支持向量机简介

报告人:王昕毅

支持向量机

- 线性可分支持向量机-硬间隔最大化
- 线性支持向量机-软间隔最大化
- 非线性可分支持向量机-核技巧及软间隔最大化
- 支持向量回归

线性可分支持向量机：

6.1 间隔与支持向量

给定训练样本集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$, $y_i \in \{-1, +1\}$, 分类学习最基本的想法就是基于训练集 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. 但能将训练样本分开的划分超平面可能有很多, 如图 6.1 所示, 我们应该努力去找到哪一个呢?

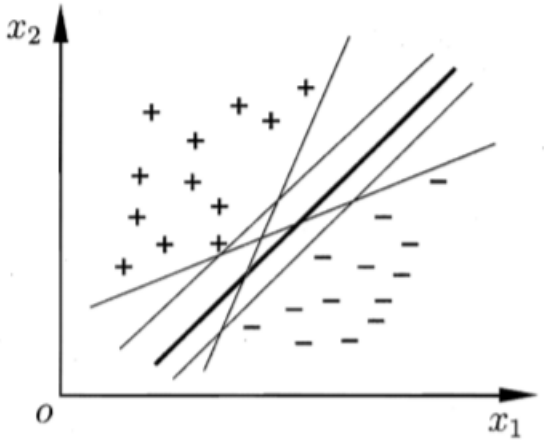


图 6.1 存在多个划分超平面将两类训练样本分开

直观上看，应该去找位于两类训练样本“正中间”的划分超平面，即图 6.1 中深色的那个，因为该划分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 6.1 中的训练 样本更接近两个类的分隔界，这将使许多划分超平面出现错误，而红色的超平 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鲁棒的，对未见 示例的泛化能力最强.

在样本空间中, 划分超平面可通过如下线性方程来描述:

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=0\ , \tag{6.1}$$

其中 $\boldsymbol{w}=(w_1;w_2;\ldots;w_d)$ 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, 划分超平面可被法向量 $\boldsymbol{w}$ 和位移 b 确定, 下面我们将其记为 $(\boldsymbol{w},b)$. 样本空间中任意点 $\boldsymbol{x}$ 到超平面 $(\boldsymbol{w},b)$ 的距离可写为

$$r=\frac{|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b|}{||\boldsymbol{w}||}\ . \tag{6.2}$$

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

$$\begin{cases}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\geqslant+1, & y_i=+1\ ;\\ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\leqslant-1, & y_i=-1\ .\end{cases} \tag{6.3}$$

如图 6.2 所示, 距离超平面最近的这几个训练样本点使式(6.3)的等号成立, 它们被称为“支持向量”(support vector), 两个异类支持向量到超平面的距离之和为

$$\gamma=\frac{2}{||\boldsymbol{w}||}\ , \tag{6.4}$$

它被称为“间隔”(margin).

$$\max_{w,b}\quad \gamma$$
$$\text{s.t.}\quad y_i\left(\frac{w}{\|w\|}\cdot x_i+\frac{b}{\|w\|}\right)\geqslant\gamma,\quad i=1,2,\cdots,N$$

$$\max_{w,b}\quad \frac{\hat{\gamma}}{\|w\|}$$
$$\text{s.t.}\quad y_i(w\cdot x_i+b)\geqslant\hat{\gamma},\quad i=1,2,\cdots,N$$

$$\min_{w,b}\quad \frac{1}{2}\|w\|^2$$
$$\text{s.t.}\quad y_i(w\cdot x_i+b)-1\geqslant 0,\quad i=1,2,\cdots,N$$

欲找到具有“最大间隔”(maximum margin)的划分超平面, 也就是要找到能满足式(6.3)中约束的参数 $\boldsymbol{w}$ 和 b, 使得 γ 最大, 即

$$\begin{aligned} \max_{\boldsymbol{w},b} \quad & \frac{2}{\|\boldsymbol{w}\|} \\ \text{s.t.} \quad & y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geqslant 1, \quad i = 1, 2, \ldots, m. \end{aligned} \tag{6.5}$$

显然, 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(6.5)可重写为

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{s.t.} \quad & y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geqslant 1, \quad i = 1, 2, \ldots, m. \end{aligned} \tag{6.6}$$

这就是支持向量机(Support Vector Machine, 简称 SVM)的基本型.

拉格朗日对偶性:

假设 $f(x)$，$c_i(x)$，$h_j(x)$ 是定义在 $\mathbf{R}^n$ 上的连续可微函数．考虑约束最优化问题

$$\min_{x \in \mathbf{R}^n} f(x) \tag{C.1}$$

$$\text{s.t.} \quad c_i(x) \leqslant 0, \quad i = 1,2,\cdots,k \tag{C.2}$$

$$h_j(x) = 0, \quad j = 1,2,\cdots,l \tag{C.3}$$

称此约束最优化问题为原始最优化问题或原始问题．

首先，引进广义拉格朗日函数（generalized Lagrange function）

$$L(x,\alpha,\beta) = f(x) + \sum_{i=1}^{k} \alpha_i c_i(x) + \sum_{j=1}^{l} \beta_j h_j(x) \tag{C.4}$$

原始问题：$\min_x \theta_P(x) = \min_x \max_{\alpha,\beta:\alpha_i \geqslant 0} L(x,\alpha,\beta)$

对偶问题：$\max_{\alpha,\beta:\alpha_i \geqslant 0} \min_x L(x,\alpha,\beta)$

原始问题与约束最优化问题等价，对偶问题与原始问题在满足KKT条件的情况下等价，对偶问题一般比原始问题更容易求解，故一般求解对偶问题

KKT条件：

$$\nabla_x L(x^*, \alpha^*, \beta^*) = 0$$

$$\alpha_i^* c_i(x^*) = 0, \quad i = 1, 2, \cdots, k$$

$$c_i(x^*) \leqslant 0, \quad i = 1, 2, \cdots, k$$

$$\alpha_i^* \geqslant 0, \quad i = 1, 2, \cdots, k$$

$$h_j(x^*) = 0 \quad j = 1, 2, \cdots, l$$

满足KKT条件时原始问题的最优解与对偶问题的最优解等价，且X可以用α和β表示求出

我们希望求解式(6.6)来得到大间隔划分超平面所对应的模型

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b \,, \tag{6.7}$$

其中 $\boldsymbol{w}$ 和 b 是模型参数. 注意到式(6.6)本身是一个凸二次规划(convex quadratic programming) 问题, 能直接用现成的优化计算包求解, 但我们可以有更高效的办法.

对式(6.6)使用拉格朗日乘子法可得到其“对偶问题”(dual problem). 具体来说, 对式(6.6) 的每条约束添加拉格朗日乘子 $\alpha_i \geqslant 0$, 则该问题的拉格朗日函数可写为

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \alpha_i \left(1 - y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b)\right) \,, \tag{6.8}$$

其中 $\boldsymbol{\alpha} = (\alpha_1; \alpha_2; \ldots; \alpha_m)$. 令 $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ 对 $\boldsymbol{w}$ 和 b 的偏导为零可得

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \,, \tag{6.9}$$

$$0 = \sum_{i=1}^{m} \alpha_i y_i \,. \tag{6.10}$$

将式(6.9)代入(6.8), 即可将 $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ 中的 $\boldsymbol{w}$ 和 b 消去, 再考虑式(6.10)的约束, 就得到式(6.6)的对偶问题

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j \tag{6.11}$$

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

$$\alpha_i \geqslant 0 \,, \quad i = 1, 2, \ldots, m \,.$$

解出 $\boldsymbol{\alpha}$ 后, 求出 $\boldsymbol{w}$ 与 b 即可得到模型

$$\begin{aligned} f(\boldsymbol{x}) &= \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} + b \\ &= \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x} + b \,. \end{aligned} \tag{6.12}$$

