Support Vector Machines

笔记：Alvin

2017-07-07

目录

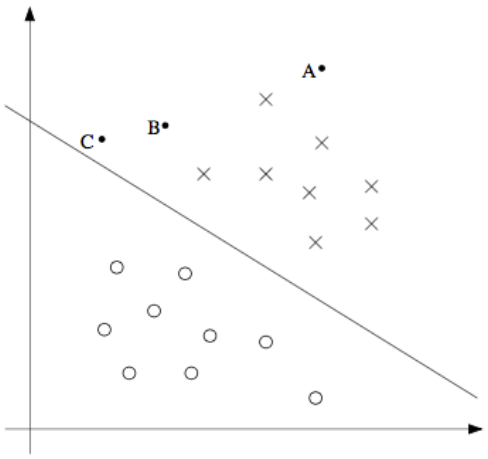
1 支持向量机

这次讲义注意讨论支持向量机学习算法。SVMs 被认为是“off-the-shelf”最好的监督学习算法。在讲它之前，我们先会讨论一下间隔 Margins 以及分离带有大“gap”的数据。接着我们会讨论最优化间隔分类器，从而转到一个题外话：Lagrange duality。我们还会探索它的内核，一种能够在高维特征空间中可以有效应用 SVMs 的方法。最后我们会以 SMO 算法结尾，这个算法提供了支持向量机的高效的实现。

1.1 Margins:Intuition

这一节主要讨论 SVMs 的间隔 margins，讲述关于 margins 的直觉以及我们的预测器的把握。在第三节中会形式化地表述他们。

考虑一个逻辑回归，概率 $p(y=1|x;\theta)$ 是由 $h_\theta(x)=g(\theta^T x)$ 建模。当且仅当输入的 x 满足 $h_\theta(x)\geq 0.5$ 时，或者当且仅当 $\theta^T x\geq 0$ 时，我们预测“1”。考虑一个正向的训练集 (y=1)，$\theta^T x$ 越大，那么 $h_\theta(x)=p(y=1|x;w,b)$ 就越大，因此标签为“1”的把握程度也就越高。所以，当 $\theta^T x >> 0$ 的时候，我们可以认为预测器会自信地预测“y=1”。类似的，我们认为逻辑回归则是一个当 $\theta^T x << 0$ 的时候非常有把握得到 y=0 的预测器。给定一个训练集，似乎我们可以找到一个很好的拟合，当 $y^{(i)}=1$ 时，能够找到 θ 使得 $\theta^T x^{(i)} >> 0$，并且当 $y^{(i)}=0$ 时，能够找到 θ 使得 $\theta^T x^{(i)} << 0$，因为这样可以反映出一个对于所有训练样本的非常有把握的分类器。这似乎是一个漂亮的操作，之后我们会用间隔函数来形式化这样一个想法。



那么对于一个不一样的直觉，考虑上面的图，x 代表着正向的训练样本，o 代表的是负向的训练样本，还有一条决策边界（这是一条由等式 $\theta^T x = 0$ 得到的直线，也称为 separating hyperplane）以及三个标记着 A，B，C 的点。

注意，点 A 离决策边界很远，如果我们要预测 A 处的 y，似乎 y 有绝大的把握是为 1。相反的，点 C 离决策边界十分近，似乎预测 y 为 1 的把握要大于 y 为 0。因此，我认为在 A 处预测要比在 C 处预测更有把握。然而，点 B 的情况处于这两种情况之间，更通用些，所以说，离决策边界越远，某种意义上我们的预测就越是自信。所以，最终我们认为，给定一个训练集，如果我们能够在我们的训练集上找到一个可以使我们做正确且有把握（意思是离边界较远）的预测的一条决策边界，那该多好。我们会在后面用到几何间隔。

1.2 符号表示

为了更加方便地讨论 SVMs，我们首先引入一个新的有关分类器的符号。我们将考虑一个带有标签 y 和特征 x 的二类分类问题的线性分类器。这里我们用 $y \in \{-1, 1\}$ 来代表两种类别。同时，这里不用 θ 作为参数，而是使用 w, b，我们将分类器写成：

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

这里如果 $z > 0$，那么 $g(z) = 1$，否则 $g(z) = -1$。这里的 b 与之前的 θ_0 形似，w 与 $[\theta_1 ... \theta_n]^T$ 相似。

根据以上 g 的定义，我们的分类器可以直接预测 1 或者-1（感知器算法）。而没有首先跳过估计 y 的概率为 1 的中间步骤（逻辑回归中做的）。

（上面两部分不是特别理解，需要结合 lecture video）

1.3 几何间隔函数 Functional and geometric margins

我们首先形式化几何间隔函数。给定训练样本 $(x^{(i)}, y^{(i)})$，我们定义 w, b 的间隔函数：

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

注意，如果 $y^{(i)} = 1$，为了使得间隔函数更大（为了我们的预测更有把握和正确），我们需要使得 $w^T x + b$ 是一个较大的正数。相反，如果 $y^{(i)} = -1$，

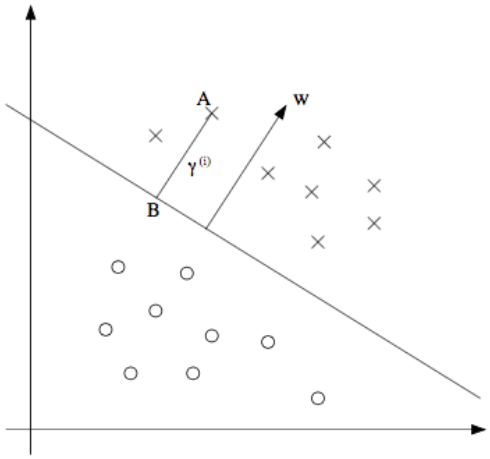
为了使得间隔函数更大（为了我们的预测更有把握和正确），我们需要使得 $w^T x + b$ 是一个较大的负数。那么如果 $y^{(i)}(w^T x + b) > 0$，则我们在训练集上的预测就是正确的。（check this yourself）因此，一个大的间隔函数代表着一个有把握的准确的预测。

