一种训练支撑向量机的改进贯序最小优化算法*

孙 剑, 郑南宁, 张志华

(西安交通大学 人工智能与机器人研究所,陕西 西安 710049)
E-mail: sj@aiar.xjtu.edu.cn; nnzheng@xjtu.edu.cn
http://www.aiar.xjtu.edu.cn

摘要: 对于大规模问题,分解方法是训练支撑向量机主要的一类方法.在很多分类问题中,有相当比例的支撑向量对应的拉格朗日乘子达到惩罚上界,而且在训练过程中到达上界的拉格朗日乘子变化平稳.利用这一统计特性,提出了一种有效的缓存策略来加速这类分解方法,并将其具体应用于 Platt 的贯序最小优化(sequential minimization optimization,简称 SMO)算法中.实验结果表明,改进后的 SMO 算法的速度是原有算法训练的 2~3 倍.

关 键 词: 支撑向量机;模式分类;二次规划;缓存策略;贯序最小优化算法
中图法分类号: TP181 文献标识码: A

支撑向量机(support vector machine,简称 SVM)源于线性可分问题的最优分类超平面,即以最大间隔将数据分开.对于线性不可分问题,通过选择某种非线性映射,将输入空间映射到一个维数更高的特征空间,在这个空间构造最优分类超平面.SVM 是基于统计学习理论[1,2]中的结构风险最小(structure risk minimization,简称 SRM)归纳原理,能有效地避免经典学习方法中过学习、维数灾难、局部极小等非常棘手的问题,在小样本条件下仍然具有良好的范化能力.有关 SVM 的详细论述可以参阅文献[3].

SVM 在解决分类、回归和密度函数估计等机器学习问题方面获得了非常好的结果,并且已经应用于手写体字符识别、人脸检测、文本、语音分类等实际问题.但是,训练时间长仍是 SVM 目前的一大缺点.对于大规模问题(样本大于 104),目前的 SVM 训练算法速度都比较慢,尤其对于 M 类问题,最少需要训练 M–1 个分类器(按一对多方式),增加新类别时,又需要全部重新训练.所以,在 SVM 的研究中,如何提高训练速度,减少训练时间,建立实用的学习算法,仍是一个亟待解决的问题.

给定训练样本$\{(x_i,y_i)|i=1,\ldots,N;x_i\in R^d,y_i\in\{-1,+1\}\}$,SVM 实现下面的决策函数:

$$f(x)=\mathrm{sign}\left(\sum_{i=1}^{N}y_i\alpha_i k(x_i,x)-b\right), \tag{1}$$

其中 α_i 是每个样本对应的拉格朗日乘子, b 是阈值.

首先通过选择满足 Mercer 条件的核函数 $k(x,y)$,确定输入空间到特征空间的某种映射 $\phi(x)$, $k(x_i,x_j)$ 等于该映射 $\phi(x)$ 构成的特征空间上的内积,即 $k(x_i,x_j)=(\phi(x_i)\cdot\phi(x_j))$,这样甚至不需要知道映射 $\phi(x)$ 的具体形式,也能在输入空间简单地计算特征空间上的内积,有关核空间的理论可以参阅文献[2].训练一个 SVM 相当于解一个凸二次规划(quadratic programming,简称 QP)问题:

$$\max_{\alpha} L_D=\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1,j=1}^{N}\alpha_i Q(i,j)\alpha_j\ , \tag{2}$$

* 收稿日期: 2000-12-07; 修改日期: 2001-11-05
基金项目: 国家自然科学基金资助项目(60175006;60024301);国家创新研究群体科学基金项目(60024301)
作者简介: 孙剑(1976－),男,山东单县人,博士生,主要研究领域为计算机视觉,机器学习;郑南宁(1952－),男,江苏南京人,博士,教授,博士生导师,中国科学院院士,主要研究领域为模式识别与智能系统;张志华(1971－),男,湖北阳新人,博士生,主要研究领域为模式识别与智能系统.

