第 5 章 非线性判别分析

CHAPTER 5

在很多情况下，数据分布情况复杂，样本集非线性可分，进行线性判别分析会有较高的错误率，可以利用非线性判别实现分类。非线性判别分析并没有特定函数形式，因此不适合采用类似于线性判别函数设计的参数估计方法。不同的设计思路产生不同的方法，本章主要学习常见的几种非线性判别分析方法，包括近邻法、二次判别函数设计、决策树方法、Logistic 回归。

5.1 近邻法

近邻法是一种经典的非线性判别分析方法，采用距离度量作为判别函数，直接根据训练样本对未知类别样本进行分类决策。首先了解利用距离度量进行判别的含义，再介绍近邻法决策规则。

5.1.1 最小距离分类器

距离度量是模式识别中常用的方法，如果类别可分，则同一类样本差别相对较小，不同类的样本差别相对较大，差别可以采用数据间的距离来衡量，因此，可以设计基于距离的分类器。

对于两类情况，设两类 ω_1、ω_2 各自的均值向量为 $\boldsymbol{\mu}_1$、$\boldsymbol{\mu}_2$，待分类样本为 $\boldsymbol{x}$，计算 $\boldsymbol{x}$ 到两类均值的距离，将 $\boldsymbol{x}$ 归为距离小的那一类，即

$$\text{若 } g(\boldsymbol{x}) = \|\boldsymbol{x}-\boldsymbol{\mu}_1\|^2 - \|\boldsymbol{x}-\boldsymbol{\mu}_2\|^2 \lesseqgtr 0, \quad \boldsymbol{x} \in \begin{cases}\omega_1 \\ \omega_2\end{cases} \tag{5-1}$$

则分类决策面为 $\boldsymbol{\mu}_1$、$\boldsymbol{\mu}_2$ 连线的垂直平分面，如图 5-1 所示。

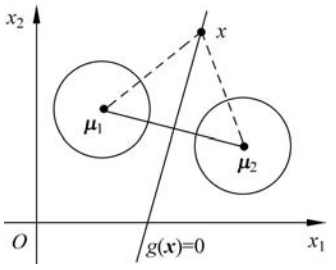


图 5-1 两类的最小距离分类器

对于正态分布模式的贝叶斯决策，在样本集分布呈现特殊情况（$P(\omega_i)=P(\omega_j)$，$\boldsymbol{\Sigma}_j=\sigma^2\boldsymbol{I}$，$i,j=1,2,\cdots,c$）时，设计的判别函数为最小距离分类器（见 2.7 节）。

如果是多类情况，则定义各类的判别函数为

$$g_j(\boldsymbol{x}) = \|\boldsymbol{x}-\boldsymbol{\mu}_j\|^2, \quad j=1,2,\cdots,c \tag{5-2}$$

决策时，哪一类的判别函数（距离）最小，则决策为哪一类，即决策规则为

$$若\ g_i(\boldsymbol{x}) < g_j(\boldsymbol{x}),\quad \forall i \neq j,\quad i,j=1,2,\cdots,c,\quad 则\ \boldsymbol{x} \in \omega_i \tag{5-3}$$

很明显，最小距离分类器仅适用于类别线性可分且类间有明显距离的特殊情况，但用均值点作为类的代表点，用距离作为判别函数的思路很有启发性。

5.1.2 分段线性距离分类器

图 5-2 所示为正常血压和高血压数据，采用最小距离分类器，设计的分界面导致很多数据被错分类，如图 5-2(a)所示。将高血压数据分为 3 个子类，各子类均值点作为该子类的代表点，距离作为判别函数，设计多类情况下的最小距离分类器，得到的分界面由多段线性分界面组成，如图 5-2(b)所示。

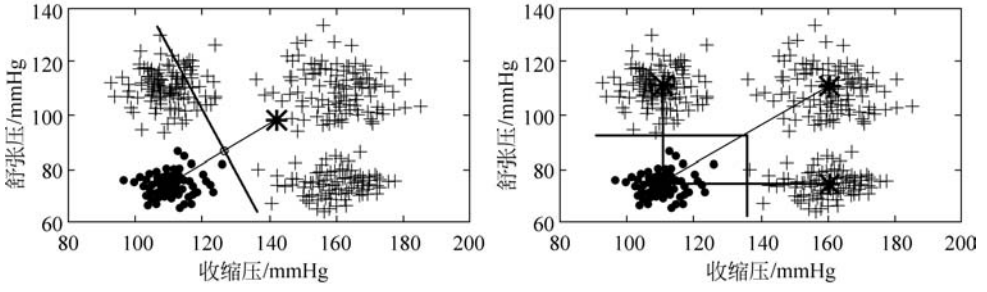


图 5-2 非线性可分情况下距离分类器示例

对于类别数据呈现多峰分布的情况，将每类划分为若干子类，验证待分类样本 $\boldsymbol{x}$ 到 ω_j 类各子类均值的距离，最小的距离作为该类的判别函数值，将样本归入最近的子类所属的类别，这种分类器称为分段线性距离分类器。

分段线性距离分类器的判别函数表示为

$$g_j(\boldsymbol{x}) = \min_{l=1,2,\cdots,c_j} \| \boldsymbol{x} - \boldsymbol{\mu}_j^l \|,\quad j=1,2,\cdots,c \tag{5-4}$$

c_j 为 ω_j 类的子类数目。

分段线性距离分类器的决策规则为

$$若\ g_i(\boldsymbol{x}) = \min_{j=1,2,\cdots,c} g_j(\boldsymbol{x}),\quad 则\ \boldsymbol{x} \in \omega_i \tag{5-5}$$

分段线性距离分类器分类效果与子类的划分有密切关系，如果对样本没有充足的先验知识，子类划分不一定合理，则会影响分类的效果。

5.1.3 近邻法及仿真实现

1. 最近邻法

将分段线性距离分类器的设计思路发展到极端，把每个样本点都作为一个子类，计算各点与待测样本 $\boldsymbol{x}$ 的距离，将 $\boldsymbol{x}$ 归入距离最近的点所在的类，这种方法称为最近邻法（Nearest Neighbor，NN）。

最近邻法的判别函数如式(5-4)所示，仅将变量 c_j 修改为 ω_j 类的样本数目 N_j；决策规则如式(5-5)所示，不再赘述。

式(5-4)计算的是欧氏距离，也可以采用不同的距离度量，或者采用相似性度量(见 7.2 节)，把最小距离变更为最大相似度量，确定最近邻，实现分类。

由于把每个样本点都作为一个子类，因此最近邻法的决策面由不同类样本点间连线的垂直平分面构成，判别函数是分段线性判别函数(Piecewise Linear Discriminant Function)。

2. k 近邻法

最近邻法根据距离待测样本最近的一个样本的类别进行归类，容易受到样本分布以及噪声的影响，导致决策错误，因此，引入投票机制，选择距离待测样本最近的 k 个样本，将待测样本归入多数近邻所在的类别，称为 k 近邻法(kNN)，最近邻法实际是 $k=1$ 的特例。

k 值需要根据样本情况进行选择，通常选择样本总数的一个很小的比例即可。两类情况下，一般选择 k 为奇数，避免两类得票相等。

k 近邻法(含最近邻法)简单易于决策，根据已知类别的样本直接对待测样本进行决策，不需要事先训练出一个判别函数，在训练样本趋于无穷时接近最优，但计算量大，耗时大，没有考虑决策的风险。

【例 5-1】 对样本集 ω_1：$\left\{\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}0\\1\end{bmatrix},\begin{bmatrix}0\\-1\end{bmatrix}\right\}$，$\omega_2$：$\left\{\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}0\\2\end{bmatrix},\begin{bmatrix}0\\-2\end{bmatrix},\begin{bmatrix}-2\\0\end{bmatrix}\right\}$进行下列处理。

(1) 采用样本均值作为两类的代表点，按最小距离法分类；

(2) 对样本 $\boldsymbol{x}=[1\quad 2]^{\mathrm{T}}$，按最近邻法分类；

(3) 对 $\boldsymbol{x}=[1\quad 2]^{\mathrm{T}}$ 按 3 近邻法分类。

解：(1) $\boldsymbol{\mu}_1=[1/3\quad 0]^{\mathrm{T}}$、$\boldsymbol{\mu}_2=[-1/2\quad 0]^{\mathrm{T}}$，$\boldsymbol{\mu}_1$和$\boldsymbol{\mu}_2$连线的垂直平分线为 $x_1=-1/12$。

(2) $g_1(\boldsymbol{x})=\min\{2,\sqrt{2},\sqrt{10}\}=\sqrt{2}$，$g_2(\boldsymbol{x})=\min\{\sqrt{5},1,\sqrt{17},\sqrt{13}\}=1$。

因为

$$g_1(\boldsymbol{x})>g_2(\boldsymbol{x})$$

所以

$$\boldsymbol{x}\in\omega_2$$

(3) 最近的三个距离为 1、$\sqrt{2}$、2，对应的 3 个近邻为$[0\quad 2]^{\mathrm{T}}$、$[0\quad 1]^{\mathrm{T}}$、$[1\quad 0]^{\mathrm{T}}$，3 个近邻中有 2 个属于 ω_1 类，所以 $\boldsymbol{x}\in\omega_1$。

【例 5-2】 导入 fisheriris 数据集，根据原理设计程序对样本$[5.5\quad 2.3\quad 4\quad 1.3]^{\mathrm{T}}$ 进行最近邻以及 5 近邻判别。

设计思路：

计算待测样本与所有样本之间的距离，将距离排序，找最小距离和前 5 个最小距离，对应的样本类别即为分类类别。

程序如下：

```
clc,clear,close all;
load fisheriris;
X = [5.5,2.3,4,1.3];          k = 5;
N = length(meas);
distance = pdist2(X,meas);                  % 计算待测样本和各样本之间的欧氏距离
[D,index] = sort(distance);                 % 距离升序排序
```

```
disp('最近邻归类:');              celldisp(species(index(1))); %输出最近邻所在的类
n1 = sum(count(species(index(1:k)),'setosa'));
n2 = sum(count(species(index(1:k)),'versicolor'));
n3 = sum(count(species(index(1:k)),'virginica')); %计算前k个近邻中各类的个数
if n1 > n2 && n1 > n3
    disp('k近邻归类:setosa');
elseif n2 > n1 && n2 > n3
    disp('k近邻归类:versicolor');
else
    disp('k近邻归类:virginica');
end
```

运行程序,在命令窗口输出对待测样本归类的结果:

```
最近邻归类:
ans{1} = versicolor
k近邻归类:
versicolor
```

3. 近邻法的快速算法

近邻法在样本数目较多时取得好的性能,但计算待测样本与大量训练样本之间的距离,计算量大,导致算法效率降低,因此需要快速算法。Kd树[①](K Dimensional Tree)是一种对 K 维空间中的点进行存储以便对其进行快速搜索的二叉树结构,利用Kd树可以省去对大部分数据点的搜索,从而减少搜索的计算量。

1) Kd树的构建

取数据的某一维,将数据从小到大排序,以数据点在该维度的中值作为切分超平面,将该维一分为二,小于该中值的数据点挂在其左子树,大于该中值的数据点挂在其右子树。再按同样的方式切分另一维,直到所有维度处理完毕。

【例5-3】 对二维平面点集合$\{[2\quad 3]^{\mathrm{T}},[5\quad 4]^{\mathrm{T}},[9\quad 6]^{\mathrm{T}},[4\quad 7]^{\mathrm{T}},[8\quad 1]^{\mathrm{T}},[7\quad 2]^{\mathrm{T}}\}$构建Kd树。

解:(1) 选择要切分的维度。分别计算 x_1 维和 x_2 维的方差,选择方差大的 x_1 维做切分。

(2) 切分 x_1 维。

按数据 x_1 维的数值排序:2、4、5、7、8、9。

确定中值:7(2、4、5、7、8、9的中值为(5+7)/2=6,由于中值需在点集合之内,取后一个7)。

确定节点:$[7\quad 2]^{\mathrm{T}}$。

分割空间:以 $x_1=7$ 将 x_1 维一分为二,$[2\quad 3]^{\mathrm{T}}$、$[4\quad 7]^{\mathrm{T}}$、$[5\quad 4]^{\mathrm{T}}$ 挂在$[7\quad 2]^{\mathrm{T}}$ 节点的左子树,$[8\quad 1]^{\mathrm{T}}$、$[9\quad 6]^{\mathrm{T}}$ 挂在$[7\quad 2]^{\mathrm{T}}$ 节点的右子树。

---

① 本书统一用 n 表示样本维数,而Kd树的 K 表示维数,此处与资料上的名称保持一致,注意与 kNN中 k 的区别。

(3) 切分 x_2 维——$[7\quad 2]^T$ 节点的左子树。

按数据 x_2 维的数值排序：3、4、7。

确定中值：4。

确定节点：$[5\quad 4]^T$。

分割空间：以 $x_2=4$ 将 $[7\quad 2]^T$ 节点的左边空间一分为二，$[2\quad 3]^T$ 挂在 $[5\quad 4]^T$ 节点的左子树，$[4\quad 7]^T$ 挂在 $[5\quad 4]^T$ 节点的右子树。

