EURASIP Journal on Applied Signal Processing 2004:8, 1135–1141
© 2004 Hindawi Publishing Corporation

An Improved Way to Make Large-Scale SVR Learning Practical

Quan Yong
Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai 200030, China
Email: quanysjtu@sjtu.edu.cn

Yang Jie
Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai 200030, China
Email: jieyang@sjtu.edu.cn

Yao Lixiu
Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai 200030, China
Email: lxyao@sjtu.edu.cn

Ye Chenzhou
Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai 200030, China
Email: chenzhou.ye@sjtu.edu.cn

Received 31 May 2003; Revised 9 November 2003; Recommended for Publication by John Sorensen

We first put forward a new algorithm of reduced support vector regression (RSVR) and adopt a new approach to make a similar mathematical form as that of support vector classification. Then we describe a fast training algorithm for simplified support vector regression, sequential minimal optimization (SMO) which was used to train SVM before. Experiments prove that this new method converges considerably faster than other methods that require the presence of a substantial amount of the data in memory.

Keywords and phrases: RSVR, SVM, sequential minimal optimization.

1. INTRODUCTION

In the last few years, there has been a surge of interest in support vector machine (SVM) [1]. SVM has empirically been shown to give good generalization performance on a wide variety of problems. However, the use of SVM is still limited to a small group of researchers. One possible reason is that training algorithms for SVM are slow, especially for large problems. Another explanation is that SVM training algorithms are complex, subtle, and sometimes difficult to implement.

In 1997, a theorem [2] was proved that introduced a whole new family of SVM training procedures. In a nutshell, Osuna's theorem showed that the global SVM training problem can be broken down into a sequence of smaller subproblems and that optimizing each subproblem minimizes the original quadratic problem (QP). Even more recently, the sequential minimal optimization (SMO) algorithm was introduced [3, 4] as an extreme example of Osuna's theorem in practice. Because SMO uses a subproblem of size two, each subproblem has an analytical solution. Thus, for the first time, SVM could be optimized without a QP solver.

In addition to SMO, other new methods [5, 6] have been proposed for optimizing SVM online without a QP solver. While these other online methods hold great promise, SMO is the only online SVM optimizer that explicitly exploits the quadratic form of the objective function and simultaneously uses the analytical solution of the size two cases.

Support vector regression (SVR) have nearly the same situation as SVM. In 1998, Smola and Schölkopf [7] gave an overview of the basic idea underlying SVMs for regression and function estimation. They also generalized SMO so that it can handle regression problems. A detailed discussion can also be found in Keerthi [8] and Flake [9]. Because one has to consider four variables, α_i, α_i^*, α_j, and α_j^*, in the regression, the training algorithm actually becomes very complex, especially, when data is nonsparse and when there are many support vectors in the solution—as is often the case in regression—because kernel function evaluations tend to dominate the runtime in this case, most of these variables do not converge to zero and its rate of convergence slows down dramatically.

In this work, we propose a new way to make SVR—a new regression technique based on the structural risk minimization principle—has a similar mathematical form as that of support vector classification, and derives a generalization of SMO to handle regression problems. Simulation results indicate that the modification to SMO for regression problem yields dramatic runtime improvements.

We now briefly outline the contents of the paper. In Section 2, we describe previous works for train SVM and SVR. In Section 3, we outline our reduced SVR approach and simplify its mathematical form so that we can express SVM and SVR in a same form. Then we describe a fast training algorithm for simplified SVR, sequential minimal optimization. Section 4 gives computational and graphical results that show the effectiveness and power of Reduced Support Vector Recognition (RSVR). Section 5 concludes the paper.

