SVM详解

guoliqiang@pku.edu.cn

**C-SVM**

# 线性分类器

## 定义

* 1. 空间中的超平面。



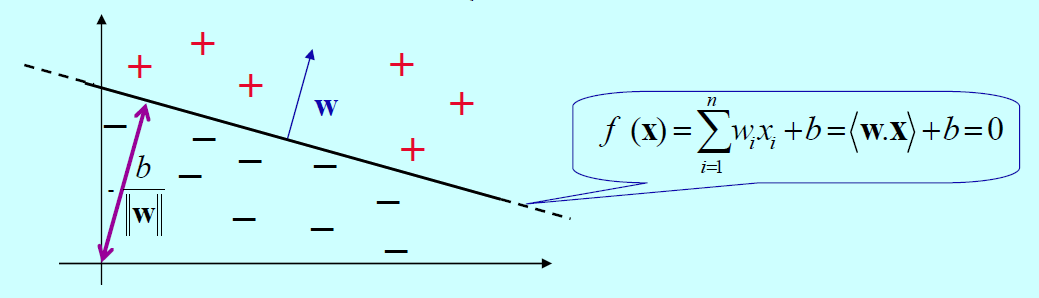
* 1. 由向量****和截距定义。



* 1. 分类规则定义如下：

(1)

注意：等于0的情况被分到了负例中。对于的绝对值很小的情况，很难有好的处理方法；因为细微的变动（比如超平面稍微转一个小角度）就可能导致结果类别的改变。理想情况，期望的值都是很大的正数或很小的负数，这样就能更加确信它是属于其中某一类别。



图[1]：线性分类器示意图

1. 向量的方向与超平面垂直。



1. 相当于原点到超平面的距离；正负表示原点是在超平面的上侧还是下侧。



1. 补充知识：点到直线的距离公式

设直线与点，则点到直线的距离为：



(2)



