《机器学习导论》第 8 讲：SVM

支持向量机（Support Vector Machine， SVM）是一种面向二分类任务的机器学习模型，是 90 年代兴起的统计机器学习领域的代表性成果，再加上核函数的引入，使得 SVM 及相关技术被广泛用于各类机器学习任务中，是从 90 年代后期一直到深度学习兴起之前最主流的机器学习技术。

1、线性可分支持向量机

线性可分问题指在二分类训练数据集上可以使用超平面（Hyperplane）将两类样本完全判决开。如果数据集的输入特征个数是一个，超平面则为一个判决点；特征个数为二，超平面则为一条直线；特征个数为三则为一个平面。

线性可分问题是二分类问题的理想状况，支持向量机则通过该理想状况来评判并获得最优分类器，即在能够正确区分两类样本的候选分类器中使用何种标准来选出最优判决超平面，即能够尽可能分对未见的测试样本。

直观上感觉位于两类训练样本“正中间”的判决超平面最好，相比于其他判决超平面它更有利于分对出现在两类样本中间的测试样本。而这也是机器学习则更关注的在测试样本上的预测能力（泛化能力）。

这个直观上的位于“正中间”的判决超平面应该距离正负两类样本都足够远，由此，间隔（margin）被提出用于度量在判决超平面引入后两类样本被判决的程度。

1.1 判决超平面与 SVM 模型

给定训练数据集$D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_m,y_m)\}$，其中，$y_i\in\{-1,+1\}$，$\boldsymbol{x}\in\mathbb{R}^n$。支持向量机要学习得到的判决超平面使用式（**）的线性方程来描述：

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

相应地，用于判决的支持向量机模型为：

$$f(\boldsymbol{x})=sgn(\boldsymbol{w}^T\boldsymbol{x}+b)=\begin{cases}1, & \boldsymbol{w}^T\boldsymbol{x}+b\geqslant 0\\ -1, & \boldsymbol{w}^T\boldsymbol{x}+b<0\end{cases}$$

1.2 代价函数

该超平面不仅要能够正确分类训练数据集 D 中的所有样本，而且期望能让判决超

平面尽可能位于“正中间”，并且距离两类样本都尽可能的远，也就是间隔尽可能的大。如图 2 所示，图 2(a)的超平面导致的间隔相对于图 2(b)要大，相应地，判决面位于“正中间”的程度更高。而线性可分支持向量机则要找到那个间隔(margin)最大的并位于间隔中间的判决超平面，其中，间隔在式（）中给出，具体推导过程在式（）中给出。

$$margin = \frac{2}{||\boldsymbol{w}||}$$

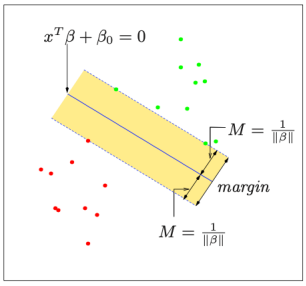


图*.* 间隔

综合正确分类两类样本和间隔最大化，线性可分情况下支持向量机的代价函数在式（）中给出：

$$\max_{\boldsymbol{w},b} \frac{2}{\| \boldsymbol{w} \|}$$

$$s.t.\, y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geqslant 1, i = 1, \ldots, m$$

其中，表达所有训练样本被正确分类使用$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geqslant 1$进行约束。这是一种新的表示训练误差的方式，多用在$y_i \in \{-1, +1\}$的二分类模型的代价函数中。即对于任意$y_i = 1$，均要求$\boldsymbol{w}^T\boldsymbol{x}_i + b \geqslant 1$；若$y_i = -1$，则有$\boldsymbol{w}^T\boldsymbol{x}_i + b \leqslant -1$。相比于$\boldsymbol{w}^T\boldsymbol{x}_i + b \geqslant 0$，这个约束更严格，而且利于训练误差的表征。使用这种表示方法，分类正确则模型输出与样本标注同号，其乘积为正数；反之异号，乘积为负。

