7.2 支持向量机[1]

第一节 了解 SVM

支持向量机，因其英文名为 support vector machine，故一般简称 SVM，通俗来讲，它是一种二类分类模型，其基本模型定义为特征空间上的间隔最大的线性分类器，其学习策略便是间隔最大化，最终可转化为一个凸二次规划问题的求解。

7.2.1 线性分类

理解 SVM，咱们必须先弄清楚一个概念：线性分类器。

7.2.1.1 分类标准

考虑一个二类的分类问题，数据点用 x 来表示，类别用 y 来表示，可以取 1 或者-1[2]，分别代表两个不同的类，且是一个 n 维向量，w^T 中的 T 代表转置。一个线性分类器的学习目标便是要在 n 维的数据空间中找到一个分类超平面，其方程可以表示为：

$$w^T x + b = 0$$

7.2.1.2 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

---

[1] 本节新书初稿的第 7 章第 2 节，在博客文章《支持向量机通俗导论（理解 SVM 的三层境界）》：http://blog.csdn.net/v_july_v/article/details/7624837 的基础上优化整理而成，同时，本节主要参考了 pluskid 所写的支持向量机系列：http://blog.pluskid.org/?page_id=683。后面，本节再经过又一轮的改进优化之后，将收录到今 2014 年出版的新书中，有任何反馈或改进意见，欢迎通过微博或博客向我反馈，thanks。July、二零一四年八月十八日。

[2] 可能有读者对类别取 1 或-1 有疑问，事实上，这个 1 或-1 的分类标准起源于 logistic 回归。

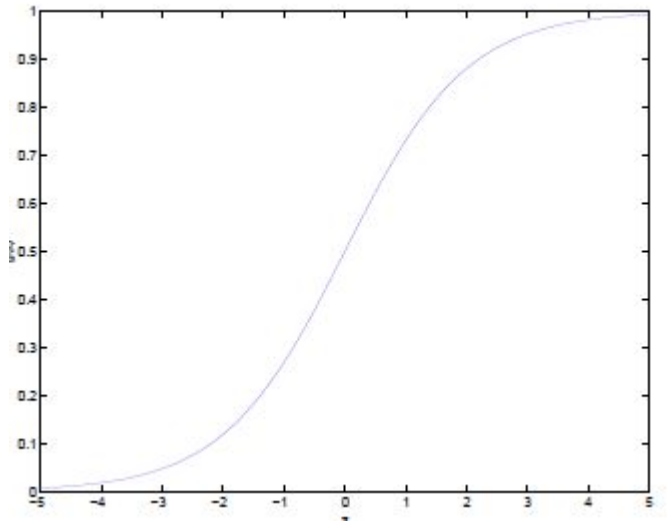
(0,1)上，映射后的值被认为是属于 y=1 的概率。

假设函数

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

其中 x 是 n 维特征向量，函数 g 就是 logistic 函数。

而 $g(z) = \frac{1}{1 + e^{-z}}$ 的图像是



可以看到，通过 logistic 函数将自变量从无穷映射到了(0, 1)，而假设函数就是特征属于 y=1 的概率。

$$\begin{aligned} P(y = 1 \mid x; \theta) &= h_\theta(x) \\ P(y = 0 \mid x; \theta) &= 1 - h_\theta(x) \end{aligned}$$

从而，当我们要判别一个新来的特征属于哪个类时，只需求$h_\theta(x)$，若大于 0.5 就是 y=1 的类，反之属于 y=0 类。

再审视一下$h_\theta(x)$，发现$h_\theta(x)$只和$\theta^T x$有关，$\theta^T x$>0，那么$h_\theta(x)$>0.5，g(z)只不过是用来映射，真实的类别决定权还在$\theta^T x$。还有当$\theta^T x$>>0 时，$h_\theta(x)$=1，反之$h_\theta(x)$=0。如果我们只从$\theta^T x$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$\theta^T x$>>0，而是 y=0 的特征$\theta^T x$<<0。Logistic 回归就是要学习得到θ，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。

7.2.1.3 形式化标示

- 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θ替换成 w 和 b。
  - 以前的$\theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$，其中认为$x_0 = 1$，现在我们替换为 b；
  - 后面的$\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$ 替换为$w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$（即$w^T x$）。

这样，我们让$\theta^T x = w^T x + b$，进一步$h_\theta(x) = g(\theta^T x) = g(w^T x + b)$。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

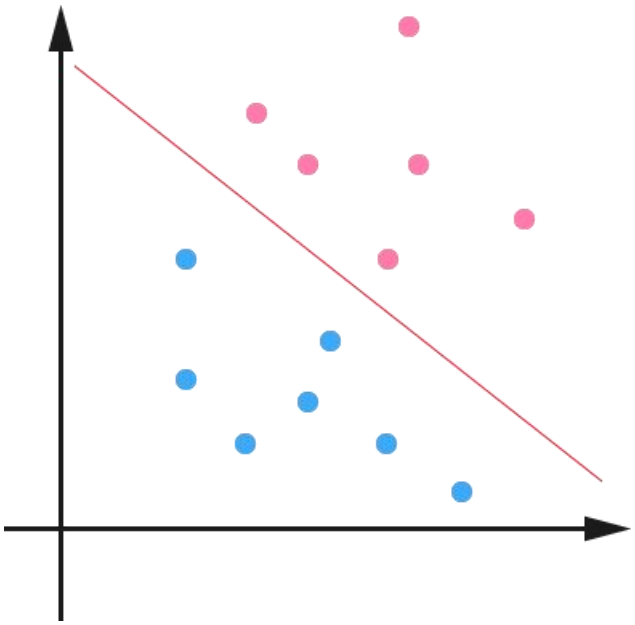
$$h_{w,b}(x) = g(w^T x + b)$$

上面提到过我们只需考虑$\theta^T x$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：

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

7.2.2 线性分类的一个例子

假定现在有一个二维平面，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，另一种为蓝颜色的点。如果我们要在这个二维平面上找到一个可行的超平面的话，那么这个超平面可以是下图中那根红颜色的线(在二维空间中，超平面就是一条直线)。

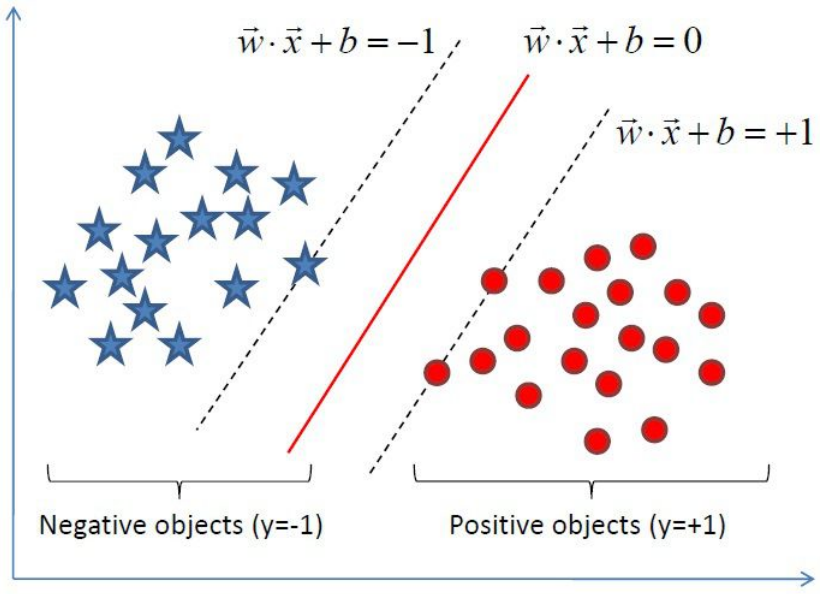


从上图中我们可以看出，这条红颜色的线作为一个超平面，把红颜色的点和蓝颜色的点分开来了，在超平面一边的数据点所对应的 y 全是 -1 ，而在另一边全是 1。

接着，我们可以令分类函数：

$$f(x) = w^T x + b$$

显然，如果 f(x)=0 ，那么 x 是位于超平面上的点。我们不妨要求对于所有满足 f(x) > 0 的点对应 y=1 的数据点。如下图所示：



上图中，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 $f(x) = w^T x + b$ 实质是一样的。

