2.4 支持向量机

最小距离的最大化

上一节介绍的对数几率回归模型是一个线性二分类模型，本节将要介绍的支持向量机（Support Vector Machine, SVM）也是一个类似的二分类模型，那么两者主要有什么不同呢？笔者认为可以从以下两点进行把握：第一，对数几率回归采用极大似然估计，仅累加分类正确样本的概率值；而 SVM 则基于“最大间隔”思想，会同时考虑分类正确或错误两类样本。第二，对数几率回归的决策面由所有训练样本决定，而 SVM 的决策面则仅由少量训练样本（称为“支持向量”）决定。本节将以这两点作为基本线索，来对 SVM 的原理进行介绍。然后，基于原理进行代码实现。

实际上，正如 1.3 节所介绍的，SVM 与统计学习理论的提出紧密相关，基于 SVM 发展出来的“核方法”影响了众多的机器学习模型（比如核化对率回归、核化 PCA、核化 LDA 等）。对于“核方法”，本节也会作一些基本介绍。

2.4.1 二分类与决策面

如图 2.4.1 所示，决策面$y = \boldsymbol{w}^T\boldsymbol{x} + b = 0$将样本（图中未画出）分为两个类别：对应$y > 0$一边的“正类”和对应$y < 0$一边的“负类”。向量$\boldsymbol{w}$是决策面的法向量，垂直于决策面（习题 2.4.1）。由点到直线的距离公式，可得原点到决策面的距离为$|b|/\|\boldsymbol{w}\|$。而任意样本向量$\boldsymbol{x}$到决策面的距离为$\left|\boldsymbol{w}^T\boldsymbol{x} + b\right|/\|\boldsymbol{w}\|$，如果去掉绝对值，则得到带符号距离$(\boldsymbol{w}^T\boldsymbol{x} + b)/\|\boldsymbol{w}\|$（$y > 0$一边为正，$y < 0$一边为负）。

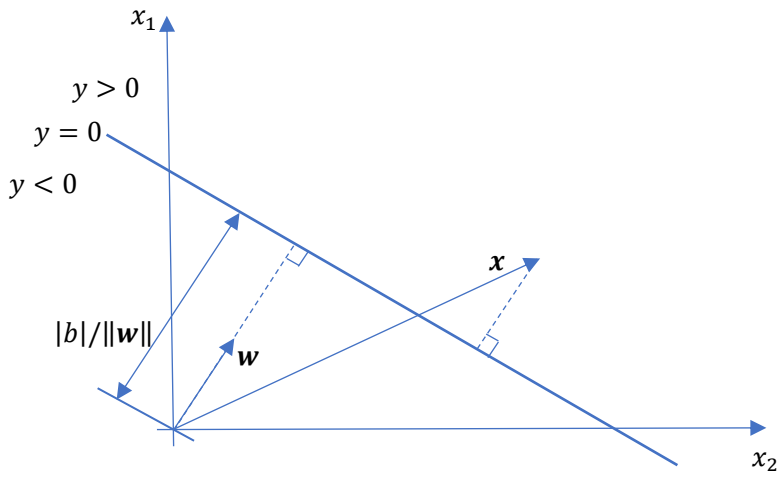

图 2.4.1 二分类与决策面

再来看图 2.4.2，该图画出了两类样本点：空心圆表示“正类”样本点，而实心圆表示“负类”样本点。该图中，除了与图 2.4.1 一致的决策面$y = 0$之外，还画出了另外 4 个用虚线表示的决策面。显然，这些决策面都能将两类样本点完全分开，从这个意义上讲这些决策面没有好坏之分，都一样好。实际上，对于图 2.4.2，类似的一样好的决策面有无穷多个（读

者可以试试画出更多）。那么，这些决策面真的一样好吗？这就引出了 SVM 的“最大间隔”思想，也正由于此 SVM 也被称为“最大间隔分类器”。

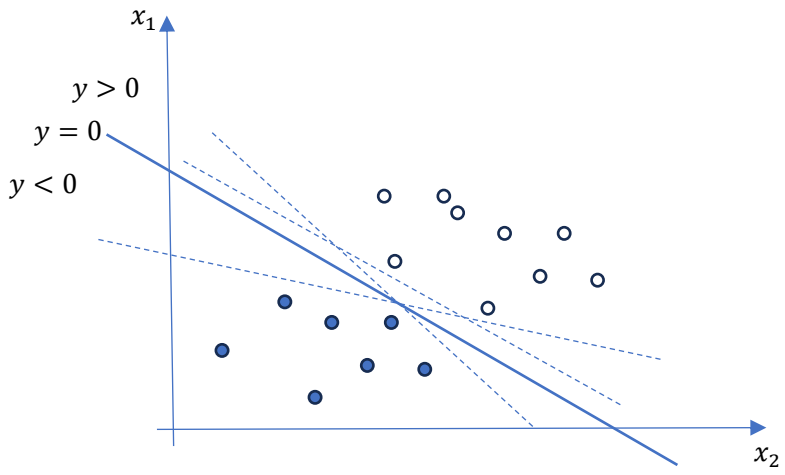

图 2.4.2 应该选择哪一个决策面？

2.4.2 最大间隔分类器

首先定义“间隔（Margin）”这个概念。所谓间隔，指的是决策面与全部样本点之间的最小距离。如图 2.4.3 所示，对于所有样本点，决策面$y = 0$（图中粗实线）与负类样本点n_1之间的距离（图中细实线）最小，这个最小距离就被称为间隔。显然，不同的决策面得到的间隔是不同的。比如图 2.4.3 所示的长虚线决策面，其与正类样本点p_1之间的距离（图中短虚线）最小，即为其间隔。显然，此间隔比粗实线决策面的间隔还更小。

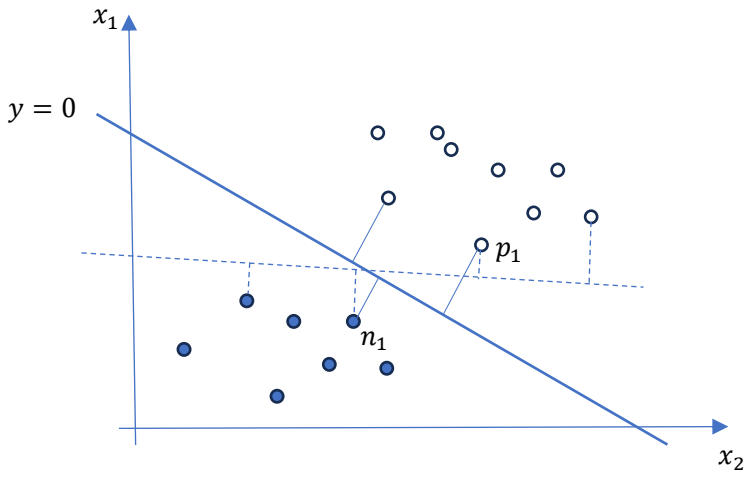

图 2.4.3 间隔

由此，SVM 的“最大间隔（Max Margin）”思想就很清楚了：找到具有最大间隔的决策面。如图 2.4.4 所示，在所有能将两类样本完全分开的决策面中，具有最大间隔的决策面只有一个，如图中粗实线所示。与图 2.4.3 中的两个决策面相比较，间隔取到了最大，这样就不仅能将两类样本完全分开，而且使两类样本分得最开。读者不妨自己再多验证几个决策面，以帮助理解“最大间隔决策面”这个重要思想。

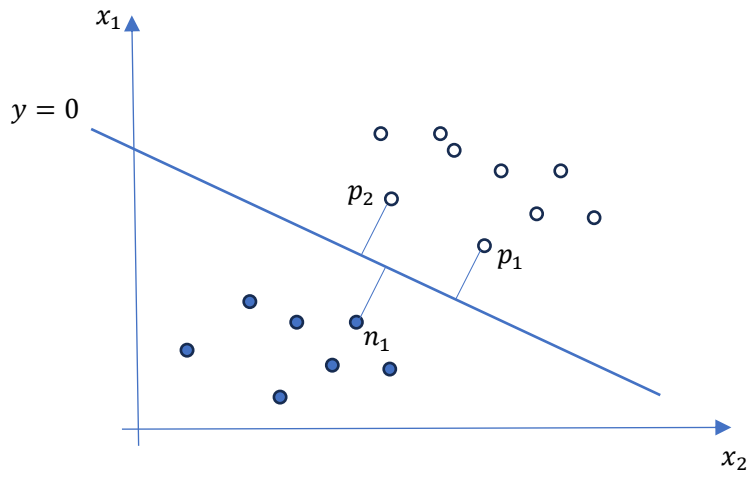

图 2.4.4 最大间隔与最大间隔决策面

观察图 2.4.4，还有两点值得重点注意。第一点，最大间隔决策面到正负样本的最小距离（即间隔）一定是一样的。因为如果不一样，则间隔并未取到最大，矛盾。第二点，决定最大间隔决策面的是少量的距离决策面最近的“关键”正负样本，称之为“支持向量（Support Vector, SV）”，这正是 SVM 名称的由来。比如，图 2.4.4 中有三个 SV：正类的p_1和p_2两个样本点，负类的样本点n_1。

