**6、支持向量机**（Support Vector Machine）

支持向量机（support vector machines，SVM）是一种二类分类模型。它的基本模型是定义在特征空间上的间隔最大的线性分类器，间隔最大使它有别于感知机；支持向量机还包括核技巧，这使它成为实质上的非线性分类器。支持向量机的学习策略就是间隔最大化，可形式化为一个求解凸二次规划（convex quadratic programming）的问题，也等价于正则化的合页损失函数的最小化问题。支持向量机的学习算法是求解凸二次规划的最优化算法。

支持向量机学习方法包含构建由简至繁的模型：线性可分支持向量机（linear support vector machine in linearly separable case）、线性支持向量机（linear support vector machine）及非线性支持向量机（non-linear support vector machine）。简单模型是复杂模型的基础，也是复杂模型的特殊情况。当训练数据线性可分时，通过硬间隔最大化（hard margin maximization），学习一个线性的分类器，即线性可分支持向量机，又称为硬间隔支持向量机；当训练数据近似线性可分时，通过软间隔最大化（soft margin maximization），也学习一个线性的分类器，即线性支持向量机，又称为软间隔支持向量机；当训练数据线性不可分时，通过使用核技巧（kernel trick）及软间隔最大化，学习非线性支持向量机。

当输入空间为欧氏空间或离散集合、特征空间为希尔伯特空间时，核函数（kernel function）表示将输入从输入空间映射到特征空间得到的特征向量之间的内积。通过使用核函数可以学习非线性支持向量机，等价于隐式地在高维的特征空间中学习线性支持向量机。这样的方法称为核技巧。核方法（kernel method）是比支持向量机更为一般的机器学习方法。

重要名词：

定义的function:

优化目标：学习的目标是在特征空间中找到一个分离超平面，能将实例分到不同的类，支持向量机的学习策略就是间隔最大化。

Loss function:

支持向量（support vector）：在线性可分情况下，训练数据集的样本点中与分离超平面距离最近的样本点的实例称为支持向量（support vector）。

核函数（kernel function）表示将输入从输入空间映射到特征空间得到的特征向量之间的内积。

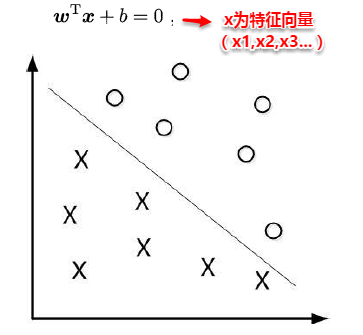
SVM的核函数：线性核、多项式核、高斯核、拉普拉斯核、Sigmoid核。

硬间隔（hard margin）:所有样本都必须正确划分。

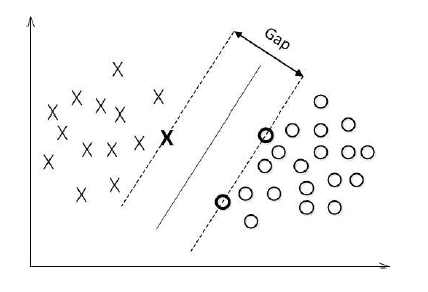
软间隔（soft margin）:允许某些样本不满足约束条件，不是所有的样本都必须正确划分。

**6.1 函数间隔与几何间隔**

对于二分类学习，假设现在的数据是线性可分的，这时分类学习最基本的想法就是找到一个合适的超平面，该超平面能够将不同类别的样本分开，类似二维平面使用ax+by+c=0来表示，超平面实际上表示的就是高维的平面，如下图所示：

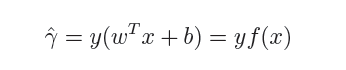
[](https://camo.githubusercontent.com/6216eb76f4f58e2a8bcf7cb8020a159612d2fb19/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266366132656338612e706e67)

对数据点进行划分时，易知：当超平面距离与它最近的数据点的间隔越大，分类的鲁棒性越好，即当新的数据点加入时，超平面对这些点的适应性最强，出错的可能性最小。因此需要让所选择的超平面能够最大化这个间隔Gap（如下图所示）， 常用的间隔定义有两种，一种称之为函数间隔，一种为几何间隔，下面将分别介绍这两种间隔，并对SVM为什么会选用几何间隔做了一些阐述。

