第 5 章 支持向量机

CHAPTER 5

支持向量机(Support Vector Machine, SVM)是机器学习中一类常用的分类和回归模型。SVM 通过将样本映射到高维空间,并在这个空间建立最大距离的分隔超平面,很好地解决了低维空间的不可分难题,同时又引入核函数解决了高维空间的计算复杂问题。SVM 建立在坚实的数学理论基础上,是一种非统计的二分类算法,也可以解决多分类和回归问题,SVM 尤其在解决小样本、非线性和高维问题中表现良好。

5.1 支持向量机模型

线性 SVM 与逻辑回归使用相同的模型,考虑 SVM 最初设计为解决二分类问题,用 SVM 解决多分类问题,常采用组合多个二分类器来构建多分类器。所以,以下从最简单的二分类问题来引入 SVM 的模型构建过程。回顾逻辑回归的二分类模型定义为:

$$y = g(x) \quad y \in \{0,1\} \tag{5-1}$$

其中 0 表示负类(Negative class),1 表示正类(Positive class)。其中 g 为 Sigmoid 函数,输出在[0,1],详细参考图 3.1。

在线性 SVM 中,采用相同形式的定义:

$$y = g(x) \quad y \in \{-1,1\} \tag{5-2}$$

其中−1 表示负类,1 表示正类。其中 g 不再采用 Sigmoid 函数,而是采用

$$g(z) = \begin{cases} 1 & (z \geqslant 0) \\ -1 & (z < 0) \end{cases}$$

的形式。

回顾基于线性函数的逻辑回归的模型定义:

$$H_{\boldsymbol{\theta}}(\boldsymbol{x}) = g(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}) \tag{5-3}$$

其中$\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x} = \theta_m x_m + \cdots + \theta_2 x_2 + \theta_1 x_1 + \theta_0 x_0$,且 $x_0 = 1$。而在 SVM 中,采用式(5-4)替换$\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}$:

$$\begin{cases} \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} = w_m x_m + \cdots + w_2 x_2 + w_1 x_1 \\ b = w_0 \end{cases} \tag{5-4}$$

则可以定义线性 SVM 的模型为:

$$H_w(\boldsymbol{x}) = g(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b) \tag{5-5}$$

对于包含 n 个训练样本的数据集(x_i, y_i),其中 $i \in \{1,2,\cdots,n\}$,每个样本包含 m 个特征,

表示为向量 $\boldsymbol{x}_i=\{x_{i1},x_{i2},\cdots,x_{im}\}$，定义 $x_{i0}=1$。其中 $y_i\in\{-1,1\}^n$，所以权重参数 $\boldsymbol{w}\in\mathbf{R}^m$。

采用线性 SVM 分类样本的原理，如图 5.1 所示。对于二维的特征 X_1 和特征 X_2，可以在二维坐标平面画出线性分界线，如图 5.1 中黑色虚线所示。但是线性分界线并不唯一，稍微平移或者旋转后只要确保不把两类数据分错，仍然可以达到线性分类的效果。那么是否存在最优的决策边界？对线性可分问题，就是二维平面上，对于所有样本是否存在那个最优分界线？SVM 正是为寻找这个最优决策边界而提出的。

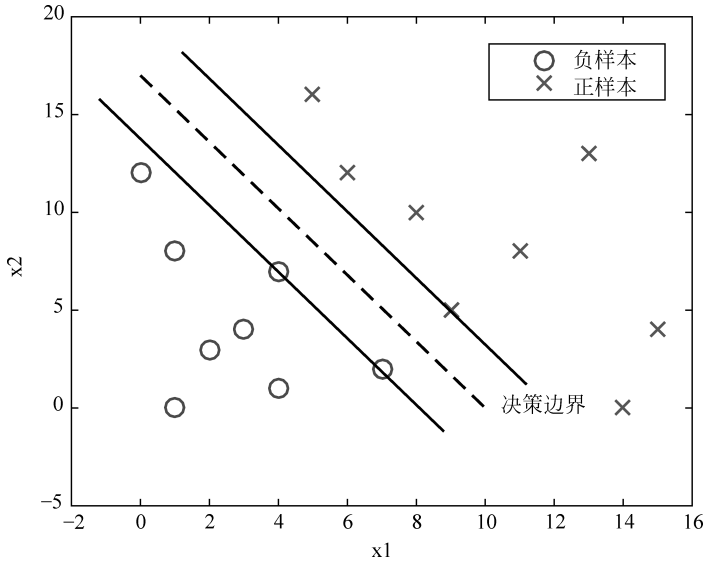


图 5.1　线性 SVM 分类原理

以如图 5.1 所示的分类问题为例，SVM 假定在最优分界线两边存在等距离的两个平行的分界线，如图 5.1 中黑色实线所示，这两条分界线将无限接近正或负样本，并确保分类正确的前提下，其间隔距离是最大的两条分界线，位于两个分界线上的样本点称为支持向量(Support vector)，两条分界线的间隔定义为 SVM 的几何间隔(Geometrical margin)。所以，SVM 又称为最大几何间隔机器学习算法。

对于三维特征可以画出 SVM 的决策边界，称为超平面(Hyperplane)，如图 5.2 所示。

然而，当样本的特征空间维度大于 3 时，不能再用几何的方法来解释 SVM 决策边界。而且，在采用 SVM 训练寻找最优决策边界时，更希望能够得到量化的输出。因此，引入 SVM 的函数间隔(Function margin)定义，来求解最大几何间隔的最优决策边界。

