Improved Scheme for Fast Approximation to Least Squares Support Vector Regression

Zhang Yuchen(张宇宸)[1], *Zhao Yongping*(赵永平)[1*], *Song Chengjun*(宋成俊)[2], *Hou Kuanxin*(侯宽新)[3], *Tuo Jinkui*(脱金奎)[4], *Ye Xiaojun*(叶小军)[5]

1. School of Mechanical Engineering, Nanjing University of Science and Technology, Nanjing, 210094, P. R. China;
2. Military Ammunition in Shenyang Representative Office, Shenyang, 110045, P. R. China;
3. Civil Aviation Flight University of China, Guanghan, 618307, P. R. China;
4. Heilongjiang North Tool Company Limited, Mudanjiang, 157013, P. R. China;
5. New Star Research Institute of Applied Technology, Hefei, 230031, P. R. China

(Received 9 February 2012; revised 2 May 2012; accepted 12 May 2012)

Abstract: The solution of normal least squares support vector regression (LSSVR) is lack of sparseness, which limits the real-time and hampers the wide applications to a certain degree. To overcome this obstacle, a scheme, named I^2FSA-LSSVR, is proposed. Compared with the previously approximate algorithms, it not only adopts the partial reduction strategy but considers the influence between the previously selected support vectors and the will-selected support vector during the process of computing the supporting weights. As a result, I^2FSA-LSSVR reduces the number of support vectors and enhances the real-time. To confirm the feasibility and effectiveness of the proposed algorithm, experiments on benchmark data sets are conducted, whose results support the presented I^2FSA-LSSVR.

Key words: support vector regression; kernel method; least squares; sparseness

CLC number: TP391 **Document code**: A **Article ID**: 1005-1120(2014)04-0413-07

1 Introduction

Since Suykens, et al[1] proposed least squares support vector machine (LSSVM), it has drawn much attention and obtained wide applications in many fields due to its high computational efficiency[2-4]. However, sparseness and robustness are two obvious weaknesses of LSSVM[5]. The lack of robustness is caused by the squared loss function used by LSSVM, while the equality constraints bring the loss of sparseness. The former can be alleviated using the weighted strategy, and the latter is realized with the pruning methods to a certain extent. As opposed to the lack of robustness, realizing sparseness seems more emergent, because it involves the real-time problem in the testing phase. Hence, many efforts were made to circumvent this drawback. A simple approach[6] was proposed to realize sparseness by sorting the support valve spectrum, i. e., deleting the smallest absolute values of supporting weights. In the following, Ref. [7] presented a more sophisticated pruning mechanism that omits the training samples bearing the least errors after deleted, and, furthermore, this algorithm was accelerated[8]. Based on sequential minimal optimization, another pruning algorithm was proposed to realize the sparse solution of LSSVM[9]. Recently, these pruning algorithms have been accelerated using the iterative methodology[10]. According to sequential forward greedy manner, Ref. [11] proposed a fast sparse approximation scheme for LSSVM, named as fast sparse approximated LSSVM (FSA-LSSVM). In the re-

Foundation item: Supported by the National Natural Science Foundation of China (51006052).

* **Corresponding author**: Zhao Yongping, Associate Professor, E-mail: y. p. zhao@163. com.

gression domain, it is called as fast sparse approximated least squares support vector regression (FSA-LSSVR). For LSSVM, there are not non-support vectors, i. e., every training sample is a support vector and makes contribution to the objective function. However, when FSA-LSSVR prunes the training samples, the complete reduction is adopted. That is to say, the pruned samples are totally discarded. Aiming at this inappropriate action, an improved FSA-LSSVR (IFSA-LSSVR) was proposed[12], i. e., during the process of updating the kernel matrix iteratively, the partial reduction strategy is adopted. However, there still exists certain limitation. In other words, the supporting weights of previously selected support vectors are fixed while selecting a new support vector, which means the influence between the previously selected support vectors and the will-selected support vector is not considered. Therefore, on the basis of IFSA-LSSVR, a further improved scheme, viz. the improved IFSA-LSSVR (I^2FSA-LSSVR), is proposed. I^2FSA-LSSVR not only adopts the partial reduction strategy but also considers the influence between the previously selected support vectors and the will-selected support vector.