[](https://camo.githubusercontent.com/1863681dae3c4d317fbd11cfa67956f552cdd0ce/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266366130366435612e706e67)

**6.1.1 函数间隔**

在超平面w'x+b=0确定的情况下，|w'x\*+b|能够代表点x*距离超平面的远近，易知：当w'x*+b>0时，表示x*在超平面的一侧（正类，类标为1），而当w'x*+b<0时，则表示x*在超平面的另外一侧（负类，类别为-1），因此（w'x*+b）y\* 的正负性恰能表示数据点x\*是否被分类正确。于是便引出了**函数间隔**的定义（functional margin）:

[](https://camo.githubusercontent.com/c0c50f14aac95c23774981efe193e005755cab3c/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266363930613134622e706e67)

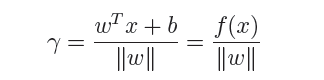
而超平面（w,b）关于所有样本点（Xi，Yi）的函数间隔最小值则为超平面在训练数据集T上的函数间隔：

[4.png](https://camo.githubusercontent.com/77568bd8fb238ab5f3abefdcc8765624d01d0535/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266363930616332362e706e67)

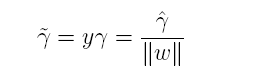
可以看出：这样定义的函数间隔在处理SVM上会有问题，当超平面的两个参数w和b同比例改变时，函数间隔也会跟着改变，但是实际上超平面还是原来的超平面，并没有变化。例如：w1x1+w2x2+w3x3+b=0其实等价于2w1x1+2w2x2+2w3x3+2b=0，但计算的函数间隔却翻了一倍。从而引出了能真正度量点到超平面距离的概念--几何间隔（geometrical margin）。

**6.1.2 几何间隔**

**几何间隔**代表的则是数据点到超平面的真实距离，对于超平面w'x+b=0，w代表的是该超平面的法向量，设x*为超平面外一点x在法向量w方向上的投影点，x与超平面的距离为r，则有x*=x-r(w/||w||)，又x*在超平面上，即w'x*+b=0，代入即可得：

[](https://camo.githubusercontent.com/c8da84f7f11b93929da1381762e36416f1dbd216/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266363937643439392e706e67)

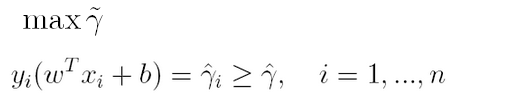
为了得到r的绝对值，令r呈上其对应的类别y，即可得到几何间隔的定义：

[](https://camo.githubusercontent.com/c25576bc8c84ae9d434f936f5032205ab02588f7/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266363936666431302e706e67)

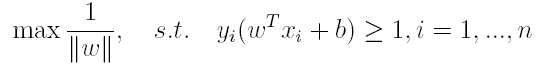
从上述函数间隔与几何间隔的定义可以看出：实质上函数间隔就是|w'x+b|，而几何间隔就是点到超平面的距离。

**6.2 最大间隔与支持向量**

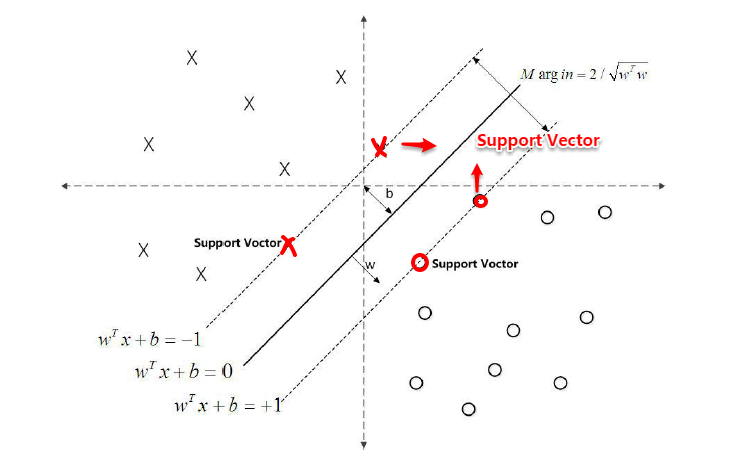
通过前面的分析可知：函数间隔不适合用来最大化间隔，因此这里我们要找的最大间隔指的是几何间隔，于是最大间隔分类器的目标函数定义为：

