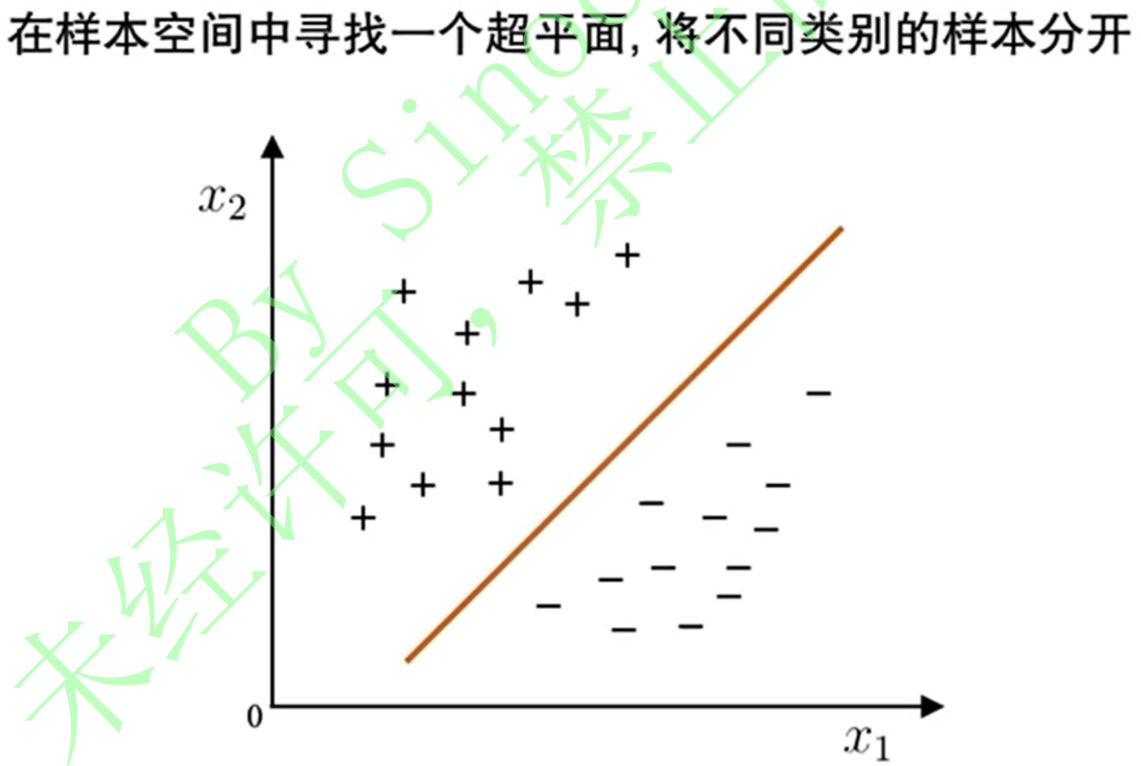
南大周志华《机器学习》课程笔记

> Introduction：最近自学机器学习课程，注意到了南京大学周志华老师的课程。我是在学堂在线平台观看的，注意到b站上也有相应视频，但b站上并未获得授权，随时有消失的可能。
>
> 周志华老师的网络教学视频中，与其西瓜书相比确实少了一些内容。但幸运的是，缺失的内容实际上对于初学者来说并不会产生太大影响。目前这一笔记也遵循视频内容，相比西瓜书中也会有一些缺失，敬请谅解。可能以后如果有机会和时间，我会再阅读周志华老师的书籍将缺失内容补全。
>
> 一切内容敬请关注我的个人Page页面。
>
> 全系列笔记请见：click here
> **About Me:**点击进入我的Personal Page

