文章编号: 1001-0920 (2014) 01-0050-07 DOI: 10.13195/j.kzyjc.2012.1791

基于递归约简的在线自适应最小二乘支持向量回归机

刘毅男, 张胜修, 张 超

(第二炮兵工程大学 自动控制工程系，西安 710025)

摘 要: 鉴于传统在线最小二乘支持向量机在解决时变对象的回归问题时, 模型跟踪精度不高, 支持向量不够稀疏, 结合迭代策略和约简技术, 提出一种在线自适应迭代约简最小二乘支持向量机. 该方法考虑新增样本与历史数据共同作用对现有模型产生的约束影响, 寻求对目标函数贡献最大的样本作为新增支持向量, 实现了支持向量稀疏化, 提高了在线预测精度与速度. 仿真对比分析表明该方法可行有效, 较传统方法回归精度高且所需支持向量数目最少.

关键词: 最小二乘支持向量回归机；在线；自适应；迭代策略；约简技术

中图分类号: TP273 **文献标志码:** A

Online adaptive least squares support vector regression based on recursion and reduction

LIU Yi-nan, *ZHANG Sheng-xiu*, *ZHANG Chao*

(Department of Automatic Control Engineering，The Second Artillery Engineering University，Xi'an 710025，China. Correspondent：LIU Yi-nan，E-mail：lyn8307@gmail.com)

Abstract: The tracking accuracy of the traditional online least squares support vector regression in solving regression problem of the time-varying objects is not high enough and support vectors are not sparse. To deal with this problem, an online adaptive recursive reduced least squares support vector regression is proposed by combining with the iterative strategy and reduced technique. The method considers the constrainable impact on the existing model, which is caused by the joint action of new samples and historical data. Meantime, the training sample leading to the largest reduction in the target function is chosen as the best new support vectors. Then the regression model is simplified, and the prediction time is shortened. Finally, simulation analysis illustrates the effectiveness and feasibility of the presented method. Compared with the traditional algorithms, the method is more accurate and sparse.

Key words: least squares support vector regression；online；adaptive；iterative strategy；reduced technique

0 引 言

支持向量机(SVM)自20世纪90年代由Vapnik等[1]提出以来, 受到广泛重视. SVM将问题归结为求解凸二次优化QP(quadratic programming), 根据有限样本信息在模型复杂度和学习能力之间寻求最佳折衷, 在小样本学习问题上表现尤为出色. 但随着信息化时代的到来, 数据量空前增大且日趋复杂, 大规模数据集的学习问题已成为SVM的应用瓶颈. 尽管人们提出了许多快速训练算法, 如Chunking算法[2]、SVMlight[3]、SVMTorch[4]、SMO[5]等, 却难以满足较高实时性的要求. 为此, Suykens等[6]提出了最小二乘支持向量机(LSSVM), 将SVM的训练转化为线性方程组的求解, 一定程度上加速了训练过程, 但丧失了解的稀疏性与鲁棒性. 针对此问题, 人们引入样本加权[7-8]、剪枝算法[9]等方法进行弥补, 但优化计算在一定程度上忽略了部分向量的约束作用. 在解决回归问题的最小二乘支持向量回归机(LSSVR)研究中, Zhao等结合约简技术及迭代策略, 提出了递归约简最小二乘支持向量回归机(RR-LSSVR)[10], 并进行了改进[11], 该算法在考虑全部样本产生的约束条件下进行优化计算, 同时, 较其他算法在同等泛化性能下解更具稀疏性, 实时性更好.

上述研究均是离线训练算法, 在实际应用中, 更多的研究对象具有时变特性, 即学习样本会随着时

收稿日期: 2012-12-01；**修回日期:** 2013-02-27.

基金项目: 国家863计划项目(2011AA7053016).

作者简介: 刘毅男(1983−), 男, 博士生, 从事智能控制、飞行器控制的研究；张胜修(1963−), 男, 教授, 博士生导师, 从事飞行器控制、仿真与决策等研究.

间推移发生变化, 这对SVM的在线学习和自适应能力提出了新的挑战. 近年来, 在离线算法的研究基础上, 一些学者提出了不同的在线式学习算法, 如Cauwenberghs等[12]提出的增量与减量SVM, Wang等[13]提出的快速在线SVR等, 但因对增减量数据的判断更新过程复杂, 算法实时性不高. 此外, 张浩然等[14]提出了LSSVM在线式学习算法, 张淑宁等[15-16]提出了在线鲁棒LSSVR, 算法实时性得以改进, 但在模型新增样本时, 忽略了历史数据对模型的约束影响.

针对上述问题, 本文结合RR-LSSVR的迭代约简思想, 提出在线自适应RR-LSSVR. 在获取研究对象初始样本集的基础上, 对新增样本进一步迭代计算, 结合历史数据寻求最优支持向量, 即在线学习的同时, 考虑新增样本与历史数据共同作用对现有模型产生的约束影响, 既保持了解的稀疏性, 又增强了训练模型的在线自适应能力. 仿真研究表明了所提出方法的可行性和有效性.