[](https://camo.githubusercontent.com/d02e52951c54c05b8f2fb527175dbecd22049c90/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266363961663136332e706e67)

一般地，我们令r^为1（这样做的目的是为了方便推导和目标函数的优化），从而上述目标函数转化为：

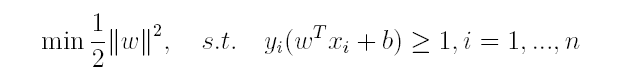
[](https://camo.githubusercontent.com/f254588530dd4e7fad23f0c88532a277801d717a/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266363937626231642e706e67)

对于y(w'x+b)=1的数据点，即下图中位于w'x+b=1或w'x+b=-1上的数据点，我们称之为**支持向量**（support vector），易知：对于所有的支持向量，它们恰好满足y\*(w'x\*+b)=1，而所有不是支持向量的点，有y\*(w'x\*+b)>1。

[](https://camo.githubusercontent.com/fb2b56682fa052c622c39089f781f920dd04f794/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266366138333863342e706e67)

**6.3 从原始优化问题到对偶问题**

对于上述得到的目标函数，求1/||w||的最大值相当于求||w||^2的最小值，因此很容易将原来的目标函数转化为：

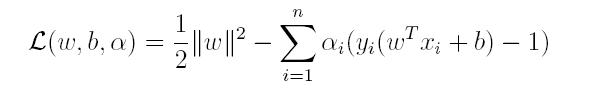
[](https://camo.githubusercontent.com/6b86dfa524a12635e391e14e655604493662a1fb/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266363937386362622e706e67)

即变为了一个带约束的凸二次规划问题，按书上所说可以使用现成的优化计算包（QP优化包）求解，但由于SVM的特殊性，一般我们将原问题变换为它的**对偶问题**，接着再对其对偶问题进行求解。为什么通过对偶问题进行求解，有下面两个原因：

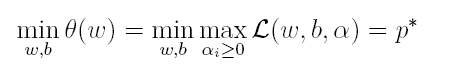
\* 一是因为使用对偶问题更容易求解；

\* 二是因为通过对偶问题求解出现了向量内积的形式，从而能更加自然地引出核函数。

对偶问题，顾名思义，可以理解成优化等价的问题，更一般地，是将一个原始目标函数的最小化转化为它的对偶函数最大化的问题。对于当前的优化问题，首先我们写出它的朗格朗日函数：

[](https://camo.githubusercontent.com/4c31006db592ed2b61ca16d5547cfa46e96ac843/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266393333326265372e706e67)

上式很容易验证：当其中有一个约束条件不满足时，L的最大值为 ∞（只需令其对应的α为 ∞即可）；当所有约束条件都满足时，L的最大值为1/2||w||^2（此时令所有的α为0），因此实际上原问题等价于：

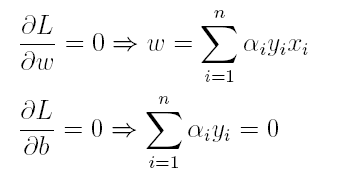
[](https://camo.githubusercontent.com/d8d4fb588898df7329d777e92e6109f7450a7103/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266393333323163352e706e67)

由于这个的求解问题不好做，因此一般我们将最小和最大的位置交换一下（需满足KKT条件） ，变成原问题的对偶问题：

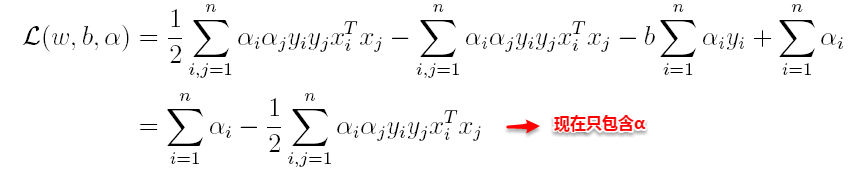
[13.png](https://camo.githubusercontent.com/49f28adb667e512f30f642e07b71ec8b59ee5c02/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266393333303936372e706e67)

