第六章 分类分析

分类分析（Classification Analysis）是一种重要的数据分析形式。人们通过找出一组数据的共同特点，按照一定的模式建立分类模型，实现对未知数据的类别预测，如对仪器仪表的故障类型、医疗数据对应的治疗方案、基于运营商数据的个人征信、欺诈信息等进行预测。

本章首先描述分类分析的基本概念，着重介绍经典的分类方法，包括决策树、朴素贝叶斯、支持向量机、人工神经网络等算法。同时针对不同分类模型的评估指标与选择方法展开讨论，以验证分类模型的有效性。最后对一些应用较为广泛的组合分类技术进行介绍。

6.1 分类分析的基本概念

解决分类问题的方法很多，分类器的任务都是实现对未知样本所属类别的预测，不同分类算法主导的分类过程存在一些共通性。下面对分类分析的基本原理及其一般过程进行介绍。

6.1.1 分类的基本原理

与聚类不同，分类是一种有监督学习。给定样本数据，其中包含各属性值及其对应的类标签（即样本所属类别），对样本数据进行观测与分析，选择合适的分类方法构造分类模型，使用分类模型预测未知样本数据的类标签。

给定某一样本数据集 $D=\{t_1,t_2,\ldots,t_n\}$，其中样本 $t_i \subseteq D$，对应类标签集合 $C=\{C_1,C_2,\ldots,C_m\}$。分类问题定义为从数据集到类集合的映射 $f:D\to C$，即数据库中的样本 t_i 分配到某个类 C_j 中，有 $C_j=\{t_i \mid f(t_i)=C_j,1\le i\le n,\text{且}t_i\in D\}$。

数据分类的过程总体可分作三个阶段：模型构建、模型测试和模型应用。

（1）模型构建

在模型构建阶段，建立描述预先定义的数据类或概念集的分类模型。用来建模的样本（数据、对象、案例或记录）集合称为**训练集**（Training Set），分类算法通过分析或从训练集学习来构造分类模型，分类模型可用数学公式、决策树等表示。这一步通常称为“训练”。

（2）模型测试

在模型构建后，需要对上述分类模型的预测准确率进行评估。由于训练过程中分类模型趋向于过拟合数据，若继续使用训练集来度量模型的准确度，其结果往往过于乐观，难以准确评估。

因此，使用独立于训练集的**测试集**（Test Set）参与模型准确率的评估，将分类模型预测的类标签与真实的类标签比较，若相同则预测正确。最终，计算分类模型在给定检验集上的准确率

（3）模型应用

在模型测试之后，使用分类模型对未知样本进行分类。

一个完整的分类过程如图 6.1 所示：

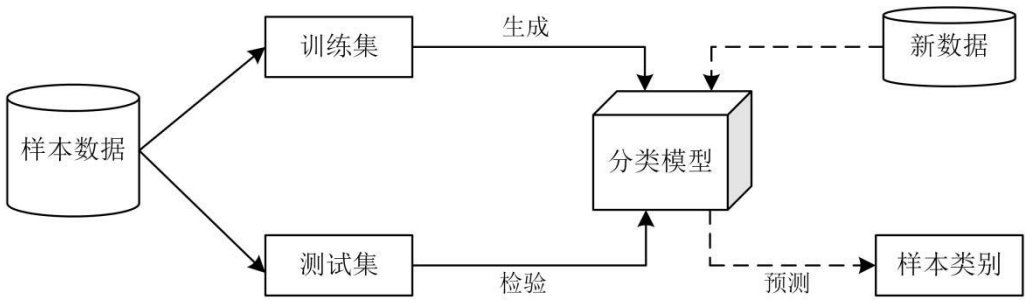


图 6.1 分类分析的一般过程

6.1.2 主要分类方法

在分类问题中，有很多构造分类模型的方法。较常用的方法有：决策树、朴素贝叶斯、支持向量机、人工神经网络等。上述分类方法的具体内容将在后续章节中作详细介绍。

（1）决策树（Decision Tree）

决策树是一种类似于流程图的树结构，它采用自顶而下的递归方式，经过一批训练集的训练生成一棵决策树。决策树在根结点和各内部结点上选择合适的分裂属性，根据该属性的不同取值向下建立分支。决策树分类不单具有较高的准确率，且简单易学，不要求使用者了解很多背景知识。

（2）朴素贝叶斯（Naïve Bayes）

朴素贝叶斯分类是统计学分类方法，它可以预测某个给定样本属于一个特定类的概率。如果支持某项属性的事件发生得越多，则该属性成立的可能性就愈大。对于给出的待分类项，求解在此条件下各个类别出现的概率，出现概率最大的类别即为预测类别。朴素贝叶斯分类模型需要的参数较少，往往具有高准确率和很快的速度。

（3）支持向量机（Support Vector Machine，SVM）

支持向量机一种经典的机器学习方法。通过寻找一个超平面作为两类训练样本点的分割，SVM 寻找最大间隔分类器，保证最小的分类错误率。最佳超平面的寻找问题可转化为一个含约束的优化问题，通过优化算法解决。支持向量机在解决小样本、非线性及高维模式识别中表现出许多特有的优势。

（4）人工神经网络（Artificial Neural Networks，ANN）

人工神经网络是一种类似于大脑神经突触连接的结构，训练重点是构造逻辑单元并反复调整权重系数。将训练集中的样本输入到网络中，根据当前输出结果和理想输出之间的差值来调整网络中的权重值，直至达到相应的终止条件。人工神经网络具有高度的自组织和自学习能力，但易陷入局部最优解。

6.2 决策树

决策树最早产生于 20 世纪 60 年代，是用于分类和回归的主要技术之一，也是应用最广泛的逻辑方法之一。决策树是以实例为基础的归纳学习算法，着眼从一组无次序、无规则的数据中推理出以树形结构表示的分类规则。一般而言，决策树具有分类精度高，操作简单等优点。

6.2.1 决策树的基本原理

决策树是一种描述对实例进行分类的树形结构，包含三类节点：根节点、内部节点和叶节点。其中，根节点包含所有样本，位于顶层。每个内部节点对应着按某种分类规则划分出的各类样本子集，位于中间。叶节点对应着分类结果，位于底层。

决策树分类采用自顶而下的递归方式：训练过程中，由根节点向下划分生成内部节点，每个内部节点向下继续划分。具体如何划分以及是否划分取决于当前节点在某个特征属性上的分类结果，最终在决策树的叶节点得到分类结论。从决策树的根节点到某个叶节点的路径对应着一条分类规则，整个决策树对应着一组规则。

最早被提出的决策树算法是 CLS 算法，它确立了决策树“**分而治之**”的基本训练流程：

1）由训练数据集生成根节点；

2）为当前节点选择某一分类属性作为划分依据；

3）根据当前节点属性不同的取值，将训练集划分为若干子集，每个取值形成一个分支。针对当前划分的若干个子集，重复步骤 2）~ 3）；

4）达成下列条件之一时，停止划分，将该节点标记为叶节点：①节点的所有样本属于同一类；②没有剩余属性；③如果某一分支没有样本，则以该节点中占大多数的样本类别创建一个叶节点；④决策树深度已达到设定的最大值。

例 6.1 在电动汽车中，电池组中所有单体之间的一致性问题是影响动力电池安全性的关键因素，有必要对汽车电池包内的每个单体进行筛查并确定其是否异常。以某批电池检测样本为例，利用三个充放电周期实验的充放电电压数据构造了分类属性 A、B、C。计算电池电压曲线与电压平均值曲线的差值，再对所有时刻的数据求平均值，归一化得到属性 A。计算电池所有时刻电压曲线与电压平均值

曲线差值的绝对值之和，归一化得到属性 B。计算电池所有时刻电压的平均值，归一化得到属性 C。

实验所得的训练数据集 D 如表 6.1 所示。

属性 $A=\{a_1,a_2,a_3\}$，其中 $a_1(>0.06)$、$a_2(0.04\sim0.06)$、$a_3(<0.04)$；

属性 $B=\{b_1,b_2,b_3\}$，其中 $b_1(>0.77)$、$b_2(0.65\sim0.77)$、$b_3(<0.65)$；

属性 $C=\{c_1,c_2,c_3\}$，其中 $c_1(>0.18)$、$c_2(0.12\sim0.18)$、$c_3(<0.12)$。

试用 CLS 算法构造决策树。

表 6.1 电池故障实验数据

序号	属性 A	属性 B	属性 C	是否异常
1	> 0.06	> 0.77	> 0.18	是
2	> 0.06	0.65~0.77	> 0.18	是
3	< 0.04	< 0.65	< 0.12	否
4	0.04~0.06	> 0.77	0.12~0.18	是
5	0.04~0.06	0.65~0.77	< 0.12	否
6	0.04~0.06	< 0.65	< 0.12	否
7	< 0.04	0.65~0.77	> 0.18	是
8	> 0.06	0.65~0.77	0.12~0.18	是
9	< 0.04	< 0.65	0.12~0.18	否
10	0.04~0.06	> 0.77	> 0.18	是
11	< 0.04	0.65~0.77	< 0.12	否
12	0.04~0.06	0.65~0.77	0.12~0.18	是
13	0.04~0.06	0.65~0.77	> 0.18	是
14	> 0.06	< 0.65	0.12~0.18	否
15	< 0.04	> 0.77	0.12~0.18	是

由上述训练数据集生成根节点，位于顶端，在属性集中选择某一属性作为根节点的划分依据。CLS 算法并没有明确指出按照什么样的标准选择属性，这里不妨选择属性 B。按属性 B 的不同取值可将训练集 D 划分为三个不同子集 D_{b1}、D_{b2}、D_{b3}。其中子集 D_{b1} 中只包含一种类别的样本，子集 D_{b3} 同理，这两个子集不再向下继续划分。子集 D_{b2} 数据如表 6.2 所示：

表 6.2 属性 B 取值为 0.65~0.77 的样本子集

序号	属性 A	属性 B	属性 C	是否异常
1	> 0.06	0.65~0.77	> 0.18	是
2	0.04~0.06	0.65~0.77	< 0.12	否
3	< 0.04	0.65~0.77	> 0.18	是

4	> 0.06	0.65~0.77	0.12~0.18	是
5	< 0.04	0.65~0.77	< 0.12	否
6	0.04~0.06	0.65~0.77	0.12~0.18	是
7	0.04~0.06	0.65~0.77	> 0.18	是

子集D_{b2}仍包含两种类别的样本，故应继续选择其它属性对子集D_{b2}进行划分。可选择属性A将其划分为三个不同子集V_{a1}、V_{a2}、V_{a3}如表 6.3 所示

表 6.3 由属性 A 继续划分的样本子集

序号	属性 A	属性 B	属性 C	是否异常
1	> 0.06	0.65~0.77	> 0.18	是
2	> 0.06	0.65~0.77	0.12~0.18	是

序号	属性 A	属性 B	属性 C	是否异常
1	0.04~0.06	0.65~0.77	< 0.12	否
2	0.04~0.06	0.65~0.77	0.12~0.18	是
3	0.04~0.06	0.65~0.77	> 0.18	是

序号	属性 A	属性 B	属性 C	是否异常
1	< 0.04	0.65~0.77	> 0.18	是
2	< 0.04	0.65~0.77	< 0.12	否

子集V_{a1}中只含一种类别，无须继续划分。子集V_{a2}、V_{a3}需要按属性C继续划分。这里以V_{a3}为例，它可继续划分为两个不同子集如表 6.4 所示

表 6.4 由属性 C 继续划分的样本子集

序号	属性 A	属性 B	属性 C	是否异常
1	< 0.04	0.65~0.77	> 0.18	是

序号	属性 A	属性 B	属性 C	是否异常
1	< 0.04	0.65~0.77	< 0.12	否

此时每个子集均只含一种类别，无法继续划分，故该节点停止向下分裂，对其余节点同理。最终可生成决策树如图 6.2 所示。

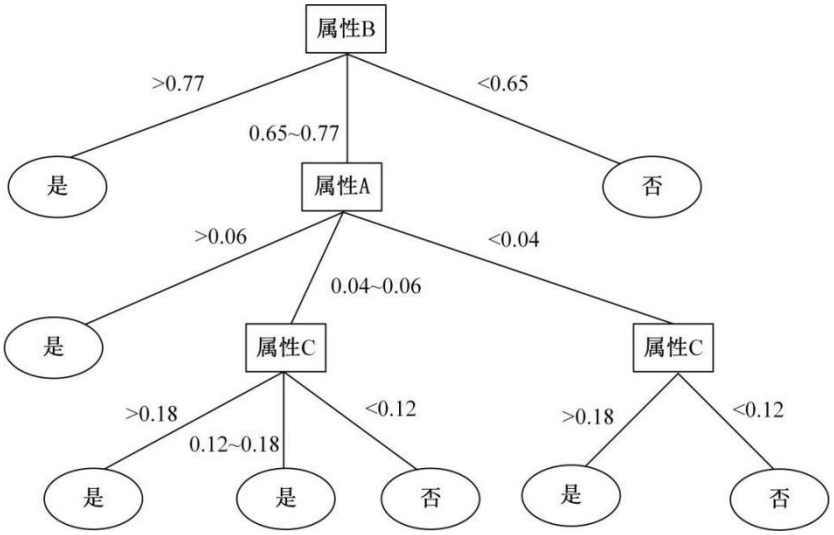


图 6.2 CLS 算法生成的决策树

图 6.2 所示决策树可以用如下的 IF-THEN 分类规则描述：

IF $B > 0.77$ THEN 电池异常

IF $0.65 \le B \le 0.77$ AND $A>0.06$ THEN 电池异常

IF $0.65 \le B \le 0.77$ AND $0.04 \le A \le 0.06$ AND $C > 0.18$ THEN 电池异常

IF $0.65 \le B \le 0.77$ AND $0.04 \le A \le 0.06$ AND $0.12 \le C \le 0.18$ THEN 电池异常

IF $0.65 \le B \le 0.77$ AND $0.04 \le A \le 0.06$ AND $C < 0.12$ THEN 电池正常

IF $0.65 \le B \le 0.77$ AND $A<0.04$ AND $C>0.18$ THEN 电池异常

IF $0.65 \le B \le 0.77$ AND $A<0.04$ AND $C<0.12$ THEN 电池正常

IF $B<0.65$ THEN 电池正常

由例 6.1 可以发现，最终生成的决策树较为繁杂。实际上除叶节点外，其余每一节点的属性选择对决策树的生成和分类性能影响较大。而 CLS 算法没有确切的属性选择标准，有较大的改进空间。

针对此问题，后续许多决策树算法引入**属性选择度量**，属性选择度量又称为分支指标，是选择当前节点最优分支属性的准则。

6.2.2 属性选择度量

本节主要介绍以下三种常见度量：信息增益（Information Gain）、增益率（Gain Ratio）和基尼指数（Gini Index）。

假设当前样本集合 D 包含 n 个类别的样本 $C_1, C_2, \cdots C_n$，其中第 k 类样本 C_k 在所有样本中出现的频率为 p_k。属性 A 将 D 划分成 m 份，D_i 表示 D 的第 i 个子集，$|D|$ 和 $|D_i|$ 分别表示 D 和 D_i 中的样本数量。

（1）信息增益

信息增益由熵（Entropy）值的变化来确定。熵是信息论中的一个概念，用于刻画随机变量不确定性的大小。

样本 D 的信息熵定义为

$$Ent(D) = Ent(p_1, p_2, \cdots, p_n) = -\sum_{k=1}^{n} p_k \log_2 p_k \tag{6.1}$$

$Ent(D)$ 的值越小，则样本 D 的不确定性越小，即样本的纯度越高。决策树的分支原则是使划分后的样本子集纯度越高越好，或者说熵越小越好。

条件熵是指在特定条件下，随机变量的不确定性。样本集合 D 在属性 A 划分的条件下，子集的熵定义为

$$Ent(D, A) = -\sum_{i=1}^{m} \frac{|D_i|}{|D|} Ent(D_i) \tag{6.2}$$

划分前后样本数据集的熵的差值称为信息增益，信息增益可以用来衡量熵的期望减小值。用属性 A 对样本集 D 进行划分后所获得的信息增益为

$$Gain(D, A) = Ent(D) - Ent(D, A) \tag{6.3}$$

$Gain(D, A)$ 越大，说明划分后熵的减少量越大，节点就趋向于更纯，越有利于分类。因此，应选择获得最大信息增益的属性作为分支属性。

（2）增益率

增益率定义为

$$Gain\_ratio(D, A) = \frac{Gain(D, A)}{SplitInfo(D, A)} \tag{6.4}$$

其中，*SplitInfo(D,A)*的计算方式如下：

$$SplitInfo(D, A) = -\sum_{i=1}^{m} \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|} \tag{6.5}$$

