Selecting a Reduced Set for Building Sparse Support Vector Regression in the Primal

Liefeng Bo, Ling Wang, and Licheng Jiao

Institute of Intelligent Information Processing
Xidian University, Xi'an 710071, China
{blf0218, wliiip}@163.com
http://see.xidian.edu.cn/graduate/lfbo

Abstract. Recent work shows that Support vector machines (SVMs) can be solved efficiently in the primal. This paper follows this line of research and shows how to build sparse support vector regression (SVR) in the primal, thus providing for us scalable, sparse support vector regression algorithm, named SSVR-SRS. Empirical comparisons show that the number of basis functions required by the proposed algorithm to achieve the accuracy close to that of SVR is far less than the number of support vectors of SVR.

1 Introduction

Support vector machines (SVMs) [1] are powerful tools for classification and regression. Though very successful, SVMs are not preferred in application requiring high test speed since the number of support vectors typically grows linearly with the size of the training set [2]. For example in on-line classification and regression, in addition to good generalization performance, high test speed is also desirable. Reduced set (RS) methods [3-4] have been proposed for reducing the number of support vectors. Since these methods operate as a post-processing step, they do not directly approximate the quantity we are interested in. Another alternative is the reduced support vector machines (RSVM) [5], where the decision function is expressed as a weighted sum of kernel functions centered on a random subset of the training set. Though simple and efficient, RSVM may result in a lower accuracy than the reduced set methods when their number of support vectors is kept in the same level.

Traditionally, SVMs are trained by using decomposition techniques such as SVMlight [6] and SMO [7], which solve the dual problem by optimizing a small subset of the variables each iteration. Recently, some researchers show that both linear and non-linear SVMs can be solved efficiently in the primal. As for linear SVMs, finite Newton algorithm [8-9] has proven to be more efficient than SMO. As for non-linear SVM, recursive finite Newton algorithm [10-11] is as efficient as the dual domain method. Intuitively, when our purpose is to compute an approximate solution, the primal optimization is preferable to the dual optimization because it directly minimizes the quantity we are interested in. On the contrary, introducing approximation in the dual may not be wise since there is indeed no guarantee that an approximate dual solution yields a good approximate primal solution. Chapelle

[10] compares the approximation efficiency in the primal and dual domain and validates this intuition.

In this paper, we develop a novel algorithm, named SSVR-SRS for building the reduced support vector regression. Unlike our previous work [11] where recursive finite Newton algorithm is suggested to solve SVR accurately, SSVR-SRS aims to find a sparse approximation solution, which is closely related to SpSVM-2 [12] and kernel matching pursuit (KMP) [13], and can be regarded as extension of the key idea of matching pursuit to SVR. SSVR-SRS iteratively builds a set of basis functions to decrease the primal objective function by adding one basis function at one time. This process is repeated until the number of basis functions has reached some specified value. SSVR-SRS can find the approximate solution at a rather low cost, i.e. $O(nm^2)$ where n is the number of training samples and m the number of all picked basis functions. Our experimental results demonstrate the efficiency and effectiveness of the proposed algorithms.

The paper is organized as follows. In Section 2, support vector regression in the primal is introduced. SSVR-SRS is discussed in Section 3. Comparisons with RSVM, LIBSVM 2.82 [14] and the reduced set method are reported in Section 4. Some conclusions and remarks are given in Section 5.

2 Support Vector Regression in the Primal

Consider a regression problem with training samples $\{\mathbf{x}_i, y_i\}_{i=1}^n$ where $\mathbf{x}_i$ is the input sample and y_i is the corresponding target. To obtain a linear predictor, SVR solves the following optimization problem

$$\begin{aligned} &\min_{\mathbf{w},b}\left(\frac{\|\mathbf{w}\|^2}{2}+C\sum_{i=1}^{n}\left(\xi_i^p+\overline{\xi}_i^p\right)\right) \\ &s.t.\ \mathbf{w}\cdot\mathbf{x}_i+b-y_i\le\varepsilon+\xi_i \\ &\quad y_i-\mathbf{w}\cdot\mathbf{x}_i+b\le\varepsilon+\overline{\xi}_i \\ &\quad \xi_i,\overline{\xi}_i\ge 0,\ i=1,2,\cdots n \end{aligned} \quad . \tag{1}$$

Eliminating the slack variables $\left\{\xi_i,\overline{\xi}_i\right\}_{i=1}^n$ and dividing (1) by the factor C, we get the unconstrained optimization problem

$$\min_{\mathbf{w},b}\left(L_\varepsilon(\mathbf{w},b)=\sum_{i=1}^{n}l_\varepsilon(\mathbf{w}\cdot\mathbf{x}_i+b-y_i)+\lambda\|\mathbf{w}\|^2\right), \tag{2}$$

where $\lambda=\frac{1}{2C}$ and $l_\varepsilon(r)=\max(|r|-\varepsilon,0)^p$. The most popular selections for p are 1 and 2. For convenience of expression, the loss function with p=1 is referred to as insensitive linear loss function (ILLF) and that with p=2 insensitive quadratic loss function (IQLF).

