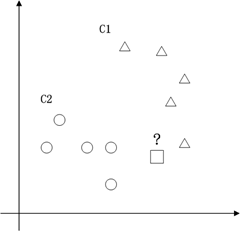
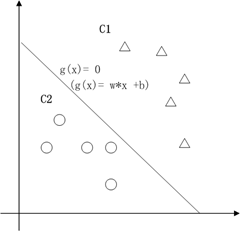
支持向量机(Support Vector Machine)是Cortes和Vapnik于1995年首先提出的，它在解决小样本、非线性及高维模式识别中表现出许多特有的优势，并能够推广应用到函数拟合等其他机器学习问题中。

**一、数学部分**

1.1二维空间

支持向量机的典型应用是分类，用于解决这样的问题：有一些事物是可以被分类的，但是具体怎么分类的我们又说不清楚，比如说下图中三角的就是C1类， 圆圈的就是C2类，这都是已知的，好，又来了一个方块，这个方块是属于C1呢还是属于C2呢，说不清楚。SVM算法就是试着帮您把这件事情说清楚的。

[](http://hi.csdn.net/attachment/201105/24/0_13062201781H4x.gif)在二维空间里（这时候样本有两个参照属性），SVM就是在C1和C2中间划一条线g(x)=0，线儿上边的属于C1类，线儿下边的属于C2类，这时候方块再来，咱就有章程了。

[](http://hi.csdn.net/attachment/201105/24/0_1306220179lBJX.gif)

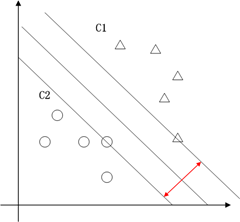
关于g(x) = 0得再啰嗦几句，g(x)里边的x不是横坐标，而是一个向量，[clip_image002[4]](http://hi.csdn.net/attachment/201105/24/0_1306220180FlW3.gif)也不是解析几何里边的斜率，也是向量。[clip_image004](http://hi.csdn.net/attachment/201105/24/0_1306220181r3x6.gif)是一个向量积。比如在解析几何意义上的直线y = -x-b,换成向量表示法就是[clip_image002[12]](http://hi.csdn.net/attachment/201105/24/0_1306220182Vimi.gif) ，这里w就是那个[clip_image008](http://hi.csdn.net/attachment/201105/24/0_1306220183EB38.gif)，x就是那个[clip_image010](http://hi.csdn.net/attachment/201105/24/0_1306220184CETp.gif)。

对C1类中的点：g(x) > 0；对于 C2类中的点：g(x) < 0 ;

如果我们用y来表示类型，+1代表C1类，-1代表C2类。

那么对于所有训练样本而言，都有：[clip_image012](http://hi.csdn.net/attachment/201105/24/0_1306220185CxQt.gif)，那么g(x) = 0 就能够正确分割所有训练样本的那条线，只要把g(x) = 0这条线给找出来就能凑合用了。

这也就只能凑合用，因为满足这个条件的g(x) = 0 太多了，追求完美的我们要的是最优的那条线。怎么才是最优的呢？直觉告诉我们g(x) = 0这条线不偏向C1那边，也不偏向C2那边，就应该是最优的了吧。对，学名叫分类间隔，下图红线的长度就是分类间隔。

[](http://hi.csdn.net/attachment/201105/24/0_1306220187O4YU.gif)

在二维空间中，求分类间隔，可以转化为求点到线的距离，点到线的距离可以表示为[clip_image002[6]](http://hi.csdn.net/attachment/201105/24/0_1306220191fixT.gif)(向量表示)。为简单计，把整个二维空间归一化(等比放大或缩小)，使得对于所有的样本，都有|g(x)|>=1，也就是让C1和C2类中离g(x)=0最近的训练样本的|g(x)|=1，这时分类间隔就是[clip_image004[4]](http://hi.csdn.net/attachment/201105/24/0_1306220192ngPt.gif)，这个间隔越大越好，那么|[clip_image006[8]](http://hi.csdn.net/attachment/201105/24/0_1306220193zl55.gif)|越小越好。