*SplitInfo(D,A)*反映属性 A 的纯度，A 的取值越少，A 的纯度就越高，*Ent(D,A)* 的值也就越小，因此最后得到的信息增益率也就越高。

（3）基尼指数

基尼指数度量数据分区或样本数据集 D 对所有类别的不纯度，定义为：

$$Gini(D) = \sum_{k=1}^{|n|} \sum_{k' \neq k} p_k p_{k'} = 1 - \sum_{k=1}^{|n|} p_k^2 \tag{6.6}$$

基尼指数反映了从数据集中随机抽取的样本其类别标志不一致的概率。*Gini(D)*越小，则数据集 D 的纯度越高，反之，*Gini(D)*越大，则数据集 D 的纯度越低。

类似地，定义在属性 A 下数据集的基尼指数为

$$Gini_A(D) = \frac{x_1}{N} Gini(A_1) + \frac{x_2}{N} Gini(A_2) + \cdots + \frac{x_m}{N} Gini(A_m) \tag{6.7}$$

式中 $A_1, A_2, \cdots, A_m$ 表示属性 A 的 m 个不同取值，$x_1, x_2, \cdots, x_m$ 表示各种取值对应的样本数，x_{ij} 表示 x_i 个样本中类 j 所对应的样本数。$Gini(A_i)$ 的定义如下：

$$Gini(A_i) = 1 - \sum_{j=1}^{k} \left(\frac{x_{ij}}{x_i} \right)^2 \tag{6.8}$$

基尼指数越小表明该属性越适合作为分支的属性。

除了上述三类属性选择度量，还有其他一些度量方法。例如基于统计 χ^2 检验的 CHAID 算法、基于 G 统计量的算法、基于最小描述长度（Minimum Description Length）原理的属性选择量度等。目前尚未发现其中某一种显著优于其他度量方法。

6.2.3 树剪枝

在决策树生成时，受样本数据中噪声点的影响，决策树容易**过拟合**，其预测准确度也会降低。剪枝是用测试集数据对决策树进行检验、校正和修正的方法之一，常用于处理这种过拟合问题。

常用的剪枝方法有两种，分别是**前剪枝**（Pre-pruning）和**后剪枝**（Post-pruning）。

（1）前剪枝

在决策树的生长过程中，根据某些测试条件决定是否继续对不纯的训练子集进行划分。一般会提前设定一个指标或阈值，当决策树达到预定的条件时就停止划分，当前的节点就成为叶节点。

前剪枝的主要方法有参数控制法和分裂阈值法。参数控制法利用某些参数（例如节点的大小、树的深度等）限制树的生长。分裂阈值法通过设定一个分裂阈值，当分裂后的信息增益不小于该阈值，才保留分支，否则停止分裂。

前剪枝可以有效减小决策树的规模及计算量，但也可能导致生成决策树的不纯度增大。此外，如何设置指标是前剪枝的核心问题，在很多情况下，选择恰当的指标是比较困难的。

（2）后剪枝

后剪枝首先需要决策树充分生长，直到叶节点都有最小的不纯度值为止，然后在树的主体中删除一些不必要的子树。后剪枝由叶节点开始逐步向根节点方向逐层进行，一边修剪一边使用验证集检验决策树的准确度。

如果删除某个节点的子节点后，决策树的准确率（或其它评价指标）并没有降低，那么就将该节点变为叶节点。产生一系列修剪过的决策树候选之后，利用测试数据对各候选决策树进行评价，保留分类错误率最小的决策树。

后剪枝虽然可以使树得到更充分的生长，但计算量大且复杂。两种剪枝技术各有利弊，在实际应用中也可以交叉使用。目前并未发现某一种剪枝技术显著优于其他技术。

6.2.4 决策树算法

综合前述内容，决策树的训练通常分为属性选择、决策树生成，剪枝三个步骤。决策树的基本算法如下：

输入：数据集 $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$，属性集 $A=\{a_1,a_2,\cdots,a_d\}$

输出：以 node 为根节点的一棵决策树

---

1. 生成根节点 node；
2. if D 中样本全属于同一类别 C then
3. 将 node 标记为 C 类叶节点；return
4. end if
5. if D 中样本在 A 上取值相同 then
6. 将 node 标记为叶节点,类别标记为 D 中样本数最多的类；return
7. end if
8. 从属性集 A（或剩余属性）中选择最优划分属性 a_*；
9. for a_* 的每一个值 a_*^v do
10. 为 node 生成一个分支；令 D_v 表示 D 中在 a_* 上取值为 a_*^v 的样本子集；
11. if D_v 为空 then
12. 将其标记为叶节点，类别标记为 D 中样本数最多的类；return
13. else
14. 以 TreeGenerate(D_v, $A\setminus\{a_*\}$)为分支节点
15. end if
16. end for

显然，决策树算法的关键是如何选择最优划分属性。然而在实际应用中，前述的 CLS 算法未指明节点属性选择的依据，分类的主观性较强。后来在其基础上形成了一系列改进算法，目前较常用的决策树算法有 ID3 算法、C4.5 算法以及 CART 算法。

（1）ID3 算法

ID3 算法采用信息增益作为度量标准，在选择根节点和各个内部节点属性时，选择当前样本集中具有最大信息增益值的属性作为划分标准。ID3 算法建树方法简单，学习能力较强，但偏向于选择取值较多的属性作为分支属性。其次，ID3 只能构造出离散数据集的决策树，而对连续性属性不能直接进行处理，且对噪声数据敏感，抗噪性能较差。

例 6.2 以表 6.1 所示的数据为例，使用 ID3 算法构造决策树。

首先计算样本总的信息熵 $Ent(D)=-\frac{9}{15}\log_2\frac{9}{15}-\frac{6}{15}\log_2\frac{6}{15}=0.970$。

样本集在属性 A 划分的条件下，各子集的熵为：

$$\begin{cases} Ent(D,a_1)=-\frac{3}{4}\log_2\frac{3}{4}-\frac{1}{4}\log_2\frac{1}{4}=0.811 \\ Ent(D,a_2)=-\frac{4}{6}\log_2\frac{4}{6}-\frac{2}{6}\log_2\frac{2}{6}=0.918 \\ Ent(D,a_3)=-\frac{2}{5}\log_2\frac{2}{5}-\frac{3}{5}\log_2\frac{3}{5}=0.971 \end{cases}$$

$$Ent(D,A)=\frac{4}{15}\times 0.811+\frac{6}{15}\times 0.918+\frac{5}{15}\times 0.971=0.907$$

类似的，可求出 $Ent(D,B)=0.403$， $Ent(D,C)=0.367$。

属性 A 的信息增益为 $Gain(D,A)=Ent(D)-Ent(D,A)=0.063$，其余两个属性的信息增益为 $Gain(D,B)=0.567$， $Gain(D,C)=0.603$。

依照上面的计算，属性 C 具有最高信息增益，故将其作为划分属性，将样本训练集分为三个子集 D_{c_1}、D_{c_2}、D_{c_3}，样本子集 D_{c_1} 与 D_{c_3} 均只含一类样本，不再继续划分。而子集 D_{c_2} 则需要继续划分。计算样本子集 D_{c_2} 总的信息熵：

$$Ent(D,c_2)=-\frac{4}{6}\log_2\frac{4}{6}-\frac{2}{6}\log_2\frac{2}{6}=0.918$$

计算样本子集在用属性 A 划分的条件下，子集的熵为：

$$\begin{cases} Ent(D_{c_2},a_1)=-\frac{1}{2}\log_2\frac{1}{2}-\frac{1}{2}\log_2\frac{1}{2}=1 \\ Ent(D_{c_2},a_2)=-\frac{2}{2}\log_2\frac{2}{2}=0 \\ Ent(D_{c_2},a_3)=-\frac{1}{2}\log_2\frac{1}{2}-\frac{1}{2}\log_2\frac{1}{2}=1 \end{cases}$$

$$Ent(D_{c_2},A)=\frac{2}{6}\times 1+\frac{2}{6}\times 0+\frac{2}{6}\times 1=0.667$$

同理可计算得出 $Ent(D_{c_2},B)=0$，信息增益 $Gain(D_{c_2},A)=0.251$，$Gain(D_{c_2},B)=0.918$。

属性 B 的信息增益较大，故选择它作为第二轮划分属性，其划分所得的各个子集均只含一类样本，故无法继续划分。由此建立完整的决策树如图 6.3 所示

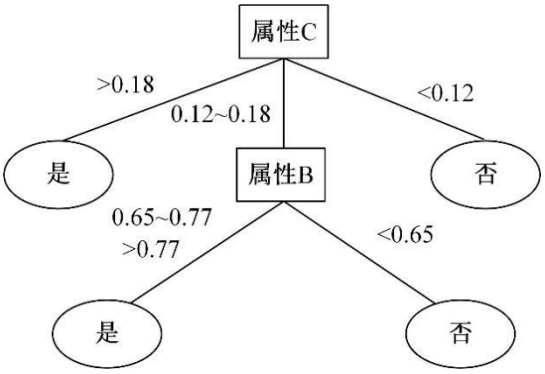


图 6.3　ID3 算法生成的决策树

图 6.3 所示的决策树可以用如下的 IF-THEN 分类规则描述：

IF $C>0.18$ THEN　电池异常

IF $0.12\le C\le 0.18$ AND　$B\ge 0.65$　THEN　电池异常

IF $0.12\le C\le 0.18$ AND　$B<0.65$　THEN　电池正常

IF $C<0.18$　THEN　电池正常

（2）C4.5 算法

C4.5 算法是基于 ID3 算法改进而来的决策树算法，它采用信息增益率作为判定划分属性好坏的标准，可以有效减少特征属性值的多少对算法的影响。它克服了 ID3 的部分缺陷，可以采用二分法处理连续属性，还能通过忽略、补全等方法处理缺失值。在构造树的过程中，由于要对数据集进行多次的顺序扫描和排序，可能会导致算法的低效。

例 6.3 以表 6.1 所示的数据为例，使用 C4.5 算法构造决策树。

在使用 ID3 算法构造决策树时已经计算出样本数据集 D 中各属性的信息增益：$Gain(D,A)=0.063$，$Gain(D,B)=0.567$，$Gain(D,C)=0.603$。只需再计算此时各属性在 D 上的分裂信息，以 A 属性为例：

$$SplitInfo(D,A)=-\frac{4}{15}\log_2\frac{4}{15}-\frac{5}{15}\log_2\frac{5}{15}-\frac{6}{15}\log_2\frac{6}{15}=1.566$$

类似可计算 $SplitInfo(D,B)=1.530$，$SplitInfo(D,C)=1.585$。

若用属性 A 对 D 进行划分，所得的信息增益率为

$$Gain\_ratio(D,A)=\frac{Gain(D,A)}{SplitInfo(D,A)}=0.040$$

同理可计算 $Gain\_ratio(D,B)=0.370$，$Gain\_ratio(D,C)=0.385$。选择具有最高信息增益率的 C 属性作为根节点的划分属性，向下建立分支。根据 C 属性建立三个子集 D_{c_1}、D_{c_2}、D_{c_3}，样本子集 D_{c_1} 和 D_{c_3} 已分类完成。然后依次计算其它属性对子集划分后的信息增益率。计算其它属性在子集 D_{c_2} 上的分裂信息：

$$SplitInfo(D_{c_2},A)=-\frac{2}{6}\log_2\frac{2}{6}-\frac{2}{6}\log_2\frac{2}{6}-\frac{2}{6}\log_2\frac{2}{6}=1.585$$

类似可计算 $SplitInfo(D_{c_2},B)=1.585$。

若用属性 A 对 D 进行划分，所得的信息增益率为

$$Gain\_ratio(D_{c_2},A)=\frac{Gain(D_{c_2},A)}{SplitInfo(D_{c_2},A)}=0.158$$

同理可计算 $Gain\_ratio(D_{c_2},B)=0.580$。选择具有最高信息增益率的 B 属性作为当前节点的划分属性，向下建立分支。其划分的各个子集均只含一类样本，故无法继续划分。由此建立了完整的决策树， C4.5 与上例中 ID3 所生成的决策树一致。

（3）CART 算法

CART 算法是一种以基尼指数作为属性选择度量的方法。CART 算法假设决策树是一棵二叉树，因此在每个分支节点出将当前的样本数据集分割成两个互不相交的子集，以基尼指数最小的属性为最佳的分支变量。CART 算法对异常点和干扰数据的抵抗性强，在面对诸如存在缺失值、变量数多等问题时 CART 显得非常稳健。然而它存在偏向多值属性的问题，且计算量较大。

例 6.4 以表 6.1 所示的数据为例，使用 CART 算法构造决策树。

首先计算样本数据 D 中属性的基尼指数，找出基尼指数减小幅度最大的属性作为根节点属性。根节点V_0的基尼指数为

$$Gini(V_0)=1-(\frac{9}{15})^2-(\frac{6}{15})^2=0.480$$

1) 计算用属性 A 对 D 划分后各样本子集的基尼指数减小值

按$\{a_1\}/\{a_2,a_3\}$划分 D 时，基尼指数为

$$Gini(D_{\{a_1\}/\{a_2,a_3\}},V_0)=\frac{4}{15}\times[1-(\frac{3}{4})^2-(\frac{1}{4})^2]+\frac{11}{15}\times[1-(\frac{6}{11})^2-(\frac{5}{11})^2]=0.464$$

基尼指数减小值

$$\Delta Gini(D_{\{a_1\}/\{a_2,a_3\}},V_0)=0.48-0.464=0.016$$

类似的可以计算 $\Delta Gini(D_{\{a_2\}/\{a_1,a_3\}},V_0)=0.006$， $\Delta Gini(D_{\{a_3\}/\{a_1,a_2\}},V_0)=0.040$

2) 计算用属性 B 对 D 划分后各样本子集的基尼指数减小值

$$\Delta Gini(D_{\{b_1\}/\{b_2,b_3\}},V_0)=0.116,$$

$$\Delta Gini(D_{\{b_2\}/\{b_1,b_3\}},V_0)=0.023,\Delta Gini(D_{\{b_3\}/\{b_1,b_2\}},V_0)=0.262$$

3) 计算用属性 C 对 D 划分后各样本子集的基尼指数减小值

$$\Delta Gini(D_{\{c_1\}/\{c_2,c_3\}},V_0)=0.160,$$

$$\Delta Gini(D_{\{c_2\}/\{c_1,c_3\}},V_0)=0.006,\Delta Gini(D_{\{c_3\}/\{c_1,c_2\}},V_0)=0.262$$

根据计算结果，对 D 按 $\{b_3\}/\{b_1,b_2\}$ 进行划分或对 D 按 $\{c_3\}/\{c_1,c_2\}$ 进行划分时，基尼指数减小幅度最大。此时出现了多个属性基尼指数减少值相等的情况，则按属性出现的先后顺序进行选择，故选择 B 对 D 按 $\{b_3\}/\{b_1,b_2\}$ 进行划分，向下生成子集 D_{b_3} 和 $D_{\{b_1,b_2\}}$。子集 D_{b_3}，只含一类样本，故无法继续划分。$D_{\{b_1,b_2\}}$ 的节点 V_{11} 的基尼指数为

$$Gini(V_{11}) = 1-(\frac{2}{11})^2-(\frac{9}{11})^2 = 0.298$$

同理按上述方法可计算属性 A、C 对 $D_{\{b_1,b_2\}}$ 划分后的基尼指数减少值：计算可知 $Gini(D_{\{c_3\}/\{c_1,c_2\}},V_{11})$，$\Delta Gini(D_{\{c_3\}/\{c_1,c_2\}},V_{11})$ 最大，故应由属性 C 按 $\{c_3\}/\{c_1,c_2\}$ 对 $D_{\{b_1,b_2\}}$ 进行划分，划分后各子样本均只含一类样本，故分类结束，生成决策树如图 6.4 所示。

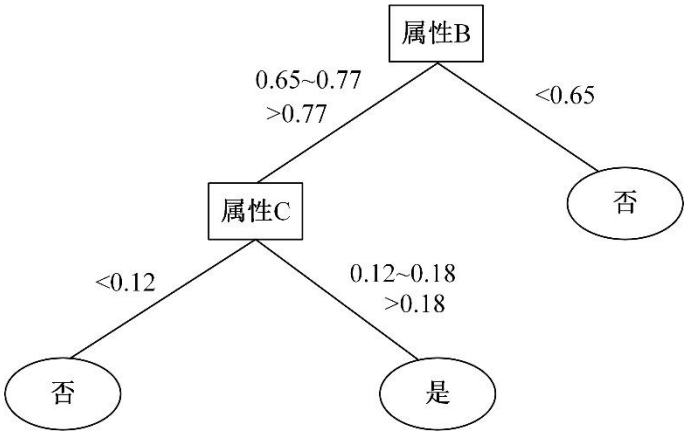


图 6.4　CART 算法生成的决策树

图 6.4 示的决策树可以用如下的 IF-THEN 分类规则描述：

IF　$B \geq 0.65$　AND　$C \geq 0.12$　THEN　电池异常

IF　$B \geq 0.65$　AND　$C < 0.12$　THEN　电池正常

IF　$B < 0.65$　THEN　电池正常

6.3 贝叶斯分类

贝叶斯法则最初是概率论中常用的归纳推理方法，后来逐步发展为一种系统的统计推断方法，被广泛地应用到统计决策、统计推断、统计估算等诸多领域。

贝叶斯分类法基于**贝叶斯定理**，运用概率推理的方式对样本数据进行分类。贝叶斯分类算法在机器学习领域有着十分重要的地位和作用，它具有模型可解释、精度高、速度高等优点。