2 Least Squares Support Vector Regression

Given the training samples set $\{(\boldsymbol{x}_\mathrm{i}, d_i)\}_{i=1}^N$ of the size N, where $\boldsymbol{x}_i \in \mathbf{R}^n$ and $d_i \in \mathbf{R}$, LSSVR[1] was given as[1]

$$\min_{w,b,e}\left\{\frac{1}{2}\boldsymbol{w}^\mathrm{T}\boldsymbol{w} + \frac{C}{2}\sum_{i=1}^N e_i^2\right\} \tag{1}$$

$$\text{s. t.}\quad d_i = \boldsymbol{w}^\mathrm{T}\varphi(\boldsymbol{x}_i) + b + e_i,\ i = 1,\cdots,N$$

where $\boldsymbol{w}$ can control the model complexity. b is the bias, $\boldsymbol{e} = [e_1,\cdots,e_N]^\mathrm{T}$ represents the training errors, C the regularization parameter, $\varphi(\cdot)$ the feature map realizing the transformation from the finite-dimensional input space to the high-dimensional feature space. To solve Eq. (1), the Lagrangian function is constructed.

$$L(\boldsymbol{w}, b, \boldsymbol{e}; \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^\mathrm{T}\boldsymbol{w} + \frac{C}{2}\sum_{i=1}^N e_i^2 + \sum_{i=1}^N \alpha_i \cdot$$

$$(d_i - \boldsymbol{w}^\mathrm{T}\varphi(\boldsymbol{x}_i) - b - e_i) \tag{2}$$

where $\boldsymbol{\alpha} = [\alpha_1,\cdots,\alpha_N]^\mathrm{T}$ is the Lagrange multiplier vector. According to the dual theorem, the optimality conditions are

$$\begin{cases} \dfrac{\partial L}{\partial \boldsymbol{w}} = 0 \rightarrow \boldsymbol{w} = \sum_{i=1}^N \alpha_i \varphi(\boldsymbol{x}_i) \\ \dfrac{\partial L}{\partial b} = 0 \rightarrow \sum_{i=1}^N \alpha_i = 0 \\ \dfrac{\partial L}{\partial e_i} = 0 \rightarrow \alpha_i = Ce_i \\ \dfrac{\partial L}{\partial \alpha_i} = 0 \rightarrow \boldsymbol{w}^\mathrm{T}\varphi(\boldsymbol{x}_i) + b + e_i - d_i = 0 \end{cases} \tag{3}$$

Plugging the variables $\boldsymbol{w}$ and $\boldsymbol{e}$ into Eq. (2), its Wolfe dual problem is

$$\min\left\{L = \frac{1}{2}\sum_{i,j=1}^N \alpha_i\alpha_j\varphi(\boldsymbol{x}_i)^\mathrm{T}\varphi(\boldsymbol{x}_j) + \sum_{i=1}^N \frac{\alpha_i^2}{2C} - \sum_{i=1}^N \alpha_i d_i\right\} \tag{4}$$

$$\text{s. t.}\quad \sum_{i=1}^N \alpha_i = 0$$

Transform the constrained problem Eq. (4) into the unconstrained equation

$$\min\left\{L = \frac{1}{2}[b\quad \boldsymbol{\alpha}^\mathrm{T}]\begin{bmatrix} 0 & \mathbf{1}^\mathrm{T} \\ \mathbf{1} & \bar{\boldsymbol{K}} \end{bmatrix}\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} - [b\quad \boldsymbol{\alpha}^\mathrm{T}]\begin{bmatrix} 0 \\ \boldsymbol{d} \end{bmatrix}\right\} \tag{5}$$

where $\bar{\boldsymbol{K}} = \boldsymbol{K} + \dfrac{\boldsymbol{I}}{C}$, $\boldsymbol{d} = [d_1,\cdots,d_N]^\mathrm{T}$, $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^\mathrm{T}\varphi(\boldsymbol{x}_j)$, $\mathbf{1} = [1_1,\cdots,1_N]^\mathrm{T}$. $\boldsymbol{I}$ is an identity matrix of appropriate dimension, and $k(\cdot,\cdot)$ the kernel function implicitly computing the inner product of two vectors in the feature space. It is very easy to obtain the optimal solution of Eq. (5) as follows

$$\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{1}^\mathrm{T} \\ \mathbf{1} & \bar{\boldsymbol{K}} \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ \boldsymbol{d} \end{bmatrix} \tag{6}$$