其中 $Q(i,j)=y_iy_jk(x_i,x_j)$, $0\le\alpha_i\le C$, $\sum_{i=1}^{N}\alpha_iy_i=0$. Q 为 $N\times N$ 矩阵, $k(x_i,x_j)$ 是核函数.

对于小规模的 QP 问题,经典最优化算法,如牛顿法、拟牛顿法等都可以较好地求解,但当训练集很大,特别是支撑向量数目也很大时,多数算法需要的内存与 Q 矩阵的大小成比例.例如,假设 Q 矩阵的元素用四字节的浮点数表示,N=20000,需要 1.5GBytes 内存,N=50000,需要 9.5TBytes 内存,解 QP 问题的经典方法不再可行.分解算法(将大的 QP 问题分解成一系列小的 QP 子问题进行迭代求解)是目前有效解决大规模问题的主要方法.

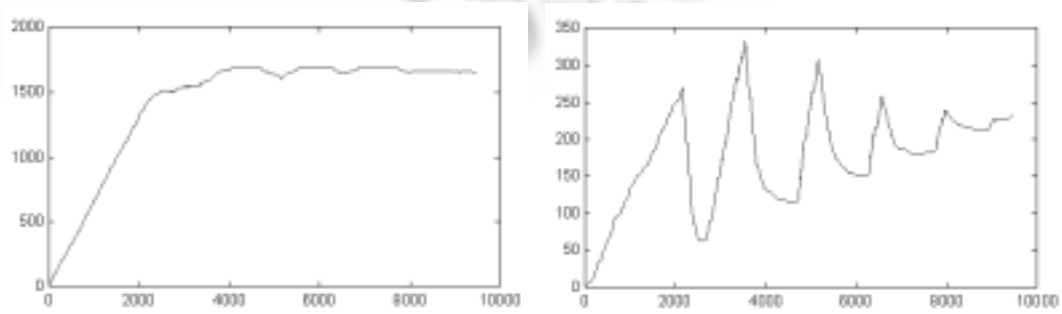
Chunking[4]算法是首先被提出来的一种分解算法:从训练样本中任意选择一个小的子集,求解这个子集的最优,保留这个子集的支撑向量,在剩余的样本中启发式地加入若干样本构成新的子集,再求解新子集的最优,反复迭代直至收敛.但当问题的支撑向量数也很大时,子问题的求解也很困难.Osuna[5]提出的工作集(working set)方法克服了上述限制:选择一个固定大小的工作集 B,求解集 B 上的 QP 问题,考察所有不满足 Karush-Kuhn-Tucker(KKT)条件的样本,启发式地选择一些样本与集 B 中 $\alpha_i=0$ 的样本交换,反复迭代直到所有的样本都满足 KKT 条件.每一个 QP 子问题仍然使用迭代数值优化算法求解.Osuna 证明每次迭代使目标函数单调递增,因为目标函数上有界,所以经过有限次迭代后算法将收敛并得到最优解,但最优解在一般情况下不惟一.Platt[6]的贯序最小优化(sequential minimization optimization,简称 SMO)算法是 Osuna 的算法中工作集的势等于 2 的特例,两变量的 QP 子问题的解可以求解析解,从而避免了使用迭代算法,而且 SMO 对于线性 SVM 和稀疏数据特别有效.Keerthi[7]指出了 SMO 算法中使用单一阈值的低效性,用双阈值方法改进了 Platt 原来的算法,进一步提高了 SMO 的训练速度.Chang[8]在理论上对分解算法的收敛性进行了证明.

在各种分解算法中,决策函数的计算量非常大,这是因为在选择工作集和检验停止条件时,通常依赖于 KKT 条件,需要计算每个样本的决策函数;另一方面,每次迭代的子优化计算也需要计算决策函数.如果能减小计算决策函数的代价,则可以降低算法整体的计算量.但简单地对每个决策函数建立缓存,每次优化后更新缓存的代价则可能抵消甚至超过缓存降低的代价.有效的缓存机制应当利用训练中的某种规律或特性.

