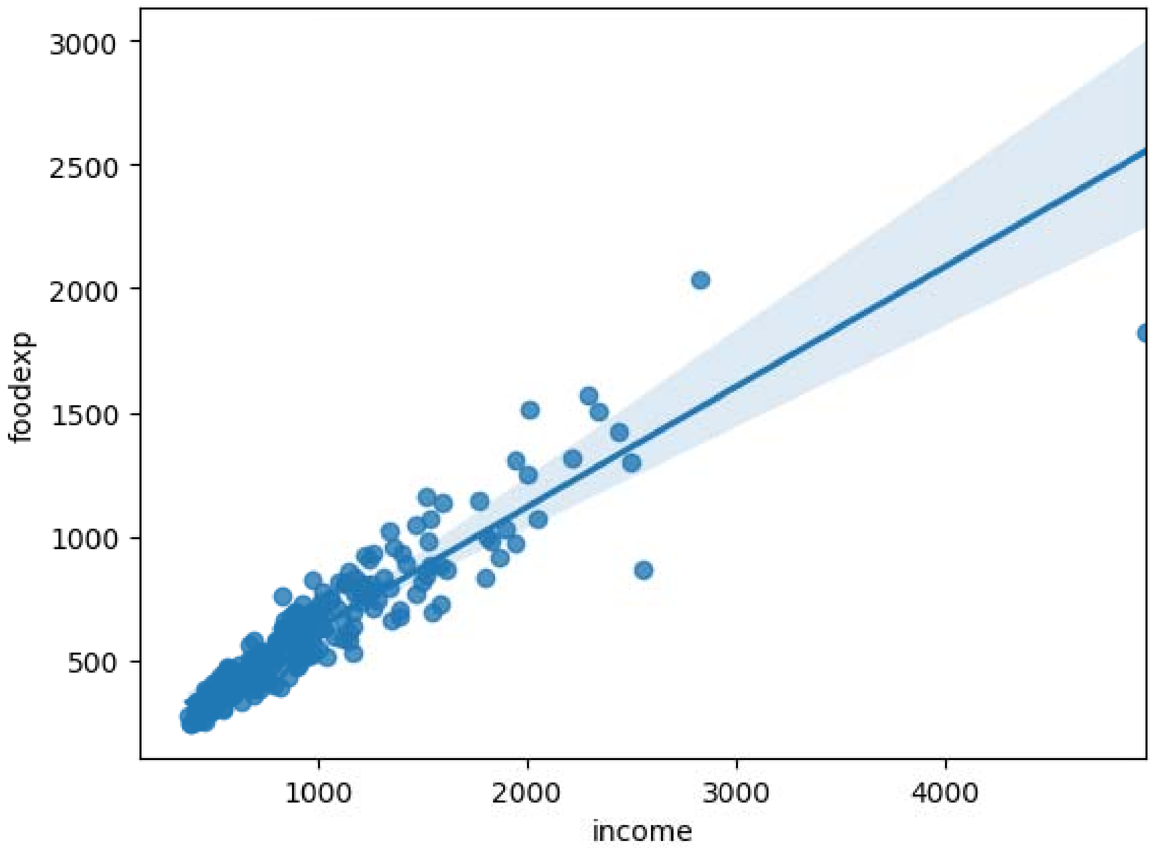


图 4.1 使用 Engle 的食品开支数据进行线性回归

例 Cobb-Douglas 生产函数(Cobb and Douglas,1928)是经济学常用的生产函数，其回归方程可写为

$$\ln y = \alpha + \beta \ln k + \gamma \ln l + \varepsilon \qquad (4.16)$$

其中，$\ln y$为产出对数，$\ln k$为资本对数，而$\ln l$为劳动力对数。

此对数线性模型(log-linear model)是否“靠谱”？

数据集 `cobb_douglas.csv` 提供了 Cobb and Douglas(1928)所用的原始数据。

首先，导入 pandas 模块，并使用其 `read_csv()`函数载入数据：

```
In [9]: import pandas as pd
   ...: data = pd.read_csv('cobb_douglas.csv')
```

考察该数据框的形状，以及前 5 个观测值：

```
In [10]: data.shape
Out[10]: (24, 7)

In [11]: data.head()
Out[11]:
   year   k   l   y      lnk      lnl      lny
```

```
0  1899  100  100  100  4.605170  4.605170  4.605170
1  1900  107  105  101  4.672829  4.653960  4.615120
2  1901  114  110  112  4.736198  4.700481  4.718499
3  1902  122  118  122  4.804021  4.770685  4.804021
4  1903  131  123  124  4.875197  4.812184  4.820282
```

结果显示，该数据为 1899-1922 年的年度时间序列，共 24 个观测值。其中，`y`，`k` 与 `l` 分别为美国制造业的产出、资本与劳动力指数(已将 1899 年取值标准化为 100)，而 `lny`，`lnk` 与 `lnl` 则为相应的对数。

其次，使用 statsmodels 的 `ols` 类进行线性回归：

```
In [12]: model = smf.ols('lny ~ lnk + lnl', data=data)
```

其中，公式“lny ~ lnk + lnl”表示把响应变量 lny 对特征变量 lnk 与 lnl 进行回归。

使用 fit()方法估计此模型，并将输出结果赋值给 results:

```
In [13]: results = model.fit()
```

通过 results 的 params 属性，考察回归系数的估计值：

```
In [14]: results.params
Out[14]:
Intercept   -0.177310
lnk          0.233054
```

```
lnl          0.807278
dtype: float64
```

结果显示，所得回归方程可写为

$$\widehat{\ln y} = -0.177 + 0.233 \ln k + 0.807 \ln l \qquad (4.17)$$

可通过 mpl_toolkits 模块的 mplot3d 子模块画三维拟合图。先导入 Numpy，Matplotlib 以及 mpl_toolkits 的 mplot3d 子模块：

```
In [15]: import numpy as np
    ...: import matplotlib.pyplot as plt
```

```
   ...: from mpl_toolkits import mplot3d
```

根据变量 `lnk` 的最小值与最大值，定义变量 `lnk` 的一个网格(100 个等分点)：

```
In [16]: xx = np.linspace(data.lnk.min(),
                          data.lnk.max(), 100)
```

其中，`data.lnk.min()`与 `data.lnk.max()`分别为变量 `lnk` 的最小值与最大值。

根据变量 `lnl` 的最小值与最大值，定义其 100 等分的网格：

```
In [17]: yy = np.linspace(data.lnl.min(),
                              data.lnl.max(), 100)
```

考察所生成的 xx 网格与 yy 网格之形状：

```
In [18]: xx.shape, yy.shape
Out[18]: ((100,), (100,))
```

结果显示，xx 与 yy 均为100×1的向量。

根据横轴的 xx 网格与纵轴的 yy 网格，使用 Numpy 的 `meshgrid()` 函数，生成一个二维的网格(包含 xx 与 yy 所有可能取值的坐标组合)：

```
In [19]: XX, YY = np.meshgrid(xx, yy)
```

其中，输出结果 XX 与 YY 分别为此二维网格的横坐标与纵坐标。

考察 XX 与 YY 的形状：

```
In [20]: XX.shape, YY.shape
Out[20]: ((100, 100), (100, 100))
```

结果显示，XX 与 YY 均为100×100的矩阵。

将(XX,YY)的每个坐标输入回归拟合方程(4.17)，即可得到相应的响应变量拟合值：

```
In [21]: ZZ = results.params[0] + XX * results.params[1] +  YY * results.params[2]
```

其中，results.params[0]为回归方程的常数项，而results.params[1]与 results.params[2]分别为 lnk 与 lnl 的回归系数。

最后，输入以下命令，画三维散点图与拟合回归平面：、

```
In [22]: fig = plt.figure()
    ...: ax = plt.axes(projection='3d')
    ...: ax.scatter3D(data.lnk, data.lnl, data.lny, c='b')
    ...: ax.plot_surface(XX, YY, ZZ, rstride=8, cstride=8, alpha=0.3, cmap='viridis')
    ...: ax.set_xlabel('lnk')
    ...: ax.set_ylabel('lnl')
    ...: ax.set_zlabel('lny')
```

其中，第 2 行命令“ax = plt.axes(projection='3d')”表示，在此 ax 画轴上画三维图。

第 3 行命令使用 mplot3d 的 `scatter3D()` 方法将样本数据画三维散点图，其中参数“`c='b'`”表示颜色为蓝色。

第 4 行命令使用 mplot3d 的 `plot_surface()` 方法画拟合的回归平面。

其中，参数“`rstride=8`”与“`cstride=8`”分别指定行(row)与列(column)方向的画图步幅(stride)为 8；

参数“`alpha=0.3`”控制平面的透明度，而“`cmap='viridis'`”表示使用“`viridis`”作为“色图”(color map)。

最后 3 行命令分别在 `x`，`y` 与 `z` 轴添加标签，结果参见图 4.2(可用鼠标调节视图的角度)。

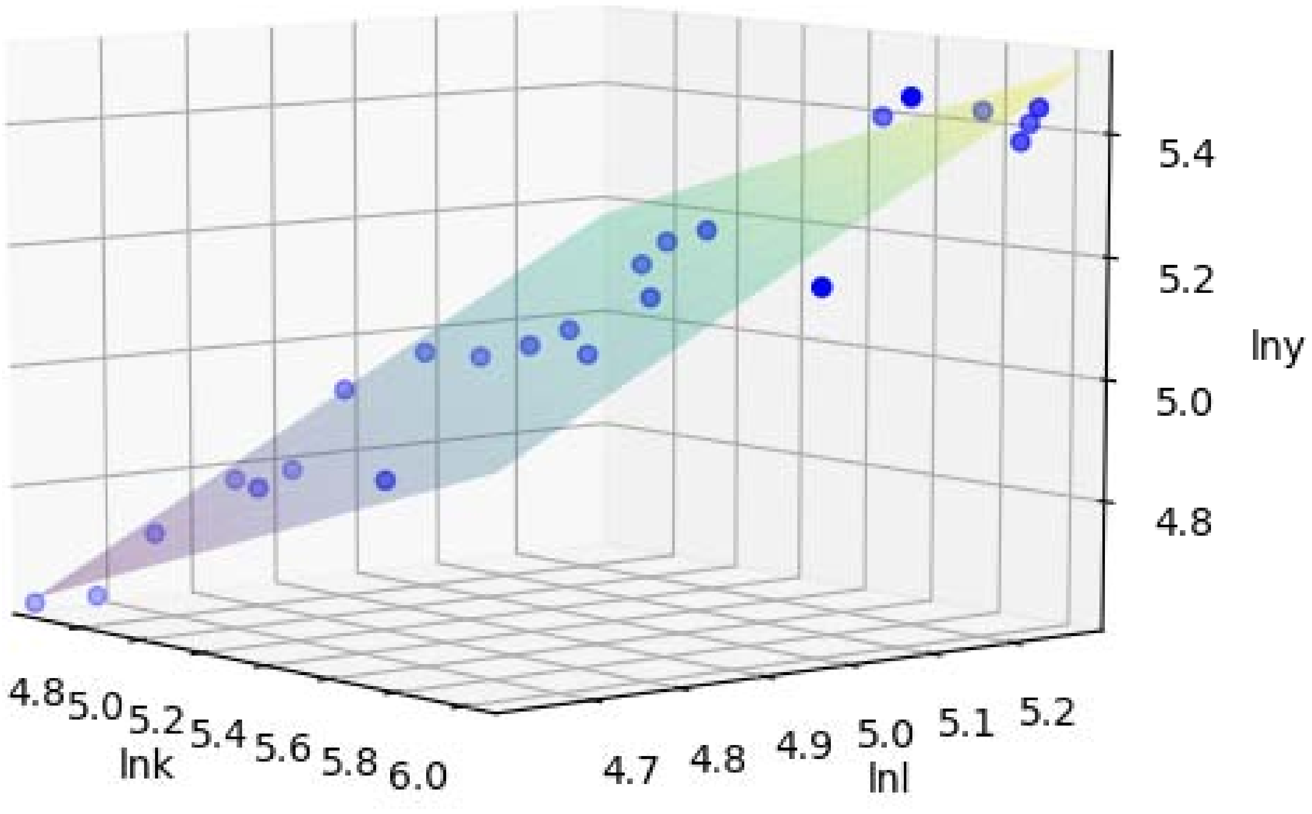


图 4.2 Cobb-Douglas 生产函数的拟合图

从图 4.2 可见，美国制造业产出、资本与劳动力的散点主要分布在拟合平面的附近，故其函数关系大致为线性。

由于线性模型最为简单，故本章将以线性回归为例，说明机器学习的一些重要原理，包括过拟合与泛化能力、偏差与方差的权衡、模型评估方法等。

4.4 最小二乘法

普通最小二乘法(Ordinary Least Square，简记 OLS)是估计线性回归模型的基本方法。

以一元回归为例，此时仅有 1 个特征变量x_i。

OLS 的任务为根据训练数据 $\left\{x_i, y_i\right\}_{i=1}^{n}$ 来估计回归方程 $f(x_i)=\alpha+\beta x_i$。

