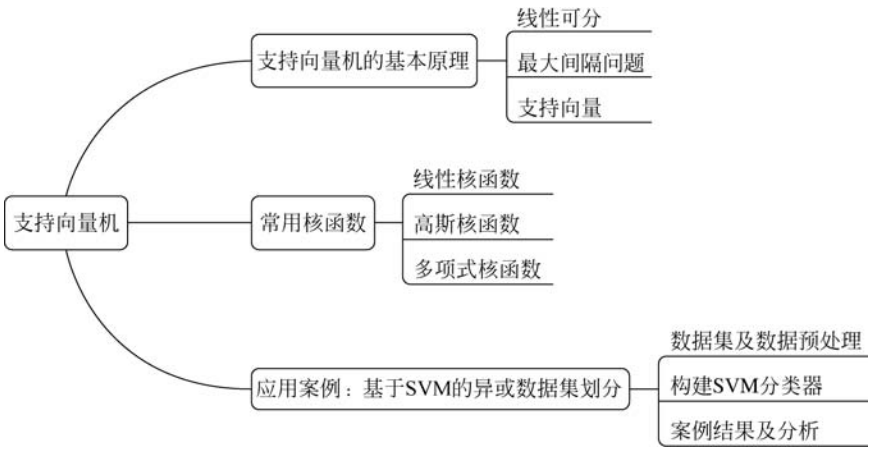
第5章

支持向量机

[思维导图]



支持向量机(Support Vector Machine，SVM)是一类按监督学习(Supervised Learning)方式对数据进行二元分类的广义线性分类器(Generalized Linear Classifier)。本章主要介绍SVM的相关概念、算法实现和实践应用。

5.1 支持向量机的基本原理

5.1.1 线性可分

要了解支持向量机，首先要了解什么是线性可分。在一维空间中，也就是在一个坐标

轴上，要分开两个可分的点集合，只需要找到一个点即可，图 5-1 展示了一维空间的线性可分。

图 5-1　一维空间的线性可分示意图

在二维空间中，要分开两个线性可分的点集合，需要找到一条分类直线，图 5-2 展示了二维空间的线性可分。

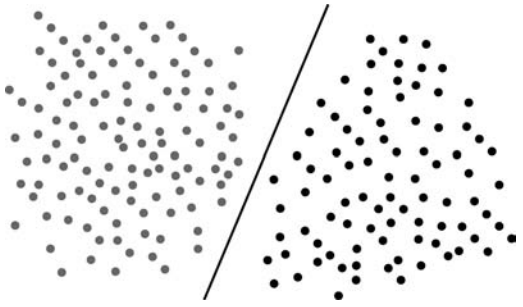


图 5-2　二维空间的线性可分示意图

在三维空间中，要分开两个线性可分的点集合，需要找到一个分类面，图 5-3 展示了三维空间的线性可分。

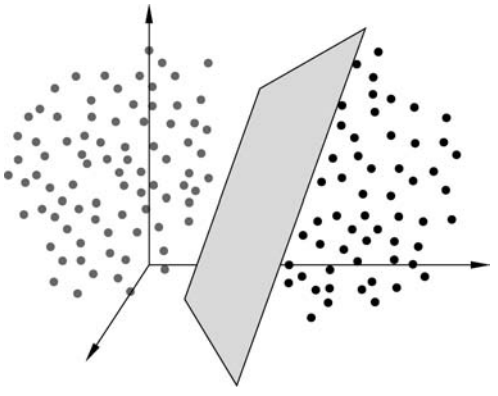


图 5-3　三维空间的线性可分示意图

在 n 维空间中，要分开两个线性可分的点集合，则需要找到一个超平面（Hyper Plane）。以二维空间为例，如式(5.1)所示，假设给定训练样本集 D：

$$D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\},y_i\in\{-1,+1\}\quad i=1,2,\cdots,m \tag{5.1}$$

