SVM

支持向量机，其英文名为support vector machine，一般简称SVM。SVM是90年代中期发展起来的基于统计学习理论的一种机器学习方法，通过寻求结构化风险最小来提高学习机泛化能力，实现经验风险和置信范围的最小化，从而达到在统计样本量较少的情况下，亦能获得良好统计规律的目的。它是一种常见的有监督二类分类模型，其基本模型定义为特征空间上间隔最大化分类器，其学习策略便是间隔最大化.

**1.1 SVM基本模型及几何间隔最大化**

给定包含N个训练样本集合。其中yi表示样本属于某个类别。SVM分类学习的基本思想就是在样本集合S所表示的特征空间中寻找一个最优超平面，将不同的类别的样本区分开。假设样本集合S在特征空间的分布如下图所示。其中“”和“o”分别表示正负样本。可以看到能够将正、负样本区分开的超平面有很多，那么应该如何选择超平面呢？

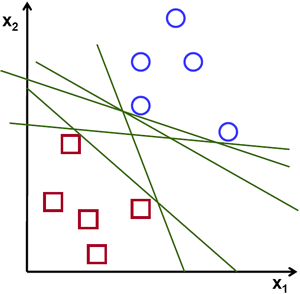


图 1.1.1

直观上讲，我们希望找到这样一个超平面，正、负样本到它的举例都比较远，如下图中间位置所示差平面。因为当待检测样本出现一定扰动时，这样的超平面仍能保证正确的分类结果。从统计学习角度考虑，希望寻找到的超平面对噪声具有最好鲁棒性，也就是说对待检测样本具有最好的泛化能力。而SVM就是一种寻找最优超平面的方法。



图 1.1.2

针对样本集合S上的二分类问题，SVM的目的是寻找一个最优超平面。直观上这个超平时所有样本到这个超平面的距离都“最远”。那么从数学计算角度来说，这个问题应该如何描述呢？

在样本空间中，一个 超平面可以用如下公式描述：

 （1）

其中，表示的是超平所对应的法向量，法向量的维度和样本的特征向量都是d;b表示的是超平面的偏移量。从而可以将任意一个超平明可以用一对参数表示。

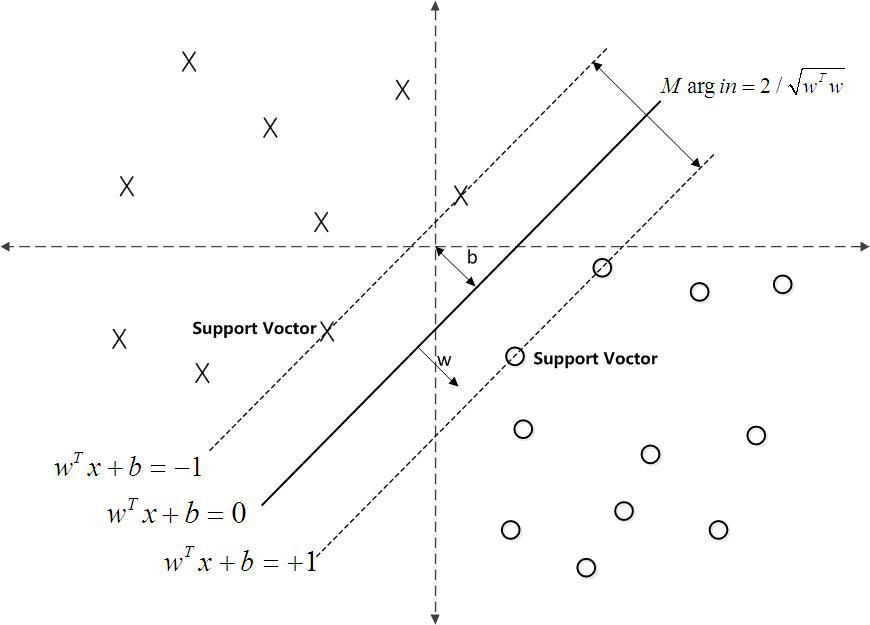


图 1.1.3

如果这个超平面能够将所有正负样本分开，则应如上图所示，有

 （2）

即所对集合S中的任意样本xi来说，若yi=+1，则该样本应该满足；否则该样本应该满足。观察到类别y的取值，方程（2）可以进一步写成：

 （3）

而这个样本点到这个超平面的距离可以由下面的公式计算：

 （4）

对于样本集合中的任意样本，我们都可以通过上述公式计算点到分类超平面的距离r，那么我们就得到一个样本点到超平面的距离的集合，我们将min(R)定义为超平面到样本点的距离，直观上讲这个距离表示了我们能够获取的正确分类的最大距离，这个距离越大，那么这个超平面对误差的“容忍”能力就越好，即对噪声的鲁棒性更高。而SVM寻找的就是能够正确分类样本的所有超平面中，使得min(R)最大的一个超平面。用数学表示就是

