文章编号: 1001-0920(2004)08-0852-05

基于改进聚类算法的分布式 SVM 及其应用

桂卫华, 李勇刚, 阳春华, 陈志盛
(中南大学 信息科学与工程学院, 湖南 长沙 410083)

摘 要: 针对 RPCL 聚类算法存在的缺点, 提出一种改进算法, 并在此基础上得到了一种分布式支持向量机(DSVM). 针对 SVM 算法中阈值难以确定的问题, 提出了一种两段学习算法. 最后将 DSVM 应用于氧化铝高压溶出过程苛性比值的软测量, 现场数据的仿真结果表明该方法具有较高的精度, 能满足实际生产的需要.

关键词: 支持向量机; RPCL 聚类算法; 软测量; 苛性比值
中图分类号: TP18 **文献标识码**: A

Distributed SVM based on improved clustering algorithm and its application

GUI Wei-hua, LI Yong-gang, YANG Chun-hua, CHEN Zhi-sheng
(College of Information Science and Engineering, Central South University, Changsha 410083, China. Correspondent: LI Yong-gang, E-mail: lyg-tod@163.com)

Abstract: An improved rival penalized competitive learning (IRPCL) clustering algorithm is proposed. A distributed support vector machine (DSVM) is constructed. Aiming at the difficulty of computing bias of SVM, a two-phase algorithm is proposed. DSVM is applied in soft sensing for ratio of soda to alumina (RSA) in the process of high-pressure digestion of alumina. Simulation result shows that the method possesses high precision and can meet actual demands.

Key words: support vector machine; RPCL clustering algorithm; soft sensing; RSA

1 引 言

作为一种建模工具, Cortes 等[1]提出的基于结构风险最小化原理的支持向量机具有良好的泛化能力, 且不存在局部最优问题, 使得 SVM 受到了广泛关注[2,3]. 自 SVM 提出以来, 人们便提出了多种学习算法, 如分解算法[4]、SMO 算法[5]、增量学习算法[6]等. 这些算法中, 阈值均根据最优化理论取边界上的一点或其平均值, 但这并不是最优阈值. 本文提出一种两段学习算法, 保证了 SVM 模型的最优化.

SVM 学习算法的复杂度取决于支持向量的个数, 因而对于大规模数据样本而言, 其计算量非常大[6]. 另外, 对不同区域的输入变量, 其扰动幅度不同, 如果用单一模型描述输入输出关系, 会影响模型的泛化能力. 为解决这些问题, 一种有效的方法是采用分而治之的策略[7,8], 将复杂问题分解为若干个较为简单的问题来处理, 即首先将学习样本聚类, 分别用不同的 SVM 描述不同类别的学习样本; 然后利用一种模糊分类器确定输入对每个 SVM 模型的隶属度, 根据隶属度对每个模型的输出进行综合而得到整个分布式 SVM 的输出.

RPCL 算法是一种非常有效的聚类算法, 但它存在不足之处. 本文分析了RPCL 算法存在的缺陷,

收稿日期: 2003-08-21; **修回日期**: 2003-09-28.
基金项目: 国家 863 计划项目(2001AA411040); 国家 973 计划资助项目(2002CB312200).
作者简介: 桂卫华(1950—), 男, 湖北武汉人, 教授, 博士生导师, 从事大系统理论、鲁棒控制和复杂生产过程的建模及优化控制等研究; 李勇刚(1974—), 男, 湖南长沙人, 博士生, 从事复杂生产过程的建模及优化控制的研究.

提出一种改进算法 该算法能加快聚类速度, 提高聚类精度

2　改进的 RPCL 聚类算法

RPCL 算法的基本思想是[9]: 对于输入, 不仅竞争获胜单元的权矢量被修正, 以适应输入值, 同时对次胜单元采用惩罚的方法, 使之远离输入值 这样, RPCL 算法在将获胜单元吸引过来的同时, 将次胜者推开, 从而能自动确定数据集的类数 但RPCL 算法存在如下缺陷: 在权矢量调整过程中, 若选取的样本位于某个类中心的附近时, 算法能将获胜单元向类中心方向吸引, 同时将次胜单元向远离类中心的方向推开; 而当选取的样本位于某个类的边缘时, 算法可能将获胜单元向偏离类中心方向吸引, 随机选取样本会造成权矢量在类中心和边缘之间来回波动, 从而影响聚类速度及精度

