第5章

逻辑回归算法

5.1 导　　读

5.1.1 逻辑回归基本概念

逻辑回归(Logistic Regression,LR)属于广义线性自回归(Generalized Linear Autoregression,GLAR),因此与多元线性回归分析很类似。它们在模型形式上大致相同,都具有 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$ 的形式,其中 $\boldsymbol{w}$ 和 b 是待求参数。其区别在于两者的因变量不同。多元线性回归直接将 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$ 作为因变量,即 $y=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$。而逻辑回归则不同,它是通过函数 L 将 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$ 对应一个隐状态 p,$p=L(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)$,然后根据 p 与 $1-p$ 的大小决定因变量的值。如果 L 是一个逻辑函数,那么其属于逻辑回归;如果 L 是一个多项式函数,那么其属于多项式回归。

逻辑回归大多用于分类,其原理就是由 sigmoid 函数处理线性模型的输出值,将输出值保持在(0,1)区间,用于二分类任务。它不仅可以预测出类别,还可以获得属于某个类别的概率预测。

逻辑回归属于一种用于解决监督学习(supervised learning)问题的学习算法,进行逻辑回归的主要目的是将训练数据的标签值与预测值之间的误差化为最小。分类是监督学习中的一个核心问题,当输出变量 Y 取有限个离散值时,预测问题便成为分类问题。此时的输入变量 x 既可以是离散的,也可以是连续的。分类器的功能是利用监督学习从数据中学习一个分类模型或分类决策函数。而分类就是分类器对新的输入对应的输出进行的预测。

例如,在现实生活中,如果要用数字表达某些事物是否出现,那么可以用 0 表示它不出现,用 1 表示它出现。能够用 0、1 表示事物状态的就称为二分类问题。如果需要在两个类别之间进行分类,那么逻辑回归的方法就非常适用。例如,一个班进行考试,有 10 道题,每道题 10 分。如果分数低于 60 分就是不及格,用 0 表示;否则是及格,用 1 表示。这个分类任务如图 5.1 所示。

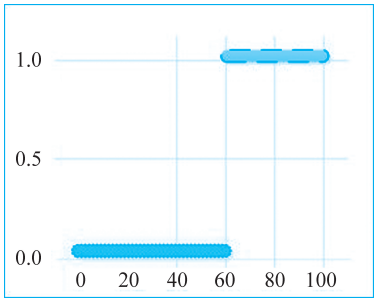


图 5.1　考试分数分类任务

5.1.2 逻辑回归应用

逻辑回归可以用于估计某个事件发生的可能性，也可以用于分析某个问题的影响因素有哪些。

目前，逻辑回归在数据挖掘、疾病自动诊断、经济预测等领域得到普遍应用。例如，分析一组胃癌病情案例，首先随机选择实验对象，并将其分为两组，一组是胃癌组，另一组是非胃癌组，两组人群必定在体征与生活方式等方面有所不同。因此，是否胃癌就作为因变量，值为“是”或“否”。而自变量有很多，包括年龄、性别、饮食习惯、幽门螺旋杆菌感染等。自变量既可以是连续的，也可以是离散的。最后通过逻辑回归分析得到自变量的权重，从而可以大致了解导致胃癌的危险因素有哪些，同时根据该权值判断出的危险因素预测一个人患胃癌的可能性。

对于二分类领域，可以用 sigmoid 函数得出概率值，适用于根据分类概率排名的领域，如搜索排名等。例如，在医学研究中，逻辑回归常用于分析某种疾病的危险因素，例如，分析年龄、性别、饮食习惯等是否属于胃癌的危险因素。

对于多分类领域，如手写字识别、信用评估、测量市场营销的成功度、预测某个产品的收益或亏损、预测特定的某天是否会发生地震或海啸，可以用逻辑回归扩展的 softmax 函数。其中，使用得最多的是二元逻辑回归分析，它不仅使用简单方便，而且容易描述和理解。因此本章以二元逻辑回归为例进行介绍。