从对偶问题(6.11)解出的 α_i 是式(6.8)中的拉格朗日乘子，它恰对应着训练样本 $(\boldsymbol{x}_i, y_i)$. 注意到式(6.6)中有不等式约束，因此上述过程需满足 KKT (Karush-Kuhn-Tucker) 条件，即要求

$$\begin{cases} \alpha_i \geqslant 0 ; \\ y_i f(\boldsymbol{x}_i) - 1 \geqslant 0 ; \\ \alpha_i \left(y_i f(\boldsymbol{x}_i) - 1\right) = 0 \,. \end{cases} \tag{6.13}$$

于是，对任意训练样本 $(\boldsymbol{x}_i, y_i)$，总有 $\alpha_i = 0$ 或 $y_i f(\boldsymbol{x}_i) = 1$. 若 $\alpha_i = 0$，则该样本将不会在式(6.12) 的求和中出现，也就不会对 $f(\boldsymbol{x})$ 有任何影响；若 $\alpha_i > 0$，则必有 $y_i f(\boldsymbol{x}_i) = 1$，所对应的样本点位于最大间隔边界上，是一个支持向量. 这显示出支持向量机的一个重要性质：训练完成后，大部分的训练样本都不需保留，最终模型仅与支持向量有关.

不难发现式(6.11)是一个二次规划问题，可使用通 用的二次规划算法来求解;然而该问题的规模正比于训练样本数这会在实际 任务中造成很大的开销.为了避开这个障碍，人们通过利用问题本身的特性，提 出了很多高效算法， SMO (Sequential Minimal Optimization) 是其中一个著名 的代表 [Platt， 1998].

SMO 算法：

序列最小最优化算法(SMO)可以高效的求解上述SVM问题，它把原始求解N个参数二次规划问题分解成很多个子二次规划问题分别求解。

SMO 的基本思路是先固定 α_i 之外的所有参数, 然后求 α_i 上的极值. 由于存在约束 $\sum_{i=1}^{m} \alpha_i y_i = 0$, 若固定 α_i 之外的其他变量, 则 α_i 可由其他变量导出. 于是, SMO 每次选择两个变量 α_i 和 α_j, 并固定其他参数. 这样, 在参数初始化后, SMO 不断执行如下两个步骤直至收敛:

- 选取一对需更新的变量 α_i 和 α_j;
- 固定 α_i 和 α_j 以外的参数, 求解式(6.11)获得更新后的 α_i 和 α_j.

SMO干了什么?

首先，整个对偶问题的二次规划表达如下：

$$\max_{\vec{\alpha}} \quad \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$s.t. \quad \sum_{i=1}^{n}\alpha_i y_i = 0$$

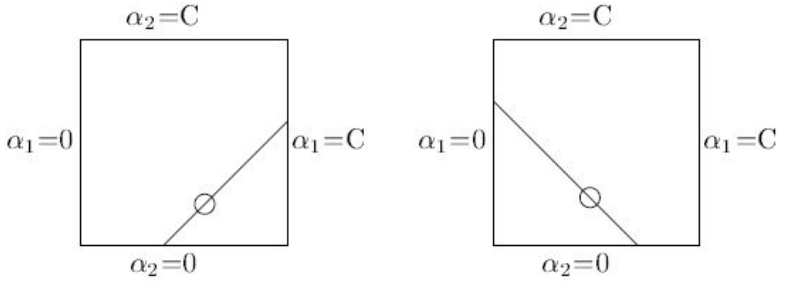
$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, n$$

SMO在整个二次规划的过程中也没干别的，总共干了两件事：

- 选取一对参数 (α_i, α_j)
- 固定 $\vec{\alpha}$ 向量的其他参数，将 (α_i, α_j) 代入上述表达式进行求最优解获得更新后的 (α_i, α_j)

SMO不断执行这两个步骤直至收敛。

因为有约束 $\sum_{i=1}^{n}\alpha_i y_i = 0$ 存在，实际上 α_i 和 α_j 的关系也可以确定。 $\alpha_i y_i + \alpha_j y_j = C$ 这两个参数的和或者差是一个常数。



$$y_1 \neq y_2 \Rightarrow \alpha_1 - \alpha_2 = \gamma$$

$$y_1 = y_2 \Rightarrow \alpha_1 + \alpha_2 = \gamma$$

所以虽然宣传上说是选择了一对 (α_i, α_j)，但还是选择了其中一个，将另一个写作关于它的表达式代入目标函数求解。

为什么SMO跑的那么快，比提出之前的算法不知道高到哪里去了?

正如上面提到的，在固定其他参数以后，这就是一个**单变量**二次规划问题，仅有的约束也是这个变量 $\alpha_i \geq 0$，显然有闭式解。不必再调用数值优化算法。

KKT条件是对偶问题最优解的**必要条件**:

$$\begin{cases} \alpha_i \geq 0 \\ y_i f(\mathbf{x}_i) - 1 \geq 0 \\ \alpha_i \left(y_i f(\mathbf{x}_i) - 1\right) = 0 \end{cases}$$

除了第一个非负约束以外，其他约束都是根据目标函数推导得到的最优解必须满足的条件，如果违背了这些条件，那得到的解必然不是最优的，目标函数的值会减小。

所以在SMO迭代的两个步骤中，只要 (α_i, α_j) 中有一个违背了KKT条件，这一轮迭代完成后，目标函数的值必然会增大。Generally speaking，KKT条件违背的程度越大，迭代后的优化效果越明显，增幅越大。

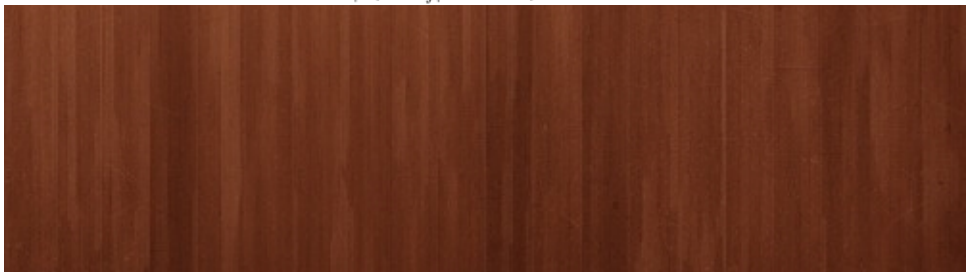
怎样跑的更快?

和梯度下降类似，我们要找到使之优化程度最大的方向（变量）进行优化。所以SMO先选取违背KKT条件程度最大的变量，那么第二个变量应该选择使目标函数值增大最快的变量，但是这个变量怎么找呢？比较各变量优化后对应的目标函数值的变化幅度？这个样子是不行的，复杂度太高了。

SMO使用了一个启发式的方法，当确定了第一个变量后，选择使两个变量对应样本之间最大的变量作为第二个变量。直观来说，更新两个差别很大的变量，比起相似的变量，会带给目标函数更大的变化。间隔的定义也可以借用偏差函数

$$E_i = \max\left(y_i f(\mathbf{x}_i) - 1, 0\right)$$

我们要找的也就是使对于 α_i 来说使 $|E_i - E_j|$ 最大的 α_j



求出α之后可根据公式 $\boldsymbol{w}=\sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i$ 求出w

如何确定偏移项 b 呢? 注意到对任意支持向量 $(\boldsymbol{x}_s, y_s)$ 都有 $y_s f(\boldsymbol{x}_s)=1$, 即