(4) 切分 x_2 维——$[7\quad 2]^T$ 节点的右子树。

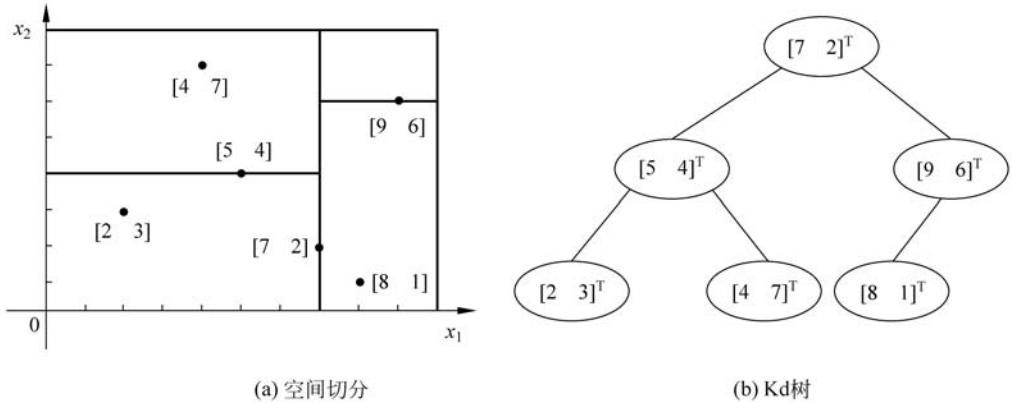
按数据 x_2 维的数值排序：1、6。

确定中值：6。

确定节点：$[9\quad 6]^T$。

分割空间：以 $x_2=6$ 将 $[7\quad 2]^T$ 节点的右边空间一分为二，$[8\quad 1]^T$ 挂在 $[9\quad 6]^T$ 节点的左子树。

Kd 树构建完成，对二维空间的切分如图 5-3(a)所示，构建的 Kd 树如图 5-3(b)所示。



(a) 空间切分　　(b) Kd树

图 5-3　Kd 树的构建

2) 最近邻搜索

给定点 $\boldsymbol{x}$，查询数据集中与其距离最近点的过程即为最近邻搜索，采用 Kd 树实现。过程如下：

(1) 在 Kd 树中找出包含目标点 $\boldsymbol{x}$ 的叶节点。

从根节点出发，向下访问 Kd 树，如果 $\boldsymbol{x}$ 当前维的坐标小于切分点的坐标，则移动到左子节点，否则移动到右子节点，直到叶节点为止，以此叶节点为“当前最近点”。顺序存储经过的节点。

(2) 按存储顺序向上回溯，在每个节点处进行以下操作：

① 如果当前节点比当前最近点距离搜索点更近，则更新当前最近点为当前节点。

② 以搜索点为圆心，以当前最近点到搜索点的距离为半径，确定圆，判断该圆是否与父节点的超平面相交；如果相交，则进入父节点的另外一侧搜索最近邻节点；如果不相交，则继续向上回溯。

当搜索到根节点时，搜索完成，得到最近邻节点。

【例 5-4】 利用例 5-3 建好的 Kd 树搜索 $[2\quad 4.5]^T$ 的最近邻。

解：(1) 在 Kd 树中找出包含目标点 $\boldsymbol{x}$ 的叶节点。从根节点$[7\quad 2]^{\mathrm{T}}$出发，当前维为x_1，$2<7$，移动到$[7\quad 2]^{\mathrm{T}}$的左子节点$[5\quad 4]^{\mathrm{T}}$；当前维更新为x_2，$4.5>4$，移动到$[5\quad 4]^{\mathrm{T}}$的右子节点$[4\quad 7]^{\mathrm{T}}$；$[4\quad 7]^{\mathrm{T}}$为叶节点，设为当前最近点，到待搜索点$[2\quad 4.5]^{\mathrm{T}}$的距离为$\sqrt{10.25}$。存储经过的节点，即搜索路径为：$[7\quad 2]^{\mathrm{T}}$、$[5\quad 4]^{\mathrm{T}}$、$[4\quad 7]^{\mathrm{T}}$。

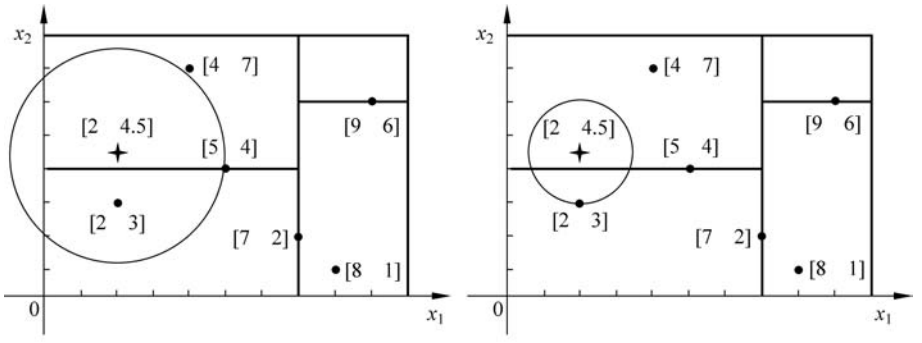
(2) 向上回溯到$[5\quad 4]^{\mathrm{T}}$，$[5\quad 4]^{\mathrm{T}}$到待搜索点$[2\quad 4.5]^{\mathrm{T}}$的距离为$\sqrt{9.25}$，$\sqrt{9.25}<\sqrt{10.25}$，更新当前最近点为$[5\quad 4]^{\mathrm{T}}$。

(3) 以$[2\quad 4.5]^{\mathrm{T}}$为圆心、以$\sqrt{9.25}$为半径的圆和父节点$[5\quad 4]^{\mathrm{T}}$的超平面$x_2=4$相交，如图 5-4(a)所示，进入$[5\quad 4]^{\mathrm{T}}$的另一侧查找，修改当前点为$[2\quad 3]^{\mathrm{T}}$；更新搜索路径为$[7\quad 2]^{\mathrm{T}}$、$[2\quad 3]^{\mathrm{T}}$。

(4) $[2\quad 3]^{\mathrm{T}}$到待搜索点$[2\quad 4.5]^{\mathrm{T}}$的距离为$\sqrt{2.25}$，$\sqrt{2.25}<\sqrt{9.25}$，更新当前最近点为$[2\quad 3]^{\mathrm{T}}$。

(5) 向上回溯到$[7\quad 2]^{\mathrm{T}}$，$[7\quad 2]^{\mathrm{T}}$到待搜索点$[2\quad 4.5]^{\mathrm{T}}$的距离大于$\sqrt{2.25}$，不修改当前最近点。

(6) 以$[2\quad 4.5]^{\mathrm{T}}$为圆心、以$\sqrt{2.25}$为半径的圆和父节点$[7\quad 2]^{\mathrm{T}}$的切分平面$x_1=7$不相交，如图 5-4(b)所示，搜索完成，最近邻为当前最近点$[2\quad 3]^{\mathrm{T}}$。



(a) 第一次回溯　　(b) 第二次回溯

图 5-4　利用 Kd 树搜索最近邻

4. 仿真实现

近邻法原理简单，主要问题在于如何提高搜索效率。MATLAB 中提供了相应的搜索函数，对其进行简要介绍。

1) ExhaustiveSearcher 模型

存储训练数据、距离度量和参数值用于穷举最近邻搜索，穷举搜索算法在 K 值大($K>10$)时比 Kd 树算法更有效，在距离度量选择方面比 Kd 树算法更灵活。

(1) NS=CREATENS(X,'NSMethod','exhaustive')：利用训练样本矩阵 X 创建 ExhaustiveSearcher 模型，X 的每一行为一个样本，列数为样本维数。

(2) NS=ExhaustiveSearcher(X)：利用训练样本矩阵 X 创建 ExhaustiveSearcher 模型。

(3) NS=ExhaustiveSearcher(X,Name,Value)：指定参数创建 ExhaustiveSearcher 模型，参数如表 5-1 所示。

表 5-1　ExhaustiveSearcher 函数参数表

参数名称	取值及含义
Distance	距离度量方式，可取'euclidean'（默认）、'chebychev'、'cityblock'、'correlation'、'cosine'、'hamming'、'jaccard'、'minkowski'、'mahalanobis'、'seuclidean'、'spearman'或定制的距离函数
Cov	采用马氏距离（mahalanobis）度量时指定 n×n 的正定协方差矩阵，默认为 nancov(X)
P	采用闵氏距离（minkowski）度量时指定指数，默认为 2
Scale	采用标准欧氏距离（seuclidean）度量指定尺度参数，默认为 nanstd(X)

2）KDTreeSearcher 模型

存储使用 Kd 树算法的最近邻搜索结果，包括训练数据、距离度量及其参数、每个叶节点中的最大数据点数量。

（1）NS＝CREATENS(X,'NSMethod','kdtree')：利用训练样本矩阵 X 创建 KDTreeSearcher 模型。

（2）NS＝KDTreeSearcher(X)：利用训练样本矩阵 X 创建 KDTreeSearcher 模型。

（3）NS＝KDTreeSearcher(X,Name,Value)：指定参数创建 KDTreeSearcher 模型，参数如表 5-2 所示。

表 5-2　KDTreeSearcher 函数参数

参数名称	取值及含义
Distance	距离度量方式，可取：'euclidean'（默认）、'chebychev'、'cityblock'、'minkowski'
P	采用闵氏距离（minkowski）度量时指定指数，默认为 2
BucketSize	每个叶节点中的最大数据点数量，默认为 50

3）knnsearch 函数

指定模型实现近邻搜索。

（1）IDX＝knnsearch(NS,Y)：从模型 NS 存储的训练样本 X 中找到 Y 中每个样本的最近邻 IDX。Y 的每行为一个样本；IDX 为一个列向量，存储 Y 中样本在 X 中的最近邻的索引。

（2）[IDX,D]＝knnsearch(NS,Y)：同时返回列向量 D，存储 Y 中样本和其最近邻的距离。

（3）[IDX,D]＝knnsearch(NS,Y,'NAME1',VALUE1,...,'NAMEN',VALUEN)：指定参数实现最近邻搜索，参数如表 5-3 所示。

表 5-3　knnsearch 函数参数

参数名称	取值及含义
Distance	采用 ExhaustiveSearcher 模型，取值见表 5-1；采用 KDTreeSearcher 模型，取值见表 5-2；部分距离可以设置参数，见表 5-1 和表 5-2
K	要搜索的近邻数目，默认为 1；K 不为 1 时，IDX 和 D 均为矩阵
IncludeTies	逻辑值，指定是否包括和第 K 个近邻距离并列的邻点，默认为 false
SortIndices	逻辑值，指定输出的距离和索引是否按照距离从小到大排列，默认为 true

4）rangesearch 函数

指定模型实现一定范围内的近邻搜索。

（1）IDX＝rangesearch(NS,Y,RADIUS)：从模型 NS 存储的训练样本 X 中找到和 Y 中样本距离不超过 RADIUS 的样本。IDX 为元胞数组，每一项存储 Y 中每一个样本的近邻索引，按距离升序排列。

（2）[IDX,D]＝rangesearch(NS,Y,RADIUS)：元胞数组 D 存储 Y 中样本和各近邻的距离。

（3）[IDX,D]＝rangesearch(NS,Y,RADIUS,'NAME1',VALUE1,...,'NAMEN',VALUEN)：指定参数搜索，参数有：'Distance'、'SortIndices'以及距离度量参数，同 knnsearch 函数。

【例 5-5】 导入 fisheriris 数据集，随机选择 147 个训练样本，3 个测试样本，采用 knnsearch、rangesearch 函数进行近邻搜索。

程序如下：

```
clc,clear,close all;
load fisheriris                          %导入数据
rng(1);                                  %设定随机数生成模式
N = size(meas,1);                        %样本数
testIdx = randsample(N,3);               %生成 1:N 的 3 个随机整数
trainIdx = ~ismember(1:N,testIdx);       %生成索引向量,随机数对应数值为 0,其他数对应数值为 1
testing = meas(testIdx,3:4);             %测试样本,仅采用原始数据的第 3 维和第 4 维
training = meas(trainIdx,3:4);           %训练样本
labeltrain = species(trainIdx,:);        %训练样本的类别标签
labeltest = species(testIdx,:);          %测试样本的类别标签
Mdl = KDTreeSearcher(training);          %生成 KDTreeSearcher 模型
[Idx1,D1] = knnsearch(Mdl,testing);      %搜索最近邻
testresult = labeltrain(Idx1)            %最近邻对应的类别即为分类结果
radius = 0.2;                            %设定搜索范围的半径
[Idx2,D2] = rangesearch(Mdl,testing,radius); %搜索范围
gscatter(training(:,1),training(:,2),labeltrain,'rgb','+.o',3); %绘制训练样本
hold on
plot(testing(:,1),testing(:,2),'kx','MarkerSize',6,'LineWidth',2);%绘制测试样本
plot(training(Idx1,1),training(Idx1,2),'m*'); %绘制测试样本的最近邻
for i = 1:length(Idx2)
    viscircles(testing(i,:),radius,'color','k','LineWidth',1,'LineStyle','-.');
end                                      %绘制以测试样本为中心的圆,搜索范围
legend('setosa','versicolor','virginica','待测样本','最近邻点');
xlabel('花瓣长/cm'),ylabel('花瓣宽/cm');
hold off
```