为后续优化方便，可将最大化$\| \boldsymbol{w} \|^{-1}$替换为最小化$\| \boldsymbol{w} \|^2$，该替换在求解前述两个最值时会得到相同的$\boldsymbol{w}$。相应地，代价函数（式*）被重写为：

$$\min_{\boldsymbol{w},b} \frac{1}{2} \| \boldsymbol{w} \|^2 \quad 式\ 4$$

$$s.t.\, y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geqslant 1, i = 1, \ldots, m$$

在上述代价函数设计中，还有两个问题需要进一步展开。一是间隔的计算，二是

它由判决超平面$\boldsymbol{w}$引出，其大小由与超平面平行的$\boldsymbol{w}^T\boldsymbol{x} + b = 1$和$\boldsymbol{w}^T\boldsymbol{x} + b = -1$计算。根据点到直线距离公式，位于$\boldsymbol{w}^T\boldsymbol{x} + b = 1$上任意一点$\boldsymbol{x}'$到超平面$\boldsymbol{w}^T\boldsymbol{x} + b = 0$的距离可写为：

批注 [MOU1]: 说明 wx+b=a 和 wx+b=1 是等价的。见雷明 P64 页描述。;

$$dist = \frac{|\boldsymbol{w}^T\boldsymbol{x} + b|}{\| \boldsymbol{w} \|} = \frac{1}{\| \boldsymbol{w} \|}$$

因此可得$margin = \frac{2}{||\boldsymbol{w}||}$。

1.3 最小化代价函数

在式（）中给出的代价函数是一个带线性不等式的凸二次规划问题，可使用通用优化算法进行求解。针对支持向量机这一特定问题，常先使用拉格朗日乘子法将优化目标和多个约束整合为一个最小化问题；然后，将其变换为最大最小对偶问题；最后，使用通用的二次规划算法求出最优解。具体过程如下：

1.3.1 代价函数的拉格朗日对偶

拉格朗日乘子法通过引入多个参数（拉格朗日乘子）将多个约束同优化目标集成起来，再进行最优化。在式（）中给出的代价函数便可通过该方法将带有多个线性不等式约束的优化问题变换成一个无约束最小化问题（见式*。*），它也被称为式（）的对偶问题（dual problem）。加在m个约束前面的α_i被称为拉格朗日乘子。

$$\min_{\boldsymbol{w},b,\boldsymbol{\alpha}} L(\boldsymbol{w},b,\boldsymbol{\alpha}) = \min_{\boldsymbol{w},b,\boldsymbol{\alpha}} \left(\frac{1}{2}\| \boldsymbol{w} \|^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b))\right) \text{式 5}$$

拉格朗日乘子法通过引入多个参数（拉格朗日乘子）将多个约束同优化目标集成起来，再进行最优化。在式（）中给出的代价函数便可通过该方法将带有多个线性不等式约束的优化问题变换成一个无约束最小化问题（见式*。*），它也被称为式（）的对偶问题（dual problem）。加在m个约束前面的α_i（$\alpha_i \geqslant 0$）被称为拉格朗日乘子。

式（）中变量为$\boldsymbol{w}, b$和$\boldsymbol{\alpha}$，拉格朗日乘子法为最小化式（）采用两步进行求解，先固定拉格朗日乘子$\boldsymbol{\alpha}$，通过调整和$\boldsymbol{w}$和b取其最小值，这样式（）可变为关于$\boldsymbol{\alpha}$的函数；然后，通过调整拉格朗日乘子，求解其最大值。

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \min_{\boldsymbol{w},b} L(\boldsymbol{w},b,\boldsymbol{\alpha}) \\ = \quad &\max_{\boldsymbol{\alpha}} \{\min_{\boldsymbol{w},b} \{\frac{1}{2}\| \boldsymbol{w} \|^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b))\}\} \end{aligned} \text{式 7}$$

通过求解$L(\boldsymbol{w},b,\boldsymbol{\alpha})$ 相对于$\boldsymbol{w}$和b求偏导为 0 可求解第一步的$\min_{\boldsymbol{w},b} L(\boldsymbol{w},b,\boldsymbol{\alpha})$。具体地，通过$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i = 0$和$\frac{\partial L}{\partial b} = \sum_{i=1}^{m} \alpha_i y_i = 0$可得：

