[**berkeleysong**](http://www.cnblogs.com/berkeleysong/)

关于研究机器学习 数据挖掘 Linux后台开发的心路历程

[**svm 算法通俗介绍**](http://www.cnblogs.com/berkeleysong/articles/3251245.html)

[**http://blog.csdn.net/v\_july\_v/article/details/7624837**](http://blog.csdn.net/v_july_v/article/details/7624837)

作者：July、pluskid ；致谢：白石、JerryLead

出处：结构之法算法之道blog。

前言

第一层、了解SVM

  1.0、什么是支持向量机SVM

  1.1.、线性分类

    1.1.1、分类标准

    1.1.2、1或-1分类标准的起源：logistic回归

    1.1.3、形式化标示类

  1.2、线性分类的一个例子

  1.3、函数间隔Functional margin与几何间隔Geometrical margin

    1.3.1、函数间隔Functional margin

    1.3.2、点到超平面的距离定义：几何间隔Geometrical margin

  1.4、最大间隔分类器Maximum Margin Classifier的定义

  1.5、到底什么是Support Vector

第二层、深入SVM

  2.1、从线性可分到线性不可分

    2.1.1、从原始问题到对偶问题的求解

    2.1.2、线性不可分的情况

  2.2、核函数Kernel

    2.2.1、如何处理非线性数据

    2.2.2、特征空间的隐式映射：核函数

    2.2.3、几个核函数

    2.2.4、核函数的本质

  2.3、使用松弛变量处理 outliers 方法

  2.4、小结

第三层、证明SVM

  3.1、线性学习器

    3.1.1、感知机算法

    3.1.2、松弛变量

  3.2、最小二乘法

  3.3、核函数特征空间

  3.4、SMO算法

  3.5、SVM的应用

    3.5.1、文本分类

参考文献及推荐阅读

**前言**

    动笔写这个支持向量机(support vector machine)是费了不少劲和困难的，从5月22日凌晨两点在微博上说我要写了，到此刻真正动笔要写此文，中间竟然隔了近半个月(而后你会发现，我写完此文得花一个半月，修改完善又得再花一个月，故前后加起来至8月底，写这个SVM便要花足足近3个月)。原因很简单，一者这个东西本身就并不好懂，要深入学习和研究下去需花费不少时间和精力，二者这个东西也不好讲清楚，尽管网上已经有朋友已经写得不错了(见文末参考链接)，但在描述数学公式的时候还是显得不够。得益于同学白石的数学证明，我还是想尝试写一下，希望本文在兼顾通俗易懂的基础上，真真正正能足以成为一篇完整概括和介绍支持向量机的导论性的文章。

    本文作为[Top 10 Algorithms in Data Mining](http://blog.csdn.net/v_july_v/article/category/1061301)系列第二篇文章，将主要结合支持向量机导论、数据挖掘导论及网友Free Mind的支持向量机系列而写(于此，还是一篇学习笔记，只是加入了自己的理解，有任何不妥之处，还望海涵)，宏观上整体认识支持向量机的概念和用处，微观上深究部分定理的来龙去脉，证明及原理细节，力求深入浅出 & 通俗易懂。

    在本文中，你将看到，理解SVM分三层境界，

第一层、了解SVM(你只需要对SVM有个大致的了解，知道它是个什么东西便已足够)；

第二层、深入SVM(你将跟我一起深入SVM的内部原理，通宵其各处脉络，以为将来运用它时游刃有余)；

第三层、证明SVM(当你了解了所有的原理之后，你会有大笔一挥，尝试证明它的冲动)；

    以此逐层深入，从而照顾到水平深浅度不同的读者，在保证浅显直白的基础上尽可能深入，还读者一个较为透彻清晰的SVM。

    同时，阅读本文之前，请读者注意以下两点：

若读者用~~IE6~~浏览器阅读本文，将有大部分公式无法正常显示(显示一半或者完全无法显示)，故若想正常的阅读本文请尽量使用chrome等浏览器，谢谢大家。

本文中出现了诸多公式，若想真正理解本文之内容，我希望读者，能拿张纸和笔出来，把本文所有定理.公式都亲自推导一遍或者直接打印下来，在文稿上演算（读本blog的最好办法便是直接把某一篇文章**打印**下来，随时随地思考.演算.讨论）。

    Ok，还是那句原话，有任何问题，欢迎任何人随时不吝指正 & 赐教，感谢。

**第一层、了解SVM**

***1.0*、什么是支持向量机SVM**

    要明白什么是SVM，便得从分类说起。

    分类作为数据挖掘领域中一项非常重要的任务，目前在商业上应用最多(比如分析型CRM里面的客户分类模型，客户流失模型，客户盈利等等，其本质上都属于分类问题)。而分类的目的则是学会一个分类函数或分类模型(或者叫做分类器)，该模型能吧数据库中的数据项映射到给定类别中的某一个，从而可以用于预测未知类别。

    其实，若叫分类，可能会有人产生误解，以为凡是分类就是把一些东西或样例按照类别给区分开来，实际上，分类方法是一个机器学习的方法，分类也成为模式识别，或者在概率统计中称为判别分析问题。

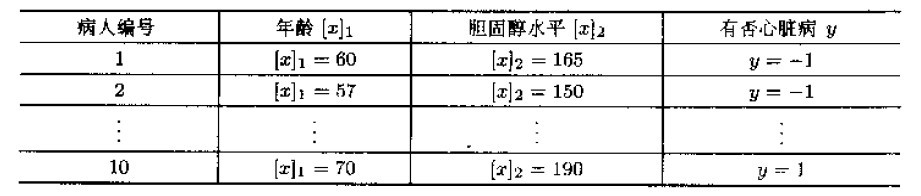
    你甚至可以想当然的认为，分类就是恰如一个商场进了一批新的货物，你现在要根据这些货物的特征分门别类的摆放在相关的架子上，这一过程便可以理解为分类，只是它由训练有素的计算机程序来完成。

    来举个例子，比如心脏病的确诊中，如果我要完全确诊某人得了心脏病，那么我必须要进行一些高级的手段，或者借助一些昂贵的机器，那么若我们没有那些高科技医疗机器，怎么判断某人是否得了心脏病呢？

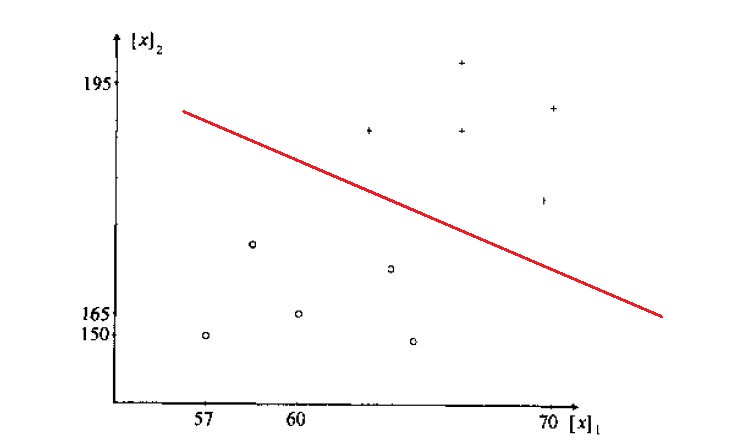
    当然了，古代中医是通过望、闻、问、切“四诊”，但除了这些，我们在现代医学里还是可以利用一些比较容易获得的临床指标进行推断某人是否得了心脏病。如作为一个医生，他可以根据他以往诊断的病例对很多个病人(假设是500个)进行彻底的临床检测之后，已经能够完全确定了哪些病人具有心脏病，哪些没有。因为，在这个诊断的过程中，医生理所当然的记录了他们的年龄，胆固醇等10多项病人的相关指标。那么，以后，医生可以根据这些临床资料，**对后来新来的病人通过检测那10多项年龄、胆固醇等指标**，以此就能**推断或者判定病人是否有心脏病**（虽说这个做法不能达到100%的标准，但也能达到80、90%的正确率），而这一根据以往临场病例指标分析来推断新来的病例的技术，即成为分类classification技术。

    OK，既然讲到了病例诊断这个例子，接下来咱们就以这个例子来简单分析下SVM。

假定是否患有心脏病与病人的年龄和胆固醇水平密切相关，下表对应10个病人的临床数据(年龄用[x1]表示，胆固醇水平用[x2]表示)：



    这样，问题就变成了一个在二维空间上的分类问题，可以在平面直角坐标系中描述如下：根据病人的两项指标和有无心脏病，把每个病人用一个样本点来表示，有心脏病者用“+”形点表示，无心脏病者用圆形点，如下图所示：



    如此我们很明显的看到，是可以在平面上用一条直线把圆点和“+”分开来的。当然，事实上，还有很多线性不可分的情况，下文将会具体描述。

    So，本文将要介绍的支持向量机SVM算法便是一种分类方法。

所谓支持向量机，顾名思义，分为两个部分了解，一什么是支持向量(简单来说，就是支持 or 支撑平面上把两类类别划分开来的超平面的向量点，下文将具体解释)，二这里的“机”是什么意思。我先来回答第二点：这里的“机（machine,机器）”便是一个算法。在机器学习领域，常把一些算法看做是一个机器，如分类机(当然，也叫做分类器)，而支持向量机本身便是一种监督式学习的方法(至于具体什么是监督学习与非监督学习，请参见此系列[Machine L&Data Mining](http://blog.csdn.net/v_july_v/article/category/1061301)第一篇)，它广泛的应用于统计分类以及回归分析中。

   支持向量机(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。

    对于不想深究SVM原理的同学(比如就只想看看SVM是干嘛的)，那么，了解到这里便足够了，不需上层。而对于那些喜欢深入研究一个东西的同学，甚至究其本质的，咱们则还有很长的一段路要走，万里长征，咱们开始迈第一步吧(相信你能走完)。

***1.1*、线性分类**

    OK，在讲SVM之前，咱们必须先弄清楚一个概念：线性分类器(也可以叫做感知机，这里的机表示的还是一种算法，本文第三部分、证明SVM中会详细阐述)。

**1.1.1、分类标准**

    这里我们考虑的是一个两类的分类问题，数据点用 x 来表示，这是一个 n 维向量，而类别用 y 来表示，可以取 1 或者 -1 ，分别代表两个不同的类。一个线性分类器就是要在 n 维的数据空间中找到一个[超平面](http://zh.wikipedia.org/wiki/%E8%B6%85%E5%B9%B3%E9%9D%A2)，其方程可以表示为：

wTx+b=0

    上面给出了线性分类的定义描述，但或许读者没有想过：为何用y取1 或者 -1来表示两个不同的类别呢？其实，这个1或-1的分类标准起源于logistic回归，为了完整和过渡的自然性，咱们就再来看看这个logistic回归。

**1.1.2、1或-1分类标准的起源：logistic回归**

    Logistic回归目的是从特征学习出一个0/1分类模型，而这个模型是将特性的线性组合作为自变量，由于自变量的取值范围是负无穷到正无穷。因此，使用logistic函数（或称作sigmoid函数）将自变量映射到(0,1)上，映射后的值被认为是属于y=1的概率。

