对外经济贸易大学
UNIVERSITY OF INTERNATIONAL BUSINESS AND ECONOMICS

支持向量机导论

信息学院-黄浩

2022年3月14日星期一

概述

SLT & SVM的地位和作用

- 是统计学习方法的优秀代表
- 有严密的数学依据，得到了严格的数学证明
- 有力反驳 —— “复杂的理论是没有用的，有用的是简单的算法”等错误观点
- 充分表明 —— “**没有什么比一个好的理论更实用了**”等基本的科学原则

SLT & SVM的数学基础

- 概率论与数理统计
- 泛函分析

SLT&SVM所坚持的“基本信念”

- **传统的估计高维函数依赖关系的方法所坚持的信念**
  - 实际问题中总存在较少数目的一些“**强特征**”，**用它**们的简单函数（**如**线性组合）就能较好地逼近未知函数。因此，需要**仔细地选择一个低维的特征空间，在**这个空间中用常规的统计技术来求解一个**逼近**。

- **SLT&SVM所坚持的信念**
  - 实际问题中存在较大数目的一些“**弱特征**”，**它**们“**巧妙的**”线性组合可较好地逼近未知的依赖关系。因此，**采用什么样的“弱特征”并不十分重要，而形成“巧妙的”**线性组合更为重要。

SLT&SVM与传统方法的区别

- 要较好地实现**传统方法**，需要人工选择（构造）一些数目相对较少的“**巧妙的特征**”
- **SVM方法**则是自动地选择（构造）一些数目较少的“**巧妙的特征**”
- 在实际应用中，可通过**构造两层（或多层）SVM**来选择“**巧妙的特征**”

SLT & SVM集以下模型于一身：

- 结构风险最小化（SRM）模型
- 数据压缩模型
- 构造复合特征的一个通用模型

  在希尔伯特空间中的内积回旋可以看作是构造特征的一种标准途径。
- 对实际数据的一种模型

  一个小的支持向量集合可能足以对不同的机器代表整个训练集。

SLT中的基本概念

- **统计方法** —— 从观测自然现象或者专门安排的实验所得到的数据去推断该事务可能的规律性。

- **统计学习理论** —— 在研究**小样本**统计估计和预测的过程中发展起来的一种新兴理论。

【**注意**】：这里所说的“**小**样本”**是相**对于无穷样本而言的，故只要样本数**不是无**穷，都可称为小样本，更严格地说，应该称为“**有限样本**”。

统计学习理论中的基本概念（续）

- **机器学习**
  - 主要研究从采集样本出发得出目前尚不能通过原理分析得到的规律,并利用这些规律对未来数据或无法观测的数据进行预测。
- **模式识别**
  - 对表征事务或现象的各种形式(数值、文字及逻辑关系等)信息进行处理和分析,以对事务或现象进行描述、辨认、分类和解释的过程。
- **统计学习理论**
  - 一种研究有限样本估计和预测的数学理论

统计学习理论的发展简况

- 学习过程的数学研究
  - F. Rosenblatt于1958,1962年把感知器作为一个学习机器模型
- 统计学习理论的开始
  - Novikoff(1962)证明了关于感知器的第一个定理
- 解决不适定问题的正则化原则的发现
  - Tikhonov(1963), Ivanov(1962), Phillips(1962)
- Vanik和Chervonenkis(1968)提出了**VC熵**和**VC维**的概念
  - 提出了统计学习理论的核心概念
  - 得到了关于收敛速度的非渐进界的主要结论

SLT的发展简况(续)

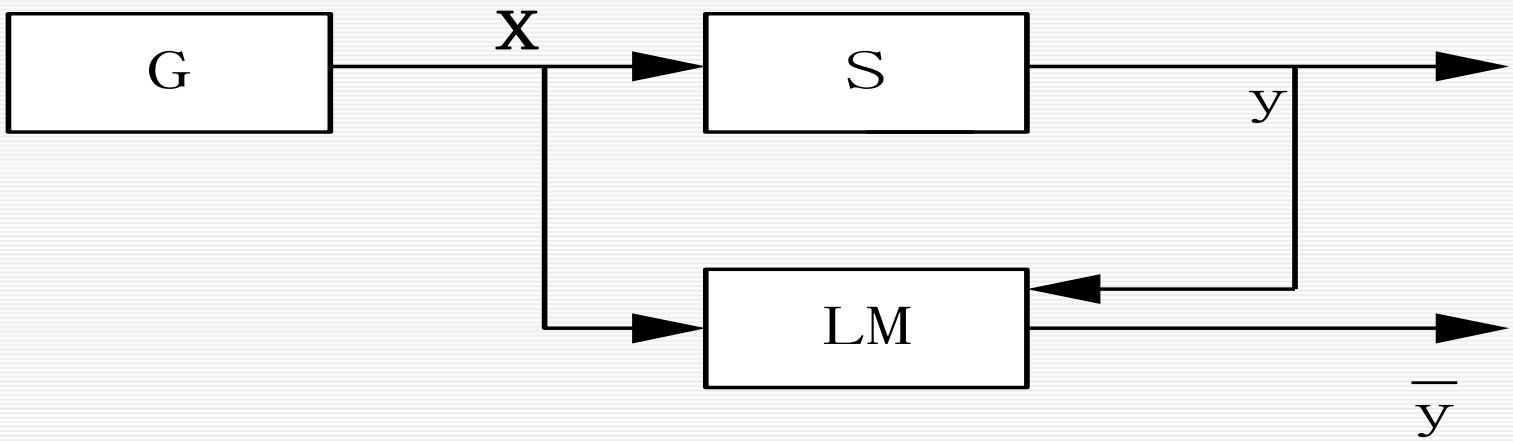
- Vapnik和Chervonenkis(1974)提出了**结构风险最小化（SRM）归纳原则**。
- Vapnik和Chervonenkis(1989)发现了经验风险最小化归纳原则和最大似然方法一致性的充分必要条件,**完成了**对经验风险最小化归纳推理的分析。
- 90**年代中期,有限**样本情况下的机器学习理论研究逐渐成**熟起来,形成了**较完善的理论体系—统计学习理论(Statistical Learning Theory,简称SLT)

统计学习理论的基本内容

- 机器学习的基本问题
- 统计学习理论的核心内容

机器学习的基本问题

- 机器学习问题的表示



学习问题的表示

- 产生器(G)，产生随机向量x属于R^n，它们是从固定但未知的概率分布函数F(x)中独立抽取的。
- 训练器(S)，对每个输入向量x返回一个输出值y，产生输出的根据是同样固定但未知的条件分布函数 F(y|x)。
- 学习机器(LM)，它能够实现一定的函数集f(x, a)，a属于A，其中A是参数集合。

机器学习的基本问题

- 机器学习就是从给定的函数集f(x,α)(α是参数)中,选择出能够最好地逼近训练器响应的函数。
- 机器学习的目的可以形式化地表示为：根据n个独立同分布的观测样本 $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$

  在一组函数 $\{f(x,\alpha)\}$ 中求出一个最优函数 $f(x,\alpha_0)$ 对训练器的响应进行估计,使期望风险最小 $R(\alpha) = \int L(y, f(x,\alpha)) dP(x,y)$

  其中 $P(x,y)$ 是未知的,对于不同类型的机器学习问题有不同形式的损失函数。

三类基本的机器学习问题

- 模式识别
- 函数逼近（回归估计）
- 概率密度估计

【补充说明】：用有限数量信息解决问题的**基本原则** —— **在解决一个给定问题时，要设法避免把解决一个更为一般的问题作为其中间步骤。**

- 上述原则意味着，当解决模式识别或回归估计问题时，**必须设法去“直接”寻找待求的函数，而不是首先估计密度，然后用估计的密度来构造待求的函数。**

- **密度估计是**统计学中的一个全能问题，即知道了密度就可以解决各种问题。一般地，估计密度是一个不适定问题(ill-posed problem)，**需要大量**观测才能较好地解决。

- 实际上，需要解决的问题（如决策规则估计或回归估计）是很特殊的，**通常只需要有某一合理数量的观测就可以解决。**

经验风险最小化原则

- 对于未知的概率分布,**最小化**风险函数, **只有**样本的信息**可以利用**,这导致了定义的期望风险是无法直接计算和最**小化的**。
- **根据概率**论中大数定理,**可用算**术平均代替数据期望,**于是定**义了经验风险

$$R_{emp}(w) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i, w))$$

来逼近期望风险。

- 经验风险最小化(ERM)**原**则：使用对参数w**求**经验风险$R_{emp}(w)$**的最小**值代替求期望风险 $R(w)$**的最小**值。

经验风险最小化

- 从期望风险最小化到经验风险最小化没有可靠的依据,只是直观上合理的想当然。

  - 期望风险和经验风险都是w的函数,概率论中的大数定理只说明了当样本趋于无穷多时经验风险将在概率意义上趋近于期望风险,并没有保证两个风险的w是同一点,更不能保证经验风险能够趋近于期望风险。