6.3.1 贝叶斯的基本原理

在介绍贝叶斯分类之前，需要了解一些与贝叶斯定理相关的概率基础知识：

（1）先验概率

它是根据历史数据或主观判断所确定的各事件发生的概率。

（2）后验概率

它是基于试验或调查所得到的各事件发生的概率，亦可看作是在考虑相关背景和前提下得到的一个条件概率。

（3）条件概率

它是某一事件在另一事件已经发生的条件下发生的概率。

假设 A、B 为两个随机事件，事件 A 发生的概率为 $P(A)$，事件 B 发生的概率为 $P(B)$，在事件 A 已发生的前提下事件 B 发生的条件概率记作 $P(B|A)$。也可将 $P(A)$ 称作先验概率，$P(B|A)$ 称作在 A 发生的条件下 B 的后验概率，二者关系如下：

$$P(AB)=P(B|A)P(A) \tag{6.9}$$

其中 $P(AB)$ 表示随机事件 A、B 同时发生的概率，当 A、B 相互独立时有

$$P\left(AB\right)=P(A)P(B) \tag{6.10}$$

（4）全概率公式

若样本空间 D 被划分为 n 个子集 $B_1,B_2,\cdots,B_n$，它们两两互斥、互不相容，且每个子集发生的概率 $P(B_i)>0,i=1,2,\cdots,n$。则在样本空间上事件 A 发生的概率为

$$P(A)=\sum_{i=1}^{n}P(B_i)P(A|B_i) \tag{6.11}$$

（5）贝叶斯定理

当 $P(A)>0$ 时，由式（6.11）可得到贝叶斯定理

$$P(B_i|A)=\frac{P(B_iA)}{P(A)}=\frac{P(B_i)P(A|B_i)}{\sum_{i=1}^{n}P(B_i)P(A|B_i)} \tag{6.12}$$

6.3.2 朴素贝叶斯分类

可以发现，贝叶斯定理提供了一种由 $P(B_i)$、$P(A|B_i)$ 和 $P(A)$ 计算后验概率 $P(B_i|A)$ 的方法。将其应用于分类问题中：若事件 A 表示样本数据的属性集合，事件 B 表示数据的类标签集合。则对于某个待分类样本，可以通过式（6.12）计算出该样本隶属于某个类别的后验概率，将后验概率最大的类别视作该样本所属类别。

具体地，给定一个训练样本集 D，假设 D 中样本包含 m 个类别，记作

$C_1, C_2, \cdots, C_m$。训练样本含 n 个属性，记作 $A_1, A_2, \cdots, A_n$，则每个训练样本现给定某个未知类别的待分类样本 $\boldsymbol{X}$，$\boldsymbol{X}$ 可以表示为一组 n 维向量：$\boldsymbol{X} = \{x_1, x_2, \cdots, x_n\}$，其中 x_i 表示该样本在属性 A_i 上的测试值。贝叶斯分类的任务是预测样本 $\boldsymbol{X}$ 所属的类标签。

由式（6.12）可计算各类别的后验概率 $P(C_i \mid \boldsymbol{X})$：

$$P(C_i \mid \boldsymbol{X}) = \frac{P(C_i)P(\boldsymbol{X} \mid C_i)}{P(\boldsymbol{X})} \tag{6.13}$$

其中，$P(\boldsymbol{X})$ 对于所有类别来说都是一个相同常数。故想要比较不同类别后验概率的大小，只需要计算 $P(C_i)P(\boldsymbol{X} \mid C_i)$ 并比较大小即可。下面主要介绍类先验概率 $P(C_i)$ 以及条件概率 $P(\boldsymbol{X} \mid C_i)$ 的计算方法。

（1）计算先验概率 $P(C_i)$

类先验概率可通过式（6.14）来估计：

$$P(C_i) = \left|C_{i,D}\right| \Big/ |D| \tag{6.14}$$

其中，$\left|C_{i,D}\right|$ 是样本 D 中属于 C_i 类的样本数，$|D|$ 是 D 的总样本数量。

在有些情况下无法计算类先验概率，则通常假定这些类等概率。即 $P(C_1) = P(C_2) = \cdots = P(C_m)$，此时只需要比较 $P(\boldsymbol{X} \mid C_i)$ 的大小即可。

（2）计算条件概率 $P(\boldsymbol{X} \mid C_i)$

待分类样本 $\boldsymbol{X} = \{x_1, x_2, \cdots, x_n\}$，其中 x_k 表示该样本在属性 A_k 上的测试值。计算 $P(\boldsymbol{X} \mid C_i)$ 主要的困难在于，它是所有属性上的联合概率，在有限的样本条件下是很难估计准确的。基于有限的训练样本直接估计联合概率，计算会比较困难。

为简化计算，在朴素贝叶斯分类中作如下假设：各数据属性之间相互独立，不存在任何关联性和依赖关系。因此

$$P(\boldsymbol{X} \mid C_i) = P(x_1 \mid C_i)P(x_2 \mid C_i)P(x_3 \mid C_i) \cdots P(x_n \mid C_i) \tag{6.15}$$

下面考虑概率 $P(x_k \mid C_i)$ 的计算，对于每个属性，应首先明确该属性是离散的还是连续的。

如果 A_k 是离散属性，则可作如下估计：

$$P(x_k \mid C_i) = \left|D_{C_i, x_k}\right| \Big/ \left|C_{i,D}\right| \tag{6.16}$$

其中，$\left|D_{C_i, x_k}\right|$ 是 C_i 类数据样本中属性 A_k 的值为 x_k 的样本数。

如果 A_k 是连续值属性，通常假定连续值属性服从均值为 μ、标准差为 σ 的高斯分布，如式（6.17）定义

$$g(x,\mu,\sigma)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{6.17}$$

因此可计算 $P(x_k \mid C_i)$ 如下

$$P(x_k \mid C_i)=g(x_k,\mu_{C_i},\sigma_{C_i}) \tag{6.18}$$

（3）**计算概率乘积** $P(C_i)P(\boldsymbol{X} \mid C_i)$

朴素贝叶斯分类法将预测 $\boldsymbol{X}$ 属于具有最高后验概率的类，也就是说，当且仅当式（6.19）成立时，朴素贝叶斯分类预测 $\boldsymbol{X}$ 属于类 C_i 。

$$P(C_i \mid \boldsymbol{X})>P(C_j \mid \boldsymbol{X}),\quad 1\le j\le m,\ j\ne i \tag{6.19}$$

朴素贝叶斯分类的算法流程如下：

输入：待预测样本 $\boldsymbol{X}$，属性集 A，类别集合 $C=\{C_1,C_2,\ldots,C_n\}$

输出：样本 $\boldsymbol{X}$ 预测类别

---

1. for C 中的每一个类别 C_i ；
2. 计算该类样本在所有样本中出现的概率 $P(C_i)$
3. 由式（6.14）计算 $P(\boldsymbol{X} \mid C_i)$
4. 计算 $P(\boldsymbol{X} \mid C_i)P(C_i)$
5. end for
6. 找出最大值 $P(\boldsymbol{X} \mid C_m)P(C_m)$
7. 判定 $\boldsymbol{X} \in C_m$

例 6.5 以表 6.1 中的数据为参考，使用朴素贝叶斯分类来预测样本 $\boldsymbol{X}=\{A=a_3,B=b_2,C=c_2\}$ 的预测类标签（电池异常记作 Y_1，电池正常记作 Y_2 ）。

1）计算各类别先验概率：

$$P(Y_1)=9/15=0.6 \qquad P(Y_2)=6/15=0.4$$

2）计算类条件概率

$$P(A=a_3 \mid Y_1)=2/9=0.222 \qquad P(A=a_3 \mid Y_2)=3/6=0.500$$

$$P(B=b_2 \mid Y_1)=5/9=0.556 \qquad P(B=b_2 \mid Y_2)=2/6=0.333$$

$$P(C=c_2 \mid Y_1)=4/9=0.444 \qquad P(C=c_1 \mid Y_2)=2/6=0.333$$

由式（6.15）可计算：

$$P(\boldsymbol{X} \mid Y_1)=P(A=a_3 \mid Y_1)P(B=b_2 \mid Y_1)P(C=c_2 \mid Y_1)=0.055$$

$$P(\boldsymbol{X} \mid Y_2)=P(A=a_3 \mid Y_2)P(B=b_2 \mid Y_2)P(C=c_2 \mid Y_2)=0.055$$

3）计算概率 $P(\boldsymbol{X} \mid Y_i)P(Y_i)$

$P(\boldsymbol{X}\mid Y_1)P(Y_1)=0.055\times 0.6=0.033$

$P(\boldsymbol{X}\mid Y_2)P(Y_2)=0.055\times 0.4=0.022$

4）找出上述结果最大的类

由于 $P(\boldsymbol{X}\mid Y_1)P(Y_1)>P(\boldsymbol{X}\mid Y_2)P(Y_2)$，故样本 $\boldsymbol{X}$ 电池状态异常（即 $\boldsymbol{X}\in Y_1$）。

需注意，若某个属性值在训练集中没有与某个类同时出现过，则由式（6.15）计算出的连乘式概率为零。一个零概率将消除乘积中涉及的所有其他概率的影响。为了避免其他属性携带的信息被训练集中未出现的属性值抹去，在估计概率值时常用“**拉普拉斯校准**”。

具体来说，令 N_k 表示第 k 个属性 A_k 所有可能的取值数，则式（6.16）可修正为：

$$P(x_k\mid C_i)=\frac{\left|D_{C_i,x_k}\right|+1}{\left|C_{i,D}\right|+N_k} \tag{6.20}$$

拉普拉斯校准避免了因训练集样本不充分而导致概率估值为零的问题，且在训练集较大时，修正过程对概率估计造成的变化可以忽略不计，使得估值趋向于实际概率值。

朴素贝叶斯分类算法简单、容易实现、速度快，且模型需要的估计的参数很少，对缺失的数据不敏感。理论上朴素贝叶斯算法是分类错误概率最小的分类器。但实际上，由于朴素贝叶斯分类假定各属性之间相互独立，忽略了各变量之间可能存在的依赖关系，这种假设在一定程度上降低了朴素贝叶斯的分类准确率。

与朴素贝叶斯分类相比，贝叶斯信念网络（Bayesian Belief Network）允许各属性之间存在依赖关系，弥补了朴素贝叶斯的上述不足。一个完整的贝叶斯网络由有向无环图与概率关系表两个部分组成，它提供一种描述事件间因果关系的图形结构，使得不确定性推理在逻辑上更清晰，可理解性更强，对此感兴趣的读者可以深入阅读相关文献资料。

6.4 支持向量机

支持向量机（SVM）是建立在统计学习理论基础上的一种预知性机器学习方法。它根据结构最小化准则构造分类器，能够使类与类之间的间隔最大化。SVM 对样本维度的数量不太敏感，它在解决小样本、非线性及高维模式识别中表现出许多特有的优势，并能够推广应用到函数拟合等其他机器学习问题中。

6.4.1 支持向量机的基本原理

不失一般性，可将分类问题限制于二分类，即数据样本具有两种类别。假设样本数据有两个特征属性 A_1 和 A_2，每个样本在二维坐标系中对应一个样本点，

用不同颜色以区分类别，如图 6.5（a）所示。

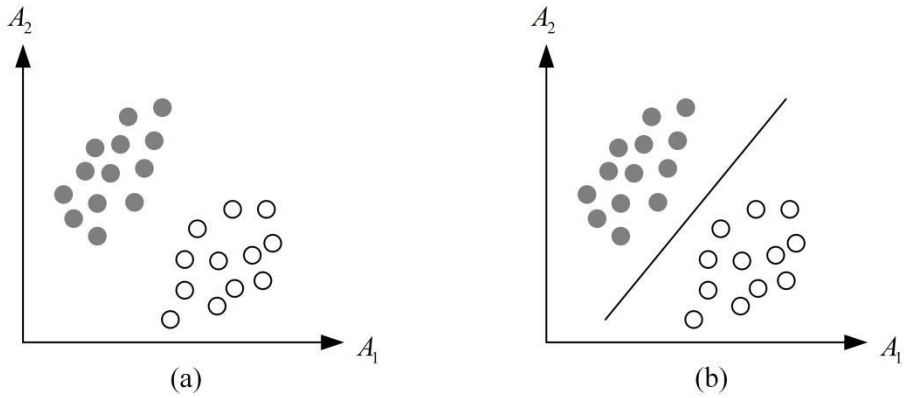


图 6.5 二维空间下线性可分的样本示例

此时坐标系中若存在一条直线（即线性函数）可将白、灰两类样本点准确无误地分开，则称样本数据**线性可分**（Linear Separable），如图 6.5（b）所示。若找不出这样一条直线，说明样本数据非线性可分。下面对线性可分进行数学描述：

设给定的数据集 $D=\{(\boldsymbol{X}_1,y_1),(\boldsymbol{X}_2,y_2),\cdots,(\boldsymbol{X}_n,y_n)\}$，其中 $\boldsymbol{X}_i$ 为训练样本，对应的样本类别为 y_i，y_i 不妨取值+1 与-1。若样本点线性可分，则正样本和负样本一定落在直线的两侧，如图 6.6 所示。二维空间中分类直线 L 可以表示为

$$w_1 x_1 + w_2 x_2 + b = 0 \tag{6.21}$$

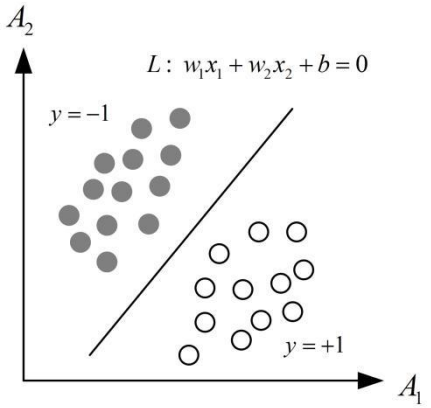


图 6.6 线性可分的数学表达示例

在直线 L 两侧，对于右下方的点 $y_i=+1$，需要满足 $w_1x_1+w_2x_2+b>0$；对于左上方的点 $y_i=-1$，需要满足 $w_1x_1+w_2x_2+b<0$。结合上面两个不等式，可以得到样本线性可分的定义如下

$$y_i(w_1x_1+w_2x_2+b)>0 \quad \forall \tag{6.22}$$

推广至更高维数，线性可分的样本点难以直接绘出，可用向量的形式简洁描述如下

$$y_i(\boldsymbol{w}^T \bullet \boldsymbol{X}_i + b) > 0 \quad \forall i \tag{6.23}$$

其中，$\boldsymbol{w}$ 为权重向量矩阵，$\boldsymbol{w} = (w_1, w_2, \ldots, w_n)^T$，$\boldsymbol{X}_i = (x_1, x_2, \ldots, x_n)^T$，$n$ 为属性数。b 是标量，称为偏置。

对于线性可分的样本数据集，存在无数个满足上述条件的超平面。如图 6.7（a）所示，图中 L_1、L_2 均可将样本点准确分开，然而其效果有优劣之分。受噪声影响，样本点可能在附近发生扰动，如图 6.7（b）所示，若白点发生轻微扰动，超平面 L_2 会将此样本错误分类。而超平面 L_1 距样本点都比较远，其分类结果仍然准确无误。

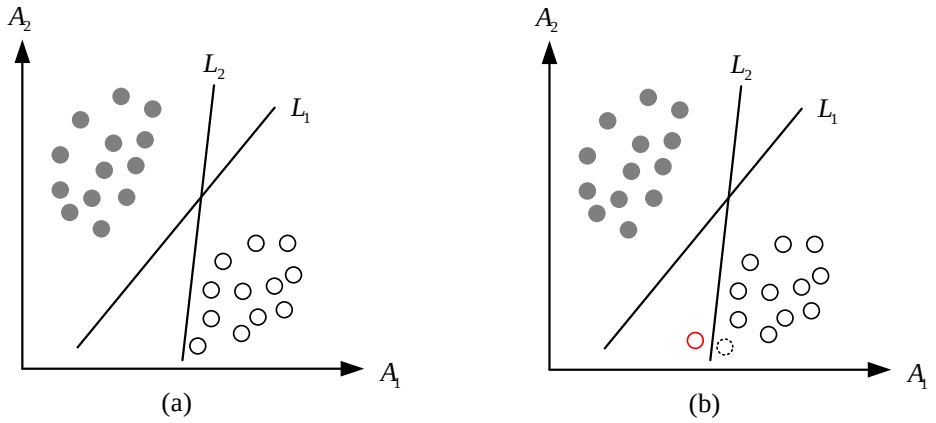


图 6.7 样本点的轻微扰动

从图像上直观来看，这是 L_2 距离白色样本点过近导致的。超平面 L_1 比 L_2 容错率更高，鲁棒性更强。SVM 分类的任务是找一个“最佳”超平面，它不应该过于偏向任何一侧的样本点，否则容易出现错误。换言之，最佳超平面是一条位于两类样本“正中间”的直线。