一般地，可以将决策面写为式(2.4.1)，其中引入了特征变换函数$\varphi(\boldsymbol{x})$。特征变换有什么好处呢？回顾 1.2.1 节谈到的两阶段学习范式：“特征提取”+“模型训练”。好的“特征提取”能够得到好的“特征表示”，从而简化问题。举个简单的例子，直角坐标下单位圆的方程表示为$x^2+y^2=1$，是一个非线性函数（二次函数）。如果引入直角坐标到极坐标的特征变换（$x=rcosa, y=rsina$），则极坐标下单位圆的方程表示为$r=1$，简化为了一个常值函数。这个特征变换（$x=rcosa, y=rsina$）就是一个好的“特征提取”，能够得到好的“特征表示”。

$$y=\boldsymbol{w}^T\varphi(\boldsymbol{x})+b \tag{2.4.1}$$

用$t_n\epsilon\{-1,1\}, n=1,\dots,N$表示训练样本的类别标签（$-1$表示负类，1表示正类），$N$为总的训练样本数。假设这些训练样本在特征空间$\varphi(\boldsymbol{x})$里线性可分，则有$t_n y(\boldsymbol{x}_n)\geq 0$（习题 2.4.2）。比如图 2.4.4 中，特征空间$\varphi(\boldsymbol{x})=\boldsymbol{x}$，决策面$y=0$将两类样本完全正确分开，满足$t_n y(\boldsymbol{x}_n)\geq 0$。

对于特征空间$\varphi(\boldsymbol{x})$里线性可分的训练集，任意样本点$\boldsymbol{x}_n$到决策面的距离为：

$$\frac{t_n y(\boldsymbol{x}_n)}{\|\boldsymbol{w}\|}=\frac{t_n(\boldsymbol{w}^T\phi(\boldsymbol{x}_n)+b)}{\|\boldsymbol{w}\|} \tag{2.4.2}$$

那么，SVM 的最优化目标可表示为：

$$argmax_{\boldsymbol{w},b}\left\{\frac{1}{\|\boldsymbol{w}\|}\underset{\boldsymbol{n}}{min}[t_n(\boldsymbol{w}^T\phi(\boldsymbol{x}_n)+b)]\right\} \tag{2.4.3}$$

这个式子分两层来看。首先，大括号里面的式子表示在N个训练样本中，找出到决策面（由$\boldsymbol{w}$和b确定）最近的距离值，这就是前面所说的“间隔”。注意因子$\frac{1}{\|\boldsymbol{w}\|}$与样本点$\boldsymbol{x}_n$无关，可以提到$min$运算的外面。其次，大括号外面，则要从所有决策面中找到使得“间隔”最大的决策面（记为$\boldsymbol{w}^*$和b^*），即“最大间隔决策面”。式(2.4.3)就是 SVM 最优化问题的标准提法：找到“最大间隔决策面”。

2.4.3 最优化问题的转化

式(2.4.3)难以求解，需要转化成更容易求解的等价问题。为此，注意到式(2.4.2)中，参数$\boldsymbol{w}$和b的大小缩放并不会改变样本点$\boldsymbol{x}_n$到决策面的距离（习题 2.4.3），因此可以规定样本点到决策面的最小距离（即间隔）满足$t_n y(\boldsymbol{x}_n) = 1$。由此，所有的训练样本均满足：

$$t_n(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n) + b) \geq 1, n = 1, \dots, N \tag{2.4.4}$$

注意，式(2.4.4)共有N个，一个训练样本对应一个。具有最小距离的样本点构成等式约束条件，而其他样本点则构成不等式约束条件。显然，对于最大间隔决策面，至少正负样本点各有一个满足等式约束条件。如图 2.4.5 所示，$y = 0$为最大间隔决策面，$y = 1$和$y = -1$分别为正负样本的最大间隔面（满足$t_n y(\boldsymbol{x}_n) = 1$），其上分别有 2 个正样本（$p_1$和$p_2$）和 1 个负样本（$n_1$）。除了最大间隔面上的 3 个样本点，其他样本点均满足$t_n y(\boldsymbol{x}_n) > 1$。

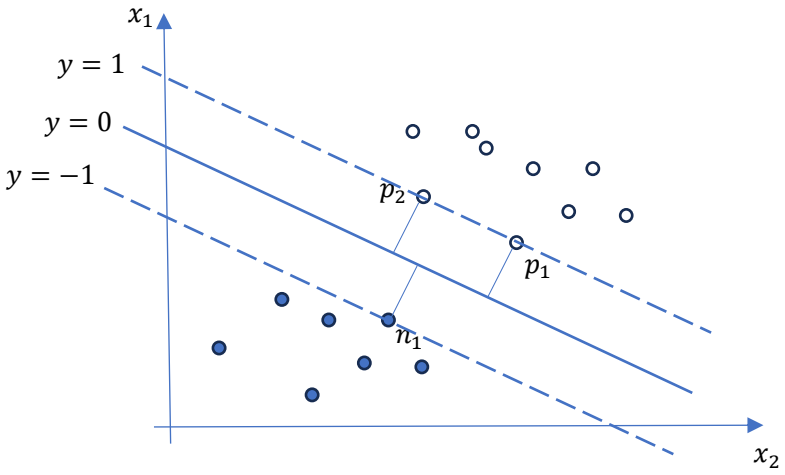

图 2.4.5 最大间隔面和最大间隔决策面

由此，式(2.4.3)的最优化问题就转化为了最大化$\frac{1}{\|\boldsymbol{w}\|}$，也就是最小化$\|\boldsymbol{w}\|^2$：

$$argmin_{\boldsymbol{w},b}\frac{1}{2}\|\boldsymbol{w}\|^2 \tag{2.4.5}$$

式(2.4.5)在$\|\boldsymbol{w}\|^2$前面乘了个因子$\frac{1}{2}$，是为了后面推导的方便。式(2.4.4)和式(2.4.5)一起构成了一个凸二次规划问题：在线性约束条件下最小化凸二次目标函数。关于“凸函数”和“凸优化”，读者可以回顾下 2.3.3 节的介绍。

为了求解式(2.4.4)和式(2.4.5)定义的约束最优化问题，引入拉格朗日乘子$a_n \geq 0$，对应式(2.4.4)里第n个约束条件。由此，得到拉格朗日函数：

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{n=1}^{N} a_n\{t_n(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n) + b) - 1\} \tag{2.4.6}$$

其中，$\boldsymbol{a} = (a_1, \dots, a_N)^T$。$L(\boldsymbol{w}, b, \boldsymbol{a})$分别对$\boldsymbol{w}$和$b$求导，并将导数置 0，易得：

$$w = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\boldsymbol{x}_n) \tag{2.4.7}$$

$$0 = \sum_{n=1}^{N} a_n t_n \tag{2.4.8}$$

利用式(2.4.7)和式(2.4.8)，可以消掉式(2.4.6)中的$\boldsymbol{w}$和b，得到：

$$L(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m) \tag{2.4.9}$$

其中，$k(\boldsymbol{x}_n,\boldsymbol{x}_m)=\boldsymbol{\phi}(\boldsymbol{x}_n)^T\boldsymbol{\phi}(\boldsymbol{x}_m)$，称为核函数。注意，式(2.4.9)还需满足$a_n\geq 0$和式(2.4.8)这两个约束条件。由此，式(2.4.3)的最优化问题进一步转化为了关于$\boldsymbol{a}$的凸二次规划问题。这个转化后的问题也被称为式(2.4.4)和式(2.4.5)的对偶问题，其中核函数$k(\boldsymbol{x}_n,\boldsymbol{x}_m)$的引入使得无需直接考虑特征变换函数$\varphi(\boldsymbol{x})$，这对于$\varphi(\boldsymbol{x})$的维数较高甚至无穷维的情况特别方便。举个例子，假设$\boldsymbol{x}=(x_1,x_2)$为 2 维，而$\varphi(\boldsymbol{x})=(a_1x_1,a_1x_2,a_2x_1^2,a_2x_2^2,\dots,a_nx_1^n,a_nx_2^n,\dots)$为无穷维，则$\boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{\phi}(\boldsymbol{x}')$根本就无法得到。