然而，对于带有上面的 g 的一个线性的分类器，间隔函数的一个特点使得它不能对有把握"confidence"作一个很好的测量。如果我们在 g 上将 w 替换为 2w，b 替换为 2b，由于 $g(w^T x + b) = g(2w^T x + 2b)$，这就不能够改变 $h_{w,b}(x)$，g，因此 $h_{w,b}(x)$ 是依赖于符号（sign）而不是 $w^T x + b$ 的大小。但是将 (w, b) 替换为 $(2w, 2b)$ 会使得间隔函数乘以 2，这样就可以扩大间隔函数却不改变任何有意义的值。因此加入一些诸如 $\|w\|_2 = 1$ 的正规化条件就显得有意义了。例如，我们可以将 (w, b) 替换为 $(w/\|w\|_2, b/\|w\|_2)$，然后考虑带有 $(w/\|w\|_2, b/\|w\|_2)$ 的间隔函数。我们之后会回来继续讨论它。

给定一个训练集 $S = \{(x^{(i)}, y^{(i)}); i = 1, ..., m\}$，并定义关于 S 的带参数 (w, b) 的间隔函数为该训练样本的最小的间隔函数。因此用 $\hat{\gamma}$ 来表示：

$$\hat{\gamma} = \min_{i=1,...,m} \hat{\gamma}^{(i)}$$

接下来，我们讨论一下几何间隔（geometric margins）。考虑下面的图片：



对应 (w, b) 的决策边界如上，以及向量 w。向量 w 与该条 separating hyperplane 垂直。考虑 A 点，它代表的是一些训练样本中的输入值 $x^{(i)}$，并且标签 $y = 1$。它离决策边界的距离就是 AB 线段的长度。

那么我们怎样才能找到 $\gamma^{(i)}$? 易得 $w/\|w\|$ 是 w 方向上的单位向量。由于 A 代表的是 $x^{(i)}$，那么点 B 就是 $x^{(i)} - \gamma^{(i)} \cdot w/\|w\|$，由于该点坐落在决策边界上，所有该边界上的点都满足等式 $w^T x + b = 0$ ，因此:

$$w^T(x^{(i)} - \gamma^{(i)}\frac{w}{\|w\|}) + b = 0$$

解出 $\gamma^{(i)}$ 就等于:

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

这是对于图中在 A 上的一个正向训练样本的情况是奏效的。更通用地，我们定义对于一个训练样本的 (w, b) 的几何间隔:

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

注意，如果 $\|w\| = 1$ 那么这个间隔函数就等于几何间隔——因此给了我们两种不同的表示间隔的符号的方法。类似的，哪怕重新调节参数几何间隔也是不变的，例如我们将 w 替换为 2w，b 替换为 2b，那么集合间距并不会改变。具体地说，由于调节参数的不变性，当尝试去拟合训练数据的 w 和 b 时，我们便能够在 w 上随意得调节参数，而不影响关键的数据。例如说我们想要 $\|w\| = 1$ 或者 $\|w_1\| = 5$ 或者 $\|w_1 + b\| + \|w_2\| = 2$，这些都可以通过调节 w 和 b 满足。

最后，给定一个训练集 $S = \{(x^{(i)}, y^{(i)}); i = 1, ..., m\}$，我们同样可以定义在 S 上的带参数 (w, b) 的集合间隔使其为在单个训练样本上的最小的几何间隔:

$$\gamma = \min_{i=1,...,m} \gamma^{(i)}$$

1.4 最大间隔分类器 The optimal margin classifier

给定一个训练集，根据之前讲的我们自然的想要的就是找到一个最大化（几何）间隔的决策边界，因为它可以反映出在训练集上预测的把握性以及对训练数据的好的拟合。具体地说就是这会得到一个将正向训练样本和负向训练样本用一个“gap”（几何间隔）分离开的分类器。

现在我们将假定有一个线性分离的训练集，就是说用一些 separating hyperplane 很容易将正向与负向的样本分开。那么我们如何寻找到一条能够得到最大几何间隔的线呢? 我们可以设定下面的最优化问题:

$$
\begin{aligned}
max_{\gamma,w,b} \quad & \gamma \\
& s.t.\ y^{(i)}(w^T x^{(i)} + b) \geq \gamma, \quad i = 1, ..., m \\
& \|w\| = 1
\end{aligned}
$$

例如我们要最大化 γ，对应于每一个带有至少为 γ 的边缘函数的训练样本。$\|w\| = 1$ 的限制更是保证了间隔函数与几何函数的相等，因此我们也能够保证所有几何间隔至少为 γ。

因此，解决这个问题将可以得到关于训练集在 (w, b) 上的的最大可能的的几何边界。

如果上面能够解出来的话，那我们早就做好了。但是“$\|w\| = 1$”限制是一个严重的（非凸的），并且这个问题事实上不是任何一种形式（this problem certainly isn't in any format that we can plug into standard optimization software to solve。）因此，我们将算法转变成更好的方法，考虑：

$$
\begin{aligned}
max_{\hat{\gamma},w,b} \quad & \frac{\hat{\gamma}}{\|w\|} \\
& s.t.\ y^{(i)}(w^T x^{(i)} + b) \geq \hat{\gamma}, \quad i = 1, ..., m \\
& \|w\| = 1
\end{aligned}
$$

