第3章

判别函数分类器设计

3.1 判别函数简介

判别函数是统计模式识别中用以对模式进行分类的一种较简单的函数。在特征空间中，通过学习，不同的类别可以得到不同的判别函数，比较不同类别的判别函数值的大小，就可以进行分类。统计模式识别方法把特征空间划分为决策区对模式进行分类，一个模式类同一个或几个决策区相对应。

设有 c 个类别，对于每一个类别 $\omega_i(i=1,2,\cdots,c)$ 定义一个关于特征向量 $\boldsymbol{X}$ 的单值函数 $g_i(\boldsymbol{X})$：①如果 $\boldsymbol{X}$ 属于第 i 类，那么 $g_i(\boldsymbol{X})>g_j(\boldsymbol{X})(i,j=1,2,\cdots,c,j\neq i)$；②如果 $\boldsymbol{X}$ 在第 i 类和第 j 类的分界面上，那么 $g_i(\boldsymbol{X})=g_j(\boldsymbol{X})(i,j=1,2,\cdots,c,j\neq i)$。

人们已研究出多种求取决策边界的算法，线性判别函数的决策边界是一个超平面方程式，其中的系数可以从已知类别的学习样本集求得。F. 罗森布拉特的错误修正训练程序是求取两类线性可分分类器决策边界的早期方法之一。在用线性判别函数不可能对所有学习样本正确分类的情况下，可以规定一个准则函数（例如对学习样本的错分数最少）并用使准则函数达到最优的算法求取决策边界。用线性判别函数的模式分类器也称为线性分类器或线性机，这种分类器计算简单，不要求估计特征向量的类条件概率密度，是一种非参数分类方法。

当用贝叶斯决策理论进行分类器设计时，在一定的假设下也可以得到线性判别函数，这无论对于线性可分或线性不可分的情况都是适用的。在问题比较复杂的情况下可以用多段线性判别函数（见近邻法分类、最小距离分类）或多项式判别函数对模式进行分类。一个二阶的多项式判别函数可以表示为与它相应的决策边界是一个超二次曲面。

本章介绍线性判别函数和非线性判别函数，用以对酒瓶的颜色进行分类，其中实现线性判别函数分类的方法有 LMSE 分类算法和 Fisher 分类，实现非线性判别函数分类的方法有基于核的 Fisher 分类和支持向量机。

3.2 线性判别函数

判别函数分为线性判别函数和非线性判别函数。最简单的判别函数是线性判别函数，它是由所有特征量的线性组合构成的。我们现在对两类问题和多类问题分别进行讨论。

1. 两类问题

对于两类问题，也就是 $\boldsymbol{W}_i=(\omega_1,\omega_2)^{\mathrm{T}}$。

1）二维情况

取二维特征向量 $\boldsymbol{X}=(x_1,x_2)^{\mathrm{T}}$，这种情况下的判别函数 $g(x)=\omega_1x_1+\omega_2x_2+\omega_3$，其中，$\omega_i(i=1,2,3)$为参数；$x_1$ 和 x_2 为坐标值，判别函数 $g(x)$具有以下性质：当 $x\in\omega_1$ 时，$g_i(x)>0$；当 $x\in\omega_2$ 时，$g_i(x)<0$；当 x 不定时，$g_i(x)=0$。这是二维情况下由判别边界分类。

2）n 维情况

对于 n 维情况，现抽取 n 维特征向量：$\boldsymbol{X}=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$，判别函数为 $g(x)=\boldsymbol{W}_0\boldsymbol{X}+\omega_{n+1}$。其中，$\boldsymbol{W}_0=(\omega_1,\omega_2,\cdots,\omega_n)^{\mathrm{T}}$ 为权向量；$\boldsymbol{X}=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$ 为模式向量。另外一种表示方法是 $g(x)=\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}$。其中，$\boldsymbol{W}=(\omega_1,\omega_2,\cdots,\omega_n,\omega_{n+1})^{\mathrm{T}}$ 为增值权向量；$\boldsymbol{X}=(x_1,x_2,\cdots,x_n,1)^{\mathrm{T}}$ 为增值模式向量。

在这种情况下，当 $x\in\omega_1$ 时，$g(x)>0$；当 $x\in\omega_2$ 时，$g(x)<0$；$g_1(x)=0$ 为边界，当 $n=2$ 时，边界为一条直线，当 $n=3$ 时，边界为一个平面，当 $n>3$ 时，边界为超平面。

2. 多类问题

对于多类问题，模式有 $\omega_1,\omega_2,\cdots,\omega_M$ 个类别，可以分为下面三种情况。

1）第一种情况

每个模式类与其他模式可用单个判别平面分开，这时 M 个类有 M 个判别函数，且具有性质

$$g_i(x)=\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{X} \tag{3-1}$$

式中，$\boldsymbol{W}_i=(\omega_{i1},\omega_{i2},\cdots,\omega_{in+1})^{\mathrm{T}}$ 为第 i 个判别函数的权向量。当 $x\in\omega_i$ 时，$g_i(x)>0$，其他情况下 $g_i(x)<0$，也就是每一个类别可以用单个判别边界与其他类别相分开。

2）第二种情况

每个模式类和其他模式类之间可以用判别平面分开，这样就有 $\dfrac{M(M-1)}{2}$ 个平面，对于两类问题，$M=2$，则有 1 个判别平面，同理对于三类问题，就有 3 个判别平面。判别函数为

$$g_{ij}(x)=\boldsymbol{W}_{ij}^{\mathrm{T}}\boldsymbol{X} \tag{3-2}$$

式中，$i\neq j$，判别边界为 $g_{ij}(x)=0$，条件为：当 $x\in\omega_i$ 时，$g_{ij}(x)>0$；当 $x\in\omega_j$ 时，$g_{ij}(x)<0$。

3）第三种情况

每类都有一个判别函数，存在 M 个判别函数：$g_k(x)=\boldsymbol{W}_k\boldsymbol{X}(k=1,2,\cdots,M)$，边界为 $g_i(x)=g_j(x)$，条件为：当 $x\in\omega_i$ 时，$g_i(x)$最大；其他情况下 $g_i(x)$小。也就是说，要判别 $\boldsymbol{X}$ 属于哪一个类，先把 $\boldsymbol{X}$ 代入 M 个判别函数，判别函数最大的那个类就是 $\boldsymbol{X}$ 所属类别。

3.3 线性判别函数的实现

对于给定的样本集 $\boldsymbol{X}$,要确定线性判别函数 $g(x)=\boldsymbol{W}^{\mathrm{T}}x+\omega_0$ 的各项系数 $\boldsymbol{W}$ 和 ω_0,可以通过以下步骤来实现:

① 收集一组具有类别标志的样本 $\boldsymbol{X}=\{x_1,x_2,\cdots,x_N\}$;

② 按照需要确定准则函数 J;

③ 用最优化技术求准则函数 J 的极值解 ω^* 和 ω_0^*,从而确定判别函数,完成分类器的设计。

对于未知样本 x,计算 $g(x)$,判断其类别。即对于一个线性判别函数,主要任务是确定线性方程的两个参数,一个是权向量 $\boldsymbol{W}$,另一个是阈值 ω_0。

在计算机中想要实现线性判别函数,可以通过"训练"和"学习"的方式,将已知样本放入到计算机的"训练"程序,经过多次迭代,从而得到准确函数。

下面具体介绍各种分类器的设计。

3.4 基于 LMSE 的分类器设计

3.4.1 LMSE 分类法简介

LMSE 是 Least Mean Square Error 的英文缩写,中文的意思是最小均方误差,常称作 LMSE 算法。

提到 LMSE 分类算法就不能不提感知器算法和自适应算法,因为 LMSE 算法本身就是自适应算法中最常用的方法,而感知器和自适应线性元件在历史上几乎是同时提出的,并且两者在对权值的调整的算法非常相似,它们都是基于纠错学习规则的学习算法。感知器算法存在如下问题:不能推广到一般的前向网络中;函数不是线性可分时,得不出任何结果。而由美国斯坦福大学的 Widrow-Hoff 在研究自适应理论时提出的 LMSE 算法,由于其易实现因而很快得到了广泛应用,成为自适应滤波的标准算法。下面介绍自适应过程。

自适应过程是一个不断逼近目标的过程。它所遵循的途径以数学模型表示,称为自适应算法。通常采用基于梯度的算法,其中 LMSE 算法尤为常用。自适应算法可以用硬件(处理电路)或软件(程序控制)两种办法实现。前者依据算法的数学模型设计电路,后者则将算法的数学模型编制成程序并用计算机实现。算法有很多种,选择算法很重要,它决定了处理系统的性能质量和可行性。

自适应均衡器的原理就是按照某种准则和算法对其系数进行调整,最终使自适应均衡器的代价(目标)函数最小化,达到最佳均衡的目的。而各种调整系数的算法就称为自适应算法,自适应算法是根据某个最优准则来设计的。最常用的自适应算法有逼零算法、最陡下降算法、LMSE 算法、RLS 算法以及各种盲均衡算法等。

自适应算法所采用的最优准则有最小均方误差准则、最小二乘准则、最大信噪比准则和统计检测准则等，其中最小均方误差准则和最小二乘准则是目前最为流行的自适应算法准则。LMSE 算法和 RLS 算法由于采用的最优准则不同，因此这两种算法在性能、复杂度等方面均有许多差别。

一种算法性能的好坏可以通过几个常用的指标来衡量，例如收敛速度——通常用算法达到稳定状态（即与最优值的接近程度达到一定值）的迭代次数表示；误调比——实际均方误差相对于算法的最小均方误差的平均偏差；运算复杂度——完成一次完整迭代所需的运算次数；跟踪性能——对信道时变统计特性的自适应能力。

3.4.2 LMSE 算法原理

LMSE 算法是针对准则函数引进最小均方误差这一条件而建立起来的。这种算法的主要特点是在训练过程中判定训练集是否线性可分，从而可对结果的收敛性做出判断。

LMSE 算法属于监督学习的类型，而且是“模型无关”的，它是通过最小化输出和期望目标值之间的偏差来实现的。

LMSE 算法属于自适应算法中常用的算法，它不同于 C 均值算法和 ISODATA 算法，后两种属于基于距离度量的算法，直观且容易理解。LMSE 算法通过调整权值函数求出判别函数，进而将待测样本代入判别函数求值，最终做出判定，得出答案。

1. 准则函数

LMSE 算法以最小均方差作为准则，因均方差为

$$E\{[r_i(\boldsymbol{X})-\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{X}]^2\} \tag{3-3}$$

因而准则函数为

$$J(\boldsymbol{W}_i,\boldsymbol{X})=\frac{1}{2}E\{[r_i(\boldsymbol{X})-\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{X}]^2\} \tag{3-4}$$

准则函数在 $r_i(\boldsymbol{X})-\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{X}=0$ 时取得最小值。准则函数对 $\boldsymbol{W}_i$ 的偏导数为

$$\frac{\partial J}{\partial \boldsymbol{W}_i}=E\{-\boldsymbol{X}[r_i(\boldsymbol{X})-\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{X}]\} \tag{3-5}$$

2. 迭代方程

将式(3-5)代入迭代方程，得到

$$\boldsymbol{W}_i(k+1)=\boldsymbol{W}_i(k)+\alpha_k\boldsymbol{X}(k)[r_i(\boldsymbol{X})-\boldsymbol{W}_i^{\mathrm{T}}(k)\boldsymbol{X}(k)] \tag{3-6}$$

对于多类问题来说，M 类问题应该有 M 个权函数方程，而对于每一个权函数方程来说，如 $\boldsymbol{X}(k)\in\omega_i$，则

$$r_i[\boldsymbol{X}(k)]=1,\quad i=1,2,\cdots,M \tag{3-7}$$

否则

$$r_i[\boldsymbol{X}(k)]=0,\quad i=1,2,\cdots,M \tag{3-8}$$

3.4.3 LMSE 算法步骤

(1) 设各个权向量的初始值为 0，即 $\boldsymbol{W}_0(0)=\boldsymbol{W}_1(0)=\boldsymbol{W}_2(0)=\cdots=\boldsymbol{W}_M(0)=0$。

(2) 输入第 k 次样本 $\boldsymbol{X}(k)$,计算 $d_i(k)=\boldsymbol{W}_i^{\mathrm{T}}(k)\boldsymbol{X}(k)$。

(3) 确定期望输出函数值:若 $\boldsymbol{X}(k)\in\omega_i$,则 $r_i[\boldsymbol{X}(k)]=1$,否则 $r_i[\boldsymbol{X}(k)]=0$。