运行程序，绘制的样本点、近邻点如图 5-5 所示，并在命令窗口输出 3 个待测样本的归类结果。

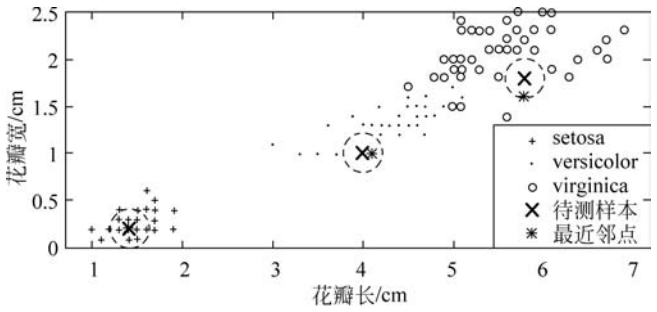


图 5-5　fisheriris 数据集的近邻搜索

5.2　二次判别函数

二次判别(Quadratic Discriminant)也是一种比较常用的固定函数类型的分类方法，一般形式为

$$\begin{aligned} g(\boldsymbol{x}) &= \boldsymbol{x}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{x} + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0 \\ &= \sum_{i=1}^{n} w_{ii}x_i^2 + 2\sum_{i=1}^{n-1}\sum_{k=i+1}^{n} w_{ik}x_i x_k + \sum_{i=1}^{n} w_i x_i + w_0 \end{aligned} \tag{5-6}$$

其中，$\boldsymbol{W}$ 是 $n\times n$ 的实对称矩阵，$\boldsymbol{w}$ 是 n 维列向量。

由式(5-6)可以看出，二次判别函数中有太多的参数需要确定，如果采用类似于线性判别函数设计的方法，则计算复杂，在样本数不足的情况下，不能保证结果的可靠性和推广能力。因此，需要采用其他的设计方法。

在 2.7 节了解到，在一般正态分布情况下，贝叶斯决策判别函数为二次函数，如果可以用正态分布模拟样本分布，则直接定义 ω_j 类的二次判别函数为

$$g_j(\boldsymbol{x}) = C_j^2 - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^{\mathrm{T}}\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j) \tag{5-7}$$

C_j^2 是一个调节项，受协方差矩阵和先验概率的影响，可以通过调节该参数调整类错误率。

例如，在例 2-15 中，针对两类正态分布的样本，利用 fitcdiscr 函数设计了二次判别函数。

如果只有一类数据可以用正态分布模拟，另一类比较均匀地分布在第一类附近，则可以只对第一类求解二次判别函数，决策规则为

$$\text{若 } g(\boldsymbol{x}) \gtrless 0, \quad \text{则决策 } \boldsymbol{x} \in \begin{cases} \omega_1 \\ \omega_2 \end{cases} \tag{5-8}$$

【例 5-6】 导入 fisheriris 数据集，选择其中的两类，采用正态分布模拟样本分布，设计二次判别函数，绘制二次决策面。

程序如下：

```
clc,clear,close all;
load fisheriris
index1 = strcmp(species(:),'versicolor');
```

```
index2 = strcmp(species(:),'virginica');          % 确定要选择的两类数据在数据集中的索引
training1 = meas(index1,3:4);
training2 = meas(index2,3:4);                     % 获取两类数据
training = [training1;training2];
index = [species(index1);species(index2)];        % 获取训练样本类别标记
gscatter(training(:,1),training(:,2),index,'rb','*o',5); % 绘制样本
hold on
mu1 = mean(training1); mu2 = mean(training2);     % 估计两类样本均值
sigma1 = cov(training1); sigma2 = cov(training2); % 估计两类协方差矩阵
minx = min(training(:,:)); maxx = max(training(:,:)); % 确定样本的取值范围
step = 0.1;
[x1Grid,x2Grid] = meshgrid(minx(:,1):step:maxx(:,1),minx(:,2):step:maxx(:,2));
xGrid = [x1Grid(:),x2Grid(:)];                    % 确定待测样本
for i = 1:length(xGrid)
    gx(i) = -(xGrid(i,:) - mu1)/(sigma1) * (xGrid(i,:) - mu1)'...
            (xGrid(i,:) - mu2)/(sigma2) * (xGrid(i,:) - mu2)'; % 计算各样本对应判别函数取值
end
contour(x1Grid,x2Grid,reshape(gx,size(x1Grid)),[0 0],'k--'); % 绘制决策面
legend('versicolor','virginica','决策面');
xlabel('花瓣长/cm'),ylabel('花瓣宽/cm');
hold off
```

程序运行结果如图 5-6 所示。程序中两类判别函数中调节项设为 0。

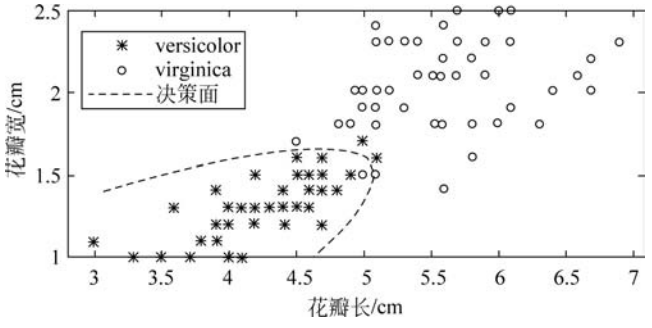


图 5-6　二次判别函数设计

5.3　决策树

人们在决策时，往往会针对多个方面依次判断，模式识别中也可以采用类似的多级决策方式，利用一定的训练样本，依次分类，直到获得最终可以接受的类，用树形表示决策的过程，并形成决策规则。这种从数据中“学习”出决策规则的方法称为决策树(Decision Tree)，是一种非线性分类器。

5.3.1　基本概念

一般而言，一棵决策树包含一个根节点、若干内部节点和若干叶节点。根节点包含所要

进行分类的样本集，按照某种规则（常用某一种特征测试），将样本集一分为几个子集，对应几个子节点，称为内部节点；子节点再分，直到叶节点，对应决策结果。

叶节点有三种情形：

（1）当前节点所含样本全属于同一类别，不需要再划分，该节点为叶节点。

（2）当前特征为空，或所有样本当前特征取值相同，无法划分，该节点为叶节点，标记类别为该节点所含样本最多的类别。

（3）当前节点所含样本集为空，无法划分，该节点为叶节点，标记类别为其父节点所含样本最多的类别。

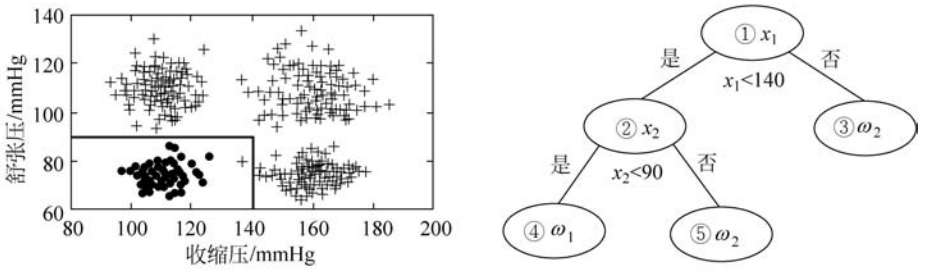
以判断是否是高血压为例，说明决策树的生成以及决策。用 x_1 表示收缩压，用 x_2 表示舒张压，血压为二维样本 $\boldsymbol{x}=[x_1 \quad x_2]^{\mathrm{T}}$。图 5-7(a)所示为正常血压 ω_1 和高血压 ω_2 二维样本集$\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N\}$，生成的决策树如图 5-7(b)所示。根节点①包括所有样本，考查第一个特征 x_1，根据是否满足 $x_1<140$ 将样本集一分为二，即节点①有两个子节点②和③；节点③中所有样本全部属于高血压 ω_2 类，为叶节点，标记类别为 ω_2 类；节点②非叶节点，考查第二个特征 x_2，根据是否满足 $x_2<90$ 将样本集一分为二，即节点②有两个子节点④和⑤；节点④中所有样本全部属于正常血压 ω_1 类，为叶节点，标记类别为 ω_1 类；节点⑤中所有样本全部属于高血压 ω_2 类，为叶节点，标记类别为 ω_2 类。从根节点①到叶节点③④⑤存在 3 条通路，通路上的判断条件即为决策规则，即

若 $x_1\geqslant 140$，则 $\boldsymbol{x}\in\omega_2$。

若 $x_1<140$ 且 $x_2<90$，则 $\boldsymbol{x}\in\omega_1$。

若 $x_1<140$ 且 $x_2\geqslant 90$，则 $\boldsymbol{x}\in\omega_2$。

图 5-7 所示的决策树生成中，节点采用的分枝准则为“特征 $x_i<\alpha$ 或者 $x_i\geqslant\alpha$”，这种决策树称为普通二进制分类树（Ordinary Binary Classification Tree，OBCT），也可以根据不同的条件生成不同的决策树。



(a) 血压数据　　(b) 生成的决策树

图 5-7　决策树示例

5.3.2　决策树的构建

从图 5-7 所示的决策树生成可以看出，决策树的构建就是选取特征和确定分枝准则的过程，每一个分枝应该产生比父节点样本集更有利于分类的样本子集，也就是新子集中的样本都更好地属于特定的类。因此，决策树的构建首先要有能衡量分类有利性的指标量，以便合理选择分枝采用的某个特征，以及确定分枝准则。

1. 信息增益

熵(Entropy)用来度量对某个事件进行观察后得到的信息量,设该事件有 c 种可能,每种可能对应的概率为 P_j,$j=1,2,\cdots,c$,熵为

$$H=-(P_1\log_2 P_1+P_2\log_2 P_2+\cdots+P_c\log_2 P_c)=-\sum_{j=1}^{c}P_j\log_2 P_j \tag{5-9}$$

例如,一个 4 类分类问题,各类样本数相同,根节点包括所有样本,样本属于某一类的不确定性最大,即纯度最低,对应的熵为

$$H=-4\times 0.25\times\log_2 0.25=2$$

如果某个内部节点只包含其中两类的样本且数量相等,则样本属于某一类的不确定性相对于根节点要低,即纯度有所提高,对应的熵为

$$H=-2\times 0.5\times\log_2 0.5=1$$

熵的值较根节点有所下降。而对于叶节点,只包含同一类样本,纯度最高,此时的熵为

$$H=-1\times\log_2 1=0$$

熵最小,不确定性最小,纯度最高。

因此,对于某个节点上的样本,熵也称为熵不纯度,反映了该节点上的特征对样本分类的不纯度,取值越小,不纯度越低。

在决策树分枝的时候,希望分枝后的子集相对于原来的子集纯度有所提高,或说不纯度下降,因此,定义信息增益(Information Gain)为不纯度减少量。

如果某个节点上,把 N 个样本划分成 m 组,每组 N_m 个样本,则信息增益为

$$\Delta H(N)=H(N)-[P_1H(N_1)+P_2H(N_2)+\cdots+P_mH(N_m)] \tag{5-10}$$

其中,$P_m=N_m/N$。

2. ID3 方法

交互式二分法(Interactive Dichotomizer-3,ID3)是最早比较著名的决策树构建方法,对于每个非叶节点,计算比较采用不同特征进行分枝的信息增益,选择带来最大信息增益的特征作为分枝特征,以实现决策树的生成与构建,方法步骤如下:

(1) 计算当前节点包含的所有样本的熵不纯度。

(2) 比较采用不同特征进行分枝将会得到的信息增益,选取具有最大信息增益的特征赋予当前节点。特征的取值个数决定了该节点下的分枝数目。

(3) 如果后继节点只包含一类样本,则停止该枝的生长,该节点成为叶节点。

(4) 如果后继节点仍包含不同类样本,则再次进行以上步骤,直到每一枝都到达叶节点。

【例 5-7】 以表 5-4 中的 1～14 号样本为训练样本,15～19 号样本为测试样本,采用 ID3 算法设计决策树,并对测试样本进行决策。

表 5-4 血压数据表

序号	年龄/岁	体重	饮食偏好	父辈血压	高血压
1	34	超重	油腻	高	是
2	28	正常	均衡	高	是
3	42	偏重	清淡	高	是

续表

序号	年龄/岁	体重	饮食偏好	父辈血压	高血压
4	45	超重	均衡	正常	是
5	50	偏重	油腻	正常	是
6	61	偏重	油腻	正常	是
7	65	正常	清淡	高	是
8	36	超重	均衡	正常	否
9	31	正常	清淡	高	否
10	27	偏重	油腻	高	否
11	44	偏重	清淡	正常	否
12	43	偏重	均衡	正常	否
13	61	正常	清淡	正常	否
14	66	正常	清淡	正常	否
15	25	正常	均衡	正常	否
16	35	偏重	均衡	正常	是
17	48	超重	油腻	高	是
18	40	正常	清淡	正常	否
19	63	偏重	均衡	高	是
20	62	正常	均衡	正常	否

解：(1) 连续特征数据调整。表 5-4 中的年龄取值太多，如果直接按年龄分枝，将导致太多的子节点。因此把年龄分为三个级别，40 岁以下为青年，40～59 岁为中年，60 岁及以上为老年，如表 5-5 所示。