When Eq. (6) is solved, for a new input sample $\boldsymbol{x}$, its target can be predicted by the following regression machine

$$f(\boldsymbol{x}) = \boldsymbol{w}^\mathrm{T}\varphi(\boldsymbol{x}) + b = \sum_{i=1}^N \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b \tag{7}$$

where $\boldsymbol{\alpha}$ and b are from Eq. (6). From Eq. (7), it is easily understood that the solution of LSSVR is not sparse, i. e., every training sample is support vector, which hampers its use in those practical applications demanding extremely fast responses. That is to say, the real-time of LSSVR is limited.

To realize the sparseness and enhance the real-time of LSSVR, in the following, the algorithm I^2FSA-LSSVR is proposed.

3 I^2FSA-LSSVR

A method of realizing sparseness for LSSVR, named the I^2FSA-LSSVR, is described. It is a greedy algorithm, which iteratively builds the regression machine by adding one training sample from the training set at one time. I^2FSA-LSSVR consists of two important parts, viz. the selection of so-called support vector and the solution of subproblem. Starting with an empty index set $P=\varnothing$ and a full index set $Q=\{1,2,\cdots,N\}$, I^2FSA-LSSVR first selects a new support vector from the training set $\{(\boldsymbol{x}_i,d_i),i\in Q\}$ according to some criterion. Then, the index s is removed from Q and added to P. After determining the training sample to be included, I^2FSA-LSSVR solves the subproblem containing the new support vector and all previously picked support vectors. This procedure is repeated until a certain stopping criterion is satisfied.

If the sample $\boldsymbol{x}_s$ is selected as the support vector at the $(n+1)$th iteration, the inverse matrix in Eq. (6) becomes as

$$\boldsymbol{R}^{n+1}=\begin{bmatrix}0 & \boldsymbol{1}_{|P|}^{\mathrm{T}} & 1\\ \boldsymbol{1}_{|P|} & \overline{K}_{PP} & \bar{\boldsymbol{k}}_s\\ 1 & \bar{\boldsymbol{k}}_s^{\mathrm{T}} & k_{ss}\end{bmatrix}^{-1} \quad (8)$$

where $\bar{\boldsymbol{k}}_s=[k_{P_1s},k_{P_2s},\cdots,k_{P_{|P|}s}]^{\mathrm{T}}$, $|\cdot|$ represents the cardinality, and P_i the ith element of the index set P. Given that

$$\boldsymbol{R}^n=\begin{bmatrix}0 & \boldsymbol{1}_{|P|}^{\mathrm{T}}\\ \boldsymbol{1}_{|P|} & \overline{K}_{PP}\end{bmatrix}^{-1} \quad (9)$$

The following updating equation can be found via Sherman-Morrion formula[13]

$$\boldsymbol{R}^{n+1}=\begin{bmatrix}\boldsymbol{R}^n & \boldsymbol{0}\\ \boldsymbol{0}^{\mathrm{T}} & 0\end{bmatrix}+\lambda\begin{bmatrix}\boldsymbol{\beta}\\ -1\end{bmatrix}[\boldsymbol{\beta}^{\mathrm{T}} \quad -1] \quad (10)$$

where $\boldsymbol{\beta}=\boldsymbol{R}_n\begin{bmatrix}1\\ \bar{\boldsymbol{k}}_s\end{bmatrix}$, $\lambda=(k_{ss}-[1 \quad \bar{\boldsymbol{k}}_s^{\mathrm{T}}]\boldsymbol{\beta})^{-1}$. From Eq. (10), b and $\boldsymbol{\alpha}$ can be obtained for constructing the regression machine. However, as we know, there are not non-support vectors for LSSVR, that is, each training sample is a support vector. In other words, each sample makes contribution to the cost function in Eq. (1). Different from the complete reduction strategy in Ref. [11], where the updating Eq. (10) is used, Ref. [12] adopts the partial reduction strategy. Accordingly, the following updating equation is considered.