对于给定的包含 n 个训练样本的训练数据集$(\boldsymbol{x}_i,y_i)$，其中 $\boldsymbol{x}$ 为样本特征向量，y 是分类标签，定义 $y_i\in\{-1,1\}^n$ 其中 i 表示第 i 个样本，对第 i 个样本定义其函数间隔为：

$$\hat{r}_i=y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b) \tag{5-6}$$

对于式(5-6)，如果 $y_i=1$，根据 SVM 中 g 函数的定义可知，$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\geqslant 0$，反之 $y_i=-1$，$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b<0$，因此可以表示函数间隔为 $\hat{r}_i=|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b|$。再根据某一个样本的函数间隔，定义整体训练数据集上的全局函数间隔为：

$$\hat{r}=\min_{i=1,2,\cdots,n}(\hat{r}_i) \tag{5-7}$$

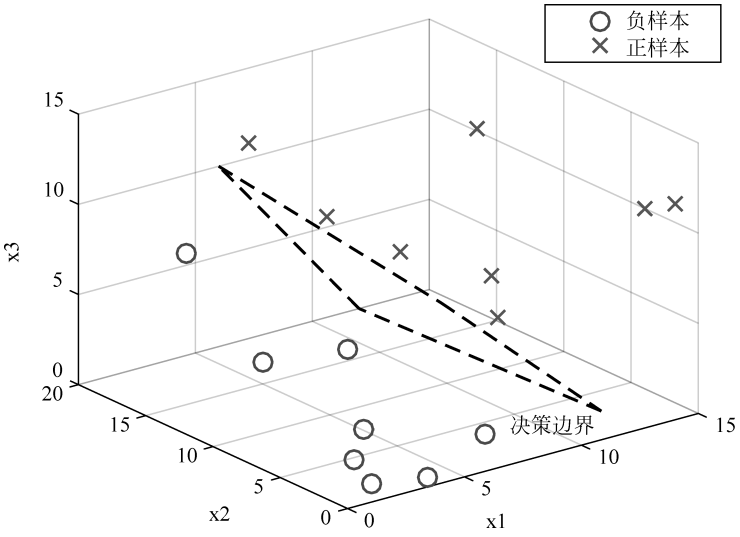


图 5.2　三维 SVM 决策边界

但是，仅仅采用函数间隔并不能很好地描述几何间隔最大化问题。比如当成比例地改变 $\boldsymbol{w}$ 和 b 的值时，函数间隔的值成比例增大，但是几何间隔却没有变化。尽管几何间隔可以描述 SVM 的分类实质，有直观的优点，但是不利于量化求解。所以，比较几何间隔和函数间隔的差别后，发现几何间隔就是函数间隔除以 $\|\boldsymbol{w}\|$，其中 $\|\boldsymbol{w}\|$ 为 $\boldsymbol{w}$ 的二阶范数，对于向量为向量各元素的平方和再开方，对于矩阵为矩阵转置与矩阵乘积，也就是最大特征矩阵开方，当 $\|\boldsymbol{w}\|=1$ 时，函数间隔与几何间隔表达式相同。

所以，SVM 定义局部间隔如下：

$$r_i = y_i\left(\left(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right)^{\mathrm{T}}\boldsymbol{x}_i + \frac{b}{\|\boldsymbol{w}\|}\right) \tag{5-8}$$

则，SVM 定义训练集上的全局间隔为：

$$r = \min_{i=1,2,\cdots,n}(r_i) \tag{5-9}$$

因此，SVM 定义最大间隔分类模型为：

$$\begin{aligned} &\max_{w,b,r} \quad r \\ &\text{subject} \quad \text{to} \quad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geqslant r, i = 1,2,\cdots,n \\ &\|\boldsymbol{w}\| = 1 \end{aligned} \tag{5-10}$$

最后，为便于量化计算，可以用函数间隔来改进式(5-10)，表示为：

$$\begin{aligned} &\max_{w,b,r} \quad \frac{\hat{r}}{\|\boldsymbol{w}\|} \\ &\text{subject} \quad \text{to} \quad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geqslant \hat{r}, i = 1,2,\cdots,n \end{aligned} \tag{5-11}$$

5.2　支持向量机代价函数

SVM 模型定义好之后，就需要寻找让模型取得最优值的权重参数。其中最优的含义即使得代价函数取得最接近全局最小值的参数组合。回顾以上 SVM 模型定义的几何间隔，

SVM 代价函数即正类的支持向量和负类的支持向量都尽量远离 SVM 的决策边界。若选择二次函数为 SVM 代价函数，考虑函数间隔为 1 时，则其几何意义即将距离决策边界最近的点到决策边界的距离定义为$\frac{1}{\|\boldsymbol{w}\|}$，根据 SVM 模型的定义，要求解$\frac{1}{\|\boldsymbol{w}\|}$的最大值相当于求二次函数$\frac{1}{2}\|\boldsymbol{w}\|^2$的最小值。所以，SVM 的代价函数定义为：

$$\begin{aligned}&\min_{w,b,r} \quad \frac{1}{2}\|\boldsymbol{w}\|^2\\&\text{subject} \quad \text{to} \quad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b)\geqslant 1, i=1,2,\cdots,n\end{aligned} \tag{5-12}$$

在式(5-12)中，只有线性约束，是一个典型的二次规划(Quadratic Programming，QP)问题。以上 QP 问题的最优解为拉格朗日鞍点(Lagrangian's saddle node)，用函数表示为：

$$L(\boldsymbol{w},b,\alpha)=\frac{1}{2}\|\boldsymbol{w}\|^2-\sum_{i=1}^{n}\alpha_i\left[y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b)-1\right] \tag{5-13}$$

其中$\alpha_i\geqslant 0$为拉格朗日乘子(Lagrange multiplier)。对于式(5-13)，可以转换为拉格朗日对偶问题更容易求解，首先固定α，让函数 L 关于 $\boldsymbol{w}$ 和 b 最小化，分别对 $\boldsymbol{w}$ 和 b 求偏导，具体为：

$$\frac{\partial L}{\partial \boldsymbol{w}}=0 \quad \Rightarrow \quad \boldsymbol{w}=\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i$$

$$\frac{\partial L}{\partial b}=0 \quad \Rightarrow \quad \sum_{i=1}^{n}\alpha_i y_i=0$$

将计算结果代入之前的$L(\boldsymbol{w},b,\alpha)$，可以得到：

$$L(\boldsymbol{w},b,\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j) \tag{5-14}$$