然后我们最大化 $\hat{\gamma}/\|w\|$，使得对于所有的训练数据，间隔函数值至少为 $\hat{\gamma}$。由于函数间隔与几何间隔之间的关系 $\gamma = \hat{\gamma}/\|w\|$，这样子就给了我们想要的。我们还摆脱了限制 $\|w\| = 1$ 的限制。现在我们又有一个不那么方便的对象（非凸）$\frac{\hat{\gamma}}{\|w\|}$ 函数，并且我们依然没有任何“off-the-shelf”的软件可以解决这种形式的最优化问题。

记得之前讲到如果我们在 g 上改变 w 和 b 的大小并不会造成什么变化。这里就要用到这个关键的想法。我们引入 w 和 b 的限制使得：

$$\hat{\gamma} = 1$$

由于扩大 w 和 b 会导致函数间隔扩大，这就是扩大限制并且能够通过重现调整 w 和 b 来满足。将其考虑到我们的问题中就可以知道最大化 $\hat{\gamma}/\|w\| = 1/\|w\|$ 和最小化 $\|w\|^2$ 是一样的，因此我们转换成下面的最优化问题：

$$\begin{aligned} min_{\gamma,w,b} \quad & \frac{1}{2}\|w\|^2 \\ s.t. \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, ..., m \end{aligned}$$

现在我们将问题转换为一个能够被高效处理的问题。上面的最优化问题是一个凸二次规划问题并且只有线性限制。它的解就是我们的最大间隔分类器。该最优化问题可以使用 QP（quadratic programming）来解决。

接下来我们要讨论一个话题叫做拉格朗日对偶。它会得到我们最优化问题的对偶形式，它在我们利用核算法高效地得到高纬空间下最大间隔分类器上起了很关键的作用。对偶形式还给我们提供了解决上述最优化问题的一个有效的算法，并且该算法要比普通的 QP 程序要好。

1.5 拉格朗日对偶性 Lagrange duality

这一节先介绍一下拉格朗日对偶性，关于解决限制最优化问题。

考虑下面形式的问题:

$$\begin{aligned} min_w \quad & f(w) \\ s.t. \quad & h_i(w) = 0, \quad i = 1, ..., l \end{aligned}$$

我们定义拉格朗日 Lagrangian 为:

$$\mathcal{L}(w, \beta) = f(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

这里的 β_i 称为是拉格朗日乘子（Lagrange multipliers）。然后我们令 $\mathcal{L}$ 的偏导数为 0:

$$\frac{\partial \mathcal{L}}{\partial w_i} = 0; \frac{\partial \mathcal{L}}{\partial \beta_i} = 0$$

然后解出 w 和 β。

在这一节中，我们将其泛化为限制最优化问题，这里有等式限制的同时还有不等式限制。由于时间关系，在课堂上不会有拉格朗日对偶性的证明，我们只给出主要的结论以及如何应用到我们的最大间隔分类器的最优化问题上。

考虑下面的原始最优化问题:

$$\begin{aligned} min_w \quad & f(w) \\ s.t. \quad & g_i(w) \leq 0, \quad i = 1, ..., k \\ & h_i(w) = 0, \quad i = 1, ..., l \end{aligned}$$

为了解决它，我们引入广义拉格朗日函数：

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

这里的 α_i β_i 称为是拉格朗日乘子（Lagrange multipliers），考虑下面的等式：

$$\theta_{\mathcal{P}}(w) = \max_{\alpha, \beta: \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

这里的 $\mathcal{P}$ 下标代表的是“原始的”。给定一些 w，如果 w 违反了任何的初始限制（例如，对一些 i，$g_i(w) > 0$ 或者 $h_i(w) \neq 0$）那么应该能够验证到：

$$\begin{aligned} \theta_{\mathcal{P}}(w) &= \max_{\alpha, \beta: \alpha_i \geq 0} f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w) \\ &= \infty \end{aligned}$$

反之，如果一个限制确实被一个特定的 w 满足，那么 $\theta_{\mathcal{P}}(w) = f(w)$，因此：

$$\theta_{\mathcal{P}}(w) = \begin{cases} f(w) \text{ 如果 w 满足原始问题约束} \\ \infty \end{cases}$$

因此，对于问题中所有满足原始约束的 w 的值，$\theta_{\mathcal{P}}$ 都取同样的值，并且如果约束失效，那么就是正无穷。因此我们考虑极小化问题：

$$\min_w \theta_{\mathcal{P}}(w) = \min_w \max_{\alpha, \beta: \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

我们发现这个原始的问题是一样的。为了之后的方便，我们定义对象 $p^* = min_w \theta_{\mathcal{P}}(w)$ ，我们称它为原始问题的值。

接着我们看一个不同的问题，我们定义：

$$\theta_{\mathcal{D}}(\alpha,\beta) = \min_{w} \mathcal{L}(w,\alpha,\beta)$$

这里的 $\mathcal{D}$ 代表的是对偶。注意，在原始问题中的定义我们对 α,β 最优化（最大化），这里是对 w 最小化。

我们可以展示对偶最优问题：

$$\max_{\alpha,\beta:\alpha_i\geq 0} \theta_{\mathcal{D}}(\alpha,\beta) = \max_{\alpha,\beta:\alpha_i\geq 0} \min_{w} \mathcal{L}(w,\alpha,\beta)$$

除了 max 与 min 交换了一下外与之前的原始问题类似。我们同样定义了对偶问题对象的最优值为 $d^* = max_{\alpha,\beta:\alpha_i\geq 0}\theta_{\mathcal{D}}(w)$

然而

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

注意 max min 总是小于或等于 min max，然后在这样的条件下，我们有

$$d^* = p^*$$

因此我们可以根据原始问题解决对偶问题。接下来看看这些条件是什么?