这样就将原问题的求最小变成了对偶问题求最大（用对偶这个词还是很形象），接下来便可以先求L对w和b的极小，再求L对α的极大。

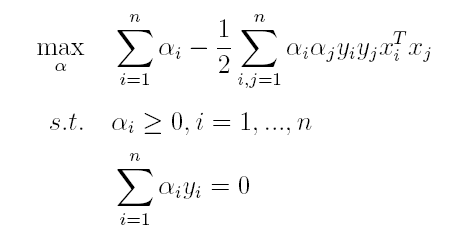
（1）首先求L对w和b的极小，分别求L关于w和b的偏导，可以得出：

[](https://camo.githubusercontent.com/05b992d3342565745cdbf9c098f48ec9e9a1c4fb/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266393333336536362e706e67)

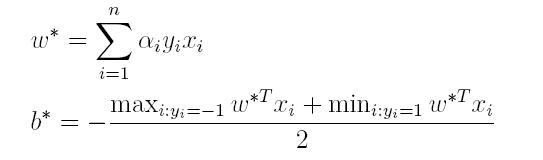
将上述结果代入L得到：

[](https://camo.githubusercontent.com/7f9fff60bd1176e7c91ce5257c3633bc732f6035/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266393335616532312e706e67)

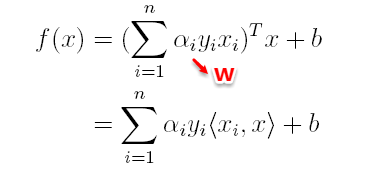
（2）接着L关于α极大求解α（通过SMO算法求解，此处不做深入）。

[](https://camo.githubusercontent.com/d59863246b172956d2a69801d146c420d1494be1/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266393333386139642e706e67)

（3）最后便可以根据求解出的α，计算出w和b，从而得到分类超平面函数。

[](https://camo.githubusercontent.com/91a54bd29d61b971e0d539ed5af5cb34bc5029f1/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266393334313963612e706e67)

在对新的点进行预测时，实际上就是将数据点x\*代入分类函数f(x)=w'x+b中，若f(x)>0，则为正类，f(x)<0，则为负类，根据前面推导得出的w与b，分类函数如下所示，此时便出现了上面所提到的内积形式。

[](https://camo.githubusercontent.com/fd89b3538ab2f6d66bc160b867d80b590a54be32/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266393335333136362e706e67)

这里实际上只需计算新样本与支持向量的内积，因为对于非支持向量的数据点，其对应的拉格朗日乘子一定为0，根据最优化理论（K-T条件），对于不等式约束y(w'x+b)-1≥0，满足：

[19.png](https://camo.githubusercontent.com/53c485ec65f55e457a676de77c345f02b5361f1d/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266393333633934372e706e67)

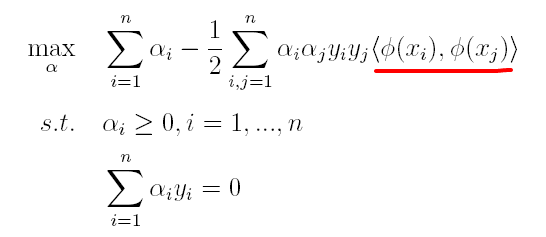
**6.4 核函数**

由于上述的超平面只能解决线性可分的问题，对于线性不可分的问题，例如：异或问题，我们需要使用核函数将其进行推广。一般地，解决线性不可分问题时，常常采用**映射**的方式，将低维原始空间映射到高维特征空间，使得数据集在高维空间中变得线性可分，从而再使用线性学习器分类。如果原始空间为有限维，即属性数有限，那么总是存在一个高维特征空间使得样本线性可分。若∅代表一个映射，则在特征空间中的划分函数变为：