1　最小二乘支持向量机

对于一个给定的训练数据集$\{(x_i,d_i)\}_{i=1}^N$, 其中$x_i\in\boldsymbol{R}^m$为系统输入, $d_i\in\boldsymbol{R}$为系统输出, 则LSSVR可以描述为约束优化问题

$$\min_{\omega,e} J(\omega,e)=\frac{1}{2}\omega^{\mathrm{T}}\omega+\frac{\gamma}{2}\sum_{i=1}^{N}e_i^2\gamma>0; \tag{1}$$

$$\text{s.t. } d_i=\omega^{\mathrm{T}}\phi(x_i)+b+e_i,$$
$$i=1,2,\cdots,N. \tag{2}$$

其中: $\gamma>0$为正则化参数, b为偏置量, e_i为第i个数据实际输出与预测输出间的误差, ω为超平面的法向量, $\phi(\cdot)$为由输入空间到特征空间的映射. 由此构建优化问题的拉格朗日函数

$$L(\omega,b,e;\alpha)=$$
$$\frac{1}{2}\omega^{\mathrm{T}}\omega+\frac{\gamma}{2}\sum_{i=1}^{N}\alpha_i\{d_i-\omega^{\mathrm{T}}\phi(x_i)-b-e_i\}, \tag{3}$$

其中$\alpha_i\in\boldsymbol{R}(i=1,2,\cdots,N)$为拉格朗日乘子. 根据KKT条件, 有

$$\begin{cases}\dfrac{\partial L}{\partial\omega}=0\to\omega=\displaystyle\sum_{i=1}^{N}\alpha_i\phi(x_i),\\ \dfrac{\partial L}{\partial b}=0\to\displaystyle\sum_{i=1}^{N}\alpha_i=0,\\ \dfrac{\partial L}{\partial e_i}=0\to\alpha_i=\gamma e_i,\\ \dfrac{\partial L}{\partial\alpha_i}=0\to\omega^{\mathrm{T}}\phi(x_i)+b+e_i-d_i=0.\end{cases} \tag{4}$$

消去向量ω和e, 化简为

$$\begin{bmatrix}\mathbf{0} & \mathbf{1}^{\mathrm{T}}\\ \mathbf{1} & K\end{bmatrix}\begin{bmatrix}b\\ \alpha\end{bmatrix}=\begin{bmatrix}\mathbf{0}\\ d\end{bmatrix}. \tag{5}$$

其中

$$\mathbf{1}=[1_1,1_2,\cdots,1_N]^{\mathrm{T}},$$
$$d=[d_1,d_2,\cdots,d_N]^{\mathrm{T}}.$$
$$K_{ij}=\kappa(x_i,x_j)=\phi(x_i)^{\mathrm{T}}\phi(x_j)+\delta_{ij}/\gamma.$$
$$\delta_{ij}=\begin{cases}1,\ i=j;\\ 0,\ i\neq j;\end{cases}$$
$$i,j=1,2,\cdots,N.$$

$\kappa(\cdot,\cdot)$为核函数. 当由式(4)求出α和b后, 对于某一测试输入, 可通过下式预测其对应输出:

$$f(x)=\sum_{i=1}^{N}\alpha_i\kappa(x_i,x)+b. \tag{6}$$

2　基于递归约简的LSSVR

由式(4)中的$\alpha_i=\gamma e_i$可知, 若训练误差不为零, 则α_i不为零, 每个样本都成为支持向量, 造成大规模数据时算法实时性变差. 应用约简技术, 假定训练样本集中子集S的样本为支持向量, 其他非支持向量对应的$\alpha_i=0$, 则$\omega=\displaystyle\sum_{i\in S}\alpha_i\kappa(x_i,\cdot)$. 将其代入式(1)得

$$\min\Big\{L(b,\alpha_S)=\frac{1}{2}\alpha_S^{\mathrm{T}}K\alpha_S+\frac{\gamma}{2}\sum_{i=1}^{N}\Big(d_i-$$
$$\sum_{j\in S}\alpha_j\phi(x_j)^{\mathrm{T}}\phi(x_i)-b\Big)^2\Big\}, \tag{7}$$

其中$K_{ij}=\kappa(x_i,x_j),\ i,j\in S$. 根据文献[17]的推证, 将式(7)用矩阵形式表示为

$$\min\Big\{L=[b\ \ \alpha_S^{\mathrm{T}}]\Big(\begin{bmatrix}\mathbf{0} & \mathbf{0}^{\mathrm{T}}\\ \mathbf{0} & K/\gamma\end{bmatrix}+$$
$$\begin{bmatrix}\mathbf{1}^{\mathrm{T}}\\ \hat{K}\end{bmatrix}[\mathbf{1}\ \ \hat{K}^{\mathrm{T}}]\Big)\begin{bmatrix}b\\ \alpha_S\end{bmatrix}-$$
$$2\begin{bmatrix}b\\ \alpha_S\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{1}^{\mathrm{T}}\\ \hat{K}\end{bmatrix}d\Big\}. \tag{8}$$

其中: $\hat{K}_{ij}=\kappa(x_i,x_j),\ i\in S,\ j\in\{1,2,\cdots,N\}$; $\mathbf{1}$为适维单位向量, $\mathbf{0}$为适维零向量. 取

$$\frac{\partial L}{\partial b}=0,\ \frac{\partial L}{\partial\alpha_S}=0,$$

得到式(8)的最优解表达式

$$(R+ZZ^{\mathrm{T}})\begin{bmatrix}b\\ \alpha_S\end{bmatrix}=Zd. \tag{9}$$

其中

$$R=\begin{bmatrix}\mathbf{0} & \mathbf{0}^{\mathrm{T}}\\ \mathbf{0} & K/\gamma\end{bmatrix},\ Z=[1^{\mathrm{T}}\ \ \hat{K}]^{\mathrm{T}}.$$

如果$R+ZZ^{\mathrm{T}}$奇异, 则可对其进行正则化处理, 得到约简LSSVR

$$f(x)=\sum_{i\in S}\alpha_i\kappa(x_i,x)+b. \tag{10}$$

可见, 约简LSSVR的解具有稀疏性.