分类学习最基本的思想就是基于训练集 D 在样本空间中找到一个划分超平面，将不同类别的样本分开。如图 5-4 所示，存在多个划分超平面将两类训练样本分开。直观上看，应该寻找位于两类训练样本“正中间”的划分超平面，如图 5-4 所示的加粗划分线。该划分线即为超平面，对训练样本局部扰动的“容忍”性最好。例如，由于训练集的局限性或噪声的因素，训练集外的样本可能比图 5-4 中的训练样本更接近两个类的分隔边界，这将会造成许多划分超平面出现错误，而超平面受影响最小。换言之，这个划分超平面所产生

的分类结果是最健壮的，对未见实例的泛化能力最强。

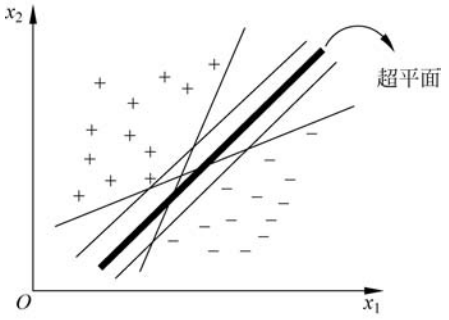


图 5-4　存在多个划分超平面将两类训练样本分开

在样本空间中，划分超平面可通过线性方程来描述，如式(5.2)所示。

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=0 \tag{5.2}$$

其中，$\boldsymbol{w}=(w_1,w_2,\cdots,w_d)$为法向量，决定了超平面的方向；$b$ 为位移项，决定了超平面与原点之间的距离。显然，划分超平面可由法向量 $\boldsymbol{w}$ 和位移 b 确定，将其记为$(\boldsymbol{w},b)$，如式(5.3)所示。样本主空间中任意点 x 到超平面$(\boldsymbol{w},b)$的距离 r 可表示为：

$$r=\frac{|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b|}{\|\boldsymbol{w}\|} \tag{5.3}$$

假设超平面$(\boldsymbol{w},b)$能将训练样本正确分类，即对于$(\boldsymbol{x},y_i)\in D$，若 $y_i=1$，则有 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b>0$；若 $y_i=-1$，则有 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b<0$。令

$$\begin{cases}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\geqslant 1, & y_i=1\\ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\leqslant -1, & y_i=-1\end{cases} \tag{5.4}$$

5.1.2　最大间隔问题

(1) 如何计算点到超平面的距离。

支持向量与间隔如图 5-5 所示，距离超平面最近的这几个训练样本点使式(5.4)中的等号成立，称其为“支持向量”(Support Vector)，如式(5.5)所示，两个异类支持向量到超

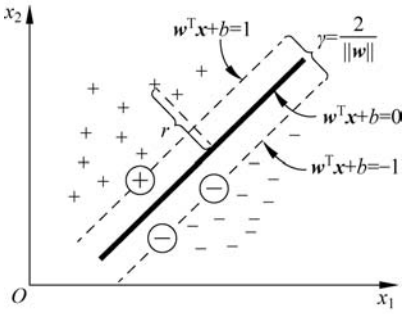


图 5-5　支持向量与间隔

平面的距离之和 γ 可以表示为：

$$\gamma = \frac{2}{\| \boldsymbol{w} \|} \tag{5.5}$$

称其为“间隔”(Margin)，如图 5-5 所示。

想要找到具有“最大间隔”(Maximum Margin)的划分超平面，也就是要找到能满足式(5.4)中约束的参数 $\boldsymbol{w}$ 和 b，使得 γ 最大，如式(5.6)所示。

$$\begin{aligned} &\max_{\boldsymbol{w},b} \frac{2}{\| \boldsymbol{w} \|} \\ &\text{s.t.} \quad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b) \geqslant 1, \quad i = 1,2,\cdots,m \end{aligned} \tag{5.6}$$

(2) 模型表示。

为了最大化间隔，仅需最大化 $\| \boldsymbol{w} \|^{-1}$，这等价于最小化 $\| \boldsymbol{w} \|^{2}$。于是，式(5.6)可重写为式(5.7)。

$$\begin{aligned} &\min_{\boldsymbol{w},b} \frac{1}{2} \| \boldsymbol{w} \|^{2} \\ &\text{s.t.} \quad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geqslant 1, \quad i = 1,2,\cdots,m \end{aligned} \tag{5.7}$$