希望在(x,y)平面上找到一条最佳拟合直线(fitted line)，使得所有样本点$\left\{x_i, y_i\right\}_{i=1}^{n}$到此拟合线的距离最近，参见图 4.3。如何度量此距离？

在此平面上，任意给定一条直线，$\hat{\alpha}+\hat{\beta}x_i$，可以计算每个点(观测值)到这条线的距离(在$y$轴方向的垂直距离)，$e_i \equiv y_i-\hat{\alpha}-\hat{\beta}x_i$，称为“残差”(residual)。

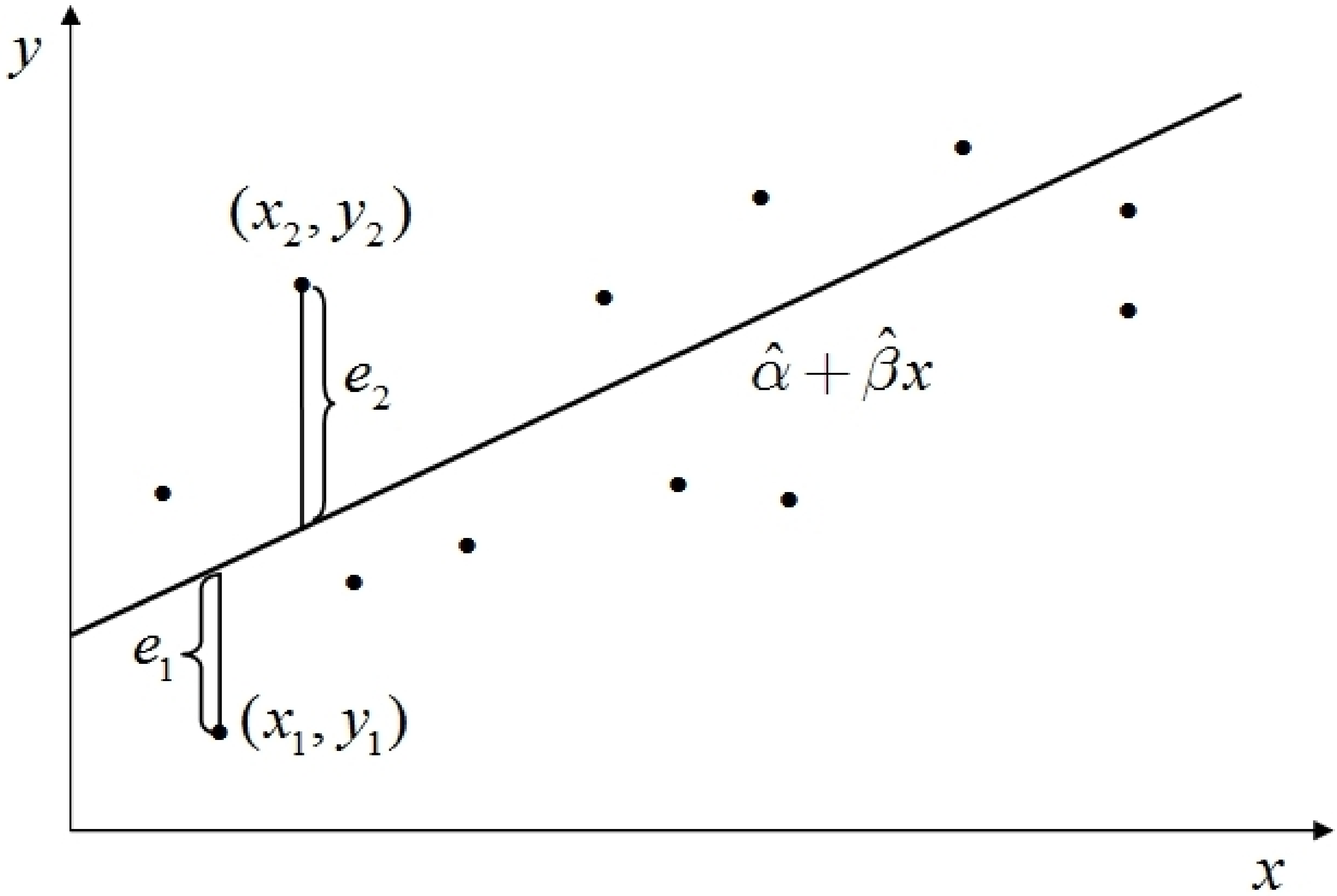


图 4.3 一元线性回归的示意图

如果直接把残差加起来，即$\sum_{i=1}^{n} e_i$，则会出现正负残差相抵的现象。

解决方法之一为使用绝对值，即$\sum_{i=1}^{n} |e_i| = \sum_{i=1}^{n} \left| y_i - \hat{\alpha} - \hat{\beta} x_i \right|$，但绝对值不易运算(比如无法微分)。

考虑残差的平方之和，$\sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{\alpha} - \hat{\beta} x_i)^2$，称为**残差平方和**(Sum of Squared Residuals，简记 SSR；或 Residual Sum of Squares，简记 RSS)。

最小二乘法就是选择$\hat{\alpha}, \hat{\beta}$，使得残差平方和最小化；故名“最小二乘”(二乘即平方)。在数学上，可将 OLS 的“目标函数”(objective function)，也称为“损失函数”(loss function)，写为

$$\min_{\hat{\alpha},\hat{\beta}} \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{\alpha} - \hat{\beta} x_i)^2 \qquad (4.18)$$

下面考虑更一般的多元回归情形。

为方便表达，引入向量与矩阵符号。

由于线性回归方程(4.12)的右边包含特征向量$\mathbf{x}_i$的线性组合，故可写为

$$y_i = \mathbf{x}_i'\boldsymbol{\beta} + \varepsilon_i \quad (i = 1, \cdots, n) \qquad (4.19)$$

其中，特征向量$\mathbf{x}_i \equiv (x_{i1}\ x_{i2}\ \cdots\ x_{ip})'$，而参数向量$\boldsymbol{\beta} \equiv (\beta_1\ \beta_2\ \cdots\ \beta_p)'$。

由于此方程对所有个体$i = 1, \cdots, n$均成立，故将这n个方程叠放(stack)在一起可得：

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} \mathbf{x}_1' \\ \mathbf{x}_2' \\ \vdots \\ \mathbf{x}_n' \end{pmatrix}}_{\mathbf{X}} \boldsymbol{\beta} + \underbrace{\begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}}_{\boldsymbol{\varepsilon}} \tag{4.20}$$

定义响应向量$\mathbf{y} \equiv (y_1\ y_2\ \cdots\ y_n)'$，数据矩阵$\mathbf{X} \equiv (\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_n)'$，残差向量$\boldsymbol{\varepsilon} \equiv (\varepsilon_1\ \varepsilon_2\ \cdots\ \varepsilon_n)'$，则有

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{4.21}$$

此方程简洁地概括了所有的训练数据$\left\{\mathbf{x}_i, y_i\right\}_{i=1}^{n}$。其中，

$\mathbf{X} \equiv (\mathbf{x}_1\ \mathbf{x}_2 \cdots \mathbf{x}_n)'$ 称为“数据矩阵”(data matrix)或“设计矩阵”(design matrix)，包含所有特征向量的信息：

$$\mathbf{X} \equiv \begin{pmatrix} \mathbf{x}_1' \\ \mathbf{x}_2' \\ \vdots \\ \mathbf{x}_n' \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} = \begin{pmatrix} 1 & x_{12} & \cdots & x_{1p} \\ 1 & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n2} & \cdots & x_{np} \end{pmatrix}$$

(4.22)

其中，设计矩阵的每行为个体观测值(observation)，而每列为变量(variable)，呈现出二维表的数据格式。设计矩阵的第 1 列元素通常都为 1，对应于回归方程的常数项。

为了估计未知参数向量$\boldsymbol{\beta}$，对于$\boldsymbol{\beta}$的一个任意假想值(hypothetical value) $\tilde{\boldsymbol{\beta}}$ (读作 beta tilde)，记第i个数据的拟合误差(即残差，residual)为$e_i = y_i - \mathbf{x}_i'\tilde{\boldsymbol{\beta}}$。

类比方程(4.21)，不难得到残差向量的表达式：

$$\mathbf{e} \equiv (e_1\ e_2 \cdots e_n)' = \mathbf{y} - \mathbf{X}\tilde{\boldsymbol{\beta}} \tag{4.23}$$

最小二乘法寻找能使残差平方和$\sum_{i=1}^{n} e_i^2$最小的$\tilde{\boldsymbol{\beta}}$。

从几何上来看，如果是一元回归，就是寻找最佳拟合的回归直线，使观测值y_i到该回归直线的距离之平方和最小，参见图 4.3。

如果是二元回归，则要寻找最佳拟合的回归平面，参见图 4.4。

如果是更多元的回归，则寻找最佳拟合的回归“超平面”(hyperplane)。

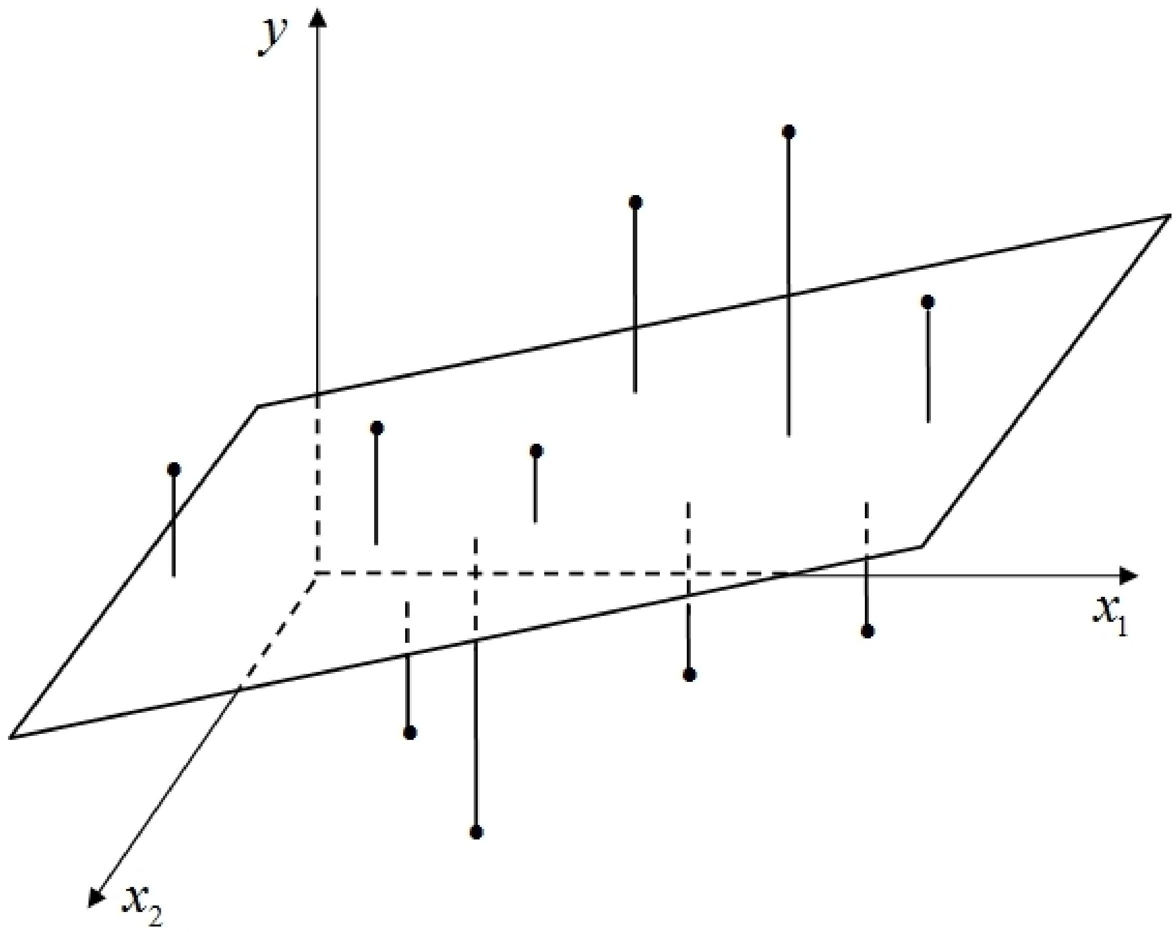


图 4.4 二元线性回归的示意图

在代数上，OLS 的最小化问题可写为

$$\min_{\tilde{\boldsymbol{\beta}}}\ \text{SSR}(\tilde{\boldsymbol{\beta}}) = \sum_{i=1}^{n} e_i^2 = \mathbf{e}'\mathbf{e} \qquad \text{(将平方和写成向量内积的形式)}$$

$$= (\mathbf{y} - \mathbf{X}\tilde{\boldsymbol{\beta}})'(\mathbf{y} - \mathbf{X}\tilde{\boldsymbol{\beta}}) \qquad \text{(代入残差向量的表达式)}$$

$$= (\mathbf{y}' - \tilde{\boldsymbol{\beta}}'\mathbf{X}')(\mathbf{y} - \mathbf{X}\tilde{\boldsymbol{\beta}}) \qquad \text{(矩阵转置的运算性质)}$$

$$= \mathbf{y}'\mathbf{y} - \mathbf{y}'\mathbf{X}\tilde{\boldsymbol{\beta}} - \tilde{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{y} + \tilde{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{X}\tilde{\boldsymbol{\beta}} \qquad \text{(乘积展开)}$$

$$= \mathbf{y}'\mathbf{y} - 2\mathbf{y}'\mathbf{X}\tilde{\boldsymbol{\beta}} + \tilde{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{X}\tilde{\boldsymbol{\beta}} \qquad \text{(合并同类项)}$$

