Int. J. Mach. Learn. & Cyber. (2017) 8:1853–1861
DOI 10.1007/s13042-016-0563-6

ORIGINAL ARTICLE

Improved sparse LSSVMS based on the localized generalization error model

Binbin Sun[1] · Wing W. Y. Ng[2] · Patrick P. K. Chan[2]

Received: 3 August 2015 / Accepted: 1 July 2016 / Published online: 9 July 2016

Abstract The least squares support vector machine (LSSVM) is computationally efficient because it converts the quadratic programming problem in the training of SVM to a linear programming problem. The sparse LSSVM is proposed to promote the predictive speed and generalization capability. In this paper, two sparse LSSVM algorithms: the SMRLSSVM and the RQRLSSVM are proposed based on the Localized Generalization Error of the LSSVM. Experimental results show that the RQRLSSVM yields both better generalization capability and sparseness in comparison to other sparse LSSVM algorithms.

Keywords Least squares support vector machine (LSSVM) · Localized generalization error model (L-GEM) · Sensitivity measure · Sparsity

1 Introduction

Suykens proposes the least squares support vector machine (LSSVM) [1] as an extension of the support vector machine (SVM) by replacing the inequality constraints in the SVM by equality constraints in the LSSVM. With the equality constraints, solving the LSSVM problem is converted from the quadratic programming in the SVM into a linear programming problem. The LSSVM is widely studied empirically and applied in a lot of applications [2].The generalization performance and the speed of the LSSVM are comparable to that of the SVM [3]. The optimization problem being solved by the LSSVM has the same formation for both of the classification and regression tasks. The number of hyper parameters of the LSSVM is less than that of the SVM for regression problems. More importantly, the LSSVM is much more computational efficient in comparison to the SVM because of the use of linear programming instead of quadratic programming.

However, owing to the conversion of the inequality constraints to equality constraints, the LSSVM loses the sparseness and is a full dense machine, i.e. all training samples are used as support vectors. It leads to a slower prediction speed of the LSSVM. Hence many algorithms are developed to impose sparseness to the LSSVM to overcome the full dense limitation. Current research results show that sparse LSSVMs usually yield the same or better generalization capability and faster prediction speed in comparison to full dense LSSVMs [4–7]. In fact, the sparsity issue is also a research problem in training the SVM [8–11], therefore some algorithms developed for training sparse SVM are directly applied to train sparse LSSVM.

The procedure to obtain a sparse LSSVM can be treated as an optimization problem. There are two major strategies of sparse LSSVM training algorithms: one-step and iterative methods. One-step methods start with a full dense LSSVM and remove support vectors (training samples) based on the evaluation of objective functions or criteria. These methods complete the sparsity and LSSVM training in one step. Iterative methods perform forward growing or backward pruning algorithms iteratively to achieve the

✉ Wing W. Y. Ng
wingng@ieee.org

[1] Department of Computer Science, Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen 518055, People's Republic of China

[2] School of Computer Science and Engineering, Machine Learning and Cybernetics Research Center, South China University of Technology, Guangzhou 510006, People's Republic of China

sparsity of the LSSVM. One-step methods avoid the intensive computations in iterative methods and obtain the same or better generalization capability. The coupled compressive algorithm [5] and the IP-LSSVM [12] are examples of the one-step methods. The iterative methods update the objective function after a sample or a batch of samples are pruned or added. The SMO-based method [13] iteratively prunes samples yielding the minimum changes to a dual objective function. There are also methods performing forward growing of samples instead of pruning which leads to the largest reduction of objective functions [6, 7, 14].

In this paper, we proposed two one-step methods to obtain a sparse LSSVM based on the localized generalization error model (L-GEM): the SMRLSSVM and the RQRLSSVM. There are two major components in the L-GEM: the training mean square error and the sensitivity measure. The SMRLSSVM uses the sensitivity measure as the criterion to select support vectors while the RQRLSSVM selects support vectors using both components of the L-GEM.

Section 2 briefly introduces concepts of the LSSVM and the sparse LSSVM. Both the SMRLSSVM and the RQRLSSVM are proposed in Sect. 3. Simulations results on both an artificial dataset and benchmark datasets are discussed in Sects. 4.1 and 4.2, respectively. Section 5 concludes this work.

2 LSSVM and sparse LSSVM

In the following of this section, we will introduce the LSSVM and the sparse LSSVM in Sects. 2.1 and 2.2, respectively.

2.1 Least squares support vector machines

In this work, we focus on the supervised classification tasks. It is easy to extend to supervised regression as described in introduction section. The given dataset is denoted by $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, where $\mathbf{x}_i \in R^n$ and $y_i \in \{\pm 1\}$ denote feature vector and the true classification of an sample, respectively. The training of the LSSVM is to solve the following optimization problem:

$$\begin{aligned} \min & J(w, b, e) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \gamma\frac{1}{2}\sum_{i=1}^{l} e_i^2 \\ \text{s.t.} \quad & y_i\left(\mathbf{w}^T\varphi(\mathbf{x}_i) + b\right) = 1 - e_i,\ i = 1, \ldots, l \end{aligned} \tag{1}$$

where $\varphi(\mathbf{x}_i)$, $\mathbf{w}$, b, and γ denote a function projecting samples from the input space to the feature space, the weight vector, the offset and the corresponding penalty parameter controlling the trade-off of the margin (inversely proportional to $\mathbf{w}$) and the error ($\sum_{i=1}^{l} e_i^2$) of the LSSVM. As constraints in optimizing the LSSVM are all equalities and the error term in the objective function is in the form of a summation of squares, these constraints can be replaced by $(\mathbf{w}^T\varphi(\mathbf{x}_i) + b) = y_i - e_i$ for $i = 1, \ldots, l$. This replacement yields the same solution as the original one.