 （5）

而通过上面的方程（3）可知，那么min(R)能够取得的最小值为：

 （6）

那么SVM计算的目标就是求得满足公式（2）或者（3）的所有超平面中，使得公式（6）取得最大值的超平面。其对应的数学描述为：

 （7）

公式（7）中可以到，待优化的目标w处于目标函数的分母上，这会增加目标函数求解的难度。因此，一般情况下会将目标函数写成：

（8）

在上述公式中，将w转移到分子上，并且把最大化改成了最小化，从而实现与公式（7）的等价变化。公式（8）就是线性可分情况下SVM的数学公式的表达。

1.2线性可分SVM求解及对偶问题

通过1,1小节的推导，我们得到了SVM的基本模型的数学描述，如公式（8）。那么这个数学模型应该如何求解呢？公式（8）的目标函数为，这是一个凸二次优化问题。对于上述问题可以通过拉格朗日对偶性（Lagrange Duality）变换到对偶变量 (dual variable) 的优化问题，即通过求解与原问题等价的对偶问题（dual problem）得到原始问题的最优解。这样做的优点在于：（1）对偶问题往往更容易求解；（2）可以自然的引入核函数，进而推广到非线性分类问题。对公式（8）引入拉格朗日乘子，得到该问题的拉格朗日函数为：

 （9）

我们定义

 （10）

对于公式（10），容易验证，当某个约束条件不满足公式（8）时，例如有一个样本点，使得，那么有，这时只要令就可以得到，这说明此时公式（10）是没有最大值的解的。但当公式（8）中所有的约束条件都满足，即对于任意样本点，都使得成立，那么结合可以知道在公式（10）中，那么就有。从而可以推论出能取到的最大值就是。在原问题公式（8）中，我们希望，而在通过对公式（10）的分析，可以发现在满足原问题的约束条件（）下，那么就可以把原问题（8）转化成一个新的求解问题即：

（11）

在公式（11）中，有两个求极致的过程，第一次在的条件下求解的最大值；第二次在无约束条件下求解的最小值。一般说来在有约束条件写求解极值问题的难度要大于五约束条件下的求极值问题，因此我们将两次求极值的过程调换一下，得到

 （12）

公式（12）中，我们可以首先在无约束条件下计算极值，从而使第一次求极值的过程变得简单。但是一个更重要的条件是公式（11）和公式（12）必须是等价的，只有这样才能保证对公式（12）求得的极值就是公式（11）的极值，也就是原问题公式（8）的解。那么公式（11）和公式（12）是否等价呢？一般情况下这两个问题并不是等价的。假设公式（11）的最优值为p\*，公式（12）的最优值为d\*。那么应该有,这个在直观上的理解就是是在一系列的最小值寻找最大的那个值，而则是在一系列最大的值中寻找最小的那个。也就是说最大值中最小的一个总也比最小值中最大的一个要大。尽管公式（11）和公式（12）并不是一个等价问题，但是公式（12）却给出了公式（11）的最优值的下界。而且在满足有些条件的情况下，这两者是相等的，即。那么在这些条件下就可以通过求解公式（12）来求解公式（11）的最优解，从而使原问题（8）得到解决。公式（11）和公式（12）就称互为对偶问题。

上面说到，需要满足一定的条件，公式（11）的最优解才和公式（12）的最优解是等价的。那么这个条件是什么呢？这个条件就是KKT(**K**arush-**K**uhn-**T**ucker)条件。KKT条件的数学表达为，对于一个能够表示成下列标准形式的最优化数学模型