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \quad \text{式 8}$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

将上述两式代入式 7，可得式 4 的拉格朗日对偶问题：

$$\max_{\boldsymbol{\alpha}} \min_{\boldsymbol{w},b} L(\boldsymbol{w},b,\boldsymbol{\alpha}) = \max_{\boldsymbol{\alpha}} \left(\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \right)$$
$$s.t. \sum_{i=1}^{m} \alpha_i y_i = 0,$$
$$\alpha_i \geqslant 0, \quad i = 1, \cdots, m$$

式 10

式（10）是式（4）原始问题的拉格朗日对偶问题，也是关于$\boldsymbol{\alpha}$的凸二次规划问题，可使用通用的二次规划或专用的 SMO（序贯最小化，Sequential Minimal Optimization）算法进行求解。

在得到$\boldsymbol{\alpha}^*$之后，代入式 8 可得。根据“KKT 条件”定理可知最优解$\boldsymbol{w}^*$、b^*满足$\boldsymbol{w}^{*T}\boldsymbol{x}_j + b^* = y_j$，因此选择任意一个$\alpha_j^* > 0$计算可得$b^*$。

$$\boldsymbol{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \boldsymbol{x}_i$$

$$b^* = y_j - \sum_{i=1}^{m} \alpha_i^* y_i \boldsymbol{x}_i^T \boldsymbol{x}_j$$

根据获得的 w*和 b*，可知式 1 中的判决超平面为：

$$\sum_{i=1}^{m} \alpha_i^* y_i \boldsymbol{x}_i^T \boldsymbol{x} + b^* = 0$$

相应地，支持向量机的判决模型为：

$$f(\boldsymbol{x}) = sgn(\boldsymbol{w}^{*T}\boldsymbol{x} + b^*) = \begin{cases} 1, & \boldsymbol{w}^{*T}\boldsymbol{x} + b^* \geqslant 0 \\ -1, & \boldsymbol{w}^{*T}\boldsymbol{x} + b^* < 0 \end{cases}$$

式（判决模型）

值得一提的是α_i对应着训练样本$(\boldsymbol{x}_i, y_i)$，在优化后得到的$\boldsymbol{\alpha}^*$中，绝大多数的$\alpha_i = 0$，只有少数$\alpha_i > 0$，这意味着m个训练样本中只有少数$\alpha_i > 0$对应的训练样本$(\boldsymbol{x}_i, y_i)$对最后的分割超平面起作用，被称为支持向量（support vector）。而这些样本也正位于$\boldsymbol{w}^{*T}\boldsymbol{x} + b^* = \pm 1$这两个超平面上。

1.3.2 使用 KKT 条件进行求解

1.3.2.1 KKT 条件

“KKT（Karush-Kuhn-Tucker）条件”是最优化理论的重要工具之一，常用来求解带有等式和不等式约束的非线性优化问题，是 Karush(1939)以及 Kuhn 和 Tucker(1951)先后独立发表的，是拉格朗日乘子法的推广。

批注 [MOU2]：前者工作是一篇硕士论文，之后，后二人独立发表，并引起关注

假设 f、g、h 是连续可导函数，面向如下带有等式和不等式约束的优化问题：

$$\begin{aligned} &\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ s.t.\quad &g_i(\boldsymbol{x}) \leqslant 0 \quad i=1,\cdots,p \\ &h_j(\boldsymbol{x}) = 0 \quad j=1,\cdots,q \end{aligned}$$

通过引入拉格朗日乘子$\boldsymbol{\alpha}$和\beta，并构造拉格朗日函数，将等式约束和不等式约束整合进优化目标，形成无约束优化问题：

$$L(\boldsymbol{x},\boldsymbol{\alpha},\boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^{p}\alpha_i g_i(\boldsymbol{x}) + \sum_{j=1}^{q}\beta_j h_j(\boldsymbol{x})$$