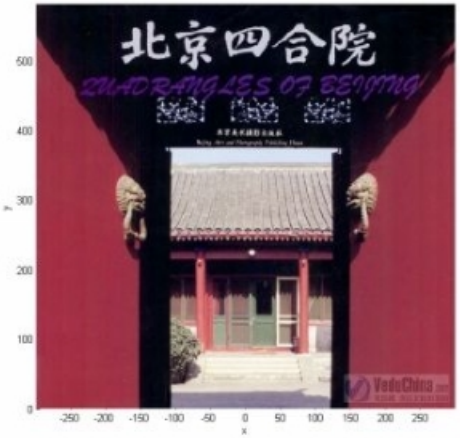
$$y_s\left(\sum_{i\in S}\alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_s+b\right)=1\ , \tag{6.17}$$

其中 $S=\{i \mid \alpha_i>0,\ i=1,2,\ldots,m\}$ 为所有支持向量的下标集. 理论上, 可选取任意支持向量并通过求解式(6.17)获得 b, 但现实任务中常采用一种更鲁棒的做法: 使用所有支持向量求解的平均值

$$b=\frac{1}{|S|}\sum_{s\in S}\left(y_s-\sum_{i\in S}\alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_s\right)\ . \tag{6.18}$$

核函数：前面我们介绍了线性情况下的支持向量机，它通过寻找一个线性的超平面来达到对数据进行分类的目的。不过，由于是线性方法，所以对非线性的数据就没有办法处理了。不论是任何高级的分类器，只要它是线性的，就没法处理，SVM 也不行。因为这样的数据本身就是线性不可分的。一个简单的例子：

下面这张图位于第一、二象限内。我们关注红色的门，以及“北京四合院”这几个字下面的紫色的字母。我们把红色的门上的点看成是“+”数据，紫色字母上的点看成是“-”数据，它们的横、纵坐标是两个特征。显然，在这个二维空间内，“+” “-”两类数据不是线性可分的。

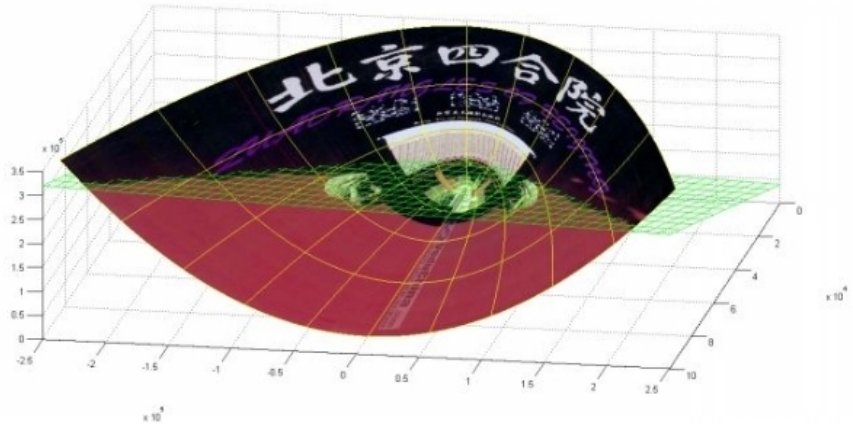


我们现在考虑核函数 $K(v_1, v_2) = < v_1, v_2 >^2$，即“内积平方”。
这里面 $v_1 = (x_1, y_1), v_2 = (x_2, y_2)$ 是二维空间中的两个点。

这个核函数对应着一个二维空间到三维空间的映射，它的表达式是：
$P(x, y) = (x^2, \sqrt{2}xy, y^2)$
可以验证，

$$\begin{aligned} < P(v_1), P(v_2) > &= < (x_1^2, \sqrt{2}x_1y_1, y_1^2), (x_2^2, \sqrt{2}x_2y_2, y_2^2) > \\ &= x_1^2x_2^2 + 2x_1x_2y_1y_2 + y_1^2y_2^2 \\ &= (x_1x_2 + y_1y_2)^2 \\ &= < v_1, v_2 >^2 \\ &= K(v_1, v_2) \end{aligned}$$

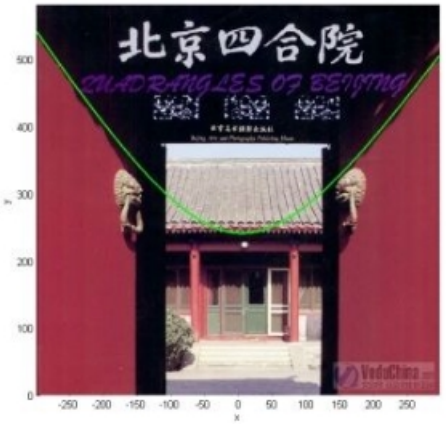
在P这个映射下，原来二维空间中的图在三维空间中的像是这个样子：



（前后轴为x轴，左右轴为y轴，上下轴为z轴）

注意到绿色的平面可以完美地分割红色和紫色，也就是说，两类数据在三维空间中变成线性可分的了。

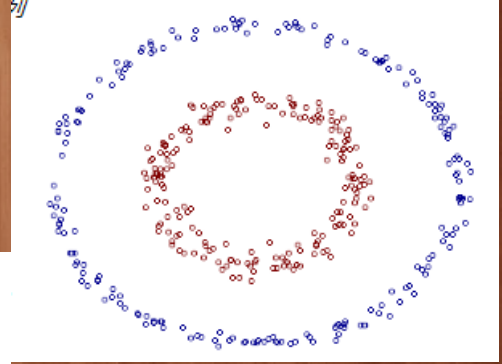
而三维中的这个判决边界，再映射回二维空间中是这样的：



这是一条双曲线，它不是线性的。

如上面的例子所说，核函数的作用就是隐含着一个从低维空间到高维空间的映射，而这个映射可以把低维空间中线性不可分的两类点变成线性可分的。
当然，这个具体例子强烈地依赖于数据在原始空间中的位置。
事实中使用的核函数往往比这个例子复杂得多。它们对应的映射并不一定能够显式地表达出来；它们映射到的高维空间的维数也比我举的例子（三维）高得多，甚至是无穷维的。这样，就可以期待原来并不线性可分的两类点变成线性可分的了。
链接：
https://www.zhihu.com/question/24627666/answer/28440943

再看一个例子，对于右图的数据集，一个理想的分界应该是一个“圆圈”而不是一条线（超平面）。如果用 X1和X2来表示这个二维平面的两个坐标的话，我们知道一条二次曲线（圆圈是二次曲线的一种特殊情况）的方程可以写作这样的形式：



$$a_1X_1 + a_2X_1^2 + a_3X_2 + a_4X_2^2 + a_5X_1X_2 + a_6 = 0$$

注意上面的形式，如果我们构造另外一个五维的空间，其中五个坐标的值分别为 $Z_1 = X_1$，$Z_2 = X_1^2$, $Z_3 = X_2$, $Z_4 = X_2^2$, $Z_5 = X_1X_2$，那么显然，上面的方程在新的坐标系下可以写作：

$$\sum_{i=1}^{5} a_iZ_i + a_6 = 0$$

关于新的坐标 Z，这正是一个 hyper plane 的方程！也就是说，如果我们做一个映射 $\phi: \mathbb{R}^2 \to \mathbb{R}^5$，将 X 按照上面的规则映射为 Z，那么在新的空间中原来的数据将变成线性可分的，从而使用之前我们推导的线性分类算法就可以进行处理了。这正是 Kernel 方法处理非线性问题的基本思想。

现在让我们再回到 SVM 的情形，假设原始的数据时非线性的，我们通过一个映射将其映射到一个高维空间中，数据变得线性可分了，这个时候，我们就可以使用原来的推导来进行计算，只是所有的推导现在是在新的空间，而不是原始空间中进行。