(4) 计算迭代方程:$\boldsymbol{W}_i(k+1)=\boldsymbol{W}_i(k)+\alpha_k\boldsymbol{X}(k)[r_i(\boldsymbol{X})-\boldsymbol{W}_i^{\mathrm{T}}(k)\boldsymbol{X}(k)]$,其中 $\alpha_k=\dfrac{1}{k}$。

(5) 循环执行步骤(2),直到满足条件:属于 ω_i 类的所有样本都满足不等式 $d_i(\boldsymbol{X})>d_j(\boldsymbol{X})$,$\forall j\neq i$。

3.4.4 LMSE 算法的 MATLAB 实现

1. 首先给定四类样本,各样本的特征向量经过增 1

程序如下:

```
pattern = struct('feature',[])
p1 = [864.45   1647.31  2665.9;
      877.88   2031.66  3071.18;
      1418.79  1775.89  2772.9;
      1449.58  1641.58  3405.12;
      864.45   1647.31  2665.9;
      877.88   2031.66  3071.18;
      1418.79  1775.89  2772.9;
      1449.58  1641.58  3405.12;
      1418.79  1775.89  2772.9;
      1449.58  1641.58  3405.12;]
pattern(1).feature = p1'
pattern(1).feature(4,:) = 1
```

pattern(1).feature 实际的矩阵形式如下:

```
p1 =
  1.0e+03 *
    0.8645    1.6473    2.6659
    0.8779    2.0317    3.0712
    1.4188    1.7759    2.7729
    1.4496    1.6416    3.4051
    0.8645    1.6473    2.6659
    0.8779    2.0317    3.0712
    1.4188    1.7759    2.7729
    1.4496    1.6416    3.4051
    1.4188    1.7759    2.7729
    1.4496    1.6416    3.4051
```

之后的三类,程序如下:

```
p2 = [2352.12  2557.04  1411.53;
2297.28  3340.14  535.62;
2092.62  3177.21  584.32;
2205.36  3243.74  1202.69;
2949.16  3244.44  662.42;
2802.88  3017.11  1984.98;
2063.54  3199.76  1257.21;
```

```
2949.16  3244.44  662.42;
2802.88  3017.11  1984.98;
2063.54  3199.76  1257.21;]
pattern(2).feature = p2'
pattern(2).feature(4,:) = 1
p3 = [1739.94  1675.15  2395.96;
1756.77  1652  1514.98;
1803.58  1583.12  2163.05;
1571.17  1731.04  1735.33;
1845.59  1918.81  2226.49;
1692.62  1867.5  2108.97;
1680.67  1575.78  1725.1;
1651.52  1713.28  1570.38;
1680.67  1575.78  1725.1;
1651.52  1713.28  1570.38;]
pattern(3).feature = p3'
pattern(3).feature(4,:) = 1
p4 = [373.3  3087.05  2429.47;
222.85  3059.54  2002.33;
401.3  3259.94  2150.98;
363.34  3477.95  2462.86;
104.8  3389.83  2421.83;
499.85  3305.75  2196.22;
172.78  3084.49  2328.65;
341.59  3076.62  2438.63;
291.02  3095.68  2088.95;
237.63  3077.78  2251.96;]
pattern(4).feature = p4'
pattern(4).feature(4,:) = 1
```

2. 设权值向量的初始值均为 0

初始化权值程序代码如下：

```
w = zeros(4,4);   % 初始化权值
```

MATLAB 程序运行结果如下：

```
w =
    0    0    0    0
    0    0    0    0
    0    0    0    0
    0    0    0    0
```

3. 计算 di(k)

程序代码如下：

```
for k = 1:4
                m = pattern(i).feature(:,j)
                m = m/norm(m)
                d(k) = w(:,k)' * m      % 计算 d
```

MATLAB 程序运行结果如下。

第一次循环结果：

```
m =
  1.0e+03 *
    0.8645
    1.6473
    2.6659
    0.0010
m =
    0.2659
    0.5067
    0.8201
    0.0003
d =
    0
m =
  1.0e+03 *
    0.8645
    1.6473
    2.6659
    0.0010
m =
    0.2659
    0.5067
    0.8201
    0.0003
d =
    0    0
m =
  1.0e+03 *
    0.8645
    1.6473
    2.6659
    0.0010
m =
    0.2659
    0.5067
    0.8201
    0.0003
d =
    0    0    0
m =
    1.0e+03 *
    0.8645
    1.6473
    2.6659
    0.0010
m =
    0.2659
    0.5067
    0.8201
    0.0003
d =
    0    0    0    0
```

最后一次运行结果：

```
m =
  1.0e+03 *
    0.2376
    3.0778
    2.2520
    0.0010
m =
    0.0622
    0.8055
    0.5894
    0.0003
d =
    0.2745    0.3263    0.3348    0.1432
m =
  1.0e+03 *
    0.2376
    3.0778
    2.2520
    0.0010
m =
    0.0622
    0.8055
    0.5894
    0.0003
d =
    0.2745    0.1397    0.3348    0.1432
m =
  1.0e+03 *
    0.2376
    3.0778
    2.2520
    0.0010
m =
    0.0622
    0.8055
    0.5894
    0.0003
d =
    0.2745    0.1397    0.1217    0.1432
m =
  1.0e+03 *
    0.2376
    3.0778
    2.2520
    0.0010
m =
    0.0622
    0.8055
    0.5894
    0.0003
d =
    0.2745    0.1397    0.1217    0.4591
```

4. 调整权值

程序代码如下：

```
for k = 1:4
                if k~ = i
                    if d(i)< = d(k)        % d(i)不是最大,则继续迭代
                        flag = 1;
                    end
                end
            end
            % 调整权值
            for k = 1:4
                w(:,k) = w(:,k) + m * (r(k) - d(k))/num
```

MATLAB 程序运行结果如下：

```
w =
   - 0.1204    0.4917    0.4122   - 0.4088
   - 0.1820    0.3990   - 0.0421    0.4791
     0.7271   - 0.3601    0.2204    0.1674
     0.0001    0.0000    0.0002    0.0000
```

5. 通过判别函数将待分类数据分类

调用 function 函数，将待测数据分类。因为调用该函数一次只能判别一个样品的类别，所以循环 30 次才能完成分类，程序代码如下：

```
for k = 1:30
    sample = sampletotall(:,k)
    y = lmseclassify(sample)
    x = sample(1)
    yy = sample(2)
    z = sample(3)
    ac(k) = y
```

运行 MATLAB 程序，最终的分类结果如下：

```
ac =
1 ~ 15 列
3    3    1    3    4    2    2    3    4    1    3    2    1    2
4
16 ~ 30 列
2    4    3    4    2    2    1    3    1    1    4    1    3    3
3
```

将该分类结果与原始分类结果对比，对照表如表 3-1 所示。

表 3-1 LMSE 分类结果与原始分类结果对照表

序 号	A	B	C	原始分类结果	LMSE 分类结果
1	1702.8	1639.79	2068.74	3	3
2	1877.93	1860.96	1975.3	3	3
3	867.81	2334.68	2535.1	1	1
4	1831.49	1713.11	1604.68	3	3
5	460.69	3274.77	2172.99	4	4
6	2374.98	3346.98	975.31	2	2
7	2271.89	3482.97	946.7	2	2
8	1783.64	1597.99	2261.31	3	3
9	198.83	3250.45	2445.08	4	4
10	1494.63	2072.59	2550.51	1	1
11	1597.03	1921.52	2126.76	3	3
12	1598.93	1921.08	1623.33	3	2
13	1243.13	1814.07	3441.07	1	1
14	2336.31	2640.26	1599.63	2	2
15	354	3300.12	2373.61	4	4
16	2144.47	2501.62	591.51	2	2
17	426.31	3105.29	2057.8	4	4
18	1507.13	1556.89	1954.51	3	3
19	343.07	3271.72	2036.94	4	4
20	2201.94	3196.22	935.53	2	2
21	2232.43	3077.87	1298.87	2	2
22	1580.1	1752.07	2463.04	3	1
23	1962.4	1594.97	1835.95	3	3
24	1495.18	1957.44	3498.02	1	1
25	1125.17	1594.39	2937.73	1	1
26	24.22	3447.31	2145.01	4	4
27	1269.07	1910.72	2701.97	1	1
28	1802.07	1725.81	1966.35	3	3
29	1817.36	1927.4	2328.79	3	3
30	1860.45	1782.88	1875.13	3	3

结果分析：从表 3-1 中可以看出有 2 个分类结果是错的，正确率为 93.3%。

6. 用三维效果图将结果直观显示

将分好类的数据用三维图像的形式直观显示出来，程序代码如下：

```
axis([0 3500 0 3500 0 3500])
 if y == 1
          plot3(x,yy,z,'g * ');        % 第一类表示为绿色
    elseif y == 2
          plot3(x,yy,z,'r * ')         % 第二类表示为红色
    elseif y == 3
```

```
            plot3(x,yy,z,'b*')          %第三类表示为蓝色
        elseif y==4
            plot3(x,yy,z,'y*')          %第四类表示为黄色
end
hold on
```