# 间隔

## 函数间隔（functional margin）

点的函数间隔 w.r.t. 为：

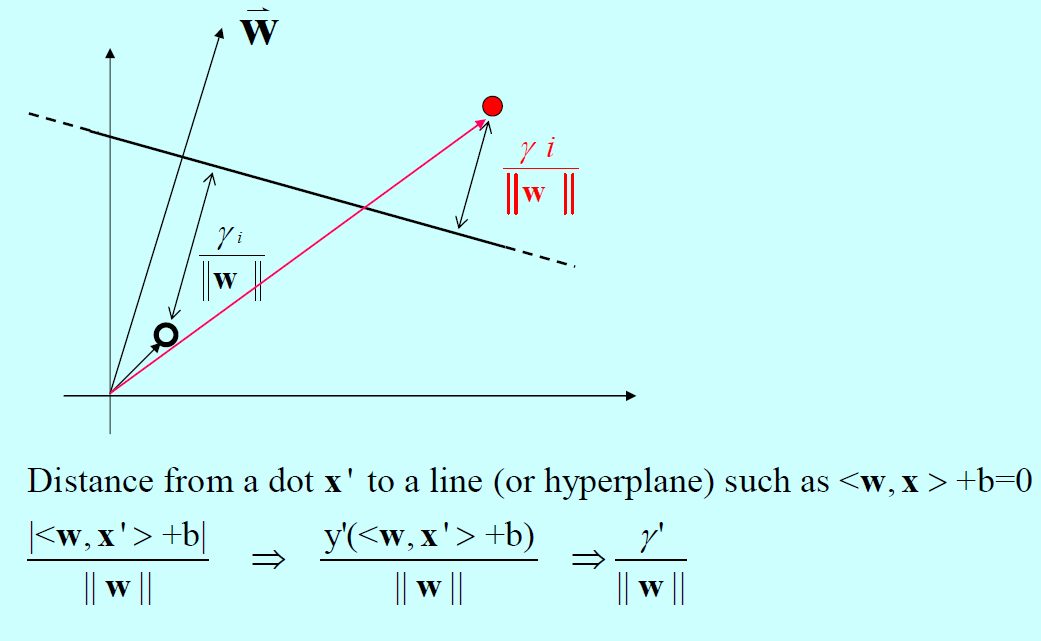
其中（3）

可以看到函数间隔为点到直线距离的分子，因为当时，乘以起到了取绝对值的作用。

## 几何间隔（geometric margin）

（4）

其中，是2-范数。可以看到几何间隔就是点到超平面的距离。



图[2]：几何间隔实际含义

1. w.r.t.超平面，训练集的函数间隔为：



（5）

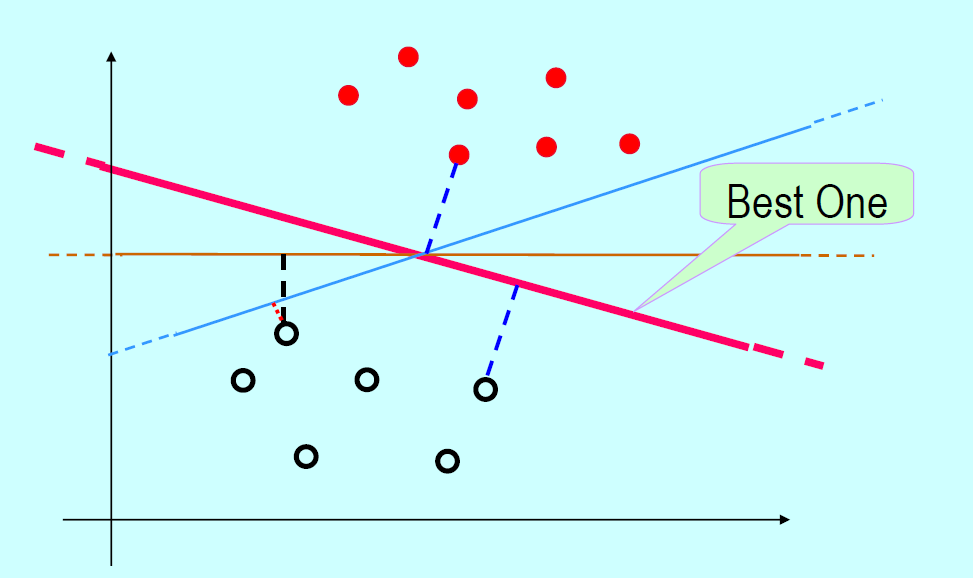


1. 训练集的函数间隔为:



（6）

从后面可以看到SVM就是在寻找某个条件下的这个最大间隔，此时便为最大间隔超平面，此超平面可以最小化泛化误差(generalization error)，如下图：



图[3]:3个不同但可以无误分开正负例的超平面

其中红颜色代表的超平面是最佳的，因为它不偏向于任何一方。（图中蓝颜色代表的距离相等）

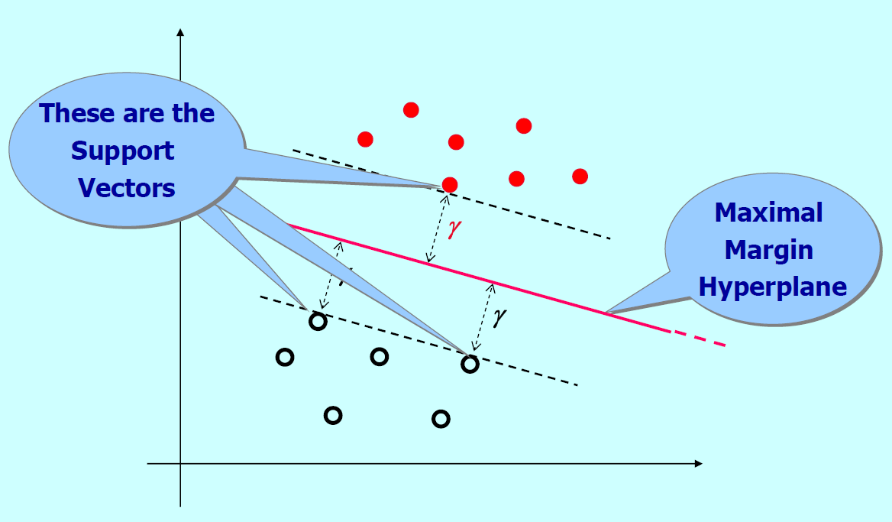


图4：最大间隔超平面与支持向量点

可以看到最大间隔超平面的确定只需要图中的3个样例点便可以，其它的点对最大超平面的确定没有任何作用，（也就是，如果你能确定训练数据中哪些是这中类型的点，完全可以去掉其余的点，只用这些点训练模型；但不幸的是在超平面确定之前，这些点是无法确定的），这三个点便被称为“支持向量”。

## 函数间隔的问题

函数间隔：

(7)

由于可以随意成倍的改变而不改变其代表的超平面，比如与代表的超平面是一样的，但是点的函数间隔分别与。为了计算方便，SVM设定，因为总能通过改变的值达到这一点，也就是SVM最终计算出的是在情况下的。需要注意的是，几何间隔并不存在此问题。



情况下得到的最大间隔超平面的几何间隔为：，这样SVM的优化目标为：



(8)

即寻找到几何间隔最大时的。

# 线性可分的SVM

## 简介

(9)