The primal Lagrangian for the LSSVM is

$$\begin{aligned} L(\mathbf{w}, \mathbf{e}, b; \boldsymbol{\alpha}) = & \frac{1}{2}\mathbf{w}^T\mathbf{w} + \gamma\frac{1}{2}\sum_{i=1}^{l} e_i^2 \\ & - \sum_{i=1}^{l} \alpha_i\left[\left(\mathbf{w}^T\varphi(\mathbf{x}_i) + b\right) - y_i + e_i\right] \end{aligned} \tag{2}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_l)$ denotes a vector consisting of l Lagrange multipliers. The optimality conditions for Eq. (2) are as follows:

$$\begin{cases} \dfrac{\partial L(\mathbf{w}, \mathbf{e}, b; \boldsymbol{\alpha})}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \displaystyle\sum_{i=1}^{l} \alpha_i\varphi(\mathbf{x}_i) \\ \dfrac{\partial L(\mathbf{w}, \mathbf{e}, b; \boldsymbol{\alpha})}{\partial b} = 0 \Rightarrow \displaystyle\sum_{i=1}^{l} \alpha_i = 0 \\ \dfrac{\partial L(\mathbf{w}, \mathbf{e}, b; \boldsymbol{\alpha})}{\partial \mathbf{e}} = 0 \Rightarrow \alpha_i = \gamma e_i \\ \dfrac{\partial L(\mathbf{w}, \mathbf{e}, b; \boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = 0 \Rightarrow \mathbf{w}^T\varphi(\mathbf{x}_i) + b - y_i + e_i = 0 \end{cases} \tag{3}$$

This optimization problem is equivalent to solving the following linear system:

$$\begin{bmatrix} \mathbf{K}_l + \gamma^{-1}\mathbf{I} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix} \tag{4}$$

where $K_l = \left[k_{ij}\right]_{i,j=1}^{l}$ and $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \langle\varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j)\rangle$ denotes the kernel function. The kernel trick $k(\mathbf{x}_i, \mathbf{x}_j) = \langle\varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j)\rangle$ is used to evaluate the similarity of sample pairs in the feature space without explicitly knowing the function $\varphi(\mathbf{x})$. The parameters of function $k()$ are called kernel parameters. The Gaussian kernel function $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/2\sigma^2)$ with kernel parameter σ is widely used. The Gaussian kernel function is used in this work as it is a popular choice of the kernel function for LSSVM [7].

The solution of the LSSVM is written in terms of the dual parameters $(\boldsymbol{\alpha}, b)$ as follows:

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b\right) \tag{5}$$

As in Eq. (3), the weight vector $\mathbf{w}$ is represented as a weighted sum of the l training samples $\mathbf{w} = \sum_{i=1}^{l} \alpha_i\varphi(\mathbf{x}_i)$. In the traditional SVM, samples with non-zero Lagrange Multipliers are called support vectors. Unlike the traditional SVM using the hinge loss as the error function, the

LSSVM using the least square loss is typically full dense, i.e. $s = l$ where s denotes the number of support vectors.

With the assumption that the weight vector $\mathbf{w}$ can be represented as a weighted sum of less number of support vectors $\mathbf{w} = \sum_{i=1}^{s} \alpha_i \varphi(\mathbf{x}_i), \mathbf{x}_i \in S \subset R^d$, and s ≪ l, we get the reduced LSSVM (RLSSVM). The RLSSVM solves the following optimization problem [4, 6]:

$$\min J(w,b,e) = \frac{1}{2}\sum_{i=1}^{s}\sum_{j=1}^{s}\alpha_i\alpha_j k_{ij} + \gamma\frac{1}{2}\sum_{i=1}^{l}\left(y_i - \sum_{j=1}^{s}\alpha_j k_{ij} - b\right)^2 \quad (6)$$

where $K_s = \left[k_{ij}\right]_{i,j=1}^{s}$, and $\mathbf{K}_{sl} = \left[k_{ij}\right]_{i=1,j=1}^{i=s,j=l}$.

This optimization problem is equivalent to solving the following linear system:

$$\left(\mathbf{R} + \mathbf{p}^T\mathbf{p}\right)\begin{bmatrix}\boldsymbol{\alpha}_s \\ b\end{bmatrix} = \mathbf{p}^T\mathbf{y} \quad (7)$$

where $\mathbf{R} = \begin{bmatrix}\gamma^{-1}\mathbf{K} & \mathbf{0} \\ \mathbf{0}^T & 0\end{bmatrix}$ and $\mathbf{p} = [\mathbf{K}_{sl}, \mathbf{1}]$.