这样一来问题就解决了吗：拿到非线性数据，就找一个映射，然后一股脑把原来的数据映射到新空间中，再做线性 SVM 即可。然而我们可以发现在上面的例子里，我们对一个二维空间做映射，选择的新空间是原始空间的所有一阶和二阶的组合，得到了五个维度；如果原始空间是三维，那么我们会得到 19 维的新空间这个数目是呈爆炸性增长的，这给 $\phi(\cdot)$ 的计算带来了非常大的困难，而且如果遇到无穷维的情况，就根本无从计算了。所以就需要 Kernel 出马了。

不妨还是从最开始的简单例子出发，设两个向量 $x_1 = (\eta_1, \eta_2)^T$ 和 $x_2 = (\xi_1, \xi_2)^T$，而 $\phi(\cdot)$ 即是到前面说的五维空间的映射，因此映射过后的内积为：

$$\langle \phi(x_1), \phi(x_2) \rangle = \eta_1\xi_1 + \eta_1^2\xi_1^2 + \eta_2\xi_2 + \eta_2^2\xi_2^2 + \eta_1\eta_2\xi_1\xi_2$$

另外，我们又注意到：

$$(\langle x_1, x_2 \rangle + 1)^2 = 2\eta_1\xi_1 + \eta_1^2\xi_1^2 + 2\eta_2\xi_2 + \eta_2^2\xi_2^2 + 2\eta_1\eta_2\xi_1\xi_2 + 1$$

二者有很多相似的地方，实际上，我们只要把某几个维度线性缩放一下，然后再加上一个常数维度，具体来说，上面这个式子的计算结果实际上和映射

$$\varphi(X_1, X_2) = (\sqrt{2}X_1, X_1^2, \sqrt{2}X_2, X_2^2, \sqrt{2}X_1X_2, 1)^T$$

之后的内积 $\langle \varphi(x_1), \varphi(x_2) \rangle$ 的结果是相等的（自己验算一下）。区别在于什么地方呢？一个是映射到高维空间中，然后再根据内积的公式进行计算；而另一个则直接在原来的低维空间中进行计算，而不需要显式地写出映射后的结果。回忆刚才提到的映射的维度爆炸，在前一种方法已经无法计算的情况下，后一种方法却依旧能从容处理，甚至是无穷维度的情况也没有问题。

我们把这里的计算两个向量在映射过后的空间中的内积的函数叫做核函数 (Kernel Function)，例如，在刚才的例子中，我们的核函数为：

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

核函数能简化映射空间中的内积运算——刚好"碰巧"的是，在我们的 SVM 里需要计算的地方数据向量总是以内积的形式出现的。对比刚才我们写出来的式子，现在我们的分类函数为：

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

其中 α 由如下 dual 问题计算而得：

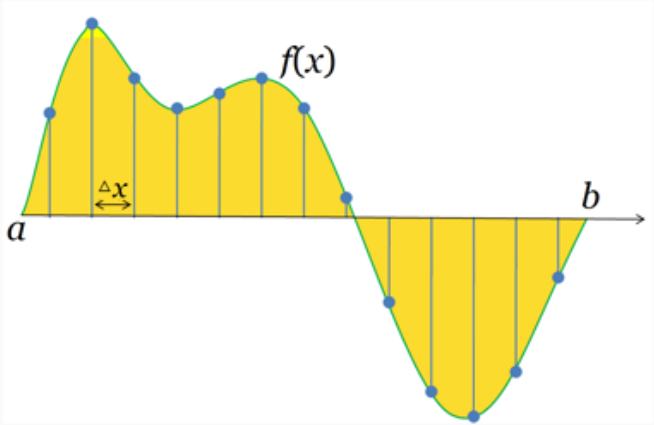
$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j)$$
$$s.t., \alpha_i \geq 0, i = 1, \ldots, n$$
$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

这样一来计算的问题就算解决了，避开了直接在高维空间中进行计算，而结果却是等价的

We know that everything in the world can be decomposed into the combination of the basic elements. For example, water is the combination of hydrogen and oxygen. Similarly, in mathematics, basis is used to represent various things in a simple and unified way.

2. Function Basis

A function is an infinite vector. As the following figure shows



source: http://en.wikipedia.org/wiki/Sampling_(signal_processing)

For a function defined on the interval $[a, b]$, we take samples by an interval Δx. If we sample the function $f(x)$ at points $a, x_1, \cdots, x_n, b$, then we can transform the function into a vector $(f(a), f(x_1), \cdots, f(x_n), f(b))^T$. When $\Delta x \to 0$, the vector should be more and more close to the function and at last, it becomes infinite.

Since functions are so close to vectors, we can also define the inner product of functions similarly. For two functions f and g sampling by interval Δx, the inner product may be defined as

$$< f, g >= \lim_{\Delta x \to 0} \sum_i f(x_i) g(x_i) \Delta x = \int f(x) g(x) dx$$

2. Eigen Decomposition

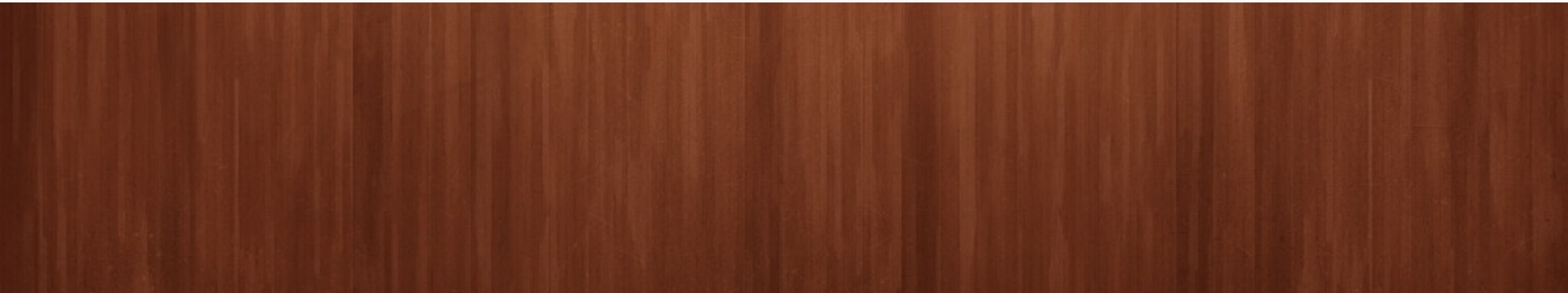
For a real symmetric matrix $\mathbf{A}$, there exists real number λ and vector $\mathbf{x}$ so that

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$$

Then λ is an eigenvalue of $\mathbf{A}$ and $\mathbf{x}$ is the corresponding eigenvector. If $\mathbf{A}$ has two different eigenvalues λ_1 and λ_2 , $\lambda_1 \neq \lambda_2$, with corresponding eigenvectors $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively,

$$\lambda_1 \mathbf{x}_1^T \mathbf{x}_2 = \mathbf{x}_1^T \mathbf{A}^T \mathbf{x}_2 = \mathbf{x}_1^T \mathbf{A} \mathbf{x}_2 = \lambda_2 \mathbf{x}_1^T \mathbf{x}_2$$

Since $\lambda_1 \neq \lambda_2$, we have $\mathbf{x}_1^T \mathbf{x}_2 = 0$, i.e., $\mathbf{x}_1$ and $\mathbf{x}_2$ are orthogonal.