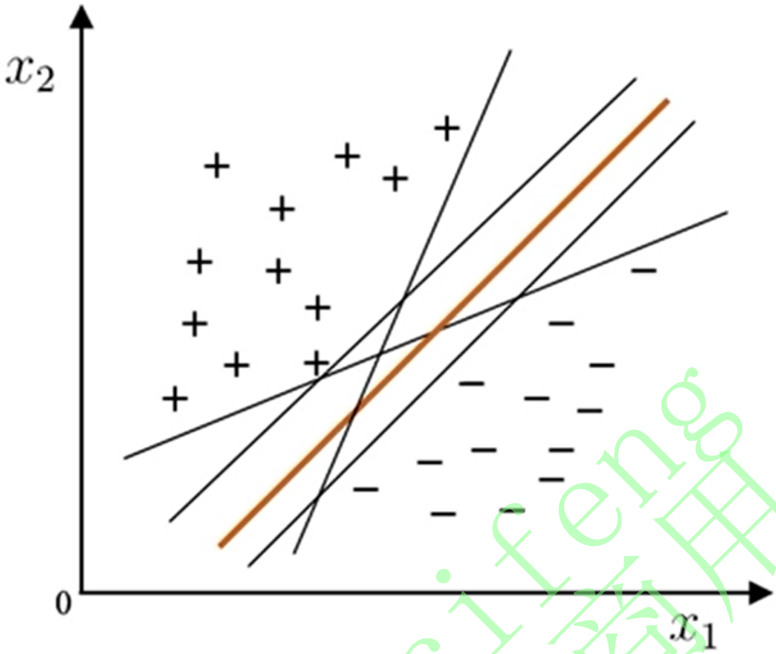
第五章 支持向量机SVM

支持向量机基本型

线性分类器回顾



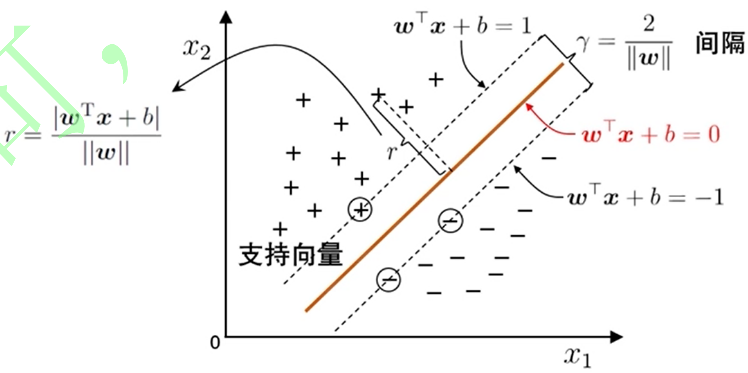
将训练样本分开的超平面可能有很多，哪一个更好呢？



"正中间"的：鲁棒性最好，泛化能力最强

间隔(Margin)
与
支持向量(Support Vector)

超平面方程: $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$



最大间隔：寻找参数$\boldsymbol{w}$和b，使得γ最大

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

$$\Downarrow$$

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

凸二次规划问题，能用优化计算包求解，但可以有更高效的办法

拉格朗日乘子法

对偶问题和解的特性

由上可知，这是一个约束优化问题，因而可以采用拉格朗日乘子法来进行。关于拉格朗日乘子法，推荐CSDN上的一个解释，总体来看写的还是很清晰的：https://blog.csdn.net/qq_28087491/article/details/126134823

拉格朗日乘子法

- 第一步：引入拉格朗日乘子$\alpha_i \geq 0$得到拉格朗日函数

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

- 第二步：令$L(\boldsymbol{w}, b, \boldsymbol{\alpha})$对$\boldsymbol{w}$和$b$的偏导为零可得

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \ , \quad 0 = \sum_{i=1}^{m} \alpha_i y_i$$

这里计算得到的是对偶问题的下界

我们希望得到原问题最好的下界，因此对其求极大

- 第三步：回代可得

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

最终模型：$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b = \boxed{\sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i^\top \boldsymbol{x}} + b$