值得注意的是，式(2.4.6)需要相对于$\boldsymbol{w}$和b最小化，而相对于$\boldsymbol{a}$最大化，因此式(2.4.9)需要相对于$\boldsymbol{a}$最大化。习惯上，可以将式(2.4.9)改写为相对于$\boldsymbol{a}$的最小化：

$$\begin{aligned} &argmin_{\boldsymbol{a}}\left\{\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}a_na_mt_nt_mk(\boldsymbol{x}_n,\boldsymbol{x}_m)-\sum_{n=1}^{N}a_n\right\}\\ &\text{s.t. } a_n\geq 0,n=1,\dots,N\\ &\quad \textstyle\sum_{n=1}^{N}a_nt_n=0 \end{aligned} \tag{2.4.10}$$

2.4.4 线性不可分的情况

前面的讨论假定了训练样本在特征空间$\varphi(\boldsymbol{x})$里线性可分，从而有$t_ny(\boldsymbol{x}_n)\geq 1$，正负样本能够被最大间隔决策面完全正确分开。那么，对于线性不可分的情况，应该如何来考虑呢？如图 2.4.6 所示，两类样本互相重叠，找不到任何一个决策面能够将其完全正确分开（该图中画出了一个决策面，读者可以尝试画出更多的决策面）。由此看来，必须允许部分样本被分错！当然，可以要求被分错的样本尽可能少，而且错的程度尽可能低（读者可以尝试在图 2.4.6 中找一下这样的决策面）。

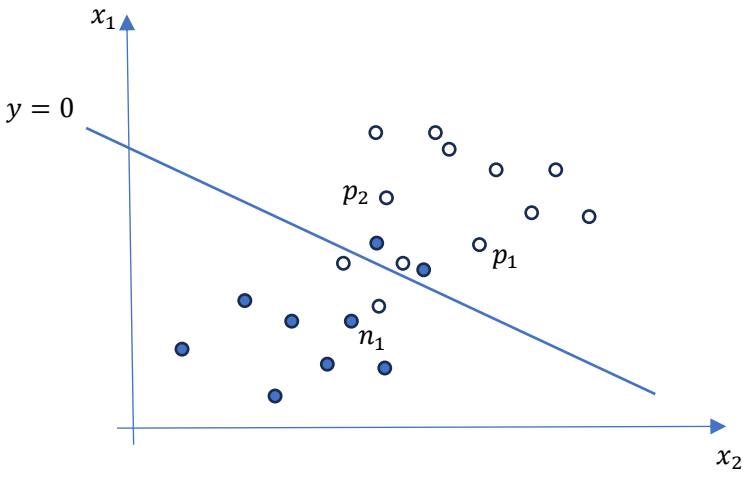

图 2.4.6 线性不可分

为此，对于每个样本点引入一个松弛变量$\xi_n\geq 0,n=1,\dots,N$：满足$\xi_n=0$的点位于正确一边的间隔面上或其内部，比如图 2.4.7 中的负类点n_3、n_4和n_5，正类点p_4和p_5；满足$0<\xi_n<1$的点位于正确一边的间隔面与决策面之间，比如图 2.4.7 中的负类点n_2和正类点p_3；满足$\xi_n=1$的点位于决策面上，比如图 2.4.7 中的正类点p_2；而满足$\xi_n>1$的点则位于决策面的错误一边，比如图 2.4.7 中的负类点n_1和正类点p_1。读者可以与图 2.4.5 进行比较，以帮助理解。值得注意的是，满足$\xi_n>1$的点同样可以细分为三种情况：$1<\xi_n<2$（决策面和错误一边的间隔面之间），$\xi_n=2$（错误一边的间隔面上），$\xi_n>2$（错误一边的间隔面内部）。

可见，ξ_n的值越大，则正确的程度越低；反之，则错误的程度越低。另外，对于ξ_n不等于 0 的情况，还可以得到$\xi_n = |t_n - y(\boldsymbol{x}_n)|$。

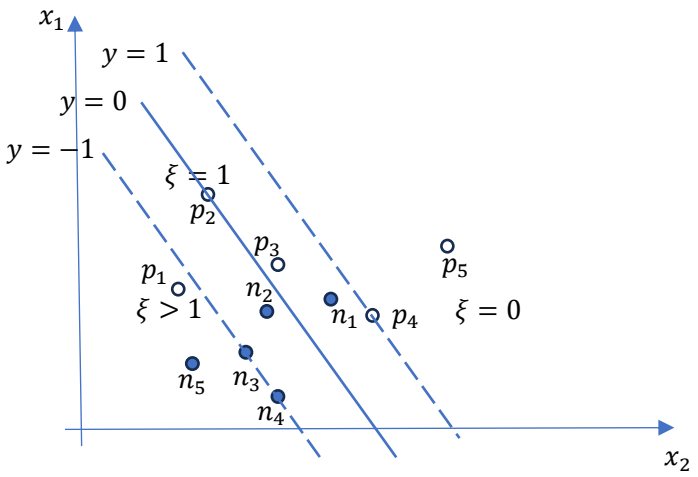

图 2.4.7 线性不可分与松弛变量ξ_n

综合以上几种情况，则引入松弛变量ξ_n之后，所有的样本点须满足：

$$t_n(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n) + b) \geq 1 - \xi_n, n = 1, \dots, N \tag{2.4.11}$$

由于允许部分样本被分错，图 2.4.7 中定义的间隔也被称为“软间隔（Soft Margin）”。相应地，图 2.4.5 中定义的间隔被称为“硬间隔（Hard Margin）”。基于线性约束条件(2.4.11)，软间隔 SVM 的最优化问题可以写为：

$$argmin_{\boldsymbol{w},b}(\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n) \tag{2.4.12}$$

其中超参数$C > 0$用来控制“最大化间隔”（对应式中第一项）和“松弛变量惩罚”（对应式中第二项）之间的权衡：C越大，则$\sum_{n=1}^{N}\xi_n$权重更大；C越小，则$\frac{1}{2}\|\boldsymbol{w}\|^2$权重更大。实际上，1.2.3 节提到了“正则项”的概念，此处就是一个例子：$\frac{1}{2}\|\boldsymbol{w}\|^2$是损失项，$\sum_{n=1}^{N}\xi_n$是正则项，$C$是正则化系数。正则项和正则化系数一起（$C\sum_{n=1}^{N}\xi_n$）控制软间隔 SVM 的复杂度，从而提高其推广能力。具体来讲，由于被误分类样本点满足$\xi_n > 1$，则$\sum_{n=1}^{N}\xi_n$就是被误分类样本点总数的上界。C越大，则$\sum_{n=1}^{N}\xi_n$必须越小（被误分类样本越少），从而模型复杂度越低。如果$C \to \infty$，则退化为式(2.4.5)定义的硬间隔 SVM：线性可分数据，两类样本被完全正确分开。

类似式(2.4.6)，得到软间隔 SVM 的拉格朗日函数：

$$L(\boldsymbol{w}, b, \xi_n, \boldsymbol{a}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n$$
$$-\sum_{n=1}^{N}a_n\{t_n(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n) + b) - 1 + \xi_n\} - \sum_{n=1}^{N}\mu_n\xi_n \tag{2.4.13}$$

其中，$a_n \geq 0$和$\mu_n \geq 0$是拉格朗日乘子。注意，拉格朗日乘子μ_n对应软间隔 SVM 新增的约束条件$\xi_n \geq 0$。

类似地，$L(\boldsymbol{w}, b, \xi_n, \boldsymbol{a})$分别对$\boldsymbol{w}$、$b$和$\xi_n$求导，并将导数置 0，易得：

$$w = \sum_{n=1}^{N}a_n t_n \boldsymbol{\phi}(\boldsymbol{x}_n) \tag{2.4.14}$$
$$0 = \sum_{n=1}^{N}a_n t_n \tag{2.4.15}$$

$$C - \mu_n = a_n \tag{2.4.16}$$

利用式(2.4.14)~(2.4.16)，可以消掉式(2.4.13)中的$\boldsymbol{w}$、b和ξ_n，得到：

$$L(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m) \tag{2.4.17}$$

这个式子形式上和式(2.4.9)完全一样，但是要注意约束条件有所不同。具体来讲，$a_n \geq 0$是一样的，但新增的$\mu_n \geq 0$和式(2.4.16)意味着$a_n \leq C$，从而要求$0 \leq a_n \leq C$。这个约束也常被称为“盒子约束（Box Constraints）”。类似式(2.4.10)，可以将式(2.4.17)进一步改写为相对于$\boldsymbol{a}$的最小化：

$$\begin{aligned} & argmin_{\boldsymbol{a}}\left\{\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m) - \sum_{n=1}^{N} a_n\right\} \\ & \text{s.t. } 0 \leq a_n \leq C, n = 1, \ldots, N \\ & \sum_{n=1}^{N} a_n t_n = 0 \end{aligned} \tag{2.4.18}$$

