迭代重加权最小二乘支持向量机快速算法研究

温 雯[1] 郝志峰[1,2] 邵壮丰[3]
(广东工业大学计算机学院 广州 510006)[1] (华南理工大学计算机科学与工程学院 广州 510641)[2]
(中国电信广东互联网与增值业务运营中心 广州 510110)[3]

摘 要 迭代重加权(Iteratively Reweighted)方法是提高最小二乘支持向量机(LS-SVM)稳健性的重要手段,但由于涉及到多次加权和重复训练,该方法需要大量运算,无法广泛应用。通过数值推导,获得了求解迭代重加权最小二乘支持向量机(IRLS-SVM)的快速算法,大幅度减少了其运算复杂度。引入了3种经典的加权函数,并在多个仿真数据集和实际数据集上进行实验,证实了IRLS-SVM能获得相当稳健的学习结果,所提出的快速算法也确实能够大幅度减少训练时间。实验结果同时表明,在快速训练算法的框架下,3种不同的权重函数可能要求不同的训练时间。

关键词 支持向量机,稳健性,异常样本,快速算法

Study on the Fast Training Algorithm of Iteratively Re-weighted Least Squares Support Vector Machine

WEN Wen[1] HAO Zhi-feng[1,2] SHAO Zhuang-feng[3]
(Faculty of Computer, Guangdong University of Technology, Guangzhou 510006, China)[1]
(College of Computer Science and Engineering, South China University of Technology, Guangzhou 510641, China)[2]
(Internet and Value-added Service Center of Guangdong Branch, China Telecom Corporation Limited, Guangzhou 510110, China)[3]

Abstract Iteratively reweighted method is an important approach to improve the robustness of least squares support vector machine(LS-SVM). However, the reweighting and retraining procedure demands a lot of computational time, which makes it impossible for practical applications. In this paper, the iteratively reweighted least squares support vector machine (IRLS-SVM) was studied. An improved training algorithm of IRLS-SVM was proposed. It is based on novel numerical method, and can effectively reduce the computational complexity of IRLS-SVM. Three different weight functions were implemented in the IRLS-SVM. Experiments on simulated instances and real-world datasets demonstrate the validity of this algorithm. Meanwhile, the results reveal that different weight function may require different computational time for the fast training algorithm of IRLS-SVM.

Keywords Support vector machines, Robustness, Outliers, Fast algorithm

1 引言

支持向量机(SVM)是由Vapnik等人提出的一种机器学习算法,其基础为小样本统计学习理论[1,2]。该算法具备良好的泛化能力和学习性能,已广泛应用于模式分类、函数估计以及密度估计等领域[3,4]。由于引入了结构风险最小化的目标,支持向量机(SVM)算法具备较以往算法更好的推广能力。然而这并不能保证SVM算法的稳健性。所谓稳健性,是指对于偏离理论假定时的不敏感性[5]。具体来说,针对回归问题,稳健性就是学习机对于异常样本的不敏感性,即学习机的学习效果不会受到训练集中异常样本的影响。

SVM算法对结构风险的控制是通过控制模型的复杂度($\|w\|^2$)和经验风险(损失函数)来实现的。其中对稳健性起关键作用的就是SVM算法的损失函数。Christmann和Messem通过其最新的理论研究发现,如果SVM使用Lipschitz连续的损失函数以及有界的BIF核,则算法在影响函数(Impact function)的意义下是稳健的[6]。根据这一结论,采用ε-不敏感损失函数、Huber损失函数[7]和pinball损失函数[8]的支持向量回归算法是相对稳健的,而采用误差平方和(Sum Squared Error:SSE)损失函数的最小二乘支持向量机(LS-SVM)回归算法则无法满足稳健性条件。这一结论与过去的经验研究相吻合。

早在LS-SVM[9]算法提出的初期,Suykens等人似乎已经意识到该算法对异常样本的敏感性,因此在2002年提出了加权LS-SVM算法(Weighted Least Squares Support Vector Machine,又为WLS-SVM)[10]来减少异常样本对回归机的负面影响。可以说对SVM进行加权的思路是正确而有实际意义的。这种思路也正是传统回归中用于解决稳健性问题的经典思路[11,12]。此外,对于采用其他损失函数的SVM回归算法,也有研究者提出了不同的加权方案,譬如Chuang等人提出的鲁棒支持向量回归机算法[13],以及Zhang和Guo提出的重加权算法[14]。这类算法都是借鉴传统稳健回归中重加权

到稿日期:2009-09-29 返修日期:2009-12-14 本文受信息安全国家重点实验室开放课题基金(20090401)资助。
温 雯(1981-),女,博士,讲师,主要研究方向为模式识别、机器学习、图像处理等,E-mail:wwen@gdut.edu.cn;郝志峰(1969-),男,教授,博士生导师,主要研究方向为数据挖掘、机器学习、仿生算法等。

最小二乘(Iteratively reweighted least squares methods)策略[15],通过对样本进行重加权,逐步减少异常样本的影响,修正回归机的估计值。但由于需要进行多次的权重设置和重复学习,所耗费的时间也是相当巨大的。