运行 MATLAB 程序,三维效果图如图 3-1 所示。

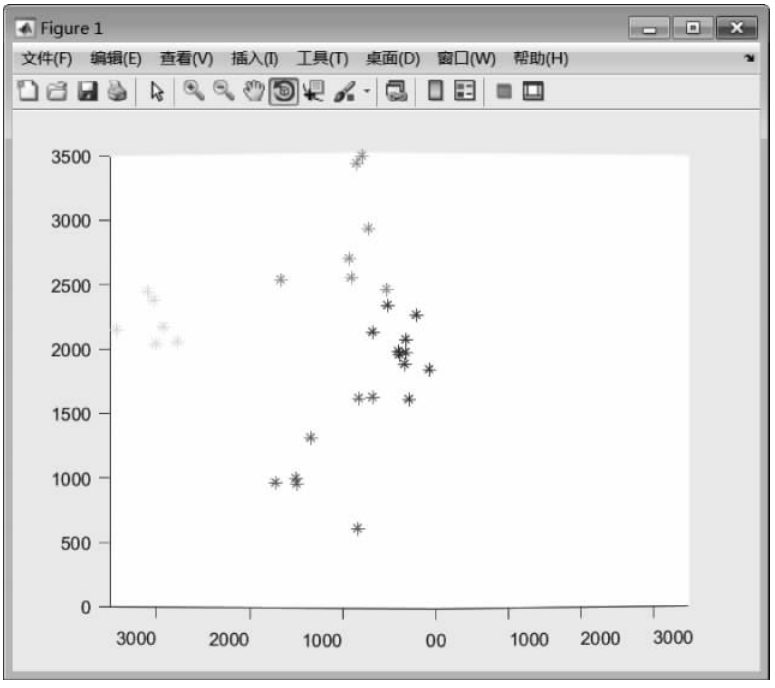

图 3-1 LMSE 算法分类结果三维效果图

7. 完整的 MATLAB 程序

完整的 MATLAB 程序代码如下:

```
%主程序
%输入数据
clear all
close all
clc
s1 = xlsread('C:\Users\Administrator\Desktop\ln.xls')    %从 Excel 表格直接读入数据
samp = s1(30:59,1:3)                                     %后 30 个数据
sampletotall = samp'                                     %转置
ac = [ ]                                                 %定义分类矩阵
for k = 1:30
    sample = sampletotall(:,k)
    y = lmseclassify(sample)                             %调用 function 函数
    x = sample(1)
```

```
    yy = sample(2)
    z = sample(3)
    ac(k) = y
axis([0 3500 0 3500 0 3500])
    if y == 1
        plot3(x,yy,z,'g * ')            % 第一类用绿色表示
    elseif y == 2
        plot3(x,yy,z,'r * ')            % 第二类用红色表示
    elseif y == 3
        plot3(x,yy,z,'b * ')            % 第三类用蓝色表示
    elseif y == 4
        plot3(x,yy,z,'y * ')            % 第四类用黄色表示
    end
    hold on
end

% 利用 LMSE 算法进行分类的 function 函数
function y = lmseclassify(sample)
 clc;
 pattern = struct('feature',[])
p1 = [
    864.45    1647.31  2665.9;
    877.88    2031.66  3071.18;
    1418.79   1775.89  2772.9;
    1449.58   1641.58  3405.12;
    864.45    1647.31  2665.9;
    877.88    2031.66  3071.18;
    1418.79   1775.89  2772.9;
    1449.58   1641.58  3405.12;
    1418.79   1775.89  2772.9;
    1449.58   1641.58  3405.12;]
pattern(1).feature = p1'
pattern(1).feature(4,:) = 1
p2 = [2352.12  2557.04  1411.53;
     2297.28  3340.14  535.62;
     2092.62  3177.21  584.32;
     2205.36  3243.74  1202.69;
     2949.16  3244.44  662.42;
     2802.88  3017.11  1984.98;
     2063.54  3199.76  1257.21;
     2949.16  3244.44  662.42;
     2802.88  3017.11  1984.98;
     2063.54  3199.76  1257.21;]
pattern(2).feature = p2'
pattern(2).feature(4,:) = 1
p3 = [1739.94  1675.15  2395.96;
     1756.77  1652     1514.98;
     1803.58  1583.12  2163.05;
     1571.17  1731.04  1735.33;
     1845.59  1918.81  2226.49;
```

```
       1692.62  1867.5   2108.97;
       1680.67  1575.78  1725.1;
       1651.52  1713.28  1570.38;
       1680.67  1575.78  1725.1;
       1651.52  1713.28  1570.38;]
pattern(3).feature = p3'
pattern(3).feature(4,:) = 1
p4 = [373.3   3087.05  2429.47;
     222.85  3059.54  2002.33;
     401.3   3259.94  2150.98;
     363.34  3477.95  2462.86;
     104.8   3389.83  2421.83;
     499.85  3305.75  2196.22;
     172.78  3084.49  2328.65;
     341.59  3076.62  2438.63;
     291.02  3095.68  2088.95;
     237.63  3077.78  2251.96;]
pattern(4).feature = p4'
pattern(4).feature(4,:) = 1

w = zeros(4,4)       % 初始化权值
 flag = 1;
 num = 0;
 num1 = 0;
 d = [ ]
 m = [ ]
 r = [ ];
s = xlsread('C:\Users\Administrator\Desktop\ln.xls')
 while flag
     flag = 0;
     num1 = num1 + 1
     for j = 1:10
          for i = 1:4
               num = num + 1;
               r = [0 0 0 0];
               r(i) = 1;
               for k = 1:4
                    m = pattern(i).feature(:,j)
                    m = m/norm(m)
                    d(k) = w(:,k)' * m;        % 计算 d
               end
               for k = 1:4
                    if k~ = i
                         if d(i)< = d(k)     % d(i)不是最大,则继续迭代
                              flag = 1;
                         end
                    end
               end
               % 调整权值
               num;
```

```
                num1;
                for k = 1:4
                    w(:,k) = w(:,k) + m * (r(k) - d(k))/num;
                end
            end
        end
        if num1 > 200                                  % 超过迭代次数则退出
            flag = 0;
        end
    end
    sample(4) = 1
     h = [];
    for k = 1:4
         h(k) = w(:,k)' * sample;                      % 计算判别函数
    end
    [maxval,maxpos] = max(h);
    y = maxpos;
```

3.4.5　结论

学习样本的维数问题：

因为样本类别不均匀(第一类 4 个样本,第二类 8 个样本,第三类 9 个样本,第四类 10 个样本),程序不运行,后来将数据重复添加进去,保证了程序的正常运行。程序相关部分代码如下：

```
        for j = 1:10
                ……
                  for k = 1:4
                      m = pattern(i).feature(:,j)
                      d(k) = w(:,k)' * m      % 计算 d
                  end
                ……
        end
p1 = [864.45  1647.31  2665.9;877.88  2031.66  3071.18;1418.79  1775.89  2772.9;1449.58
     1641.58  3405.12;864.45  1647.31  2665.9;877.88  2031.66  3071.18;1418.79  1775.89
     2772.9;1449.58  1641.58  3405.12;1418.79  1775.89  2772.9;1449.58  1641.58
     3405.12;]
```

注意：其中 864.45　1647.31　2665.9；877.88　2031.66　3071.18；1418.79　1775.89　2772.9；1449.58　1641.58　3405.12；1418.79　1775.89　2772.9；1449.58　1641.58　3405.12 是重复添加的样本数据,目的是凑够 10 个数据以便程序能进行循环。

3.5 基于 Fisher 的分类器设计

3.5.1 Fisher 判别法简介

Fisher 判别法是 1936 年由 R. A. Fisher 首先提出的。Fisher 判别法是一种线性判别法,线性判别又称线性准则。与线性准则相对应的还有非线性准则,其中一些在变换条件下可以转化为线性准则,因此对应于 d 维特征空间,线性判别函数虽然最简单,但是在应用上却具有普遍意义,便于对分类问题的理解与描述。

基于线性判别函数的线性分类方法,虽然使用有限样本集合来构造,从严格意义上来讲属于统计分类方法。也就是说,对于线性分类器的检验,应建立在样本扩充的条件下,以基于概率的尺度来评价才是有效的评价。尽管线性分类器的设计在满足统计学的评价下并不严格与完美,但是由于其具有的简单性与实用性,在分类器设计中还是获得了广泛的应用。

3.5.2 Fisher 判别法的原理

设计线性分类器首先要确定准则函数,然后再利用训练样本集确定该分类器的参数,以求使所确定的准则达到最佳。在使用线性分类器时,样本的分类由其判别函数值决定,而每个样本的判别函数值是其各分量的线性加权和再加上一个阈值 y_0。

如果只考虑各分量的线性加权和,则它是各样本向量与向量 $\boldsymbol{w}$ 的向量点积。如果向量 $\boldsymbol{w}$ 的幅度为单位长度,则线性加权和又可被看作各样本向量在向量 $\boldsymbol{w}$ 上的投影。

Fisher 判别法的基本原理是,对于 d 维空间的样本,投影到一维坐标上,样本特征将混杂在一起,难以区分。Fisher 判别法的目的,就是要找到一个最合适的投影轴 $\boldsymbol{w}$,使两类样本在该轴上投影的交叠部分最少,从而使分类效果为最佳。如何寻找一个投影方向,使得样本集合在该投影方向上最易区分,这就是 Fisher 判别法所要解决的问题。Fisher 投影原理如图 3-2 所示。

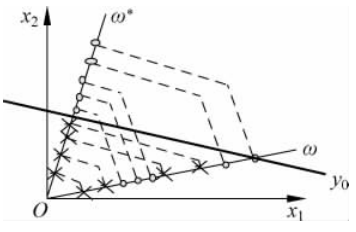

图 3-2 Fisher 投影原理图

Fisher 准则函数的基本思路:即向量 $\boldsymbol{w}$ 的方向选择应能使两类样本投影的均值之差尽可能大些,而使类内样本的离散程度尽可能小。

3.5.3 Fisher 分类器设计

已知 N 个 d 维样本数据集合 $\chi=\{x_1,x_2,\cdots,x_N\}$,其中类别为 $\omega_i(i=1,2)$,样本容量为 N_i,其子集为 x_i,以投影坐标向量 $\boldsymbol{w}$ 与原特征向量 $\boldsymbol{x}$ 作数量积,可得投影表达式为 $y_n=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n(n=1,2,\cdots,N)$。

相应地,y_n 也为两个子集 y_1 和 y_2。如果只考虑投影向量 $\boldsymbol{w}$ 的方向,不考虑其长度,即

默认其长度为单位1，则 y_n 即为 $\boldsymbol{x}_n$ 在 $\boldsymbol{w}$ 方向上的投影。Fisher 准则的目的就是寻找最优投影方向，使得 $\boldsymbol{w}$ 为最好的投影向量 $\boldsymbol{w}^*$。

样本在 d 维特征空间的一些描述量如下。

(1) 各类样本均值向量 $\boldsymbol{m}_i$

$$\boldsymbol{m}_i = \frac{1}{N_i}\sum_{j=1}^{N_i}\boldsymbol{x}_j, \quad i = 1,2 \tag{3-9}$$

(2) 样本类内离散度矩阵 $\boldsymbol{S}_i$ 与总类内离散度矩阵 $\boldsymbol{S}_w$

$$\boldsymbol{S}_i = \sum_{j=1}^{N_i}(\boldsymbol{x}_j - \boldsymbol{m}_i)(\boldsymbol{x}_j - \boldsymbol{m}_i)^{\mathrm{T}}, \quad i = 1,2 \tag{3-10}$$

$$\boldsymbol{S}_w = \boldsymbol{S}_1 + \boldsymbol{S}_2 \tag{3-11}$$

(3) 样本类间离散度矩阵 $\boldsymbol{S}_{\mathrm{b}}$

$$\boldsymbol{S}_{\mathrm{b}} = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^{\mathrm{T}} \tag{3-12}$$

如果在一维上投影，则有各类样本均值向量 $\bar{\boldsymbol{m}}_i$

$$\bar{\boldsymbol{m}}_i = \frac{i}{N_i}\sum_{j=1}^{N_i} y_j, \quad i = 1,2 \tag{3-13}$$

样本类内离散度矩阵 $\bar{\boldsymbol{S}}_i$ 与总类内离散度矩阵 $\bar{\boldsymbol{S}}_w$

$$\bar{\boldsymbol{S}}_i = \sum_{j=1}^{N_i}(y_j - \bar{\boldsymbol{m}}_i)^2, \quad i = 1,2 \tag{3-14}$$

$$\bar{\boldsymbol{S}}_w = \bar{\boldsymbol{S}}_1 + \bar{\boldsymbol{S}}_2 \tag{3-15}$$

Fisher 准则函数的定义原则为，希望投影后，在一维空间中样本类别区分清晰，即两类样本的距离越大越好，也就是均值之差 $(\bar{\boldsymbol{m}}_1 - \bar{\boldsymbol{m}}_2)$ 越大越好；各类样本内部密集，即类内离散度 $\bar{\boldsymbol{S}}_w = \bar{\boldsymbol{S}}_1 + \bar{\boldsymbol{S}}_2$ 越小越好。根据上述两条原则，构造 Fisher 准则函数

$$J_{\mathrm{F}}(\boldsymbol{w}) = \frac{(\bar{\boldsymbol{m}}_1 - \bar{\boldsymbol{m}}_2)^2}{\bar{\boldsymbol{S}}_1 + \bar{\boldsymbol{S}}_2} \tag{3-16}$$

使得 $J_{\mathrm{F}}(\boldsymbol{w})$ 为最大值的 $\boldsymbol{w}$ 即为要求的投影向量 $\boldsymbol{w}^*$。

式(3-16)称为 Fisher 准则函数，需进一步转化为 $\boldsymbol{w}$ 的显函数，为此要对 $\bar{\boldsymbol{m}}_1$，$\bar{\boldsymbol{m}}_2$ 等项进一步演化。由于

$$\bar{\boldsymbol{m}}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} y_j = \frac{1}{N_i}\sum_{j=1}^{N_i}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_j = \boldsymbol{w}^{\mathrm{T}}\left(\frac{1}{N_i}\sum_{j=1}^{N_i}\boldsymbol{x}_j\right) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{m}_i \tag{3-17}$$

则有

$$(\bar{\boldsymbol{m}}_1 - \bar{\boldsymbol{m}}_2)^2 = (\boldsymbol{w}^{\mathrm{T}}\boldsymbol{m}_1 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{m}_2)^2 = \boldsymbol{w}^{\mathrm{T}}(\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^{\mathrm{T}}\boldsymbol{w} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}\boldsymbol{w} \tag{3-18}$$

其中 $\boldsymbol{S}_{\mathrm{b}} = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^{\mathrm{T}}$ 为类间离散矩阵。再由类内离散度

$$\bar{\boldsymbol{S}}_i = \sum_{j=1}^{N_i}(y_j - \bar{\boldsymbol{m}}_i)^2 = \sum_{j=1}^{N_i}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_j - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{m}_i)^2 = \boldsymbol{w}^{\mathrm{T}}\left[\sum_{j=1}^{N_i}(\boldsymbol{x}_j - \boldsymbol{m}_i)(\boldsymbol{x}_j - \boldsymbol{m}_i)^{\mathrm{T}}\right]\boldsymbol{w} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_i\boldsymbol{w} \tag{3-19}$$

其中 $\boldsymbol{S}_i = \sum_{j=1}^{N_i}(\boldsymbol{x}_j - \boldsymbol{m}_i)(\boldsymbol{x}_j - \boldsymbol{m}_i)^{\mathrm{T}}$。

所以总类内离散度为

$$\bar{\boldsymbol{S}}_w = \bar{\boldsymbol{S}}_1 + \bar{\boldsymbol{S}}_2 = \boldsymbol{w}^{\mathrm{T}}(\boldsymbol{S}_1 + \boldsymbol{S}_2)\boldsymbol{w} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_w\boldsymbol{w} \tag{3-20}$$

将式(3-18)与式(3-20)代入式(3-16),得到 Fisher 准则函数对于变量 $\boldsymbol{w}$ 的显式函数为

$$J_{\mathrm{F}}(\boldsymbol{w}) = \frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{w}\boldsymbol{w}} \tag{3-21}$$

对 $\boldsymbol{x}_n$ 的分量作线性组合 $y_n=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n(n=1,2,\cdots,N)$,从几何意义上看,$\|\boldsymbol{w}\|=1$,则每个 y_n 就是相对应的 $\boldsymbol{x}_n$ 到方向为 $\boldsymbol{w}$ 的直线上的投影。$\boldsymbol{w}$ 的方向不同,将使样本投影后的可分离程序不同,从而直接影响识别效果。