假设 f 和 g 是凸的，并且 h_i 是线性。假设限制的 g_i 是可见的，这就意味着对于所有的 i 存在一些 w 使得 $g_i(w) < 0$。

在上面的假设之下，存在着 w^*,α^*,β^* 以致于 w^* 是原始问题的解，而 α^*,β^* 是对偶问题的解，并且 $p^* = d^* = \mathcal{L}(w^*,\alpha^*,\beta^*)$。并且 w^*,α^*,β^* 满足 KKT（Karush-Kuhn-Tucker）条件，

$$\begin{aligned}
\frac{\partial}{\partial w_i}\mathcal{L}(w^*,\alpha^*,\beta^*) &= 0, \quad i = 1,...,n \\
\frac{\partial}{\partial \beta_i}\mathcal{L}(w^*,\alpha^*,\beta^*) &= 0, \quad i = 1,...,l \\
\alpha_i^* g_i(w^*) &= 0, \quad i = 1,...,k \\
g_i(w^*) &\leq 0, \quad i = 1,...,k \\
\alpha^* &\geq 0, \quad i = 1,...,k
\end{aligned}$$

如果存在 w^*, α^*, β^* 满足 KKT 条件，那么就有一组原始问题和对偶问题的解。

注意等式 $\alpha_i^* g_i(w^*) = 0, \quad i = 1, ..., k$ 叫做 KKT 对偶互补条件。具体地说，它的意思是，如果 $\alpha_i^* > 0$，那么 $g_i(w^*) = 0$。这是说明 SVM 只有一些支持向量是很关键的；当讨论在 SMO 算法时，KKT 对偶互补条件也会提供我们收敛测试。

1.6 最大间隔分类器 Optimal margin classifiers

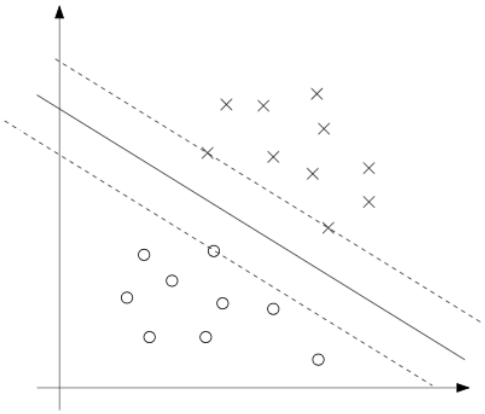
之前我们已经知道了以下的最优化问题来找到最大间隔分类器：

$$\begin{aligned} min_{\gamma,w,b} \quad & \frac{1}{2}\|w\|^2 \\ s.t. \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, ..., m \end{aligned}$$

我们可以将约束写成：

$$g_i(w) = -y^{(i)}(w^T x^{(i)} + b) + 1 \leq 0$$

对于每一个训练样本为就有一个这样的约束。注意，从 KKT 对偶互补条件中，我们只对那些函数间隔等于 1 的训练样本有 $\alpha_i > 0$。考虑到下面的图，实线表示的就是一个最大的间隔分解线。



最小的间隔的点就是那些离决策边界最近的点（图中有三个点），它们坐落在虚线表示的决策边界上。因此，只有三个 α_i——也就是对应这三个训练样例——在外面等最优化问题中在最优解上是非零的。这三个点就叫

做这个问题的支持向量（support vectors）。有一个事实在后面非常有用，它就是支持向量的数量要比训练集的数量少得多。

我们继续，我们会用到这个问题的对偶形式，其中需要小心的是我们会就内积 $< x^{(i)}, x^{(j)} >$ 尝试在输入特征空间中的点之间去写出我们的算法。就这些内积表示出的算法在核函数中也是挺关键的。

我们为我们的最优化问题构造拉格朗日函数：

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

注意到这里只有 α_i 拉格朗日乘子，没有 β_i，因为该问题只有不等式约束。

让我们寻找该问题的对偶形式。我们首先需要最小化关于 w, b（固定的 α）的 $\mathcal{L}(w, b, \alpha)$ 来得到 $\theta_{\mathcal{D}}$，那么只需求其偏导，并令它为 0:

$$\nabla_w \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0$$

这就求出了

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

对 b 求偏导，我们有：

$$\frac{\partial}{\partial b}\mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

我们将得到的 w 的定义代入拉格朗日函数，我们就能得到：

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} - b \sum_{i=1}^{m} \alpha_i y^{(i)}$$

从对 b 的偏导的结果得知，拉格朗日函数最后一项为 0，因此，我们有

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)}$$

同时记得最小化关于 w 和 b 的 $\mathcal{L}(w, b, \alpha)$，再加上约束 $\alpha_i \geq 0$ 以及 $\frac{\partial}{\partial b}\mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$，我们就能得到以下的对偶最优化问题：

$$max_\alpha \;\; W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j < x^{(i)}, x^{(j)} >$$
$$s.t. \;\; \alpha_i \geq 0, \;\; i = 1, ..., m$$
$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

你应该能够验证 $p^* = d^*$ 和 KKT 条件在我们的最优化问题下是满足的。因此我们可以解决对偶问题而不用解决原始问题。具体地说就是在上面的对偶问题中，我们有一个参数为 α_i 的最大化问题。至于解决该问题的具体算法在之后会给出，但是如果我们却是能解决它，那么我们就能使用 $w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$ 来找到作为 α 函数的原始的 w。通过考虑原始问题找到 w^*，那么原始问题中的截距项也能够找到：

$$b^* = -\frac{max_{i:y^{(i)}=-1}(w^*)^T x^{(i)} + min_{i:y^{(i)}=1}(w^*)^T x^{(i)}}{2}$$

(Check for yourself that this is correct.)