记 $\mathrm{BSV}^t=\{(x_i,y_i)\mid\alpha_i^t=C\}$ 为第 t 次迭代后 $\alpha_i^t=C$ 的样本构成的集合,$\mathrm{Non\text{-}BSV}^t=\{(x_i,y_i)\mid 0<\alpha_i^t<C\}$ 为第 t 次优化后 $0<\alpha_i^t<C$ 的样本构成的集合(下文省略上标 t).记|BSV|是 BSV 的势,即 BSV 中元素的个数,|Non-BSV|是 Non-BSV 的势.这样,决策函数可以写成:

$$f(x)=\sum_{i\in\mathrm{Non-BSV}}\alpha_iy_ik(x_i,x)+\sum_{i\in\mathrm{BSV}}C\cdot y_ik(x_i,x)-b. \tag{3}$$

在大规模分类问题的训练中,我们注意到:在迭代初期,|BSV|和|Non-BSV|迅速增大;很快,|BSV|变化比较稳定,直到到达中止条件时,这些 BSV 成为支撑向量,而|Non-BSV|的变化较剧烈;很多样本一旦进入 BSV 集合后就不再变化,如图 1 所示,其中,水平坐标为训练的迭代次数.



(a) The plot of |BSV| in training phase (b) The plot of |Non-BSV| in training phase

(a) |BSV|在训练过程中的变化曲线 (b) |Non-BSV|在训练过程中的变化曲线

Fig.1 The plots of |BSV| and |Non-BSV| in training phase

图 1 |BSV|和|Non-BSV|在训练过程中的变化曲线

针对上述特性,本文对分解算法中决策函数的计算提出一种有效的缓存机制:当|BSV|的变化相对稳定后,

对每个样本建立一个缓存,建立式(3)右边的第 2 项;以后计算决策函数时只需计算式(3)右边的第 1 项;当 BSV 有变化时,对所有缓存更新.

本文第 1 节描述了 SMO 算法及其计算复杂度.第 2 节给出应用缓存机制的 SMO 改进算法,并分析了其计算复杂度.第 3 节进行实验评测,验证了改进算法的有效性.最后是小结.

1 SMO 算法与分析

Platt 的 SMO 算法是工作集的势等于 2 的分解算法,这样的两变量子问题最优解可以用解析的方法计算;同时,Platt 提出的启发式工作集选择策略也非常简单、有效.

SMO 的停止条件.对于凸 QP 问题,一个解是最优点的充要条件是 KKT 条件满足并且 Hess 矩阵半正定.经过简化的 KKT 条件为

$$\alpha_i = 0 \Rightarrow y_i f(x_i) \ge 1, \quad 0 < \alpha_i < C \Rightarrow y_i f(x_i) = 1, \quad \alpha_i = C \Rightarrow y_i f(x_i) \le 1,$$

当所有样本都满足上式时,算法停止,得到最优解.

两变量 QP 问题的解析解.令 $s = y_1 y_2$, $E_i = f(x_i) - y_i$,

$$L = \max(0, \alpha_2 + s\alpha_1 - (s+1)/2C), \quad H = \min(0, \alpha_2 + s\alpha_1 - (s-1)/2C),$$

$$\alpha_2^{\text{new}} = \alpha_2 + y_2(E_1 - E_2)/(k(x_1,x_1) + k(x_2,x_2) - 2k(x_1,x_2)),$$

$$\alpha_1^{\text{new}} = \alpha_1 + s(\alpha_2 - \alpha_2^{\text{new,clipped}}),$$

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

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

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

SMO 的启发式选择策略.它由两个嵌套的循环组成,外循环选择 α_1,内循环选择 α_2 (只考虑违反 KKT 条件的样本).