至此, 如何从含有海量数据的训练样本集中选定具有代表意义的子集S即成为首先要解决的问题. 显然组合式的穷举搜索不具实时性, 自由选择又无法保证所选样本为最优支持向量. 迭代策略较好地解决了这一问题, 如果在第n步迭代环节中已选定支持向量个数为$|S|$, 则式(8)的最优解可由下式获得:

$$\left(\begin{bmatrix}\mathbf{0} & \mathbf{0}^{\mathrm{T}}\\ \mathbf{0} & K/\gamma\end{bmatrix}+\begin{bmatrix}\mathbf{1}^{\mathrm{T}}\\ \hat{K}\end{bmatrix}[\mathbf{1}\ \ \hat{K}^{\mathrm{T}}]\right)\begin{bmatrix}b^n\\ \alpha_S^n\end{bmatrix}=\begin{bmatrix}\mathbf{1}^{\mathrm{T}}\\ \hat{K}\end{bmatrix}d, \tag{11}$$

即

$$\left(\begin{bmatrix}\mathbf{0} & \mathbf{0}^{\mathrm{T}}\\ \mathbf{0} & K/\gamma\end{bmatrix}+\begin{bmatrix}N & \mathbf{1}^{\mathrm{T}}\hat{K}^{\mathrm{T}}\\ \hat{K}\mathbf{1} & \hat{K}\hat{K}^{\mathrm{T}}\end{bmatrix}\right)\begin{bmatrix}b^n\\ \alpha_S^n\end{bmatrix}=\begin{bmatrix}\mathbf{1}^{\mathrm{T}}\\ \hat{K}\end{bmatrix}d. \tag{12}$$

其中: α_S为由支持向量集S所决定的α子向量, $\mathbf{1}$和$\mathbf{0}$分别为适维全1或全0向量, 上标n表示第n步迭代时相应变量的取值. 将式(12)的解带入式(8)得

$$L^n=-\begin{bmatrix}b^n\\ \alpha_S^n\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{1}^{\mathrm{T}}\\ \hat{K}\end{bmatrix}d. \tag{13}$$

其中

$$\begin{bmatrix}b^n\\ \alpha_S^n\end{bmatrix}=U^n\begin{bmatrix}\mathbf{1}^{\mathrm{T}}\\ \hat{K}\end{bmatrix}d,$$

$$U^n=\left(\begin{bmatrix}\mathbf{0} & \mathbf{0}^{\mathrm{T}}\\ \mathbf{0} & K/\gamma\end{bmatrix}+\begin{bmatrix}N & \mathbf{1}^{\mathrm{T}}\hat{K}^{\mathrm{T}}\\ \hat{K}\mathbf{1} & \hat{K}\hat{K}^{\mathrm{T}}\end{bmatrix}\right)^{-1}. \tag{14}$$

设在第$n+1$步迭代中, 确定x_q为支持向量, 则U^{n+1}较U^n增加关于x_q的核函数表达行和列, 表示为

$$U^{n+1}=\left(\begin{bmatrix}\mathbf{0} & \mathbf{0}^{\mathrm{T}} & \mathbf{0}\\ \mathbf{0} & K/\gamma & \kappa_q/\gamma\\ \mathbf{0} & \kappa_q^{\mathrm{T}}/\gamma & \kappa_{qq}/\gamma\end{bmatrix}+\begin{bmatrix}N & \mathbf{1}^{\mathrm{T}}\hat{K}^{\mathrm{T}} & \mathbf{1}^{\mathrm{T}}\hat{\kappa}_q\\ \hat{K}\mathbf{1} & \hat{K}\hat{K}^{\mathrm{T}} & \hat{K}\hat{\kappa}_q\\ \kappa_q^{\mathrm{T}}\mathbf{1} & \kappa_q^{\mathrm{T}}\hat{K}^{\mathrm{T}} & \hat{\kappa}_q^{\mathrm{T}}\hat{\kappa}_q\end{bmatrix}\right)^{-1}. \tag{15}$$

其中

$$\kappa_q=\begin{bmatrix}\kappa(x_q,x_i)\\ \vdots\\ \kappa(x_q,x_j)\end{bmatrix},\ i,\cdots,j\in S;$$

$$\hat{\kappa}_q=\begin{bmatrix}\kappa(x_q,x_1)\\ \vdots\\ \kappa(x_q,x_i)\end{bmatrix},\ i=1,2,\cdots,N.$$

假设U^n已于第n步迭代中得到, 根据Sherman-Morrison公式, U^{n+1}可表示为关于U^n的函数

$$U^{n+1}=\begin{bmatrix}U^n & \mathbf{0}\\ \mathbf{0}^{\mathrm{T}} & 0\end{bmatrix}+\lambda\begin{bmatrix}\beta\\ -\mathbf{1}\end{bmatrix}[\beta^{\mathrm{T}}\ \ -\mathbf{1}]. \tag{16}$$