  - 即使有办法使这些条件在样本数无穷大时得到保证, 也无法认定在这些前提下得到的经验风险最小化方法在样本数有限时仍能得到好的结果。

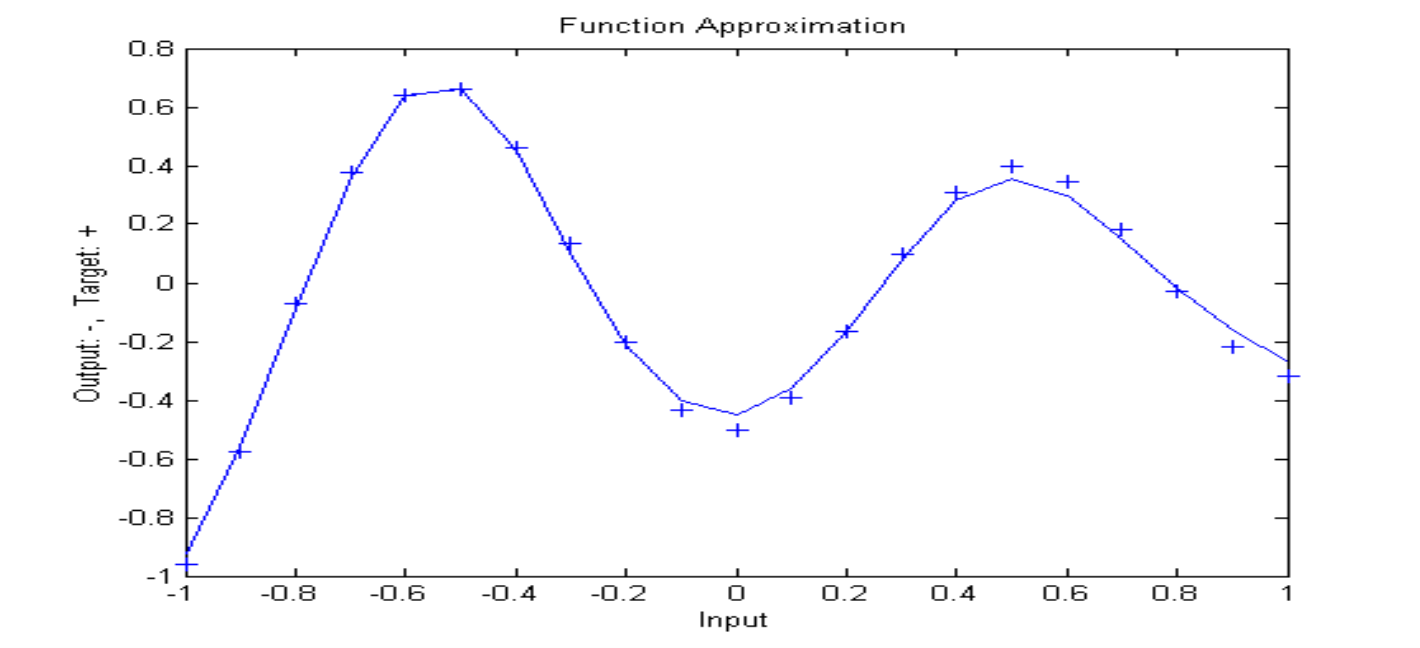
复杂性与推广能力

- 学习机器对未来输出进行正确预测的能力称作**推广能力**（也称为**“泛化能力”**）。

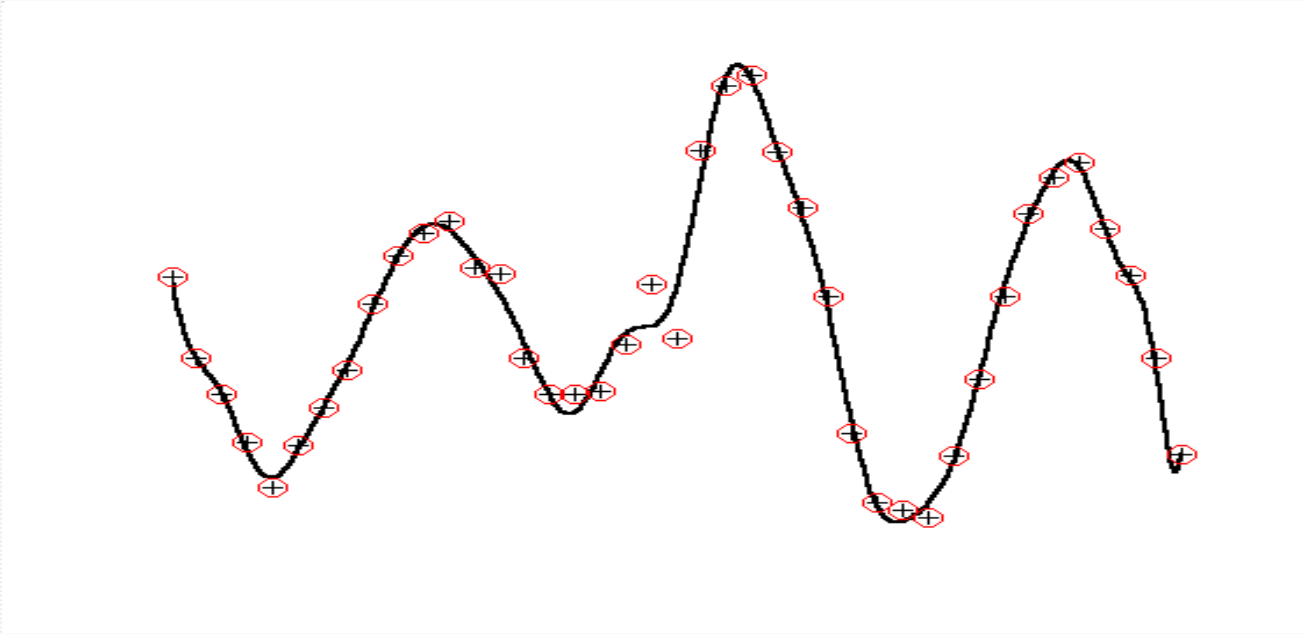
- **在某些情况下**,训练误差过小反而导致推广能力的下降,这就是**过学习**问题。

- **神**经网络的过学习问题是经验风险最小化原则失败的一**个典型例子**。

用三角函数拟合任意点


Function Approximation
Output: -, Target: +
Input
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

学习的示例



复杂性与推广能力（续）

- 在有限样本情况下，
  - 经验风险最小并不一定意味着期望风险最小；
  - 学习机器的复杂性不但与所研究的系统有关,而且要和有限的学习样本相适应；
  - **学习精度和推广性之间似乎是一对不可调和的矛盾,采用复杂的学习机器虽然容易使得学习误差更小,却往往丧失推广性；**
  - 传统的解决办法（例如：采用正则化、模型选择、噪声干扰等方法以控制学习机器的复杂度）缺乏坚实的理论基础。

统计学习理论的核心内容

- SLT被认为是目前针对有限样本统计估计和预测学习的最佳理论，它从理论上较为系统地研究了经验风险最小化原则成立的条件、有限样本下经验风险与期望风险的关系及如何利用这些理论找到新的学习原则和方法等问题。

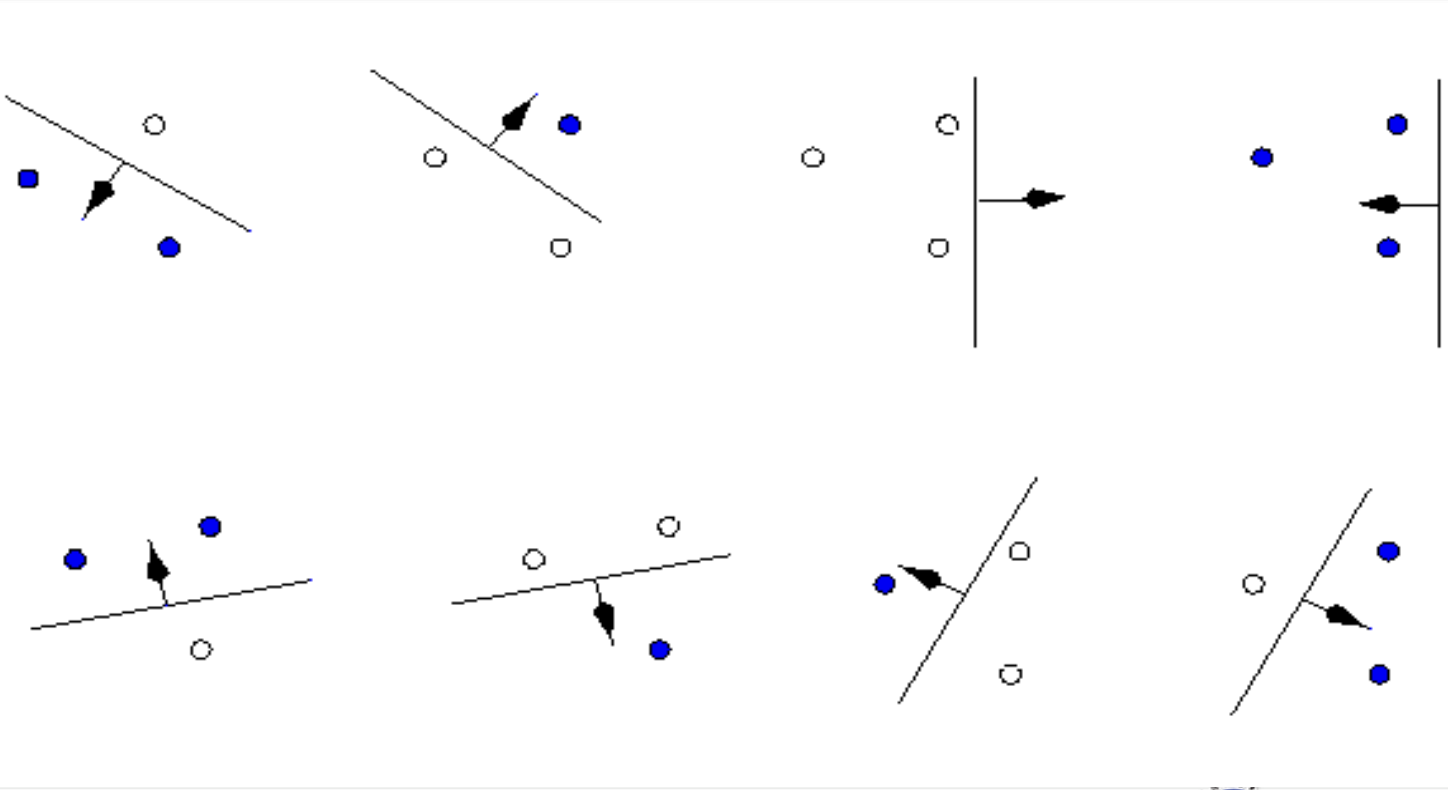
- SLT的主要内容包括：
  - 基于经验风险原则的统计学习过程的一致性理论
  - 学习过程收敛速度的非渐进理论
  - 控制学习过程的推广能力的理论
  - 构造学习算法的理论

VC维(函数的多样性)

- 为了研究经验风险最小化函数集的学习一致收敛速度和推广性，SLT定义了一些指标来衡量函数集的性能，其中最重要的就是VC维(Vapnik-Chervonenkis Dimension)。

- **VC维**：对于一个指示函数（即只有0和1两种取值的函数）集，如果存在h个样本能够被函数集里的函数按照所有可能的2^h种形式分开，则称函数集能够把h个样本打散，函数集的VC维就是能够打散的最大样本数目。

- 如果对任意的样本数，总有函数能打散它们，则函数集的VC维就是无穷大。



VC维（续）

- 一般而言,VC维越大, 学习能力就越强,但学习机器也越复杂。
- 目前还没有通用的关于计算任意函数集的VC维的理论,只有对一些特殊函数集的VC维可以准确知道。
- N维实数空间中线性分类器和线性实函数的VC维是n+1。
- Sin(ax)的VC维为无穷大。
- .....