这就是支持向量机的基本模型。

5.1.3　支持向量

样本中距离超平面最近的点被称为支持向量，如图 5-6 所示。

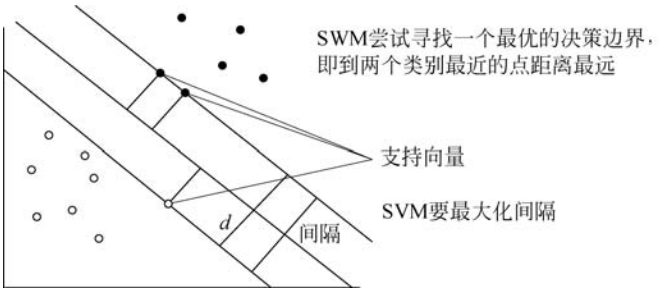


图 5-6　支持向量

5.2　常用核函数

支持向量机在机器学习领域享有较高知名度的一个主要原因是该方法易于通过核化来解决非线性分类的复杂问题，即不能使用线性超平面作为决策边界来区分样本的类别的问题。该方法的逻辑是针对线性不可分数据，建立非线性组合，通过映射函数 ϕ 把原始特征投影到一个高维空间，特征在该空间变得线性可分。原始特征被映射到高维空间使之变得线性可分，如图 5-7 所示。可以将一个二维数据集转换为一个新的在三维空间上表示的特征，通过线性超平面可以将图 5-7 中所示的两类分开，如果把它投射到原来的特征空间上，将形成非线性边界。

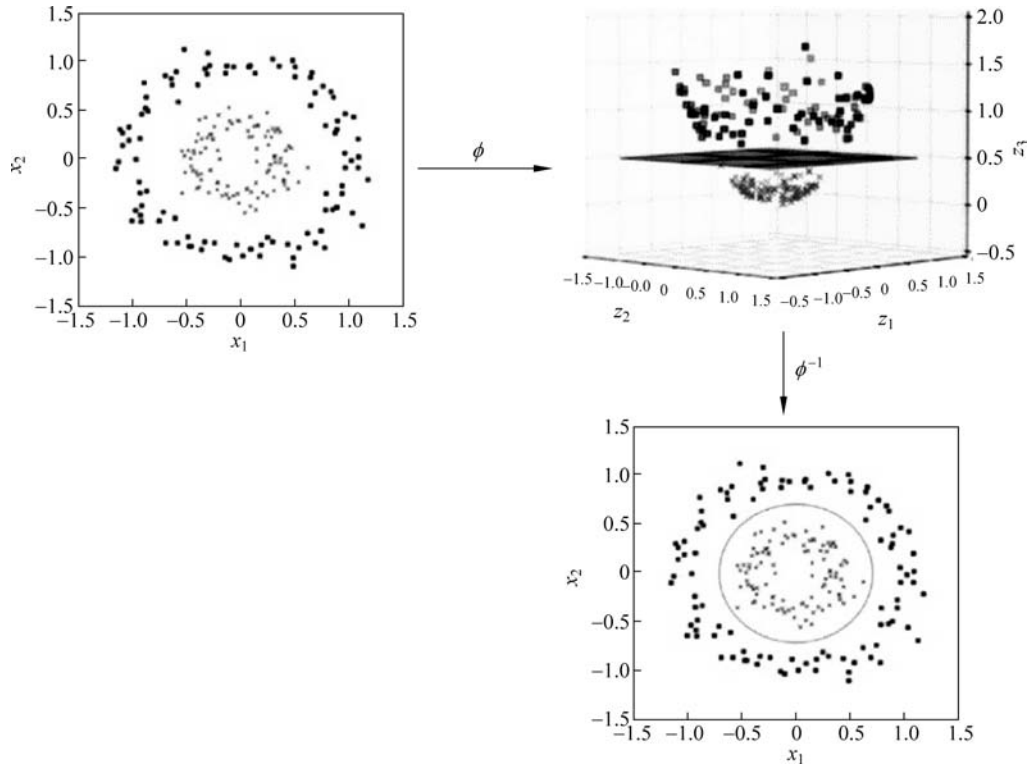


图 5-7　原始特征被映射到高维空间使之变得线性可分

然而，这种映射方法的问题是构建特征的计算成本非常高，特别是在处理高维数据时尤为明显。因此，提出核函数方法来减少高维数据计算问题。实际上，核函数方法中只需要用 $\phi((\boldsymbol{x}^{(i)})^{\mathrm{T}}) \cdot \phi(\boldsymbol{x}^{(j)})$ 替换 $\boldsymbol{x}^{(i)\mathrm{T}} \cdot \boldsymbol{x}^{(j)}$，可大大减少高维空间上的计算成本，如式(5.8)所示。