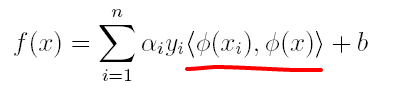
[20.png](https://camo.githubusercontent.com/4b376deb3368bca9c583de8d813d1036350dfbfa/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373266393334333033652e706e67)

按照同样的方法，先写出新目标函数的拉格朗日函数，接着写出其对偶问题，求L关于w和b的极大，最后运用SOM求解α。可以得出：

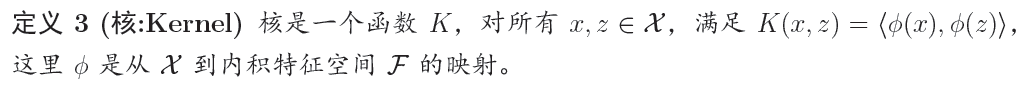
（1）原对偶问题变为：

[](https://camo.githubusercontent.com/e074ec51e2cd120276e4cd95d653016febcb4d97/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373330636336386233622e706e67)

（2）原分类函数变为：

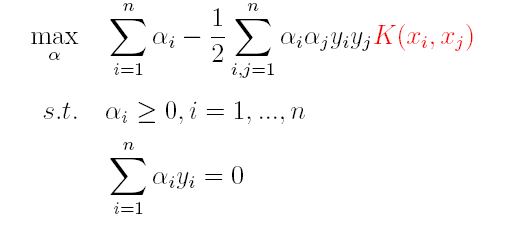
​ [](https://camo.githubusercontent.com/03f5e1c9ab83bc09d2be4c8c292e8e1eea6935f2/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373330636331623637332e706e67)

求解的过程中，只涉及到了高维特征空间中的内积运算，由于特征空间的维数可能会非常大，例如：若原始空间为二维，映射后的特征空间为5维，若原始空间为三维，映射后的特征空间将是19维，之后甚至可能出现无穷维，根本无法进行内积运算了，此时便引出了**核函数**（Kernel）的概念。

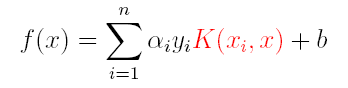
[](https://camo.githubusercontent.com/098f47ea5b4dde579bf5b943871bcd0b8c2fa36b/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373330636334396164632e706e67)

因此，核函数可以直接计算隐式映射到高维特征空间后的向量内积，而不需要显式地写出映射后的结果，它虽然完成了将特征从低维到高维的转换，但最终却是在低维空间中完成向量内积计算，与高维特征空间中的计算等效\*\*（低维计算，高维表现）\*\*，从而避免了直接在高维空间无法计算的问题。引入核函数后，原来的对偶问题与分类函数则变为：

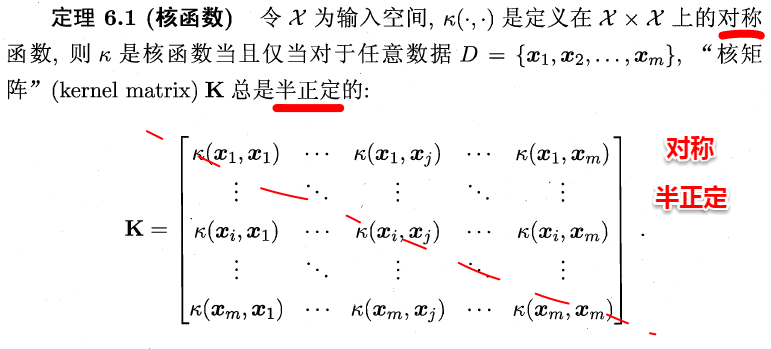
（1）对偶问题：

[](https://camo.githubusercontent.com/7c46639c5d8152a0e878ba791aaaddf63cb5c4c0/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373330636331373362322e706e67)