(4.24)

在$\tilde{\boldsymbol{\beta}}$的参数空间，OLS 的损失函数$\mathrm{SSR}(\tilde{\boldsymbol{\beta}})$是$\tilde{\boldsymbol{\beta}}$的二次(型)函数，参见一维的示意图 4.5。

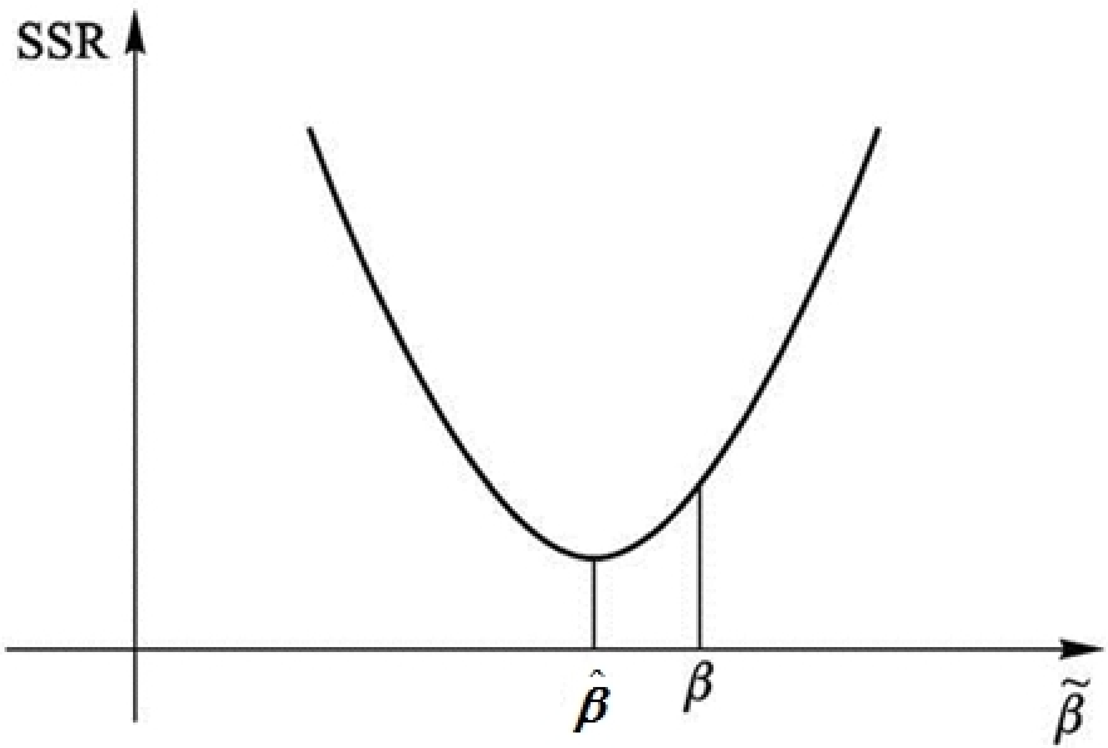


图 4.5　参数的假想值$\tilde{\boldsymbol{\beta}}$、真实值$\boldsymbol{\beta}$与 OLS 估计值$\hat{\boldsymbol{\beta}}$

使用向量微分的规则，将$\text{SSR}(\tilde{\boldsymbol{\beta}})$对向量$\tilde{\boldsymbol{\beta}}$求导，可得最小化问题的一阶条件(即梯度向量为 **0**)：

$$\frac{\partial(\text{SSR})}{\partial\tilde{\boldsymbol{\beta}}} = \frac{\partial\left(\mathbf{y}'\mathbf{y} - 2\mathbf{y}'\mathbf{X}\tilde{\boldsymbol{\beta}} + \tilde{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{X}\tilde{\boldsymbol{\beta}}\right)}{\partial\tilde{\boldsymbol{\beta}}} \tag{4.25}$$

$$= -2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}\tilde{\boldsymbol{\beta}} = \mathbf{0}$$

其中，由于$\mathbf{y}'\mathbf{y}$不含$\tilde{\boldsymbol{\beta}}$($\mathbf{y}'\mathbf{y}$相当于常数)，故$\dfrac{\partial(\mathbf{y}'\mathbf{y})}{\partial\tilde{\boldsymbol{\beta}}} = \mathbf{0}$。将上式移项可知，最小二乘估计量$\hat{\boldsymbol{\beta}}$须满足：

$$\underbrace{(\mathbf{X}'\mathbf{X})}_{p\times p}\underbrace{\hat{\boldsymbol{\beta}}}_{p\times 1} = \underbrace{\mathbf{X}'}_{p\times n}\underbrace{\mathbf{y}}_{n\times 1} \qquad (4.26)$$

表达式(4.26)为包含 p 个方程与 p 个未知数的线性方程组，称为**正规方程组**(normal equations)。

如果 $p\times p$ 维矩阵 $(\mathbf{X}'\mathbf{X})$ 可逆，可求解 OLS 估计量：

$$\hat{\boldsymbol{\beta}} \equiv (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \qquad (4.27)$$

什么时候 $(\mathbf{X}'\mathbf{X})$ 可逆呢？

这要求数据矩阵$\mathbf{X}$满列秩，$rank(\mathbf{X}) = p$，即矩阵$\mathbf{X}$没有多余的列向量。

换言之，不存在某个列向量可被其他列向量线性表出的情形(比如，一个变量是另一变量的两倍)。

如果数据矩阵 $\mathbf{X}$ 不满列秩，则称存在**严格多重共线性**(strict multicolinearity)。

因此，使用 OLS 的前提是不存在严格多重共线性。

命题 4. 2 如果数据矩阵$\mathbf{X}$满列秩，则$(\mathbf{X}'\mathbf{X})^{-1}$存在。

证明：根据第 3 章命题 3.3，我们知道矩阵$(\mathbf{X}'\mathbf{X})$半正定。由于正定矩阵一定可逆，故只需进一步证明$(\mathbf{X}'\mathbf{X})$正定即可。

记数据矩阵$\mathbf{X}$的第k列为$\mathbf{x}_{\bullet k}=(x_{1k}\ x_{2k}\cdots x_{nk})'$，对应于第$k$个特征变量，则$\mathbf{X}=(\mathbf{x}_{\bullet 1}\cdots\mathbf{x}_{\bullet p})$。需要注意$\mathbf{x}_{\bullet k}$与上文所定义的$\mathbf{x}_i$不同，其中$\mathbf{x}_i=(x_{i1}\ x_{i2}\cdots x_{ip})'$为$\mathbf{X}$的第$i$行。

由于$\mathbf{X}$满列秩，故$\left\{\mathbf{x}_{\bullet 1},\cdots,\mathbf{x}_{\bullet p}\right\}$线性无关。这意味着，对于任意$n$维列向量$\mathbf{c}\neq\mathbf{0}$，

$$\mathbf{Xc} = (\mathbf{x}_{\bullet 1} \cdots \mathbf{x}_{\bullet p}) \begin{pmatrix} c_1 \\ \vdots \\ c_p \end{pmatrix} = c_1 \mathbf{x}_{\bullet 1} + \cdots + c_p \mathbf{x}_{\bullet p} \neq \mathbf{0} \qquad (4.28)$$

因此，二次型$\mathbf{c'X'Xc} = (\mathbf{Xc})'\mathbf{Xc} > 0$，故$\mathbf{X'X}$为正定矩阵。 ■

OLS 一般不适用于“高维数据”(high-dimensional data)。

由于高维数据的变量个数大于样本容量(即 $p > n$)，故 $rank(\mathbf{X}) \le n < p$(矩阵$\mathbf{X}$的秩小于或等于其行数$n$)，故数据矩阵$\mathbf{X}$不满列秩。此时，$(\mathbf{X'X})^{-1}$不存在，故 OLS 没有唯一解。

对于高维回归，一般须进行“正则化”(regularization)处理，即在损失函数中加入“惩罚项”(penalty term)，进行“惩罚回归”(penalized regression)。

4.5 OLS 的正交性与几何解释

从正规方程组(4.26)出发，移项可得：

$$\mathbf{X}'\mathbf{y} - \mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{0} \qquad (4.29)$$

将矩阵$\mathbf{X}'$从左边提出可得：

$$\mathbf{X}'(\underbrace{\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}}_{=\mathbf{e}}) = \mathbf{X}'\mathbf{e} = \mathbf{0} \qquad (4.30)$$

其中，$\mathbf{e} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}$为残差向量。这意味着，残差向量$\mathbf{e}$与数据矩阵$\mathbf{X}$正交。更具体地，将上式展开写：

$$\mathbf{X}'\mathbf{e} = \underbrace{\begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_{12} & x_{22} & \cdots & x_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1p} & x_{2p} & \cdots & x_{np} \end{pmatrix}}_{\mathbf{X}'} \underbrace{\begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix}}_{\mathbf{e}} = \underbrace{\begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{\mathbf{0}} = \mathbf{0} \tag{4.31}$$

由于$\mathbf{X}'$的第k行就是$\mathbf{X}$的第k列$\mathbf{x}_{\bullet k} = (x_{1k}\ \ x_{2k} \cdots x_{nk})'$，故$\mathbf{X}'$的第$k$行包含第$k$个变量的全部观测值，因此残差向量$\mathbf{e}$与每个变量都正交。

这个性质称为 OLS 的**正交性**(orthogonality)。

定义响应变量y_i的拟合值(fitted value)或预测值(predicted value)为

$$\hat{y}_i \equiv \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \cdots + \hat{\beta}_p x_{ip} \quad (i = 1, \cdots, n) \tag{4.32}$$

将$\hat{\boldsymbol{\beta}}$代入模型“$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$”，并令$\boldsymbol{\varepsilon} = \mathbf{0}$，可得拟合值向量的表达式：

$$\hat{\mathbf{y}} \equiv (\hat{y}_1 \ \hat{y}_2 \cdots \hat{y}_n)' = \mathbf{X}\hat{\boldsymbol{\beta}} \tag{4.33}$$

容易证明，拟合值向量也与残差向量正交，因为

$$\hat{\mathbf{y}}'\mathbf{e} = (\mathbf{X}\hat{\boldsymbol{\beta}})'\mathbf{e} = \hat{\boldsymbol{\beta}}' \underbrace{\mathbf{X}'\mathbf{e}}_{=\mathbf{0}} = \hat{\boldsymbol{\beta}}' \cdot \mathbf{0} = 0 \tag{4.34}$$

