一个多元逐步回归的二叉树分类器的设计与实现*

黄莎白　　刘白桦

（中国科学院沈阳自动化研究所）

〔提要〕作者设计并实现了一个基于多变元逐步回归的二叉树分类器。在树结构和特征子集的选择中采用了穷举法，比有限制条件的选择更合理更优化。用FORTRAN语言实现的“遍历”二叉树，充分利用了FORTRAN处理可调数组的能力，并采取适当技巧，从而最大限度地利用了计算机内存。该通用分类器，可用来对任何具有统计数据的模式进行分类。在对白血球的分类中，取得了五类97%，六类92.2%的高识别率。

一　基本原理

在数理统计中，将自变量作为非随机变量，而因变量作为随机变量的一类问题归为回归分析问题。利用回归分析设计分类器，回归平面作为每级分类器的分类界面，逐级地线性分类以逼近一个尽可能优的非线性分类，即是一个多元逐步回归的二叉树分类器的实质。

设计一个二叉树分类器要完成三个任务：决定二叉树分类器的结构；在每个非终极点选择较好的特征子集；在每个非终极节点选择一个决策规则。

1．决定二叉树分类器的结构

分类的原则是使残差

$$Q = L_{yy} - L_{yx}L_{xx}^{-1}L_{xy} = \min \qquad (1)$$

式中$L_{yy} = \sum_{i=1}^{n}(y_i - \overline{y})^2$是观测值$y$的总离差平方和，是一定值，它反映观测值$y_1, y_2, \cdots\cdots, y_n$围绕均值的分散程度。于是决定二叉树分类器的结构就变成选择一种分类结构使

$$L_{yx}L_{xx}^{-1}L_{xy} = \max \qquad (2)$$

设共分p类，令

$$D = [D(1)\ D(2)\cdots D(p)]' \qquad (3)$$

式中$D(I) = 1$或-1，$I = 1,2,\cdots, p$，并令$y^* = yD$。

容易证明$L_{y^*y^*} = D'L_{yy}D$，$L_{y^*x} = D'L_{yx}$，$L_{xy^*} = L_{xy}D$，于是

$$L_{y^*x}L_{xx}^{-1}L_{xy^*} = D'L_{yx}L_{xx}^{-1}L_{xy}D = L_{xx}^{-1}(D'L_{yx})^2 \qquad (4)$$

因此问题归结为选择一个D的形式（即非终极节点的树结构形式），使

$$L_{xx}^{-1}(D'L_{yx})^2 = \max$$

2．逐步回归

所谓逐步回归即是在每个非终极节点选择一个特征子集。当自变量个数较多时，要建立回归方程的计算量是相当大的，但是每个自变量与因变量的关系显然不是同等密切的。按照某种原则逐步“引入”重要自变量，“剔除”不重要自变量的过程称为逐步回归。当既不能引进新变量又不能剔除已选变量时，回归过程结束，建立因变量对已选变量的回归方程。

下面叙述选入与剔除自变量的准则。

若已引入的自变量是$x_1, x_2, \cdots, x_{k-1}$则因变量

$$y = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_{k-1}x_{k-1}$$，对应残差为Q

若再引入自变量x_k，则因变量

$$y = \hat{\beta}_{0*} + \hat{\beta}_{1*}x_1 + \cdots + \hat{\beta}_{k-1*}x_{k-1} + \hat{\beta}_{k*}x_k$$ 对应残差为Q_*

令 $\hat{\beta}_* = \begin{vmatrix} \hat{\beta}_{1*} \\ \vdots \\ \hat{\beta}_{k-1*} \end{vmatrix}$，$\hat{\beta}_{k*} = \hat{\beta}_k$，$\hat{\beta} = \begin{vmatrix} \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_{k-1} \end{vmatrix}$

则正规方程为

* 收到本文的时间是1984年6月24日。

$$\begin{vmatrix} L_{xx} & L_{xx_k} \\ L_{x_kx} & L_{x_kx_k} \end{vmatrix} \begin{Bmatrix} \hat{\beta}_* \\ \hat{\beta}_k \end{Bmatrix} = \begin{Bmatrix} L_{xy} \\ L_{x_ky} \end{Bmatrix} \quad (5)$$

