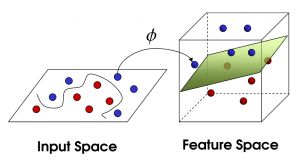
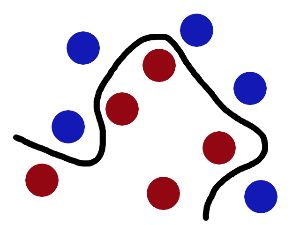
大侠没有棍可以很好帮他分开两种球了，现在怎么办呢？当然像所有武侠片中一样大侠桌子一拍，球飞到空中。然后，凭借大侠的轻功，大侠抓起一张纸，插到了两种球的中间。

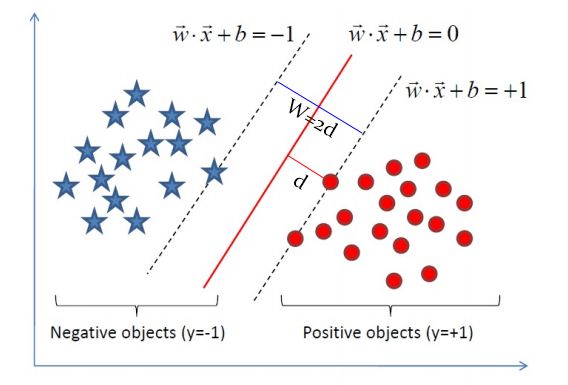


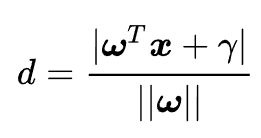
现在，从空中的魔鬼的角度看这些球，这些球看起来像是被一条曲线分开了。



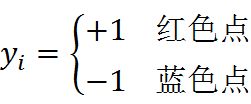
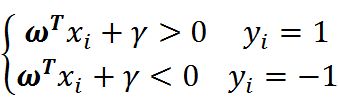
再之后，无聊的大人们，把这些球叫做**data**，把棍子叫做**classifier**, 找到最大间隙的**trick**叫做**optimization**，拍桌子叫做**kernelling,** 那张纸叫做**hyperplane**

**SVM基本原理**

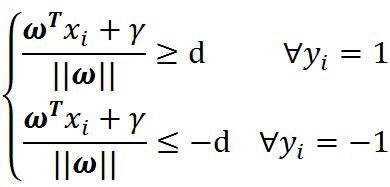


IMG_256

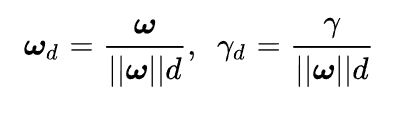
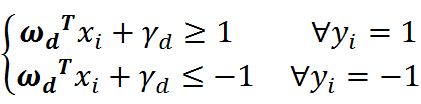
分类器的好坏的评定依据是分类间隔W=2d的大小，即分类间隔W越大，我们认为这个超平面的分类效果越好。此时，求解超平面的问题就变成了求解分类间隔W最大化的为题。W的最大化也就是d最大化。