初始化 $\alpha_i = 0$, $b = 0$.

(1) 如果第 1 次进入外循环或者内循环的优化没有进展,α_1 在所有样本中贯序地选择;否则 α_1 在 Non-BSV 中贯序地选择;

(2) α_2 首先在 Non-BSV 中选择,使 $|E_1 - E_2|$ 最大;如果优化没有进展,α_2 在 Non-BSV 中贯序地选择;如果优化仍没有进展,α_2 在所有样本中贯序地选择;

(3) 当没有违反 KKT 条件的样本时,算法中止,得到最优解 α_i^*, b^*,否则返回(1).

SMO 中大量的计算集中在核函数 $k(x_i, x_j)$ 的计算上,任何能够加速对核函数 $k(x_i, x_j)$ 计算的方法都将减少训练时间.加速的方法有两类:一种是使用缓存机制,在缓存中保存一定已经计算的 $k(x_i, x_j)$ 的值,缓存的命中率越高,算法整体速度越快,但是分解算法对核函数存取的结构性很差,简单的最少最近使用策略(least recent usage,简称 LRU)对于 SMO 并不一定有效[9],合适的缓存策略是关键[10];另一种方法是减少不必要的核函数计算,因为 Non-BSV 集合的决策函数计算频繁,Platt 对 Non-BSV 集合使用了缓存,Non-BSV 样本的 E_i 在每次 α 优化后更新: $E_i^{\text{new}} = E_i^{\text{old}} + y_1\Delta\alpha_1 k_{1i} + y_2\Delta\alpha_2 k_{2i} + \Delta b$,下次更新前可以直接使用.而对于非 Non-BSV 的样本, E_i 要通过决策函数计算:

$$E_i = \sum_j^N \alpha_j y_j k(x_j, x_i) - b - y_i = \sum_j^{j\in\text{Non-BSV}} \alpha_j y_j k(x_j, x_i) + \sum_j^{j\in\text{BSV}} C \cdot y_j k(x_j, x_i) - b - y_i. \tag{4}$$

设在整个训练过程中,缓存的命中率为 $p(0 < p < 1)$,共需要计算决策函数 k 次,子优化的迭代次数为 t, t 次迭代中|Non-BSV|的平均值为 ta_nbsv, ka_nbsv 是 $k(1-p)$ 次计算决策函数时|Non-BSV|的平均值, ka_bsv 是 $k(1-p)$ 次计算决策函数时|BSV|的平均值.

以计算一次核函数的计算为单位,SMO 的计算复杂度为

$$O(k \cdot (1-p) \cdot (ka\_nbsv + ka\_bsv) + 2t \cdot ta\_nbsv). \tag{5}$$

前一项表示所有未击中缓存的核函数计算量,后一项表示每次优化后更新缓存的计算量.表 1 给出了实验中通过标准 SMO 算法训练后得到的各组数据.

2 缓存机制与改进的 SMO 算法

记 $A_+^t=\{(x_i,y_i)\mid \alpha_i^t=C,\alpha_i^t<C\}$, $A_-^t=\{(x_i,y_i)\mid \alpha_i^t<C,\alpha_i^{t-1}=C\}$,则第 t 次迭代后, $BSV^t=(BSV^{t-1}-A_-^t)\cup A_+^t$, (下文省略上标 t).很明显,在 BSV 变化缓慢的情况下,如果每次计算不在缓存中的决策函数时只计算变化的增量,就可以有效地减少核函数的计算,特别是在训练的后期,BSV 几乎不变化.