1.2多维空间

现在我们已经在2维空间中抽象出一个数学问题，求满足如下条件的g(x)=0：

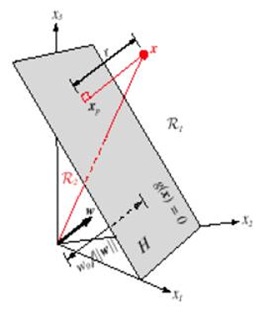
[clip_image008[4]](http://hi.csdn.net/attachment/201105/24/0_1306220194AMqf.gif)，即在满足[clip_image010[4]](http://hi.csdn.net/attachment/201105/24/0_1306220195bWjp.gif)条件下能使[clip_image012[4]](http://hi.csdn.net/attachment/201105/24/0_13062201987rZi.gif)取最小值的那个w。在二维空间中，w可以近似的理解为斜率，在样本确定，斜率确定的情况下，[clip_image014](http://hi.csdn.net/attachment/201105/24/0_1306220198mUb2.gif)中的那个b也是可以确定的，整个[clip_image016](http://hi.csdn.net/attachment/201105/24/0_1306220199Dqp5.gif) = 0也就确定了。

现在我们讨论的只是二维空间，但是我们惊喜的发现，在二维空间中的结论可以很容易的推广到多维空间。比如说：

我们仍然可以把多维空间中的分割面(超平面)表示为[clip_image018](http://hi.csdn.net/attachment/201105/24/0_130622020046IH.gif)。

多维空间中点到面的距离仍然可以表示为[clip_image002[7]](http://hi.csdn.net/attachment/201105/24/0_13062202030yVr.gif)。如下图，平面表示为[clip_image018[1]](http://hi.csdn.net/attachment/201105/24/0_1306220203DdYh.gif)，x是[clip_image020](http://hi.csdn.net/attachment/201105/24/0_130622020490PI.gif)在面上的投影，r是x到面的距离，简单推导如下：

w向量垂直于平面[clip_image022](http://hi.csdn.net/attachment/201105/24/0_1306220205JXDj.gif)，有：[clip_image024](http://hi.csdn.net/attachment/201105/24/0_130622020836xy.gif)，把上式带入[clip_image018[2]](http://hi.csdn.net/attachment/201105/24/0_1306220209Ek1y.gif)中得到[clip_image026](http://hi.csdn.net/attachment/201105/24/0_13062202122OE7.gif)，化简得到[clip_image028](http://hi.csdn.net/attachment/201105/24/0_1306220215iPmm.gif)，所以[clip_image030](http://hi.csdn.net/attachment/201105/24/0_1306220216838X.gif)，向量x到平面[clip_image018[3]](http://hi.csdn.net/attachment/201105/24/0_1306220219yGcy.gif)的距离[clip_image032](http://hi.csdn.net/attachment/201105/24/0_1306220220gNbb.gif)，这和二维空间中结论也是一致的。

[](http://hi.csdn.net/attachment/201105/24/0_1306220224Qm20.gif)

现在我们把SVM从2维空间推广到多维空间，即求满足如下条件的g(x)=0：

[clip_image036](http://hi.csdn.net/attachment/201105/24/0_1306220225tztn.gif)。

1.3拉格朗日因子

这是一个典型的带约束条件的求极值问题，目标函数是[clip_image006[9]](http://hi.csdn.net/attachment/201105/24/0_1306220228mCL0.gif)的二次函数，约束函数是[clip_image006[10]](http://hi.csdn.net/attachment/201105/24/0_1306220229nnpG.gif)的线性函数：二次规划问题。求解二次规划问题的一般性方法就是添加拉格朗日乘子，构造拉格朗日函数(理论上这儿应该还有一些额外的数学条件，拉格朗日法才是可用，就略过了)。

具体求解步骤如下：

1、构造拉格朗日函数

[clip_image038](http://hi.csdn.net/attachment/201105/24/0_1306220230iHr1.gif)

其中[clip_image006[11]](http://hi.csdn.net/attachment/201105/24/0_1306220230yYgT.gif)和b是未知量。

2、对[clip_image006[12]](http://hi.csdn.net/attachment/201105/24/0_1306220231UlN0.gif)和b求偏导数，令偏导数为0。

[clip_image040](http://hi.csdn.net/attachment/201105/24/0_13062202327p7M.gif), 即[clip_image042](http://hi.csdn.net/attachment/201105/24/0_1306220232hGjN.gif)

[clip_image044](http://hi.csdn.net/attachment/201105/24/0_1306220233JVJZ.gif)

3、把上式带回拉格朗日函数，得到拉格朗日对偶问题，把问题转化为求解[clip_image046](http://hi.csdn.net/attachment/201105/24/0_13062202340huS.gif)