或

$$\begin{cases} L_{xx}\hat{\beta}_* + L_{xx_k}\hat{\beta}_k = L_{xy} \\ L_{x_kx}\hat{\beta}_* + L_{x_kx_k}\hat{\beta}_k = L_{x_ky} \end{cases}$$

由此得

$$\hat{\beta}_k = (L_{x_kx_k} - L_{x_kx}L_{xx}^{-1}L_{xx_k})^{-1}(L_{x_ky} - L_{x_kx}L_{xx}^{-1}L_{xy})$$

$$\hat{\beta}_* = L_{xx}^{-1}L_{xy} - L_{xx}^{-1}L_{xx_k}\hat{\beta}_k = \hat{\beta} - L_{xx}^{-1}L_{xx_k}\hat{\beta}_k$$

而

$$\hat{\beta}_{0*} = \bar{y} - \bar{x}\hat{\beta}_* - \bar{x}_k\hat{\beta}_k$$

Q_*一般小于Q。x_k是否有必要引入，取决于$Q-Q_*$的大小，差越大，x_k越重要。

$$Q_* = L_{yy} - (L_{yx}L_{yx_k})\begin{Bmatrix} \hat{\beta}_* \\ \hat{\beta}_k \end{Bmatrix}$$

$$= L_{yy} - [L_{yx}L_{yx_k}]\begin{Bmatrix} \hat{\beta} - L_{xx}^{-1}L_{xx_k}\hat{\beta}_k \\ \hat{\beta}_k \end{Bmatrix}$$

$$= L_{yy} - L_{yx}\hat{\beta} + L_{yx}L_{xx}^{-1}L_{xx_k}\hat{\beta}_k - L_{yx_k}\hat{\beta}_k$$

因 $Q = L_{yy} - L_{yx}\hat{\beta}$

故 $Q_* = Q - (L_{yx_k} - L_{yx}L_{xx}^{-1}L_{xx_k})\hat{\beta}_k$

代入$\hat{\beta}_k$的表达式，得

$$Q_* = Q' - (L_{yx_k} - L_{yx}L_{xx}^{-1}L_{xx_k})(L_{x_kx_k} - L_{x_kx}L_{xx}^{-1}L_{xx_k})^{-1}(L_{x_ky} - L_{x_kx}L_{xx}^{-1}L_{xy}) \quad (6)$$

对矩阵 $L = \begin{vmatrix} L_{xx} & L_{xx_k} \\ L_{x_kx} & L_{x_kx_k} \end{vmatrix}$ 作$(k-1)$次变换，即引入$x_1, x_2, \cdots, x_{k-1}$，得当前矩阵

$$L = \begin{vmatrix} L_{xx}^{-1} & L_{xx}^{-1}L_{xx_k} \\ -L_{x_kx}L_{xx}^{-1} & L_{x_kx_k} - L_{x_kx}L_{xx}^{-1}L_{xx_k} \end{vmatrix} \quad (7)$$

故 $L_{x_kx_k} - L_{x_kx}L_{xx}^{-1}L_{xx_k}$ 为当前矩阵的(k,k)元素，记作L_{kkc}；$L_{yx_k} - L_{yx}L_{xx}^{-1}L_{xx_k}$为当前矩阵的$(y,k)$元素，记作$L_{ykc}$；$L_{x_ky} - L_{x_kx}L_{xx}^{-1}L_{xy}$为当前矩阵的$(k,y)$元素，记作$L_{kyc}$。所以

$$Q_* = Q - L_{ykc}L_{kkc}^{-1}L_{ykc} \quad (8)$$

可见x_k是否引入，由已经引入若干自变量的当前矩阵的(k,k)元素，(k,y)元素和(y,k)元素估价。引入无量纲量$(Q-Q_*)/Q$衡量。

令$(Q_*-Q)/Q = T/L_{yy} = L_{ykc}L_{kkc}^{-1}L_{kyc}/L_{yyc}$（$Q$即当前矩阵的$(y,y)$元素）。取阈值$T_i>0$，$T_d<0$，于是“引入”与“剔除”规则为：选择$x_k$，当$T/L_{yy} \geqslant T_i$时，引入；当其使$T/L_{yy} > T_d$时，剔除。