由式(5-14)可知，此时的拉格朗日函数只包含一个参数需要优化，也就是α，如果可以求到最优的α，也就可以找到最优的 $\boldsymbol{w}$ 和 b。此时，可以采用序列最小优化(Sequential Minimal Optimization，SMO)算法来优化以上函数，具体为：

$$\begin{aligned}&\max_{\alpha}\left(\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j)\right)\\&\text{subject} \quad \text{to} \quad \alpha_i\geqslant 0, i=1,2,\cdots,n\\&\sum_{i=1}^{n}\alpha_i y_i=0\end{aligned} \tag{5-15}$$

至此仅仅得到了解决线性可分问题的代价函数，然而现实中的很多问题为线性不可分的情况。

例如，图 5.3 为一个典型的线性不可分的实例。但是，如果将图 5.3 中的样本采用一个径向基(Radial Basis Function，RBF)函数映射到三维空间，就可以很容易地实现线性可分，再在二维空间画出决策边界线，如图 5.4 所示。

以上 MATLAB 实现 SVM 基于 RBF 核函数分类的代码如下，首先产生随机样本，画出数据图：

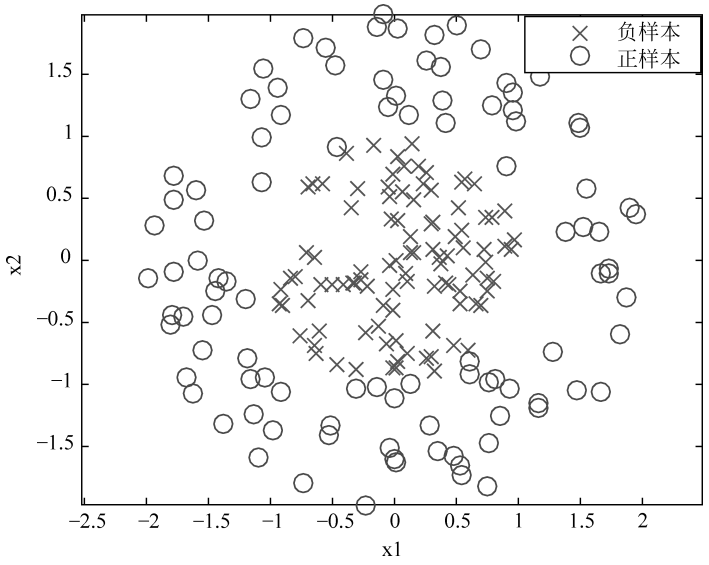


图 5.3　线性不可分样本

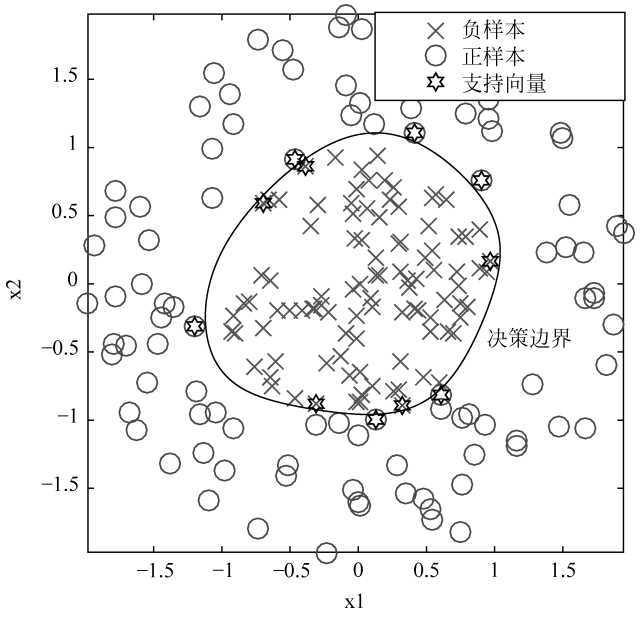


图 5.4　SVM 画出决策边界线

```
rng(1);  r = sqrt(rand(100,1)); t = 2 * pi * rand(100,1);
data1 = [r. * cos(t), r. * sin(t)];
r2 = sqrt(3 * rand(100,1) + 1);     t2 = 2 * pi * rand(100,1);
data2 = [r2. * cos(t2), r2. * sin(t2)];
figure;  plot(data1(:,1),data1(:,2),'rx');
hold on
plot(data2(:,1),data2(:,2),'bo');
xlabel('x1');  ylabel('x2');
```

```
axis equal
hold off
```

训练 SVM 模型，代码如下：

```
data3 = [data1;data2];
theclass = ones(200,1);
theclass(1:100) = -1;
% 训练 SVM 分类器
cl = fitcsvm(data3,theclass,'KernelFunction','rbf',…
    'BoxConstraint',Inf,'ClassNames',[-1,1]);
```

预测分类，代码如下：

```
d = 0.02;
[x1Grid,x2Grid] = meshgrid(min(data3(:,1)):d:max(data3(:,1)),…
    min(data3(:,2)):d:max(data3(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
[~,scores] = predict(cl,xGrid);
```

画分类边界线，代码如下：

```
figure;
h(1:2) = gscatter(data3(:,1),data3(:,2),theclass,'rb','xo');
hold on
h(3) = plot(data3(cl.IsSupportVector,1),data3(cl.IsSupportVector,2),'kh');
contour(x1Grid,x2Grid,reshape(scores(:,2),size(x1Grid)),[0 0],'k');
legend(h,{'-1','+1','Support Vectors'});
xlabel('x1')
ylabel('x2')
text(0.9,-0.4,'Decision boundary')
axis equal
hold off
```

从以上简单分类实例分析可知，假设有一个并不能确定是否线性可分的样本集合，对于 SVM 模型即可以统一定义为优化以下代价函数：

$$\begin{aligned}\min_{w,b,\xi} \quad & \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}+C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}^{T}\phi(\boldsymbol{x}_i)+b)\geqslant 1-\xi_i, \\ & \xi_i\geqslant 0\end{aligned} \tag{5-16}$$