为了让权矢量尽快向相应的类中心移动, 选取样本时, 应考虑样本的空间分布情况 当样本在类中心附近时, 应以较大的概率选中它, 使权矢量能以较大的概率向类中心移动; 反之, 当样本在类边缘时, 则以较小的概率选中它 为实现这种思想, 必须确定样本与其相应类中心之间的距离 但类中心未知, 必须采用一种间接方法衡量样本与类中心之间的距离 为此, 本文给出如下定义:

定义1　设 $d_{ij} = \| x_i - x_j \|$ 表示样本 i 与 j 之间的距离, 找出与样本 i 距离最近的 L 个样本, 其距离为 $d_{1i}, d_{2i}, \cdots, d_{Li}$, 则称

$$D_i = 1 \Big/ \left(\frac{1}{L} \sum_{k=1}^{L} d_{ki} \right) \tag{1}$$

为样本 i 的区域密度 D_i 越大(说明样本 i 附近样本较多), 它距类中心越近; 反之说明样本 i 距类中心越远, 即处于类边缘 这样, 可确定样本 i 被选中的概率为

$$p_i = D_i \Big/ \sum_{k=1}^{N} D_k, \tag{2}$$

其中 N 为样本点总数 按上述思想, 可以给出改进 RPCL 算法如下:

1) 设定初始类个数 m 及循环次数 T, 初始化 $w_i (i = 1, 2, \cdots, m)$, 并设 $t = 1$.

2) 计算样本 i 被选中的概率 $p_i (i = 1, 2, \cdots, N)$.

3) 产生随机数 $\zeta \in [0 \quad 1]$, 若 $0 \leq \zeta < p_1$, 则选中样本 x_1; 如果 $\sum_{j=1}^{k-1} p_j \leq \zeta < \sum_{j=1}^{k} p_j$, 则选中样本 $x_k (k = 2, 3, \cdots, N)$.

4) 设被选中的样本等于 x, 对 $i = 1, 2, \cdots, m$, 令

$$u_i = \begin{cases} 1, & i = c; \\ -1, & i = r; \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

其中: c 表示获胜单元; r 表示次胜单元, 即

$$\gamma_c \| x - w_c \|^2 = \min_j \gamma_j \| x - w_j \|^2, \tag{4}$$

$$\gamma_r \| x - w_r \|^2 = \min_{j \neq c} \gamma_j \| x - w_j \|^2. \tag{5}$$

其中: $\gamma_j = n_j \Big/ \sum_{k=1}^{m} n_k$, n_k 是 $u_k = 1$ 的次数

5) 由下式修改权矢量 w_i, 即

$$\Delta w_i(t) = \begin{cases} k_c (x - w_i(t)), & u_i = 1; \\ -k_r (x - w_i(t)), & u_i = -1; \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

其中: $0 \leq k_c, k_r \leq 1$ 为学习率

6) $t = t + 1$, 如果 $t < T$, 则转 2).

7) 对于每个样本, 找出与其最近的类中心, 并归入相应的类

8) 计算每个类包含的样本个数, 如果小于给定值 ξ, 则剔除相应的类中心

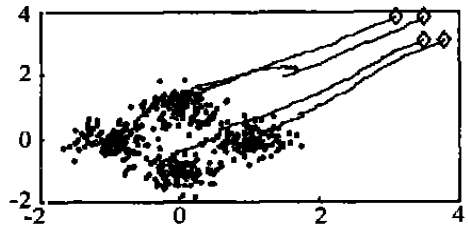
由于样本的类数未知, 必须事先假定有较多的类别, 算法中的最后两步就是在聚类完成后删除多余的类 本算法中, L 是一个非常重要的参数, L 过小或过大, D 都不能真实反映样本的区域密度, 从而影响算法的效率 本文取 $L = [N/m]$, $[x]$ 表示不超过 x 的最大整数