3. 建立界面方程

最后在每个非终极节点建立界面方程。

经过逐步回归，假定共引入了k个自变量，它们是$x_1, x_2, \cdots, x_k$，且到此时为止，共进行了f次筛选变换，那么这时的当前矩阵

$$L^{f} = \begin{matrix} \\ k \\ m-k \\ p \end{matrix} \begin{matrix} \begin{matrix} k & m-k & p \end{matrix} \\ \begin{vmatrix} L_{11}^{(f)} & L_{12}^{(f)} & L_{13}^{(f)} \\ L_{21}^{(f)} & L_{22}^{(f)} & L_{23}^{(f)} \\ L_{31}^{(f)} & L_{32}^{(f)} & L_{33}^{(f)} \end{vmatrix} \\ \begin{matrix} x_1, \cdots, x_k, x_{k+1}, \cdots, x_m \end{matrix} \\ \begin{matrix} y_1, \cdots, y_p \end{matrix} \end{matrix}$$

这时的回归系数为

$$\hat{\beta} = L_{13}^{(f)}; \quad \hat{\beta}_0 = \bar{y} - \bar{x}\hat{\beta};$$

$\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_k)$。界面方程为

$$\begin{Bmatrix} y_1 \\ \vdots \\ y_p \end{Bmatrix} = \hat{\beta}^T \begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix} + \hat{\beta}_0{}^T \quad (9)$$

二 分类器的设计和实现

1. 三个任务的设计和实现

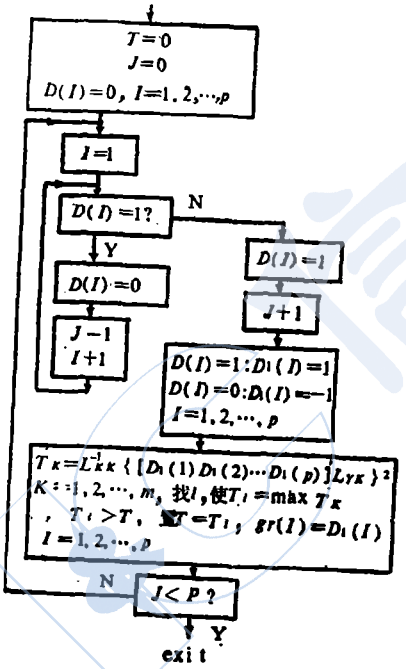
（1）寻找最好的结构形式

首先要决定二叉树的结构（图1），即在每个非终极节点（包括根节点）决定原始数据

的分组。设共p类，其中某些类为一组，其余类为另一组。为了找到最好的树结构形式，我们采用了穷举法：考虑一个除了 0 0 … 0 与 1 1 … 1 的所有的 p 位二进数的集合（共2^p-2个数），让该集合中的每个二进制数与$D(I)(I=1,2,\cdots,p)$的元素一一对应（1 对应 1，0 对应 −1），如二进制数为 101110，则$D=(1,-1,1,1,1,-1)$。计算指标

$$T_k=L_{kk}^{-1}\{[D(1)D(2)\cdots D(p)]L_{yk}\}^2$$

的值。$k=1,2,\cdots,m$，m为自变量（即特征）的总数。找 i，使$T_i=\max T_k$。每计算一次，二进制数加 1，再计算对应此种结构的指标值，若此时的T_i^*大于已经找到的T_i，则对应于T_i^*的结构优于已经找到的对应于T_i的结构。依次比较下一种结构，直到比较完(2^p-2)种分组，就找到了最好的分组结构。具有$D(I)=1$的所有类放在左边组，其余$D(I)=-1$的所有类放到右边的组。