KKT条件：

（由拉格朗日乘子法引入的KKT条件）

$$\begin{cases} \alpha_i \geq 0\,; \\ 1 - y_i f(\boldsymbol{x}_i) \leq 0\,; \\ \alpha_i\left(1 - y_i f(\boldsymbol{x}_i)\right) = 0\,. \end{cases} \Longrightarrow \text{必有 } \alpha_i = 0 \text{ 或 } y_i f(\boldsymbol{x}_i) = 1$$

以α为系数的点在最终预测的函数中并未出现

表示恰好在间隔的界限之上，否则再过去就大于1了

因此，最终存在于其中使用到的点必然出现在间隔上，而实际上大量的点是不会出现在间隔上的。换言之，支持向量机的解具有**稀疏性**：训练完成后，最终模型仅与支持向量有关。

支持向量机(Support vector Machine, SVM)便因此得名。

求解方法

在第三步中，可以将计算过程交给许多凸优化的包进行求解。但计算机人通常追求求解速度，甚至愿意为此损失一定的精度。

□ 第三步：回代可得

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \boxed{\alpha_i \alpha_j} y_i y_j \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$$

$$\text{s.t.} \quad \boxed{\sum_{i=1}^{m} \alpha_i y_i = 0}\,, \quad \alpha_i \geqslant 0\,, \quad i = 1, 2, \ldots, m$$

注意看第三步的内容，上面的计算式中包含了α_a和α_j，而下式中的约束条件要求$\alpha_1 y_1 + \alpha_2 y_2 + \ldots + \alpha_m y_m = 0$。那么就可以思考。我们只要在这m项中挑选两项，那么其余m-2项都可以固定为常数c，这样简单处理既保证可以满足约束条件，又极大减少了变量的个数，从而方便了我们进行计算。

因此，可以得到下面的求解方法——**SMO**

基本思路：不断执行如下两个步骤直至收敛

- 第一步：选取一对需更新的变量α_i 和 α_j
- 第二步：固定 α_i和 α_j以外的参数，求解对偶问题更新 α_i 和 α_j

仅考虑 α_i 和 α_j 时，对偶问题的约束 $0=\sum_{i=1}^{m}\alpha_i y_i$ 变为

$$\alpha_i y_i+\alpha_j y_j=c\ ,\quad \alpha_i\geqslant 0\ ,\quad \alpha_j\geqslant 0$$

将α_j 用α_i 表示后，该式中只包含α_i 这么一个未知变量

用 α_i 表示 α_j，代入对偶问题

进行循环迭代

$$\max_{\boldsymbol{\alpha}}\ \sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j$$

闭式解指可以用公式表示的准确解

有闭式解

对任意支持向量$(\boldsymbol{x}_s,y_s)$有 $y_s f(\boldsymbol{x}_s)=1$ 由此可解出 b

为提高鲁棒性，通常使用所有支持向量求解的平均值

解得α_i后，可以回代$\alpha_i y_i+\alpha_j y_j=0$这一约束，得到新的$\alpha_j$，从而可以再回代问题公式进行循环迭代优化。

那么，α_i,α_j应该如何选取初始值呢?

当然可以随机取值，这样通过多次迭代后可以得到结果。但如果为了尽快获得较好的结果，应该选择每次更新后可以使得目标函数改善最多的。因为每一次改变越多，也就是走的越快。因此，违反原本KKT条件越多的，用其来进行更新目标函数的提升也就越大。这也就形成了SVM最著名和最经典的解法——SMO方法。

寻找最初的两个点时，先寻找违反KKT条件最多的点，理论上第二个点也应该寻找违反KKT条件最多的点，但由于其计算量较大，通常采用这样一个简化的方法：找到第一个点后，寻找距离第一个点间隔最远的点，从而减少计算量。这一方法必然是收敛的，因为每次寻找违反KKT条件最大的去进行更新，直到最后找不到，那么函数也就到了所想要的收敛的区域。