如何寻找这条“正中间”的直线？这与离直线最近的样本点位置密切相关，这些样本点称作**支持向量**，是 SVM 寻找最佳分类超平面的关键。如图 6.8（a）所示，圈出的样本点距离直线 L 最近，由于直线位于中间，它与两侧支持向量的距离 d_0 应当相等。两侧支持向量与超平面距离之和定义为超平面的**间隔（Margin）**。

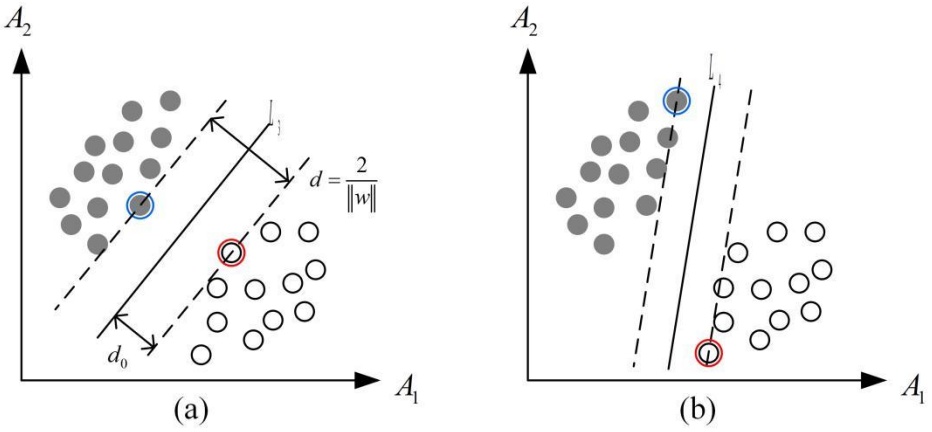


图 6.8 最佳分类超平面

然而，位于样本点“正中间”的直线也不止一条，如图 6.8（b）所示，超平面 L_4 也在样本“正中间”，通过对比我们可以发现 L_3 的间隔较大，L_4 的间隔较小。从直观上看 L_3 距离两类样本比 L_4 更远，分类结果相对来说更加可靠。所以最佳分类超平面需要使得间隔最大化，且位于间隔的“正中间”。

综上所述，SVM 要寻找的最优分类超平面应满足以下条件：

1）超平面准确无误地分开了两类样本。

2）超平面与两类支持向量的距离相等。

3）超平面使得间隔最大化。

根据平面几何知识，支持向量 $\boldsymbol{x}_0$ 到分类超平面的距离为

$$d_0 = \frac{\left|\boldsymbol{w}^T \boldsymbol{x}_0 + b\right|}{\|\boldsymbol{w}\|} = \frac{y_0\left(\boldsymbol{w}^T \boldsymbol{x}_0 + b\right)}{\|\boldsymbol{w}\|} \tag{6.24}$$

其中 $\|\boldsymbol{w}\| = \sqrt{\boldsymbol{w}_1^2 + \boldsymbol{w}_2^2 + \cdots \boldsymbol{w}_n^2}$ 。

最佳超平面应使间隔 $d = 2d_0$ 最大化，即 SVM 寻找最佳分类超平面的问题转化为距离参数的最大化问题：

$$\begin{aligned} &\max_{w,b} d \\ &s.t.\ \ y_i(\boldsymbol{w}^T \bullet \boldsymbol{X}_i + b) \geq 0\ ,\ \ \forall i \end{aligned} \tag{6.25}$$

解得 $\boldsymbol{w}$ 与 b 值即可得到最优分类超平面。然而我们注意到 d_0 不仅受 $\boldsymbol{w}$ 与 b 的影响，对于不同的分类超平面，与之对应的支持向量也是不同的，难以方便地计算 d_0 的最优值，此时需要对 d_0 的表达式进行化简。

我们对 $\boldsymbol{w}$ 与 b 作等比例缩放变换，即令 $\boldsymbol{w}_1 = \lambda \boldsymbol{w}$，$b_1 = \lambda b$，$\lambda \neq 0$。原超平面方程：$L: \boldsymbol{w}^T \boldsymbol{x}_i + b = 0$，变换后超平面方程：$L_1: \boldsymbol{w}_1^T \boldsymbol{x}_i + b = \lambda\left(\boldsymbol{w}^T \boldsymbol{x}_i + b\right) = 0$，可以发现 $\boldsymbol{w}$ 与 $\boldsymbol{b}$ **等比例缩放之后不改变原超平面的位置**。

若$\left|\boldsymbol{w}^T\boldsymbol{x}_0+b\right|=q$，选取$\lambda=\frac{1}{q}$，令$\boldsymbol{w}=\lambda\boldsymbol{w}$，$b=\lambda b$，缩放后$\left|\boldsymbol{w}^T\boldsymbol{x}_0+b\right|=1$。对于不同位置的超平面，我们总可以选择合适的$\lambda$使得$\left|\boldsymbol{w}^T\boldsymbol{x}_0+b\right|=1$。则式（6.24）可简化为：

$$d_0=\frac{\left|\boldsymbol{w}^T\boldsymbol{x}_0+b\right|}{\|\boldsymbol{w}\|}=\frac{1}{\|\boldsymbol{w}\|} \tag{6.26}$$

式中$\boldsymbol{w}$与b均为等比例变换之后的值，当然也可以将$\left|\boldsymbol{w}^T\boldsymbol{x}_0+b\right|$放缩至其它常数。为方便数学描述，在支持向量机中习惯性地缩放为1。

由上述分析可知，其余样本点到超平面的距离均大于d_0，可将线性可分的表达改写为

$$y_i(\boldsymbol{w}^T\bullet\boldsymbol{X}_i+b)\geq 1\forall i \tag{6.27}$$

两侧的支持向量应该满足：

$$\begin{aligned}\boldsymbol{w}^T\bullet\boldsymbol{X}_i+b=+1 \quad y_i=+1\\ \boldsymbol{w}^T\bullet\boldsymbol{X}_i+b=-1 \quad y_i=-1\end{aligned} \tag{6.28}$$

分类超平面的间隔

$$d=2d_0=\frac{2}{\|\boldsymbol{w}\|} \tag{6.29}$$

优化问题可简化改写为

$$\begin{aligned}&\max_{w,b}\frac{2}{\|\boldsymbol{w}\|}\\ &s.t.\ \ y_i(\boldsymbol{w}^T\bullet\boldsymbol{X}_i+b)\geq 1,\ \ \forall i\end{aligned} \tag{6.30}$$

6.4.2 支持向量机求解

为方便求导可将目标函数定为$\frac{1}{2}\|\boldsymbol{w}\|^2$，该优化问题可转换为：

$$\begin{aligned}&\min_{w,b}\frac{1}{2}\|\boldsymbol{w}\|^2\\ &s.t.\ \ y_i(\boldsymbol{w}^T\bullet\boldsymbol{X}_i+b)\geq 1,\ \ \forall i\end{aligned} \tag{6.31}$$

对式（6.31）使用拉格朗日乘子法可得到“对偶问题”，满足约束条件的优化问题可由拉格朗日算符$L(\boldsymbol{w},b)$转换

$$L(\boldsymbol{w},b,\boldsymbol{\alpha})=\frac{1}{2}\|\boldsymbol{w}\|^2-\sum_{i=1}^{n}\alpha_i\left(y_i(\boldsymbol{w}^T\bullet\boldsymbol{X}_i+b)-1\right) \tag{6.32}$$

式中α_i是拉格朗日乘数，$\alpha_i\geq 0$，对每条约束条件都要添加。拉格朗日函数关于$\boldsymbol{w}$和b最小化：

$$\frac{\partial L}{\partial b}=0 \Rightarrow \sum_{i=0}^{n}\alpha_i y_i=0$$

$$\frac{\partial L}{\partial \boldsymbol{w}}=0 \Rightarrow \boldsymbol{w}=\sum_{i=0}^{n}\boldsymbol{X}_i\alpha_i y_i$$

代入原目标函数可得对偶问题如下：

$$\begin{aligned}&\max_{\boldsymbol{\alpha}}\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{X}_j^T\boldsymbol{X}_i\\&s.t.\sum_{i=1}^{n}\alpha_i y_i=0\ \alpha_i\geq 0\end{aligned}\tag{6.33}$$

这是一个二次规划问题，其中$\boldsymbol{\alpha}=(\alpha_1,\alpha_1,\cdots,\alpha_n)$。为求解参数$\boldsymbol{\alpha}$，可用常见的二次规划算法求解。除此之外，有一些较简便的算法，较为常见的有序列最小最优化（SMO）算法。SMO 算法每次选择两个变量α_i、α_j，并固定其它参数，仅优化这两个变量。解出$\boldsymbol{\alpha}$之后，计算权重向量$\boldsymbol{w}$。

要得到最终的分类超平面，还需确定偏置 b 的值。所有的支持向量都满足

$$y_s(\boldsymbol{w}^T\cdot\boldsymbol{X}_s+b)=y_s\left(\sum_{i\in S}\alpha_i y_i\boldsymbol{X}_i^T\boldsymbol{X}_s+b\right)\tag{6.34}$$

其中 S 为所有支持向量的下标集合，使用所有支持向量求解的平均值计算偏置 b

$$b=\frac{1}{|S|}\sum_{s\in S}\left(\frac{1}{y_s}-\sum_{i\in S}\alpha_i y_i\boldsymbol{X}_i^T\boldsymbol{X}_s\right)\tag{6.35}$$

可得到分类模型：

$$f(\boldsymbol{X})=\boldsymbol{w}^T\boldsymbol{X}+b=\sum_{i=1}^{n}\alpha_i y_i\boldsymbol{X}_i^T\boldsymbol{X}+b\tag{6.36}$$

由于优化问题存在不等式约束，上述过程需要满足 KKT 条件：

$$\begin{cases}\alpha_i\geq 0\\ y_i f(\boldsymbol{X}_i)-1\geq 0\\ \alpha_i(y_i f(\boldsymbol{X}_i)-1)=0\end{cases}\tag{6.37}$$

由 KKT 条件可以看出，对任何训练样本，总有$\alpha_i=0$或$y_i f(\boldsymbol{X}_i)=1$。当$\alpha_i=0$时，该样本对最终形成的分类超平面没有影响。当$y_i f(\boldsymbol{X}_i)=1$时，该样本点是一个支持向量。故整个 SVM 模型的训练结果仅与各支持向量有关。

在前面的分析中，我们假定 SVM 必须将所有样本都分类正确，然而实际上很难找到这样的分类函数，而且难以确定线性可分的结果是否过拟合。为了改进泛化能力，我们需要放宽约束条件，通过建立柔性边界允许一定程度的误分类，此时可称之为“软间隔支持向量机”，如图 6.9 所示。

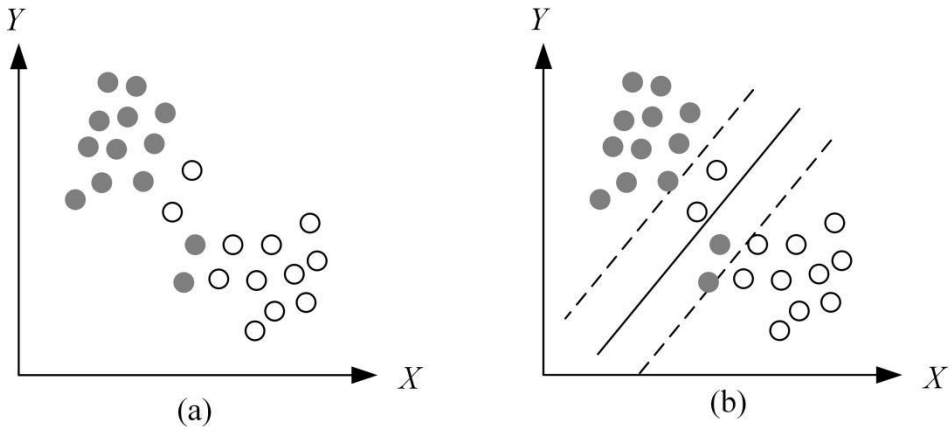


图 6.9 软间隔支持向量机示意图

具体来说，对每个训练样本及类标签$(\boldsymbol{X}_i, y_i)$设置“松弛变量”ξ_i（$\xi_i \geq 0$），ξ_i 刻画的是对应样本不满足约束的程度。可将上述优化问题（6.31）改写为

$$\begin{aligned} &\min_{w,b} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{s.t.} \quad y_i(\boldsymbol{w}^T \bullet \boldsymbol{X}_i + b) \geq 1 - \xi_i, \quad \forall i \end{aligned} \tag{6.38}$$

其中比例因子$C > 0$，C 是人为设定的超参数，通过不断变化 C 的值同时测试算法的准确度，从而选择较合适的 C 值。

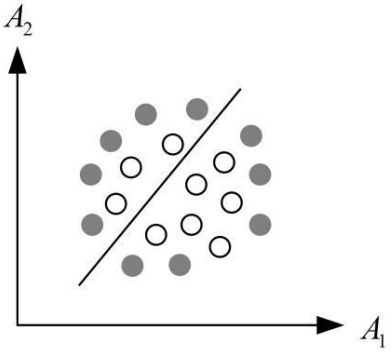


图 6.10 样本数据非线性可分

此外，在分类问题中，很多情况下训练集是非线性可分的。在此情况下不存在 $\boldsymbol{w}$ 和 b 满足上述优化问题的限制条件。即使引入了松弛变量，用线性函数划分还是存在极大的误差，如图 6.10 所示。在此情况下，一般在 SVM 中定义核函数，实现训练集向更高维空间（特征）的映射，将其转化为线性可分的情况处理。常见的核函数主要有线性核函数、多项式核函数、高斯核函数等，对此感兴趣的读者可以查阅相关资料深入学习。

例 6.6 检测滚动轴承的质量

在生产现场，轴承的检测主要是通过时域信号完成的。在以“异声”为主要

考虑指标的轴承质量检测中，可选择相关性较大的两个指标，它们分别是：加速度信号均方根值的分贝值 L，超过某幅值的点数 M。质量检测将产品分为合格 P 与不合格 N 两个类别，选择合格与不合格样本各 50 个作为 SVM 的训练样本。$y_i \in \{-1,1\}$，质量合格时记为 1，否则记为 -1。试求取 SVM 的分类超平面。

1）绘制样本散点图

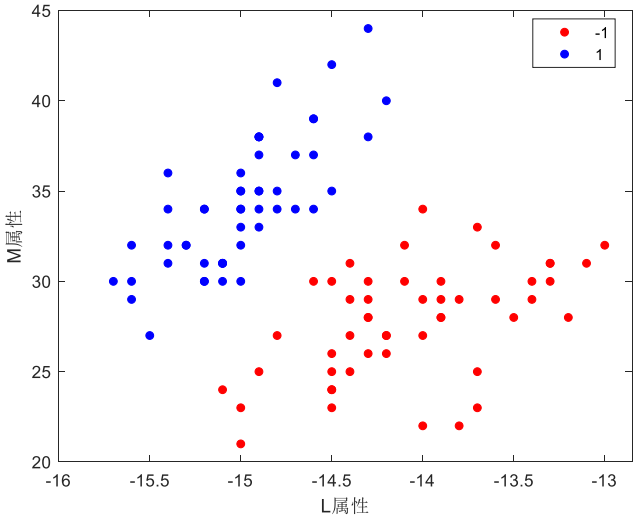


图 6.11 数据样本散点图

2）优化问题求解

构造式（6.38）所示的优化问题，通过 SMO 算法编程求得最优解，可确定各支持向量 SVs 属性值与其对应的拉格朗日乘数 α_i 如下：

表 6.5 支持向量及拉格朗日乘数表

序号	L	M	α	序号	L	M	α
1	-15.1	30	1	6	-14.8	27	0.5431
2	-14.6	34	1	7	-14.4	31	1
3	-15	30	1	8	-14.6	30	1
4	-14.5	35	0.5431	9	-14	34	1
5	-15.5	27	1	10	-14.4	30	1