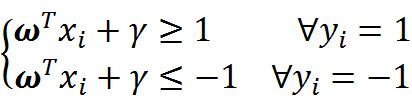
①如果正确分类则有：

假设分类面正好在中间则有：



两边同除d：

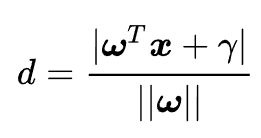


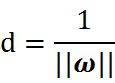
②等效为

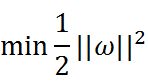
为什么标记为1和-1呢？因为这样标记方便我们将上述方程变成如下形式：

IMG_256

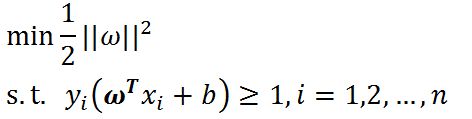
③IMG_256



将我们的目标函数进一步化简：，是凸函数

求解d的最大化问题即为||w||的最小化问题，故等效为，1/2是为凑求导的。

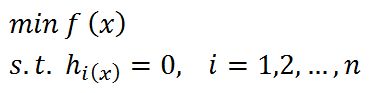
问题等效为：



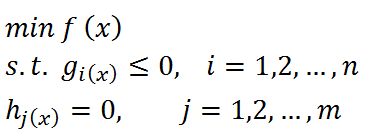
④a》无约束优化问题，可以写为：

IMG_256

b》有等式约束的优化问题，可以写为：



c》有不等式约束的优化问题，可以写为：



对于第(a)类的优化问题，尝尝使用的方法就是费马大定理(Fermat)，即使用求取函数f(x)的导数，然后令其为零，可以求得候选最优值，再在这些候选值中验证；如果是凸函数，可以保证是最优解。这也就是我们高中经常使用的求函数的极值的方法。对于第(b)类的优化问题，常常使用的方法就是拉格朗日乘子法（Lagrange Multiplier) ，即把等式约束h\_i(x)用一个系数与f(x)写为一个式子，称为拉格朗日函数，而系数称为拉格朗日乘子。通过拉格朗日函数对各个变量求导，令其为零，可以求得候选值集合，然后验证求得最优值。对于第(c)类的优化问题，常常使用的方法就是KKT条件。同样地，我们把所有的等式、不等式约束与f(x)写为一个式子，也叫拉格朗日函数，系数也称拉格朗日乘子，通过一些条件，可以求出最优值的**必要条件**，这个条件称为KKT条件。

⑤我们知道我们要求解的是最小化问题，所以一个直观的想法是如果我能够构造一个函数，使得该函数在可行解区域内与原目标函数完全一致，而在可行解区域外的数值非常大，甚至是无穷大，那么这个**没有约束条件的新目标函数的优化问题**就与原来**有约束条件的原始目标函数的优化问题**是等价的问题。这就是使用拉格朗日方程的目的，它将约束条件放到目标函数中，**从而将有约束优化问题转换为无约束优化问题。**

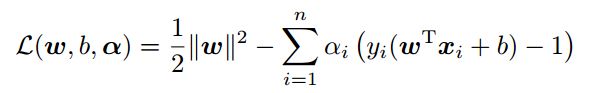
在拉格朗日优化我们的问题这个道路上，**需要进行下面二个步骤：**

（1）将有约束的原始目标函数转换为无约束的新构造的拉格朗日目标函数

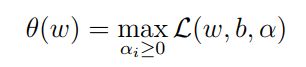
（2）使用拉格朗日对偶性，将不易求解的优化问题转化为易求解的优化

⑤-1**第一步，将有约束的原始目标函数转换为无约束的新构造的拉格朗日目标函数**

构造拉格朗日函数：



其中αi是拉格朗日乘子，αi大于等于0（自己规定），是我们构造新目标函数时引入的系数变量。

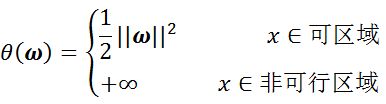
令

当样本点不满足约束条件时，即**在可行解区域外：**IMG_256

此时，要取max，我们将αi设置为正无穷，此时θ(w)显然也是正无穷

当样本点满足约束条件时，即**在可行解区域内：**IMG_256

正如前面所说，显然θ(w)为原目标函数本身。我们将上述两种情况结合一下，就得到了新的目标函数：



此时，再看我们的初衷，就是为了建立一个在可行解区域内与原目标函数相同，在可行解区域外函数值趋近于无穷大的新函数，现在我们做到了。

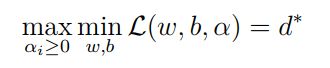
**取min则把可行解区域外的点（θ为无穷大）去掉，即所有点都分类正确。**

**问题等效成为：**

IMG_256

⑤-2 **第二步，将不易求解的优化问题转化为易求解的优化**

看一下我们的新目标函数，先求最大值，再求最小值。这样的话，我们首先就要面对带有需要求解的参数w和b的方程，而αi又是不等式约束，这个求解过程不好做。所以，我们需要使用拉格朗日函数对偶性，将最小和最大的位置交换一下，这样就变成了：



交换以后的新问题是原始问题的对偶问题，这个新问题的最优值用d\*来表示。而且d\*<=p\*。我们关心的是d=p的时候，这才是我们要的解。需要什么条件才能让d=p呢？

