支持向量机通俗导论

——理解 SVM 的三层境界

作者：July · pluskid
致谢：白石 · JerryLead

出处：结构之法算法之道 blog
http://blog.csdn.net/v_july_v/article/details/7624837

目录

前言

动笔写这个支持向量机（support vector machine）是费了不少劲和困难的，原因很简单，一者这个东西本身就并不好懂，要深入学习和研究下去需花费不少时间和精力，二者这个东西也不好讲清楚，尽管网上已经有朋友写得不错了（见文末参考链接），但在描述数学公式的时候还是显得不够。得益于同学白石的数学证明，我还是想尝试写一下，希望本文在兼顾通俗易懂的基础上，真真正正能足以成为一篇完整概括和介绍支持向量机的导论性的文章。

本文在写的过程中，参考了不少资料，包括《支持向量机导论》、《统计学习方法》及网友 pluskid 的支持向量机系列[1]等等，于此，还是一篇学习笔记，只是加入了自己的理解和总结，有任何不妥之处，还望海涵。全文宏观上整体认识支持向量机的概念和用处，微观上深究部分定理的来龙去脉，证明及原理细节，力保逻辑清晰 & 通俗易懂。

同时，阅读本文时建议大家尽量使用 chrome 等浏览器，如此公式才能更好的显示，再者，阅读时可拿张纸和笔出来，把本文所有定理.公式都亲自推导一遍或者直接打印下来（可直接打印网页版或本文文末附的 PDF，享受随时随地思考、演算的极致快感），在文稿上演算。

Ok，还是那句原话，有任何问题，欢迎任何人随时不吝指正 & 赐教，感谢。

[1] http://blog.pluskid.org/?page_id=683

第 1 层　了解 SVM

1.1　什么是 SVM

要明白什么是支持向量机 *Support Vector Machines, SVM*，便得从分类说起。

分类作为数据挖掘领域中一项非常重要的任务，它的目的是学会一个分类函数或分类模型（或者叫做分类器），该模型能吧数据库中的数据项映射到给定类别中的某一个，从而可以用于预测未知类别。

本文将要介绍的支持向量机算法便是一种分类方法。

支持向量机

所谓支持向量机，顾名思义，分为两个部分了解：一，什么是支持向量（简单来说，就是支持或支撑平面上把两类类别划分开来的超平面的向量点，下文将具体解释）；二，这里的“机（machine，机器）”便是一个算法。在机器学习领域，常把一些算法看做是一个机器，如分类机（当然，也叫做分类器），而支持向量机本身便是一种监督式学习的方法（至于具体什么是监督学习与非监督学习，请参见此系列Machine Learning & Data Mining 第一篇），它广泛的应用于统计分类以及回归分析中。

而支持向量机是 90 年代中期发展起来的基于统计学习理论的一种机器学习方法，通过寻求结构化风险最小来提高学习机泛化能力，实现经验风险和置信范围的最小化，从而达到在统计样本量较少的情况下，亦能获得良好统计规律的目的。

对于不想深究支持向量机原理的同学（比如就只想看看支持向量机是干嘛的），那么，了解到这里便足够了，不需上层。而对于那些喜欢深入研究一个东西的同学，甚至究其本质的，咱们则还有很长的一段路要走，万里长征，咱们开始迈第一步吧（相信你能走完）。

1.2　线性分类

OK，在讲 SVM 之前，咱们必须先弄清楚一个概念：线性分类器（也可以叫做感知机，这里的机表示的还是一种算法，本文第三部分“证明 SVM”中会详细阐述）。

1.2.1　分类标准

这里我们考虑的是一个两类的分类问题，数据点用 $\boldsymbol{x}$ 来表示，这是一个 n 维向量，$\boldsymbol{w}^{\mathrm{T}}$ 上标中的“T”代表转置，而类别用 y 来表示，可以取 1 或者 –1，分别代表两个不同的类。一个线性分类器就是要在 n 维的数据空间中找到一个超平面，其方程可以表示为：

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=0 \tag{1.2.1}$$

上面给出了线性分类的定义描述，但或许读者没有想过：为何用 y 取 1 或者 –1 来表示两个不同的类别呢？其实，这个 1 或 –1 的分类标准起源于 Logistic 回归，为了完整和过渡的自然性，咱们就再来看看这个 Logistic 回归。

1.2.2 1 或 −1 分类标准的起源：Logistic 回归

Logistic 回归目的是从特征学习出一个 0/1 分类模型，而这个模型是将特性的线性组合作为自变量，由于自变量的取值范围是负无穷到正无穷。因此，使用 Logistic 函数（或称作 Sigmoid 函数）将自变量映射到 $(0,1)$ 上，映射后的值被认为是属于 $y=1$ 的概率。形式化表示就是：假设函数

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}\right) \tag{1.2.2}$$

其中 $\boldsymbol{x}$ 是 n 维特征向量，函数 g 就是 Logistic 函数。对于一元变量的情形，$g(z)=\frac{1}{1+\mathrm{e}^{-z}}$ 的图像是

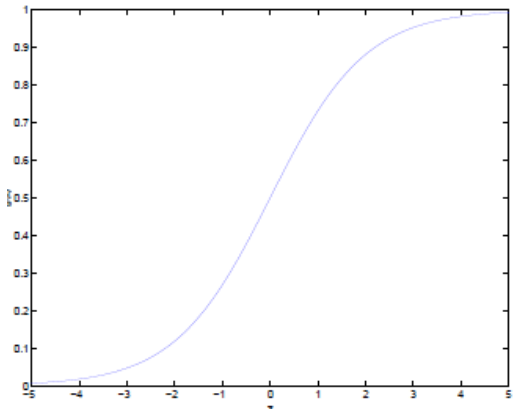

图 1.1: $g(z)$ 的函数图像

可以看到，$g(\cdot)$ 将 $(-\infty,+\infty)$ 映射到了 $(0,1)$，从而1.2.2式可以作为 $\boldsymbol{x}$ 对应 $y=1$ 的概率

$$\Pr\{y=1\,|\,\boldsymbol{x};\boldsymbol{\theta}\} = h_{\boldsymbol{\theta}}(\boldsymbol{x}) \tag{1.2.3}$$

$$\Pr\{y=0\,|\,\boldsymbol{x};\boldsymbol{\theta}\} = 1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}) \tag{1.2.4}$$

当我们要判别一个新来的数据点 $\boldsymbol{x}$ 属于哪个类时，只需求 $h_{\boldsymbol{\theta}}(\boldsymbol{x})$，若大于 0.5 就是 $y=1$ 的类，反之属于 $y=0$ 类。

再审视一下 $h_{\boldsymbol{\theta}}(\boldsymbol{x})$，发现 $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ 只和 $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}$ 有关，$\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}>0$，那么 $h_{\boldsymbol{\theta}}(\boldsymbol{x})>0.5$，$g(\boldsymbol{z})$ 只不过是用来映射，真实的类别决定权还在 $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}$。还有当 $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}\gg 0$ 时，$h_{\boldsymbol{\theta}}(\boldsymbol{x})=1$，反之 $h_{\boldsymbol{\theta}}(\boldsymbol{x})=0$。如果我们只从 $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}$ 出发，希望模型达到的目标无非就是让训练数据中 $y=1$ 的特征 $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}\gg 0$，而使 $y=0$ 的特征 $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}\ll 0$。Logistic 回归就是要学习得到 $\boldsymbol{\theta}$，使得正例的特征远大于 0，负例的特征远小于 0，强调在全部训练实例上达到这个目标。

1.2.3 形式化表示

我们这次使用的结果标签是 $y=-1$，$y=1$，替换在 logistic 回归中使用的 $y=0$ 和 $y=1$。同时将 $\boldsymbol{\theta}$ 替换成 $\boldsymbol{w}$ 和 b。以前的 $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}=\theta_0+\theta_1x_1+\theta_2x_2+\cdots+\theta_nx_n$，其中认为 $x_0=1$。现在我们替换 θ_0 为 b，后面替换 $\theta_1x_1+\theta_2x_2+\cdots+\theta_nx_n$ 为 $w_1x_1+w_2x_2+\cdots+w_nx_n$（即 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$）。这样，我们让 $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$，进一步 $h_{\boldsymbol{\theta}}(\boldsymbol{x})=g(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x})=g(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)$。也就是说除了 y 由 $y=0$ 变为 $y=-1$，只是标记不同外，与 logistic 回归的形式化表示没区别。

再明确下假设函数

$$h_{\boldsymbol{w},b}(\boldsymbol{x}) = g(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b) \tag{1.2.5}$$

上面提到过我们只需考虑 $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}$ 的正负问题，而不用关心 $g(z)$，因此我们这里将 $g(z)$ 做一个简化，将其简单映射到 $y=-1$ 和 $y=1$ 上。映射关系如下：

$$g(z) = \begin{cases} 1, & z\geq 0 \\ -1, & z<0 \end{cases} \tag{1.2.6}$$

于此，想必已经解释明白了为何线性分类的标准一般用 1 或者 –1 来表示。

注：上小节来自 Jerrylead 所做的斯坦福机器学习课程的笔记。

1.3 线性分类的一个例子

下面举个简单的例子，一个二维平面（一个超平面，在二维空间中的例子就是一条直线），如下图所示，平面上有两种不同的点，分别用两种不同的颜色表示，一种为红颜色的点，另一种则为蓝颜色的点，红颜色的线表示一个可行的超平面。

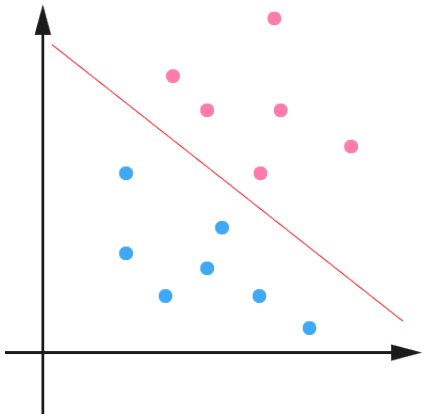

图 1.2: 一个二维平面上线性分类问题的例子

从图1.2中我们可以看出，这条红颜色的线把红颜色的点和蓝颜色的点分开来了。而这条红颜色的线就是我们上面所说的超平面，也就是说，这个所谓的超平面的的确确便把这两种不同颜色的数据点分隔开来，在超平面一边的数据点所对应的 y 全是 -1，而在另一边全是 1。

接着，我们令分类函数[1]

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b \tag{1.3.1}$$

显然，如果 $f(\boldsymbol{x}) = 0$，那么 $\boldsymbol{x}$ 是位于超平面上的点。我们不妨要求对于所有满足 $f(\boldsymbol{x}) < 0$ 的点，其对应的 y 等于 -1，而 $f(\boldsymbol{x}) > 0$ 则对应 $y = 1$ 的数据点。[2] [3]

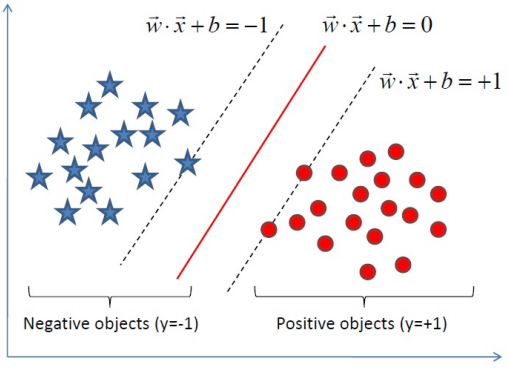

图 1.3: 一个二维平面上线性分类问题的例子：正样例和负样例

当然，有些时候，或者说大部分时候数据并不是线性可分的，这个时候满足这样条件的超平面就根本不存在（不过关于如何处理这样的问题我们后面会讲），这里先从最简单的情形开始推导，就假设数据都是线性可分的，亦即这样的超平面是存在的。

更进一步，我们在进行分类的时候，将数据点 $\boldsymbol{x}$ 代入 $f(\boldsymbol{x})$ 中，如果得到的结果小于 0，则赋予其类别 -1，如果大于 0 则赋予类别 1。如果 $f(\boldsymbol{x}) = 0$，则很难办了，分到哪一类都不是。

[1] 提醒：下文很大篇幅都在讨论着这个分类函数。

[2] 注：图1.3中，定义特征到结果的输出函数 $u = \vec{w}\cdot\vec{x} - b$，与我们之前定义的 $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$ 实质是一样的。

[3] 有一朋友飞狗来自 Mare_Desiderii，看了上面的定义之后，问到：请教一下 SVM functional margin 为 $\hat{\gamma} = y(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b) = yf(\boldsymbol{x})$ 中的 y 是只取 1 和 -1 吗？y 的唯一作用就是确保 functional margin 的非负性？真是这样的么？当然不是，详情请见本文评论下第 43 楼。

请读者注意，下面的篇幅将按下述 3 点走：

(1) 咱们就要确定上述分类函数 $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$（$\boldsymbol{w} \cdot \boldsymbol{x}$ 表示 $\boldsymbol{w}$ 与 $\boldsymbol{x}$ 的内积）中的两个参数 $\boldsymbol{w}$ 和 b，通俗理解的话 $\boldsymbol{w}$ 是法向量，b 是截距（再次说明：定义特征到结果的输出函数 $u = \vec{w} \cdot \vec{x} - b$，与我们最开始定义的 $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$ 实质是一样的）。