求解 Fisher 准则函数的条件极值,即可解得使 $J_{\mathrm{F}}(\boldsymbol{w})$ 为极值的 $\boldsymbol{w}^*$。对求取其极大值时的 $\boldsymbol{w}^*$,可以采用拉格朗日乘子算法解决,令分母非 0,即 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{w}\boldsymbol{w}=c\neq 0$。

构造拉格朗日函数

$$L(\boldsymbol{w},\lambda) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}\boldsymbol{w} - \lambda(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{w}\boldsymbol{w} - c) \tag{3-22}$$

对 $\boldsymbol{w}$ 求偏导,并令其为 0,即

$$\frac{\partial L(\boldsymbol{w},\lambda)}{\partial \boldsymbol{w}} = \boldsymbol{S}_{\mathrm{b}}\boldsymbol{w} - \lambda\boldsymbol{S}_{w}\boldsymbol{w} = 0 \tag{3-23}$$

得到

$$\boldsymbol{S}_{\mathrm{b}}\boldsymbol{w}^* = \lambda\boldsymbol{S}_{w}\boldsymbol{w}^* \tag{3-24}$$

由于 $\boldsymbol{S}_w$ 非奇异,两边左乘 $\boldsymbol{S}_w^{-1}$,得到 $\boldsymbol{S}_w^{-1}\boldsymbol{S}_{\mathrm{b}}\boldsymbol{w}^*=\lambda\boldsymbol{w}^*$,该式为矩阵 $\boldsymbol{S}_w^{-1}\boldsymbol{S}_{\mathrm{b}}$ 的特征值问题。其中,拉格朗日算子 λ 为矩阵 $\boldsymbol{S}_w^{-1}\boldsymbol{S}_{\mathrm{b}}$ 的特征值;$\boldsymbol{w}^*$ 即对应于特征值 λ 的特征向量,即最佳投影的坐标向量。

矩阵特征值的问题有标准的求解方法。在此给出一种直接求解方法,不求特征值而直接得到最优解 $\boldsymbol{w}^*$。

由于

$$\boldsymbol{S}_{\mathrm{b}} = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^{\mathrm{T}} \tag{3-25}$$

所以

$$\boldsymbol{S}_{\mathrm{b}}\boldsymbol{w}^* = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^{\mathrm{T}}\boldsymbol{w}^* = (\boldsymbol{m}_1 - \boldsymbol{m}_2)R$$

式中,$R=(\boldsymbol{m}_1-\boldsymbol{m}_2)^{\mathrm{T}}\boldsymbol{w}^*$ 为限定标量。进而,由于

$$\lambda\boldsymbol{w}^* = \boldsymbol{S}_w^{-1}\boldsymbol{S}_{\mathrm{b}}\boldsymbol{w}^* = \boldsymbol{S}_w^{-1}(\boldsymbol{S}_{\mathrm{b}}\boldsymbol{w}^*) = \boldsymbol{S}_w^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)R \tag{3-26}$$

得到

$$\boldsymbol{w}^* = \frac{R}{\lambda}\boldsymbol{S}_w^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2) \tag{3-27}$$

忽略比例因子 R/λ,得到最优解 $\boldsymbol{w}^*=\boldsymbol{S}_w^{-1}(\boldsymbol{m}_1-\boldsymbol{m}_2)$。因此,使得 $J_{\mathrm{F}}(\boldsymbol{w})$ 取极大值时的 $\boldsymbol{w}$ 即为 d 维空间到一维空间的最佳投影方向

$$\boldsymbol{w}^* = \boldsymbol{S}_w^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2) \tag{3-28}$$

向量 $\boldsymbol{w}^*$ 就是使 Fisher 准则函数 $J_{\mathrm{F}}(\boldsymbol{w})$ 达到极大值的解,也就是按 Fisher 准则将 d 维 X 空间投影到一维 Y 空间的最佳投影方向,$\boldsymbol{w}^*$ 的各分量值是对原 d 维特征向量求加权和的权值。

由式(3-28)表示的最佳投影方向是容易理解的,因为其中的$(\boldsymbol{m}_1-\boldsymbol{m}_2)$项是一向量,对与$(\boldsymbol{m}_1-\boldsymbol{m}_2)$平行的向量投影可使两均值点的距离最远。

但是如何使类间分得较开,同时又使类内密集程度较高这样一个综合指标来看,则需根据两类样本的分布离散程度对投影方向作相应的调整,这就体现在对$(\boldsymbol{m}_1-\boldsymbol{m}_2)$向量按 $\boldsymbol{S}_w^{-1}$ 作一线性变换,从而使 Fisher 准则函数达到极值点。

以上讨论了线性判别函数加权向量 $\boldsymbol{w}$ 的确定方法，并讨论了使 Fisher 准则函数极大的 d 维向量 $\boldsymbol{w}^*$ 的计算方法。由 Fisher 判别函数得到了最佳一维投影后，还需确定一个阈值点 y_0，一般可采用以下几种方法确定 y_0，即

$$y_0 = \frac{\bar{\boldsymbol{m}}_1 + \bar{\boldsymbol{m}}_2}{2} \tag{3-29}$$

$$y_0 = \frac{N_1 \bar{\boldsymbol{m}}_1 + N_2 \bar{\boldsymbol{m}}_2}{N_1 + N_2} \tag{3-30}$$

$$y_0 = \frac{\bar{\boldsymbol{m}}_1 + \bar{\boldsymbol{m}}_2}{2} + \frac{\ln(P(\omega_1)/P(\omega_2))}{N_1 + N_2 - 2} \tag{3-31}$$

式(3-29)是根据两类样本均值之间的平均距离来确定阈值点的。式(3-30)既考虑了样本均值之间的平均距离，又考虑了两类样本的容量大小作阈值位置的偏移修正。式(3-31)既使用了先验概率 $P(\omega_i)$，又考虑了两类样本的容量大小作阈值位置的偏移修正，目的都是使得分类误差尽可能小。

为了确定具体的分界面，还要指定线性方程的常数项。实际工作中可以采用对 y_0 进行逐次修正的方式，选择不同的 y_0 值，计算其对训练样本集的错误率，找到错误率较小的 y_0 值。

对于任意未知类别的样本 $\boldsymbol{x}$，计算它的投影点 $y=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$，决策规则为

$$\begin{cases} y > y_0, & \boldsymbol{x} \in \omega_1 \\ y < y_0, & \boldsymbol{x} \in \omega_2 \end{cases} \tag{3-32}$$

3.5.4　Fisher 算法的 MATLAB 实现

1. 流程图

根据上面所介绍的 Fisher 判别函数，可得出如图 3-3 所示的流程图。

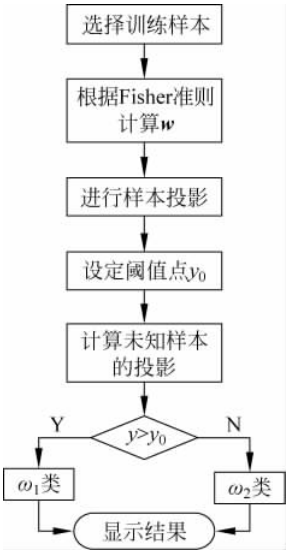

图 3-3　Fisher 分类器设计流程图

2. 样本均值

利用 MATLAB 程序得到训练样本均值,程序代码如下:

```
clear,close all;
N = 29;                                  % N 为训练样本总个数
X = [1495.18    1957.44    3498.02    % X 为训练样本
     1125.17    1594.39    2937.73
     1269.07    1910.72    2701.97
     ……
……]
m1 = mean(X(1:11,:));                   % 求得第一类样本均值
m2 = mean(X(12:29,:));                  % 求得第二类样本均值
```

3. 投影向量

Fisher 准则的目的就是寻找最优投影方向,使得 $\boldsymbol{w}$ 为最好的投影向量 $\boldsymbol{w}^*$。

利用如下 MATLAB 程序求得最佳投影向量:

```
S1 = 0;S2 = 0;                                  % 初始化类离散度
for i = 1:11
    S1 = S1 + (X(i,:) - m1) * (X(i,:) - m1)';   % 求得第一类的类内离散度
end
for i = 12:29
    S2 = S2 + (X(i,:) - m2) * (X(i,:) - m2)';   % 求得第二类的类内离散度
end
Sw = S1 + S2;                                   % 求得总类内离散度
W = inv(Sw) * (m1 - m2);                        % 求得最佳投影方向
```

4. 阈值点

本设计器采用 $y_0=\boldsymbol{w}^*(\boldsymbol{m}_1+\boldsymbol{m}_2)^{\mathrm{T}}/2$ 来确定阈值点,由于该式既考虑了样本均值之间的平均距离,又考虑了两类样本的容量大小作阈值位置的偏移修正,因此采用它可以使得分类误差尽可能小。

5. 输出分类结果

对于任意未知类别的样本 $\boldsymbol{x}$,计算它的投影点 $y=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$,决策规则为:当 $y>y_0$ 时,$\boldsymbol{x}\in\omega_1$;当 $y<y_0$ 时,$\boldsymbol{x}\in\omega_2$。

输出分类结果的 MATLAB 程序如下:

```
for i = 1:22
y = W * x(i,:)'                 % 确定投影点
```

```
if y>y0                % 当 y>y0 时,测试样本属于第一类
  disp('一')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'r+','MarkerSize',6,'LineWidth',2)
else
  disp('二')           % 当 y<y0 时,测试样本属于第二类
  hold on,plot3(x(i,1),x(i,2),x(i,3),'b+','MarkerSize',6,'LineWidth',2)
end
end
```

3.5.5　识别待测样本类别

本节内容以兑酒为例。不同类型的酒是由多种成分按不同的比例构成的，兑酒时需要三种原料(X,Y,Z)，现在已测出不同酒中三种原料的含量，需要判定它属于四种类型中的哪一种。样本中，前 29 组数据用于学习，后 22 组用于识别。

1. 选择分类方法

由于 Fisher 分类法一次只能将样本分成两类，因此，首先要将样本分成两大类，即一类、二类，然后再继续往下分，将其分成 1、2、3、4 类。分类流程图如图 3-4 所示。

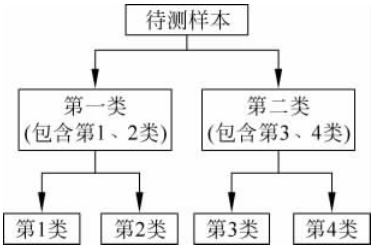

图 3-4　Fisher 分类流程图

将样本分成两大类有 3 种分法，如表 3-2 所示。

表 3-2　Fisher 分类方法

种　　类	分 类 方 法
第一种	第 1、2 类作为第一类，第 3、4 类作为第二类
第二种	第 1、3 类作为第一类，第 2、4 类作为第二类
第三种	第 1、4 类作为第一类，第 2、3 类作为第二类

根据所给的训练样本数据，利用 MATLAB 程序得出训练样本分布图，如图 3-5 所示。

观察训练样本分布图可知，如果将第 1、2 类分在一起作为第一类，第 3、4 类分在一起作为第二类，这样很难将它们分开，因此排除这种分类方法，应选择第二、三种分类方法。