Non-linear SVR can be obtained by using the map $\phi(\bullet)$ which is determined implicitly by a kernel function $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)\bullet\phi(\mathbf{x}_j)$. The resulting optimization is

$$\min_{\mathbf{w},b}\left(L_\varepsilon(\mathbf{w},b) = \sum_{i=1}^{n} l_\varepsilon\left(\mathbf{w}\cdot\phi(\mathbf{x}_i) - y_i\right) + \lambda\|\mathbf{w}\|^2 \right), \tag{3}$$

where we have dropped b for the sake of simplicity. Our experience shows that the generalization performance of SVR is not affected by this drop. According to the representer theory [15], the weight vector $\mathbf{w}$ can be expressed in terms of training samples,

$$\mathbf{w} = \sum_{i=1}^{n} \beta_i \phi(\mathbf{x}_i). \tag{4}$$

Substituting (4) into (3), we have

$$\min_{\boldsymbol{\beta}}\left(L_\varepsilon(\boldsymbol{\beta}) = \sum_{i=1}^{n} l_\varepsilon\left(\sum_{j=1}^{n} \beta_i k(\mathbf{x}_i \mathbf{x}_j) - y_i \right) + \lambda \sum_{i=1}^{n} \beta_i \beta_j k(\mathbf{x}_i \mathbf{x}_j) \right). \tag{5}$$

Introducing the kernel matrix $\mathbf{K}$ with $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{K}_i$ the i-th row of $\mathbf{K}$, (5) can be rewritten as

$$\min_{\boldsymbol{\beta}}\left(L_\varepsilon(\boldsymbol{\beta}) = \sum_{i=1}^{n} l_\varepsilon(\mathbf{K}_i\boldsymbol{\beta} - y_i) + \lambda\boldsymbol{\beta}^T\mathbf{K}\boldsymbol{\beta} \right). \tag{6}$$

A gradient descent algorithm is straightforward for IQLF; however, it is not applicable to ILLF since it is not differentiable. Inspired by the Huber loss function [16], we propose an insensitive Huber loss function (IHLF)

$$l_{\varepsilon,\Delta}(z) = \begin{cases} 0 & if\ |z| \le \varepsilon \\ (|z| - \varepsilon)^2 & if\ \varepsilon < |z| < \Delta, \\ (\Delta - \varepsilon)(2|z| - \Delta - \varepsilon) & if\ |z| \ge \Delta \end{cases} \tag{7}$$

to approximate ILLF. We emphasize that Δ is strictly greater than ε, ensuring that IHLF is differentiable.

The properties of IHLF are controlled by two parameters: ε and Δ. With certain ε and Δ values, we can obtain some familiar loss functions: (1) for $\varepsilon = 0$ and an appropriate Δ, IHLF becomes the Huber loss function; (2) for $\varepsilon = 0$ and $\Delta = \infty$, IHLF becomes the quadratic (Gaussian) loss function; (3) for $\varepsilon = 0$ and $\Delta \to \varepsilon$, IHLF approaches the linear (Laplace) loss function; (4) for $0 < \varepsilon < \infty$ and $\Delta = \infty$, IHLF becomes the insensitive quadratic loss function; and, (5) for $0 < \varepsilon < \infty$ and $\Delta \to \varepsilon$, IHLF approaches the insensitive linear loss function.

Introducing IHLF into the optimization problem (6), we have the following primal objective function:

$$\min_{\boldsymbol{\beta}}\left(L_{\varepsilon,\Delta}(\boldsymbol{\beta}) = \sum_{i=1}^{n} l_{\varepsilon,\Delta}(\mathbf{K}_i\boldsymbol{\beta} - y_i) + \lambda\boldsymbol{\beta}^T\mathbf{K}\boldsymbol{\beta} \right). \tag{8}$$