表 5-5　血压数据表

序号	年龄	体重	饮食偏好	父辈血压	高血压
1	青年	超重	油腻	高	是
2	青年	正常	均衡	高	是
3	中年	偏重	清淡	高	是
4	中年	超重	均衡	正常	是
5	中年	偏重	油腻	正常	是
6	老年	偏重	油腻	正常	是
7	老年	正常	清淡	高	是
8	青年	超重	均衡	正常	否
9	青年	正常	清淡	高	否
10	青年	偏重	油腻	高	否
11	中年	偏重	清淡	正常	否
12	中年	偏重	均衡	正常	否
13	老年	正常	清淡	正常	否
14	老年	正常	清淡	正常	否
15	青年	正常	均衡	正常	否
16	青年	偏重	均衡	正常	是
17	中年	超重	油腻	高	是
18	中年	正常	清淡	正常	否
19	老年	偏重	均衡	高	是
20	老年	正常	均衡	正常	否

(2) 生成决策树。

首先,计算不考虑任何特征时熵不纯度。14 个人中高血压 7 人,正常血压 7 人,根节点熵不纯度为

$$H(14,7)=-[0.5\times\log_2(0.5)+0.5\times\log_2(0.5)]=1$$

其次,根节点分枝。考查采用不同特征划分样本后的信息增益。

采用“年龄”特征划分样本集:青年组 5 人,高血压 2 人;中年组 5 人,高血压 3 人;老年组 4 人,高血压 2 人,熵不纯度为

$$H_{\text{age}}=\frac{5}{14}H(5,2)+\frac{5}{14}H(5,3)+\frac{4}{14}H(4,2)=0.9793$$

信息增益为

$$\Delta H_{\text{age}}(14)=H(14,7)-H_{\text{age}}=0.0207$$

采用“体重”特征划分样本集:正常组 5 人,高血压 2 人;偏重组 6 人,高血压 3 人;超重组 3 人,高血压 2 人,熵不纯度为

$$H_{\text{weight}}=\frac{5}{14}H(5,2)+\frac{6}{14}H(6,3)+\frac{3}{14}H(3,2)=0.9721$$

信息增益为

$$\Delta H_{\text{weight}}(14)=H(14,7)-H_{\text{weight}}=0.0279$$

采用“饮食偏好”特征划分样本集:油腻组 4 人,高血压 3 人;清淡组 6 人,高血压 2 人;均衡组 4 人,高血压 2 人,熵不纯度为

$$H_{\text{diet}}=\frac{4}{14}H(4,3)+\frac{6}{14}H(6,2)+\frac{2}{14}H(4,2)=0.9111$$

信息增益为

$$\Delta H_{\text{diet}}(14)=H(14,7)-H_{\text{diet}}=0.0889$$

采用“父辈血压”特征划分样本集:高血压组 6 人,高血压 4 人;正常血压组 8 人,高血压 3 人,熵不纯度为

$$H_{\text{parent}}=\frac{6}{14}H(6,4)+\frac{8}{14}H(8,3)=0.9389$$

信息增益为

$$\Delta H_{\text{parent}}(14)=H(14,7)-H_{\text{parent}}=0.0611$$

采用“饮食偏好”特征划分样本集信息增益最大,用该特征将根节点一分为三,如图 5-8(a)所示。

再次,下一级节点分枝。

节点②,4 个样本,为序号 1、5、6、10,高血压 3 人,熵不纯度为

$$H(4,3)=-\left(\frac{3}{4}\times\log_2\frac{3}{4}+\frac{1}{4}\times\log_2\frac{1}{4}\right)=0.8113$$

依次采用“年龄”“体重”“父辈血压”特征划分样本集,各自的熵不纯度和信息增益为

$$H_{\text{age}}=\frac{2}{4}H(2,1)+\frac{1}{4}H(1,1)+\frac{1}{4}H(1,1)=0.5000,$$

$$\Delta H_{\text{age}}(4)=H(4,3)-H_{\text{age}}=0.311$$

$$H_{\text{weight}}=\frac{3}{4}H(3,2)+\frac{1}{4}H(1,1)=0.6887,$$

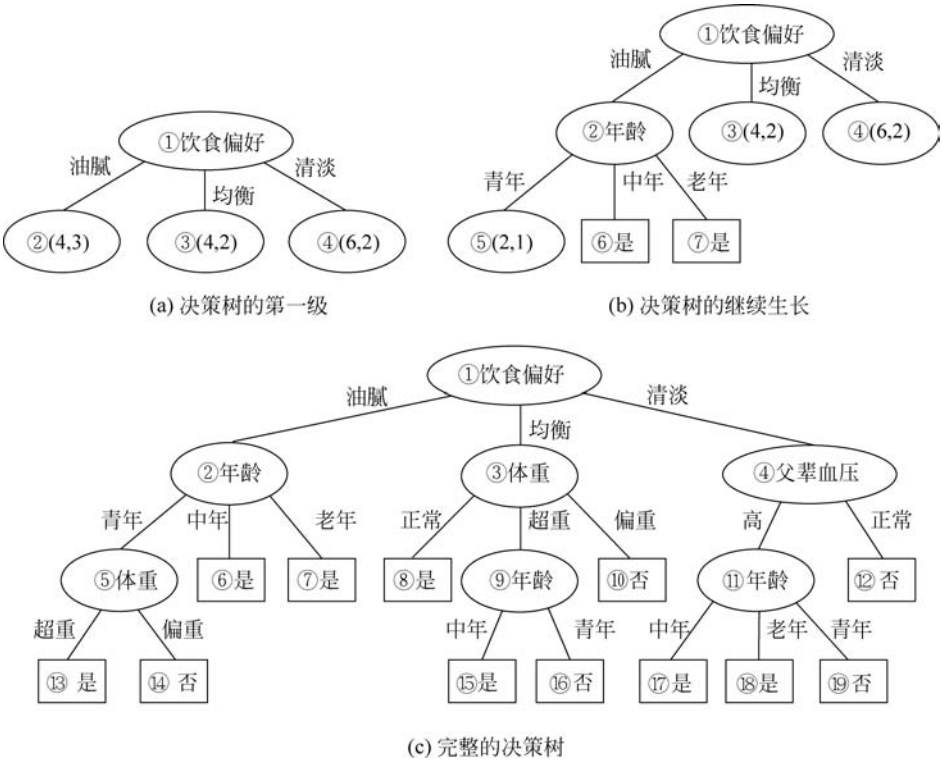
$$\Delta H_{\text{weight}}(4)=H(4,3)-H_{\text{weight}}=0.1226$$

$$H_{\text{parent}}=\frac{2}{4}H(2,1)+\frac{2}{4}H(2,2)=0.5000,$$

$$\Delta H_{\text{parent}}(4)=H(4,3)-H_{\text{parent}}=0.311$$

采用“年龄”和“父辈血压”划分样本集信息增益一样大，选择“年龄”特征将节点②一分为三，如图 5-8(b)所示，其中，节点⑥和⑦中各有一个高血压类样本，为叶节点，标记为高血压类。

然后，同样的方法依次处理直到叶节点，生成的决策树共有 4 级，如图 5-8(c)所示，方框表示叶节点。



(a) 决策树的第一级　(b) 决策树的继续生长

(c) 完整的决策树

图 5-8　用 ID3 方法对表 5-4 中数据判读是否高血压的决策树

(3) 利用决策树对测试样本进行决策。

首先，确定决策规则。生成的决策树有 12 个叶节点，从根节点到叶节点 12 条通路，对应 12 条决策规则，例如：

若(饮食偏好==油腻)且(年龄==青年)且(体重==超重)，则(血压=高血压)；

若(饮食偏好==油腻)且(年龄==青年)且(体重==偏重)，则(血压=正常血压)；

……

若(饮食偏好==清淡)且(父辈血压==正常)，则(血压=正常血压)。

其次，将待测样本代入决策规则进行判断。

15 号样本，饮食偏好==均衡，体重==正常，节点⑧，判断为高血压，判断错误。

16 号样本，饮食偏好＝＝均衡，体重＝＝偏重，节点⑩，判断为正常血压，判断错误。

17 号样本，饮食偏好＝＝油腻，年龄＝＝中年，节点⑥，判断为高血压，判断正确。

18 号样本，饮食偏好＝＝清淡，父辈血压＝＝正常，节点⑫，判断为正常血压，判断正确。

19 号样本，饮食偏好＝＝均衡，体重＝＝偏重，节点⑩，判断为正常血压，判断错误。

20 号样本，饮食偏好＝＝均衡，体重＝＝正常，节点⑧，判断为高血压，判断错误。

例题中样本过少，叶节点往往只有一个样本，可能会抓住由于偶然性带来的假象，分类器判断准确率较差。可以采用剪枝的方式提高泛化能力，将在下节学习。

ID3 方法虽然名为交互式二分法，但实际根据特征的取值可以划分为多个子节点。

【例 5-8】 采用例 5-7 中的数据，基于 ID3 方法设计决策树生成程序。

程序如下：

```
clc,clear,close all;
    X = ["序号" "年龄" "体重" "饮食偏好" "父辈血压";"1" "青年" "超重" "油腻" "高";...
    "2" "青年" "正常" "均衡" "高";"3" "中年" "偏重" "清淡" "高";...
    "4" "中年" "超重" "均衡" "正常";"5" "中年" "偏重" "油腻" "正常";...
    "6" "老年" "偏重" "油腻" "正常";"7" "老年" "正常" "清淡" "高";...
    "8" "青年" "超重" "均衡" "正常";"9" "青年" "正常" "清淡" "高";...
    "10" "青年" "偏重" "油腻" "高";"11" "中年" "偏重" "清淡" "正常";...
    "12" "中年" "偏重" "均衡" "正常";"13" "老年" "正常" "清淡" "正常";...
    "14" "老年" "正常" "清淡" "正常"];        %训练样本,含序号和特征变量名
Y = [1;1;1;1;1;1;1;2;2;2;2;2;2;2];            %样本类别标签,1:高血压,2:正常血压
code_total = 1;                               %当前节点总数
node{1,1} = 1:size(X,1) - 1;                  %存放节点包含的样本序号
node{1,2} = 0;                                %存放节点级别,当前根节点,为 0 级
node{1,3} = code_total;                       %存放节点编号
node{1,4} = [];                               %内部节点存放节点分枝特征,叶节点存放类别标签
node_cur = 1;                                 %当前节点编号
training{1,1} = X; training{1,2} = node_cur;  %存放可分枝节点包含的样本及对应节点编号
label{1} = Y;                                 %存放可分枝节点包含的样本对应的类别标签
stacknum = 0;                                 %处理到的可分枝节点序号,初始化为 0
while stacknum~ = size(training,1)            %尚有未处理完的可分枝节点时进行循环处理
    stacknum = stacknum + 1;                  %当前处理到的可分枝节点序号
    level = node{stacknum,2} + 1;             %当前节点所在级别
    label_cur = label{stacknum};              %当前节点样本对应的类别标签
    FatherI = calEnt(label_cur);              %当前节点的信息熵
    code_cur = training{stacknum,1};          %当前节点的训练样本
    node_cur = training{stacknum,2};          %当前节点的编号
    [N,n] = size(code_cur(2:end,2:end)); %获取当前样本集的样本数目及可用于分枝的特征变量数
    deltaI = zeros(n,1);                      %各特征对应信息增益初始值
    for i = 1:n                               %对每一个可用于分枝的特征进行计算
        data = code_cur(2:end,i + 1);         %获取当前样本集的同一特征
        sta = tabulate(data);                 %统计该特征的取值情况
        sonI = 0;                             %子节点信息熵初始值
```

```
        for j = 1:size(sta,1)                    % 对于该特征的每种取值进行处理
            templabel = label_cur(strcmp(data,sta(j,1))); % 获取各个取值对应的类别标签
            sonI = sonI + length(templabel)/N * calEnt(templabel); % 计算按该特征分枝对应的熵
        end
        deltaI(i) = FatherI - sonI;              % 计算按该特征分枝对应的信息增益
    end
    [~,pos] = max(deltaI);                       % 找最大信息增益
    node{node_cur,4} = code_cur(1,pos + 1);      % 存放当前节点的分枝特征变量名
    data = code_cur(2:end,pos + 1);              % 获取分枝特征对应的数据
    code_cur(:,pos + 1) = []; % 从当前数据集中去掉该特征数据,在后续节点不再用该特征分枝
    sta = tabulate(data);                        % 统计分枝特征取值情况
    for j = 1:size(sta,1) % 对分枝特征每种取值进行处理,有几种取值就对应几个子节点
        code_total = code_total + 1;             % 节点数增加
        number = strcmp(data,sta(j,1));          % 获取当前取值对应的样本序号
        number = [false;number];                 % 增加一行,对应特征变量名所在行
        node{end + 1,1} = code_cur(number,1);
        node{end,2} = level;
        node{end,3} = code_total;                % 增加节点,存放样本序号、级别以及编号
        templabel = label_cur(number(2:end)); % 新节点样本对应的类别标签
        if length(unique(templabel)) == 1        % 标签只有一种时,为叶节点,存放类别标签
            node{end,4} = templabel(1);
        else                                     % 标签有多种时,存储该节点的信息
            number(1) = true;
            training{end + 1,1} = code_cur(number,:); % 存储数据,含特征变量名
            training{end,2} = code_total;        % 存储数据对应的节点编号
            label{end + 1} = templabel;          % 存储数据对应的类别标签
        end
    end
end
function Entroph = calEnt(source)                % 计算当前序列的信息熵
    Entroph = 0;
    sta = tabulate(source);
    [c,k] = size(sta);
    for j = 1:c
        if sta(j,k)~ = 0
            Entroph = Entroph - sta(j,k)/100 * log2(sta(j,k)/100);
        end
    end
end
```