2.4.5 最优化问题的求解

关于式(2.4.10)和(2.4.18)的求解，常用的算法是SMO（Sequential Minimal Optimization）算法。这个算法的核心思想如下：每次迭代过程中，启发式地选取一对参数a_i和a_j进行优化，其他参数保持不变。由于两个变量的二次规划问题有解析解，所以每次选取的参数a_i和a_j可直接求解。

首先来看参数a_i和a_j的选取。可以采用一些启发式的规则，比如a_i选取违反KKT条件最严重的；a_j选取的标准是使得误差变化最大。说明一下，所谓KKT条件（Karush-Kuhn-Tucker conditions）指的是最优化问题(2.4.10)和(2.4.18)的充要条件。式(2.4.10)的KKT条件为：

$$a_n \geq 0 \tag{2.4.19}$$

$$t_n y(\boldsymbol{x}_n) - 1 \geq 0 \tag{2.4.20}$$

$$a_n\{t_n y(\boldsymbol{x}_n) - 1\} = 0 \tag{2.4.21}$$

由此可知，要么$a_n = 0$（样本点在间隔面内部），要么$t_n y(\boldsymbol{x}_n) = 1$（样本点在间隔面上，是支持向量）。参见图2.4.5。

式(2.4.18)的KKT条件为：

$$a_n \geq 0 \tag{2.4.22}$$

$$t_n y(\boldsymbol{x}_n) - 1 + \xi_n \geq 0 \tag{2.4.23}$$

$$a_n\{t_n y(\boldsymbol{x}_n) - 1 + \xi_n\} = 0 \tag{2.4.24}$$

$$\mu_n \geq 0 \tag{2.4.25}$$

$$\xi_n \geq 0 \tag{2.4.26}$$

$$\mu_n \xi_n = 0 \tag{2.4.27}$$

类似的，要么$a_n = 0$（样本点在间隔面内部），要么$t_n y(\boldsymbol{x}_n) = 1 - \xi_n$（样本点在间隔面上或间隔面外部，是支持向量）。参见图2.4.7。

对于a_j的选取，所谓误差指的就是预测值$y(\boldsymbol{x}_i)$和真实值t_i的差：

$$E_i = y(\boldsymbol{x}_i) - t_i, i = 1,2 \tag{2.4.28}$$

注意，利用式(2.4.14)（同式(2.4.7)），可得：

$$y(\boldsymbol{x}_i) = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\boldsymbol{x}_n)^T \boldsymbol{\phi}(\boldsymbol{x}_i) + b = \sum_{n=1}^{N} a_n t_n k(\boldsymbol{x}_i, \boldsymbol{x}_n) + b \tag{2.4.29}$$

其中，用到核函数的定义（见2.4.3节）。

选取了参数a_i和a_j之后，下一步就是利用两个变量二次规划问题的解析解直接求解a_i和

a_j。本节针对最优化问题 (2.4.18)（(2.4.10)是其特殊情况）给出两个变量的解析解[1]：

$$a_2 = a_2^{old} + \frac{t_2(E_1 - E_2)}{\eta} \tag{2.4.30}$$

$$\eta = k(\boldsymbol{x}_1, \boldsymbol{x}_1) + k(\boldsymbol{x}_2, \boldsymbol{x}_2) - \mathbf{2}k(\boldsymbol{x}_1, \boldsymbol{x}_2) \tag{2.4.31}$$

$$L \le a_2 \le H \rightarrow a_2^{new} \tag{2.4.32}$$

$$L = \max(0, a_2 - a_1), H = \min(C, C + a_2 - a_1), \text{当} t_1 \ne t_2 \tag{2.4.33}$$

$$L = \max(0, a_2 + a_1 - C), H = \min(C, a_2 + a_1), \text{当} t_1 = t_2 \tag{2.4.34}$$

$$a_1^{new} = a_1^{old} + t_1 t_2 (a_2^{old} - a_2^{new}) \tag{2.4.35}$$

$$b_1^{new} = b^{old} - E_1 - t_1 k(\boldsymbol{x}_1, \boldsymbol{x}_1)(a_1^{new} - a_1^{old}) - t_2 k(\boldsymbol{x}_2, \boldsymbol{x}_1)(a_2^{new} - a_2^{old}) \tag{2.4.36}$$

$$b_2^{new} = b^{old} - E_2 - t_1 k(\boldsymbol{x}_1, \boldsymbol{x}_2)(a_1^{new} - a_1^{old}) - t_2 k(\boldsymbol{x}_2, \boldsymbol{x}_2)(a_2^{new} - a_2^{old}) \tag{2.4.37}$$

$$b^{new} = b_1^{new}, \text{当} 0 < a_1^{new} < C \tag{2.4.38}$$

$$b^{new} = b_2^{new}, \text{当} 0 < a_2^{new} < C \tag{2.4.39}$$

$$b^{new} = (b_1^{new} + b_2^{new})/2, \text{当} a_1^{new} \text{和} a_2^{new} \text{为} 0 \text{或} C \tag{2.4.40}$$

值得特别注意的是，尽管决定决策面的是少量关键样本（支持向量），求解的过程仍然要用到所有N个训练样本。

2.4.6 使用求解的 SVM 进行预测

由 2.4.5 节求解得到的$\boldsymbol{a}$和b，对于测试样本$\boldsymbol{x}_t$就可以直接应用式(2.4.29)得到$y(\boldsymbol{x}_t)$的值，该值的符号即为$\boldsymbol{x}_t$的预测类别：负号为负类，正号为正类；该值的绝对值则反映了$\boldsymbol{x}_t$到决策面的距离，这个距离值越大则预测类别的可信度越高。

2.4.5 节中，为了求解$\boldsymbol{a}$和b，需要用到所有N个训练样本。那么，进行预测时也需要用到所有N个训练样本吗？实际上，如果$a_n = 0$（样本点在间隔面内部），则对应的训练样本$\boldsymbol{x}_n$将不会出现在式(2.4.29)中。也就是说，只有$a_n > 0$的训练样本会在预测时用到，这些训练样本就是前面多次提到的“支持向量（SV）”。由于 SV 是少量关键训练样本，这就带来了两方面的好处：第一，避免对数据的过拟合，保证 SVM 良好的推广能力；第二，预测过程的计算复杂度低。

2.4.7 核函数与核方法

2.4.3 节定义了核函数：$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \phi(\boldsymbol{x}_n)^T \phi(\boldsymbol{x}_m)$。该函数的引入使得无需直接考虑特征变换函数$\varphi(\boldsymbol{x})$，这对于$\varphi(\boldsymbol{x})$的维数较高甚至无穷维的情况特别方便。那么，核函数有什么要求呢？如何构造核函数呢？有哪些常用的核函数呢？本节就这些问题作一个基本的解答。

首先，核函数定义为两个特征向量的内积，因此其具有对称性，即$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = k(\boldsymbol{x}_m, \boldsymbol{x}_n)$。

其次，核函数构成的对称阵（称为核矩阵或 Gram 矩阵）是半正定阵。具体来讲，对于N个训练样本点，两两之间计算$K_{nm} = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$，就得到具有$N \times N$个元素的方阵。又因为核函数的对称性，这个方阵还是对称阵。而对称阵具有实特征值，如果这些实特征值都大于等于 0，那么就满足核矩阵是半正定阵的要求。

另外，核函数$k(\boldsymbol{x}_n, \boldsymbol{x}_m)$本质上是在度量两个样本点$\boldsymbol{x}_n$和$\boldsymbol{x}_m$之间的相似性，因此跟实际应用紧密相关，需要结合具体应用来进行考量。

关于构造核函数。显然，可以通过选择特征变换函数$\varphi(\boldsymbol{x})$来构造核函数。比如对于最简单的恒等映射$\varphi(\boldsymbol{x}) = \boldsymbol{x}$，可以得到$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \boldsymbol{x}_n{}^T \boldsymbol{x}_m$，这就是最简单的“线性核”。也可以直接构造核函数，比如二次多项式核$k(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^T \boldsymbol{z})^2$，易推得其对应的$\varphi(\boldsymbol{x})$包含所有的二次