3 Selecting a Reduced Set in the Primal

In a reduced SVR, it is desirable to decrease the primal objective function as much as possible with as few basis functions as possible. The canonical form of this problem is given by

$$\begin{aligned} &\min\left(L_{\varepsilon,\Delta}(\boldsymbol{\beta}) = \sum_{i=1}^{n} l_{\varepsilon,\Delta}\left(\mathbf{K}_i\boldsymbol{\beta} - y_i\right) + \lambda\boldsymbol{\beta}^T\mathbf{K}\boldsymbol{\beta} \right), \\ &s.t. \left\|\boldsymbol{\beta}\right\|_0 \le m \end{aligned} \tag{9}$$

where $\|\bullet\|_0$ is the l^0 norm, counting the nonzero entries of a vector and m is the specified maximum size of basis functions. However, there are several difficulties in solving (9). First, the constraint is not differentiable, so gradient descent algorithms can not be used. Second, the optimization algorithms can become trapped in a shallow local minimum because there are many minima to (9). Finally, an exhaustive search over all possible choices ($\|\boldsymbol{\beta}\|_0 \le m$) is computational prohibitive since the number of possible combinations is $\sum_{i=1}^{m} \binom{n}{m}$, too large for current computers.

Table 1. Flowchart of SSVR-SRS

```
Algorithm 3.1 SSVR-SRS
1. Set $P = \varnothing$, $Q = \{1, 2, \cdots, n\}$, $\boldsymbol{\beta} = \mathbf{0}$;
2. Select a new basis function from $Q$; let $s$ be its index and set
   $P = P \cup \{s\}$ and $Q = Q - \{s\}$;
3. Solve the sub-problem with respect to $\boldsymbol{\beta}_P$ and the remaining variables
   are fixed at zero.
4. Check whether the number of basis functions is equal to $m$, if so,
   stop; otherwise go to step 2.
```

In this paper, we will compute an approximate solution using a matching pursuit-like method, named SSVR-SRS, to avoid optimizing (9) directly. SSVR-SRS starts with an empty set of basis functions and selects one basis function at one time to decrease the primal objective function until the number of basis functions has reached a specified value. Flowchart of SSVR-SRS is shown in Table 1. The final decision function takes the form

$$f(\mathbf{x}) = \sum_{i \in P} \beta_i k(\mathbf{x}, \mathbf{x}_i). \tag{10}$$

The set of the samples associated with the non-zero weights is called reduced set. Because here the reduced set is restricted to be a subset of training set, we consider this method as "selecting a reduced set".

3.1 Selecting Basis Function

Let $\mathbf{K}_P$ the sub-matrix of $\mathbf{K}$ made of the columns indexed by P, $\mathbf{K}_{XY}$ the sub-matrix of $\mathbf{K}$ made of the rows indexed by X and the columns indexed by Y and $\boldsymbol{\beta}_P$ the sub-vector indexed by P.

How do we select a new basis function from Q? A natural idea is to optimize the primal objective function with respect to the variables $\boldsymbol{\beta}_P$ and β_j for each $j \in Q$ and select the basis function giving the least objective function value. The selection process can be described as a two-layer optimization problem,

$$s = \arg\min_{j\in Q}\left(\min_{\boldsymbol{\beta}_P,\beta_j}\left(L_{\varepsilon,\Delta}(\boldsymbol{\beta}) = \sum_{i=1}^{n} l_{\varepsilon,\Delta}\left(\mathbf{K}_{iP}\boldsymbol{\beta}_P + \mathbf{K}_{ij}\boldsymbol{\beta}_j - y_i\right) + \lambda\begin{bmatrix}\boldsymbol{\beta}_P\\ \beta_j\end{bmatrix}^T\begin{bmatrix}\mathbf{K}_{PP} & \mathbf{K}_{Pj}\\ \mathbf{K}_{jP} & \mathbf{K}_{jj}\end{bmatrix}\begin{bmatrix}\boldsymbol{\beta}_P\\ \beta_j\end{bmatrix}\right)\right). \quad (11)$$

This basis function selection method, called pre-fitting, has appeared in kernel matching pursuit for least squares problem. Unfortunately, pre-fitting needs to solve the $|P|+1$ dimensional optimization problem $|Q|$ times, the cost of which is obviously higher than that of optimizing the sub-problem.

A cheaper method is to select the basis function that best fits the current residual vector in terms of a specified loss function. This method originated from matching pursuit [17] for least squares problem and was extended to an arbitrary differentiable loss function in gradient boosting [18]. However, our case is more complicated due to the occurrence of the regularization term, and thus we would like to select the basis function that fits the current residual vector and the regularization term as well as possible. Let the current residual vector be

$$\begin{cases}\mathbf{r}\left(\boldsymbol{\beta}_P^{opt}\right) = \mathbf{K}_P\boldsymbol{\beta}_P^{opt} - \mathbf{y}\\ r_i\left(\boldsymbol{\beta}_P^{opt}\right) = \mathbf{K}_{iP}\boldsymbol{\beta}_P^{opt} - y_i\end{cases}, \quad (12)$$