我们为每一个样本建立一个缓存 $cacheBSV_i, i=1,...,N$,当 BSV 开始缓慢变化时计算 $cacheBSV_i$ 的初始值;以后每当有 α_j 进入或离开 BSV 时,更新 $cacheBSV_i$:

$$cacheBSV_i^{\text{new}}=\begin{cases}cacheBSV_i^{\text{old}}+C\cdot y_j k(x_i,x_j) & \alpha_j\in A_+\\ cacheBSV_i^{\text{old}}-C\cdot y_j k(x_i,x_j) & \alpha_j\in A_-\end{cases}, \tag{6}$$

式(4)改写为

$$E_i=f(x_i)-y_i=\sum_{j\in \text{Non-BSV}}\alpha_j y_j k(x_j,x_k)+cacheBSV_i-y_i. \tag{7}$$

图 2 表示了我们的缓存机制.

记训练过程中 α_j 进入或离开 BSV 的总次数为 io_bsv ,改进的 SMO 算法其计算复杂度为

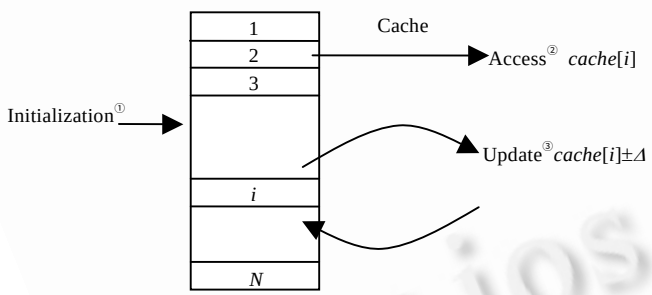
$$O(k\cdot(1-p)\cdot(ka\_nbsv+0)+2t\cdot ta\_nbsv+N\cdot io\_bsv). \tag{8}$$

式(8)的第 3 项是为更新 $cacheBSV_i$ 而增加的计算.比较式(5)和式(8),当满足

$$N\cdot io\_bsv<k\cdot(1-p)\cdot ka\_bsv \tag{9}$$

时,改进后的算法将减小计算量,即

$$\frac{k\cdot(1-p)}{N}-\frac{io\_bsv}{ka\_bsv}>0. \tag{10}$$



①初始化,②使用,③更新.

Fig 2 Buffering mechanism

图 2 缓存机制

对于 QP 优化问题, $k\cdot(1-p)$ 随着样本数 N 的增大而以 $N^q(q>1)$ 的速度增大时, ka_bsv/N 的比值依赖于问题和给定的样本, ka_bsv 随着样本数 N 的线性增大, io_bsv 的变化不但与 N 有关,而且与训练算法有关,精确的定量分析比较困难,但根据 BSV 的缓慢变化趋势和我们的实验计算, io_bsv 近似需随着样本数 N 线性增长.所以,样本数 N 越大,式(10)的左边越大,缓存机制发挥的作用也越大.

减少的计算量为 $(k\cdot(1-p)\cdot ka_bsv-N\cdot io_bsv)$次核函数的计算.随着问题规模的增大,特别是在最终的支撑向量中有相当一部分到达上界(在多数线性不可分的分类问题中都具有这样的性质),改进算法减少的计算量将成比例增加.

可以看到,缓存机制利用的是 BSV 变化平缓、缓存更新带来的计算小于重复计算 BSV 核函数这一特性,所以我们对算法的改进并不依赖于 SMO 算法,对于各种分解算法都可以加入如图 2 所示的缓存机制.例如,Osuna 将样本分为 B,N 两个集合,(1) 解 B 上的 QP 子问题,(2) 当存在 $j\in N, f(x_j)y_j<1, i\in B, \alpha_i=0$ 时,交换 x_i 与 x_j 在 B 上的 QP 子问题,反复迭代.在其算法中加入我们的缓存机制将是非常直观的.

3 实验与讨论

为便于比较,我们同样使用 Platt 等人使用的大规模分类任务:Adult Data(家庭收入预测)和 MNIST(手写数

字识别库).系统测试平台为 733MHz Pentium III 处理器,Windows NT 4.0,Microsoft Visual C++ 6.0 编译器,缺省编译器优化选项.因为 SMO 算法主要集中在核函数的计算上,所以我们以核函数的计算次数作为评测标准,这样可以独立于不同的测试环境.在计算 KKT 条件时,精度越高,收敛速度越慢,我们取精度为 0.001.