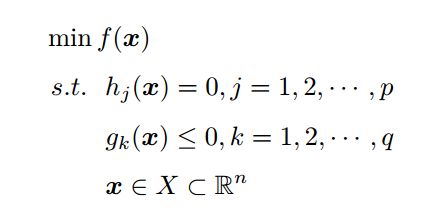
（1）首先必须满足这个优化问题是凸优化问题。

（2）其次，需要满足KKT条件。

凸优化问题的定义是：**求取最小值的目标函数为凸函数的一类优化问题。**目标函数是凸函数我们已经知道，这个优化问题又是求最小值。所以我们的最优化问题就是凸优化问题。

⑤-3 KKT（Karush-Kuhn-Tucker）条件

过一些条件，可以求出最优值的必要条件，这个条件就是接下来要说的KKT条件。一个最优化模型能够表示成下列标准形式：



KKT条件是说最优值条件必须满足以下条件：

1. 条件一：经过拉格朗日函数处理之后的新目标函数L(w,b,α)对x求导为零：
2. 条件二：

（3）条件三： ；

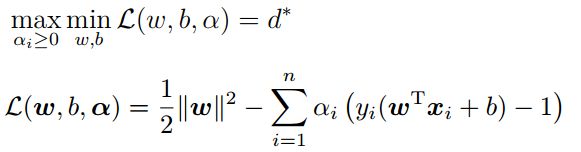
显然，条件二已经满足了。另外两个条件为啥也满足呢？

这里原谅我省略一系列证明步骤，感兴趣的可以移步这里：<https://blog.csdn.net/xianlingmao/article/details/7919597>

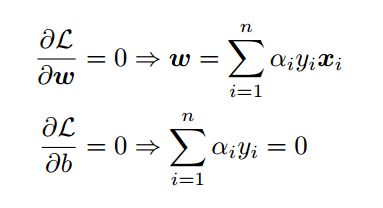
这里已经给出了很好的解释。现在，凸优化问题和KKT都满足了，问题转换成了对偶问题。而求解这个对偶学习问题，可以分为三个步骤：首先要让L(w,b,α)关于w和b最小化，然后求对α的极大，最后利用SMO算法求解对偶问题中的拉格朗日乘子。现在，我们继续推导。

⑥ 对偶问题求解

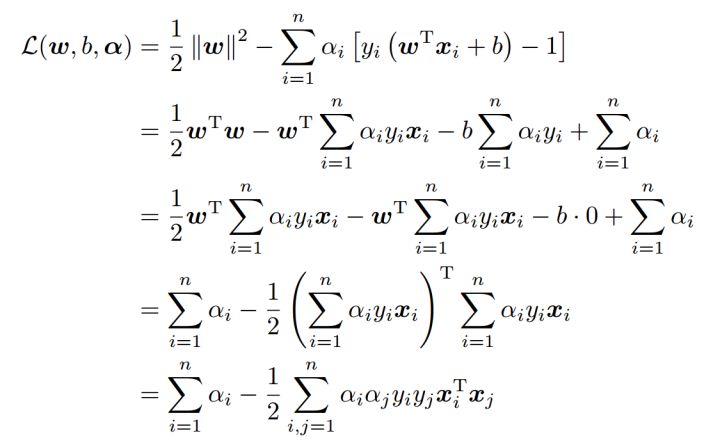
**第一步**



首先固定α，要让L(w,b,α)关于w和b最小化，我们分别对w和b偏导数，令其等于0：



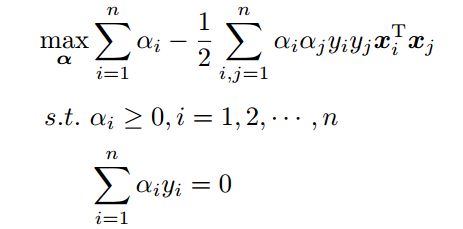
代回L(w,b,α)得到：



此时的L(w,b,α)函数只含有一个变量，即αi。

**第二步：**

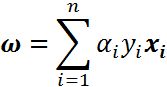
在上一步，内侧的最小值求解完成，我们求解外侧的最大值，从上面的式子得到：

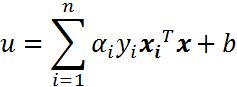


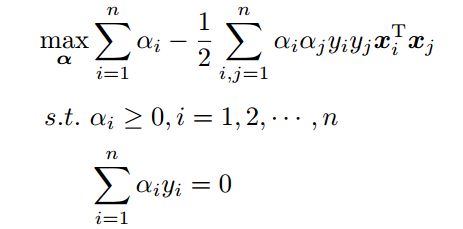
现在我们的优化问题变成了如上的形式。对于这个问题，我们有更高效的优化算法，即序列最小优化（SMO）算法。我们通过这个优化算法能得到α，再根据α，我们就可以求解出w和b，进而求得我们最初的目的：找到超平面，即"决策平面"。