其中 $\boldsymbol{w}$ 为权重矩阵，b 为偏差，与线性回归中的定义相同。其中非负参数 ξ_i 称为松弛变量，允许某些样本点的函数间隔小于 1，即在最大间隔区间中，或者允许某些样本点在对方区域中，即函数间隔为负数。其中非负参数 C 为惩罚因子，是离群点的权重，其值越大则表明离群点对目标函数的影响越大，即越不希望看到离群点。上式中对训练集中的样本 $\boldsymbol{x}_i$，可以通过函数 ϕ 映射到高维空间。可以定义 SVM 的核函数为 $K(\boldsymbol{x}_i,\boldsymbol{x}_j)\equiv\phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j)$，常用的 SVM 核函数有以下 4 种：

（1）线性核：

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j$$

(2) 多项式核：

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=(\gamma \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j+r)^d,\quad \gamma>0$$

(3) 径向基(Radial Basis Function，RBF)核：

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\exp(-\gamma\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2),\quad \gamma>0$$

(4) S(Sigmoid)函数核：

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\tanh(\gamma \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j+r)$$

其中线性核函数没有参数需要设置。多项式核函数需要设置 3 个参数：d 设置多项式核函数的最高次数，γ 参数决定了样本特征映射到高维空间后的分布，r 为偏差。

SVM 的线性核函数，主要用于线性可分的样本数据，其输入和输出具有相同的维度，参数少且速度快，对于未知样本可以先从线性核函数开始测试，误差最小则为最优核函数。SVM 的多项式核函数，可以实现输入到高维空间的映射，参数多，而且当多项式次数较高时，计算复杂度很高。SVM 的 RBF 核函数又称为高斯核函数，可以实现输入到高维，甚至无限维的映射，在实际工程中广泛应用。采用 S 核函数的 SVM，实质上实现了一种多层的神经网络。

在选择 SVM 核函数时，应该充分考虑样本特点，尽量选择与数据分布相同或接近的核函数，可参考原则：若特征的数量和样本数接近，应选线性核 SVM；若特征数量远小于样本数据量，当样本数量正常时，可选择 RBF 核 SVM，当样本数量庞大时，则需要增加特征后再选择。

5.3　支持向量机实例分析

视频讲解

5.3.1　实例一：SVM 解决线性可分问题

采用 SVM 进行线性分类。下载开源 SVM 软件包 LIBSVM，基于软件包的 MATLAB/Octave 接口(选择包描述中“一个简单的 MATLAB 接口”)，实现样本线性 SVM 分类。

1. 问题描述

1) 安装 LIBSVM

下载 LIBSVM 的 MATLAB 接口，根据包中说明文件构建 LIBSVM 源码，具体命令参考软件包中基于 UNIX 和 Windows 的 MATLAB 和 Octave 构建指令。如果 LIBSVM 构建成功，可以得到 4 个后缀为 mexglx(Windows 系统为 mexw32)的文件。有了可以用于 MATLAB/Octave 中运行的二进制码，还需要让本实例源码可以找到该文件。可以有以下 3 种方式添加路径：

(1) 将二进制文件链接到工作目录。

(2) 添加二进制文件的目录到 MATLAB/Octave 的系统路径中。

(3) 直接复制二进制文件到工作目录中。

2) 线性 SVM 分类

回顾 SVM 分类问题的代价函数：

$$\min_{w,b,\xi} \quad \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}+C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to} \quad y_i(\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}_i)+b)\geqslant 1-\xi_i, i=1,2,\cdots,n$$

$$\xi_i\geqslant 0, i=1,2,\cdots,n$$

在求解这个优化问题后，SVM 分类器预测 1 的条件是，$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}+b\geqslant 0$，反之则预测结果为$-1$，所以 SVM 决策的边界服从直线 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}+b=0$。

2. 实例分析参考解决方案

对于二维分类问题，首先考虑两个特征的分类问题。用以下命令，导入本书配套源码中的文件 twofeature.txt 到 MATLAB/Octave 中：

```
[trainlabels, trainfeatures] = libsvmread('twofeature.txt');
```

注意：这个文件被格式化为 LIBSVM 要求的格式，所以如果采用 MATLAB/Octave 常用命令导入将不能正常工作。

导入文件后，向量 trainlabels 应该包含训练数据的分类标记，矩阵 trainfeatures 应该包含每个训练样本的 2 个特征。现在画出数据的二维图，用不同的符号显示正和负，画出的图应该类似图 5.5 的图形。在图 5.5 中，可以看到两类数据之间有一个很明显的分界线间隙。然而，蓝色类中有一个红色点在左边远离的地方出现。现在来观察这个远离边界的红色点对 SVM 决策分界线的影响。设置代价为 $C=1$。在 SVM 优化问题中 C 是一个正的惩罚因子，其作用是惩罚错误分类的训练样本。相对于较小的 C，较大的 C 可以阻止错误分类。首先，用 $C=1$ 执行分类，调用以下语句训练模型：

```
model = svmtrain(trainlabels, trainfeatures, '-s 0 -t 0 -c 1');
```

