



第7章 支持向量机

支持向量机(support vector machines. SVM)

- 线性可分支持向量机(linear support vector machine in linearly separable case).
硬间隔最大化(hard margin maximization) ;
- 线性支持向量机(linear support vector machine)
训练数据近似线性可分时, 通过软间隔最大化(soft margin maximization) ;
- 非线性支持向量机(non-linear support vector machine)
当训练数据线性不可分时, 通过使用核技巧(kernel trick)

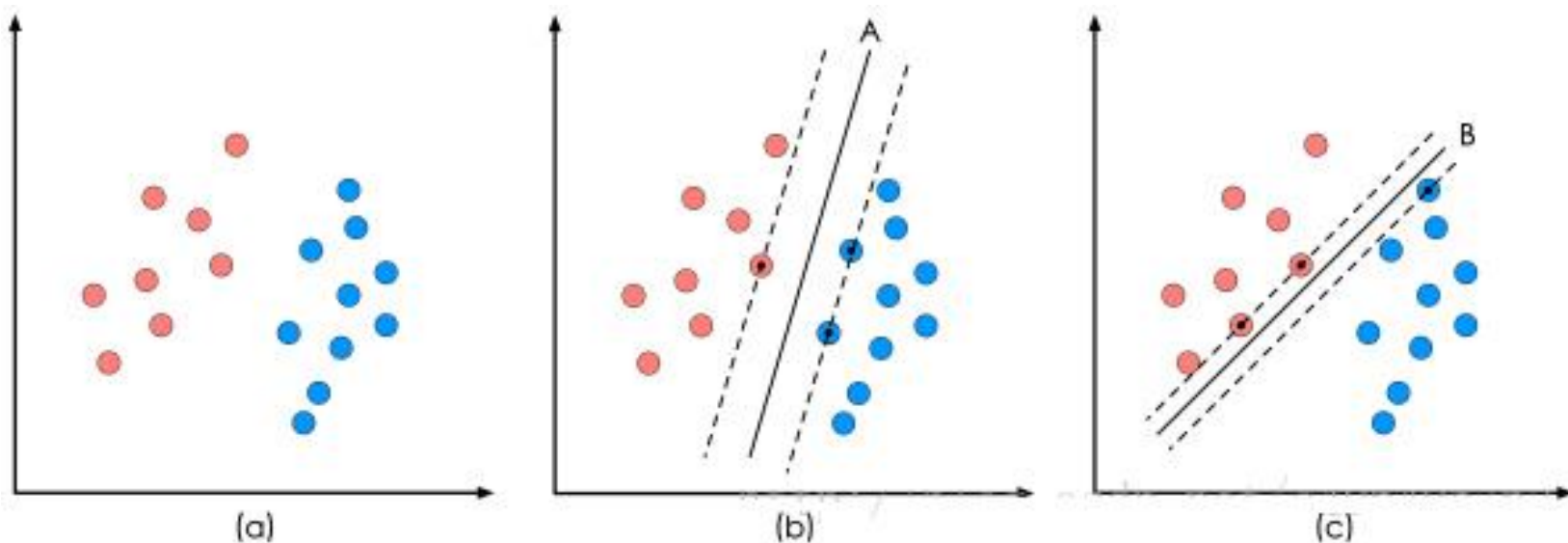
主要内容

- 线性可分支持向量机和线性支持向量机
- 非线性支持向量机与核函数

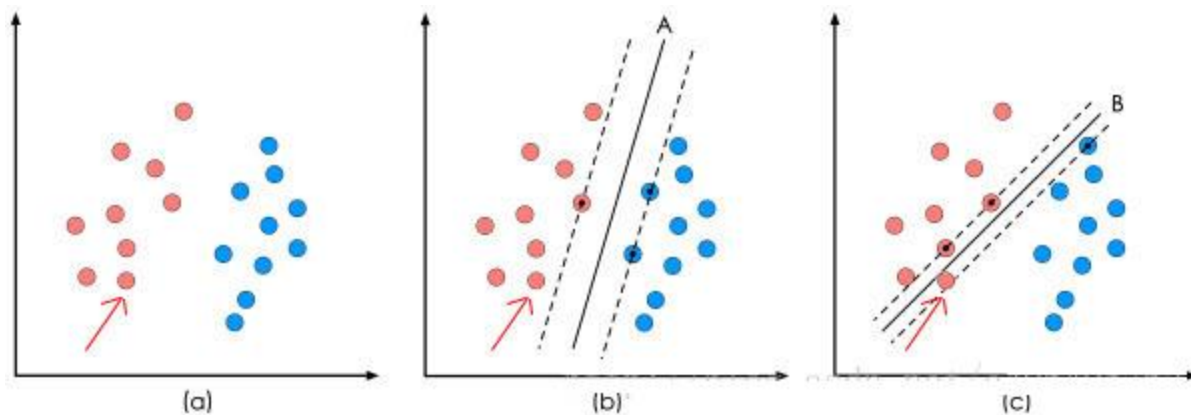


7.1 线性可分支支持向量机和 线性支持向量机

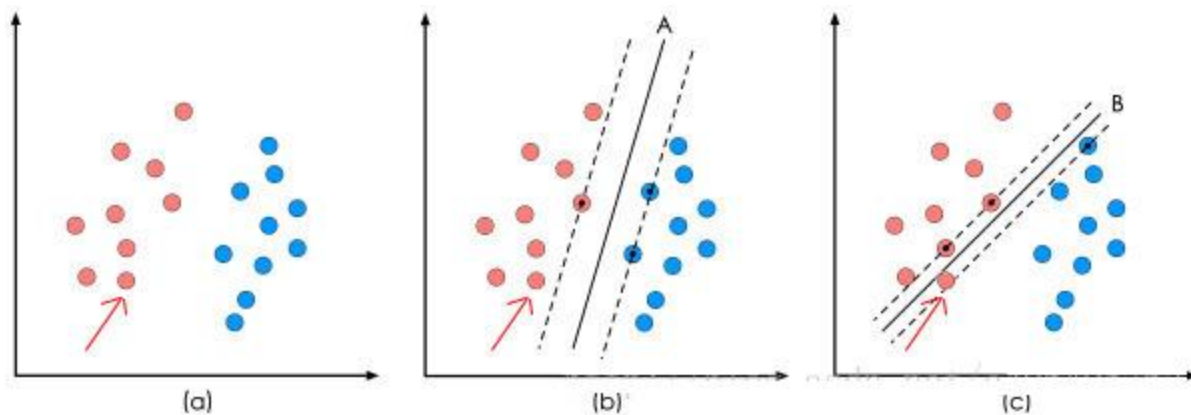
1、线性可分支持向量机



上图中的(a)是已有的数据，红色和蓝色分别代表两个不同的类别。数据显然是线性可分的，但是将两类数据点分开的直线显然不止一条。上图的(b)和(c)分别给出了B、C两种不同的分类方案，其中黑色实线为分界线，术语称为“**决策面**”。每个决策面对应了一个线性分类器。虽然从分类结果上看，分类器A和分类器B的效果是相同的。但是他们的性能是有差距的。