VC维（续）

Open problem: 对于给定的学习函数集,**如何用理**论或实验的方法计算其VC维是当前统计学习理论研**究中有待解决的一个**难点问题。

三个里程碑定理

收敛的充分(必要)条件(VC熵) $\lim_{x\to\infty}\frac{H(n)}{n}=0$

快收敛速度的充分条件 $\lim_{x\to\infty}\frac{H_{ann}(n)}{n}=0$

与概率测度无关的快收敛充要条件 $\lim_{x\to\infty}\frac{G(n)}{n}=0$

推广性的界

- SLT系统地研究了经验风险和实际风险之间的关系,也即推广性的界。
- 根据SLT中关于函数集推广性界的理论,对于指示函数集中所有的函数,经验风险 $R_{emp}(\alpha)$ 和实际风险 $R(\alpha)$ 之间至少以概率 $1-\eta$ 满足如下关系:

$$R(\alpha) \le R_{emp}(\alpha) + \sqrt{\frac{h(\ln(2n/h)+1) - \ln(\eta/4)}{n}}$$

其中，h是函数集的VC维,n是样本数。

推广性的界（续1）

$$R(\alpha) \le R_{emp}(\alpha) + \sqrt{\frac{h(\ln(2n/h)+1) - \ln(\eta/4)}{n}}$$

- 学习机器的实际风险由两部分组成:
  - 训练样本的经验风险
  - 置信范围(同置信水平 $1-\eta$有关,而且同学习机器的VC维和训练样本数有关。

$$R(\alpha) \le R_{emp}(\alpha) + \Phi(\frac{n}{h})$$

- 在训练样本有限的情况下,学习机器的VC维越高,则置信范围就越大,导致实际风险与经验风险之间可能的差就越大。

推广性的界（续2）

- 在设计分类器时，不但要使经验风险最小化，还要使VC维尽量小，从而缩小置信范围，使期望风险最小。

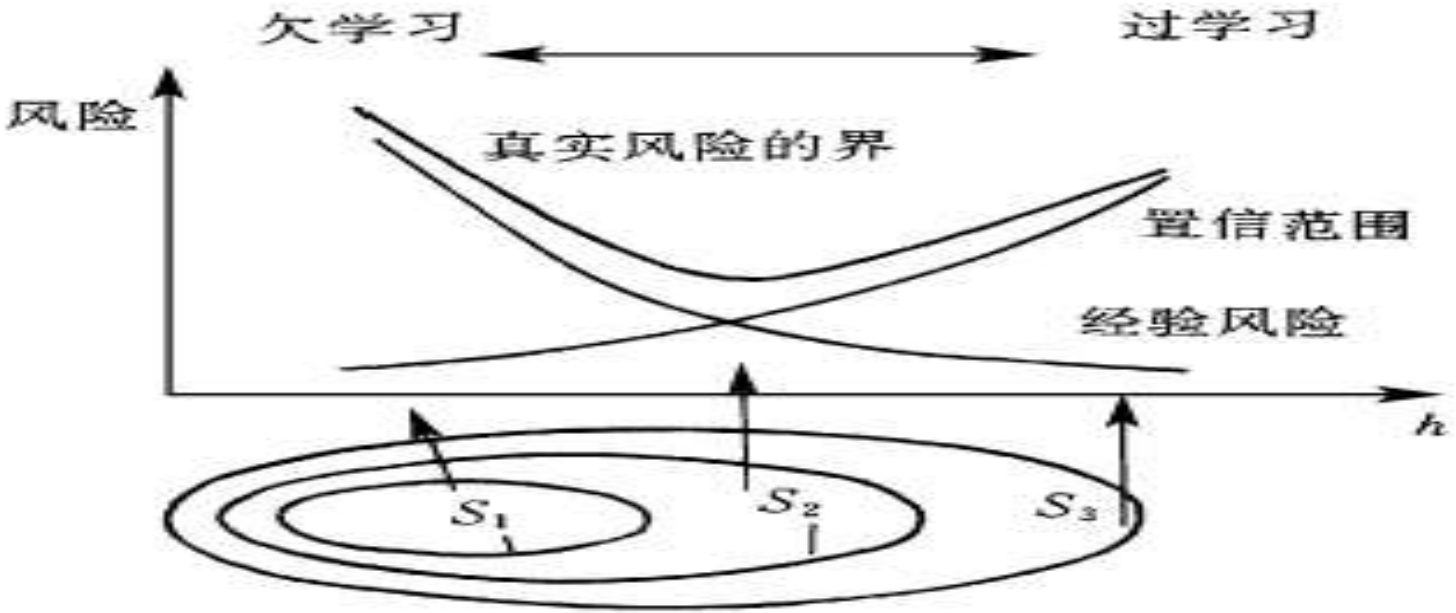
- 寻找反映学习机器的能力的更好参数，从而得到更好的界是SLT今后的重要研究方向之一。

结构风险最小化

- 传统机器学习方法中普遍采用的经验风险最小化原则在样本数目有限时是不合理的,**因此，需要同**时最小化经验风险和置信范围。

- 统计学习理论提出了一种新的策略,**即把函数集构造**为一**个函数子集序列,使各个子集按照**VC维的大小排列;**在每个子集中**寻找最小经验风险,**在子集**间折衷考虑经验风险和**置信范**围,**取得**实际风险的最小。这种思想称作结构风险最小化(Structural Risk Minimization)，**即SRM准**则。

结构风险最小化（续1）



函数集子集：$S_1 \subset S_2 \subset S_3$

VC 维：$h_1 \leqslant h_2 \leqslant h_3$

结构风险最小化（续2）

- 实现SRM原则的两种思路
  - **在每个子集中求最小**经验风险,**然后**选择使最小经验风险和置信范围之和最小的子集。
  - 设计函数集的某种结构使每个子集中都能取得最小的经验风险,**然后只需**选择适当的子集使置信范围最小,则这个子集中使经验风险最小的函数就是最优函数。**支持向量机方法**实际上就是这种思路的实现。

了解SVM

- 支持向量机，因其英文名为support vector machine，故一般简称SVM，通俗来讲，它是一种二类分类模型，其基本模型定义为特征空间上的间隔最大的线性分类器，其学习策略便是间隔最大化，最终可转化为一个凸二次规划问题的求解。

了解SVM

分类标准的起源：Logistic回归

理解SVM，必须先弄清楚一个概念：线性分类器。给定一些数据点，它们分别属于两个不同的类，现在要找到一个线性分类器把这些数据分成两类。如果用x表示数据点，用y表示类别（y可以取1或者-1，分别代表两个不同的类），一个线性分类器的学习目标便是要在n维的数据空间中找到一个超平面（hyper plane），这个超平面的方程可以表示为（ w^T中的T代表转置）

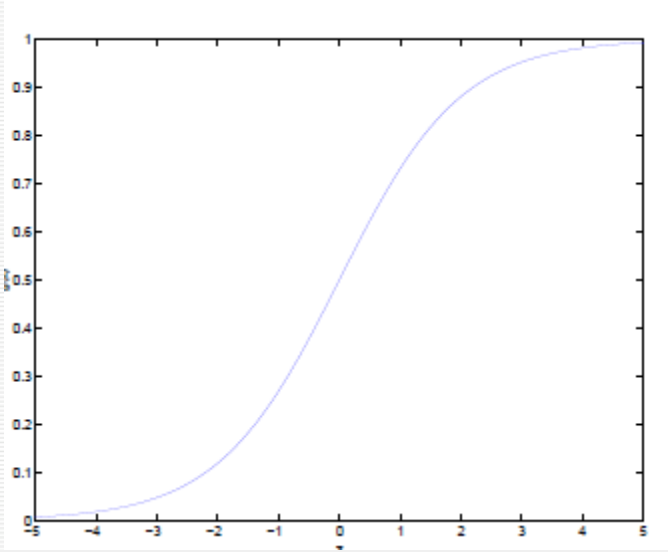
$$w^T x + b = 0$$

- 类别取1或-1的标准起源于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。
- 假设函数如下，其中x是n维特征向量，函数g就是logistic函数。

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

了解SVM

$$g(z) = \frac{1}{1 + e^{-z}}$$



- 当要判别一个新来的特征属于哪个类时，只需求 $h_\theta(x)$ 即可，若 $h_\theta(x)$ 大于0.5就是y=1的类，反之属于y=0类。

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。

$$P(y = 1 \mid x; \theta) = h_\theta(x)$$
$$P(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

了解SVM

此外，$h_\theta(x)$只和$\theta^T x$有关，$\theta^T x>0$，那么$h_\theta(x) > 0.5$，而g(z)只是用来映射，真实的类别决定权还是在于$\theta^T x$。再者，当$\theta^T x \gg 0$时，$h_\theta(x)$=1，反之$h_\theta(x)$=0。如果我们只从$\theta^T x$出发，希望模型达到的目标就是让训练数据中y=1的特征$\theta^T x \gg 0$，而是y=0的特征$\theta^T x \ll 0$。Logistic回归就是要学习得到θ，使得正例的特征远大于0，负例的特征远小于0，而且要在全部训练实例上达到这个目标。

接下来，尝试把logistic回归做个变形。首先，将使用的结果标签y = 0和y = 1替换为y = -1,y = 1，然后将$\theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$（$x_0 = 1$）中的$\theta_0$替换为b，最后将后面的$\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$替换为$\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$（即$w^T x$）。如此，则有了$\theta^T x = w^T x + b$。也就是说除了y由y=0变为y=-1外，线性分类函数跟logistic回归的形式化表示$h_\theta(x) = g(\theta^T x) = g(w^T x + b)$没区别。

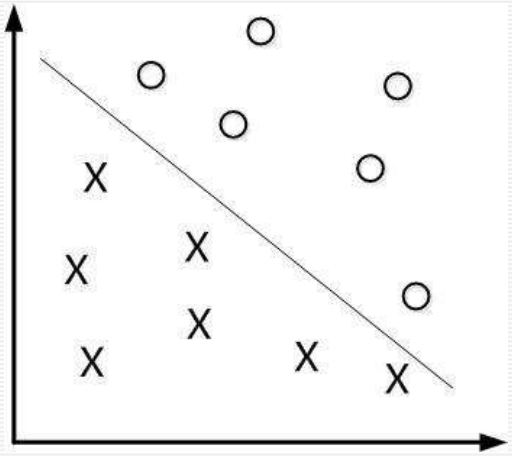
进一步，可以将假设函数$h_{w,b}(x) = g(w^T x + b)$中的g(z)做一个简化，将其简单映射到y=-1和y=1上。映射关系如下：

$$g(z) = \begin{cases} 1, & z \geq 0 \\ -1, & z < 0 \end{cases}$$

了解SVM

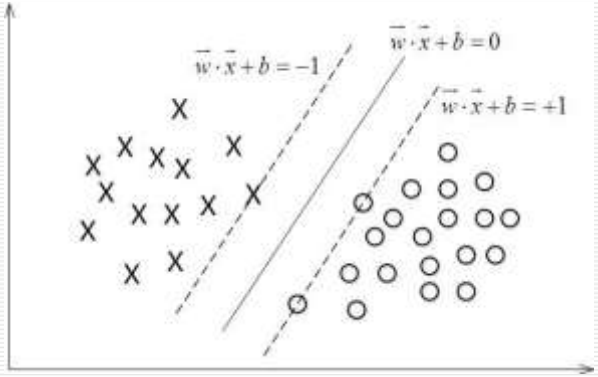
线性分类的一个例子

下面举个简单的例子。如下图所示，现在有一个二维平面，平面上有两种不同的数据，分别用圈和叉表示。由于这些数据是线性可分的，所以可以用一条直线将这两类数据分开，这条直线就相当于一个超平面，超平面一边的数据点所对应的y全是-1 ，另一边所对应的y全是1。



了解SVM

这个超平面可以用分类函数 $f(x) = w^T x + b$ 表示，当f(x) 等于0的时候，x便是位于超平面上的点，而f(x)大于0的点对应 y=1 的数据点，f(x)小于0的点对应y=-1的点，如下图所示：