The LSSVM uses all training sample as support vectors. As shown in [13], pruning training samples to reduce the number of support vectors is an instance of sparse LSSVM training methods. Therefore, pruned training samples will not be used as support vectors of the LSSVM in the coming training turns. The optimization problems of training the sparse LSSVM remain the same except the number of training samples l is set to be the number of support vectors (S). However, the reduction of the number of training samples may reduce the generalization capability of the sparse LSSVM owing to information loss by the reduced training sample set. So, sparse LSSVMs without training samples eliminating are preferred. On the other hand, using constraints without the bias term is another simple modification to boost the training efficiency [13]. In this case, the LSSVM without bias terms solves the optimization problem same as the kernel ridge regression or the Gaussian process regression does. The linear system being solved is $(\mathbf{K} + \gamma^{-1}\mathbf{I})\boldsymbol{\alpha} = \mathbf{y}$.

2.2 Sparse LSSVM

According to the equality constraints in Eq. (3), the LSSVM is full dense. It is because the use of least square error loss function as the objective and Lagrange multipliers of all support vectors are rarely equal to zero. As aforementioned, traditional SVMs also encounter the sparsity problem, therefore some sparse methods for SVM can be applied to obtain sparse LSSVM directly [8]. Some of the sparse methods with a pruning criterion are inspired by the sparse neural networks training methods and the pruning procedure of decision trees. In this section, we focus on methods proposed for sparse LSSVMs directly.

Suykens proposes a simple method to obtain sparse LSSVM by sorting the support value spectrum and pruning the support vector with the minimum absolute value of α iteratively [15]. In the LSSVM dual space, the magnitude of α is proportional to the modulus of error e. Therefore, the minimum absolute value of α infers to the minimum absolute value of e. So, the support vector yielding the minimum squared error of training sample (e^2) is pruned, i.e. $\arg\left(\min_i |\alpha_i|\right)$.

The IP-LSSVM achieves sparsity by only keeping support vectors in margin as in SVM's and maintains the simplicity of the LSSVM [12]. The IP-LSSVM eliminates samples with negative or small positive Lagrange multipliers. Hence, samples located on the border between two classes or being misclassified are kept and will serve as support vectors ($\alpha_i \gg 0$). The pruning criterion of the IP-LSSVM is $\arg\left(\min_i \alpha_i\right)$. For the LSSVM with the Gaussian kernel and a large value of kernel parameter σ, most of the Lagrange parameters α_i are positive. When all the Lagrange parameters are positive, the IP-LSSVM is equivalent to the algorithm proposed by Suykens [15].

The sparse LSSVM based on SMO [13] solves the training problems using the Sequential Minimal Optimization (SMO) without the bias term and iteratively removes the sample yielding minimum change of the dual objective function. Samples being removed will not participate in training in later iterations. The sample yielding the minimum change to the dual objective function as in Eq. (8) is pruned. The training of the LSSVMs is speeded up by caching intermediate results from previous iterations.

$$\min_i \left|\frac{1}{2}\alpha_i^2 K(\mathbf{x}_k,\mathbf{x}_k) - \alpha_i\left(\sum_{j=1}^{l}\alpha_j K(\mathbf{x}_i,\mathbf{x}_j)\right)\right| \quad (8)$$

With some derivations, one will find that the selection criterion in Eq. (8) is equivalent to $\min_i(|\alpha_i|)$ for the Gaussian kernel LSSVM.

The FSALSSVM, the RRLSSVM, and the IRRLSSVM are proposed based on the same criterion and select sample with the largest reduction to the objective function iteratively [6, 7, 14]. In the growing iterations, training samples yielding the largest reduction

to the target function is selected to construct the support vector set. The trained RRLSSVM is believed to be the global minimum of the RLSSVM. Owing to the fact that the construction procedure of the RRLSSVM is a greedy growing algorithm; the RRLSSVM may not guarantee to achieve the real global optimal.

The *1*-norm term (or lasso loss) is introduced to the objective function of the LSSVM training to enhance sparsity in [16]. On the other hand, sparsity can also be achieved by applying different Lagrange multipliers to both different samples and different features [17]. Sparse kernel density estimation with a localized regularization is proposed to construct sparse LSSVMs in [18].

Sparse LSSVMs usually use fewer support vectors which lead to a faster prediction. In general, sparse LSSVMs yield better generalization capability in comparison to full dense LSSVMs because sparse LSSVMs reduce the chance of over-fitting.

3 L-GEM based Sparse LSSVMs

In this section, two new sparse LSSVMs, i.e. SMLSSVM and RQLSSVM, are proposed based on the L-GEM error upper bound for LSSVM [2]. The error bound based on L-GEM is denoted by R_{SM} (Q). The empirical error and the sensitivity measure are two major components of the L-GEM error bound for LSSVM. For the SMLSSVM, only the sensitivity measure term is used to evaluate the importance of each sample. On the other hand, the RQLSSVM uses both of the two major components of the L-GEM as the criterion for sample pruning.

3.1 L-GEM for LSSVM

The localized generalization error model (L-GEM) evaluates the generalization capability of a classifier by qualifying an upper bound of the mean square errors of it within a given neighborhood (i.e. the Q-neighborhood) of the given training samples [2, 19–22]. The Q-neighborhood of a sample $\mathbf{x}_b$ $(S_Q(\mathbf{x}_b))$ is defined as $S_Q(\mathbf{x}_b) = \{\mathbf{x}|\mathbf{x} = \mathbf{x}_b + \Delta\mathbf{x}; |\Delta\mathbf{x}_i| \le Q \,\forall i = 1, \ldots, n\}$ which consists of all non-training samples located within a n-dimensional hypercube centered at the training sample $\mathbf{x}_b$. The Q-neighborhood of the whole dataset (S_Q) is defined to be the union of all $S_Q(\mathbf{x}_b)$.