5.2 概　　述

逻辑回归作为深度学习的基础，也是一种被广泛使用的统计模型和分类器，但它与深度学习的黑盒机制有很大不同，其解决问题的原理比较容易理解。逻辑回归是一个把线性回归映射为概率的模型，即把实数空间的输出映射到(0,1)区间，从而得到概率，其目的是解决类似“成功或失败”“有或无”或“通过或拒绝”等非此即彼的问题。

5.2.1 probability 和 odds 定义

probability 指的是发生的次数与总次数之比，例如抛硬币：

$$p=\frac{\text{正面向上的次数}}{\text{总次数}} \tag{5.1}$$

p 的取值范围为[0,1]。

odds 指的是发生的次数与没有发生的次数之比：

$$\text{odds}=\frac{\text{正面向上的次数}}{\text{反面向上的次数}} \tag{5.2}$$

odds 的取值范围为$[0,+\infty)$。

注意： odds 表示样例作为正例的相对可能性。

抛硬币出现正反面的事件服从伯努利分布。如果 X 是伯努利分布中的随机变量，则 X 的取值为{0,1}，非 0 即 1。例如，抛硬币出现正反面的概率分别为

$$P(X=1)=p$$

$$P(X=0)=1-p \tag{5.3}$$

相应的 odds 为

$$\text{odds}=\frac{p}{1-p},\quad \text{odds}\in[0,+\infty) \tag{5.4}$$

5.2.2　logit 函数和 sigmoid 函数及其特性

对 odds 取对数，扩展 odds 的取值范围到实数空间$[-\infty,+\infty]$，这就是 logit 函数：

$$\text{logit}(p)=\log_e\frac{p}{1-p},\quad p\in(0,1),\quad \text{logit}(p)\in[-\infty,+\infty] \tag{5.5}$$

注意：在后面的表述中，均省略 log 的底 e。

接下来，用线性回归模型表示 logit(p)，因为线性回归模型和 logit 函数的输出有同样的取值范围。

设

$$\text{logit}(p)=\theta_1x_1+\theta_2x_2+b \tag{5.6}$$

图 5.2 为 logit(p)的图形。注意，$p\in(0,1)$。

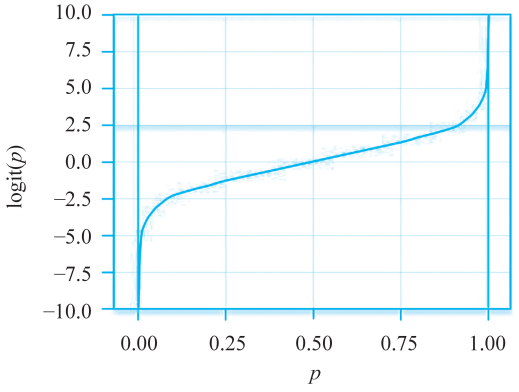


图 5.2　logit(p)的图形

由 $\text{logit}(p)=\theta_1x_1+\theta_2x_2+b$ 得

$$\log\frac{p}{1-p}=\theta_1x_1+\theta_2x_2+b \tag{5.7}$$

设 $\theta_1x_1+\theta_2x_2+b=z$，得

$$\log\frac{p}{1-p}=z \tag{5.8}$$

等式两边取 e 的幂，得

$$\frac{p}{1-p}=e^z \tag{5.9}$$

化简，得

$$p=e^z(1-p)$$
$$p=e^z-e^zp$$
$$p(1+e^z)=e^z$$

$$p=\frac{e^z}{1+e^z} \tag{5.10}$$

分子、分母同除以 e^z，得

$$p=\frac{1}{1+e^{-z}},\quad p\in(0,1) \tag{5.11}$$

由此得到 sigmoid 函数：

$$\text{sigmoid}(x)=\frac{1}{1+e^{-z}},\quad \text{sigmoid}(x)\in(0,1) \tag{5.12}$$