本文以迭代重加权最小二乘支持向量机(Iteratively reweighted least squares support vector machine,IRLS-SVM)为研究对象,针对重加权所引起的算法运算量增加的问题,首先提出了一种能够根据权重变化,求解单次加权 LS-SVM 的迭代更新算法;而后在该算法的基础上,提出并实现了若干种稳健权重设置方案下的快速 IRLS-SVM 学习算法;最后从运算时间和求解精度上比对了该算法和经典 LS-SVM 以及 ε-SVR 的性能。

2 最小二乘支持向量机回归算法

记 $z_i=(x_i,y_i)$,其中 x_i 为第 i 个样本的输入值,y_i 为对应的输出值。对于给定样本集 $Z=\{z_1,\cdots,z_N\}$ 的回归问题,首先通过一个映射 $\varphi(x):R^d\to R^{\tilde{d}}$ 将输入值映射到一个高维特征空间,在该特征空间中回归函数可以表示成:

$$y(x)=w^T\varphi(x)+b \tag{1}$$

有别于传统回归,SVM 在最小化经验风险的同时,也追求结构风险的最小化,即其目标是最小化风险函数:

$$R(\gamma)=\frac{1}{2}w^Tw+\frac{1}{2}\gamma\cdot\sum_{i=1}^{N}L(y_i,y(x_i)) \tag{2}$$

其中损失函数可以有多种表现形式,LS-SVM 选择平方和作为损失函数,并将不等式约束条件变成等式约束条件。具体地,LS-SVM 回归可以表示为如下优化问题[10]。

$$\begin{aligned}&\min J(w,e)=\frac{1}{2}w^Tw+\frac{1}{2}\gamma\sum_{i=1}^{N}e_i^2\\&\text{s.t. } y_i=w^T\varphi(x_i)+b+e_i,i=1,\cdots,N\end{aligned} \tag{3}$$

式中,$e_i(i=0,1,\cdots,N)$为对应第 i 个样本的估计误差。我们期望从该优化问题(3)中获取形如 $y(x)=w^T\varphi(x)+b$ 的估计式,以对未来的样本进行估计和诊断。但是 $\varphi(x)$的具体表达式往往无法知道,因此需要采用一定的技巧将问题加以转换。

利用拉格朗日乘子及矩阵变换方法,优化问题(3)可以转换成对偶空间中的线性方程组(4)的形式。

$$\begin{bmatrix}0 & 1_v^T\\ 1_v & \Omega+\frac{1}{\gamma}I\end{bmatrix}\begin{bmatrix}b\\ \alpha\end{bmatrix}=\begin{bmatrix}0\\ y\end{bmatrix} \tag{4}$$

式中,$y=[y_1,\cdots,y_N]^T$ 为样本输出向量,$1_v=[1,\cdots,1]^T$,$\alpha=[\alpha_1,\cdots,\alpha_N]^T$ 为拉格朗日乘子,$\Omega_{ij}=\varphi(x_i)^T\varphi(x_j)=K(x_i,x_j)$ 为核矩阵,其中 $i=1,\cdots,N,j=1,\cdots,N$。该线性方程组表达方式非常简洁且相对容易求解,这也正是 LS-SVM 引起了众多研究者及实际应用者关注的主要原因。求解式(4)后,可以获得函数估计式:

$$y(x)=\sum_{i=1}^{N}\alpha_iK(x,x_i)+b \tag{5}$$

虽然 LS-SVM 具有模型简单、求解便捷等优点,但由于使用平方和误差作为损失函数,异常样本对估计函数的负面影响将被扩大,因此如何控制学习机对异常样本的敏感性便显得尤为重要。Suykens 提出的加权最小二乘支持向量机(Weighted least squares support vector machine,WLS-SVM)是对这一问题的经典尝试。

WLS-SVM 的基本思想是对每个样本(x_i,y_i)的误差变量 e_i 赋予一个权重因子 v_i,输入样本集可表示成(x_i,y_i,v_i),其中 $x_i\in R^d$,$y_i\in R$,$0\leqslant v_i\leqslant 1$,$i=1,2,\cdots,N$,优化函数变为:

$$\begin{aligned}&\min J^*(w^*,e^*)=\frac{1}{2}\|w^*\|^2+\frac{1}{2}C\sum_{i=1}^{N}v_ie_i^{*2}\\&\text{s.t. } y_i=w^{*T}\varphi(x_i)+b^*+e_i^*,i=1,\cdots,N\end{aligned} \tag{6}$$