其中

$$\beta=U^n\begin{bmatrix}\mathbf{1}^{\mathrm{T}}\hat{\kappa}_q\\ \kappa_q/\gamma+\hat{K}\hat{\kappa}_q\end{bmatrix}, \tag{17}$$

$$\lambda=(\kappa_{qq}/\gamma+\hat{\kappa}_q^{\mathrm{T}}\hat{\kappa}_q-[\hat{\kappa}_q^{\mathrm{T}}\mathbf{1}\ \ \hat{\kappa}_q^{\mathrm{T}}\hat{K}^{\mathrm{T}}+\hat{\kappa}_q^{\mathrm{T}}/\gamma]\beta)^{-1}. \tag{18}$$

根据式(16), 求得第$n+1$步迭代中α和b的值为

$$\begin{bmatrix}b^{n+1}\\ \alpha_S^{n+1}\\ \alpha_q^{n+1}\end{bmatrix}=U^{n+1}\begin{bmatrix}\hat{\mathbf{0}}\\ \hat{d}_S\\ \hat{d}_q\end{bmatrix}=\begin{bmatrix}U^n\begin{bmatrix}\hat{\mathbf{0}}\\ \hat{d}_S\end{bmatrix}\\ \mathbf{0}\end{bmatrix}+\lambda\left(\beta^{\mathrm{T}}\begin{bmatrix}\hat{\mathbf{0}}\\ \hat{d}_S\end{bmatrix}-\hat{d}_q\right)\begin{bmatrix}\beta\\ -\mathbf{1}\end{bmatrix}. \tag{19}$$

其中: $\hat{\mathbf{0}}=\sum_{i=1}^{N}d_i$, $\hat{d}_S=\hat{K}d$. 将式(14)代入(19)并化简可得

$$\begin{bmatrix}b^{n+1}\\ \alpha_S^{n+1}\\ \alpha_q^{n+1}\end{bmatrix}=\begin{bmatrix}b^n\\ \alpha_S^n\\ \mathbf{0}\end{bmatrix}+\lambda\left(\beta^{\mathrm{T}}\begin{bmatrix}\hat{\mathbf{0}}\\ \hat{d}_S\end{bmatrix}-\hat{d}_q\right)\begin{bmatrix}\beta\\ -\mathbf{1}\end{bmatrix}. \tag{20}$$

将式(20)代入(8)可得目标函数的更新方程为

$$L_q^{n+1}=L^n-\lambda\left(\beta^{\mathrm{T}}\begin{bmatrix}\hat{\mathbf{0}}\\ \hat{d}_S\end{bmatrix}-\hat{d}_q\right)^2. \tag{21}$$

因此, 定义评价函数为

$$\xi_q^{n+1}=L^n-L_q^{n+1}, \tag{22}$$

等价于

$$\xi_q^{n+1}=\lambda\left(\beta^{\mathrm{T}}\begin{bmatrix}\hat{\mathbf{0}}\\ \hat{d}_S\end{bmatrix}-\hat{d}_q\right)^2. \tag{23}$$

由以上推导可知, ξ_q^{n+1}为一个正值, 计算非支持向量集合中关于各元素$x_i(i=\{1,2,\cdots,N\}\backslash S)$的$\xi_i^{n+1}$值, 若所得$\xi_i^{n+1}$越大, 则$L_i^{n+1}$越小, 即对应样本$x_i$对式(8)所表示约束优化问题的贡献越大. 这样, 将之前未被选作支持向量的样本按照各自评价值的大小排序, 选取评价值最大的样本作为下一个支持向量, 得到支持向量的选择标准为

$$q=\underset{i\in\{1,2,\cdots,N\}\backslash S}{\arg\max}\ \xi_i^{n+1}. \tag{24}$$

迭代由$n=0$开始, 至所选支持向量达到预先定义的正整数$M(M\ll N)$, 即$n=M$时停止, 这样, 初始回归模型实现了考虑全部样本约束下的支持向量稀疏且最优, 实现了回归模型初始化.

3 基于递归约简的在线自适应LSSVR

现实中, 绝大多数的研究对象都具有时变特性, 因为研究对象参数变化或初始建模样本不完备等原

因导致的回归模型老化, 需要学习算法具有在线自适应调节的能力, 所以, 在初始回归模型基础上, 基于迭代策略和约简技术, 实现模型在线更新.