J记1的个数，T记指标函数值

图 1 决定二叉树结构的流程图

对每一个二进制数所对应的结构，计算指标函数值，记下最大值及相应结构形式。对所有可能的结构“穷举”，找到最好的树结构。

2）选择特征子集（图 2）由上面找到的最好的分组结构D得$y_*=yD$，从而建立

$$L=\begin{vmatrix} L_{xx} & L_{xy*} \\ L_{y*x} & L_{y*y*} \end{vmatrix}$$

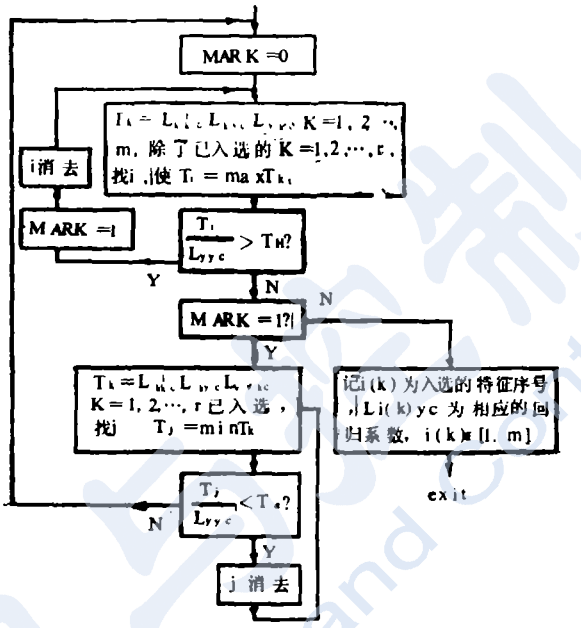


图 2 选择特征子集流程图

T_H，T_d为预先设定的入、出阈值。

开始置MARK为 0，考察$x_1,x_2,\cdots,x_m$是否有可引入变量，若x_i入选，将MARK置 1，同时作 i 消去。若已入选 r 个变量——$x_1,x_2,\cdots,x_r$，考察$x_{r+1},\cdots,x_m$中是否还有可入选变量，若没有，此时MARK为 1，考察在已入选的变量中有否可剔除的变量，若x_j应剔除，则作 j 消去。继续考察，若没有可剔除的变量，再置 MARK 为 零，考察有否可入选变量，此时，若没有可入选变量，由于MARK的控制，也就不再去剔除，记下已入选的特征序号及相应的回归系数，至出口。

3）最后决策

因为已找到了一个有效的特征子集和相应的回归系数$\{L_{i(1)yc},L_{i(2)yc},\cdots,L_{i(r)yc}\}$，于是可写成一个线性判别函数

$$\begin{cases} f = \beta_0 + \sum_{k=1}^{q} L_{i\ x\ y\bar{c}}\, x_{i(k)} \\ \beta_0 = \overline{y} - \sum_{k=1}^{r} L_{i\ k\ yc}\, \overline{x_{i(k)}} \end{cases} \qquad (10)$$

所有具有 $f \geqslant 0$ 的测试样本被放在左边的子树，$f < 0$ 的测试样本被放到右边的子树。

在每个非终极节点重复以上三步，直至二叉树分类器完成。

2．用FORTRAN语言遍历二叉树

定义二叉树为节点的有限集合。该集合或者是空集，或者是由一个根及两个不相交的左子树和右子树的二叉树组成。这个定义是递归的，即是用树来定义树。具有一个节点的树必然仅由根组成，而具有n>1个节点的树则借助于少于 n 个节点的树来定义。

于是就可以用这样的一种方式在计算机内表示二叉树：每个节点有两个链接，即LLINK和 RLINK，以及作为“指向树的指针”的一个链接变量W，如果树是空的（叶子），则$W=\lambda$，否则 W 是树的根节点（或子树的根节点）的地址，而 LLINK(W) 和 RLINK(W)分别是指向左边和右边子树的指针。这些规则递归地定义了任何二叉树形的内存表示。如图 3 的二叉树形由图 4 的内存表示。

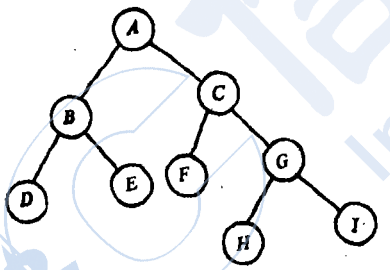


图 3　一种二叉树形

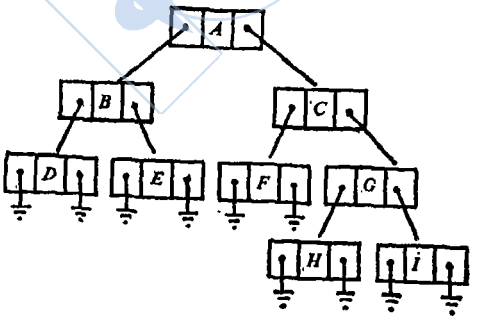


图 4　表示图 3 二叉树形的内存结构

所谓“遍历”或“走遍”一棵树，是系统地检查树的节点，使每个节点恰好被访问一次。遍历一棵二叉树有三种主要方法：先根次序、中根次序和后根次序，这三种方法都是递归定义的。当二叉树为空时，就“遍历”了。否则按表 1 所示的三步进行。