$$\widetilde{\boldsymbol{R}}^{n+1}=\begin{bmatrix}\widetilde{\boldsymbol{R}}^n & \boldsymbol{0}^{\mathrm{T}}\\ \boldsymbol{0} & 0\end{bmatrix}+\tilde{\lambda}\begin{bmatrix}\tilde{\boldsymbol{\beta}}\\ -1\end{bmatrix}[\tilde{\boldsymbol{\beta}}^{\mathrm{T}} \quad -1] \quad (11)$$

where $\tilde{\boldsymbol{\beta}}=\widetilde{\boldsymbol{R}}^n\tilde{\boldsymbol{k}}_s$, $\tilde{\lambda}=(\tilde{k}_{ss}-\tilde{\boldsymbol{k}}_s^{\mathrm{T}}\tilde{\boldsymbol{\beta}})^{-1}$. Here,

$$\tilde{\boldsymbol{k}}_s=\begin{bmatrix}0 & \boldsymbol{1}_{|P|}^{\mathrm{T}}\\ \boldsymbol{1}_{|O|} & \overline{K}_{OP}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}1\\ \overline{K}_{Os}\end{bmatrix} \quad (12)$$

$$\tilde{k}_{ss}=[1 \quad \overline{K}_{Os}]\begin{bmatrix}1\\ \overline{K}_{Os}\end{bmatrix} \quad (13)$$

where $O=\{1,2,\cdots,N\}$ is a full index set. Plugging Eq. (11) into Eq. (6) yields the following equation

$$\begin{bmatrix}\tilde{b}^{n+1}\\ \tilde{\boldsymbol{\alpha}}_P^{n+1}\\ \tilde{\alpha}_s^{n+1}\end{bmatrix}=\widetilde{\boldsymbol{R}}^{n+1}\begin{bmatrix}\tilde{0}\\ \tilde{\boldsymbol{d}}_P\\ \tilde{d}_s\end{bmatrix}=\begin{bmatrix}\widetilde{\boldsymbol{R}}^n\begin{bmatrix}\tilde{0}\\ \tilde{\boldsymbol{d}}_P\end{bmatrix}\\ 0\end{bmatrix}+ \tilde{\lambda}\left(\tilde{\boldsymbol{\beta}}^{\mathrm{T}}\begin{bmatrix}\tilde{0}\\ \tilde{\boldsymbol{d}}_P\end{bmatrix}-\tilde{d}_s\right)\begin{bmatrix}\tilde{\boldsymbol{\beta}}\\ -1\end{bmatrix} \quad (14)$$

where $\tilde{d}_s=[1 \quad \overline{K}_{sO}]\begin{bmatrix}0\\ \boldsymbol{d}_O\end{bmatrix}$, $\tilde{0}=\sum_{i=1}^{N}d_i$. Further,

$$\begin{bmatrix}\tilde{b}^{n+1}\\ \tilde{\boldsymbol{\alpha}}_P^{n+1}\\ \tilde{\alpha}_s^{n+1}\end{bmatrix}=\begin{bmatrix}\tilde{b}^{n}\\ \tilde{\boldsymbol{\alpha}}_P^{n}\\ 0\end{bmatrix}+\tilde{\lambda}\left(\tilde{\boldsymbol{\beta}}^{\mathrm{T}}\begin{bmatrix}\tilde{0}\\ \tilde{\boldsymbol{d}}_P\end{bmatrix}-\tilde{d}_s\right)\begin{bmatrix}\tilde{\boldsymbol{\beta}}\\ -1\end{bmatrix} \quad (15)$$

From Eq. (15), we obtain the following regression machine.

$$f(\boldsymbol{x})=\sum_{i\in P}\tilde{\alpha}_i k(\boldsymbol{x}_i,\boldsymbol{x})+\tilde{b} \quad (16)$$

From Eqs. (12, 13), it is not difficult to know that during the process of finding $\tilde{\alpha}_i\,(i\in P)$ and $\tilde{b}$, the contribution of the training samples indexed by Q is considered as well. As a result, the selected support vectors are reduced obviously, which means that the real-time of the improved algorithm, IFSA-LSSVR, is enhanced. Up to now, there have been two significant problems to solve, namely, the criterion of selecting support vector and the stopping criterion. In what follows, we will give them.

Regarding the criterion of selecting support vectors, a natural idea that the sample making the most contribution to the cost function of Eq. (1) is chosen as the support vector, arises.