考虑包含历史数据和新加入数据共同对回归模型产生约束, 保留初始样本集. 每次更新加入一个新采集样本, 记为 (x_{N+1}, d_{N+1}), 当前模型支持向量数目为 M, 则式 (22) 评价函数变为

$$\xi_q^{N+1} = L^N - L_q^{N+1}, \tag{25}$$

等价于

$$\xi_q^{N+1} = \lambda\Big(\beta^{\mathrm{T}} \begin{bmatrix} \hat{\mathbf{0}} \\ \hat{d}_S \end{bmatrix} - \hat{d}_q\Big)^2. \tag{26}$$

新采集样本与未选作支持向量的样本一起, 按照标准

$$q = \underset{i\in\{1,2,\cdots,N+1\}\setminus S}{\arg\max} \xi_i^{N+1} \tag{27}$$

选定新的支持向量. 注意到, 在计算过程中, $\hat{K}_{ij}$ 和 $\hat{\kappa}_q$ 的取值范围发生变化, 分别为

$$\hat{K}_{ij} = \kappa(x_i, x_j),\ i \in S,\ j \in \{1, 2, \cdots, N+1\};$$

$$\hat{\kappa}_q = \begin{bmatrix} \kappa(x_q, x_1) \\ \vdots \\ \kappa(x_q, x_i) \end{bmatrix},\ i = 1, 2, \cdots, N+1.$$

至此, 新加入的支持向量已确定, 存储样本总数为 $N+1$, 为了避免由于数据不断增多而导致的模型复杂化和存储空间消耗, 需要对历史数据进行剔除. 采用窗式移动的方式进行样本更新, 保持支持向量数目始终为 M, 存储样本总数始终为 N. 其依据是考虑到研究对象的时变特性, 新增支持向量是结合新采集样本和历史数据综合确定的, 能够反映研究对象的最新变化, 而最早确定的历史支持向量随着时间推移对时变对象回归模型的贡献度降低, 因此予以剔除.

首先, 将初始回归方程表示为如式 (5) 所示的形式, 其内部元素个数和内容有所变化, 表示为

$$\mathbf{1} = [1_{S_1}, 1_{S_2}, \cdots, 1_{S_N}]^{\mathrm{T}},$$
$$d = [d_{S_1}, d_{S_2}, \cdots, d_{S_N}]^{\mathrm{T}}.$$

核函数矩阵为

$$Q = \begin{bmatrix} \kappa(x_{S_1}, x_{S_1}) & \cdots & \kappa(x_{S_M}, x_{S_1}) \\ \vdots & \ddots & \vdots \\ \kappa(x_{S_1}, x_{S_M}) & \cdots & \kappa(x_{S_M}, x_{S_M}) \end{bmatrix},$$

则有

$$K = Q + \delta_{ij}/\gamma = \begin{bmatrix} \kappa(x_{S_1}, x_{S_1}) + 1/\gamma & \cdots & \kappa(x_{S_M}, x_{S_1}) \\ \vdots & \ddots & \vdots \\ \kappa(x_{S_1}, x_{S_M}) & \cdots & \kappa(x_{S_M}, x_{S_M}) + 1/\gamma \end{bmatrix}.$$

由式 (5) 可求得

$$b = \frac{\mathbf{1}^{\mathrm{T}} P d}{\mathbf{1}^{\mathrm{T}} P \mathbf{1}}, \tag{28}$$

$$\alpha = P\Big(d - \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}} P d}{\mathbf{1}^{\mathrm{T}} P \mathbf{1}}\Big), \tag{29}$$

其中 $P = K^{-1}$. 将矩阵 K 分块, 表示为

$$K = \begin{bmatrix} f & F^{\mathrm{T}} \\ F & W \end{bmatrix}.$$

其中

$$f = \kappa(x_{S_1}, x_{S_1}) + 1/\gamma,$$
$$F = [\kappa(x_{S_1}, x_{S_2}), \kappa(x_{S_1}, x_{S_3}), \cdots, \kappa(x_{S_1}, x_{S_M})]^{\mathrm{T}},$$
$$W = \begin{bmatrix} \kappa(x_{S_2}, x_{S_2}) + 1/\gamma & \cdots & \kappa(x_{S_M}, x_{S_2}) \\ \vdots & \ddots & \vdots \\ \kappa(x_{S_2}, x_{S_M}) & \cdots & \kappa(x_{S_M}, x_{S_M}) + 1/\gamma \end{bmatrix}.$$

在下一个时刻, 新选定的支持向量 $(x_{S_{M+1}}, d_{S_{M+1}})$ 加入, 旧样本 (x_{S_1}, d_{S_1}) 被抛弃, 核函数矩阵变化为

$$Q' = \begin{bmatrix} \kappa(x_{S_2}, x_{S_2}) & \cdots & \kappa(x_{S_{M+1}}, x_{S_2}) \\ \vdots & \ddots & \vdots \\ \kappa(x_{S_2}, x_{S_{M+1}}) & \cdots & \kappa(x_{S_{M+1}}, x_{S_{M+1}}) \end{bmatrix},$$

矩阵 K 相应变化为 K', 即

$$K' = Q' + \delta_{ij}/\gamma = \begin{bmatrix} \kappa(x_{S_2}, x_{S_2}) + 1/\gamma & \cdots & \kappa(x_{S_{M+1}}, x_{S_2}) \\ \vdots & \ddots & \vdots \\ \kappa(x_{S_2}, x_{S_{M+1}}) & \cdots & \kappa(x_{S_{M+1}}, x_{S_{M+1}}) + 1/\gamma \end{bmatrix} = \begin{bmatrix} W & V \\ V & v \end{bmatrix}.$$

其中

$$v = \kappa(x_{S_{M+1}}, x_{S_{M+1}}) + 1/\gamma,$$
$$V = [\kappa(x_{S_{M+1}}, x_{S_2}), \kappa(x_{S_{M+1}}, x_{S_3}), \cdots, \kappa(x_{S_{M+1}}, x_{S_M})]^{\mathrm{T}}.$$