注 1：具体的推导过程读者可以参考李航老师的《统计学习方法》相应章节。

项。

更有效的构造核函数的方式为：利用简单的核函数按照一些规则来构造新的核函数。比如$k(\boldsymbol{x},\boldsymbol{z})=(\boldsymbol{x}^T\boldsymbol{z}+c)^2$，其中$c>0$。其对应的$\varphi(\boldsymbol{x})$则包含常量、一次项和二次项。类似地，$k(\boldsymbol{x},\boldsymbol{z})=(\boldsymbol{x}^T\boldsymbol{z})^M$包含所有的$M$阶项。$k(\boldsymbol{x},\boldsymbol{z})=(\boldsymbol{x}^T\boldsymbol{z}+c)^M$，其中$c>0$，包含$M$阶及以下的项。

一个常用的核函数是高斯核：

$$k(\boldsymbol{x},\boldsymbol{z})=\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{z}\|^2}{2\sigma^2}\right),\sigma>0 \tag{2.4.41}$$

这个核函数可以由线性核、指数核按照一些规则构造出来。进而，如果把其中的线性核替换为一般的非线性核又可以得到一个新的核函数。

另外，拉普拉斯核也比较常用：

$$k(\boldsymbol{x},\boldsymbol{z})=\exp\left(-\|\boldsymbol{x}-\boldsymbol{z}\|/\sigma\right),\sigma>0 \tag{2.4.42}$$

表示定理表明：对于任意损失函数与单调递增的正则化项，最优解都可以表示为核函数$k(\boldsymbol{x}_n,\boldsymbol{x}_m)$的线性组合。这个定理显示出核函数的巨大威力。由此，发展出一系列基于核函数的学习方法，统称为“核方法”。典型的做法是：通过引入核函数（称为核技巧或核替换）将线性模型拓展为对应的非线性模型。比如前面提到的核化对率回归、核化 PCA、核化 LDA 这些。

核方法的关键在于核函数的选取，但核函数如何选取是没有理论支撑的、是不可解释的。这就意味着，核函数的选取依赖于具体的应用、依赖于经验，所以也被称为“核函数工程”。

2.4.8 软间隔 SVM 的核心代码实现

本节将详细介绍软间隔 SVM（硬间隔 SVM 是其特例）的代码实现，并给出代码的简单运行实例。相信读者在此基础上能够在实验课中进一步完善和改进代码，完成更具挑战性和实用性的应用任务。

1．简单数据集 1：

```
3.542485 1.977398    -1
3.018896 2.556416    -1
7.551510 -1.580030    1
2.114999 -0.004466    -1
8.127113 1.274372     1
7.108772 -0.986906    1
```

上面给出了该数据集的前 6 行，一行表示一个样本，总共有 100 行。每行有 3 列，前两列表示两个特征，最后一列是类别标签，注意其取值是-1 和 1，分别对应负类和正类。将这个数据集可视化出来的结果（代码类似 2.3.6 节的 plotBestFit()）如图 2.4.8 所示：

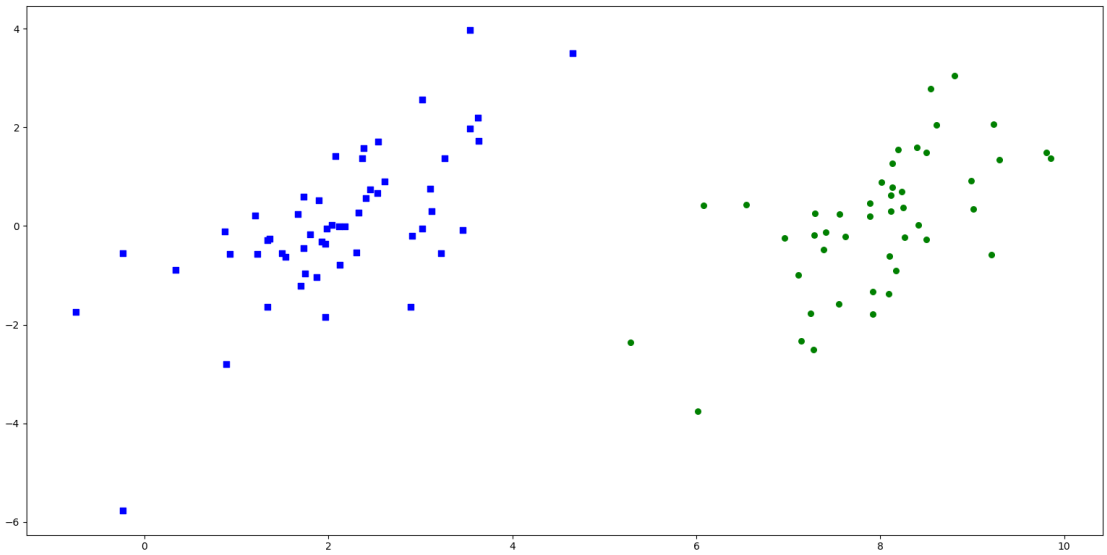

图 2.4.8 简单数据集 1

由图 2.4.8 可见，这个数据集是一个线性可分的数据集，正负类样本数基本平衡（正类对应圆形点，负类对应正方形点）。采用类似 2.3.6 节的 loadDataSet()函数把简单数据集 1 装入到 dataList（对应两个特征）和 labelList（对应类别标签）两个列表中。

2．线性核 SVM 训练代码：

```
def smoSimple(dataList, labelList, C, toler, maxIter):
    dataMat = np.mat(dataList); labelMat = np.mat(labelList).transpose()
    b = 0; n, m = np.shape(dataMat)
    alphas = np.mat(np.zeros((n, 1)))
    iter = 0
    while (iter < maxIter):
        alphaPairsChanged = 0
        for i in range(n):
            fXi = float(np.multiply(alphas, labelMat).T * (dataMat * dataMat[i, :].T)) + b
            Ei = fXi - float(labelMat[i])
            if ((labelMat[i]*Ei < -toler) and (alphas[i] < C)) or ((labelMat[i]*Ei > toler) and
(alphas[i] > 0)): #检查一个样本是否违反 KKT 条件
                j = selectJrand(i, n)
                fXj = float(np.multiply(alphas, labelMat).T * (dataMat * dataMat[j, :].T))
+ b
                Ej = fXj - float(labelMat[j])
                alphaIold = alphas[i].copy(); alphaJold = alphas[j].copy();
                if (labelMat[i] != labelMat[j]):
                    L = max(0, alphas[j] - alphas[i])
                    H = min(C, C + alphas[j] - alphas[i])
                else:
                    L = max(0, alphas[j] + alphas[i] - C)
```

```
                    H = min(C, alphas[j] + alphas[i])
                if L==H: print("L==H"); continue
                eta = 2.0 * dataMat[i, :] * dataMat[j, :].T - dataMat[i, :] * dataMat[i, :].T -
dataMat[j, :] * dataMat[j, :].T
                if eta >= 0: print("eta>=0"); continue
                alphas[j] -= labelMat[j] * (Ei - Ej)/eta
                alphas[j] = clipAlpha(alphas[j], H, L)
                if (abs(alphas[j] - alphaJold) < 0.00001): print("j not moving enough");
continue
                alphas[i] += labelMat[j]*labelMat[i]*(alphaJold - alphas[j])#在相反的方
向上以同 j 一样的量更新 i
                b1 = b - Ei - labelMat[i] * (alphas[i] - alphaIold) * dataMat[i, :] *
dataMat[i, :].T - labelMat[j] * (alphas[j] - alphaJold) * dataMat[i, :] * dataMat[j, :].T
                b2 = b - Ej - labelMat[i] * (alphas[i] - alphaIold) * dataMat[i, :] *
dataMat[j, :].T - labelMat[j] * (alphas[j] - alphaJold) * dataMat[j, :] * dataMat[j, :].T
                if (0 < alphas[i]) and (C > alphas[i]): b = b1
                elif (0 < alphas[j]) and (C > alphas[j]): b = b2
                else: b = (b1 + b2)/2.0
                alphaPairsChanged += 1
                print("iter: %d i:%d, pairs changed %d" % (iter,i,alphaPairsChanged))
        if (alphaPairsChanged == 0): iter += 1
        else: iter = 0
        print("iteration number: %d" % iter)
    return b, alphas
```

这个函数实现了线性核 SVM 的训练。输入：dataList—训练样本特征列表；labelList—训练样本类别标签列表；C—a_n上限（见式(2.4.18)）；toler—误差门限；maxIter—迭代次数。输出：学习到的参数 b 和 alphas（即a_n）。