2. PREVIOUS METHODS FOR TRAINING SVM AND SVR

2.1. *SMO for SVM*

The QP problem to train an SVM is shown below:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j k(\vec{x}_i, \vec{x}_j), \\ \text{subject to} \quad & 0 \le \alpha_i \le C,\ i = 1, \dots, n, \\ & \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned} \tag{1}$$

The QP problem in (1) is solved by the SMO algorithm. A point is an optimal point of (1) if and only if the Karush-Kuhn-Tucker (KKT) conditions are fulfilled and $Q_{ij} = y_i y_j k(\vec{x}_i, \vec{x}_j)$ is positive semidefinite. Such a point may be a nonunique and nonisolated optimum. The KKT conditions are particularly simple; the QP problem is solved when, for all i,

$$\begin{aligned} \alpha_i = 0 &\Longrightarrow y_i f(\vec{x}_i) \ge 1, \\ 0 < \alpha_i < C &\Longrightarrow y_i f(\vec{x}_i) = 1, \\ \alpha_i = C &\Longrightarrow y_i f(\vec{x}_i) \le 1. \end{aligned} \tag{2}$$

Unlike other methods, SMO chooses to solve the smallest possible optimization problem at every step. In each time, SMO chooses two Lagrange multipliers to jointly optimize, finds the optimal values for these multipliers, and updates the SVM to reflect the new optimal values. The advantage of SMO lies in the fact that solving for two Lagrange multipliers can be done analytically. Thus, an entire inner iteration due to numerical QP optimization is avoided.

In addition, SMO does not require extra matrix storage. Thus, very large SVM training problems can fit inside of the memory of an ordinary personal computer or workstation. Because of these advantages, SMO is well suited for training SVM and becomes the most popular training algorithm.

2.2. *Training algorithms for SVR*

Chunking, which was introduced in [10], relies on the observation that only the SVs are relevant for the final form of the hypothesis. Therefore, the large QP problem can be broken down into a series of smaller QP problems, whose ultimate goal is to identify all of the nonzero Lagrange multipliers and discard all of the zero Lagrange multipliers. Chunking seriously reduces the size of the matrix from the number of training examples squared to approximately the number of nonzero Lagrange multipliers squared. However, chunking still may not handle large-scale training problems, since even this reduced matrix may not fit into memory.

Osuna [2, 11] suggested a new strategy for solving the QP problem and showed that the large QP problem can be broken down into a series of smaller QP subproblems. As long as at least one example that violates the KKT conditions is added to the examples for the previous subproblem, each step reduces the overall objective function and maintains a feasible point that obeys all of the constraints. Therefore, a sequence of QP subproblems that always add at least one violator will asymptotically converge.

Based on the SMO, Smola [7] generalized SMO to train SVR. Consider the constrained optimization problem for two indices, say (i, j). Pattern dependent regularization means that C_i may be different for every pattern (possibly even different for α_i, α_i^*). For regression, one has to consider four different cases, (α_i, α_j), (α_i, α_j^*), (α_i^*, α_j), and (α_i^*, α_j^*). Thus, one obtains from the summation constraint $(\alpha_i - \alpha_i^*) + (\alpha_j - \alpha_j^*) = (\alpha_i^{\text{old}} - \alpha_i^{*\,\text{old}}) + (\alpha_j^{\text{old}} - \alpha_j^{*\,\text{old}}) = \gamma$ for regression. Exploiting $\alpha_j^{(*)} \in [0, C_j^{(*)}]$ yields $\alpha_i^{(*)} \in [L, H]$, where L, H are defined as the boundary of feasible regions for regression. SMO has better scaling with training set size than chunking for all data sets and kernels tried. Also, the memory footprint of SMO grows only linearly with the training set size. SMO should thus perform well on the largest problems, because it scales very well.

3. REDUCED SVR AND ITS SMO ALGORITHM

Most of those already existing training methods are originally designed to only be applicable to SVM. Compared with SVM, SVR has more complicated form. For SVR, there are two sets of slack variables, $(\xi_1, \dots, \xi_n)$ and $(\xi_1^*, \dots, \xi_n^*)$, and their corresponding dual variables, $(\alpha_1, \dots, \alpha_n)$ and $(\alpha_1^*, \dots, \alpha_n^*)$. The analytical solution to the size-two QP problems must be generalized in order to work on regression problems. Even though Smola has generalized SMO to handle regression problems, one has to distinguish four different cases, (α_i, α_j), (α_i, α_j^*), (α_i^*, α_j), and $(\alpha_i^* \alpha_j^*)$. This makes the training algorithm more complicated and difficult to implement. In this paper, we propose a new way to make SVR have the similar mathematical form as that of support vector classification, and derive a generalization of SMO to handle regression problems.