 （13）

如果它取到最优解x\*时，满足如下条件：

 （14）

则原问题就可以转化成对偶问题进行求解。

很幸运，我们的问题（11）就是满足KKT条件的。因此将问题（11）转化成对应的对偶问题进行求解。即求解公式（12），那么新问题就变成了：

 （15）

在公式（15）中，第一次求极值是无约束条件下的求极值问题，我们直接进行求导得到：

 （16）

令上述求导等于零，即

 （17）

得到：

（18）

将公式（18）带回公式中可以得到：

 （19）

此时的拉格朗日函数只包含了一个变量，那就是。如果能求出的值就可以得到问题（12）的最优值，而根据公式（18）可以求得w的值w\*，而参数b则可以根据如下公式求得

 （20）

现在我们的问题变成了对公式（19）进行求解，这时的目标函数变为：

 （21）

其中，表示的是由构成的向量。针对这个问题，目前最常用的求解方法就是SMO（Sequential Minimal Optimization）算法，我们将在后面的章节中介绍SMO方法的求解过程。

现在假设我们已经求解出了问题（21）的最优解，那么我们就可以计算出求解得到w\*和b\*的值，将带回到超平面对方的方程中，得到

 （22）

可以发现，超平面f（x）对应的方程只和训练集中的样本点相关。而对于的取值，我们在对公式（10）的分析中可以知道，对于所有满足的样本，其对应的值必然为0，否则就会出现公式（10）无法取得极值的问题。也就是说只有满足的才是有值的，从而可以进一步发现公式（22）中的只有一部分样本点对超平面的参数起作用。我们再进一步看看这些对超平面其作用的点具有什么性质，从公式可以得知这些点满足，也就是说当=-1时，=-1，当=+1时，=+1，从图1.1.3可以看出这些点刚好位于超平面的最左和最右的位置上。也可以说超平面的参数有这些点来控制，SVM中将这些不为0的点称之为支持向量，这也就是支持向量机名称的由来。而且在公式（22）中可以发现，在判断一个新的样本点x是否为正样本（计算函数f(x)的值）时，我们需要计算的是新的采样点x和这些支持向量的内积，这种计算可以使SVM很自然地引入核（kernel）的概念，从而将线性SVM映射到非线性空间，从而实现SVM对非线性问题的求解，我们将在后面的章节中介绍基于核的SVM方法。在本小节的最后，我们来分析一下此时样本点到超平面f(x)的最大距离为，因为w\*的值由这些支持向量过程，所以这个最大间隔也有这些支持向量唯一确定了。

**1.2 核函数SVM**

**1.2.1 核函数SVM**

通过1.1小节的学习，我们了解到了SVM的基本数学模型和支持向量的基本概念。但是目前的SVM的性能还是比较弱的，因为目前的SVM只能处理线性可分的数据集。但是在现实应用时，样本的数据并不一定是线性可分。如下图中左边的例子，其中“x”表示的是负样本，“o”表示的是正样本。



对于这样的样本分布来说，我们很难在二维空间下寻找到一条直线将数据完全分开，但是仔细观察可以发现这些样本的分布类似一个圆环形状，只是圆环的半径大小不一样而已，因此如果我们可以根据输入数据计算出类似半径的一个计算量，那么这个问题应该就可以很好地解决了。利用在本样例中，我们的每一个样本数据输入的是一个二维特征向量， 即在任意中，。而我们希望计算的类似半径的一个计算量为，从而将原始数据表示成：。这个三维的样本数据在三维空间中的分布将会是下图中右边所示的样本，这样我们就可以在三维空间中利用1.2小节学习到线性可分SVM将样本区分开来。

上述例子中是因为我们可以观察到数据在空间中的分布才想到将数据映射到三维空间中进行计算，但在很对实际的应用中，数据的维度往往很高，而且分布也很复杂，我们几乎无法观测其在多维空间中的分布也几乎不可能人为的想到一些“合适”的映射到高位空间的方法将数据样本分开。那么能否让SVM自己处理这个情况呢？答案是肯定的，为了完成这个目的我们需要在原来的线性SVM中引入“核”（Kernel）的概念。

我们用表示将样本的特征映射到高维空间的一个表示，在上面的例子中表示将转化到一个映射关系。可以看到这种映射只是通过对样本原有数据进行计算得到一些新的维度的特征，经过映射后样本的特征向量还是一个向量，只是维度有所不同而已。将经过特征映射后的超平面表示为：

 （23）

其中和为模型的参数，为人工定义的一种映射关系。类似公式（8），原问题的数学描述为

 （24）

同样按照1.2小节的思路将其转化成对偶问题，得到对偶问题的数学描述为：

 （25）

公式（25）中表示两个向量和映射到高维空间后的内积，更一般化的表示为

 （26）

其中表示两个样本在核（Kernel）空间下的内积，将公式（26）带回到公式（25）中，得到在SVM在核空间下的表示

（26）

而通过求解对偶问题得到原问题的解为：

 （27）

从公式（24）到公式（26）只是做了形式化的变化，其数学描述的本质还是一样的，那我们为什么还要做这样的变化呢？其中一个重要的原因就是将高维空间下内积计算表示成为核空间下的内积计算后可以使我们不在关系的具体的表达形式，而只需要关系核空间下的内积计算的具体表达形式，从而大大化简了计算的难度。为什么这样说呢？