式中,v_i 由下式决定:

$$v_i=\begin{cases}1, & \text{if}\left|e_i/\hat{s}\right|\leqslant c_1\\ \dfrac{c_2-|e_i|}{c_2-c_1}, & \text{if } c_1\leqslant\left|e_i/\hat{s}\right|\leqslant c_2\\ 10^{-4}, & \text{otherwise}\end{cases} \tag{7}$$

c_1,c_2 的建议取值分别为 2.5 和 3.0(这是因为:在高斯分布 N(0,1)的假设下,绝对值小于 2.5 的样本出现概率为 99.38%;而绝对值大于 3.0 的样本出现概率仅为 0.13%),$\hat{s}$为 e_i 的标准偏差的稳健性估计,可以由式(8)或式(9)给出。

$$\hat{s}=\frac{IQR}{2*0.6745} \tag{8}$$

IQR 为四分位数间距,即将误差值排序后,分别位于75%和25%位置的两个数据间的差值。

$$\hat{s}=1.483MAD(x_i) \tag{9}$$

$MAD(x_i)$代表 Median Absolute Deviation,即绝对离差的中位数。

WLS-SVM 要求在训练 LS-SVM 的基础上,获得误差的分布信息,进而设置权重 v_i,$i=1,\cdots,N$,最后再次训练加权后的 LS-SVM。因此其运算复杂度相当于训练两次 LS-SVM 的计算复杂度。

3 WLS-SVM 的快速训练算法

注意到不管采用哪种加权方案,WLS-SVM 都会要求重新训练一次 LS-SVM,因此将耗费较多的运算时间。针对这一问题,我们从数值计算的角度给出了一种 WLS-SVM 的快速学习算法。为了便于讨论,首先将 WLS-SVM 的对偶问题改写成如下形式,见式(10)。

$$\begin{bmatrix}\Omega+V_\gamma & 1_v\\ 1_v^T & 0\end{bmatrix}\begin{bmatrix}\alpha\\ b\end{bmatrix}=\begin{bmatrix}y\\ 0\end{bmatrix} \tag{10}$$

式中,$V_\gamma=diag\left\{\frac{1}{\gamma v_1},\cdots,\frac{1}{\gamma v_N}\right\}$是一个对角矩阵。对照式(4),可以发现它与 LS-SVM 模型非常相似,唯一的不同之处在于传统 LS-SVM 的对偶问题中 $\tilde{V}_\gamma=diag\left\{\frac{1}{\gamma},\cdots,\frac{1}{\gamma}\right\}$。这启发我们可以尝试通过 LS-SVM 的训练结果来快速获得 WLS-SVM 的结果。若记 $A=\begin{bmatrix}\Omega+V_\gamma & 1_v\\ 1_v^T & 0\end{bmatrix}$,$\tilde{A}=\begin{bmatrix}\Omega+\tilde{V}_\gamma & 1_v\\ 1_v^T & 0\end{bmatrix}$,$X=\begin{bmatrix}\alpha\\ b\end{bmatrix}$,$Y=\begin{bmatrix}y\\ 0\end{bmatrix}$,则我们的目的是利用 $\tilde{A}^{-1}$ 快速获得 A^{-1},从而避免对 A 重新求逆。为了清晰描述我们的算法原理,首先引入 Sherman-Morrison-Woodbury 公式。

Sherman-Morrison-Woodbury 公式

给定一个可逆矩阵 A、列向量 u_1 和 u_2,假设 $1+u_1^TA^{-1}u_2\neq 0$,则有以下公式成立:

$$(A+u_1u_2^T)^{-1}=A^{-1}-\frac{A^{-1}u_1u_2^TA^{-1}}{1+u_2^TA^{-1}u_1} \tag{11}$$

受 Sherman-Morrison-Woodbury 公式启发,我们可以采

用一种迭代更新的方法由 $\tilde{A}^{-1}$ 快速计算获得 A^{-1}。令 $A_0^{-1}=\tilde{A}^{-1}$，$\tilde{X}=\begin{bmatrix}\tilde{\alpha}\\ \tilde{b}\end{bmatrix}$（其中 $\tilde{\alpha},\tilde{b}$ 是传统 LS-SVM 在最优超参数下求得的 α 向量和偏置 b）。在每一步迭代中，只考虑与矩阵 $\tilde{A}$ 在第 k 行第 k 列上元素值不一样的对应矩阵，即在第 k 次迭代，则考虑：

$$A_k=A_{k-1}+\begin{bmatrix}0 & \cdots & 0 & \cdots & 0\\ \vdots & \ddots & \vdots & \ddots & \vdots\\ 0 & \cdots & \frac{1}{\gamma}\left(\frac{1}{v_k}-1\right) & \cdots & 0\\ \vdots & \ddots & \vdots & \ddots & \vdots\\ 0 & \cdots & 0 & \cdots & 0\end{bmatrix}\begin{matrix}\\ \\ k^{th}Row\\ \\ \\ \end{matrix}$$
$$k^{th}Col. \qquad (12)$$

显然，当 $k=l$（l 是所有训练样本的样本总数）时，有 $A_l=A$。令

$$u_{1(k)}=(0,\cdots,0,\frac{1}{\gamma},0,\cdots 0)^T$$

$$u_{2(k)}=(0,\cdots,0,\frac{1}{v_k}-1,0,\cdots,0)^T$$

式中，$u_{1(k)}$ 的第 k 个元素是 $1/\gamma$，且 $u_{2(k)}$ 的第 k 个元素是 $1/v_k-1$，则有 $A_k=A_{k-1}+u_{1(k)}u_{2(k)}^T$。因此，根据 Sherman-Morrison-Woodbury 公式，有：

$$A_k^{-1}=A_{k-1}^{-1}-\frac{A_{k-1}^{-1}u_{1(k)}u_{2(k)}^TA_{k-1}^{-1}}{1+u_{2(k)}^TA_{k-1}^{-1}u_{1(k)}} \qquad (13)$$