[clip_image048](http://hi.csdn.net/attachment/201105/24/0_130622023714zf.gif)

4、最后把问题转化为求解满足下列等式的[clip_image046[1]](http://hi.csdn.net/attachment/201105/24/0_1306220238Hf4H.gif)

[clip_image050](http://hi.csdn.net/attachment/201105/24/0_1306220238xIN0.gif)

1.4线性化

好，现在我们再来梳理一下svm的分类逻辑，在空间中找一个分割面（线）把样本点分开，分割面（线）的最优条件就是分类间隔最大化，分类间隔是基于点到平面（直线）的距离来计算的。问题是所有的分割面都是平面，所有的分割线都是直线吗？显然不是。

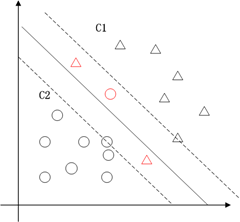
比如特征是房子的面积x，这里的x是实数，结果y是房子的价格。假设我们从样本点的分布中看到x和y符合3次曲线，那么我们希望使用x的三次多项式来逼近这些样本点。

在二维空间中这是非线性的，这样我们前面的推理都没法用了------点到曲线的距离？不知道怎么算。但是如果把x映射到3维空间[clip_image052](http://hi.csdn.net/attachment/201105/24/0_1306220239qiqi.gif)，那么对于[clip_image054](http://hi.csdn.net/attachment/201105/24/0_1306220240qd8p.gif)来说，[clip_image056](http://hi.csdn.net/attachment/201105/24/0_1306220241ddRG.gif)就是线性的，也就是说，对于低维空间中非线性的线(面)，在映射到高维空间中时，就能变成线性的。于是我们还需要把问题做一个小小的修正，我们面临的问题是求解：

[clip_image050[1]](http://hi.csdn.net/attachment/201105/24/0_1306220241a9ZP.gif)，这里面引入了一个Kernel，核函数，用于样本空间的线性化。

1.5松弛变量

      上面就是一个比较完整的推导过程，但是经验表明把上述条件丢给计算机进行求解，基本上是无解的，因为条件太苛刻了。实际上，最经常出现的情况如下图红色部分，在分类过程中会出现噪声，如果对噪声零容忍那么很有可能导致分类无解。

[](http://hi.csdn.net/attachment/201105/24/0_13062202422jm6.gif)

为了解决这个问题又引入了松弛变量。把原始问题修正为：

[clip_image002[10]](http://hi.csdn.net/attachment/201105/24/0_1306220243x6xG.gif)

按照拉格朗日法引入拉格朗日因子：

[clip_image004[6]](http://hi.csdn.net/attachment/201105/24/0_13062202475SMA.gif)

对上式分别求[clip_image006[18]](http://hi.csdn.net/attachment/201105/24/0_1306220250yipz.gif)的导数得到：

[clip_image008[6]](http://hi.csdn.net/attachment/201105/24/0_1306220251tZ2r.gif), 即[clip_image010[6]](http://hi.csdn.net/attachment/201105/24/0_1306220252Q10d.gif)

[clip_image012[6]](http://hi.csdn.net/attachment/201105/24/0_1306220255kTKN.gif)

[clip_image014[4]](http://hi.csdn.net/attachment/201105/24/0_1306220255Zw4I.gif)

带回[clip_image016[4]](http://hi.csdn.net/attachment/201105/24/0_1306220256Uz5y.gif)得到拉格朗日的对偶问题：

[clip_image018[10]](http://hi.csdn.net/attachment/201105/24/0_1306220257y02T.gif)

另外当目标函数取极值时，约束条件一定是位于约束边界(KKT条件)，也就是说：

[clip_image020[4]](http://hi.csdn.net/attachment/201105/24/0_1306220257hAMn.gif)

[clip_image022[4]](http://hi.csdn.net/attachment/201105/24/0_1306220258AN87.gif)

分析上面式子可以得出以下结论：

[clip_image024[4]](http://hi.csdn.net/attachment/201105/24/0_1306220259TFvR.gif)时：[clip_image026[4]](http://hi.csdn.net/attachment/201105/24/0_1306220263rEZz.gif)可以不为零，就是说该点到分割面的距离小于[clip_image028[4]](http://hi.csdn.net/attachment/201105/24/0_13062202665XZ7.gif)，是误分点。