运行程序,可以在工作区查看 node 变量,node 为 19×4 的元胞数组,每一行表示一个节点的信息,分别为节点存放的样本序号、节点的级别、节点的编号、分枝特征(叶节点存放类别标签),和图 5-8(c)所示一致。

3. C4.5 算法

C4.5 算法采用信息增益率(Gain Ratio)代替信息增益来选择最优划分特征,增益率的

定义为

$$\Delta H_{\mathrm{R}}(N)=\frac{\Delta H(N)}{H(N)} \tag{5-11}$$

C4.5 算法增加了处理连续特征的功能，采用了二分法将连续特征离散化为二值特征。设特征 x_i，$i=1,2,\cdots,n$，在训练样本上共包含了 m 个值，将这些值按照从小到大的顺序排列，记为$\{x_i^1,x_i^2,\cdots,x_i^m\}$；设定一个阈值 T，将数据分为两个子集 x_i^-、x_i^+；不同的阈值将导致不同的分割子集，对每种情况计算信息增益率，选择信息增益率最大的划分方案实现特征二值化，再进行决策树的生成。

阈值 T 可以取相邻两个取值的中值

$$T=\frac{x_i^j+x_i^{j+1}}{2},\quad j=1,2,\cdots,m-1 \tag{5-12}$$

【例 5-9】 以表 5-4 中的 1～14 号样本为训练样本，采用 C4.5 算法生成决策树。

解：(1) 根节点熵不纯度为 $H(14,7)=1$。

(2) 根节点分枝。考查采用不同特征划分样本后的信息增益率。

采用“体重”“饮食偏好”“父辈血压”特征划分样本集，信息增益率分别为 0.0279、0.0889、0.0611。

对于“年龄”特征，根节点包含的样本在该特征上的取值排序为{27,28,31,34,36,42,43,44,45,50,61,61,65,66}；阈值 T 取值为{27.5,29.5,32.5,35,39,42.5,43.5,44.5,47.5,55.5,63,65.5}，当 $T=27.5$ 时，“年龄$<T$”组 1 人，高血压 0 人；“年龄$>T$”组 13 人，高血压 7 人，熵不纯度为

$$H_{\mathrm{age}}^1=\frac{1}{14}H(1,0)+\frac{13}{14}H(13,7)=0.9246$$

信息增益率为

$$\Delta H_{\mathrm{R,age}}(14)=\frac{H(14,7)-H_{\mathrm{age}}^1}{H(14,7)}=0.0754$$

依次修改阈值 T，计算“年龄”特征最大信息增益率为 0.0754，对应阈值 $T=27.5$。

采用“饮食偏好”特征划分样本集信息增益率最大，因此，用该特征将根节点一分为三，如图 5-9(a)所示。

(3) 下一级节点分枝。

节点②，4 个样本，高血压 3 人，熵不纯度为 $H(4,3)=0.8113$。采用“体重”“父辈血压”特征划分样本集，信息增益率分别为 0.1511 和 0.3837。

对于“年龄”特征，节点②包含的样本在该特征上的取值排序为{27,34,50,61}，阈值 T 取值为{30.5,42,55.5}；当 $T=30.5$ 时，“年龄$<T$”组 1 人，高血压 0 人；“年龄$>T$”组 3 人，高血压 3 人，熵不纯度为 0，信息增益率为 1；当 $T=42$，$T=55.5$ 时，信息增益率分别为 0.3837 和 0.1511。采用“年龄 $T=30.5$”划分样本集信息增益率最大，选择“年龄”特征将节点②一分为二，子节点⑤中只有一个正常血压样本，为叶节点，标记为正常血压类；子节点⑥中有 3 个高血压类样本，为叶节点，标记为高血压类，如图 5-9(b)所示。

(4) 同样的方法依次处理直到叶节点，生成的决策树如图 5-9(c)所示。

C4.5 算法不需要事先将连续特征离散化为少量级别，但在生成决策树的过程中，取不

同阈值将连续特征二值化，划分方案增多。另外，C4.5 算法还可以处理部分特征缺失的样本。

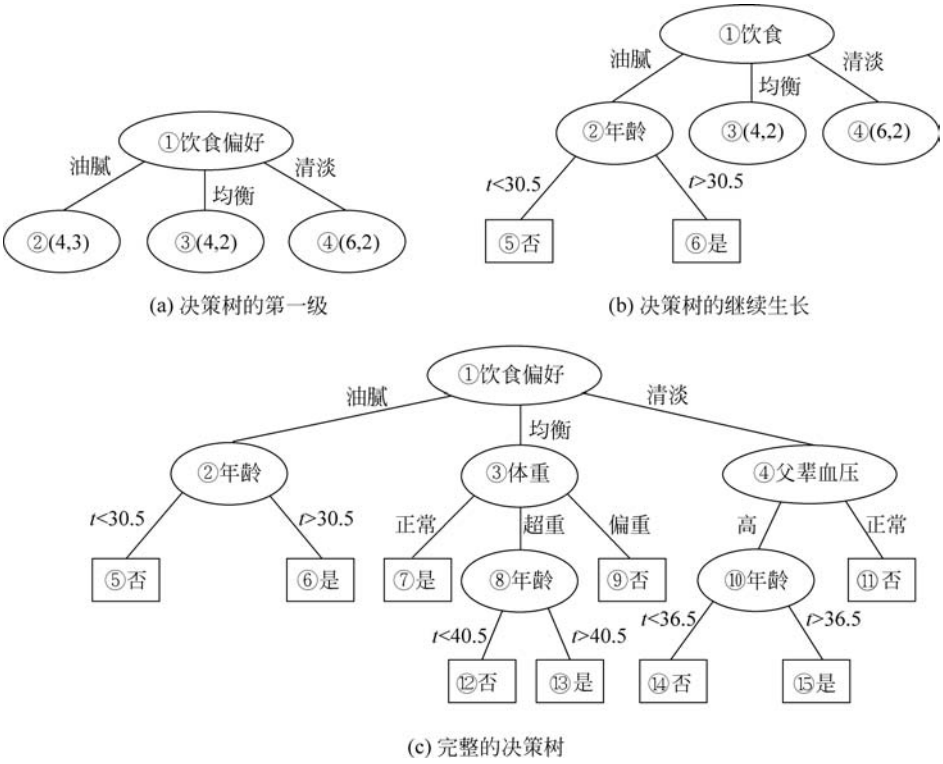


图 5-9　用 C4.5 算法对表 5-4 中数据判读是否高血压的决策树

4. CART 算法

分类和回归树(Classification And Regression Tree，CART)算法也是一个很著名的决策树算法，核心思想和 ID3 方法相同，主要不同之处在于，CART 在每一个节点上都采用二分法，最终构成一棵二叉树；CART 既可用于分类，也可以用于构造回归树对连续变量进行回归。下面主要介绍 CART 构建用于分类的决策树。

CART 分类树算法使用基尼系数(Gini Index)来选择节点分枝特征。设样本集 $\mathscr{X}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N\}$ 有 c 个类别，其中第 j 类的概率为 P_j，基尼系数定义为

$$\mathrm{Gini}(\mathscr{X})=\sum_{j=1}^{c}P_j(1-P_j)=1-\sum_{j=1}^{c}P_j^2 \tag{5-13}$$

从定义可以看出，基尼系数反映了从样本中随机抽取两个样本，其类别标记不一样的概率。因此，基尼系数越小，两个样本来自不同类的概率越小，样本集不纯度越低。

如果各类的样本数为 N_j，则样本集 $\mathscr{X}$ 的基尼系数为

$$\mathrm{Gini}(\mathscr{X})=1-\sum_{j=1}^{c}\left(\frac{N_j}{N}\right)^2 \tag{5-14}$$

决策树生成中根据某个特征 $x_{i,i=1,2,\cdots,n}$ 将样本集划分为两部分 $\mathscr{X}_1$ 和 $\mathscr{X}_2$，各自的样本数为 N_1 和 N_2，按照 x_i 划分的样本集的基尼系数为

$$\mathrm{Gini}(\mathscr{X},x_i)=\frac{N_1}{N}\mathrm{Gini}(\mathscr{X}_1)+\frac{N_2}{N}\mathrm{Gini}(\mathscr{X}_2) \tag{5-15}$$

使用基尼系数来代替信息熵存在一定的误差，是熵模型的一个近似替代。但是，由于其避免了大量的对数运算，因此减少了计算量，简化了模型的运用。

【例 5-10】 以表 5-4 中的 1～14 号样本为训练样本，采用 CART 算法生成分类决策树。

解：(1) 根节点分枝。考查采用不同特征划分样本后的基尼系数。

采用“体重”特征划分样本集：正常组 5 人，高血压 2 人，非正常组 9 人，高血压 5 人；偏重组 6 人，高血压 3 人，非偏重组 8 人，高血压 4 人；超重组 3 人，高血压 2 人，非超重组 11 人，高血压 5 人。三种划分情况的基尼系数分布为

$$\mathrm{Gini}^1_{\mathrm{weight}}=\frac{5}{14}\left[1-\left(\frac{2}{5}\right)^2-\left(\frac{3}{5}\right)^2\right]+\frac{9}{14}\left[1-\left(\frac{5}{9}\right)^2-\left(\frac{4}{9}\right)^2\right]=0.4889$$

$$\mathrm{Gini}^2_{\mathrm{weight}}=\frac{6}{14}\left[1-\left(\frac{3}{6}\right)^2-\left(\frac{3}{6}\right)^2\right]+\frac{8}{14}\left[1-\left(\frac{4}{8}\right)^2-\left(\frac{4}{8}\right)^2\right]=0.5$$

$$\mathrm{Gini}^3_{\mathrm{weight}}=\frac{3}{14}\left[1-\left(\frac{2}{3}\right)^2-\left(\frac{1}{3}\right)^2\right]+\frac{11}{14}\left[1-\left(\frac{5}{11}\right)^2-\left(\frac{6}{11}\right)^2\right]=0.4848$$

最小的基尼系数为 0.4848。

同理，采用“饮食偏好”特征划分样本集，按照油腻和非油腻划分，基尼系数最小，为 0.4500。

采用“父辈血压”特征划分样本集，基尼系数为 0.4583。

对于“年龄”特征，根节点包含的样本在该特征上的取值排序为{27，28，31，34，36，42，43，44，45，50，61，61，65，66}；阈值 T 取值为{27.5，29.5，32.5，35，39，42.5，43.5，44.5，47.5，55.5，63，65.5}，当 $T=27.5$ 时，“年龄$<T$”组 1 人，高血压 0 人；“年龄$>T$”组 13 人，高血压 7 人，基尼系数为

$$\mathrm{Gini}^1_{\mathrm{age}}=\frac{1}{14}\left[1-1^2\right]+\frac{13}{14}\left[1-\left(\frac{7}{13}\right)^2-\left(\frac{6}{13}\right)^2\right]=0.4615$$

T 取其他值情况下，基尼系数依次为 0.5、0.4848、0.5、0.4889、0.5、0.4898、0.4583、0.4889、0.5、0.5、0.4615，最小基尼系数为 0.4583，对应划分阈值 $T=44.5$。

采用“饮食偏好”特征划分样本集基尼系数最小，用该特征将根节点一分为二。

(2) 下一级节点分枝。

节点②，4 个样本，为序号 1、5、6、10，高血压 3 人，采用“体重”特征划分样本集，偏重组 3 人，高血压 2 人，超重组 1 人，高血压 1 人，基尼系数为 0.3333。

采用“父辈血压”特征划分样本集，基尼系数为 0.25。

对于“年龄”特征，根节点包含的样本在该特征上的取值排序为{27，34，50，61}；阈值 T 取值为{30.5，42，55.5}，当 $T=30.5$ 时，基尼系数为最小值 0。

采用“年龄”特征划分样本集基尼系数最小，用年龄≤30.5 将根节点一分为二。

(3) 同样的方法依次处理直到叶节点，生成的决策树如图 5-10 所示，方框表示叶节点。节点⑧采用“饮食偏好”和“年龄”划分基尼系数相等，此处采用“饮食偏好”划分。最终的决策树有 8 个叶节点，对应 8 条决策规则。