各支持向量在散点图中被圈出，如图 6.12 所示

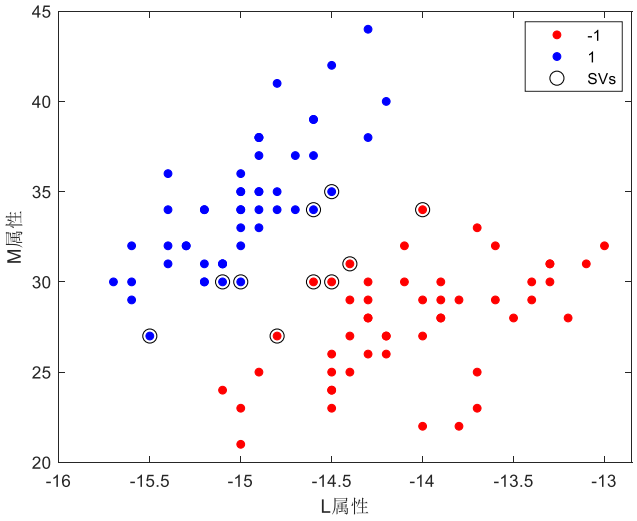


图 6.12 支持向量示意图

3）计算分类超平面

由 α_i 可确定权重向量 $\boldsymbol{w}$ 与偏置 b：

$$\boldsymbol{w} = (w_1, w_2)^T = (-2.537, 0.345)^T$$

$$b = -47.867$$

$\boldsymbol{w}$ 与 b 对应分类超平面如图 6.13 所示，其数学表达为

$$-0.2537x_1 + 0.345x_2 - 47.867 = 0$$

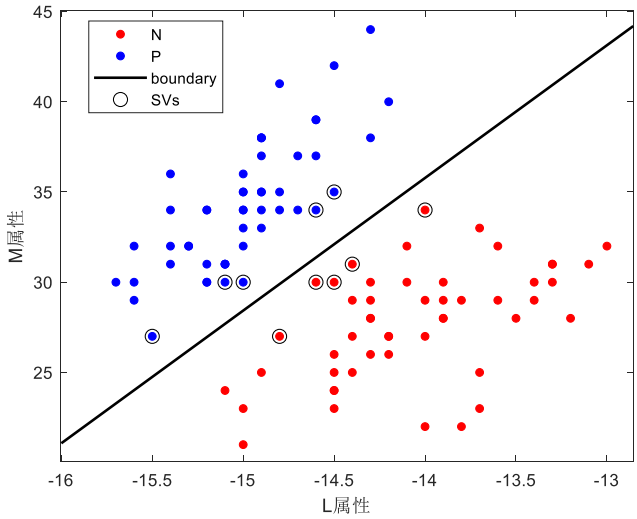


图 6.13 分类超平面示意图

6.5 人工神经网络

人工神经网络也称为神经网络或类神经网络，它是一种应用类似于大脑神经突触连接的结构进行分布式并行信息处理的算法数学模型。这种网络具有良好的

自适应和自学习能力，在优化、信号处理与模式识别、智能控制、故障诊断等领域都有着广泛的应用。本章以 BP 神经网络为例，简要介绍人工神经网络在分类学习中的应用。

6.5.1 人工神经网络拓扑

人工神经网络的基本处理单元是神经元，在 1943 年，基于神经元的生理结构，人们建立了单个神经元的数理模型（MP 模型），其结构模型如图 6.14 所示。

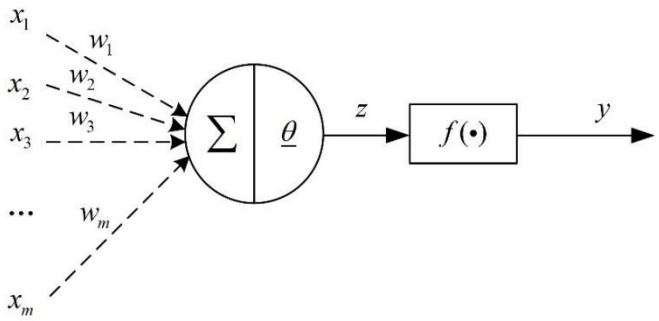


图 6.14　MP 神经元的结构模型

一个基本的神经元包括输入信号、求和单元、激活函数及输出信号。神经元接收到来自 m 个神经元传递的输入信号，乘以各自对应的权重 w_i 后求和，作为该神经元的总输入。将其与神经元阈值比较，通过激活函数处理，便得到神经元的输出。根据图 6.14 所示的模型，可以用式（6.39）来描述一个神经元：

$$\begin{aligned} z &= \sum_{i=1}^{m} w_i x_i - \theta \\ y &= f(z) \end{aligned} \tag{6.39}$$

其中，x_i 是输入信号，w_i 为输入权值，z 为加法器输出，$f(\bullet)$ 为激活函数，y 为神经元的输出。

将大量神经元按一定的结构连接起来，每层神经元两两相连，便构成多层神经网络。一般来说，一个典型多层网络具有三个层次：输入层、隐藏层、输出层。其拓扑结构如图 6.15 所示：

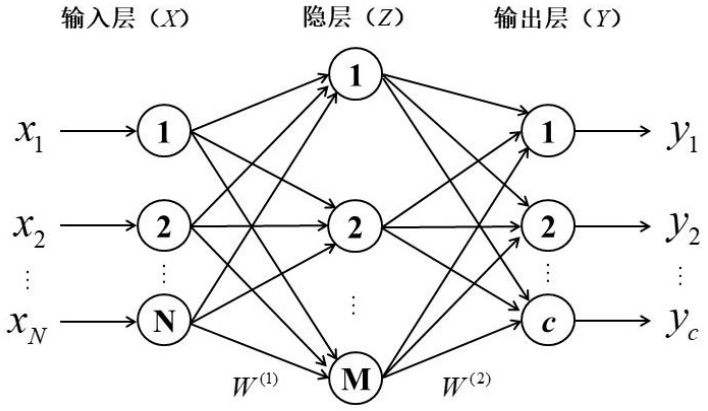


图 6.15　分层神经网络示例

输入层是人工神经网络的起始，用于接收外部环境的输入信号，将其传递给隐藏层的各个单元。隐藏层是神经网络的内部处理单元，用于对输入信号进行处理，既可以是一层，也可以是多层，层数视具体需求而定。输出层用于输出神经网络信号。

当每次训练结束，将输出结果与期望输出进行比较，判别网络输出的误差是否在允许范围内。为使评估模型的预测值 $\hat{y}$ 与真实值 y 的差距，可构造损失函数（Loss Function），也称代价函数。常见的损失函数如下：

1）绝对值损失函数

$$Loss=\sum_{i=1}^{n}\left|y_i-\hat{y}_i\right|=\sum_{i=1}^{n}\left|y_i-(wx_i+b)\right| \tag{6.40}$$

2）平方损失函数

$$Loss=\frac{1}{2}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2=\frac{1}{2}\sum_{i=1}^{n}(y_i-(wx_i+b))^2 \tag{6.41}$$

损失函数越小，说明所得结果越接近期望值。为求导计算方便，BP 神经网络算法将以平方损失函数为目标函数进行训练。

除此之外，**激活函数**也是神经网络的重要组成部分。没有激活函数，整个网络便只有加法和乘法级联等线性运算，神经网络难以解决非线性问题。而阶跃函数具有不连续、不光滑等性质，不便于参与求导等运算，故实际中常常使用其它激活函数，常用的有以下几种：

1）Sigmoid 函数（S 型函数）

$$f(x)=\frac{1}{1+e^{-x}} \tag{6.42}$$

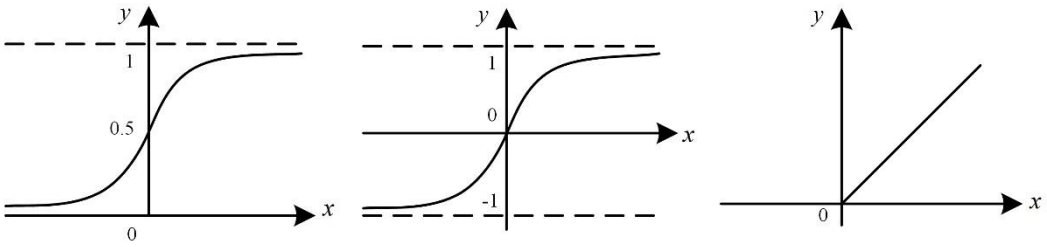
2）Tanh 函数，也称双曲正切函数

$$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{6.43}$$

3）ReLU（Rectified Linear Unit）函数

$$f(x)=\max\{0,x\} \tag{6.44}$$

常见激活函数的图像如图 6.16 所示



(a) Sigmoid函数　　(b) Tanh函数　　(c) ReLU函数

图 6.16　常见激活函数图像

还有许多其它函数可以作为人工神经网络的激活函数，但对这些函数的要求是连续可导，可以允许在少数点上不可导。

6.5.2 反向传播过程

针对多层神经网络的训练算法有很多，本节主要介绍误差反向传播算法（BP）。不失一般性，考虑一个典型单隐层 BP 神经网络模型，如图 6.17 所示：

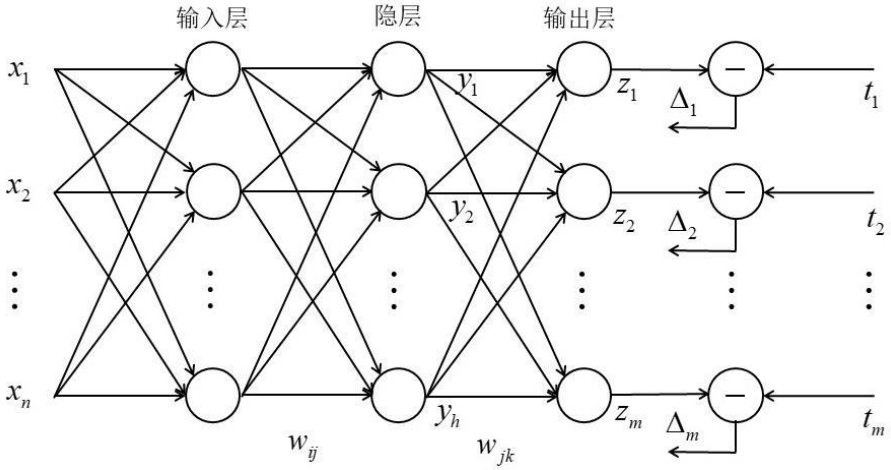


图 6.17　BP 神经网络模型

设网络的输入为 $\boldsymbol{x}=(x_1,x_2,\ldots,x_n)^T$，隐藏层有 h 个神经元，输出为 $\boldsymbol{y}=(y_1,y_2,\ldots,y_h)^T$。输出层有 m 个神经元，输出为 $\boldsymbol{z}=(z_1,z_2,\ldots,z_m)^T$，期望输出为 $\boldsymbol{t}=(t_1,t_2,\ldots,t_m)^T$。$w_{ij}$ 表示输入层第 i 个神经元到隐藏层第 j 个神经元的权值，w_{jk} 表示隐藏层第 j 个神经元到输出层第 k 个神经元的权值。隐藏层到输出层的激活函数为 f，输出层的激活函数为 g。

BP 神经网络中的参数更新均由梯度下降算法实现，梯度下降算法也称最速下降算法，它基于这样一个事实：若实值函数 $f(x)$ 在点 x_k 处可微且有定义，那么函数 $f(x)$ 在 x_k 点沿着负梯度（梯度的反方向）下降最快，沿着梯度下降方向求解最小值。算法步骤如下：

1）任取一点 w_i，将 $f(w)$ 对 w 求偏导，计算梯度 $\frac{\partial f(w)}{\partial w}|_{w=w_i}$

2）令 k=0。若 $\frac{\partial f(w)}{\partial w}|_{w=w_i}=0$，退出运算。

否则应按式（6.49）调整 w_i 的值：

$$w_i(t+1)=w_i(t)-\eta\frac{\partial f(w)}{\partial w}|_{w=w_i} \tag{6.45}$$

式中 η 称为学习率，它控制算法每一轮迭代的更新步长。步长太大则易震荡，太小则收敛速度慢，此处算法可自动根据斜率调整步长，同时自动确定下一次更新的方向（即负梯度方向）。

为计算方便，以网络误差平方和（即平方损失函数）为目标函数，采用梯度下降法对单隐层 BP 神经网络进行训练，一次完整的 BP 迭代过程包括正向传播与误差的反向传播等过程，基本步骤如下：

（1）设置模型参数初始值

神经网络学习之前要确定网络的初始权值与阈值，初始参数的设置直接影响着神经网络的学习性能。一般的经验是初始权重通常在 –1 到 1 之间取值，后续再更新。

（2）计算正向传播过程中各节点的输出，包括隐层各节点和输出层各节点

$$y_j=f(\sum_{i=1}^{n}w_{ij}x_i-\theta_j) \tag{6.46}$$

$$z_k=g(\sum_{j=1}^{h}w_{jk}y_j-\gamma_k) \tag{6.47}$$

式中的 y_j 表示隐藏层第 j 个神经元的输出；式中的 z_k 表示输出层第 k 个神经元的输出。式中阈值 θ_j 与 γ_k 可看作固定输入为 –1.0 的“哑结点”所对应的连接权重。这样，权重和阈值的学习便可统一为权重的学习，即令 $w_{0j}=\theta_j$，$w_{0k}=\gamma_k$，$x_0=y_0=-1$。则式（6.46）及（6.47）可以简化为

$$y_j=f(\sum_{i=0}^{n}w_{ij}x_i) \tag{6.48}$$

$$z_k=g(\sum_{j=0}^{h}w_{jk}y_j) \tag{6.49}$$

（3）计算输出误差

此时，网络输出与目标输出的均方误差为

$$Loss = \frac{1}{2}\sum_{k=1}^{m}(t_k - z_k)^2 \tag{6.50}$$

（4）误差反向传播，调整权值

按梯度下降算法调整权值，使误差减小。每次权值的调整为

$$\Delta w_{pq} = -\eta \frac{\partial Loss}{\partial w_{pq}} \tag{6.51}$$

式中，η在神经网络中称为学习率。可以证明：按这个方法调整，误差会逐渐减小。BP 神经网络的调整顺序为：

a）先调整隐藏层到输出层的权值。设v_k为输出层第 k 个神经元的输入，则

$$v_k = \sum_{j=0}^{h} w_{jk} y \tag{6.52}$$

$$\frac{\partial Loss}{\partial w_{jk}} = \frac{\partial Loss}{\partial z_k}\frac{\partial z_k}{\partial v_k}\frac{\partial v_k}{\partial w_{jk}} = -(t_k - z_k)g'(v_k)y_j \triangleq -\delta_k y_j \tag{6.53}$$

于是隐藏层到输出层的权值调整迭代公式为

$$w_{jk}(t+1) = w_{jk}(t) + \eta\delta_k \tag{6.54}$$

b）再调整输入层到隐藏层的权值，同理

$$\frac{\partial Loss}{\partial w_{ij}} = \frac{\partial Loss}{\partial y_j}\frac{\partial y_j}{\partial u_j}\frac{\partial u_j}{\partial w_{ij}} \tag{6.55}$$

式中，u_j为隐藏层第 j 个神经元的输入，即

$$u_j = \sum_{i=0}^{n} w_{ij} x \tag{6.56}$$

注意：隐藏层第 j 个神经元与输出层的各个神经元都有连接，即$\frac{\partial Loss}{\partial y_j}$涉及所有的权值$w_{ij}$，因此

$$\frac{\partial Loss}{\partial y_j} = \sum_{k=0}^{m}\frac{\partial Loss}{\partial z_k}\frac{\partial z_k}{\partial v_k}\frac{\partial v_k}{\partial y_j} = -\sum_{k=0}^{m}(t_k - z_k)g'(v_k)w_{jk} \tag{6.57}$$

于是

$$\frac{\partial Loss}{\partial w_{ij}} = -\sum_{k=0}^{m}\left[(t_k - z_k)g'(v_k)w_{jk}\right]f'(u_j)x_i \triangleq -\delta x \tag{6.58}$$

因此，从输入层到隐藏层的权值调整迭代公式为

$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_j x_i \tag{6.59}$$

（5）判断是否满足终止条件

根据实际需求设置合适的终止条件，例如训练误差是否以及达到一个很小的值，若满足则停止，否则重复步骤 2~4。

例 6.7 风电功率预测技术是指对未来一段时间内风电场所能输出的功率大小进行预测，以便安排调度计划。可利用 BP 神经网络算法对风电功率进行预测，假设有三层 BP 神经网络结构如图 6.18 所示，使用 Sigmoid 函数作为网络的激活函数。模型输入变量为与风电功率相关的风速 v，输入层到隐藏层的权值和阈值为 w_h、θ_h，隐藏层到输出层的权值和阈值为 w_o、θ_o。隐藏层的输出为 y_h，输出层的输出为风电功率 p_v，对神经网络的初始赋值如表 6.6 所示，试计算该网络第一次迭代的过程。