利用文献[9]中的算例, 对改进 RPCL 算法及传统 RPCL 算法进行仿真研究 算例中待聚类的数据集是两维数据, 由 4 个高斯分布的类别组成, 其类中心分别为 $(-1, 0)$, $(1, 0)$, $(0, 1)$, $(0, -1)$, 方差均为 0.1, 每个类的数据量均为 100 组, 共 400 组样本 实验中取学习速率分别为 $k_c = 0.02$ 和 $k_r = 0.01$. 权矢量修改的最大次数 T 取 1 000, 预先设定样本类数为 4 实验结果如图 1 所示, 图中: 右上角的 4 个菱形表示权矢量的初始值, 左下角的 4 个菱形表示权矢量的最终值, 4 条曲线表示权矢量的运动轨迹 从图中可以看出, 在传统 RPCL 算法中, 由于受边缘数据的影响, 权矢量经较大的波折后才到达类中心; 而改进算法每个权矢量都迅速地向类中心收敛, 聚类速度比传统 RPCL 算法有了较大的提高

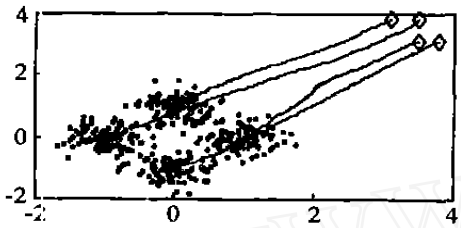
3　分布式支持向量机模型

3.1　支持向量机及两段学习算法

机器学习问题就是从给定的 p 个样本 $\{x_i, y_i\} (i = 1, 2, \cdots, p)$ 中找出一个函数 f 来描述输入输出关



(a) 传统RPCL 算法聚类结果



(b) 改进RPCL 算法聚类结果

图 1 聚类结果比较

系, 其中 $x_i \in R^n$ 和 $y_i \in R$ 分别为样本输入和输出. 支持向量机用于非线性函数逼近的基本思想是: 通过一个非线性映射 Φ, 将输入数据 x 映射到一个 m 维($m > n, m$ 为支持向量个数) 特征空间 F, 将非线性逼近问题转化为高维特征空间中的线性函数逼近问题. 即函数具有如下形式:

$$f(x) = w^{\mathrm{T}}\phi(x) + b, w \in R^m, b \in R. \tag{7}$$

根据结构风险最小化准则[2], 函数逼近的目的就是寻找权值 w 和阈值 b, 最小化

$$J = \frac{1}{2}w^{\mathrm{T}}w + C\sum_{i=1}^{p}L(y_i - f(x_i)). \tag{8}$$

其中: C 为折衷因子, L 为风险函数. 在实际应用中, 经常会引入松弛因子 ξ_i 和 ξ_i^*. 这样, 函数逼近便成为寻找最优超平面, 即

$$\begin{aligned}&\min_{w,b,\xi,\xi^*} J = \frac{1}{2}w^{\mathrm{T}}w + C\sum_{i=1}^{p}(\xi_i + \xi_i^*);\\ &\text{s.t. } y_i - w^{\mathrm{T}}\phi(x_i) - b \le \varepsilon + \xi_i,\\ &\quad w^{\mathrm{T}}\phi(x_i) + b - y_i \le \varepsilon + \xi_i^*,\\ &\quad \xi_i, \xi_i^* \ge 0.\end{aligned} \tag{9}$$

其中 ε 为允许的最大误差值. 利用Lagrange 优化方法[2], 可得到问题(9) 的对偶问题为

$$\begin{aligned}&\max_{\alpha,\alpha^*} L =\\ &-\frac{1}{2}\sum_{i,j=1}^{p}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) -\\ &\varepsilon\sum_{i=1}^{p}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{p}y_i(\alpha_i - \alpha_i^*);\\ &\text{s.t. } \sum_{i=1}^{p}(\alpha_i - \alpha_i^*) = 0, \alpha_i, \alpha_i^* \in [0\quad C].\end{aligned} \tag{10}$$