1. SVM设定最大间隔分割超平面的在某个训练集合上的函数间隔为1，这点可以从限制条件得出。也就是，训练数据中点的函数间隔至少为1，此时最大间隔分割超平面在训练集上的函数间隔为1。

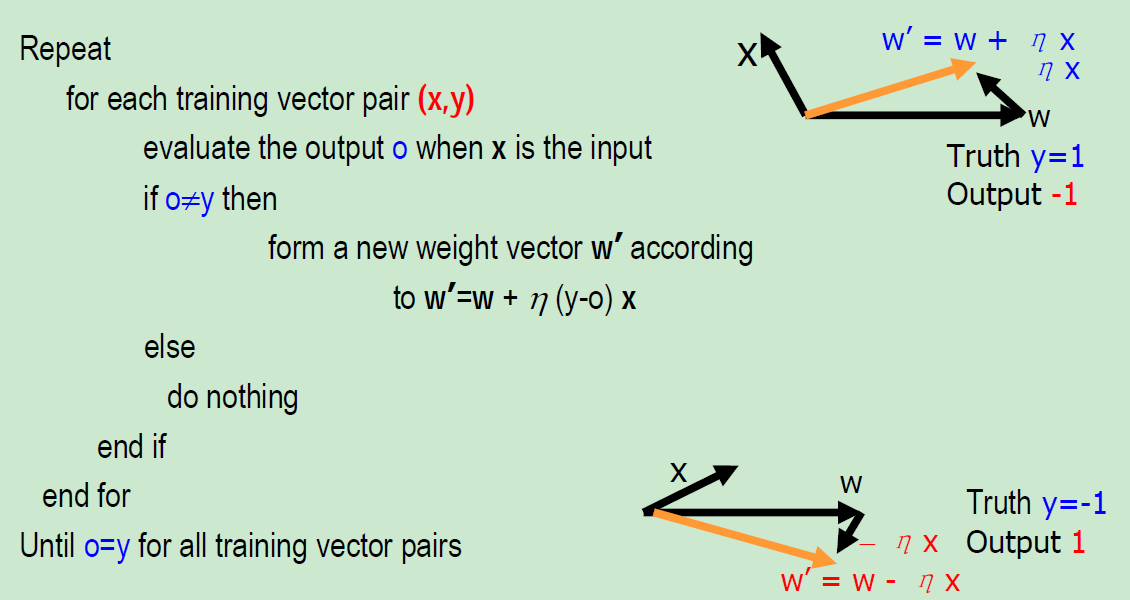


1. 公式9定义的SVM只有训练数据集合上确实存在这样一个超平面的情况下才会有解，因此称为线性可分的SVM，然而现实情况中不总是存在有这样的超平面，此问题的解决方法可以从下文中看到。

3. 关于

把b也看作一维，对应的的值始终为1， 这样相当于超平面和所有的点都会移动，直到超平面移动到到原点的位置。

这样，所有正例的x和<w, b>(下图中的w)的夹角都会小于90度，所有负例的x和<w, b>的夹角都会大于90度。，这样可以得出如下更新规则：



## 凸函数与凸集

1. 对于，如果任意，并且对于任意有：



（10）



则称为凸函数(下凸)。



1. 对于集合，如果并对于任何，点，则称是凸的。



当目标函数和约束集合是凸的时候，目标函数的局部最小值也是全局最小值，因为：对于任意，根据局部最小值的定义，存在充分接近1的使得：



（11）



根据凸集的性质可得：

（12）



因此得出是全局最小值。凸集和凸函数的性质保证了SVM的最优化问题是一个可解的问题。



## 带有限制条件的最优化问题

### 简介

其一般形式为如下所示：

（13）



1. 可以看到线性SVM的求解可以转换成此类问题。
2. 当目标函数是一次或二次函数，限制条件都是一次函数时，其就是中学期间学过的线性规划问题。求解此问题最著名的方法就是拉格朗日乘子法。（由于SVM中的超平面属于凸函数，因此其极值与最值相同）



### 拉格朗日乘子法

拉格朗日乘子法是在一种在等式限制条件下寻找目标函数极值的有效策略，例如：

（14）



拉格朗日函数为：

（15）



求解转换为求解，通过其求解过程可以看到的极值和的极值是相同的，求解过程如下：



（16）



可以看到若在取得极值，则根据上式有：这样：



（17）



很容易得出是的极值，至于此时的是不是具有约束条件的目标函数的极值[[1]](#footnote-0)。



简单的说就是在满足约束条件的的极小值点，的梯度一定可以由各个约束条件的梯度线性组合获得，这也就是如下公式存在的理由。



（18）



### KKT条件

不幸的是最初的拉格朗日乘子法不能处理限制条件含有不等式的情况，KKT条件使得具有不等式限制条件的最优化问题同样可以转换非约束条件函数求极值问题。

形式化描述如下：

（19）



其拉格朗日乘子法公式为：

（18）



注：，并且是凸集，函数为凸函数。



KKT条件将原问题最优化转化为最优化。



一个点是最优解的充要条件为，存在与使得：











从下文中可以看到，的最优解等价于对偶问题 的最优解。



令设是一个原问题的可行解，是上述问题的一个可行解。则：



证明

对于，从的定义出发，有：



因为的可行性意味着，同时的可行性意味着，



可见，对偶问题的上界由原问题给出：



那么，如果其中，并且，则与分别是原问题和对偶问题的解，同时



因为既然值是相等的，



中的不等号全部可以替换为等号。

#### KKT证明

##### 必要性



可以理解为最小值点的梯度可以由限制函数在此点的梯度线性组合而得到。这个问题的证明请参考 [Nash 96]