Adult Data*是一份人口普查数据,每个家庭有 14 个属性,经量化后得到一个 123 维的二值向量,共 32 562 个样本,用来预测该家庭的年收入是否超过 5 万美元.选择高斯核函数 $K(x_1,x_2)=\exp(-\|x_1-x_2\|^2/2\sigma^2)$, $\sigma^2=10$,C 取 1(与 Platt 相同).

MNIST**是 AT&T 贝尔实验室的手写数字识别评测数据库.样本来自 250 个人,共 60 000 个样本,每个样本是一个 28×28 大小的灰度图像,构成一个 784 维的向量.我们选择数字 8 作为一类,其余数字作为另一类构成两类问题.选择 4 次多项式函数 $k(x_1,x_2)=((x_1\cdot x_2)/784)^4$,$C$ 取 10.

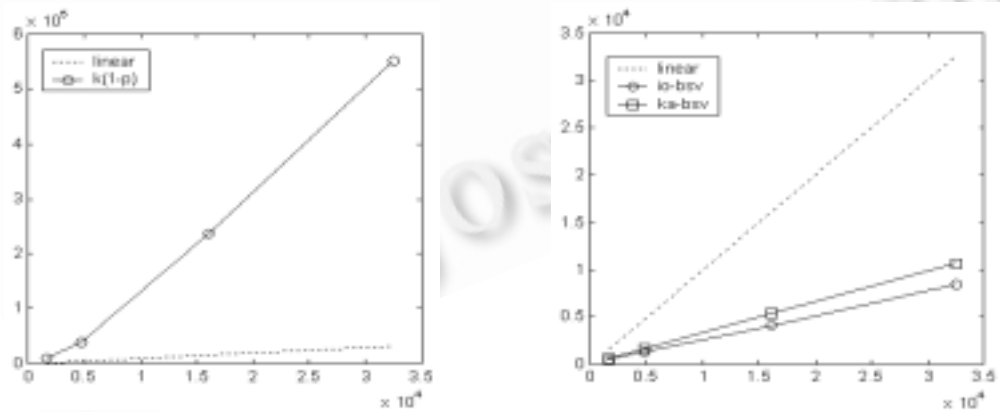
表 1 给出了训练过程中与计算量相关的变量实际数值,因为缓存机制在第 1 次结束内循环后才开始生效,所以所有变量都是从第 1 次内循环结束后开始统计.为便于分析,图 3(a)给出了 $k\cdot(1-p)$ 相对于 N 的曲线关系,对于 Adult 数据, $k\cdot(1-p)\approx N^{1.27}$.图 3(b)给出了 *io_bsv* 和 *ka_bsv* 相对于 N 的曲线关系,其中虚线表示斜率=1 的直线.很明显, *io_bsv* 和 *ka_bsv* 随 N 线性增长,且位于斜率=1 的直线下方, *io_bsv*/*ka_bsv* 近似为常数.对于我们的实验数据,不等式(10)成立.

表 2 给出了修改前后 SMO 算法的比较,减少的计算量和我们上一节分析计算的结果一致.改进的 SMO 算法将计算量平均减少了 50%~70%,即速度是原算法的 2~3 倍.