在“决策面”不变的情况下，我又添加了一个红点。可以看到，分类器B依然能很好的分类结果，而分类器C则出现了分类错误。显然分类器B的“决策面”放置的位置优于分类器C的“决策面”放置的位置，SVM算法也是这么认为的，它的依据就是分类器B的分类间隔比分类器C的分类间隔大。这里涉及到第一个SVM独有的概念“分类间隔”。



在保证决策面方向不变且不会出现错分样本的情况下移动决策面，会在原来的决策面两侧找到两个极限位置(越过该位置就会产生错分现象)，如虚线所示。虚线的位置由决策面的方向和距离原决策面最近的几个样本的位置决定。而这两条平行虚线正中间的分界线就是在保持当前决策面方向不变的前提下的最优决策面。两条虚线之间的垂直距离就是这个最优决策面对应的分类间隔。显然每一个可能把数据集正确分开的方向都有一个最优决策面（有些方向无论如何移动决策面的位置也不可能将两类样本完全分开），而不同方向的最优决策面的分类间隔通常是不同的，那个具有“最大间隔”的决策面就是SVM要寻找的最优解。而这个真正的最优解对应的两侧虚线所穿过的样本点，就是SVM中的支持样本点，称为“**支持向量**”。

2、数学建模

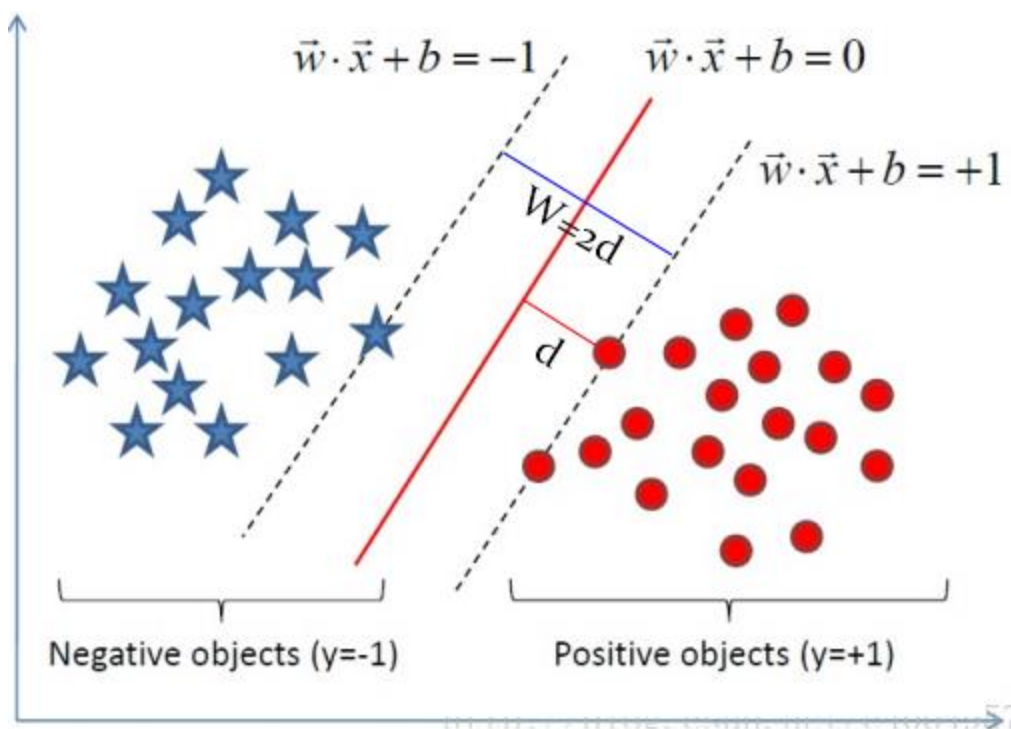
求解这个“决策面”的过程，就是最优化。一个最优化问题通常有两个基本的因素：

- 1) 目标函数，也就是你希望什么东西的什么指标达到最好；
- 2) 优化对象，你期望通过改变哪些因素来使你的目标函数达到最优。在线性SVM算法中，目标函数显然就是那个“分类间隔”，而优化对象则是决策面。所以要对SVM问题进行数学建模，首先要对上述两个对象（“分类间隔”和“决策面”）进行数学描述。按照一般的思维习惯，我们先描述决策面。

数学建模的时候，先在二维空间建模，然后再推广到多维。

“分类间隔”方程

现在，我们依然对于一个二维平面的简单例子进行推导。



间隔的大小实际上就是支持向量对应的样本点到决策面的距离的二倍。

图中的距离d我们怎么求？

点 (x_0, y_0) 到直线 $AX + BY + C = 0$ 的距离

$$d = \left| \frac{Ax_0 + By_0 + C}{\sqrt{A^2 + B^2}} \right|$$

现在，将直线方程扩展到多维，求得我们现在的超平面方程，对公式进行如下变形：

$$d = \frac{|w^T X + b|}{\|w\|}$$

这个d就是“分类间隔”。其中 $\|w\|$ 表示w的二范数，求所有元素的平方和，然后再开方。

对于二维平面： $w = (w_1, w_2)$

$$\|w\| = \sqrt{w_1^2 + w_2^2}$$

目的是为了找出一个分类效果好的超平面作为分类器。分类器的好坏的评定依据是分类间隔 $W=2d$ 的大小，即分类间隔 W 越大，我们认为这个超平面的分类效果越好。此时，求解超平面的问题就变成了求解分类间隔 W 最大化的为题。 W 的最大化也就是 d 最大化的。