1. ，显然成立。



简单设，考虑公式1中由组成的那些梯度，最值得出现必然在可行域的边缘（或交叉点），这样，那些和这个边缘（交叉点）有关的其值必然是0，这样成立；对于那些和这个边缘（交叉点）无关的，必须令，因为公式中最小值点的梯度可以由限制函数在此点的梯度线性组合，“此点的梯度”只可能是那些“生成”此点的限制条件函数的梯度，还有一个重要原因此时的值等于的值，因此时，必然。



1. ，显然成立。



1. ，保证此点是最小值。



#### 充分性

#### http://blog.pluskid.org/?p=702

### 基于KKT条件的公式推导

与凸函数说明是最小值点，这样：



由于因此



是时得到的，此时由于， ，所以最大值时只能，于是：



进而：



由于鞍点位置上：=，（最值只可能是鞍点），因此：





这样SVM求解公式：



可以写成：



将下面两项代入上述目标函数得到：



进而变为如下形式：



这样SVM的求解便是求解，在结果中，每个可以反映出当前样本对SVM分类器的影响权重，可以看到只有那些支持向量的点的值不为0，其余点的均等于0.



这样求得的值，依据便可以求得的值，值可以通过以下公式计算出：



我们肯定能分别找到一个支持向量的正例和负例，此时有：



这样：



这样便得到了，分类过程如下：



其中SV代表支持向量点的集合。

可以看到超平面可以被那些支持向量的点表示出来。

#### 有用中间推导公式：



此公式理解：

对于支持向量的点：对于非支持向量的点：

由于支持向量点：满足：





其中：从前面得知这样，几何间隔可以改写为

### 对偶目标函数与原始目标函数的差值

当时，我们根据可以得到此时的，因此：



此差值称为可行间隙，可见只有在取得最优值时，可行间隙才为0.支持向量点的个数越少，其对应的超平面泛化能力越强。（有个误差公式p88）



根据的计算过程，在软最大间隔SVM中有：





因为此时有两个限制条件：

与

可以看出的点是存在误差的点，当点直接跨到线另一侧时，是一个很大的数，这样的点很有可能是噪音点。当时的点是那些完美支持向量的点。



因此，我们可以得出：



换句话说，如果不符合上述条件，那此时的解一定违反KKT条件。