(2) 那如何确定 $\boldsymbol{w}$ 和 b 呢？答案是寻找两条边界端或极端划分直线中间的最大间隔（之所以要寻最大间隔是为了能更好的划分不同类的点，下文你将看到：为寻最大间隔，导出 $\frac{1}{2}\|\boldsymbol{w}\|^2$，继而引入拉格朗日函数和对偶变量 $\boldsymbol{\alpha}$，化为对单一因数对偶变量 $\boldsymbol{\alpha}$ 的求解，当然，这是后话），从而确定最终的最大间隔分类超平面和分类函数；

(3) 进而把寻求分类函数 $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$ 的问题转化为对 $\boldsymbol{w}$、b 的最优化问题，最终化为对偶因子的求解。

总结成一句话即是：从最大间隔出发（目的本就是为了确定法向量 $\boldsymbol{w}$），转化为求对变量 $\boldsymbol{w}$ 和 b 的凸二次规划问题。亦或如下所示。[4]

研究者 July

为确定分类函数 $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$ 中的参数 $\boldsymbol{w}$ 和 b，于是寻找最大分类间隔，导出 $\frac{1}{2}\|\boldsymbol{w}\|^2$，继而引入朗格朗日函数，化为对单一因子对偶变量 $\boldsymbol{\alpha}$ 的求解，如此，求 $\boldsymbol{w} \cdot \boldsymbol{x}$ 与求 $\boldsymbol{\alpha}$ 等价，而求 $\boldsymbol{\alpha}$ 的解法即为 SMO。把求分类函数 $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$ 的问题转化到求最大分类间隔，继而再转化为对 $\boldsymbol{w}$、b 的最优化问题，即凸二次规划问题，妙。

1.4 函数间隔与几何间隔 *Functional Margin & Geometric Margin*

一般而言，一个点距离超平面的远近可以表示为分类预测的确信或准确程度。

在超平面 $\boldsymbol{w} \cdot \boldsymbol{x} + b$ 确定的情况下，$|\boldsymbol{w} \cdot \boldsymbol{x} + b|$ 能够相对的表示点 $\boldsymbol{x}$ 到距离超平面的远近，而 $\boldsymbol{w} \cdot \boldsymbol{x} + b$ 的符号与类标记 y 的符号是否一致表示分类是否正确，所以，可以用量 $y \cdot \boldsymbol{w} \cdot \boldsymbol{x} + b$ 的正负性来判定或表示分类的正确性和确信度。

于此，我们便引出了定义样本到分类间隔距离的函数间隔的概念。

1.4.1 函数间隔 *Functional Margin*

我们定义函数间隔为

$$\hat{\gamma} = y(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b) = yf(\boldsymbol{x}) \tag{1.4.1}$$

接着，我们定义超平面 $(\boldsymbol{w}, b)$ 关于训练数据集 T 的函数间隔为超平面 $(\boldsymbol{w}, b)$ 关于 T 中所有样本点 $(\boldsymbol{x}_i, y_i)$ 的函数间隔最小值，其中 $\boldsymbol{x}$ 是特征，y 是结果标签，i 表示第 i 个样本，有

$$\hat{\gamma} = \min \hat{\gamma}_i, i = 1, 2, \cdots, n \tag{1.4.2}$$

然与此同时，问题就出来了。上述定义的函数间隔虽然可以表示分类预测的正确性和确信度，但在选择分类超平面时，只有函数间隔还远远不够，因为如果成比例的改变 $\boldsymbol{w}$ 和 b，如将他们改变为 $2\boldsymbol{w}$ 和 $2b$，虽然此时超平面没有改变，但函数间隔的值 $yf(\boldsymbol{x})$ 却变成了原来的 4 倍。

其实，我们可以对法向量 $\boldsymbol{w}$ 加些约束条件，使其表面上看起来规范化，如此，我们很快又将引出真正定义点到超平面的距离——几何间隔的概念。[5]

1.4.2 点到超平面的距离定义：几何间隔 *Geometric Margin*

在给出几何间隔的定义之前，咱们首先来看下，如图1.4所示，对于一个点 $\boldsymbol{x}$，令其垂直投影到超平面上的

[4] 有点需要注意，如读者@ 酱爆小八爪所说：从最大分类间隔开始，就一直是凸优化问题。

[5] 很快你将看到，几何间隔就是函数间隔除以个 $\|\boldsymbol{w}\|$，即 $\frac{yf(\boldsymbol{x})}{\|\boldsymbol{w}\|}$。

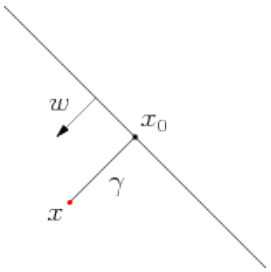

图 1.4: 几何间隔

对应的为 $\boldsymbol{x}_0$，由于 $\boldsymbol{w}$ 是垂直于超平面的一个向量，γ 为样本 $\boldsymbol{x}$ 到分类间隔的距离，我们有[6]

$$\boldsymbol{x} = \boldsymbol{x}_0 + \gamma \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \tag{1.4.3}$$

又由于 $\boldsymbol{x}_0$ 是超平面上的点，满足 $f(\boldsymbol{x}_0) = 0$，代入超平面的方程即可算出[7]

$$\gamma = \frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b}{\|\boldsymbol{w}\|} = \frac{f(\boldsymbol{x})}{\|\boldsymbol{w}\|} \tag{1.4.4}$$

不过这里的 γ 是带符号的，我们需要的只是它的绝对值，因此类似地，也乘上对应的类别 y 即可，因此实际上我们定义几何间隔为[8] [9]

$$\tilde{\gamma} = y\gamma = \frac{\hat{\gamma}}{\|\boldsymbol{w}\|} \tag{1.4.5}$$

正如本文评论下读者**popol1991**留言：函数间隔 $y(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b) = yf(\boldsymbol{x})$ 实际上就是 $|f(\boldsymbol{x})|$，只是人为定义的一个间隔度量；而几何间隔 $\frac{|f(\boldsymbol{x})|}{\|\boldsymbol{w}\|}$ 才是直观上的点到超平面距离。

想想二维空间里的点到直线公式：假设一条直线的方程为 $ax + by + c = 0$，点 P 的坐标是 (x_0, y_0)，则点到直线距离为 $\frac{|ax_0+by_0+c|}{\sqrt{a^2+b^2}}$。

点到平面的距离

若点坐标为 (x_0, y_0, z_0)，平面为 $Ax + By + Cz + D = 0$，则点到平面的距离为：

$$d = \left| \frac{Ax_0 + By_0 + Cz_0 + D}{\sqrt{A^2 + B^2 + C^2}} \right| \tag{1.4.6}$$

那么如果用向量表示，设 $\boldsymbol{w} = (a, b)$，$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + c$，那么这个距离正是 $\frac{|f(\boldsymbol{x})|}{\|\boldsymbol{w}\|}$。

1.5 最大间隔分类器 *Maximum Margin Classifier*

于此，我们已经很明显的看出，函数间隔和几何间隔相差一个 $\|\boldsymbol{w}\|$ 的缩放因子。按照我们前面的分析，对一个数据点进行分类，当它的间隔越大的时候，分类正确的把握越大。对于一个包含 n 个点的数据集，我们可以很自然地定义它的间隔为所有这 n 个点的间隔中最小的那个。于是，为了使得分类的把握尽量大，我们希望所选择的超平面能够最大化这个间隔值。

通过上节，我们已经知道

1. 函数间隔明显是不太适合用来最大化的一个量，因为在超平面固定以后，我们可以等比例地缩放 $\boldsymbol{w}$ 的长度和 b 的值，这样可以使得 $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$ 的值任意大，亦即函数间隔 $\hat{\gamma}$ 可以在超平面保持不变的情况下被取得任意大。
2. 而几何间隔则没有这个问题，因为除上了这个分母，所以缩放 $\boldsymbol{w}$ 和 b 的时候的值是不会改变的，它只随着超平面的变动而变动，因此，这是更加合适的一个间隔。

[6] $\|\cdot\|$ 表示的是范数，关于范数的概念可参见百度百科：http://baike.baidu.com/view/637132.htm。
[7] 有的书上会写成把 $\|\boldsymbol{x}\|$ 分开相除的形式，如本文参考文献及推荐阅读条目 11，其中，$\|\boldsymbol{w}\|$ 为 $\boldsymbol{w}$ 的 2-范数。
[8] 注：别忘了，上面 $\hat{\gamma}$ 的定义，$\hat{\gamma} = y(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b) = yf(\boldsymbol{x})$。
[9] 代入相关式子可以得出：$y_i \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} + \frac{b}{\|\boldsymbol{w}\|}$???

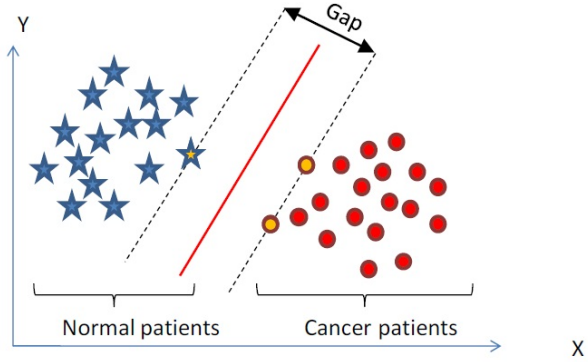

图 1.5: 最大化分类间隔

这样一来，我们的最大间隔分类器的目标可以定义为

$$\max \tilde{\gamma} \tag{1.5.1}$$

当然，还需要满足一些条件，根据间隔的定义，我们有

$$y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) = \hat{\gamma}_i \geq \hat{\gamma}, i = 1, 2, \cdots, n \tag{1.5.2}$$

其中 $\hat{\gamma} = \tilde{\gamma}\|\boldsymbol{w}\|$（等价于 $\tilde{\gamma} = \frac{\hat{\gamma}}{\|\boldsymbol{w}\|}$，固有稍后的 $\hat{\gamma} = 1$ 时，$\tilde{\gamma} = \frac{1}{\|\boldsymbol{w}\|}$），出于方便推导和优化的目的，我们可以令 $\tilde{\gamma} = 1$（对目标函数的优化没有影响，至于为什么，请见本文评论下第 42 楼回复），此时，上述的目标(1.5.1)转化为

$$\max \frac{1}{\|\boldsymbol{w}\|} \tag{1.5.3}$$

$$s.t. \quad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 1, i = 1, 2, \cdots, n \tag{1.5.4}$$

通过求解这个问题，我们就可以找到一个间隔最大的分类器。如图1.6所示，中间的红色线条是最优超平面，另外两条线到红线的距离都是等于 $\tilde{\gamma}$ 的。[10]

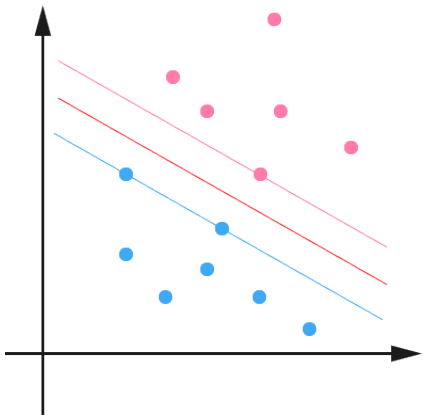

图 1.6: 最大间隔分类器

通过最大化间隔，我们使得该分类器对数据进行分类时具有了最大的把握。但，这个最大分类间隔器到底是用来干嘛的呢？很简单，支持向量机通过使用最大分类间隔来设计决策最优分类超平面，而为何是最大间隔，却不是最小间隔呢？因为最大间隔能获得最大稳定性与区分的确信度，从而得到良好的推广能力（超平面之间的距离越大，分离器的推广能力越好，也就是预测精度越高，不过对于训练数据的误差不一定是最小的）。

[10] 这里 γ 便是上文所定义的几何间隔，当令 $\hat{\gamma}$=1 时，γ 便为 $\frac{1}{\|\boldsymbol{w}\|}$，而我们上面得到的目标函数便是在相应的约束条件下，要最大化这个 $\frac{1}{\|\boldsymbol{w}\|}$ 值。

1.6 到底什么是支持向量 *Support Vector*

1.5节，我们介绍了最大间隔分类器，但并没有具体阐述到底什么是支持向量，本节，咱们来重点阐述这个概念。咱们不妨先来回忆一下1.5节的图1.6。

从图上可以看到两个支撑着中间的间隙的超平面，它们到中间的纯红线——分离超平面的距离相等，即我们所能得到的最大的几何间隔 $\tilde{\gamma}$，而“支撑”这两个超平面的必定会有一些点，而这些“支撑”的点便叫做支持向量支持向量。

很显然，由于这些支持向量刚好在边界上，所以它们满足 $y(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)=1$（还记得我们把函数间隔定为 1 了吗？1.5节中：“处于方便推导和优化的目的，我们可以令 $\hat{\gamma}$=1”），而对于所有不是支持向量的点，也就是在“阵地后方”的点，则显然有 $y(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)>1$。当然，通常除了 k 近邻之类的“Memory-based Learning”算法，通常算法也都不会直接把所有的点记忆下来，并全部用来做后续推断中的计算。不过，如果算法使用了核方法进行非线性化推广的话，就会遇到这个问题了。核方法在2.2节中介绍）。