3、约束条件

看起来，我们已经顺利获得了目标函数的数学形式。但是为了求解 w 的最大值。我们不得不面对如下问题：

(1)如何判断超平面是否将样本点正确分类？

(2)求距离 d 的最大值，首先需要找到支持向量上的点，怎么在众多的点中选出支持向量上的点呢？

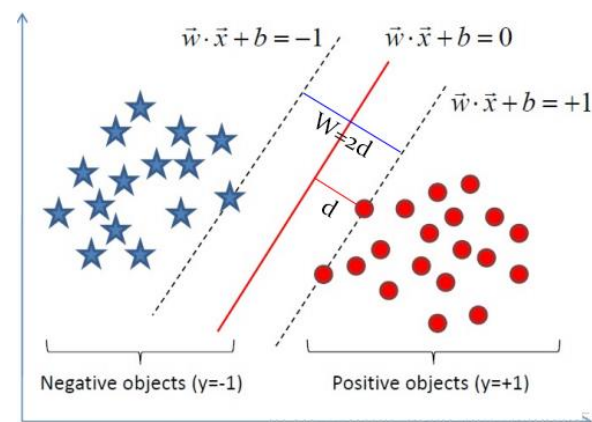
上述需要面对的问题就是约束条件，也就是说优化的变量 d 的取值范围受到了限制和约束。事实上约束条件一直是最优化问题里最让人头疼的东西。但既然已经知道了这些约束条件确实存在，就不得不用数学语言对他们进行描述。但SVM算法通过一些巧妙的小技巧，将这些约束条件融合到一个不等式里面。

二维平面上有两种点，我们分别对它们进行标记：

红颜色的圆点标记为1，我们人为规定其为正样本；
蓝颜色的五角星标记为-1，我们人为规定其为负样本。

对每个样本点 x_i 加上一个类别标签 y_i ：

$$y_i = \begin{cases} +1 & \text{红色点} \\ -1 & \text{蓝色点} \end{cases}$$



如果我们的超平面方程能够完全正确地对上图的样本点进行分类，就会满足下面的方程：

$$\begin{cases} W^T X_i + b > 0 & y_i = +1 \\ W^T X_i + b < 0 & y_i = -1 \end{cases}$$

上述公式推导

假设决策面正好处于间隔区域的中轴线上，并且相应的支持向量对应的样本点到决策面的距离为 d ，那么公式进一步写成：

$$\begin{cases} \frac{w^T X_i + b}{\|w\|} \geq d & \forall y_i = +1 \\ \frac{w^T X_i + b}{\|w\|} \leq -d & \forall y_i = -1 \end{cases}$$

对于所有分类标签为1的样本点，它们到直线的距离都大于等于 d (支持向量上的样本点到超平面的距离)。对于所有分类标签为-1的样本点，它们到直线的距离都小于等于 d 。公式两边都除以 d ，就可以得到：

$$\begin{cases} w_d^T X_i + b_d \geq 1 & \forall y_i = +1 \\ w_d^T X_i + b_d \leq -1 & \forall y_i = -1 \end{cases} \quad w_d = \frac{w}{\|w\|d}, \quad b_d = \frac{b}{d}$$

因为 $\|w\|$ 和 d 都是标量。所上述公式的两个矢量，依然描述一条直线的法向量和截距。

$$w_d^T x + b_d = 0$$

上述两个公式，都是描述一条直线，数学模型代表的意义是一样的。现在，让对 w_d 和 b_d 重新起个名字，就叫它们 w 和 b 。因此，对于存在分类间隔的两类样本点，我们一定可以找到一些超平面，使其对于所有的样本点均满足下面的条件：

$$\begin{cases} w^T x_i + b \geq 1 & \forall y_i = +1 \\ w^T x_i + b \leq -1 & \forall y_i = -1 \end{cases}$$

上述方程即给出了SVM最优化问题的约束条件。这时候，可能有人会问了，为什么标记为1和-1呢？因为这样标记方便将上述方程变成如下形式：

$$y_i (w^T x_i + b) \geq 1 \quad \forall x_i$$

正是因为标签为1和-1，才方便我们将约束条件变成一个约束方程，从而方便计算。

4、线性SVM优化问题基本描述

目标函数

$$d = \frac{|w^T X + b|}{\|w\|}$$

优化目标是是**d最大化**。用支持向量上的样本点求解d的最大化的问题的。那么支持向量上的样本点有什么特点呢？

$$|w^T X_i + b| = 1 \quad \forall X_i$$

所有支持向量上的样本点，都满足如上公式。

目标函数简化后

$$d = \frac{1}{\|w\|}$$

只关心支持向量上的点。求解d的最大化问题变成了 $\|w\|$ 的最小化问题。
进而 $\|w\|$ 的最小化问题等效于

$$\min \frac{1}{2} \|w\|^2$$

为了在进行最优化的过程中对目标函数求导时比较方便，但这绝对不影响最优化问题最后的求解。将最终的目标函数和约束条件放在一起进行描述:

$$\min \frac{1}{2} \|w\|^2$$

$$s.t. \quad y_i (w^T x_i + b) \geq 1, i = 1, 2, \dots, n$$

n 是样本点的总个数，缩写s.t.表示“Subject to”，是“服从某某条件”的意思。上述公式描述的是一个典型的不等式约束条件下的二次型函数优化问题，同时也是支持向量机的基本数学模型。

已经得到支持向量机的基本数学模型，接下来的问题就是如何根据数学模型，求得最优解。求解方法有一个前提，就是目标函数必须是**凸函数**。理解凸函数先明确另一个概念：**凸集**。在凸几何中，凸集(convex set)是在凸组合下闭合的放射空间的子集。看一幅图可能更容易理解：

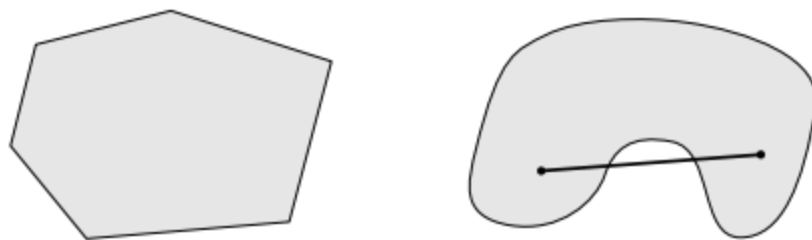


Figure 1: Examples of a convex set (a) and a non-convex set (b).