For $\mathbf{A} \in \mathcal{R}^{n \times n}$, we can find n eigenvalues a long with n orthogonal eigenvectors. As a result, $\mathbf{A}$ can be decomposited as

$$\mathbf{A} = \mathbf{Q}\mathbf{D}\mathbf{Q}^T$$

where $\mathbf{Q}$ is an orthogonal matrix (i.e., $\mathbf{Q}\mathbf{Q}^T = \mathbf{I}$) and $\mathbf{D} = \mathbf{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$. If we write $\mathbf{Q}$ column by column

$$\mathbf{Q} = (\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_n)$$

then

$$\begin{aligned}
\mathbf{A} = \mathbf{Q}\mathbf{D}\mathbf{Q}^T &= (\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_n) \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} \begin{pmatrix} \mathbf{q}_1^T \\ \mathbf{q}_2^T \\ \vdots \\ \mathbf{q}_n^T \end{pmatrix} \\
&= (\lambda_1 \mathbf{q}_1, \lambda_2 \mathbf{q}_2, \cdots, \lambda_n \mathbf{q}_n) \begin{pmatrix} \mathbf{q}_1^T \\ \mathbf{q}_2^T \\ \vdots \\ \mathbf{q}_n^T \end{pmatrix} \\
&= \sum_{i=1}^{n} \lambda_i \mathbf{q}_i \mathbf{q}_i^T
\end{aligned}$$

Here $\{\mathbf{q}_i\}_{i=1}^n$ is a set of orthogonal basis of $\mathcal{R}^n$.

3. Kernel Function

A function $f(\mathbf{x})$ can be viewed as an infinite vector, then for a function with two independent variables $K(\mathbf{x},\mathbf{y})$, we can view it as an infinite matrix. Among them, if $K(\mathbf{x},\mathbf{y}) = K(\mathbf{y},\mathbf{x})$ and

$$\int\int f(\mathbf{x})K(\mathbf{x},\mathbf{y})f(\mathbf{y})d\mathbf{x}d\mathbf{y} \geq 0$$

for any function f, then $K(\mathbf{x},\mathbf{y})$ is symmetric and positive definite, in which case $K(\mathbf{x},\mathbf{y})$ is a kernel function.

Similar to matrix eigenvalue and eigenvector, there exists eigenvalue λ and eigenfunction $\psi(\mathbf{x})$ so that

$$\int K(\mathbf{x},\mathbf{y})\psi(\mathbf{x})d\mathbf{x} = \lambda\psi(\mathbf{y})$$

For different eigenvalues λ_1 and λ_2 with corresponding eigenfunctions $\psi_1(\mathbf{x})$ and $\psi_2(\mathbf{x})$, it is easy to show that

$$\begin{aligned}\int \lambda_1\psi_1(\mathbf{x})\psi_2(\mathbf{x})d\mathbf{x} &= \int\int K(\mathbf{y},\mathbf{x})\psi_1(\mathbf{y})d\mathbf{y}\psi_2(\mathbf{x})d\mathbf{x} \\ &= \int\int K(\mathbf{x},\mathbf{y})\psi_2(\mathbf{x})d\mathbf{x}\psi_1(\mathbf{y})d\mathbf{y} \\ &= \int \lambda_2\psi_2(\mathbf{y})\psi_1(\mathbf{y})d\mathbf{y} \\ &= \int \lambda_2\psi_2(\mathbf{x})\psi_1(\mathbf{x})d\mathbf{x}\end{aligned}$$

Therefore,

$$< \psi_1, \psi_2 >= \int \psi_1(\mathbf{x})\psi_2(\mathbf{x})d\mathbf{x} = 0$$

Again, the eigenfunctions are orthogonal. Here ψ denotes the function (the infinite vector) itself.

For a kernel function, infinite eigenvalues $\{\lambda_i\}_{i=1}^{\infty}$ along with infinite eigenfunctions $\{\psi_i\}_{i=1}^{\infty}$ may be found. Similar to matrix case,

$$K(\mathbf{x},\mathbf{y}) = \sum_{i=0}^{\infty} \lambda_i\psi_i(\mathbf{x})\psi_i(\mathbf{y})$$

which is the Mercer's theorem. Here $< \psi_i, \psi_j >= 0$ for $i \neq j$. Therefore, $\{\psi_i\}_{i=1}^{\infty}$ construct a set of orthogonal basis for a function space.

4. Reproducing Kernel Hilbert Space

Treat $\{\sqrt{\lambda_i}\psi_i\}_{i=1}^{\infty}$ as a set of orthogonal basis and construct a Hilbert space $\mathcal{H}$. Any function or vector in the space can be represented as the linear combination of the basis. Suppose

$$f = \sum_{i=1}^{\infty} f_i \sqrt{\lambda_i}\psi_i$$

we can denote f as an infinite vector in $\mathcal{H}$:

$$f = (f_1, f_2, \dots)_{\mathcal{H}}^T$$

For another function $g = (g_1, g_2, \dots)_{\mathcal{H}}^T$, we have

$$< f, g >_{\mathcal{H}} = \sum_{i=1}^{\infty} f_i g_i$$

For the kernel function K, here I use $K(\mathbf{x}, \mathbf{y})$ to denote the evaluation of K at point $\mathbf{x}, \mathbf{y}$ which is a scalar, use $K(\cdot, \cdot)$ to denote the function (the infinite matrix) itself, and use $K(\mathbf{x}, \cdot)$ to denote the $\mathbf{x}$th "row" of the matrix, i.e., we fix one parameter of the kernel function to be $\mathbf{x}$ then we can regard it as a function with one parameter or as an infinite vector. Then

$$K(\mathbf{x}, \cdot) = \sum_{i=0}^{\infty} \lambda_i \psi_i(\mathbf{x})\psi_i$$

In space $\mathcal{H}$, we can denote

$$K(\mathbf{x}, \cdot) = (\sqrt{\lambda_1}\psi_1(\mathbf{x}), \sqrt{\lambda_2}\psi_2(\mathbf{x}), \cdots)_{\mathcal{H}}^T$$

Therefore

$$< K(\mathbf{x}, \cdot), K(\mathbf{y}, \cdot) >_{\mathcal{H}} = \sum_{i=0}^{\infty} \lambda_i \psi_i(\mathbf{x})\psi_i(\mathbf{y}) = K(\mathbf{x}, \mathbf{y})$$

This is the *reproducing* property, thus $\mathcal{H}$ is called reproducing kernel Hilbert space (RKHS).

Now it is time to return to the problem from the beginning of this article: how to map a point into a feature space? If we define a mapping

$$\mathbf{\Phi}(\mathbf{x}) = K(\mathbf{x}, \cdot) = (\sqrt{\lambda_1}\psi_1(\mathbf{x}), \sqrt{\lambda_2}\psi_2(\mathbf{x}), \cdots)^T$$

then we can map the point $\mathbf{x}$ to $\mathcal{H}$. Here $\mathbf{\Phi}$ is not a function, since it points to a vector or a funtion in the feature space $\mathcal{H}$. Then

$$< \mathbf{\Phi}(\mathbf{x}), \mathbf{\Phi}(\mathbf{y}) >_{\mathcal{H}} = < K(\mathbf{x}, \cdot), K(\mathbf{y}, \cdot) >_{\mathcal{H}} = K(\mathbf{x}, \mathbf{y})$$

As a result, we do not need to actually know what is the mapping, where is the feature space, or what is the basis of the feature space. For a symmetric positive-definite function K , there must exist at least one mapping $\mathbf{\Phi}$ and one feature space $\mathcal{H}$ so that

$$< \mathbf{\Phi}(\mathbf{x}), \mathbf{\Phi}(\mathbf{y}) > = K(\mathbf{x}, \mathbf{y})$$

which is the so-called *kernel trick*.

http://songcy.net/posts/story-of-basis-and-kernel-part-2/
https://blog.csdn.net/haolexiao/article/details/72171523?utm_source=itdadao&utm_medium=referral

软间隔与正则化

在前面的讨论中，我们一直假定训练样本在样本空间或特征空间中是线性可分的，即存在一个超平面能将不同类的样本完全划分开. 然而，在现实任务 中往往很难确定合适的核函数使得训练样本在特征空间中线性可分;退一步说， 即使恰好找到了某个核函数使训练集在特征空间中线性可分，也很难断定这个 貌似线性可分的结果不是由于过拟合所造成的. 缓解该问题的一个办法是允许支持向量机在一些样本上出错.为此，要引 入"软间隔" (soft margin)的概念， 如图 6.4所示