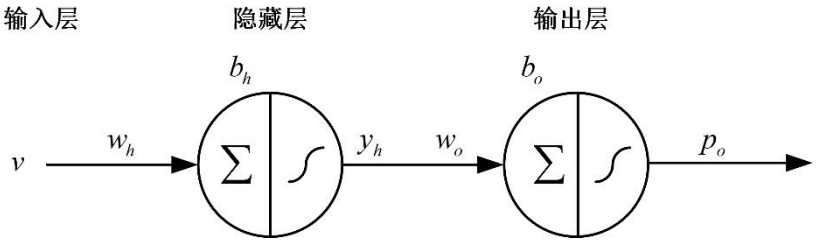


图 6.18 风电功率 BP 神经网络示意图

表 6.6 网络参数初始化

v	p	w_h	w_o	θ_h	θ_o	η
0.7	1	0.3	0.2	-0.1	-0.2	0.5

（1）信号前向传播，计算各节点输出

1）输入层到隐藏层

先计算隐藏层神经元的输入

$$z_h = w_h v - \theta_h = 0.3\times 0.7 + 0.1 = 0.310$$

隐藏层神经元输出为

$$y_h = f(z_h) = \frac{1}{1+e^{-z_h}} = \frac{1}{1+e^{-0.31}} = 0.577$$

2）隐藏层到输出层

先计算输出层神经元的输入

$$z_o = w_o y_h - \theta_o = 0.2\times 0.577 + 0.2 = 0.315$$

隐藏层神经元输出为

$$p_o = f(z_o) = \frac{1}{1+e^{-z_o}} = \frac{1}{1+e^{-0.315}} = 0.578$$

（2）计算输出误差

$$Loss=\frac{1}{2}(p-p_o)^2=\frac{1}{2}(1-0.578)^2=0.089$$

（3）误差反向传播过程，更新参数

1）计算输出层节点的误差率

由链式传导法则，误差对权值 w_o 的偏导数为

$$\frac{\partial Loss}{\partial w_o}=\frac{\partial Loss}{\partial p_o}\frac{\partial p_0}{\partial z_o}\frac{\partial z_0}{\partial w_o}=-0.422\times0.244\times0.577=-0.059$$

其中，

$$\frac{\partial Loss}{\partial p_o}=-(p-p_o)=-(1-0.578)=-0.422$$

$$\frac{\partial p_o}{\partial z_o}=p_o(1-p_o)=0.578\times(1-0.578)=0.244$$

$$\frac{\partial z_o}{\partial w_o}=y_h=0.577$$

同理，误差对阈值 θ_o 的偏导数为

$$\frac{\partial Loss}{\partial \theta_o}=\frac{\partial Loss}{\partial p_o}\frac{\partial p_0}{\partial z_o}\frac{\partial z_0}{\partial \theta_o}=-0.422\times0.244\times(-1)=0.127$$

其中，$\frac{\partial z_o}{\partial \theta_o}=-1$。

2）计算隐藏层节点的误差率

由链式传导法则，误差对权值 w_h 的偏导数为

$$\frac{\partial Loss}{\partial w_h}=\frac{\partial Loss}{\partial p_o}\frac{\partial p_0}{\partial z_o}\frac{\partial z_0}{\partial y_h}\frac{\partial y_h}{\partial z_h}\frac{\partial z_h}{\partial w_h}=-0.422\times0.244\times0.2\times0.244\times0.7=-0.00317$$

其中，

$$\frac{\partial z_o}{\partial y_h}=w_o=0.2$$

$$\frac{\partial y_h}{\partial z_h}=y_h(1-y_h)=0.577\times(1-0.577)=0.244$$

$$\frac{\partial z_h}{\partial w_h}=v=0.7$$

同理，误差对阈值 θ_h 的偏导数为

$$\frac{\partial Loss}{\partial \theta_h}=\frac{\partial Loss}{\partial p_o}\frac{\partial p_0}{\partial z_o}\frac{\partial z_0}{\partial y_h}\frac{\partial y_h}{\partial z_h}\frac{\partial z_h}{\partial \theta_h}=-0.422\times0.244\times0.2\times0.244\times(-1)=0.00502$$

其中，$\frac{\partial z_h}{\partial \theta_h}=-1$。

3）更新各节点权值与阈值

由梯度下降算法，下面对 w_h、w_o、θ_h、θ_o 的值进行更新：

$$w_o^{(1)} = w_o^{(0)} - \eta \frac{\partial Loss}{\partial w_0} = 0.2 - 0.5 \times (-0.059) = 0.230$$

$$\theta_o^{(1)} = \theta_o^{(0)} - \eta \frac{\partial Loss}{\partial \theta_0} = -0.2 - 0.5 \times (0.127) = -0.264$$

$$w_h^{(1)} = w_h^{(0)} - \eta \frac{\partial Loss}{\partial w_h} = 0.3 - 0.5 \times (-0.00317) = 0.30158$$

$$\theta_h^{(1)} = \theta_h^{(0)} - \eta \frac{\partial Loss}{\partial \theta_h} = -0.1 - 0.5 \times (0.00502) = -0.10251$$

以上完成了一次完整的训练，新的权值和阈值会改变神经网络的输出值，使其更接近期望值。随着训练次数的增加，误差会越来越小。

BP 神经网络是目前应用最为广泛的神经网络之一，具有高度的自组织和学习能力、良好的鲁棒性和容错性，而且可以实现大规模数据的并行处理。但 BP 神经网络收敛速度慢，并且容易陷入局部最优解。针对 BP 神经网络的上述问题，在局部极小值方面，可在优化初值选取和改变网络结构等方向对此进行改善；在收敛速度方面，可以采用自适应学习率和引入陡度因子等方法加快训练速度。

6.6 分类模型的评价与选择

一个性能良好的分类模型既要能够很好地拟合训练数据集，而且还应当尽可能准确地预测未知样本的类标签。为了检验分类器在未知样本上的表现，应使用未参与训练的样本测试集对其进行验证。本节将介绍分类器的性能评价指标、评价方法以及如何选择恰当的分类器。

6.6.1 分类器评价指标

对分类器进行评价，首先需要熟悉一些术语，这些术语汇总于混淆矩阵中。混淆矩阵（Confusion Matrix）是一种评价分类器性能的常用方法，它描述的是样本数据的实际类别和预测类别之间的关系。一个简单二分类问题的混淆矩阵如表 6.7 所示。

表 6.7 二分类结果的混淆矩阵

混淆矩阵		实际类别	
		正例	反例
预测类别	正例	*TP*	*FP*
	反例	*FN*	*TN*

其中 *TP*、*FP*、*TN*、*FN* 解释如下：

1）真正例（*TP*）：表示预测结果为正，实际为正的样本数。

2）假正例（*FP*）：表示预测结果为正，实际为负的样本数。

3）真反例（*TN*）：表示预测结果为负，实际为负的样本数。

4）假反例（*FN*）：表示预测结果为负，实际为正的样本数。

混淆矩阵展示了分类模型检验的结果，但不够直观。需要使用更多性能度量对模型预测结果进行评价。下面介绍一些常用的分类器度量指标：

对于样本类别分布相对平衡的数据集，通常选择**准确率**和**错误率**对分类器性能进行评估：

（1）准确率（Accuracy）

分类器在检验集上的准确率是被该分类器正确分类的样本所占的百分比。即

$$Accuracy=\frac{TP+TN}{P+N} \tag{6.60}$$

（2）错误率（Error Rate）

与准确率同理，可定义分类器的错误率如下

$$Error\ rate=\frac{FP+FN}{P+N} \tag{6.61}$$

对于类不平衡问题，准确率与错误率有时并不能准确反映分类器的分类效果。例如对一批工件进行缺陷检测，假设已经训练出一个分类模型，测试集包含 98 个正常工件与 2 个故障工件，分类器预测结果如下：

表 6.8 工件故障检测结果

混淆矩阵		实际类别	
		正常	故障
预测类别	正常	97	2
	故障	0	1

对上例计算准确率，$Accuracy=98\%$，分类的结果似乎很不错。但我们可以发现分类模型对故障样本的识别能力不佳，3 个故障样本中仅 1 个被识别。因此，需要采用其它的度量指标，**灵敏性**和**特效性**往往用于类不平衡问题的分类度量：

（3）灵敏性（Sensitive）

灵敏性用于评估分类器正确地识别正样本的情况，定义如下

$$Sensitivity=\frac{TP}{P} \tag{6.62}$$

（4）特效性（Specificity）

同理，特效性用于评估分类器正确地识别负样本的情况，定义如下

$$Specificity = \frac{TN}{N} \tag{6.63}$$

对于前述工件检测问题，可计算该分类器灵敏性为 100%，特效性为 33%。较低的特效性反映出分类模型对故障工件（负样本）的识别效果很差。除了灵敏性和特效性，在类不平衡问题中也可采用**精度**、**召回率**及 ***F* 度量**作为指标，它们用于衡量模型对正样本的识别能力。

（5）精度（Precision）

精度是预测为正的样本中预测正确的样本所占的比例

$$Precision = \frac{TP}{TP + FP} \tag{6.64}$$

（6）召回率（Recall）

召回率是预测为正的样本在实际正样本中所占的比例，其实它就是灵敏性

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{6.65}$$

需要注意的是，精度和召回率是相互矛盾的。一般情况下，当精度高时，召回率往往偏低；当召回率高时，精度又会偏低。通常只有在一些简单的分类任务中，才可能是精度和召回率都很高。

（7）*F* 度量

由于精度和召回率是一对矛盾的度量，*F* 度量是精度和召回率的调和均值，它赋予二者相等的权重，定义如下

$$F_1 = \frac{2 \times precision \times recall}{precision + recall} \tag{6.66}$$

然而在很多时候对精度和召回率的重视程度并不相同，常常也使用 F_β 度量方法。F_β 度量是精度和召回率的加权度量，它赋予召回率权重是精确率的 β 倍，定义如下

$$F_\beta = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall} \tag{6.67}$$

其中，β 是非负实数。

6.6.2 分类器评价方法

在简要介绍分类模型的性能评价指标之后，需要利用它们对分类器的分类结果进行评估。通常使用测试集对分类模型进行测试和验证。故训练集和测试集的划分方法对分类模型的性能评估结果影响较大。本节将介绍三种常用的分类器评

价方法：保持法、交叉验证法、自助法。

（1）保持法

保持法（Hold out）是最常用的评价方法之一，将给定的样本数据随机地划分为两个集合，即训练集和验证集为了保证随机性，需将数据集进行多次随机划分，对多次划分所得的性能指标取平均值，作为评价分类器的依据。

然而保持法训练出的模型依赖于训练集的划分。训练集越小，模型方差越大；训练集太大，则验证集就少了，估计出来的准确率不太可靠。一般情况下，2/3 的数据分配到训练集，其余 1/3 的数据分配到验证集。

（2）交叉验证

若数据有限，采用保持法容易导致过拟合，可采用交叉检验。最常用的交叉检验为“k 折交叉检验”。将原始数据集随机划分为 k 个（$k>1$）互斥的子集 $D_1, D_2, \ldots, D_k$，每个子集的大小近似相等。每次选择其中之一作为测试集，其余 $k-1$ 个子集作为训练集。例如，第一次迭代时，使用子集 D_1 作为测试集，其余子集一起作训练集；第 i 次迭代时，使用子集 D_i 作为测试集，其余子集一起作训练集。这样得到 k 组性能指标，对它们取平均即可。交叉检验的基本思路如图 6.19 所示（k=10）。

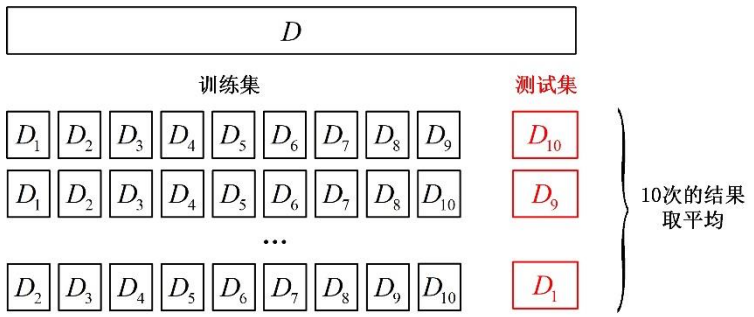


图 6.19　10 折交叉检验示意图

在交叉验证中，每个样本都参与了分类模型的训练。在数据集有限的情况下使用交叉验证，相当于同时训练 k 个模型取均值，也相当于扩充了数据集。一般建议采用 10-折交叉检验估计准确率。

（3）自助法

自助法从给定样本数据中有放回的均匀抽样，即每次随机地从原始数据集中抽取一个样本，然后放回。k 次抽样后便得到含有 k 个样本的数据集，将其作为训练集，而其它没有抽到的数据则构成测试集。需要注意，某个样本被抽到再放回后，后续仍有可能被重复抽到。当数据集过小，不适合用保持法和交叉验证法划分数据集时，一般考虑自助法。

6.6.3 分类器选择方法

若通过上述方法由数据集产生了多个分类模型，需要比较各分类模型之间的性能优劣，选择最合适的分类模型。统计显著性检验、ROC 曲线都是常用的分类器选择方法。

对于不同的分类器，实验得出其性能上的差异有可能只是偶然的。我们不妨假设两个分类器分类结果相同，对它进行**统计显著性检验**，根据检验结果判断是否可以推翻假设。

（1）统计显著性检验

较常用的显著性检验是 *t*-检验，为便于讨论，本节以错误率为性能度量。

现由同一个训练集训练了分类模型 M_1 和 M_2，采用 10-折交叉检验的方法测试模型，得到了每个模型的平均错误率。假设：两个分类模型分类结果相同，即二者平均错误率之差为 0，进行 *t*-检验。

对 M_1 和 M_2 使用相同的检验集，在交叉验证的每一轮都计算得出 M_1 和 M_2 的错误率。设 $err(M_1)_i$（或 $err(M_2)_i$）是模型 M_1（或 M_2）在交叉验证第 i 轮的错误率，最终对 M_1 和 M_2 的错误率分别取平均值得到 $\overline{err}(M_1)$ 和 $\overline{err}(M_2)$。

t-检验计算 k 个样本具有 $k-1$ 自由度的 *t*-统计量，如（6.68）所示：

$$t = \frac{\overline{err}(M_1) - \overline{err}(M_2)}{\sqrt{\text{var}(M_1 - M_2)/k}} \tag{6.68}$$

其中 $\text{var}(M_1 - M_2)$ 是两个模型差的方差，计算如下

$$\text{var}(M_1 - M_2) = \frac{1}{k}\sum_{i=1}^{k}\left[err(M_1)_i - err(M_2)_i - \overline{err}(M_1) + \overline{err}(M_2)\right]^2 \tag{6.69}$$

计算 t 并选择显著水平 sig，通常使用 5%或 1%的显著水平。在统计学相关资料中查阅 *t*-分布表，假设采用 5%的显著水平，由于 *t*-分布是对称的，表格通常只显示分布上部的百分点，因此找置信界 $z = sig/2 = 0.025$ 的表值。若 $t > z$ 或 $t < -z$，则 t 落在拒斥域，可拒绝两个分类模型相同的原假设，说明两个模型之间有显著差别，在此情况下可选择具有较低错误率的模型。否则，不能拒绝原假设，说明两个模型之间的测试差异可能只是偶然因素导致的。

注意，若有两个检验集，则两个模型之间的方差估计为

$$\text{var}(M_1 - M_2) = \sqrt{\frac{\text{var}(M_1)}{k_1} + \frac{\text{var}(M_2)}{k_2}} \tag{6.70}$$

其中，k_1 和 k_2 分别用于 M_1 和 M_2 的交叉验证样本数，查表时自由度取两个模型的最小值。

接下来介绍一种分类器性能的可视化工具：**ROC 曲线**。ROC 曲线全称是接收者操作特征（Receiver Operating Characteristic，ROC）。在机器学习中，ROC

曲线一般针对二分类问题，曲线下方的面积主要反映分类器的准确率。

（2）ROC 曲线

ROC 曲线多用于二分类任务中，它是以真正例率 *TPR* 为纵坐标，假正例率 *FPR* 为横坐标绘制的曲线。其中，*TPR* 表示分类器正确地识别正样本的比例，即

$$TPR = \frac{TP}{P} \tag{6.71}$$

FPR 表示模型将负样本错误标记为正的比例，即

$$FPR = \frac{FP}{N} \tag{6.72}$$

绘制 ROC 曲线之前，首先将测试样本按降序排列，将最有可能属于正类的数据放在列表的顶部，而最不可能属于正类的样本放在列表底部，按正类概率大小进行递减排序。

从坐标系左下角（原点）开始（$TPR = FPR = 0$），检查列表顶部的样本数据的实际类别。若它是真正例，则 *TP* 增加，从而纵坐标 *TPR* 增加，坐标向上移动，并绘制一个点。若它是假正例，则 *FP* 增加，从而横坐标 *FPR* 增加，坐标向右移动，并绘制一个点。该过程对每个数据样本重复，最终可得到完整的 ROC 曲线。