    形式化表示就是

    假设函数

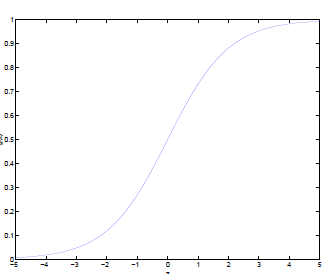
Image(37).png

其中x是n维特征向量，函数g就是logistic函数。

而

Image(38).png

的图像是



可以看到，将无穷映射到了(0,1)。

    而假设函数就是特征属于y=1的概率。

Image(40).png

当我们要判别一个新来的特征属于哪个类时，只需求，若大于0.5就是y=1的类，反之属于y=0类。

    再审视一下

Image(41).png

，发现只和

Image(42).png

有关，>0，那么

Image(43).png

，g(z)只不过是用来映射，真实的类别决定权还在

。还有当

时

Image(44).png

，

=1，反之

=0。如果我们只从

出发，希望模型达到的目标无非就是让训练数据中y=1的特征

，而是y=0的特征

Image(45).png

。Logistic回归就是要学习得到

Image(46).png

，使得正例的特征远大于0，负例的特征远小于0，强调在全部训练实例上达到这个目标。

**1.1.3、形式化标示**

     我们这次使用的结果标签是y=-1,y=1，替换在logistic回归中使用的y=0和y=1。同时将

替换成w和b。以前的

Image(47).png

，其中认为

Image(48).png

。现在我们替换

为b，后面替换

Image(49).png

为

Image(50).png

（即

Image(51).png

）。这样，我们让

Image(52).png

，进一步

Image(53).png

。也就是说除了y由y=0变为y=-1，只是标记不同外，与logistic回归的形式化表示没区别。

    再明确下假设函数

Image(54).png

上面提到过我们只需考虑

的正负问题，而不用关心g(z)，因此我们这里将g(z)做一个简化，将其简单映射到y=-1和y=1上。映射关系如下：

Image(55).png

    于此，想必已经解释明白了为何线性分类的标准一般用1 或者-1 来标示。

    注：上小节来自jerrylead所作的斯坦福机器学习课程的笔记。

***1.2*、线性分类的一个例子**

    下面举个简单的例子，一个二维平面(一个超平面，在二维空间中的例子就是一条直线)，如下图所示，平面上有两种不同的点，分别用两种不同的颜色表示，一种为红颜色的点，另一种则为蓝颜色的点，红颜色的线表示一个可行的超平面。



    从上图中我们可以看出，这条红颜色的线把红颜色的点和蓝颜色的点分开来了。而这条红颜色的线就是我们上面所说的超平面，也就是说，这个所谓的超平面的的确确便把这两种不同颜色的数据点分隔开来，在超平面一边的数据点所对应的 y 全是 -1 ，而在另一边全是 1 。

    接着，我们可以令分类函数（提醒：下文很大篇幅都在讨论着这个分类函数）：

 f(x)=wTx+b

    显然，如果 f(x)=0 ，那么 x 是位于超平面上的点。我们不妨要求对于所有满足 f(x)<0 的点，其对应的 y 等于 -1 ，而 f(x)>0 则对应 y=1 的数据点。

    （有一朋友飞狗来自Mare\_Desiderii，看了上面的定义之后，问道：请教一下SVM functional margin 为 γˆ=y(wTx+b)=yf(x)中的Y是只取1和-1 吗？y的唯一作用就是确保functional margin的非负性？真是这样的么？当然不是，详情请见本文评论下第43楼）

    当然，有些时候（或者说大部分时候）数据并不是线性可分的，这个时候满足这样条件的超平面就根本不存在(不过关于如何处理这样的问题我们后面会讲)，这里先从最简单的情形开始推导，就假设数据都是线性可分的，亦即这样的超平面是存在的。

    更进一步，我们在进行分类的时候，将数据点 x代入 f(x) 中，如果得到的结果小于 0 ，则赋予其类别 -1 ，如果大于 0 则赋予类别 1 。如果 f(x)=0，则很难办了，分到哪一类都不是。

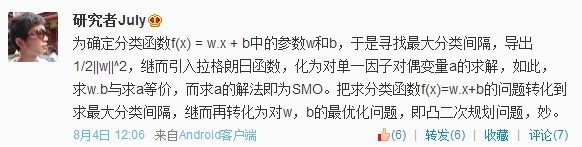
   请读者注意，下面的篇幅将按下述3点走：

咱们就要确定上述分类函数f(x) = w.x + b（w.x表示w与x的内积）中的两个参数w和b，通俗理解的话w是法向量，b是截距；

那如何确定w和b呢？答案是寻找两条边界端或极端划分直线中间的最大间隔（之所以要寻最大间隔是为了能更好的划分不同类的点，下文你将看到：为寻最大间隔，导出1/2||w||^2，继而引入拉格朗日函数和对偶变量a，化为对单一因数对偶变量a的求解，当然，这是后话）；

进而把寻求分类函数f(x) = w.x + b的问题转化为对w，b的最优化问题。

    总结成一句话即是：从最大间隔出发（目的本就是为了确定法向量w），转化为求对变量w和b的凸二次规划问题。亦或如下图所示（有点需要注意，如读者@酱爆小八爪所说：从最大分类间隔开始，就一直是凸优化问题）：



***1.3*、函数间隔Functional margin与几何间隔Geometrical margin**

    一般而言，一个点距离超平面的远近可以表示为分类预测的确信或准确程度。在超平面w\*x+b=0确定的情况下，|w\*x+b|能够相对的表示点x到距离超平面的远近，而w\*x+b的符号与类标记y的符号是否一致表示分类是否正确，所以，可以用量y\*(w\*x+b)的正负性来判定或表示分类的正确性和确信度，于此，我们便引出了函数间隔functional margin的概念。

**1.3.1、函数间隔Functional margin**

    我们定义函数间隔functional margin 为：

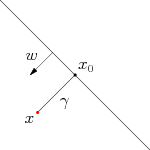
       γˆ=y(wTx+b)=yf(x)

    接着，我们定义超平面(w，b)关于训练数据集T的函数间隔为超平面(w，b)关于T中所有样本点(xi，yi)的函数间隔最小值，其中，x是特征，y是结果标签，i表示第i个样本，有：

γˆ=minγˆi    (i=1，...n)

    然与此同时，问题就出来了。上述定义的函数间隔虽然可以表示分类预测的正确性和确信度，但在选择分类超平面时，只有函数间隔还远远不够，因为如果成比例的改变w和b，如将他们改变为2w和2b，虽然此时超平面没有改变，但函数间隔的值f(x)却变成了原来的4倍。其实，我们可以对法向量w加些约束条件，使其表面上看起来规范化，如此，我们很快又将引出真正定义点到超平面的距离--**几何间隔geometrical margin**的概念。

**1.3.2、点到超平面的距离定义：几何间隔Geometrical margin**



    在给出几何间隔的定义之前，咱们首先来看下，如上图所示，对于一个点 x ，令其垂直投影到超平面上的对应的为 x0 ，由于 w 是垂直于超平面的一个向量，我们有

x=x0+γw∥w∥

                （||w||表示的是范数，关于范数的概念参见：<http://baike.baidu.com/view/637132.htm>）

    又由于 x0 是超平面上的点，满足 f(x0)=0 ，代入超平面的方程即可算出： γ

γ=wTx+b∥w∥=f(x)∥w∥

（有的书上会写成把||w|| 分开相除的形式，如本文参考文献及推荐阅读条目9，其中，||w||为w的二阶泛数）

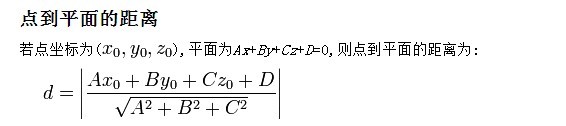
    不过，这里的 γ 是带符号的，我们需要的只是它的绝对值，因此类似地，也乘上对应的类别 y即可，因此实际上我们定义 **几何间隔geometrical margin** 为(注：别忘了，上面ˆγ的定义，ˆγ=y(wTx+b)=yf(x))：

γ˜=yγ=γˆ∥w∥

**（代人相关式子可以得出：yi\*(w/||w|| + b/||w||)）**

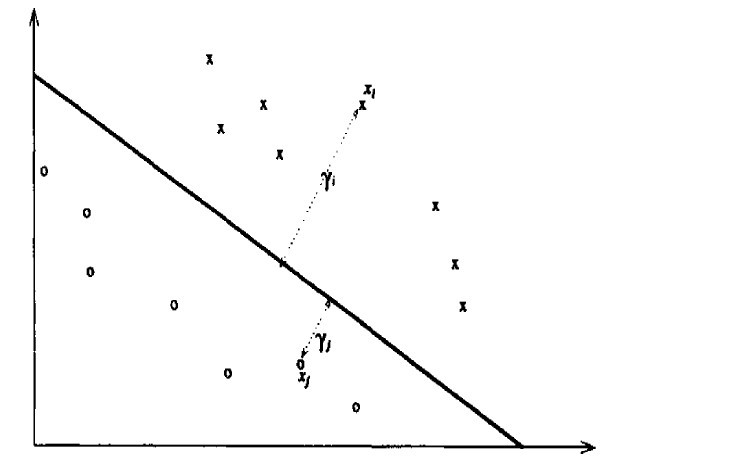
    正如本文评论下读者popol1991留言：函数间隔y\*(wx+b)=y\*f(x)实际上就是|f(x)|，只是人为定义的一个间隔度量；而几何间隔|f(x)|/||w||才是直观上的点到超平面距离。

    想想二维空间里的点到直线公式：假设一条直线的方程为ax+by+c=0,点P的坐标是(x0,y0)，则点到直线距离为|ax0+by0+c|/sqrt(a^2+b^2)。如下图所示：



    那么如果用向量表示，设w=(a,b),f(x)=wx+c,那么这个距离正是|f(p)|/||w||。

    OK，下图中xi，和xj分别到超平面的距离：



***1.4*、最大间隔分类器Maximum Margin Classifier的定义**

    于此，我们已经很明显的看出，函数间隔functional margin 和 几何间隔geometrical margin 相差一个 ∥w∥ 的缩放因子。按照我们前面的分析，对一个数据点进行分类，当它的 margin 越大的时候，分类的 confidence 越大。对于一个包含 n 个点的数据集，我们可以很自然地定义它的 margin 为所有这 n 个点的 margin 值中最小的那个。于是，为了使得分类的 confidence 高，我们希望所选择的超平面hyper plane 能够最大化这个 margin 值。

    通过上节，我们已经知道：

*1*、functional margin 明显是不太适合用来最大化的一个量，因为在 hyper plane 固定以后，我们可以等比例地缩放 w 的长度和 b 的值，这样可以使得 f(x)=wTx+b 的值任意大，亦即 functional margin γˆ 可以在 hyper plane 保持不变的情况下被取得任意大，

*2*、而 geometrical margin 则没有这个问题，因为除上了 ∥w∥ 这个分母，所以缩放 w 和 b 的时候 γ˜ 的值是不会改变的，它只随着 hyper plane 的变动而变动，因此，这是更加合适的一个 margin 。

    这样一来，我们的 maximum margin classifier 的目标函数可以定义为：

**maxγ***˜˜*˜˜

    当然，还需要满足一些条件，根据 margin 的定义，我们有

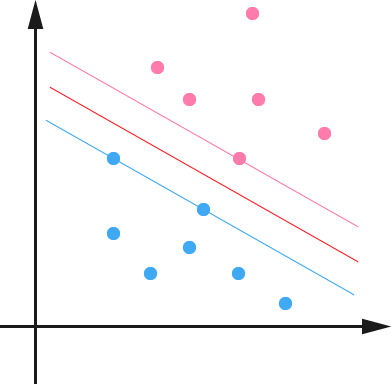
Image(12).jpeg

    其中 γˆ=γ˜∥w∥  (等价于 ˜γ = ˆγ / ∥w∥，故有稍后的 γˆ =1 时， ˜γ = 1 / ||w||)，处于方便推导和优化的目的，我们可以令 γˆ=1(对目标函数的优化没有影响，至于为什么，请见本文评论下第42楼回复，貌似博客系统故障，原来评论在第42楼的回复现在第42+81楼，即第123楼显示) ，此时，上述的目标函数 **˜γ**转化为(其中，s.t.，即subject to的意思，它导出的是约束条件)：

Image(13).jpeg

    通过求解这个问题，我们就可以找到一个 margin 最大的 classifier ，如下图所示，中间的红色线条是 Optimal Hyper Plane ，**另外两条线到红线的距离都是等于**

γ˜**的**( γ˜ 便是上文所定义的geometrical margin，当令 γˆ=1时， γ˜ 便为1/||w||，而我们上面得到的目标函数便是在相应的约束条件下，要最大化这个1/||w||值)：