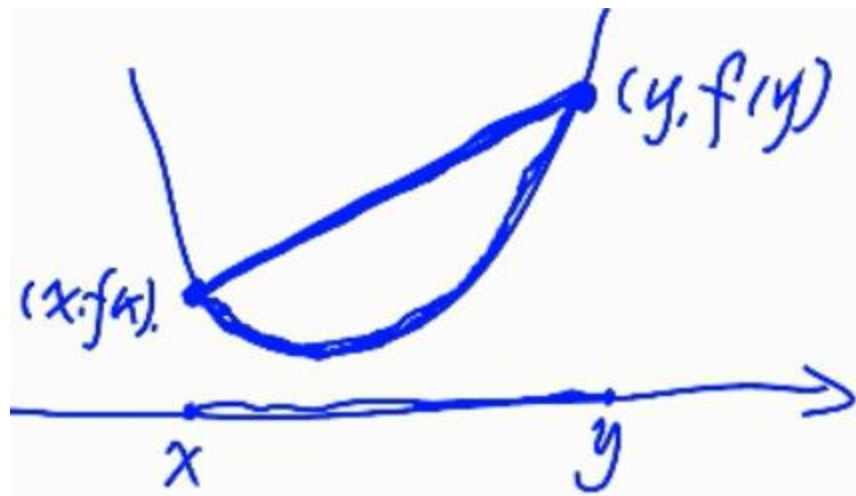
左右量图都是一个集合。如果**集合中任意2个元素连线上的点也在集合中**，那么这个集合就是凸集。显然，上图中的左图是一个凸集，上图中的右图是一个非凸集。

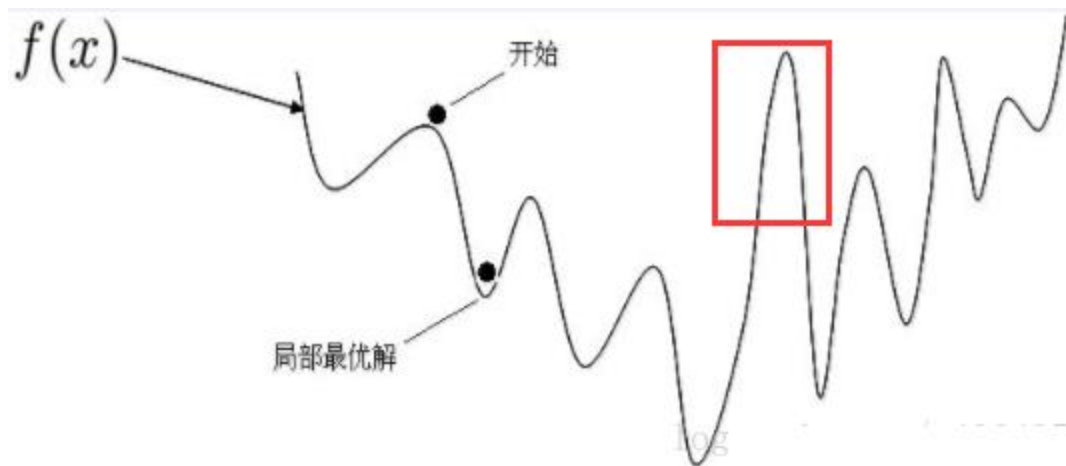
定义：若这里凸集C即某个区间L，那么，设函数f为定义在区间L上的函数，若对L上的任意两点 x_1 ， x_2 和任意的实数 λ ， λ 属于 $(0,1)$ ，总有：

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2)$$

则函数f称为L上的凸函数，当且仅当其上镜图（在函数图像上方的点集）为一个凸集。

几何意义：**函数任意两点连线上的值大于对应自变量处的函数值。**





像上图这样的函数，它整体就是一个**非凸函数**，无法获得全局最优解的，只能获得**局部最优解**。比如红框内的部分，如果单独拿出来，它就是一个凸函数。

对于的目标函数：

$$\min \frac{1}{2} \|w\|^2$$

很显然，它是一个凸函数。所以，可以使用接下来讲述的方法求取最优解。

最优化问题有如下几类：

无约束优化问题，可以写为

$$\min f(x)$$

有等式约束的优化问题，可以写为：

$$\min f(x)$$

$$s.t. \quad h_j(x) = 0, \quad j = 1, 2, \dots, n$$

有不等式约束的优化问题，可以写为：

$$\min f(x)$$

$$s.t. \quad g_i(x) \leq 0 \quad i = 1, 2, \dots, m$$

$$h_j(x) = 0 \quad j = 1, 2, \dots, n$$

对于第(a)类的优化问题，是我们高中经常使用的求函数的极值的方法。常用的方法包括（1）直接法：梯度=0，求驻点。（2）迭代法：梯度下降法、牛顿法、拟牛顿法。

对于第(b)类的优化问题，常常使用的方法就是**拉格朗日乘子法**（Lagrange Multiplier），即把等式约束 $h_i(x)$ 用一个系数与 $f(x)$ 写为一个式子，称为拉格朗日函数，而系数称为拉格朗日乘子。通过拉格朗日函数对各个变量求导，令其为零，可以求得候选值集合，然后验证求得最优值。

$$L(x, \beta) = f(x) + \sum_{j=0}^n \beta_j h_j$$

对于第(c)类的优化问题，常常使用的方法就是KKT条件。同样地，把所有的等式、不等式约束与 $f(x)$ 写为一个式子，也叫**拉格朗日函数**，系数也称拉格朗日乘子，通过一些条件，可以求出最优值的必要条件，这个条件称为KKT条件。

$$L(x, \lambda, \mu) = f(x) + \sum_{j=0}^n \beta_j h_j(x) + \sum_{i=0}^m \alpha_i g_i(x)$$

$$\frac{\partial L(x, \beta, \alpha)}{\partial x} = 0, \quad h_j(x) = 0, \quad \alpha_i \geq 0, \quad g_i(x) \leq 0 \quad \alpha_i g_i(x) = 0$$

现在再看一下最优化问题：

$$\min \frac{1}{2} \|w\|^2$$

$$s.t. \quad y_i (w^T x_i + b) \geq 1, \quad i = 1, 2, \dots, n$$

要求解的是最小化问题，所以一个直观的想法是如果能够构造一个函数，使得该函数在可行解区域内与原目标函数完全一致，而在可行解区域外的数值非常大，甚至是无穷大，那么这个没有约束条件的新目标函数的优化问题就与原来有约束条件的原始目标函数的优化问题是等价的问题。这就是使用拉格朗日方程的目的，它将约束条件放到目标函数中，从而将有约束优化问题转换为无约束优化问题。