因为后续转化到优化下面这个式子的时候：

$$\max \frac{1}{\|w\|}, \quad s.t., y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n$$

为了求解方便，会令 yf(x)为 1，即 yf(x)是 $y(w^x + b)$，还是 $y(w^x - b)$，对我们要优化的式子 max1/||w|| 已无影响。

从而在我们进行分类的时候，将数据点 x 代入 f(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。

7.2.3 函数间隔 Functional margin 与几何间隔 Geometrical 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准确程度。在超平面 w*x+b=0 确定的情况下，|w*x+b|能够相对的表示点 x 到距离超平面的远近，而 w*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(y*(w*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 functional margin 的概念。

我们定义函数间隔 functional margin 为：

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

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

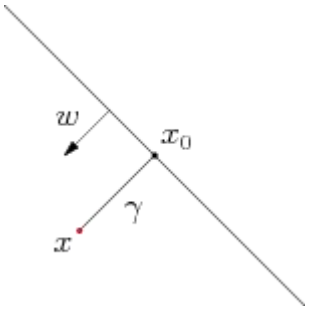
$\hat{\gamma}$ = min $\hat{\gamma}$ i (i=1，...n)

但问题也就出来了：如果成比例的改变 w 和 b，如将他们改变为 2w 和 2b，虽然此时超平面没有改变，但函数间隔的值 f(x)却变成了原来的 2 倍，所以只有函数间隔还远远不够。

事实上，我们可以对法向量 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 geometrical margin 的概念[3]。

对于一个点 x ，令其垂直投影到超平面上的对应点为 x0 ，w 是垂直于超平面的一个向量，$\hat{\gamma}$ 为样本 x 到分类间隔的距离，



我们有

$$x = x_0 + \gamma \frac{w}{\|w\|}$$

其中，‖w‖表示的是范数。

又由于 x0 是超平面上的点，满足 f(x0)=0 ，代入超平面方程 $w^T x + b = 0$ 即可算出：

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

不过这里的 $\hat{\gamma}$ 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 geometrical margin 为：

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

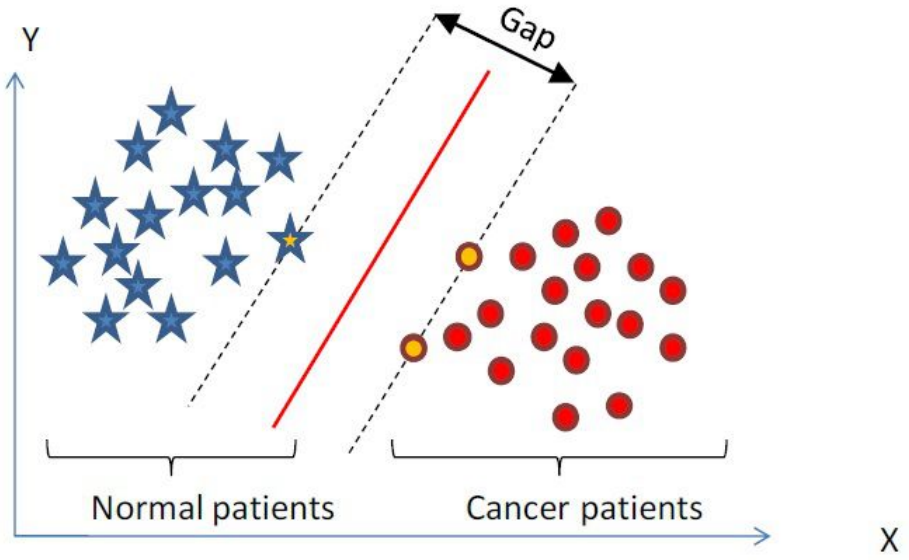
综上，函数间隔 y*(wx+b)=y*f(x)实际上就是|f(x)|，只是人为定义的一个间隔度量；而几何间隔|f(x)|/||w||

[3] 很快你将看到，几何间隔就是函数间隔除以个||w||，即 yf(x) / ||w||。

才是直观上的点到超平面距离。

7.2.4 最大间隔分类器 Maximum Margin Classifier 的定义

事实上，对一个数据点进行分类，当它的 margin 越大的时候，分类的 confidence 越大。于是，为了使得分类的 confidence 尽量高，我们希望所选择的超平面 hyper plane 能够最大化这个 margin 值。下面考虑求一个几何间隔最大的分类超平面。



由前面的分析可知：functional margin 明显是不太适合用来最大化的一个量，因为在 hyper plane 固定以后，我们可以等比例地缩放 w 的长度和 b 的值，这样可以使得 $f(x) = w^T x + b$ 的值任意大，亦即 functional margin 可以在 hyper plane 保持不变的情况下被取得任意大；而 geometrical margin 因为除上了 $\|w\|$ 这个分母，所以缩放 w 和 b 的时候 $\hat{\gamma}$ 的值是不会改变的，它只随着 hyper plane 的变动而变动，因此，这是更加合适的一个 margin 。

这样一来，我们的 maximum margin classifier 的目标函数可以定义为：

$\max \tilde{\gamma}$，满足一定的约束条件，即 s. t. $y_i(w^T x_i + b) = \hat{\gamma}_i \geq \hat{\gamma}, \quad i = 1, \ldots, n$

其中 $\hat{\gamma} = \tilde{\gamma}||w||$ [4]，出于方便推导和优化的目的，我们可以令 $\hat{\gamma}$ =1(对目标函数的优化没有影响)，此时，上述的目标函数 $\hat{\gamma}$ 转化为：

$$\max \frac{1}{\|w\|}, \quad s.t., y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n$$

其中，s.t.，即 subject to 的意思，它导出的是约束条件。

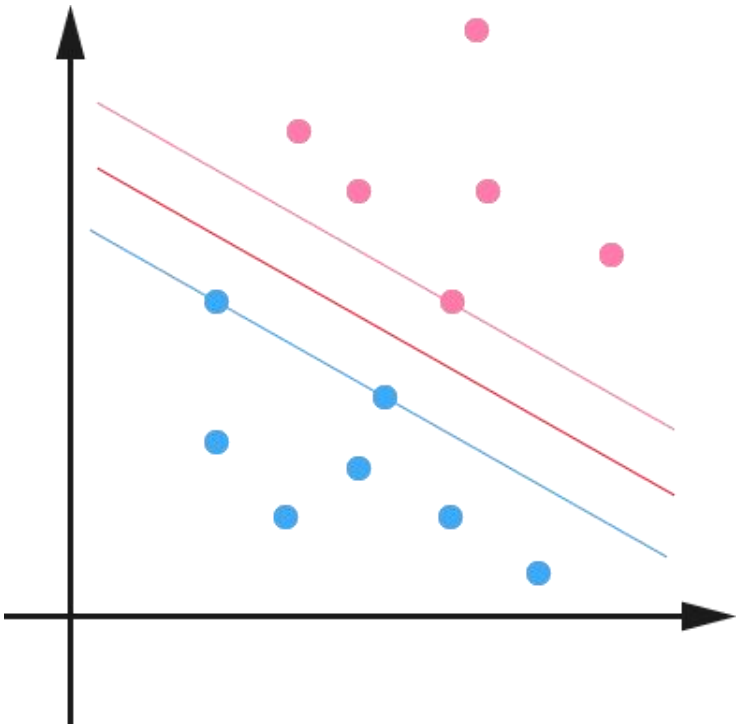
接下来，就是如何求 w 和 b 的问题了。

针对这个问题，我们就可以找到一个 margin 最大的 classifier，通过最大化 margin，我们使得该分类器在对数据进行分类时具有了最大的 confidence，从而设计决策最优分类超平面。