该问题最优解的必要条件为：

$$\begin{cases} \nabla_{\boldsymbol{x}} L(\boldsymbol{x},\boldsymbol{\alpha},\boldsymbol{\mu}) &= 0 \\ \alpha_i \geqslant 0 \\ \alpha_i g_i(\boldsymbol{x}) = 0 \\ g_i(\boldsymbol{x}) \leqslant 0 \\ h_j(\boldsymbol{x}) = 0 \end{cases}$$

如果$\boldsymbol{f(x)}$是凸函数，那么上述条件则是最优解的充分必要条件。

需要注意的是，定理中的 KKT 条件是取得极值的必要条件；但当优化目标是凸优化问题时，则 KKT 条件是取得最小值的充分必要条件。

另外，也可以检查候选解符合或违背 KKT 条件的程度，来帮助优化算法寻找更优的解。

1.3.2.2 使用原始问题 KKT 条件进行求解

根据式 4 中的原始优化目标函数，可列出式 4 最优解的 KKT 条件：

$$\begin{cases} \nabla_{\boldsymbol{w}} L(\boldsymbol{w},b,\boldsymbol{\alpha}) = 0 \\ \nabla_{b} L(\boldsymbol{w},b,\boldsymbol{\alpha}) = 0 \\ \alpha_i \geqslant 0 \\ \alpha_i(1-y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)) = 0 \\ 1-y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b) \leqslant 0 \end{cases}$$

因为式 4 的优化目标为关于 w 的二次凸函数，所以上述 KKT 条件为最优解的充要条件。因此，根据第一个条件，可解出 w*。

根据$\boldsymbol{w}^* = \sum_{i=1}^{m}\alpha_i^* y_i \boldsymbol{x}_i$可知存在\alpha_i>0，因为如果所有\alpha_i=0，则 w*=0，此解无意义。然后，根据第四个条件，当\alpha_i>0 时，可知$1-y_i(\boldsymbol{w}^{*T}\boldsymbol{x}_i+b^*)=0$，即在上一节中计算 b*时使用的$y_i = \boldsymbol{w}^{*T}\boldsymbol{x}_i + b^*$，其中，$y_i \in \{-1,+1\}$。

2、软间隔支持向量机

前文所提线性可分情况是肯定存在一个超平面能将正负两类样本完全划分开。但更多的情况是总有少量或部分样本使得约束$y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)\geqslant 1$不成立，这种情况被称为线性不可分。因此在线性可分 SVM 的基础上，引入松弛因子$\xi\geqslant 0$，将约束放松为$y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)\geqslant 1-\xi_i$。

松弛因子的引入即使在线性可分情况下也有积极意义，尤其由于个别离群样本导致的间隔十分狭窄，导致模型过拟合。因此，通过引入松弛因子适当容忍少数训练样本错分，从而使模型预测性能得到保证是个合适的做法。相应地，线性可分支持向量机中的间隔被称为硬间隔（hard margin），容忍少数错分的间隔被称为软间隔（soft margin）。

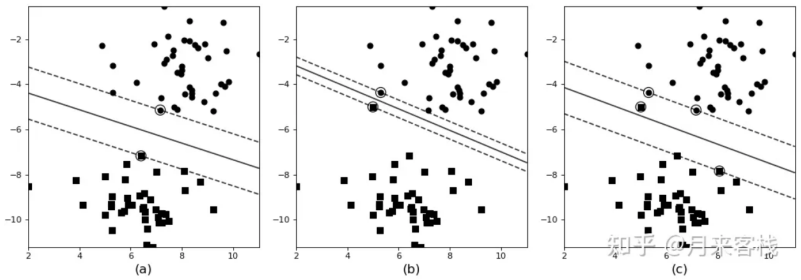
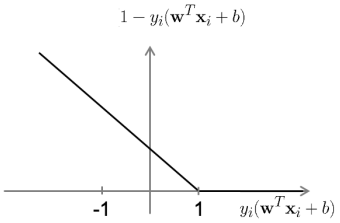
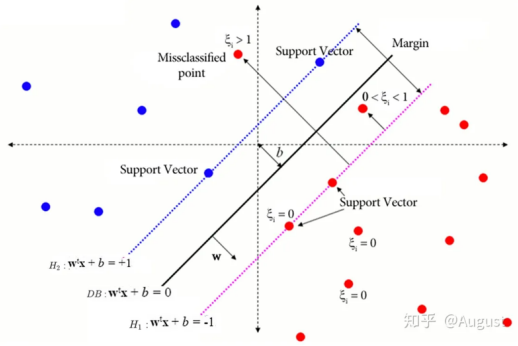