即，若令 $\dot{a}_{ij}^{(k)}$ 是 A_k^{-1} 的第 i 行、第 j 列的元素，则有：

$$\dot{a}_{ij}^{(k)}=\dot{a}_{ij}^{(k-1)}-\dot{a}_{ik}^{(k-1)}\cdot\frac{1}{\gamma}\left(\frac{1}{v_k}-1\right)\cdot\dot{a}_{kj}^{(k-1)}/\left(1+\frac{1}{\gamma}\cdot\dot{a}_{kk}^{(k-1)}\cdot\left(\frac{1}{v_k}-1\right)\right) \qquad (14)$$

由于 $A_l=A$，$A^{-1}=A_l^{-1}$，则更新后的 X 可以根据式(14)计算而得。

$$X=A_l^{-1}Y \qquad (15)$$

X 中的元素即对应着 WLS-SVM 的 α 和 b。因此，获得 LS-SVM 的训练结果之后，可以用以下迭代算法快速地求解 WLS-SVM。

算法 1 WLS-SVM 的快速求解算法

1. 训练 LS-SVM，并将 LS-SVM 对偶问题的逆矩阵 $\tilde{A}^{-1}$ 保存到 M 中。

2. 采用给定方法对样本进行加权。

3. 设置 $k=1$。

4. 对于第 k 个样本，如果权重 $v_k<1.0$，则根据式(14)更新矩阵 M 的各个元素；否则转而执行步骤 5。

5. 令 $k=k+1$。如果 $k\leqslant l$，转而执行步骤 4；如果 $k>l$，执行步骤 6。

6. 根据式(15)计算 α 和 b，输出它们作为 WLS-SVM 的训练结果。

直观地，算法 1 的流程如图 1 所示。由于采用式(14)对逆矩阵进行更新，针对快速求解算法的复杂度有以下结论：显然在 WLS-SVM 的快速求解算法中，对矩阵 $\tilde{A}^{-1}$，只有当 $v_k<1$ 时，才需要进行迭代更新。而在每次更新中，将产生复杂度为 l^2 的计算量。令 m 代表对应 $v_k<1$ 的样本数，为了获得 WLS-SVM 的解，总共需要进行复杂度为 ml^2 的计算。由于异常样本的数量通常不会太多，因此在合适的权重方案下，大部分样本的权重都将被设置为 1.0，即 $v_k<1$ 的样本数只占少部分，即有 $m<<l$。而直接求解 WLS-SVM 至少需要复杂度为 al^3 的计算量，其中 $a>1$。因此，使用快速求解算法后，WLS-SVM 的运算时间复杂度将大幅度减少，为迭代重加权提供了时间上的保证。当然，从空间复杂度来看，算法 1 需要额外的空间用于存储中间矩阵 M，即它所要求增加的存储量是一个 $l*l$ 的对称矩阵，在目前硬件的发展水平下，这是完全可以接受的。

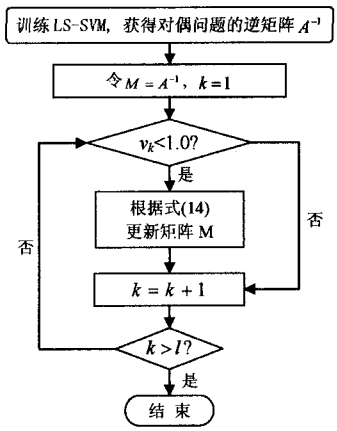

图 1 WLS-SVM 的快速求解算法流程图

4 IRLS-SVM 及其快速算法

在异常样本数目较少的情况下，WLS-SVM 可以正确地设置样本权重，从而较好地消除噪声影响；但随着异常样本数量的增加，根据一次学习获得的信息进行权重设置其正确性逐渐降低。事实上，针对这个问题，Suykens 在文献[10]中曾经提到可以采用多次重(复)加权的方法：即在每一轮的学习中根据上一轮训练结果，对样本设置权重，重新训练，并将这种学习过程进行若干次重复，从而逐渐纠正有偏差的学习结果。这种方法正是迭代重加权最小二乘支持向量机(IRLS-SVM)的思路。然而 IRLS-SVM 面临若干关键难点：其一是权重的设计和选择；其二是加权训练终止条件的确定；其三是快速训练算法的设计和实现。最后一点恰恰是 IRLS-SVM 最困难的一个问题，因为重复训练将导致运算量的大幅度增加，也正是因为如此，Suykens 并不推荐使用重复加权的方案。但是根据前一节所提出的算法 1，发现大幅度减少一次加权训练的运算时间是有可能的，这为多次加权训练创造了条件。在本小节中，将提出实现 IRLS-SVM 的一种完整算法，并基于算法 1 提出适用于 IRLS-SVM 快速训练算法，使其能够根据异常样本数量自适应调整加权的次数，从而较快速地获得稳健的学习结果。