如下图所示，中间的红色线条是 Optimal Hyper Plane，另外两条线到红线的距离都是等于 $\hat{\gamma}$ 的[5]：

[4] 等价于 $\hat{\gamma}$ = $\hat{\gamma}$ /||w||，故有稍后的 $\hat{\gamma}$ =1 时， $\hat{\gamma}$ = 1 / ||w||。

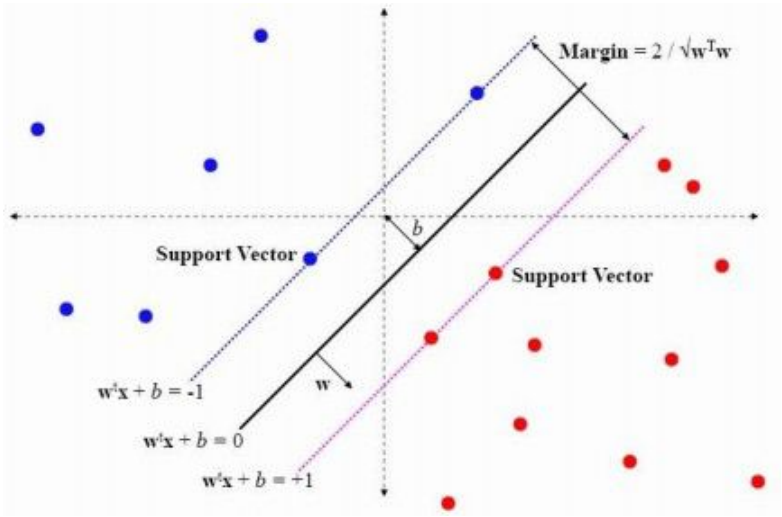
[5] $\hat{\gamma}$ 便是上文所定义的 geometrical margin，当令 $\hat{\gamma}$ =1 时， $\hat{\gamma}$ 便为 1/||w||，而我们上面得到的目标函数便是在相应的约束条件下，要最大化这个 1/||w||值。



此外，什么是 Support Vector 呢？

通过上图，我们可以看到两个支撑着中间的 gap 的超平面，到中间的纯红线 separating hyper plane 的距离相等，即我们所能得到的最大的 geometrical margin，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 Support Vector。

换言之，Support Vector 便是下图中那蓝色虚线和粉红色虚线上的点：



很显然，由于这些 supporting vector 刚好在边界上，所以它们满足$y(w^Tx+b)=1$，而对于所有

不是支持向量的点，则显然有$y(w^Tx+b)>1$。

第二节 深入 SVM

7.2.6 从线性可分到线性不可分

7.2.6.1 从原始问题到对偶问题的求解

根据我们之前得到的目标函数（subject to 导出的则是约束条件）：

$$\max \frac{1}{\|w\|}, \quad s.t., y_i(w^Tx_i+b)\geq 1, i=1,\ldots,n$$

由于求$\frac{1}{||w||}$的最大值相当于求$\frac{1}{2}||w||^2$的最小值，所以上述目标函数等价于：

$$\min \frac{1}{2}\|w\|^2 \quad s.t., y_i(w^Tx_i+b)\geq 1, i=1,\ldots,n$$

这样，我们的问题成为了一个凸优化问题，因为现在的目标函数是二次的，约束条件是线性的，所以它是一个凸二次规划问题。这个问题可以用任何现成的 QP（Quadratic Programming） 的优化包进行求解。一言以蔽之：在一定的约束条件下，目标最优，损失最小。

但由于这个问题的特殊结构，我们可以过 Lagrange Duality 变换到对偶变量 (dual variable) 的优化问题，即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。

那什么是 Lagrange duality？简单地来说，通过给每一个约束条件加上一个 Lagrange multiplier(拉格朗日乘值)，即引入拉格朗日乘子α，如此我们便可以通过拉格朗日函数将约束条件融合到目标函数里去[6]：

---

[6] 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。

$$\mathcal{L}(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i\left(y_i(w^Tx_i+b)-1\right)$$

然后我们令

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

容易验证：

- 当某个约束条件不满足时，例如$y_i(w^Tx_i+b)<1$，那么我们显然有$\theta(w)=\infty$（只要令$\alpha_i=\infty$即可）。
- 而当所有约束条件都满足时，则有$\theta(w)=\frac{1}{2}\|w\|^2$，亦即我们最初要最小化的量$\frac{1}{2}\|w\|^2$。

因此，在要求约束条件得到满足的情况下最小化$\theta(w)$，实际上等价于直接最小化α_i，约束条件为$\alpha_i \geq 0$, i=1, ⋯, n，如果约束条件没有得到满足，$\theta(w)$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：

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

这里用p^*表示这个问题的最优值，且和我们最初的问题是等价的。如果直接求解，那么一上来便得面对 w 和 b 两个参数，而α_i又是不等式约束，这个求解过程不好做。不妨把最小和最大的位置交换一下，变成：

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

交换之后的新问题是交换之前问题的对偶问题，这个新问题的最优值用d^*来表示。而且，我们有$d^* \leq p^*$，在满足某些条件的情况下，这两者等价，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。

也就是说，下面我们可以先求 L 对 w、b 的极小，再求 L 对α的极大。而且，之所以从 minmax 的原始问题p^*，转化为 maxmin 的对偶问题d^*，一者因为d^*是p^*的近似解，二者，转化为对偶问题后，更容易求解。

7.2.6.2 KKT 条件

上段说“$d^* \leq p^*$在满足某些条件的情况下，两者等价”，这所谓的“满足某些条件”就是要满足 KKT 条件。那 KKT 条件的表现形式是什么呢？一般地，一个最优化数学模型能够表示成下列标准形式：

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

其中，f(x)是需要最小化的函数，h(x)是等式约束，g(x)是不等式约束，p 和 q 分别为等式约束和不等式约束的数量。同时，我们得明白以下两个定理：

- 凸优化的概念：$\mathcal{X} \subset \mathbb{R}^n$为一凸集，$f : \mathcal{X} \to \mathbb{R}$为一凸函数。凸优化就是要找出一点$x^* \in \mathcal{X}$，使得每一$x \in \mathcal{X}$满足$f(x^*) \leq f(x)$。
- KKT 条件的意义：它是一个非线性规划（Nonlinear Programming）问题能有最优化解法的充要条件。

而 KKT 条件就是指上面最优化数学模型的标准形式中的最小点 x* 必须满足下面的条件：

1. $h_j(\mathbf{x}_*)=0,\ j=1,\ldots,p,\ g_k(\mathbf{x}_*)\le 0,\ k=1,\ldots,q,$

2. $\nabla f(\mathbf{x}_*)+\sum_{j=1}^{p}\lambda_j\nabla h_j(\mathbf{x}_*)+\sum_{k=1}^{q}\mu_k\nabla g_k(\mathbf{x}_*)=\mathbf{0},$

$\lambda_j\neq 0,\ \mu_k\ge 0,\ \mu_k g_k(\mathbf{x}_*)=0.$

经过论证，我们这里的问题是满足 KKT 条件的[7]，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 L(w，b，a) 关于 w 和 b 最小化，然后求对 α 的极大，最后利用 SMO 算法求解对偶因子。