OK，到此为止，算是了解到了 SVM 的第一层，对于那些只关心怎么用 SVM 的朋友便已足够，不必再更进一层深究其更深的原理。

第 2 层　深入 SVM

2.1　从线性可分到线性不可分

2.1.1　从原始问题到对偶问题

当然，除了在上文中所介绍的从几何直观上之外，支持向量的概念也可以从其优化过程的推导中得到。虽然1.5节给出了优化目标，却没有讲怎么来求解。现在就让我们来处理这个问题。回忆一下之前得到的优化目标

$$\max \frac{1}{\|\boldsymbol{w}\|} \tag{2.1.1}$$

$$s.t.\ \ y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 1, i = 1, 2, \cdots, n \tag{2.1.2}$$

由于求 $\frac{1}{\|\boldsymbol{w}\|}$ 的最大值相当于求 $\frac{1}{2}\|\boldsymbol{w}\|^2$ 的最小值，所以上述问题等价于（$\boldsymbol{w}$ 由分母变成分子，从而也有原来的“最大化”问题变为“最小化”问题，很明显，两者问题等价）

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{2.1.3}$$

$$s.t.\ \ y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 1, i = 1, 2, \cdots, n \tag{2.1.4}$$

到这个形式以后，就可以很明显地看出来，它是一个凸优化问题，或者更具体地说，它是一个二次优化问题——目标函数是二次的，约束条件是线性的。这个问题可以用任何现成的 QP *Quadratic Programming* 的优化包进行求解。

虽然这个问题确实是一个标准的 QP 问题，但是它也有它的特殊结构，通过 Lagrange 对偶变换到对偶变量 *Dual Variable* 的优化问题之后，可以找到一种更加有效的方法来进行求解，而且通常情况下这种方法比直接使用通用的 QP 优化包进行优化要高效得多。

也就说，除了用解决 QP 问题的常规方法之外，还可以应用 Lagrange 对偶性，通过求解对偶问题得到最优解，这就是线性可分条件下支持向量机的对偶算法，这样做的优点在于：一者对偶问题往往更容易求解；二者可以自然的引入核函数，进而推广到非线性分类问题。

接下来，你将看到“对偶变量的优化问题”等类似的关键词频繁出现，便是解决此凸优化问题的第二种更为高效的解——对偶变量的优化求解。

至于上述提到，关于什么是 Lagrange 对偶性？简单地来说，通过给每一个约束条件加上一个 Lagrange 乘子，即引入 Lagrange 对偶变量 $\boldsymbol{\alpha}$，如此我们便可以通过 Lagrange 函数将约束条件融和到目标函数里去（也就是说把条件融合到一个函数里头，现在只用一个函数表达式便能清楚的表达出我们的问题）。

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1\right) \tag{2.1.5}$$

然后我们令

$$\theta(\boldsymbol{w}) = \max_{\alpha_i > 0} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) \tag{2.1.6}$$

容易验证，当某个约束条件不满足时，例如 $y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) < 1$，那么我们显然有 $\theta(\boldsymbol{w}) = +\infty$（只要令 $\alpha_i = +\infty$ 即可）。而当所有约束条件都满足时，则有 $\theta(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2$，亦即我们最初要最小化的量。因此，在要求约束条件得到满足的情况下最小化 $\frac{1}{2}\|\boldsymbol{w}\|^2$，实际上等价于直接最小化 $\theta(\boldsymbol{w})$（当然，这里也有约束条件，就是

$\alpha_i \geq 0, i = 1, 2, \cdots, n$，因为如果约束条件没有得到满足，$\theta(\boldsymbol{w})$ 会等于无穷大，自然不会是我们所要求的最小值。具体写出来，我们现在的目标函数变成了

$$\min_{\boldsymbol{w},b} \theta(\boldsymbol{w}) = \min_{\boldsymbol{w},b} \max_{\alpha_i \geq 0} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = p^* \tag{2.1.7}$$

这里用 p^* 表示这个问题的最优值，这个问题和我们最初的问题是等价的。不过，现在我们来把最小和最大的位置交换一下（稍后，你将看到，当(2.1.8)满足了一定的条件之后，这个式子 d^* 便是(2.1.7)式 p^* 的对偶形式表示）。

$$\max_{\alpha_i \geq 0} \min_{\boldsymbol{w},b} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = d^* \tag{2.1.8}$$

当然，交换以后的问题不再等价于原问题，这个新问题的最优值用 d^* 来表示。并且，我们有 $d^* \leq p^*$，这在直观上也不难理解，最大值中最小的一个总也比最小值中最大的一个要大吧！总之，第二个问题的最优值 d^* 在这里提供了一个第一个问题的最优值 p^* 的一个下界，在满足某些条件的情况下，这两者相等，这个时候我们就可以通过求解第二个问题来间接地求解第一个问题。

上段说“在满足某些条件的情况下”，这所谓的“满足某些条件”就是要先满足 Slater 条件，进而就满足 KKT 条件。理由如下 3 点所述（观点来自 frestyle）：

1. 在凸优化问题中，d^* 和 p^* 相同的条件是 Slater 条件，该条件保证鞍点 *Saddle Point* 存在。
2. 至于 KKT 条件，首先原问题的最优值可以通过求 Lagrange 函数的鞍点（如果有的话）来得到，再者，KKT 条件里面进一步引入了更强的前提，也就是在满足 Slater 条件的同时（前面说了，Slater 条件保证鞍点存在），$f(\cdot)$ 和 $g_i(\cdot)$ 都是可微的，这样鞍点不仅存在，而且能通过对 Lagrange 函数求导得到，
3. 所以 KKT 条件是一个点是最优解的条件，而不是 $d^* = p^*$ 的条件，当然这个 KKT 条件对后边简化对偶问题很关键。

那 KKT 条件的表现形式是什么呢？据维基百科：KKT 条件的介绍，一般地，一个最优化数学模型能够表示成下列标准形式

$$\min f(\boldsymbol{x}) \tag{2.1.9}$$

$$s.t. \quad h_j(\boldsymbol{x}) = 0, j = 1, 2, \cdots, p \tag{2.1.10}$$

$$g_k(\boldsymbol{x}) \leq 0, k = 1, 2, \cdots, q \tag{2.1.11}$$

$$\boldsymbol{x} \in X \subset \mathbb{R}^n \tag{2.1.12}$$

所谓 Karush-Kuhn-Tucker 最优化条件，就是指上式的最小点 $\boldsymbol{x}^*$ 必须满足下面的条件：

1. $h_j(\boldsymbol{x}^*) = 0$，$g_k(\boldsymbol{x}^*) \leq 0$，$j = 1, 2, \cdots, p$，$k = 1, 2, \cdots, q$
2. $\nabla f(\boldsymbol{x}^*) + \Sigma_{j=1}^{p} \lambda_j \nabla h_j(\boldsymbol{x}^*) + \Sigma_{k=1}^{q} \mu_k \nabla g_k(\boldsymbol{x}^*) = \boldsymbol{0}$，$\lambda_j \neq 0$，$\mu_k \geq 0$，$\mu_k g_k(\boldsymbol{x}^*) = 0$

经过论证，我们这里的问题是满足 KKT 条件的（首先已经满足 Slater 条件，再者 $f(\cdot)$ 和 $g_i(\cdot)$ 也都是可微的，即 $\mathcal{L}$ 对 $\boldsymbol{w}$ 和 b 都可导），因此现在我们便转化为求解第二个问题。也就是说，现在，咱们的原问题通过满足一定的条件，已经转化成了对偶问题。而求解这个对偶学习问题，分为 3 个步骤，首先要让 $\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha})$ 关于 $\boldsymbol{w}$ 和 b 最小化，然后求对 $\boldsymbol{\alpha}$ 的极大，最后利用 SMO 算法求解对偶因子。

第一步 首先固定 $\boldsymbol{\alpha}$，要让 $\mathcal{L}$ 关于 $\boldsymbol{w}$ 和 b 最小化，我们分别对 $\boldsymbol{w}$ 和 b 求偏导数，即令 $\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}}$ 和 $\frac{\partial \mathcal{L}}{\partial b}$ 等于 0。[1]

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0 \Rightarrow \boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \tag{2.1.13}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.1.14}$$

[1] 对 $\boldsymbol{w}$ 求导结果的解释请看本文评论下第 45 楼回复

以上结果代回(2.1.5)式得到

$$\begin{aligned}\mathcal{L}(\boldsymbol{w},b,\boldsymbol{\alpha}) &= \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n}\alpha_i\left[y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\right)-1\right]\\ &= \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} - \boldsymbol{w}^{\mathrm{T}}\sum_{i=1}^{n}\alpha_i y_i\boldsymbol{x}_i - b\sum_{i=1}^{n}\alpha_i y_i + \sum_{i=1}^{n}\alpha_i\\ &= \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\sum_{i=1}^{n}\alpha_i y_i\boldsymbol{x}_i - \boldsymbol{w}^{\mathrm{T}}\sum_{i=1}^{n}\alpha_i y_i\boldsymbol{x}_i - b\cdot 0 + \sum_{i=1}^{n}\alpha_i\\ &= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i y_i\boldsymbol{x}_i\right)^{\mathrm{T}}\sum_{i=1}^{n}\alpha_i y_i\boldsymbol{x}_i\\ &= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j \end{aligned} \tag{2.1.15}$$

从上面的最后一个式子，我们可以看出，此时的 Lagrange 函数只包含了一个变量，那就是 α_i，然后下文的第 2 步，求出 α_i 了便能求出 $\boldsymbol{w}$ 和 b，由此可见，上文第 1.2 节提出来的核心问题：分类函数 $f(\boldsymbol{x})=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$ 也就可以轻而易举的求出来了。

由此看出，使用 Lagrange 定理解凸最优化问题可以使用一个对偶变量表示，转换为对偶问题后，通常比原问题更容易处理，因为直接处理不等式约束是困难的，而对偶问题通过引入 Lagrange 乘子（又称为对偶变量）来解。

第二步 求对 $\boldsymbol{\alpha}$ 的极大，即是关于对偶变量 $\boldsymbol{\alpha}$ 的优化问题，从上面的式子得到

$$\max_{\boldsymbol{\alpha}}\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j \tag{2.1.16}$$

$$s.t.\ \alpha_i\geq 0, i=1,2,\cdots,n \tag{2.1.17}$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \tag{2.1.18}$$

如前面所说，这个问题有更加高效的优化算法，即我们常说的 SMO 算法。

不得不提醒下读者：经过上面第一个步骤的求 $\boldsymbol{w}$ 和 b，得到的 Lagrange 函数已经没有了变量 $\boldsymbol{w}$ 和 b，只有 $\boldsymbol{\alpha}$，而反过来，求得的 $\boldsymbol{\alpha}$ 将能导出 $\boldsymbol{w}$ 和 b 的解，最终得出分离超平面和分类决策函数。为何呢？因为如果求出了 α_i，根据 $\boldsymbol{w}=\Sigma_{i=1}^{n}\alpha_i y_i\boldsymbol{\alpha}_i$，即可求出 $\boldsymbol{w}$。然后通过 $b=-\frac{1}{2}\big(\max\limits_{i:y_i=-1}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+\min\limits_{i:y_i=1}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i\big)$，即可求出 b。

2.1.2 序列最小最优化算法 *SMO*

细心的读者读至上节末尾处，怎么求对偶变量 $\boldsymbol{\alpha}$ 的值可能依然心存疑惑。实际上，关于 $\boldsymbol{\alpha}$ 的求解过程即是我们常说的 SMO 算法，这里简要介绍下。

$$\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}y_i y_j\alpha_i\alpha_j\langle\boldsymbol{x}_i,\boldsymbol{x}_j\rangle \tag{2.1.19}$$

$$s.t.\ 0\leq\alpha_i\leq C, i=1,2,\cdots,n \tag{2.1.20}$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \tag{2.1.21}$$

要解决的是在参数 $\boldsymbol{\alpha}$ 上求 W 的最大值的问题，至于 $\boldsymbol{x}_i$ 和 y_i 都是已知数。其中 C 是一个参数，用于控制目标函数中两项（“寻找间隔最大的超平面”和“保证数据点偏差量最小”）之间的权重。和上文最后的式子对比一下，可以看到唯一的区别就是现在对偶变量 $\boldsymbol{\alpha}$ 多了一个上限 C，关于 C 的具体由来请查看下文第 2.3 节。

要了解这个 SMO 算法是如何推导的，请跳到3.5节。

2.1.3 线性不可分的情况

OK，为过渡到下节2.2节所介绍的核函数，让我们再来看看上述推导过程中得到的一些有趣的形式。首先就是关于我们的超平面，对于一个数据点 $\boldsymbol{x}$ 进行分类，实际上是通过把 $\boldsymbol{x}$ 代入到 $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$ 算出结果然后根据其正负号来进行类别划分的。而前面的推导中我们得到 $\boldsymbol{w} = \Sigma_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i$，因此分类函数为

$$f(\boldsymbol{x}) = \left(\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i\right)^{\mathrm{T}}\boldsymbol{x} + b = \sum_{i=1}^{n}\alpha_i y_i \langle \boldsymbol{x}_i, \boldsymbol{x}\rangle + b \tag{2.1.22}$$