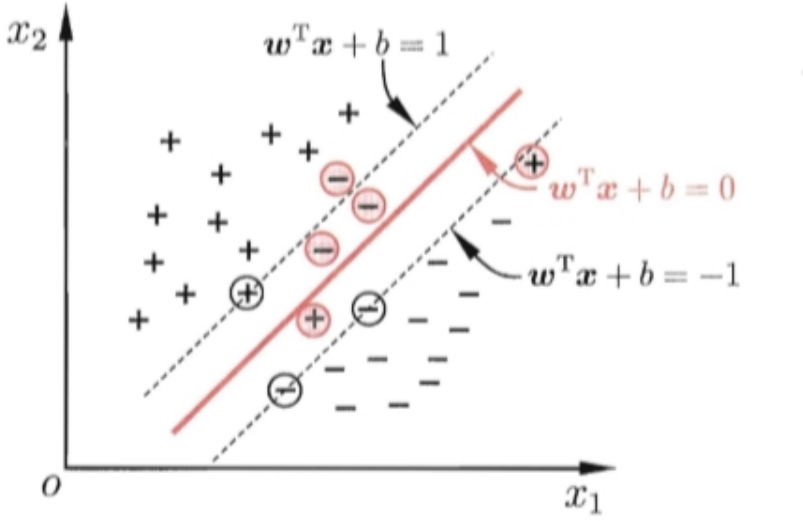


图 6.4 软间隔示意图. 红色圈出了一些不满足约束的样本.

前面介绍的支持向量机形式是要求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

$$y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b)\geqslant 1\ .$$

当然, 在最大化间隔的同时, 不满足约束的样本应尽可能少. 于是, 优化目标可写为

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m} \ell_{0/1}\left(y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) - 1\right) \ , \tag{6.29}$$

其中 $C > 0$ 是一个常数, $\ell_{0/1}$ 是 "0/1损失函数"

$$\ell_{0/1}(z) = \begin{cases} 1, & \text{if } z < 0; \\ 0, & \text{otherwise.} \end{cases} \tag{6.30}$$

显然, 当 C 为无穷大时, 式(6.29)迫使所有样本均满足约束(6.28), 于是式(6.29)等价于(6.6); 当 C 取有限值时, 式(6.29)允许一些样本不满足约束.

参数C调和间隔大小以及支持向量机对误分类的忍耐程度

然而, $\ell_{0/1}$ 非凸、非连续, 数学性质不太好, 使得式(6.29)不易直接求解. 于是, 人们通常用其他一些函数来代替 $\ell_{0/1}$, 称为“替代损失”(surrogate loss). 替代损失函数一般具有较好的数学性质, 如它们通常是凸的连续函数且是 $\ell_{0/1}$ 的上界. 图 6.5 给出了三种常用的替代损失函数:

$$\text{hinge 损失: } \ell_{hinge}(z) = \max(0, 1-z)\ ; \tag{6.31}$$

$$\text{指数损失(exponential loss): } \ell_{exp}(z) = \exp(-z)\ ; \tag{6.32}$$

$$\text{对率损失(logistic loss): } \ell_{log}(z) = \log(1+\exp(-z))\ . \tag{6.33}$$

若采用 hinge 损失, 则式(6.29)变成

$$\min_{\boldsymbol{w},b}\ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\max\left(0, 1-y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\right)\right)\ . \tag{6.34}$$

引入“松弛变量”(slack variables) $\xi_i \geqslant 0$, 可将式(6.34)重写为

$$\min_{\boldsymbol{w},b,\xi_i} \ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i \tag{6.35}$$

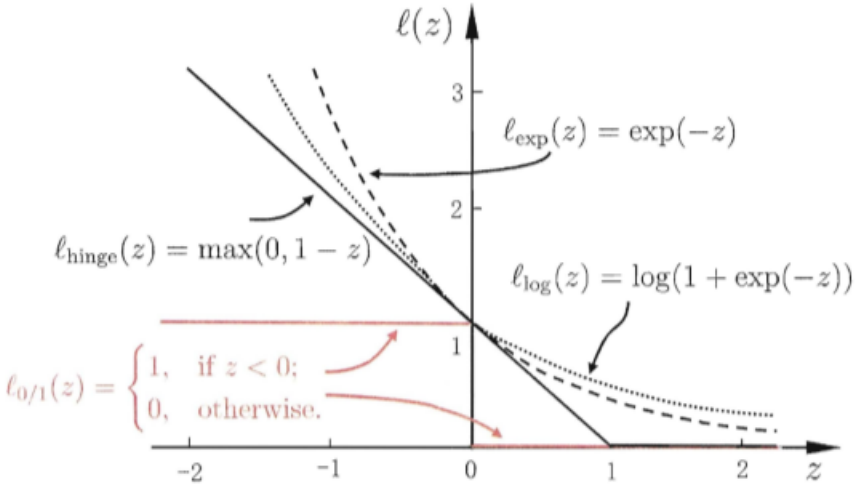


图 6.5 三种常见的替代损失函数: hinge损失、指数损失、对率损失

$$\text{s.t.} \ \ y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geqslant 1 - \xi_i$$

$$\xi_i \geqslant 0 \ , \ i = 1, 2, \ldots, m.$$

这就是常用的“软间隔支持向量机”.

与硬间隔求解方式类似，通过拉格朗日乘子法求解对偶问题

于是, 对任意训练样本 $(\boldsymbol{x}_i, y_i)$, 总有 $\alpha_i = 0$ 或 $y_i f(\boldsymbol{x}_i) = 1 - \xi_i$. 若 $\alpha_i = 0$, 则该样本不会对 $f(\boldsymbol{x})$ 有任何影响; 若 $\alpha_i > 0$, 则必有 $y_i f(\boldsymbol{x}_i) = 1 - \xi_i$, 即该样本是支持向量: 由式(6.39) 可知, 若 $\alpha_i < C$, 则 $\mu_i > 0$, 进而有 $\xi_i = 0$, 即该样本恰在最大间隔边界上; 若 $\alpha_i = C$, 则有 $\mu_i = 0$, 此时若 $\xi_i \leqslant 1$ 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. 由此可看出, 软间隔支持向量机的最终模型仅与支持向量有关, 即通过采用 hinge 损失函数仍保持了稀疏性.

$$C = \alpha_i + \mu_i \ .$$

$$\xi_i \geqslant 0 \ , \ \mu_i \xi_i = 0 \ .$$

$$\alpha_i \geqslant 0 \ , \quad \mu_i \geqslant 0 \ ,$$

$$\alpha_i \left(y_i f \left(\boldsymbol{x}_i \right) - 1 + \xi_i \right) = 0$$

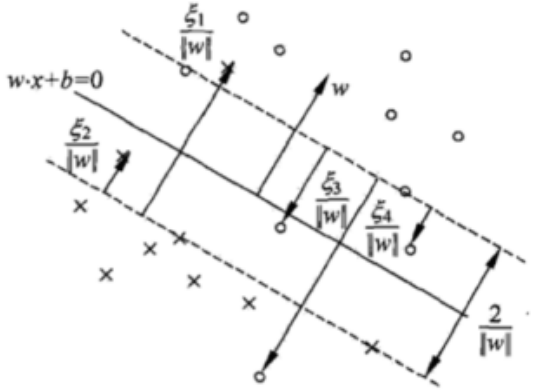


图 7.5 软间隔的支持向量

软间隔的支持向量 x_i 或者在间隔边界上，或者在间隔边界与分离超平面之间，或者在分离超平面误分一侧. 若 $\alpha_i^* < C$，则 $\xi_i = 0$，支持向量 x_i 恰好落在间隔边界上；若 $\alpha_i^* = C$，$0 < \xi_i < 1$，则分类正确，x_i 在间隔边界与分离超平面之间；若 $\alpha_i^* = C$，$\xi_i = 1$，则 x_i 在分离超平面上；若 $\alpha_i^* = C$，$\xi_i > 1$，则 x_i 位于分离超平面误分一侧.