[clip_image030[4]](http://hi.csdn.net/attachment/201105/24/0_13062202675r8L.gif)时：[clip_image026[5]](http://hi.csdn.net/attachment/201105/24/0_1306220268p1KW.gif)为零，[clip_image032[4]](http://hi.csdn.net/attachment/201105/24/0_1306220269y3fF.gif)大于零：表示该点到分割面的距离大于[clip_image028[5]](http://hi.csdn.net/attachment/201105/24/0_1306220270338D.gif)是正确分类点。

[clip_image034](http://hi.csdn.net/attachment/201105/24/0_13062202714zq5.gif)时：[clip_image026[6]](http://hi.csdn.net/attachment/201105/24/0_1306220272cHNc.gif)为零，[clip_image036[4]](http://hi.csdn.net/attachment/201105/24/0_13062202725oSs.gif)，该点就是支持向量。

再用数学语言提炼一下：

令[clip_image038[4]](http://hi.csdn.net/attachment/201105/24/0_1306220275brSC.gif)，其对[clip_image040[4]](http://hi.csdn.net/attachment/201105/24/0_1306220276NHQQ.gif)的偏导数为：

[clip_image042[4]](http://hi.csdn.net/attachment/201105/24/0_13062202779Twm.gif)。

KKT条件可以表示为：

[clip_image044[4]](http://hi.csdn.net/attachment/201105/24/0_1306220278z4qG.gif)

用[clip_image046[6]](http://hi.csdn.net/attachment/201105/24/0_1306220278eewU.gif)表示该KKT条件就是：

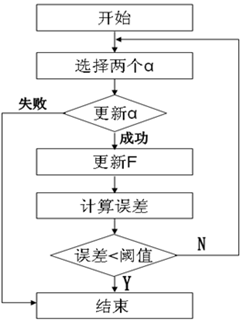
[clip_image048[9]](http://hi.csdn.net/attachment/201105/24/0_13062205628A2y.gif)

若[clip_image050[6]](http://hi.csdn.net/attachment/201105/24/0_1306220562w0QS.gif)，则

所有的[clip_image052[4]](http://hi.csdn.net/attachment/201105/24/0_1306220563ka76.gif) 大于所有的[clip_image054[4]](http://hi.csdn.net/attachment/201105/24/0_13062205631MBK.gif)。这里b作为中间数被忽略了，因为b是可以由[clip_image056[4]](http://hi.csdn.net/attachment/201105/24/0_1306220563myMy.gif)推导得到的。

**二、算法部分**

对于样本数量比较多的时候（几千个），SVM所需要的内存是计算机所不能承受的。目前，对于这个问题的解决方法主要有两种：块算法和分解算法。这里，libSVM采用的是分解算法中的SMO(串行最小化)方法，其每次训练都只选择两个样本。基本流程如下：

[](http://hi.csdn.net/attachment/201105/24/0_1306220563A4Cf.gif)

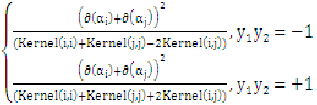
这里有两个重要的算法，一个是[clip_image060](http://hi.csdn.net/attachment/201105/24/0_13062205647zgg.gif)的选择，另一个是[clip_image060[1]](http://hi.csdn.net/attachment/201105/24/0_13062205649MZz.gif)的更新。

2.1[clip_image062](http://hi.csdn.net/attachment/201105/24/0_13062205641oav.gif)的选择算法

选择两个和KKT条件违背的最严重的两个[clip_image056[5]](http://hi.csdn.net/attachment/201105/24/0_1306220564CDJn.gif)，包含两层循环：

外层循环：优先选择遍历非边界样本，因为非边界样本更有可能需要调整，而边界样本常常不能得到进一步调整而留在边界上。在遍历过程中找出[clip_image064](http://hi.csdn.net/attachment/201105/24/0_13062205648448.gif)的所有样本中[clip_image066](http://hi.csdn.net/attachment/201105/24/0_1306220565AP7p.gif)值最大的那个(这个样本是最有可能不满足[clip_image068](http://hi.csdn.net/attachment/201105/24/0_1306220565TFcF.gif)条件的样本。

内层循环：对于外层循环中选定的那个样本[clip_image056[6]](http://hi.csdn.net/attachment/201105/24/0_1306220565Kg0A.gif)，找到这样的样本[clip_image070](http://hi.csdn.net/attachment/201105/24/0_1306220565z3yM.gif)，使得：