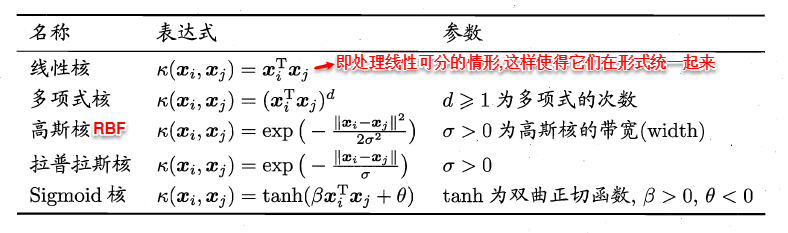
（2）分类函数：

[](https://camo.githubusercontent.com/a06eb9c340ef145b15d1455b209d9ab202405401/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373330636330353935392e706e67)

因此，在线性不可分问题中，核函数的选择成了支持向量机的最大变数，若选择了不合适的核函数，则意味着将样本映射到了一个不合适的特征空间，则极可能导致性能不佳。同时，核函数需要满足以下这个必要条件：

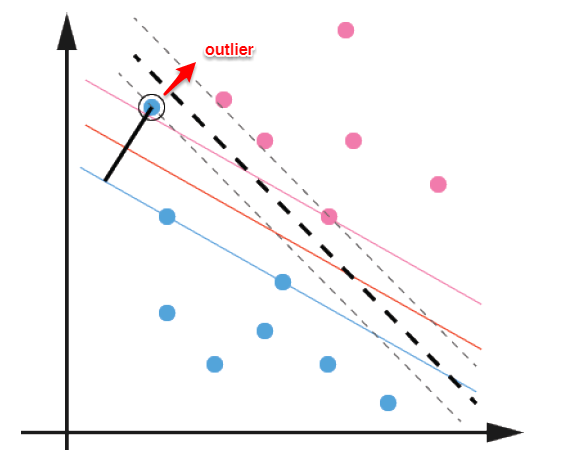
[](https://camo.githubusercontent.com/66a900a9f8a143235b63bdbbdd7bd57de3e1c674/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373330636363343638632e706e67)

由于核函数的构造十分困难，通常我们都是从一些常用的核函数中选择，下面列出了几种常用的核函数：

[](https://camo.githubusercontent.com/21f9aa9b720a9709052e7c239de46881825761fc/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373330636363353431612e706e67)

**6.5 软间隔支持向量机**

前面的讨论中，我们主要解决了两个问题：当数据线性可分时，直接使用最大间隔的超平面划分；当数据线性不可分时，则通过核函数将数据映射到高维特征空间，使之线性可分。然而在现实问题中，对于某些情形还是很难处理，例如数据中有**噪声**的情形，噪声数据（**outlier**）本身就偏离了正常位置，但是在前面的SVM模型中，我们要求所有的样本数据都必须满足约束，如果不要这些噪声数据还好，当加入这些outlier后导致划分超平面被挤歪了，如下图所示，对支持向量机的泛化性能造成很大的影响。

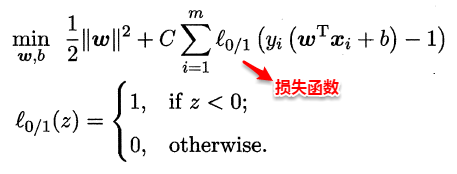
[](https://camo.githubusercontent.com/5be593fadf1fb49922645967574710c7a2fa4e4a/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373330636363653638652e706e67)

为了解决这一问题，我们需要允许某一些数据点不满足约束，即可以在一定程度上偏移超平面，同时使得不满足约束的数据点尽可能少，这便引出了\*\*“软间隔”支持向量机\*\*的概念

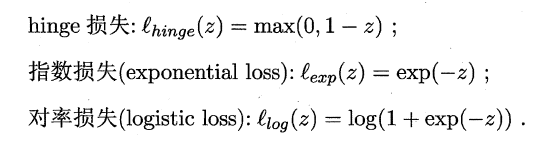
\* 允许某些数据点不满足约束y(w'x+b)≥1；

\* 同时又使得不满足约束的样本尽可能少。

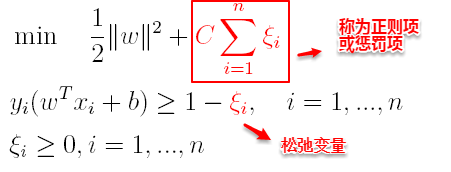
这样优化目标变为：

[](https://camo.githubusercontent.com/18bfb70c264bab2b8da52c1844b07735afac6f1d/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373330636336633966652e706e67)