where $\boldsymbol{\beta}_P^{opt}$ is the optimal solution obtained by solving the sub-problem, and the index of basis function can be obtained by solving the following two-layer optimization problem,

$$s = \arg\min_{j\in Q}\left(\min_{\beta_j}\left(L_{\varepsilon,\Delta}(\beta_j) = \sum_{i=1}^{n} l_{\varepsilon,\Delta}\left(r_i\left(\boldsymbol{\beta}_P^{opt}\right) + \mathbf{K}_{ij}\beta_j\right) + \lambda\begin{bmatrix}\boldsymbol{\beta}_P^{opt}\\ \beta_j\end{bmatrix}^T\begin{bmatrix}\mathbf{K}_{PP} & \mathbf{K}_{Pj}\\ \mathbf{K}_{jP} & \mathbf{K}_{jj}\end{bmatrix}\begin{bmatrix}\boldsymbol{\beta}_P^{opt}\\ \beta_j\end{bmatrix}\right)\right). \quad (13)$$

Note that unlike pre-fitting, here $\boldsymbol{\beta}_P^{opt}$ is fixed.

$L_{\varepsilon,\Delta}(\beta_j)$ is one dimensional, piecewise quadratic function and can be minimized exactly. However, in practice, it is not necessary to solve it precisely. A simpler method is to compare the square of the gradient of $L_{\varepsilon,\Delta}(\beta_j)$ at $\beta_j = 0$ for all $j \in Q$,

$$\left(\nabla L_{\varepsilon,\Delta}(0)\right)^2 = \left(\mathbf{g}^T\mathbf{K}_j + 2\lambda\boldsymbol{\beta}_P^{opt T}\mathbf{K}_{Pj}\right)^2, \quad (14)$$

where

$$g_i = \begin{cases} 0 & if \left|r_i\left(\boldsymbol{\beta}_P^{opt}\right)\right| \le \varepsilon \\ 2sign\left(r_i\left(\boldsymbol{\beta}_P^{opt}\right)\right)\left(\left|r_i\left(\boldsymbol{\beta}_P^{opt}\right)\right| - \varepsilon\right) & if\ \varepsilon < \left|r_i\left(\boldsymbol{\beta}_P^{opt}\right)\right| < \Delta \\ 2sign\left(r_i\left(\boldsymbol{\beta}_P^{opt}\right)\right)(\Delta - \varepsilon) & if \left|r_i\left(\boldsymbol{\beta}_P^{opt}\right)\right| \ge \Delta \end{cases}, \tag{15}$$

where $sign(z)$ is 1 if $z \ge 0$; otherwise $sign(z)$ is -1. To be fair, the square of the gradient should be normalized to

$$\frac{\left(\bar{\mathbf{g}}^T \bar{\mathbf{K}}_j\right)^2}{\left\|\bar{\mathbf{g}}\right\|_2^2 \left\|\bar{\mathbf{K}}_j\right\|_2^2}, \tag{16}$$

where $\bar{\mathbf{g}} = \begin{bmatrix} \mathbf{g} \\ 2\lambda\boldsymbol{\beta}_P^{opt} \end{bmatrix}$ and $\bar{\mathbf{K}}_j = \begin{bmatrix} \mathbf{K}_j \\ \mathbf{K}_{Pj} \end{bmatrix}$. This is an effective criterion because the gradient measures how well the j-th basis function fits the current residual vector and the regularization term. If set $\varepsilon = 0$, $\Delta = \infty$ and $\lambda = 0$, this criterion is exactly the one in the back-fitting version of KMP.

If each $j \in Q$ is tried, then the total cost of selecting a new basis function is $O\left(n^2\right)$, which is still more than what we want to accept. This cost can be reduced to $O(n)$ by only considering a random subset O of Q and selecting the next basis function only from O rather than performing an exhaustive search over Q,

$$s = \arg\min_{j \in O \subset Q} \left(\frac{-\left(\bar{\mathbf{g}}^T \bar{\mathbf{K}}_j\right)^2}{\left\|\bar{\mathbf{g}}\right\|_2^2 \left\|\bar{\mathbf{K}}_j\right\|_2^2} \right). \tag{17}$$

In the paper, we set $|O| = 100$.