软最大间隔SVM的可行间隙计算如下：





又因为：，所以：

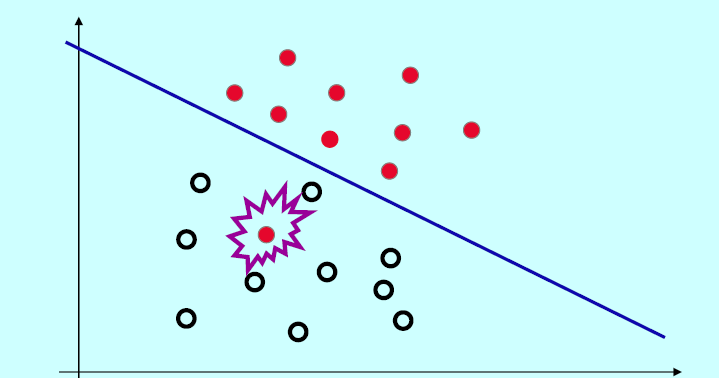


其中：



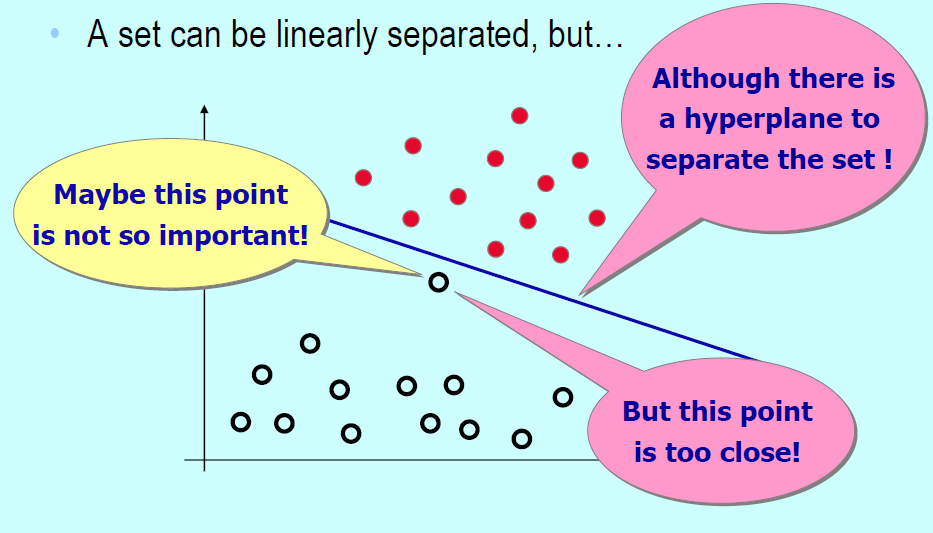
# 线性不可分问题

二分类问题的不总是可以完全线性可分的，比如下图：



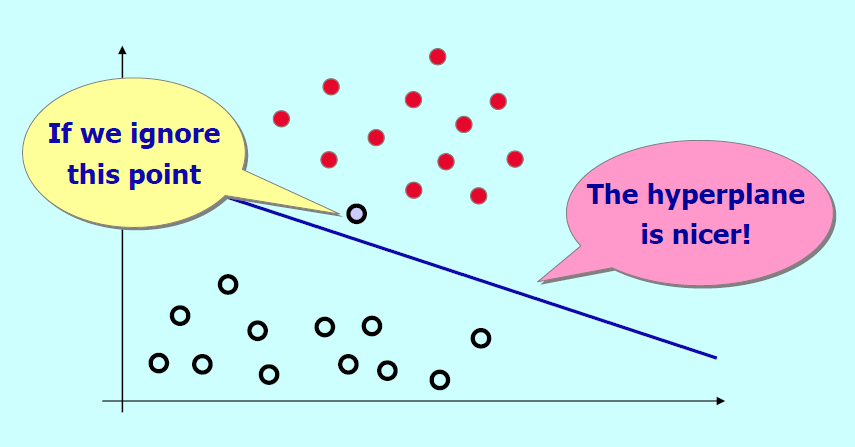
图[5] 线性不可分示意图

即使是线性可分的，可能由于噪音数据导致计算出一个很差的最大分割超平面，比如下图：



图[6]噪音数据对最大分割超平面的影响

如果我么可以这样选择最大分割超平面便会好很多，也就是忽略掉噪音数据，如下图所示：



图[7] 忽略噪音数据后计算出的最大分割超平面

这样便有了软间隔SVM，引入松弛变量（slack variable），限制条件有原始的：

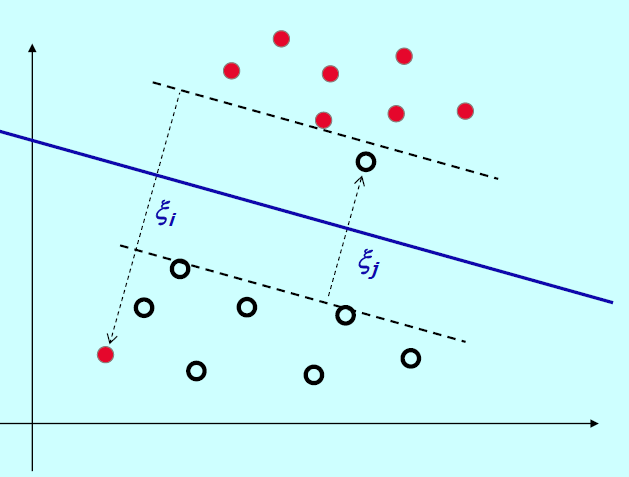




调整为：



其含义如下图所示：



图[8] 松弛变量的含义

这样软间隔SVM的最优化弓手由原始的：



调整为：



其中C是认为设定的数值，用来平衡最大间隔误差和训练误差。这样对偶问题由



调整为：



代入上式得到：



由于，且可以归结为所以上式



因此最终得到：

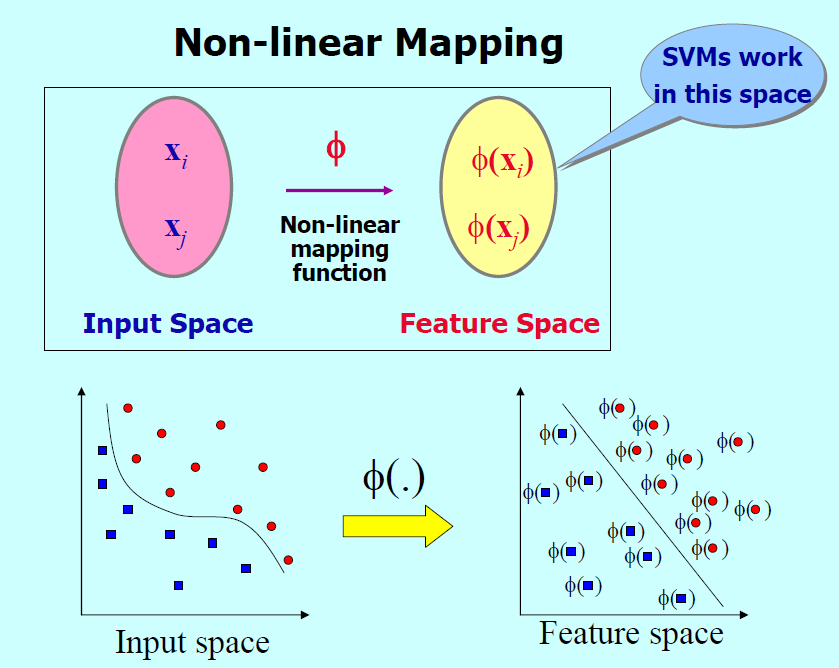


和非软间隔SVM相比，公式只增加了一个限制条件软间隔SVM只是用来处理噪音数据，对于线性不可分的情况还需要另外一个工具，那就是核函数，其原理如下。



## 核函数

通过一个非线性映射函数，将原空间中的样本映射到一个新空间，在这个新空间它是线性可分的。如下所示：



图[9]映射函数含义

幸运的是，在计算过程中我们不需要显示的计算新的映射空间的表示方法（其也是很难计算的，有时候新空间是无限维的），因为从计算公式中我们可以看到，其只需要计算的内积，这样我们如果直接能计算出的内积便可。此时计算公式调整为



引入核函数，另其



注意：核函数必须满足条件

核函数具有如下3个好处：

1. 将线性不可分转换成线性可分。
2. 不用显式表示新空间。
3. SVM可以隐式利用新空间。

### 常用的核函数

1. 多项式核函数