由分块矩阵的计算方法, 可得

$$P = K^{-1} = \begin{bmatrix} f & F^{\mathrm{T}} \\ F & W \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & W^{-1} \end{bmatrix} + rr^{\mathrm{T}} z, \tag{30}$$

$$P' = K'^{-1} = \begin{bmatrix} W & V^{\mathrm{T}} \\ V & v \end{bmatrix}^{-1} = \begin{bmatrix} W^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + r' r'^{\mathrm{T}} z'. \tag{31}$$

其中

$$r = (-1, F^{\mathrm{T}} W^{-1})^{\mathrm{T}},\ z = \frac{1}{f - F^{\mathrm{T}} W^{-1} F}; \tag{32}$$

$$r' = (V^{\mathrm{T}} W^{-1}, -1)^{\mathrm{T}},\ z' = \frac{1}{v + V^{\mathrm{T}} W^{-1} V}; \tag{33}$$

f 为由于样本集窗式移动由所删除样本构成的核函数元素和向量; v 为由于样本集窗式移动由所增加样本形成的核函数元素和向量. 由式 (31) 可以看出, 求得下一时刻的 $P'=K'^{-1}$ 关键是求出 W^{-1}. 将式 (32) 代入 (30), 可得

$$P=\begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & W^{-1}\end{bmatrix}+\begin{bmatrix}\mathbf{1}/z & -F^{\mathrm{T}}W^{-1}/z\\ -W^{-1}F/z & -W^{-1}FF^{\mathrm{T}}W^{-1}/z\end{bmatrix}. \tag{34}$$

因为前一时刻 $P=K^{-1}$ 已知, 可由 P 阵第 1 个元素 P_{11} 求得 $1/z$, 并根据其他行列元素推得 W^{-1}, 然后根据式 (31) 求得下一时刻的 $P'=K'^{-1}$, 完成递推计算. 递推计算避免了直接求逆, 大大减小了算法计算复杂度, 提高了实时性. 利用计算结果更新回归模型的参数值 α, 实现模型在线自适应更新. 此外, 在存储样本集中找出对应于 S_1 的样本并剔除, 保持样本存储总数为 N.

综合以上推证, 在线自适应 RR-LSSVR 的实现过程归纳如下, 其中 Step 1 ∼ Step 4 为回归模型初始化, Step 5 ∼ Step 9 为在线自适应 RR-LSSVR.

Step 1: 令 $\alpha^0=0,\ b^0=0, S=\varnothing,\ T=\{1,2,\cdots,N\}$, 取正整数 M, $n=0$.

Step 2: 若 $n>M$ 则停止, 否则转至 Step 3.

Step 3: 若 $n=0$, 则在样本集 T 中根据式 (8) 求得 q, 计算

$$U^{n+1}=\left(\begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \kappa_{qq}/\gamma\end{bmatrix}+\begin{bmatrix}N & \mathbf{1}^{\mathrm{T}}\hat{\kappa}_q\\ \hat{\kappa}_q^{\mathrm{T}}\mathbf{1} & \hat{\kappa}_q^{\mathrm{T}}\hat{\kappa}_q\end{bmatrix}\right)^{-1},$$

$$\begin{bmatrix}b^{n+1}\\ \alpha_q^{n+1}\end{bmatrix}=U^{n+1}\begin{bmatrix}\hat{\mathbf{0}}\\ \hat{d}_q\end{bmatrix};$$

否则由式 (24) 获得样本 x_q, 并通过式 (16) 和 (20) 计算 U^{n+1}, α_S^{n+1}, α_q^{n+1} 和 b^{n+1}.

Step 4: 令 $S=S\bigcup\{q\},\ P=P\backslash\{q\},\ n=n+1$, 转至 Step 2.

Step 5: 采集新样本与之前未选作支持向量的样本构成备选样本集合, 按照式 (27) 获得新支持向量 (x_q,d_q).

Step 6: 根据选定的新加入支持向量和需剔除的旧支持向量, 按照式 (34) 计算求得 W^{-1}.

Step 7: 应用新选支持向量, 根据式 (31) 求得 P'.

Step 8: 更新回归模型的参数值 α.

Step 9: 在存储样本集中找出对应于 S_1 的样本并剔除, 转至 Step 5.

4 仿真分析

通过仿真分析, 验证基于迭代约简的在线自适应 LSSVR 的可行性和有效性. 实验环境为: Intel i5-2320(@3GHz) 处理器, 3G 内存, Windows XP 操作系统的计算机, 在 Matlab2007a 软件平台上完成程序设计及仿真. 仿真中, 将本文方法与在线 SVR[13]、经典在线 LSSVR[14]、在线鲁棒 LSSVR[16]进行比较, 采用对研究对象模型静态采样、动态依次单个注入样本的方式模拟在线训练. 因为本文提出的在线自适应 RR-LSSVR 需要前期样本进行共同约束建模, 从训练样本中随机抽取部分作为在线自适应 RR-LSSVR 前期回归数据, 然后对 4 种方法的回归效果进行比较. 在实际生产生活中, 针对某一研究对象的前期数据获取一般是可行的, 因此, 仿真中抽取同一研究对象的部分同批次样本作为初始回归数据, 不影响实验比较结果.