Substituting Eq. (6) into Eq. (5) gives rise to the following equation

$$\tilde{L}=-\frac{1}{2}\begin{bmatrix} b & \boldsymbol{\alpha}_P^{\mathrm{T}} \end{bmatrix}\begin{bmatrix} 0 \\ \boldsymbol{d}_P \end{bmatrix} \tag{17}$$

The deviation of Eq. (17) caused by choosing the training sample $\boldsymbol{x}_s$ as the support vector is

$$\Delta\tilde{L}=\tilde{L}^{(n)}-\tilde{L}^{(n+1)}=\frac{\tilde{\lambda}}{2}\left(\tilde{\boldsymbol{\beta}}^{\mathrm{T}}\begin{bmatrix}\tilde{0}\\ \tilde{\boldsymbol{d}}_P\end{bmatrix}-\tilde{d}_s\right)^2 \tag{18}$$

The larger the $\Delta\tilde{L}$ is, the most important the sample $\boldsymbol{x}_s$. During each iteration, the sample causing the largest deviation will be chosen as the support vector. From Eq. (18), the criterion of choosing support vector is obtained.

$$s=\arg\min_{i\in Q}\left\{\Delta\tilde{L}_i=\frac{\tilde{\lambda}}{2}\left(\tilde{\boldsymbol{\beta}}^{\mathrm{T}}\begin{bmatrix}\tilde{0}\\ \tilde{\boldsymbol{d}}_P\end{bmatrix}-\tilde{\boldsymbol{d}}_i\right)^2\right\} \tag{19}$$

As opposed to the criterion in Refs. [11,12] , here the criterion not only adopts the partial reduction strategy but also considers the influence between the previously selected support vectors and the will-selected support vector. Therefore, during the process of calculating the cost function at the $(n+1)$th iteration, the updated weights is adopted instead of the fixed weights, i. e. , the weights at the nth iteration.

As for the stopping criterion, the pre-defined number of support vectors M can be utilized to control the sparseness, thus satisfying our requirement of real-time. Of course, similar to Refs. [11] and [12] , according to the prediction errors, the stopping criterion is obtained as well

$$\max(|\boldsymbol{r}_Q^n|)<\varepsilon \tag{20}$$

where $r_i^n=\begin{cases}-d_i & n=0\\ \sum\limits_{j\in P}\tilde{\alpha}_j k(\boldsymbol{x}_j,\boldsymbol{x}_i)+\tilde{b}^n-d_i & n>0\end{cases}$, ε is a pre-chosen positive small number which can control the number of support vectors. The larger the ε is, the fewer the support vectors. When ε approaches zero, I^{2}FSA-LSSVR goes to the normal LSSVR. As an extreme, when $\varepsilon=0$, I^{2}FSA-LSSVR is completely equivalent to the normal LSSVR. In summary, the flowchart of realizing I^{2}FSA-LSSVR is depicted as follows:

(1) To initialize $\tilde{\boldsymbol{\alpha}}^0=\boldsymbol{0}^{\mathrm{T}}$, $\tilde{b}=0$, $O=Q=\{1,\cdots,N\}$, $P=\varnothing$, $\tilde{0}=\sum\limits_{i=1}^{N}d_i$, $n=0$;

(2) If $Q=\varnothing$ or $\max(|\boldsymbol{r}_Q^n|)<\varepsilon$, stop; else find the index s according to Eq. (19);

(3) If $n=0$, $\tilde{\boldsymbol{R}}^{n+1}=\left(\begin{bmatrix}0 & 1\\ \boldsymbol{1}_{|O|} & \bar{K}_{Os}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}0 & 1\\ \boldsymbol{1}_{|O|} & \bar{K}_{Os}\end{bmatrix}\right)^{-1}$ and $\begin{bmatrix}\tilde{b}^{n+1}\\ \tilde{\alpha}_s^{n+1}\end{bmatrix}=\tilde{\boldsymbol{R}}^{n+1}\begin{bmatrix}\tilde{0}\\ \tilde{d}_s\end{bmatrix}$; else compute $\tilde{R}^{n+1}$, $\tilde{\boldsymbol{\alpha}}_P^{n+1}$, $\tilde{\alpha}_s^{n+1}$, $\tilde{b}^{n+1}$ and $\tilde{\boldsymbol{\beta}}$ from Eqs. (11,15);

(4) $P=P\cup\{s\}$, $Q=Q/\{s\}$;

(5) To calculate $\boldsymbol{r}_Q^{n+1}=K_{QP}\tilde{\boldsymbol{\alpha}}_P^{n+1}+\tilde{b}^{n+1}-\boldsymbol{d}_Q$, $n=n+1$, and go to step (2).