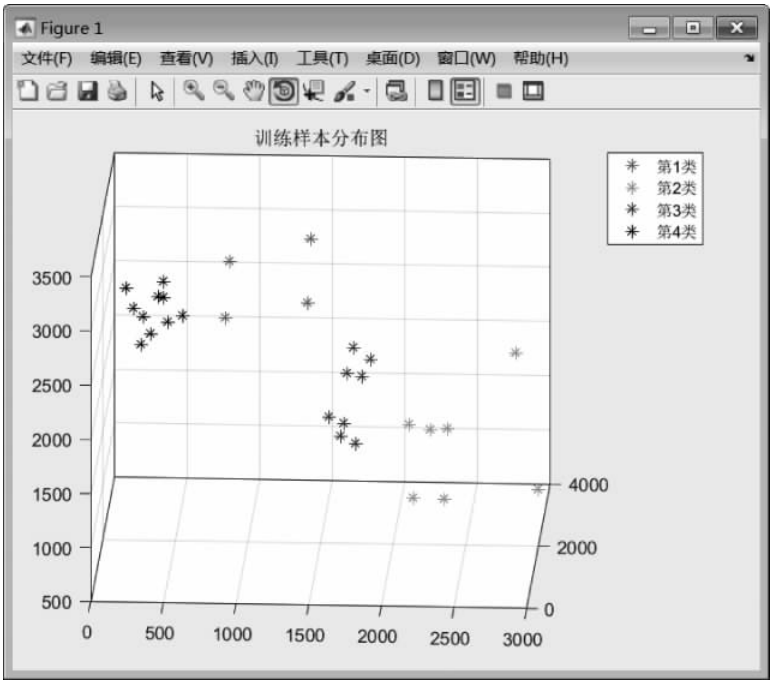

图 3-5 训练样本分布图

2. MATLAB 程序

(1) 选择第二种方法的相关程序及仿真结果。

训练样本分布图程序代码如下：

```
clear,close all;
N = 29;
X = [864.45    1647.31    2665.9;
     877.88    2031.66    3071.18;
     1418.79   1775.89    2772.9;
     1449.58   1641.58    3405.12;
     2352.12   2557.04    1411.53;
     2297.28   3340.14    535.62;
     2092.62   3177.21    584.32;
     2205.36   3243.74    1202.69;
     2949.16   3244.44    662.42;
     2802.88   3017.11    1984.98;
     2063.54   3199.76    1257.21;
     1739.94   1675.15    2395.96;
     1756.77   1652       1514.98;
     1803.58   1583.12    2163.05;
     1571.17   1731.04    1735.33;
     1845.59   1918.81    2226.49;
     1692.62   1867.5     2108.97;
     1680.67   1575.78    1725.1;
```

```
      1651.52    1713.28    1570.38;
      373.3      3087.05    2429.47;
      222.85     3059.54    2002.33;
      401.3      3259.94    2150.98;
      363.34     3477.95    2462.86;
      104.8      3389.83    2421.83;
      499.85     3305.75    2196.22;
      172.78     3084.49    2328.65;
      341.59     3076.62    2438.63;
      291.02     3095.68    2088.95;
      237.63     3077.78    2251.96;
      ]
fig = figure;
plot3(X(1:4,1),X(1:4,2), X(1:4,3),'r * ')
hold on,plot3(X(5:11,1),X(5:11,2), X(5:11,3),'g * ')
hold on,plot3(X(12:19,1),X(12:19,2), X(12:19,3),'b * ')
hold on,plot3(X(20:29,1),X(20:29,2), X(20:29,3),'k * ');grid;box
title('训练样本分布图')
legend('第 1 类','第 2 类','第 3 类','第 4 类')
```

测试样本分为一(1、3)类、二(2、4)类程序代码如下：

```
clear,close all;
N = 29;
X = [864.45    1647.31    2665.9
     877.88    2031.66    3071.18
     1418.79   1775.89    2772.9
     1449.58   1641.58    3405.12
     1739.94   1675.15    2395.96
     1756.77   1652       1514.98
     1803.58   1583.12    2163.05
     1571.17   1731.04    1735.33
     1845.59   1918.81    2226.49
     1692.62   1867.5     2108.97
     1680.67   1575.78    1725.1
     1651.52   1713.28    1570.38
     2352.12   2557.04    1411.53
     2297.28   3340.14    535.62
     2092.62   3177.21    584.32
     2205.36   3243.74    1202.69
     2949.16   3244.44    662.42
     2802.88   3017.11    1984.98
     2063.54   3199.76    1257.21
     373.3     3087.05    2429.47
     222.85    3059.54    2002.33
     401.3     3259.94    2150.98
     363.34    3477.95    2462.86
     104.8     3389.83    2421.83
     499.85    3305.75    2196.22
     172.78    3084.49    2328.65
     341.59    3076.62    2438.63
     291.02    3095.68    2088.95
     237.63    3077.78    2251.96]
```

```
fig = figure;
plot3(X(1:12,1),X(1:12,2), X(1:12,3),'b + ')
hold on,plot3(X(13:29,1),X(13:29,2), X(13:29,3),'r + ');grid;box
title('分为一、二类分布图')
m1 = mean(X(1:12,:));
m2 = mean(X(13:29,:));
S1 = 0;S2 = 0;
for i = 1:12
    S1 = S1 + (X(i,:) - m1) * (X(i,:) - m1)';
end
for i = 13:29
    S2 = S2 + (X(i,:) - m2) * (X(i,:) - m2)';
end
Sw = S1 + S2;
W = inv(Sw) * (m1 - m2);
W = W./norm(W)
x = [1702.8   1639.79  2068.74
    1877.93  1860.96  1975.3
    867.81   2334.68  2535.1
    1831.49  1713.11  1604.68
    460.69   3274.77  2172.99
    2374.98  3346.98  975.31
    2271.89  3482.97  946.7
    1783.64  1597.99  2261.31
    198.83   3250.45  2445.08
    1494.63  2072.59  2550.51
    1597.03  1921.52  2126.76
    1598.93  1921.08  1623.33
    1243.13  1814.07  3441.07
    2336.31  2640.26  1599.63
    354      3300.12  2373.61
    2144.47  2501.62  591.51
    426.31   3105.29  2057.8
    1507.13  1556.89  1954.51
    343.07   3271.72  2036.94
    2201.94  3196.22  935.53
    2232.43  3077.87  1298.87
    1580.1   1752.07  2463.04
    1962.4   1594.97  1835.95
    1495.18  1957.44  3498.02
    1125.17  1594.39  2937.73
    24.22    3447.31  2145.01
    1269.07  1910.72  2701.97
    1802.07  1725.81  1966.35
    1817.36  1927.4   2328.79
    1860.45  1782.88  1875.13];
y0 = W * (m1 + m2)'/2;
for i = 1:30
if W * x(i,:)'> y0
  disp('一')
```

```
  hold on,plot3(x(i,1),x(i,2),x(i,3),'g * ','MarkerSize',6,'LineWidth',2)
else
  disp('二')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'k * ','MarkerSize',6,'LineWidth',2)
end
end
legend('训练样本一类','训练样本二类','测试样本一类','测试样本二类')
```

程序运行完之后，出现如图 3-6 所示的一、二类数据分类结果图界面。

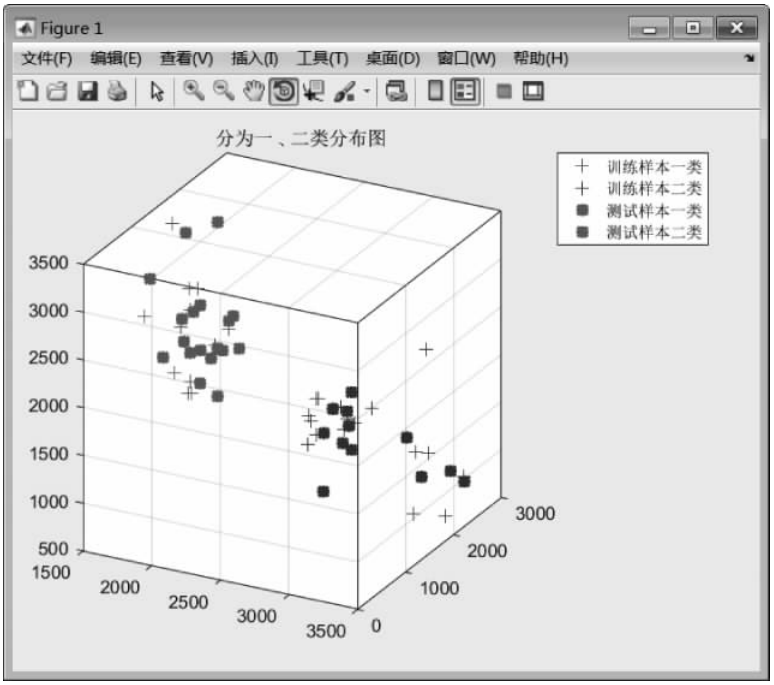

图 3-6　数据的一、二类分类结果图界面

运行 MATLAB 程序的结果如下：

```
W =
    0.2363   - 0.9187   0.3166
一
一
一
一
二
二
二
二
一
二
一
一
一
```

```
一
二
二
二
二
二
一
二
二
二
二
一
一
一
一
二
一
一
一
一
```

测试样本一类分为 1、3 类的程序代码如下：

```
N = 12;
X = [864.45    1647.31    2665.9
     877.88    2031.66    3071.18
     1418.79   1775.89    2772.9
     1449.58   1641.58    3405.12
     1739.94   1675.15    2395.96
     1756.77   1652       1514.98
     1803.58   1583.12    2163.05
     1571.17   1731.04    1735.33
     1845.59   1918.81    2226.49
     1692.62   1867.5     2108.97
     1680.67   1575.78    1725.1
     1651.52   1713.28    1570.38]
fig = figure;
plot3(X(1:4,1),X(1:4,2), X(1:4,3),'r.')
hold on,plot3(X(5:12,1),X(5:12,2), X(5:12,3),'b.');grid;box
title('分为 1、3 类分布图')
m1 = mean(X(1:4,:));
m2 = mean(X(5:12,:));
S1 = 0;S2 = 0;
for i = 1:4
    S1 = S1 + (X(i,:) - m1) * (X(i,:) - m1)';
end
for i = 5:12
    S2 = S2 + (X(i,:) - m2) * (X(i,:) - m2)';
end
Sw = S1 + S2;
W = inv(Sw) * (m1 - m2);
```

```
W = W./norm(W)
x = [1702.8    1639.79    2068.74
    1877.93   1860.96    1975.3
    867.81    2334.68    2535.1
    1831.49   1713.11    1604.68
    1783.64   1597.99    2261.31
    1494.63   2072.59    2550.51
    1597.03   1921.52    2126.76
    1598.93   1921.08    1623.33
    1243.13   1814.07    3441.07
    1507.13   1556.89    1954.51
    1580.1    1752.07    2463.04
    1962.4    1594.97    1835.95
    1495.18   1957.44    3498.02
    1125.17   1594.39    2937.73
    1269.07   1910.72    2701.97
    1802.07   1725.81    1966.35
    1817.36   1927.4     2328.79
    1860.45   1782.88    1875.13
];
  hold on,plot3(1495.18,1957.44,3498.02,'r + ','MarkerSize',6,'LineWidth',2)
  hold on,plot3(1557.27,1746.27,1879.13,'b + ','MarkerSize',6,'LineWidth',2)
y0 = W * (m1 + m2)'/2;
for i = 1:18
if W * x(i,:)'> y0
  disp('1')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'r + ','MarkerSize',6,'LineWidth',2)
else
  disp('3')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'b + ','MarkerSize',6,'LineWidth',2)
end
end
legend('训练样本 1 类','训练样本 3 类','测试样本 1 类','测试样本 3 类')
```

程序运行完之后，出现如图 3-7 所示的 1、3 类数据分类结果图界面。

运行 MATLAB 程序的结果如下：

```
W =
    -0.4737    0.0499    0.8793
3
3
1
3
3
1
3
3
1
```