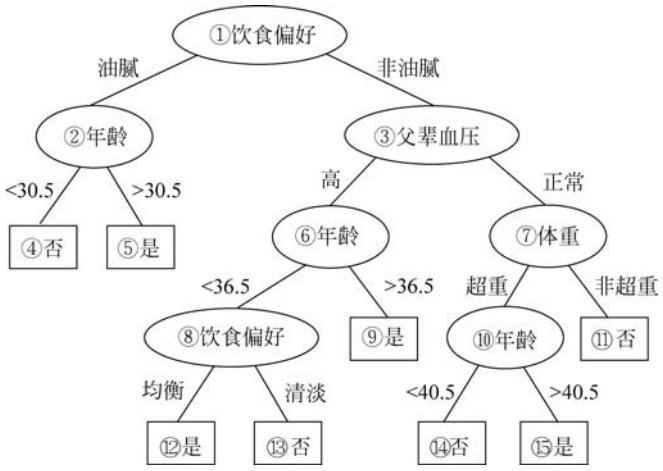


图 5-10　用 CART 算法对表 5-4 中数据判读是否高血压的决策树

5.3.3　过学习与决策树的剪枝

从 5.3.2 节的例子可以看出，决策树构建的同时，建立了决策规则，可以利用决策规则对未知类别的样本进行分类。依据有限样本全部正确划分为准则建立决策规则，需要考虑为未来数据分析时的成功率，即分类器的泛化性能。如果一个算法在训练数据上表现良好，但在测试数据或未来的新数据上的表现与在训练数据上差别很大，称该算法遇到了过学习或过适应。

在有限的样本下，如果决策树生长得很大（分枝多、级别多），则可能会抓住由于偶然性或噪声带来的假象，导致过学习。例如例 5-7 利用 ID3 算法生成的决策树，仅 14 个训练样本，却生成了 12 条决策规则，每条决策规则对应的样本数很少，很有可能是偶然现象，导致了对于 5 个待测样本进行判断时错误率较高。

因此，要控制决策树规模，即剪枝（Prunning），防止出现过学习。决策树剪枝的方法有两种基本策略：先剪枝和后剪枝。

1. 先剪枝

先剪枝指的是在决策树生长过程中判断节点是继续分枝还是直接作为叶节点，某些节点不再分枝，将减小决策树的规模。

先剪枝的关键在于确定节点是否继续分枝的判断条件，有多种方法，比如：设定信息增益阈值，若小于该值，则停止生长，但该阈值不易设定；统计已有节点的信息增益分布，若继续生长得到的信息增益与该分布相比不显著，则停止生长等。剪枝主要是为了提高决策树泛化性能，可以通过实时检测当前决策树对于测试样本集的决策性能来判断节点是否继续分枝：性能有所提高，则继续分枝；性能没有提高，则直接作为叶节点。

因此，生成先剪枝决策树时，对于每个节点，进行如下操作：

（1）将节点视为叶节点，标记为该节点训练样本中数目最多的类别。

（2）计算当前决策树对于测试样本集的分类精度。

（3）寻找进行分枝的最优特征并进行分枝。

（4）对分枝后的子节点进行标记。

（5）验证分枝后的决策树分类精度是否有提高：若有，则进行分枝；若没有，则剪枝（即

禁止分枝),当前节点作为叶节点。

【例 5-11】 以表 5-5 中的 1～14 号样本为训练样本,15～20 号样本为测试样本,采用 ID3 算法生成先剪枝决策树。

解:(1) 根节点是否分枝判断。

首先,标记。当前节点 14 个样本,7 个高血压,7 个正常血压,标记为“高血压”。

其次,计算分类精度。当前为根节点,认为所有的样本均为“高血压”,对于测试样本,6 个样本 3 个判断正确,分类精度为 3/6。

再次,寻找进行分枝的最优特征并分枝。由例 5-7 可知,采用“饮食偏好”特征划分样本集信息增益最大,用该特征将根节点划分为三个子节点②③④,各自的样本数及高血压样本数分别为油腻(4,3),均衡(4,2),清淡(6,2)。

然后,节点②③④分别标记为“高血压”“高血压”和“正常血压”。

最后,验证当前决策树的分类精度。测试样本集中 16、17、18、19 号样本分类正确,分类精度 4/6,有提高,进行分枝。

(2) 节点②是否分枝判断。

首先,寻找进行分枝的最优特征并分枝。采用“年龄”特征划分样本集信息增益最大,用该特征将根节点划分为三个子节点,各自的样本数及高血压样本数分别为青年(2,1),中年(1,1),老年(1,1)。

其次,标记。三个子节点均被标记为“高血压”。

最后,验证当前决策树的分类精度。测试样本集中 16、17、18、19 号样本被正确分类,分类精度为 4/6,没有提高,禁止该分枝,节点②为叶节点,标记为“高血压”。

(3) 节点③是否分枝判断。

首先,寻找进行分枝的最优特征并分枝。采用“体重”特征划分样本集信息增益最大,用该特征将根节点划分为三个子节点,各自的样本数及高血压样本数分别为正常(1,1),超重(2,1),偏重(1,0)。

其次,标记。子节点分别被标记为“高血压”“高血压”和“正常血压”。

最后,验证当前决策树的分类精度。测试样本集中 17、18 号样本被正确分类,分类精度为 2/6,降低了,禁止该分枝,节点③为叶节点,标记为“高血压”。

(4) 节点④是否分枝判断。

首先,寻找进行分枝的最优特征并分枝。采用“父辈血压”特征划分样本集信息增益最大,用该特征将根节点划分为两个子节点,各自的样本数及高血压样本数分别为高血压(3,2),正常血压(3,0)。

其次,标记。子节点分别被标记为“高血压”和“正常血压”。

最后,验证当前决策树的分类精度。测试样本集中 16、17、18、19 号样本被正确分类,分类精度为 4/6,没有提高,禁止分枝,节点④为叶节点,标记为“正常血压”。

图 5-11 基于 ID3 算法利用表 5-5 中数据生成先剪枝决策树

没有可分枝的节点,决策树生成到此为止,最终形成的先剪枝决策树如图 5-11 所示。

由例 5-11 可知,构建决策树时进行先剪枝,很多

分枝未展开，降低了过学习的风险，减少了训练时间、测试时间开销；有些分枝虽不能提升泛化性能，但在其基础上的后续划分有可能提高性能，因此，先剪枝有可能带来欠学习的风险。

2. 后剪枝

后剪枝指的是在决策树充分生长后对其进行修剪，核心思想是合并分枝，从叶节点出发，如果消除具有相同父节点的节点后能够提高决策树的泛化性能则消除，并以其父节点作为新的叶节点；不断地从叶节点往上回溯，直到合并操作不再合适为止。

因此，生成后剪枝决策树时，进行如下操作：

(1) 生成一棵完整的决策树，并计算决策树对于测试样本集的分类精度。

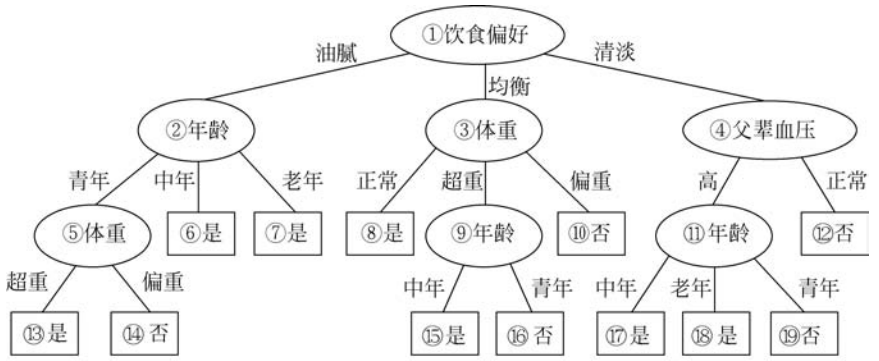
(2) 从最高级开始合并节点，对其父节点进行标记。

(3) 计算合并分枝后的决策树分类精度，并判断是否有提高：若有，则剪枝(即进行合并)；若没有，则不剪枝(保留分枝)。

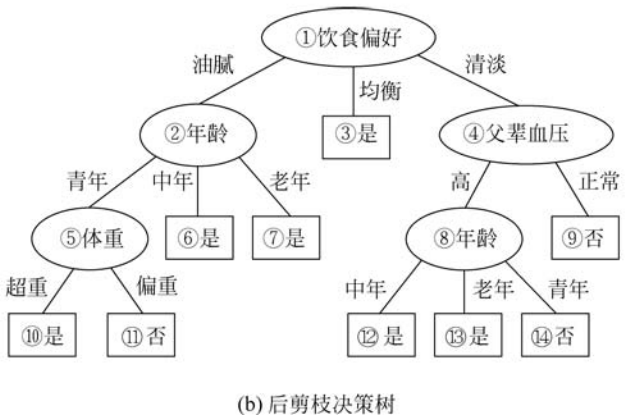
(4) 向上回溯，重复(2)、(3)的合并、判断操作，直到不再需要合并为止。

【例 5-12】 以表 5-5 中的 1～14 号样本为训练样本，15～20 号样本为测试样本，采用 ID3 算法生成后剪枝决策树。

解：(1) 生成完整的决策树如图 5-12(a)所示，对测试样本进行分类，6 个样本，两个正确，分类精度为 2/6。



(a) 未剪枝决策树



(b) 后剪枝决策树

图 5-12　基于 ID3 算法利用表 5-5 中数据生成后剪枝决策树

(2) 合并第 3 级的节点⑬和⑭,其父节点⑤,2 个样本中 1 个高血压样本,标记为“高血压”;对测试样本进行验证,6 个样本中 2 个判断正确,精度不变,不剪枝。

(3) 合并第 3 级的节点⑮和⑯,其父节点⑨,2 个样本中 1 个高血压样本,标记为“高血压”;对测试样本进行验证,6 个样本中 2 个判断正确,精度不变,不剪枝。

(4) 合并第 3 级的节点⑰、⑱和⑲,其父节点⑪,3 个样本中 2 个高血压样本,标记为“高血压”;对测试样本进行验证,6 个样本中 2 个判断正确,精度不变,不剪枝。

(5) 合并第 2 级的节点⑤、⑥和⑦,其父节点②,4 个样本中有 3 个高血压样本,标记为“高血压”;对测试样本进行验证,6 个样本中 2 个判断正确,精度不变,不剪枝。

(6) 合并第 2 级的节点⑧、⑨和⑩,其父节点③,4 个样本中有 2 个高血压样本,标记为“高血压”;对测试样本进行验证,6 个样本中 4 个判断正确,精度提高,剪枝。

(7) 合并第 2 级的节点⑪和⑫,其父节点④,6 个样本中有 2 个高血压样本,标记为“正常血压”;对测试样本进行验证,6 个样本中 4 个判断正确,精度没有提高,不剪枝。

(8) 合并第 1 级的节点②、③和④,其父节点为根节点,标记为“高血压”,对测试样本进行验证,6 个样本中 3 个判断正确,精度没有提高,不剪枝。

至此,没有节点可以再合并,最后生成的后剪枝决策树如图 5-12(b)所示。

由例 5-12 可知,构建决策树时进行后剪枝,比先剪枝保留了更多的分支,欠学习风险小,泛化性能一般优于先剪枝决策树;但是由于需要先生成完整决策树,训练时间和测试时间开销大。

5.3.4 仿真实现

MATLAB 中提供了生成分类决策树的相应函数,对其进行简要介绍。

1) ClassificationTree 类

表示用于分类的二分决策树,使用 fitctree 函数创建 ClassificationTree 类对象,使用 predict 函数对样本进行分类决策,有多种属性和函数。

2) fitctree 函数

使用 CART 算法创建 ClassificationTree 类对象,调用格式如下。

(1) TREE = fitctree(TBL,Y):利用表 TBL 中的数据训练 ClassificationTree 类 TREE,Y 可以是类别标号矩阵、TBL 中变量名或者用字符串表示的公式。

(2) TREE=fitctree(X,Y):利用 X 中的数据训练 ClassificationTree 类 TREE。X 为 N×n 的矩阵,Y 指定 N 个样本的类别标号。

(3) TREE=fitctree(...,'PARAM1',val1,'PARAM2',val2,…):指定参数训练决策树。部分参数如表 5-6 所示。

表 5-6 fitctree 函数部分参数

参数名称	取值及含义
ClassNames	类名称矩阵,指定参与训练的类别,使用 Y 中的全部或部分元素对应的类,默认为全部
Cost	方阵,Cost(i,j)为 ω_i 类样本归为 ω_j 类的损失,默认为 0～1 损失
MaxNumSplits	最大分支节点数,默认为 size(X,1)－1

续表

参数名称	取值及含义
MinLeafSize	指定叶节点最少要包含的样本数，默认为 1
MinParentSize	指定内部节点最少要包含的样本数。默认为 10
Prune	取'on'(默认)，进行剪枝；取'off'，不剪枝
PruneCriterion	指定剪枝标准，可取'error'(默认)或'impurity'
SplitCriterion	指定分枝准则，可取'gdi'(基尼的多样性指数，默认)、'twoing'、'deviance'
Weights	样本权值向量，默认为全 1

3) predict 函数

(1) label= predict(TREE,X)：利用训练好的 ClassificationTree 模型 TREE 对矩阵或表 X 中的数据进行分类决策，返回分类标签向量。

(2) label=predict(TREE,X,'Subtrees',value)：根据 Subtrees 指定级别的剪枝子树进行预测。'Subtrees'可取按升序排列的非负整数的向量，0 表示不修剪的完整决策树，max(TREE. PruneList)表示只含根节点；取'all'，要在 0:max(TREE. PruneList)整个范围的剪枝子树上进行决策。

(3) [label,score,node,cnum]= predict(…)：同时返回分类得分矩阵(后验概率)score、预测的节点序号向量 node 和预测的类别编号向量 cnum。