The traditional L-GEM model is not designed for enhancing the sparsity of LSSVMs, so modification is proposed here. The localized generalization error bound $R_{SM}(Q)$ is defined as the integral of the mean squares error of non-training samples located within S_Q:

$$\begin{aligned} R_{SM}(Q) &= \int_{S_Q} (f(\mathbf{x}) - y)^2 p(\mathbf{x}) d\mathbf{x} \\ &= \sum_{b=1}^{l} \int_{S_Q(\mathbf{x}_b)} (f(\mathbf{x}) - y)^2 p(\mathbf{x}) d\mathbf{x} \\ &= \sum_{b=1}^{l} \int_{S_Q(\mathbf{x}_b)} ((f(\mathbf{x}) - f(\mathbf{x}_b)) + (f(\mathbf{x}_b) - y_b) \\ &\quad + (y_b - y))^2 p(\mathbf{x}) d\mathbf{x} \\ &\le \sum_{b=1}^{l} \left(\sqrt{err_b^2} + \sqrt{E_{S_{Q_b}}\left((\Delta y)^2\right)} + A \right)^2 + \varepsilon \end{aligned} \tag{9}$$

where $err_b^2 = (f(\mathbf{x}_b) - y_b)^2$ and

$$E_{S_Q(\mathbf{x}_b)}((\Delta y)^2) = \int_{S_Q(\mathbf{x}_b)} (f(\mathbf{x}) - f(\mathbf{x}_b))^2 p(\mathbf{x}) d\mathbf{x}$$

denote the empirical mean square error and the sensitivity measure term (SM), respectively. ε is a constant and ignored in this work because they are fixed for all trained LSSVMs after the training dataset is given. $A = \int_{S_Q(\mathbf{x}_b)} (y - y_b)^2 p(\mathbf{x}) d\mathbf{x}$ and A is set to be 0 as we assume all unseen samples in the Q-neighborhood of $\mathbf{x}_b$ are in the same class as $\mathbf{x}_b$.

$E_{S_{Q(\mathbf{x}_b)}}$ for the Gaussian kernel support vector machines $f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i, \mathbf{x}_j) + b$ and

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \Big/ 2\sigma^2 \right)$$

is as follows:

$$\begin{aligned} E_{S_Q(\mathbf{x}_b)}\left((\Delta y)^2\right) &= \int_{S_{Q(\mathbf{x}_b)}} (f(\mathbf{x}) - f(\mathbf{x}_b))^2 p(\mathbf{x}) d\mathbf{x} \\ &= f^2(\mathbf{x}_b) - 2f(\mathbf{x}_b) I_1 + I_2 \end{aligned} \tag{10}$$

where $I_1 = \int_{S_Q(\mathbf{x}_b)} f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}$ and $I_2 = \int_{S_Q(\mathbf{x}_b)} f^2(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}$. Assume that input features are independent to each other, using the *erf* function $erf(x) = \frac{2}{\sqrt{\pi}} \int_x^{+\infty} e^{-t^2} dx$, we have:

$$\begin{aligned} I_1 &= \left(\frac{\sqrt{\pi}\sigma}{2\sqrt{2}Q} \right)^n \sum_{j=1}^{l} \alpha_j y_j \prod_{i=1}^{n} \left(erf\left(\frac{Q + x_{bi} - x_{ji}}{\sqrt{2\sigma^2}} \right) \right. \\ &\left. - erf\left(\frac{-Q + x_{bi} - x_{ji}}{\sqrt{2\sigma^2}} \right) \right) I_2 = \left(\frac{\sqrt{\pi}\sigma}{4Q} \right)^n \\ &\sum_{j,k=1}^{l} \alpha_j \alpha_k y_j y_k \exp\left(- \frac{\sum_{i=1}^{n} (x_{ki} - x_{ji})^2}{4\sigma^2} \right) \\ &\prod_{i=1}^{n} \left(\begin{array}{c} erf\left(\frac{2(Q + x_{bi}) - (x_{ji} + x_{ki})}{\sqrt{2\sigma^2}} \right) \\ - erf\left(\frac{2(-Q + x_{bi}) - (x_{ji} + x_{ki})}{\sqrt{2\sigma^2}} \right) \end{array} \right) \end{aligned} \tag{11}$$

3.2 SMRLSSVM

According to the definition of the SM and Eq. (10), the SM of a support vector indicates how much it affects the output of the LSSVM with respect to a given magnitude of input changes. Support vectors yielding small SM values do not have significant affects to final outputs of the LSSVM. Hence the SMRLSSVM prunes away samples yielding small SM values.

The sparse LSSVM training algorithm of the SMRLSSVM is as follows:

Algorithm 1. SMRLSSVM

Input: $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, Gaussian kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/2\sigma^2)$, the user defined percentage of support vectors for the sparse LSSVM (r).

Output: sparse LSSVM $f(\mathbf{x}) = \mathrm{sgn}(\sum_{i=1}^{S} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b)$, support vector set S and corresponding Lagrange parameters α_S and b.