其中: α_i, α_i^* 为 Lagrange 因子; $k(x, y) = \phi(x)^{\mathrm{T}}\phi(x)$ 称为核函数, 它是满足Mercer 条件的任意对称函数[2]. 显然, 支持向量机的学习是一个不等式约束下的二次函数寻优问题, 存在唯一解. 根据Lagrange 优化理论, 对于问题(10) 的最优解, 函数 $f(x)$ 可表示为

$$f(x) = \sum_{i=1}^{p}(\alpha_i - \alpha_i^*)k(x_i, x) + b. \tag{11}$$

根据 KKT 条件[2], 最优点应满足如下条件:

$$\begin{aligned}&\alpha_i\left(\varepsilon + \xi_i - y_i + \sum_{j=1}^{p}(\alpha_j - \alpha_j^*)k(x_j, x_i) + b\right) = 0,\\ &\alpha_i^*\left(\varepsilon + \xi_i^* + y_i - \sum_{j=1}^{p}(\alpha_j - \alpha_j^*)k(x_j, x_i) - b\right) = 0;\end{aligned} \tag{12}$$

及

$$(C - \alpha_i)\xi_i = 0, (C - \alpha_i^*)\xi_i^* = 0. \tag{13}$$

容易证明, 解中只有少部分 α_i, α_i^* 不为零, 且其中至少有一个为零, 与其对应的样本 x_i 就是支持向量. 根据式(12) 和(13), 可计算阈值

$$\begin{aligned}&b = y_i - \varepsilon - \sum_{j=1}^{p}(\alpha_j - \alpha_j^*)k(x_j, x_i)\alpha_i \in (0\quad C),\\ &b = y_i + \varepsilon - \sum_{j=1}^{p}(\alpha_j - \alpha_j^*)k(x_j, x_i)\alpha_i^* \in (0\quad C).\end{aligned} \tag{14}$$

对任意不为零的Lagrange 因子, 都可由式(14) 求出一个阈值 b, 因此如何确定阈值便成了一个难题. 在众多的 SVM 算法中, 阈值一般有 2 种取法[4~7]: 从所有阈值中任取一个或取其平均值. 这样, 实际上大多数(甚至全部) 不为零的Lagrange 因子, 并没有满足KKT 条件(12), 因此这并不是最优解. 为此, 本文提出一种两段学习算法.

首先利用分解算法[5] 求出Lagrange 因子 α_i, α_i^*; 找出其中不为零的因子, 与其对应的样本点即为支持向量. 这样, 输入输出之间的关系可描述为

$$f(x) = \sum_{i=1}^{S}\beta_i k(x_i, x) + b. \tag{15}$$

其中: S 为支持向量机的个数, $x_i (i = 1, 2, \cdots, S)$ 为支持向量, β_i 实际上是那些不为零的Lagrange 因子, b 为阈值.

确定了支持向量后, 第 2 步工作是寻找最优权值 β_i 和阈值 b, 使得函数值与实际值拟合得最好, 即

$$\min J = \frac{1}{p}\sum_{i=1}^{p}(y_i - f(x_i))^2. \tag{16}$$

这是一个线性回归问题, 可用最小二乘法求解, 即

$$\beta = (K^{\mathrm{T}}K)^{-1}K^{\mathrm{T}}Y. \tag{17}$$

其中: $\beta = [\beta_1, \cdots, \beta_S, b]^T$, $K = [k_1, \cdots, k_p]^T$, $k_i = [k(x_1, x_i), \cdots, k(x_S, x_i), 1]^T$ $(i = 1, 2, \cdots, p)$, $Y = [y_1, \cdots, y_p]^T$.

普通的 SVM 学习算法可同时确定支持向量、权值及阈值 而两段学习算法是在确定支持向量后再利用最小二乘法计算权值及阈值, 虽然增加了计算量, 但能够求解出在均方误差意义下的最优解, 因此计算量的增加是值得的