3.2 Optimizing the Sub-problem

After a new basis function is included, the weights of basis functions, β_P are no longer optimal in terms of the primal objective function. This can be corrected by the so-called back-fitting method, which solves the sub-problem containing a new basis function and all previously picked basis functions. Thus, the sub-problem is a $|P|$ dimensional minimization problem expressed as

$$\min_{\boldsymbol{\beta}_P} \left(L_{\varepsilon,\Delta}\left(\boldsymbol{\beta}_P\right) = \sum_{i=1}^{n} l_{\varepsilon,\Delta}\left(\mathbf{K}_{iP}\boldsymbol{\beta}_P - y_i\right) + \lambda \boldsymbol{\beta}_P{}^T \mathbf{K}_{PP} \boldsymbol{\beta}_P \right). \tag{18}$$

$L_{\varepsilon,\Delta}\left(\boldsymbol{\beta}_P\right)$ is a piecewise quadratic convex function and continuously differentiable with respect to $\boldsymbol{\beta}_P$. Although $L_{\varepsilon,\Delta}\left(\boldsymbol{\beta}_P\right)$ is not twice differentiable, we still can use the finite Newton algorithm by defining the generalized Hessian matrix [11].

Define the sign vector $\mathbf{s}(\boldsymbol{\beta}_P)=\left[s_1(\boldsymbol{\beta}_P),\cdots,s_n(\boldsymbol{\beta}_P)\right]^T$ by

$$s_i(\boldsymbol{\beta}_P)=\begin{cases}1 & if\ \ \varepsilon<r_i(\boldsymbol{\beta}_P)<\Delta \\ -1 & if\ -\Delta<r_i(\boldsymbol{\beta}_P)<-\varepsilon \\ 0 & otherwise\end{cases}, \tag{19}$$

the sign vector $\bar{\mathbf{s}}(\boldsymbol{\beta}_P)=\left[\bar{s}_1(\boldsymbol{\beta}_P),\cdots,\bar{s}_n(\boldsymbol{\beta}_P)\right]^T$ by

$$\bar{s}_i(\boldsymbol{\beta}_P)=\begin{cases}1 & if\ \ r_i(\boldsymbol{\beta}_P)\geq\Delta \\ -1 & if\ r_i(\boldsymbol{\beta}_P)\leq-\Delta \\ 0 & otherwise\end{cases}, \tag{20}$$

and the active matrix

$$\mathbf{W}(\boldsymbol{\beta}_P)=diag\left\{w_1(\boldsymbol{\beta}_P),\cdots,w_n(\boldsymbol{\beta}_P)\right\} \tag{21}$$

by $w_i(\boldsymbol{\beta}_P)=s_i^2(\boldsymbol{\beta}_P)$. The gradient of $L_{\varepsilon,\Delta}(\boldsymbol{\beta}_P)$ with respect to $\boldsymbol{\beta}_P$ is

$$\nabla L_{\varepsilon,\Delta}(\boldsymbol{\beta}_P)=2\mathbf{K}_P^T\mathbf{W}(\boldsymbol{\beta}_P)\mathbf{r}(\boldsymbol{\beta}_P)-2\varepsilon\mathbf{K}_P^T\mathbf{s}(\boldsymbol{\beta}_P)+2(\Delta-\varepsilon)\mathbf{K}_P^T\bar{\mathbf{s}}(\boldsymbol{\beta}_P)+2\lambda\mathbf{K}_{PP}\boldsymbol{\beta}_P\,. \tag{22}$$

The generalized Hessian is

$$\nabla^2 L_{\varepsilon,\Delta}(\boldsymbol{\beta}_P)=2\mathbf{K}_P^T\mathbf{W}(\boldsymbol{\beta}_P)\mathbf{K}_P+2\lambda\mathbf{K}_{PP}\,. \tag{23}$$

The Newton step at the k-th iteration is given by

$$\boldsymbol{\beta}_P^{k+1}=\boldsymbol{\beta}_P^k-t\left(\nabla^2 L_{\varepsilon,\Delta}(\boldsymbol{\beta}_P^k)\right)^{-1}\nabla L_{\varepsilon,\Delta}(\boldsymbol{\beta}_P^k)\,. \tag{24}$$

The step size t can be found by a line search procedure that minimizes the one dimensional piecewise-smooth, convex quadratic function. Since the Newton step is much more expensive, the line search does not add to the complexity of the algorithm.

3.3 Computational Complexity

In SSVR-SRS, the most time-consuming operation is computing the Newton step (24). When a new basis function is added, it involves three main steps: computing the column $\mathbf{K}_s$, which is $O(n)$, computing the new elements of the generalized Hessian, which is $O(nm)$ and inverting the generalized Hessian that can be computed in an incremental manner [12], which is $O(m^2)$. When the active matrix $\mathbf{W}(\boldsymbol{\beta}_P)$ is changed, the inversion of the generalized Hessian needs to be updated again, which is $O(cm^2)$. In most cases, c is a small constant, so it is reasonable to consider $O(nm)$ as an expensive cost since $n\gg m$. Adding up these costs till m basis functions are chosen, we get an overall complexity of $O(nm^2)$.