即可把线性回归模型输出的实数空间取值映射为概率。

从图 5.3 可以看到，sigmoid 函数是一个 S 形的曲线，自变量 x 无论取何值，其输出值都在[0,1]内。一个分类问题只有两种答案：一种是“是”，另一种是“否”。若 1 对应“是”，则 0 对应“否”。

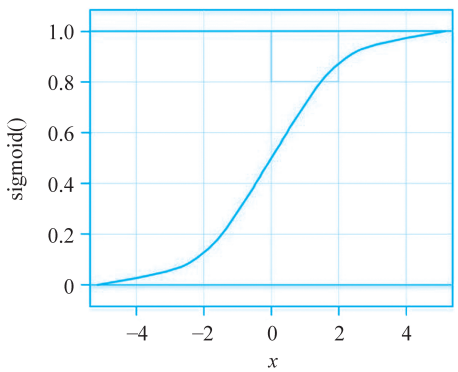


图 5.3　sigmoid 函数图像

阈值是可以自己设定的。假设分类的阈值是 0.5。记 $y=\text{sigmoid}(x)$。由图 5.3 可以看到，$x=0$ 是一个分界点。当 $x<0$ 时，$y<0.5$，sigmoid 函数输出为 0；当 $x>0$ 时 $y>0.5$，sigmoid 函数输出为 1。结果也可以理解为概率。换句话说，概率小于 0.5 的分类为 0，概率大于 0.5 的分类为 1，这就达到了分类的目的。

逻辑回归主要用于二元分类问题。对于给定的输入，在已知各项参数的情况下，就可以给出 0 或 1 的判断。例如，考试得了 60 分，通过逻辑回归函数就可以判断及格了。可见，在已知参数的情况下，逻辑回归函数根据输入可以给出 0 或 1 的结果。

5.2.3　最大似然估计

引入假设函数 $h_{\boldsymbol{\theta}}(\boldsymbol{X})$，设 $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{X}$ 为线性回归模型，$\boldsymbol{\theta}$ 和 $\boldsymbol{X}$ 均为列向量，例如：

$$\boldsymbol{\theta}=\begin{bmatrix}b\\ \theta_1\\ \theta_2\end{bmatrix},\boldsymbol{X}=\begin{bmatrix}1\\ x_1\\ x_2\end{bmatrix} \tag{5.13}$$

求 $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{X}$：

$$\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{X}=b\times 1+\theta_1\times x_1+\theta_2\times x_2=\theta_1x_1+\theta_2x_2+b \tag{5.14}$$

设 $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{X}=z$，则假设函数为

$$h_{\boldsymbol{\theta}}(\boldsymbol{X})=\frac{1}{1+e^{-z}}=P(Y=1\mid \boldsymbol{X};\boldsymbol{\theta}) \tag{5.15}$$

式(5.15)代表了 $Y=1$ 的概率。

$$1-h_{\boldsymbol{\theta}}(\boldsymbol{X})=P(Y=0\mid \boldsymbol{X};\boldsymbol{\theta}) \tag{5.16}$$