4 Experimental Results

To show the effectiveness and feasibility of the proposed I^{2}FSA-LSSVR, 11 experiments on benchmark data sets (available from http://www. liaad. up. pt/~ ltorgo/Regression/ or http://archive. ics. uci. edu/ml/), i. e. , stock, winequality_red, winequality_white, bank8FM, bank32NH, puma8NH, puma32H, cpu_small, cpu_act, total_UPDRS, and motor_UPDRS, are completed on an processor of Intel(R) Core™ i7 950 with 4 GB RAM using MATLAB R2007a complier under a Windows XP operation system. The Gaussian kernel function $k(\boldsymbol{x}_i,\boldsymbol{x}_j)=\exp\left(-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2}{2\gamma^2}\right)$ is chosen to construct I^{2}FSA-LSSVR. As for the choice of kernel parameter γ and the regularization parameter C, they are determined using the cross-validation method[14] from the pairs $\{2^{-4},\cdots,2^4\}\times\{2^{-4},\cdots,2^{10}\}$. For each data set, the input data are normalized in the range of [0, 1], but there are no normalizations on target values. The detailed specifications about these data sets are listed in Table 1. To compare conveniently, two performance indexes, the root mean squared errors (RMSE) and the normalized mean square error (NMSE), are defined, respectively, in the following

$$\mathrm{RMSE}=\sqrt{\frac{\sum\limits_{i=1}^{N}(\hat{d}_i-d_i)^2}{N}} \tag{21}$$