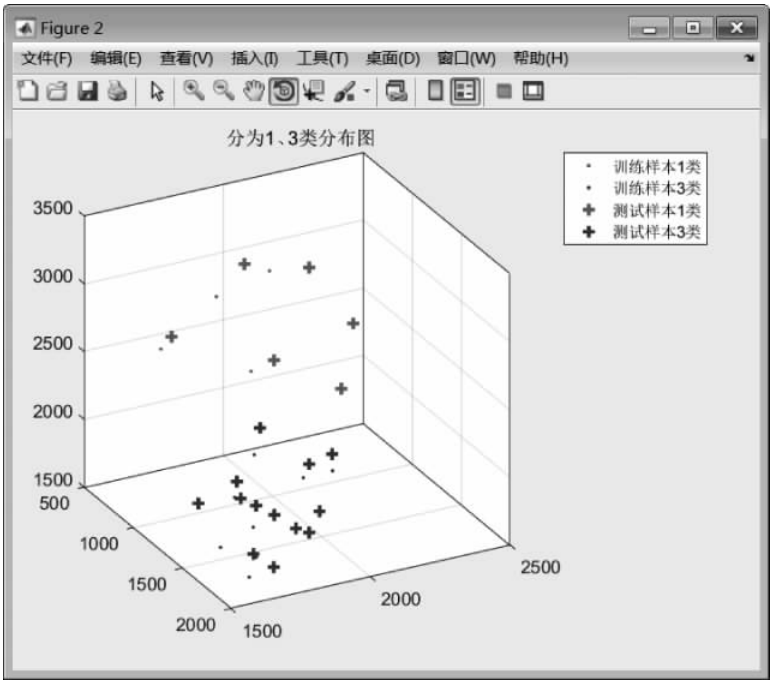

图 3-7　测试样本的 1、3 类分类结果图界面

```
3
3
3
1
1
1
3
3
3
```

测试样本二类分为 2、4 类的程序代码如下：

```
clear,close all;
N = 17;
X = [2352.12    2557.04    1411.53;
     2297.28    3340.14    535.62;
     2092.62    3177.21    584.32;
     2205.36    3243.74    1202.69;
     2949.16    3244.44    662.42;
     2802.88    3017.11    1984.98;
     2063.54    3199.76    1257.21;
     373.3      3087.05    2429.47;
     222.85     3059.54    2002.33;
     401.3      3259.94    2150.98;
     363.34     3477.95    2462.86;
```

```
      104.8    3389.83    2421.83;
      499.85   3305.75    2196.22;
      172.78   3084.49    2328.65;
      341.59   3076.62    2438.63;
      291.02   3095.68    2088.95;
      237.63   3077.78    2251.96;]
fig = figure;
plot3(X(1:7,1),X(1:7,2), X(1:7,3),'r.')
hold on,plot3(X(8:17,1),X(8:17,2), X(8:17,3),'b.');grid;box
title('分为 2、4 类分布图')
m1 = mean(X(1:7,:));
m2 = mean(X(8:17,:));
S1 = 0;S2 = 0;
for i = 1:7
    S1 = S1 + (X(i,:) - m1) * (X(i,:) - m1)';
end
for i = 8:17
    S2 = S2 + (X(i,:) - m2) * (X(i,:) - m2)';
end
Sw = S1 + S2;
W = inv(Sw) * (m1 - m2);
W = W./norm(W)
x = [460.69     3274.77    2172.99
    2374.98    3346.98    975.31
    2271.89    3482.97    946.7
    198.83     3250.45    2445.08
    2336.31    2640.26    1599.63
    354        3300.12    2373.61
    2144.47    2501.62    591.51
    426.31     3105.29    2057.8
    343.07     3271.72    2036.94
    2201.94    3196.22    935.53
    2232.43    3077.87    1298.87
    24.22      3447.31    2145.01
];
  hold on,plot3(2232.43,3077.87,1298.87,'r + ','MarkerSize',6,'LineWidth',2)
  hold on,plot3(362.51,3150.03,2472,'b + ','MarkerSize',6,'LineWidth',2)
y0 = W * (m1 + m2)'/2;
for i = 1:12
if W * x(i,:)'> y0
  disp('2')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'r + ','MarkerSize',6,'LineWidth',2)
else
  disp('4')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'b + ','MarkerSize',6,'LineWidth',2)
end
end
legend('训练样本 2 类','训练样本 4 类','测试样本 2 类','测试样本 4 类')
```

程序运行完之后，出现如图 3-8 所示的 2、4 类数据分类结果图界面。

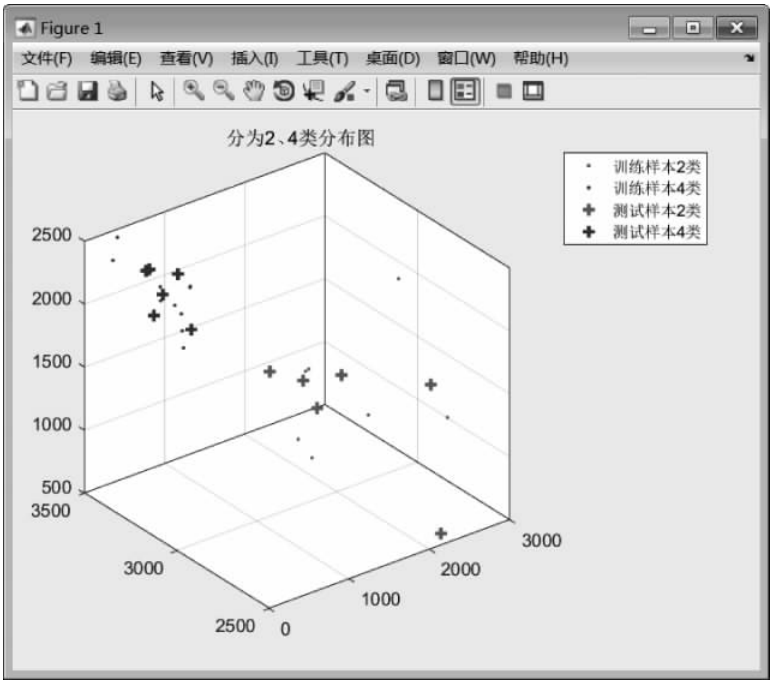

图 3-8 测试样本的 2、4 类分类结果图界面

运行 MATLAB 程序的结果如下：

```
W =
    0.8696   -0.0333   -0.4926
4
2
2
4
2
4
2
4
4
2
2
4
```

(2) 选择第三种方法的相关程序及仿真结果。

测试样本分为一(1、4)类、二(2、3)类的程序代码如下：

```
clear,close all;
N=29;
X = [864.45    1647.31    2665.9
     877.88    2031.66    3071.18
     1418.79   1775.89    2772.9
     1449.58   1641.58    3405.12
     373.3     3087.05    2429.47;
     222.85    3059.54    2002.33;
```

```
        401.3    3259.94    2150.98;
        363.34   3477.95    2462.86;
        104.8    3389.83    2421.83;
        499.85   3305.75    2196.22;
        172.78   3084.49    2328.65;
        341.59   3076.62    2438.63;
        291.02   3095.68    2088.95;
        237.63   3077.78    2251.96;
        2352.12  2557.04    1411.53;
        2297.28  3340.14    535.62;
        2092.62  3177.21    584.32;
        2205.36  3243.74    1202.69;
        2949.16  3244.44    662.42;
        2802.88  3017.11    1984.98;
        2063.54  3199.76    1257.21;
        1739.94  1675.15    2395.96;
        1756.77  1652       1514.98;
        1803.58  1583.12    2163.05;
        1571.17  1731.04    1735.33;
        1845.59  1918.81    2226.49;
        1692.62  1867.5     2108.97;
        1680.67  1575.78    1725.1;
        1651.52  1713.28    1570.38;
        ]
fig = figure;
plot3(X(1:14,1),X(1:14,2), X(1:14,3),'r + ')
hold on,plot3(X(15:29,1),X(15:29,2), X(15:29,3),'b + ');grid;box
title('分为一、二类分布图')
m1 = mean(X(1:14,:));
m2 = mean(X(15:29,:));
S1 = 0;S2 = 0;
for i = 1:14
    S1 = S1 + (X(i,:) - m1) * (X(i,:) - m1)';
end
for i = 15:29
    S2 = S2 + (X(i,:) - m2) * (X(i,:) - m2)';
end
Sw = S1 + S2;
W = inv(Sw) * (m1 - m2);
W = W./norm(W)
x = [1702.8    1639.79    2068.74
    1877.93   1860.96    1975.3
    867.81    2334.68    2535.1
    1831.49   1713.11    1604.68
    460.69    3274.77    2172.99
    2374.98   3346.98    975.31
    2271.89   3482.97    946.7
    1783.64   1597.99    2261.31
    198.83    3250.45    2445.08
    1494.63   2072.59    2550.51
    1597.03   1921.52    2126.76
    1598.93   1921.08    1623.33
    1243.13   1814.07    3441.07
    2336.31   2640.26    1599.63
    354       3300.12    2373.61
    426.31    3105.29    2057.8
```

```
        2144.47    2501.62    591.51
        1507.13    1556.89    1954.51
        343.07     3271.72    2036.94
        2201.94    3196.22    935.53
        2232.43    3077.87    1298.87
        1580.1     1752.07    2463.04
        1962.4     1594.97    1835.95
        1495.18    1957.44    3498.02
        1125.17    1594.39    2937.73
        24.22      3447.31    2145.01
        1269.07    1910.72    2701.97
        1802.07    1725.81    1966.35
        1817.36    1927.4     2328.79
        1860.45    1782.88    1875.13];
y0 = W * (m1 + m2)'/2;
for i = 1:30
if W * x(i,:)'> y0
  disp('一')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'r * ','MarkerSize',6,'LineWidth',2)
else
  disp('二')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'b * ','MarkerSize',6,'LineWidth',2)
end
end
legend('训练样本一类','训练样本二类','测试样本一类','测试样本二类')
```

程序运行完之后,出现如图 3-9 所示的数据分类结果图界面。

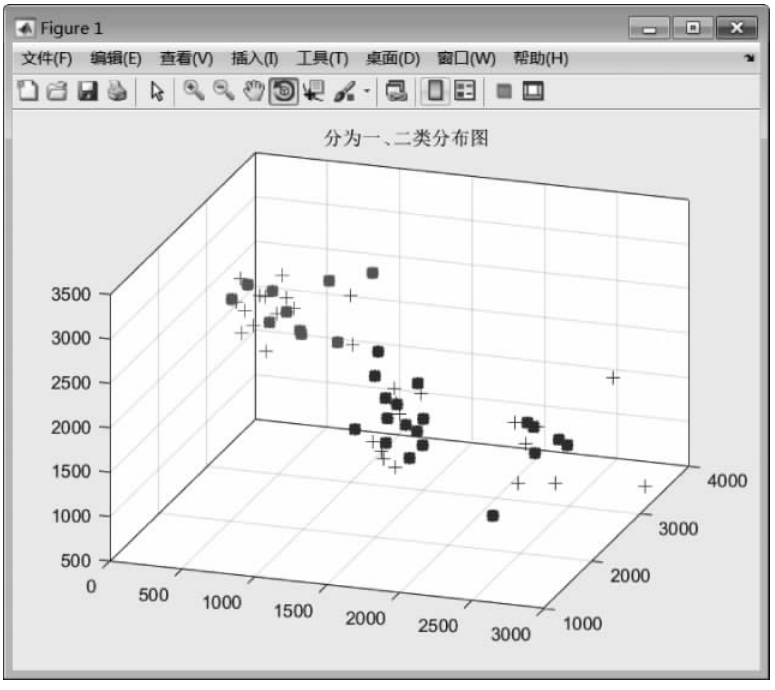

图 3-9 按第三种方法分为一、二类的结果图界面

运行 MATLAB 程序之后的结果如下：

```
W =
    -0.8228    0.2321    0.5188
二
二
二
一
二
二
一
二
二
二
二
二
一
二
二
二
一
二
二
一
二
二
一
二
二
一
二
二
二
二
二
二
一
一
一
一
一
二
二
二
二
```

测试样本一类分为 1、4 类的程序代码如下：

```
N = 14;
X = [864.45    1647.31    2665.9
     877.88    2031.66    3071.18
     1418.79   1775.89    2772.9
     1449.58   1641.58    3405.12
     373.3     3087.05    2429.47;
     222.85    3059.54    2002.33;
     401.3     3259.94    2150.98;
     363.34    3477.95    2462.86;
     104.8     3389.83    2421.83;
     499.85    3305.75    2196.22;
     172.78    3084.49    2328.65;
     341.59    3076.62    2438.63;
     291.02    3095.68    2088.95;
     237.63    3077.78    2251.96;
     ]
```

```
fig = figure;
plot3(X(1:4,1),X(1:4,2), X(1:4,3),'r.')
hold on,plot3(X(5:14,1),X(5:14,2), X(5:14,3),'b.');grid;box
title('分为 1、4 类分布图')
m1 = mean(X(1:4,:));
m2 = mean(X(5:14,:));
S1 = 0;S2 = 0;
for i = 1:4
    S1 = S1 + (X(i,:) - m1) * (X(i,:) - m1)';
end
for i = 5:14
    S2 = S2 + (X(i,:) - m2) * (X(i,:) - m2)';
end
Sw = S1 + S2;
W = inv(Sw) * (m1 - m2);
W = W./norm(W)
x = [867.81   2334.68  2535.1
    460.69   3274.77  2172.99
    198.83   3250.45  2445.08
    1243.13  1814.07  3441.07
    354      3300.12  2373.61
    426.31   3105.29  2057.8
    343.07   3271.72  2036.94
    1495.18  1957.44  3498.02
    1125.17  1594.39  2937.73
    24.22    3447.31  2145.01
    1269.07  1910.72  2701.97
];
  hold on,plot3(1495.18,1957.44,3498.02,'r + ','MarkerSize',6,'LineWidth',2)
  hold on,plot3(1557.27,1746.27,1879.13,'b + ','MarkerSize',6,'LineWidth',2)
y0 = W * (m1 + m2)'/2;
for i = 1:11
if W * x(i,:)'> y0
  disp('1')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'r + ','MarkerSize',6,'LineWidth',2)
else
  disp('4')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'b + ','MarkerSize',6,'LineWidth',2)
end
end
legend('训练样本 1 类','训练样本 4 类','测试样本 1 类','测试样本 4 类')
```