7.2.6.3 对偶问题求解的 3 个步骤

1. 首先固定 α，要让 L 关于 w 和 b 最小化，分别对 w 和 b 求偏导数，即令 ∂L/∂w 和 ∂L/∂b 等于零：

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

将结果代入之前的 L，得到：

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

从而有：

[7] 首先已经满足 Slater condition，再者 f 和 gi 也都是可微的，即 L 对 w 和 b 都可导。

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

上述具体的推导过程，如下图所示：

$$\mathcal{L}(\mathrm{w},\mathrm{b},\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{m}\alpha_i\left[y^{(i)}\left(w^Tx^{(i)}+b\right)-1\right]$$

$$=\frac{1}{2}\mathrm{w}^T\mathrm{w}-\sum_{i=1}^{m}\alpha_i y^{(i)}w^Tx^{(i)}-\sum_{i=1}^{m}\alpha_i y^{(i)}b+\sum_{i=1}^{m}\alpha_i$$

$$=\frac{1}{2}\mathrm{w}^T\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}-\sum_{i=1}^{m}\alpha_i y^{(i)}w^Tx^{(i)}-\sum_{i=1}^{m}\alpha_i y^{(i)}b+\sum_{i=1}^{m}\alpha_i$$

$$=\frac{1}{2}\mathrm{w}^T\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}-w^T\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}-\sum_{i=1}^{m}\alpha_i y^{(i)}b+\sum_{i=1}^{m}\alpha_i$$

$$=-\frac{1}{2}\mathrm{w}^T\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}-\sum_{i=1}^{m}\alpha_i y^{(i)}b+\sum_{i=1}^{m}\alpha_i$$

$$=-\frac{1}{2}\mathrm{w}^T\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}-b\sum_{i=1}^{m}\alpha_i y^{(i)}+\sum_{i=1}^{m}\alpha_i$$

$$=-\frac{1}{2}\left(\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}\right)^T\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}-b\sum_{i=1}^{m}\alpha_i y^{(i)}+\sum_{i=1}^{m}\alpha_i$$

$$=-\frac{1}{2}\sum_{i=1}^{m}\alpha_i y^{(i)}\left(x^{(i)}\right)^T\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}-b\sum_{i=1}^{m}\alpha_i y^{(i)}+\sum_{i=1}^{m}\alpha_i$$

$$=-\frac{1}{2}\sum_{i=1,j=1}^{m}\alpha_i y^{(i)}\left(x^{(i)}\right)^T\alpha_j y^{(j)}x^{(j)}-b\sum_{i=1}^{m}\alpha_i y^{(i)}+\sum_{i=1}^{m}\alpha_i$$

最后，得到：

$$\mathcal{L}(w,b,\alpha)=\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j-\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j-b\sum_{i=1}^{n}\alpha_i y_i+\sum_{i=1}^{n}\alpha_i$$
$$=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j$$

从上面的最后一个式子，我们可以看出，此时的拉格朗日函数只包含了一个变量，那就是α_i，然后下文的第 2 步，求出了α_i便能求出 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$f(x)=w^Tx+b$也就可以轻而易举的求出来了。

2. 求对α的极大，即是关于对偶问题的最优化问题。经过上面第一个步骤的求 w 和 b，得到的拉格朗日函数式子已经没有了变量 w 和 b，只有α，从上面的式子得到：

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

这样，求得的α将能导出 w，b 的解，最终得出分离超平面和分类决策函数[8]。

3. 如前所说，在求得 L(w, b, a) 关于 w 和 b 最小化，以及对α的极大之后，最后一步便是利用 SMO 算法求解对偶因子。

---

[8] 求出了α_i，根据 $w=\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}$，即可求出 w。然后通过 $b^*=-\frac{\max_{i:y^{(i)}=-1}w^{*T}x^{(i)}+\min_{i:y^{(i)}=1}w^{*T}x^{(i)}}{2}$，即可求出 b。

7.2.6.4 序列最小最优化 SMO 算法

$$\begin{aligned}\max_\alpha \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \\ \text{s.t.} \quad & 0 \le \alpha_i \le C, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0,\end{aligned}$$

上述式子解决的是在参数$\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$上求最大值 W 的问题，至于$x^{(i)}$和$y^{(i)}$都是已知数，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 dual variable α 多了一个上限 C，C 是一个参数，用于控制目标函数中两项（“寻找 margin 最大的超平面”和“保证数据点偏差量最小”）之间的权重[9]。

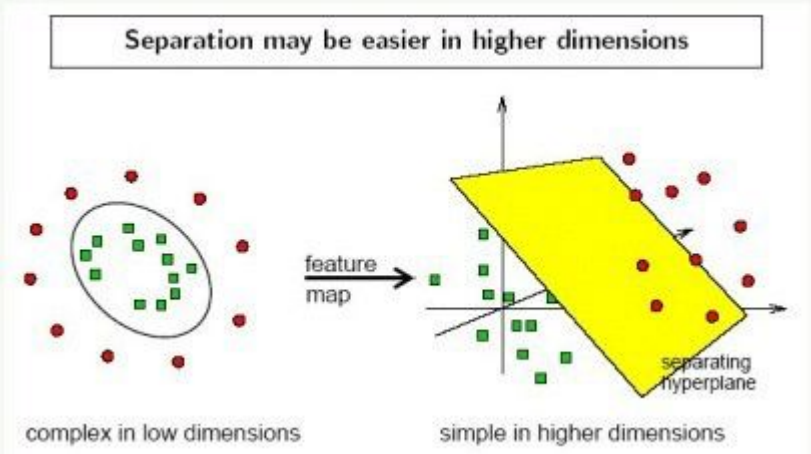
到目前为止，我们的 SVM 还比较弱，只能处理线性的情况，下面我们将引入核函数，进而推广到非线性分类问题。

7.2.7 核函数 Kernel

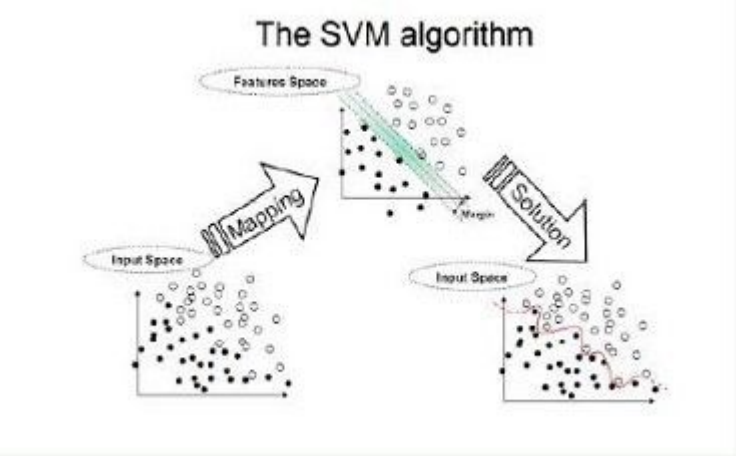
7.2.7.1 特征空间的隐式映射：核函数

在线性不可分的情况下，支持向量机通过一种事先选择的非线性映射（核函数）将输入空间映射到高维特征空间，在这个空间中构造最优分类超平面，从而把平面上本身不好分的非线性数据分开了：

[9] 要了解这个 SMO 算法是如何推导的，请跳到下文第 7.2.10 节 SMO 算法。



这种方法使得在高维属性空间中有可能需要训练的数据实现超平面的分割，避免了在原输入空间中进行非线性曲面分割计算，且在处理高维输入空间的分类时，其工作原理如下图所示：



而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：

$$f(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi_i(\mathbf{x}) + b,$$

这里ϕ：X->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 F，
2. 然后在特征空间使用线性学习器分类。

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：

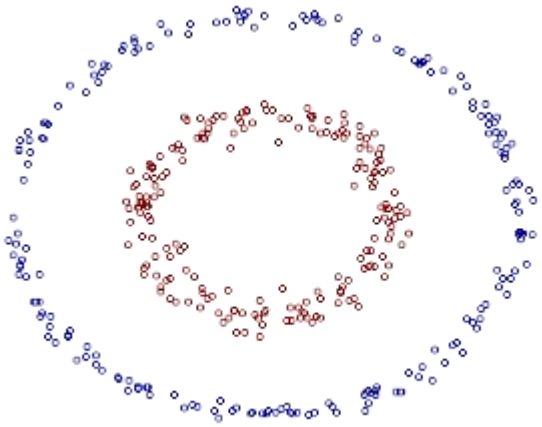
$$f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i y_i \langle \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}) \rangle + b.$$

如果有一种方式可以在特征空间中直接计算内积〈φ(xi • φ(x)〉，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。