首先引入 3 种经典的权重函数：Huber 权重函数，Bisquare 权重函数以及 Suykens 权重函数。前两种是传统稳健回归中常用的权重函数[15,16]，最后一种则是 Suykens 在经典 WLS-SVM 中所用的权重函数(事实上，Suykens 权重函数也是来源于稳健回归[17])。对于第 t 次迭代中的第 k 个样本，3 种权重函数可以分别用式(16)－式(18)表示。

Huber 权重函数：

$$v(t)_k=\begin{cases}1.0, & |u(t)_k|\leqslant 1.345\\ 1.345/|u(t)_k|, & |u(t)_k|>1.345\end{cases} \qquad (16)$$

Bisquare 权重函数：

$$v(t)_k=\begin{cases}\left[1.0-\left(\frac{u(t)_k}{4.685}\right)^2\right]^2, & |u(t)_k|\leqslant 4.685\\ 10^{-4}, & |u(t)_k|>4.685\end{cases} \qquad (17)$$

Suykens 权重函数：

$$v(t)_k=\begin{cases}1, & \text{if } |u(t)_k|\leqslant 2.5\\ \dfrac{3.0-|u(t)_k|}{3.0-2.5}, & \text{if } 2.5\leqslant|u(t)_k|\leqslant 3.0\\ 10^{-4}, & \text{otherwise}\end{cases} \quad (18)$$

在以上 3 种权重函数中，$u(t)_k=e(t)_k/\hat{s}(t)$，$e(t)_k$ 是第 t 次学习后第 k 个样本对应的误差，$\hat{s}(t)$ 则由式(19)给出。

$$\hat{s}(t)=\frac{IQR(t)}{2*0.6745} \quad (19)$$

式中，$IQR(t)$代表第 t 次循环中的误差项的四分位数间距，即将第 t 次迭代学习获得的误差值$\{e(t)_k|k=0,\cdots,l\}$排序后，分别位于 75%和 25%位置的两个数据间的差值。

3 种权重函数的直观图形如图 2 所示。从图 2 可以看出，3 种权重函数的特点都是对位于高斯分布尾部的样本赋予小权重值，因为处于这一位置的样本往往是偏离了正常分布的异常样本。其次，根据我们的观察，无论使用上述哪一种权重函数，随着重复加权次数的增加，异常样本和正常样本的权重都将逐渐趋向于一个稳定的水平。因此，IRLS-SVM 可以选择在样本权重变化异常微小(即小于某个阈值)的情况下的终止加权和重复训练。再者，针对 IRLS-SVM 耗费运算量过大的情况，可以使用前一小节中的结论，充分利用算法 1 的迭代更新方法，减少运算复杂度，加快算法的运算速度，即只需要将更新公式(14)修改为如下形式：

$$\dot{a}_{ij}^{(k)}=\dot{a}_{ij}^{(k-1)}-\dot{a}_{ik}^{(k-1)}\cdot\frac{1}{\gamma}\left(\frac{1}{v(t)_k}-\frac{1}{v(t-1)_k}\right)\cdot\dot{a}_{kj}^{(k-1)}/$$

$$\left(1+\frac{1}{\gamma}\cdot\dot{a}_{kk}^{(k-1)}\cdot\left(\frac{1}{v(t)_k}-\frac{1}{v(t-1)_k}\right)\right) \quad (20)$$

式中，$v(t)_k$ 代表第 t 次加权后第 k 个样本对应的权重。

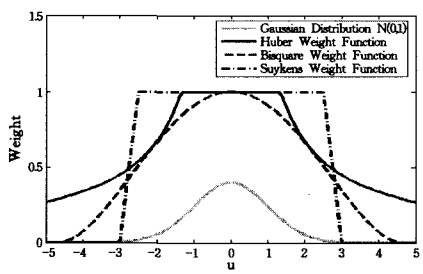

图 2　3 种不同的权重函数：Huber 权重函数，Bisquare 权重函数以及 Suykens 权重函数

以下给出加速 IRLS-SVM 的完整算法。

算法 2　加速 IRLS-SVM

1. 设置各样本权重初始值 $v(0)_k=1.0,k=1,2,\cdots,l$。训练 LS-SVM，并将 LS-SVM 对偶问题的逆矩阵 $\tilde{A}^{-1}$ 保存到 M 中。选择式(16)－式(18)中的一个作为权重函数。

2. 令迭代次数 $t=0$。

3. 根据选定的权重函数对样本进行加权。

4. 采用算法(1)对 WLS-SVM 进行快速训练，即执行以下步骤：

4.1 令 $k=1,flag=0$；

4.2 对于第 k 个样本，如果 $|v(t)_k-v(t-1)_k|>\delta_1$，则根据式(20)更新矩阵 M 的各个元素，并置 $flag=1$；否则，直接执行步骤 4.3；