我们回到本小节最开始的例子，在这个例子中，我们将定义为这样一个映射关系：

 （28）

其中，，那么此时的高维空间的内积为：

 （29）

根据映射关系的定义，将和带入公式（29），得到

 （30）

通过对公式（29）的计算可以发现，计算高维空间的内积可以分成两个步骤，首先计算内积在高维空间下的表示，即公式（29），但这个表示中还存在一些我们自己计算出来的特征值，因此还需要将映射关系所定义为“额外的”特征值的计算表达式带回到公式（29）中得到公式（30），通过上述两个步骤后，就将高维空间的内积的计算完全用原始数据的特征值表示出来了。我们为什么需要将高维空间中内积的计算表示成原始数据的特征值的形式呢？其中一个直观的原因如果不表示成原始数据的特征值的形式我们就要存储这些“计算”出来的特征值，如果将原始数据的特征维度映射到一个很高的维度，比如映射到几千维、甚至几万维，那么势必会带来不必要的存储空间的开销。另外一个更重要的原因是，如果我们想把这个特征维度提高到无限维怎么办？在这种情况下就必须使用核（Kernel）的概念。

再看公式（29）到（30）的计算，可以发现，在内积算法的过程中只是起到了中间过渡的作用，其实完全没有必要写出的具体表达形式，可以直接将内积的计算公式表示成：

 （31）

而且不难发现，不但在内积的计算上我们可以直接定义其表达式，而且在最后的超平面的表达形式上我们也可以直接使用这个内积的计算公式，如公式（27）。

现在我们将直接定义内积的计算公式，而不用关心其具体映射关系的表达形式。上面的例子中，我们只是将内积的计算表达成更加复杂的多项式的形式，那么可不可以定义成更加复杂的情况呢？当然可以，比如我们可以将内积定义成如下形式：

 （32）

其中表示的是底数e的指数运算，表示的是两个向量之间的欧式距离。在这个内积的计算公式下，我们无法直接写出映射关系的表达形式，但这并不影响我们关于内积的计算。如果一定要分析这个内积是两个多少维的向量进行计算得到，那么你会发现这是两个无限维的特征向量进行内积计算得到的。对这个内积进行泰勒（Taylor）展开得到：

 （33）

可以看到表示的是一个无穷数的和，而其中每一项展开后都是原始特征值的一个n维多项式，而对于多项式中的每一项我们都需要用内积的形式表示，因为n是无穷大的，因此要表示这个内积的维度也会是无穷维的。这个例子就显示出了定义核函数另一个重要作用，即我们可以把原始特征维度映射成无穷维。那么我们为什么要将原始数据映射成无穷维呢？因为在很多分类的任务中，样本在原始的特征维度下是线性不可分的，但我们又难以通过观察的方式确定需要将样本映射到多高的维度才能将正负样本区分开来，因此一个简单的方式就是直接将原始特征映射到无穷维，让SVM自动地帮我们在高维空间中计算出区分样本的超平面。

至此，我们成功地通过引入核函数来使SVM能够胜任线性不可分问题。而核函数之所以可以引入进来，一个重要的原因是SVM得到超平面可以写成支持向量点的内积的形式。而核函数的引入也极大的增加的SVM的实用性，使得SVM成为一个重要的分类算法。

**1.2.2 核函数**

通过上一小节的分析可以知道，核函数对SVM处理线性不可分样本有着重要的作用，那么什么样的内积的定义才能作为核函数呢？我们引入问题：给定一个函数，我们能否使用来替代计算，也就说，是否能够找出一个，使得对于所有的和，都有？比如给出了，是否能够认为是一个有效的核函数。

下面来解决这个问题，给定N个训练样本，每一个样本对应一个特征向量。那么，我们可以将任意两个和带入中，计算得到。这样我们一共可以得到N\*N个这样的值，用矩阵表示为：