Table 1 The statistics in the SMO training

表 **1** SMO 训练中的统计量

	N	*k*(1−*p*)	*io_bsv*	*ka_bsv*	*ka_nbsv*	BSV	Non-BSV
Adult1	1 605	10 827	456	581	128	584	107
Adult4	4 781	43 710	1 326	1 642	306	1 652	232
Adult7	16 101	237 024	4 125	5 346	940	5 378	544
Adult	32 562	551 996	8444	10 619	1 780	10 656	1 018
MNIST1	500	4 715	39	67	31	62	44
MNIST2	2 000	34 802	190	274	181	259	224
MNIST3	5 000	135 767	350	721	485	701	533
MNIST4	20 000	307 735	1 467	3 007	1 977	2 921	2 188



(a) The plot of $k(1-p)$ relate to N (b) The plots of *io_bsv*, *ka_bsv* relate to N

(a) Adult 训练中 $k(1-p)$与 N 的关系 (b) Adult 训练中 *io_bsv*, *ka_bsv* 与 N 的关系

Fig 3 The plots of $k(1-p)$, *io_bsv* and *ka_bsv* relate to N in Adult data

图 3 Adult 训练中 $k(1-p)$, *io_bsv* 和 *ka_bsv* 相对于 N 的关系曲线

*该数据可由ftp://ftp.ics.uci.edu/pub/machine-learning-database/adult获得.

**该数据可由http://www.research.att.com/~yann/ocr/mnist/index.html获得.

Table 2 The comparison of computations between improved SMO and original SMO

表 2 改进前后的 SMO 计算量比较

	The computing number of kernel function①	
	SMO+Buffering②	SMO
Adult1	2 587 815	8 145 618
Adult4	22 406 766	75 864 350
Adult7	226 937 669	1 517 467 798
Adult	1 131 809 727	6 958 485 835
MNIST1	172 554	467 642
MNIST2	6 846 486	16 018 429
MNIST3	68 234 522	164 474 762
MNIST4	648 170 169	1 543 145 961

①核函数计算次数,②缓存.

4 小 结

针对训练 SVM 的分解算法,本文提出一种高效的计算决策函数的缓存策略,该策略充分利用了 SVM 训练过程中到达上界的拉格朗日乘子变化平稳的特性,有效地降低了计算决策函数的代价,采用该缓存策略的 SMO 算法在运算速度上得到了显著的提高,而且缓存策略很容易推广到其他分解算法中.

如果用 SVM 解决问题线性可分问题时,BSV 占总样本的比例经常比较小,这时,我们的缓存机制作用减小,在极端的情况下不会减少计算量,但也不会带来额外的计算.但实际问题通常都很复杂,在高维特征空间也很难做到线性可分,这时 BSV 占总样本的比例经常比较大,更有利于缓存机制发挥作用.

References:

[1] Bian, Zhao-qi, Zhang, Xue-gong. Pattern Recognition. 2nd ed., Beijing: Tsinghua University Press, 1999 (in Chinese).

[2] Vapnik, V. The Nature of Statistical Learning Theory. New York: Springer-Verlag, 1995.

[3] Burges, C.J.C. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 1998,2(2): 955~974.

[4] Boser, B.E., Guyon, I.M., Vapnik, V.N. A training algorithm for optimal margin classifiers. In: Haussler, D., ed. Proceedings of the 5th Annual ACM Workshop on COLT. Pittsburgh: ACM Press, 1992. 144~152.

[5] Osuna, E., Freund, R., Girosi, F. Training support vector machines: an application to face detection. In: Plummer, D., ed. Proceedings of the Compute Vision and Pattern Recognition. 1997. 130~136.

[6] Platt, J. Fast training of support vector machines using sequential minimal optimization. Advances in Kernel Methods-Support Vector Learning. Cambridge, MA: MIT Press, 1999. 185~208.

[7] Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., *et al*. Improvements to platt's SMO algorithm for SVM classifier design. Technical Report. Bangalore, India: Department of CSA, IISc, 1999.

[8] Chang, C.C, Hsu, C.W, Lin, C.J The analysis of decomposition methods for support vector machines. In: Dean, T., ed. Proceedings of the 16th International Joint Conference on Artificial Intelligence, 1999. 200~204.

[9] Flake, G.W., Lawrence, S. Efficient SVM regression training with SMO. Technical Report, NEC Research, 1999.