这里的形式的有趣之处在于，对于新点 $\boldsymbol{x}$ 的预测，只需要计算它与训练数据点的内积即可（$\langle\cdot,\cdot\rangle$ 表示向量内积），这一点至关重要，是之后使用核函数进行非线性推广的基本前提。此外，所谓“支持向量”也在这里显示出来——事实上，所有非支持向量所对应的系数 $\boldsymbol{\alpha}$ 都是等于零的，因此对于新点的内积计算实际上只要针对少量的“支持向量”而不是所有的训练数据即可。

为什么非支持向量对应的 $\boldsymbol{\alpha}$ 等于零呢？直观上来理解的话，就是这些“后方”的点——正如我们之前分析过的一样，对超平面是没有影响的，由于分类完全有超平面决定，所以这些无关的点并不会参与分类问题的计算，因而也就不会产生任何影响了。

回忆一下我们2.1.1节中通过 Lagrange 乘数法得到的优化目标：

$$\max_{\alpha_i \geq 0} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n}\alpha_i\big(y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1\big) \tag{2.1.23}$$

注意到如果 $\boldsymbol{x}_i$ 是支持向量的话，(2.1.23)式中红颜色的部分是等于 0 的（因为支持向量的函数间隔等于 1），而对于非支持向量来说，函数间隔会大于 1，因此红颜色部分是大于零的，而 α_i 又是非负的，为了满足最大化，α_i 必须等于 0。这也就是这些非支持向量的点的局限性。

从1.5节到上述所有这些东西，便得到了一个最大间隔分类器，这就是一个简单的支持向量机。当然，到目前为止，我们的支持向量机还比较弱，只能处理线性可分的情况，不过，在得到了目标函数的对偶形式之后，通过核函数推广到非线性可分的情况就变成了一件非常容易的事情。[2]

2.2 核函数 *Kernel*

2.2.1 特征空间的隐式映射：核函数

在上文中，我们已经了解到了支持向量机处理线性可分的情况，而对于非线性的情况，支持向量机的处理方法是选择一个核函数 $\kappa(\cdot,\cdot)$，通过将数据映射到高维空间，来解决在原始空间中线性不可分的问题。由于核函数的优良品质，这样的非线性扩展在计算量上并没有比原来复杂多少，这一点是非常难得的。当然，这要归功于核方法——除了支持向量机之外，任何将计算表示为数据点的内积的方法，都可以使用核方法进行非线性扩展。

Minsky 和 Papert 早就在 20 世纪 60 年代就已经明确指出线性学习器计算能力有限。为什么呢？因为总体上来讲，现实世界复杂的应用需要有比线性函数更富有表达能力的假设空间，也就是说，目标概念通常不能由给定属性的简单线性函数组合产生，而是应该一般地寻找待研究数据的更为一般化的抽象特征。

而下文我们将具体介绍的核函数则提供了此种问题的解决途径，从下文你将看到，核函数通过把数据映射到高维空间来增加第一节所述的线性学习器的能力，使得线性学习器对偶空间的表达方式让分类操作更具灵活性和可操作性。因为训练样例一般是不会独立出现的，它们总是以成对样例的内积形式出现，而用对偶形式表示学习器的优势在为在该表示中可调参数的个数不依赖输入属性的个数，通过使用恰当的核函数来替代内积，可以隐式得将非线性的训练数据映射到高维空间，而不增加可调参数的个数（当然，前提是核函数能够计算对应着两个输入特征向量的内积）。

[2] 相信你还记得本节开头所说的：“通过求解对偶问题得到最优解，这就是线性可分条件下支持向量机的对偶算法，这样做的优点在于：一者对偶问题往往更容易求解；二者可以自然的引入核函数，进而推广到非线性分类问题”。

简而言之：在线性不可分的情况下，支持向量机通过某种事先选择的非线性映射（核函数）将输入变量映射到一个高维特征空间，在这个空间中构造最优分类超平面2.1。我们使用支持向量机进行数据集分类工作的过程首先是同预先选定的一些非线性映射将输入空间映射到高维特征空间。

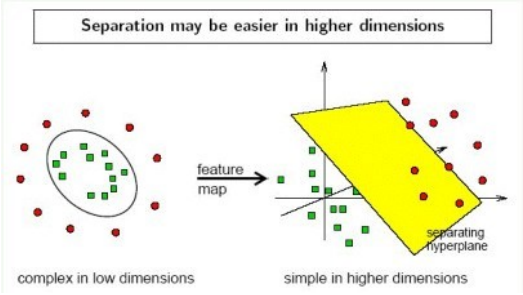

图 2.1: 二维平面上线性不可分的数据，通过映射到更高维的特征空间中，变得线性可分了

在高维特征空间中对训练数据通过超平面进行分类，避免了直接在原输入空间中寻找非线性函数来进行分类。支持向量机的分类函数具有这样的性质：它是一组以支持向量为参数的非线性函数的线性组合，因此分类函数的表达式仅和支持向量的数量有关，而独立于空间的维度，在处理高维输入空间的分类时，这种方法尤其有效，其工作原理如图2.2所示。

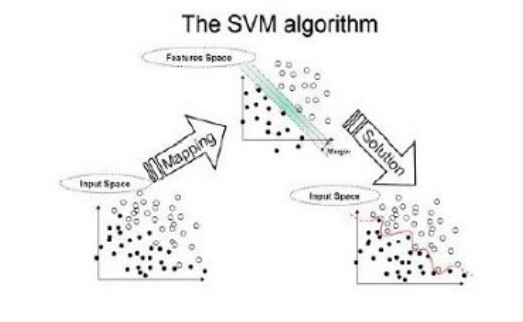

图 2.2: 支持向量机

具体点说：在我们遇到核函数之前，如果用原始的方法，那么在用线性学习器学习一个非线性关系，需要选择一个非线性特征集，并且将数据写成新的表达形式，这等价于应用一个固定的非线性映射，将数据映射到特征空间，在特征空间中使用线性学习器，因此，考虑的假设集是这种类型的函数：

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \phi_i(\boldsymbol{x}) + b \tag{2.2.1}$$

其中 $\phi : \mathcal{X} \to \mathcal{F}$ 是从输入空间到某个特征空间的映射，这意味着建立非线性学习器分为两步：

1. 首先使用一个非线性映射将数据变换到一个特征空间 $\mathcal{F}$；
2. 然后在特征空间使用线性学习器分类。

在上文我提到过对偶形式，而这个对偶形式就是线性学习器的一个重要性质，这意味着假设可以表达为训练点的线性组合，因此决策规则可以用测试点和训练点的内积来表示：

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle + b \tag{2.2.2}$$

如果有一种方式可以在特征空间中直接计算内积$\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle$，就像在原始输入点的函数中一样，就有可能将两个步骤融合到一起建立一个非线性的学习器，这样直接计算法的方法称为核函数方法，于是，核函数便横空出世了。

这里我直接给出一个定义：核是一个函数 κ，对所有 $\boldsymbol{x}, \boldsymbol{z} \in \mathcal{X}$，满足 $\kappa(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle$，这里 $\phi(\cdot)$ 是从原始输入空间 $\mathcal{X}$ 到内积特征空间 $\mathcal{F}$ 的映射。

总而言之，举个简单直接点的例子，如@Wind所说：如果不是用核技术，就会先计算线性映射 $\phi(\boldsymbol{x}_1)$ 和 $\phi(\boldsymbol{x}_2)$,然后计算这两个特征的内积，使用了核技术之后，先把 $\phi(\boldsymbol{x}_1)$ 和 $\phi(\boldsymbol{x}_2)$ 的一般表达式 $\langle\phi(\boldsymbol{x}_1),\phi(\boldsymbol{x}_2)\rangle = \kappa(\langle\boldsymbol{x}_1,\boldsymbol{x}_2\rangle)$ 计算出来，这里的 $\langle\cdot,\cdot\rangle$ 表示内积，$\kappa(\cdot,\cdot)$ 就是对应的核函数，这个表达往往非常简单，所以计算非常方便。

OK，接下来，咱们就进一步从外到里，来探探这个核函数的真面目。

2.2.2 如何处理非线性数据

在2.1节中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。举个例子来说，如图2.3所示的两类数据，分别分布为两个圆圈的形状，这样的数据本身就是线性不可分的，此时咱们该如何把这两类数据分开呢？

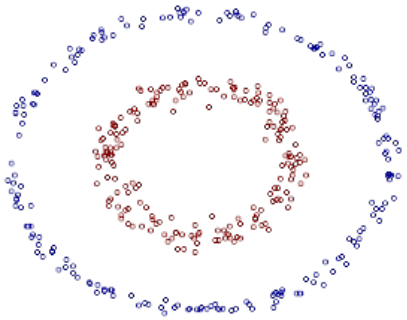

图 2.3: 包含两个不同类别的点的数据

事实上，上图所述的这个数据集，是用两个半径不同的圆圈加上了少量的噪音生成得到的，所以，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 X_1 和 X_2 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：

$$a_1X_1 + a_2X_1^2 + a_3X_2 + a_4X_2^2 + a_5X_1X_2 + a_6 = 0 \tag{2.2.3}$$

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 $Z_1 = X_1, X_2, = X_1^2, Z_3 = X_2, Z_4 = X_2^2, Z_5 = X_1X_2$，那么显然(2.2.3)式在新的坐标系下可以写作

$$\sum_{i=1}^{5} a_iZ_i + a_6 = 0 \tag{2.2.4}$$

关于新的坐标 $Z_1, Z_2, \cdots, Z_5$，这正是一个超平面的方程！也就是说，如果我们做一个映射 $\phi : \mathbb{R}^2 \to \mathbb{R}^5$，将 X_1, X_2 按照上面的规则映射为 $Z_1, Z_2, \cdots, Z_5$，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。这正是核方法处理非线性问题的基本思想。

在进一步描述核方法的细节之前，不妨再来看看这个例子映射过后的直观例子。当然，你我可能无法把 5 维空间画出来，不过由于我这里生成数据的时候就是用了特殊的情形，具体来说，我这里的超平面实际的方程是这个样子（圆心在 X_2 轴上的一个正圆）：

$$\text{<NOT AVAILABLE>} \tag{2.2.5}$$

因此我只需要把它映射到 $Z_1 = X_2^1, Z_2 = X_2^2, Z_3 = X_2$ 这样一个三维空间中即可，图2.4即是映射之后的结果[3]，将坐标轴经过适当的旋转，就可以很明显地看出，数据是可以通过一个平面来分开的。

现在让我们再回到支持向量机中的情形，假设原始的数据是非线性可分的，我们通过一个映射 $\phi(\cdot)$ 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。当然，推导过程也并不是可以简单地直接类比的，例如，原本我们要求超平面的法向量 $\boldsymbol{w}$，但是如果映射之后得到的新空间的维度是无穷维的[4]，要表示一个无穷维的向量描述

[3]下面的 gif 动画，先用 Matlab 画出一张张图片，再用 Imagemagick 拼贴成。

[4]确实会出现这样的情况，比如后面会提到的高斯核函数 *Gaussian Kernel*。

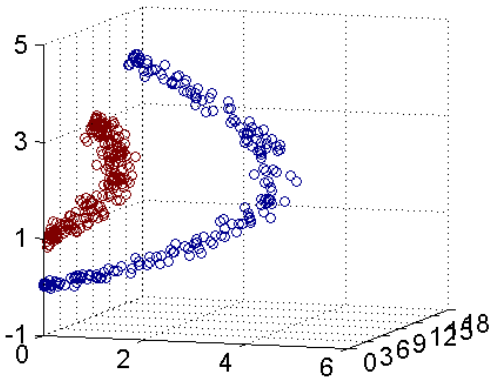

图 2.4: 数据映射之后的结果

起来就比较麻烦。于是我们不妨先忽略过这些细节，直接从最终的结论来分析，回忆一下，我们上一次2.1节中得到的最终的分类函数是这样的：

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + b \tag{2.2.6}$$

现在则是在映射过后的空间，即：

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle + b \tag{2.2.7}$$

而其中的 $\boldsymbol{\alpha}$ 也是通过求解如下的对偶问题而得到的

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \alpha_i \alpha_j y_i y_j \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle \tag{2.2.8}$$

$$s.t.\ \alpha_i \geq 0, i = 1, 2, \cdots, n \tag{2.2.9}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.2.10}$$

这样一来问题就解决了吗？似乎是的：拿到非线性可分的数据，就找一个映射 $\phi(\cdot)$，然后一股脑把原来的数据映射到新空间中，再按照线性可分情况下支持向量机的求解方法来做即可。不过事实上没有这么简单！其实刚才的方法稍想一下就会发现有问题：在最初的例子里，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。所以就需要核函数出马了。

不妨还是从最开始的简单例子出发，设两个向量 $\boldsymbol{x}_1 = (\eta_1, \eta_2)$ 和 $\boldsymbol{x}_2 = (\xi_1, \xi_2)$，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：

$$\langle \phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2) \rangle = \eta_1 \xi_1 + \eta_1^2 \xi_1^2 + \eta_2 \xi_2 + \eta_2^2 \xi_2^2 + \eta_1 \eta_2 \xi_1 \xi_2 \tag{2.2.11}$$

另外，我们又注意到

$$(\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle + 1)^2 = 2\eta_1 \xi_1 + \eta_1^2 \xi_1^2 + 2\eta_2 \xi_2 + \eta_2^2 \xi_2^2 + 2\eta_1 \eta_2 \xi_1 \xi_2 + 1 \tag{2.2.12}$$