注：有的资料上定义特征到结果的输出函数 $u = \vec{w}\vec{x} - b$，与这里定义的 $f(x) = w^T x + b$ 实质是一样的。为什么？因为无论是 $u = \vec{w}\cdot\vec{x} - b$，还是 $f(x) = w^T x + b$，不影响最终优化结果。下文你将看到，当我们转化到优化

$\max \frac{1}{\|w\|}$， $s.t., y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n$ 的时候，为了求解方便，会把yf(x)令为1，即yf(x)是y(w^x + b)，还是y(w^x - b)，对我们要优化的式子max1/||w||已无影响。

UIBE 对外经济贸易大学 UNIVERSITY OF INTERNATIONAL BUSINESS AND ECONOMICS

了解SVM

函数间隔Functional margin与几何间隔Geometrical margin

接下来的问题是，如何确定这个超平面呢？从直观上而言，这个超平面应该是最适合分开两类数据的直线。而判定“最适合”的标准就是这条直线离直线两边的数据的间隔最大。所以，得寻找有着最大间隔的超平面。

在超平面w*x+b=0确定的情况下，|w*x+b|能够表示点x到距离超平面的远近，而**通过观察w*x+b的符号与类标记y的符号是否一致可判断分类是否正确**，所以，可以用(y*(w*x+b))的正负性来判定或表示分类的正确性。于此，我们便引出了函数间隔（functional margin）的概念。

定义函数间隔（用 $\hat{\gamma}$ 表示）为： $\hat{\gamma} = y(w^T x + b) = yf(x)$

而超平面(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），便为超平面(w, b)关于训练数据集T的函数间隔：

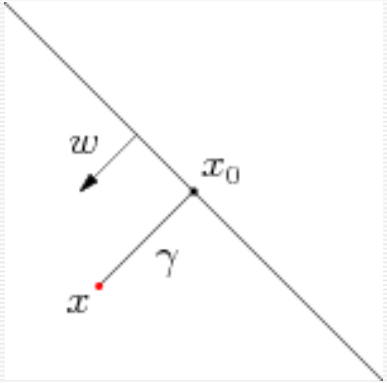
$\hat{\gamma} = \min \hat{\gamma}_i \ (i=1, ...n)$

但这样定义的函数间隔有问题，即如果成比例的改变w和b（如将它们改成2w和2b），则函数间隔的值f(x)却变成了原来的2倍（虽然此时超平面没有改变），所以只有函数间隔还远远不够。

事实上，我们可以对法向量w加些约束条件，从而引出真正定义点到超平面的距离--几何间隔（geometrical margin）的概念。

了解SVM

假定对于一个点 x ，令其垂直投影到超平面上的对应点为 x0 ，w 是垂直于超平面的一个向量，γ 为样本x到超平面的距离，如下图所示：



$$x = x_0 + \gamma \frac{w}{||w||}$$

了解SVM

其中||w||为w的二阶范数（范数是一个类似于模的表示长度的概念），$\frac{w}{||w||}$是单位向量（一个向量除以它的模称之为单位向量）。

又由于x0 是超平面上的点，满足 f(x0)=0，代入超平面的方程 $w^T x + b = 0$ ，可得$w^T x_0 + b = 0$，即$w^T x_0 = -b$。

随即让此式 $x = x_0 + \gamma \frac{w}{\|w\|}$ 的两边同时乘以w^T，再根据$w^T x_0 = -b$和$w^T w = \|w\|^2$，即可算出γ：

$$\gamma = \frac{w^T x + b}{\|w\|} = \frac{f(x)}{\|w\|}$$

为了得到γ的绝对值，令γ乘上对应的类别 y，即可得出几何间隔（用$\tilde{\gamma}$表示）的定义：

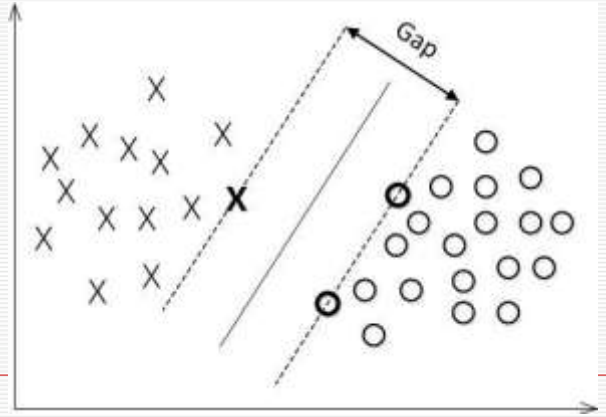
$$\tilde{\gamma} = y\gamma = \frac{\hat{\gamma}}{\|w\|}$$

了解SVM

最大间隔分类器Maximum Margin Classifier的定义

从上述函数间隔和几何间隔的定义可以看出：几何间隔就是函数间隔除以||w||，而且函数间隔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的距离。

对一个数据点进行分类，当超平面离数据点的“间隔”越大，分类的确信度（confidence）也越大。所以，为了使得分类的确信度尽量高，需要让所选择的超平面能够最大化这个“间隔”值。这个间隔就是下图中的Gap的一半。



了解SVM

通过由前面的分析可知：函数间隔不适合用来最大化间隔值，因为在超平面固定以后，可以等比例地缩放w的长度和b的值，这样可以使得 $f(x) = w^T x + b$ 的值任意大，亦即函数间隔 $\hat{\gamma}$ 可以在超平面保持不变的情况下被取得任意大。但几何间隔因为除上了 $\|w\|$，使得在缩放w和b的时候几何间隔 $\tilde{\gamma}$ 的值是不会改变的，它只随着超平面的变动而变动，因此，这是更加合适的一个间隔。换言之，这里要找的最大间隔分类超平面中的"间隔"指的是几何间隔。

于是最大间隔分类器（maximum margin classifier）的目标函数可以定义为：

$$\max \tilde{\gamma}$$

同时需满足一些条件，根据间隔的定义，有

$$y_i(w^T x_i + b) = \hat{\gamma}_i \geq \hat{\gamma}, \quad i = 1, \ldots, n$$

了解SVM

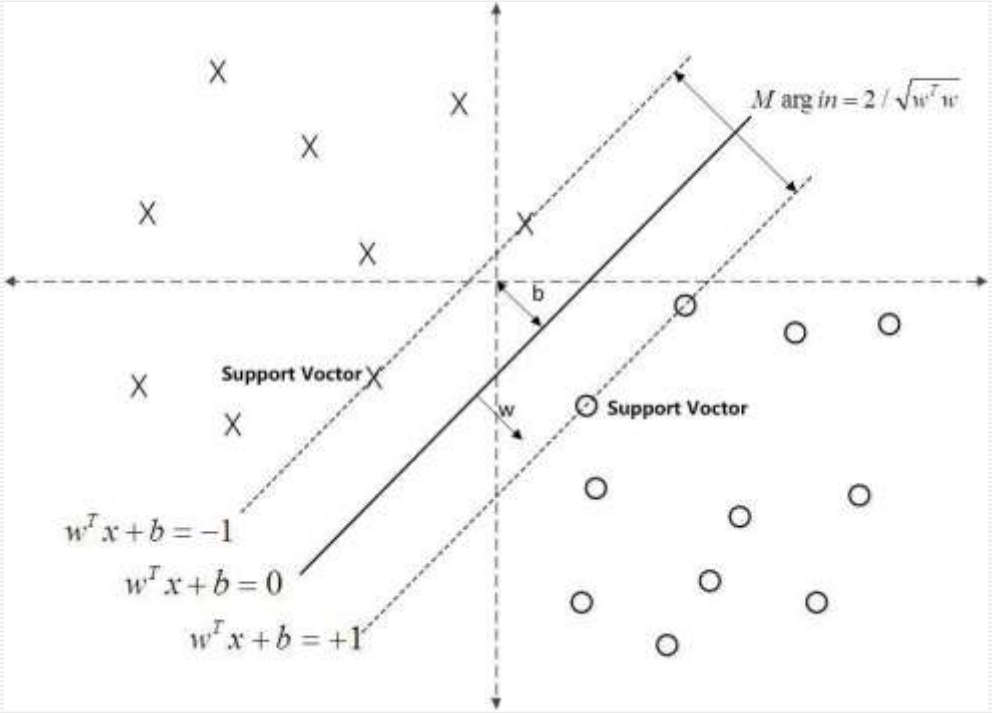
回顾下几何间隔的定义 $\tilde{\gamma} = y\gamma = \frac{\hat{\gamma}}{\|w\|}$ ，可知：如果令函数间隔 $\hat{\gamma}$ 等于1（之所以令 $\hat{\gamma}$ 等于1，是为了方便推导和优化，且这样做对目标函数的优化没有影响），则有 $\tilde{\gamma}$ = 1 / ||w||且 $y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n$ ，从而上述目标函数转化成了

$$\max \frac{1}{\|w\|}, \quad s.t., y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n$$

相当于在相应的约束条件 $y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n$ 下，最大化这个1/||w||值，而1/||w||便是几何间隔 $\tilde{\gamma}$ 。

如下图所示，中间的实线便是寻找到的最优超平面（Optimal Hyper Plane），其到两条虚线边界的距离相等，这个距离便是几何间隔 $\tilde{\gamma}$ ，两条虚线间隔边界之间的距离等于2 $\tilde{\gamma}$ ，而虚线**间隔边界上的点则是支持向量。**由于这些支持向量刚好在虚线间隔边界上，所以它们满足 $y(w^T x + b) = 1$ （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 $\hat{\gamma}$ =1），而对于所有不是支持向量的点，则显然有 $y(w^T x + b) > 1$ 。

了解SVM



如果仅仅关心如何使用SVM，了解到此基本足够。

深入SVM

从线性可分到线性不可分

从原始问题到对偶问题的求解

接着考虑之前得到的目标函数：

$$\max \frac{1}{\|w\|} \quad s.t., y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n$$

由于求$\frac{1}{\|w\|}$的最大值相当于求$\frac{1}{2}\|w\|^2$的最小值，所以上述目标函数等价于（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）：

$$\min \frac{1}{2}\|w\|^2 \quad s.t., y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n$$

深入SVM

- 因为现在的目标函数是二次的，约束条件是线性的，所以它是一个凸二次规划问题。这个问题可以用现成的**QP (Quadratic Programming)** 优化包进行求解。一言以蔽之：在一定的约束条件下，目标最优，损失最小。
- 此外，由于这个问题的特殊结构，还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。
- 那什么是拉格朗日对偶性呢？简单来讲，通过给每一个约束条件加上一个拉格朗日乘子（**Lagrange multiplier**），定义拉格朗日函数（通过拉格朗日函数将约束条件融合到目标函数里去，从而只用一个函数表达式便能清楚的表达出我们的问题）：

$$\mathcal{L}(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i\left(y_i(w^Tx_i+b)-1\right)$$

$$\theta(w)=\max_{\alpha_i\geq 0}\mathcal{L}(w,b,\alpha)$$

深入SVM

容易验证，当某个约束条件不满足时，例如$y_i(w^Tx_i+b)<1$，那么显然有$\theta(w)=\infty$（只要令$\alpha_i=\infty$即可）。而当所有约束条件都满足时，则最优值为$\theta(w)=\frac{1}{2}\|w\|^2$，亦即最初要最小化的量。