表 1

先根次序的遍历	中根次序的遍历	后根次序的遍历
访　问　根	遍历左子树	遍历左子树
遍历左子树	访　问　根	遍历右子树
遍历右子树	遍历右子树	访　问　根

我们采用了先根次序算法。

实现此算法的计算机内存表示见图 5 。

我们使用的计算机是M68000。用 M68000 的 FOR77 遍历二叉树有许多困难，第一，FORTRAN 不是递归的；第二，也是最主要的，计算机的内存很难满足统计识别的庞大的原始数据所需的存储空间。这是所有搞统计识别研究的同志都会遇到的具体问题，我们探讨并实现了解决此问题的方法，相信它有普遍的参考意义。

二叉树是递归定义的，所以用具有递归调用功能的语言实现解释是很方便的，但 FORTRAN 语言不具备递归调用功能，故难以编制遍历二叉树的解释程序。为此，我们用 FORTRAN语言中最易处理的数据结构——数组构成链接表，再按一定算法规则造一个循环程序跟踪链接表，代替递归解释，以实现二叉树遍历。为克服第二个困难，必须充分利用内存和外存。将当前节点不用的全部数据作为文件存入磁盘，而当前节点要用的全部数据都放在一

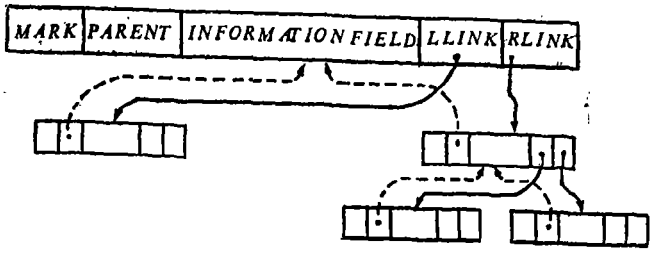


图 5　先根次序算法的计算机内存表示

个数组内，不设工作区，让运算结果覆盖数据区，最大限度地利用内存，下面分别叙述。

1）链接表$DS(I,J)$

表 2

← Information Field →								
M	P_R	L	R	P	N	1	2	… …
4	0	0	0	0	0	1	2	3 …
4	0	0	0	0	0	0	0	0 …
4	0	0	0	0	0	0	0	0 …
4	0	0	0	0	0	0	0	0 …
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

每个节点占链接表（表2）的一行，包括标记M，父亲的节点号PR，左儿子的节点号L，右儿子的节点号R，本节点的原始数据类别数P和样本数N，以及本节点包含的类别序号1,2,…,P(P,N及类别号是信息域的内容)。链接表所需的行数即二叉树的节点总数（包括根和叶子）与类别数的关系是：节点总数$=2P-1$。

首先链接表置初值，M全置或4，根节点包含所有的类，其余为零。算法规则为

①若$M=4$，变1，处理左儿；

②若$M=1$，变2，处理右儿；

③若$M=2$，变3，找父；

④若$P=1$ M变0，送类别序号，找父；

⑤若$M=3$ $PR=0$，则父为根，结束；

⑥在本节点选左、右儿的类别数P及样本数N；

⑦在本节点送左、右儿的类别序号。

2）在处理当前节点时，根据分组结构，将原始数据分成左、右两部分记盘。必须考虑到在处理左、右儿节点时能找到这些文件，为此让这些文件的名字与节点号有关。我们的作法是：定义一个字符型数组$CH(I)$，I为当前节点号。由于这些文件要在多次重入的子程序中写盘，又要在主程序中读盘，为了不破坏文件名字，必须将字符型数组放在FORTRAN的有名公用区中（因无名公用区不能用数据块辅程序赋初值）。如：

```
主程序  CHARACTER*4 CH(11)
        ⋮
        END
        BLOCK DATA
        CHARACTER*4 CH(11)
        COMMON/CH/CH(11)
        DATA CH(1),CH(2),CH(3),
…,CH(11)/'C1','C2',…,'C11'
        END
```