4) view 函数

(1) view(TREE)：在命令窗口输出决策树 TREE 的文字描述，以查看生成的决策树。

(2) view(TREE,'Mode',Value)：指定决策树的查看方式。'Mode'可取'text'(默认)或'graph'，后者将打开分类树查看器实现交互式查看。

5) prune 函数

(1) tree1= prune(TREE)：生成决策树 TREE 的最佳剪枝决策树。

(2) tree1 = prune(TREE,Name,Value)：指定参数生成剪枝决策树。参数包括：'Alpha'，剪枝代价；'Level'，0 到 max(TREE. PruneList)的数值，剪枝级别，叶节点对应 0 级，内部节点的级别是其到最远叶节点的级差；'Nodes'，取值在 1 到 TREE. NumNodes 的数值向量，指定的节点为 tree1 的叶节点。

【例 5-13】 以表 5-4 中的 1～14 号样本为训练样本，采用 fitctree 函数生成分类决策树，并对 15～20 号样本进行测试。

程序如下：

```
clc,clear,close all;
Age = [34;28;42;45;50;61;65;36;31;27;44;43;61;66];                    % 年龄特征
Weight = {'超重';'正常';'偏重';'超重';'偏重';'偏重';'正常';…
          '超重';'正常';'偏重';'偏重';'偏重';'正常';'正常''};          % 体重特征
Diet = {'油腻';'均衡';'清淡';'均衡';'油腻';'油腻';'清淡';…
        '均衡';'清淡';'油腻';'清淡';'均衡';'清淡';'清淡'};            % 饮食偏好特征
Parent = {'高';'高';'高';'正常';'正常';'正常';'高';…
           '正常';'高';'高';'正常';'正常';'正常';'正常'};             % 父辈血压特征
X = table(Age,Weight,Diet,Parent);                                     % 创建表
Y = {'是';'是';'是';'是';'是';'是';'是';'否';'否';'否';'否';'否';'否';'否'}; % 类别标签
```

```
ctree = fitctree(X,Y,'Prune','off','MinParentSize',2);
                                                    %生成不剪枝决策树,内部节点最少2个样本
view(ctree)                          %在命令窗口输出决策树的文字描述
view(ctree,'mode','graph')                                    %决策树图形显示
Age = [25;35;48;40;63;62];
Weight = {'正常';'偏重';'超重';'正常';'偏重';'正常''};
Diet = {'均衡';'均衡';'油腻';'清淡';'均衡';'均衡'};
Parent = {'正常';'正常';'高';'正常';'高';'正常'};
test = table(Age,Weight,Diet,Parent);                         %测试样本表
[label,score,node,cnum] = predict(ctree,test);                %对测试样本进行决策分类
```

程序运行,将在命令窗口输出文字描述的决策树,图形显示的决策树如图5-13所示,和图5-10一致,最后一个节点采用“饮食偏好”和“年龄”划分基尼系数相等,程序中“年龄”特征在前,因此采用“年龄”划分。6个测试样本分类结果依次为否、否、是、否、是、否,16号样本判断错误。

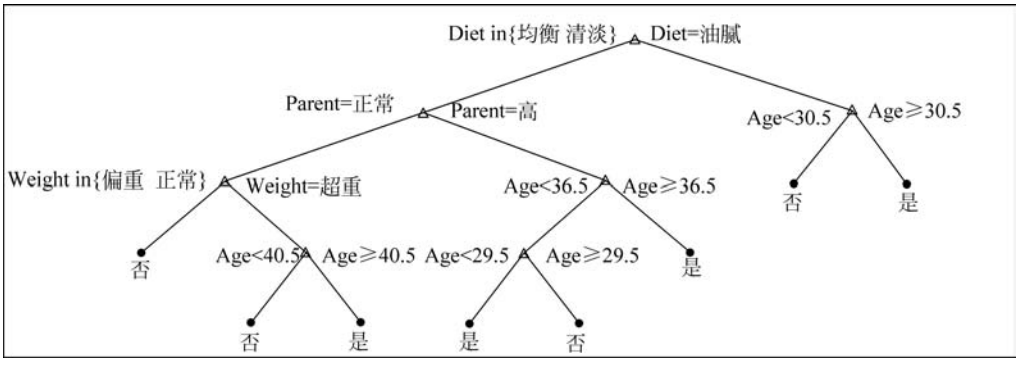


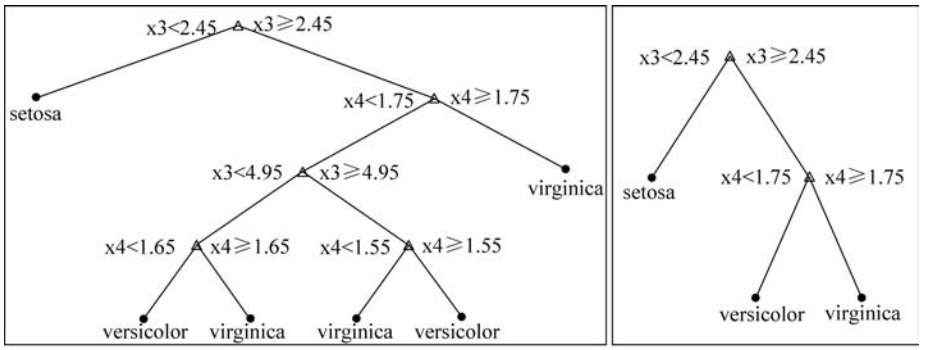
图5-13 fitctree生成的判读是否高血压的未剪枝决策树

【例5-14】 利用fisheriris数据集设计决策树,生成剪枝决策树,并利用不同剪枝决策树对样本$[4.8 \quad 3.5 \quad 1.5 \quad 0.2]^{T}$进行判别。

程序如下:

```
clc,clear,close all;
load fisheriris
ctree = fitctree(meas,species,'Prune','off','MinParentSize',5); %生成不剪枝决策树
view(ctree,'mode','graph')
subtree = prune(ctree,'Level',2);                             %生成2级剪枝决策树
view(subtree,'Mode','graph');
pattern = [4.8 3.5 1.5 0.2];
label1 = predict(ctree,pattern)                               %使用不剪枝决策树决策
label2 = predict(ctree,pattern,'Subtrees','all')              %使用各级剪枝决策树决策
label3 = predict(subtree,pattern)                             %使用2级剪枝决策树决策
```

程序运行,生成的决策树如图5-14所示,同时在命令窗口输出决策结果:label1 = {'setosa'};label2为1×5的元胞数组,5个元素均为{'setosa'},分别对应0~4级剪枝决策树的决策结果;label3 ={'setosa'}。



(a) 不剪枝决策树　　(b) 2级剪枝决策树

图 5-14　利用 fisheriris 数据集生成的决策树

5.4　Logistic 回归

Logistic 回归(Logistic Regression)是一种经典分类方法,也称为对数概率回归(也有文献译为“逻辑回归”“逻辑斯谛回归”等),是采用对数概率模型描述样本属于某类的可能性与样本特征之间的关系,用训练数据估计模型中的系数,进而实现分类的方法。

5.4.1　基本原理

考虑二分类任务,采用线性判别函数 $g(\boldsymbol{x})=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0$ 对样本 $\boldsymbol{x}$ 进行归类判别,设类别标签 $y\in\{0,1\}$,将样本归类,需要根据 $g(\boldsymbol{x})$的值判断 y 的值,因此,有

$$y=f[g(\boldsymbol{x})] \tag{5-16}$$

$f(\cdot)$称为 Link 函数(Link Function)。

例如:当 $g(\boldsymbol{x})>0$ 时,$\boldsymbol{x}\in\omega_1$,$y=1$;当 $g(\boldsymbol{x})<0$ 时,$\boldsymbol{x}\in\omega_2$,$y=0$;当 $g(\boldsymbol{x})=0$ 时,可以任意判别。Link 函数 $f(\cdot)$实际是单位阶跃函数,即

$$y=\begin{cases}0, & g(\boldsymbol{x})<0\\ 0.5, & g(\boldsymbol{x})=0\\ 1, & g(\boldsymbol{x})>0\end{cases} \tag{5-17}$$

函数如图 5-15 所示。

单位阶跃函数不连续,用单调可微的 Logistic 函数近似表达单位阶跃函数为

$$y=\frac{1}{1+\mathrm{e}^{-g(x)}} \tag{5-18}$$

Logistic 函数将 $g(\boldsymbol{x})$的值转换为一个接近 0 或 1 的 y 值,在 $g(\boldsymbol{x})=0$ 附近变化很陡,图形如图 5-15 中虚线所示。

由式(5-17)可知,y 可以看作将样本 $\boldsymbol{x}$ 归为 ω_1 类的可能性,则 $1-y$ 是将样本归为 ω_2 类的可能性,当 $y>1-y$ 时,$\boldsymbol{x}\in\omega_1$;当 $y<1-y$ 时,$\boldsymbol{x}\in\omega_2$。定义概率为

$$\frac{y}{1-y}=\mathrm{e}^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0} \tag{5-19}$$

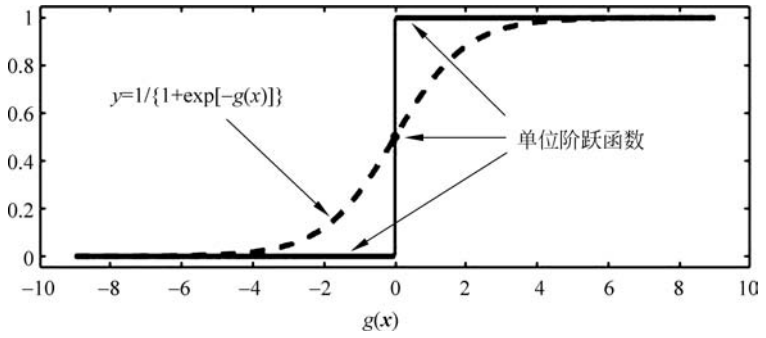


图 5-15　单位阶跃函数和 Logistic 函数

反映了 $\boldsymbol{x}$ 归为 ω_1 类的相对可能性。

对概率取对数得到 logit 函数

$$\text{logit}(\boldsymbol{x})=\ln\left(\frac{y}{1-y}\right)=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0 \tag{5-20}$$

表明样本属于某类的可能性与样本之间呈线性关系。很明显

$$\text{当 logit}(\boldsymbol{x})\gtrless 0\text{ 时，}\quad \boldsymbol{x}\in\begin{cases}\omega_1\\ \omega_2\end{cases} \tag{5-21}$$

如果能够确定 $\boldsymbol{w}$ 和 w_0，则可以确定 logit 函数，从而实现分类。

进一步，样本属于某一类的可能性用后验概率来表示，logit 函数重写为

$$\ln\left[\frac{P(y=1\mid\boldsymbol{x})}{P(y=0\mid\boldsymbol{x})}\right]=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0 \tag{5-22}$$

其中，

$$P(y=1\mid\boldsymbol{x})=\frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0)}}=\frac{\mathrm{e}^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0}}{1+\mathrm{e}^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0}} \tag{5-23}$$

$$P(y=0\mid\boldsymbol{x})=1-P(y=1\mid\boldsymbol{x})=\frac{\mathrm{e}^{-(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0)}}{1+\mathrm{e}^{-(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0)}}=\frac{1}{1+\mathrm{e}^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0}} \tag{5-24}$$

采用最大似然估计的方法确定 $\boldsymbol{w}$ 和 w_0，给定数据集 $\mathcal{X}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N\}$，对应的类别标号为 $\mathcal{Y}=\{y_1,y_2,\cdots,y_N\}$，对于每一个样本 $\boldsymbol{x}_i$，$i=1,2,\cdots,N$，有 $y_i\in\{0,1\}$，则

$$P(y_i\mid\boldsymbol{x}_i;\boldsymbol{w},w_0)=[P(y_i=1\mid\boldsymbol{x}_i;\boldsymbol{w},w_0)]^{y_i}[1-P(y_i=1\mid\boldsymbol{x}_i;\boldsymbol{w},w_0)]^{1-y_i} \tag{5-25}$$

定义对数似然函数为

$$h(\boldsymbol{w},w_0)=\sum_{i=1}^{N}\ln P(y_i\mid\boldsymbol{x}_i;\boldsymbol{w},w_0) \tag{5-26}$$

将式(5-23)、式(5-24)和式(5-25)代入式(5-26)，得

$$h(\boldsymbol{w},w_0)=\sum_{i=1}^{N}\{-y_i\ln[1+\mathrm{e}^{-(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+w_0)}]-(1-y_i)\ln(1+\mathrm{e}^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+w_0})\} \tag{5-27}$$

对数似然函数求最大值得到最优的 $\boldsymbol{w}$ 和 w_0，或者对负对数似然函数求最小值，即

$$-h(\boldsymbol{w},w_0)=\sum_{i=1}^{N}\{y_i\ln[1+\mathrm{e}^{-(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+w_0)}]+(1-y_i)\ln(1+\mathrm{e}^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+w_0})\}\tag{5-28}$$