在继续之前，我们再多关注一下：

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

给出了 α 的最优值 w。假设我们模型拟合了一个训练集，然后想要对一个新的输入值做一个预测。我们就会计算出 $w^T x + b$，然后预测 y=1 当且仅当等式大于 0。但是利用上面的等式，我们还能写成：

$$w^T x + b = (\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)})^T x + b = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} < x^{(i)}, x > + b$$

因此，如果我们找到了 α_i，为了能够预测，我们需要计算一个等式，它依赖于在 x 和训练集中点的内积。我们还知道 α_i 将总为 0，除了支持向量上。因此上面和中的许多项都为 0。然后我们只需要找到内积与支持向量之间的关系，然后作出预测。

通过检测最优化问题的对偶形式，我们最问题的结构更敏锐，并且能就 x 和支持向量间的内积写出整个算法，在下一节中，我们会探索应用核函数到分类问题中去，然后学习一个在高维空间上有效率运行的算法——支持向量机。

1.7 核 Kernels

回顾之前的线性回归的讨论，我们有一个预测问题，它的输入 x 是房屋的居住面积。然后我们利用特征 x, x^2, x^3 来得到一个三次方程。为了区分这两类变量，我们称这个问题的输入属性（input attributes）为原始输入（在房屋的例子中就是居住面积 x）。然后它们会被映射到一些新的数量集中，然后应用到学习算法中去。我们称这些数为输入特征（input features）。我们用 ϕ 来表示这样的特征映射（feature mapping），也就是从输入属性到输入特征的映射。例如在上面的例子就是：

$$\phi(x) = \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}$$

我们在 SVMs 中并不用原始输入 x，而是利用一些特征 $\phi(x)$ 来学习。因此我们需要将上述的算法中的 x 替换为 $\phi(x)$。由于算法可以写成关于内积 $\langle x, z\rangle$ 的形式，这就意味着我们要将所有的内积替换成 $\langle \phi(x), \phi(z)\rangle$ 的形式。具体地说就是给定一个特征映射 ϕ，我们定义对应的内核函数（Kernel）为：

$$K(x, z) = \phi(x)^T \phi(z)$$

这样我们就可以将之前算法中的 $\langle x, z\rangle$ 替换为 $K(x, z)$，然后我们的算法就可以用这些特征 ϕ 来学习了。

现在给定 ϕ，我们可以通过找到 $\phi(x)$ 和 $\phi(z)$ 并取它们的内积来计算 $K(x, z)$。但是通常更有趣的是，尽管 $\phi(x)$ 计算成本很高（也许是因为它是一个非常高纬的向量），但 $K(x, z)$ 可能比较好计算。在这样的设定下，通过在我们的算法中利用一个有效的方式来计算 $K(x, z)$，我们就能得到 SVMs 在给定的 ϕ 的高纬特征空间中学习，但并没有找到向量 $\phi(x)$。

让我们看一个例子。假设 $x, z \in \mathbb{R}^n$，然后考虑：

$$K(x, z) = (x^T z)^2$$

同样可以将其写成：

$$
\begin{aligned}
K(x,z) &= (\sum_{i=1}^{n} x_i z_i)(\sum_{j=1}^{n} x_j z_j) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j z_i z_j \\
&= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j)
\end{aligned}
$$

因此我们可以看到 $K(x,z) = \phi(x)^T \phi(z)$，映射关系 ϕ（这里给出 n=3 的情况）为:

$$
\phi(x) = \begin{bmatrix}
x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3
\end{bmatrix}
$$

注意计算高纬的 $\phi(x)$ 需要 $\mathcal{O}(n^2)$，找到 $K(x,z)$ 只需要 $\mathcal{O}(n)$，也就是输入属性的维度的线性时间。

对于相关的核，考虑 (Check this yourself):

$$
\begin{aligned}
K(x,z) &= (x^T z + c)^2 \\
&= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) + \sum_{i=1}^{n} (\sqrt{2c} x_i)(\sqrt{2c} z_i) + c^2
\end{aligned}
$$

对应的特征映射就是

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \\ \sqrt{2c}x_1 \\ \sqrt{2c}x_2 \\ \sqrt{2c}x_3 \\ c \end{bmatrix}$$

其中参数 c 控制着在 x_i 和 x_ix_j 项之间的权重。

更广义地，内核 $K(x,z) = (x^Tz + c)^d$ 对应的特征映射到一个 $\binom{n+d}{d}$ 的特征空间，对应所有从 $x_{i1}, x_{i2}, ..., x_{ik}$ 到 d 的形式。尽管在这样的 $\mathcal{O}(n^d)$ 维空间，计算 $K(x,z)$ 只需要 $\mathcal{O}(n)$ 时间，因此我们根本不需要将高纬的向量表示出来。

现在，我们再讨论一些核不一样的观点。直觉上我们也许认为如果 $\phi(x)$ 和 $\phi(z)$ 很相近，我们也许期望 $K(x,z) = \phi(x)^T\phi(z)$ 会很大。相反，如果 $\phi(x)$ 和 $\phi(z)$ 差的较远，例如都快互相垂直，那么 $K(x,z) = \phi(x)^T\phi(z)$ 会很小。因此我们就能够认为 $K(x,z)$ 可以度量 $\phi(x)$ 和 $\phi(z)$ 之间有多近，x 和 z 有多相似。

鉴于这一设想，假设你在研究的一些学习问题，你想到了一些函数 $K(x,z)$ 可以测量 x 和 z 有多相似，例如你选择：

$$K(x,z) = exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$$

这是一个衡量 x 和 z 之间相似程度的表示，如果结果接近 1，那就代表 x 和 z 很接近，反之如果结果 x 和 z 很远，那么接近 0。在 SVM 中，我们能够将 K 的定义作为核吗？在这个特殊的情况下是可以的。（核称作为高斯核 Gaussian kernel，并且对应一个无限维特征映射 ϕ）但是给定函数 K，

我们怎么分辨它是有效的核呢? 例如，我们能够识别是否有一些特征映射 ϕ 满足 $K(x,z) = \phi(x)^T\phi(z)$?