如果假是有效的核函数，那么根据核函数定义

 （34）

可见，矩阵应该是个对称阵。更强的结论是这个是一个半正定矩阵。也就说如果我们定义的是一个半正定矩阵，那么这个就是一个有效的核函数，我们可以放心的实用在SVM中。

更一般性的核函数有Mercer定理给出：

如果函数是上的映射。如果是一个有效核函数（也称为Mercer核函数），那么当且仅当对于训练样本，其相应的核函数矩阵是对称半正定的。

通过上面的分析，我们就可以判断自己定义的核函数是否为一个有效的核函数了。但是既然核函数的本质是将样本映射到高维空间后来处理在原始特征空间线性不可分的情况，那么映射后特征空间的好坏对SVM能否有效地区分样本至关重要。而且在很多核函数在定义的时候我们无法写出对应的映射函数，因此也就无法估计出映射后特征空间的形态。因此在使用基于核函数的SVM方法时，核函数的选取变得十分重要。

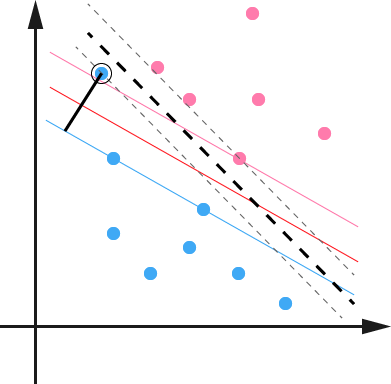
下面列举了一些常见的核函数：

|  |  |  |
| --- | --- | --- |
| 名称 | 表达式 | 参数说明 |
| 多项式核函数 |  | d为多项式的次数，为参数 |
| 高斯核函数 |  | 高斯核的带宽 |
| Sigmoid核函数 |  | tanh为双曲正切函数，为参数 |

此外，还可以通过不同核函数的组合还定义新的核函数，利用将不同的核函数进行相加或者相乘运算得到新的核函数。不过这种组合核函数在实际的应用中并不常用，因为组合核函数需要引入更多的计算量，而且组合的模式难以确定，例如利用线性加权组合出一个新的核函数，那么如何选择最优的权重系数呢？这势必会引入额外的计算量。而且在处理实际的问题时，往往会尝试使用不同的核函数来确定最终的选择。

1.4 松弛变量SVM

在最开始讨论SVM的时候，我们就假定，数据是线性可分的，亦即我们可以找到一个可行的超平面将数据完全分开。后来为了处理非线性数据，使用kernel方法对原来的线性 SVM 进行了推广，使得非线性的的情况也能处理。虽然通过映射 将原始数据映射到高维空间之后，能够线性分隔的概率大大增加，但是对于某些情况还是很难处理。例如可能并不是因为数据本身是非线性结构的，而只是因为数据有噪音。对于这种偏离正常位置很远的数据点，我们称之为奇异点（outlier），在我们原来的 SVM 模型里，outlier的存在有可能造成很大的影响，因为超平面本身就是只有少数几个支持向量（support vector）组成的，如果这些支持向量（support vector）里又存在奇异点的话，那么将会对SVM的分类性能产生很大的影响。下图中蓝色和粉色分别代表不同的样本，其中蓝色最左侧点（用黑色圆圈标出）的分布位置和其他蓝色样本的位置很不同，这在实际问题是常见的，可能是由于采样时噪声的影响造成的。如果我们一定要是使SVM在这种情况也能将全部样本分开，则可以得到图中黑色虚线标示出来的超平面，但是可以看出这个超平面的间隔并不大，这将会降低算法的泛化能力。但是如果我们排除这个奇异点再进行SVM学习，则可以得到红色标示的超平面，可以看到这个超平面的间隔要明显大于考虑奇异点时得到的超平面，因此可能使我们得到的SVM模型具有更加的泛化能力。



通过上面的分析，可以看到为了在有噪声的样本中尽量提高SVM的泛化能力，我们需要尽量的将奇异点排除在SVM的学习过程中。但是很遗憾，在实际问题中，我们不能直接观察出哪些点是奇异点。在这种情况下怎么办呢？首先，来看一下这些奇异点在SVM学得最优的超平面下的表现。用表示这个奇异点，用来表示这超平面，因为这个奇异点已经在支持向量意外，因此一定有：，即当前超平面无法将这个点正确的区分开来。但是如果我们能使这个奇异点满足这个条件，那么我们还是可以利用这个最优的超平面将这个歧义样本区分看来。当然这个条件太宽泛了，因此我们可以定义一个范围，例如要求对于所有样本点满足：