为提高模型的鲁棒性，通常会对所有支持向量带入求解b，采用所有的解的平均值作为最终的b值。

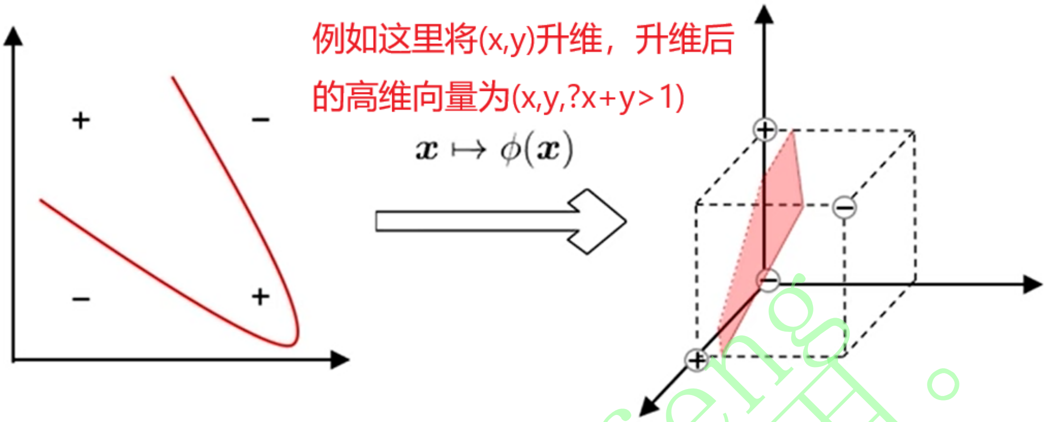
这就是支持向量机的基本型，基本型指的是原本的问题是线性可分的问题，通过寻找线性可分问题中正中间的这一个划分，在该划分过程中寻找一个简单的优化技术，通过采用拉格朗日乘子法，最后转化成了一个闭式解的问题(甚至可以迭代优化)，从而获得更加友好和高效的方法。

然而，现实中很多数据并非线性可分的，这就需要在线性的基本型SVM之上进行一些修改。

特征空间映射

若不存在一个能正确划分两类样本的超平面，怎么办？

将样本从原始空间映射到一个更高维的特征空间，使样本在这个特征空间内线性可分



如果原始空间是有限维(属性数有限)，那么一定存在一个高维特征空间使样本线性可分

这样，我们对这些高维向量的划分就如之前基本型一致了，只是将原来的x这里都变成$\phi(x)$。

设样本 $\boldsymbol{x}$ 映射后的向量为 $\phi(\boldsymbol{x})$，划分超平面为 $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}) + b$

原始问题

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$\text{s.t.} \ \ y_i(\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}_i) + b) \geq 1, \ \ i = 1, 2, \ldots, m.$$

对偶问题

$$\max_{\boldsymbol{\alpha}} \ \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j)$$
$$\text{s.t.} \ \sum_{i=1}^{m} \alpha_i y_i = 0 \,, \quad \alpha_i \geqslant 0 \,, \quad i = 1, 2, \ldots, m$$

预测

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}) + b = \sum_{i=1}^{m} \alpha_i y_i \phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}) + b$$

可以注意到，对x升维后需要去计算更高维的两个向量的内积，这意味着计算开销和等待时间是非常长的。但幸运的是，我们可以观察到这两个高维向量始终以内积的形式出现，因此如果我们不需要直接计算这两个高维向量的内积，而是通过计算某个可以代替该内积的值就可以大大加快计算效率和计算速度，因此，我们甚至不需要知道每个高维向量的值，而只需要知道内积结果即可。

而基于这一思想，就引出**核函数**这一概念。

核函数(kernel function)