**SMO算法**

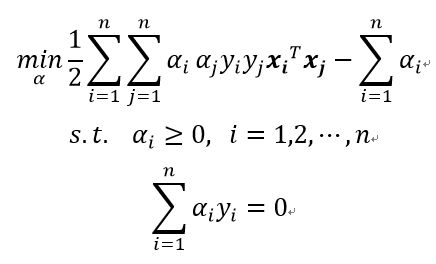
先来定义特征到结果的输出函数为：IMG_256

由上文知

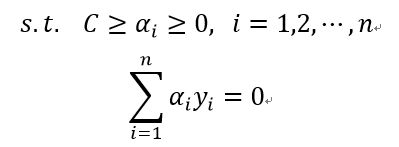
将上述公式带入输出函数中：

拉格朗日对偶后得到最终的目标化函数：

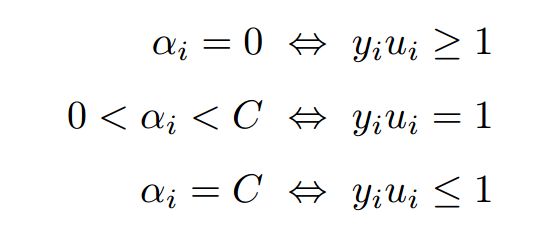
将目标函数变形，在前面增加一个符号，将最大值问题转换成最小值问题：



实际上，对于上述目标函数，是存在一个假设的，即数据100%线性可分。但是，目前为止，我们知道几乎所有数据都不那么"干净"。这时我们就可以通过引入所谓的**松弛变量**(slack variable)，来允许有些数据点可以处于超平面的错误的一侧。这样我们的优化目标就能保持仍然不变，但是此时我们的约束条件有所改变：

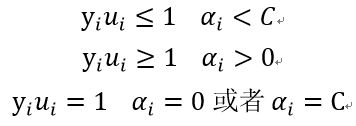


根据KKT条件可以得出其中αi取值的意义为：

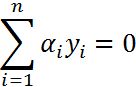


* 对于第1种情况，表明αi是正常分类，在边界线外部；
* 对于第2种情况，表明αi是支持向量，在边界上；
* 对于第3种情况，表明αi是在两条边界之间。

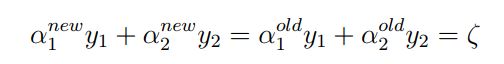
而最优解需要满足KKT条件，即上述3个条件都得满足，以下几种情况出现将会不满足：



也就是说，如果存在不能满足KKT条件的αi，那么需要更新这些αi，这是第一个约束条件。此外，更新的同时还要受到第二个约束条件的限制，即：

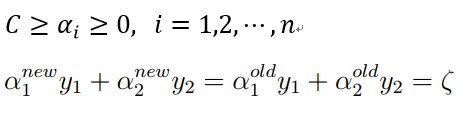


因为这个条件，我们同时更新两个α值，因为只有成对更新，才能保证更新之后的值仍然满足和为0的约束，假设我们选择的两个乘子为α1和α2：

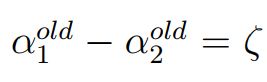
ξ为常数

因为两个因子不好同时求解，所以可以先求第二个乘子α2的解（α2 new），得到α2的解（α2 new）之后，再用α2的解（α2 new）表示α1的解（α1 new ）。为了求解α2 new ，得先确定α2 new的取值范围。假设它的上下边界分别为H和L，那么有：IMG_256