图 1. SVM软间隔与硬间隔图

2.1 代价函数

代价函数中的训练样本误差从$\sum_{i=1}^{m}\alpha_i(1-y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b))$变为$C\sum_{i=1}^{m}\max(0,1-y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b))$，后者也被称为折页损失或Hinge损失（如图*所示）。相应地，软间隔支持向量机的代价函数可写为$L(\boldsymbol{w},b)=\frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{i=1}^{m}\max(0,1-y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b))$。为了便于后续优化求解，引入松弛变量(slack variable) $\xi_i\geqslant 0$，代价函数被改写为式**，其拉格朗日函数为式**。





$$\min_{\mathbf{w},b,\xi_i} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i$$
$$s.t. \quad y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geqslant 1 - \xi_i$$
$$\xi_i \geqslant 0, i = 1, 2, \cdots, m$$

式（带松弛变量）

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i + \sum_{i=1}^{m}\alpha_i(1 - \xi_i - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)) - \sum_{i=1}^{m}\beta_i\xi_i$$

式（拉格朗日函数）

2.2 最小化代价函数

类似于线性可分 SVM，通过 $\max_{\alpha,\beta}\min_{w,b,\xi} L(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\beta})$ 来最小化拉格朗日函数：先固定拉格朗日乘子 alpha 和 mu，最小化式（拉格朗日函数）；然后再调整 alpha 和 mu 进行最大化。具体地，令 $L(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu})$ 相对 w、b 和\xi 求导为 0 可得：

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{w}} = \boldsymbol{w}^\top - \sum_{n=1}^{N}\alpha_n y_n \boldsymbol{x}_n{}^\top, \qquad \boldsymbol{w} = \sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i$$
$$\frac{\partial \mathfrak{L}}{\partial b} = \sum_{n=1}^{N}\alpha_n y_n, \qquad \sum_{i=1}^{m}\alpha_i y_i = 0$$
$$\frac{\partial \mathfrak{L}}{\partial \xi_n} = C - \alpha_n - \gamma_n. \quad \Rightarrow \quad C = \alpha_i + \beta_i$$

将其带入式（拉格朗日函数）可得：

$$\mathfrak{D}(\xi,\alpha,\gamma)=\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}y_iy_j\alpha_i\alpha_j\left\langle\boldsymbol{x}_i,\boldsymbol{x}_j\right\rangle-\sum_{i=1}^{N}y_i\alpha_i\left\langle\sum_{j=1}^{N}y_j\alpha_j\boldsymbol{x}_j,\boldsymbol{x}_i\right\rangle$$
$$+C\sum_{i=1}^{N}\xi_i-b\sum_{i=1}^{N}y_i\alpha_i+\sum_{i=1}^{N}\alpha_i-\sum_{i=1}^{N}\alpha_i\xi_i-\sum_{i=1}^{N}\gamma_i\xi_i\,. \tag{12.39}$$
$$=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}y_iy_j\alpha_i\alpha_j\left\langle\boldsymbol{x}_i,\boldsymbol{x}_j\right\rangle+\sum_{i=1}^{N}\alpha_i+\sum_{i=1}^{N}(C-\alpha_i-\gamma_i)\xi_i$$

由\sum\alpha y_i =0 和 C=\alpha_i+\beta_i，可得式（带松弛变量）的对偶问题：

$$\begin{aligned}\max_{\boldsymbol{\alpha}}\quad&\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_jy_iy_j\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j\\ \text{s.t.}\quad&\sum_{i=1}^{m}\alpha_iy_i=0\,,\\ &0\leqslant\alpha_i\leqslant C\,,\quad i=1,2,\ldots,m\,.\end{aligned}$$