式(5.16)代表了 $Y=0$ 的概率。注意，$Y\in\{0,1\}$，Y 非 0 即 1。

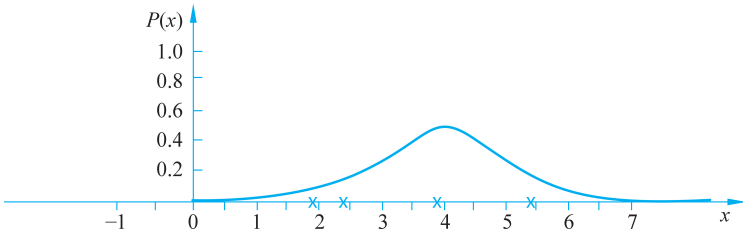
伯努利分布函数如下：

$$f(k;p)=p^k(1-p)^{1-k},k\in\{0,1\} \tag{5.17}$$

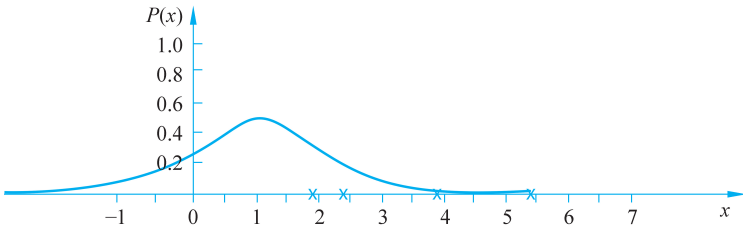
注意：$f(k;p)$表示 k 为 0 或 1 的概率，也就是 P_k。

最大似然估计的目的就是找到一个最符合数据的概率分布。

例如，图 5.4 中的×表示数据点，相应的概率的乘积就是似然函数的输出。显然，图 5.4(a)的似然函数值比图 5.4(b)的大，所以图 5.4(a)的分布更符合数据。而最大似然估计就是找到一个最符合当前数据的分布。



(a) $\boldsymbol{\theta}$=4



(b) $\boldsymbol{\theta}$=1

图 5.4　数据的概率分布

根据伯努利分布，定义似然函数：

$$\begin{aligned}L(\boldsymbol{\theta}\mid\boldsymbol{x})&=P(Y\mid\boldsymbol{X};\boldsymbol{\theta})\\&=\prod_{i=1}^{m}P(y_i\mid x_i;\boldsymbol{\theta})\\&=\prod_{i=1}^{m}h_{\boldsymbol{\theta}}(x_i)^{y_i}(1-h_{\boldsymbol{\theta}}(x_i))^{1-y_i}\end{aligned} \tag{5.18}$$

其中，x_i 为各个数据样本，共有 m 个数据样本。最大似然估计的目的就是让式(5.18)的输出值尽可能大。对式(5.18)取对数，以方便计算。取对数可以把乘积转换为加法，而且不影响优化目标：

$$\begin{aligned}L(\boldsymbol{\theta}\mid\boldsymbol{x})&=\log(P(\boldsymbol{Y}\mid h_{\boldsymbol{\theta}}(\boldsymbol{X})))\\&=\sum_{i=1}^{m}(y_i\log(h_{\boldsymbol{\theta}}(x^{(i)}))+(1-y_i)\log(1-h_{\boldsymbol{\theta}}(x^{(i)})))\end{aligned} \tag{5.19}$$

只要在式(5.19)前面加一个负号，即可把求最大值转换为求最小值。设 $h_{\boldsymbol{\theta}}(\boldsymbol{X})=\hat{\boldsymbol{Y}}$，得出损失函数 $J(\boldsymbol{\theta})$。只要最小化这个函数，就能通过求导得到 $\boldsymbol{\theta}$：

$$J(\boldsymbol{\theta})=-\sum_{i=1}^{m}(\boldsymbol{Y}\log\hat{\boldsymbol{Y}}-(1-\boldsymbol{Y})\log(1-\hat{\boldsymbol{Y}})) \tag{5.20}$$

深度学习中的交叉熵和式(5.20)一样，只不过式(5.20)是交叉熵中的项分类问题。对于多分类，可对 $J(\boldsymbol{\theta})$进行推广，获得交叉熵(cross entropy)：

$$\text{CrossEntropy}(\boldsymbol{Y},\hat{\boldsymbol{Y}}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{c=1}^{N_c}\boldsymbol{Y}\log\hat{\boldsymbol{Y}} \tag{5.21}$$

其中，c 为分类编号，N_c 为所有的分类数量。

5.2.4 参数的获取：梯度下降法优化参数

根据式(5.20)画出对数似然(log likelihood)函数，也就是损失函数的图形，如图 5.5 所示。

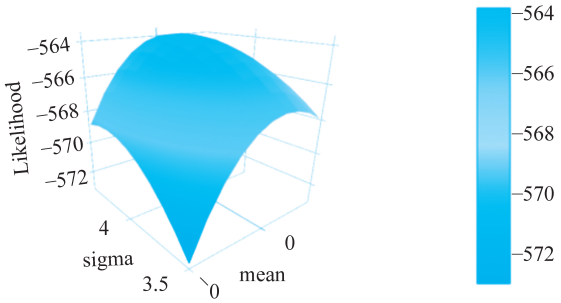


图 5.5 损失函数的图形

注意：损失函数要对原始的对数似然函数取负号，使极大似然值的最大化问题变成最小化问题。

如图 5.5 所示，损失函数是一个凸函数。使用梯度下降法寻找损失函数的极小值，即求出当 $\boldsymbol{\theta}$ 取什么值时损失函数可以达到极小值。