AUC（Area Under Curve）即 ROC 曲线下方的面积，该曲线下方的面积大小与分类模型的优劣密切相关，反映模型正确分类的统计概率。一般来说，AUC 的取值在 0.5~1 之间，AUC 的值越接近于 1，说明该模型的准确率越好；当 AUC 的值越接近于 0.5，即 ROC 曲线越接近于对角线，说明模型的准确率越低。

图 6.20 为两个不同分类模型 M_1 和 M_2 测试结果绘制的 ROC 曲线图，对比可明显看出 M_1 的 AUC 值较小，其 ROC 曲线接近于对角线，说明分类模型 M_1 的分类效果较差，分类模型 M_2 的分类准确性更高。

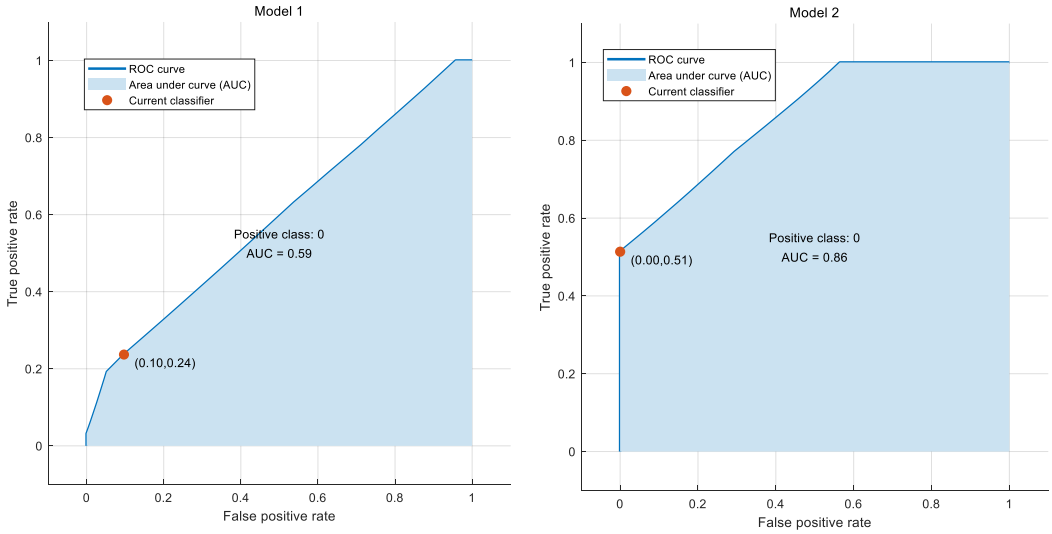


图 6.20 不同模型的 ROC 曲线示意图

6.7 组合分类技术

组合分类器是由多个分类器组合而成的复合模型，将多个学习器进行结合，往往能提高分类的准确性，常可以获得比单一学习器显著优越的泛化性能。本章将详细介绍装袋、提升和随机森林等应用较为广泛的组合分类方法。

6.7.1 组合分类方法简介

组合分类的一般流程如图 6.21 所示，将样本数据集 D 划分为 k 个独立的训练集 $D_1, D_1, \cdots, D_k$，其中训练集 D_i 用于创建分类器 M_i，由此可以得到 k 个分类模型（也称作基分类器）。将它们按某种策略进行组合，创建一个复合分类模型 M。对于给定的待分类样本数据，每个基分类器通过投票预测类别。若超过半数的基分类器预测类别正确，则组合分类就正确。当且仅当一半以上的基分类器都分类错误时，组合分类器才会误分类，所以组合分类器往往比基分类器更加准确。

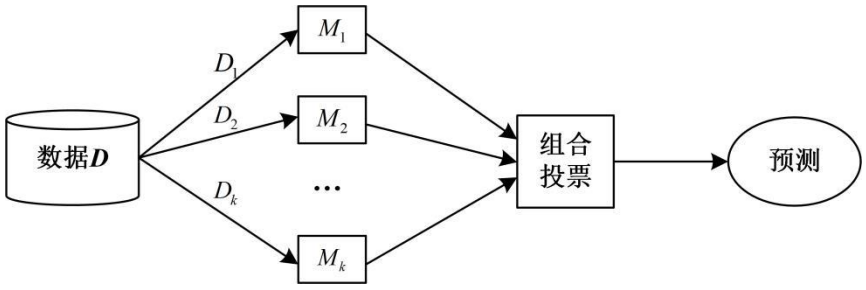


图 6.21 组合分类的基本原理

较常用的组合分类方法有 Bagging、Boosting、Stacking 等，简要介绍如下：

（1）Bagging 算法

Bagging 算法基于本章 6.6.2 中提到的自助法，它使用训练集的子集和并行学习的模型来训练每个模型。Bagging 通常对分类任务使用简单投票法，即每个基分类器使用相同的权重投票，最终将票数最高的类赋予样本数据。应用广泛的随机森林实质上 Bagging 的变体，是一种用于决策树的 Bagging 方法。

（2）Boosting 算法

Boosting 算法先从初始训练集训练出一个基分类器 M_i，更新样本权重，使分类器 M_i 更加关注被 M_i 上一个分类器错误分类的训练样本。如此反复更新，直到基分类器的数目达到指定值，最终将这些分类器进行加权集合。

（3）Stacking 算法

Stacking 指训练一个用于组合所有个体分类器的有层次的模型，即首先训练多个不同的基分类器，然后再以这些基分类器的输出作为下一阶段的输入来训练一个二级分类器，依此类推，最终得到多层复合模型。

6.7.2 装袋算法过程

Bagging 算法过程如下：在原始数据集中随机有放回地抽样 m 次，则可以得到一个含 m 个数据样本的训练集 D_1。照这样重复 k 次便可得到 k 个训练集 $D_1, D_2, \ldots, D_k$。由训练集 D_i 学习得到分类模型 M_i。

对一个未知的样本 X 分类，每个基分类器投票，装袋分类器统计得票，将得票数最高的类作为最终的预测结果。若出现两种类别收到相同票数，则随机选择一个即可。

Bagging 算法过程如下：

输入：训练样本集 D，基分类器个数 k，基分类算法，未知样本 $\boldsymbol{X}$

输出：一个复合分类模型，样本 $\boldsymbol{X}$ 预测类别

---

1. for i = 1 to k do；
2. 对 D 进行有放回抽样，创建训练集 D_i
3. 使用 D_i 与基分类算法导出分类模型 M_i
4. 使用 M_i 预测样本 X 的类别，算作一票
5. end for
6. 将票数最多的类别视为 $\boldsymbol{X}$ 所属类别

装袋分类器的准确率通常高于单个分类器的准确率，且 Bagging 主要关注于降低基分类器的方差，它在不剪枝决策树、人工神经网络等易受样本扰动的学习器上作用更为明显。

6.7.3 提升算法过程

在提升（boosting）方法中，对每个训练样本赋予初始权重，并迭代地学习 k 个分类器。如果样本被分类错误，则它的权重增加；如果样本正确分类，则它的权重减小，因此样本数据的权重大小反映了对它们分类的困难程度。在训练新的分类器时更加关注被上一个分类器错误分类的样本数据，这样建立了一个互补的分类器系列。本节主要介绍提升算法中较为主流的 Adaboost。

Adaboost 的主要思想如下：给定数据集 D，$D=\{(x_1,y_1),(x_2,y_2),\ldots(x_m,y_m)\}$。首先对每个训练样本赋予相等的权重 $1/m$，接着从 D 中有放回地抽样，得到训练集 D_1，通过训练集 D_1 构造分类器 M_1。然而使用 D 作为检验集计算 M_1 的错误率，根据分类情况调整训练样本的权重，然后使用调整后的 D 继续构造下一轮分类器，直到达到基分类器数量上限。

分类器 M_i 的错误率计算如下

$$error(M_i)=\sum_{j=1}^{m} w_j \times h(x_j) \tag{6.73}$$

其中 h_j 是样本 x_j 的权重。$err(x_j)$ 是样本 x_j 的误分类误差。若 x_j 被误分类，则 $err(x_j)=1$，否则 $err(x_j)=0$。若分类器 M_i 的错误率大于 0.5，则丢弃，重新产生新的 D_i 和 M_i。

训练数据 D 的权重调整规则如下：若一个样本在第 i 次被正确分类，则其权重乘以 $error(M_i)/(1-error(M_i))$。当所有被正确分类的样本权重都已更新，对所有样本权重进行规范化，使权重之和保持不变。

当所有分类器都构造完毕，根据各个分类器的分类情况赋予相应的投票权重。分类器的错误率越低，则它的投票权重就应该越高。可以使用式（6.74）所示的分类器权重计算方法：

$$w_i=\log\frac{1-error(M_i)}{error(M_i)} \tag{6.74}$$

经过所有分类器的加权投票之后，选择具有最大权重的类作为预测结果。

Adaboost 的算法流程如下：

输入：训练样本集 $D=\{(x_1,y_1),(x_2,y_2),\ldots(x_m,y_m)\}$，训练轮数 k，基学习算法，未知样本 $\boldsymbol{X}$

输出：复合分类模型，样本 $\boldsymbol{X}$ 预测类别

训练过程：

1. 初始化每个样本的权重 $h_j=\dfrac{1}{m}$

---

2. for i = 1 to k do
3. 对 D 进行有放回抽样，创建训练集 D_i
4. 使用 D_i 与基学习算法导出分类模型 M_i
5. 以 D 作为测试集计算 $error(M_i)$
6. if $error(M_i) > 0.5$ then
7. 终止循环
8. end if
9. for D 中每个被正确分类的样本 do
10. 更新权重 $h_j^{(i+1)} = h_j^{(i)} \times \frac{error(M_i)}{1-error(M_i)}$
11. 规范化 D 中每个样本的权重
12. end for

预测过程：

1. for i = 1 to k do
2. 对每个类的权重初始化为 0
3. 各个分类器投票权重 $w_i = \log\frac{1-error(M_i)}{error(M_i)}$
4. M_i 对 X 的类别预测 $c = M_i(X)$
5. 将 w_i 加到类别 c 的权重
6. end for
7. 将权重最大的类别作为 $\boldsymbol{X}$ 所属类别

与 Bagging 算法相比，Adaboost 算法充分考虑了每个分类器的权重，大部分情况下 Adaboost 得到的结果精度更高、偏差更小，然而 Adaboost 较为关注被误分类的样本，存在结果对样本数据集过分拟合的风险。

后续一些提升算法对此进行了改进，使用较为广泛的有 Xgboost 算法。Xgboost 由华盛顿大学的陈天奇博士提出，在 Kaggle 竞赛中使用，具有防过拟合效果更好、损失函数更精确、处理稀疏矩阵效率更高等优点，感兴趣的读者可查阅相关论文深入学习。

6.7.4 随机森林

随机森林（Random Forest，RF）是 Bagging 方法的一个扩展变体，它是一种用于决策树的 Bagging 方法。

若给定训练样本集 $D=\{(x_1,y_1),(x_2,y_2),\ldots(x_m,y_m)\}$，为组合分类器产生 k 棵决策树的一般过程如下：在第 i 次迭代中，使用有放回抽样，由 D 产生含 d 个样本的训练集 D_i，以此构造决策树 M_i。假设在决策树当前节点有 n 个可用属性，

在此节点属性中随机挑选 f 个属性（f 一般远小于 n），在 f 个属性中选择一个最优划分属性进行划分。使树增长到最大规模，不剪枝。照此重复 k 次便可得到 k 棵决策树，分类时每棵树都投票给它预测的类别，最终得票最多的类即视为预测结果。

随机森林的起始性能往往较差，因为各节点属性挑选的随机性，个体决策树的性能有所降低。随着个体决策树数目的增加，随机森林通常会收敛到更低的泛化误差。

另外，随机森林的性能对每次划分所考虑的属性数 f 较为敏感，按经验，通常选取 $f=\log_2 n+1$ 个属性。随机森林的训练效率往往优于 Bagging，因为在构建个体决策树时，Bagging 构建的是“确定型”决策树，需要对节点的所有可用属性进行考察。而随机森林仅需要在一个规模较小的属性子集中考察。故随机森林计算难度较小，相对于 Bagging 和 Adaboost，它的训练速度更快。

6.8 实例

电力变压器是整个电力设备中最关键的组成部分之一，其运行可靠性直接影响电力系统的安全运行。然而变压器在运行过程中，受到制造水平、温度、污染等影响，不可避免地会出现绝缘劣化及潜伏性故障或缺陷。随着故障的缓慢发展，变压器油与油中的固体有机绝缘材料在运行电压下因电、热、氧化等多种因素作用会逐渐变质，裂解成低分子气体。裂解出来的气体形成气泡在油中经过对流、扩散作用，就会不断地溶解在油中。由此可见，油中溶解气体的成分和含量在一定程度上反映出变压器的内部故障，可作为识别变压器故障类型的特征量。

电力变压器的内部故障一般可分为两类：过热故障和放电故障。主要用于分析的气体种类为：H_2、CH_4、C_2H_6、C_2H_4、C_2H_2。通过检测变压器油中上述五类气体的含量特征可以判断电力变压器的内部故障类型。

现对一批电力变压器样本的油中气体含量进行检测，得到七十组实验数据如表所示，挑出其中 50 组样本作为训练集，其余 20 组样本作为测试集。试选用合适的分类方法建立分类模型，并测试分类模型的性能。

50 组样本训练集如表 6.9 所示：

表 6.9 变压器故障训练样本

序号	H_2	CH_4	C_2H_6	C_2H_4	C_2H_4	故障类别
1	44.3	17.3	3.6	23.3	10.4	放电故障
2	673.6	423.5	77.5	988.9	344.4	放电故障
3	550	53	34	20	0	放电故障
4	13.6	5.3	8.2	29	2.1	正常
5	4670	3500	2120	5040	2560	放电故障

6	26.6	22.7	22.5	109	0	过热故障
7	274	376	55	1002	17	过热故障
8	80	20	6	20	62	放电故障
9	170	320	53	520	3.2	过热故障
10	5.8	2.7	1.8	0.8	0	正常
11	42.1	28	10	51	5.8	放电故障
12	56	285	96	28	7	过热故障
13	200	48	14	117	131	放电故障
14	0.33	0.26	0.04	0.27	0	正常
15	14	11	3	22	1	正常
16	53.6	17.7	13.2	5	0	正常
17	39.1	24.50	18.37	11.43	6.53	正常
18	20.6	19.89	7.4	61.27	1.51	过热故障
19	110	5	2.7	3.2	1.7	正常
20	160	130	33	96	0	过热故障
21	259	863	393	994	6	过热故障
22	33.66	2.9703	33.17	27.72	2.48	正常
23	15	12	5.3	3.2	0.2	过热故障
24	220	340	42	480	14	正常
25	56	286	96	928	7	过热故障
26	30	110	137	52	22.3	正常
27	19	10	7.9	9.5	8.5	放电故障
28	350	1001	298	1001	7.9	过热故障
29	565	93	34	47	0	放电故障
30	217.5	40	4.9	51.8	67.5	放电故障
31	2.4	1.4	1	0.3	0	正常
32	57	77	19	21	0	过热故障
33	60	40	6.9	110	70	放电故障
34	25.1	411.91	320.9	1832.8	18.4	过热故障
35	78	161	86	353	10	放电故障
36	181	262	210	528	0	过热故障
37	120	120	33	84	0.55	过热故障
38	33	26	6	5.3	0.2	正常
39	428	1660	533	4094	11.4	过热故障
40	90	149	32.4	486	19.2	过热故障
41	93	58	43	37	0	过热故障
42	335	67	18	143	170	放电故障
43	172.9	334.1	172.9	812.5	37.7	过热故障
44	32	11.38	8.2	30.6	3.9	正常
45	115.9	75	14.7	25.3	6.8	放电故障
46	54	7	7.4	8.6	5.4	放电故障

47	7.5	5.7	3.4	2.6	3.2	正常
48	59	28	9	70	15	放电故障
49	80	10	4	1.5	0	正常
50	170	24	7	17	54	放电故障