其中是用户设定的参数

1. 高斯核函数（RBF核函数）



其中是用户输入的参数

1. Sigmoid核函数





其中与是用户输入的参数

这样SVM最终的计算公式如下：

训练：



预测:



# SMO算法

## 简介

### 更新规则

每次迭代只优化两个点的最小子集。参考公式：



其中 ，这两个条件在推理迭代更新中使用到了。

不妨设两个点分别为与，由于需要满足，所以：



同时，这样就约束目标函数在一条直线上寻找最优解。我们不妨首先计算，由于与，的取值范围可以缩小到：



若则：





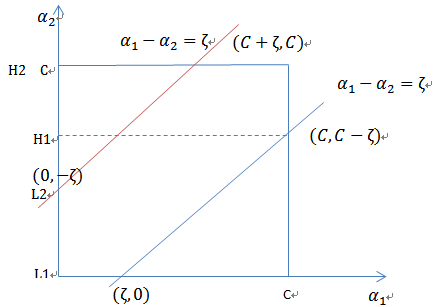
两边同乘以得：



所以：



参见下图：



若：



两边同乘以得：



所以：



定义

其中或

定义



当允许改变时，要使目标函数进一步最优，可以通过下面公式更新：



得到的需要处于范围，因此需要做一步剪辑，如下：



这样：



这个更新步骤的证明如下：

定义



目标公式



中都可以看做为常数，这样将作为目标得到：





由于，这意味着：其中，在计算出这个方程可以用来计算，这样将代入上式得到：



由于驻点满足：



因此：



注意的值只能是1与-1，公式转换的时候可以被利用到。



得出：



进而：



最终得到：



最后的到的要根据：



进行剪辑，确保的值在要求范围内。



在特殊情况下，可能不为正，如果核函数K不满足Mercer定理，那么目标函数可能变得非正定，可能出现负值。即使K是有效的核函数，如果训练样本中出现相同的特征x，那么仍有可能为0。SMO算法在不为正值的情况下仍有效。为保证有效性，我们可以推导出就是的二阶导数，，没有极小值，最小值在边缘处取到（类比），时更是单调函数了，最小值也在边缘处取得，而的边缘就是U和V。这样将和分别代入中即可求得的最小值，相应的还是也可以知道了。具体计算公式如下：



### 值的更新规则



由于所以有



两两想减得：



进而改造成：



可见：如果并且则，上述两种方法计算出的值相同。如果只有其中一个值处于，则取值对应的那个，因为另一个计算的要大。如果或，则处于与之间的任何数值都不违反下面条件，一般取值