 （35）

其中。

现在看公式（35），因为的引入，使我们不在要求样本满足，而是对满足的条件进行一定的松弛，这样我们就可以用过松弛变量来对每一个点进行松弛，而不必考虑哪些点是奇异点了。但在实际的应用时，我们也无法寻找每一个松弛变量的最优值，这也是没有必要的，因为我们更希望在整体上数据上越小越好。一个极端的例子就是所有的=0，这个时候的SVM的就和没有松弛变量的情况是一样的了。那么如何从整体上考虑这个松弛变量的值呢？还记SVM的目标函数吧——，我们在目标函数中引入对松弛变量的约束，得到新的目标函数：

 （36）

其中C是一个参数，用于控制目标函数中两项（“寻找最大间隔的超平面”和“保证数据点偏差量最小”）之间的权重。注意，其中ξ是需要优化的变量，而C是一个事先确定好的常量。完整SVM公式为：

 （37）

公式（37）就是引入松弛变量后的SVM数学表达式。

在1.1小节中，我们将线性可分SVM的数学求解问题转化成为其对偶问题，那么引入松弛变量后的SVM的对偶问题时什么呢？用类似1.1小节中的方法，引入拉格朗日因子，得到

 （38）

分别对求导，得到

 （39）

将公式（39）带入公式（37）的目标函数中，得到：

 （40）

可以发现，和线性SVM的目标函数一样。但是这时却多了一个约束条件。而作为拉格朗日因子，要求，因此的条件就变成了。所以引入松弛变量后的SVM变成：

 （41）

和之前的结果对比一下，可以看到唯一的区别就是现在对偶问题中对多了一个上限 C 。而核（Kernel）化的非线性形式也是一样的，只要把换成即可。这样一来，一个完整的，可以处理线性和非线性并能容忍噪音和奇异点的支持向量机数学表达式即为：

 （42）

1.5 SMO算法

既然我们知道SVM算法和其对偶问题的数学表达形式，那么SVM该如何求解呢？1.1小节提到SMO算法求解，这一小节我们就介绍SMO算法。

SMO算法由Microsoft Research的John C. Platt在1998年提出，并成为最快的二次规划优化算法，特别针对线性SVM和数据稀疏时性能更优。

回顾SVM 对应的对偶问题优化问题为：

 （43）

在该问题中，和C都是已知数,我们要解决的是在参数上求最大值W。因为有约束条件的存在，使得不同的之间的却只并不是独立，SMO的思路就是每次选择两个参数，如和，对于其他所有参数我们假定它们是固定，那么就可以利用梯度下降法对w求的最小化，利用我们对利用梯度下降法求得一个w的最小值，并更新，则可以利用约束条件，求得得值。SMO之所以高效就是因为在固定其他参数后，对一个参数优化过程很高效。SMO算法具体的计算过程如下：

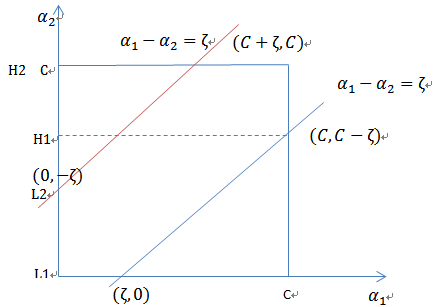
假设我们选取了初始值满足了问题中的约束条件。接下来，我们固定，这样W就是和的函数。并且和满足条件：

 （44）

由于都是已知固定值，因此为了方面，可将等式右边标记成实数值，则公式（44）变为

 （45）

当和异号时，也就是一个为1，一个为-1时，他们可以表示成一条直线，斜率为1。如下图：



横轴是，纵轴是，和既要在矩形方框内，也要在直线上，因此

 （46）

同理，当和同号时，

 （47）

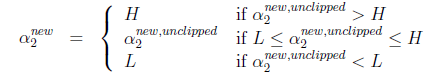
然后我们将和表示：

 （48）

代入W中，得

 （49）

展开后W可以表示成这样形式的二次方程。其中X,Y,Z是固定值。这样，通过对W进行求导可以得到，但同时受到公式（46）或者（47）的约束，因此最后求得的结果根据如下公式计算：