$$K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \phi((\boldsymbol{x}^{(i)})^{\mathrm{T}}) \cdot \phi(\boldsymbol{x}^{(j)}) \tag{5.8}$$

以下介绍三种 SVM 常用的核函数，包括线性核函数、高斯核函数以及多项式核函数。

5.2.1　线性核函数

线性核函数主要用于线性可分的情况，它的特征空间到输入空间的维度是一样的，其参数少、速度快，具体形式如式(5.9)所示。

$$K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = (\boldsymbol{x}^{(i)})^{\mathrm{T}} \cdot \boldsymbol{x}^{(j)} \tag{5.9}$$

对于线性可分数据，其分类效果很理想，因此通常首先尝试用线性核函数来做分类。

5.2.2　高斯核函数

高斯核又称径向基函数(Radial Basis Function，RBF)，就是某种沿径向对称的标量函数。通常定义为空间中任一点 $\boldsymbol{x}^{(i)}$ 到某一中心 $\boldsymbol{x}^{(j)}$ 之间欧氏距离的单调函数，其作用往往是局部的，即当 $\boldsymbol{x}^{(j)}$ 远离 $\boldsymbol{x}^{(i)}$ 时函数取值很小。高斯核函数的定义如式(5.10)所示。

$$K(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)})=\exp\left(-\frac{\|\boldsymbol{x}^{(i)}-\boldsymbol{x}^{(j)}\|^2}{2\sigma^2}\right) \tag{5.10}$$

该公式常被化简为式(5.11)。

$$K(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)})=\exp(-\gamma\|\boldsymbol{x}^{(i)}-\boldsymbol{x}^{(j)}\|^2) \tag{5.11}$$

其中，$\gamma=\frac{1}{2\sigma^2}$为需要优化的自由参数。

高斯核函数是一种局部性强的核函数，其可以将一个样本映射到一个更高维的空间内，该核函数是应用最广的一个，无论是大样本还是小样本都有比较好的性能，而且其相对于多项式核函数参数要少，因此大多数情况下，在不知道用什么核函数时，优先使用高斯核函数。

5.2.3　多项式核函数

多项式核函数是一种非标准核函数，它非常适合于正交归一化后的数据，其具体形式如式(5.12)所示。

$$K(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)})=(((\boldsymbol{x}^{(i)})^{\mathrm{T}}\cdot\boldsymbol{x}^{(j)})+1)^d \tag{5.12}$$

多项式核函数可以实现将低维的输入空间映射到高维的特征空间，但是多项式核函数的参数多，当多项式的阶数比较高的时候，核矩阵的元素值将趋于无穷大或者无穷小，计算复杂度会大到无法计算，但是计算结果还算稳定。

5.3　应用案例：基于 SVM 的异或数据集划分

5.3.1　数据集及数据预处理

异或逻辑就是同 1 异 0，而异或数据集顾名思义就是将二维坐标中的数据通过异或的逻辑方式划分为两类数据集。在二维坐标系下随机生成一些点，这些点作为数据总集合，点的数量就是数据集的数量，然后根据每个点的 x 坐标和 y 坐标的关系，对 x 坐标和 y 坐标做异或运算，得到值为 1 的点划分为一类，得到的值为 0 的点划分为另一类，这样得到的数据集就是需要的异或数据集。异或数据集是一类比较经典的非线性数据集。

创建一个简单异或数据集的代码如例 5-1 所示，调用 NumPy 的 logical_xor()函数形成一个异或门，其中将 100 个样本的分类标签设为 1，100 个样本的分类标签为－1。