定义：核是一个函数 K，对所有 x，z(-X，满足 $K(\mathbf{x}, \mathbf{z}) = \langle \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{z}) \rangle,$ ，这里 φ 是从 X 到内积特征空间 F 的映射。

7.2.7.2 核函数：如何处理非线性数据

我们已经知道，如果是线性方法，所以对非线性的数据就没有办法处理。举个例子，如下图所示的两类数据，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，我们该如何把这两类数据分开呢？



此时，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：

$$a_1 X_1 + a_2 X_1^2 + a_3 X_2 + a_4 X_2^2 + a_5 X_1 X_2 + a_6 = 0$$

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 $Z_1 = X_1, Z_2 = X_{21}, Z_3 = X_2, Z_4 = X_{22}, Z_5 = X_1X_2$，那么显然，上面的方程在新的坐标系下可以写作：

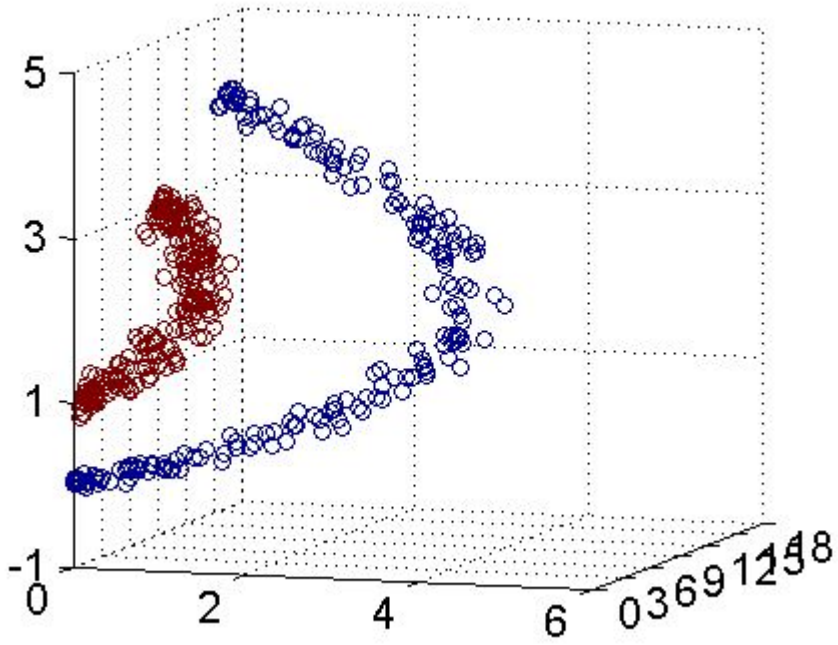
$$\sum_{i=1}^{5} a_i Z_i + a_6 = 0$$

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 ϕ:R2→R5 ，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。

再进一步描述 Kernel 的细节之前，不妨再来看看这个例子映射过后的直观例子。具体来说，这里的超平面实际的方程是这个样子（圆心在 X2 轴上的一个正圆）：

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

因此我只需要把它映射到 $Z_1 = X_{21}, Z_2 = X_{22}, Z_3 = X_2$ 这样一个三维空间中即可，下图即是映射之后的结果，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：



回忆一下，我们上一次 7.2.2 节中得到的最终的分类函数是这样的：

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

映射过后的空间是：

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

而其中的α也是通过求解如下 dual 问题而得到的：

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle \\ &s.t., \alpha_i \geq 0, i = 1, \ldots, n \\ &\sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned}$$

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，这会给 $\phi(\cdot)$ 的计算带来了非常大的困难，而且如果遇到无穷维的情况，根本无从计算。此时需要引入 Kernel。

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$\phi(\cdot)$即是之前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，那么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：

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

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。对比刚才我们上面写出来的式子，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：

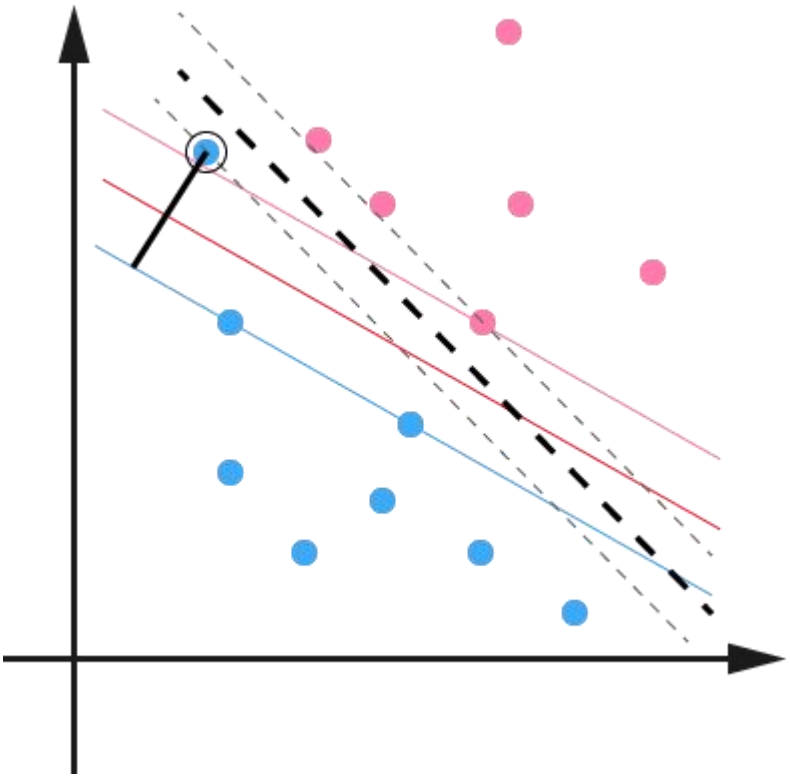
$$\max_{\alpha} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \kappa(x_i, x_j)$$
$$s.t., \alpha_i \geq 0, i = 1, \ldots, n$$
$$\sum_{i=1}^{n}\alpha_i y_i = 0$$

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。

7.2.8 使用松弛变量处理 outliers 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，在上文 2.2 节使用 Kernel 方法对原来的线性 SVM 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的。

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 outlier ，在我们原来的 SVM 模型里，outlier 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 support vector 组成的，如果这些 support vector 里又存在 outlier 的话，其影响就很大了。例如下图：



用黑圈圈起来的那个蓝点是一个 outlier ，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 outlier 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[10]，同时 margin 也相应变小了。当然，更严重的情况是，如果这个 outlier 再

[10] 这只是一个示意图，并没有严格计算精确坐标。

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，SVM 允许数据点在一定程度上偏离一下超平面。例如上图中，黑色实线所对应的距离，就是该 outlier 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。

我们原来的约束条件为：

$$y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, n$$

现在考虑到 outlier 问题，约束条件变成了：

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n$$

其中 $\xi_i \geq 0$ 称为松弛变量 (slack variable) ，对应数据点 x_i 允许偏离的 functional margin 的量。当然，如果我们运行 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：

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

其中 参数 C 是一个事先确定好的常量，用于控制目标函数中两项（“寻找 margin 最大的超平面”和“保证数据点偏差量最小”）之间的权重。而 ξ 是需要优化的变量之一。完整表达如下：

$$\begin{aligned} &\min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \\ &s.t., y_i(w^T x_i + b) \geq 1 - \xi_i, i = 1, \ldots, n \\ &\quad \xi_i \geq 0, i = 1, \ldots, n \end{aligned}$$

用之前的方法将限制或约束条件加入到目标函数中，得到新的拉格朗日函数，如下所示：

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

分析方法和前面一样，转换为另一个问题之后，我们先让$\mathcal{L}$针对 w、b 和ξ_i 最小化：

$$\frac{\partial\mathcal{L}}{\partial w}=0\Rightarrow w=\sum_{i=1}^{n}\alpha_i y_i x_i$$
$$\frac{\partial\mathcal{L}}{\partial b}=0\Rightarrow\sum_{i=1}^{n}\alpha_i y_i=0$$
$$\frac{\partial\mathcal{L}}{\partial \xi_i}=0\Rightarrow C-\alpha_i-r_i=0,\quad i=1,\ldots,n$$