4.3 令 $k=k+1$，转而执行步骤 4.2，直到 $k>l$。

5. 如果 $flag$ 为真，令 $t=t+1$，转而执行步骤 3；否则，执行步骤 6。

6. 根据式(15)计算 α 和 b，输出它们作为 IRLS-SVM 的训练结果。

在 IRLS-SVM 中，步骤 4 中的变量 $flag$ 保证了 IRLS-SVM 在以下条件满足时停止迭代：如果在前后两次加权过程中，任何一个样本的权重变化量都不超过阈值 δ，停止迭代。由于权重的值域为(0,1]，δ 可以设置为(0,0.01]中的一个较小的数。在我们的实验中，通常将 δ 设置为 0.005。在这种迭代终止条件下，IRLS-SVM 可以针对每一次学习的结果，纠正权重设置，并在学习结果无法再改善的时候，自动终止重复加权的过程，从而保证算法的稳健性。并且在 IRLS-SVM 中，快速训练算法大大减少了迭代加权过程中产生的重复计算，从而保证了算法的效率。

5　实验结果与分析

为了考察算法的有效性，我们分别在若干个仿真数据集和实际数据集上进行了测试，主要从运算时间和测试精度两个方面考察 IRLS-SVM 的算法性能。为了使结果更加客观，实验中不仅记录了经典 LS-SVM 回归算法的测试结果，还加入了相同数据集下 ε-SVR 的测试结果。所有程序在 VC++6.0 环境下编写和运行；其中，我们编写了经典 LS-SVM 以及 IRLS-SVM 的对应算法，包括未加速的 IRLS-SVM(即在算法 2 步骤 4 中直接使用普通的训练方法)和加速的 IRLS-SVM(即算法 2)；ε-SVR 的求解则使用文献[18]中的公开源代码(本文使用的版本为 Libsvm-2.86)。实验中硬件配置条件为：633MHz 的 Celeron 2，960MB 的内存。所用操作系统为 Windows XP。

首先使用非线性回归中的基准函数 sinc 函数进行仿真实验。训练样本由两部分构成：一部分是正常样本，由 sinc 附加小方差的正态扰动随机产生，见式(21)；另一部分为异常样本，由输出值分布于[－1,3]的随机均匀分布产生。

$$f(x)=\frac{\sin x}{x}+\xi,x\in[-15,15]\quad \xi\sim N(0,0.09) \quad (21)$$

仿真实验中考虑了 5 个训练样本集，样本总数从 166 递增至 226。每个样本集中正常样本个数均为 151 个，由式(20)等间隔采样产生；异常样本个数从 15 个递增至 75 个(即 6 个训练集中的异常样本比例大约从 9%递增至 33%)。正态扰动和异常样本均由 Matlab 工具箱随机生成。测试样本集由不加扰动的信号函数产生，且设置不同的采样间隔，以确保测试样本与训练样本不同。测试样本个数为 197。

各算法测试精度对比如表 1 所列，训练时间的相关结果如表 2 所列。表 1 的结果显示，使用 3 种加权函数中的任意一种，IRLS-SVM 都可以获得远远好于 LS-SVM 的测试精度，这说明通过重复加权的方式，确实可以大幅度提升最小二乘支持向量机的稳健性。对于仿真数据集，Suykens 加权方案和 Bisquare 加权方案都较为出色，可以稳定地获得比 ε-SVR 好得多的结果。表 1 同时显示，LS-SVM 的测试精度是 3 种算法中最差的，这与文献[6]中的研究结果相一致(因为 LS-SVM 模型中的 SSE 损失函数并不满足稳健性要求，而 ε-SVR 相对而言更能满足稳健性要求)。从训练时间来看，加速 IRLS-SVM 可以节省大量的训练时间，远远少于未经加速的 IRLS-SVM 所需的时间，这使得重复加权的方式具备运算时间上的可操作性：只需要增加较少的训练时间就能使 IRLS-SVM 获得较 ε-SVR 更好的测试精度。此外，对于 3 种

不同的权重函数，IRLS-SVM 所需要的时间也不尽相同：大多数情况下 Suykens 权重方案需要的训练时间最少，Bisqare 所需的时间最多。这是因为在 Suykens 权重方案下，多数正常样本能够稳定地被赋予恒定权重 1.0，每次迭代学习中需要变化更新的只有介于正常和异常之间的少量的样本权重。与此相反，Bisquare 权重函数是连续函数，在每次学习中，都有较多的样本需要变更其权重值，从而导致迭代更新的次数增加，训练时间也同步增加。

表 1 仿真数据集测试结果：预测精度对比

数据集	ε-SVR (MAE)	LS-SVM (MAE)	IRLS-SVM (MAE)		
			Suykens	Huber	Bisquare
Sinc 166	0.0533	0.2001	**0.0282**	0.0305	0.0288
Sinc 181	0.0456	0.2353	0.0242	0.0336	**0.0237**
Sinc 196	0.0559	0.3054	**0.0277**	0.0522	0.0294
Sinc 211	0.0564	0.3802	0.0450	0.0523	**0.0370**
Sinc 226	0.0660	0.3292	0.0423	0.0613	**0.0369**