因此，在要求约束条件得到满足的情况下最小化$\frac{1}{2}\|w\|^2$，实际上等价于直接最小化$\theta(w)$（当然，这里也有约束条件，就是$\alpha_i \geq 0, i=1,\ldots,n$），因为如果约束条件没有得到满足，$\theta(w)$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，目标函数变成了：

$$\min_{w,b}\theta(w)=\min_{w,b}\max_{\alpha_i\geq 0}\mathcal{L}(w,b,\alpha)=p^*$$

交换以后的新问题是原始问题的对偶问题，这个新问题的最优值用d^*来表示。而且有$d^*\leq p^*$，在满足某些条件的情况下，这两者相等，这个时候就可以通过求解对偶问题来间接地求解原始问题。

换言之，之所以从minmax的原始问题p^*，转化为maxmin的对偶问题d^*，一者因为d^*是p^*的近似解，二者，转化为对偶问题后，更容易求解。

下面可以先求L 对w、b的极小，再求L 对α的极大。

深入SVM

KKT条件

上文中提到“$d^* \le p^*$ 在满足某些条件的情况下，两者等价”，这所谓的“满足某些条件”就是要满足KKT条件。

一般地，一个最优化数学模型能够表示成下列标准形式：

$$\min f(\mathbf{x})$$

$$\text{s.t.} \quad h_j(\mathbf{x}) = 0, j = 1,\ldots, p,$$

$$g_k(\mathbf{x}) \le 0, k = 1,\ldots, q,$$

$$\mathbf{x} \in \mathbf{X} \subset \Re^n$$

- 其中，f(x)是需要最小化的函数，h(x)是等式约束，g(x)是不等式约束，p和q分别为等式约束和不等式约束的数量。
- 同时，要明白以下两点：

深入SVM

- 凸优化的概念：$\mathcal{X} \subset \mathbb{R}^n$ 为一凸集，$f: \mathcal{X} \to \mathbb{R}$ 为一凸函数。凸优化就是要找出一点 $x^* \in \mathcal{X}$，使得每一 $x \in \mathcal{X}$ 满足 $f(x^*) \leq f(x)$。
- KKT条件的意义：它是一个非线性规划（Nonlinear Programming）问题能有最优化解法的必要和充分条件。

而KKT条件就是指上面最优化数学模型的标准形式中的最小点 x* 必须满足下面的条件：

1. $h_j(\mathbf{x}_\cdot) = 0, j = 1,\ldots,p,\ g_k(\mathbf{x}_\cdot) \leq 0, k = 1,\ldots,q,$

2. $\nabla f(\mathbf{x}_\cdot) + \sum_{j=1}^{p} \lambda_j \nabla h_j(\mathbf{x}_\cdot) + \sum_{k=1}^{q} \mu_k \nabla g_k(\mathbf{x}_\cdot) = \mathbf{0},$

$\lambda_j \neq 0,\ \mu_k \geq 0,\ \mu_k g_k(\mathbf{x}_\cdot) = 0.$

- 也就是说，原始问题通过满足KKT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，然后求对的极大，最后利用SMO算法求解对偶问题中的拉格朗日乘子。

深入SVM

对偶问题求解的3个步骤

（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

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$$

将以上结果代入之前的L：

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i (w^T x_i + b) - 1 \right)$$

深入SVM

$$\mathcal{L}(w,b,\alpha)=\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j-\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j-b\sum_{i=1}^{n}\alpha_i y_i+\sum_{i=1}^{n}\alpha_i$$
$$=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j$$

$$\mathcal{L}(\mathrm{w},\mathrm{b},\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{m}\alpha_i\left[y^{(i)}\left(w^T x^{(i)}+b\right)-1\right]$$

$$=\frac{1}{2}\mathrm{w}^T\mathrm{w}-\sum_{i=1}^{m}\alpha_i y^{(i)}w^T x^{(i)}-\sum_{i=1}^{m}\alpha_i y^{(i)}b+\sum_{i=1}^{m}\alpha_i$$

$$=\frac{1}{2}\mathrm{w}^T\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}-\sum_{i=1}^{m}\alpha_i y^{(i)}w^T x^{(i)}-\sum_{i=1}^{m}\alpha_i y^{(i)}b+\sum_{i=1}^{m}\alpha_i$$

$$=\frac{1}{2}\mathrm{w}^T\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}-w^T\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}-\sum_{i=1}^{m}\alpha_i y^{(i)}b+\sum_{i=1}^{m}\alpha_i$$

$$=-\frac{1}{2}\mathrm{w}^T\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}-\sum_{i=1}^{m}\alpha_i y^{(i)}b+\sum_{i=1}^{m}\alpha_i$$

$$=-\frac{1}{2}\mathrm{w}^T\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}-b\sum_{i=1}^{m}\alpha_i y^{(i)}+\sum_{i=1}^{m}\alpha_i$$

$$=-\frac{1}{2}\left(\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}\right)^T\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}-b\sum_{i=1}^{m}\alpha_i y^{(i)}+\sum_{i=1}^{m}\alpha_i$$

$$=-\frac{1}{2}\sum_{i=1}^{m}\alpha_i y^{(i)}\left(x^{(i)}\right)^T\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}-b\sum_{i=1}^{m}\alpha_i y^{(i)}+\sum_{i=1}^{m}\alpha_i$$

$$=-\frac{1}{2}\sum_{i,j=1}^{m}\alpha_i y^{(i)}\left(x^{(i)}\right)^T\alpha_j y^{(j)}x^{(j)}-b\sum_{i=1}^{m}\alpha_i y^{(i)}+\sum_{i=1}^{m}\alpha_i$$

深入SVM

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α_i（求出了α_i便能求出w，和b，由此可见，上文提出来的核心问题：分类函数$f(x)=w^Tx+b$也就可以轻而易举的求出来了）。

（2）、求对α*的极大*，即是关于对偶问题的最优化问题。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α。从上面的式子得到：

$$\max_{\alpha} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j$$
$$s.t., \alpha_i \geq 0, i=1,\ldots,n$$
$$\sum_{i=1}^{n}\alpha_i y_i = 0$$

这样，求出了α_i，根据$w=\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}$，即可求出w，然后通过$b^*=-\frac{\max_{i:y^{(i)}=-1} w^{*T}x^{(i)}+\min_{i:y^{(i)}=1} w^{*T}x^{(i)}}{2}$，即可求出b，最终得出分离超平面和分类决策函数。

UIBE 对外经济贸易大学 UNIVERSITY OF INTERNATIONAL BUSINESS AND ECONOMICS

深入SVM

（3）在求得L(w, b, a) 关于 w 和 b 最小化，以及对α的极大之后，最后一步则可以利用SMO算法求解对偶问题中的拉格朗日乘子α。

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$
$$s.t., \alpha_i \geq 0, i = 1, \ldots, n$$
$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

上述式子要解决的是在参数$\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$上求最大值W的问题，至于$x^{(i)}$和$y^{(i)}$都是已知数。

到目前为止，我们的 SVM 还比较弱，只能处理线性的情况，下面我们将引入核函数，进而推广到非线性分类问题。

深入SVM

线性不可分的情况

首先是关于 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 $f(x)=w^Tx+b$ 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

$w=\sum_{i=1}^{n}\alpha_i y_i x_i$

因此**分类函数**为：

$$f(x)=\left(\sum_{i=1}^{n}\alpha_i y_i x_i\right)^T x+b$$
$$=\sum_{i=1}^{n}\alpha_i y_i\langle x_i,x\rangle+b$$

深入SVM

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（$\langle \cdot, \cdot \rangle$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。

回忆一下通过 Lagrange multiplier得到的目标函数：

$$\max_{\alpha_i \geq 0} \mathcal{L}(w, b, \alpha) = \max_{\alpha_i \geq 0} \frac{1}{2} \|w\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i (w^T x_i + b) - 1 \right)$$

深入SVM

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α_i又是非负的，为了满足最大化，α_i必须等于 0 。这也就是这些非 Supporting Vector 的点的局限性。

至此，我们便得到了一个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")。

深入SVM

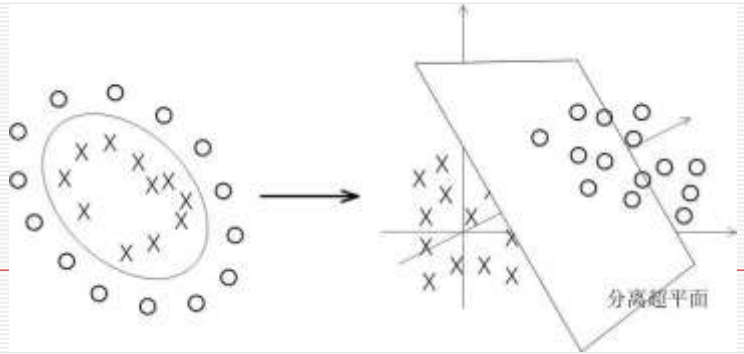
核函数Kernel

特征空间的隐式映射：核函数

事实上，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。对于非线性的数据SVM如何处理？

对于非线性的情况，SVM 的处理方法是选择一个核函数 $\kappa(\cdot,\cdot)$ ，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。

具体来说，在线性不可分的情况下，支持向量机首先在低维空间中完成计算，然后通过核函数将输入空间映射到高维特征空间，最终在高维特征空间中构造出最优分离超平面，从而把平面上本身不好分的非线性数据分开。如图所示，一堆数据在二维空间无法划分，从而映射到三维空间里划分：



深入SVM

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：

$$f(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi_i(\mathbf{x}) + b,$$

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。
而由于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：

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i y_i \langle \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}) \rangle + b.$$

深入SVM

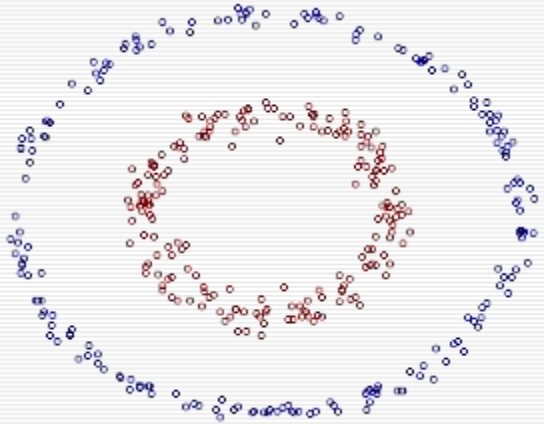
如果有一种方式可以**在特征空间中直接计算内积〈φ(x*i* · φ(x)〉**，就像在原始输入点的函数中一样，就有可能将两个步骤融合到一起建立一个非线性的学习器，**这样直接计算法的方法称为核函数方法。**

核是一个函数K，对所有x，z，满足 $K(\mathbf{x},\mathbf{z}) = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle,$ ，这里φ是从X到内积特征空间F的映射。