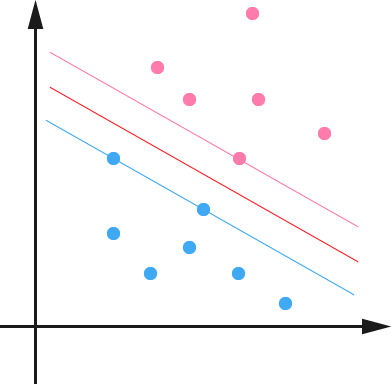


    通过最大化 margin ，我们使得该分类器对数据进行分类时具有了最大的 confidence 。但，这个最大分类间隔器到底是用来干嘛的呢？很简单，**SVM 通过使用最大分类间隙Maximum Margin Classifier 来设计决策最优分类超平面**，而为何是最大间隔，却不是最小间隔呢？因为最大间隔能获得最大稳定性与区分的确信度，从而得到良好的推广能力(超平面之间的距离越大，分离器的推广能力越好，也就是预测精度越高，不过对于训练数据的误差不一定是最小的.2012.08.21updated)。

    So，对于什么是Support Vector Machine ，我们可以先这样理解，如上图所示，我们可以看到 hyper plane 两边的那个 gap 分别对应的两条平行的线（在高维空间中也应该是两个 hyper plane）上有一些点，显然两个超平面hyper plane 上都会有点存在，否则我们就可以进一步扩大 gap ，也就是增大 γ˜ 的值了。这些点，就叫做 support vector。下文1.5节将更为具体描述。

***1.5*、到底什么是Support Vector**

    上节，我们介绍了Maximum Margin Classifier，但并没有具体阐述到底什么是Support Vector，本节，咱们来重点阐述这个概念。咱们不妨先来回忆一下上节1.4节最后一张图：



    可以看到两个支撑着中间的 gap 的超平面，它们到中间的纯红线separating hyper plane 的距离相等，即我们所能得到的最大的 geometrical margin **γ˜** 。而“支撑”这两个超平面的必定会有一些点，而这些“支撑”的点便叫做支持向量Support Vector。

    很显然，由于这些 supporting vector 刚好在边界上，所以它们是满足 y(wTx+b)=1 （还记得我们把 functional margin 定为 1 了吗？上节中：“处于方便推导和优化的目的，我们可以令 γˆ=1”），而对于所有不是支持向量的点，也就是在“阵地后方”的点，则显然有 y(wTx+b)>1 。当然，通常除了 K-Nearest Neighbor 之类的 Memory-based Learning 算法，通常算法也都不会直接把所有的点记忆下来，并全部用来做后续 inference 中的计算。不过，如果算法使用了**Kernel 方法进行非线性化推广**的话，就会遇到这个问题了。Kernel 方法在下文第二部分2.2节中介绍）。

    OK，到此为止，算是了解到了SVM的第一层，对于那些只关心怎么用SVM的朋友便已足够，不必再更进一层深究其更深的原理。

**第二层、深入SVM**

***2.1*、从线性可分到线性不可分**

**2.1.1、从原始问题到对偶问题的求解**

    当然，除了在上文中所介绍的从几何直观上之外，支持向量的概念也可以从其优化过程的推导中得到。虽然上文1.4节给出了目标函数，却没有讲怎么来求解。现在就让我们来处理这个问题。回忆一下之前得到的目标函数（subject to导出的则是约束条件）：

Image(14).jpeg

     由于求

Image(57).png

的最大值相当于求

Image(58).png

的最小值，所以上述问题等价于（w由分母变成分子，从而也有原来的max问题变为min问题，很明显，两者问题等价）：

Image(15).jpeg

到这个形式以后，就可以很明显地看出来，它是一个凸优化问题，或者更具体地说，它是一个二次优化问题——目标函数是二次的，约束条件是线性的。这个问题可以用任何现成的 [QP (Quadratic Programming)](http://en.wikipedia.org/wiki/Quadratic_programming) 的优化包进行求解；

虽然这个问题确实是一个标准的 QP 问题，但是它也有它的特殊结构，通过 [Lagrange Duality](http://en.wikipedia.org/wiki/Lagrange_duality#The_strong_Lagrangian_principle:_Lagrange_duality) 变换到对偶变量 (dual variable) 的优化问题之后，可以找到一种更加有效的方法来进行求解，而且通常情况下这种方法比直接使用通用的 QP 优化包进行优化要高效得多。

    也就说，除了用解决QP问题的常规方法之外，还可以应用拉格朗日对偶性，通过求解对偶问题得到最优解，这就是线性可分条件下支持向量机的对偶算法，这样做的优点在于：一者对偶问题往往更容易求解；二者可以自然的引入核函数，进而推广到非线性分类问题。

    ok，接下来，你将看到“**对偶变量dual variable的优化问题**”等类似的关键词频繁出现，便是解决此凸优化问题的第二种更为高效的解--对偶变量的优化求解。

    至于上述提到，关于什么是Lagrange duality？简单地来说，通过给每一个约束条件加上一个 Lagrange multiplier(拉格朗日乘值)，即引入拉格朗日对偶变量***α***，如此我们便可以通过拉格朗日函数将约束条件融和到目标函数里去(也就是说把条件融合到一个函数里头，现在只用一个函数表达式便能清楚的表达出我们的问题)：



   然后我们令

Image(17).jpeg

    容易验证，当某个约束条件不满足时，例如 yi(wTxi+b)<1，那么我们显然有 θ(w)=∞（只要令 αi=∞ 即可）。而当所有约束条件都满足时，则有 θ(w)=12∥w∥2 ，    亦即我们最初要最小化的量。因此，在要求约束条件得到满足的情况下最小化 12∥w∥2    实际上等价于直接最小化 θ(w)     （当然，这里也有约束条件，就是 αi≥0,i=1,…,n）   ，因为如果约束条件没有得到满足，θ(w)     会等于无穷大，自然不会是我们所要求的最小值。具体写出来，我们现在的目标函数变成了：

Image(18).jpeg

    这里用 p∗ 表示这个问题的最优值，这个问题和我们最初的问题是等价的。不过，现在我们来把最小和最大的位置交换一下（稍后，你将看到，当下面式子满足了一定的条件之后，这个式子*d*便是上式*P* 的对偶形式表示）：

Image(19).jpeg

    当然，交换以后的问题不再等价于原问题，这个新问题的最优值用 d∗ 来表示。并，我们有 d∗≤p∗ ，这在直观上也不难理解，最大值中最小的一个总也比最小值中最大的一个要大吧！  总之，**第二个问题**的最优值 d∗ 在这里提供了一个第一个问题的最优值 p∗ 的一个下界，在满足某些条件的情况下，这两者相等，这个时候我们就可以通过求解第二个问题来间接地求解第一个问题。

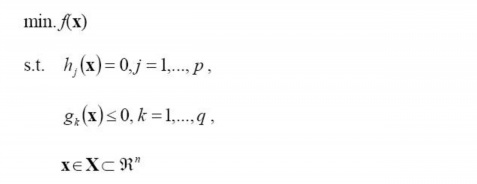
    注：上段说“在满足某些条件的情况下”，这所谓的“满足某些条件”就是要先满足Slater's Condition，进而就满足KKT条件。理由如下3点所述（观点来自frestyle）：

在convex problem中，d\*和p\*相同的条件是Slater's Condition，Slater's condition保证临界点saddle point存在。

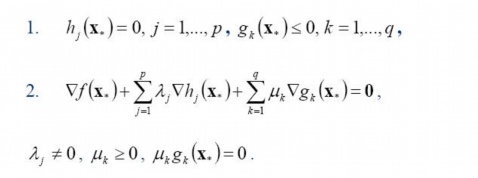
至于KKT条件，首先原问题的最优值可以通过求Lagrangian的临界点saddle point（如果有的话）来得到，再者，KKT theorem里面进一步引入了更强的前提，也就是在满足Slater condition的同时（前面说了，Slater's condition保证临界点saddle point存在），f和gi都是可微的，这样saddle point不仅存在，而且能通过对Lagrangian求导得到，

所以KKT条件是一个点是最优解的条件，而不是d\*=p\*的条件，当然这个KKT条件对后边简化dual problem很关键。

    那KKT条件的表现形式是什么呢？据维基百科：[KKT 条件](http://en.wikipedia.org/wiki/Karush%E2%80%93Kuhn%E2%80%93Tucker_conditions)的介绍，一般地，一个最优化数学模型能够表示成下列标准形式：



    所谓 Karush-Kuhn-Tucker 最优化条件，就是指上式的最小点 x\* 必须满足下面的条件：

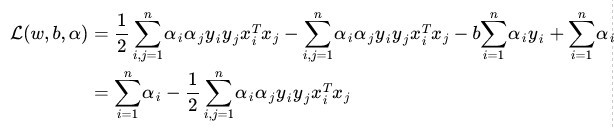


    经过论证，我们这里的问题是满足 KKT 条件的（首先已经满足Slater condition，再者f和gi也都是可微的，即L对w和b都可导），因此现在我们便转化为求解第二个问题。也就是说，现在，咱们的原问题通过满足一定的条件，已经转化成了对偶问题。而求解这个对偶学习问题，分为3个步骤，首先要让L(w，b，a) 关于 w 和 b 最小化，然后求对α的极大，最后利用SMO算法求解对偶因子。

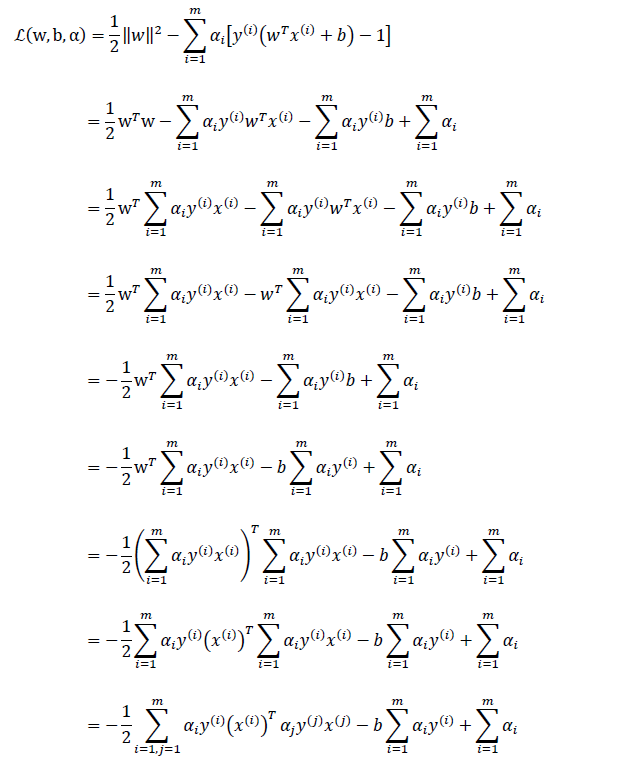
    **（1）**、首先固定*α，*要让 L 关于 w 和 b 最小化，我们分别对w，b求偏导数，即令 ∂L/∂w 和 ∂L/∂b 等于零（对w求导结果的解释请看本文评论下第45楼回复~~，貌似博客系统故障，原来评论在第45楼的回复现在第45+81楼，即第126楼显示~~）：

∂L∂w=0∂L∂b=0⇒w=∑i=1nαiyixi⇒∑i=1nαiyi=0

    带回上述的 L： ，得到：



    提醒：有读者可能会问上述推导过程如何而来？说实话，其具体推导过程是比较复杂的，如下图所示：



      最后，得到：

    如 jerrylead所说：“倒数第4步”推导到“倒数第3步”使用了线性代数的转置运算，由于ai和yi都是实数，因此转置后与自身一样。“倒数第3步”推导到“倒数第2步”使用了(a+b+c+…)(a+b+c+…)=aa+ab+ac+ba+bb+bc+…的乘法运算法则。最后一步是上一步的顺序调整。