拉格朗日函数

使用拉格朗日获得的函数，使用求导的方法求解依然困难。进而，需要对问题再进行一次转换，即使用一个数学技巧：拉格朗日对偶。

所以，显而易见的是，在拉格朗日优化我们的问题这个道路上，需要进行下面二个步骤：

- (1)将有约束的原始目标函数转换为无约束的新构造的拉格朗日目标函数
- (2)使用拉格朗日对偶性，将不易求解的优化问题转化为易求解的优化

第一步：将有约束的原始目标函数转换为无约束的新构造的拉格朗日目标函数

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^N \alpha_i [y_i (w^T x_i + b) - 1]$$

其中 α_i 是拉格朗日乘子， α_i 大于等于0，是我们构造新目标函数时引入的系数变量(我们自己设置)。令：

$$\theta_P(w) = \max_{\alpha_i \geq 0} L(w, b, \alpha)$$

当样本点不满足约束条件时，即在**可行解**区域外：

$$y_i(w^T x_i + b) < 1$$

此时，将 α_i 设置为正无穷，此时 $\theta_p(w)$ 显然也是正无穷。

当样本点满足约束条件时，即在可行解区域内：

$$y_i(w^T x_i + b) \geq 1$$

此时，显然 $\theta(w)$ 为原目标函数本身。将上述两种情况结合一下，就得到了新的目标函数：

$$\theta_P(w) = \begin{cases} \frac{1}{2} \|w\|^2 & x \in \text{可行域} \\ +\infty & x \in \text{非可行区域} \end{cases}$$

初衷，就是为了建立一个在**可行解区域内**与原目标函数相同，在可行解区域外函数值趋近于无穷大的新函数现在，问题变成了求新目标函数的最小值，即：

$$\min_{w, b} \theta_p(w) = \min_{w, b} \max_{\alpha_i \geq 0} L(w, b, \alpha) = p^*$$

这里用 p^* 表示这个问题的最优值，且和最初的问题是等价的。

第二步：将不易求解的优化问题转化为易求解的优化

新目标函数，先求最大值，再求最小值。首先就要面对带有需要求解的参数 w 和 b 的方程，而 α_i 又是不等式约束，这个求解过程不好做。所以，需要使用拉格朗日函数对偶性，将最小和最大的位置交换一下，这样就变成：

$$\max_{\alpha_i \geq 0} \min_{w, b} L(w, b, \alpha) = d^*$$

交换以后的新问题是原始问题的对偶问题，这个新问题的最优值用 d^* 来表示。而且 $d^* \leq p^*$ 。关心的是 $d=p$ 的时候，这才是要的解。需要什么条件才能让 $d=p$ 呢？

首先必须满足这个优化问题是凸优化问题。
其次，需要满足KKT条件。

凸优化问题的定义是：求取最小值的目标函数为凸函数的一类优化问题。目标函数是凸函数，这个优化问题又是求最小值。所以最优化问题就是凸优化问题。

KKT条件

使用拉格朗日函数对的目标函数进行了处理，生成了一个新的目标函数。通过一些条件，可以求出最优值的必要条件，这个条件就是接下来要说的KKT条件。一个最优化模型能够表示成下列标准形式：

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & h_j(x) = 0, \quad j = 1, 2, \dots, p \\ & g_k(x) \leq 0, \quad k = 1, 2, \dots, q \\ & x \in X \subset R^n \end{aligned}$$

KKT条件的全称是Karush-Kuhn-Tucker条件，KKT条件是说最优值条件必须满足以下条件：

- 条件一：经过拉格朗日函数处理之后的新目标函数 $L(w, b, \alpha)$ 对 α 求导为零；
- 条件二： $h(x) = 0$ ；
- 条件三： $\alpha^* g(x) = 0$ ；

优化问题：

$$\min \frac{1}{2} \|w\|^2$$

$$s.t. \quad y_i (w^T x_i + b) \geq 1, i = 1, 2, \dots, n$$

凸优化问题和KKT都满足了，问题转换成了对偶问题。而求解这个对偶学习问题，可以分为三个步骤：首先要让 $L(w, b, \alpha)$ 关于 w 和 b 最小化，然后求对 α 的极大，最后利用SMO算法求解对偶问题中的拉格朗日乘子。

对偶问题求解

$$\max_{\alpha_i \geq 0} \min_{w, b} L(w, b, \alpha) = d^*$$

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i [y_i (w^T x_i + b) - 1]$$

首先固定 α ，要让 $L(w, b, \alpha)$ 关于 w 和 b 最小化，分别对 w 和 b 偏导数，令其等于0，即：

$$\begin{aligned} \nabla_w L(w, b, \alpha) &= w - \sum_{i=1}^N \alpha_i y_i x_i = 0 \\ \nabla_b L(w, b, \alpha) &= -\sum_{i=1}^N \alpha_i y_i = 0 \end{aligned} \quad \Rightarrow \quad \begin{aligned} w &= \sum_{i=1}^N \alpha_i y_i x_i \\ \sum_{i=1}^N \alpha_i y_i &= 0 \end{aligned}$$

将上述结果带回 $L(w,b,\alpha)$ 得到：

$$\begin{aligned} L(w, b, \alpha) &= \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \bullet x_j) - \sum_{i=1}^N \alpha_i [y_i ((\sum_{j=1}^N \alpha_j y_j x_j) \bullet x_i + b) - 1] \\ &= \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i^T x_j) \end{aligned}$$

上面的最后一个式子，此时的 $L(w,b,\alpha)$ 函数只含有一个变量，即 α_i 。

现在内侧的最小值求解完成，求解外侧的最大值，从上面的式子得到

$$\max_{\alpha} \left[\sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i^T x_j) \right]$$

$$s.t. \quad \alpha_i \geq 0, i = 1, 2, \dots, n$$

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

现在优化问题变成了如上的形式。对于这个问题，有更高效率的优化算法，即序列最小优化（SMO）算法。通过这个优化算法能得到 α ，再根据 α ，就可以求解出 w 和 b ，进而求得最初的目的：找到超平面，即**决策平面**。

软间隔

将目标函数变形，在前面增加一个符号，将最大值问题转换成最小值问题

$$\min_{\alpha} \left[\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i^T x_j) - \sum_{i=1}^N \alpha_i \right]$$