然后可以根据公式（44）求得。这样我们就可以一次更新公式（43）中所有的，从而求得最后的优化结果。

1.4 SVM处理多类分类问题和回归问题

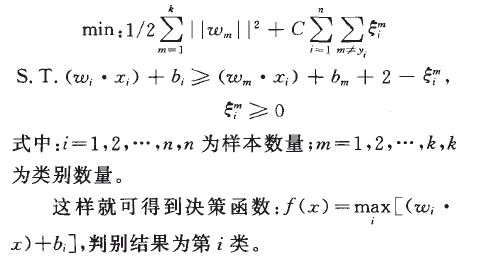
1.4.1 多类问题

　　SVM算法最初是为二值分类问题设计的，当处理多类问题时，就需要构造合适的多类分类器。目前，构造SVM多类分类器的方法主要有两类

　　（1）直接法，直接在目标函数上进行修改，将多个分类面的参数求解合并到一个最优化问题中，通过求解该最优化问题“一次性”实现多类分类。这种方法看似简单，但其计算复杂度比较高，实现起来比较困难，只适合用于小型问题中；

（2）间接法，主要是通过组合多个二分类器来实现多分类器的构造，常见的方法有one-against-one和one-against-all两种。

对于方法（1），其数学表达式可以写成：



虽然这种种指导思想看起来简单，但由于它的最优化问题求解过程太复杂，计算量太大，实现起来比较困难，因此未被广泛应用。本文不作过多介绍。

相比于方法（1），方法（2）在无论在应用和求解上都更加灵活。这方法在解决多分类问题时，有四种主要的划分解构：

**方法一：一对多法（one-versus-rest,简称OVR SVMs）：**

训练时依次把某个类别的样本归为一类,其他剩余的样本归为另一类，这样k个类别的样本就构造出了k个SVM。分类时将未知样本分类为具有最大分类函数值的那类。

假如我有四类要划分（也就是4个Label），他们是A、B、C、D。于是我在抽取训练集的时候，分别抽取

　　（1）A所对应的向量作为正集，B，C，D所对应的向量作为负集；

　　（2）B所对应的向量作为正集，A，C，D所对应的向量作为负集；

　　（3）C所对应的向量作为正集，A，B，D所对应的向量作为负集；

　　（4）D所对应的向量作为正集，A，B，C所对应的向量作为负集；

使用这四个训练集分别进行训练，然后得到四个训练结果文件。在测试的时候，把对应的测试向量分别利用这四个训练结果文件进行测试。最后每个测试都有一个结果f1(x),f2(x),f3(x),f4(x)。于是最终的结果便是这四个值中最大的一个作为分类结果。这种方法有种缺陷,因为训练集是1:M,这种情况下存在biased.因而不是很实用。可以在抽取数据集的时候，从完整的负集中再抽取三分之一作为训练负集。

**方法二：一对一法（one-versus-one,简称OVO SVMs或者pairwise）**

　　其做法是在任意两类样本之间设计一个SVM，因此k个类别的样本就需要设计k(k-1)/2个SVM。当对一个未知样本进行分类时，最后得票最多的类别即为该未知样本的类别。

假设有四类A,B,C,D四类。在训练的时候我选择A,B; A,C; A,D; B,C; B,D;C,D所对应的向量作为训练集，然后得到六个训练结果，在测试的时候，把对应的向量分别对六个结果进行测试，然后采取投票形式，最后得到一组结果。

　　投票是这样的：

　　A=B=C=D=0;

　　(A,B)-classifier 如果是A win,则A=A+1;otherwise,B=B+1;

　　(A,C)-classifier 如果是A win,则A=A+1;otherwise, C=C+1;

　　...

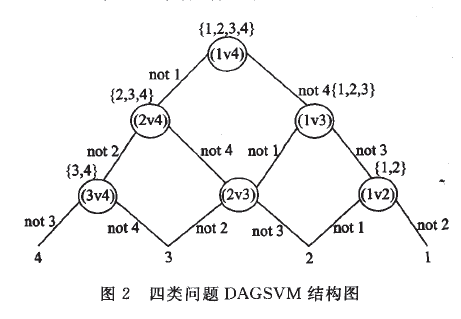
　　(C,D)-classifier 如果是A win,则C=C+1;otherwise,D=D+1;

　　The decision is the Max(A,B,C,D)