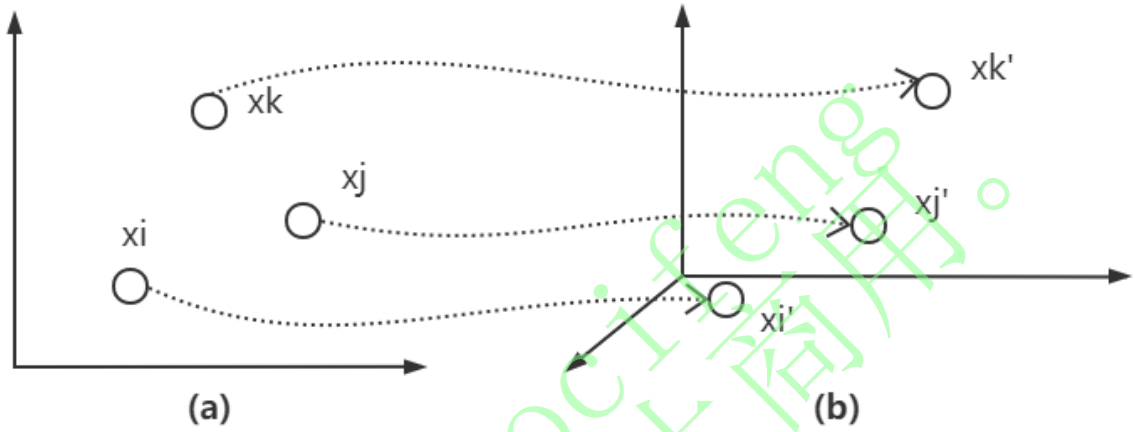
因此，基本思想就是设计一个核函数$\kappa()$，使得$\kappa(x_i,x_j)=\phi(x_i)^T\phi(x_j)$。这样，我们就可以绕开显示考虑特征映射和计算高维内积的困难。

关于Mercer定理

Mercer定理：若一个对称函数所对应的核矩阵半正定，则它就可以作为核函数来使用。

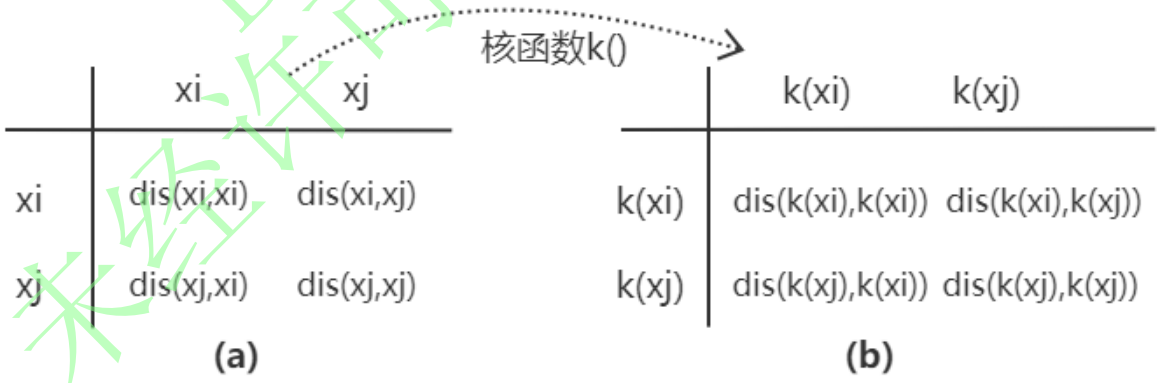
> 根据Mercer定理，我们就知道了如何去寻找我们的目标核函数。下面解释核矩阵等概念。

实际上，核函数的功能是实现空间的变换。



假定上图图(a)中，有点$x_i.x_j,x_k$，右侧图(b)为图a对应的甚高维空间（我们不知道有多高，甚至其可以无穷高维，所以称为甚高维空间），x_i',x_j',x_k'为左侧低维空间中$x_i.x_j$，x_k所对应的点，根据升维映射向量$\phi()$可得到$x_i'=\phi(x_i),x_j'=\phi(x_j),x_k'=\phi(x_k)$。在左侧空间中，可以计算得到点$x_i.x_j$之间的距离。换言之，这样的一些距离就对应了左侧这样一个低微空间。