3.1. RSVR and its simplified formulation

Recently, the RSVM [12] was proposed as an alternate of the standard SVM. Similar to (1), we now use a different regression objective which not only suppresses the parameter w, but also suppresses b in our nonlinear formulation. Here we first introduce an additional term $b^2/2$ to SVR and outline the key modifications from standard SVR to RSVR. Hence we arrive at the formulation stated as follows,

$$\begin{aligned}\text{minimize}\quad & \frac{1}{2}(w^T w + b^2) + C\sum_{i=1}^{n}(\xi_i + \xi_i^*),\\ \text{subject to}\quad & y_i - w\varphi(\vec{x}_i) - b \le \varepsilon + \xi_i,\\ & w\varphi(\vec{x}_i) + b - y_i \le \varepsilon + \xi_i^*,\\ & \xi_i, \xi_i^* \ge 0 \quad i = 1,\dots,n.\end{aligned} \tag{3}$$

It is interesting to note that very frequently the standard SVR problem and our variant (3) give the same w. In fact, from [12] we can see the result which gives sufficient conditions that ensure that every solution of RSVM is also a solution of standard SVM for a possibly larger C. The same conclusion can be generalized to the RSVR case easily. Later we will show computationally that this reformulation of the conventional SVM formulation yields similar results to SVR.

By introducing two dual sets of variables, we construct a Lagrange function from both the objective function and the corresponding constraints. It can be shown that this function has a saddle point with respect to the primal and dual variables at the optimal solution

$$\begin{aligned}L = {} & \frac{1}{2}w^T w + \frac{1}{2}b^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)\\ & - \sum_{i=1}^{n}\alpha_i(\varepsilon + \xi_i - y_i + w\varphi(\vec{x}_i) + b)\\ & - \sum_{i=1}^{n}\alpha_i^*(\varepsilon + \xi_i^* + y_i - w\varphi(\vec{x}_i) - b)\\ & - \sum_{i=1}^{n}(\eta_i\xi_i + \eta_i^*\xi_i^*).\end{aligned} \tag{4}$$

It is understood that the dual variables in (4) have to satisfy positivity constraints, that is, $\alpha_i, \alpha_i^*, \eta_i, \eta_i^* \ge 0$. It follows from the saddle point condition that the partial derivatives of L with respect to the primal variables (w, b, ξ_i, ξ_i^*) have to vanish for optimality.

$$\begin{aligned}\frac{\partial L}{\partial b} &= b + \sum_{i=1}^{n}(\alpha_i^* - \alpha_i) = 0 \Longrightarrow b = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*),\\ \frac{\partial L}{\partial w} &= w - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\varphi(\vec{x}_i) = 0 \Longrightarrow w = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\varphi(\vec{x}_i),\\ \frac{\partial L}{\partial \xi_i^{(*)}} &= C - \alpha_i^{(*)} - \eta_i^{(*)} = 0.\end{aligned} \tag{5}$$

Substituting (5) into (4) yields the dual optimization problem

$$\begin{aligned}\text{minimize}\quad & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\varphi(\vec{x}_i)\varphi(\vec{x}_j)\\ & + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\\ & + \varepsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)y_i,\\ \text{subject to}\quad & \alpha_i, \alpha_i^* \in [0, C], \quad i = 1,\dots,n.\end{aligned} \tag{6}$$

The main reason for introducing our variant (3) of the RSVR is that its dual (6) does not contain an equality constraint, as does the dual optimization problem of original SVR. This enables us to apply in a straightforward manner the effective matrix splitting methods, such as those of [13], that process one constraint of (3) at a time through its dual variable, without the complication of having to enforce an equality constraint at each step on the dual variable α. This permits us to process massive data without bringing it all into fast memory.