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

$$\max_{\alpha}\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j\langle x_i,x_j\rangle$$

不过，由于我们得到$C-\alpha_i-r_i=0$而又有 ri >= 0（作为 Lagrange multiplier 的条件），因此有$\alpha_i\le C$，所以整个 dual 问题现在写作：

$$\max_{\alpha}\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j\langle x_i,x_j\rangle$$
$$s.t.,\ 0\le\alpha_i\le C,i=1,\ldots,n$$
$$\sum_{i=1}^{n}\alpha_i y_i=0$$

把前后的结果对比一下（错误修正：图中的 Dual formulation 中的 Minimize 应为 maxmize）：

Primal formulation:

$$\text{Minimize } \frac{1}{2}\sum_{i=1}^{n} w_i^2 + C\sum_{i=1}^{N}\xi_i \text{ subject to } y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1 - \xi_i \text{ for } i = 1,\ldots,N$$

Objective function　　Constraints

Dual formulation:

$$\text{Minimize } \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j \vec{x}_i\cdot\vec{x}_j \text{ subject to } 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{N}\alpha_i y_i = 0$$

Objective function　　Constraints

for $i = 1,\ldots,N$.

可以看到唯一的区别就是现在 dual variable α 多了一个上限 C 。而 Kernel 化的非线性形式也是一样的，只要把 $\langle x_i, x_j \rangle$ 换成 $\kappa(x_i, x_j)$ 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 outliers 的支持向量机才终于介绍完毕了。

总而言之，SVM 它本质上即是一个分类方法，用 w^T+b 定义分类函数，于是求 w、b，为寻最大间隔，引出 $1/2\|w\|^2$，继而引入拉格朗日因子，化为对拉格朗日乘子 a 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 w、b 与求 a 等价，而 a 的求解可以用一种快速学习算法 SMO，至于核函数，是为处理非线性情况，若直接映射到高维计算恐维度爆炸，故在低维计算，等效高维表现。

第三节 扩展 SVM

7.2.9 损失函数

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。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 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[11]：

（1）0-1 损失函数

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

（2）平方损失函数

$$L(Y, f(X)) = (Y - f(X))^2$$

（3）绝对损失函数

$$L(Y, f(X)) = |Y - f(X)|$$

（4）对数损失函数

$$L(Y, P(Y|X)) = -\log P(Y|X)$$

给定一个训练数据集

$$T = \{(x_1, y_1), (x_2, y_2), \ldots (x_i, y_i) \ldots, (x_N, y_N)\}$$

模型 f(X)关于训练数据集的平均损失称为经验风险，如下：

$$R_{emp}(f) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i))$$

[11] 引用自《统计学习方法》。

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。

经验风险最小化的策略认为，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：

$$\min_{f \in F} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) \qquad (1)$$

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：

$$R_{srm}(f) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f)$$

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)值就越小，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最优化问题：

$$\min_{f \in F} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f) \qquad (2)$$

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（1）和式（2）。

如此，SVM 有第二种理解，即最优化加损失最小，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，也许会有不同收获”[12]。

7.2.10 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 SMO 算法，但并未提到其具体解法。

[12] 关于损失函数，还可以看看张潼的这篇《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，他还有另外一篇论文《Statistical analysis of some multi-category large margin classification methods》，在多分类情况下 margin loss 的分析，这两篇对 Boosting 和 SVM 使用的损失函数分析的很透彻。

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 的这篇文章来看看 SMO 的解法是怎样的。

7.2.10.1 SMO 算法的解法

咱们首先来定义特征到结果的输出函数为

$$u = \vec{w} \cdot \vec{x} - b$$

再三强调，这个 u 与我们之前定义的 $f(x) = w^T x + b$ 实质是一样的。

接着，咱们重新定义下原始的优化问题，权当重新回顾，如下：

$$\min_{w,b} \frac{1}{2} \|\vec{w}\|^2 \text{ subject to } y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, \forall i$$

求导得到：

$$\vec{w} = \sum_{i=1}^{N} y_i \alpha_i \vec{x}_i, \quad b = \vec{w} \cdot \vec{x}_k - y_k \text{ for some } \alpha_k > 0$$

代入 $u = \vec{w} \cdot \vec{x} - b$ 中，可得 $u = \sum_{j=1}^{N} y_j \alpha_j K(\vec{x}_j, \vec{x}) - b$。

引入对偶因子后，得：

$$\min_{\alpha} \Psi(\vec{\alpha}) = \min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j (\vec{x}_i \cdot \vec{x}_j) \alpha_i \alpha_j - \sum_{i=1}^{N} \alpha_i$$

s.t： $\alpha_i \geq 0, \forall i,$ 且 $\sum_{i=1}^{N} y_i \alpha_i = 0.$ [13]

经过加入松弛变量后，模型修改为：

$$\min_{w,b,\xi} \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{N}\xi_i \text{ subject to } y_i(\vec{w}\cdot\vec{x}_i - b) \geq 1 - \xi_i, \forall i$$

$$0 \leq \alpha_i \leq C, \forall i$$

从而最终我们的问题变为：

$$\min_{\alpha} \Psi(\vec{\alpha}) = \min_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j K(\vec{x}_i, \vec{x}_j)\alpha_i\alpha_j - \sum_{i=1}^{N}\alpha_i$$
$$0 \leq \alpha_i \leq C, \forall i,$$
$$\sum_{i=1}^{N} y_i \alpha_i = 0.$$

继而，根据 KKT 条件可以得出其中取值的意义为：

$$\alpha_i = 0 \Leftrightarrow y_i u_i \geq 1,$$
$$0 < \alpha_i < C \Leftrightarrow y_i u_i = 1,$$
$$\alpha_i = C \Leftrightarrow y_i u_i \leq 1.$$

这里的还是拉格朗日乘子(问题通过拉格朗日乘法数来求解)

1. 对于第 1 种情况，表明 α_i 是正常分类，在边界内部（我们知道正确分类的点 yi*f(xi)>=0）；

[13] 这里得到的 min 函数与我们之前的 max 函数实质也是一样的，因为把符号变下，即有 min 转化为 max 的问题，且 yi 也与之前的 $y^{(i)}$ 等价，yj 亦如此。

2. 对于第 2 种情况，表明了α_i是支持向量，在边界上；

3. 对于第 3 种情况，表明了α_i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：

- y_iu_i<=1 但是α_i<C 则是不满足的,而原本α_i=C
- y_iu_i>=1 但是α_i>0 则是不满足的而原本α_i=0
- y_iu_i=1 但是α_i=0 或者α_i=C 则表明不满足的，而原本应该是 0<α_i<C