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

$$\varphi(X_1, X_2) = \left(\sqrt{2} X_1, X_1^2, \sqrt{2} X_2, X_2^2, \sqrt{2} X_1 X_2, 1 \right) \tag{2.2.13}$$

之后的内积 $\langle \varphi(\boldsymbol{x}_1), \varphi(\boldsymbol{x}_2) \rangle$ 的结果是相同的，那么区别在于什么地方呢？ [5]

[5] 公式说明：上面之中，最后的两个式子，第一个算式，是带内积的完全平方式，可以拆开，然后，通过凑一个得到，第二个算式，也是根据第一个算式凑出来的。

1. 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；
2. 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隐式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：

$$\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle + 1)^2 \tag{2.2.14}$$

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

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}) + b \tag{2.2.15}$$

其中 $\boldsymbol{\alpha}$ 由如下的对偶问题求解而得。

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \alpha_i \alpha_j y_i y_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{2.2.16}$$

$$s.t.\ \alpha_i \geq 0, i = 1, 2, \cdots, n \tag{2.2.17}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.2.18}$$

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！当然，因为我们这里的例子非常简单，所以我可以手工构造出对应于 $\varphi(\cdot)$ 的核函数出来，如果对于任意一个映射，想要构造出对应的核函数就很困难了。

2.2.3 几个核函数

通常人们会从一些常用的核函数中选择；根据问题和数据的不同，选择不同的参数，实际上就是得到了不同的核函数。

多项式核

$$\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle + R)^d \tag{2.2.19}$$

显然刚才我们举的例子是这里多项式核的一个特例（$R = 1, d = 2$）。虽然比较麻烦，而且没有必要，不过这个核所对应的映射实际上是可以写出来的，该空间的维度是 $\binom{m+d}{d}$，其中 m 是原始空间的维度。

高斯核

$$\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp\left\{ -\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}{2\sigma^2} \right\} \tag{2.2.20}$$

这个核就是最开始提到过的会将原始空间映射为无穷维空间的那个家伙。不过，如果 σ 选得很大的话，高次特征上的权重实际上衰减得非常快，所以实际上（数值上近似一下）相当于一个低维的子空间；反过来，如果 σ 选得很小，则可以将任意的数据映射为线性可分——当然，这并不一定是好事，因为随之而来的可能是非常严重的过拟合问题。不过，总的来说，通过调控参数 σ，高斯核实际上具有相当高的灵活性，也是使用最广泛的核函数之一。图2.5所示的例子便是把低维线性不可分的数据通过高斯核函数映射到了高维空间。

线性核

$$\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) = \langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle \tag{2.2.21}$$

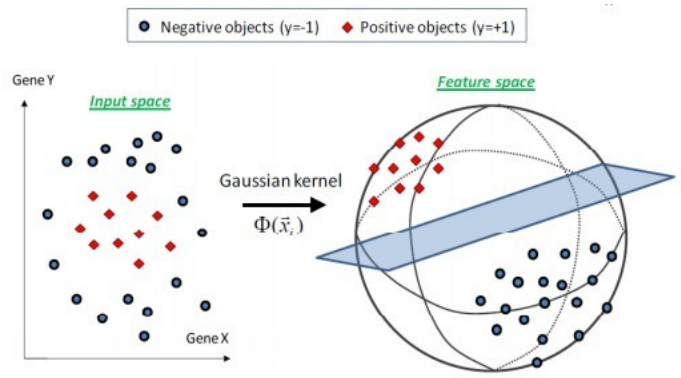

图 2.5: 应用高斯核函数的例子

这实际上就是原始空间中的内积。这个核存在的主要目的是使得“映射后空间中的问题”和“映射前空间中的问题”两者在形式上统一起来了。意思是说，咱们有的时候，写代码，或写公式的时候，只要写个模板或通用表达式，然后再代入不同的核，便可以了，于此，便在形式上统一了起来，不用再分别写一个线性的，和一个非线性的。

2.2.4 核函数的本质

上面说了这么一大堆，读者可能还是没明白核函数到底是个什么东西？我再简要概括下，即以下三点：

1. 实际中，我们会经常遇到线性不可分的样例，此时，我们的常用做法是把样例特征映射到高维空间中去（如图2.1所示，映射到高维空间后，相关特征便被分开了，也就达到了分类的目的）；
2. 但进一步，如果凡是遇到线性不可分的样例，一律映射到高维空间，那么这个维度大小是会高到可怕的（如上文中 19 维乃至无穷维的例子），那咋办呢？
3. 此时，核函数就隆重登场了，核函数的价值在于它虽然也是讲特征进行从低维到高维的转换，但核函数绝就绝在它事先在低维上进行计算，而将实质上的分类效果表现在了高维上，也就如上文所说的避免了直接在高维空间中的复杂计算。

经过前面内容的讲解，我们已经知道，当把内积 $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ 变成 $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$ 之后，求 $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$ 有两种方法。

第一种是先找到这种映射，然后将输入空间中的样本映射到新的空间中，最后在新空间中去求内积 $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$。以多项式 $x_1 + x_2 + x_1^2 + x_2^2 + c = 0$ 为例，对其进行变换，$c_1 = x_1, c_2 = x_2, c_3 = x_1^2, c_4 = x_2^2$，得到：$c_1 + c_2 + c_3 + c_4 + c = 0$，也就是说通过把输入空间从二维向四维映射后，样本由线性不可分变成了线性可分，但是这种转化带来的直接问题是维度变高了，这意味着，首先可能导致后续计算变复杂，其次可能出现维数灾难[6]，对于学习器而言就是：特征空间维数可能最终无法计算，而它的泛化能力（学习器对训练样本以外数据的适应性）会随着维度的增长而大大降低，这也违反了“Occam 剃刀”原则，最终可能会使得内积 $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$ 无法求出，于是也就失去了这种转化的优势了。

第二种是找到某种方法，它不需要显式地将输入空间中的样本映射到新的空间中而能够在输入空间中直接计算出内积 $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$。它其实是对输入空间向高维空间的一种隐式映射，它不需要显式地给出那个映射，在输入空间就可以计算 $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$，这就是核函数方法。

最后引用一个例子[7]举例说明下核函数解决非线性问题的直观效果。

假设现在你是一个农场主，圈养了一批羊群，但为预防狼群袭击羊群，你需要搭建一个篱笆来把羊群围起来。但是篱笆应该建在哪里呢？你很可能需要依据牛群和狼群的位置建立一个“分类器”，比较图2.6这几种不同的分类器，我们可以看到支持向量机完成了一个很完美的解决方案。这个例子从侧面简单说明了支持向量机使

[6] Curse of Dimensionality，统计中也译为“维数祸根”。
[7] http://www.yaksis.com/posts/why-use-svm.html

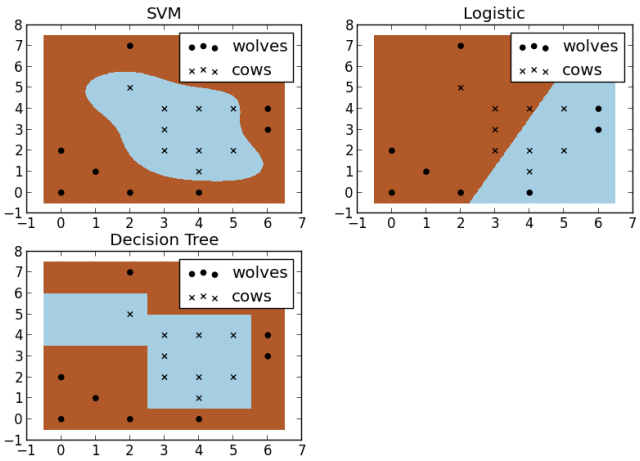

图 2.6: 用不同分类器构建“篱笆”的结果

用非线性分类器的优势，而 Logistic 回归以及决策树都是使用了直线方法。[8]

2.3 使用松弛变量处理离群点的方法

在本文第一节最开始讨论支持向量机的时候，我们就假定，数据是线性可分的，亦即我们可以找到一个可行的超平面将数据完全分开。后来为了处理非线性数据，在2.2节使用核函数方法对原来的线性支持向量机进行了推广，使得非线性的的情况也能处理。虽然通过映射 $\phi(\cdot)$ 将原始数据映射到高维空间之后，能够线性分隔的概率大大增加，但是对于某些情况还是很难处理。

例如可能并不是因为数据本身是非线性结构的，而只是因为数据有噪音。对于这种偏离正常位置很远的数据点，我们称之为离群点 *Outlier* ，在我们原来的支持向量机模型里，离群点的存在有可能造成很大的影响，因为超平面本身就是只有少数几个支持向量组成的，如果这些支持向量里又存在离群点的话，其影响就很大了。

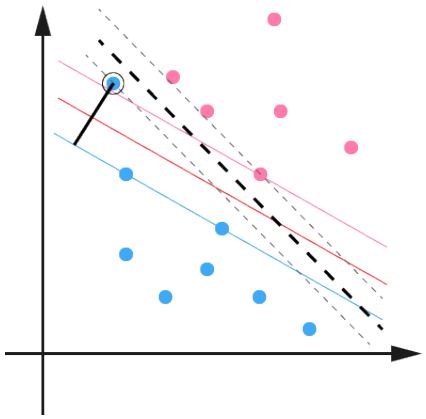

图 2.7: 数据中存在离群点的情况

如图2.7所示，用黑圈圈起来的那个蓝点是一个离群点，它偏离了自己原本所应该在的那个半空间，如果直接忽略掉它的话，原来的分隔超平面还是挺好的，但是由于这个离群点的出现，导致分隔超平面不得不被挤歪了，变成途中黑色虚线所示（这只是一个示意图，并没有严格计算精确坐标），同时间隔也相应变小了。当然，更严重的情况是，如果这个离群点再往右上移动一些距离的话，我们将无法构造出能将数据分开的超平面来。

为了处理这种情况，支持向量机允许数据点在一定程度上偏离一下超平面。例如图2.7中，黑色实线所对应

[8] 对核函数有进一步兴趣的，还可以参看：http://www.cnblogs.com/vivounicorn/archive/2010/12/13/1904720.html。

的距离，就是该离群点偏离的距离，如果把它移动回来，就刚好落在原来的超平面上，而不会使得超平面发生变形了。

插播下一位读者@Copper_PKU的理解：换言之，在有松弛的情况下，离群点也属于支持向量，同时，对于不同的支持向量，Lagrange 参数的值也不同，如此篇论文“Large Scale Machine Learning”中图所示（图2.8），对于远离分类平面的点值为 0；对于边缘上的点值在 $[0, \frac{1}{L}]$ 之间，其中，L 为训练数据集个数，即数据集大小；对于离群点和内部的数据值为 $\frac{1}{L}$。更多请参看本文文末参考资料第 51 条。

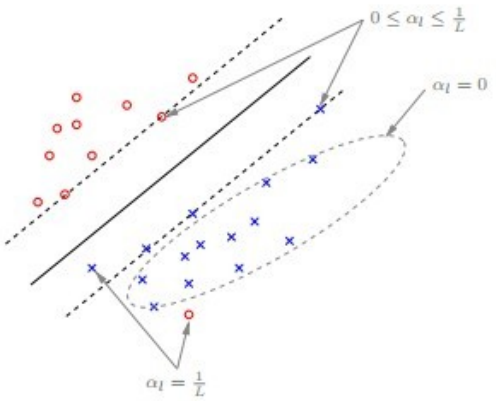

图 2.8: 离群点也属于支持向量

OK，继续回到咱们的问题。我们，原来的约束条件为：

$$y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \ge 1,\ i = 1, 2, \cdots, n \tag{2.3.1}$$

现在考虑到离群点，约束条件变成了

$$y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \ge 1 - \xi_i,\ i = 1, 2, \cdots, n \tag{2.3.2}$$

其中 $\xi_i\,(i = 1, 2, \cdots, n)$ 称为松弛变量 *Slack Variable*，对应数据点 $\boldsymbol{x}_i$ 允许偏离的函数间隔的量。当然，如果我们允许 ξ_i 任意大的话，那任意的超平面都是符合条件的了。所以，我们在原来的目标函数后面加上一项，使得这些 ξ_i 的总和也要最小，新的优化目标为

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{2.3.3}$$

$$s.t.\ y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \ge 1 - \xi_i,\ i = 1, 2, \cdots, n \tag{2.3.4}$$

$$\xi_i \ge 0,\ i = 1, 2, \cdots, n \tag{2.3.5}$$

其中 C 是一个参数，用于控制目标函数中两项（“寻找间隔最大的超平面”和“保证数据点偏差量最小”）之间的权重。注意，其中 $\boldsymbol{\xi}$ 是需要优化的变量（之一），而 C 是一个事先确定好的常量。

和前面类似，采用 Lagrange 乘数法进行求解，可以写出

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{r}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left(y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1 + \xi_i\right) - \sum_{i=1}^{n} r_i\xi_i \tag{2.3.6}$$

求偏导之后令偏导数为 0 可以分别得到

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i \tag{2.3.7}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n}\alpha_i y_i = 0 \tag{2.3.8}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \Rightarrow C - \alpha_i - r_i = 0,\ i = 1, 2, \cdots, n \tag{2.3.9}$$