深入SVM

核函数：如何处理非线性数据

来看个核函数的例子。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。



事实上，图所述的这个数据集，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 X_1 和 X_2 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：

$$a_1X_1 + a_2X_1^2 + a_3X_2 + a_4X_2^2 + a_5X_1X_2 + a_6 = 0$$

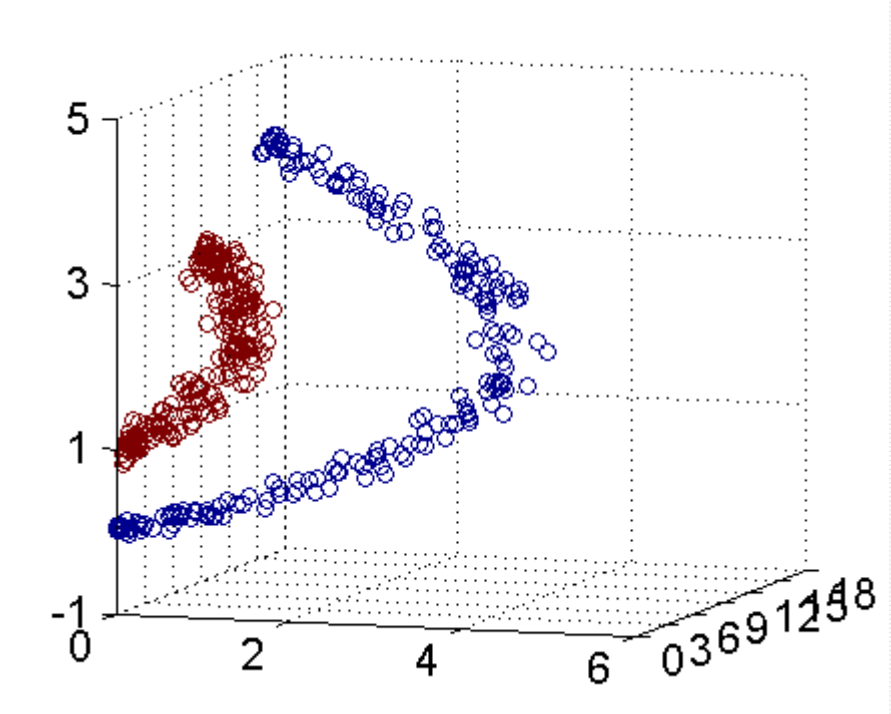
深入SVM

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 $Z_1 = X_1$，$Z_2 = X_1^2$，$Z_3 = X_2$，$Z_4 = X_2^2$，$Z_5 = X_1 X_2$，那么显然，上面的方程在新的坐标系下可以写作：

$$\sum_{i=1}^{5} a_i Z_i + a_6 = 0$$

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 $\phi : R_2 \to R_5$，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。

深入SVM



深入SVM

核函数相当于把原来的分类函数：

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle x_i, x \rangle + b$$

映射成：

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle \phi(x_i), \phi(x) \rangle + b$$

而其中的α可以通过求解如下 dual 问题而得到的：

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle$$
$$s.t., \alpha_i \geq 0, i = 1, \ldots, n$$
$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

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，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。不过事实上好像并没有这么简单。
细想一下，上面的方法是不是有问题？

深入SVM

- 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（一阶、二阶和三阶的组合），那么我们会得到：3(一次)+3(二次交叉)+3(平方)+3(立方)+1(x1*x2*x3)+2*3(交叉，一个一次一个二次，类似x1*x2^2) = 19维的新空间，这个数目是呈指数级爆炸性增长的，从而势必这给 $\phi(\cdot)$ 的计算带来非常大的困难，而且如果遇到无穷维的情况，就根本无从计算了。

不妨还是从最开始的简单例子出发，设两个向量 $x_1 = (\eta_1, \eta_2)^T$ 和 $x_2 = (\xi_1, \xi_2)^T$，而 $\phi(\cdot)$ 即是到前面说的五维空间的映射，因此映射过后的内积为：

$$\langle \phi(x_1), \phi(x_2) \rangle = \eta_1\xi_1 + \eta_1^2\xi_1^2 + \eta_2\xi_2 + \eta_2^2\xi_2^2 + \eta_1\eta_2\xi_1\xi_2$$

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，这里说的前面所述的映射方式，回顾之前的映射规则，再看第一个推导，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。

深入SVM

$$(\langle x_1, x_2 \rangle + 1)^2 = 2\eta_1\xi_1 + \eta_1^2\xi_1^2 + 2\eta_2\xi_2 + \eta_2^2\xi_2^2 + 2\eta_1\eta_2\xi_1\xi_2 + 1$$

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

$$\varphi(X_1, X_2) = (\sqrt{2}X_1, X_1^2, \sqrt{2}X_2, X_2^2, \sqrt{2}X_1X_2, 1)^T$$

之后的内积 $\langle \varphi(x_1), \varphi(x_2) \rangle$ 的结果是相等的，那么区别在于什么地方呢？

1. 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；
2. 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。**

深入SVM

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：

$$\kappa(x_1, x_2) = (\langle x_1, x_2 \rangle + 1)^2$$

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：

$$\sum_{i=1}^{n} \alpha_i y_i \kappa(x_i, x) + b$$

深入SVM

其中 α 由如下 dual 问题计算而得:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j)$$

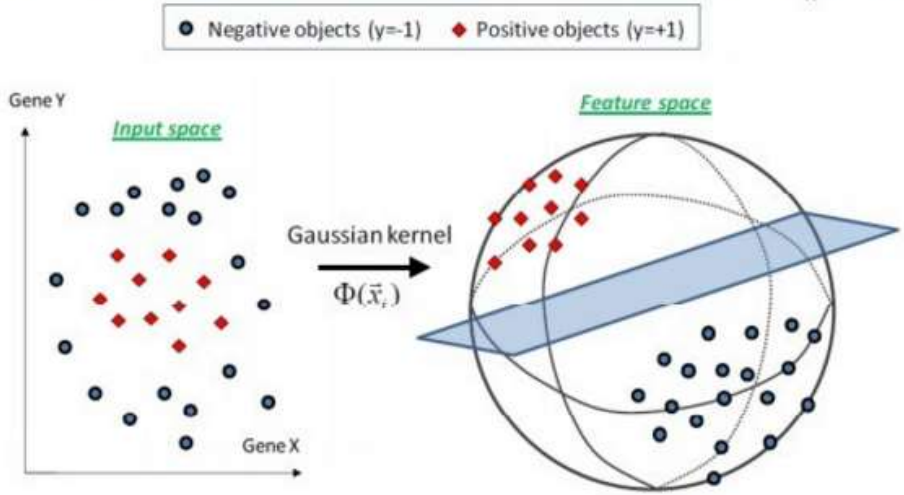
$$s.t., \alpha_i \geq 0, i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

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。

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 $\binom{m+d}{d}$，其中 m 是原始空间的维度。
- 高斯核 $\kappa(x_1, x_2) = \exp\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right)$，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 σ 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 σ 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 σ，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 $\kappa(x_1, x_2) = \langle x_1, x_2 \rangle$，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)。

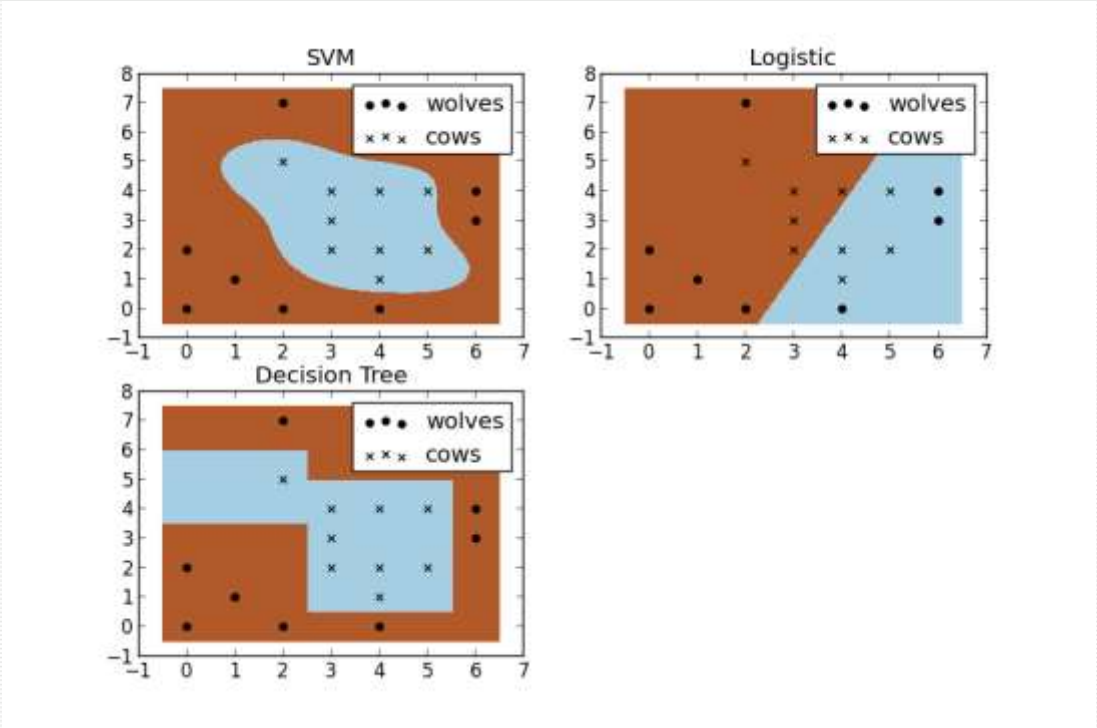
深入SVM

核函数的本质

- 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(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。怎么办？
- 核函数的价值在于它虽然也是将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。

假设现在有一个农场主，圈养了一批牛群，但为预防狼群袭击牛群，需要搭建一个篱笆来把牛群围起来。但是篱笆应该建在哪里呢？可能需要依据牛群和狼群的位置建立一个“分类器”，比较下图这几种不同的分类器，我们可以看到SVM完成了一个很完美的解决方案。

深入SVM

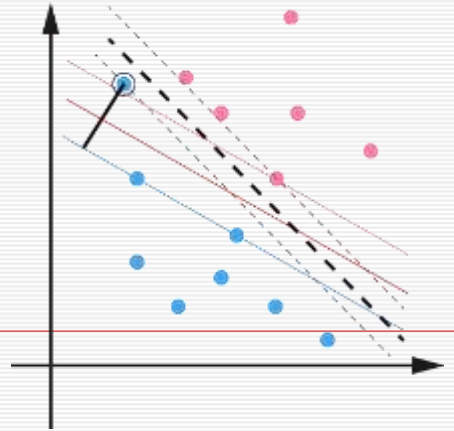

SVM
wolves
cows
Logistic
wolves
cows
Decision Tree
wolves
cows

深入SVM

使用松弛变量处理 outliers 方法

假定数据是线性可分的，亦即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 outlier ，在原来的 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：