仿真实验选用 3 组 benchmark 数据集, 分别为 total_UPDRS, winequality_red, winequality_white. 为了便于分析仿真结果, 定义评估测试指标均方根误差如下所示:

$$\mathrm{RMSE}=\sqrt{\sum_{i=1}^{N}(\hat{d}_i-d_i)^2\Big/N}. \tag{35}$$

核函数选用高斯核函数

$$\kappa(x_i,x_j)=\exp\Big(-\frac{\|x_i-x_j\|}{2\sigma^2}\Big). \tag{36}$$

仿真验证前, 将每组数据集的输入数据进行归一化处理, 同时, 基于经典离线 SVR 和 LSSVR, 在区间

$$\{2^{-10},2^{-9},\cdots,2^{10}\}\times\{2^{-5},2^{-4},\cdots,2^{5}\}$$

上采用留一法交叉验证获取模型最优参数值 (γ^*,σ^*), 代入到 4 种在线回归方法中. 随机抽取测试样本的 20% 作为在线自适应 RR-LSSVR 的初始样本数据, 3 种在线 LSSVR 方法设定支持向量数目为训练样本总数的 8%.

4 种方法在 3 个数据集上的参数设置和训练结果如表 1 所示. 由于在线 SVR 根据 KKT 条件判断不断调整支持向量和非支持向量的划分, 不能通过设定支持向量数目的方式与在线 LSSVR 方法比较预测精度的变化趋势. 因此, 图 1 ∼ 图 3 仅给出 3 种在线 LSSVR 方法回归精度随支持向量数目的变化趋势.

由表 1 可见: 在 3 个数据集上开展仿真实验, 在线自适应 RR-LSSVR 测试精度明显高于其他 3 种方法, 基本接近于全样本的 LSSVR 离线回归精度; 在线 SVR 的测试精度仅高于经典在线 LSSVR; 在线鲁棒 LSSVR 由于具有鲁棒判别策略, 测试精度高于在线 SVR 与经典在线 LSSVR. 由图 1 ∼ 图 3 可见, 在训练

表 1　仿真结果对比

数据集	参数设定	方法	训练样本数	测试样本数	RMSE	训练时间/s
total_UPDRS	$C^*=8,\varepsilon^*=0.01,\sigma^*=0.02$	在线 SVR	3 000	2 875	9.475 7	25.327 5
	$\gamma^*=32,\ \sigma^*=0.25$	离线 LSSVR	3 000	2 875	8.563 6	14.682 1
		经典在线 LSSVR	3 000	2 875	9.968 2	22.364 8
		在线鲁棒 LSSVR	3 000	2 875	9.321 4	28.846 1
		在线自适应 RR-LSSVR	3 000	2 875	8.578 0	29.162 2
total_UPDRS	$C^*=4.8,\varepsilon^*=0.01,\sigma^*=0.025$	在线 SVR	1 000	599	6.399 3e-01	8.103 8
	$\gamma^*=32,\ \sigma^*=0.5$	离线 LSSVR	1 000	599	6.195 8e-01	2.160 4
		经典在线 LSSVR	1 000	599	7.636 2e-01	6.165 3
		在线鲁棒 LSSVR	1 000	599	6.365 4e-01	11.302 6
		在线自适应 RR-LSSVR	1 000	599	6.188 2e-01	11.741 9
total_UPDRS	$C^*=9.68,\varepsilon^*=0.01,\sigma^*=0.05$	在线 SVR	3 500	1 398	7.602 9e-01	32.472 9
	$\gamma^*=64,\ \sigma^*=1$	离线 LSSVR	3 500	1 398	7.159 8e-01	19.813 9
		经典在线 LSSVR	3 500	1 398	8.120 5e-01	28.684 0
		在线鲁棒 LSSVR	3 500	1 398	7.523 9e-01	35.594 3
		在线自适应 RR-LSSVR	3 500	1 398	7.163 5e-01	36.217 3