实际上，对于上述目标函数，是存在一个假设的，即数据100%线性可分。但是目前为止，几乎所有数据都不那么“干净”。这时就可以通过引入所谓的松弛变量(slack variable)，来允许有些数据点可以处于超平面的错误的一侧，为此要引入“软间隔”(soft margin)的概念。

具体来说，前面介绍的支持向量机形式是要求所有样本都满足约束，即所有样本都必须划分正确，这称为“硬间隔”（hard margin），而软间隔则是允许某些样本不满足约束： $y_i(w^T x_i + b) \geq 1$

引入松弛变量 $\xi_i \geq 0$ ，使得下式成立： $y_i(w^T x_i + b) \geq 1 - \xi_i$

在最大化间隔的同时，不满足约束条件的样本应该尽可能少，于是优化目标可以写为：

$$\min_{w, b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \quad (1)$$

(1) 式就是常见的软间隔支持向量机

与硬间隔时类似，这仍然是一个二次规划问题，于是类似

$$\max_{\alpha_i \geq 0} \min_{w, b} L(w, b, \alpha) = d *$$

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i [y_i (w^T x_i + b) - 1]$$

的获取方法，通过拉格朗日乘子法可得到：

$$\begin{aligned} L(w, b, \alpha, \varepsilon, u) = & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \\ & + \sum_{i=1}^m \alpha_i [1 - \xi_i - y_i (w^T x_i + b)] - \sum_{i=1}^m u_i \xi_i \quad (2) \end{aligned}$$

其中 α_i ， u_i 是拉格朗日乘子

令 (2) 式对 ω , b , ξ_i 的偏导数为0可得：

$$\begin{aligned} W &= \sum_{i=1}^m \alpha_i y_i X_i \\ 0 &= \sum_{i=1}^m \alpha_i y_i \\ C &= \alpha_i + u_i \end{aligned} \quad (3)$$

将上面3个式子带入 (2) 中得到对偶问题：

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j X_i^T X_j \\ s.t. \quad & \sum_{i=1}^m \alpha_i y_i = 0 \quad 0 \leq \alpha_i \leq C \end{aligned}$$

与硬间隔对比可以发现，优化目标仍然不变，只是约束条件有所改变。

根据KKT条件

$$\begin{cases} \alpha_i \geq 0, u_i \geq 0 \\ y_i f(x_i) - 1 + \zeta_i \geq 0 \\ \alpha_i [y_i f(x_i) - 1 + \zeta_i] = 0 \\ \zeta_i \geq 0, u_i \zeta_i = 0 \end{cases}$$

于是，对于任意训练样本，总有 $\alpha_i=0$ 或者 $y_i f(x_i)=1-\xi_i$ 。若 $\alpha_i=0$ ，则该样本不会对 $f(x)$ 有任何影响；若 $\alpha_i>0$ ，则必有 $y_i f(x_i)=1-\xi_i$ ，即该样本是支持向量。

并且，由(5)知道，若 $\alpha_i < C$ ，则 $\mu_i > 0$ ，进而有 $\xi_i = 0$ ，所以此时有： $y f(x) = 1$ ，即该样本恰好在最大间隔边界上；若 $\alpha_i = C$ ，则有 $\mu_i = 0$ ，此时若 $\xi_i \leq 1$ 则该样本落在最大间隔内部；若 $\xi_i > 1$ 则该样本被错误分类。由此可以看出，软间隔支持向量机的最终模型仅与支持向量有关

讨论：

$$\begin{array}{ll} \alpha_i = 0 & \Leftrightarrow y_i u_i \geq 1 \text{ 对于第1种情况，表明}\alpha_i\text{是正常分类，在边界内部；} \\ 0 < \alpha_i < C & \Leftrightarrow y_i u_i = 1 \text{ 对于第2种情况，表明}\alpha_i\text{是支持向量，在边界上；} \\ \alpha_i = C & \Leftrightarrow y_i u_i \leq 1 \text{ 对于第3种情况，表明}\alpha_i\text{是在两条边界之间。} \end{array}$$

最优解需要满足KKT条件，即上述3个条件都得满足，以下几种情况出现将会不满足：

$$y_i u_i \leq 1 \quad \alpha_i < C$$

$$y_i u_i \geq 1 \quad \alpha_i > 0$$

$$y_i u_i = 1 \quad \alpha_i = 0 \text{ 或者 } \alpha_i = C$$

也就是说，如果存在不能满足KKT条件的 α_i ，那么需要更新这些 α_i ，这是第一个约束条件。此外，更新的同时还要受到第二个约束条件的限制，即：

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

SMO算法

1996年，John Platt发布了一个称为SMO的强大算法，用于训练SVM。SM表示序列最小化(Sequential Minimal Optimizaion)。Platt的SMO算法是将大优化问题分解为多个小优化问题来求解的。这些小优化问题往往很容易求解，并且对它们进行顺序求解的结果与将它们作为整体来求解的结果完全一致的。在结果完全相同的同时，SMO算法的求解时间短很多。

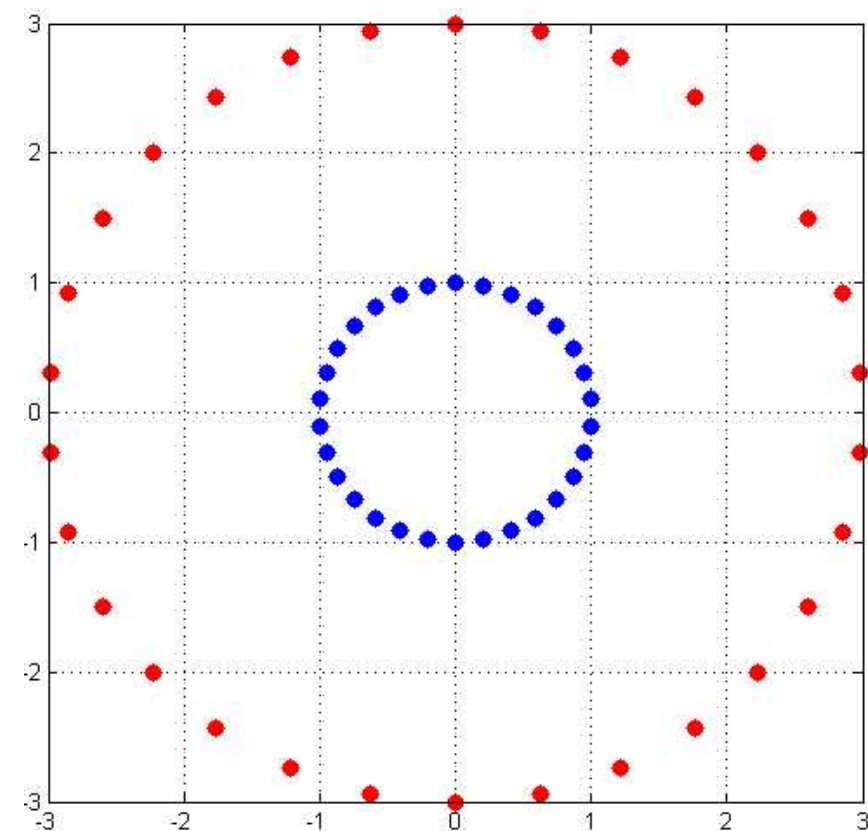
SMO算法的目标是求出一系列 α 和 b ，一旦求出了这些 α ，就很容易计算出权重向量 w 并得到分隔超平面。