所以要找出不满足 KKT 条件的这些 a_i 并更新这些 a_i（这也是应用此 SMO 算法的目的），但这些 a_i 又受到另外一个约束，即

$$\sum_{i=1}^{m}\alpha_i y^{(i)}=0$$

L 对 a 和 b 求偏导，得到：

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

因此，我们通过另一个方法，即同时更新 a_i 和 a_j，要求满足以下等式：

$$a_i^{new}y_i+a_j^{new}y_j=a_i^{old}y_i+a_j^{old}y_j=常数$$

就能保证和为 0 的约束。

利用 $y_ia_i+y_ja_j$=常数，消去 ai，可得到一个关于单变量 a_j 的一个凸二次规划问题，不考虑其约束 0<=a_j<=C, 可以得其解为：

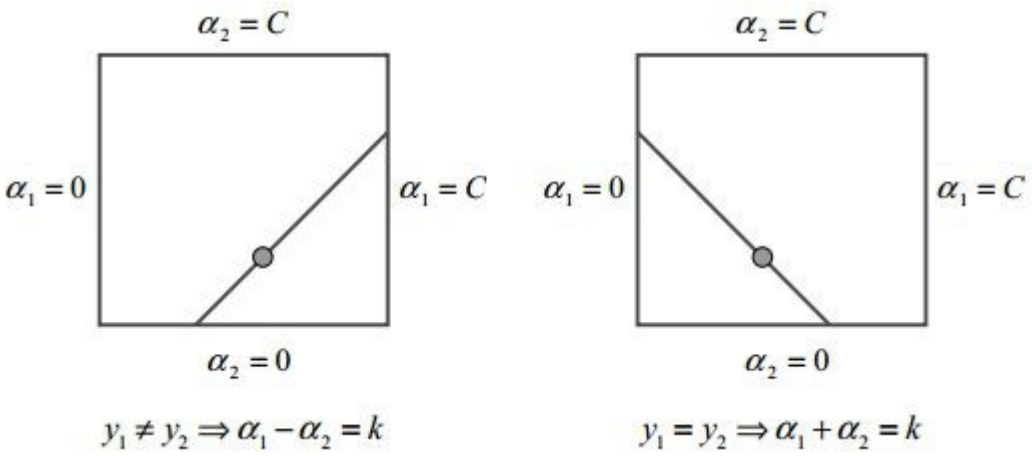
$$\alpha_j^{new} = \alpha_j + \frac{y_j(E_i - E_j)}{\eta}$$

这里$\eta = K(\vec{x}_1, \vec{x}_1) + K(\vec{x}_2, \vec{x}_2) - 2K(\vec{x}_1, \vec{x}_2)$,，表示旧值。

然后考虑约束 0<=a_j<=C 可得到 a 的解析解为：

$$\alpha_j^{new,clipped} = \begin{cases} H & if \quad \alpha_j^{new} \geq H \\ \alpha_j^{new} & if \quad L < \alpha_j^{new} < H \\ L & if \quad \alpha_j^{new} \leq L \end{cases}$$

把 SMO 中对于两个参数求解过程看成线性规划来理解的话，那么下图所表达的便是约束条件：$a_i^{new} y_i + a_j^{new} y_j = a_i^{old} y_i + a_j^{old} y_j = 常数$



根据 y_i 和 y_j 同号或异号，可得出两个拉格朗日乘子的上下界分别为（公式有问题，L max，H min）：

$$\begin{cases} L = \max(0, \alpha_j - \alpha_i), H = \max(C, C + \alpha_j - \alpha_i) & if \quad y_i \neq y_j \\ L = \max(0, \alpha_j + \alpha_i - C), H = \max(C, \alpha_j - \alpha_i) & if \quad y_i = y_j \end{cases}$$

对于α_i，有 $\alpha_i^{new} = \alpha_i + y_i y_j (\alpha_j - \alpha_j^{new,clipped})$ 。

那么如何求得 a_i 和 a_j 呢？

- 对于 a_i，即第一个乘子，可以通过刚刚说的那 3 种不满足 KKT 的条件来找；
- 而对于第二个乘子 a_j 可以找满足条件 ：$\max |E_i - E_j|$ 求得。

而 b 的更新则是：

$$b_1 = b - E_i - y_i(a_i - a_i^{old})k(x_i, x_i) - y_j(a_j - a_j^{old})k(x_i, x_j)$$

$$b_2 = b - E_j - y_i(a_i - a_i^{old})k(x_i, x_i) - y_j(a_j - a_j^{old})k(x_j, x_j)$$

在满足下述条件：

$$b := \begin{cases} b_1 & \text{if } 0 < \alpha_i < C \\ b_2 & \text{if } 0 < \alpha_j < C \\ (b_1 + b_2)/2 & \text{otherwise} \end{cases}$$

下更新 b，且每次更新完两个乘子的优化后，都需要再重新计算 b，及对应的 E_i 值。 最后更新所有 a_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

$$f(x) = \sum_{j=1}^{n} \alpha_j y_j k(x_j, x) + b$$

此外，这里也有一篇类似的文章，大家可以参考下。

7.2.10.2 SMO 算法的步骤

这样，SMO 的主要步骤如下：

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.

}

意思是，

第一步选取一对 a_i 和 a_j，选取方法使用启发式方法；

第二步，固定除 a_i 和 a_j 之外的其他参数，确定 W 极值条件下的 a_i，由 a_j 表示 a_i。

假定在某一次迭代中，需要更新对应的拉格朗日乘子，那么这个小规模的二次规划问题写为：

$$L_s = \max_{\alpha}\{(\alpha_1+\alpha_2)+\sum_{i=3}^{n}\alpha_i-\frac{1}{2}\left\|\alpha_1 y_1\phi(\boldsymbol{x_1})+\alpha_1 y_1\phi(\boldsymbol{x_1})+\sum_{i=3}^{n}\alpha_i y_i\phi(\boldsymbol{x_i})\right\|^2\}$$

$$s.t.\,\alpha_1 y_1+\alpha_2 y_2=-\sum_{i=3}^{n}\alpha_i y_i, 0<\alpha_i<C, \forall i$$

那么在每次迭代中，如何更新乘子呢？更新过程如下图所示：

更新拉格朗日乘子α_1, α_2

- 步骤1：计算上下界L和H
  - $L = \max(0, \alpha_2^{old} - \alpha_1^{old}), H = \min(C, C + \alpha_2^{old} - \alpha_1^{old}), if\ y_1 \neq y_2$
  - $L = \max(0, \alpha_2^{old} + \alpha_1^{old} - C), H = \min(C, \alpha_2^{old} + \alpha_1^{old}), if\ y_1 = y_2$

- 步骤2：计算L_s的二阶导数
  - $\eta = 2\phi(\boldsymbol{x_1})^t\phi(\boldsymbol{x_2}) - \phi(\boldsymbol{x_1})^t\phi(\boldsymbol{x_1}) - \phi(\boldsymbol{x_2})^t\phi(\boldsymbol{x_2})$

- 步骤3：更新L_s

$$\alpha_2^{new} = \alpha_2^{old} - \frac{y_2(e_1 - e_2)}{\eta}$$

$$e_i = g^{old}(\boldsymbol{x_i}) - y_i$$

- 步骤4：计算变量α_2

$$\alpha^{temp} = \begin{cases} H, & \text{if}\alpha_2^{new} \geq H \\ \alpha_2^{new}, & \text{if } L \leq \alpha_2^{new} \leq H \\ L, & \text{if } \alpha_2^{new} \leq L \end{cases}$$

- 步骤5：更新α_1

$$\alpha_1^{new} = \alpha_1^{old} + y_1y_2(\alpha_2^{old} - \alpha^{temp})$$

知道了如何更新乘子，那么选取哪些乘子进行更新呢？具体选择方法有以下两个步骤：

1. 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 a_2；

2. 步骤 2：在所有不违反 KKT 条件的乘子中，选择使$|E_1 - E_2|$最大的 a_1（注：别忘了，其中$E_i = u_i - y_i$，而$u = \vec{w} \cdot \vec{x} - b$，求出来的 E 代表函数 u_i 对输入 x_i 的预测值与真实输出类标记 y_i 之差）。

值得一提的是，每次更新完两个乘子的优化后，都需要再重新计算 b，及对应的 E_i 值。

与此同时，乘子的选择务必遵循两个原则：

- 使乘子能满足 KKT 条件
- 对一个满足 KKT 条件的乘子进行更新，应能最大限度增大目标函数的值（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 a_i 和 a_j 进行调整，其它分量则保持固定不变，在得到解 a_i 和 a_j 之后，再用 a_i 和 a_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。

7.2.10.3、SMO 算法的实现

行文至此，我相信，SVM 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，max1/||w||，min(1/2||w||)^2，凸二次规划，拉格朗日函数，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，为什么要这样那样，太多东西可以追究，最后实现。

至于上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，大家可以看看，此外这里还有一份 libsvm 的注释文档。