我们可以得出：



违背上述条件也就是违背了KKT条件。

## LibSVM计算阈值b的方法

LibSVM并不在每次更新过程中计算 值，而是在所有计算完毕后在计算b值，对于完美支持向量的点，有，并且其中，这样



这样对任意完美支持向量的点都可以计算出一个b值出来，最后取其平均值作为最终的b值。

## LibSVM 计算的方法



## SMO算法停止条件

原始目标函数值与对偶目标函数值的间隙只有在驻点（优化点）才会消失，这个间隙称为可行间隙，其计算方式如下：

原始目标 - 对偶目标为



又因为：，所以：



其中



因而其停止条件可以用如下公式衡量：



检查它是否小于某个值，比如0.001。

## LibSVM选择的方法与停止条件

对于那些完美支持向量的点：



其中的定义在代码中。

进而我们可以看到。定义：





LibSVM先在中选择一个使值最大的，这是的最大值是最近接实际的b值的，然后选择，根据公式：



要使相对于变化的最大也就是最大化。

当和值很接近时，算法终止，因为这两个值都是一次一次在接近真实的b值，真实的b值得到了，算法自然停止了。

## 总结

SMO算法每步选择两个元素和，共同优化，在其它参数固定的前提下，找到这两个参数元素的最优值，并更新相应的向量。两个点的优化可以有解析解，并且没有矩阵操作，不需要再内存中存储核矩阵，核矩阵的引入可以提高速度，但会增加空间复杂度。



SMO算法虽然每次都选择两个参数进行优化，这两个参数起其中至少有一个参数是违背KKT条件的，经过一次这样的优化步骤，优化目标就会更优一步。依据Osuna的定理[[2]](#footnote-1)，这个过程可以保证收敛于最优解。为了加快收敛速度，SMO采用了一些启发式的规则来选择需要优化的两个参数。如下：

SMO采用了两种启发式规则，分别应用在两个元素和上。应用在上的启发规则体现在SMO算法的外围循环上。外围循环首先遍历整个训练集合判断当前点是不是违背如下的KKT条件。



如果找到一个违背此KKT条件的点，它变被选择用来做优化。此轮遍历整个训练数据结束后，外围循环将会缩小遍历范围，此时它只遍历那些值非0且非C的点集，同样检查当前点是不是违背上述的KKT条件，外围循环会循环到这些点全部都遵循KKT条件为止，此后外围循环会回到开始状态继续遍历整个训练集合。这样外围循环在“一次遍历整个训练数据集合”与“若干次遍历值非0且非C的点集”之间转换直到算法结束。这个启发规则保证算法优先处理那些最可能违背KKT条件的点，即值非0且非C的点，因为这样的点，即使在其它点的值发生改变后，其值也很可能不会改动。所以SMO算法首先保证那些值非0且非C的点集遵循KKT条件，再扫描整个训练数据集。



Notice that the KKT conditions are checked to be within of fulfillment. Typically, is set to be 10-3. Recognition systems typically do not need to have the KKT conditions fulfilled to high accuracy: it is acceptable for examples on the positive margin to have outputs between 0.999 and 1.001. The SMO algorithm (and other SVM algorithms) will not converge as quickly if it is required to produce very high accuracy output.

一旦第一个需要优化的点找到，SMO会依据另外一条启发式规则选择第二个需要优化的点。此条启发规则要最大化第一个点值的优化步伐，依据公式：



由于计算K比较耗时，SMO采用最大化的办法来最大化第一个点值的优化步伐。计算中SMO会缓存那些值非0且非C的点的E值。也就是说当*E*1 正值时，SMO选择最小的*E*2，当为负值时，SMO选择最大的*E*2.在一些非常规的情况下，SMO依据上述第二条启发规则选择出来的第二个点并不能优化目标函数，比如这两个点对应的特征向量是完全一样的，此时会造成目标函数为半正定的。在这种情况下，SMO使用第二条启发规则下的一条分级规则来寻找一对能优化目标函数的点，它是这样的，如果第二条启发规则找到的点不能给优化目标函数带来正面效果，SMO开始遍历所有值非0且非C的点，搜索出一个可以给优化目标函数带来正面效果的点，如果不能找到任何一个这样的点，SMO开始遍历训练集中所有的点，直到找到一个合适的点。这两种遍历都是起始于一个随机位置，为了避免SMO偏向于训练数据开始位置的点。如果最终也没能搜索到一个符合条件的点，此时第一个被选中的参数将被跳过，SMO继续依据第一条启发规则选择出下一个被优化的第一个点。



## 伪代码

target = desired output vector

point = training point matrix

procedure takeStep(i1,i2)

if (i1 == i2) return 0

alph1 = Lagrange multiplier for i1

y1 = target[i1]

E1 = SVM output on point[i1] – y1 (check in error cache)

s = y1\*y2

Compute L, H

// case 1: C=0 此时，ai=0.