1. 求导过程

需要求 $J(\boldsymbol{\theta})$ 对 θ_j 的导数 $\frac{\partial J(\boldsymbol{\theta})}{\partial\theta_j}$，注意：

$$\hat{\boldsymbol{Y}} = \frac{1}{1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}x}} \tag{5.22}$$

利用

$$\frac{\mathrm{d}}{\mathrm{d}x}\log_a f(x) = \frac{1}{f(x)\ln a}f'(x) \tag{5.23}$$

结合式(5.23)，将式(5.22)代入式(5.20)，可得

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\log(1-\hat{\boldsymbol{Y}}) &= \frac{\partial}{\partial\theta_j}\log\frac{1}{1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}x}}\\ &= \frac{\partial}{\partial\theta_j}(\log 1-\log(1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}x}))\\ &= \frac{\partial}{\partial\theta_j}(-\log(1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}x}))\\ &= -\frac{1}{1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}x}}\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}x}(-x_j)\\ &= \left(1-\frac{1}{1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}x}}\right)x_j\end{aligned} \tag{5.24}$$

再求$\frac{\partial}{\partial\theta_j}\log(1-\hat{\boldsymbol{Y}})$：

$$\frac{\partial}{\partial\theta_j}\log(1-\hat{\boldsymbol{Y}})=\frac{\partial}{\partial\theta_j}\log\frac{\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}x}}{1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}x}}$$

$$=\frac{\partial}{\partial\theta_j}(-\boldsymbol{\theta}^{\mathrm{T}}x-\log(1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}x})) \tag{5.25}$$

将式(5.25)代入式(5.24)，得

$$\frac{\partial}{\partial\theta_j}\log(1-\hat{\boldsymbol{Y}})=-x_j+x_j\left(1-\frac{1}{1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}x}}\right)$$

$$=-\frac{1}{1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}x}}x_j \tag{5.26}$$

将利用式(5.26)求得的$\frac{\partial}{\partial\theta_j}\log(1-\hat{\boldsymbol{Y}})$和$\frac{\partial}{\partial\theta_j}\log(\hat{\boldsymbol{Y}})$代入$\frac{\partial J(\boldsymbol{\theta})}{\partial\theta_j}$(注意，$i$ 是数据点的序号，j 是特征的数量)：

$$\boldsymbol{X}=\begin{bmatrix}x_1^{(1)} & x_2^{(1)} & \cdots & x_m^{(1)}\\ x_1^{(2)} & x_2^{(2)} & \cdots & x_m^{(2)}\\ x_1^{(3)} & x_2^{(3)} & \cdots & x_m^{(3)}\end{bmatrix} \tag{5.27}$$

展开并整理得(注意，$\hat{\boldsymbol{Y}}=\frac{1}{1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}x}}$)

$$\frac{\partial J(\boldsymbol{\theta})}{\partial\theta_j}=-\sum_{i=1}^{m}y^{(i)}x_j^{(i)}\left(1-\frac{1}{1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}^{(i)}}}\right)-(1-y^{(i)})x_j^{(i)}\frac{1}{1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}^{(i)}}}$$

$$=\sum_{i=1}^{m}\left(\frac{1}{1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}^{(i)}}}-y^{(i)}\right)x_j^{(i)}$$

$$=\sum_{i=1}^{m}(\hat{y}^{(i)}-y^{(i)})x_j^{(i)} \tag{5.28}$$

从式(5.13)可以发现 $\boldsymbol{\theta}$ 中的 b 对应 $\boldsymbol{X}$ 中的 1，由此可得

$$\frac{\partial J(\boldsymbol{\theta})}{\partial b}=\sum_{i=1}^{m}(\hat{y}^{(i)}-y^{(i)}) \tag{5.29}$$