* MAE 为 mean absolute error，即平均绝对误差

表 2 仿真数据集测试结果：训练时间对比

数据集	样本总数	异常样本个数	ε-SVR 训练时间(秒)	LS-SVM 训练时间(秒)	IRLS-SVM			
					权重函数	迭代加权次数	训练时间(秒)	
							未加速算法	加速算法
Sinc 166	166	15	0.11	0.11	Suykens	6	0.70	0.21
					Huber	5	0.58	0.21
					Bisquare	4	0.50	0.21
Sinc 181	181	30	0.13	0.14	Suykens	5	0.75	0.22
					Huber	5	0.75	0.26
					Bisquare	4	0.64	0.28
Sinc 196	196	45	0.16	0.17	Suykens	6	1.09	0.30
					Huber	6	1.06	0.36
					Bisquare	5	0.92	0.39
Sinc 211	211	60	0.22	0.20	Suykens	8	1.00	0.41
					Huber	6	1.17	0.44
					Bisquare	6	1.16	0.52
Sinc 226	226	75	0.24	0.25	Suykens	6	1.66	0.42
					Huber	5	1.42	0.48
					Bisquare	7	1.88	0.63

为了进一步验证算法的有效性，我们还选择源自实际问题的 Motorcycle 数据集[10]、Boston Housing 数据集[19]以及 Abalone 数据集[20]进行测试。Motorcycle 数据集是模拟头部受到撞击后产生的加速度的相关数据，共有 133 个样本，该数据集输入值的采样并不是等距的，且在某些情况下对应的同一个输入值可能有不同的输出值，而且输出值具有异方差性。Boston Housing 数据集包含 506 个样本，每个样本有 13 个输入特征和 1 个输出特征。Abalone 数据集是源自生物学研究领域的一个数据集，它共包含 4177 个样本，每个样本有 7 个输入特征和 1 个输出特征。我们将样本集随机分为 5 等份，其中 4 份用于训练，1 份用于测试，算法测试精度和训练时间分别记录在表 3 和表 4 中。实际数据集上的测试结果与仿真数据集上的结果具有相似的趋势：迭代重加权算法具有远远优于 LS-SVM 的稳健性，而且也具有较 ε-SVR 更好的测试精度。而从训练时间上来看，加速算法大幅度减少了 IRLS-SVM 的运算时间，且在 Suykens 和 Huber 的权重方案下，加速算法能够在增加不太多训练时间的前提下获得更稳健的测试结果。

表 3 实际数据集测试结果：测试精度对比

数据集	ε-SVR (MAE)	LS-SVM (MAE)	IRLS-SVM (MAE)		
			Suykens	Huber	Bisquare
Motorcycle	11.36	12.81	10.71	**10.01**	10.13
Boston Housing	6.72	5.87	3.91	**3.46**	3.83
Abalone	1.54	1.86	1.49	1.47	**1.46**

* MAE 为 mean absolute error，即平均绝对误差

表 4 实际数据集测试结果：训练时间对比

数据集	ε-SVR 训练时间(秒)	LS-SVM 训练时间(秒)	IRLS-SVM			
			权重函数	加权次数	训练时间(秒)	
					未加速算法	加速算法
Motorcycle	0.08	0.08	Suykens	4	0.30	0.11
			Huber	14	0.59	0.33
			Bisquare	11	0.56	0.31
Boston Housing	1.42	1.59	Suykens	7	14.11	3.63
			Huber	11	18.24	6.41
			Bisquare	11	23.42	8.42
Abalone	105.5	191.2	Suykens	19	3632.6	311.2
			Huber	16	3058.2	242.3
			Bisquare	13	2485.4	324.1

结束语 为了提升 LS-SVM 的稳健性，本文对迭代重加权模式下的 LS-SVM(即 IRLS-SVM)进行了深入研究。通过数值推导，获得了求解 IRLS-SVM 的快速算法，该算法能大幅度减少运算复杂度，为迭代重加权扫除了运算时间上的障碍；此外，我们还通过实验研究了迭代重加权中的 3 种常见的加权函数(Suykens 权重函数、Huber 函数和 Bisquare 函数)，发现无论使用哪种加权函数 IRLS-SVM 都能获得较为稳健的学习结果。但 3 种权重函数可能会对快速算法的训练时间造成一定的影响。

从研究结果来看，迭代加权的方式确实可以提高最小二乘支持向量机的稳健性。虽然该方法主要从改进求解过程的角度入手，但其本质是改变了损失函数，即通过迭代加权的处理，IRLS-SVM 的模型实际上已经有异于 LS-SVM 模型，其优化目标中的损失函数不再是简单的 SSE 函数，而是对样本进行了加权之后(在某些权重函数下对异常样本进行了几乎是“抛弃”处理的权重设置之后)的 SSE。这就充分降低了异常样本对算法的影响，使得 IRLS-SVM 抗干扰能力更强、更加稳健。从这个角度来说，IRLS-SVM 稳健性的提高符合文献[6]所阐述的规律。

(下转第 297 页)

经过上述转换，得到可执行模型 OPN。要保证 OPN 模型可执行，还必须对图 5 和图 6 中的转移、位置以及对象进行详细的描述，如转移的动作函数(可根据作战规则模型细化)、各种令牌的传递(可根据信息交换关系细化)等。按照 OPN 建模要求完成相关细节建模后，就可以利用对象 Petri 网仿真环境执行 OPN 模型，进行体系结构的验证评估。