由于$\mathbf{e}=\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}}=\mathbf{y}-\hat{\mathbf{y}}$，故

$$\mathbf{y}=\hat{\mathbf{y}}+\mathbf{e} \tag{4.35}$$

因此，响应变量$\mathbf{y}$可分解为相互正交的拟合值$\hat{\mathbf{y}}$与残差$\mathbf{e}$之和，参见图 4.6。

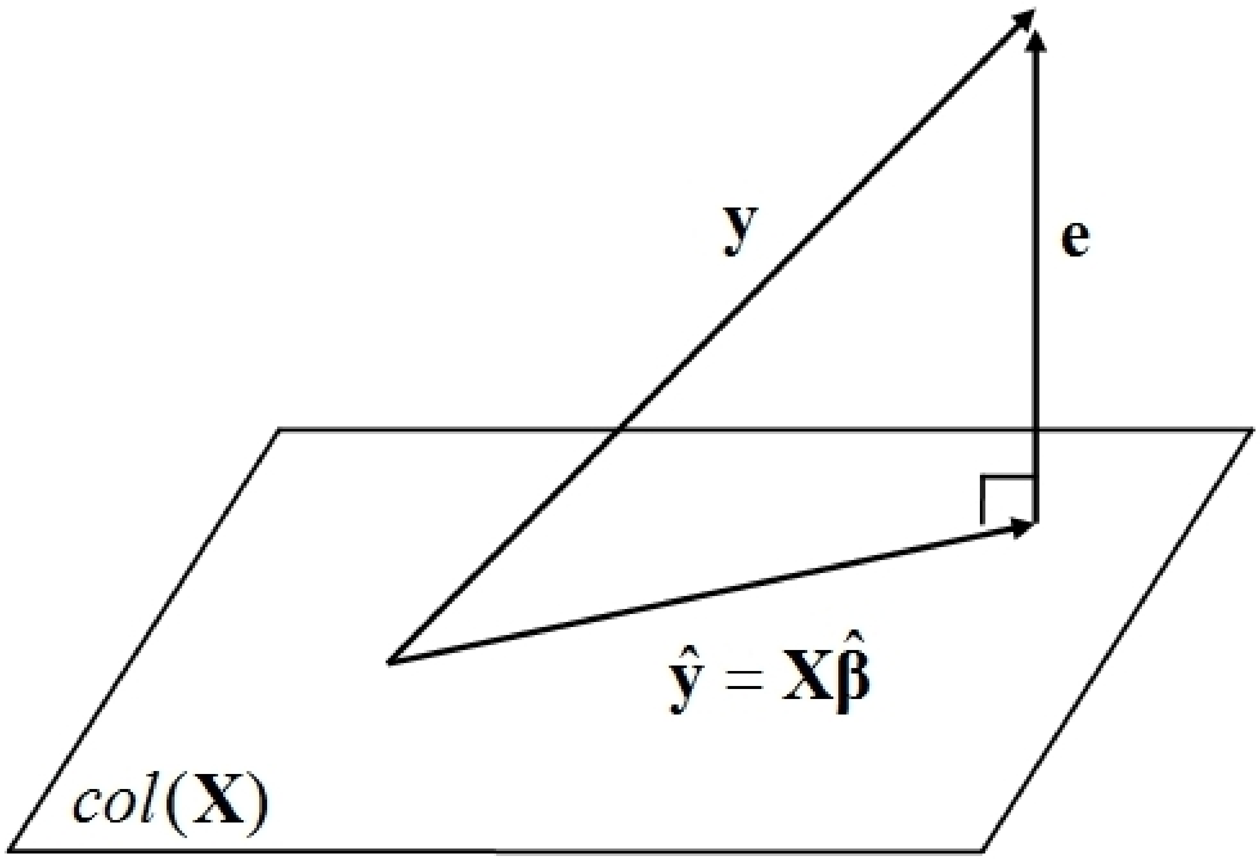


图 4.6 最小二乘法的正交性

在图 4.6 中，$\hat{\mathbf{y}}$ 可视为 $\mathbf{y}$ 向 $\mathbf{X}$ 的列空间(column space) $col(\mathbf{X})$ 这一超平面所作的“投影”(projection)，其中列空间 $col(\mathbf{X})$ 为矩阵 $\mathbf{X}$ 列向量的所有线性组合之集合：

$$\text{矩阵}\mathbf{X}\text{的列空间} \equiv col(\mathbf{X}) \equiv \left\{ c_1 \mathbf{x}_{\bullet 1} + c_2 \mathbf{x}_{\bullet 2} + \cdots c_p \mathbf{x}_{\bullet p}, \quad \forall c_1, c_2, \cdots, c_p \right\} \tag{4.36}$$

由于拟合值为各变量的线性组合，即 $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$，故拟合值向量 $\hat{\mathbf{y}}$ 正好在超平面 $col(\mathbf{X})$ 上。

而根据 OLS 的正交性，残差向量 $\mathbf{e}$ 与 $\hat{\mathbf{y}}$ 正交。

从 OLS 的正交性得到启发，也可以从几何角度推导 OLS，而不必通过向量微分。OLS 的损失函数可写为

$$\min_{\tilde{\boldsymbol{\beta}}} \ \text{SSR}(\tilde{\boldsymbol{\beta}}) = \sum_{i=1}^{n} e_i^2 = \mathbf{e}'\mathbf{e} = \|\mathbf{e}\|^2 \qquad (4.37)$$

我们的目标是最小化残差向量的长度$\|\mathbf{e}\|$。而残差向量$\mathbf{e}$的长度，就是$\mathbf{y}$到超平面$col(\mathbf{X})$的距离。

只有当此残差向量$\mathbf{e}$垂直于$col(\mathbf{X})$时，$\mathbf{y}$到$col(\mathbf{X})$的距离才能最小化。因此，残差向量$\mathbf{e}$正交于$\mathbf{X}$，故$\mathbf{X}'\mathbf{e} = \mathbf{0}$，这正是 OLS 的正规方程组。

总之，OLS 的实质就是响应变量$\mathbf{y}$向数据矩阵$\mathbf{X}$的列空间所作的投影。

将 OLS 的解$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$代入方程(4.33)可得：

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \underbrace{\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'}_{\equiv \mathbf{P_X}}\mathbf{y} \equiv \mathbf{P_X}\mathbf{y} \tag{4.38}$$

其中，$n \times n$矩阵$\mathbf{P_X} \equiv \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$称为“投影矩阵”(projection matrix)，其作用是将$\mathbf{y}$(或任何n维列向量)投影到$\mathbf{X}$的列空间$col(\mathbf{X})$。

进一步，残差向量$\mathbf{e}$可写为

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{P_X y} = \underbrace{(\mathbf{I} - \mathbf{P_X})}_{\equiv \mathbf{M_X}} \mathbf{y} \equiv \mathbf{M_X y} \qquad (4.39)$$

其中，$n \times n$矩阵$\mathbf{M_X} \equiv \mathbf{I} - \mathbf{P_X} = \left[\mathbf{I} - \mathbf{X}(\mathbf{X'X})^{-1}\mathbf{X'}\right]$称为“消灭矩阵”(annihilator matrix)，其作用是将$\mathbf{y}$向$\mathbf{X}$的列空间$col(\mathbf{X})$投影，然后得到此投影的残差。

消灭矩阵在线性代数的“施密特正交化”(Gram-Schmidt orthogonalization)中起着重要作用，因为施密特正交化本质上就是投影之后计算残差的运算。

4.6 施密特正交化与 QR 分解

虽然 OLS 的显式解$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$简洁漂亮，但在实践中一般并不使用此公式求解，因为计算逆矩阵$(\mathbf{X}'\mathbf{X})^{-1}$可能太费时。回到正规方程组：

$$(\mathbf{X}'\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{y} \quad (4.40)$$

使用“高斯消元法”(Gaussian elimination)求解此线性方程组。若变量个数p很大，则求解此p个方程与p个未知数方程组的时间开销依然较大。

可考虑使用“施密特正交化”，将满列秩的数据矩阵$\mathbf{X} = (\mathbf{x}_{\bullet 1} \cdots \mathbf{x}_{\bullet p})$变为正交矩阵。

经过施密特正交化之后，$\mathbf{X}$的列向量之间均正交，而且每个列向量的长度都被标准化为 1。将施密特正交化的过程用矩阵表示，就是所谓“QR 分解”(QR decomposition)：

$$\mathbf{X} = \mathbf{QR} \qquad (4.41)$$

其中，矩阵$\mathbf{Q}_{n\times p}$的所有列向量都相互正交，且每个列向量的长度均为 1，称为“标准正交向量”(orthonormal vectors)；而$\mathbf{R}$为“上三角矩阵”(upper diagonal matrix)。

不难证明，$\mathbf{Q}'\mathbf{Q} = \mathbf{I}$(参见习题)。

将方程(4.41)代入正规方程组(4.40)可得：

$$(\mathbf{X}'\mathbf{X})\hat{\boldsymbol{\beta}} = (\mathbf{R}'\underbrace{\mathbf{Q}'\mathbf{Q}}_{=\mathbf{I}}\mathbf{R})\hat{\boldsymbol{\beta}} = \mathbf{R}'\mathbf{R}\hat{\boldsymbol{\beta}} = \underbrace{\mathbf{R}'\mathbf{Q}'}_{=\mathbf{X}'}\mathbf{y} = \mathbf{X}'\mathbf{y} \quad (4.42)$$

在上式两边同时左乘$\mathbf{R}'^{-1}$可得：

$$\mathbf{R}\hat{\boldsymbol{\beta}} = \mathbf{Q}'\mathbf{y} \quad (4.43)$$

方程组(4.43)很容易用高斯消元法求解，因为$\mathbf{R}$为上三角矩阵。

4.7 拟合优度

OLS 的样本回归线(超平面)为离所有样本点最近的直线(超平面)。但此最近的直线，究竟离这些样本点有多近？希望有一个绝对的度量，以衡量样本回归线对数据的拟合优良程度。这可通过平方和分解公式来度量。

命题 4.3（平方和分解公式） 如果回归方程有常数项，则可将响应变量的离差平方和 $\sum_{i=1}^{n}(y_i-\overline{y})^2$ 分解为

$$\sum_{i=1}^{n}(y_i-\overline{y})^2=\sum_{i=1}^{n}(\hat{y}_i-\overline{y})^2+\sum_{i=1}^{n}e_i^2 \qquad (4.44)$$

其中，$\bar{y} \equiv \dfrac{1}{n}\sum_{i=1}^{n} y_i$ 为样本均值。

上式表明，导致 y_i 偏离其样本均值 $\bar{y}$ 的因素可分为两部分，即可由模型解释的部分 $\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2$，以及无法由模型解释的残差部分 $\sum_{i=1}^{n} e_i^{\,2}$。平方和分解公式之所以成立，正是由于 OLS 的正交性。

证明 将“$y_i - \bar{y}$”分解为“$y_i - \hat{y}_i + \hat{y}_i - \bar{y} = e_i + \hat{y}_i - \bar{y}$”可得：

$$\begin{aligned}\sum_{i=1}^{n}(y_i-\overline{y})^2 &= \sum_{i=1}^{n}(y_i-\hat{y}_i+\hat{y}_i-\overline{y})^2 = \sum_{i=1}^{n}(e_i+\hat{y}_i-\overline{y})^2 \\ &= \sum_{i=1}^{n}e_i^{\,2}+\sum_{i=1}^{n}(\hat{y}_i-\overline{y})^2+2\sum_{i=1}^{n}e_i\left(\hat{y}_i-\overline{y}\right)\end{aligned} \tag{4.45}$$

因此，只要证明交叉项$\sum_{i=1}^{n}e_i(\hat{y}_i-\overline{y})=\sum_{i=1}^{n}e_i\hat{y}_i-\overline{y}\sum_{i=1}^{n}e_i=0$即可。

首先，根据 OLS 的正交性，

$$\sum_{i=1}^{n}e_i\hat{y}_i=\hat{\mathbf{y}}'\mathbf{e}=0 \tag{4.46}$$

其次，在有常数项的情况下，根据方程(4.31)的第一行可得

$$(1\ 1\cdots 1)\begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix} = \sum_{i=1}^{n} e_i = 0 \tag{4.47}$$

因此，$\sum_{i=1}^{n} e_i(\hat{y}_i - \overline{y}) = 0$，故平方和分解公式成立。 ■

根据平方和分解公式，y_i的离差平方和$\sum_{i=1}^{n}(y_i-\bar{y})^2$可分解为模型可解释的部分$\sum_{i=1}^{n}(\hat{y}_i-\bar{y})^2$与不可解释的部分$\sum_{i=1}^{n}e_i^2$。如果模型可解释的部分所占比重越大，则样本回归线的拟合程度越好。

定义 拟合优度(goodness of fit)R^2为

$$0 \le R^2 \equiv \frac{\sum_{i=1}^{n}(\hat{y}_i-\bar{y})^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2} = 1-\frac{\sum_{i=1}^{n}e_i^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2} \le 1 \qquad (4.48)$$

拟合优度R^2也称为**可决系数**(coefficient of determination)。

可以证明，在有常数项的情况下，拟合优度等于观测值y与拟合值$\hat{y}$之间相关系数的平方，故名R^2：

$$R^2 = \left[\mathrm{Corr}(y, \hat{y})\right]^2 \tag{4.49}$$

R^2越高，则拟合程度越好。然而，如果向回归方程中增加特征变量，则R^2必然只增不减，因为至少可让新增变量的系数为 0 而保持R^2不变。

过高的R^2可能意味着模型在样本内过度拟合，反而导致其样本外预测的能力下降，详见下节。

4.8 过拟合与泛化能力

线性模型可视为一阶泰勒近似。为此，可以考虑加入高次项。但如果加入太多高次项，则会引起**过拟合**(overfit)。我们使用“模拟数据”(simulated data)考察过拟合问题。

使用模拟数据的好处在于，知道数据生成过程，即数据中真正的信号部分 $f(\mathbf{x})$，故可将估计结果 $\hat{f}(\mathbf{x})$ 与 $f(\mathbf{x})$ 相比。假设数据生成过程为

$$y_i = \sin(2\pi x_i) + \varepsilon_i \quad (i = 1, \cdots, 10) \tag{4.50}$$

其中，特征变量 $x_i \sim U[0,1)$(在区间 $[0,1)$ 服从均匀分布)，扰动项 $\varepsilon_i \sim N(0, 0.3^2)$(均值为 0，标准差为 0.3 的正态分布)，而样本容量为 $n = 10$。