2. 搜寻下山过程

现在已经得到了 θ_j 和偏差 b 的梯度，如果用这个梯度对参数进行更新，就要定义学习率 η，防止下山的时候跑得太快而跑过头。一般学习率的取值都比较小，然后重复下面的步骤，直到收敛：

$$\theta_j\leftarrow\theta_j-\eta\frac{\partial J(\boldsymbol{\theta})}{\partial\theta_j} \tag{5.30}$$

$$b\leftarrow b-\eta\frac{\partial J(\boldsymbol{\theta})}{\partial b} \tag{5.31}$$

5.2.5　模型评估方法

通常来说，如果对学习器的泛化误差进行评估并进而做出选择，即可通过实验测试获取结果。因此，需使用一个测试集(testing set)测试学习器对新样本的判别能力，然后将测试

集上的测试误差(testing error)近似为泛化误差。通常假设测试样本也是从样本真实分布中独立同分布采样而得的。但要注意一点,测试集应该尽可能与训练集互斥,即训练集中不能出现测试样本,且不能在训练过程中使用过。

训练集中为什么不能出现测试样本呢?为理解这一点,以下面的场景为例加以说明:老师给同学们出了10道练习题,考试时老师的试题是同样的这10道题,那么这个考试成绩能否真实、有效地反映同学们的学习情况呢?答案是否定的,可能有的同学只是掌握了这10道题,不会其他知识,却能轻易得高分。希望得到泛化能力强的模型,就像希望同学们把课程学好,并能够将所学知识举一反三一样。训练样本相当于给同学们出练习题,测试过程就是课程考试。很明显,如果用测试样本进行训练,则得到的估计结果一定是过于"乐观"的。

可是,我们只有一个包含 m 个样本的数据集 $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$,怎样才能做到既能训练又能测试呢?适当地处理数据集 D,从中产生训练集 S 和测试集 T。下面介绍几种常见的做法。

1. 留出法

留出法(hold-out)直接将数据集 D 划分为训练集 S 和测试集 T 两个互斥的集合,即 $D=S\cup T$,$S\cap T=\varnothing$。在 S 上训练出模型后,用 T 评估其测试误差,作为对泛化误差的估计。

以二分类任务为例,假定 D 包含1000个样本,将其划分为 S(包含700个样本)和 T(包含300个样本)。用 S 进行训练后,如果模型在 T 上有90个样本分类错误,那么其错误率为(90/300)×100%=30%,相应的精度为1−30%=70%。

在这里要注意的一个问题是,训练集与测试集的划分要尽可能使数据分布保持一致,避免因数据划分过程引入额外的偏差而影响最终结果,例如在分类任务中至少要保持样本的类别比例相似。如果从采样(sampling)的角度看待数据集的划分过程,那么保留类别比例的采样方式常常被叫作分层采样(stratified sampling)。例如,通过对 D 进行分层采样而获得含70%样本的训练集 S 和含30%样本的测试集 T,若 D 包含500个正例、500个反例,则分层采样得到的 S 应包含350个正例、350个反例,而 T 则包含150个正例和150个反例;若 S 和 T 中样本类别比例差别很大,则误差估计将由于训练数据与测试数据分布的差异而产生偏差。

另一个需注意的问题是,即便有了训练集和测试集的样本比例,仍存在多种分割初始数据集 D 的方式。如上所述,可以首先对 D 中的样本进行排序,然后在训练集中放入前350个正例,或者放入后350个正例。这些不同的划分将导致不同的训练集和测试集,模型评估会产生不同结果。因此,单次使用留出法得到的估计结果是不够稳定的。在使用留出法时,一般要进行若干次随机划分,重复进行实验评估后取平均值作为留出法的评估结果。例如,进行100次随机划分,每次产生一个训练集和一个测试集用于实验评估,100次后就得到100个结果,而留出法返回的则是这100个结果的平均值。