Table 1 Experimental results on benchmark data sets

Data set	Algorithm	RMSE	NMSE	trTime/s	teTime/s	#SV	trNum	teNum
stock ($C=2^5$, $\gamma=2^{-2}$)	Normal LSSVR	7.374E-01	1.294E-02	0.41	0.45	600	600	350
	FSA-LSSVR	7.473E-01	1.329E-02	0.37	0.16	220	600	350
	IFSA-LSSVR	7.487E-01	1.335E-02	0.37	0.14	190	600	350
	I^{2}FSA-LSSVR	7.402E-01	1.304E-02	105.99	0.11	150	600	350
winequality_red ($C=2^6$, $\gamma=2^1$)	Normal LSSVR	6.184E-01	5.876E-01	1.14	1.22	1 000	1 000	599
	FSA-LSSVR	6.243E-01	5.987E-01	0.20	0.09	75	1 000	599
	IFSA-LSSVR	6.248E-01	5.998E-01	0.05	0.03	15	1 000	599
	I^{2}FSA-LSSVR	6.261E-01	6.023E-01	10.83	0.02	5	1 000	599
winequality_white ($C=2^6$, $\gamma=2^0$)	Normal LSSVR	7.119E-01	6.477E-01	15.27	10.00	3 500	3 500	1 398
	FSA-LSSVR	7.259E-01	6.734E-01	11.03	1.68	590	3 500	1 398
	IFSA-LSSVR	7.246E-01	6.710E-01	1.22	0.34	120	3 500	1 398
	I^{2}FSA-LSSVR	7.259E-01	6.735E-01	927.49	0.09	35	3 500	1 398
bank8FM ($C=2^7$, $\gamma=2^0$)	Normal LSSVR	3.248E-02	4.517E-02	26.13	33.87	4 500	4 500	3 692
	FSA-LSSVR	3.307E-02	4.684E-02	10.94	3.70	495	4 500	3 692
	IFSA-LSSVR	3.300E-02	4.664E-02	2.79	1.34	180	4500	3 692
	I^{2}FSA-LSSVR	3.313E-02	4.700E-02	4 217.38	0.73	95	4 500	3 692
bank32NH ($C=2^2$, $\gamma=2^1$)	Normal LSSVR	8.167E-02	4.359E-01	28.42	37.49	4 500	4 500	3 692
	FSA-LSSVR	8.330E-02	4.535E-01	58.95	10.42	1 270	4 500	3 692
	IFSA-LSSVR	8.328E-02	4.533E-01	15.05	3.76	455	4 500	3 692
	I^{2}FSA-LSSVR	8.325E-02	4.530E-01	3 337.11	0.58	70	4 500	3 692
puma8NH ($C=2^0$, $\gamma=2^{-1}$)	Normal LSSVR	3.303	3.485E-01	26.21	33.77	4 500	4 500	3 692
	FSA-LSSVR	3.369	3.626E-01	80.09	11.75	1 510	4 500	3 692
	IFSA-LSSVR	3.369	3.626E-01	46.32	6.13	810	4 500	3 692
	I^{2}FSA-LSSVR	3.368	3.623E-01	8 937.42	1.50	200	4 500	3 692
puma32H ($C=2^9$, $\gamma=2^1$)	Normal LSSVR	2.675E-02	7.478E-01	28.52	37.69	4 500	4 500	3 692
	FSA-LSSVR	2.721E-02	7.738E-01	73.73	11.76	1 420	4 500	3 692
	IFSA-LSSVR	2.722E-02	7.743E-01	0.44	0.28	35	4 500	3 692
	I^{2}FSA-LSSVR	2.723E-02	7.749E-01	716.34	0.12	15	4 500	3 692
cpu_small ($C=2^7$, $\gamma=2^{-1}$)	Normal LSSVR	3.081	2.714E-02	32.90	32.95	5 000	5 000	3 192
	FSA-LSSVR	3.139	2.817E-02	16.72	3.99	600	5 000	3 192
	IFSA-LSSVR	3.141	2.820E-02	2.59	1.01	155	5 000	3 192
	I^{2}FSA-LSSVR	3.116	2.776E-02	4 413.89	0.51	80	5 000	3 192
cpu_act ($C=2^8$, $\gamma=2^0$)	Normal LSSVR	3.220	3.295E-02	34.41	34.27	5 000	5 000	3 192
	FSA-LSSVR	3.190	3.223E-02	3.28	1.37	200	5 000	3 192
	IFSA-LSSVR	3.211	3.264E-02	1.87	0.81	120	5 000	3 192
	I^{2}FSA-LSSVR	3.284	3.416E-02	2 237.85	0.27	40	5 000	3 192
total_UPDRS ($C=2^9$, $\gamma=2^{-1}$)	Normal LSSVR	9.115	7.466E-01	11.31	18.39	3 000	3 000	2 875
	FSA-LSSVR	9.255	7.696E-01	3.81	2.04	330	3 000	2 875
	IFSA-LSSVR	9.283	7.742E-01	1.31	0.84	140	3 000	2 875
	I^{2}FSA-LSSVR	9.223	7.643E-01	708.96	0.20	35	3 000	2 875
motor_UPDRS ($C=2^8$, $\gamma=2^{-1}$)	Normal LSSVR	6.958	7.528E-01	11.48	18.21	3 000	3 000	2 875
	FSA-LSSVR	7.079	7.791E-01	4.76	2.29	380	3 000	2 875
	IFSA-LSSVR	7.071	7.775E-01	0.41	0.31	50	3 000	2 875
	I^{2}FSA-LSSVR	7.057	7.745E-01	702.80	0.20	35	3 000	2 875

$$\mathrm{NMSE}=\frac{1}{\Delta^2 N}\sum_{i=1}^{N}(d_i-\hat{d}_i)^2 \qquad (22)$$

where $\Delta^2=\frac{1}{N-1}\sum_{i=1}^{N}(d_i-\bar{d})^2$ with $\bar{d}$ being the mean of the measured value, $\hat{d}_i$ is the prediction value, and d_i the measured value.