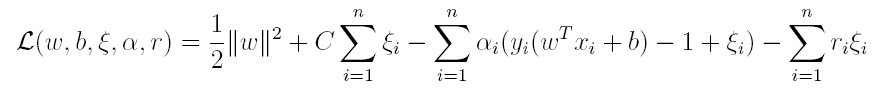
如同阶跃函数，0/1损失函数虽然表示效果最好，但是数学性质不佳。因此常用其它函数作为“替代损失函数”。

[](https://camo.githubusercontent.com/7969c015f59598f5d1223c05521b9fba61308fe8/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373330636335653561392e706e67)

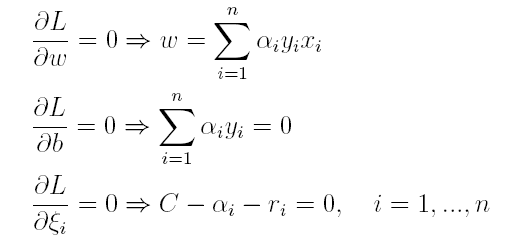
支持向量机中的损失函数为**hinge损失**，引入\*\*“松弛变量”\*\*，目标函数与约束条件可以写为：

[](https://camo.githubusercontent.com/9b57e48bf62902c0068383badf5012dfef712aae/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373331376161333431312e706e67)

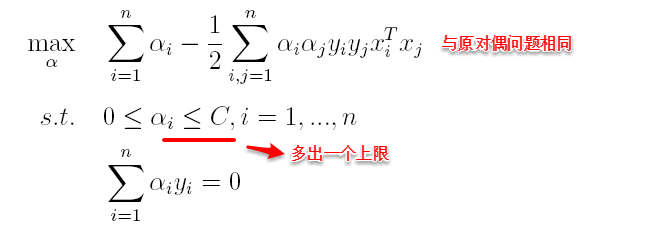
其中C为一个参数，控制着目标函数与新引入正则项之间的权重，这样显然每个样本数据都有一个对应的松弛变量，用以表示该样本不满足约束的程度，将新的目标函数转化为拉格朗日函数得到：

[](https://camo.githubusercontent.com/c7e29be9a7ea2cdb0756b6fd139ff6cd99033ec2/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373331376134633936652e706e67)

按照与之前相同的方法，先让L求关于w，b以及松弛变量的极小，再使用SMO求出α，有：

[](https://camo.githubusercontent.com/2819d3b5e60d5e5d848ca5310062177636024702/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373331376136646666322e706e67)

将w代入L化简，便得到其对偶问题：

[](https://camo.githubusercontent.com/f7fa8148028e191f5fef6280212da1e1ece349bb/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373331376162363634362e706e67)

将“软间隔”下产生的对偶问题与原对偶问题对比可以发现：新的对偶问题只是约束条件中的α多出了一个上限C，其它的完全相同，因此在引入核函数处理线性不可分问题时，便能使用与“硬间隔”支持向量机完全相同的方法。

**6.6 SVM的特点**

SVM有如下主要几个特点：

(1)  非线性映射是SVM方法的理论基础,SVM利用内积核函数代替向高维空间的非线性映射；

(2)  对特征空间划分的最优超平面是SVM的目标,最大化分类边际的思想是SVM方法的核心；

(3)  支持向量是SVM的训练结果,在SVM分类决策中起决定作用的是支持向量。

(4)  SVM 是一种有坚实理论基础的新颖的小样本学习方法。它基本上不涉及概率测度及大数定律等,因此不同于现有的统计方法。从本质上看,它避开了从归纳到演绎的传统过程,实现了高效的从训练样本到预报样本的“转导推理”,大大简化了通常的分类和回归等问题。