我们还可以把式(6.29)中的 0/1 损失函数换成别的替代损失函数以得到其他学习模型, 这些模型的性质与所用的替代函数直接相关, 但它们具有一个共性: 优化目标中的第一项用来描述划分超平面的“间隔”大小, 另一项 $\sum_{i=1}^{m} \ell(f(\boldsymbol{x}_i), y_i)$ 用来表述训练集上的误差, 可写为更一般的形式

$$\min_{f} \ \Omega(f) + C \sum_{i=1}^{m} \ell(f(\boldsymbol{x}_i), y_i), \tag{6.42}$$

其中 $\Omega(f)$ 称为“结构风险”(structural risk), 用于描述模型 f 的某些性质; 第二项 $\sum_{i=1}^{m} \ell(f(\boldsymbol{x}_i), y_i)$ 称为“经验风险”(empirical risk), 用于描述模型与训练数据的契合程度; C 用于对二者进行折中. 从经验风险最小化的角度来看, $\Omega(f)$ 表述了我们希望获得具有何种性质的模型(例如希望获得复杂度较小的模型), 这为引入领域知识和用户意图提供了途径; 另一方面, 该信息有助于削减假设空间, 从而降低了最小化训练误差的过拟合风险. 从这个角度来说, 式(6.42)称为“正则化”(regularization)问题, $\Omega(f)$ 称为正则化项, C 则称为正则化常数. L_p 范数 (norm) 是常用的正则化项, 其中 L_2 范数 $\|\boldsymbol{w}\|_2$ 倾向于 $\boldsymbol{w}$ 的分量取值尽量均衡, 即非零分量个数尽量稠密, 而 L_0 范数 $\|\boldsymbol{w}\|_0$ 和 L_1 范数 $\|\boldsymbol{w}\|_1$ 则倾向于 $\boldsymbol{w}$ 的分量尽量稀疏, 即非零分量个数尽量少.

正则化可理解为一种 "罚函数法"即对不希 望得到的结采施以惩罚，从而使得优化过程趋向于 希望目标

支持向量机间隔最大化可以考虑为防止过拟合

支持向量回归

现在我们来考虑回归问题. 给定训练样本 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$, $y_i \in \mathbb{R}$, 希望学得一个形如式(6.7)的回归模型, 使得 $f(\boldsymbol{x})$ 与 y 尽可能接近, $\boldsymbol{w}$ 和 b 是待确定的模型参数.

对样本 $(\boldsymbol{x}, y)$, 传统回归模型通常直接基于模型输出 $f(\boldsymbol{x})$ 与真实输出 y 之间的差别来计算损失, 当且仅当 $f(\boldsymbol{x})$ 与 y 完全相同时, 损失才为零. 与此不同, 支持向量回归(Support Vector Regression, 简称 SVR)假设我们能容忍 $f(\boldsymbol{x})$ 与 y 之间最多有 ϵ 的偏差, 即仅当 $f(\boldsymbol{x})$ 与 y 之间的差别绝对值大于 ϵ 时才计算损失. 如图 6.6 所示, 这相当于以 $f(\boldsymbol{x})$ 为中心, 构建了一个宽度为 2ϵ 的间隔带, 若训练样本落入此间隔带, 则认为是被预测正确的.

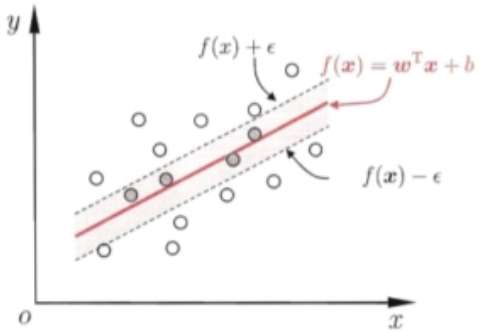


图 6.6 支持向量回归示意图. 红色显示出 ϵ-间隔带, 落入其中的样本不计算损失.

于是, SVR 问题可形式化为

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m} \ell_\epsilon(f(\boldsymbol{x}_i) - y_i) , \tag{6.43}$$

其中 C 为正则化常数, ℓ_ϵ 是图 6.7 所示的 ϵ-不敏感损失 (ϵ-insensitive loss) 函数

$$\ell_\epsilon(z) = \begin{cases} 0, & \text{if } |z| \leqslant \epsilon ; \\ |z| - \epsilon, & \text{otherwise.} \end{cases} \tag{6.44}$$

间隔带两侧的松弛程度可有所不同.

引入松弛变量 ξ_i 和 $\hat{\xi}_i$, 可将式(6.43)重写为

$$\min_{\boldsymbol{w},b,\xi_i,\hat{\xi}_i} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}(\xi_i + \hat{\xi}_i) \tag{6.45}$$

$$\text{s.t.} \quad f(\boldsymbol{x}_i) - y_i \leqslant \epsilon + \xi_i ,$$

$$y_i - f(\boldsymbol{x}_i) \leqslant \epsilon + \hat{\xi}_i ,$$

$$\xi_i \geqslant 0, \hat{\xi}_i \geqslant 0 , \ i = 1, 2, \ldots, m.$$

个人理解为：L2范数（结构风险最小化）+经验风险最小化

类似式(6.36), 通过引入拉格朗日乘子 $\mu_i \geqslant 0$, $\hat{\mu}_i \geqslant 0$, $\alpha_i \geqslant 0$, $\hat{\alpha}_i \geqslant 0$, 由拉格朗日乘子法可得到式(6.45)的拉格朗日函数

$$\begin{aligned} & L(\boldsymbol{w}, b, \boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}, \boldsymbol{\xi}, \hat{\boldsymbol{\xi}}, \boldsymbol{\mu}, \hat{\boldsymbol{\mu}}) \\ & = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}(\xi_i + \hat{\xi}_i) - \sum_{i=1}^{m}\mu_i\xi_i - \sum_{i=1}^{m}\hat{\mu}_i\hat{\xi}_i \\ & + \sum_{i=1}^{m}\alpha_i\big(f(\boldsymbol{x}_i) - y_i - \epsilon - \xi_i\big) + \sum_{i=1}^{m}\hat{\alpha}_i\big(y_i - f(\boldsymbol{x}_i) - \epsilon - \hat{\xi}_i\big) . \end{aligned} \tag{6.46}$$

将式(6.7)代入, 再令 $L(\boldsymbol{w}, b, \boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}, \boldsymbol{\xi}, \hat{\boldsymbol{\xi}}, \boldsymbol{\mu}, \hat{\boldsymbol{\mu}})$ 对 $\boldsymbol{w}$, b, ξ_i 和 $\hat{\xi}_i$ 的偏导为零可得

$$\boldsymbol{w} = \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i)\boldsymbol{x}_i , \tag{6.47}$$

$$0 = \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i) , \tag{6.48}$$

$$C = \alpha_i + \mu_i , \tag{6.49}$$

$$C = \hat{\alpha}_i + \hat{\mu}_i . \tag{6.50}$$

将式(6.47)–(6.50)代入式(6.46), 即可得到 SVR 的对偶问题

$$\begin{aligned} \max_{\boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}} \quad & \sum_{i=1}^{m} y_i(\hat{\alpha}_i - \alpha_i) - \epsilon(\hat{\alpha}_i + \alpha_i) \\ & - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j \\ \text{s.t.} \quad & \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i) = 0 , \\ & 0 \leqslant \alpha_i, \hat{\alpha}_i \leqslant C . \end{aligned} \tag{6.51}$$