假设现在的 K 的确是一个有效的核，对应着一些 ϕ 上的映射。现在考虑有限的 m 个点集（不是所有的训练样本）$\{x^{(1)}, ..., x^{(m)}\}$。令一个矩阵 K 为 $m \times m$ 由 $K_{ij} = K(x^{(i)}, x^{(j)})$。这个矩阵叫做核矩阵（Kernel matrix）。注意，我们要覆盖标记并利用 K 来代表函数 $K(x,z)$ 的核以及核矩阵 K。

现在, 如果 K 是有效的核, 那么 $K_{ij} = K(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T\phi(x^{(j)}) = \phi(x^{(j)})^T\phi(x^{(i)}) = K(x^{(j)}, x^{(i)}) = K_{ji}$，因此 K 必须是对称的。令 $\phi_k(x)$ 代表向量 $\phi(x)$ 的第 k 个坐标，我们从 z 开始找我们能找到:

$$\begin{aligned}
z^T K z &= \sum_i \sum_j z_i K_{ij} z_j \\
&= \sum_i \sum_j z_i \phi(x^{(i)})^T \phi(x^{(j)}) z_j \\
&= \sum_i \sum_j z_i \sum_k \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j \\
&= \sum_k \sum_i \sum_j z_i \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j \\
&= \sum_k \left(\sum_i z_i \phi_k(x^{(i)}) \right)^2 \\
&\geq 0
\end{aligned}$$

第二步到最后一步使用的技巧在 Problem set 1 Q1 上的是一样的。因为 z 是任意的，那么 K 就是半正定的（positive semi-definite，$K \geq 0$）。

因此，我们可以知道，如果 K 是一个有效的核（例如对应一些特征映射 ϕ），那么对应的核矩阵 $K \in \mathbb{R}^{m \times m}$ 是对称半正定。更通用地说，这就证明出了 K 是有效核的一个充要条件（也叫做 Mercer Kernel）。

Mercer 定理 1.1 令 $K : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$。那么 K 为一个有效核的充要条件是对于任意的 $\{x^{(1)}, ..., x^{(m)}\}$ $m < \infty$，对应的核矩阵是对称半正定的。

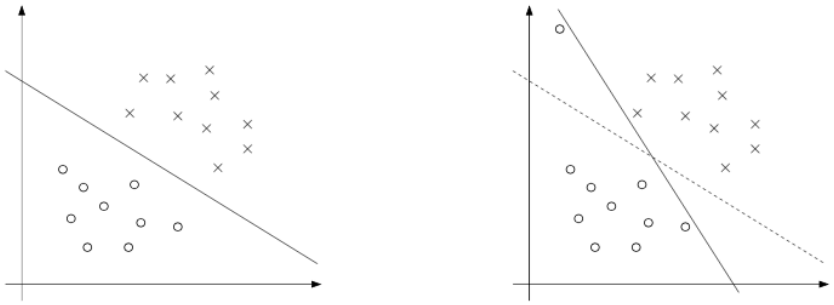
给定一个函数 K，而非去寻找它的特征映射函数 ϕ，该定理给出了另一种判断它是否是一个有效核的方法。在 problem set 2 中会有练习。

在课堂上，我们也讨论了一些其他的核函数。例如数字识别问题，给定一个手写的数字图像（16×16 像素），我们需要去分辨出是哪一个数字。可

以使用多项式和函数 $K(x,z) = (x^T z)^d$ 或者是高斯核函数，支持向量机在这个问题上的效果是非常好的。同时这也是非常令人惊讶的，因为这里的输入属性 x 仅仅是一个图像像素的 256 维的向量密度值，并且系统对其他信息一无所知，甚至是对像素旁边是哪个像素也不知道。另一个在课堂上提到的问题是，当我们想要分类的对象 x 是字符串时，（例如 x 是一个氨基酸序列）那么似乎对于大多数学习算法而言很难去构建一个合理的小型的特征集，尤其当不同的字符串还有着不同的长度。然而，令 $\phi(x)$ 是一个计数着在 x 中每一个长度为 k 的子串出现的次数的特征向量。若考虑英文字符串，则有 26^k 个这样的字符串。因此，$\phi(x)$ 是一个 26^k 维的向量；哪怕是适当的 k 值，这个值也是非常庞大的，以致于很难有效的进行计算。当然，利用（动态规划）字符串匹配算法可能有效地计算出 $K(x,z) = \phi(x)^T \phi(z)$，即我们能够隐形地在庞大的维数下计算，而不是在特征向量空间中显性地计算。

核函数在支持向量机下的应用应该已经讲的比较清楚了，因此我们不会在这上面逗留太久。记住，核的思想的应用要远远超出支持向量机。具体地说，如果你有任何学习算法，并且在输入属性之中只有内积 $\langle x,z\rangle$，那么你就可以将其替换为核 $K(x,z)$，这样你就能够神奇地使你的算法在高维特征空间下高效地执行。例如，核技巧可以与感知机来得到一个核感知算法。这类在之后的课堂上也会提到，这种方法一贯被称为“核技巧”。

1.8 正规化与不可分离情况 Regularization and the non-separable case



迄今为止的 SVM 的数据被认为是线性可分离的。通过 ϕ 来将数据映射到一个高维的特征空间通常可以提高数据可分离的概率，但是我们不能保证它总是能够做到。并且，在某些情况下寻找一个分离超平面也许不是我们想要做的，因为很容易被异常值影响。例如上面的左图是最大间隔分类，

而当有一个异常点出现在左上处时（右图），它就使得决策边界夸张地旋转了，并且得到的是一个非常小的间隔。

为了使得算法可以对非线性分离的数据集也有效，并且能够不太被异常数据影响，我们给出优化的最优化问题（利用 ℓ_1 正规化）：

$$\begin{aligned} max_{\gamma,w,b} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ s.t. \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, ..., m \\ & \xi_i \geq 0, \quad i = 1, ..., m \end{aligned}$$