可以通过迭代策略求解。

【例 5-15】 三维空间两类分类问题，样本集为

$$\omega_1:\{[0\quad 0\quad 0]^{\mathrm{T}},[1\quad 0\quad 1]^{\mathrm{T}},[1\quad 0\quad 0]^{\mathrm{T}},[1\quad 1\quad 0]^{\mathrm{T}}\}$$

$$\omega_2:\{[0\quad 0\quad 1]^{\mathrm{T}},[0\quad 1\quad 1]^{\mathrm{T}},[0\quad 1\quad 0]^{\mathrm{T}},[1\quad 1\quad 1]^{\mathrm{T}}\}$$

试用 Logistic 回归求解判别函数权向量，并对样本$[0\quad 0.6\quad 0.8]^{\mathrm{T}}$进行判别。

程序如下：

```
clc,clear,close all;
gX = @(W,X) W(1) * X(:,1) + W(2) * X(:,2) + W(3) * X(:,3) + W(4);    % 线性函数
logisticfun = @(W,X) 1./(1 + exp( - gX(W,X)));                       % Logistic 函数
training = [0 0 0;1 0 0;1 0 1;1 1 0;0 0 1;0 1 1;0 1 0;1 1 1];
label = [1;1;1;1;0;0;0;0];                                           % 训练样本集类别标签
negloglikfun = @(W) - sum(label. * log(logisticfun(W,training)) + ...
                          (1 - label). * log(1 - logisticfun(W,training))); % 负对数似然函数
W0 = [0;0;0;0];                                                      % 迭代求解初始值
opts = optimset('fminsearch');
opts.MaxFunEvals = Inf;
opts.MaxIter = 10000;                                                % 求最小值的相关设置
WHatML = fminsearch(negloglikfun,W0,opts)          % 求无约束多变量函数的最小值
pattern = [0 0.6 0.8];                                               % 待测样本
logitX = WHatML(1) * pattern(1) + WHatML(2) * pattern(2) + WHatML(3) * pattern(3) + WHatML(4);
if logitX > 0                              % 计算 logit 函数值并判断类别
    result = 1;
else
    result = 0;
end
```

程序中利用了 fminsearch 函数实现负对数似然函数求最小值，该函数利用 Nelder-Mead 单纯性算法求极值。程序中最小值对应的权向量 WHatML 为[23.5306　−24.3546　−24.086212.5739]$^{\mathrm{T}}$，但是受初值的影响会发生变化。待测样本类别标签为 0，归为第二类。

5.4.2　多分类任务

设训练样本集为 $\mathcal{Y}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N\}$，各自的类别标号 $y_i\in\{1,2,\cdots,c\}$，$c>2$ 为类别数。记将样本 $\boldsymbol{x}$ 标记为 j 的概率为 $P_j=P(y=j\mid\boldsymbol{x})$，$j=1,2,\cdots,c$，$\sum_{j=1}^{c}P_j=1$，改写式(5-22)为

$$\ln\left(\frac{P_j}{\sum_{l\neq j}P_l}\right)=\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{x}+w_{j0}\tag{5-29}$$

$$P_j=\frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{x}+w_{j0})}}=\frac{\mathrm{e}^{\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{x}+w_{j0}}}{1+\mathrm{e}^{\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{x}+w_{j0}}} \tag{5-30}$$

$$\sum_{l\neq j}P_l=1-P_j=\frac{\mathrm{e}^{-(\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{x}+w_0)}}{1+\mathrm{e}^{-(\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{x}+w_0)}}=\frac{1}{1+\mathrm{e}^{\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{x}+w_0}} \tag{5-31}$$

同样采用最大似然估计的方法确定 $\boldsymbol{w}_j^{\mathrm{T}}$ 和 w_{j0}。

5.4.3 仿真实现

MATLAB 提供了 mnrfit 函数拟合多分类 Logistic 回归模型，mnrval 函数计算预测概率，主要的调用格式如下：

(1) B=mnrfit(X,Y)：根据矩阵 X 中的数据和对应的响应变量 Y 拟合多分类 Logistic 回归模型。X 为 N×n 的矩阵，函数自动实现样本的增广化；Y 可以是 N×c 的矩阵，Y(i,j)表示 X 中第 i 行的数据对应第 j 类的输出；Y 可以是一个 N 维的列向量，元素值为 1 到 c 的整数值，也可以是一个 N 维的 categorical 数组，表明每个样本的类别；输出变量 B 为估计的(n+1)×(c−1)的权向量矩阵，第一行为常数项，其余行依次对应数据的各维变量；B 的每一列为该类相对于其余 c−1 类的 Logistic 回归分类权向量。

(2) PHAT=mnrval(B,X)：根据多分类 Logistic 回归模型系数 B，对 X 中的数据进行预测，返回 N×c 的预测概率矩阵 PHAT。

另有 fitglm 函数创建广义线性回归模型 GeneralizedLinearModel，设置参数'Link'为'logit'，实现 Logistic 回归分析；使用 predict 函数实现预测。

【例 5-16】 利用 MATLAB 函数对例 5-15 中的数据进行 Logistic 回归分析。

程序如下：

```
clc,clear,close all;
gX = @(W,X) W(1) + W(2) * X(:,1) + W(3) * X(:,2) + W(4) * X(:,3); % 线性函数,常数项为第一项
logisticfun = @(W,X) 1./(1 + exp( - gX(W,X)));                    % Logistic 函数 P(y = 1|X)
training = [0 0 0;1 0 0;1 0 1;1 1 0;0 0 1;0 1 1;0 1 0;1 1 1];
label = [1;1;1;1;2;2;2;2];
B = mnrfit(training,label);              % 拟合模型,输出系数矩阵 B,同线性函数中的 w
pattern = [0 0.6 0.8];                                            % 待测样本
P = mnrval(B,pattern);                                            % 预测,输出概率矩阵
Py1 = logisticfun(B,pattern);                                     % 根据式(5 - 23)计算 P(y = 1|X)
Py2 = 1 - Py1;                                                    % 计算 P(y = 2|X)
logitX = log(Py1/Py2); % 计算 logit 函数,或 logitX = log(P(1)/P(2))或 logitX = gX(B,pattern);
if logitX > 0
    result = 1
else
    result = 2
end
```

程序运行，系数矩阵 $\boldsymbol{B}$ 为$[37.0123\quad 148.2485\quad -93.8968\quad -141.8509]^{\mathrm{T}}$，即 $\ln\left[\frac{P(y=1|x)}{P(y=2|x)}\right]=37.0123+148.2485x_1-93.8968x_2-141.8509x_3$。预测概率矩阵 $\boldsymbol{P}$ 为

[0　1]，即 $P(y=1|\boldsymbol{x})=0$，$P(y=2|\boldsymbol{x})=1$，与 Py1、Py2 一致。对待测样本计算 logit 函数值为负值，输出 result=2，即将待测样本归为第 2 类。

【例 5-17】 利用 fisheriris 数据集拟合 Logistic 回归模型，并对样本[6.3　2.8　4.9　1.7]^{T} 进行归类。

程序如下：

```
clc,clear,close all;
load fisheriris
sp = categorical(species);                    % 将类别标记 species 转化为 categorical 数组
iris = ["setosa" "versicolor" "virginica"];   % 类名称
B = mnrfit(meas,sp);                          % 拟合 Logistic 回归模型
pattern = [6.3 2.8 4.9 1.7];                  % 待测样本
P = mnrval(B,pattern);                        % 对待测样本进行预测,输出预测概率矩阵 P
[~,pos] = max(P);                             % 找出最大概率对应的类
result = iris(pos);                           % 输出归类结果
disp('Logistic 回归分类:')
disp(result)
```

程序运行，预测概率矩阵 $\boldsymbol{P}$ 为[0　0.3977　0.6023]，即待测样本属于 setosa、versicolor 和 virginica 类的概率分别为 0、0.3977 以及 0.60230，归入最大概率对应的类，在命令窗口输出：

Logistic 回归分类：

```
virginica
```

5.5　非线性判别分析的实例

【例 5-18】 对由不同字体的数字图像构成的图像集，实现基于最近邻法和 Logistic 回归的分类器设计及判别。

1. 设计思路

两种分类方法均采用和例 2-16 相同的预处理方法，提取同样的特征。在最近邻法中，测试样本在训练样本集中寻找最近邻，根据近邻的类别归类。设计方案如图 5-16 所示。Logistic 回归分类要先拟合多分类 Logistic 回归模型，再根据概率实现分类决策，设计方案如图 5-17 所示。

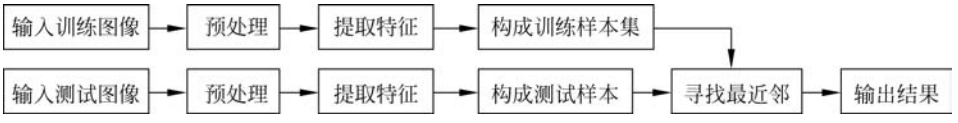


图 5-16　最近邻归类设计方案框图

2. 程序设计

1）生成训练样本

读取训练图像文件，经过图像预处理，提取特征，生成训练样本。同例 2-16。

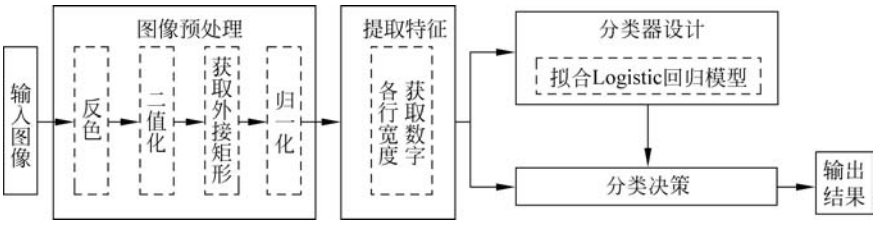


图 5-17　Logistic 回归分类设计方案框图

2）拟合多分类 Logistic 回归模型

```
Labeltrain1 = categorical(labeltrain); % 将 0～9 的数字转换为 categorical 数组
B = mnrfit(training,labeltrain1);
```

3）生成测试样本

读取测试图像文件，经过图像预处理，提取特征，生成测试样本。同例 2-16。

4）对测试样本进行决策归类

（1）利用训练样本生成 KDTreeSearcher 模型，并寻找最近邻，根据最近邻的类别决策归类，并测算检测率。

```
Mdl = KDTreeSearcher(training);
[Idx,D] = knnsearch(Mdl,testing);
testnb = labeltrain(Idx);
ratio1 = sum(testnb == labeltest)/N
```

（2）利用 Logistic 回归模型系数 $\boldsymbol{B}$ 对测试样本进行预测，根据预测概率大小归类，并测算检测率。

```
P = mnrval(B,testing);
[～,testlogit] = max(P,[],2); % 预测概率矩阵 P 每一行求最大，对应的位置即是类别对应的正整数
testlogit = testlogit - 1;
ratio2 = sum(testlogit == labeltest)/N
```

3. 实验结果

运行程序，采用 10 个数字的共 50 幅图像进行训练，采用 30 幅图像进行测试，在命令窗口输出 ratio1＝0.7667 和 ratio2＝0.8000，即对于测试图像，最近邻法正确率达到 76.67%，Logistic 回归分类正确率达到 80%。

习题

1. 简述最小距离分类器的分类方法及特点。
2. 简述近邻法的分类思路及特点。
3. 简述决策树方法的分枝过程和决策过程，并说明决策树剪枝优化的思路。
4. 对例 5-10 中 CART 算法生成的决策树进行剪枝。
5. 已知二维空间三类样本均服从正态分布$\boldsymbol{\mu}_1=[1\quad 1]^{\mathrm{T}}$，$\boldsymbol{\mu}_2=[4\quad 4]^{\mathrm{T}}$，$\boldsymbol{\mu}_3=[8\quad 1]^{\mathrm{T}}$，

$\boldsymbol{\Sigma}_1=\boldsymbol{\Sigma}_2=\boldsymbol{\Sigma}_3=2\boldsymbol{I}$，编写程序，基于这三类生成1000个二维向量的数据集，分别采用欧氏距离和马氏距离，利用数据集设计最小距离分类器。

6. 编写程序，随机生成二维向量作为待测样本，利用上题的训练样本，采用最近邻和5近邻方法对待测样本进行归类。

7. 利用3类样本 $\omega_1:\left\{\begin{bmatrix}-5\\-5\end{bmatrix},\begin{bmatrix}-5\\-4\end{bmatrix},\begin{bmatrix}-4\\-5\end{bmatrix},\begin{bmatrix}-5\\-6\end{bmatrix},\begin{bmatrix}-6\\-5\end{bmatrix}\right\}$，$\omega_2:\left\{\begin{bmatrix}5\\5\end{bmatrix},\begin{bmatrix}5\\4\end{bmatrix},\begin{bmatrix}4\\5\end{bmatrix},\begin{bmatrix}5\\6\end{bmatrix},\begin{bmatrix}6\\5\end{bmatrix}\right\}$，$\omega_3:\left\{\begin{bmatrix}-5\\5\end{bmatrix},\begin{bmatrix}-5\\4\end{bmatrix},\begin{bmatrix}-4\\5\end{bmatrix},\begin{bmatrix}-5\\6\end{bmatrix},\begin{bmatrix}-6\\5\end{bmatrix}\right\}$，拟合Logistic回归模型，并对数据$[-2\ -2]^{\mathrm{T}}$进行归类。