Algorithm:

1. Train a full dense LSSVM (α, b) according to Eq. (4) and fine tune hyper-parameters γ and σ via a minimization of the L-GEM.
2. For all samples, compute the SM of each sample using Eq. (10).
3. Prune r percents of training samples yielding the smallest SM values. Other samples are used to form the support vector set S.
4. Retrain the RLSSVM (α_S, b) on S and D according to Eq. (6)

3.3 RQRLSSVM

In contrast to pruning samples yielding either minimum α magnitudes in other sparse LSSVM training algorithms or minimum SM in the SMRLSSVM, minimizing the whole L-GEM should lead to a sparse LSSVM yielding better generalization capability. So in the RQRLSSVM, both the training error and the SM in the L-GEM are used as the pruning criterion. Samples yielding both a small training error and a small SM are pruned. In the full dense LSSVM, the training error is proportional to the Lagrange parameters α. Hence, the RQRLSSVM prunes samples yielding small values of both the Lagrange parameters α and the SM.

The sparse LSSVM training algorithm of the RQRLSSVM is as follows:

Algorithm 2. RQRLSSVM

Input: $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, Gaussian kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \big/ 2\sigma^2\right)$, the user defined percentage of support vectors for the sparse LSSVM (r).

Output: sparse LSSVM $f(\mathbf{x}) = \mathrm{sgn}\left(\sum_{i=1}^{S} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b\right)$, support vector set S and Lagrange parameters α and b.

Algorithm:

1. Train a full dense LSSVM (α, b) according to Eq. (4) and fine tune hyper-parameters γ and σ using the L-GEM.
2. For all samples, compute the SM of each sample using Eq. (10).
3. Find the set of samples yielding the smallest r percents of SM and the set of samples yielding the smallest r percent of α. Samples appearing in both sets are pruned. Remaining samples are used to form the support vector set S.
4. Retrain the RQRLSSVM (α_S, b) on S and D according to Eq. (6)

4 Experiments and discussion

The SMRLSSVM (SMR) and the RQRLSSVM (RQR) are compared with five other sparse methods on 10 datasets in this section. The sparse methods includes: the Spectral-based One-step LSSVM (SpecO) [15], the Spectral-based Iterative LSSVM (SpecI) [15], the IP-LSSVM (IP) [12], the SMO-based Sparse LSSVM (SMO) [13], and the Improved Recursive Reduced LSSVM (IRR) [7]. Eight UCI datasets include the Winston breast cancer (wbc), the credit approval (crx), the German credit (ger), the heart disease (hea), the hepatitis (hep), the ionosphere (ion), the pima (pima), and the sonar (son). Two artificial datasets are the chessboard (chb) and the exclusive OR (xor). The characteristics of UCI datasets are shown in Table 1. Datasets are randomly divided into training and testing sets with 2/3 and 1/3 of samples, respectively. Twenty independent runs are performed to reduce the random effects in our experiments. Program codes are modified based on the traditional LSSVM toolkit LS-SVMLab v1.7. The hyper-parameters are selected for the full dense LSSVM using the method proposed in [2].

There are two pruning strategies to obtain sparsity for the LSSVM: one-step and iterative. The first one prunes a predefined number of samples in one turn while the later one prunes a small portion of samples and update the LSSVM iteratively. Experimental results show that these two strategies yield almost the same generalization capability. However, in some cases, the iterative strategy is less efficient while usually uses less support vectors implying for a better sparsity. Section 4.1 examines characteristics

Table 1 Characteristics of eleven datasets

datasets	chb	xor	wbc	crx	ger	hea	hep	ion	pima	son
#sample	270	400	683	653	1000	270	80	351	768	208
#feature	2	2	9	15	24	13	19	33	8	60

of support vectors being selected and decision boundaries of trained sparse LSSVM on the artificial dataset chb. Section 4.2 shows experimental results of different methods on 10 datasets.

4.1 Experiments on artificial datasets

Experiments are carried out using the 3-by-3 chessboard (chb) problem. Figure 1 visualizes pruning results of different methods, training samples, selected support vectors (i.e. samples not being pruned) and corresponding classification decision boundaries for the chb in Fig. 1. In Fig. 1, there are two plots for the results of each method. The top one shows training samples in two classes (green pluses and red dots) and selected support vectors, i.e. samples remained after pruning (blue squares and green circles). The lower one shows contours formed by real-valued sparse LSSVM outputs and different colors show region being classified into different classes. Testing accuracy and number of support vectors of each method are listed on the left-hand-side and the lower-center of the corresponding top-plot.

Comparing to the original full dense LSSVM, all the 7 sparse LSSVMs achieve a better testing accuracies. The RQR achieves the best testing accuracy, i.e. 97.78 % while the SMR, the SMO, and the SpecI achieves the second best testing accuracy of 95.56 %. However, the number of support vectors selected by the SMO is 151 which means only 16.11 % of training samples are pruned. The LSSVM trained using the SMO is not sparse because the bias term b is removed from its optimization objective function. The support vectors pruned by both the SpecO and the IP are almost the same as we explained in Sect. 2.2. Decision boundaries of the sparse LSSVM generated by the iterative SpecI are smoother than that of the one-step SpecO and yield better generalization capabilities. The SMR generates the smoothest classifier because it trains the sparse LSSVM using samples yielding the largest sensitivity. Therefore, the sparse LSSVM is less sensitive after training by the SRM. However, the error term is ignored in the support vector selection, so decision boundaries of the sparse LSSVM trained using the SMR are distorted and yield worse testing accuracy. The IRR is a forward growing algorithm which selects representative samples as support vectors instead of pruning in other methods. The sparsity of the LSSVM generated by the IRR is the largest among all methods. However, the IRR uses a greedy procedure to select training samples which leads to a suboptimal selection in iterations and the selection in each turn is heavily influenced by the selection in the previous turn. Overall, the RQR achieves the best testing accuracy (97.78 %) and the smoothest classifier with the most reasonable decision boundaries. The decision boundaries of the sparse LSSVM yielded by the RQR are the best reproduction of the real decision boundaries of the 3×3 chessboard. The sparse LSSVM trained by the RQR yields the smoothest decision boundaries which avoid over-fitting in comparison to other methods in experiments. Moreover, the RQR only selects 12.22 % of training samples as the support vectors which yields a good sparsity.