这种方法虽然好,但是当类别很多的时候,model的个数是n\*(n-1)/2,代价还是相当大的。

**方法三：有向无环图（DAG）**

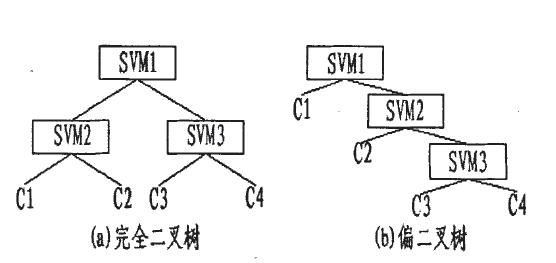
DAG-SvMS是由PIatt提出的决策导向的循环图DAG导出的，是针对“一对一"SvMS存在误分，拒分现象提出的。这种方法的训练过程类似于“一对一”方法，k类别问题需要求解k(k-1)／2个支持向量机分类器，这些分类器构成一个有向无环图。该有向无环图中含有k(k-1)／2个内部节点和k个叶结点，每个节点对应一个二类分类器。其解构如下图所示



DAG-SVMS简单易行，只需要使用k一1个决策函数即可得出结果，较“一对一"方法提高了测试速度，而且不存在误分、拒分区域；另外，由于其特殊的结构，故有一定的容错性，分类精度较一般的二叉树方法高。然而，由于存在自上而下的“误差积累”现象是层次结构固有弊端，故DAG-SVMS也逃脱不掉。即如果在某个结点上发生了分类错误，则会把分类错误延续到该结点的后续结点上。

**方法四：策树方法**

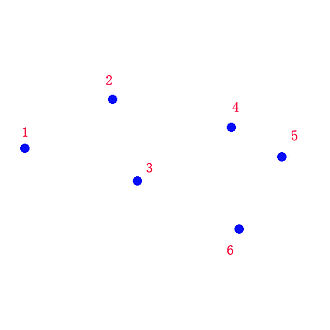
决策树的基本思想是从根节点开始，采用某种方法将该节点所包含的类别划分为两个子类，然后再对两个子类进一步划分，如此循环，直到子类中只包含一个类别为止，这样，就得到了一个倒立的二叉树。最后，在二叉树各决策节点训练支持向量机分类器，实现对识别样本的分类。决策树支持向量机多分类方法有很多种，不同方法的主要区别在于设计树结构的方法不同。



完全二叉树结构分类时使用的平均分类器数目为log2k，偏二叉树使用的平均分类器数为(k+1)／2-1／k，具有其他层次结构的二叉树使用的分类器平均值介于二者之间。完全二叉树分类时所需要的分类器数目最少，因此具有较少支持向量的完全二叉树的分类器速度也是较快的。

**排序支持向量机（Ranking SVM）**

在有些问题中，我们往往需要对一组序列进行排序，如集合中，满足如下序关系，如下图所示



一种直观的方法就是利用**one-versus-rest，**即我们对每一个点训练SVM，使这个超平面将大于和小于的样本分开。即对于任意样本点，将所有大于的样本作为正样本，所有小于的样本作为负样本，从而根据得到一组正负样本，即可以根据这组样本训练得到一个SVM。按照这种思路，如果要将左右样本分开，需要训练n个SVM，这种方法在样本数量较大时显然是不可取的。

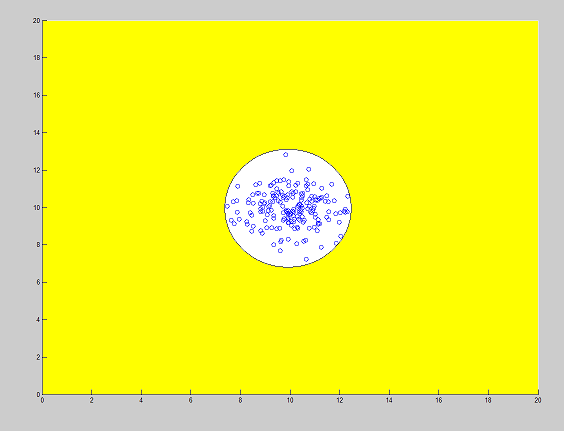
现在我们换一种思路处理这个问题，对于有序关系的样本，我们并不一定要知道直接排序它的排序位置，如果我们知道任意两个样本之间的序关系，则必要时可以根据这个样本和其他所有样本的序关系在确定该样本的位置，因此我们可以让SVM只是学习样本之间的序关系就行了，因此我们重新一组训练样本，即：每对(xi, xj) 满足xi < xj,定义（xi,xj,+1）

从而定义新的目标函数



一类支持向量机（one-class SVM）

在生成式模型中，我们往往会遇到一类问题，即在集合中，所有样本都属于一类，如下图所示：



我们希望用SVM做一类分类，那么新的SVM的目标函数为



SVM的应用——目标识别