L(

    从上面的最后一个式子，我们可以看出，此时的拉格朗日函数只包含了一个变量，那就是ai，然后下文的第2步，求出了ai便能求出w，和b，由此可见，上文第1.2节提出来的核心问题：分类函数**w^T + b**也就可以轻而易举的求出来了。

    由此看出，使用拉格朗日定理解凸最优化问题可以使用一个对偶变量表示，转换为对偶问题后，通常比原问题更容易处理，因为直接处理不等式约束是困难的，而对偶问题通过引入拉格朗日乘子(又称为对偶变量)来解。

    **（2）**、求对α的极大，即是关于对偶变量**dual variable**α（下文将一直**用粗体+下划线表示）**的优化问题，从上面的式子得到：

    (不得不提醒下读者：经过上面第一个步骤的求w和b，得到的拉格朗日函数式子已经没有了变量w，b，只有a，而反过来，求得的a将能导出w，b的解，最终得出分离超平面和分类决策函数。为何呢？因为如果求出了ai

，根据

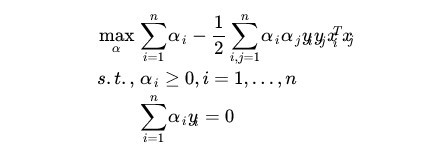
Image(21).jpeg

，

即可求出w。然后通过

Image(60).png

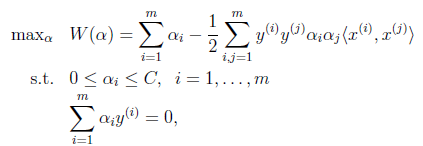
，即可求出b )



    如前面所说，这个问题有更加高效的优化算法，即我们常说的SMO算法。

     **（3）**、序列最小最优化SMO算法。细心的读者读至上节末尾处，怎么求对偶变量α的值可能依然心存疑惑。实际上，关于a的求解过程即是我们常说的SMO算法，这里简要简单介绍下。

      OK，当：



    要解决的是在参数

Image(62).png

上求最大值W的问题，至于

Image(63).png

和

Image(64).png

都是已知数（其中

Image(65).png

 是一个参数，用于控制目标函数中两项（“寻找 margin 最大的超平面”和“保证数据点偏差量最小”）之间的权重。和上文最后的式子对比一下，可以看到唯一的区别就是现在 dual variable

Image(66).png

 多了一个上限  ，具体由来请查看下文第2.3节）。

    按照坐标上升的思路，我们首先固定除

Image(67).png

以外的所有参数，然后在

上求极值。等一下，这个思路有问题，因为如果固定

以外的所有参数，那么

将不再是变量（可以由其他值推出），由上面式子中的最后一个条件可以得出：

Image(68).png

    因此，我们需要一次选取两个参数做优化，比如

和

Image(69).png

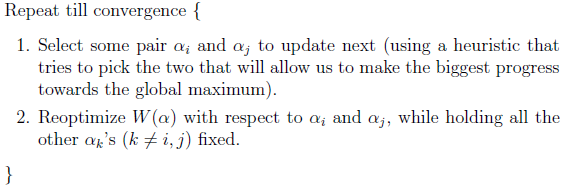
，此时

可以由

和其他参数表示出来。这样回带到W中，W就只是关于

的函数了，可解。

    这样，SMO的主要步骤如下：



    意思是，

第一步选取一对

Image(71).png

和

Image(72).png

，选取方法使用启发式方法；

第二步，固定除

和

之外的其他参数，确定W极值条件下的

，

由

表示。

    SMO算法每次迭代只选出两个分量ai和aj进行调整，其它分量则保持固定不变，在得到解ai和aj之后，再用ai和aj改进其它分量。与通常的分解算法比较，尽管它可能需要更多的迭代次数，但每次迭代的计算量比较小，所以该算法表现出整理的快速收敛性，且不需要存储核矩阵，也没有矩阵运算。

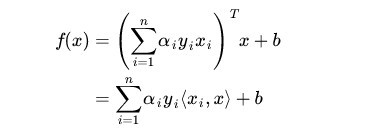
    更多请参看文末参考文献和推荐阅读中的条目6《支持向量机--算法、理论和扩展》和条目11《统计学习方法》的相关章节，或跳至下文3.4节。

**2.1.2、线性不可分的情况**

    OK，为过渡到下节2.2节所介绍的核函数，让我们再来看看上述推导过程中得到的一些有趣的形式。首先就是关于我们的 hyper plane ，对于一个数据点 x 进行分类，实际上是通过把 x 带入到 f(x)=wTx+b      算出结果然后根据其正负号来进行类别划分的。而前面的推导中我们得到

      w=∑ni=1αiyixi ，

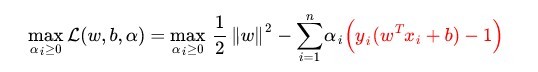
    因此**分类函数**为：



    这里的形式的有趣之处在于，对于新点 *x*的预测，只需要计算它与训练数据点的内积即可（⋅,⋅表示向量内积），这一点至关重要，是之后使用 Kernel 进行非线性推广的基本前提。此外，所谓 Supporting Vector 也在这里显示出来——事实上，所有非 Supporting Vector 所对应的系数 *α* 都是等于零的，因此对于新点的内积计算实际上只要针对少量的“支持向量”而不是所有的训练数据即可。

    为什么非支持向量对应的 α 等于零呢？直观上来理解的话，就是这些“后方”的点——正如我们之前分析过的一样，对超平面是没有影响的，由于分类完全有超平面决定，所以这些无关的点并不会参与分类问题的计算，因而也就不会产生任何影响了。

    回忆一下我们2.1.1节中通过 Lagrange multiplier得到的目标函数：



     注意到如果 xi 是支持向量的话，上式中红颜色的部分是等于 0 的（因为支持向量的 functional margin 等于 1 ），而对于非支持向量来说，functional margin 会大于 1 ，因此红颜色部分是大于零的，而 αi 又是非负的，为了满足最大化，αi 必须等于 0 。这也就是这些非 Supporting Vector 的点的局限性。

    从1.5节到上述所有这些东西，便得到了一个maximum margin hyper plane classifier，这就是所谓的支持向量机（Support Vector Machine）。当然，**到目前为止，我们的 SVM 还比较弱，只能处理线性的情况**，不过，在得到了对偶dual 形式之后，通过**Kernel 推广到非线性**的情况就变成了一件非常容易的事情了(相信，你还记得本节开头所说的：通过求解对偶问题得到最优解，这就是线性可分条件下支持向量机的对偶算法，这样做的优点在于：一者对偶问题往往更容易求解；二者可以自然的引入核函数，进而推广到非线性分类问题)。

***2.2*、核函数Kernel**

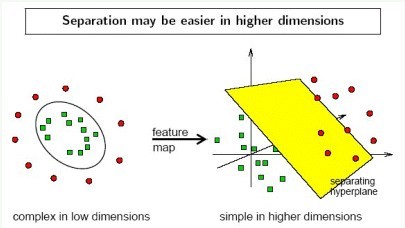
    咱们首先给出核函数的来头：

在上文中，我们已经了解到了SVM处理线性可分的情况，而对于非线性的情况，SVM 的处理方法是选择一个核函数 κ(⋅,⋅) ，通过将数据映射到高维空间，来解决在原始空间中线性不可分的问题。由于核函数的优良品质，这样的非线性扩展在计算量上并没有比原来复杂多少，这一点是非常难得的。当然，这要归功于核方法——除了 SVM 之外，任何将计算表示为数据点的内积的方法，都可以使用核方法进行非线性扩展。

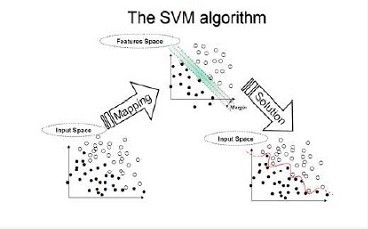
    也就是说，Minsky和Papert早就在20世纪60年代就已经明确指出线性学习器计算能力有限。为什么呢？因为总体上来讲，现实世界复杂的应用需要有比线性函数更富有表达能力的假设空间，也就是说，目标概念通常不能由给定属性的简单线性函数组合产生，而是应该一般地寻找待研究数据的更为一般化的抽象特征。

    而下文我们将具体介绍的核函数则提供了此种问题的解决途径，从下文你将看到，核函数通过把数据映射到高维空间来增加第一节所述的线性学习器的能力，使得线性学习器对偶空间的表达方式让分类操作更具灵活性和可操作性。我们知道，**训练样例一般是不会独立出现的，它们总是以成对样例的内积形式出现**，而用**对偶形式**表示学习器的优势在为在该表示中可调参数的个数不依赖输入属性的个数，**通过使用恰当的核函数来替代内积，可以隐式得将非线性的训练数据映射到高维空间**，而不增加可调参数的个数(当然，前提是核函数能够计算对应着两个输入特征向量的内积)。

    ***1、***简而言之：在线性不可分的情况下，支持向量机通过某种事先选择的非线性映射(核函数)将输入变量映射到一个高维特征空间，在这个空间中构造最优分类超平面。我们使用SVM进行数据集分类工作的过程首先是同预先选定的一些非线性映射将输入空间映射到高维特征空间(下图很清晰的表达了通过映射到高维特征空间，而把平面上本身不好分的非线性数据分了开来)：



    使得在高维属性空间中有可能最训练数据实现超平面的分割，避免了在原输入空间中进行非线性曲面分割计算。SVM数据集形成的分类函数具有这样的性质：它是一组以支持向量为参数的非线性函数的线性组合，因此分类函数的表达式仅和支持向量的数量有关，而独立于空间的维度，在处理高维输入空间的分类时，这种方法尤其有效，其工作原理如下图所示：



    ***2、***具体点说：在我们遇到核函数之前，如果用原始的方法，那么在用线性学习器学习一个非线性关系，需要选择一个非线性特征集，并且将数据写成新的表达形式，这等价于应用一个固定的非线性映射，将数据映射到特征空间，在特征空间中使用线性学习器，因此，考虑的假设集是这种类型的函数：

740718_7761.jpeg

    这里*ϕ*：X->F是从输入空间到某个特征空间的映射，这意味着建立非线性学习器分为两步：

首先使用一个非线性映射将数据变换到一个特征空间F，

然后在特征空间使用线性学习器分类。

    在上文我提到过对偶形式，而这个对偶形式就是线性学习器的一个重要性质，这意味着假设可以表达为训练点的线性组合，因此决策规则可以用测试点和训练点的内积来表示：

741121_9867.jpeg

    如果有一种方式可以**在特征空间中直接计算内积〈φ(xi · φ(x)〉**，就像在原始输入点的函数中一样，就有可能将两个步骤融合到一起建立一个非线性的学习器，**这样直接计算法的方法称为核函数方法，**于是，核函数便横空出世了。

    这里我直接给出一个定义：核是一个函数K，对所有x，z(-X，满足

741445_1451.jpeg

，这里φ是从X到内积特征空间F的映射。

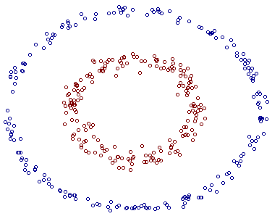
    ***3、***总而言之，举个简单直接点的例子，则是如果不是用核技术，就会先计算线性映射phy(x1)和phy(x2),然后计算这两个特征的内积，使用了核技术之后，先把phy(x1)和phy(x2)的通用表达式子：< phy(x1)，phy(x2) >=k( <x1,x2> )计算出来，注意到这里的< ， >表示内积，k( , )就是对应的核函数，这个表达往往非常简单，所以计算非常方便。

    ....