使用二次规划可以求解得到\alpha^*，进而根据求导（w)式，可得 w^*。再根据 KKT 条件。

$$\begin{cases}\nabla_{\boldsymbol{w}}L(\boldsymbol{w},b,\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta})=0\\ \nabla_bL(\boldsymbol{w},b,\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta})=0\\ \nabla_{\xi}L(\boldsymbol{w},b,\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta})=0\\ \alpha_i\geqslant0\\ 1-\xi_i-y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)\leqslant0\\ \alpha_i(1-\xi_i-y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b))=0\\ \beta_i\geqslant0\\ \xi_i\geqslant0\\ \beta_i\xi_i=0\end{cases}$$

由 KKT 条件可知存在一个\alpha_j^*，0<\alpha_j^*<C，此时\beta_j>0，所以\xi_j=0，因此，b*应满足 $1-y_j(\boldsymbol{w}^{*T}\boldsymbol{x}_j+b^*)=0$，即 $\boldsymbol{w}^{*T}\boldsymbol{x}_j+b^*=y_j$。据此可得：

$$b^*=y_j-\boldsymbol{w}^{*T}\boldsymbol{x}_j=y_j-\sum_{i=1}^{m}\alpha_i^*y_i\boldsymbol{x}_i^T\boldsymbol{x}_j$$

最终得到形式如式（判决模型）的软间隔支持向量机。

值得一提的是，拉格朗日乘子\alpha_i 依旧是影响判决模型的主要因素。从 w*和 b*来看，当\alpha_i=0 时，对应的训练样本（xi，yi）对判决模型没有影响；当 0<\alpha_i<C 时，必有\beta_i=C-\alpha_i>0，由 KKT 条件最后一项可知\xi_i=0，对应的样本 x_i 则在最大间隔上，为支持向量。当\alpha_i=C 时，根据 KKT 条件 3 可知\beta_i=0，对应样本在\xi_i<1 时落在间隔内，如果\xi_i>1 则样本落入对面间隔外，即错分。

3、核函数

软间隔 SVM 能够解决一定程度的线性不可分问题，仍属线性模型。对于形如二次函数、指数函数等非线性判决超平面，常使用带有核函数(kernel)的 SVM 来进行解决。其核心思想是将在欧式空间下非线性可分的两类样本经过空间变换后变得线性可分。

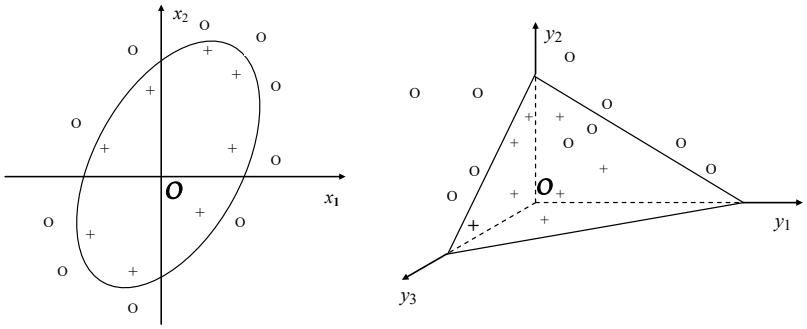
使用空间变换将问题转换为线性可分问题

例如，下图(a)所示的数据集是一个可使用二次函数进行判决的非线性分类问题。通过变换\phi(x)数据集在另一个由 x1^2 、x2^2 和\sqrt{2}x1x2 张成的空间内则能找到平面将两类样本分隔开。

$$w_1x_1^2 + w_2x_2^2 + \sqrt{2}w_3x_1x_2 + b = 0 \qquad (4\text{-}1)$$

$$\phi(\vec{x}) = \phi(x_1, x_2) = (y_1, y_2, y_3) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \qquad (4\text{-}2)$$