由KKT条件得:

$$\begin{cases} \alpha_i(f(\boldsymbol{x}_i) - y_i - \epsilon - \xi_i) = 0 , \\ \hat{\alpha}_i(y_i - f(\boldsymbol{x}_i) - \epsilon - \hat{\xi}_i) = 0 , \\ \alpha_i\hat{\alpha}_i = 0 , \ \xi_i\hat{\xi}_i = 0 , \\ (C - \alpha_i)\xi_i = 0 , \ (C - \hat{\alpha}_i)\hat{\xi}_i = 0 . \end{cases} \tag{6.52}$$

可以看出, 当且仅当 $f(\boldsymbol{x}_i) - y_i - \epsilon - \xi_i = 0$ 时 α_i 能取非零值, 当且仅当 $y_i - f(\boldsymbol{x}_i) - \epsilon - \hat{\xi}_i = 0$ 时 $\hat{\alpha}_i$ 能取非零值. 换言之, 仅当样本 $(\boldsymbol{x}_i, y_i)$ 不落入 ϵ-间隔带中, 相应的 α_i 和 $\hat{\alpha}_i$ 才能取非零值. 此外, 约束 $f(\boldsymbol{x}_i) - y_i - \epsilon - \xi_i = 0$ 和 $y_i - f(\boldsymbol{x}_i) - \epsilon - \hat{\xi}_i = 0$ 不能同时成立, 因此 α_i 和 $\hat{\alpha}_i$ 中至少有一个为零.

将式(6.47)代入(6.7), 则 SVR 的解形如

$$f(\boldsymbol{x}) = \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i)\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x} + b . \tag{6.53}$$

能使式(6.53)中的 $(\hat{\alpha}_i - \alpha_i) \neq 0$ 的样本即为 SVR 的支持向量, 它们必落在 ϵ-间隔带之外. 显然, SVR 的支持向量仅是训练样本的一部分, 即其解仍具有稀疏性.

表示定理：

定理 6.2 (表示定理) 令 $\mathbb{H}$ 为核函数 κ 对应的再生核希尔伯特空间, $\|h\|_{\mathbb{H}}$ 表示 $\mathbb{H}$ 空间中关于 h 的范数, 对于任意单调递增函数 $\Omega : [0, \infty] \mapsto \mathbb{R}$ 和任意非负损失函数 $\ell : \mathbb{R}^m \mapsto [0, \infty]$, 优化问题

$$\min_{h \in \mathbb{H}} \quad F(h) = \Omega(\|h\|_{\mathbb{H}}) + \ell\big(h(\boldsymbol{x}_1), h(\boldsymbol{x}_2), \ldots, h(\boldsymbol{x}_m)\big) \tag{6.57}$$

的解总可写为

$$h^*(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i) \ . \tag{6.58}$$

表示定理对损失函数没有限制, 对正则化项 Ω 仅要求单调递增, 甚至不要求 Ω 是凸函数, 意味着对于一般的损失函数和正则化项, 优化问题(6.57)的最优解 $h^*(\boldsymbol{x})$ 都可表示为核函数 $\kappa(\boldsymbol{x}, \boldsymbol{x}_i)$ 的线性组合; 这显示出核函数的巨大威力.

核线性判别分析

我们先假设可通过某种映射 $\phi : \mathcal{X} \mapsto \mathbb{F}$ 将样本映射到一个特征空间 $\mathbb{F}$, 然后在 $\mathbb{F}$ 中执行线性判别分析, 以求得

$$h(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}) \ . \tag{6.59}$$

类似于式(3.35), KLDA 的学习目标是

$$\max_{\boldsymbol{w}} J(\boldsymbol{w}) = \frac{\boldsymbol{w}^{\mathrm{T}}\mathbf{S}_b^{\phi}\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}\mathbf{S}_w^{\phi}\boldsymbol{w}} \ , \tag{6.60}$$

其中 $\mathbf{S}_b^{\phi}$ 和 $\mathbf{S}_w^{\phi}$ 分别为训练样本在特征空间 $\mathbb{F}$ 中的类间散度矩阵和类内散度矩阵. 令 X_i 表示第 $i \in \{0,1\}$ 类样本的集合, 其样本数为 m_i; 总样本数 $m = m_0 + m_1$. 第 i 类样本在特征空间 $\mathbb{F}$ 中的均值为

$$\boldsymbol{\mu}_i^{\phi} = \frac{1}{m_i}\sum_{\boldsymbol{x}\in X_i}\phi(\boldsymbol{x}) \ , \tag{6.61}$$

两个散度矩阵分别为

$$\mathbf{S}_b^{\phi} = (\boldsymbol{\mu}_1^{\phi} - \boldsymbol{\mu}_0^{\phi})(\boldsymbol{\mu}_1^{\phi} - \boldsymbol{\mu}_0^{\phi})^{\mathrm{T}} \ ; \tag{6.62}$$

$$\mathbf{S}_w^{\phi} = \sum_{i=0}^{1}\sum_{\boldsymbol{x}\in X_i}\left(\phi(\boldsymbol{x}) - \boldsymbol{\mu}_i^{\phi}\right)\left(\phi(\boldsymbol{x}) - \boldsymbol{\mu}_i^{\phi}\right)^{\mathrm{T}} \ . \tag{6.63}$$

通常我们难以知道映射 ϕ 的具体形式, 因此使用核函数 $\kappa(\boldsymbol{x}, \boldsymbol{x}_i) = \phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x})$ 来隐式地表达这个映射和特征空间 $\mathbb{F}$. 把 $J(\boldsymbol{w})$ 作为式(6.57)中的损失函数 ℓ, 再令 $\Omega \equiv 0$, 由表示定理, 函数 $h(\boldsymbol{x})$ 可写为

$$h(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i) \ , \tag{6.64}$$

于是由式(6.59)可得

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i \phi(\boldsymbol{x}_i) \ . \tag{6.65}$$

令 $\mathbf{K} \in \mathbb{R}^{m\times m}$ 为核函数 κ 所对应的核矩阵, $(\mathbf{K})_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. 令 $\mathbf{1}_i \in \{1,0\}^{m\times 1}$ 为第 i 类样本的指示向量, 即 $\mathbf{1}_i$ 的第 j 个分量为 1 当且仅当 $\boldsymbol{x}_j \in X_i$, 否则 $\mathbf{1}_i$ 的第 j 个分量为 0. 再令

$$\hat{\boldsymbol{\mu}}_0 = \frac{1}{m_0}\mathbf{K}\mathbf{1}_0 \ , \tag{6.66}$$

$$\hat{\boldsymbol{\mu}}_1 = \frac{1}{m_1}\mathbf{K}\mathbf{1}_1 \ , \tag{6.67}$$

$$\mathbf{M} = (\hat{\boldsymbol{\mu}}_0 - \hat{\boldsymbol{\mu}}_1)(\hat{\boldsymbol{\mu}}_0 - \hat{\boldsymbol{\mu}}_1)^{\mathrm{T}} \ , \tag{6.68}$$

$$\mathbf{N} = \mathbf{K}\mathbf{K}^{\mathrm{T}} - \sum_{i=0}^{1} m_i \hat{\boldsymbol{\mu}}_i \hat{\boldsymbol{\mu}}_i^{\mathrm{T}} \ . \tag{6.69}$$

于是, 式(6.60)等价为

$$\max_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{M}\boldsymbol{\alpha}}{\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{N}\boldsymbol{\alpha}} \ . \tag{6.70}$$

显然, 使用线性判别分析求解方法即可得到 $\boldsymbol{\alpha}$, 进而可由式(6.64)得到投影函数 $h(\boldsymbol{x})$.

Thank you