    OK，接下来，咱们就进一步从外到里，来探探这个核函数的真面目。

**2.2.1、如何处理非线性数据**

    在2.1节中我们介绍了线性情况下的支持向量机，它通过寻找一个线性的超平面来达到对数据进行分类的目的。不过，由于是线性方法，所以对非线性的数据就没有办法处理了。举个例子来说，则是如下图所示的两类数据，分别分布为两个圆圈的形状，这样的数据本身就是线性不可分的，你准备如何把这两类数据分开呢(下文将会有一个相应的三维空间图)？



    上图所述的这个数据集，就是用两个半径不同的圆圈加上了少量的噪音生成得到的，所以，一个理想的分界应该是一个“圆圈”而不是一条线（超平面）。如果用 X1 和 X2 来表示这个二维平面的两个坐标的话，我们知道一条二次曲线（圆圈是二次曲线的一种特殊情况）的方程可以写作这样的形式：

a1X1+a2X21+a3X2+a4X22+a5X1X2+a6=0

    注意上面的形式，如果我们构造另外一个五维的空间，其中五个坐标的值分别为 Z1=X1, Z2=X21, Z3=X2, Z4=X22, Z5=X1X2，那么显然，上面的方程在新的坐标系下可以写作：

∑i=15aiZi+a6=0

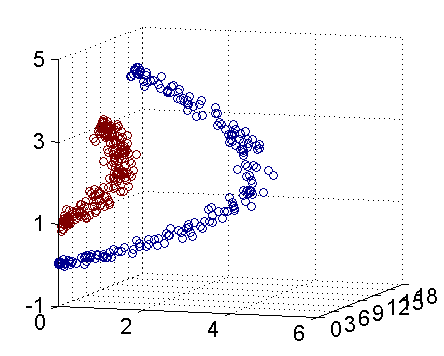
    关于新的坐标 Z ，这正是一个 hyper plane 的方程！也就是说，如果我们做一个映射 ϕ:R2→R5 ，将 X 按照上面的规则映射为 Z ，那么在新的空间中原来的数据将变成线性可分的，从而使用之前我们推导的线性分类算法就可以进行处理了。这正是 Kernel 方法处理非线性问题的基本思想。

**2.2.2、特征空间的隐式映射：核函数**

    再进一步描述 Kernel 的细节之前，不妨再来看看这个例子映射过后的直观例子。当然，你我可能无法把 5 维空间画出来，不过由于我这里生成数据的时候就是用了特殊的情形，具体来说，我这里的超平面实际的方程是这个样子（圆心在 X2 轴上的一个正圆）：

a1X21+a2(X2−c)2+a3=0

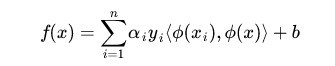
    因此我只需要把它映射到 Z1=X21, Z2=X22, Z3=X2 这样一个三维空间中即可，下图即是映射之后的结果，将坐标轴经过适当的旋转，就可以很明显地看出，数据是可以通过一个平面来分开的(pluskid：下面的gif 动画，先用 Matlab 画出一张张图片，再用 Imagemagick 拼贴成)：



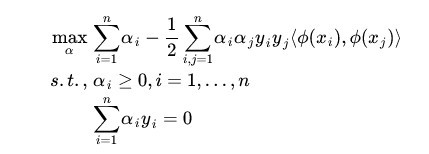
    现在让我们再回到 SVM 的情形，假设原始的数据时非线性的，我们通过一个映射 ϕ(⋅) 将其映射到一个高维空间中，数据变得线性可分了，这个时候，我们就可以使用原来的推导来进行计算，只是所有的推导现在是在新的空间，而不是原始空间中进行。当然，推导过程也并不是可以简单地直接类比的，例如，原本我们要求超平面的法向量 w ，但是如果映射之后得到的新空间的维度是无穷维的（确实会出现这样的情况，比如后面会提到的 高斯核Gaussian Kernel ），要表示一个无穷维的向量描述起来就比较麻烦。于是我们不妨先忽略过这些细节，直接从最终的结论来分析，回忆一下，我们**上一次2.1节**中得到的最终的分类函数是这样的：



    现在则是在映射过后的空间，即：



    而其中的 **α 也是通过求解如下 dual 问题**而得到的：



    这样一来问题就解决了吗？似乎是的：拿到非线性数据，就找一个映射

Image(73).png

Image(74).png

Image(75).png

Image(76).png

 ，然后一股脑把原来的数据映射到新空间中，再做线性 SVM 即可。不过事实上没有这么简单！其实刚才的方法稍想一下就会发现有问题：在最初的例子里，我们对一个二维空间做映射，选择的新空间是原始空间的所有一阶和二阶的组合，得到了五个维度；如果原始空间是三维，那么我们会得到 19 维的新空间，这个数目是呈爆炸性增长的，这给  的计算带来了非常大的困难，而且如果遇到无穷维的情况，就根本无从计算了。所以就需要 Kernel 出马了。

    不妨还是从最开始的简单例子出发，设两个向量

Image(27).jpeg

和

Image(28).jpeg

，而

Image(29).jpeg

即是到前面**2.2.1节**说的五维空间的映射，因此映射过后的内积为：

Image(30).jpeg

        （公式说明：上面的这两个推导过程中，所说的前面的五维空间的映射，这里说的前面便是文中2.2.1节的所述的映射方式，仔细看下2.2.1节的映射规则，再看那第一个推导，其实就是计算x1，x2各自的内积，然后相乘相加即可，第二个推导则是直接平方，去掉括号，也很容易推出来）

    另外，我们又注意到：

Image(31).jpeg

     二者有很多相似的地方，实际上，我们只要把某几个维度线性缩放一下，然后再加上一个常数维度，具体来说，上面这个式子的计算结果实际上和映射

Image(32).jpeg

     之后的内积

Image(33).jpeg

的结果是相等的，那么区别在于什么地方呢？

一个是映射到高维空间中，然后再根据内积的公式进行计算；

而另一个则**直接在原来的低维空间中进行计算，而不需要显式地写出映射后的结果**。

    （公式说明：上面之中，最后的两个式子，第一个算式，是带内积的完全平方式，可以拆开，然后，通过凑一个得到，第二个算式，也是根据第一个算式凑出来的）

    回忆刚才提到的映射的维度爆炸，在前一种方法已经无法计算的情况下，后一种方法却依旧能从容处理，甚至是无穷维度的情况也没有问题。

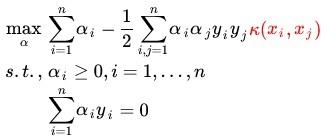
    我们把这里**的计算两个向量在隐式映射过后的空间中的内积的函数叫做核函数** (Kernel Function) ，例如，在刚才的例子中，我们的核函数为：

Image(34).jpeg

    核函数能**简化映射空间中的内积运算**——刚好“碰巧”的是，在我们的**SVM 里需要计算的地方数据向量总是以内积的形式出现**的。对比刚才我们上面写出来的式子，现在我们的**分类函数**为：

Image(35).jpeg

    其中  由如下 dual 问题计算而得：



    这样一来计算的问题就算解决了，避开了直接在高维空间中进行计算，而结果却是等价的！当然，因为我们这里的例子非常简单，所以我可以手工构造出对应于的核函数出来，如果对于任意一个映射，想要构造出对应的核函数就很困难了。

**2.2.3、几个核函数**

    最理想的情况下，我们希望知道数据的具体形状和分布，从而得到一个刚好可以将数据映射成线性可分的，然后通过这个得出对应的

Image(37).jpeg

进行内积计算。然而，第二步通常是非常困难甚至完全没法做的。不过，由于第一步也是几乎无法做到，因为对于任意的数据分析其形状找到合适的映射本身就不是什么容易的事情，所以，人们通常都是“胡乱”选择映射的，所以，根本没有必要精确地找出对应于映射的那个核函数，而只需要“胡乱”选择一个核函数即可——我们知道它对应了某个映射，虽然我们不知道这个映射具体是什么。由于我们的计算只需要核函数即可，所以我们也并不关心也没有必要求出所对应的映射的具体形式。

    当然，也并不是任意的二元函数都可以作为核函数，所以除非某些特殊的应用中可能会构造一些特殊的核（例如用于文本分析的文本核，注意其实使用了 Kernel 进行计算之后，其实完全可以去掉原始空间是一个向量空间的假设了，只要核函数支持，原始数据可以是任意的“对象”——比如文本字符串），通常人们会从一些常用的核函数中选择（根据问题和数据的不同，选择不同的参数，实际上就是得到了不同的核函数），例如：

多项式核，显然刚才我们举的例子是这里多项式核的一个特例（R = 1，d = 2）。虽然比较麻烦，而且没有必要，不过这个核所对应的映射实际上是可以写出来的，该空间的维度是

Image(38).jpeg

，其中

Image(77).png

 是原始空间的维度。

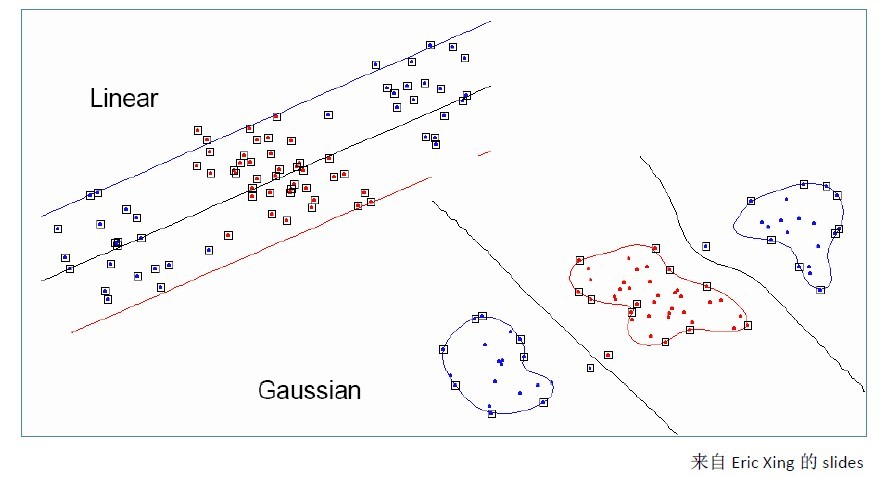
高斯核

Image(39).jpeg

，这个核就是最开始提到过的会将原始空间映射为无穷维空间的那个家伙。不过，如果

Image(40).jpeg

选得很大的话，高次特征上的权重实际上衰减得非常快，所以实际上（数值上近似一下）相当于一个低维的子空间；反过来，如果选得很小，则可以将任意的数据映射为线性可分——当然，这并不一定是好事，因为随之而来的可能是非常严重的过拟合问题。不过，总的来说，通过调控参数，高斯核实际上具有相当高的灵活性，也是使用最广泛的核函数之一。下图所示的例子便是把低维线性不可分的数据通过高斯核函数映射到了高维空间：



线性核

Image(41).jpeg

，这实际上就是原始空间中的内积。这个核存在的主要目的是使得“映射后空间中的问题”和“映射前空间中的问题”两者在形式上统一起来了(意思是说，咱们有的时候，写代码，或写公式的时候，只要写个模板或通用表达式，然后再代入不同的核，便可以了，于此，便在形式上统一了起来，不用再分别写一个线性的，和一个非线性的)。

**2.2.4、核函数的本质**

        上面说了这么一大堆，读者可能还是没明白核函数到底是个什么东西？我再简要概括下，即以下三点：

实际中，我们会经常遇到线性不可分的样例，此时，我们的常用做法是把样例特征映射到高维空间中去(如上文2.2节最开始的那幅图所示，映射到高维空间后，相关特征便被分开了，也就达到了分类的目的)；

但进一步，如果凡是遇到线性不可分的样例，一律映射到高维空间，那么这个维度大小是会高到可怕的(如上文中19维乃至无穷维的例子)。那咋办呢？

此时，核函数就隆重登场了，核函数的价值在于它虽然也是讲特征进行从低维到高维的转换，但核函数绝就绝在它事先在低维上进行计算，而将实质上的分类效果表现在了高维上，也就如上文所说的避免了直接在高维空间中的复杂计算。

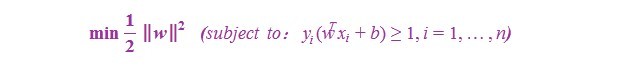
    此外，读者来信，向我提出了一个问题：

    In a SVM without using slack variables, if we remove one of the support vectors from the training set, what will happen to the maximal margin? List all the possibilities and give a sample for each possible situation, i.e., generate a training set, indicate which point is to be removed and clarify the change of the maximal margin.