4.2 Experiments on 10 datasets

All the 8 UCI benchmarking datasets and the 2 artificial datasets are normalized to follow a standard normal distribution $X \sim N(0,1)$. For all pruning procedures, 5 % of samples are pruned once. For all sparse algorithms, the proportion of the number of support vectors remained is determined using a 10-fold cross validation.

The averages and standard deviations of testing classification accuracies over 20 independent runs using different methods for the 10 datasets are shown in Table 2. The t test is performed between the RQR and other methods for experiments using each database. Experiments are marked by * and # when the RQR outperforms a method by 99 and 95 % statistical confidences, respectively. We also list the proportion of support vectors selected for these methods in Table 3.

Figure 2 shows both the average testing accuracies and average percentages of support vectors being selected from training samples over all datasets. The original full dense LSSVM yields the worst generalization capability because of overfitting by using all training samples as support vectors. The SMO omits the bias term and selects only one sample in a turn to avoid finding the most violating pair. Therefore, the SMO prunes the least number of training samples. The LSSVM created by the IRR has the largest sparsity over all methods. The RQR yields the best average

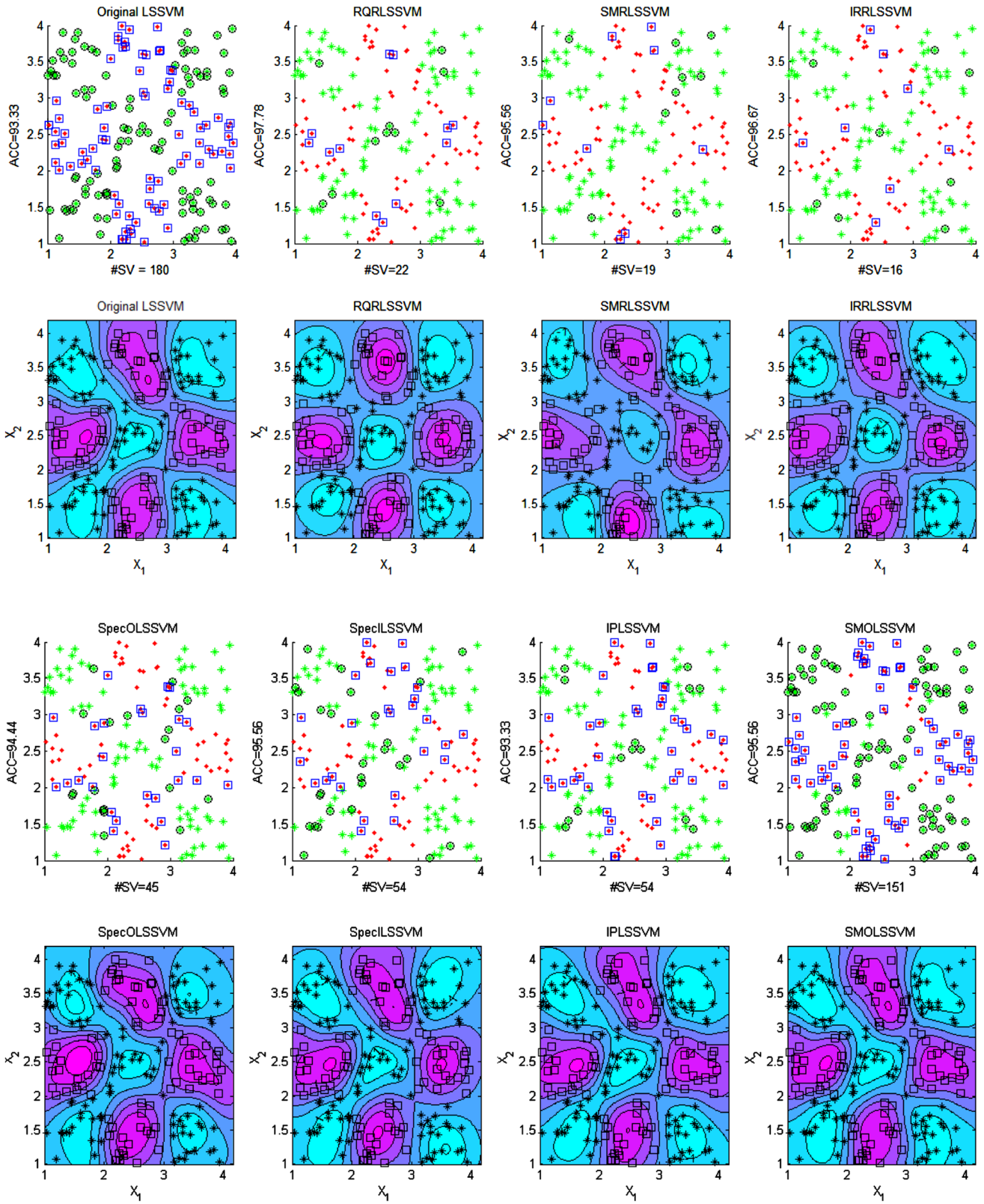

Fig. 1 Training samples, selected support vectors and corresponding classification decision boundaries using different methods for the 2-D 3 × 3 chessboard problem