首先，导入所需模块：

```
In [1]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...: import seaborn as sns
   ...: import statsmodels.formula.api as smf
```

其次，生成模拟数据，并画散点图与函数 $f(x)=\sin(2\pi x)$：

```
In [2]: np.random.seed(42)
   ...: x = np.random.rand(10)
   ...: y = np.sin(2 * np.pi * x) + np.random.normal(0,
                    0.3, 10)
   ...: plt.scatter(x, y)
   ...: plt.xlabel('x')
   ...: plt.ylabel('y')
   ...: w = np.linspace(0, 1, 100)
   ...: plt.plot(w, np.sin(2* np.pi * w))
```

其中，倒数第 2 行命令定义 w 为在$[0,1]$区间的 100 个等分点，而最后 1 行命令在 w 的这 100 个等分点画函数 $f(x)=\sin(2\pi x)$，结果参见图 4.7。

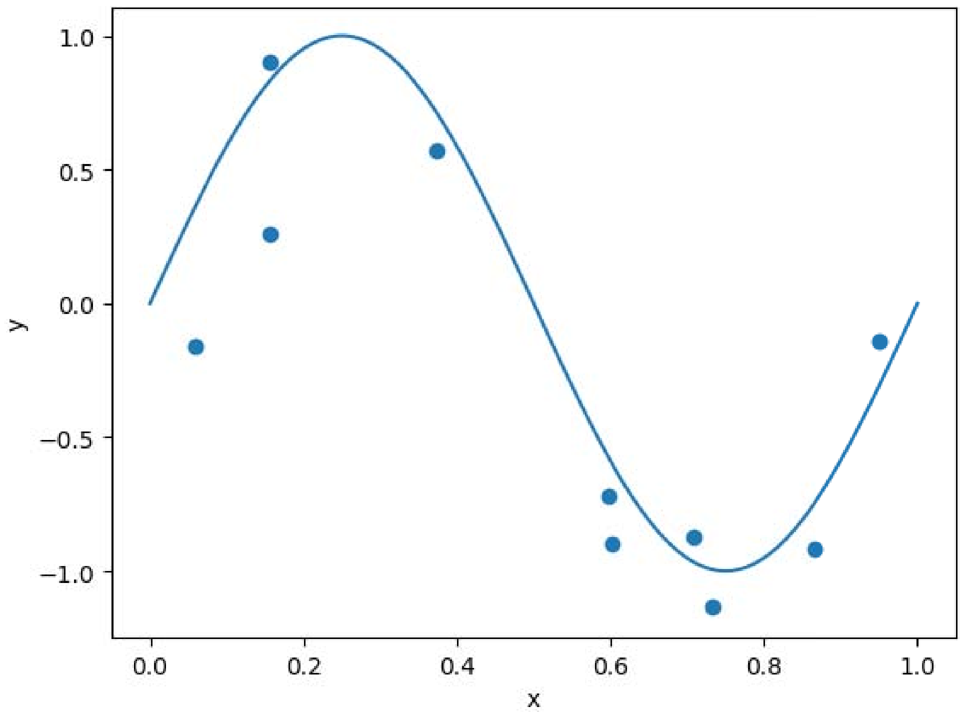


图 4.7 散点图与真实函数 $f(x)=\sin(2\pi x)$

考虑使用以下多项式(polynomial)函数来拟合此数据：

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_M x^M + \varepsilon \qquad (4.51)$$

是否多项式的阶数M越高，则拟合效果越好？

对于 1 至 9 阶多项式($M = 1, \cdots, 9$)，使用 `for` 循环分别进行 OLS 估计，然后将拟合结果(共有 9 个)，在3×3的画布上画图展示：

```
In [3]: fig, ax = plt.subplots(3, 3, sharex=True,
sharey=True, subplot_kw=dict(yticks=[]))
   ...: fig.subplots_adjust(hspace=0.1, wspace=0.1)
   ...: for m in range(1, 10):
   ...:   ax = fig.add_subplot(3, 3, m)
   ...:   ax.set_xlim([0, 1])
   ...:   ax.set_ylim([-1.5, 1.5])
   ...:   ax.set_yticks(np.arange(-1.5, 2, 0.5))
   ...:   ax.plot(w, np.sin(2 * np.pi * w), 'k',
                  linewidth=1)
   ...:   sns.regplot(x, y, order=m, ci=None,
    scatter_kws={'s': 15}, line_kws={'linewidth': 1})
   ...:   ax.text(0.6, 1, 'M = '+ str(m))
```

```
   ...: plt.tight_layout()
```

其中，第 1 个命令设定画布(fig)由 3×3 的画轴(ax)所构成，参数“sharex=True”与“sharey=True”表示所有画轴将共享 x 轴与 y 轴的尺度，而参数“subplot_kw=dict(yticks=[])”将 y 轴的标记(ticks)设为空(避免重复标记)。

第 2 个命令将这些“子图”(subplots，即 3 行与 3 列的画轴)之间的空白区域(padding)之高度(hspace)与宽度(wspace)均设为 0.1。

在 for 循环中，命令“ax.set_xlim([0, 1])”与“ax.set_ylim([-1.5, 1.5])”分别设定 x 轴与 y 轴的画图范围。

命令“`ax.set_yticks(np.arange(-1.5, 2, 0.5))`”表示在 y 轴从-1.5 至 2(不含 2)的范围内，每隔 0.5 作一个标记(tick)。

命令“`ax.plot(w, np.sin(2 * np.pi * w), 'k', linewidth=1)`”画函数 $f(x)=\sin(2\pi x)$，并将颜色设为黑色(`'k'`)，而线宽(`linewidth`)为 1。

命令“`sns.regplot(x, y, order=m, ci=None, scatter_kws={'s':15},line_kws={'linewidth': 1})`”使用 seaborn 模块的 `regplot()`函数同时画散点图与回归拟合图。

其中，参数“order=m”表示作 m 阶回归，“ci=None”表示不画置信区间，“scatter_kws={'s': 15}”以字典的格式将散点的尺寸设为 15，而参数“line_kws={'linewidth': 1}”将回归线的宽度设为 1。

命令“ax.text(0.6 ,1, 'M = '+ str(m))”在坐标(0.6, 1)的位置加上文本“'M = '+ str(m))”。

最后，命令“plt.tight_layout()”去掉图四周的多余空白，结果参见图 4.8。

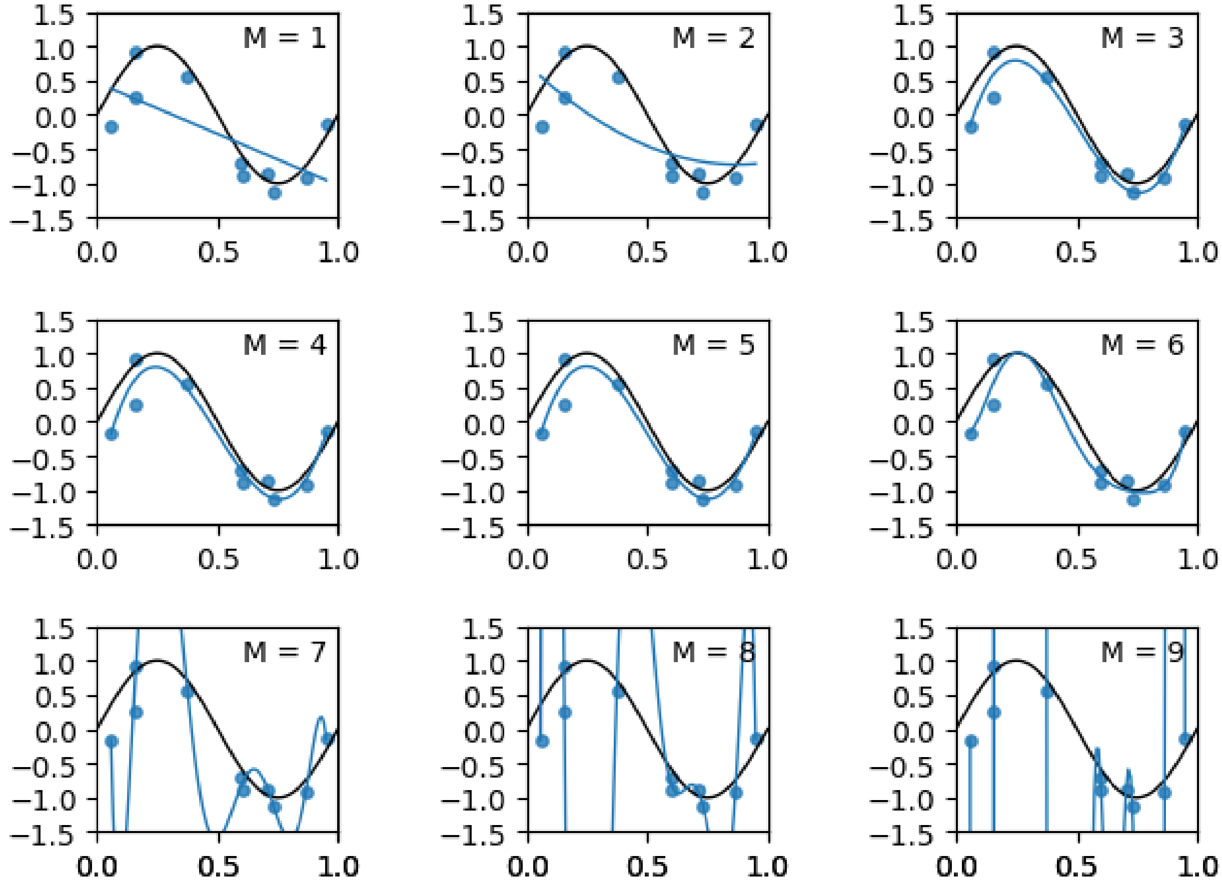


图 4.8 多项式回归的拟合效果

从图 4.8 可见，线性函数与二次函数均未能充分捕捉到真实函数 $f(x)=\sin(2\pi x)$的主要特征，处于**欠拟合**(underfit)的状态。

3-6 次函数似乎拟合得最好，可视为“good fit”、“just fit”或“ideal fit”。

6 次以上的多项式函数，其拟合效果则越来越**过拟合**(overfit)，即虽然样本内的拟合程度越来越好(比如R^2越来越高，参见图 4.9)，但所估计的 $\hat{f}(\mathbf{x})$则日益偏离真实 $f(\mathbf{x})$。

极端情况发生在$M=9$，对于此 9 阶多项式函数，其待估参数为$K=10$(含常数项)，正好等于样本容量$n=10$，使得“自由度”(degree of freedom) 降为$(n-K)=0$。

在样本内完美地拟合了观测数据($\hat{f}(\mathbf{x})$经过所有数据)；但所学得的函数$\hat{f}(\mathbf{x})$波动剧烈，与真实信号$f(\mathbf{x})$相去甚远。

为进一步考察过拟合，当多项式回归的阶数变动时，计算样本内拟合优度R^2，以及**训练误差**(training error)，即均方误差之开平方(Root Mean Square Error，简记 RMSE)：

```
In [4]: R2 = []
   ...: train_error = []
   ...: for m in range(1, 9):
   ...:     stats = np.polyfit(x, y, m, full=True)
   ...:     ssr = stats[1]
   ...:     train_error.append((ssr / 10) ** 0.5)
   ...:     r2 = 1 - ssr / (sum((y-y.mean()) ** 2))
   ...:     R2.append(r2)
   ...: R2.append(1)
   ...: train_error.append(0)
```

其中，第 1-2 个命令首先初始化 `R2` 与 `train_error` 为空的列表(empty lists)。

在 for 循环中，使用 Numpy 的 polyfit()函数进行多项式回归，其中参数“m”为多项式的阶数，而“full=True”表示返回更多信息(默认仅返回回归系数)。

把 polyfit()函数的返回值记为 stats，并提取其第 1 个元素 stats[1]为 ssr，即残差平方和(sum of squared residuals)。

命令“train_error.append((ssr / 10) ** 0.5)”将残差平方和 ssr 除以样本容量 10，再开平方，即为均方误差(RMSE)，然后加入列表 train_error。

类似地，计算拟合优度，并加入列表 R2。

由于 `polyfit()`函数不便计算“`m = 9`”的情形(10 个观测值估计 10 个未知参数)，故最后两个命令手工加上其“`R2 = 1`”与“`train_error = 0`”。

通过画图考察样本内拟合优度随着回归阶数的变化，可输入命令：

```
In [5]: plt.plot(range(1, 10), R2, 'o-')
   ...: plt.xlabel('Degree of Polynomial')
   ...: plt.ylabel('R2')
   ...: plt.title('In-sample Fit')
```