    大意是：在没有松弛变量的svm中，如果我们移去训练集中的一个支持向量，那最大margin会怎么变化呢？列举出所有的可能，每种情况给出一个例子。也就是，举出一个训练集，指出移去哪个点，并指明最大margin怎么变。

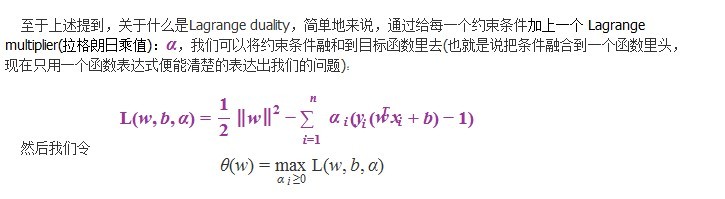
    解答：你可能会说最终的maximal margin会变大，会变小，或不变？但一切依据呢？与其胡乱猜测，不如实际推导.计算.证明！接下来，咱们回顾下上文「所有图片截取自上文」：

 maximum margin就是



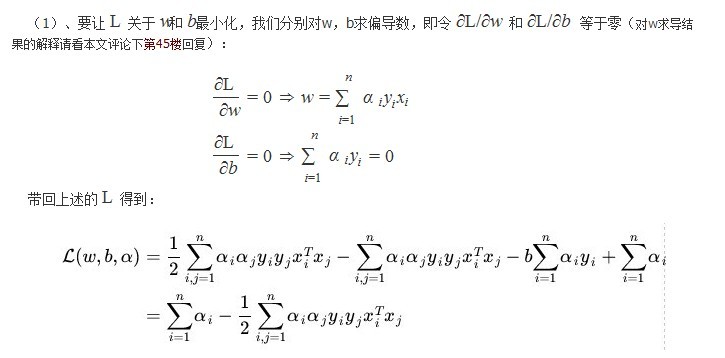
 remove后(w,b)变了，w一变，maximum margin自然也就会变了，至于如何变，请读者继续计算看具体结果。此外，还可以看看这里的分析：<http://www.cs.berkeley.edu/~russell/classes/cs194/f11/assignments/a2/a2-solution.pdf>。

 与此同时，读者自会发现到：上文中很大一部分篇幅都是在阐述怎么求及优化这个最大间隔分类超平面，包括后面的

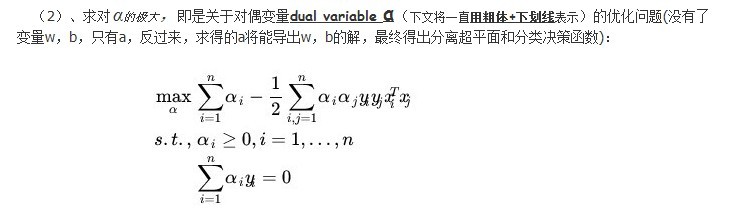


Image(44).jpeg

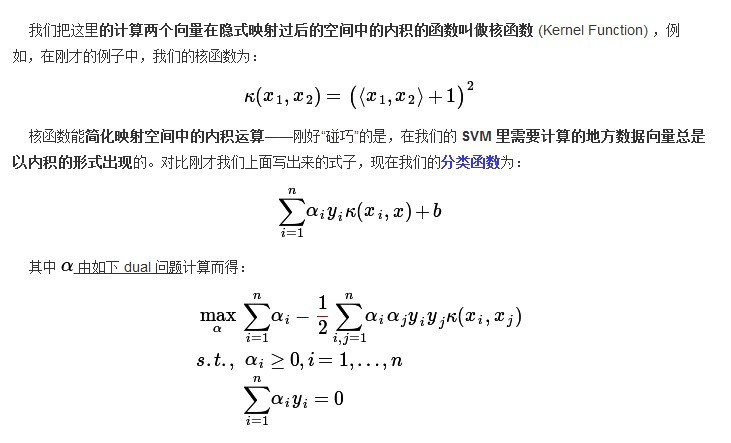
    下面，就是这两个步骤，第一步：



    第二步、对α求极大如下所示：



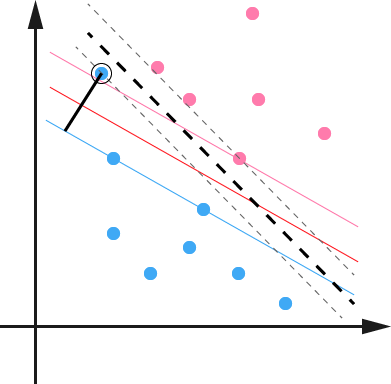
    再到后来，有了核函数，解决高维空间计算的复杂性问题，再到对对偶因子***α***的求解：SMO算法，通篇都不过是为了寻找一个最大间隔分类超平面，也即寻找一个分类函数，然后对这个分类函数各个因子的求解，如w，b，对偶因子***α****..*



    自此，你看到，上文中各个知识点是可以联系起来的，每一个步骤也都是一步一步推导下来的。

***2.3*、使用松弛变量处理 outliers 方法**

    在本文第一节最开始讨论支持向量机的时候，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，在上文2.2节使用 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 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 support vector 组成的，如果这些 support vector 里又存在 outlier 的话，其影响就很大了。例如下图：



    用黑圈圈起来的那个蓝点是一个 outlier ，它偏离了自己原本所应该在的那个半空间，如果直接忽略掉它的话，原来的分隔超平面还是挺好的，但是由于这个 outlier 的出现，导致分隔超平面不得不被挤歪了，变成途中黑色虚线所示（这只是一个示意图，并没有严格计算精确坐标），同时 margin 也相应变小了。当然，更严重的情况是，如果这个 outlier 再往右上移动一些距离的话，我们将无法构造出能将数据分开的超平面来。

    为了处理这种情况，SVM 允许数据点在一定程度上偏离一下超平面。例如上图中，黑色实线所对应的距离，就是该 outlier 偏离的距离，如果把它移动回来，就刚好落在原来的超平面上，而不会使得超平面发生变形了。具体来说，原来的约束条件

Image(48).jpeg

    现在变成

Image(49).jpeg

    其中

Image(50).jpeg

称为松弛变量 (slack variable) ，对应数据点

Image(51).jpeg

允许偏离的 functional margin 的量。当然，如果我们运行

Image(52).jpeg

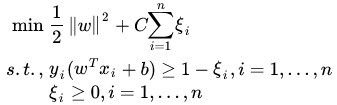
任意大的话，那任意的超平面都是符合条件的了。所以，我们在原来的目标函数后面加上一项，使得这些的总和也要最小：

Image(53).jpeg

    其中  是一个参数，用于控制目标函数中两项（“寻找 margin 最大的超平面”和“保证数据点偏差量最小”）之间的权重。注意，其中

Image(79).png

 是需要优化的变量（之一），而  是一个事先确定好的常量。完整地写出来是这个样子：



    用之前的方法将限制或约束条件加入到目标函数中，得到新的拉格朗日函数，如下所示：

Image(55).jpeg

     分析方法和前面一样，转换为另一个问题之后，我们先让

Image(56).jpeg

针对

Image(57).jpeg

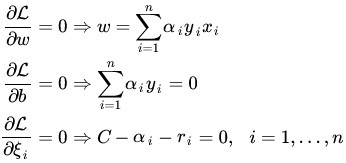
、

Image(58).jpeg

和

Image(59).jpeg

最小化：



     将

Image(80).png

 带回

Image(81).png

 并化简，得到和原来一样的目标函数：

Image(61).jpeg

     不过，由于我们得到

Image(62).jpeg

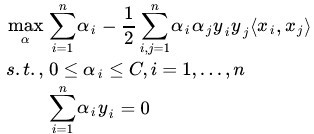
而又有

Image(63).jpeg

（作为 Lagrange multiplier 的条件），因此有

Image(64).jpeg

，所以整个 dual 问题现在写作：



    和之前的结果对比一下，可以看到唯一的区别就是现在 dual variable  多了一个上限  。而 Kernel 化的非线性形式也是一样的，只要把

Image(66).jpeg

换成

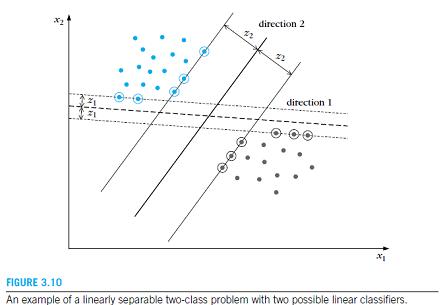
Image(67).jpeg

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 outliers 的支持向量机才终于介绍完毕了。继续深入请阅读参考文献及推荐阅读的条目6。

***2.4*、小结**

    综上所述，对于一个线性可分的两类问题，我们可以通过g(x)=w^T\*x+w0=0确定一个分类面把这两类分开。我们的目的是为了建立一个这样的分类面，事实上这样的分类面不是唯一确定的。在感知器里我们通过梯度下降法优化出一个分类器初始值的选择、迭代步长等的不同得到的分类器也不同。那么我们需要在这些分离器里找一个最好的。

    如图所示，我们可以认为direction2的分类效果要比 direction1的分类效果要好，因为direction2的裕量比direction1大。我们需要在这各个分类器中选择一个最优的。SVM是根据统计学习理论依照结构风险最小化的原则提出的，要求实现两个目的：1）两类问题能够分开（经验风险最小）2）margin最大化（风险上界最小）既是在保证风险最小的子集中选择经验风险最小的函数。



    把样本到分类面的距离

320176_8787.jpeg

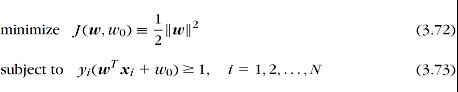
    进行归一化处理后我们得到里分类面最近的样本g(x) = 1。

    这样我们就有边界margin：

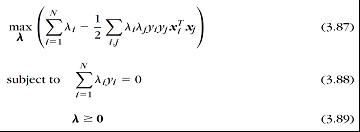
320197_3008.jpeg

，这里满足这样条件的样本点就是我们所谓的支持向量。

    这样我们就转化为一个优化问题Sergios Theodoridis：



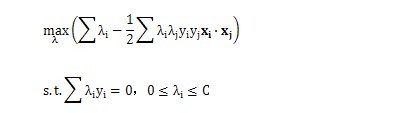
    建立拉格朗日方程并引入KKT条件得到：



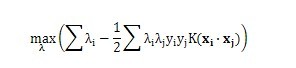
    由上式得到判别函数式：

320342_8747.jpeg

    对于不是线性可分的问题，我们可以通过加入松弛子C来解决：



    由以上讨论我们得到判别函数只与向量的內积有关，因此我们可以选择一个非线性变换将x映射到高维空间，在低维空间不可分的问题映射到高维空间后就有可能是线性可分的。这里我们不需要知道是什么形式的只需要关注內积运算即可。由此，可以通过构造核函数实现：



    这里的核函数的选择没有特别的方式，在Chih-Wei Hsu中推荐使用径向基函数。

    故不准确的说，SVM它本质上即是一个分类方法，用w^T+b定义分类函数，于是求w、b，为寻最大间隔，引出1/2||w||^2，继而引入拉格朗日因子，化为对单一因数对偶变量a的求解（求解过程中会涉及到一系列最优化或凸二次规划等问题），如此，求w.b与求a等价，而求a的解法即为SMO，至于核函数，是为处理非线性情况，若直接映射到高维计算恐维度爆炸，故在低维计算，等效高维表现。