3.2　分布式支持向量机结构

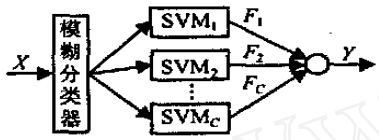


图 2　分布式支持向量机模型

分布式支持向量机结构如图 2 所示, 其中模糊分类器用于确定输入 X 对于每个子 SVM 模型的隶属度 根据隶属度将每个子 SVM 模型的输出综合, 得到最终的输出为

$$Y = \sum_{i=1}^{C} u_i F_i \tag{18}$$

其中: C 为类的个数, F_i 为第 i 个支持向量机的输出, u_i 为输入 X 对于第 i 个支持向量机的隶属度 隶属度可按如下策略求得: 输入 X 距某一类中心越近, 说明其属于该类的可能性越大, 因此它对该类的隶属度也应越大 根据这种思想, 可计算得到隶属度

$$u_i = \begin{cases} 1, d(x, c_i) = 0; \\ \dfrac{1/d(x, c_i)}{\sum_{j=1}^{C} (1/d(x, c_j))}, \text{otherwise} \end{cases} \tag{19}$$

其中: c_i 表示第 i 类的类中心, $d(x, c_i)$ 表示输入与相应的类中心的欧氏距离

4　苛性比值软测量

氧化铝高压溶出是在高温高压条件下, 利用苛性钠溶液将铝土矿中的氧化铝溶解成铝酸钠溶液, 是氧化铝生产过程中最重要的一个环节 高压溶出过程中, 溶出液的苛性比值是一个非常重要的质量指标, 不仅反映了溶出过程中的碱耗, 而且对后续生产将产生极大的影响 因此, 如何在线检测苛性比值, 并调节高压溶出条件, 达到最佳溶出效果非常重要 然而, 在实际生产中, 苛性比值是通过对溶出矿浆化学成分分析结果计算出来的, 因而存在较大滞后, 对溶出过程的控制极为不利

通过对氧化铝高压溶出过程的机理分析可知, 影响苛性比值的因素主要包括 3 个方面: 原矿浆化学成分、物理特性及溶出工况(如溶出温度、压力) 共 20 个因素 因为影响苛性比值的因素较多, 而支持向量机的复杂度与输入变量维数无关, 因此非常适合苛性比值的软测量

氧化铝厂对原矿浆及溶出矿浆的化学分析每隔 2 h 进行 1 次, 从氧化铝厂收集了一年多的实际运行数据共 4 000 多条 由于某些仪表及人为误差, 其中一些数据与实际值存在很大误差 为保证数据的可靠性, 通过对这些数据认真分析和处理, 剔除一些不合理的样本, 从中选取了 2 100 条数据样本, 其中 1 500 条样本用于建立数学模型, 另外 600 条用于检验模型的精度

对于建模数据, 预先设定其类数为 25, 则 $L = 60$; 然后计算样本的区域密度及在 RPCL 算法中被选中的概率 随机选取初始权矢量, 按改进的 RPCL 算法进行聚类 聚类完成后, 将样本归类, 其中有 6 类样本个数小于 10, 可以剔除, 从而可以将建模样本归为 19 类; 然后计算样本与其所属类中心之间的平均距离, 即

$$d = \sum_{j=1}^{m} \sum_{i=1}^{N_j} \| x_{ij} - c_j \| \Big/ \sum_{j=1}^{m} N_j \tag{20}$$

其中: m 表示最终选定的类数量, N_j 表示第 j 类样本个数, x_{ij} 表示第 j 类的第 i 个样本, c_j 表示最终得到的第 j 个类的中心 聚类的目的是最小化 d. 按同样的条件利用传统的 RPCL 算法进行聚类, 并对不同的初始值及迭代次数用 2 种聚类算法进行了 20 次仿真, 其结果如图 3 所示(聚类前所有样本数据都进行了归一化处理).

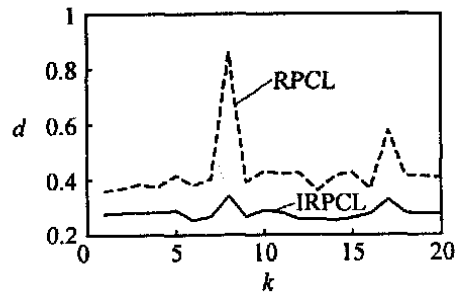


图 3　RPCL 与 IRPCL 仿真结果比较

从图 3 可以看出, 对于改进算法, d 值明显较小, 这说明它的聚类精度更高; 另外, 图 3 中的第 8 及第 17 次结果是迭代次数较小的情况, IRPCL 算法已经基本上收敛到最优值, 而 RPCL 算法还远远没有到达最优值, 这说明 IRPCL 算法的聚类速度有了