> 需要注意：将低微转换到高维，高维之中点之间的关系相比低微会发生一定的扭曲。而这种扭曲是由$\phi()$所决定的。



> 注释：为了简单描述，这里仅考虑了两个节点，实际中应该是一个邻接矩阵的形式来衡量任意两个节点之间的距离。

在原空间中我们可以定义一个距离函数如图(a)，可以知道该矩阵主对角线为全0，矩阵沿着主对角线对称，该矩阵对称、半正定。

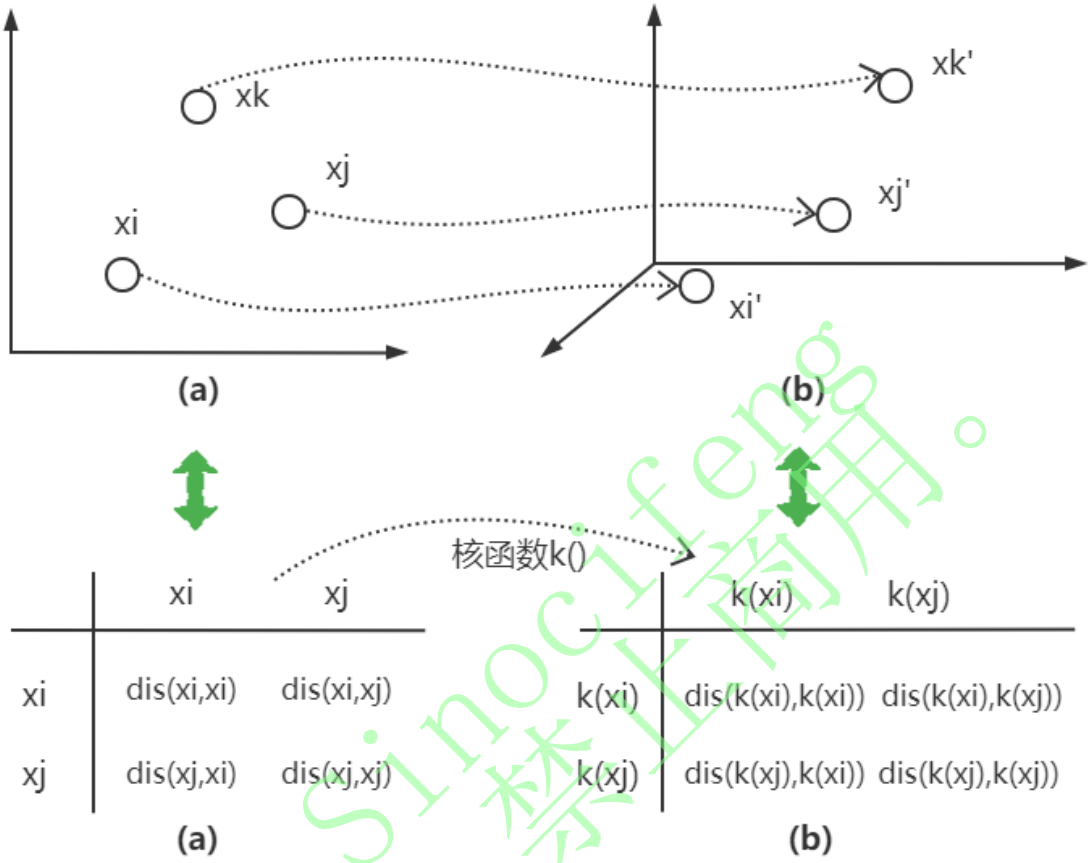
核矩阵即对于输入的(xi,yi)，通过核函数k()处理后可以得到矩阵如图(b)。如果可以观察到右侧这样的矩阵也是对称和半正定的，则该矩阵就是某一个甚高维空间内的距离矩阵。

这就是对Mercer定理进行的解释

后续介绍

任意一个核函数，都隐式地定义了一个RKHS(Reproducing Kernel Hilbert Space，再生核希尔伯特空间)。

RKHS其实值得就是核矩阵所对应的隐式甚高维空间，即如下图所示：



我们将上面的甚高维空间可以称为由核矩阵张成的RKHS(再生核希尔伯特空间)。

这样，我们就将两个向量之间的点积问题转换成了求核函数问题，我们只需要用核函数$k()$来代替升维映射向量$\phi()$即可。

思考

那么核函数$k()$与升维映射向量$\phi()$所对应的甚高维空间是否恰好一样呢?