    OK，理解到这第二层，已经能满足绝大部分人一窥SVM原理的好奇心，然对于那些想在证明层面理解SVM的则还很不够，但进入第三层理解境界之前，你必须要有比较好的数理基础和逻辑证明能力，不然你会跟我一样，吃不少苦头的。

**第三层、证明SVM**

    说实话，凡是涉及到要证明的东西.理论，便一般不是怎么好惹的东西。绝大部分时候，看懂一个东西不难，但证明一个东西则需要点数学功底，进一步，证明一个东西也不是特别难，难的是从零开始发明创造这个东西的时候，则显艰难(因为任何时代，大部分人的研究所得都不过是基于前人的研究成果，前人所做的是开创性工作，而这往往是最艰难最有价值的，他们被称为真正的先驱。牛顿也曾说过，他不过是站在巨人的肩上。你，我则更是如此)。

    正如陈希孺院士在他的著作「数理统计学简史」的第4章、最小二乘法中所讲：在科研上诸多观念的革新和突破是有着很多的不易的，或许某个定理在某个时期由某个人点破了，现在的我们看来一切都是理所当然，但在一切没有发现之前，可能许许多多的顶级学者毕其功于一役，耗尽一生，努力了几十年最终也是无功而返。

    OK，以下内容基本属于自己在看支持向量机导论一书的理解，包括自己对一些证明的理解，还是读书笔记。

**本部分导述**

3.1节线性学习器一部分中，主要阐述3个东西，感知机算法，松弛变量，及最小二乘理论，同时，基本上是贴的用相机拍的照片（为什么？懒）；

3.2节、核函数特征空间；

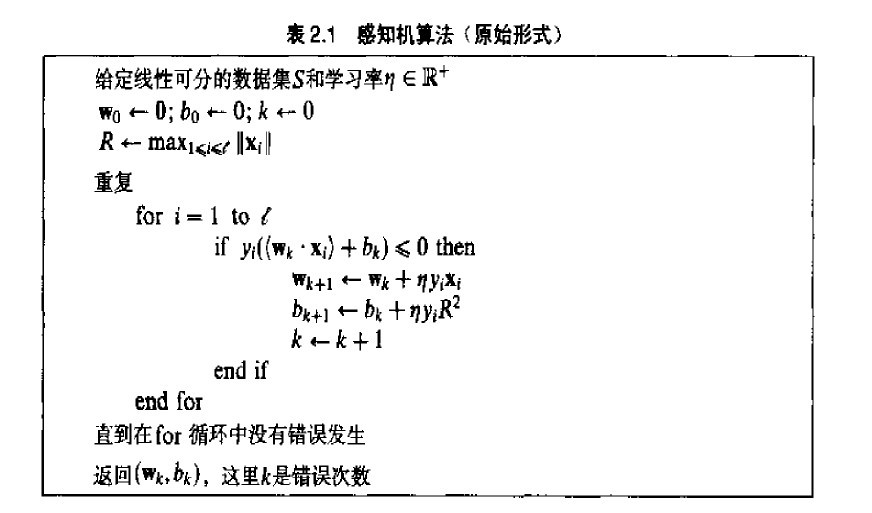
3.3节、SMO算法；

3.4节、简略谈谈SVM的应用。

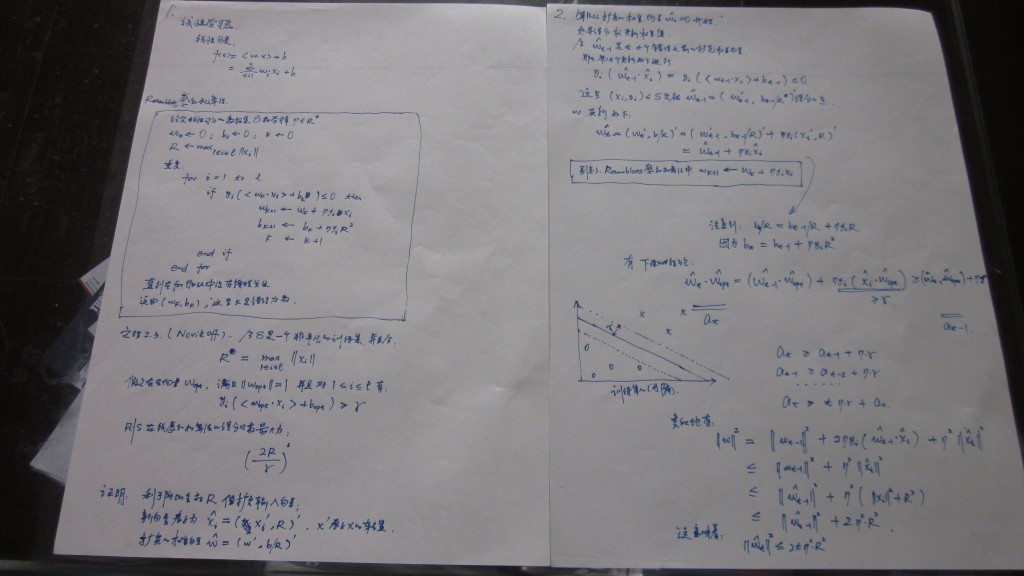
***3.1*、线性学习器**

**3.1.1、感知机算法**

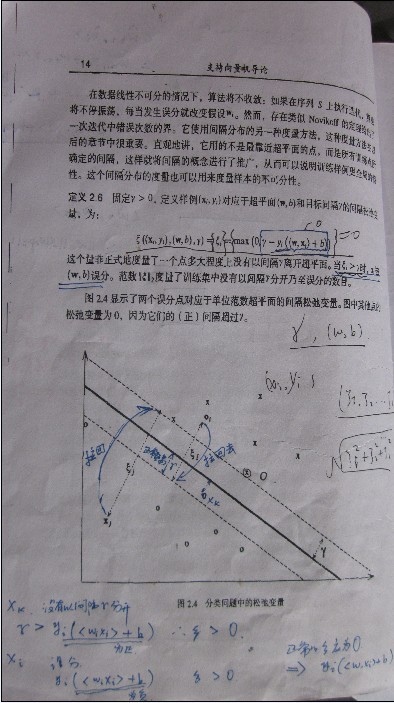
    这个感知机算法是1956年提出的，年代久远，依然影响着当今，当然，可以肯定的是，此算法亦非最优，后续会有更详尽阐述。不过，有一点，你必须清楚，这个算法是为了干嘛的：不断的训练试错以期寻找一个合适的超平面(是的，就这么简单)。

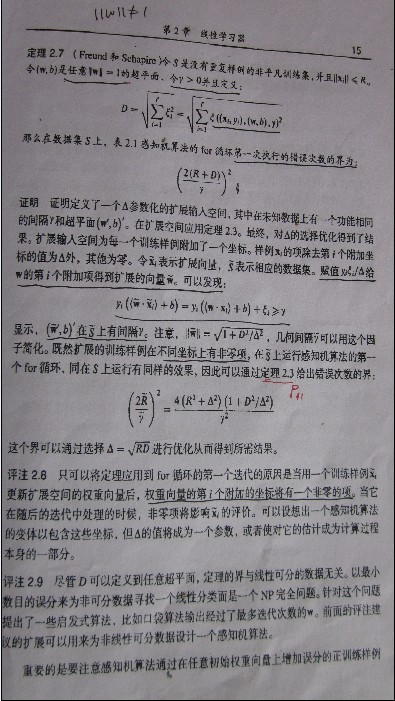


    下图算是读书笔记，若看不清，可以点击查看：[大图](http://photo.weibo.com/1580904460/photos/large/photo_id/3458285907662942)（后面一大堆的证明不过是为了证明上述感知机算法在线性条件下收敛，说白了，为了得到一个界，不至于无穷循环下去）。



**3.1.2、松弛变量**





    （点击查看大图：[左图](http://photo.weibo.com/1580904460/photos/detail/photo_id/3458252814538292#3458252814538292)，[右图](http://photo.weibo.com/1580904460/photos/detail/photo_id/3458252814538292#3458252822619770)）

    针对上面左图的说明：我们知道，当分类出现了误差，要么就是被误分，要么就是没有以正常的间隔被分开：

被误分。如上面左图所示，如果一切正常的话，那么xi出现的位置不该是那里，而是该左图中左边那个箭头所示，被“拉回去”，而既然出现在了这个不正常的位置，那么有什么后果内，这就导致了所谓的被误分，使得最终的松弛变量>0；同理，oj也不该出现在那个位置，而应该被“拉回去”。

没有以正常的间隔被分开。还是如上面所示左图，xk显然没有被以正常的间隔分开，而是过于靠近超平面。

    也就是说，对于正常的间隔大于r的点，这个值为0，而对于误分点，松弛变量大于*r*。同时，数据噪声使得单个点的间隔变小以致成为负值，因此，松弛变量可以作为数据噪声的度量，所以说，从松弛变量得来的方法适合处理噪声数据。

    还是不懂？别忘了，本文第一部分1.4节中，有：“这样一来，我们的 maximum margin classifier 的目标函数可以定义为：

maxγ˜˜ ˜˜

    当然，还需要满足一些条件，根据 margin 的定义，我们有

yi(wTxi+b)=γˆi≥γ ˆ,i=1,…,n "

***3.2*、最小二乘法**

    既然本节开始之前提到了最小二乘法，那么下面稍微简单阐述下。   

  我们口头中经常说：一般来说，平均来说。如平均来说，不吸烟的健康优于吸烟者，之所以要加“平均”二字，是因为凡事皆有例外，总存在某个特别的人他吸烟但由于经常锻炼所以他的健康状况可能会优于他身边不吸烟的朋友。而最小二乘法的一个最简单的例子便是算术平均。

    最小二乘法（又称最小平方法）是一种数学优化技术。它通过最小化误差的平方和寻找数据的最佳函数匹配。利用最小二乘法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小。用函数表示为：

Image(82).png

  使误差「所谓误差，当然是观察值与实际真实值的差量」平方和达到最小以寻求估计值的方法，就叫做最小二乘法，用最小二乘法得到的估计，叫做最小二乘估计。当然，取平方和作为目标函数只是众多可取的方法之一。

   最小二乘法的一般形式可表示为：

Image(83).png

    有效的最小二乘法是勒让德在 1805 年发表的，基本思想就是认为测量中有误差，所以所有方程的累积误差为

Image(68).jpeg

    我们求解出导致累积误差最小的参数即可。

βˆ==argminβ∑i=1ne2iargminβ∑i=1n[yi−(β0+β1x1i+⋯+βpxpi)]2

    勒让德在论文中对最小二乘法的优良性做了几点说明：

 最小二乘使得误差平方和最小，并在各个方程的误差之间建立了一种平衡，从而防止某一个极端误差取得支配地位

 计算中只要求偏导后求解线性方程组，计算过程明确便捷

最小二乘可以导出算术平均值作为估计值

    对于最后一点，从统计学的角度来看是很重要的一个性质。推理如下：假设真值为 θ, x1,⋯,xn为n次测量值, 每次测量的误差为ei=xi−θ，按最小二乘法，误差累积为

Image(69).jpeg

    求解θ 使得 L(θ) 达到最小，正好是算术平均 xˉ=∑ni=1xin。

    由于算术平均是一个历经考验的方法，而以上的推理说明，算术平均是最小二乘的一个特例，所以从另一个角度说明了最小二乘方法的优良性，使我们对最小二乘法更加有信心。

    最小二乘法发表之后很快得到了大家的认可接受，并迅速的在数据分析实践中被广泛使用。不过历史上又有人把最小二乘法的发明归功于高斯，这又是怎么一回事呢。高斯在1809年也发表了最小二乘法，并且声称自己已经使用这个方法多年。高斯发明了小行星定位的数学方法，并在数据分析中使用最小二乘方法进行计算，准确的预测了谷神星的位置。