因此，样例现在允许（函数）间隔小于 1，如果一个样例有函数间隔 $1 - \xi_i$（$\xi > 0$），那么我们就得损失一个对象的代价函数，通过增加 $C\xi_i$。参数 C 控制着一个权重，该权重用来维持使 $\|w\|^2$ 变小（之前时使得间隔变大）以及确保大多数的样本有函数间隔至少为 1。

跟之前一样，我们可以构造拉格朗日函数：

$$\mathcal{L}(w, b, \xi, \alpha, r) = \frac{1}{2}w^T w + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i[y^{(i)}(x^T w + b) - 1 + \xi_i] - \sum_{i=1}^{m} r_i\xi_i$$

这里的 α_i 和 r_i 是我们的拉格朗日乘子（被约束为 ≥ 0）。这里我们不再给出对偶求导的过程，但是在令关于 w 和 b 求导后式子为 0，和以前一样带入拉格朗日函数，然后简化，我们就能够得到下面形式的对偶问题：

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

跟以前一样，w 可以用 α_i 来表示，即 $w = \sum_{i=1}^{m}\alpha_i y^{(i)} x^{(i)}$，这样在解完对偶问题之后，我们就能利用公式

$$w^T x + b = (\sum_{i=1}^{m}\alpha_i y^{(i)} x^{(i)})^T x + b = \sum_{i=1}^{m}\alpha_i y^{(i)}\langle x^{(i)}, x\rangle + b$$

来做预测。注意，惊讶的是，在引入 ℓ_1 正规化，对对偶问题的唯一的改变是约束 $0 \leq \alpha_i$ 变成了 $0 \leq \alpha_i \leq C$，b^* 的值也不得不改变（之前的 b^* 失效）；请看下一节 Platt's paper 的注释。

还有，KKT 对偶互补条件变成（在下一节中对测试 SMO 算法的收敛会用到）：

$$\begin{aligned}\alpha_i = 0 &\Rightarrow y^{(i)}(w^T x^{(i)} + b) \geq 1 \\ \alpha_i = C &\Rightarrow y^{(i)}(w^T x^{(i)} + b) \leq 1 \\ 0 < \alpha_i < C &\Rightarrow y^{(i)}(w^T x^{(i)} + b) = 1\end{aligned}$$

那么现在就是要给出一个算法来真正解决对偶问题，在就是下一节要讲的。

1.9 序列最小最优化算法 The SMO algorithm

序列最小最优化算法，即 SMO（sequential minimal optimization）算法是由 John Platt 提出的一个有效的算法来解决 SVM 衍生出来的对偶问题。在这之前又要先接触一个题外话——坐标上升算法（Coordinate ascent）。

1.9.1 Coordinate ascent

考虑尝试去解决一个无约束最优化问题：

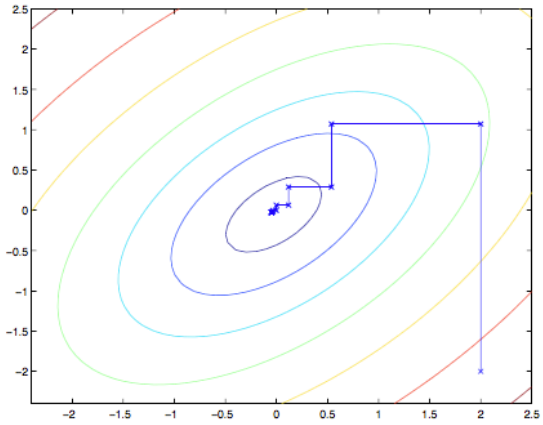
$$\max_{\alpha} \ W(\alpha_1, \alpha_2, ..., \alpha_m)$$

这里我们假设 W 是一个关于参数 α_i 的函数，并且这里忽略我们之前提到的该问题与 SVMs 之间的关系。我们已经接触过两个最优化算法来了，它们是梯度上升（下降）和牛顿方法。这里我们要接触的新算法叫做坐标上升算法（coordinate ascent）：

Loop until convergence: {
for i=1 to m,{
$\alpha_i := argmax_{\hat{\alpha_i}} W(\alpha_1, ..., \alpha_{i-1}, \hat{\alpha}_i, \alpha_{i+1}, ..., \alpha_m)$
}
}

因此，在最里面的循环中，我们可以使除了 α_i 的变量固定，然后重新最优化关于参数 α_i 的 W。在这个方法的版本中，最里面的循环按次序最优化变量（另一种更老练的版本是选择其他的次序；例如根据下一个变量选哪个可以使得在 W(α) 中能最大化提高）。

当 W 碰巧是内循环中的那种形式（“arg max）的时候，该算法则是一个相当有效的算法。这里有一副该算法计算过程的图片：



图中的椭圆是一个我们要最优化的二次方程的轮廓。坐标上身初始化的点在 (2,-2)，图中给出了它最终走向全局最大值的大致路径。注意，该算法的走的每一步的路径都与坐标轴平行，也就是说，在同一时刻只有一个变量在最优化。

1.9.2 SMO

在结束 SVM 的讨论之前，我们得讲述 SMO 算法。一些细节将在作业中涉及。

下面则是我们要解决的（对偶）最优化问题：

$$max_\alpha \quad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \tag{1}$$

$$s.t. \quad 0 \leq \alpha_i \leq C, \quad i = 1, ..., m \tag{2}$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \tag{3}$$