【例 5-1】　创建一个异或数据集。

```
1.  import matplotlib.pyplot as plt
2.  import numpy as np
3.  np.random.seed(1)
4.  X_xor = np.random.randn(200, 2)
5.  y_xor = np.logical_xor(X_xor[:, 0] > 0,X_xor[:, 1] > 0)
6.  y_xor = np.where(y_xor, 1, -1)
7.  plt.scatter(X_xor[y_xor == 1, 0],X_xor[y_xor == 1, 1],c = 'b', marker = 'x',label = '1')
8.  plt.scatter(X_xor[y_xor == -1, 0],X_xor[y_xor == -1, 1],c = 'r',marker = 's', label = '-1')
9.  plt.xlim([-3, 3])
```

```
plt.ylim([-3, 3])
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```

执行上述代码会产生具有随机噪声的 XOR 数据集，如图 5-8 所示。

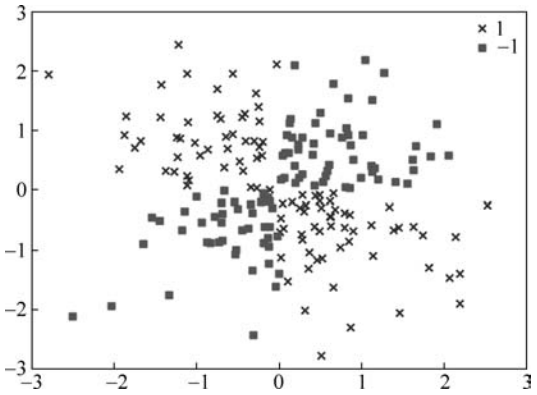


图 5-8　具有随机噪声的 XOR 数据集

显然，异或数据集并不能产生线性超平面作为决策边界来分隔样本的正类和负类，在后面的实例中，将会利用支持向量机的核方法解决异或数据集的分类问题。

5.3.2　构建 SVM 分类器

核支持向量机解决非线性数据分类问题的核心就是通过映射函数 ϕ 将样本的原始特征映射到一个使样本线性可分的更高维的空间中。SVM 算法的原理就是找到一个分割超平面，它能把数据正确地分类，并且间距最大。这里要实现的就是训练通过核支持向量机对非线性可分的异或数据集划分决策边界。

首先定义 plot_decision_regions()函数绘制分类器的模型决策区域，并通过可视化的方法展示划分的效果，划分决策区域的代码如例 5-2 所示。

【例 5-2】　划分决策区域的代码。

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.colors import ListedColormap
def plot_decision_regions(x,y,model,resolution=0.02):
    markers = ('s','x','o','^','v')
    colors = ('red','blue','lightgreen','gray','cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])
    x1_min,x1_max = x[:,0].min() - 1,x[:,0].max() + 1
    x2_min,x2_max = x[:,1].min() - 1,x[:,1].max() + 1
    xx1,xx2 = np.meshgrid(np.arange(x1_min,x1_max,resolution),
                np.arange(x2_min,x2_max,resolution))
    z = model.predict(np.array([xx1.ravel(),xx2.ravel()]).T)
```

```
    z = z.reshape(xx1.shape)
    plt.contourf(xx1,xx2,z,alpha = 0.4,cmap = cmap)
    plt.xlim(xx1.min(),xx1.max())
    plt.ylim(xx2.min(),xx2.max())
    for idx,cl in enumerate(np.unique(y)):
      plt.scatter(x = x[y == cl,0],y = x[y == cl,1],
              alpha = 0.8,c = cmap(idx),
              marker = markers[idx],label = cl)
    plt.xlabel("x1")
    plt.ylabel("x2")
    plt.show()
```

上述代码定义颜色和标记并通过 ListedColormap()函数来从颜色列表创建色度图。然后通过 NumPy 的 meshgrid()函数创建网格阵列 xx1 和 xx2，利用特征向量确定特征的最小值和最大值，通过模拟足够多的数据绘制出决策边界。调用 predict()函数来预测相应网格点的分类标签 z，在把分类标签 z 改造成与 xx1 和 xx2 相同维数的网格后，调用 Matplotlib 的 contourf()函数画出轮廓图。