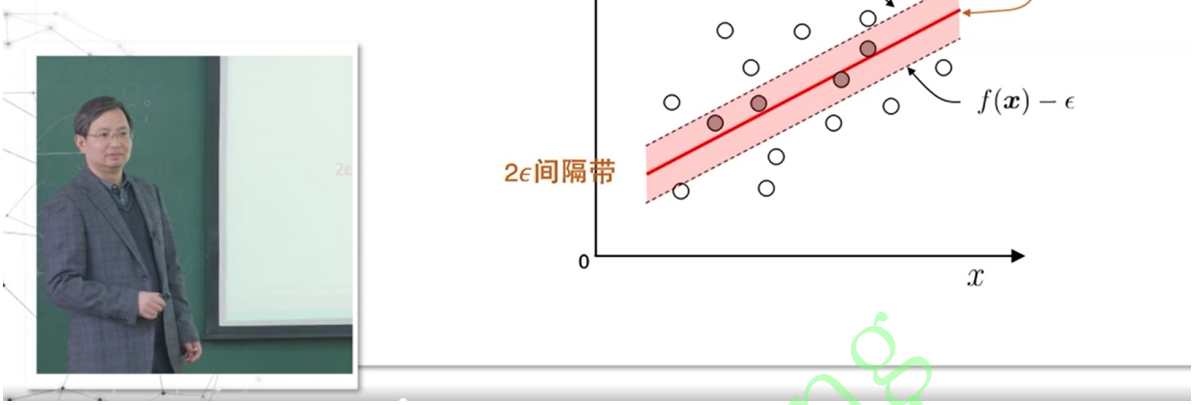
答案是无法确定的，我们无法保证两个空间一致。假设我们记录所有可能的核函数为集合A，那么$\phi()$必然包含在该集合A中。如果想要找到最合适的核函数，即$\phi()$，那么这一问题就转变为从很多可能的核函数中寻找一个最合适的核函数，而这就可以考虑使用第二章中的模型选择技术。然而，我们无法保证一定可以找到最理想的$\phi()$，因为如果可以找到$\phi()$就等价于在多项式时间内找到了一个非常困难问题的确定性最优解，即$P=NP$。

总结：**“核函数的选择”是决定支持向量机性能的关键**

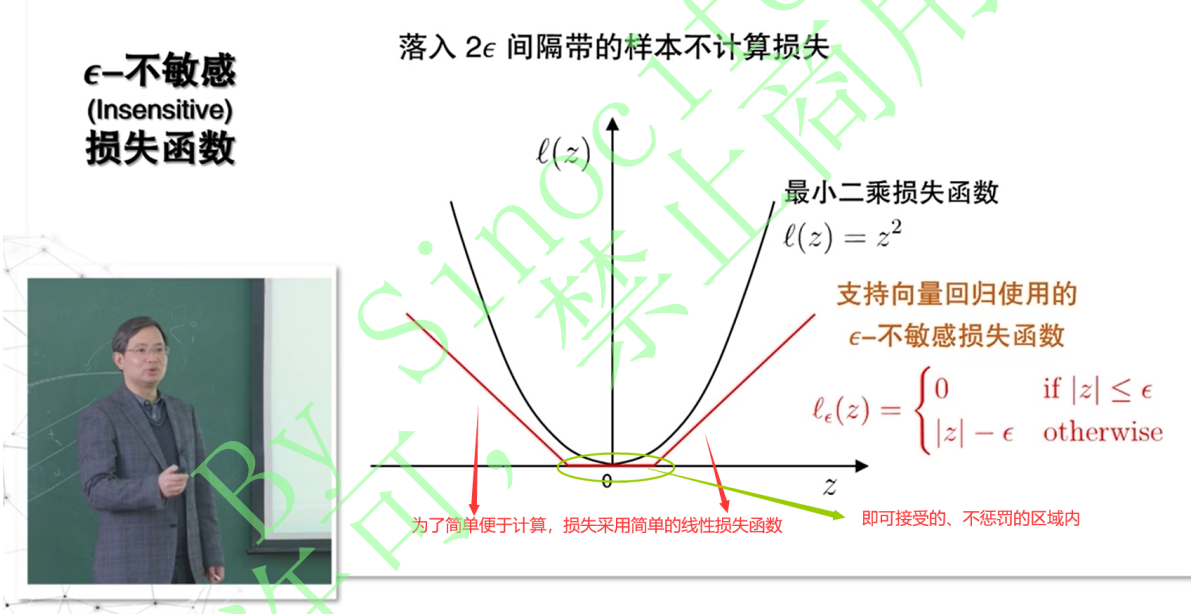
> 机器学习中所有技术都必然在某处无法找到最优，支持向量机最好的是在于其不好的地方(核函数选择)已经清楚地告知了。