（1）把 dataList 和 labelList 转换为 Numpy 矩阵 dataMat 和 labelMat，便于后面的矩阵运算。注意 dataMat 是 n*m 的矩阵，n 个样本 m 个特征。而 labelMat 在转换时作了转置操作，得到 n*1 的矩阵。

（2）b 和 alphas（n*1 的矩阵）都初始化为 0。

（3）每一次迭代过程中，“for i in range(n):”这一句意味着顺序选择第一个样本点$\boldsymbol{x}_i$。

（4）“fXi = float(np.multiply(alphas, labelMat).T * (dataMat * dataMat[i, :].T)) + b”这一句对应式(2.4.29)。注意：np.multiply()是矩阵对应元素相乘（Hadamard积），而“*”是矩阵乘法。由于采用线性核$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \boldsymbol{x}_n{}^T \boldsymbol{x}_m$，直接就是特征向量的内积。

（5）“Ei = fXi - float(labelMat[i])”这一句对应式(2.4.28)，即得到$\boldsymbol{x}_i$的预测误差。

（6）“if ((labelMat[i]*Ei < -toler) and (alphas[i] < C)) or ((labelMat[i]*Ei > toler) and (alphas[i] > 0)):”这一句检查样本点$\boldsymbol{x}_i$是否违反KKT条件(2.4.22)~(2.4.27)。如果$t_i y(\boldsymbol{x}_i) - 1 \leq 0$（$\boldsymbol{x}_i$在正确间隔面的错误一端，即正确间隔面的外部）则必须有$a_i = C$，否则就违反了KKT条件；如果$t_i y(\boldsymbol{x}_i) - 1 \geq 0$（$\boldsymbol{x}_i$在正确间隔面的正确一端，即正确间隔面的内部）则必须有$a_i = 0$，否则就违反了KKT条件。注意，超参toler可以指定一个误差门限，如果没有超过这个门限值就认为没有违反，这对于计算机的有限精度浮点运算来讲是必要的。

（7）如果选取的样本点$\boldsymbol{x}_i$没有违反KKT条件，就尝试选取下一个样本点。否则，选取$\boldsymbol{x}_j$：j = selectJrand(i, n)。这个函数后面来看。

（8）类似（4）和（5），对$\boldsymbol{x}_j$计算fXj和Ej。

（9）将a_i和a_j备份为alphaIold和alphaJold，对应式(2.4.35)中的a_1^{old}和式(2.4.30)中的a_2^{old}。

（10）根据式(2.4.33)和式(2.4.34)计算a_j的低限L和高限H。

（11）根据式(2.4.31)计算eta，然后根据式(2.4.30)计算a_j，并调用函数clipAlpha()确保其值在L和H之内。

（12）根据式(2.4.35)计算a_i。

（13）接下来计算参数b：对应公式(2.4.36)~(2.4.40)。

选取$\boldsymbol{x}_j$的函数如下：

```
def selectJrand(i, n):
    j = i #we want to select any j not equal to i
    while (j==i):
        j = int(np.random.uniform(0, n))
    return j
```

可见，采取的是随机选取（均匀分布）方式。

至此，线性核软间隔SVM的训练代码已介绍完毕，接下来就在“简单数据集1”上进行训练，看看实际效果如何。以语句“b, alphas = smoSimple(dataList, labelList, 6, 0.001, 400)”进行训练，然后将样本点、决策面、间隔面和SV都可视化出来，结果如图2.4.9所示：

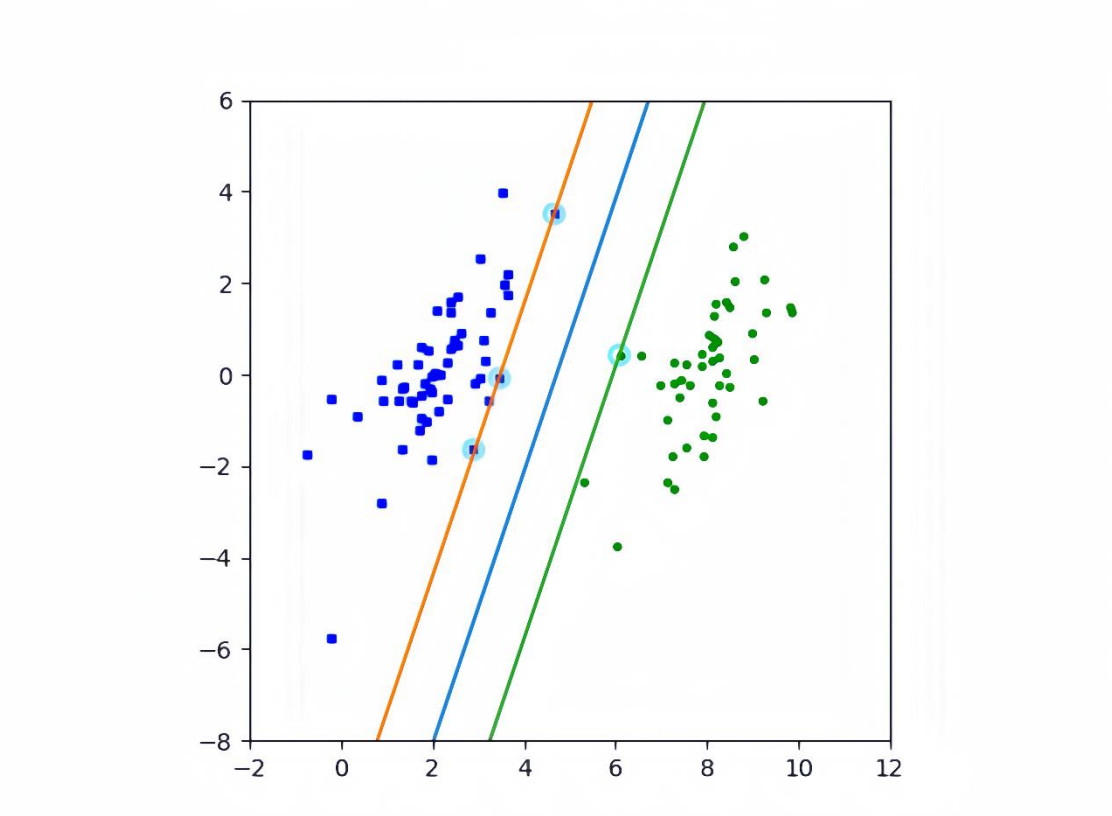

图2.4.9 简单数据集1上的训练结果

由于简单数据集 1 是线性可分数据集，所以正如所预期的，线性核 SVM 能够将其完全正确分开。图 2.4.9 中，用圆圈出来的样本点就是 SV：总共有 4 个 SV，负类有 3 个，正类

有 1 个。由于没有分错的样本点，所以 SV 都在其正确一边的间隔面上。从这个例子，可以充分体会到 SVM 的优势：100 个样本点中，起作用的只有 4 个关键样本点，既避免了对数据的过拟合，又大幅降低了预测时的计算开销。

到这里，读者一定想知道软间隔 SVM 在线性不可分数据集上的表现如何。毕竟，实际应用中的数据一般都是线性不可分的。为了方便对比，就使用 2.3.6 节用到的线性不可分数据集——称其为“简单数据集 2”，读者可以回顾下图 2.3.11，确认下这个数据集确实是线性不可分的。

同样以语句“b, alphas = smoSimple(dataList, labelList, 6, 0.001, 400)”进行训练，然后将样本点、决策面、间隔面和SV都可视化出来，结果如图2.4.10所示：