The detailed experimental results are shown in Table 1, where trTime and teTime represent the training time and the testing time, respectively; and #SV, trNum, teNum denote the number of support vectors, the number of training samples, and the number of testing samples, respectively. From Table 1, I^{2}FSA-LSSVR needs the least number of support vectors when all the algorithms reach the almost same prediction accuracy. Hence, when they come to the almost same generalization performance, the real-time of I^{2}FSA-LSSVR is the best. Compared with the FSA-LSSVR, the solution of IFSA-LSSVR is much sparser. The main reason is that when IFSA-LSSVR computes the supporting weights, the partial reduction strategy is applied. On the basis of IFSA-LSSVR, I^{2}FSA-LSSVR considers the influence between the previously selected support vectors and the will-selected support vector when calculating the supporting weights. Consequently, I^{2}FSA-LSSVR obtains the best real-time performance, that is, the sparsest solution. As for this point, the columns of teTime and #SV can answer it. However, I^{2}FSA-LSSVR needs the most training time. During each iteration, the updating computational complexity of FSA-LSSVR is max $\{O(|P|^2), O(|P| \cdot |Q|)\}$. After improvement, the computational cost of IFSA-LSSVR increases to $O(N \cdot |P|)$. Since IFSA-LSSVR needs fewer iterations compared to FSA-LSSVR, the total cost of adding up all the iterations for IFSA-LSSVR becomes less, which is supported by the training time in Table 1. At each iteration, I^{2}FSA-LSSVR needs the largest computational burden $O(N \cdot |Q| \cdot |P|)$. Although it needs the fewest support vectors, the total burden becomes the largest. The training time in Table 1 may address it. Fortunately, these algorithms are all offline, i. e., the computational burden in the training phase can be finished affordably in an offline model, which can not affect the real-time performance in the testing phase. All in all, I^{2}FSA-LSSVR needs the least number of support vectors and realizes the sparsest solution of LSSVR, thus enhancing the real-time.

5 Conclusions

Since the solution of normal LSSVR lacks the sparseness, which limits the real-time in the testing phase and hampers the wide applications to a certain degree, a fast sparse approximation scheme, FSA-LSSVR, was proposed. However, during the process of computing the supporting weights, FSA-LSSVR adopts the complete reduction strategy. To overcome this drawback, an improved scheme, IFSA-LSSVR, was presented, which takes the partial reduction strategy to calculate the supporting weights in each iteration. However, for IFSA-LSSVR there also exists limitation in calculating supporting weights because it does not consider the influence between the previously selected support vectors and the will-selected support vector. Hence, I^{2}FSA-LSSVR circumvents this problem and obtains the much sparser solution. Besides, it further enhances the real-time performance of LSSVR. To confirm the feasibility and effectiveness of the proposed I^{2}FSA-LSSVR, a lot of experiments on benchmark data sets are conducted, whose results favor it.

References:

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

[2] Zhao Y, Sun J. Boosting sparse least squares support vector regression and its application to thrust estimation[J]. Transactions of Nanjing University of Aeronautics and Astronautics, 2008, 25(4): 254-261.

[3] Zhao Y, Sun J, Wang J. Online parsimonious least squares support vector regression and its application [J]. Transactions of Nanjing University of Aeronautics and Astronautics, 2009, 20(5): 1086-1090.

[4] Su W, Zhao Y, Sun J. Novel weighted least squares support vector regression for thrust estimation on performance deterioration of aero-engine[J]. Transactions of Nanjing University of Aeronautics and Astronautics, 2012, 29(1): 25-32.

[5] Suykens J A K, De Brabanter J, Lukas L, et al. Weighted least squares support vector machines: robustness and sparse approximation[J]. Neurocom-

puting, 2002, 48(1-4):85-105.

[6] Suykens J A K, Lukas L, Vandewalle J. Sparse approximation using least square vector machines[C]// Proceedings of 2000 IEEE International Symposium on Circuits and Systems. Piscataway: IEEE, 2000: 757-760.

[7] De Kruif B J, De Vries T J A. Pruning error minimization in least squares support vector machines[J]. IEEE Transactions on Neural Networks, 2004, 14 (3):696-702.

[8] Kuh A, De Wilde P. Comments on "pruning error minimization in least squares support vector machines"[J]. IEEE Transactions on Neural Networks, 2007, 18(2):606-609.

[9] Zeng X Y, Chen X W. SMO-based pruning methods for sparse least squares support vector machines[J]. IEEE Transactions on Neural Networks, 2005, 16 (6):1541-1546.

[10] Zhao Y, Sun J. Improved scheme to accelerate sparse least squares support vector regression[J]. Journal of Systems Engineering and Electronics, 2010, 21(2): 312-317.

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

[12] Zhao Yongping, Sun Jianguo. Fast method for sparse least squares support vector regression machine[J]. Control and Decision, 2008, 23(12):1347-1352. (in Chinese)

[13] Zhang X. Matrix analysis and applications[M]. Beijing:Tsinghua University Press, 2005. (in Chinese)

[14] An S, Liu W, Venkatesh S. Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression[J]. Pattern Recognition, 2007, 40(8):2154-2162.

(Executive editor: Zhang Tong)