通过数据块辅程序，将文件名C_1,C_2,…,C_{11}分别赋给了字符型数组的元素CH(1),CH(2),…,CH(11)。于是在处理1号节点时，若左儿是2号节点，右儿是3号节点，就会分别以C_2，C_3为文件名，将原始数据分成左，右两枝记盘。当处理2号节点时，$I=2$，以CH(1)去读文件，就读出了所需的C_2文件。

3）充份利用FORTRAN处理可调数组的能力，来尽可能地利用内存。此原则贯穿在我们的程序之中，现仅就一例加以说明。

在划分原始数据为训练集与测试集时，由于要统一考虑到C方法（重复替代法）及循环分组法，存放测试集（或训练集）的数据将与原始数据矩阵一般大，而要在程序中定义偌大的两个数组，将使内存的利用率大大降低。我们的办法是在主程序中定义一个与原始数据矩阵一般大的一维数组，先将原始数据记盘（留作后面“抽”训练集用）；在子程序中定义一个可调数组，将原始数据划分成训练集及测试集两组，先“抽”测试集，边“抽”边覆盖原始数据，在一个可调数组内进行。

除了C方法外，测试集肯定小于原始数据矩阵。在“抽”完测试集，返回到主程序将测试集记盘时，由于主程序的一维数组实体调用时仍按原始数据矩阵的大小，按列存取，故记盘时会夹很多原来的数据。因此须先设法将这些原来的数据“挤掉”，再按测试集的大小记盘（图6）。

我们的作法（图7）是

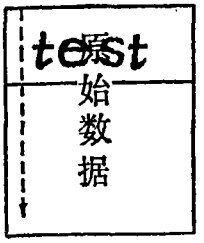


图 6 测试集记盘

```
主程序
      ⋮
   DIMENSION A(56588)
      ⋮
   CALL AAT    ; "抽"测试集
   CALL MOVE   ; "挤"掉多余的数据
   CALL WR     ; test记盘
   SUBROUTINE AAT
      ⋮
   DIMENSION A(658, 80+P)
      ⋮
   END
      SUBROUTINE MOVE
      INTEGER*1  PI
      DIMENSION A(56588)
      DO 10  K=1, 80+PI
      DO 10  I=1, NI
   10 A(K*NI+I) =A (K*658+I)
      RETURN
      END
```

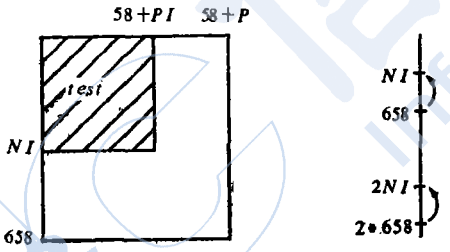


图 7 记盘的具体实现

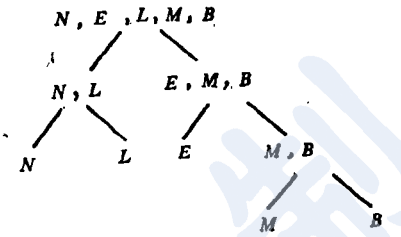
子程序 MOVE 的功能相当于矩阵按列展开，每列"挤"掉了(658-*NI*)个数据。然后子程序WR按〔*NI*×(58+*PI*)〕的大小记盘，这样，数据就完全正确了。

三 结 果

该分类器已投入运行。用一组共658个白血球的原始数据作分类，取得了较好的分类结果。

1．五类（重复替代法）

树结构为



混淆矩阵为

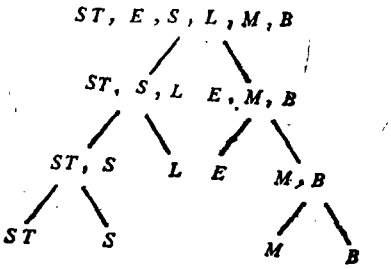
	N	*E*	*L*	*M*	*B*
N(242)	242	1	0	1	1
E(105)	0	99	0	0	5
L(109)	0	0	109	3	1
M(109)	0	0	0	103	1
B(93)	0	5	0	2	85