接下来，综合下面两个条件：



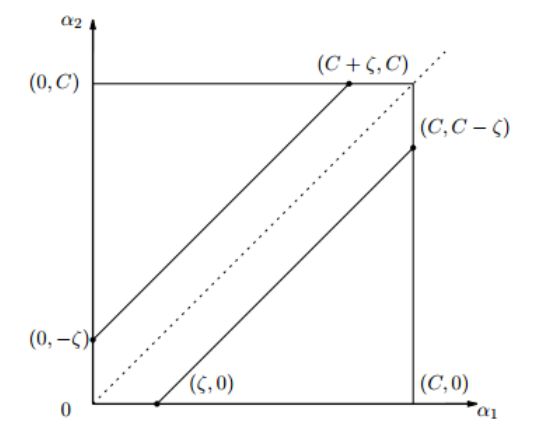
当y1不等于y2时，即一个为正1，一个为负1的时候，可以得到：



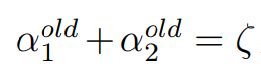
所以有：

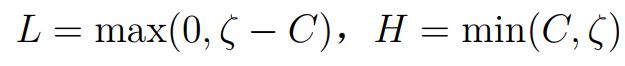
IMG_256

规划范围为，中间那部分区域：

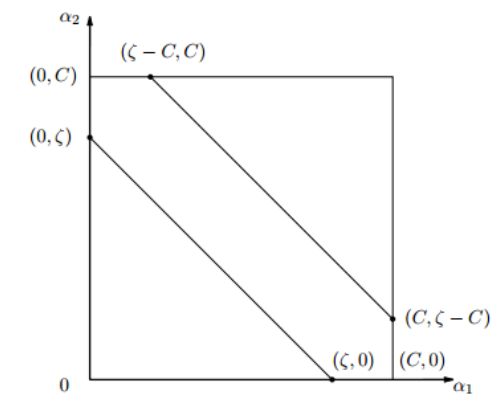


当y1等于y2时，即两个都为正1或者都为负1，可以得到：

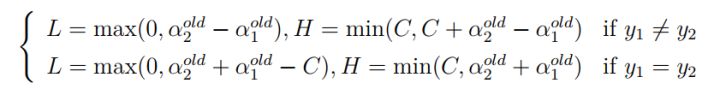




规划范围为：

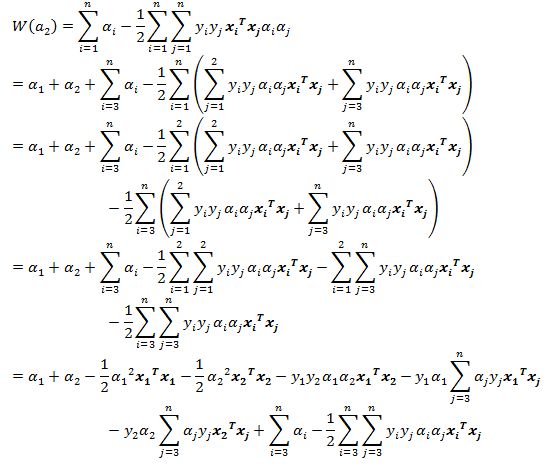


如此，根据y1和y2异号或同号，可以得出α2 new的上下界分别为

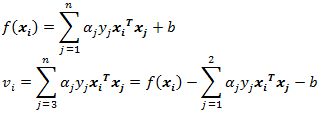


这个界限就是编程的时候需要用到的。已经确定了边界，接下来，就是推导迭代式，用于更新 α值。

我们依然假设选择的两个乘子为α1和α2。固定这两个乘子，进行推导。于是目标函数变成了：

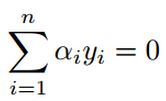


为了描述方便，我们定义如下符号：



最终目标函数变为：

IMG_256

对于这个目标函数，如果对其求导，还有个未知数α1，所以要推导出α1和α2的关系，然后用α2代替α1，这样目标函数就剩一个未知数了，我们就可以求导了，所以现在继续推导α1和α2的关系。注意第一个约束条件：

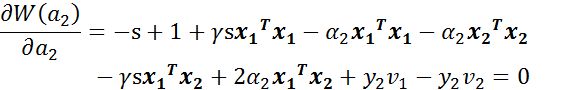
我们在求α1和α2的时候，可以将α3,α4,...,αn和y3,y4,...,yn看作常数项。因此有：

IMG_256

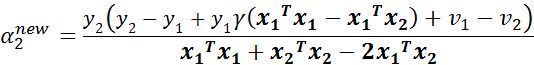
我们不必关心常数B的大小，现在将上述等式两边同时乘以y1，因为(y1y1=1)，所以：IMG_256