求异或数据集决策边界的代码如例 5-3 所示。

【例 5-3】 导入 sklearn 库中的 SVC 类，求出异或数据集的决策边界。

```
if __name__ == "__main__":
    x_xor = np.random.randn(200,2)
    #将数据集变成一个异或的数据集
    y_xor = np.logical_xor(x_xor[:,0] > 0,x_xor[:,1] > 0)
    y_xor = np.where(y_xor,1,-1)
    svm = SVC(kernel = "rbf", random_state = 0, gamma = 0.1, C = 1.0)
    svm.fit(x_xor, y_xor)
    plot_decision_regions(x_xor, y_xor, svm)
```

其中，参数 kernel＝"rbf"表示使用的核函数为高斯核函数，γ 的值设置为 0.1，也就是高斯球的截至参数，增大 γ 的值，将加大训练样本的影响范围，导致决策边界紧缩或波动。

如果使用多项式核函数，可以将 svm＝SVC(kernel＝"rbf", random_state＝0, gamma＝0.1, C＝1.0)这一行代码替换为 svm＝SVC(kernel＝'poly',degree＝2,gamma＝1,coef0＝0)。其中，参数 degree 只对多项式核函数有用，是指多项式核函数的阶数 d，gamma 为核函数系数，coef0 是指多项式核函数中的常数项。可以看到多项式核函数需要调节的参数是比较多的。

5.3.3　案例结果及分析

使用径向基核函数，γ 设置为 0.1 时，异或数据集的分类结果如图 5-9 所示。

使用高斯核函数，γ 设置为 10 时，异或数据集的分类结果如图 5-10 所示。

使用高斯核函数，γ 设置为 100 时，异或数据集的分类结果如图 5-11 所示。

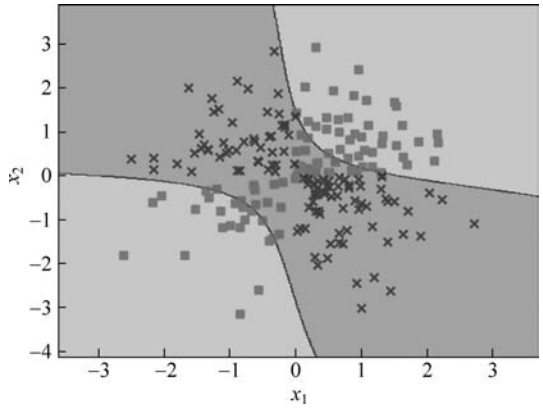


图 5-9　使用高斯核函数，$\gamma=0.1$ 时的分类结果

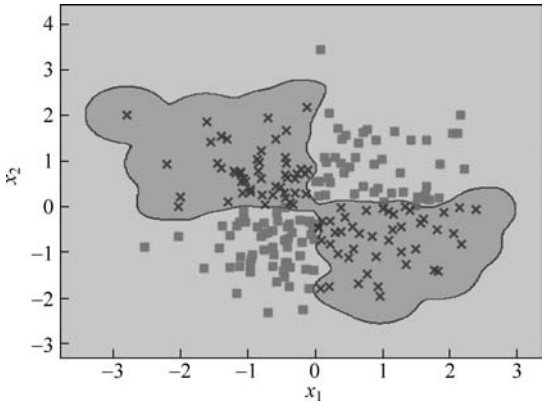


图 5-10　使用高斯核函数，$\gamma=10$ 时的分类结果

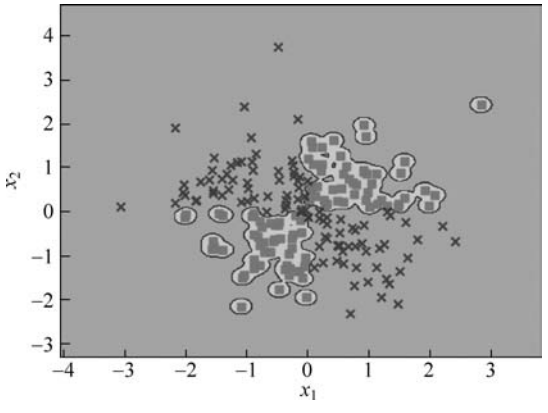


图 5-11　使用高斯核函数，$\gamma=100$ 时的分类结果

由于多项式核函数的参数比较多，在使用多项式核函数时，将参数 degree 固定为 2，参数 coef0 的值固定为 0，通过变换 γ 的值来观察可视化结果的变化，γ 设置为 0.1 的时候，异或数据集的分类结果如图 5-12 所示。