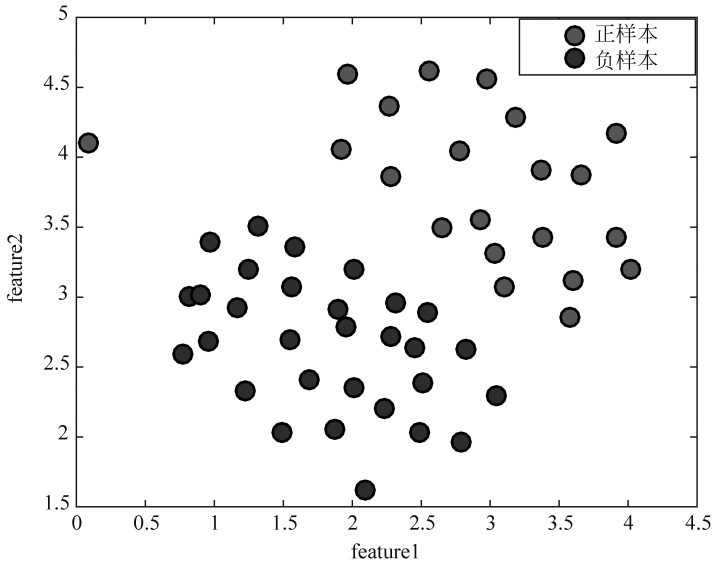


图 5.5　二维样本可视化图（见彩插）

最后一个字符串参数为 LIBSVM 的训练选项。其中“-s 0”表示采用 SVM 分类，其中“-t 0”表示线性核函数，因为本实例希望得到一个线性分界线；其中“-c 1”表示惩罚因子为 1。可以在 MATLAB/Octave 命令行下输入 svmtrain 命令，查看所有相关选项。训练结束后，model 结构体中存储了模型参数。

但是，在模型结构中没有明确地表示 $\boldsymbol{w}$ 和 b 变量，需要通过以下命令来计算获得：

```
w = model.SVs' * model.sv_coef;
b = -model.rho;
if (model.Label(1) == -1)
    w = -w; b = -b;
end
```

在获得 $\boldsymbol{w}$ 和 b 后，可以画出二维分界边界线，结果应该类似图 5.6 所示。

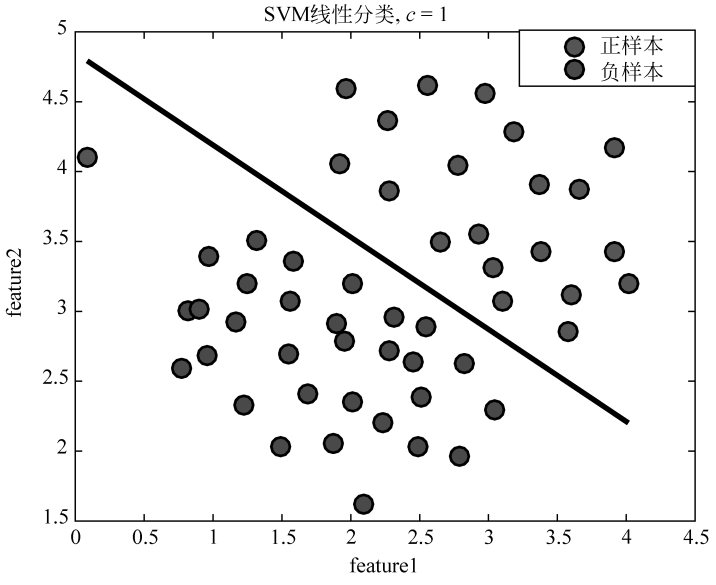


图 5.6　$c=1$ 时由 SVM 获得的决策边界（见彩插）

当 $c=1$ 时，很明显看到那个远离的红色点被错误分类，但是得到了一个看起来很合理的分界线。接下来，设置 $c=100$ 执行分类，观察当惩罚因子非常高的时候，结果会怎样？

训练模型，重新画分界线，这次因为 c 设置为 100，那个远离的点被正确分类了，但是分界线看起来似乎并没有非常完美地符合其他数据。设置 $c=100$ 执行分类，获得决策边界图如图 5.7 所示。

通过这个实例分析可知，当惩罚因子较小时，SVM 算法可以获得较为合理的分界线，但是会出现错误分类的离群点。当惩罚因子变大时，SVM 算法将尽力避免错误分类，但是为了不错误分类所有样本点，算法将减小权重从而导致训练所得的分界线出现较大偏移。

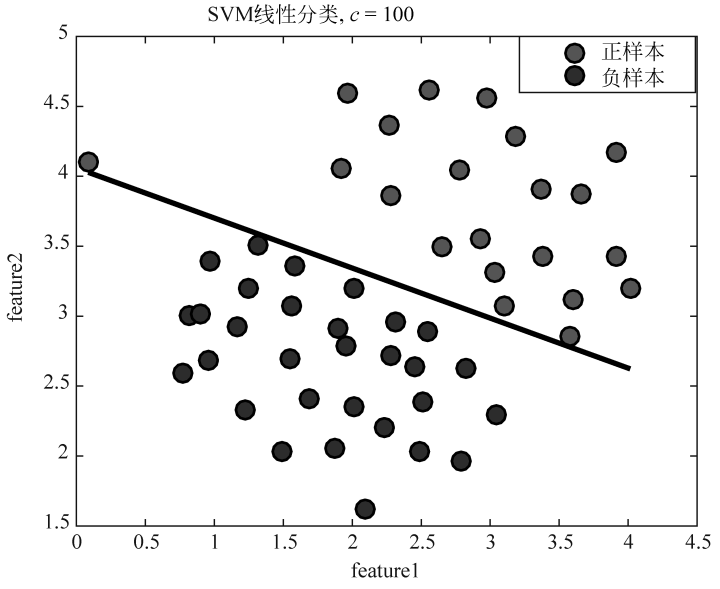


图 5.7　c＝100 时由 SVM 获得的决策边界（见彩插）

5.3.2　实例二：SVM 解决邮件分类问题

回顾第 4 章实例分析中的垃圾邮件分类问题，本实例采用 SVM 来实现，并对比 SVM 与朴素贝叶斯的实验结果。

1. 问题描述

1）数据

在本书配套的源码数据文件夹中，有 4 个与朴素贝叶斯实例相同的训练集，现在只是格式化为 LIBSVM 要求的数据格式，分别命名为：

（1）email_train-50.txt（基于 50 个邮件的文档）。

（2）email_train-100.txt（100 个文档）。

（3）email_train-400.txt（400 个文档）。

（4）email_train-all.txt（完整的 700 个训练文档）。

采用 4 个训练集分别训练 4 个线性 SVM 模型，其中参数 C 采用 SVM 中默认值。在训练完成后，测试每个模型，测试集为 email_test.txt。执行测试的命令为 svmpredict，也可以在 MATLAB/Octave 控制台上输入 svmpredict 命令，查看命令的更多信息。