程序运行完之后,出现如图 3-10 所示的 1、4 类数据分类结果界面。

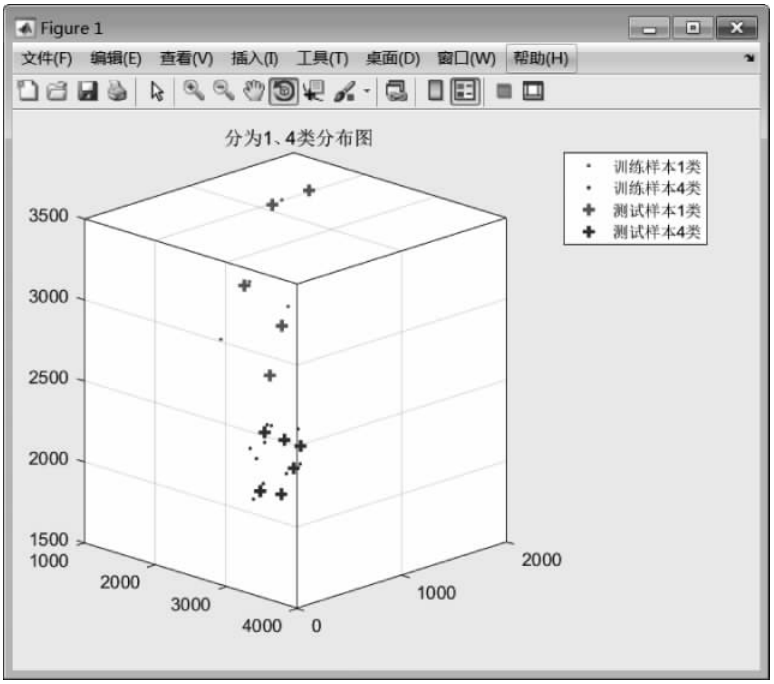

图 3-10　数据的 1、4 类分类结果图界面

运行 MATLAB 程序的结果如下：

```
W =
    0.4742   -0.7890   0.3906
1
4
4
1
4
4
4
1
1
4
1
```

测试样本二类分为 2、3 类的程序代码如下：

```
clear,close all;
N=15;
X = [2352.12    2557.04    1411.53;
     2297.28    3340.14    535.62;
     2092.62    3177.21    584.32;
     2205.36    3243.74    1202.69;
     2949.16    3244.44    662.42;
```

```
        2802.88    3017.11    1984.98;
        2063.54    3199.76    1257.21;
        1739.94    1675.15    2395.96
        1756.77    1652       1514.98
        1803.58    1583.12    2163.05
        1571.17    1731.04    1735.33
        1845.59    1918.81    2226.49
        1692.62    1867.5     2108.97
        1680.67    1575.78    1725.1
        1651.52    1713.28    1570.38
        ]
fig = figure;
plot3(X(1:7,1),X(1:7,2), X(1:7,3),'r.')
hold on,plot3(X(8:15,1),X(8:15,2), X(8:15,3),'b.');grid;box
title('分为 2、3 类分布图')
m1 = mean(X(1:7,:));
m2 = mean(X(8:15,:));
S1 = 0;S2 = 0;
for i = 1:7
    S1 = S1 + (X(i,:) - m1) * (X(i,:) - m1)';
end
for i = 8:15
    S2 = S2 + (X(i,:) - m2) * (X(i,:) - m2)';
end
Sw = S1 + S2;
W = inv(Sw) * (m1 - m2);
W = W./norm(W)
x = [1702.8        1639.79    2068.74
    1877.93       1860.96    1975.3
    1831.49       1713.11    1604.68
    2374.98       3346.98    975.31
    2271.89       3482.97    946.7
    1783.64       1597.99    2261.31
    1494.63       2072.59    2550.51
    1597.03       1921.52    2126.76
    1598.93       1921.08    1623.33
    2336.31       2640.26    1599.63
    2144.47       2501.62    591.51
    1507.13       1556.89    1954.51
    2201.94       3196.22    935.53
    2232.43       3077.87    1298.87
    1580.1        1752.07    2463.04
    1962.4        1594.97    1835.95
    1802.07       1725.81    1966.35
    1817.36       1927.4     2328.79
    1860.45       1782.88    1875.13
    ];
      hold on,plot3(2232.43,3077.87,1298.87,'r + ','MarkerSize',6,'LineWidth',2)
      hold on,plot3(362.51,3150.03,2472,'b + ','MarkerSize',6,'LineWidth',2)
    y0 = W * (m1 + m2)'/2;
```

```
for i = 1:19
if W * x(i,:)'> y0
  disp('2')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'r + ','MarkerSize',6,'LineWidth',2)
else
  disp('3')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'b + ','MarkerSize',6,'LineWidth',2)
end
end
legend('训练样本 2 类','训练样本 3 类','测试样本 2 类','测试样本 3 类')
```

程序运行完之后，出现如图 3-11 所示的 2、3 类数据分类结果图界面。

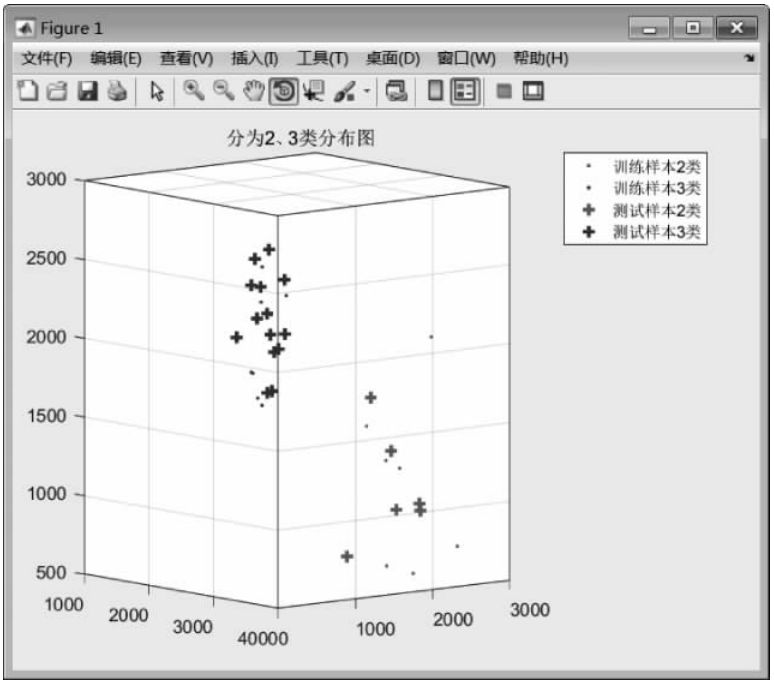

图 3-11　2、3 类数据分类结果图界面

运行 MATLAB 程序的结果如下：

```
W =
    0.3837    0.7917    - 0.4754
3
3
3
2
2
3
3
3
3
```

```
2
2
3
2
2
3
3
3
3
3
```

将两种分类方法的分类结果进行比较,即利用第二、三种分类方法进行比较,得出的分类结果如表 3-3 所示。

表 3-3 两种分类结果的比较

X	Y	Z	设计分类器分类结果 第二种(1、3,2、4 类)	设计分类器分类结果 第三种(1、4,2、3 类)
1702.8	1639.79	2068.74	3	3
1877.93	1860.96	1975.3	3	3
867.81	2334.68	2535.1	1	1
1831.49	1713.11	1604.68	3	3
460.69	3274.77	2172.99	4	4
2374.98	3346.98	975.31	2	2
2271.89	3482.97	946.7	2	2
1783.64	1597.99	2261.31	3	3
198.83	3250.45	2445.08	4	4
1494.63	2072.59	2550.51	1	3
1597.03	1921.52	2126.76	3	3
1598.93	1921.08	1623.33	3	3
1243.13	1814.07	3441.07	1	1
2336.31	2640.26	1599.63	2	2
354	3300.12	2373.61	4	4
2144.47	2501.62	591.51	2	2
426.31	3105.29	2057.8	1	4
1507.13	1556.89	1954.51	2	3
343.07	3271.72	2036.94	4	4
2201.94	3196.22	935.53	2	2
2232.43	3077.87	1298.87	3	2
1580.1	1752.07	2463.04	3	3
1962.4	1594.97	1835.95	3	3
1495.18	1957.44	3498.02	1	1
1125.17	1594.39	2937.73	1	1
24.22	3447.31	2145.01	1	4
1269.07	1910.72	2701.97	1	1
1802.07	1725.81	1966.35	3	3
1817.36	1927.4	2328.79	3	3
1860.45	1782.88	1875.13	3	3

比较这两种分类方法，方法二有两个错误分类，方法三仅有一个错误分类，所以方法三优于方法二。

3.5.6 结论

本节主要论述了 Fisher 分类法的概念、特点及其分类器设计，重点讨论了利用 Fisher 分类法设计分类器的全过程。在设计该种分类器的过程中，首先利用训练样本求得最佳投影方向 w^*，并确定阈值点 y_0；接着通过分析来归纳给定样本数据的分类情况；最后利用 MATLAB 中的相关函数、工具设计了基于 Fisher 分类法的分类器，并对测试数据进行了成功分类。整个讨论和设计过程，关键点和创新点就在于对测试数据的处理过程上，通过两种方法做到了快速且相对准确的分类。

3.6 基于支持向量机的分类法

3.6.1 支持向量机简介

从观测数据中学习归纳出系统运动规律，并利用这些规律对未来数据或无法观测到的数据进行预测一直是智能系统研究的重点。传统学习方法中采用的经验风险最小化方法(ERM)虽然将误差最小化，但不能最小化学习过程的泛化误差。ERM 方法不成功的例子就是神经网络中的过学习问题。为此，由 Vapnik 领导的贝尔实验室研究小组于 1963 年提出了一种新的非常有潜力的分类技术，支持向量机(support vector machine，SVM)是一种基于统计学习理论的模式识别方法，主要应用于模式识别领域。

支持向量机的基本思想是在样本空间或特征空间构造出最优超平面，使得超平面与不同类样本集之间的距离最大，从而达到最大的泛化能力。

3.6.2 支持向量机基本思想

SVM 是从线性可分情况下的最优分类面方法发展而来的，基本思想可用图 3-12 的两类线性可分情况说明。在图 3-12 中，实心点和空心点代表两类样本，实线 P_0、P_1 为分类线。两条虚线分别为过各类中离分类线最近的样本且平行于分类线的直线，它们之间的距离叫作分类间隔。所谓最优分类线就是要求分类线不但能将两个类正确分开(训练错误率为 0)，而且使分类间隔最大。

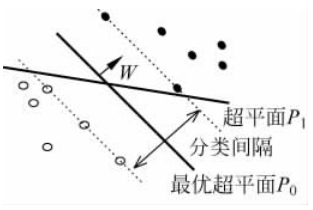

图 3-12 两类线性分割图

分类线方程为

$$\omega x + b = 0, \quad \omega \in R^m, b \in R \tag{3-33}$$

此时分类间隔为 $2/\|\omega\|$，使间隔最大等价于使 $\|\omega\|^2$ 最小，则可以通过求 $\|\omega\|^2/2$ 的极

小值获得分类间隔最大的最优超平面。

这里的约束条件为

$$y_i(\omega x_i+b)-1\geqslant 0,\quad i=1,2,\cdots,n \tag{3-34}$$

该约束优化问题可以用 Lagrange 方法求解,令

$$L(\omega,b,\alpha)=\frac{1}{2}\|\omega\|^2-\sum_{i=1}^{m}[y_i(\omega x_i+b)-1] \tag{3-35}$$

其中 $\alpha_i\geqslant 0$ 为每个样本的拉氏乘子,由 L 分别对 b 和 ω 导数为 0,可以导出

$$\sum_{i=1}^{m}\alpha_i y_i=0 \tag{3-36}$$

$$\omega=\sum_{i=1}^{m}\alpha_i y_i x_i \tag{3-37}$$

因此,解向量有一个由训练样本集的一个子集样本向量构成的展开式,该子集样本的拉氏乘子均不为 0,即支持向量。拉氏乘子为 0 的样本向量的贡献为 0,对选择分类超平面是无意义的。于是,就从训练集中得到了描述最优分类超平面的决策函数即支持向量机,它的分类功能由支持向量决定。这样决策函数可以表示为

$$f(x)=\mathrm{sgn}\left(\sum_{i=1}^{m}\alpha_i y_i(x+x_i)+b\right) \tag{3-38}$$

3.6.3 支持向量机的几个主要优点

(1) 它是专门针对有限样本情况的,其目标是得到现有信息下的最优解而不仅仅是样本数趋于无穷大时的最优值。

(2) 算法最终将转化成为一个二次型寻优问题。从理论上说,得到的将是全局最优点,解决了在神经网络方法中无法避免的局部极值问题。

(3) 算法将实际问题通过非线性变换转换到高维的特征空间(feature space)。在高维空间中构造线性判别函数来实现原空间中的非线性判别函数,特殊性质能保证机器有较好的推广能力,同时它巧妙地解决了维数问题,其算法复杂度与样本维数无关。