深入SVM

使用松弛变量处理 outliers 方法

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 蓝色间隔边界上，而不会使得超平面发生变形了。

原来的约束条件为： $y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, n$

现在考虑到outlier问题，约束条件变成了： $y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n$

其中 $\xi_i \geq 0$ 称为松弛变量 (slack variable)，对应数据点 x_i 允许偏离的 functional margin 的量。当然，如果我们运行 ξ_i 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 ξ_i 的总和也要最小：

$$\min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i$$

深入SVM

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

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$s.t., y_i(w^T x_i + b) \geq 1 - \xi_i, i = 1, \ldots, n$$
$$\xi_i \geq 0, i = 1, \ldots, n$$

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：

$$\mathcal{L}(w, b, \xi, \alpha, r) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left(y_i(w^T x_i + b) - 1 + \xi_i\right) - \sum_{i=1}^{n} r_i\xi_i$$

深入SVM

分析方法和前面一样，转换为另一个问题之后，我们先让 $\mathcal{L}$ 针对 w、b 和 ξ 最小化：

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \Rightarrow C - \alpha_i - r_i = 0, \quad i = 1, \ldots, n$$

将 w 带回 $\mathcal{L}$ 并化简，得到和原来一样的目标函数：

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

深入SVM

不过，由于我们得到$C-\alpha_i-r_i=0$而又有$r_i\geq 0$（作为 Lagrange multiplier 的条件），因此有$\alpha_i\leq C$，所以整个 dual 问题现在写作：

$$\max_{\alpha}\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j\langle x_i,x_j\rangle$$
$$s.t.,0\leq\alpha_i\leq C,i=1,\ldots,n$$
$$\sum_{i=1}^{n}\alpha_i y_i=0$$

深入SVM

Primal formulation:

Minimize $\frac{1}{2}\sum_{i=1}^{n} w_i^2 + C\sum_{i=1}^{N}\xi_i$ subject to $y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1 - \xi_i$ for $i = 1,\ldots,N$

Objective function　　Constraints

Dual formulation:

Maximize $\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j \vec{x}_i\cdot\vec{x}_j$ subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N}\alpha_i y_i = 0$

for $i = 1,\ldots,N$.　　Objective function　　Constraints

可以看到唯一的区别就是现在 dual variable $\boldsymbol{\alpha}$ 多了一个上限 $\boldsymbol{C}$ 。而 Kernel 化的非线性形式也是一样的，只要把 $\langle x_i, x_j \rangle$ 换成 $\kappa(x_i, x_j)$ 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。

深入SVM

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，化为对拉格朗日乘子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a的求解可以用一种快速学习算法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。

SMO算法

前面我们提到了求解对偶问题的序列最小最优化SMO算法，但并未提到其具体解法。首先看下最后悬而未决的问题：

$$\max_{\alpha} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle$$
$$s.t., 0 \le \alpha_i \le C, i = 1, \ldots, n$$
$$\sum_{i=1}^{n}\alpha_i y_i = 0$$

等价于

$$\min_{\alpha} \Psi(\vec{\alpha}) = \min_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j K(\vec{x}_i, \vec{x}_j)\alpha_i\alpha_j - \sum_{i=1}^{N}\alpha_i$$
$$0 \le \alpha_i \le C, \forall i,$$
$$\sum_{i=1}^{N} y_i\alpha_i = 0.$$

SMO算法

SMO算法的推导

1998年，Microsoft Research的John C. Platt在论文《Sequential Minimal Optimization：A Fast Algorithm for Training Support Vector Machines》中提出针对上述问题的解法：SMO算法，它很快便成为最快的二次规划优化算法，特别是在针对线性SVM和数据稀疏时性能更优。

咱们首先来定义特征到结果的输出函数：

$$u = \vec{w} \cdot \vec{x} - b$$

注：这个u与我们之前定义的 $f(x) = w^T x + b$ 实质是一样的。

接着，重新定义下咱们原始的优化问题，权当重新回顾，如下：

$$\min_{w,b} \frac{1}{2} \|\vec{w}\|^2 \text{ subject to } y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, \forall i$$

SMO算法

求导得到：

$$\vec{w}=\sum_{i=1}^{N} y_i\alpha_i\vec{x}_i,\quad b=\vec{w}\cdot\vec{x}_k-y_k \text{ for some } \alpha_k>0$$

代入$u=\vec{w}\cdot\vec{x}-b$中，可得 $u=\sum_{j=1}^{N} y_j\alpha_j K(\vec{x}_j,\vec{x})-b$ 。

通过引入拉格朗日乘子转换为对偶问题后，得：

$$\min_{\alpha}\Psi(\vec{\alpha})=\min_{\alpha}\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j(\vec{x}_i\cdot\vec{x}_j)\alpha_i\alpha_j-\sum_{i=1}^{N}\alpha_i$$

s.t: $\alpha_i\geq 0,\forall i,$

且 $\sum_{i=1}^{N} y_i\alpha_i=0.$

注：这里得到的min函数与我们之前的max函数实质也是一样，因为把符号变下，即由min转化为max的问题，且yi也与之前的$y^{(i)}$等价，yj亦如此。

SMO算法

经过加入松弛变量后，模型修改为：

$$\min_{w,b,\xi} \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{N}\xi_i \quad \text{subject to } y_i(\vec{w}\cdot\vec{x}_i - b) \geq 1-\xi_i, \forall i$$

$$0 \leq \alpha_i \leq C, \forall i$$

从而最终我们的问题变为：

$$\min_{\vec{\alpha}} \Psi(\vec{\alpha}) = \min_{\vec{\alpha}} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j K(\vec{x}_i, \vec{x}_j)\alpha_i\alpha_j - \sum_{i=1}^{N}\alpha_i$$
$$0 \leq \alpha_i \leq C, \forall i,$$
$$\sum_{i=1}^{N} y_i\alpha_i = 0.$$

下面要解决的问题是：在$\alpha_i = \{\alpha_1, \alpha_2, \alpha_3, ..., \alpha_n\}$上求上述目标函数的最小值。为了求解这些乘子，每次从中任意抽取两个乘子α_1和α_2，然后固定α_1和α_2以外的其它乘子$\{\alpha_3, \alpha_4, ..., \alpha_n\}$，使得目标函数只是关于$\alpha_1$和$\alpha_2$的函数。这样，不断的从一堆乘子中任意抽取两个求解，不断的迭代求解子问题，最终达到求解原问题的目的。

SMO算法

而原对偶问题的子问题的目标函数可以表达为：

$$\Psi = \frac{1}{2}K_{11}\alpha_1^2 + \frac{1}{2}K_{22}\alpha_2^2 + sK_{12}\alpha_1\alpha_2 + y_1\alpha_1 v_1 + y_2\alpha_2 v_2 - \alpha_1 - \alpha_2 + \Psi_{\text{constant}}$$

其中

$$K_{ij} = K(\vec{x}_i, \vec{x}_j),$$
$$v_i = \sum_{j=3}^{N} y_j \alpha_j^* K_{ij} = u_i + b^* - y_1\alpha_1^* K_{1i} - y_2\alpha_2^* K_{2i}$$

为了解决这个子问题，首要问题便是每次如何选取α_1和α_2。实际上，其中一个乘子是违法KKT条件最严重的，另外一个乘子则由另一个约束条件选取。

根据KKT条件可以得出目标函数中α_i取值的意义：

$$\alpha_i = 0 \Leftrightarrow y_i u_i \geq 1,$$
$$0 < \alpha_i < C \Leftrightarrow y_i u_i = 1,$$
$$\alpha_i = C \Leftrightarrow y_i u_i \leq 1.$$

SMO算法

这里的α_i还是拉格朗日乘子：

1. 对于第1种情况，表明α_i是正常分类，在间隔边界内部（我们知道正确分类的点$y_i * f(x_i) \geq 0$）；
2. 对于第2种情况，表明了α_i是支持向量，在间隔边界上；
3. 对于第3种情况，表明了α_i是在两条间隔边界之间；

而最优解需要满足KKT条件，即上述3个条件都得满足，以下几种情况出现将会出现不满足：

- $y_i u_i$ <=1但是α_i<C则是不满足的，而原本α_i=C
- $y_i u_i$ >=1但是α_i>0则是不满足的，而原本α_i=0
- $y_i u_i$ =1但是α_i=0或者α_i=C则表明不满足的，而原本应该是0<α_i<C

也就是说，如果存在不满足KKT条件的α_i，那么需要更新这些α_i，这是第一个约束条件。此外，更新的同时还要受到第二个约束条件的限制，即

$$\sum_{i=1}^{N} y_i \alpha_i = 0.$$

SMO算法

因此，如果假设选择的两个乘子α_1和α_2，它们在更新之前分别是α_1^{old}、α_2^{old}，更新之后分别是α_1^{new}、α_2^{new}，那么更新前后的值需要满足以下等式才能保证和为0的约束：

$$\alpha_1^{new} y_1 + \alpha_2^{new} y_2 = \alpha_1^{old} y_1 + \alpha_2^{old} y_2 = \zeta$$

其中，ζ是常数。

两个因子不好同时求解，所以可先求第二个乘子α_2的解（α_2^{new}），得到α_2的解（α_2^{new}）之后，再用α_2的解（α_2^{new}）表示α_1的解（α_1^{new}）。

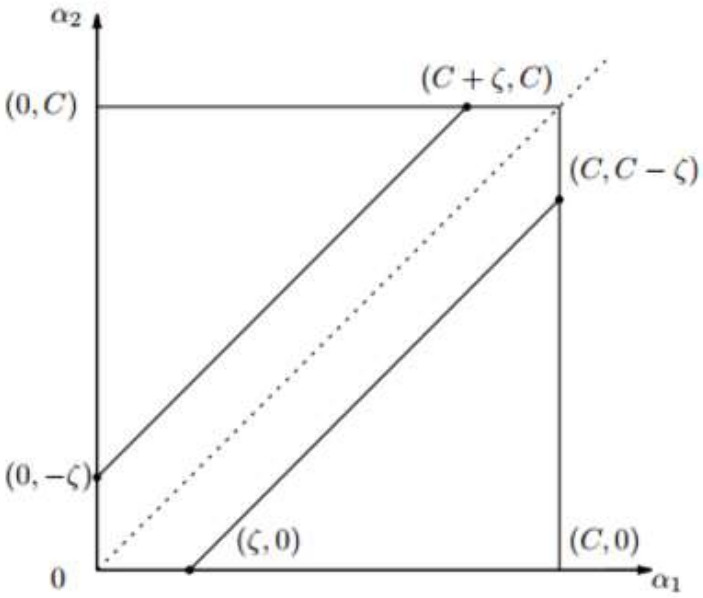
为了求解α_2^{new}，得先确定α_2^{new}的取值范围。假设它的上下边界分别为H和L，那么有：

$$L \le \alpha_2^{new} \le H$$

接下来，综合$0 \le \alpha_i \le C, \quad i = 1, \ldots, n$和$\alpha_1^{new} y_1 + \alpha_2^{new} y_2 = \alpha_1^{old} y_1 + \alpha_2^{old} y_2 = \zeta$这两个约束条件，求取$\alpha_2^{new}$的取值范围。