其中，第 1 个命令的参数`'o-'`表示既画圆点(`'o'`)，又画折线(`'-'`)。结果参见图 4.9。

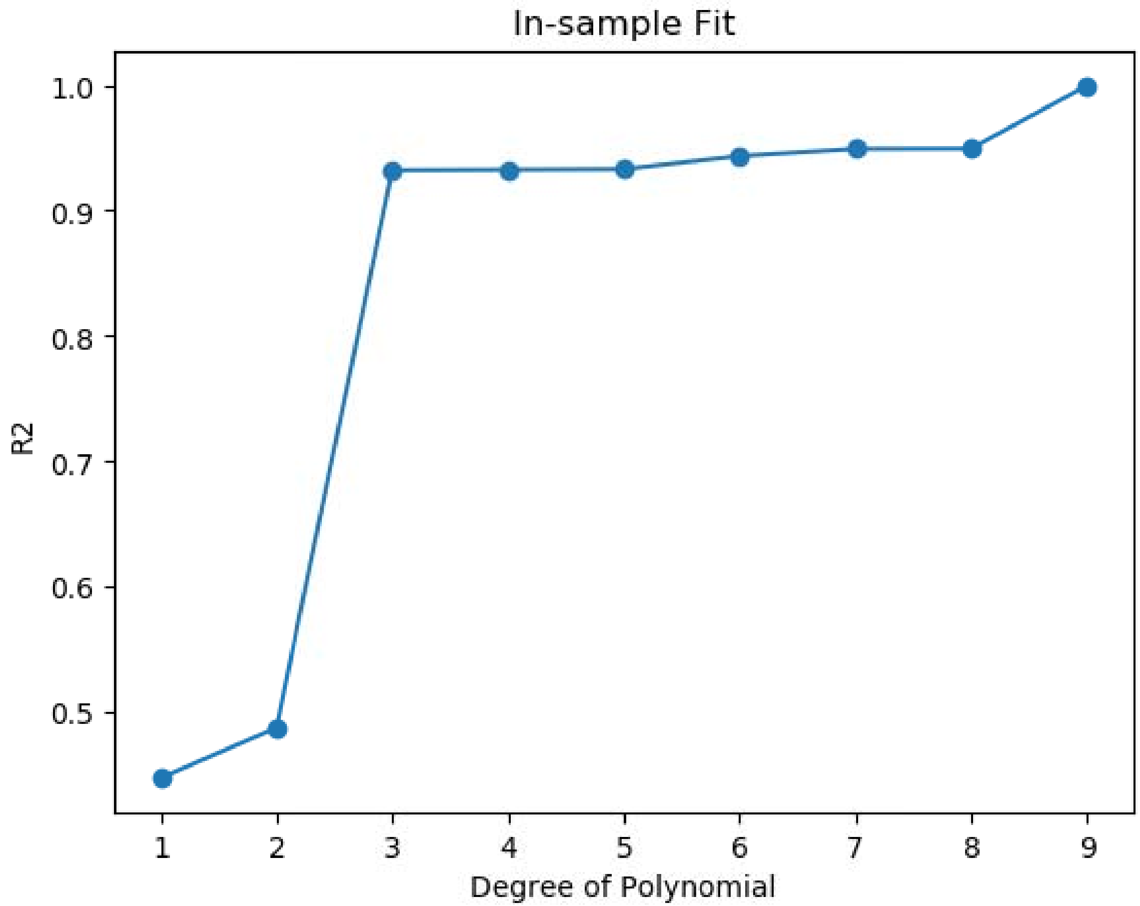


图 4.9 样本内的拟合优度

图 4.9 显示，随着多项式的阶数越来越高，样本内的拟合优度R^2越来越高，并于$M = 9$(9 阶多项式)时达到 1。

但多项式回归发现数据中真正信号的能力，当$M > 3$之后却越来越低(参见图 4.8)。

直观上，数据可视为信号与噪音的组合，参见方程(4.2)。当样本内的拟合越来越完美时，这意味着回归函数也拟合了大量的噪音(因为回归函数“跟”数据跟得太紧)；而噪音对于样本外的预测毫无意义。

在过拟合的情况下，虽然样本内的拟合优度很高，但模型在样本外的预测能力反而下降。

然而，机器学习的目的恰恰是样本外的预测能力，即将模型运用于其未见过的数据(unseen data)时，所具有的推广预测能力，即模型的**泛化能力**(generalization)。

另一方面，对于模型已经见过并据此进行优化的训练数据，即使拟合得再好，也说明不了问题：它可能只是记住了训练数据，比如此例中$M=9$的多项式回归情形。

下面继续用模拟数据考察多项式回归的泛化能力。使用同样的数据生成过程(但用不同的随机数种子)，随机生成 100 个原模型所未见过的新观测值，构成一个**测试集**(test data)，然后进行 1-9 阶多项式回归，并计算其**测试误差**(test error)：

```
In [6]: test_error = []
   ...: for m in range(1, 10):
   ...:     coefs = np.polyfit(x, y, m)
   ...:     np.random.seed(123)
   ...:     x_new = np.random.rand(100)
   ...:     y_new = np.sin(2* np.pi * x_new)+
                              np.random.normal(0, 0.3, 100)
   ...:     pred = np.polyval(coefs, x_new)
   ...:     ssr = (sum((pred - y_new) ** 2) / 100) ** 0.5
   ...:     test_error.append(ssr)
```

其中，第 1 个命令初始化 `test_error` 为一个空的列表。

在 for 循环中，命令“coefs = np.polyfit(x, y, m)”将 m 阶多项式回归的系数估计值赋值给 coefs。

命令“pred = np.polyval(coefs, x_new)”计算在测试集 x_new 上的预测值，并记为 pred；由此可计算相应的残差平方和及测试误差。

更直观地，将将训练误差与测试误差画在一张图上，可输入命令：

```
In [7]: plt.plot(range(1, 10), train_error, 'o--k',
                 label='Training Error')
   ...: plt.plot(range(1, 10), test_error, 'o-b',
                 label='Test Error')
   ...: plt.ylim(-0.05, 1)
```

```
   ...: plt.xlabel('Degree of Polynomial')
   ...: plt.ylabel('Root Mean Squared Errors')
   ...: plt.title('Training Error vs. Test Error')
```

其中，第 1 个命令的参数`'o--k'` 表示以黑色(`'k'`)画圆点(`'o'`)与虚线(`'--'`)。

第 2 个命令的参数`'o-b'` 表示以蓝色(`'b'`)画圆点(`'o'`)与实线(`'-'`)。

第 3 个命令设定 `y` 轴的画图范围，结果参见图 4.10。

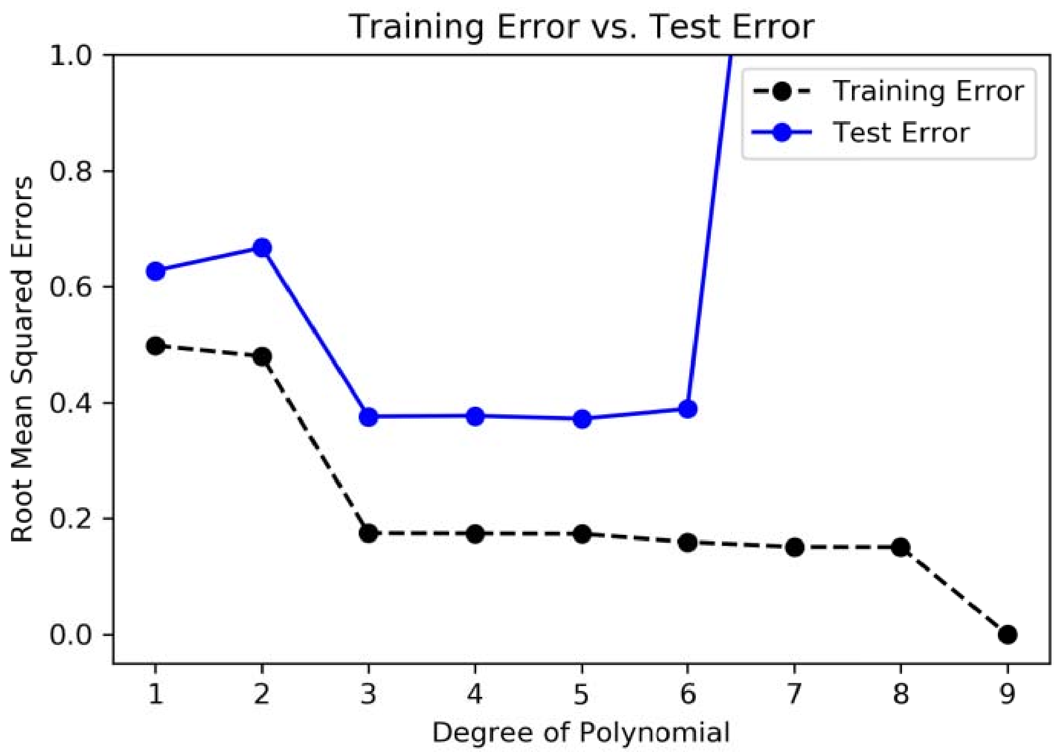


图 4.10 训练误差与测试误差

从图 4.10 可见，随着多项式阶数的上升，样本内的训练误差(`Training Error`)不断下降，并于$M=9$时训练误差降为 0。

作为对比，考察测试误差(Test Error)的具体取值：

```
In [8]: print(test_error)
[0.6272929120340168, 0.6674828980959218,
0.3755499502639364, 0.37676901692557596,
0.37162179336514245, 0.388876218260886,
1.8923251664099412, 13.382614809971427,
1124.978217995193]
```

结果显示，样本外的测试误差在$M = 5$时达到最低值(但$3 \leq M \leq 6$时差别不大)，然后不断上升；而且当自由度几乎用尽时，测试误差急剧上升，以致于超出 y 轴的画图范围。

4.9 偏差与方差的权衡

从图 4.10 可知，随着模型复杂程度的增加，测试误差一般呈现出 U 型的曲线特征，即先下降而后上升。

为什么测试误差一般呈 U 型？事实上,测试误差受到两种不同力量的影响，即偏差与方差。

偏差(bias)指的是估计量是否有系统误差(比如，系统性高估或低估)。给定 $\mathbf{x}$，则估计量 $\hat{f}(\mathbf{x})$的偏差定义为

$$\text{Bias}(\hat{f}(\mathbf{x})) \equiv \text{E}\,\hat{f}(\mathbf{x}) - f(\mathbf{x}) \qquad (4.52)$$

由上式可知，偏差度量的是在大量重复抽样过程中，$\hat{f}(\mathbf{x})$对于 $f(\mathbf{x})$的平均偏离程度。

方差(Variance)则衡量在大量重复抽样过程中，估计量 $\hat{f}(\mathbf{x})$ 本身围绕着其期望值 $\mathrm{E}\,\hat{f}(\mathbf{x})$ 的波动幅度，其定义为

$$\mathrm{Var}(\hat{f}(\mathbf{x})) \equiv \mathrm{E}\left[\hat{f}(\mathbf{x}) - \mathrm{E}\,\hat{f}(\mathbf{x})\right]^2 \qquad (4.53)$$

给定 $\mathbf{x}$，估计量 $\hat{f}(\mathbf{x})$ 的均方误差可作如下分解：

$$
\begin{aligned}
\mathrm{MSE}(\hat{f}(\mathbf{x})) &= \mathrm{E}\left[y-\hat{f}(\mathbf{x})\right]^2 \quad (\text{MSE定义}) \\
&= \mathrm{E}\left[f(\mathbf{x})+\varepsilon-\hat{f}(\mathbf{x})\right]^2 \quad (\text{代入模型}\, y = f(\mathbf{x})+\varepsilon) \\
&= \mathrm{E}\left[f(\mathbf{x})-\mathrm{E}\,\hat{f}(\mathbf{x})+\mathrm{E}\,\hat{f}(\mathbf{x})-\hat{f}(\mathbf{x})+\varepsilon\right]^2 \quad (\text{加减}\,\mathrm{E}\,\hat{f}(\mathbf{x})) \\
&= \underbrace{\left[\mathrm{E}\,\hat{f}(\mathbf{x})-f(\mathbf{x})\right]^2}_{\mathrm{Bias}^2} + \underbrace{\mathrm{E}\left[\hat{f}(\mathbf{x})-\mathrm{E}\,\hat{f}(\mathbf{x})\right]^2}_{\mathrm{Variance}} + \underbrace{\mathrm{E}(\varepsilon^2)}_{\mathrm{Var}(\varepsilon)} \\
&= \underbrace{\mathrm{Bias}^2+\mathrm{Variance}}_{\textit{reducible}} + \underbrace{\mathrm{Var}(\varepsilon)}_{\textit{irreducible}}
\end{aligned}
\tag{4.54}
$$