如何使用支持向量机

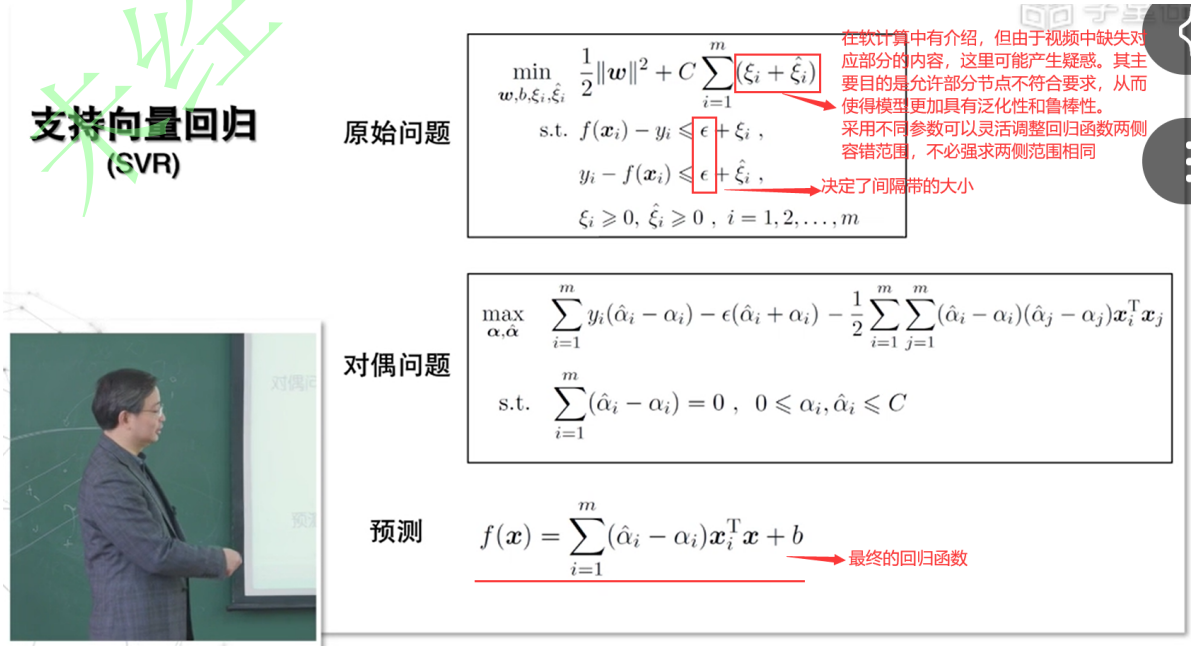
一个SVM进行回归的案例



这样，我们可以得到损失函数（即惩罚）



从而，基于之前的知识，可以对问题进行求解获得最终回归结果：



现实使用中的SVM

如何使用SVM?

- 入门级—— 实现并使用各种版本SVM
- 专业级—— 尝试、组合核函数
- 专家级—— 根据问题而设计目标函数、替代损失、进而……