4 Experiments

In this section, we evaluate the performance of SSVR-SRS on five benchmark data sets and compare them with SVM, the reduced set method and reduced SVM.

4.1 Experimental Details

SVR is constructed based on LIBSVM 2.82 where the second order information is used to select the working set. RSVM is implemented by our own Matlab code. The reduced set method determines the reduced vectors $\{\mathbf{z}_i\}_{i=1}^m$ and the corresponding expansion coefficients by minimizing

$$\left\|\mathbf{w}-\sum_{j=1}^{m}\alpha_j\phi(\mathbf{z}_j)\right\|^2, \tag{25}$$

where $\mathbf{w}=\sum_{i\in S}\beta_i\phi(\mathbf{x}_i)$ is the weight vector obtained by optimizing (5) and S is the index set of support vectors. Reduced set selection (RSS) is parallel to SSVR-SRS and determines a new basis function by

$$s=\arg\min_{j\in O\subset Q}\left(\frac{-\left(\left[\boldsymbol{\beta}_S^T,-\boldsymbol{\alpha}_P^T\right]\begin{bmatrix}\mathbf{K}_{Sj}\\ \mathbf{K}_{Pj}\end{bmatrix}\right)^2}{\left\|\left[\boldsymbol{\beta}_S^T,-\boldsymbol{\alpha}_P^T\right]\right\|_2^2 k(\mathbf{x}_j,\mathbf{x}_j)}\right). \tag{26}$$

Five benchmark data sets: Abalone, Bank8fh, Bank32fh, House8l and Friedman3 are used in our empirical study. Information on these benchmark data sets is summarized in Table 2. These data sets have been extensively used in testing the performance of diversified kinds of learning algorithms. The first four data sets are available from Torgo's homepage: http://www.liacc.up.pt/~ltorgo/Regression/DataSets.html. Friedman3 is from [19]. The noise is adjusted for a 3:1 signal-to-noise ratio.

All the experiments were run on a personal computer with 2.4 GHz P4 processors, 2 GB memory and Windows XP operation system. Gaussian kernel $k(\mathbf{x}_i,\mathbf{x}_j)=\exp\left(-\gamma\left\|\mathbf{x}_i-\mathbf{x}_j\right\|_2^2\right)$ is used to construct non-linear SVR. The free parameters in the algorithms are determined by 10-fold cross validation except that Δ in the

Table 2. Information on benchmark data sets

Problem	Training	Test	Attribute	m
Abalone	3000	1177	8	50
Bank8fh	5000	4192	8	50
Bank32h	5000	4192	32	150
House8l	15000	7784	8	300
Friedman3	30000	20000	4	240

insensitive Huber loss function is fixed to 0.3. For each training-test pair, the training samples are scaled into the interval [-1, 1], and the test samples are adjusted using the same linear transformation. For SSVR-SRS, RSS and RSVM, the final results are averaged over five random implementations.

4.2 Comparisons with LIBSVM 2.82

Table 3-4 reports the generalization performance and the number of basis functions of SVR and SSVR-SRS. As we can see, compared with SVR, SSVR-SRS achieves the impressive reduction in the number of basis functions almost without sacrificing the generalization performance.

Table 3. Test error and number of basis functions of SVR, SSVR-SRS on benchmark data sets. Error denotes root-mean-square test error, Std denotes the standard deviation of test error and NBF denotes the number of basis functions. For SSVR-SRS, λ is set to be 1e-2 on the first four data sets and 1e-3 on Friedman3 data set.

Error	SVR		SSVR-SRS		
2.107	Error	NBF	Error	Std	NBF
Abalone	2.106	1152	2.107	0.006783	18
Bank8fh	0.071	2540	0.071	0.000165	40
Bank32nh	0.082	2323	0.083	0.000488	83
House8l	30575	2866	30796	126.106452	289
Friedman3	0.115	9540	0.115	0.000211	203

Table 4. Test error and number of basis functions of SVR, SSVR-SRS on benchmark data sets. For SSVR-SRS, λ is set to be 1e-5.