2）问题

在测试的时候，准确率会被输出到控制台。记录每个训练集的分类准确率，对照解决方案中的测试结果，也可以与朴素贝叶斯的预测错误率进行对比，看看结果如何。

2. 实例分析参考解决方案

对比分类准确率。以下是 LIBSVM 的分类结果报告。

（1）50 个文档：准确率＝75.3846%（196/260）。

（2）100 个文档：准确率＝88.4615%（230/260）。

（3）400 个文档：准确率＝98.0769%（255/260）。

(4) 完整的 700 个训练文档：准确率＝98.4615% (256/260)。

表 5.1 为 SVM 与朴素贝叶斯的分类错误比较。

表 5.1　SVM 与朴素贝叶斯的分类错误比较

训练样本(Train Samples)	朴素贝叶斯	支持向量机(SVM)
50 个训练文档(50 train docs)	2.7%	24.6%
100 个训练文档(100 train docs)	2.3%	11.5%
400 个训练文档(400 train docs)	2.3%	1.9%
700 个训练文档(700 train docs)	1.9%	1.5%

分析表 5.1 中的实验结果，可以得到结论，朴素贝叶斯在较小训练数据集时比 SVM 性能好，但是 SVM 随着训练数据集样本数的增加，表现了更好的逼近特性。

5.3.3　实例三：核函数 SVM 解决线性不可分问题

视频讲解

在本实例中，主要练习带核非线性支持向量机分类。将采用 RBF 核分类线性不可分的数据。与上一个实例相似，可以采用基于 MATLAB/Octave 的 LIBSVM 接口建立一个 SVM 模型。如果未安装 LIBSVM，请参考本章实例一中 readme.txt(见本书配套资源)操作的安装步骤并运行 LIBSVM。

1. 问题描述

1) 数据

首先，导入本实例数据文件，参考本书配套源码中的文件 ex533.txt。

2) 核函数

回顾核函数的定义，基于 SVM 处理线性不可分特征，采用将样本映射到高维空间，这样常常变为线性可分。然而，如何确定特征映射函数 $\phi(x_i)$呢？这个映射函数就称为核函数。一般 SVM 采用较为容易计算的核函数。此外，由于特征映射到高维以后，可能会导致分界边界非常复杂，所以采用简单的核函数可以有效地降低计算复杂度。

3) RBF 核函数

本实例中将采用 LIBSVM 中的径向基函数(RBF)作为核函数。回顾一下，其表达式为：

$$\begin{aligned} K(\boldsymbol{x}_i,\boldsymbol{x}_j) &= \phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j) \\ &= \exp(-\gamma \| \boldsymbol{x}_i - \boldsymbol{x}_j \|^2), \quad \gamma > 0 \end{aligned}$$

注意：当令 $\gamma=\dfrac{1}{2\sigma^2}$时，也称该核函数为高斯 RBF 核函数。其次，直接计算 $\phi(\boldsymbol{x})$是没有意义的，原因在于 $\phi(\boldsymbol{x})$采用 RBF 映射后是无限维，不可能在内存中存储。

4) SVM 采用 RBF 核分类任务一

具体来练习 RBF 核选择一个非线性的分界线。用以下命令将 LIBSVM 格式的数据导入到 MATLAB/Octave 中：

```
[train_labels, train_features] = libsvmread('ex5331.txt');
```

这是一个二维分类问题，如果用不同颜色画出正样本和负样本，可以得到如图 5.8 所示图形。

从图 5.8 可知，这组数据很明显没有线性分界线，观察 RBF 如何自动画出一个非线性

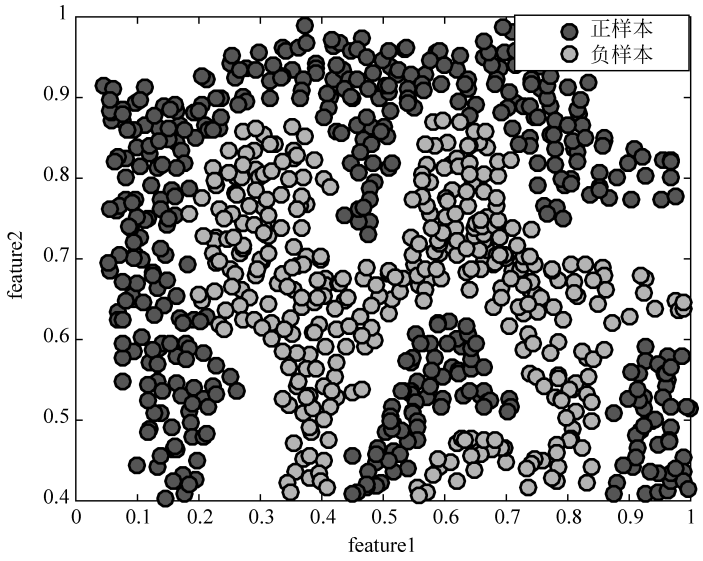


图 5.8　线性不可分样本可视化图（见彩插）

的分界线？用以前实例练习中用过的命令 svmtrain，来训练一个核函数为 RBF 的 SVM 模型，其中设置参数 $\gamma=100$。如果不记得如何使用该命令，也可以在 MATLAB/Octave 控制台中直接输入 svmtrain 查看使用方法。一旦获得模型，就可以用 plotboundary 命令画出可视化的分界线。

```
plotboundary(train_labels, train_features, model);
```