其中，平方展开后的三个交叉项均为 0，证明如下。

首先，第 1 个交叉项为

$$
\begin{aligned}
&\mathrm{E}\left[f(\mathbf{x})-\mathrm{E}\,\hat{f}(\mathbf{x})\right]\left[\mathrm{E}\,\hat{f}(\mathbf{x})-\hat{f}(\mathbf{x})\right] \qquad (\left[f(\mathbf{x})-\mathrm{E}\,\hat{f}(\mathbf{x})\right]\text{为常数}) \\
&=\left[f(\mathbf{x})-\mathrm{E}\,\hat{f}(\mathbf{x})\right]\cdot \mathrm{E}\left[\mathrm{E}\,\hat{f}(\mathbf{x})-\hat{f}(\mathbf{x})\right] \qquad (\text{求期望为线性运算}) \\
&=\left[f(\mathbf{x})-\mathrm{E}\,\hat{f}(\mathbf{x})\right]\cdot \underbrace{\left[\mathrm{E}\,\hat{f}(\mathbf{x})-\mathrm{E}\,\hat{f}(\mathbf{x})\right]}_{=0}=0
\end{aligned}
\tag{4.55}
$$

其次，根据迭代期望定律及假设 4.2，第 2 个交叉项

$$\begin{aligned}
\mathrm{E}\left[(f(\mathbf{x})-\mathrm{E}\,\hat{f}(\mathbf{x}))\varepsilon\right] &= \mathrm{E}_{\mathbf{x}}\,\mathrm{E}\left[(f(\mathbf{x})-\mathrm{E}\,\hat{f}(\mathbf{x}))\varepsilon \middle| \mathbf{x}\right] && \text{(迭代期望定律)} \\
&= \mathrm{E}_{\mathbf{x}}\left[(f(\mathbf{x})-\mathrm{E}\,\hat{f}(\mathbf{x}))\cdot \mathrm{E}(\varepsilon \mid \mathbf{x})\right] && \text{(假设4.2)} \\
&= \mathrm{E}_{\mathbf{x}}\left[(f(\mathbf{x})-\mathrm{E}\,\hat{f}(\mathbf{x}))\cdot 0\right] = 0
\end{aligned} \tag{4.56}$$

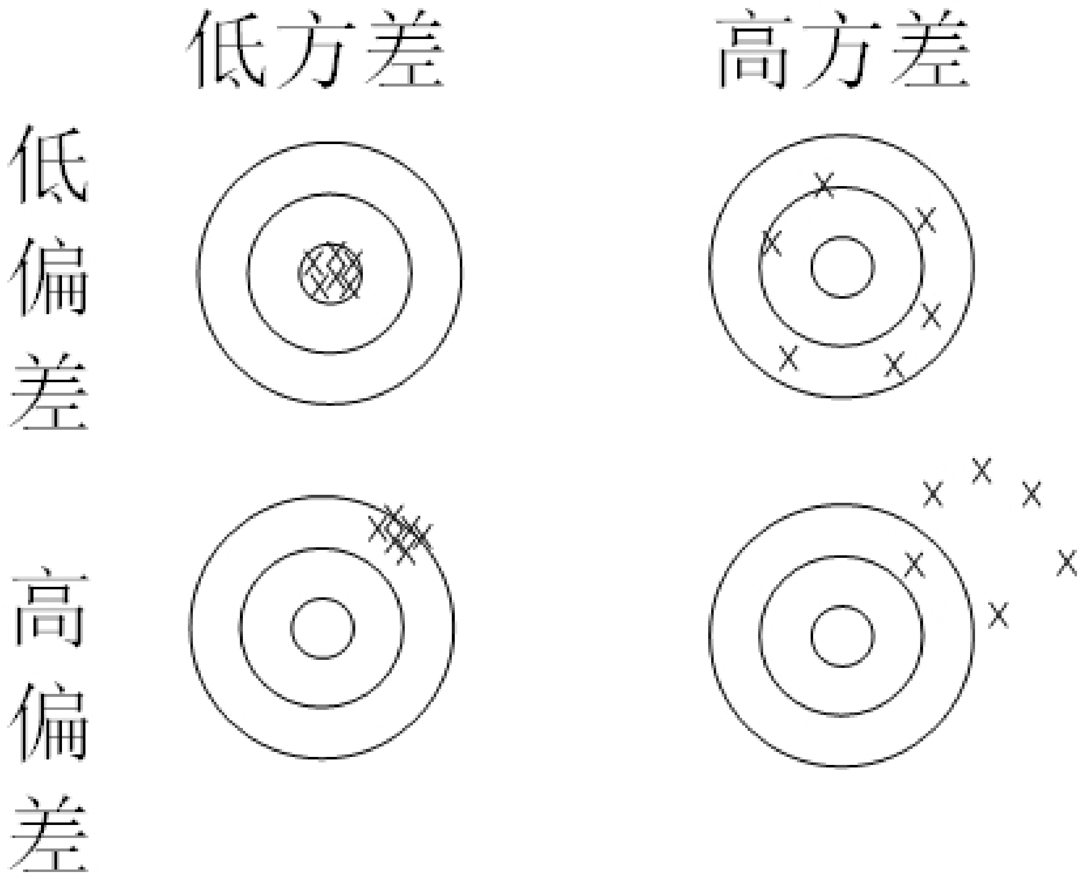
其中，在给定$\mathbf{x}$时，可将$(f(\mathbf{x})-\mathrm{E}\,\hat{f}(\mathbf{x}))$视为常数提出。

类似地，第 3 个交叉项也为 0：

$$
\begin{aligned}
\mathrm{E}\left[(\mathrm{E}\,\hat{f}(\mathbf{x})-\hat{f}(\mathbf{x}))\varepsilon\right] &= \mathrm{E}_{\mathbf{x}}\,\mathrm{E}\left[(\mathrm{E}\,\hat{f}(\mathbf{x})-\hat{f}(\mathbf{x}))\varepsilon \,\middle|\, \mathbf{x}\right] \qquad (\text{迭代期望定律}) \\
&= \mathrm{E}_{\mathbf{x}}\left[(\mathrm{E}\,\hat{f}(\mathbf{x})-\hat{f}(\mathbf{x}))\cdot \mathrm{E}(\varepsilon\,|\,\mathbf{x})\right] \qquad (\text{假设4.2}) \\
&= \mathrm{E}_{\mathbf{x}}\left[(\mathrm{E}\,\hat{f}(\mathbf{x})-\hat{f}(\mathbf{x}))\cdot 0\right] = 0
\end{aligned}
\tag{4.57}
$$

因此，估计量 $\hat{f}(\mathbf{x})$的均方误差可分解为偏差平方(Bias^2)、方差(Variance)与扰动项方差$\mathrm{Var}(\varepsilon)$之和。

偏差平方与方差均“可降低”(reducible)；在极端情况下，如果知道真实函数 $f(\mathbf{x})$，则偏差与方差均为 0，参见图 4.11。



4.11 偏差与方差的图示

在图 4.11 中，靶心为真实的 y，而射中位置为预测的 $\hat{y}$。左上角的低偏差、低方差情形为最为理想的模型，其 $\hat{y}$ 总在 y 的附近。

右上角的模型虽然平均而言系统偏差很小，但方差很大，故经常偏离靶心，存在“过拟合”(overfit)。

左下角的模型则正好相反，虽然方差很小，几乎总打在相同的地方，但遗憾的是此地并非靶心，故偏差较大，存在“欠拟合”(underfit)。

右下角的模型则偏差与方差都较大，不仅存在较大系统偏差，而且波动幅度大，故是最糟糕的模型。

另一方面，扰动项方差$\mathrm{Var}(\varepsilon)$则“不可降低”(irreducible)，即使知道真实函数$f(\mathbf{x})$，但$\mathrm{Var}(\varepsilon)$也依然存在。

本质上，扰动项ε的方差$\mathrm{Var}(\varepsilon)$决定了预测问题的困难程度。

如果扰动项方差$\mathrm{Var}(\varepsilon)$很大，即使你所学得的函数$\hat{f}(\mathbf{x})$已经很接近于真实函数$f(\mathbf{x})$，预测准确度依然可能不高；比如对金融市场的预测。

另一方面，如果扰动项方差很小，则只要$\hat{f}(\mathbf{x})$足够接近$f(\mathbf{x})$，预测准确度就可以很高；比如人脸识别。

通常来说，偏差平方与方差之间存在着此消彼长的替代关系。

当模型过于简单时(比如第 4.8 节多项式拟合案例中M为 1 或 2)，则估计量$\hat{f}(\mathbf{x})$的偏差较大，但方差较小；此时存在欠拟合。

随着模型的复杂程度增加，估计量$\hat{f}(\mathbf{x})$越来越接近于数据，故偏差变小；但由于数据包含噪音，故紧跟数据使得方差随之增大，此时易出现过拟合。

因此，在选择模型复杂程度时，存在**偏差与方差的权衡**(bias-variance trade-off)，故须选择合适的模型复杂程度，使得模型的均方误差达到最小值(比如上例中的$M=3$)，参见图 4.12。

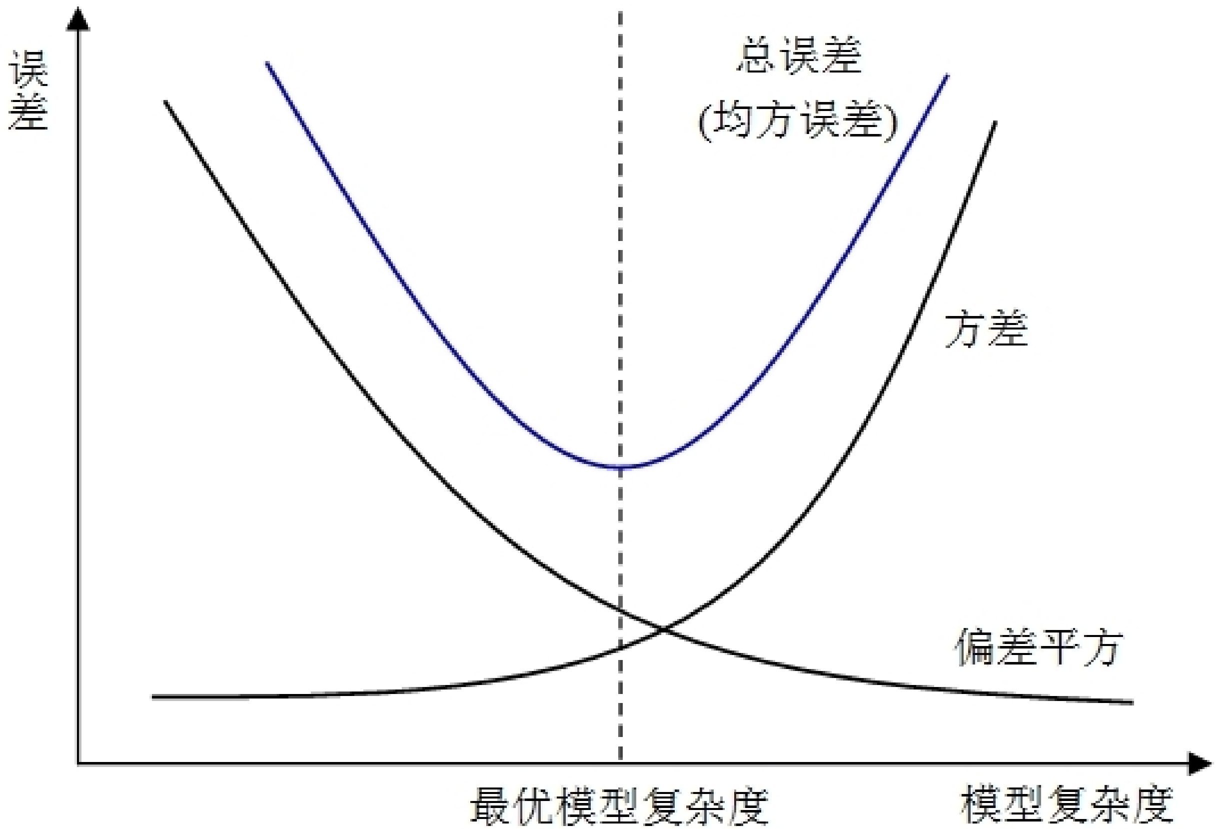


图 4.12 偏差与方差的权衡

4.10 模型评估的再抽样方法

训练误差可能是对测试误差的糟糕估计。为了纠正此偏误，统计学的传统方法是使用**信息准则**(information criterion)，对过于复杂的模型进行惩罚。比如，常用的 AIC(Akaike Information Criterion)信息准则为

$$\min_{p} \ \mathrm{AIC} \equiv \ln(\mathrm{MSE}) + \frac{2}{n} p \qquad (4.58)$$

其中，右边第一项为对模型拟合度的奖励(减少均方误差 MSE)，而第二项为对变量过多(模型过于复杂)的惩罚(为变量个数 p 的增函数)。

当 p 上升时，第一项下降而第二项上升。通过选择合适的变量个数 p，最小化 AIC，达到在模型拟合度与简洁性(parsimony)之间的权衡，以避免过拟合。

另一常用的信息准则为“贝叶斯信息准则”(Bayesian Information Criterion，简记 BIC)：

$$\min_{p} \ \text{BIC} \equiv \ln(\text{MSE}) + \frac{\ln n}{n} p \quad (4.59)$$