3.6.4 训练集为非线性情况

对于实际上难以线性分类的问题,待分类样本可以通过选择适当的非线性变换映射到某个高维的特征空间,使得在目标高维空间的这些样本线性可分,从而转化为线性可分问题。Cover 定理表明,通过这种非线性转换将非线性可分样本映射到足够高维的特征空间,非线性可分的样本将以极大的可能性变为线性可分。如果这个非线性转换为 $\phi(x)$,则超平面决策函数式可重写为

$$f(x)=\mathrm{sgn}\left(\sum_{i=1}^{m}\alpha_i y_i\phi(x)\phi(x_i)+b\right) \tag{3-39}$$

3.6.5 核函数

在上面的问题中只涉及训练样本之间的内积运算。实际上在高维空间只需进行内积运算，用原空间中的函数即可实现，甚至没有必要知道变换的形式。根据泛函的有关理论，只要一种 $K(x,x_i)$ 核函数满足 Mercer 条件，它就对应某一变换空间中的内积。因此，在最优分类面中采用适当的内积函数 $K(x,x_i)$ 就可以实现某一非线性变换后的线性分类，而计算复杂度却没有增加。核函数存在性定理表明：给定一个训练样本集，就一定存在一个相应的函数，训练样本通过核函数映射到高维特征空间的相是线性可分的。

对于一个特定的核函数，给定的样本集中的任意一个样本都可能成为一个支持向量。这意味着在一个支持向量机算法下观察到的特征在其他支持向量机算法下（其他核函数）并不能保持。因此对于解决具体问题来说，选择合适的核函数是很重要的。

常见的核函数有 3 类。

（1）多项式核函数。

$$K(x,x_i)=[(x,x_i)+1]^q \tag{3-40}$$

（2）径向基函数（RBF）。

$$K(x,x_i)=\exp\left(\frac{|x-x_i|^2}{\sigma^2}\right) \tag{3-41}$$

（3）采用 Sigmoid 函数作为内积。

$$K(x,x_i)=\tanh(v(x,x_i)+c) \tag{3-42}$$

3.6.6 多类分类问题

基本的支持向量机仅能解决两类分类问题。一些学者从两个方向研究用支持向量机解决多类分类问题：一个方向是将基本的两类支持向量机（Binary-class SVM，BSVM）扩展为多类分类支持向量机（multi-class SVM，MSVM），使支持向量机本身成为解决多类分类问题的多类分类器；另一方向则相反，将多类分类问题逐步转化为两类分类问题，即用多个两类分类支持向量机组成的多类分类器。

1. 多类分类支持向量机 MSVM

实际应用研究中多类分类问题更加常见，只要将目标函数由两类改为多类（k 类）情况，就可以很自然地将 BSVM 扩展为多类分类支持向量机 MSVM，以相似的方式可得到决策函数。

2. 基于 BSVM 的多类分类器

这种方案是为每个类构建一个 BSVM，如图 3-13 所示。对于每个类的 BSVM，其训练样本集的构成是：属于该类的样本为正样本，而不属于该类的其他所有样本为负样本，即该 BSVM 分类器将该类样本和其他样本分开。在 1-a-1 分类过程中训练样本需要重新标注，因为一个样本只有在对应类别的 BSVM 分类器才是正样本，对其他的 BSVM 分类器都是

负样本。

1) 1-a-1 分类器(One-against-one classifiers)

对于 1-a-1 分类器,解决 k 类分类问题就需要用到 BSVM,这种方案是每两个类别训练一个 BSVM 分类器,如图 3-14 所示。最后一个待识别样本的类别是由所有 $k(k-1)/2$ 个 BSVM“投票”决定的。

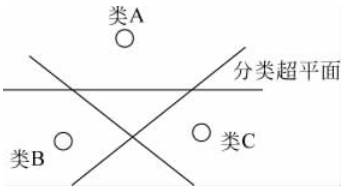

图 3-13 BSVM 分类原理图

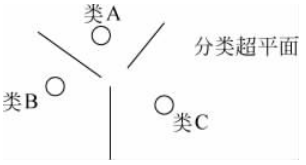

图 3-14 1-a-1 分类原理图

2) 多级 BSVM 分类器

这种方案是把多类分类问题分解为多级的两类分类子问题,如图 3-15 所示。两种典型方案中,A、B、C、D、E、F 分别表示 7 个不同的类。

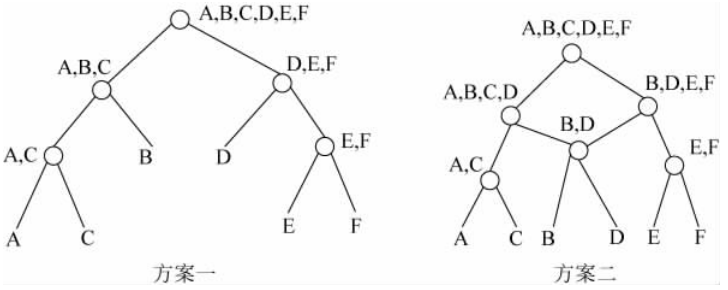

图 3-15 BSVM 多级分类

3.6.7 基于 SVM 的 MATLAB 实现

1. 建立模型流程图

基于 SVM 的数据分类设计流程如图 3-16 所示。

图 3-16 基于 SVM 的数据分类设计流程

2. 数据预处理

对训练集和测试集进行归一化预处理,采用[0,1]区间归一化。

$$f:\ x \rightarrow y = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

3. 训练和预测

以下是 MATLAB 中 LibSVM 工具箱中自带的 SVM 训练和预测的语句。

```
●model = svmtrain(train_labels, train_metrix,['libsvm_options'])
-- train_labels: 训练集的标签
-- train_metrix: 训练集的属性
-- libsvm_options: 一些选项参数
-- model: 训练得到的分类模型
●[predict_label, accuracy] = svmpredict(test_labels, test_metrix, model)
-- test_labels: 测试集的标签
-- test_metrix: 测试集的属性
-- model: 由 svmtrain 得到的分类模型
-- predict_label: 预测得到的测试集的标签
-- accuracy: 分类准确率
```

4. LibSVM 工具箱简介

LibSVM 是台湾大学林智仁教授等人开发设计的一个简单、易于使用且快速有效的 SVM 模式识别与回归的软件包。它不但提供了编译好的可在 Windows 系统中执行的文件，还提供了源代码，方便用户改进、修改以及在其他操作系统上应用。该软件还有一个特点，就是对 SVM 所涉及的参数调节相对比较少，提供了很多的默认参数，利用这些默认参数就可以解决很多问题；同时还提供了交互检验的功能。

以下是 LibSVM 工具箱的安装过程。

(1) 下载 libsvm-mat-2.91-1 的压缩文件，将其解压到 MATLAB→toolbox 的安装路径下。

(2) 在 MATLAB 软件中执行“设置路径”→“添加文件夹”命令，将该压缩包解压后添加到工具箱即可。

5. MATLAB 源程序

在程序开始时要将数据进行归一化处理，归一化程序如下：

```
function normal = normalization(x,kind)
% last modified 2009.2.24
%
if nargin < 2
    kind = 2; % kind = 1 or 2 表示第一类或第二类规范化
end

[m,n] = size(x);
normal = zeros(m,n);
%% normalize the data x to [0,1]
if kind == 1
    for i = 1:m
```

```
        ma = max( x(i,:) );
        mi = min( x(i,:) );
        normal(i,:) = ( x(i,:) - mi )./( ma - mi );
    end
end
%% normalize the data x to [ - 1,1]
if kind == 2
    for i = 1:m
        mea = mean( x(i,:) );
        va = var( x(i,:) );
        normal(i,:) = ( x(i,:) - mea )/va;
    end
end
```

SVM 的 MATLAB 完整源程序如下：

```
clear;
clc;
load SVM;

train_train = [train(1:4,:);train(5:11,:);train(12:19,:);train(20:30,:)]; % 手动划分为 4 类
train_target = [target(1:4);target(5:11);target(12:19);target(20:30)];
test_simulation = [simulation(1:6,:);simulation(7:11,:);simulation(12:24,:);
simulation(25:30,:)];
test_labels = [labels(1:6);labels(7:11);labels(12:24);labels(25:30)];

% train_train = normalization(train_train',2);
% test_simulation = normalization(test_simulation',2);
% train_train = train_train';
% test_simulation = test_simulation';

%           bestcv = 0;
%           for log2c = - 10:10,
%           for log2g = - 10:10,
%             cmd = ['- v 5 - c ', num2str(2 ^log2c), ' - g ',num2str(2 ^log2g)]; % 将训练集分
                                                                                  % 为 5 类
%             cv = svmtrain(train_target, train_train, cmd);
%             if (cv >= bestcv),
%               bestcv = cv; bestc = 2 ^log2c; bestg = 2 ^log2g;
%             end
%           end
%         end
%         fprintf('(best c = % g, g = % g, rate = % g)\n',bestc, bestg, bestcv);
%         cmd = ['- c ', num2str(bestc), ' - g ', num2str(bestg)];
%         model = svmtrain(train_target, train_train, cmd);

model = svmtrain(train_target, train_train, '- c 2 - g 0.2 - t 1'); % 核函数
  [predict_label, accuracy] = svmpredict(test_labels, test_simulation, model);
hold off
```

```
f = predict_label';
index1 = find(f == 1);
index2 = find(f == 2);
index3 = find(f == 3);
index4 = find(f == 4);
plot3(simulation(:,1),simulation(:,2),simulation(:,3),'o');
line(simulation(index1,1),simulation(index1,2),simulation(index1,3),'linestyle','none',
'marker','*','color','g');
line(simulation(index2,1),simulation(index2,2),simulation(index2,3),'linestyle','none',
'marker','<','color','r');
line(simulation(index3,1),simulation(index3,2),simulation(index3,3),'linestyle','none',
'marker','+','color','b');
line(simulation(index4,1),simulation(index4,2),simulation(index4,3),'linestyle','none',
'marker','>','color','y');
box;grid on;hold on;
xlabel('A');
ylabel('B');
zlabel('C');
title('支持向量机分类图');
```

程序运行完之后，出现如图 3-17 所示的分类结果图界面。

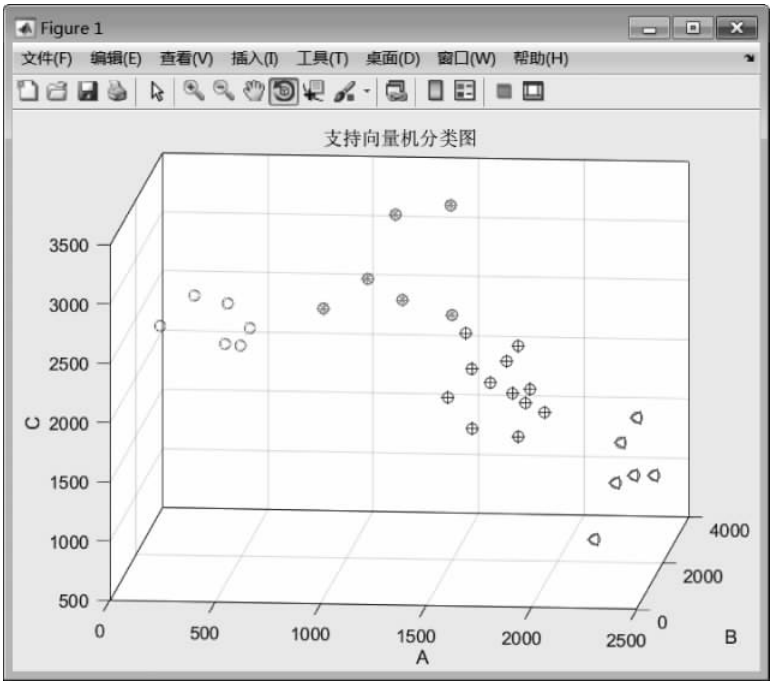

图 3-17　支持向量机分类结果图界面

在 MATLAB 命令窗口将出现如下结果：

```
Accuracy = 96.6667 % (29/30) (classification)
predict_label =
```

```
1
1
1
1
1
1
2
2
2
2
2
3
3
3
3
3
3
2
3
3
3
3
3
3
4
4
4
4
4
4
```

从程序运行结果可以看出分类正确率为96.6667%,有一个样本分错了。

3.6.8 结论

SVM具有优良的学习能力和推广能力,能够有效克服“维数灾难”和“过学习”问题,而SVM的参数是影响分类精度、回归预测的重要因素。仿真结果表明,预测结果可靠,可以为数据提供很好的参考信息。

习题

(1) 什么是判别函数法?

(2) 怎样确定线性判别函数的系数?

(3) 线性判别函数的分类器设计方法有哪些?非线性判别函数的分类器设计方法有哪些?它们有什么异同?