当设置参数 $\gamma=100$，画出分界后的结果如图 5.9 所示。

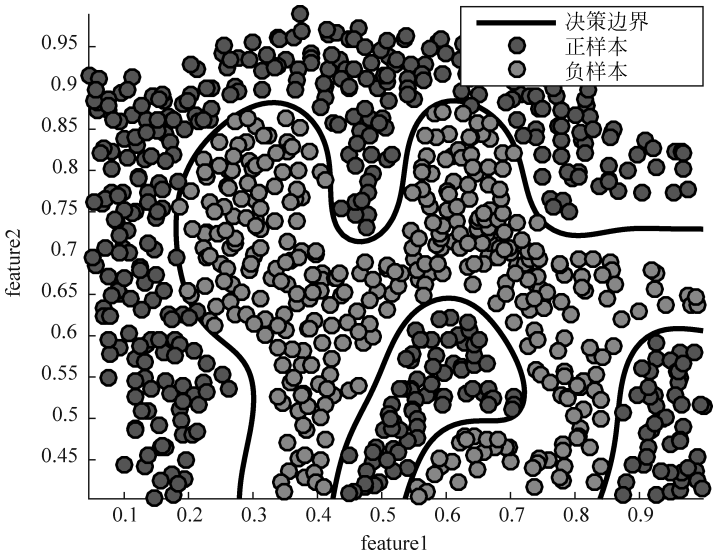


图 5.9　$\gamma=100$ 时带核函数 SVM 画出的决策边界（见彩插）

命令 Plotboundary 包含一个填充颜色的选项函数：$\sum_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$。注意：这个函数给出了用于决策分类的决定值。例如，如果 $\sum_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b \geqslant 0$，则样本 $\boldsymbol{x}$ 被分类

为正类，否则分为负类。

执行命令，可以输入以下代码：

```
plotboundary(train_labels, train_features, model, 't');
```

结果图形输出如图 5.10 所示。从图 5.10 可以直观地看到，白色区域为更高可信度的正类样本，黑色区域为更高可信度的负类样本，颜色梯度显示了对样本分类值的决策度变化。

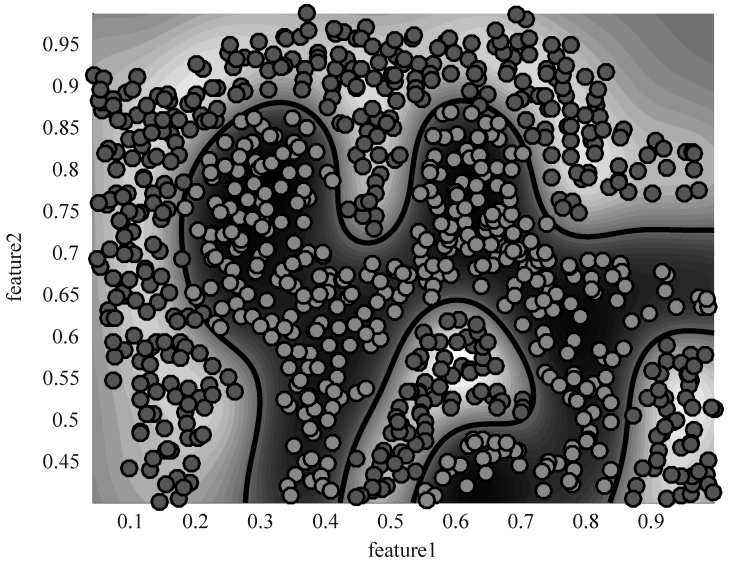


图 5.10　$\gamma=100$ 时带核函数 SVM 画出的填充颜色后的决策边界（见彩插）

5）SVM 采用 RBF 核分类任务二

通过本实例练习来观察以下 RBF 核函数中的参数 γ 对分界线的影响。导入数据文件 ex5332.txt 并画图，可以得到如图 5.11 所示图形。

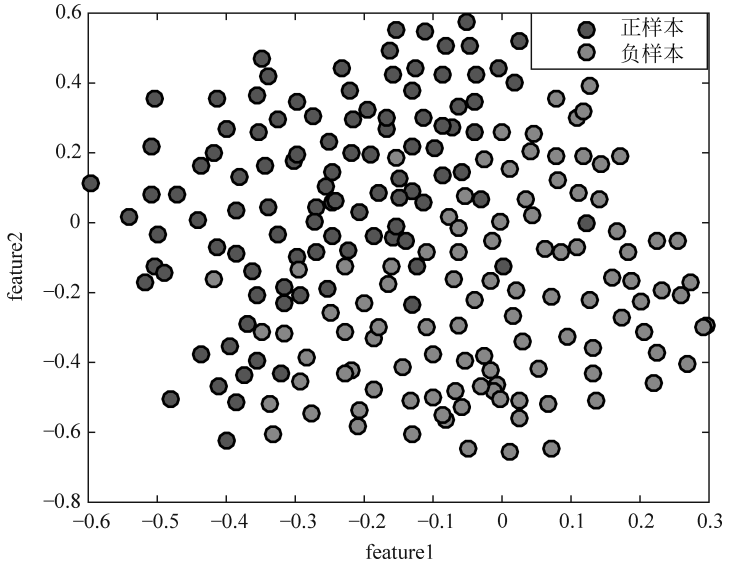


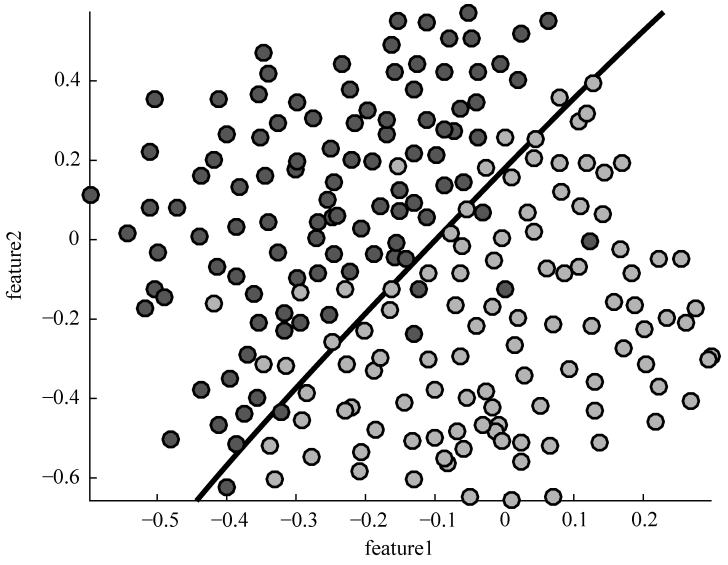
图 5.11　线性不可分样本可视化图（见彩插）

本任务要求：设置 γ 不同值 1、10、100 和 1000 训练 SVM 模型，并分别画出每个模型的分界线(不用颜色填充)。观察分界线如何跟随 γ 值的变化而变化。