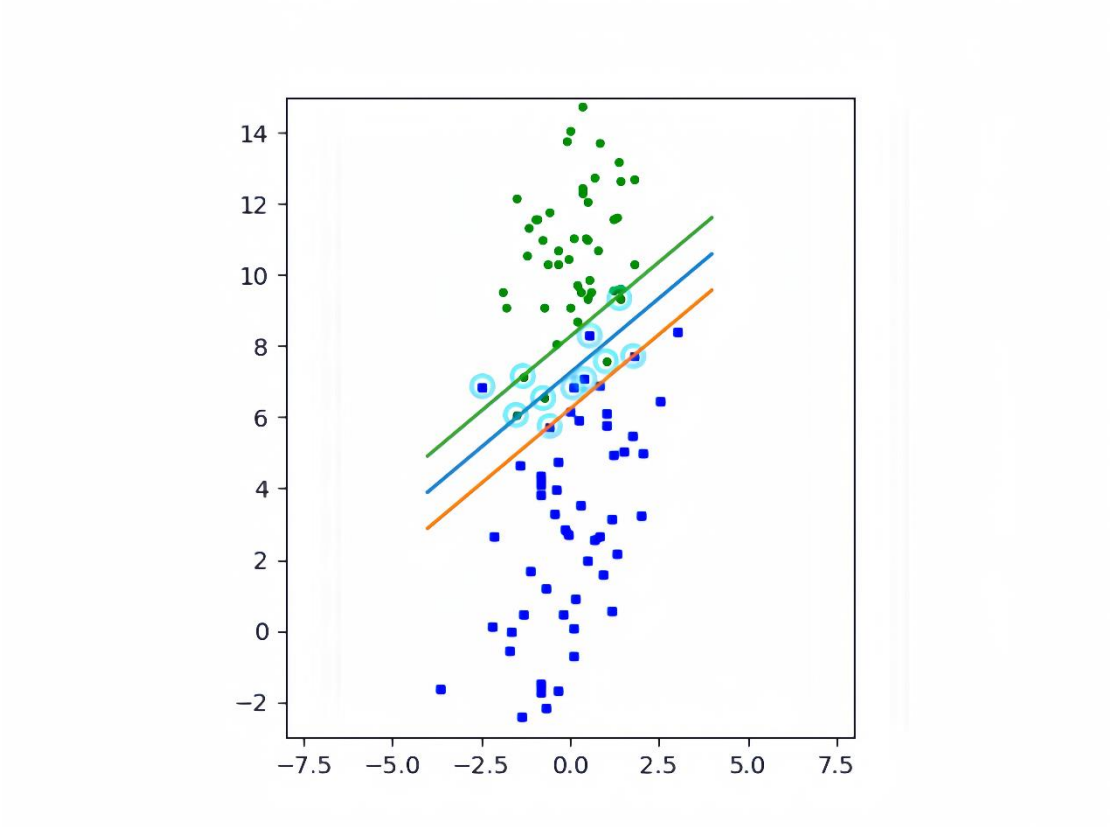

图 2.4.10 简单数据集 2 上的训练结果

正如所预期的，线性核 SVM 并不能将这个线性不可分数据集完全正确分开，共有 4 个样本点被分错，正类和负类各两个。由于是软间隔 SVM，因此在正确间隔面上和正确间隔面错误一端的样本点都是 SV，共计 11 个。

比较下图 2.4.10 和 2.3.6 节图 2.3.11 的分类结果，可以发现对率回归模型和线性核软间隔 SVM 的性能基本相当，都有 4 个样本点被分错。另外，对率回归模型在预测时并不会用到训练样本点，因此这方面相比 SVM 还更有优势。当然，前面也谈到了，非线性核 SVM（比如高斯核、拉普拉斯核）对于线性不可分数据集更有优势，这些就留给读者来进一步探索。

小故事：没有什么比一个好的理论更实用

弗拉基米尔•瓦普尼克（Vladimir N. Vapnik, 1936— ）是杰出的数学家、统计学家、计算机科学家。他出生于苏联，1958 年在乌兹别克国立大学获数学硕士学位，1964 年在莫斯科控制科学学院获统计学博士学位，此后一直在该校工作并担任计算机系主任。1990 年（苏联解体的前一年）他离开苏联来到新泽西州的美国电话电报公司贝尔实验室工作，1995 年发表了最初的 SVM 文章。当时神经网络正当红，因此这篇文章被权威期刊 Machine Learning 要求以“支持向量网络”的名义发表。

实际上，瓦普尼克在 1963 年就已提出了支持向量的概念，1968 年他与另一位苏联数学家 A. Chervonenkis 提出了以他们两人的姓氏命名的“VC 维”，1974 年又提出了结构风险最小化原则，使得统计学习理论在二十世纪七十年代就已成型。但这些工作主要是以俄文发表的，直到瓦普尼克随着东欧剧变和苏联解体导致的苏联科学家移民潮来到美国，这方面的研究才在西方学术界引起重视，统计学习理论、支持向量机、核方法在二十世纪末大红大紫。

瓦普尼克 2002 年离开美国电话电报公司加入普林斯顿的 NEC 实验室，2014 年加盟脸书（Facebook）公司人工智能实验室。1995 年之后他还在伦敦大学、哥伦比亚大学等校任教授，据说瓦普尼克在苏联根据一本字典自学了英语及其发音。他有一句名言被广为传诵：“Nothing is more practical than a good theory."

启迪：

（1）理论的建立是一门学科走向成熟的重要标志

（2）追热点不如坚持原创、坚持自我

（3）一门学科的发展过程是一个螺旋式上升的历史过程

习题 2.4

2.4.1 基于图 2.4.1 中的向量表示，说明法向量$\boldsymbol{w}$垂直于决策面上的任意向量$\boldsymbol{x}$。

2.4.2 说明 2.4.2 节里的公式$t_n y(\boldsymbol{x}_n) \geq 0$为何成立。

2.4.3 说明为何式(2.4.2)中，参数$\boldsymbol{w}$和b的大小缩放并不会改变样本点$\boldsymbol{x}_n$到决策面的距离。
2.4.4 式(2.4.5)的优化目标函数里未出现b，这意味着无需优化b吗？谈谈你的理解。
2.4.5 式(2.4.6)里求和符号的前面为什么是减号，谈谈你的理解。
2.4.6 由式(2.4.6)推导出式(2.4.7)和式(2.4.8)。
2.4.7 利用式(2.4.7)和式(2.4.8)，消掉式(2.4.6)中的$\boldsymbol{w}$和b，得到式(2.4.9)。
2.4.8 对于ξ_n不等于 0 的情况，说明$\xi_n = |t_n - y(\boldsymbol{x}_n)|$。
2.4.9 式(2.4.18)的KKT条件表明，要么$a_n = 0$（样本点在间隔面内部），要么$t_n y(\boldsymbol{x}_n) = 1 - \xi_n$（样本点在间隔面上或间隔面外部，是支持向量）。请结合图示，进一步说明后一种情况下，样本点所处位置的几种可能情况，并给出a_n、ξ_n、μ_n的取值情况。
2.4.10 使用训练好的 SVM 对测试样本进行预测时，会用到哪些训练样本呢？为什么？与 2.1 节介绍的 kNN、2.2 节介绍的决策树、2.3 节介绍的对率回归进行比较，有何不同？
2.4.11 对于二次多项式核函数$k(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^T \boldsymbol{z})^2$，请推出对应的特征变换函数$\varphi(\boldsymbol{x})$。
2.4.12 当采用类似 2.3.6 节的 loadDataSet()函数把简单数据集 1 装入到 dataList（对应两个特征）和 labelList（对应类别标签）两个列表中时，代码有什么关键不同呢？为什么？
2.4.13 从运算的角度详细分析“fXi = float(np.multiply(alphas, labelMat).T * (dataMat * dataMat[i, :].T)) + b”这句代码是如何对应式(2.4.29)的，要求给出每个矩阵的形状、每个运算的输入和输出、最终运算结果的形状。
2.4.14 可以通过向量点积来理解间隔的概念，如图 2.38 所示，正负间隔面上各有一个样本点，其差向量与权重向量$\boldsymbol{w}$的点积即为正负间隔面的距离 2，请给出证明。