// case 2: a1==0

if (L == H) return 0

k11 = kernel(point[i1],point[i1])

k12 = kernel(point[i1],point[i2])

k22 = kernel(point[i2],point[i2])

eta = k11+k22-2\*k12

if (eta > 0){

a2 = alph2 + y2\*(E1-E2)/eta

if (a2 < L) a2 = L

else if (a2 > H) a2 = H

}else{

Lobj = objective function at a2=L

Hobj = objective function at a2=H

if (Lobj < Hobj-eps) a2 = L

else if (Lobj > Hobj+eps) a2 = H

else a2 = alph2

}

if (|a2-alph2| < eps\*(a2+alph2+eps)) return 0

a1 = alph1+s\*(alph2-a2)

Update threshold to reflect change in Lagrange multipliers //b

Update weight vector to reflect change in a1 & a2, if SVM is linear

Update error cache using new Lagrange multipliers

Store a1 in the alpha array

Store a2 in the alpha array

return 1

endprocedure

procedure examineExample(i2)

y2 = target[i2]

alph2 = Lagrange multiplier for i2

E2 = SVM output on point[i2] – y2 (check in error cache)

r2 = E2\*y2 //f(x2)\*y2-1

if ((r2 < -tol && alph2 < C) || (r2 > tol && alph2 > 0)){

if (number of non-zero & non-C alpha > 1){

i1 = result of second choice heuristic

if takeStep(i1,i2) return 1

}

loop over all non-zero and non-C alpha, starting at a random point{

i1 = identity of current alpha

if takeStep(i1,i2) return 1

}

loop over all possible i1, starting at a random point{

i1 = loop variable

if (takeStep(i1,i2) return 1

}

}

return 0

endprocedure

main routine:

numChanged = 0;

examineAll = 1;

while (numChanged > 0 || examineAll){

numChanged = 0;

if (examineAll)

loop I over all training examples

numChanged += examineExample(I)

else

loop I over examples where alpha is not 0 & not C

numChanged += examineExample(I)

if (examineAll == 1) examineAll = 0

else if (numChanged == 0) examineAll = 1

}

Epsilon-SVR

# 简介

给定训练数据，的目标是寻找一个函数，使其对于，其值与真实值的差值不超过，也就是



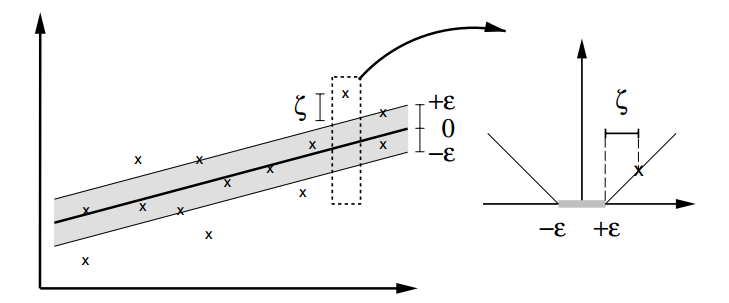
求解一个常见的方法是最小化的2-范数，同时满足约束条件，也就是：



为了适应现实中的噪音数据，类似于软间隔超平面，我们同样可以引入松弛变量，这样：



其中常数，用于在控制松弛变量所起的作用的大小。如下图



其中对于阴影部分的点，其对确定不起作用，对于其余部分的点，这些点是那些完美支持向量的点和存在误差的点。

# 公式推导

根据拉格朗日乘子法，公式



可以改写为



因为：



带入上式得：



这样：



根据KKT条件我们可以得知：

1. 对于的点是那些完美支持向量的点。
2. 是那些不起作用的点，即阴影部分中的点。
3. 是那些存在误差的点。

对于那些完美支持向量的点，有KKT调节得出：



因为，此时：



我们可以看到值可以这样计算得到：



# 求解过程

从前面我们可以得知，C-SVM的求解公式为：



从下面的推导过程我们可以看出，可以转换成C-SVM的样子，进而使用上面的方法求得最优值。只要能转换成如下C-SVM公式的格式，就可以转换成求解C-SVM问题.



其中Q是一个方阵，在C-SVM中其为的方阵，。

由：



推出：



其中：



注意：对于每条训练数据，其对应两个参数，其中。

这样我们便将转换成了C-SVM的形式，进而变可以使用上面的方法求解出最优的。

http://svms.org/tutorials/SmolaScholkopf1998.pdf

1. <http://en.wikipedia.org/wiki/Lagrange_multiplier>

   《模式识别》Theodoridis

   《支持向量机导论》P73 [↑](#footnote-ref-0)
2. <http://fourier.eng.hmc.edu/e161/lectures/gaussianprocess/node8.html>

   正定函数：除零点外恒为正数的标量函数 [↑](#footnote-ref-1)