SMO算法的工作原理是：每次循环中选择两个 α 进行优化处理。一旦找到了一对合适的 α ，那么就增大其中一个同时减小另一个。这里所谓的“合适”就是指两个 α 必须符合以下两个条件，条件之一就是两个 α 必须要在间隔边界之外，而且第二个条件则是这两个 α 还没有进行过区间化处理或者不在边界上。

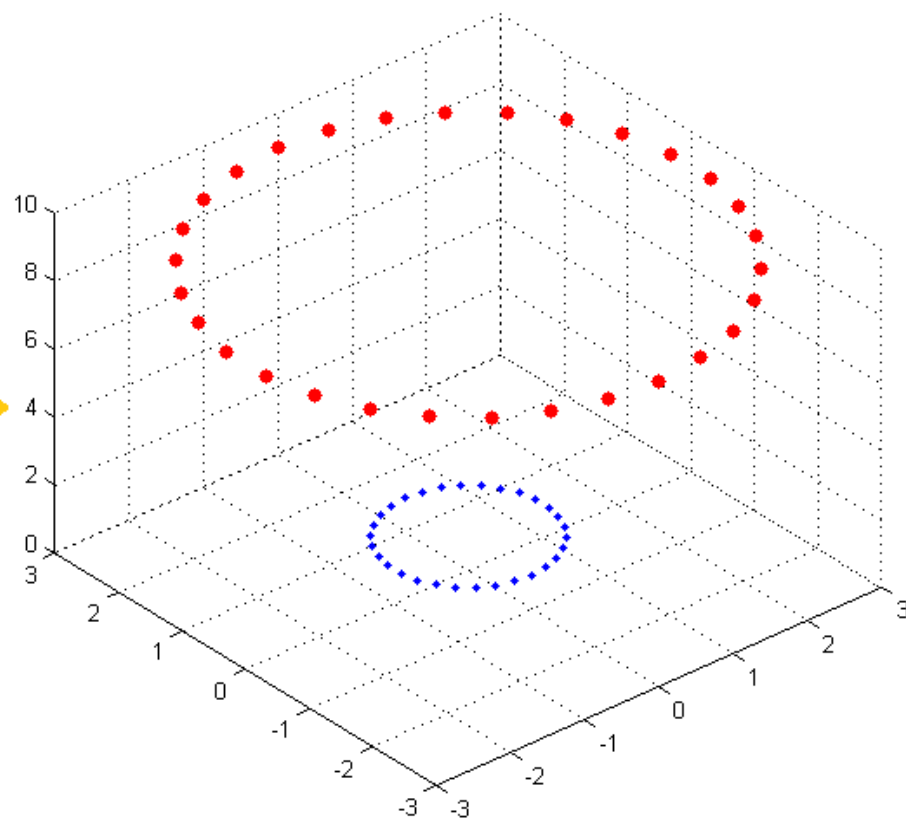


7.2非线性支持向量机 与核函数

对于以上所述的SVM，处理能力还是很弱，仅仅能处理线性可分的数据。如果数据线性不可分的时候，我们就将低维的数据映射向更高的维次，以此使数据重新线性可分。这转化的关键便是核函数。



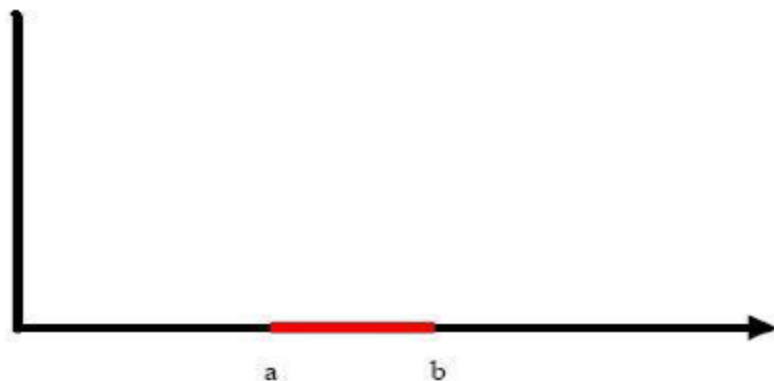
$$\{x, y \mid x^2 + y^2 = R^2\}$$



$$\{x, y, x^2 + y^2\}$$

核函数

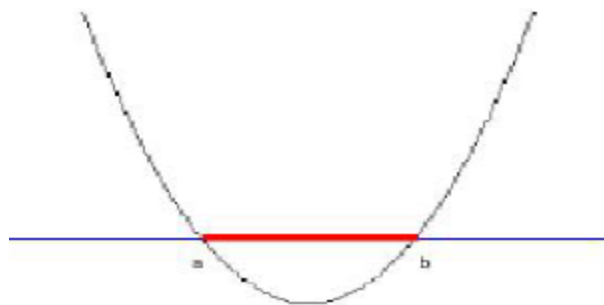
用一个二维平面中的分类问题作例子



把横轴上端点a和b之间红色部分里的所有点定为正类，两边的黑色部分里的点定为负类。试问能找到一个线性函数把两类正确分开么？

不能，因为二维空间里的线性函数就是指直线，显然找不到符合条件的直线。

但我们可以找到一条曲线，例如下面这一条：



显然通过点在这条曲线的上方还是下方就可以判断点所属的类别。这条曲线就是我们熟知的二次曲线，它的函数表达式是：

$$g(x) = c_2x^2 + c_1x + c_0$$

问题只是它不是一个线性函数，但是，做一下变换，新建一个向量y和a：

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ x_2 \end{bmatrix} \quad a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix}$$