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，大家可以看下。

读者评论

本文发表后，很多朋友给了不少意见，以下是节选的一些精彩评论：

1. “压力”陡增的评论→//@藏了个锋：我是看着 July 大神的博文长大的啊//@zlkysl：就是看了最后那一篇才决定自己的研究方向为 SVM 的。

2. @张金辉：“July 的 SVM 三重境界，不得不转的一篇。其实 Coursera 的课堂上 Andrew Ng 讲过支持向量机，但显然他没有把这作为重点，加上 Ng 讲支持向量机的方法我一时半会难以完全消化，所以听的也是一知半解。真正开始了解支持向量机就是看的这篇“三重境界”，之后才对这个算法有了大概的概念，以至如何去使用，再到其中的原理为何，再到支持向量机的证明等。总之，这篇文章开启了我长达数月的研究支持向量机阶段，直到今日”。

3. @孤独之守望者："最后，推出 svm 的 cost function 是 hinge loss，然后对比其他的方法的 cost function，说明其实他们的目标函数很像，那么问题是 svm 为什么这么 popular 呢？您可以再加些 VC dimension 跟一些 error bound 的数学，点一下，提供一个思路和方向"。

4. @夏粉_百度：“在面试时，考察 SVM 可考察机器学习各方面能力：目标函数,优化过程,并行方法，算法收敛性,样本复杂度，适用场景,调参经验，不过个人认为考察 boosting 和 LR 也还不错啊。此外，随着统计机器学习不断进步，SVM 只被当成使用了一个替代 01 损失 hinge 研究，更通用的方法被提出，损失函数研究替代损失与贝叶斯损失关系，算法稳定性研究替代损失与推广性能关系,凸优化研究如何求解凸目标函数，SVM,boosting 等算法只是这些通用方法的一个具体组建而已。”

5. @居里猴姐：关于 SVM 损失函数的问题，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》，在多分类情况下 margin loss 的分析，这两篇对 Boosting 和 SVM 使用的损失函数分析的很透彻。

6. @夏粉_百度：SVM 用了 hinge 损失，hinge 损失不可导，不如其它替代损失方便优化并且转换概率麻烦。核函数也不太用，现在是大数据时代，样本非常大，无法想象一个 n^2 的核矩阵如何存储和计算。 而且，现在现在非线性一般靠深度学习了。//@Copper_PKU:请教 svm 在工业界的应用典型的有哪些？工业界如何选取核函数，经验的方法？svm 的训练过程如何优化？

7. @Copper_PKU：July 的 svm tutorial 我个人觉得还可以加入和修改如下部分：(1) 对于支持向量解释，可以结合图和拉格朗日参数来表达，松弛中 sv 没有写出来. (2) SMO 算法部分，加入 Joachims 论文中提到的算法，以及 SMO 算法选取 workset 的方法，包括 SMO 算法的收敛判断，还有之前共轭梯度求解方法，虽然是较早的算法，但是对于理解 SMO 算法有很好的效果。模型的优化和求解都是迭代的过程，加入历史算法增强立体感。

8. //@廖临川: 之所以 sgd 对大训练集的效果更好，1.因为 SGD 优化每次迭代使用样本子集，比使用训练全集（尤其是百万数量级）要快得多；2.如果目标函数是凸的或者伪凸的，SGD 几乎必然可以收敛到全局最优；否则，则收敛到局部最优；3.SGD 一般不需要收敛到全局最优，只要得到足够好的解，就可以立即停止。//@Copper_PKU：sgd 的核心思想：是迭代训练，每拿到一个样本就算出基于当前 w(t) 的 loss function，t 代表训练第 t 次，然后进行下一 w（t+1）的更新，w(t+1)=w(t)-(learning rate) * loss function 的梯度，这个类比神经网络中 bp 中的参数训练方法。sample by sample 就是每次仅处理一个样本 而不是一个 batch。

9. //@Copper_PKU：从损失函数角度说：primal 问题可以理解为正则化项+lossfunction，求解目标是在两个中间取平衡 如果强调 loss function 最小则会 overfitting，所以有 C 参数。 //@研究者 July：SVM 还真就是在一定限定条件下，即约束条件下求目标函数的最优值问题，同时，为减少误判率，尽量让损失最小。

10. ...

其他资料

1. 《支持向量机导论》，[美] Nello Cristianini / John Shawe-Taylor 著；
2. 支持向量机导论一书的支持网站：http://www.support-vector.net/；
3. 《数据挖掘导论》，[美] Pang-Ning Tan / Michael Steinbach / Vipin Kumar 著；
4. 《数据挖掘：概念与技术》，(加)Jiawei Han;Micheline Kamber 著；
5. 《数据挖掘中的新方法：支持向量机》，邓乃扬 田英杰 著；
6. 《支持向量机--理论、算法和扩展》，邓乃扬 田英杰 著；
7. 支持向量机系列，pluskid：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；
10. 《模式识别支持向量机指南》，C.J.C Burges 著；
11. 《统计学习方法》，李航著(第 7 章有不少内容参考自支持向量机导论一书，不过，可以翻翻看看)；

12. 《统计自然语言处理》，宗成庆编著，第十二章、文本分类；
13. SVM 入门系列，Jasper：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，作者不详；
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；
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. 《神经网络与机器学习(原书第三版)》，[加] Simon Haykin 著；
23. 正态分布的前世今生：http://t.cn/zlH3Ygc；
24. 《数理统计学简史》，陈希孺院士著；
25. 《最优化理论与算法(第 2 版)》，陈宝林编著；
26. A Gentle Introduction to Support Vector Machines in Biomedicine：
http://www.nyuinformatics.org/downloads/supplements/SVM_Tutorial_2010/Final_WB.pdf，此 PPT 很赞，除了对引入拉格朗日对偶变量后的凸二次规划问题的深入度不够之外，其它都挺好，配图很精彩，本文有几张图便引自此 PPT 中；
27. 来自卡内基梅隆大学 carnegie mellon university(CMU)的讲解 SVM 的 PPT：
http://www.autonlab.org/tutorials/svm15.pdf；
28. 发明 libsvm 的台湾林智仁教授 06 年的机器学习讲义 SVM：
http://wenku.baidu.com/link?url=PWTGMYNb4HGUrUQUZwTH2B4r8pIMgLMiWIK1ymVORrds_11VOkHwp-JWab7IALDiors64JW_6mD93dtuWHwFWxsAk6p0rzchR8Qh5_4jWHC；
29. http://staff.ustc.edu.cn/~ketang/PPT/PRLec5.pdf；
30. Introduction to Support Vector Machines (SVM)，By Debprakash Patnai M.E (SSA)，
https://www.google.com.hk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCwQFjAA&url=http%3a%2f%2fwww%2epws%2estu%2eedu%2etw%2fccfang%2findex%2efiles%2fAI%2fAI%26ML-Support%2520Vector%2520Machine-1%2eppt&ei=JRR6UqT5C-iyiQfWyIDgCg&usg=AFQjCNGw1fTbpH4ltQjjmx1d25ZqbCN9nA；
31. 多人推荐过的 libsvm：http://www.csie.ntu.edu.tw/~cjlin/libsvm/；
32. 《machine learning in action》，中文版为《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. 《统计学习理论的本质》，[美] Vladimir N. Vapnik 著，非常晦涩，不做过多推荐；

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/，中文 VC 维解释 http://xiaoxia001.iteye.com/blog/1163338；

37. 来自 NEC Labs America 的 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. PAC 问题：http://www.cs.huji.ac.il/~shashua/papers/class11-PAC2.pdf；

40. 百度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. 《矩阵分析与应用》，清华张贤达著；

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. 矩阵的 wikipedia 页面：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 phd thesis. 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. 《凸优化》，作者: Stephen Boyd / Lieven Vandenberghe，原作名: Convex Optimization；

59. Large-scale Non-linear Classification: Algorithms and Evaluations，Zhuang Wang，讲了很多 SVM 算法的新进展：http://ijcai13.org/files/tutorial_slides/te2.pdf；

60. Sequential Minimal Optimization for SVM：http://www.cs.iastate.edu/~honavar/smo-svm.pdf。