[](http://hi.csdn.net/attachment/201105/24/0_1306220565Fn4f.gif)

[clip_image074](http://hi.csdn.net/attachment/201105/24/0_1306220565aECl.gif)最大，上式是更新[clip_image060[2]](http://hi.csdn.net/attachment/201105/24/0_1306220566HG2E.gif)中的一个算式，表示的是在选定[clip_image056[7]](http://hi.csdn.net/attachment/201105/24/0_1306220566MNtZ.gif)，[clip_image070[1]](http://hi.csdn.net/attachment/201105/24/0_13062205660z5D.gif)最为更新算子的情况下，[clip_image076](http://hi.csdn.net/attachment/201105/24/0_1306220566paDx.gif)最大。

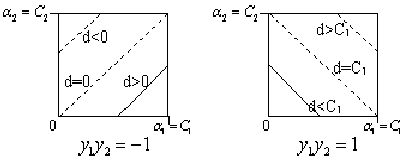
如果选择[clip_image060[3]](http://hi.csdn.net/attachment/201105/24/0_1306220566Wzpu.gif)的过程中发现KKT条件已经满足了，那么算法结束。

2.2[clip_image062[1]](http://hi.csdn.net/attachment/201105/24/0_1306220567rAo0.gif)的更新算法

由于SMO每次都只选择2个样本，那么等式约束可以转化为直线约束：

[clip_image078](http://hi.csdn.net/attachment/201105/24/0_1306220567JxJu.gif)

转化为图形表示为：

[](http://hi.csdn.net/attachment/201105/24/0_1306220567mYWK.gif)

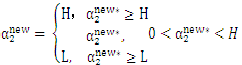
那么[clip_image081](http://hi.csdn.net/attachment/201105/24/0_130622056755qr.gif)的取值范围是：

[clip_image083](http://hi.csdn.net/attachment/201105/24/0_13062205673SI8.gif)

把[clip_image085](http://hi.csdn.net/attachment/201105/24/0_130622056843Zf.gif)带入[clip_image087](http://hi.csdn.net/attachment/201105/24/0_1306220568M3f3.gif)中，得到一个一元二次方程，求极值得到：

[clip_image089](http://hi.csdn.net/attachment/201105/24/0_1306220568M4y4.gif)

最终：

[](http://hi.csdn.net/attachment/201105/24/0_13062205689zrR.gif)

2.3其他

上面说到SVM用到的内存巨大，另一个缺陷就是计算速度，因为数据大了，计算量也就大，很显然计算速度就会下降。因此，一个好的方式就是在计算过程中逐步去掉不参与计算的数据。因为，实践证明，在训练过程中，[clip_image095](http://hi.csdn.net/attachment/201105/24/0_1306220568hs16.gif)一旦达到边界([clip_image095[1]](http://hi.csdn.net/attachment/201105/24/0_13062205685ua9.gif)=0或者[clip_image095[2]](http://hi.csdn.net/attachment/201105/24/0_1306220568Z14F.gif)=C)，[clip_image095[3]](http://hi.csdn.net/attachment/201105/24/0_1306220569BbGm.gif)的值就不会变，随着训练的进行，参与运算的样本会越来越少，SVM最终结果的支持向量（0<[clip_image095[4]](http://hi.csdn.net/attachment/201105/24/0_1306220569YY5R.gif)

LibSVM采用的策略是在计算过程中，检测active\_size中的[clip_image095[5]](http://hi.csdn.net/attachment/201105/24/0_1306220569XnoG.gif)值，如果[clip_image095[6]](http://hi.csdn.net/attachment/201105/24/0_1306220569i324.gif)到了边界，那么就应该把相应的样本去掉（变成inactived），并放到栈的尾部，从而逐步缩小active\_size的大小。

b的计算 ，基本计算公式为：[clip_image097](http://hi.csdn.net/attachment/201105/24/0_13062205699Kc6.gif)

理论上，b的值是不定的。当程序达到最优后，只要用任意一个标准支持向量机（0<[clip_image095[7]](http://hi.csdn.net/attachment/201105/24/0_1306220570J3j2.gif)<C）的样本带入上式，得到的b值都是可以的。目前，求b的方法也有很多种。在libSVM中，分别对y=+1和y=-1的两类所有支持向量求b，然后取平均值。