将 $\boldsymbol{w}$ 回代 $\mathcal{L}$ 并化简，即得到和原来一样的目标函数

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \tag{2.3.10}$$

不过，由于我们得到 $C - \alpha_i - r_i = 0$ 而又有 $r_i \geq 0$（作为 Lagrange 乘数法的条件），因此有 $\alpha_i \leq C$，所以整个对偶问题现在写作：

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \tag{2.3.11}$$

$$s.t.\ 0 \leq \alpha_i \leq C, i = 1, 2, \cdots, n \tag{2.3.12}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.3.13}$$

图2.9列出了原问题和对偶问题进行对比[9]。

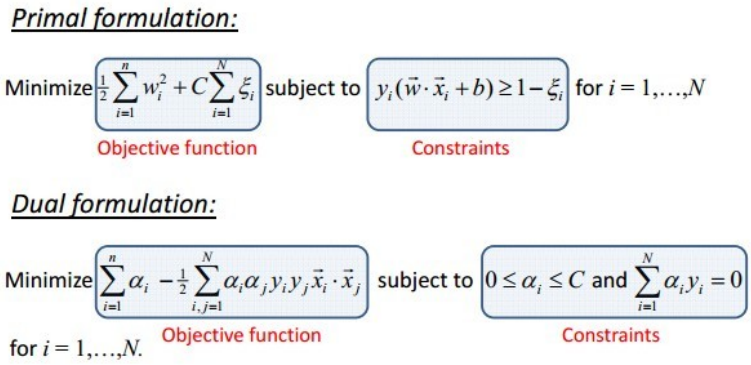

图 2.9: 引入松弛变量之后的原问题和对偶问题

可以看到唯一的区别就是现在对偶变量 $\boldsymbol{\alpha}$ 多了一个上限 C。而核化的非线性形式也是一样的，只要把 $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ 换成 $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ 即可。这样一来，一个完整的，可以处理线性和非线性并能容忍噪音和离群点的支持向量机才终于介绍完毕了。

行文至此，可以做个小结，不准确的说，支持向量机它本质上即是一个分类方法，用 $\boldsymbol{w}^{\mathrm{T}} + b$ 定义分类函数，于是求 $\boldsymbol{w}$ 和 b，为寻最大间隔，引出 $\frac{1}{2}\|\boldsymbol{w}\|^2$，继而引入 Lagrange 乘子，化为对 $\boldsymbol{\alpha}$ 的求解（求解过程中会涉及到一系列最优化或凸二次规划等问题），如此，求求 $\boldsymbol{w}$ 和 b 与求 $\boldsymbol{\alpha}$ 等价，而 $\boldsymbol{\alpha}$ 的求解可以用一种快速学习算法 SMO，至于核函数，是为处理非线性可分的情况，若直接映射到高维计算可能出现维数灾难问题，故在低维计算，等效高维表现。

OK，理解到这第二层，已经能满足绝大部分人一窥支持向量机原理的好奇心，然对于那些想在证明层面理解支持向量机的则还很不够，但进入第三层理解境界之前，你必须要有比较好的数理基础和逻辑证明能力，不然你会跟我一样，吃不少苦头的。

[9]修正：图中的 Dual formulation 中的 Minimize 应为 Maxmize。

第 3 层　证明 SVM

说实话，凡是涉及到要证明的东西、理论，便一般不是怎么好惹的东西。绝大部分时候，看懂一个东西不难，但证明一个东西则需要点数学功底，进一步，证明一个东西也不是特别难，难的是从零开始发明创造这个东西的时候，则显艰难，因为任何时代，大部分人的研究所得都不过是基于前人的研究成果，前人所做的是开创性工作，而这往往是最艰难最有价值的，他们被称为真正的先驱。牛顿也曾说过，他不过是站在巨人的肩上。你，我，更是如此。

正如陈希孺院士在他的著作《数理统计学简史》的第 4 章最小二乘法中所讲：在科研上诸多观念的革新和突破是有着很多的不易的，或许某个定理在某个时期由某个人点破了，现在的我们看来一切都是理所当然，但在一切没有发现之前，可能许许多多的顶级学者毕其功于一役，耗尽一生，努力了几十年最终也是无功而返。

话休絮烦，要证明一个东西先要弄清楚它的根基在哪，即构成它的基础是哪些理论。OK，以下内容基本是上文中未讲到的一些定理的证明，包括其背后的逻辑、来源背景等东西，还是读书笔记。

本部分导览

- 3.1节线性学习器中，主要阐述感知机算法
- 3.2节非线性学习器中，主要阐述 Mercer 定理
- 3.3节主要讨论损失函数
- 3.4节主要介绍最小二乘法
- 3.5节主要介绍 SMO 算法
- 3.6节简略谈了支持向量机的应用

3.1　线性学习器

3.1.1　感知机

这个感知机算法（算法1）是 1956 年提出的，年代久远，依然影响着当今，当然，可以肯定的是，此算法亦非最优，后续会有更详尽阐述。不过，有一点，你必须清楚，这个算法是为了干嘛的：不断的训练试错以期寻找一个合适的超平面（是的，就这么简单）。

下面，举个例子。如图3.1所示，凭我们的直觉可以看出，图中的红线是最优超平面，蓝线则是根据感知机算法在不断的训练中，最终，若蓝线能通过不断的训练移动到红线位置上，则代表训练成功。

既然需要通过不断的训练以让蓝线最终成为最优分类超平面，那么，到底需要训练多少次呢？Novikoff 定理告诉我们当间隔是正的时候感知机算法会在有限次数的迭代中收敛，也就是说 Novikoff 定理证明了感知机算法的收敛性，即能得到一个界，不至于无穷循环下去。[1]

定理 3.1.1 *Novikoff 定理*

如果分类超平面存在，仅需在序列 S 上迭代几次，在界为 $\left(\frac{2R}{\gamma}\right)^2$ 的错误次数下就可以找到分类超平面，算法停止。这里 $R=\max_{1\leq i\leq l}\|\boldsymbol{x}_i\|$，$\gamma$ 为扩充间隔。

[1] Novikoff 定理的证明请见：http://www.cs.columbia.edu/~mcollins/courses/6998-2012/notes/perc.converge.pdf。

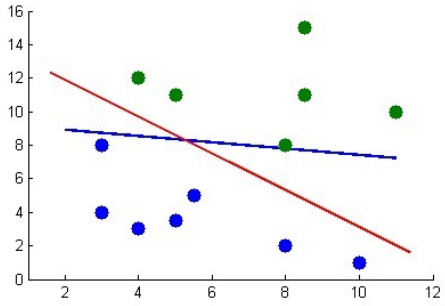

图 3.1: 感知机

根据误分次数公式可知, 迭代次数与对应于扩充（包括偏置）权重的训练集的间隔有关。顺便再解释下这个所谓的扩充间隔 γ，γ 即为样本到分类间隔的距离，即从 γ 引出的最大分类间隔。OK，还记得上文1.4.2节开头的内容么？

在给出几何间隔的定义之前，咱们首先来看下，如图1.4所示，对于一个点 $\boldsymbol{x}$，令其垂直投影到超平面上的对应的为 $\boldsymbol{x}_0$，由于 $\boldsymbol{w}$ 是垂直于超平面的一个向量，γ 为样本 $\boldsymbol{x}$ 到分类间隔的距离，我们有

$$\boldsymbol{x} = \boldsymbol{x}_0 + \gamma \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \tag{3.1.1}$$

然后后续怎么推导出最大分类间隔请回到本文第一、二部分，此处不重复板书。

同时有一点得注意：感知机算法虽然可以通过简单迭代对线性可分数据生成正确分类的超平面，但不是最优效果，那怎样才能得到最优效果呢，就是上文中第一部分所讲的寻找最大分类间隔超平面。

算法 1 感知机算法（原始形式）

输入： 线性可分的数据集 S，学习率 $\eta \in \mathbb{R}^+$

$\boldsymbol{w}_0 \leftarrow 0; b_0 \leftarrow 0; k \leftarrow 0$
$R \leftarrow \max_{1 \le i \le l} \|\boldsymbol{x}_i\|$
repeat
  for $i = 1$ to l **do**
    if $y_i\left(\langle \boldsymbol{w}_k, \boldsymbol{x}_i \rangle + b_k\right) \le 0$ **then**
      $\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k + \eta y_i \boldsymbol{x}_i$
      $b_{k+1} \leftarrow b_k + \eta y_i R^2$
      $k \leftarrow k + 1$
    end if
  end for
until 在 for 循环中没有错误发生
return $(\boldsymbol{w}_k, b_k)$，k 为错误次数

3.2 非线性学习器

3.2.1 Mercer 定理

定理 3.2.1 *Mercer 定理*

如果函数 κ 是 $\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ 上的映射。那么如果 κ 是一个有效核函数（也称为 *Mercer* 核函数），那么当且仅当对于训练样例 $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$，其相应的核函数矩阵是对称半正定的。

要理解这个 Mercer 定理，先要了解什么是半正定矩阵，要了解什么是半正定矩阵，先得知道什么是正定矩阵。[2] [3] [4]

[2] 矩阵理论“博大精深”，我自己也未能彻底理清，等我理清了再续写此节，顺便推荐我正在看的一本《矩阵分析与应用》。
[3] Mercer 定理的证明：http://ftp136343.host106.web522.com/a/biancheng/matlab/2013/0120/648.html
[4] 一个 Tutorial：http://www.cs.berkeley.edu/~bartlett/courses/281b-sp08/7.pdf

3.3 损失函数

在本文1.1节有这么一句话“支持向量机（SVM）是 90 年代中期发展起来的基于统计学习理论的一种机器学习方法，通过寻求结构化风险最小来提高学习机泛化能力，实现经验风险和置信范围的最小化，从而达到在统计样本量较少的情况下，亦能获得良好统计规律的目的。”但初次看到的读者可能并不了解什么是结构化风险，什么又是经验风险。要了解这两个所谓的“风险”，还得又从监督学习说起。

监督学习实际上就是一个经验风险或者结构风险函数的最优化问题。风险函数度量平均意义下模型预测的好坏，模型每一次预测的好坏用损失函数来度量。它从假设空间 $\mathcal{F}$ 中选择模型 f 作为决策函数，对于给定的输入 X，由 $f(X)$ 给出相应的输出 Y，这个输出的预测值 $f(X)$ 与真实值 Y 可能一致也可能不一致，用一个损失函数来度量预测错误的程度。损失函数记为 $L(Y, f(X))$。

常用的损失函数有以下几种（以下基本引用自《统计学习方法》）：

(1) 0-1 损失函数

$$L(Y, f(X)) = \begin{cases} 1, & Y \neq f(X) \\ 0, & Y = f(X) \end{cases} \tag{3.3.1}$$

(2) 平方损失函数

$$L(Y, f(X)) = (Y - f(X))^2 \tag{3.3.2}$$

(3) 绝对损失函数

$$L(Y, f(X)) = |Y - f(X)| \tag{3.3.3}$$

(4) 对数损失函数

$$L(Y, f(X)) = -\log P(Y \mid X) \tag{3.3.4}$$

给定一个训练数据集

$$T = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_N, y_N)\} \tag{3.3.5}$$

模型 $f(X)$ 关于训练数据集的平均损失成为经验风险，如下：

$$R_{\text{emp}}(f) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(\boldsymbol{x}_i)) \tag{3.3.6}$$

关于如何选择模型，监督学习有两种策略：经验风险最小化和结构风险最小化。

经验风险最小化策略认为，经验风险最小的模型就是最优的模型，则按照经验风险最小化求最优模型就是求解如下最优化问题：

$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(\boldsymbol{x}_i)) \tag{3.3.7}$$

当样本容量很小时，经验风险最小化的策略容易产生过拟合的现象。结构风险最小化可以防止过拟合。结构风险是在经验风险的基础上加上表示模型复杂度的正则化项或罚项，结构风险定义如下：

$$R_{\text{snn}}(f) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(\boldsymbol{x}_i)) + \lambda J(f) \tag{3.3.8}$$

其中 $J(f)$ 为模型的复杂度，模型 f 越复杂，$J(f)$ 值就越大，模型越简单，$J(f)$ 就越小，也就是说 $J(f)$ 是对复杂模型的惩罚。$\lambda \geq 0$ 是系数，用以权衡经验风险和模型复杂度。结构风

险最小化的策略认为结构风险最小的模型是最优的模型，所以求最优的模型就是求解下面的优化问题：

$$\min_{f\in\mathcal{F}} \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(\boldsymbol{x}_i)) + \lambda J(f) \tag{3.3.9}$$

这样，监督学习问题就变成了经验风险或结构风险函数的最优化问题，如式(3.3.7)和式(3.3.9)。

如此，SVM 有第二种理解，即最优化 + 损失最小，或如@ 夏粉 _ 百度所说“可从损失函数和优化算法角度看 *SVM*、*Boosting*、*LR* 等算法，可能会有不同收获”。[5]

关于损失函数，如下文读者评论中所述：可以看看张潼老师的这篇 *Statistical Behavior and Consistency of Classification Methods Based on Convex Risk Minimization*，各种算法中常用的损失函数基本都具有 Fisher 一致性，优化这些损失函数得到的分类器可以看作是后验概率的“代理”。此外，张潼老师还有另外一篇论文 *Statistical Analysis of Some Multi-Category Large Margin Classification Methods*，对多分类情况下的 Margin Loss 进行了分析，这两篇对 Boosting 和 SVM 使用的损失函数分析的很透彻。