BIC 准则与 AIC 准则只有第二项有差别。一般来说，$\ln n > 2$(除非样本容量很小)，故 BIC 准则对于变量过多的惩罚比 AIC 准则更为严厉，因此 BIC 准则更强调模型的简洁性。

但信息准则的原理依赖于大样本理论的一些假设，在有限样本中未必能选出预测能力最优的模型。

故机器学习一般并不使用信息准则来评估模型，而通过**再抽样**(resampling)或**重采样**的方法来度量模型的泛化能力，包括验证集法、重复验证集法、K 折交叉验证、重复 K 折交叉验证与自助法等。

在使用**验证集法**(validation set approach)时，先将样本数据随机地一分为二，其中大部分(比如 70%)作为训练集，而将其余的小部分(比如 30%)作为“验证集”(validation set)或“保留集”(hold-out set)，参见图 4.13。

在估计模型时，仅使用训练集；然后在验证集中进行样本外预测，并计算“验证集误差”(validation set error)，以此估计测试误差(test error)。

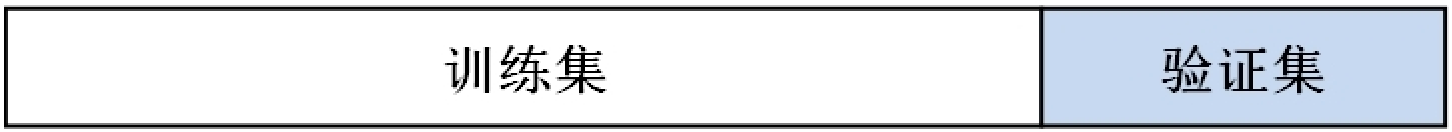


图 4.13 验证集法的示意图

验证集法虽然操作方便，但有两个缺点。

首先，验证集法的结果取决于随机分组。在使用不同的随机数种子，将数据分为不同的训练集与验证集时，其验证集误差可能波动较大。

其次，在使用验证集法时，仅使用训练集的部分数据来估计模型，浪费了验证集的那部分信息，故估计效率较低，导致出现偏差。

这意味着，与使用整个样本进行估计相比，验证集法可能高估(overestimate)测试误差。

改进验证集法的一种简单方法为**重复验证集法**(repeated training/test splits)，即重复使用验证集法，比如 100 次。

这意味着，每次均将数据集随机分组为训练集与验证集(分割比例保持不变)，然后将每次所得的验证集误差进行平均。

这种方法虽可使估计结果更为稳定，但依然无法解决验证集法的偏差问题(因为每次仅用 70%的数据进行训练)。

在机器学习的实践中，大量使用 K **折交叉验证**(*K*-fold Cross-Validation，简记 *K*-fold CV)来度量测试误差；其中 K 一般为 5 或 10。以 10 折交叉验证为例，参见图 4.14。

首先，将样本数据随机地分为大致相等的 10 个子集。

其次，每次均留出一折(约十分之一样本)作为验证集，而将其余九折(约十分之九样本)作为训练集，以训练集估计模型，然后在验证集中进行预测，并计算验证集均方误差。

最后，将所有验证集均方误差进行平均，即为“交叉验证误差”(cross-validation error)，可作为对测试误差的估计。

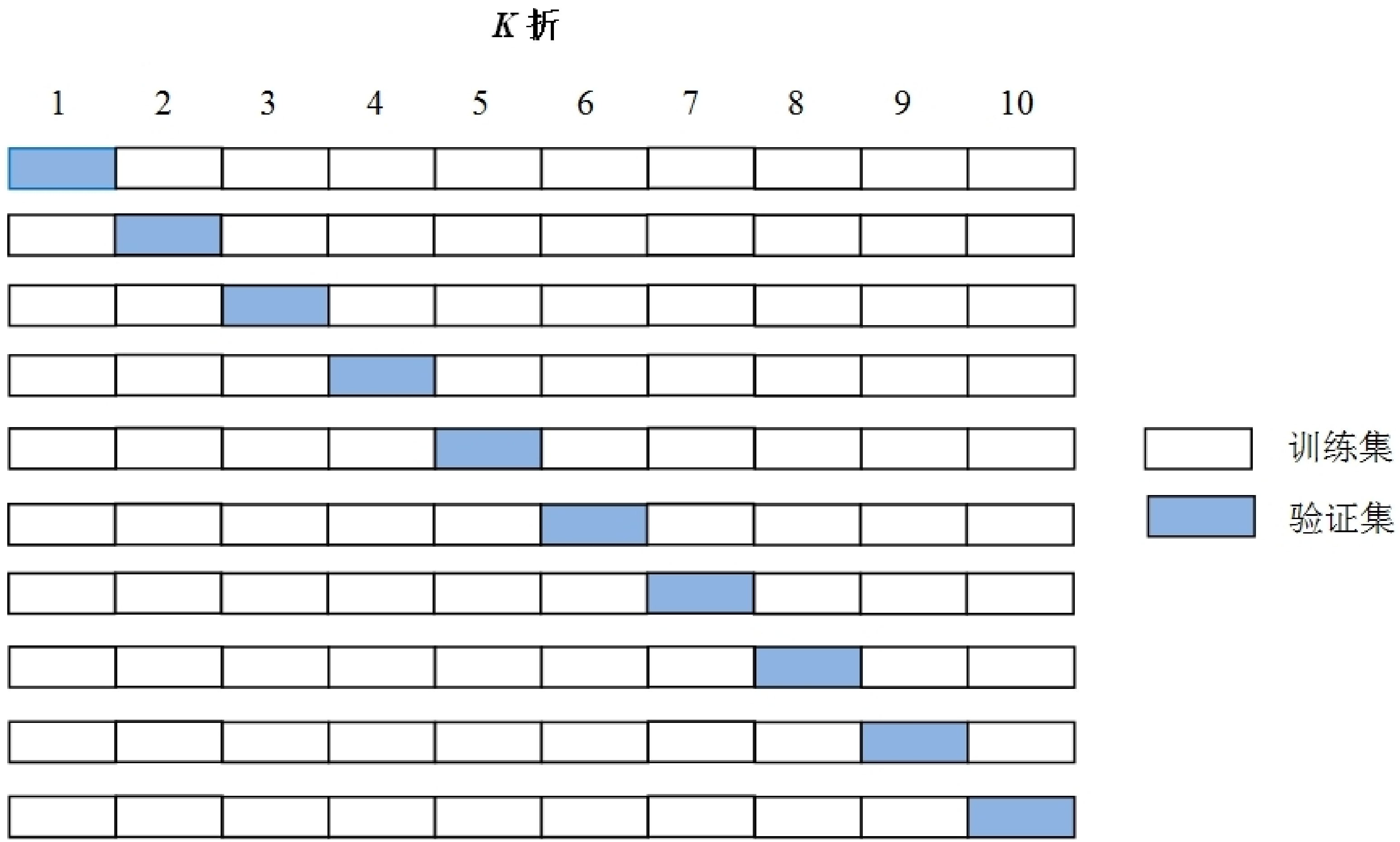


图 4.14 10 折交叉验证的示意图

如果将留出第k折所得的验证集均方误差记为MSE_k，则K折交叉验证误差为

$$CV_{(K)} \equiv \frac{1}{K}\sum_{k=1}^{K}\text{MSE}_k \tag{4.60}$$

K折交叉验证正好弥补了验证集法的缺陷。

由于交叉验证误差为K个验证集误差之平均，故很大程度上避免了验证集法的随机性与波动性。

而且，在进行交叉验证时，所有数据均有机会进入训练集，故估计效率高于验证集法。

特别地，如果样本容量较小，还可使用**留一交叉验证**(Leave-One-Out CV，简记 LOOCV)。事实上，这就是n折交叉验证，即$K=n$，其中n为样本容量。

在进行 LOOCV 时，将样本数据等分为n折，每折仅包含一个样本点，故其估计结果不再有随机性。

由于每次均使用$(n-1)$个数据去预测留出的 1 个数据，故总共需要估计n次。显然，如果样本容量n很大，则 LOOCV 的时间开销也将很大。因此，实践中常使用 5 折或 10 折交叉验证。

事实上，如何选择K，也涉及偏差与方差的权衡。

首先，考察偏差(bias)。对于验证集法，可大致视为$K = 2$ 或 3，导致训练集占整个数据集的比重不大，故出现较大偏差。而在另一极端，留一交叉验证法使用$(n-1)$个数据进行训练，故偏差很小。而常用的$K = 5$ 或 10 折交叉验证，其偏差水平则介于二者之间。

其次，考虑方差(variance)。由于 LOOCV 每次使用的训练集非常类似(只有 1 个观测值不同)，故所得结果也高度(正)相关。当我们将这n个高度相关的结果进行平均时，则所得结果的方差较大。反之，如果$K < n$，则每次使用的训练集不尽相同，故将其结果进行平均时可以降低方差。

因此，实践中一般进行$K = 5$ 或 10 折交叉验证，以大致达到偏差与方差的最佳权衡，避免偏差较高(比如验证集法)或方差较大(比如 LOOCV)。

考虑到 10 折交叉验证的结果仍可能依赖于随机分组，在实践中也可将 10 折交叉验证重复 10 次(每次使用不同的随机种子分组)，然后将这 10 次 10 折交叉验证的结果再次进行平均。

这意味着，总共对 100 个验证集误差进行平均，这种方法称为**重复 *K* 折交叉验证**(repeated *K*-fold cross-validation)。

如果原始样本较小，即使用 10 折交叉验证，在进行训练时，也会进一步损失十分之一的宝贵数据。

此时，可考虑另一种再抽样方法，即**自助法**(bootstrap)，由 Efron(1979) 提出。

自助法是一种“有放回”(with replacement)的再抽样方法，即每次随机抽取一个观测值后，再将其放回样本；如此反复，直至得到n个观测值(保持样本容量不变)的“自助样本”(bootstrap sample)。然后，使用此自助样本作为训练集，以估计模型。

由于进行有放回的再抽样，故在自助样本中，有些观测值可能被多次抽中，而有些观测值则一直未被抽中。

在获得自助样本的n次抽样中(每次仅抽一个观测值)，某个观测值一直未被抽中的概率为$\left(1-\dfrac{1}{n}\right)^n$。

如果样本容量n较大，则该观测值一直未被抽中的概率接近于其极限值：

$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n}=\lim_{n\to\infty}\left(1+\frac{-1}{n}\right)^{n}=e^{-1}\approx 0.368 \qquad (4.61)$$

因此，如果n较大，则自助样本中未出现的观测值所占比重约为36.8%(近似于三分之一)。

这些观测值称为**袋外观测值**(out-of-bag observations)。

由于袋外观测值并不出现于自助样本，故可将其构成验证集，并计算验证集误差，称为**袋外误差**(out-of-bag error)。

使用自助法的好处是，可以产生任意数量的自助样本，比如 500 个自助样本，然后将 500 个袋外误差进行平均。

但这些自助样本并非来自真正的总体，而是将原始样本视为总体，然后不断地从原始样本中进行抽样。由于原始样本与总体并不相同，而且每次仅使用约 63.2%的数据进行训练，故自助法也会产生偏差，并不常用。

在统计学中，一般使用自助法估计某统计量的标准误(standard errors)，以评估该统计量的不确定性(uncertainty)，即在多次随机抽样中的波动程度。在机器学习中，自助法则常用于“集成学习”(ensemble learning)。

综合考虑以上各种方法的优缺点，机器学习在评估模型的测试误差时，通常使用$K = 5$或 10 折交叉验证。

4.11 线性回归的 Python 案例

本节以机器学习常用的波士顿房价数据 boston，演示线性回归、验证集法与交叉验证。

Harrison and Rubinfeld (1978)曾用此数据研究空气污染对于房价的影响。

该数据集包含 1970 年波士顿 506 个社区有关房价的 14 个变量。响应变量为社区房价中位数 MEDV(median value of owner-occupied homes in $1000s)。

特征变量包括平均房间数 RM、房屋年龄 AGE(1940 年前所造房屋的比重)、黑人所占比重的平方 B、低端人口所占百分比 LSTAT、人均犯罪率 CRIM、可建 25,000 平方英尺以上大院的住宅用地比例 ZN、非零售商业用地比例 INDUS、生师比 PTRATIO(pupil-teacher ratio)、房产税率 TAX、是否毗邻查尔斯河 CHAS、距离波士顿五个就业中心的加权平均距离 DIS、高速公路可达性指标 RAD、氮氧化物浓度 NOX。

其中，有些变量由于缺乏社区数据，以“镇”(town)的数据替代，比如 B、CRIM、INDUS 与 PTRATIO。

* 本章其余内容详见教材，以及配套 Python 程序（现场演示）。