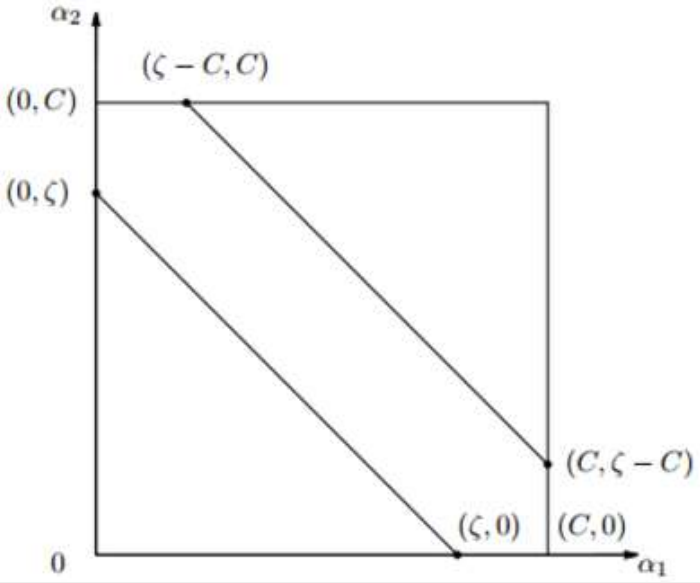
SMO算法

当y1 != y2时，根据$\alpha_1^{new} y_1 + \alpha_2^{new} y_2 = \alpha_1^{old} y_1 + \alpha_2^{old} y_2 = \zeta$可得$\alpha_1^{old} - \alpha_2^{old} = \zeta$，所以有$L = max(0, -\zeta)$，$H = min(C, C - \zeta)$，如下图所示：



SMO算法

当y1 = y2时，同样根据$\alpha_1^{new} y_1 + \alpha_2^{new} y_2 = \alpha_1^{old} y_1 + \alpha_2^{old} y_2 = \zeta$可得：$\alpha_1^{old} + \alpha_2^{old} = \zeta$，所以有$L = max(0, \zeta - C)$，$H = min(C, \zeta)$，如下图所示：



SMO算法

如此，根据y1和y2异号或同号，可得出α_2^{new}的上下界分别为：

$$\begin{cases} L = \max(0, \alpha_2^{old} - \alpha_1^{old}), H = \min(C, C + \alpha_2^{old} - \alpha_1^{old}) & \text{if } y_1 \neq y_2 \\ L = \max(0, \alpha_2^{old} + \alpha_1^{old} - C), H = \min(C, \alpha_2^{old} + \alpha_1^{old}) & \text{if } y_1 = y_2 \end{cases}$$

回顾下第二个约束条件$\alpha_1^{new} y_1 + \alpha_2^{new} y_2 = \alpha_1^{old} y_1 + \alpha_2^{old} y_2 = \zeta$，令上式两边乘以y1，可得

$$\alpha_1 + s\alpha_2 = \alpha_1^* + s\alpha_2^* = w.$$

其中，$w = -y_1 \sum_{i=3}^{n} y_i \alpha_i^*$。

因此α_1可以用α_2表示，$\alpha_1 = w - s * \alpha_2$，从而把子问题的目标函数转换为只含$\alpha_2$的问题：

$$\begin{aligned} \Psi = & \tfrac{1}{2} K_{11}(w - s\alpha_2)^2 + \tfrac{1}{2} K_{22}\alpha_2^2 + sK_{12}(w - s\alpha_2)\alpha_2 \\ & + y_1(w - s\alpha_2)v_1 - w + s\alpha_2 + y_2\alpha_2 v_2 - \alpha_2 + \Psi_{\text{constant}} \end{aligned}$$

对外经济贸易大学 OF INTERNATIONAL BUSINESS AND ECONOMICS

SMO算法

对α_2求导，可得

$$\frac{d\Psi}{d\alpha_2} = -sK_{11}(w - s\alpha_2) + K_{22}\alpha_2 - K_{12}\alpha_2 + sK_{12}(w - s\alpha_2) - y_2 v_1 + s + y_2 v_2 - 1 = 0$$

化简下：

$$\alpha_2(K_{11} + K_{22} - 2K_{12}) = s(K_{11} - K_{12})w + y_2(v_1 - v_2) + 1 - s$$

然后将$s = y_1 * y_2$、$\alpha_1 + s\alpha_2 = \alpha_1^* + s\alpha_2^* = w$、和 $K_{ij} = K(\vec{x}_i, \vec{x}_j)$，$v_i = \sum_{j=3}^{N} y_j \alpha_j^* K_{ij} = u_i + b^* - y_1\alpha_1^* K_{1i} - y_2\alpha_2^* K_{2i}$ 代入上式可得：

$$\alpha_2^{new,unc}(K_{11} + K_{22} - 2K_{12}) = \alpha_2^{old}(K_{11} + K_{22} - 2K_{12}) + y_2(u_1 - u_2 + y_2 - y_1)$$

SMO算法

令$E_i = u_i - y_i$（表示预测值与真实值之差），$\eta = K(\vec{x}_1, \vec{x}_1) + K(\vec{x}_2, \vec{x}_2) - 2K(\vec{x}_1, \vec{x}_2)$，然后上式两边同时除以$\eta$，得到一个关于单变量$\alpha_2$的解：

$$\alpha_2^{new,unc} = \alpha_2^{old} + \frac{y_2(E_1 - E_2)}{\eta}$$

这个解没有考虑其约束条件$0 \le \alpha_2 \le C$，即是未经剪辑时的解。

然后考虑约束$0 \le \alpha_2 \le C$可得到经过剪辑后的α_2^{new}的解析解为：

$$\alpha_2^{new} = \begin{cases} H, & \alpha_2^{new,unc} > H \\ \alpha_2^{new,unc}, & L \le \alpha_2^{new,unc} \le H \\ L, & \alpha_2^{new,unc} < L \end{cases}$$

SMO算法

求出了后α_2^{new}，便可以求出α_1^{new}，得$\alpha_1^{new} = \alpha_1^{old} + y_1 y_2(\alpha_2^{old} - \alpha_2^{new})$。

那么如何选择乘子α_1和α_2呢?

- 对于α_1，即第一个乘子，可以通过刚刚说的那3种不满足KKT的条件来找;
- 而对于第二个乘子α_2可以寻找满足条件：$max|E_i - E_j|$的乘子。

而b在满足下述条件:

$$b = \begin{cases} b_1 & \text{if } 0 < \alpha_1^{new} < C \\ b_2 & \text{if } 0 < \alpha_2^{new} < C \\ (b_1 + b_2)/2 & \text{otherwise} \end{cases}$$

下更新b:

$$b_1^{new} = b^{old} - E_1 - y_1(\alpha_1^{new} - \alpha_1^{old})k(x_1, x_1) - y_2(\alpha_2^{new} - \alpha_2^{old})k(x_1, x_2)$$

$$b_2^{new} = b^{old} - E_2 - y_1(\alpha_1^{new} - \alpha_1^{old})k(x_1, x_2) - y_2(\alpha_2^{new} - \alpha_2^{old})k(x_2, x_2)$$

且每次更新完两个乘子的优化后，都需要再重新计算b，及对应的Ei值。

最后更新所有α_i，y和b，这样模型就出来了，从而即可求出咱们开头提出的分类函数:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle x_i, x \rangle + b$$

SMO算法

SMO算法的步骤

1. 第一步选取一对α_i和α_j，选取方法使用启发式方法；

2. 第二步，固定除α_i和α_j之外的其他参数，确定W极值条件下的α_i，α_j由α_i表示。

假定在某一次迭代中，需要更新x_1，x_2对应的拉格朗日乘子α_1，α_2，那么这个小规模的二次规划问题写为：

$$L_s = \max_{\alpha}\{(\alpha_1+\alpha_2)+\sum_{i=3}^{n}\alpha_i-\frac{1}{2}\left\|\alpha_1 y_1\phi(\boldsymbol{x_1})+\alpha_1 y_1\phi(\boldsymbol{x_1})+\sum_{i=3}^{n}\alpha_i y_i\phi(\boldsymbol{x_i})\right\|^2\}$$

$$s.t.\,\alpha_1 y_1+\alpha_2 y_2=-\sum_{i=3}^{n}\alpha_i y_i\,,0<\alpha_i<C,\forall i$$

SMO算法

更新拉格朗日乘子α_1, α_2

- 步骤1：计算上下界L和H
  - $L = \max(0, \alpha_2^{old} - \alpha_1^{old}), H = \min(C, C + \alpha_2^{old} - \alpha_1^{old}), if\ y_1 \neq y_2$
  - $L = \max(0, \alpha_2^{old} + \alpha_1^{old} - C), H = \min(C, \alpha_2^{old} + \alpha_1^{old}), if\ y_1 = y_2$
- 步骤2：计算L_s的二阶导数
  - $\eta = 2\phi(\boldsymbol{x_1})^t\phi(\boldsymbol{x_2}) - \phi(\boldsymbol{x_1})^t\phi(\boldsymbol{x_1}) - \phi(\boldsymbol{x_2})^t\phi(\boldsymbol{x_2})$
- 步骤3：更新L_s

$$\alpha_2^{new} = \alpha_2^{old} - \frac{y_2(e_1 - e_2)}{\eta}$$

$$e_i = g^{old}(\boldsymbol{x_i}) - y_i$$

对外经济贸易大学
UNIVERSITY OF INTERNATIONAL BUSINESS AND ECONOMICS

SMO算法

– 步骤4：计算变量α_2

$$\alpha^{temp} = \begin{cases} H, & \text{if} \alpha_2^{new} \geq H \\ \alpha_2^{new}, & \text{if } L \leq \alpha_2^{new} \leq H \\ L, & \text{if } \alpha_2^{new} \leq L \end{cases}$$

– 步骤5：更新α_1

$$\alpha_1^{new} = \alpha_1^{old} + y_1 y_2 (\alpha_2^{old} - \alpha^{temp})$$

SMO算法

知道了如何更新乘子，那么选取哪些乘子进行更新呢？具体选择方法有以下两个步骤：

1. 步骤1：先"扫描"所有乘子，把第一个违反KKT条件的作为更新对象，令为a1；
2. 步骤2：在所有不违反KKT条件的乘子中，选择使|E1 −E2|最大的a2进行更新，使得能最大限度增大目标函数的值（类似于梯度下降. 此外$E_i = u_i - y_i$，而$u = \vec{w} \cdot \vec{x} - b$，求出来的E代表函数ui对输入xi的预测值与真实输出类标记yi之差）。

最后，每次更新完两个乘子的优化后，都需要再重新计算b，及对应的Ei值。

综上，**SMO**算法的基本思想是将**Vapnik**在**1982**年提出的**Chunking**方法推到极致，**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，所以该算法表现出较好的快速收敛性，且不需要存储核矩阵，也没有矩阵运算。

谢谢大家