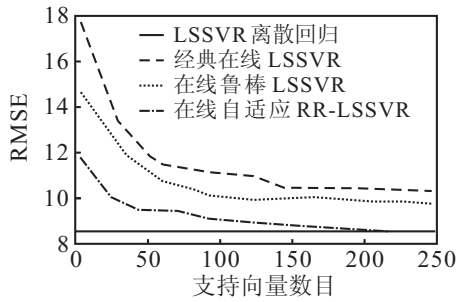

图 1　total_UPDRS 对比仿真

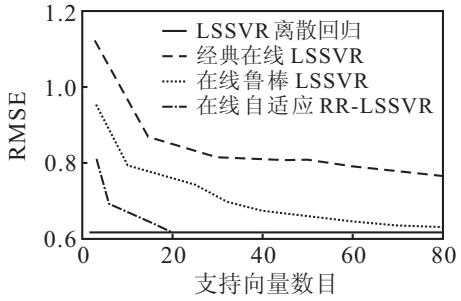

图 2　winequality_red 对比仿真

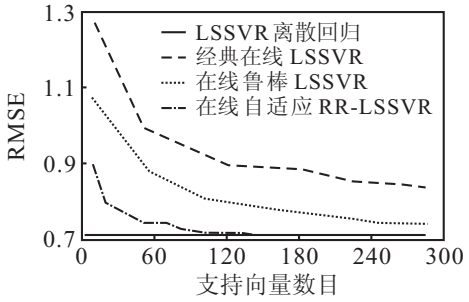

图 3　winequality_white 对比仿真

过程中, 在线自适应 RR-LSSVR 达到或接近于全样本的 LSSVR 离线回归精度明显需要支持向量更少, 3 个数据集仿真中实际需要支持向量数目均少于初始设定训练样本总数的 8%, total_UPDRS 约为 218, winequality_red 约为 24, winequality_white 约为 136, 这表明实验之初设定的支持向量最大数目对在线自适应 RR-LSSVR 而言仍有大幅减少的空间, 这样, 在线自适应 RR-LSSVR 的模型复杂度较其他两种在线 LSSVR 方法会减小, 预测速度会更快. 此外, 虽然仿真实验中在线自适应 RR-LSSVR 迭代计算所花费时间稍长, 但通过控制支持向量数目和最大存储样本数, 其平均一次数据更新时间控制在毫秒级, 完全能够满足实际应用中对于模型在线回归的要求.

5　结　论

本文结合 RR-LSSVR 的迭代约简思想, 提出了在线自适应 RR-LSSVR. 将新增样本结合历史数据共同对回归模型产生约束, 并在其中寻求最优支持向量, 摒弃了简单的样本一入一出的模型更新方式, 实现了 LSSVR 解的稀疏化, 进而使回归模型得到简化, 缩短了预测时间, 同时, 算法的回归精度较高. 仿真实验和分析验证了所提出方法的可行性和有效性, 取得了较为满意的效果, 为实际应用打下了基础.

参考文献(References)

[1] Vapnik V N. The nature of statistical learning theory[M]. New York: Springer-Verlag Press, 1995: 138-170.

[2] Osuna E, Freund R, Girosi F. An improved training algorithm for support vector machines[C]. Proc of IEEE Signal Processing Society Workshop. New York: IEEE Press, 1997: 276-285.

[3] Joachims T. Making large-scale SVM learning practical[C]. Advances in Kernel Methods: Support Vector Machine. Cambridge: MIT Press, 1999: 169-184.

[4] Collobert R, Bengio S. SVMTorch: Support vector machines for large-scale regression problems[J]. J of Machine Learning Research, 2001, 1(2): 143-160.

[5] Shevade S K, Keerthi S S, Bhattacharyya C, et al. Improvements to the SMO algorithm for SVM regression[J]. IEEE Trans on Neural Networks, 2000, 11(5): 1188-1193.

[6] Suykens J A K, Vandewalle J. Least squares support vector machine classifiers[J]. Neural Processing Letter, 1999, 9(3): 293-300.

[7] 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(1): 85-105.

[8] Shim J, Hwang C, Nau S. Robust LSSVM regression using fuzzy C-means clustering[C]. Proc of the 2nd Int Conf on Natural Computation. Xi'an: Springer Press, 2006, 9: 157-166.

[9] Suykens J A K, Lukas L, Vandewalle J. Sparse approximation using least squares vector machines[C]. Proc of IEEE Int Symposium on Circuits and Systems. New Jersey: IEEE Press, 2000: 757-760.

[10] Zhao Y P, Sun J G. Recursive reduced least squares support vector regression[J]. Pattern Recognition, 2009, 42(5): 837-842.

[11] Zhao Y P, Sun J G, Zhong H D, et al. An improved recursive reduced least squares support vector regression[J]. Neurocomputing, 2012, 87(3): 1-9.

[12] Cauwenberghs G, Poggio T. Incremental and decremental support vector machine learning[C]. Proc of the 14th Annual Neural Information Processing Systems Conf. Colorado, 2001: 409-423.

[13] Wang H, Pi D Y, Sun Y X. Online SVM regression algorithm-based adaptive inverse control[J]. Neurocomputing, 2007, 70(4): 952-959.

[14] 张浩然, 汪晓东. 回归最小二乘支持向量机的增量和在线式学习算法[J]. 计算机学报, 2006, 29(3): 400-406.
(Zhang H R, Wang X D. Incremental and online learning algorithm for regression least squares support vector machine[J]. Chinese J of Computers, 2006, 29(3): 400-406.)

[15] 张淑宁, 王福利, 尤富强, 等. 基于鲁棒学习的最小二乘支持向量机及其应用[J]. 控制与决策, 2010, 25(8): 1169-1177.
(Zhang S N, Wang F L, You F Q, et al. Robust least squares support vector machine based on robust learning algorithm and its application[J]. Control and Decision, 2010, 25(8): 1169-1177.)

[16] 张淑宁, 王福利, 何大阔, 等. 在线鲁棒最小二乘支持向量机回归建模[J]. 控制理论与应用, 2011, 28(11): 1601-1606.
(Zhang S N, Wang F L, He D K, et al. Modeling method of online robust least-squares-support-vector regression[J]. Control Theory & Applications, 2011, 28(11): 1601-1606.)

[17] Jiao L, Bo L, Wang L. Fast sparse approximation for least squares support vector machine[J]. IEEE Trans on Neural Networks, 2007, 18(3): 685-697.

[18] Cortez P, Cerdara A, Tsanas A. Uc irvine machine learning repository[DB/OL]. [2009-10-07](2012-08-20). http://archive.ics.uci.edulmll.

(责任编辑: 郑晓蕾)