Problem	SVR		SSVR-SRS		
	Error	NBF	Error	Std	NBF
Abalone	2.106	1152	2.106	0.012109	17
Bank8fh	0.071	2540	0.071	0.000259	44
Bank32nh	0.082	2323	0.083	0.000183	119
House8l	30575	2866	30967	219.680790	282
Friedman3	0.115	9540	0.115	0.000318	190

4.3 Comparisons with RSVM and RSS

Figure 1-5 compare SSVR-SRS, RSVM and RSS on the five data sets. Overall, SSVR-SRS beats its competitors and achieves the best performance in terms of the decrease of test error with the number of basis functions. In most cases, RSVM is inferior to RSS, especially in the early stage. An exception is House8l data set where RSVM gives smaller test error than RSS when the number of basis functions is beyond some threshold value. SSVR-SRS significantly outperforms RSS on Bak32nh, House8l and Friedman3 data sets, but the difference between them becomes very small on the remaining data sets. SSVR-SRS is significantly superior to RSVM on

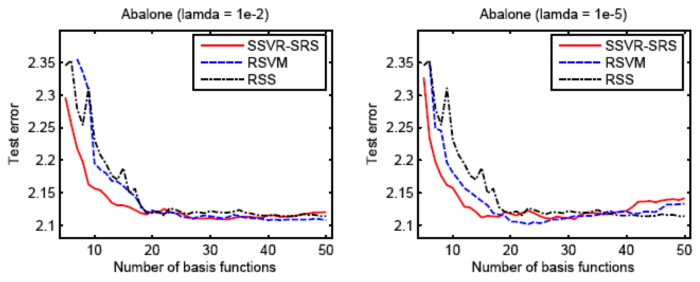

Fig. 1. Comparisons of SSVR-SRS, RSVM and RSS on Abalone

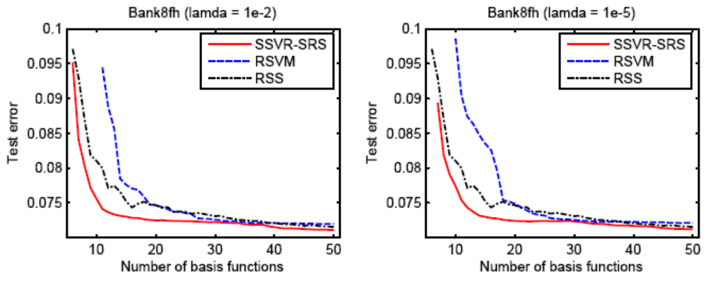

Fig. 2. Comparisons of SSVR-SRS, RSVM and RSS on Bank8fh

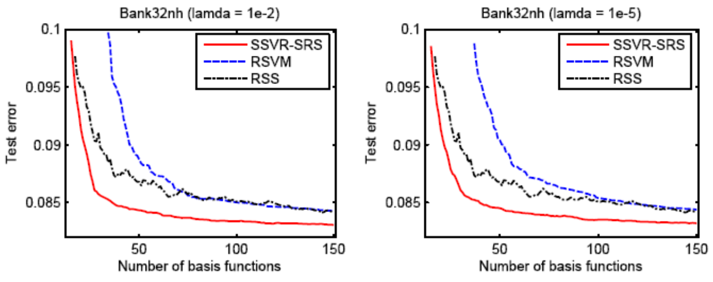

Fig. 3. Comparisons of SSVR-SRS, RSVM and RSS on Bank32nh

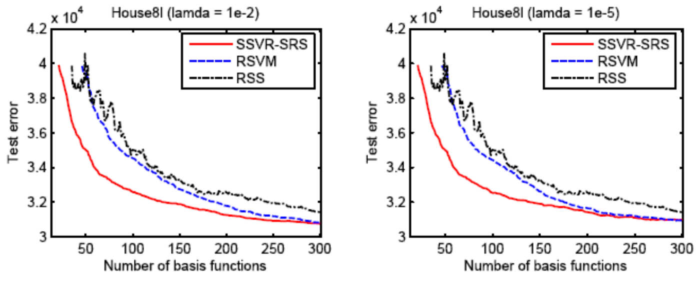

Fig. 4. Comparisons of SSVR-SRS, RSVM and RSS on House81

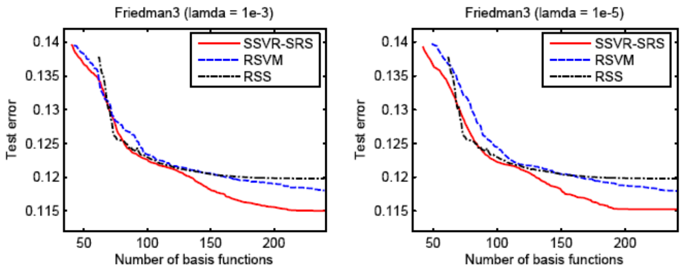

Fig. 5. Comparisons of SSVR-SRS, RSVM and RSS on Friedman3

four of the five data sets and comparable on the remaining data set. Another observation from Figure 1-5 is that SSVR-SRS with small regularization parameter starts over-fitting earlier than that with large regularization parameter, e.g. Abalone data set.