另外,我们希望将 D 训练出的模型用于评估,但留出法需要分出训练集和测试集,这就会产生一个尴尬的情况:若令绝大多数样本都在训练集 S 中,则训练出的模型可能更接近用 D 训练出的模型,然而因为测试集 T 比较小,最后的评估结果就会不够精准且变化较大。若让一些样本包含于测试集 T 里,则训练集 S 与 D 差别就会更大,被评估的模型与用

D 训练出的模型相比可能有较大差别，从而降低了评估结果的保真性(fidelity)。目前为止，这个问题还没有完美的解决方案，最普遍的做法是将 2/3～4/5 的样本用于训练，其余样本用于测试。

2. 交叉验证法

交叉验证法(cross validation)是先将数据集 D 划分为大小相似的互斥子集，即 $D=D_1 \cup D_2 \cup \cdots \cup D_k$，$D_i \cap D_j = \varnothing (i \neq j)$，互斥子集总共有 k 个。保证所有子集具有相同的数据分布，即它们都是从 D 中通过分层采样得到的。然后，每次用 $k-1$ 个子集的并集作为训练集，余下的那个子集作为测试集；最终得到 k 组训练集和测试集，从而可进行 k 次训练和测试，最终返回的是这 k 个测试结果的平均数。显然，k 的取值在很大程度上决定了交叉验证法评估结果的稳定性和保真性，为强调这一点，通常交叉验证法也被称为 k 折交叉验证(k-fold cross validation)。10 是 k 最常用的取值，此时称为 10 折交叉验证。其他常用的 k 值有 5、20 等。图 5.6 为 10 折交叉验证。

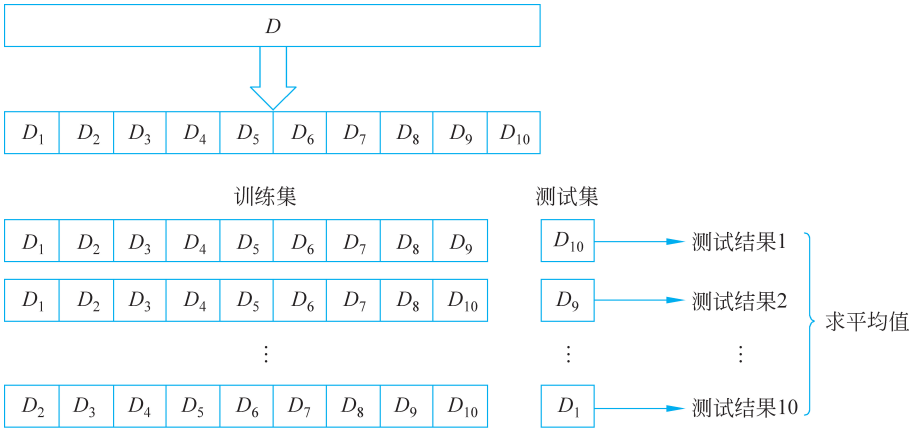


图 5.6　10 折交叉验证

与留出法相似，将数据集 D 划分为 k 个子集也有多种方式。为减小因划分为不同子集而引入的差别，k 折交叉验证通常要随机使用不同的划分重复 p 次，取这 p 次 k 折交叉验证结果的均值作为最终评估结果。

假定数据集 D 中包含 m 个样本，若令 $k=m$，则得到了交叉验证法的一个特例：留一法(Leave-One-Out，LOO)。很明显，留一法不会受随机样本划分方式的影响，因为 m 个样本有且仅有一个方式划分为 m 个子集，即每个子集包含一个样本，留一法使用的训练集只比初始数据集少了一个样本，这就使得在绝大多数情况下留一法中被实际评估的模型与期望评估中用 D 训练出的模型很相似。因此，留一法的评估结果往往被认为比较准确。然而，留一法也存在不足，在数据集比较大的情况下，训练 m 个模型的计算量可能是极大的(例如数据集包含 100 万个样本，则需训练 100 万个模型)，而这还未考虑算法调参的情况。另外，留一法的评估结果也不能保证永远比其他评估方法准确。