3.4 最小二乘法

3.4.1 什么是最小二乘法

既然本节开始之前提到了最小二乘法，那么下面引用《正态分布的前世今生》里的内容稍微简单阐述下。

我们口头中经常说：一般来说，平均来说。如平均来说，不吸烟的健康优于吸烟者，之所以要加“平均”二字，是因为凡事皆有例外，总存在某个特别的人他吸烟但由于经常锻炼所以他的健康状况可能会优于他身边不吸烟的朋友。而最小二乘法的一个最简单的例子便是算术平均。

最小二乘法（又称最小平方法）是一种数学优化技术。它通过最小化误差的平方和寻找数据的最佳函数匹配。利用最小二乘法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小。用函数表示为：

$$\min \sum_{i=1}^{n} (y - y_i)^2 \tag{3.4.1}$$

使误差平方和达到最小以寻求估计值的方法，就叫做最小二乘法，用最小二乘法得到的估计，叫做最小二乘估计。当然，取平方和作为目标函数只是众多可取的方法之一。

最小二乘法是 Legendre 在 1806 年发表的，基本思想就是认为测量中有误差，我们求解出导致累积误差最小的参数即可。

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} e_i^2 \\ &= \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_{1i} + \cdots + \beta_p x_{pi})]^2\end{aligned} \tag{3.4.2}$$

Legendre 在论文中对最小二乘法的优良性做了几点说明：

- 最小二乘使得误差平方和最小，并在各个方程的误差之间建立了一种平衡，从而防止某一个极端误差取得支配地位
- 计算中只要求偏导后求解线性方程组，计算过程明确便捷
- 最小二乘可以导出算术平均值作为估计值

[5]关于更多统计学习方法的问题，请参看：http://blog.csdn.net/qll125596718/article/details/8351337

对于最后一点，从统计学的角度来看是很重要的一个性质。推理如下：假设真值为 θ，$x_1, x_2, \cdots, x_n$ 为 n 次测量值，每次测量的误差为 $e_i = x_i - \theta$，按最小二乘法，误差累积为

$$L(\theta) = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (x_i - \theta)^2 \tag{3.4.3}$$

求解 θ 使 $L(\theta)$ 达到最小，正好是算术平均 $\bar{x} = \frac{1}{n}\Sigma_{i=1}^{n} x_i$。

由于算术平均是一个历经考验的方法，而以上的推理说明，算术平均是最小二乘的一个特例，所以从另一个角度说明了最小二乘方法的优良性，使我们对最小二乘法更加有信心。

最小二乘法发表之后很快得到了大家的认可接受，并迅速的在数据分析实践中被广泛使用。不过历史上又有人把最小二乘法的发明归功于 Gauss，这又是怎么一回事呢。Gauss 在 1809 年也发表了最小二乘法，并且声称自己已经使用这个方法多年。Gauss 发明了小行星定位的数学方法，并在数据分析中使用最小二乘方法进行计算，准确的预测了谷神星的位置。

说了这么多，貌似跟本文的主题支持向量机没啥关系呀，别急，请让我继续阐述。本质上说，最小二乘法即是一种参数估计方法，说到参数估计，咱们得从一元线性模型说起。

3.4.2 最小二乘法的解法

什么是一元线性模型呢？先来梳理下几个基本概念[6]：

- 监督学习中，如果预测的变量是离散的，我们称其为分类（如决策树，支持向量机等），如果预测的变量是连续的，我们称其为回归。
- 回归分析中，如果只包括一个自变量和一个因变量，且二者的关系可用一条直线近似表示，这种回归分析称为一元线性回归分析。
- 如果回归分析中包括两个或两个以上的自变量，且因变量和自变量之间是线性关系，则称为多元线性回归分析。
- 对于二维空间线性是一条直线；对于三维空间线性是一个平面，对于多维空间线性是一个超平面

对于一元线性回归模型, 假设从总体中获取了 n 组观察值 $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$。对于平面中的这 n 个点，可以使用无数条曲线来拟合。要求样本回归函数尽可能好地拟合这组值。综合起来看，这条直线处于样本数据的中心位置最合理。

选择最佳拟合曲线的标准可以确定为：使总的拟合误差（即总残差）达到最小。有以下三个标准可以选择：

1. 用“残差和最小”确定直线位置是一个途径。但很快发现计算“残差和”存在相互抵消的问题。
2. 用“残差绝对值和最小”确定直线位置也是一个途径。但绝对值的计算比较麻烦。
3. 最小二乘法的原则是以“残差平方和最小”确定直线位置。用最小二乘法除了计算比较方便外，得到的估计量还具有优良特性。这种方法对异常值非常敏感。

最常用的是普通最小二乘法 *Ordinary Least Square, OLS*：所选择的回归模型应该使所有观察值的残差平方和达到最小，即采用平方损失函数。

我们定义样本回归模型为：

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + e_i \;\Rightarrow\; e_i = y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i \tag{3.4.4}$$

其中 e_i 为样本 (x_i, y_i) 的误差。定义平方损失函数：

$$Q = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 \tag{3.4.5}$$

[6] 引用自http://blog.csdn.net/qll125596718/article/details/8248249

则通过 Q 最小确定这条直线，即确定 $\hat{\beta}_0$ 和 $\hat{\beta}_1$，以 $\hat{\beta}_0$ 和 $\hat{\beta}_1$ 为变量，把它们看作是 Q 的函数，就变成了一个求极值的问题，可以通过求导数得到。求 Q 对两个待估参数的偏导数并令其等于 0：

$$\begin{cases} \frac{\partial Q}{\partial \hat{\beta}_0} = 2\sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)\cdot(-1) = 0 \\ \frac{\partial Q}{\partial \hat{\beta}_1} = 2\sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)\cdot(-x_i) = 0 \end{cases} \tag{3.4.6}$$

求解可以得到

$$\hat{\beta}_0 = \frac{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{3.4.7}$$

$$\hat{\beta}_1 = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{3.4.8}$$

这就是最小二乘法的解法，就是求得平方损失函数的极值点。自此，你看到求解最小二乘法与求解 SVM 问题何等相似，尤其是定义损失函数，而后通过偏导求得极值。

OK，更多请参看陈希孺院士的《数理统计学简史》的第 4 章、最小二乘法，和本文参考条目第 59 条《凸函数》。

3.5 SMO 算法

在上文2.1.2节中，我们提到了求解对偶问题的序列最小最优化算法，但并未提到其具体解法。

事实上，SMO 算法是由 Microsoft Research 的 John C. Platt 在 1998 年发表的一篇论文 *Sequential Minimal Optimization A Fast Algorithm for Training Support Vector Machines* 中提出，它很快成为最快的二次规划优化算法，特别针对线性 SVM 和数据稀疏时性能更优。

接下来，咱们便参考 John C. Platt 的这篇文章[7]来看看 SMO 的解法是怎样的。

3.5.1 SMO 算法的解法

咱们首先来定义特征到结果的输出函数为

$$u = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} - b \tag{3.5.1}$$

再三强调，这个 u 与我们之前定义的 $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$ 实质是一样的。接着，咱们重新定义咱们原始的优化问题，权当重新回顾：

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{3.5.2}$$

$$s.t.\ y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i - b) \geq 1,\ i = 1, 2, \cdots, n \tag{3.5.3}$$

类似地，采用 Lagrange 乘数法可以得到

$$\boldsymbol{w} = \sum_{i=1}^{n} y_i \alpha_i \boldsymbol{x}_i \tag{3.5.4}$$

$$b = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_k - y_k \quad （对某个\ \alpha_k > 0） \tag{3.5.5}$$

这里的 α_i 还是 Lagrange 乘数法中引入的系数。将(3.5.4)代入(3.5.1)式并引入核函数 $\kappa(\cdot,\cdot)$ 可以得到

$$u = \sum_{i=1}^{n} \alpha_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}) - b \tag{3.5.6}$$

[7] http://research.microsoft.com/en-us/um/people/jplatt/smoTR.pdf

与此同时，和前面类似可以写出

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle - \sum_{i=1}^{n}\alpha_i \tag{3.5.7}$$

$$s.t.\ \alpha_i \geq 0, i = 1, 2, \cdots, n \tag{3.5.8}$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \tag{3.5.9}$$

引入松弛变量之后为

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{3.5.10}$$

$$s.t.\ y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i - b) \geq 1 - \xi_i,\ i = 1, 2, \cdots, n \tag{3.5.11}$$

最终的优化目标为

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n}\alpha_i \tag{3.5.12}$$

$$s.t.\ 0 \geq \alpha_i \geq C, i = 1, 2, \cdots, n \tag{3.5.13}$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \tag{3.5.14}$$

根据 KKT 条件可以得出其中 α_i 取值的意义为：

$$\alpha_i = 0 \ \Leftrightarrow\ y_i u_i \geq 1 \tag{3.5.15}$$

$$0 < \alpha_i < C \ \Leftrightarrow\ y_i u_i = 1 \tag{3.5.16}$$

$$\alpha_i = C \ \Leftrightarrow\ y_i u_i \leq 1 \tag{3.5.17}$$

1. (3.5.15)式表明是正常分类，在边界内部，我们知道正确分类的点 $y_i f(x_i) \geq 0$。
2. (3.5.16)式表明了是支持向量，在边界上。
3. (3.5.17)式表明了是在两条边界之间。

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

1. $y_i u_i \leq 1$ 但是 $\alpha_i < C$，则是不满足的,而原本 $\alpha_i = C$
2. $y_i u_i \geq 1$ 但是 $\alpha_i > 0$，则是不满足的，而原本 $\alpha_i = 0$
3. $y_i u_i = 1$ 但是 $\alpha_i = 0$ 或者 $\alpha_i = C$，则表明不满足的，而原本应该是 $0 < \alpha_i < C$

所以要找出不满足 KKT 条件的这些 α_i，并更新这些 α_i，但这些 α_i 又受到另外一个约束（参见(2.1.13)式）：

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \tag{3.5.18}$$

因此，我们通过另一个方法，即同时更新 α_i 和 α_j，要求满足(3.5.19)式，就能保证和为 0 的约束。

$$\alpha_i^{\text{new}} y_i + \alpha_j^{\text{new}} = \alpha_i^{\text{old}} y_i + \alpha_j^{\text{old}} y_j = \text{常数} \tag{3.5.19}$$

利用(3.5.19)式，消去 α_i，可得到一个关于单变量 α_j 的一个凸二次规划问题，不考虑其约束 $0 \leq \alpha_j \leq C$,可以得其解为

$$\alpha_j^{\text{new}} = \alpha_j^{\text{old}} + \frac{y_j(E_i - E_j)}{\eta} \tag{3.5.20}$$

其中

$$E_i = u_i - y_i \tag{3.5.21}$$

$$E_j = u_j - y_j \tag{3.5.22}$$

$$\eta = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_i) + \kappa(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{3.5.23}$$

然后考虑约束 $0 \leq \alpha_j \leq C$ 可以得到 α_i 的解析解为

$$\alpha_i^{\text{new, clipped}} = \begin{cases} H, & \alpha_i^{\text{new}} \geq H \\ \alpha_i^{\text{new}}, & L < \alpha_i^{\text{new}} < H \\ L, & \alpha_i^{\text{new}} \leq L \end{cases} \tag{3.5.24}$$

把 SMO 中对于两个参数求解过程看成线性规划来理解来理解的话，那么图3.2所表达的便是(3.5.19)式。根据 y_i

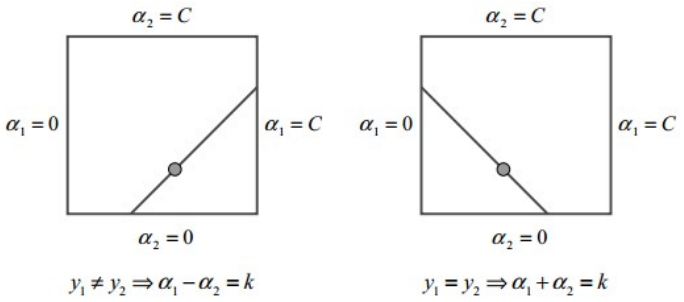

图 3.2: 约束条件

和 y_j 同号或异号，可得出两个上、下界分别为

$$\begin{cases} L = \max\{0, \alpha_j - \alpha_i\} \\ H = \max\{C, C + \alpha_j - \alpha_i\} \end{cases}, \quad y_i \neq y_j \tag{3.5.25}$$

$$\begin{cases} L = \max\{0, \alpha_j + \alpha_i - C\} \\ H = \max\{C, \alpha_j - \alpha_i\} \end{cases}, \quad y_i = y_j \tag{3.5.26}$$

对于 α_i，有

$$\alpha_i^{\text{new}} = \alpha_i + y_i y_j (\alpha_j - \alpha_j^{\text{new, clipped}}) \tag{3.5.27}$$

那么如果和求得 α_i 和 α_j 呢？对于 α_i，可以通过刚刚说的那 3 种不满足 KKT 的条件来找；而对于 α_j，可以通过 $\max |E_i - E_j|$ 求得；而 b 的更新则是

$$b = \begin{cases} b_1, & 0 < \alpha_i < C \\ b_2, & 0 < \alpha_j < C \\ \frac{1}{2}(b_1 + b_2), & \text{其它} \end{cases} \tag{3.5.28}$$

其中

$$b_1 = b - E_i - y_i(\alpha_i - \alpha_i^{\text{old}})\kappa(\boldsymbol{x}_i, \boldsymbol{x}_i) - y_j(\alpha_j - \alpha_j^{\text{old}})\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{3.5.29}$$

$$b_2 = b - E_j - y_i(\alpha_i - \alpha_i^{\text{old}})\kappa(\boldsymbol{x}_i, \boldsymbol{x}_i) - y_j(\alpha_j - \alpha_j^{\text{old}})\kappa(\boldsymbol{x}_j, \boldsymbol{x}_j) \tag{3.5.30}$$

每次更新完 α_i 和 α_j 后，都需要再重新计算 b，及对应的 E_i 和 E_j。

最后更新所有 α_i、y 和 b，这样模型就出来了，从而即可求出咱们开头提出的分类函数。[8]

3.5.2 SMO 算法的步骤

SMO 的主要步骤如图3.3所示。其中迭代的两步是：

(1) 选择接下来要更新的一对 α_i 和 α_j：采用启发式的方法进行选择，以使目标函数最大程度地接近其全局最优值

(2) 将目标函数对 α_i 和 α_j 进行优化，保持其它所有的 $\alpha_k (k \neq i, j)$ 不变

[8]可以参考：http://www.cnblogs.com/jerrylead/archive/2011/03/18/1988419.html。

Repeat till convergence {

1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).
2. Reoptimize $W(\alpha)$ with respect to α_i and α_j, while holding all the other α_k's $(k \neq i, j)$ fixed.