结束语 基于可执行模型的体系结构验证评估可以通过执行来分析验证执行流程，可以分析执行中的相关数据来评估体系结构对需求的满足程度。可执行模型的构建是该验证评估方法的关键。在具体建模中，要根据可执行模型的建模特点和元素，建立体系结构数据与它们之间的关系。因此，选择不同的可执行模型，其建模过程也有所不同。

为了更好地评估体系结构，该方法最好与体系结构综合评估的方法配合使用，综合评估方法，确定评估内容和评估指标，基于可执行模型的验证评估方法分析过程和收集各评估指标的数据，最后利用综合评估方法评估体系结构。

参 考 文 献

[1] Sowa J F, Zachman J A. Extending and Formalizing the Framework for Information Systems Architecture[J]. IBM Systems Journal, 1992, 31(3): 590-616

[2] DoD Architecture Framework Working Group. DoD Architecture Framework Version 1.0 [R]. U.S.: Department of Defense, 2003

[3] Wagenhals L W, Shin I, Kim D, et al. C4ISR Architectures: II. A Structured Analysis Approach for Architecture Design [J]. Systems Engineering, 2000, 3(4): 248-287

[4] Wagenhals L W, Haider S, Levis A H. Synthesizing Executable Models of Object Oriented Architectures [C]// Workshop on Formal Methods Applied to Defence Systems. Adelaide, Australia, 2002

[5] Pawlowski T, Barr P C, Ring S J. Applying Executable Architectures to Support Dynamic Analysis of C2 Systems[C]// 2004 Command and Control Research and Technology Symposium. www.dodccrp.org

[6] Ring S J. An Activity-Based Methodology for Development and Analysis of Integrated DoD Architectures-"The Art of Architecture"[C]// 2004 Command and Control Research and Technology Symposium The Power of Information Age Concepts and Technologies

[7] 罗雪山，等. C3I 系统理论及其[M]. 长沙：国防科技大学出版社，2001

[8] 修胜龙，罗雪山，罗爱民，等. C4ISR 体系结构描述的逻辑和行为验证[J]. 系统工程与电子技术，2005，27(2)：275-279

(上接第 228 页)

参 考 文 献

[1] 张学工. 关于统计学习理论与支持向量机[J]. 自动化学报，2000，26(1)：32-42

[2] Vapnik V. The nature of statistical learning theory [M]. John Wiley & Sons, New York, USA, 1995

[3] Cao L J, Tay F E H. Support vector machine with adaptive parameters in financial time series forecasting[J]. IEEE Transactions on Neural Networks, 2003, 14(6): 1506-1518

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

[5] Huber P J. Robust Statistics [M]. John Wiley & Sons Inc., 1981

[6] Chirstmann A, Messem A V. Bouligand Derivatives and Robsutness of Support Vector Machines for Regression[J]. Journal of Machine Learning Research, 2008, 9: 915-936

[7] Smola A J, Scholkopf B. A Tutorial on Support Vector Regression[J]. Statistics and computing, 2004, 3(14): 199-222

[8] Steinwart I, Christmann A. How SVMs can Estimate Quantiles and the Median[C]// Advances in Neural Information Processing Systems 2007. 2008: 305-312

[9] Suykens J A K, Vandewa J. Least Squares Support Vector Machine Classifiers[J]. Neural Processing Letters, 1999, 9: 293-300

[10] Suykens J A K, Brabanter J D, Lukas L, et al. Weighted Least Squares Support Vector Machines: Robustness and Sparse Approximation[J]. Neurocomputing, 2002, 48: 85-105

[11] Weisberg S. Applied Linear Regression [M]. 王静龙，梁小筠，李宝慧，译. 应用线性回归. 中国统计出版社，1998：113-121

[12] Bates D M, Donald G W. Nonlinear Regression Analysis and Its Applications [M]. 韦博成，万方焕，朱宏图，译. 北京：中国统计出版社，1996

[13] Chuang C C, Su S F, Jeng J T, et al. Robust Support Vector Regression Networks for Function Approximation with Outliers [J]. IEEE Trans. on Neural Networks, 2002, 13(6): 1322-1330

[14] Zhang J S, Guo G. Reweighted Robust Support Vector Regression Method[J]. Chinese Journal of Computers, 2005, 28(7): 1171-1178

[15] LJung L. System Identification: Theory for the User(2nd Edition). (影印版)[M]. 北京：清华大学，2002

[16] Kutner M, Nachtsheim C, Neter J. Applied Linear Regression Models (The fourth edition)[M]. McGraw-Hill Higher Education, 2004: 437-445

[17] Rousseeuw P J, Leroy A. Robust Regression and Outlier Detection[J]. John & Wiley Inc., New York, 1987: 9-11

[18] Chang C C, Lin C J. LIBSVM: a library for support vector machines, 2001. Software[OL]. http://www.csie.ntu.edu.tw/~cjlin/libsvm

[19] Boston Housing Data Set[OL]. http://archive.ics.uci.edu/ml/datasets/Housing

[20] Abalone Data set [OL]. http://archive.ics.uci.edu/ml/datasets/Abalone