平均识别率为97%。

2．六类

（1）重复替代法

树结构为



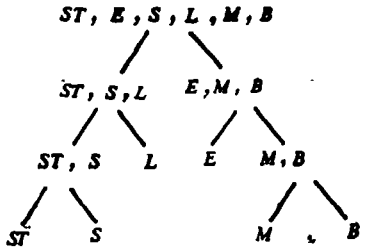
混淆矩阵为

	ST	*E*	*S*	*L*	*M*	*B*
ST(125)	102	1	17	0	0	0
E(105)	0	102	0	0	0	2
S(117)	23	0	100	0	0	1
L(109)	0	0	0	109	3	1
M(109)	0	0	0	0	105	0
B(93)	0	2	0	0	1	89

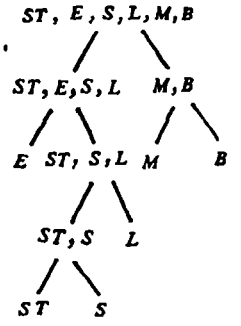
平均识别率为92.2%。

（2）一半对一半

前一半树结构为



后一半树结构为



混淆阵为

	ST	E	S	L	M	B
ST	73	2	38	0	1	0
E	1	77	1	1	2	22
S	50	1	75	0	0	1
L	0	0	2	106	5	3
M	0	5	0	0	99	4
B	0	19	0	1	1	62

平均识别率为75.7%。

四 结 论

1．该分类器由于在树结构及特征选择中都采用了穷举法，设计合理、结构优化，故对白血球的分类结果，与国内外发表的结果相比，是令人满意的。

2．分类器是通用的，使用方便。只要用交互方式输入类别数，每类原始数据的个数及文件名，便可运行程序得出分类结果。用此分类器对沈阳陆军总院的白血病前后期的病例进行过分类，得到了较好的结果。

全部白血球图象是由中国医科大学第三附属医院血液研究室协助输入的。细胞图象的区域分割、特征提取及数据整理等程序都是朱东、南枫、曲玉华、刘白桦、黄莎白合作完成的。姚筱亦同志仔细审阅了全文并提出了宝贵意见，在此表示感谢。

参 考 文 献

〔1〕 Shi Qingyun and Fu, K.S., A Method for the Design of Binary Tree Classifiers, Pattern Recognition, Vol.16, No.6, 1983, pp.393—603.

〔2〕 Robert J.Baron and Linda G.Shapiro York, Data Structures and Their Implemetation, Van Nostrand Reinhold, 1980.

〔3〕 D.E.克努特著，管纪文、苏运霖译，计算机程序设计技巧，第一卷：基本算法。

〔4〕 Jack, K. Mui and Fu, K.S., Automated Classification of Nucleated Blood Cells Using a Binary Tree Classifier, IEEE Transactions on Pattern Analysis and Machine Intelligence Vol.PAMI-2, No.5, 1980.

ABSTRACTS

Optimization Design Multivariable PID Frequency Controller Domain Parameter

Wang Qingguo et al.

In this paper the concept of target transfer function array is presented. By using this concept expected performance index is expressed. And then a multivariable PID control frequency domain design technique is introduced by parameter optimization approach. In a closed loop system designed by this technique, the frequency responce is almost same as the target array, the output transient performance is quite satisfactory, and no steady state error. This technique is used in paper-making process for the weight and moisture control system design and an expected result is given. (p.1)

The Design and Implementation of a Binary Tree Classifier Based on Multivariate Step-wise Regression

Huang Shubei Liu Beihua

A binary tree classifier based on multivariate step-wise regression was designed and implemented. The "exhaustion" method was applied in selecting both the tree structure and the feature subsets to get more reasonable and optimal result than that done by selection with constrained condition.

The "traversal" of binary tree was implemented with FORTRAN Language. Computer memory can be used sufficiently by both the advantage of FORTRAN for processing adjustable array and the programming skill.

The classifier is universal and can be used in any pattern classification in which the statistical data is available. For its application to white blood cell usage the recognition rate is as high as 97% in five classes and 92.2% in six classes. (p.6)

Realization of Optimizing Both Parameters and Structure in Model Building Using Rational Approximation

Ji Xinbo Zhang Zhiyi

On purpose to apply rational function for mathematical model building, this article proposes a practical relaxation algorithm to optimize both parameters and structure