20 组测试数据表 6.10 所示：

表 6.10 变压器故障测试样本

序号	H_2	CH_4	C_2H_6	C_2H_4	C_2H_4	故障类别
1	46	37.2	8.3	107	71.9	放电故障
2	0	0.8	0.5	0.4	0	正常
3	3.9	1.3	1.5	0.8	0	正常
4	1678	652.9	80.7	1005.9	419.1	放电故障
5	48	38.4	91.6	5.3	4.8	正常
6	46.81	36.88	8.51	7.52	0.28	正常
7	9259	8397	26782	10497	-1	过热故障
8	42.02	33.61	14.85	8.96	0.56	正常
9	164	244	103	497	8.3	过热故障
10	46.13	11.57	33.14	8.52	0.63	正常
11	5405	561	5	70	128	放电故障
12	747	2065	1029	4589	6.4	过热故障
13	0	1.18	1.6	1.1	0.6	正常
14	166.6	28.31	6.72	12.4	0.3	过热故障
15	4.32	193	118	125	0	过热故障
16	345	112.25	27.5	51.5	58.75	放电故障
17	260	130	26	84	92	放电故障
18	70.4	69.5	28.9	241.2	0.4	过热故障
19	189	157	17	62	7.4	放电故障
20	27	90	42	63	0.2	过热故障

6.8.1 数据预处理

在本实例中，主要对缺失值和特征之间量纲不一致这两个问题进行处理。

1）缺失值处理

通过 isna(). sum()方法查看了缺失值的情况，未发现缺失值，故不用进行缺失值处理。

2）标准化处理

为了降低各特征维度不一致（取值大小范围不一致）给分类模型造成的影响，通过 StandardScaler 方法对各个特征进行标准化处理。

6.8.2 参数设置

如果超参数选择不恰当，就会出现欠拟合或者过拟合的问题。本节通过网格搜索算法确定决策树模型以及 BP 神经网络的参数，通过遍历给定的参数组合来优化模型表现的方法。

1）决策树模型

决策树模型主要涉及到的参数如下：

criterion：用来决定不纯度的计算方法，可取的值为 gini 或者 entropy，即基尼系数或者熵值。

splitter：创建决策树分支选项。可取的值为 best 或者 random，前者是所有特征中找最佳的切分点，后者是在部分特征中寻找（数据量大的时候）。

max_depth：树的最大深度限制, 当树的深度到达 max_depth 时决策树都会停止运算。本次实例中，max_depth 设置的取值范围为[5, 10, 20, None]。

min_samples_split：分裂所需的最小样本数量。当叶节点的样本数量小于该参数后, 则不再生成分支。本次实例中，设置的取值范围为[2, 5, 8, 10]。

采用网格搜索方法遍历搜索最优的参数，其中，决策树的 random_state=3；cv=5，即采用了 5 折交叉验证；最终，得到最优的参数，criterion 的取值为 entropy；max_depth 的取值为 5；min_samples_split 的取值为 2；splitter 的取值为 random。

最终生成的决策树模型如图 6.22 所示

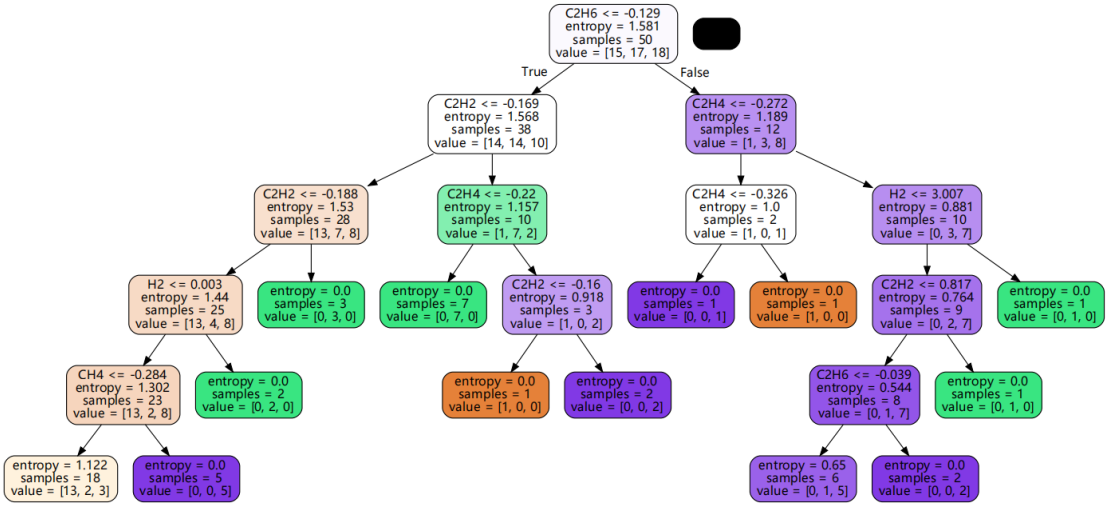


图 6.22 决策树模型

2）BP 神经网络

BP 神经网络主要涉及到的参数如下：

hidden_layer_sizes：隐藏层的层数及神经元数量。设置的取值范围为[(100,), (100, 30)]，例如 hidden_layer_sizes=(100, 30)，表示有两层隐藏层，第一层隐藏层有 100 个神经元，第二层也有 30 个神经元。

solver：默认 adam，用来优化权重。

max_iter：最大迭代次数，默认 200，本次实例设置其取值为 20。

经过网格搜索算法遍历所有的参数，得到的 BP 神经网络的最优参数为：hidden_layer_sizes= (100,), max_iter= 20, solver=adam。此外，涉及到的其他参数均采用默认值。

6.8.3 模型指标比较

为描述方便，在后面的讨论中，分别将正常、放电故障、过热故障简记为 0，1，2 三个类别。

1）混淆矩阵

决策树和 BP 神经网络得到的混淆矩阵如图 6.23 所示。

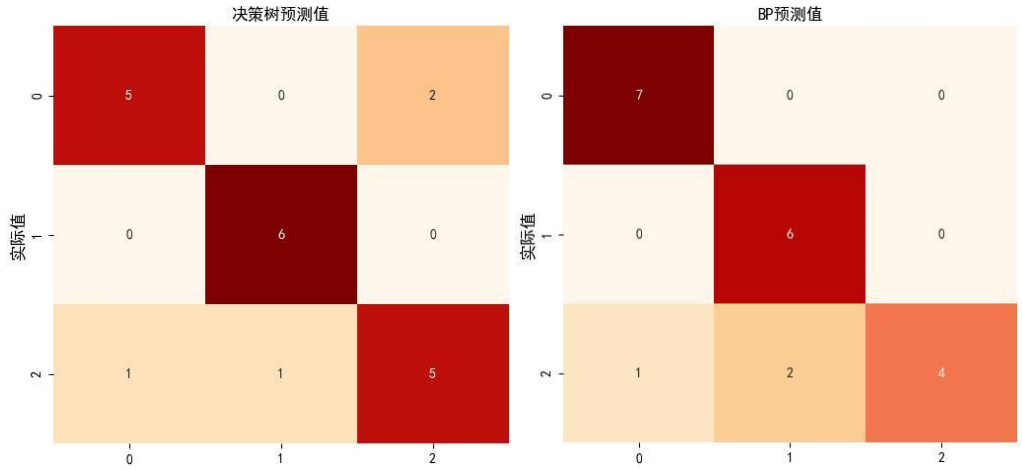


图 6.23　决策树与 BP 神经网络的混淆矩阵

由混淆矩阵可以看出，被 BP 误分类的样本数略少于决策树，但二者对过热故障的识别均出现一些偏差。

2）准确率、精度、召回率以及 F 度量

在决策树和 BP 神经网络模型的指标比较中，主要是采用了准确率、精度、召回率以及 F1 分数这四个指标。由于本次实例所采用的数据为 3 分类数据，因此，对每个样本标签都计算准确率、精度、召回率以及 F 度量，然后考虑各个类别数据量权重加权求和。各个指标的结果如表 6.11 所示。

表 6.11　模型指标比较

指标	决策树	BP 神经网络
准确率	0.8	0.85
精度	0.7988	0.8813
召回率	0.8	0.85
F1 度量	0.7962	0.8384

通过表 6.11 可以看出，就本例而言，以网格搜索方法遍历搜索最优的参数建立分类模型，BP 神经网络模型的分类效果优于决策树。

6.9 本章小结

分类分析是一种有监督学习，用于预测数据的所属类别。本章首先描述了分类分析的概念与基本步骤，并分别介绍了四种常用的分类方法（决策树、朴素贝叶斯、支持向量机、人工神经网络）。其次讨论了分类模型的评估与选择标准，最后介绍了组合分类方法。具体内容如下：

基本概念：介绍了分类分析的概念，并描述了分类分析的基本思路和一般过程，简要概述了决策树、贝叶斯、支持向量机、人工神经网络等常用分类方法。

决策树：介绍了基本的决策树理论、属性选择度量及剪枝操作，结合实例对常见的三种决策树算法（ID3，C4.5，CART）进行说明。

贝叶斯分类：介绍了贝叶斯定理等概率论相关基础知识，并描述了常用的朴素贝叶斯分类思想。

支持向量机：以二维空间为例，介绍了支持向量机的基本理论，引出对应的优化问题及相应解法。

人工神经网络：介绍了常用的激活函数及损失函数，对 BP 的原理及基本过程进行阐述。

分类模型评价与选择：介绍了分类器的评价指标及划分测试集的方法，来完成对分类模型效果的评估。

组合分类技术：描述了组合分类方法的一般流程，介绍了 Bagging、Adaboost、随机森林等组合分类技术。

习题

3-1 简述分类分析的基本过程。

3-2 分类分析与聚类分析有什么区别？

3-3 决策树中常用的属性度量指标有哪些？对应的经典算法有哪些？

3-4 表 6.12 所示数据取自 Titanic 数据集，包含四个属性和两个类标签，其中“舱位”取值 1、2、3，分别代表头等舱、二等舱和三等舱。“城市”取值 1、2、3，分别代表纽约、华盛顿和巴黎。“性别”取值 0 和 1，分别代表男性和女性，“登船港口”取值 0 和 1，分别代表南安普顿和英吉利海峡。类标签 0 和 1 分别表示遇难和幸存。试用 ID3 算法构建决策树模型。

表 6.12　Titanic 数据集

序号	舱位	城市	性别	登船港口	是否幸存
1	1	1	0	0	0
2	1	2	1	1	1
3	2	3	0	0	1
4	1	1	0	1	0
5	1	2	0	0	0
6	3	1	0	1	1
7	2	1	1	1	0
8	1	3	0	0	0
9	2	3	1	0	1
10	2	3	0	1	0
11	3	2	0	1	1
12	2	3	1	1	1
13	2	2	0	0	1
14	1	3	0	1	0
15	3	3	0	0	1
16	2	2	1	1	1
17	1	3	1	1	1
18	1	1	1	0	1
19	3	3	1	1	1
20	2	1	0	0	0

3-5 简述朴素贝叶斯分类的基本思想，并思考朴素贝叶斯的局限性。

3-6 根据表 6.12 中的数据，利用朴素贝叶斯算法来判断一位在英吉利海峡登船、乘坐三等舱的华盛顿女性幸存的可能性，并与决策树的分类结果作对比。

3-7 如何采用 SVM 算法实现对非线性可分训练集的分类？

3-8 表 6.13 所示数据取自 Iris 数据集，试就此分类问题编程实现 SVM 分类算法。

表 6.13　Iris 数据训练集

序号	属性 1	属性 2	类别	序号	属性 1	属性 2	类别
1	5.1	3.5	0	21	7	3.2	1
2	4.9	3	0	22	6.4	3.2	1
3	4.7	3.2	0	23	6.9	3.1	1
4	4.6	3.1	0	24	5.5	2.3	1
5	5	3.6	0	25	6.5	2.8	1
6	5.4	3.9	0	26	5.7	2.8	1
7	4.6	3.4	0	27	6.3	3.3	1
8	5	3.4	0	28	4.9	2.4	1
9	4.4	2.9	0	29	6.6	2.9	1
10	4.9	3.1	0	30	5.2	2.7	1
11	5.4	3.7	0	31	5	2	1
12	4.8	3.4	0	32	5.9	3	1
13	4.8	3	0	33	6	2.2	1

14	4.3	3	0	34	6.1	2.9	1
15	5.8	4	0	35	5.6	2.9	1
16	5.7	4.4	0	36	6.7	3.1	1
17	5.4	3.9	0	37	5.6	3	1
18	5.1	3.5	0	38	5.8	2.7	1
19	5.7	3.8	0	39	6.2	2.2	1
20	5.1	3.8	0	40	5.6	2.5	1

3-9 简要描述 BP 神经网络的训练迭代过程。

3-10 为什么 BP 神经网络易陷入局部最优解？

3-11 根据表 6.13 中的数据，试用标准 BP 算法训练一个单隐层网络。

3-12 简述训练集与测试集的划分方法，在小样本的情况下一般适合什么方法？

3-13 表 6.14 中数据样本已经按分类器返回概率值递减排序，请计算真正例率 TPR 和假正例率 FPR，并为该数据绘制 ROC 曲线。

表 6.14　分类样本及分类器返回概率值

序号	类	概率	序号	类	概率
1	P	0.95	6	P	0.57
2	N	0.85	7	N	0.55
3	P	0.78	8	N	0.52
4	P	0.63	9	N	0.51
5	N	0.60	10	P	0.42

3-14 以表 6.15 中的数据作为测试集，检验并比较 SVM 与 BP 两种分类器的分类效果。

表 6.15　Iris 数据测试集

序号	属性 1	属性 2	类别	序号	属性 1	属性 2	类别
1	5.4	3.4	0	11	5.9	3.2	1
2	5.1	3.7	0	12	6.1	2.8	1
3	4.6	3.6	0	13	6.3	2.5	1
4	5.1	3.3	0	14	6.1	2.8	1
5	4.8	3.4	0	15	6.4	2.9	1
6	5	3	0	16	6.6	3	1
7	5	3.4	0	17	6.8	2.8	1
8	5.2	3.5	0	18	6.7	3	1
9	5.2	3.4	0	19	6	2.9	1
10	4.7	3.2	0	20	5.7	2.6	1

3-15 与单个分类器相比，组合分类器有哪些优势？

3-16 思考装袋算法与提升算法的异同。

参考文献

[1] HAN J，KAMBER M，PEI J.数据挖掘概念与技术（第三版）[M].范晓明，孟小峰，译.北京：机械工业出版社，2012.

[2] 吕晓玲，谢邦昌.数据挖掘方法与应用[M].北京：中国人民大学出版社，2009.

[3] 范苑英，蒋军敏，石薇，等.大数据技术及应用[M].北京：机械工业出版社，2021.

[4] 石胜飞.大数据分析与挖掘[M].北京：人民邮电出版社，2018.

[5] Mehmed Kantardzic.数据挖掘：概念、模型、方法和算法（第 2 版）[M].王晓海，吴志刚，译.北京：清华大学出版社，2013.

[6] 周志华.机器学习[M].北京：清华大学出版社，2016.

[7] 包子阳，余继周，杨杉，等.智能优化算法及其 MATLAB 实例[M].北京：电子工业出版社，2021.

[8] Quinlan, J. R. Discovering rules by induction from large collections of examples[J]. Expert Systems in the Micro-electronics Age, 1979: 168-201.

[9] Quinlan, J. R. C4.5: Programs for machine learning[J]. Morgan Kaufmann, San Mateo, CA, 1993.

[10] Quinlan, J. R. Induction of decision trees[J]. Machine Learning, 1986, 1(1): 81-106.

[11] Li B, Friedman J, Olshen R, et al. Classification and regression trees[J]. Biometrics, 1984, 40(3): 358-361.

[12] Vapnik, V. (1998). Statistical Learning Theory. Wiley, New York.

[13] Vapnik, V. (1995). The Nature of Statistical Learning Theory. Springer, New York.

[14] Cortes, C. and Vapnik, V. (1995). Support vector networks. Machine Learning, 20(3): 273-297.

[15] Kohonen, T. (1988). An introduction to neural computing. Neural Networks, 1(1): 3-16.

[16] Haykin, S. (1998). Neural Networks: A Comprehensive Foundation, 2nd edition,

Prentice-Hall, Upper Saddle River, NJ.

[17] McCulloch, W.S. and W.Pitts. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics,5(4):115-133.

[18] Werbos, P. (1974). Beyond regression: New tools for prediction and analysis in the behavior science. Ph.D. thesis, Harvard University, Cambridge.

[19] Efron, B. and R.Tibshirani. (1993). An Introduction to the Bootstrap. Chapman&Hall, New York, NY.

[20] Spackman, K.A. (1989). Signal detection theory:Valuable tools for evaluating inductive learning. In Proceedings of the 6th International Workshop on Machine Learning,160-163, Ithaca, NY.

[21] Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation, 10(7):1895-1923.

[22] Freund, Y. and R. E. Schapire. (1997). A decision-theoretic generalization of on-line learning and an application to boosting(with discussions). Annals of Statistics, 28(2):337-407.

[23] Breiman, L. (1996a). Bagging predictors. Machine Learning, 24(2):123-140.

[24] Breiman, L. (2001a). Random forests. Machine Learning, 45(1):5-32.

[25] Chen T, Guestrin C. Xgboost: A scalable tree boosting system[C]. Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016: 785-794.