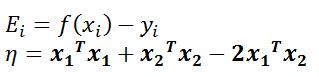
其中γ为常数By1，我们不关心这个值，s=y1y2。接下来，我们将得到的α1带入W(α2)公式得：

IMG_256

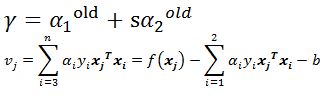
这样目标函数中就只剩下α2了，我们对其求偏导（注意：s=y1y2，所以s的平方为1，y1的平方和y2的平方均为1）：

继续化简，将s=y1y2带入方程：



我们令：

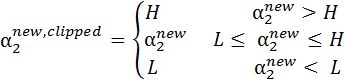
Ei为误差项，η为学习速率。

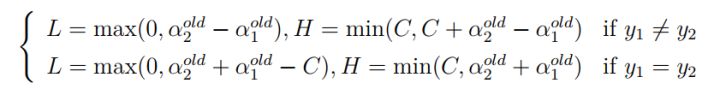
再根据我们已知的公式：

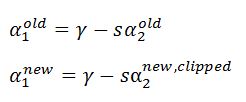
继续化简得：IMG_256

这个是没有经过剪辑是的解，需要考虑约束：IMG_256

根据之前推导的α取值范围，我们得到最终的解析解为：





又因为：

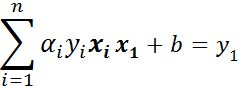
消去γ得：

IMG_256

当我们更新了α1和α2之后，需要重新计算阈值b，因为b关系到了我们f(x)的计算，也就关系到了误差Ei的计算。

我们要根据α的取值范围，去更正b的值，使间隔最大化。当α1 new在0和C之间的时候，根据KKT条件可知，这个点是支持向量上的点。因此，满足下列公式：IMG_256

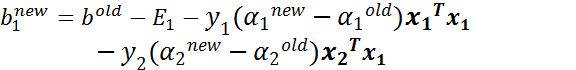
公式两边同时乘以y1得(y1y1=1)：

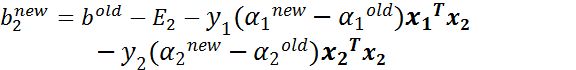


因为我们是根据α1和α2的值去更新b，所以单独提出i=1和i=2的时候，整理可得：IMG_256

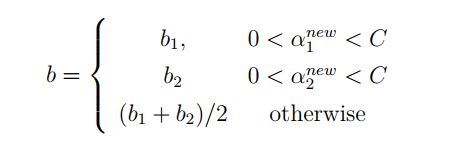
其中前两项为，对old也适用：IMG_256

整理得：





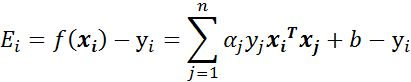
当b1和b2都有效的时候，它们是相等的，即：IMG_256

当两个乘子都在边界上，则b阈值和KKT条件一致。当不满足的时候，SMO算法选择他们的中点作为新的阈值：

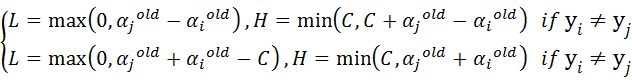
最后，更新所有的α和b，这样模型就出来了，从而即可求出我们的分类函数。

**计算步骤梳理：**

**A计算误差**



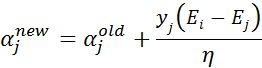
**B 计算上下界L和H**

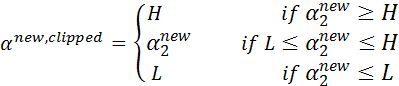


**C 计算**η

IMG_256

**D 更新αj，并根据C修剪**

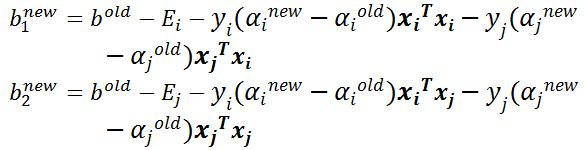




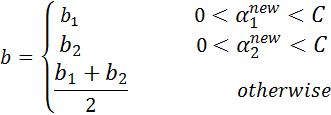
**E 更新αi**

IMG_256

**F 更新b1和b2**



**G 根据b1和b2更新b**



**得到所有α，最后可算出w**