我们说，我们有一组 α_i 满足约束 (2-3)，现在，假设我们令 $\alpha_2, ..., \alpha_m$ 保持不变，然后执行一次坐标上升步骤并重新优化关于 α_1 的对象。我们能取得一些进步吗？答案是不能，由于约束 (3) 确保了：

$$\alpha_1 y^{(1)} = -\sum_{i=2}^{m} \alpha_i y^{(i)}$$

或者，我们在等式两边都乘上 $y^{(1)}$ 得到

$$\alpha_1 = -y^{(1)} \sum_{i=2}^{m} \alpha_i y^{(i)}$$

（这一步用到了一个事实——$y^{(1)} \in \{-1, 1\}$，因此 $(y^{(1)})^2 = 1$，因此，如果我们保持 $\alpha_2, ..., \alpha_m$ 不变的话，α_1 其实是由其 α_i 来决定的，那么在该最优化问题中，如果我们不违反约束 (3) 的话，我们就不能够改变 α_1 的值。

因此，如果我们想要更新一些 α_i，那么就必须同时更新两个以上以保证满足约束。这就促动了 SMO 算法：

Repeat till convergence {
1. 挑选一些要更新的参数对 α_i, α_j（利用启发式算法，使得可以让我们能够得到最大的进步而趋向于全局的最大值。）
2. 保持其他 α_k $k \neq i, j$ 不变，（反复优化关于 α_i, α_j 的（$W(\alpha)$
}

测试该算法的收敛，我们可以检查在一些 $\sqcup\!\updownarrow$. 内是否满足 KKT 条件。这里的 $\sqcup\!\updownarrow$. 是收敛容错参数（convergence tolerance parameter），一般设置在 0.01 到 0.001 左右（详见论文与伪代码）。

说 SMO 算法是一个有效算法的关键在于对 α_i, α_j 的更新是十分高效的。现在我们简单概述一下这个高效更新的简单思想。

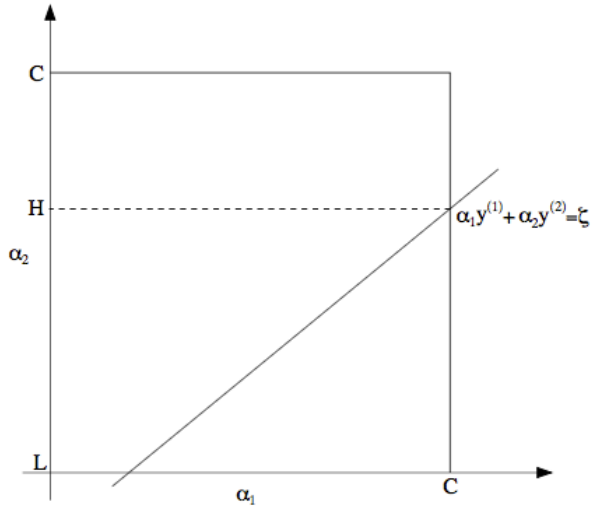
我们先设置一些 α_i 满足约束条件 (2-3)，假设我们决定保持 $\alpha_3, ..., \alpha_m$ 不变，我们想要反复优化（reoptimize）关于 α_1, α_2 的 $W(\alpha_1, \alpha_2, ..., \alpha_m)$，由约束 (3)，我们有：

$$\alpha_1 y^1 + \alpha_2 y^2 = -\sum_{i=3}^{m} \alpha_i y^{(i)}$$

由于右边是固定不变的，那么可以用常亮 ζ 表示：

$$\alpha_1 y^1 + \alpha_2 y^2 = \zeta$$

那么我们就能画出关于 α_1, α_2 的约束函数：



从约束 (2) 中，我们知道 α_1, α_2 肯定在正方形 $[0, C] \times [0, C]$ 之间。同时在直线 $\alpha_1 y^1 + \alpha_2 y^2 = \zeta$ 上。注意，由此我们知道 $L \leq \alpha_2 \leq H$；否则，(α_1, α_2) 就不能够同时满足正方形和直线的约束。在这个例子中，L=0。图中的直线 $\alpha_1 y^1 + \alpha_2 y^2 = \zeta$，但这并不一定是这样的；一般来说，会有一些下界 L 和上界 H 来限制允许的 α_2 的值从而确保 α_1, α_2 坐落在正方形 $[0, C] \times [0, C]$ 之间。

利用等式 $\alpha_1 y^1 + \alpha_2 y^2 = \zeta$ 我们能够得到用 α_2 来表示 α_1 的函数：

$$\alpha_1 = (\zeta - \alpha_2 y^{(2)}) y^{(1)}$$

因此，$W(\alpha)$ 就能写成：

$$W(\alpha_1, \alpha_2, ..., \alpha_m) = W((\zeta - \alpha_2 y^{(2)}) y^{(1)}, \alpha_2, ..., \alpha_m)$$

将 $\alpha_3, ..., \alpha_m$ 视作常数，这样就能看出它其实就是关于 α_2 的一个二次方程。例如，这就可以写成类似于对于合适的 a，b，和 c 有 $a\alpha_2^2 + b\alpha_2 + c$。如果我们忽视约束 (2)，（或者说，$L \leq \alpha_2 \leq H$），那么我们通过求导使得导函数为 0 很容易就能最大化该二次方程。我们令 $\alpha_2^{new,unclipped}$ 代表 α_2 的结果。你应该还要能够证明到如果我们求 W 的最大值并遵从正方形限制，那么我们要能找到最优的结果值，然后使得其在区间 $[L, H]$ 之间，得到：（这边不太会翻译）

$$\alpha_2^{new} = \begin{cases} H & \text{if } \alpha_2^{new,unclipped} > H, \\ \alpha_2^{new,unclipped} & \text{if } L \leq \alpha_2^{new,unclipped} \leq H, \\ L & \text{if } \alpha_2^{new,unclipped} < L. \end{cases}$$

最后找到了 α_2^{new}，然后我们利用等式 $\alpha_1 y^1 + \alpha_2 y^2 = \zeta$ 来找到 α_1^{new} 的最优值。

还有一些简单的细节可以自己阅读 Platt 的论文：一个是用启发式来选择接下来的 α_i, α_j 的更新；其次就是当 SMO 算法在允许时如何更新 b 的值。