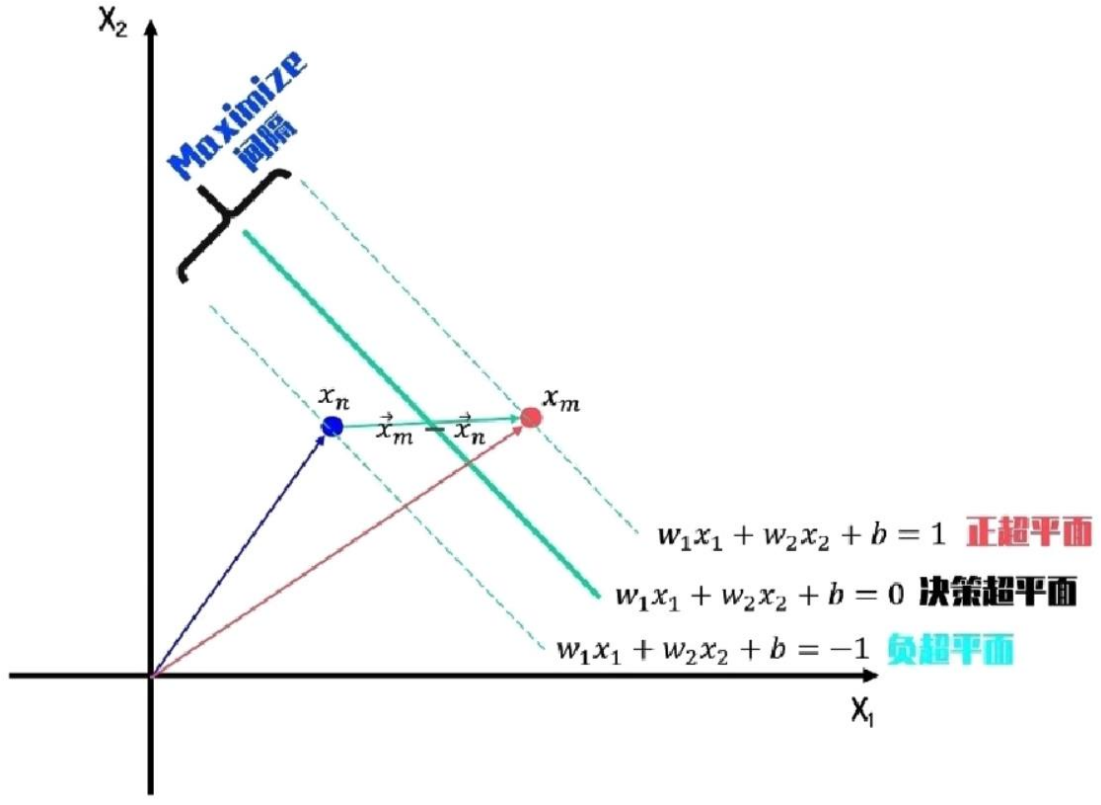

图 2.38 通过向量点积来理解间隔的概念

2.4.15 可以通过向量点积来理解权重向量$\boldsymbol{w}$与决策面垂直，如图 2.39 所示，决策面上有两个个不同样本点，其差向量与权重向量$\boldsymbol{w}$垂直（正交），请给出证明。

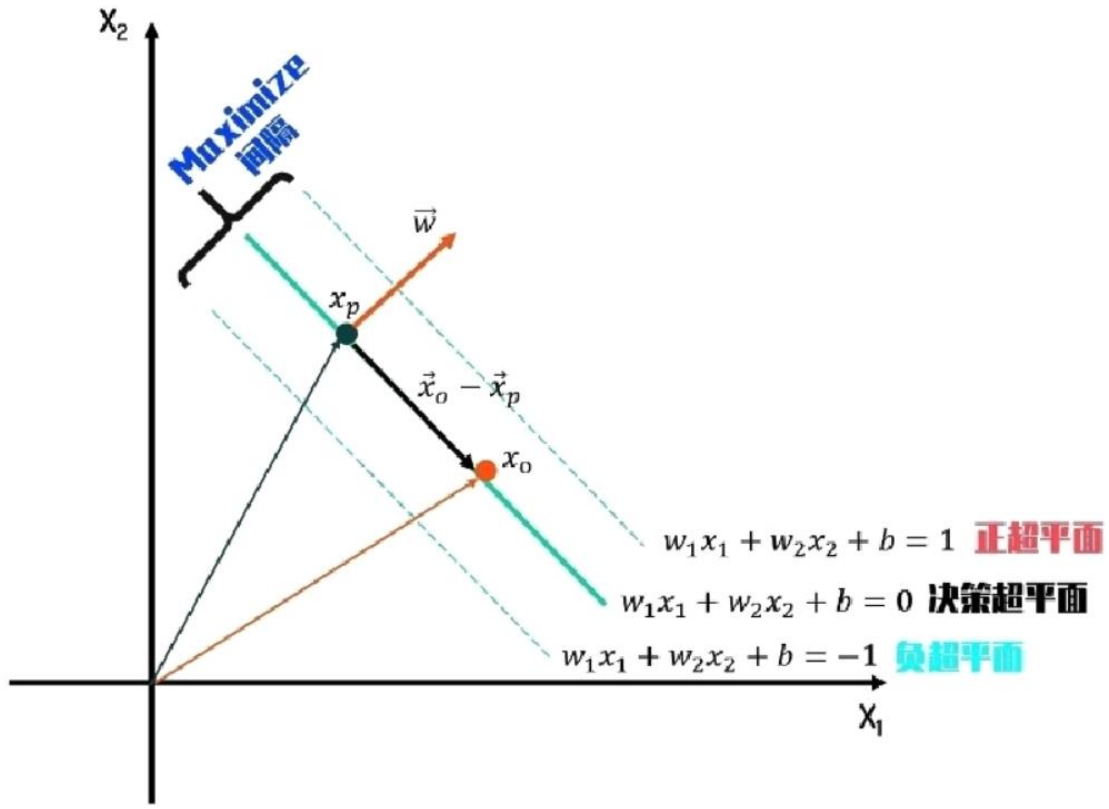

图 2.39 通过向量点积来理解权重向量$\boldsymbol{w}$与决策面垂直

2.4.16 如果$\boldsymbol{\varphi(x)}$的维数高于$\boldsymbol{x}$的维数，则特征空间对原始数据空间进行了升维，那么升维能够带来什么好处呢？请结合图 2.40 进行思考，该图中所示的 2 维原始数据（线性不可分）如何能够通过升维而成为线性可分数据。

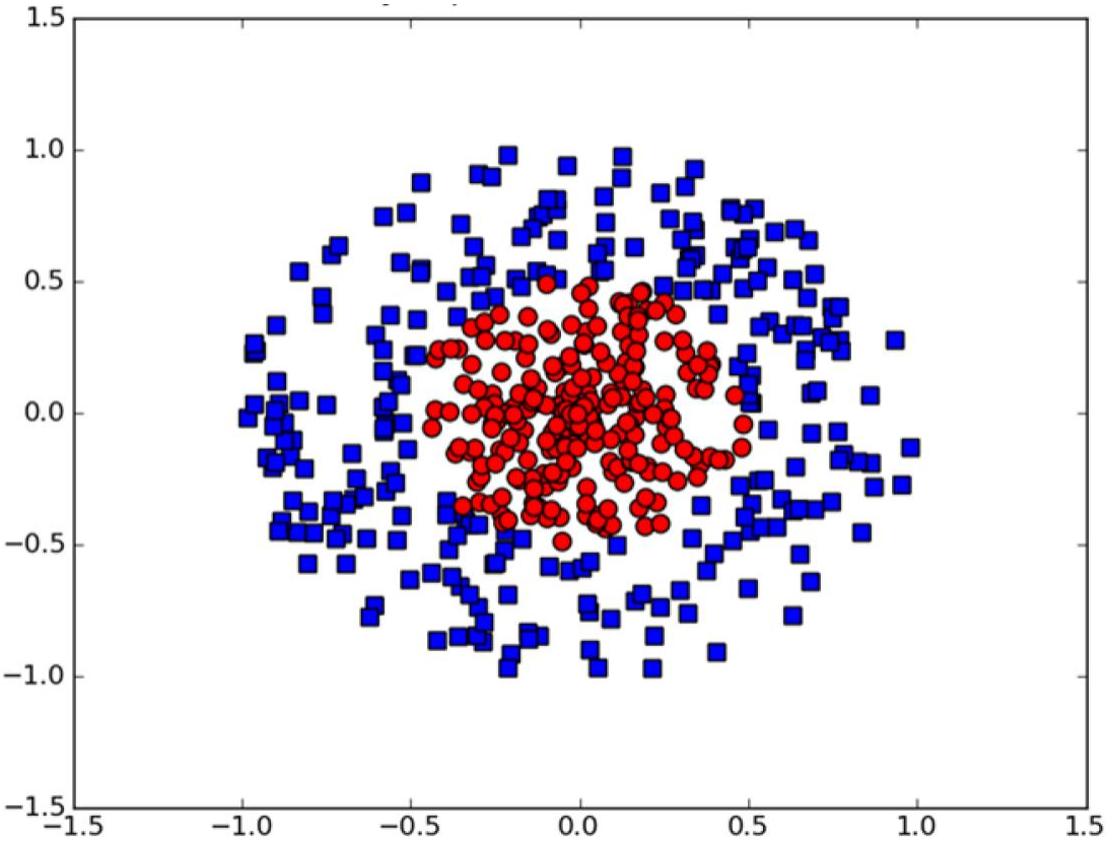

图 2.40 线性不可分 2 维原始数据

2.4.17 假设$\boldsymbol{x}$和$\boldsymbol{z}$是 2 维向量，请说明高斯核函数（式(2.4.41)）对应的特征向量$\boldsymbol{\varphi}(\boldsymbol{x})$是无穷维。提示：将$\|\boldsymbol{x}-\boldsymbol{z}\|^2$展开为 3 项，从而$k(\boldsymbol{x},\boldsymbol{z})$表示为 3 项的乘积，再应用指数函数的泰勒展开式。