这样g(x)就可以转化为 $f(y) = \langle a, y \rangle$ ，你可以把y和a分别回带一下，看看等不等于原来的g(x)。用内积的形式写你可能看不太清楚，实际上f(y)的形式就是：

$$g(x) = f(y) = a^T y$$

在任意维度的空间中，这种形式的函数都是一个线性函数，因为自变量y的次数不大于1。

原来在二维空间中一个线性不可分的问题，映射到高维空间后，变成了线性可分的！因此也形成了我们最初想解决线性不可分问题的基本思路——向高维空间转化，使其变得线性可分。

用一个具体文本分类的例子来看看这种向高维空间映射从而分类的方法如何运作，如果我们文本分类问题的原始空间是1000维的（即每个要被分类的文档被表示为一个1000维的向量），在这个维度上问题是线性不可分的。现在我們有一个2000维空间里的线性函数

$$f(x') = \langle w', x' \rangle + b$$


它能够將原问题变得可分。式中的 w' 和 x' 都是2000维的向量，只不过 w' 是定值，而 x' 是变量。现在我们的输入呢，是一个1000维的向量 x ，分类的过程是先把 x 变换为2000维的向量 x' ，然后求这个变换后的向量 x' 与向量 w' 的内积，再把这个内积的值和 b 相加，就得到了结果，看结果大于阈值还是小于阈值就得到了分类结果。

所以只需要关心那个高维空间里内积的值。而从理论上说， x' 是经由 x 变换来的，因此广义上可以把它叫做 x 的函数（因为有一个 x ，就确定了一个 x' ），而 w' 是常量，它是一个低维空间里的常量 w 经过变换得到的，所以给了一个 w 和 x 的值，就有一个确定的 $f(x')$ 值与其对应。所以，需要这样一种函数 $K(w,x)$ ，他接受低维空间的输入值，却能算出高维空间的内积值 $\langle w', x' \rangle$ 。

也就是当给了一个低维空间的输入 x 以后： $g(x)=K(w,x)+b$

$$f(x')=\langle w', x' \rangle + b$$

这两个函数的计算结果就完全一样，我们也就用不着费力找那个映射关系，直接拿低维的输入往 $g(x)$ 里面代就可以了。



这样的函数确实存在，它被称作核函数（核，kernel），而且不止一个，事实上，只要是满足了Mercer条件的函数，都可以作为核函数。核函数的基本作用就是接受两个低维空间里的向量，能够计算出经过某个变换后在高维空间里的向量内积值。这就是说，尽管给的问题是线性不可分的，但是就硬当它是线性问题来求解，只不过求解过程中，凡是要求内积的时候就用选定的核函数来算。这样求出来的 α 再和选定的核函数一组合，就得到分类器。

常用核函数

(1)线性核函数 (Linear kernel function) $K(x_i, x_j) = (x_i^T x_j)$

(2)多项式核函数 (Polynomial kernel function) $K(x_i, x_j) = (x_i^T x_j)^d$

(3)高斯核函数 (Gaussian Kernel Function) $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$

(4)sigmoid核函数 $K(x_i, x_j) = \tanh(\beta x_i^T x_j + \theta)$

\tanh 双曲正切, $\beta > 0, \theta < 0$



接下来还有两个问题：

1. 既然有很多的核函数，针对具体问题该怎么选择？
2. 如果使用核函数向高维空间映射后，问题仍然是线性不可分的，那怎么办？



对核函数的选择，现在还缺乏指导原则！

各种实验的观察结果（不光是文本分类）的确表明，某些问题用某些核函数效果很好，用另一些就很差，但是一般来讲，径向基核函数（RBF）是不会出太大偏差的一种。

在常用的核函数中，应用最广泛的是具有较好学习能力的RBF核，无论低维、高维、小样本、大样本等情况，RBF核均适应，具有较宽的收敛域，是较为理想的分类依据函数。Keerthi S S等人证明了线性核和多项式核是RBF核的特殊情况。Lin C J等说明了在某些参数情况下，Sigmoid核同RBF核具有相似的性能。

核函数

【定义】： $x, z \in X$, X 属于 R^n 空间, 非线性函数 Φ 实现输入空间 X 到特征空间 F 的映射, 其中 F 属于 R^m , $n \ll m$ 。核函数技术接收2个低维空间的向量, 能够计算出经某变换后高维空间里的向量内积值。

根据核函数技术有：

$K(x, z) = \langle \Phi(x), \Phi(z) \rangle$, 其中 \langle, \rangle 为内积, $K(x, z)$ 为核函数。

例如 $K(x_1, x_2) = (\langle x_1, x_2 \rangle + 1)^2$

加入核函数以后的分类函数为

$$\sum_{i=1}^n \alpha_i y_i K(x_i, x) + b$$

核函数应用广泛的原因：

- 核函数的引入避免了“**维数灾难**”，大大**减小了计算量**。而输入空间的维数 n 对核函数矩阵无影响，因此，核函数方法可以有效处理高维输入。
- 无需知道非线性变换函数 Φ 的形式和参数
- 核函数的形式和参数的变化会隐式地改变从输入空间到特征空间的映射，进而对特征空间的性质产生影响，最终改变各种核函数方法的性能。
- 核函数方法可以和不同的算法相结合，形成多种不同的基于核函数技术的方法，且这**两部分的设计可以单独进行**，并可以为**不同的应用选择不同的核函数**和算法。

总结

线性可分

- 求解使得超平面具有最大内间隔的 \mathbf{w}^T ， \mathbf{b} 参数。
- 将问题转化为对偶问题进行快速求解。
- 改进：加入松弛变量 ξ 和惩罚因子 \mathbf{C} 的**SVM**

松弛变量允许实际分类中一定的不准确性的存在，引入松弛变量后原先的约束条件变为：

$$y_i(w^T x_i + b) \geq 1 - \zeta_i, i = 1, 2, \dots, \zeta_i \geq 0$$

惩罚因子 \mathbf{C} 则是为了避免系统轻易放弃一些重要的数据，减小系统损失。引入 \mathbf{C} 后目标函数变为：

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \zeta_i$$

线性不可分：

- 将数据空间映射到高维空间，使原本线性不可分变为线性可分。
- 引入核函数，简化映射空间中的内积运算。它避开了直接在高维空间中进行计算，而表现形式却等价于高维空间。
- 不同的样本结构与不同的核函数结合，达到很好的分割效果