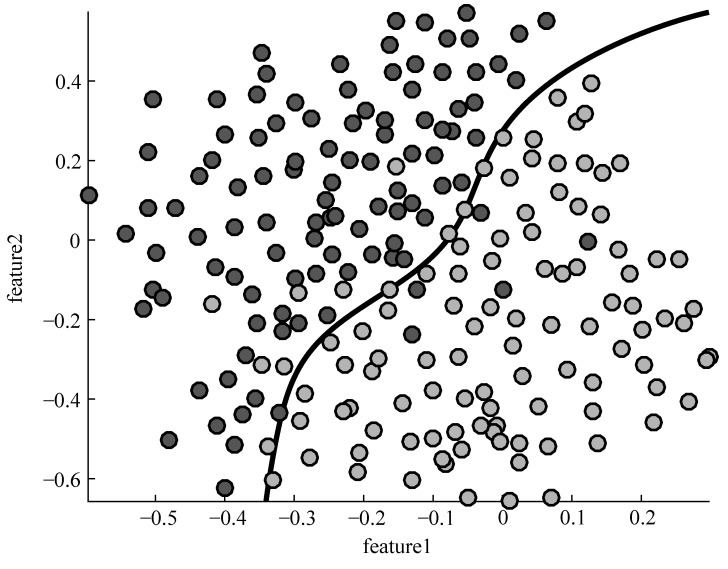
2. 实例分析参考解决方案

图 5.12 为每个模型的分界线。模型的分类准确率分别为 91.9%、93.3%、94.3% 和 100%。

由图 5.12 可知，随着 γ 增加，算法尽量避免错误分类数据，然后会导致过拟合。目前，如何在不过拟合的情况下，合适地设置 γ 值并没有很明确的方法。在实际工程应用中，一般采用经验设置初值，然后调试逼近最优参数。

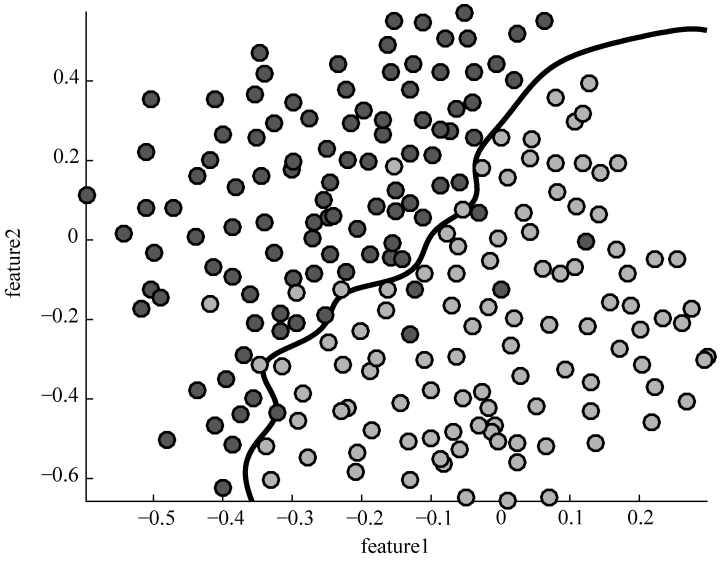


(a) $\gamma = 1$

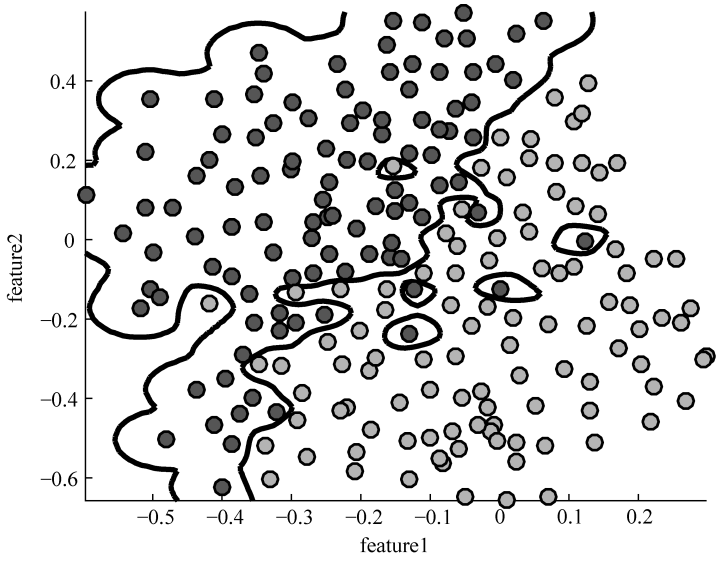


(b) $\gamma = 10$

图 5.12　不同的参数 γ 对应不同的模型分界线(见彩插)



(c) $\gamma = 100$



(d) $\gamma = 1000$

图 5.12 （续）

5.4　习题

1. 请比较支持向量机(SVM)与逻辑回归(LR)的优缺点，并考虑当一个能被正确分类，但是远离决策边界的样本点加入训练集时，SVM 和 LR 的表现。

2. 基于 Python 机器学习库 Sklearn，完成以下任务：

(1) 生成两类二维的带标签的数据集，数据集大小为 200，并为每个类分配一个或多个正态分布的点集(函数 make_blobs)，最后画出二维数据的散点图(函数 pyplot)；

(2) 采用 Sklearn 库中的 svm 函数，对以上生成的数据集实现分类(sklearn. svm. SVC)，并画出决策分界线(函数 pcolormesh)。