3. 自助法

我们希望评估的是用 D 训练出来的模型。但在留出法和交叉验证法中，为了用于测试而保留了一部分样本，因此实际评估的模型所使用的训练集比 D 小，这必然会出现因训练样本规模不同而导致的估计偏差。尽管留一法受训练样本规模变化的影响较小，但计算复

杂度太高。那么，有什么办法可以使结果不受训练样本规模不同的影响，同时还能比较高效地进行实验评估呢？

目前，一个比较好的解决方案就是自助法（bootstrapping）。它直接以自助采样法（bootstrap sampling）为基础。给定数据集 D，令其包含 m 个样本，对它进行采样产生数据集 D'：每次随机从 D 中挑选一个样本，将其复制后放入 D'，然后再将该样本放回初始数据集 D 中，使得该样本在下次采样时仍然有可能被选中。这个过程重复执行 m 次后，就得到了自助采样的结果，即一个包含 m 个样本的数据集 D'。显然，D 中有一部分样本很可能在 D' 中多次出现，而另一部分样本很可能不出现。可以做一个简单的估计，样本在 m 次采样中始终不被选中的概率是 $\left(1-\frac{1}{m}\right)^m$，取极限得

$$\lim_{m\to\infty}\left(1-\frac{1}{m}\right)^m \mapsto \frac{1}{\mathrm{e}} \approx 0.368 \tag{5.32}$$

通过自助采样，初始数据集 D 中约有 36.8%的样本未出现在采样数据集 D' 中。于是可将 D' 用作训练集，将 $D\backslash D'$ 用作测试集。这样，实际评估的模型与期望评估的模型都使用 m 个训练样本，同时仍有数据总量约 1/3 未在训练集中出现的样本用于测试。这样的测试结果也称包外估计（out-of-bag estimate）。

自助法常用于数据集较小，难以有效划分训练集和测试集的情况。此外，自助法能从初始数据集中产生多个不同的训练集，这非常有利于集成学习等方法。但是，自助法产生的数据集改变了初始数据集的分布，这会引入估计偏差。因此，在数据量足够时，更常用的还是留出法和交叉验证法。

5.2.6 调参与最终模型

诸多学习算法都需要设定一些参数。不同的参数配置，其模型的性能往往有显著差别。因此，在评估与选择模型时，除了要选择适用的学习算法，还需要设定算法参数，这就是通常所说的参数调节，简称调参。

有人可能会认为，调参和算法选择在本质上没有太大的区别。将每种参数配置都训练出模型，然后把对应最好的模型的参数作为结果。这样的考虑基本上是对的，但要注意一点，学习算法的很多参数都在实数范围内取值。因此，为每种参数配置都训练出模型的可能性很小。一般的做法是对每个参数选定一个范围和变化步长。例如，在[0，0.2]范围内以 0.05 为步长，则有 5 个实际要评估的候选参数值，最终的选定值就是从这 5 个候选值中产生的。很明显，从这种方法选出的参数值往往不是最佳的，但这个结果是计算开销和性能估计的折中，这样学习过程才变得可行。事实上，即便在这样的折中后，调参依旧存在困难。可以简单估算一下：假定算法有 3 个参数，每个参数仅考虑 5 个候选值，这样对每一组训练集和测试集就有 $5\times5\times5=125$ 个模型需考察。很多强大的学习算法有大量参数需设定，这将导致极大的调参工作量，因此，在很多应用任务中，参数的优劣决定了最终模型的性能。

还需要注意的是，通常把学习得到的模型在实际使用中遇到的数据称为测试数据。为了更好地与之区别，模型评估与选择中用于评估测试的数据集常称为验证集（validation set）。举例来说，如果要研究对比不同算法的泛化能力，估计模型在实际使用时的泛化能力就要用测试集上的判别效果，并将训练数据另外划分为训练集和验证集，在利用验证集得到