}

图 3.3: SMO 算法的主要步骤

假定在某一次迭代中，需要更新 α_1 和 α_2，那么优化目标可以写成

$$\max_{\boldsymbol{\alpha}} \alpha_1 + \alpha_2 + \sum_{i=3}^{n} \alpha_i - \frac{1}{2} \left\| \alpha_1 y_1 \phi(\boldsymbol{x}_1) + \alpha_2 y_2 \phi(\boldsymbol{x}_2) + \sum_{i=3}^{n} \alpha_i y_i \phi(\boldsymbol{x}_i) \right\|^2 \tag{3.5.31}$$

而更新 α_1 和 α_2 的步骤如下

(1) 根据(3.5.25)式计算上下界 L 和 H

(2) 计算(3.5.31)式中目标函数的二阶导数，注意到(3.5.23)式和(3.5.32)式只差一个符号

$$\eta = 2\phi(\boldsymbol{x}_1)^{\mathrm{T}} \phi(\boldsymbol{x}_2) - \phi(\boldsymbol{x}_1)^{\mathrm{T}} \phi(\boldsymbol{x}_1) - \phi(\boldsymbol{x}_2)^{\mathrm{T}} \phi(\boldsymbol{x}_2) \tag{3.5.32}$$

(3) 根据(3.5.20)式更新 α_2

(4) 根据(3.5.24)和(3.5.27)式更新 α_1

对于每次迭代选择 α_i 和 α_j 的启发式方法，其包括以下两个步骤

(1) 先“扫描”所有乘子，把第一个违反 KKT 条件的作为更新对象，令为 α_j；

(2) 在所有不违反 KKT 条件的乘子中，选择使 $|E_i - E_j|$ 最大的 α_i。

需要注意的是，每次更新完所选的 α_i 和 α_j 后，都需要再重新计算 b、E_i 和 E_j。另外，α_i 和 α_j 的选择务必遵循两个原则：

(1) 满足 KKT 条件

(2) 更新应能最大限度地增大目标函数的值（类似于梯度下降）

综上，SMO 算法的基本思想是将 Vapnik 在 1982 年提出的 Chunking 方法推到极致，SMO 算法每次迭代只选出两个分量 α_i 和 α_j 进行调整，其它分量则保持固定不变，在得到解 α_i 和 α_j 之后，再用 α_i 和 α_j 改进其它分量。与通常的分解算法比较，尽管它可能需要更多的迭代次数，但每次迭代的计算量比较小，所以该算法表现出整理的快速收敛性，且不需要存储核矩阵，也没有矩阵运算。

3.5.3 SMO 算法的实现

行文至此，我相信，对支持向量机理解到了一定程度后，是的确能在脑海里从头至尾推导出相关公式的，最初分类函数，最大化分类间隔，$\max \frac{1}{\|\boldsymbol{w}\|}$，$\min \frac{1}{2} \|\boldsymbol{w}\|^2$，凸二次规划，Lagrange 乘数法，转化为对偶问题，SMO 算法，都为寻找一个最优解，一个最优分类平面。一步步梳理下来，为什么这样那样，太多东西可以追究，最后实现。

至于下文中将阐述的核函数则为是为了更好的处理非线性可分的情况，而松弛变量则是为了纠正或约束少量“不安分”或脱离集体不好归类的因子。

台湾的林智仁教授写了一个封装 SVM 算法的 libsvm 库[9]，大家可以看看，此外这里还有一份 libsvm 的注释文档：http://www.pami.sjtu.edu.cn/people/gpliu/document/libsvm_src.pdf。

[9] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

除了在这篇论文 *Fast Training of Support Vector Machines Using Sequential Minimal Optimization* 中 Platt 给出了 SMO 算法的逻辑代码之外，这里也有一份 SMO 的实现代码：http://blog.csdn.net/techq/article/details/6171688，大家可以看下。

其余更多请参看文末参考文献和推荐阅读中的条目 6《支持向量机——算法、理论和扩展》和条目 11《统计学习方法》的相关章节。

3.6 支持向量机的应用

或许我们已经听到过，支持向量机在很多诸如文本分类、图像分类、生物序列分析和生物数据挖掘、手写字符识别等领域有很多的应用，但或许你并没强烈的意识到，支持向量机可以成功应用的领域远远超出现在已经在开发应用了的领域。

3.6.1 文本分类

一个文本分类系统不仅是一个自然语言处理系统，也是一个典型的模式识别系统，系统的输入是需要进行分类处理的文本，系统的输出则是与文本关联的类别。由于篇幅所限，其它更具体内容本文将不再详述。

OK，本节虽取标题为“证明 SVM”，但聪明的读者们想必早已看出，其实本部分并无多少证明部分（特此致歉），怎么办呢？可以参阅《支持向量机导论》一书，此书精简而有趣。本节完。

参考资料

(1) Nello Cristianini, and John Shawe-Taylor. 支持向量机导论.
(2) 支持向量机导论一书的支持网站：http://www.support-vector.net/
(3) Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. 数据挖掘导论.
(4) Jiawei Han, and Micheline Kamber. 数据挖掘：概念与技术.
(5) 邓乃扬，田英杰. 数据挖掘中的新方法：支持向量机.
(6) 邓乃扬，田英杰. 支持向量机 –理论、算法和扩展.
(7) pluskid - 支持向量机系列：http://blog.pluskid.org/?page_id=683
(8) http://www.360doc.com/content/07/0716/23/11966_615252.shtml
(9) 数据挖掘十大经典算法初探：http://blog.csdn.net/v_july_v/article/details/6142146
(10) C. J. C Burges. 模式识别支持向量机指南.
(11) 李航. 统计学习方法（第七章）.
(12) 宗成庆. 统计自然语言处理（第十二章）.
(13) Jasper - SVM 入门系列：http://www.blogjava.net/zhenandaci/category/31868.html
(14) 最近邻决策和 SVM 数字识别的实现和比较.
(15) 斯坦福大学机器学习课程原始讲义：http://www.cnblogs.com/jerrylead/archive/2012/05/08/2489725.html
(16) 斯坦福机器学习课程笔记：http://www.cnblogs.com/jerrylead/tag/Machine%20Learning；
(17) http://www.cnblogs.com/jerrylead/archive/2011/03/13/1982639.html
(18) SMO 算法的数学推导：http://www.cnblogs.com/jerrylead/archive/2011/03/18/1988419.html
(19) 数据挖掘掘中所需的概率论与数理统计知识（上）：http://blog.csdn.net/v_july_v/article/details/8308762
(20) 关于机器学习方面的文章，可以读读：http://www.cnblogs.com/vivounicorn/category/289453.html
(21) 数学系教材推荐：http://blog.sina.com.cn/s/blog_5e638d950100dswh.html
(22) Simon Haykin. 神经网络与机器学习（第三版）.
(23) 正态分布的前世今生：http://www.52nlp.cn/tag/%E6%AD%A3%E6%80%81%E5%88%86%E5%B8%83%E7%9A%84%E5%89%8D%E4%B8%96%E4%BB%8A%E7%94%9F
(24) 陈希孺. 数理统计学简史.
(25) 陈宝林. 最优化理论与算法（第 2 版）.
(26) A Gentle Introduction to Support Vector Machines in Biomedicine：http://www.nyuinformatics.org/downloads/supplements/SVM_Tutorial_2010/Final_WB.pdf
(27) http://www.autonlab.org/tutorials/svm15.pdf
(28) libsvm 的作者林智仁教授 06 年的机器学习讲义：http://www.csie.ntu.edu.tw/~cjlin/talks/MLSS.pdf，以及其 2010 年的一个 slide：http://www.csie.ntu.edu.tw/~cjlin/talks/postech.pdf
(29) http://staff.ustc.edu.cn/~ketang/PPT/PRLec5.pdf
(30) Debprakash Patnai - Introduction to Support Vector Machines (SVM)：http://www.pws.stu.edu.tw/ccfang/index.files/AI/AI%26ML-Support%20Vector%20Machine-1.ppt

(31) libsvm：http://www.csie.ntu.edu.tw/ cjlin/libsvm/；
(32) Peter Harrington. 机器学习实战.
(33) SMO 算法的提出：*Sequential Minimal Optimization A Fast Algorithm for Training Support Vector Machines*，http://research.microsoft.com/en-us/um/people/jplatt/smoTR.pdf
(34) Vladimir N. Vapnik. 统计学习理论的本质.
(35) 张兆翔，机器学习第五讲之支持向量机：http://irip.buaa.edu.cn/~zxzhang/courses/MachineLearning/5.pdf
(36) VC 维的理论解释：http://www.svms.org/vc-dimension，http://xiaoxia001.iteye.com/blog/1163338
(37) Jason Weston 关于 SVM 的讲义：http://www.cs.columbia.edu/~kathy/cs4701/documents/jason_svm_tutorial.pdf
(38) 来自 MIT 的 SVM 讲义：http://www.mit.edu/~9.520/spring11/slides/class06-svm.pdf
(39) 关于 PAC（Probably Approximately Correct）：http://www.cs.huji.ac.il/~shashua/papers/class11-PAC2.pdf
(40) 张潼老师的两篇论文：*Statistical Behavior and Consistency of Classification Methods Based on Convex Risk Minimization*，http://home.olemiss.edu/~xdang/676/Consistency_of_Classification_Convex_Risk_Minimization.pdf；*Statistical Analysis of Some Multi-Category Large Margin Classification Methods*
(41) http://jacoxu.com/?p=39
(42) 张贤达. 矩阵分析与应用.
(43) SMO 算法的实现：http://blog.csdn.net/techq/article/details/6171688
(44) 常见面试之机器学习算法思想简单梳理：http://www.cnblogs.com/tornadomeet/p/3395593.html
(45) Wikipedia - 矩阵：http://zh.wikipedia.org/wiki/%E7%9F%A9%E9%98%B5
(46) 最小二乘法及其实现：http://blog.csdn.net/qll125596718/article/details/8248249
(47) 统计学习方法概论：http://blog.csdn.net/qll125596718/article/details/8351337
(48) http://www.csdn.net/article/2012-12-28/2813275-Support-Vector-Machine
(49) A Tutorial on Support Vector Regression：http://alex.smola.org/papers/2003/SmoSch03b.pdf；SVR 简明版：http://www.cmlab.csie.ntu.edu.tw/~cyy/learning/tutorials/SVR.pdf。
(50) SVM Org：http://www.support-vector-machines.org
(51) R. Collobert. Large Scale Machine Learning. Université Paris VI. 2004.http://ronan.collobert.com/pub/matos/2004_phdthesis_lip6.pdf.
(52) Making Large-Scale SVM Learning Practical：http://www.cs.cornell.edu/people/tj/publications/joachims_99a.pdf
(53) 文本分类与 SVM：http://blog.csdn.net/zhzhl202/article/details/8197109；
(54) Working Set Selection Using Second Order Information for Training Support Vector Machines：http://www.csie.ntu.edu.tw/~cjlin/papers/quadworkset.pdf
(55) SVM Optimization: Inverse Dependence on Training Set Size：http://icml2008.cs.helsinki.fi/papers/266.pdf
(56) Large-Scale Support Vector Machines: Algorithms and Theory：http://cseweb.ucsd.edu/~akmenon/ResearchExam.pdf
(57) 凸优化的概念：http://cs229.stanford.edu/section/cs229-cvxopt.pdf
(58) Stephen Boyd, and Lieven Vandenberghe. 凸优化.
(59) Zhuang Wang - Large-scale Non-linear Classification: Algorithms and Evaluations，讲了很多 SVM 算法的新进展：http://ijcai13.org/files/tutorial_slides/te2.pdf