[10] Joachims, T. Making large-scale SVM learning practical. Advances in Kernel Methods-Support Vector Learning, Cambridge. MA: MIT Press, 1999. 169~184.

附中文参考文献:

[1] 边兆祺,张学工.模式识别.第 2 版,北京:清华大学出版社,1999.

An Improved Sequential Minimization Optimization Algorithm for Support Vector Machine Training*

SUN Jian, ZHENG Nan-ning, ZHANG Zhi-hua

(*Institute of AI and Robotics, Xi'an Jiaotong University, Xi'an* 710049, *China*)

E-mail: sj@aiar.xjtu.edu.cn; nnzheng@xjtu.edu.cn

http://www.aiar.xjtu.edu.cn

Abstract: The decomposition methods are main family to train SVM (support vector machine) for large-scale problem. In many pattern classification problems, most support vectors' Lagrangian multipliers are bound, and those multipliers change smoothly during training phases. Based on the facts, an efficient caching strategy is proposed to accelerate the decomposition methods in this paper. Platt's sequential minimization optimization (SMO) algorithm is improved by this caching strategy. The experimental results show that the modified algorithm can be 2~3 times faster than the classical SMO for large real-world data sets.

Key words: support vector machine; pattern classification; quadratic programming; caching strategy; sequential minimization optimization algorithm

* Received December 7, 2000; accepted November 5, 2001

Supported by the National Natural Science Foundation of China under Grant Nos.60175006, 60024301; the National Creative Research Group Science Foundation of China under Grant No.60024301

敬告作者

《软件学报》创刊以来,蒙国内外学术界厚爱,收到许多高质量的稿件,其中不少在发表后读者反映良好,认为本刊保持了较高的学术水平.但也有一些稿件因不符合本刊的要求而未能通过审稿.为了帮助广大作者尽快地把他们的优秀研究成果发表在我刊上,特此列举一些审稿过程中经常遇到的问题,请作者投稿时尽量予以避免,以利大作的发表.

1. 读书偶有所得,即匆忙成文,未曾注意该领域或该研究课题国内外近年来的发展情况,不引用和不比较最近文献中的同类结果,有的甚至完全不列参考文献.

2. 做了一个软件系统,详尽描述该系统的各个方面,如像工作报告,但采用的基本上是成熟技术,未与国内外同类系统比较,没有指出该系统在技术上哪几点比别人先进,为什么先进.一般来说,技术上没有创新的软件系统是没有发表价值的.

3. 提出一个新的算法,认为该算法优越,但既未从数学上证明比现有的其他算法好(例如降低复杂性),也没有用实验数据来进行对比,难以令人信服.

4. 提出一个大型软件系统的总体设想,但很粗糙,而且还没有(哪怕是部分的)实现,很难证明该设想是现实的、可行的、先进的.

5. 介绍一个现有的软件开发方法,或一个现有软件产品的结构(非作者本人开发,往往是引进的,或公司产品),甚至某一软件的使用方法.本刊不登载高级科普文章,不支持在论文中引进广告色彩.

6. 提出对软件开发或软件产业的某种观点,泛泛而论,技术含量少.本刊目前暂不开办软件论坛,只发表学术文章,但也欢迎材料丰富,反映现代软件理论或技术发展,并含有作者精辟见解的某一领域的综述文章.

7. 介绍作者做的把软件技术应用于某个领域的工作,但其中软件技术含量太少,甚至微不足道,大部分内容是其他专业领域的技术细节,这类文章宜改投其他专业刊物.

8. 其主要内容已经在其他正式学术刊物上或在正式出版物中发表过的文章,一稿多投的文章,经退稿后未作本质修改换名重投的文章.

本刊热情欢迎国内外科技界对《软件学报》踊跃投稿.为了和大家一起办好本刊,特提出以上各点敬告作者.并且欢迎广大作者和读者对本刊的各个方面,尤其是对论文的质量多多提出批评建议.