$$w_1y_1 + w_2y_2 + w_3y_3 + b = 0 \qquad (4\text{-}3)$$



（a）在 x_1 和 x_2 张成的空间中　　（b）在 $x_1^2, x_2^2, \sqrt{2}x_1x_2$ 张成的空间中

图 4.1 核函数变换前后样本在空间中的分布情况

需要指出的是为各类数据集设计合适的变换函数\phi(x)使变换后的样本在空间中线性可分是一件困难的事情。

直接使用核函数优化对偶问题

在支持向量机这一具体模型中，考虑到对偶式（*）和（*）中样本 x_i 和 x_j 对目标函数的主要影响为二者内积（inner product）或点积（dot product）。因此，直接考虑样本变换后的内积则更为直接高效。

相应地，对偶问题中的 $\max_{\alpha}\left(\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j\right)$ 则变换为式**

$$\max_{\alpha}\left(\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i,\boldsymbol{x}_j)\right)$$

其中，$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\langle\phi(\boldsymbol{x}_i),\phi(\boldsymbol{x}_j)\rangle=\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$，\left<,\right>为向量内积。

在非线性 SVM 模型的实际优化中，跳过空间映射函数\phi(x)，直接计算 $\langle\phi(\boldsymbol{x}_i),\phi(\boldsymbol{x}_j)\rangle$。需要说明的是，使用核函数的模型优化其实是隐式地在变换后空间进行的，且不需要明确定义其特征空间和映射函数\phi(x)。这种方法被称为核方法(kernel method)，也被用于扩展线性判别分析和线性回归等线性模型的非线性能力。

常见的核函数有：

1.线性核：$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\boldsymbol{x}_i^T\boldsymbol{x}_j$

2.多项式核：$K(\mathbf{x},\mathbf{y})=(\gamma\langle\mathbf{x},\mathbf{y}\rangle+c)^d$

3.径向基核（常使用高斯核）：$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\exp(\frac{||\boldsymbol{x}_i-\boldsymbol{x}_j||^2}{2\sigma^2})$

4. Sigmoid 核：$K(\mathbf{x},\mathbf{y})=\tanh(\gamma\langle\mathbf{x},\mathbf{y}\rangle+c)$

从上述核函数的主体是向量内积或距离来看，核函数也可被看作是相似度度量函数。

4、基于 sklearn 的 SVM 模型

SVM 的优化主要是对代价函数的对偶问题使用 SMO 算法或二次规划算法，这些代码实现难度较大。因此，在应用中多使用 sklearn 库里封装好的的 SVC。核心函数为构造函数 linearSVC()或 SVC()、训练函数 fit()和预测函数 predict()。其中，构造函数使用键值方式的参数以适应不同类型的 SVM 分类器的初始化。

针对线性可分的 SVM 可以调用 svm.linearSVC(penealty=l2, loss=hinge)来构造模型的代价函数，通过 svm.SVC(C=1, kernel='rbf', gamma=0.5)构造径向基函数（常用高斯函数）的核函数。在 sklearn 中径向基函数被定义为 $exp(-\gamma||x-x'||^2),\gamma>0$，如果使用高

斯核函数的参数\sigma，需要通过\gamma = 1 / (2\sigma^2)进行转换。

```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC(C=1, kernel='rbf', gamma=np.power(0.1, -2)/2) #对应
sigma=0.1
>>> clf.fit(X, y)
>>> clf.predict([[2., 2.]])
```

sklearn.svm.SVC

class sklearn.svm.**SVC**(*, *C=1.0*, *kernel='rbf'*, *degree=3*, *gamma='scale'*, *coef0=0.0*, *shrinking=True*, *probability=False*, *tol=0.001*, *cache_size=200*, *class_weight=None*, *verbose=False*, *max_iter=-1*, *decision_function_shape='ovr'*, *break_ties=False*, *random_state=None*) [source]

关于 SVC 的更多参数，请参见 sklearn.svm.SVC 文档(https://scikit-learn.org/stable)

==========

参考内容

SVM 软间隔与对偶问题，https://zhuanlan.zhihu.com/p/503762216，里面还有 sklearn 的示例代码，和拉格朗日法的例子，非常好。