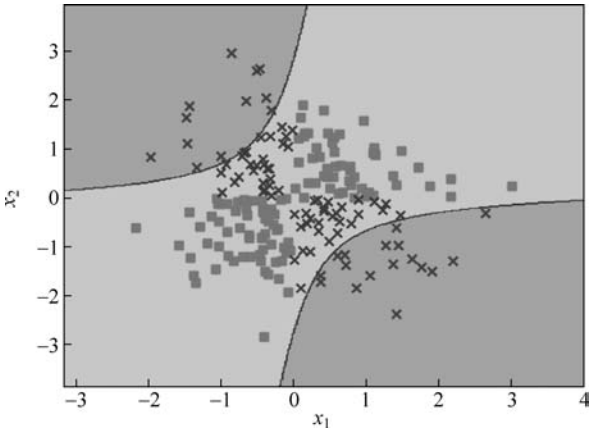


图 5-12　使用多项式核函数，$\gamma=0.1$ 时的分类结果

使用多项式核函数，γ 设置为 1 时，异或数据集的分类结果如图 5-13 所示。

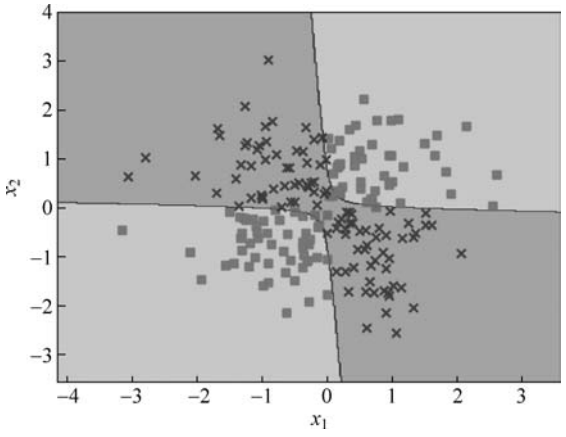


图 5-13　使用多项式核函数，$\gamma=1$ 时的分类结果

使用多项式核函数，γ 设置为 100 时，异或数据集的分类结果如图 5-14 所示。

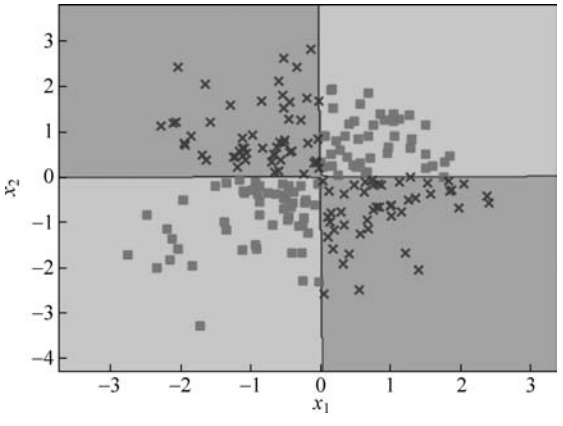


图 5-14　使用多项式核函数，$\gamma=100$ 时的分类结果

由上述结果可以发现，γ 的值比较小时，不同类别的决策边界比较宽松，γ 值比较大时，不同类别的决策边界比较紧实，在使用高斯核函数时，随着 γ 值的增大出现了过拟合的现象，说明 γ 在控制过拟合问题上也可以起到比较重要的作用。

对于两种核函数的不同决策结果，可以看到，两种核函数对于非线性分类的异或数据集的划分都有比较好的效果，而高斯核函数由于参数比较少且分类结果比较稳定，因此在解决此类问题上可以优先选择高斯核函数。使用多项式核函数训练决策边界时，相同参数出现的结果可能会略有差异，这里设置多项式阶数为 2，值比较小，训练速度比较快，如果阶数比较大，计算量会显著增加，将会使得训练时间比较长，这也是多项式核函数相对于高斯核函数的一个小缺陷。