很大的提高

改进聚类算法将样本空间分成了 19 个类, 分别用 19 个 SVM 逼近不同的样本空间. 在 SVM 学习中, 折衷因子 C 取 0.8, 允许的误差值 ϵ = 0.01. 表 1 比较了普通 SVM 学习算法与两段学习算法及分布式 SVM (DSVM) 与单一的 SVM (SSVM) 的效果. 从表 1 可以看出, 两段算法比普通算法精度高, 分布式 SVM 比单一 SVM 精度高. 对于本文方法, 其预测精度达到 97.3%, 完全能满足生产的要求. 从表 1 还可以看出, 由于有效地将样本聚类, 分布式 SVM 的支持向量个数明显减少, 能获得相对简单的模型.

表 1 不同 SVM 模型及算法的比较

	算法	标准方差	预测精度	支持向量个数
DSVM	普通	0.0751	95.1%	102
	两段	0.0415	97.3%	102
SSVM	普通	0.0912	93.9%	131
	两段	0.0754	95.0%	131

5 结 论

传统 RPCL 聚类算法在学习过程中没有考虑样本空间分布对权值调整的影响, 从而聚类速度慢、精度低. 本文在引入样本区域密度概念的基础上, 提出了一种改进聚类算法, 加快了聚类速度, 提高了聚类精度. 针对 SVM 学习过程中阈值难以确定的问题, 提出了两段学习算法, 该算法能够得出最优权值和阈值, 提高了 SVM 的精度.

将基于改进聚类算法的分布式 SVM 模型用于氧化铝高压溶出过程中苛性比值的软测量, 仿真结果表明该方法具有较高的预测精度, 能满足实际要求, 可应用于实际生产.

参考文献(References):

[1] Corinna Cortes, Vladimir Vapnik. Support vector networks[J]. *Machine Learning*, 1995, 20(3): 273-295.

[2] Bas J de Kruif, Theo J A de Vries. Support-vector-based least squares for learning non-linear dynamics [A]. *Proc of the 41st IEEE Conf on Decision and Control*[C]. Las Vegas, Nevada, 2002. 1343-1348.

[3] 王定成, 方廷健, 高理富. 支持向量机回归在线建模及应用[J]. 控制与决策, 2003, 18(1): 89-91.
(Wang D C, Fang T J, Gao L F. Support vector machines regression on-line modeling and its application [J]. *Control and Decision*, 2003, 18(1): 89-91.)

[4] Chang Chih-Chung, Hsu Chih-Wei, Lin Chih-Jen. The analysis of decomposition methods for support vector machines[J]. *IEEE Trans on Neural Networks*, 2000, 11(4): 1003-1008.

[5] John C P. Fast training of support vector machines using sequential minimal optimization[A]. *Advances in Kernel Methods-Support Vector Learning* [C]. Cambridge, MA: MIT Press, 1999. 185-208.

[6] 萧嵘, 王继成, 孙正兴, 等. 一种 SVM 增量学习算法 α-ISVM [J]. 软件学报, 2001, 12(12): 1818-1824.
(Xiao R, Wang J C, Sun Z X, et al. An incremental SVM learning algorithm α-ISVM [J]. *J of Software*, 2001, 12(12): 1818-1824.)

[7] 王旭东, 邵惠鹤, 罗荣富. 分布式 RBF 神经网络及其在软测量方面的应用[J]. 控制理论与应用, 1998, 15(4): 558-563.
(Wang X D, Shao H H, Luo R F. The distributed RBF neural network and its application in soft sensor[J]. *Control Theory and Applications*, 1998, 15(4): 558-563.)

[8] Cao Lijuan. Support vector machines experts for time series forecasting[J]. *Neurocomputing*, 2003, 51: 321-339.

[9] Xu Lei, Krzyzak Adam, Oja Erkki. Rival penalized competitive learning for clustering analysis, RBF net, and curve detection [J]. *IEEE Trans on Neural Networks*, 1993, 4(4): 636-649.

下 期 要 目