One phenomenon to note is that the reduced set selection has a large fluctuation in the generalization performance in the early stage. This is because the fact that, the different components of the weight vector $\mathbf{W}$ usually have a different impact on the generalization performance and therefore the better approximation to $\mathbf{W}$ does not necessarily leads to the better generalization performance. The fluctuation is alleviated with the increasing number of basis functions because the large number of basis functions can guarantee that each component of $\mathbf{W}$ is approximated well.

4.4 Training Time of SSVR-SRS

We do not claim that SSVR-SRS is more efficient than some state-of-the-art training decomposition algorithms such as SMO. Our main motivation is to point out that there is a way that can efficiently build a highly sparse SVR with the guaranteed generalization performance. In practice, depending on the number of basis functions, SSVR-SRS can be faster or slower than the decomposition algorithms. It is not fair to directly compare the training time of our algorithm with that of LIBSVM 2.82 since our algorithm is implemented by Matlab and however LIBSVM 2.82 by C++. But, we still list the training time in Table 5 as a rough reference.

Table 5. Training time of four algorithms on benchmark data sets

Problem	SSVR-SRS	RSVM	LIBSVM2.82	RSS
Abalone	5.73	2.59	1.70	2.85
Bank8fh	7.39	4.61	8.03	9.65
Bank32h	47.63	31.03	17.55	24.76
House8l	416.92	391.47	98.38	118.79
Fiedman3	565.59	462.57	1237.19	1276.42

5 Concluding Remarks

We have presented SSVR-SRS for building sparse support vector regression. Our method has three key advantages: (1) it directly approximates the primal objective function and is more reasonable than the post-processing methods; (2) it scales well with the number of training samples and can be applied to large scale problems; (3) it simultaneously considers the sparseness and generalization performance of the resulting learner.

This work was supported by the Graduate Innovation Fund of Xidian University (No. 05004).

References

1. Vapnik, V: Statistical Learning Theory. New York Wiley-Interscience Publication (1998)
2. Steinwart, I. Sparseness of support vector machines. Journal of Machine Learning Research 4 (2003) 1071–1105
3. Burges, C. J. C. and Schölkopf, B. Improving the accuracy and speed of support vector learning machines. Advances in Neural Information Processing System 9 (1997) 375-381
4. Schölkopf, B., Mika, S., Burges, C. J. C., Knirsch, P., K. Muller, K. R., Raetsch, G., and Smola, A. J. Input space vs. feature space in kernel-based methods. IEEE Transactions on Neural Networks 10 (1999) 1000-1017
5. Lee, Y. J. and Mangasarian, O. L. RSVM: Reduced support vector machines. In Proceedings of the SIAM International Conference on Data Mining. SIAM, Philadelphia (2001)
6. Joachims, T. Making large-scale SVM learning practical. In Advances in Kernel Methods - Support Vector Learning, MIT Press, Cambridge, Massachussetts (1999)
7. Platt, J. Sequential minimal optimization: a fast algorithm for training support vector machines. In Advance in Kernel Methods - Support Vector Learning, MIT Press, Cambridge, Massachussetts (1999)
8. Mangasarian, O. L. A finite Newton method for classification. Optimization Methods & Software 17(5) (2002) 913-929
9. Keerthi, S. S. and Decoste D. M. A modified finite Newton method for fast solution of large scale linear svms. Journal of Machine Learning Research 6 (2005) 341-361
10. Chapelle, O. Training a Support Vector Machine in the Primal. Neural Computation (2006) (Accepted)
11. Bo, L. F., Wang, L. and Jiao L. C. Recursive finite Newton algorithm for support vector regression in the primal. Neural Computation (2007), in press.
12. Keerthi, S. S., Chapelle, O., and Decoste D. Building Support Vector Machines with Reduced Classifier Complexity. Journal of Machine Learning Research 7 (2006) 1493-1515
13. Vincent, P. and Bengio, Y. Kernel matching pursuit. Machine Learning 48 (2002) 165-187
14. Fan, R. E., Chen P. H., and Lin C. J. Working Set Selection Using Second Order Information for Training Support Vector Machines. Journal of Machine Learning Research 6 (2005) 1889-1918
15. Kimeldorf, G. S. and Wahba G. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. Annals of Mathematical Statistics 41 (1970) 495-502
16. Huber, P. Robust Statistics. John Wiley, New York (1981)
17. Mallat, S. and Zhang, Z. Matching pursuit with time-frequency dictionaries. IEEE Transactions on Signal Processing 41(12) (1993) 3397-3415
18. Friedman, J. Greedy Function Approximation: a Gradient Boosting Machine. Annals of Statistics 29 (2001) 1189-1232
19. Friedman, J. Multivariate adaptive regression splines. Annals of Statistics 19(1) (1991) 1-141