Table 2 Averages and standard deviations of classification accuracies and t-test of different methods for 10 benchmark datasets over 20 independent runs

Datasets	LSSVM	SpecO	SpecI	IP	SMO	IRR	SMR	RQR
chb	92.56 ± 2.72*	93.17 ± 2.56*	93.33 ± 2.95*	93.11 ± 2.81*	95.00 ± 2.06#	95.06 ± 1.89#	95.72 ± 1.78	96.56 ± 1.69
xor	96.19 ± 2.22*	96.46 ± 2.19*	96.57 ± 2.20*	96.53 ± 2.10*	97.46 ± 1.85#	97.31 ± 2.02#	97.46 ± 1.75#	98.54 ± 1.31
wbc	96.75 ± 0.92*	96.82 ± 0.93*	96.91 ± 0.94#	96.82 ± 0.93*	97.21 ± 1.42	97.08 ± 1.03	97.28 ± 1.00	97.63 ± 0.94
crx	85.50 ± 2.13*	86.51 ± 1.95#	86.35 ± 1.83*	86.51 ± 1.95#	85.07 ± 9.37	86.86 ± 1.87#	87.41 ± 1.85	88.19 ± 1.96
ger	76.32 ± 1.55*	77.11 ± 1.64*	77.07 ± 1.57*	77.11 ± 1.64*	75.97 ± 0.81*	77.57 ± 1.72	78.07 ± 1.70	78.65 ± 1.76
hea	83.17 ± 3.35*	84.56 ± 3.12#	84.33 ± 3.34#	84.56 ± 3.12#	82.00 ± 9.39#	84.83 ± 3.09#	85.44 ± 3.14	87.06 ± 3.33
hep	70.19 ± 7.16*	73.70 ± 6.68	72.96 ± 7.89	73.52 ± 6.83	73.70 ± 8.23	74.63 ± 6.39	76.11 ± 5.82	76.85 ± 6.11
ion	94.74 ± 1.97*	95.60 ± 1.67	95.51 ± 1.94	95.64 ± 1.59	91.28 ± 2.55*	95.81 ± 1.88	95.98 ± 1.92	96.45 ± 1.69
pima	76.96 ± 2.05*	77.57 ± 2.01#	77.59 ± 2.27#	77.57 ± 2.01#	76.38 ± 5.32#	78.11 ± 2.30	78.58 ± 2.27	79.34 ± 2.24
son	85.29 ± 3.82#	87.07 ± 3.89	86.36 ± 4.26	87.43 ± 4.03	88.29 ± 3.49	86.71 ± 3.74	87.50 ± 3.85	88.29 ± 4.00

Table 3 Average percentages of training samples being selected as support vectors (%) for 10 benchmark datasets over 20 independent runs

Datasets	SpecO	SpecI	IP	SMO	IRR	SMR	RQR
chb	36.26	31.50	35.51	76.20	9.84	20.94	19.69
xor	25.93	27.11	22.20	63.21	18.36	29.21	20.46
wbc	12.79	11.02	14.29	26.05	12.46	14.09	2.47
crx	20.01	12.24	20.01	56.78	4.43	26.19	12.38
ger	39.72	23.92	39.97	84.82	10.12	22.96	11.87
hea	20.00	10.75	20.00	31.64	3.19	9.58	7.47
hep	39.56	22.31	35.53	33.56	15.28	38.63	22.22
ion	39.10	32.82	37.33	79.44	19.79	52.41	28.89
pima	17.00	11.45	17.00	67.41	3.23	10.70	4.11
son	55.74	60.65	65.23	91.55	58.95	74.67	48.14
avg	29.99	23.59	30.17	59.39	16.20	29.94	17.77

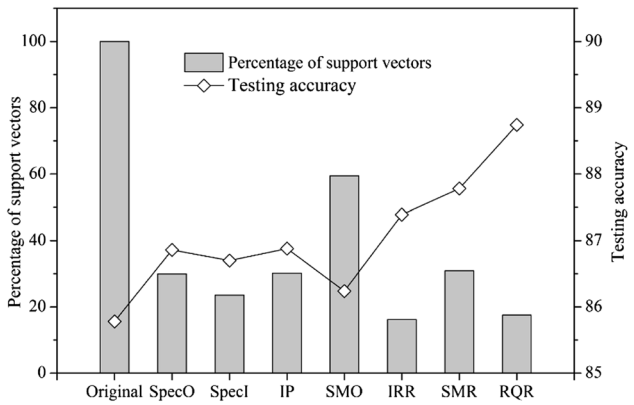

Fig. 2 Average testing accuracies and average percentages of support vectors using different sparse methods

testing accuracies and removes almost the largest portions of training samples among all methods in comparisons. The SMR method is worse than the RQR because it ignores the