    OK，更多请参看陈希孺院士的「数理统计学简史」的第4章、最小二乘法。

***3.3*、核函数特征空间**

   经过前面第一、二部分，我们已经知道，当把内积

就变成

之后，求

将有两种方法：

         1、先找到这种映射，然后将输入空间中的样本映射到新的空间中，最后在新空间中去求内积

。以多项式

 为例，对其进行变换，

，

，

，

，得到：

，也就是说通过把输入空间从二维向四维映射后，样本由线性不可分变成了线性可分，但是这种转化带来的直接问题是维度变高了，这意味着，首先可能导致后续计算变复杂，其次可能出现维度之咒，对于学习器而言就是：特征空间维数可能最终无法计算，而它的泛化能力(学习器对训练样本以外数据的适应性)会随着维度的增长而大大降低，这也违反了“奥坎姆的剃刀”，最终可能会使得内积

无法求出，于是也就失去了这种转化的优势了；

         2、或者是找到某种方法，它不需要显式的将输入空间中的样本映射到新的空间中而能够在输入空间中直接计算出内积

。它其实是对输入空间向高维空间的一种隐式映射，它不需要显式的给出那个映射，在输入空间就可以计算

，这就是传说中的核函数方法。

    OK，不再做过多介绍了，对核函数有进一步兴趣的，还可以看看此文：<http://www.cnblogs.com/vivounicorn/archive/2010/12/13/1904720.html>。

***3.4*、SMO算法**

    SMO算法是<<Minimal Optimization:A Fast Algorithm for Training Support Vector Machines》一文中提出的，作者信息：<http://research.microsoft.com/en-us/people/jplatt/>，其基本思想是将Vapnik在1982年提出的Chunking方法推到极致，即：通过将原问题分解为一系列小规模凸二次规划问题而获得原问题解的方法，每次迭代只优化由2个点组成的工作集，SMO算法每次启发式地选择两个因子同时固定其它因子来找到这两个因子的最优值，直到达到停止条件。

    上文2.1.1节已经提到过，SMO算法不过是为了解决对偶问题中对偶因子***α***的求解问题，这两篇文章：<http://www.cnblogs.com/jerrylead/archive/2011/03/18/1988419.html>，<http://www.cnblogs.com/vivounicorn/archive/2011/06/01/2067496.html>，已经介绍的相当详尽，包括算法思想及其实现，此处不再赘述。

***3.5*、SVM的应用**

    或许我们已经听到过，SVM在很多诸如文本分类，图像分类，生物序列分析和生物数据挖掘，手写字符识别等领域有很多的应用，但或许你并没强烈的意识到，SVM可以成功应用的领域远远超出现在已经在开发应用了的领域。

**3.5.1、文本分类**

    一个文本分类系统不仅是一个自然语言处理系统，也是一个典型的模式识别系统，系统的输入是需要进行分类处理的文本，系统的输出则是与文本关联的类别。由于篇幅所限，其它更具体内容本文将不再详述。

    OK，本节虽取标题为证明SVM，但聪明的读者们想必早已看出，其实本部分并无多少证明部分（特此致歉），怎么办呢？可以参阅支持向量机导论一书，此书精简而有趣。

    最后，非常感谢pluskid及诸多朋友&牛人们的文章及著作，让我有机会在其基础上总结、深入，本文基本成型。谢谢。

**参考文献及推荐阅读**

《**支持向量机导论**》，[美] Nello Cristianini / John Shawe-Taylor 著；

支持向量机导论一书的支持网站：<http://www.support-vector.net/>；

《数据挖掘导论》，[美] Pang-Ning Tan / Michael Steinbach / Vipin Kumar 著；

《数据挖掘：概念与技术》，(加)Jiawei Han;Micheline Kamber 著；

《数据挖掘中的新方法：支持向量机》，邓乃扬 田英杰 著；

《**支持向量机--理论、算法和扩展**》，邓乃扬 田英杰 著；

**支持向量机系列**，**pluskid**：<http://blog.pluskid.org/?page_id=683>；

<http://www.360doc.com/content/07/0716/23/11966_615252.shtml>；

[数据挖掘十大经典算法初探](http://blog.csdn.net/v_july_v/article/details/6142146)；

《模式识别支持向量机指南》，C.J.C Burges 著；

《**统计学习方法**》，李航著(第7章有不少内容参考自支持向量机导论一书，不过，可以翻翻看看)；

《统计自然语言处理》，宗成庆编著，第十二章、文本分类；

SVM入门系列，Jasper：<http://www.blogjava.net/zhenandaci/category/31868.html>；

最近邻决策和SVM数字识别的实现和比较，作者不详；

斯坦福大学机器学习课程原始讲义：<http://www.cnblogs.com/jerrylead/archive/2012/05/08/2489725.html>；

斯坦福机器学习课程笔记：<http://www.cnblogs.com/jerrylead/tag/Machine%20Learning/>；

<http://www.cnblogs.com/jerrylead/archive/2011/03/13/1982639.html>；

<http://www.cnblogs.com/jerrylead/archive/2011/03/18/1988419.html>；

[数据挖掘掘中所需的概率论与数理统计知识、上](http://blog.csdn.net/v_july_v/article/details/8308762)；

关于机器学习方面的文章，可以读读：<http://www.cnblogs.com/vivounicorn/category/289453.html>；

数学系教材推荐：<http://blog.sina.com.cn/s/blog_5e638d950100dswh.html>；

《神经网络与机器学习(原书第三版)》，[加] Simon Haykin 著；

正态分布的前世今生：<http://t.cn/zlH3Ygc>；

《**数理统计学简史**》，陈希孺院士著；

《最优化理论与算法(第2版)》，陈宝林编著；

A Gentle Introduction to Support Vector Machines in Biomedicine：<http://www.nyuinformatics.org/downloads/supplements/SVM_Tutorial_2010/Final_WB.pdf>，此PPT很赞，除了对引入拉格朗日对偶变量后的凸二次规划问题的深入度不够之外，其它都挺好，配图很精彩。

berkeleysong

绿色通道： 好文要顶 关注我 收藏该文[与我联系](http://msg.cnblogs.com/send/berkeleysong)

Image(84).png

Image(9).gif

[berkeleysong](http://home.cnblogs.com/u/berkeleysong/)

[关注 - 0](http://home.cnblogs.com/u/berkeleysong/followees)

[粉丝 - 2](http://home.cnblogs.com/u/berkeleysong/followers)

+加关注

0

0

(请您对文章做出评价)

[«](http://www.cnblogs.com/berkeleysong/p/3249760.html) 上一篇：[一些公司对quantitative的要求](http://www.cnblogs.com/berkeleysong/p/3249760.html)

[»](http://www.cnblogs.com/berkeleysong/p/3252663.html) 下一篇：[数据挖掘牛人的微博圈子](http://www.cnblogs.com/berkeleysong/p/3252663.html)

posted on 2013-08-11 13:32 [berkeleysong](http://www.cnblogs.com/berkeleysong/) 阅读(129) 评论(0)  [编辑](http://i.cnblogs.com/EditArticles.aspx?postid=3251245) [收藏](http://www.cnblogs.com/berkeleysong/articles/3251245.html#)

刷新评论[刷新页面](http://www.cnblogs.com/berkeleysong/articles/3251245.html#)[返回顶部](http://www.cnblogs.com/berkeleysong/articles/3251245.html#top)

注册用户登录后才能发表评论，请 登录 或 注册，[访问](http://www.cnblogs.com/)网站首页。

[【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库](http://www.ucancode.com/index.htm)

[【推荐】融云即时通讯云－专注为 App 开发者提供IM云服务](http://www.rongcloud.cn/)

**最新IT新闻**:

· [全球股市动荡或给风投市场降温：增加创业公司融资难度](http://news.cnblogs.com/n/527921/)

· [将癌细胞重编程为正常细胞](http://news.cnblogs.com/n/527920/)

· [Mozilla解释为什么选择WebExtensions扩展](http://news.cnblogs.com/n/527919/)

· [所以，你确定你真的想成为产品汪吗？！](http://news.cnblogs.com/n/527918/)

· [Apache Spark不过时的六大理由](http://news.cnblogs.com/n/527916/)

» [更多新闻...](http://news.cnblogs.com/)

**最新知识库文章**:

· [关于软件开发，你老板不知道的7件事](http://kb.cnblogs.com/page/527139/)

· [关于烂代码的那些事（中）](http://kb.cnblogs.com/page/526769/)

· [关于烂代码的那些事（上）](http://kb.cnblogs.com/page/526768/)

· [作为码农，我们为什么要写作](http://kb.cnblogs.com/page/526625/)

· [今天你写了自动化测试吗](http://kb.cnblogs.com/page/526474/)

» [更多知识库文章...](http://kb.cnblogs.com/)

**导航**

[首页](http://www.cnblogs.com/berkeleysong/)

[新随笔](http://i.cnblogs.com/EditPosts.aspx?opt=1)

[联系](http://msg.cnblogs.com/send/berkeleysong)

[订阅](http://www.cnblogs.com/berkeleysong/rss)

Image(10).gif

[管理](http://i.cnblogs.com/)

**公告**

机器学习小硕士

会写C++/C,会STL

会常见的数据结构与算法

会Linux的服务器开发

会数据挖掘常用算法

会推导机器学习公式

昵称：[berkeleysong](http://home.cnblogs.com/u/berkeleysong/)

园龄：[2年](http://home.cnblogs.com/u/berkeleysong/)

粉丝：[2](http://home.cnblogs.com/u/berkeleysong/followers/)

关注：[0](http://home.cnblogs.com/u/berkeleysong/followees/)

+加关注

**搜索**

**常用链接**

[我的随笔](http://www.cnblogs.com/berkeleysong/p/)

[我的评论](http://www.cnblogs.com/berkeleysong/MyComments.html)

[我的参与](http://www.cnblogs.com/berkeleysong/OtherPosts.html)

[最新评论](http://www.cnblogs.com/berkeleysong/RecentComments.html)

[我的标签](http://www.cnblogs.com/berkeleysong/tag/)

**随笔档案**

[2014年6月 (1)](http://www.cnblogs.com/berkeleysong/archive/2014/06.html)

[2014年5月 (56)](http://www.cnblogs.com/berkeleysong/archive/2014/05.html)

[2014年4月 (1)](http://www.cnblogs.com/berkeleysong/archive/2014/04.html)

[2013年9月 (4)](http://www.cnblogs.com/berkeleysong/archive/2013/09.html)

[2013年8月 (4)](http://www.cnblogs.com/berkeleysong/archive/2013/08.html)

**文章档案**

[2015年5月 (1)](http://www.cnblogs.com/berkeleysong/archives/2015/05.html)

[2014年9月 (3)](http://www.cnblogs.com/berkeleysong/archives/2014/09.html)

[2014年8月 (8)](http://www.cnblogs.com/berkeleysong/archives/2014/08.html)

[2014年7月 (1)](http://www.cnblogs.com/berkeleysong/archives/2014/07.html)

[2014年6月 (3)](http://www.cnblogs.com/berkeleysong/archives/2014/06.html)

[2014年5月 (20)](http://www.cnblogs.com/berkeleysong/archives/2014/05.html)

[2014年4月 (4)](http://www.cnblogs.com/berkeleysong/archives/2014/04.html)

[2014年3月 (8)](http://www.cnblogs.com/berkeleysong/archives/2014/03.html)

[2014年2月 (8)](http://www.cnblogs.com/berkeleysong/archives/2014/02.html)

[2014年1月 (1)](http://www.cnblogs.com/berkeleysong/archives/2014/01.html)

[2013年12月 (5)](http://www.cnblogs.com/berkeleysong/archives/2013/12.html)

[2013年11月 (6)](http://www.cnblogs.com/berkeleysong/archives/2013/11.html)

[2013年10月 (6)](http://www.cnblogs.com/berkeleysong/archives/2013/10.html)

[2013年9月 (2)](http://www.cnblogs.com/berkeleysong/archives/2013/09.html)

[2013年8月 (9)](http://www.cnblogs.com/berkeleysong/archives/2013/08.html)

Powered by:

[博客园](http://www.cnblogs.com/)

Copyright © berkeleysong