(5)  SVM 的最终决策函数只由少数的支持向量所确定,计算的复杂性取决于支持向量的数目,而不是样本空间的维数,这在某种意义上避免了“维数灾难”。

(6)  少数支持向量决定了最终结果,这不但可以帮助我们抓住关键样本、“剔除”大量冗余样本,而且注定了该方法不但算法简单,而且具有较好的“鲁棒”性。这种“鲁棒”性主要体现在:

        ①增、删非支持向量样本对模型没有影响;

        ②支持向量样本集具有一定的鲁棒性;

        ③有些成功的应用中,SVM 方法对核的选取不敏感

(7)  SVM学习问题可以表示为凸优化问题，因此可以利用已知的有效算法发现目标函数的全局最小值。而其他分类方法（如基于规则的分类器和人工神经网络）都采用一种基于贪心学习的策略来搜索假设空间，这种方法一般只能获得局部最优解。

(8)  SVM通过最大化决策边界的边缘来控制模型的能力。尽管如此，用户必须提供其他参数，如使用核函数类型和引入松弛变量等。

(9)  SVM在小样本训练集上能够得到比其它算法好很多的结果。支持向量机之所以成为目前最常用，效果最好的分类器之一，在于其优秀的泛化能力，这是是因为其本身的优化目标是结构化风险最小，而不是经验风险最小，因此，通过margin的概念，得到对数据分布的结构化描述，因此减低了对数据规模和数据分布的要求。SVM也并不是在任何场景都比其他算法好，对于每种应用，最好尝试多种算法，然后评估结果。如SVM在邮件分类上，还不如逻辑回归、KNN、bayes的效果好。

(10)  它基于结构风险最小化原则，这样就避免了过学习问题，泛化能力强。

(11)  它是一个凸优化问题，因此局部最优解一定是全局最优解的优点。

(12)  泛化错误率低，分类速度快，结果易解释

不足之处：

(1) SVM算法对大规模训练样本难以实施

        SVM的空间消耗主要是存储训练样本和核矩阵，由于SVM是借助二次规划来求解支持向量，而求解二次规划将涉及m阶矩阵的计算（m为样本的个数），当m数目很大时该矩阵的存储和计算将耗费大量的机器内存和运算时间。针对以上问题的主要改进有有J.Platt的SMO算法、T.Joachims的SVM、C.J.C.Burges等的PCGC、张学工的CSVM以及O.L.Mangasarian等的SOR算法。

        如果数据量很大，SVM的训练时间就会比较长，如垃圾邮件的分类检测，没有使用SVM分类器，而是使用了简单的naive bayes分类器，或者是使用逻辑回归模型分类。

(2) 用SVM解决多分类问题存在困难

        经典的支持向量机算法只给出了二类分类的算法，而在数据挖掘的实际应用中，一般要解决多类的分类问题。可以通过多个二类支持向量机的组合来解决。主要有一对多组合模式、一对一组合模式和SVM决策树；再就是通过构造多个分类器的组合来解决。主要原理是克服SVM固有的缺点，结合其他算法的优势，解决多类问题的分类精度。如：与粗集理论结合，形成一种优势互补的多类问题的组合分类器。

(3)对缺失数据敏感，对参数和核函数的选择敏感

        支持向量机性能的优劣主要取决于核函数的选取,所以对于一个实际问题而言,如何根据实际的数据模型选择合适的核函数从而构造SVM算法.目前比较成熟的核函数及其参数的选择都是人为的,根据经验来选取的,带有一定的随意性.在不同的问题领域,核函数应当具有不同的形式和参数,所以在选取时候应该将领域知识引入进来,但是目前还没有好的方法来解决核函数的选取问题.

支持向量机的主要应用和研究的热点

        目前支持向量机主要应用在模式识别领域中的文本识别,中文分类,人脸识别等;同时也应用到许多的工程技术和信息过滤等方面.

        当前研究的热点主要是对支持向量机中算法的优化,包括解决SVM中二次规划求解问题,对大规模SVM的求解问题,对SVM中QP问题的求解问题等.另外就是如何更好的构造基于SVM的多类分类器,如何提高SVM的归纳能力和分类速度等.如何根据实际问题确定核函数也是一个重要的研究热点.