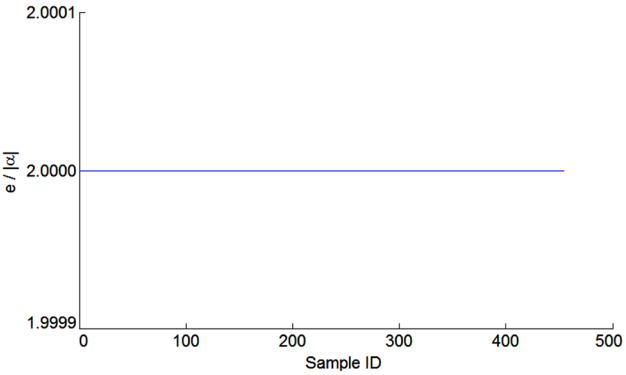

Fig. 3 $e/|\alpha|$ on *wbc* with $\gamma = 0.5$

error term during selection. The sparsity of LSSVMs generated by the RQR is only a bit worse than that of the IRR's. The squared training error and the sensitivity measure are the two major components of the localized generalization error model. These two terms are applied in a vector form instead of the original linear combination form to reduce the influence of the choice of the Q value. The RQR prunes samples yielding small values in both terms. Therefore, the sparse LSSVM yields a smoother decision boundary and is not easily overfit.

Figure 3 shows the ratios between the Lagrange parameter α for experiments of the wbc and the training error e are very stable. Therefore, a small α indicates a small e and we can use α to represent e during the sample pruning.

5 Conclusion

In this paper, we propose two new sparse LSSVM training algorithms, the SMRLSSVM and the RQRLSSVM, based on the Localized Generalization Error of the LSSVM.

Experimental results show that LSSVMs trained using the RQRLSSVM yield the best generalization capability and almost the best sparsity among other sparse algorithms in comparison. In our future works, we will explore the applications of the RQRLSSVM and the SMRLSSVM for big data problems [23] and integrations with extreme learning [24].

Acknowledgments This work was supported by the National Natural Science Foundation of China (61272201 and 61572201) and the Fundamental Research Funds for the Central Universities (2015ZZ023).

References

1. Suykens JAK, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process Lett 9(3):293–300
2. Sun BB et al (2013) Hyper-Parameter Selection for Sparse Ls-Svm Via Minimization of Its Localized Generalization Error. Int J Wavel Multiresolution Inf Process 11(3):1350030
3. van Gestel T et al (2004) Benchmarking least squares support vector machine classifiers. Mach Learn 54(1):5–32
4. Jiao LC, Bo LF, Wang L (2007) Fast sparse approximation for least squares support vector machine. IEEE Trans Neural Networks 18(3):685–697
5. Yang LX et al (2014) Sparse least square support vector machine via coupled compressive pruning. Neurocomputing 131:77–86
6. Zhao YP, Sun JG (2009) Recursive reduced least squares support vector regression. Pattern Recogn 42(5):837–842
7. Zhao YP et al (2012) An improved recursive reduced least squares support vector regression. Neurocomputing 87:1–9
8. Downs T, Gates KE, Masters A (2002) Exact simplification of support vector solutions. J Mach Learn Res 2(2):293–297
9. Huang CH (2012) A reduced support vector machine approach for interval regression analysis. Inf Sci 217:56–64
10. Lee YJ, Huang SY (2007) Reduced support vector machines: a statistical theory. IEEE Trans Neural Networks 18(1):1–13
11. Lin KM, Lin CJ (2003) A study on reduced support vector machines. IEEE Trans Neural Networks 14(6):1449–1459
12. Carvalho BPR, Braga AP (2009) IP-LSSVM: a two-step sparse classifier. Pattern Recogn Lett 30(16):1507–1515
13. Zeng XY, Chen XW (2005) SMO-based pruning methods for sparse least squares support vector machines. IEEE Trans Neural Netw 16(6):1541–1546
14. Zhao YP, Wang KK, Li F (2015) A pruning method of refining recursive reduced least squares support vector regression. Inf Sci 296:160–174
15. Suykens JA, Lukas L, Vandewalle J (2000) Sparse approximation using least square vector machines. In: IEEE Proc. Int. Symp. Circuits Syst., pp 757–760
16. Gao JB, Kwan PW, Shi DM (2010) Sparse kernel learning with LASSO and Bayesian inference algorithm. Neural Netw 23(2):257–264
17. Liu JL et al (2011) A weighted L-q adaptive least squares support vector machine classifier—Robust and sparse approximation. Expert Syst Appl 38(3):2253–2259
18. Chen S, Hong X, Harris CJ (2008) An orthogonal forward regression technique for sparse kernel density estimation. Neurocomputing 71(4–6):931–943
19. Yeung DS, Li J-C, Ng WWY, Chan PPK (2016) MLPNN training via a multiobjective optimization of training error and stochastic sensitivity. IEEE Trans Neural Netw Learn Syst 27(5):978–992
20. Ng WWY et al (2008) Feature selection using localized generalization error for supervised classification problems using RBFNN. Pattern Recogn 41(12):3706–3719
21. Yeung DS et al (2007) Localized generalization error model and its application to architecture selection for radial basis function neural network. IEEE Trans Neural Netw 18(5):1294–1305
22. Ng WWY et al (2015) Diversified sensitivity-based undersampling for imbalance classification problems. IEEE Tras Cybern 45(11):2402–2412
23. Wang XZ (2015) Learning from big data with uncertainty—editorial. J Intell Fuzzy Syst 28(5):2329–2330
24. Lu SX, Wang XZ, Zhang GQ, Zhou X (2015) Effective algorithms of the Moore–Penrose inverse matrices for extreme learning machine. Intell Data Anal 19(4):743–760