Define

$$\begin{aligned} & H = ZZ^T, \quad Z = \begin{pmatrix} d_1\varphi(\vec{x}_1^{\,0}) \\ \vdots \\ d_n\varphi(\vec{x}_n^{\,0}) \\ d_{n+1}\varphi(\vec{x}_{n+1}^{\,0}) \\ \vdots \\ d_{2l}\varphi(\vec{x}_{2n}^{\,0}) \end{pmatrix}_{2n\times 1} = \begin{pmatrix} d_1\varphi(\vec{x}_1) \\ \vdots \\ d_n\varphi(\vec{x}_n) \\ d_{n+1}\varphi(\vec{x}_n) \\ \vdots \\ d_{2n}\varphi(\vec{x}_{2n}) \end{pmatrix}_{2n\times 1},\\ & \alpha = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \\ \alpha_1^* \\ \vdots \\ \alpha_n^* \end{pmatrix}_{2n\times 1},\\ & E = dd^T, \quad d = \begin{pmatrix} d_1 \\ \vdots \\ d_n \\ d_{n+1} \\ \vdots \\ d_{2n} \end{pmatrix}_{2n\times 1} = \begin{pmatrix} 1 \\ \vdots \\ 1 \\ -1 \\ \vdots \\ -1 \end{pmatrix}_{2n\times 1},\\ & c = \begin{pmatrix} y_1 - \varepsilon \\ \vdots \\ y_l - \varepsilon \\ -y_1 - \varepsilon \\ \vdots \\ -y_n - \varepsilon \end{pmatrix}_{2n\times 1}. \end{aligned} \tag{7}$$

Thus, (6) can be expressed in a simpler way,

$$\begin{aligned} &\text{maximize} \quad c^T\alpha - \frac{1}{2}\alpha^T H\alpha - \frac{1}{2}\alpha^T E\alpha, \\ &\text{subject to} \quad \alpha_i \in [0, C], \quad i = 1, \dots, n. \end{aligned} \tag{8}$$

If we ignore the difference of matrix dimension, (8) and (2) will have the similar mathematical form. So, many training algorithms that were used in SVM can be used in RSVR.

Thus, we obtain an expression which can be evaluated in terms of dot products between the pattern to be regressed and the support vectors

$$f(\vec{x}_*) = \sum_{i=1}^{2n} \alpha_i d_i k(\vec{x}_i^{\,0}, \vec{x}_*) + b. \tag{9}$$

To compute the threshold b, we take into account that due to (5), the threshold can for instance be obtained by

$$b = \sum_{i=1}^{2n} \alpha_i d_i. \tag{10}$$

3.2. Analytic solution for RSVR

Note that there is little difference between generalized RSVR and SVR. The dual (6) does not contain an equality constraint, but we can take advantage of (10) to solve this problem. Here, b is regarded as a constant, while for conventional SVR b equals to zero.

Each step of SMO will optimize two Lagrange multipliers. Without loss of generality, let these two multipliers be α_1 and α_2. The objective function from (8) can thus be written as

$$\begin{aligned} W(\alpha_1, \alpha_2) = {} & c_1\alpha_1 + c_2\alpha_2 - \frac{1}{2}k_{11}\alpha_1^2 - \frac{1}{2}k_{22}\alpha_2^2 - sk_{12}\alpha_1\alpha_2 \\ & - d_1\alpha_1 v_1 - d_2\alpha_2 v_2 - \frac{1}{2}\alpha_1^2 - \frac{1}{2}\alpha_2^2 - s\alpha_1\alpha_2 \\ & - d_1\alpha_1 u_1 - d_2\alpha_2 u_2 + W_{\text{constant}}, \end{aligned} \tag{11}$$

where

$$\begin{aligned} k_{ij} &= k(\vec{x}_i^{\,0}, \vec{x}_j^{\,0}), \qquad s = d_1 d_2, \\ v_i &= \sum_{j=3}^{2l} d_j \alpha_j^{\text{old}} k_{ij} = f^{\text{old}}(\vec{x}_i) + b^{\text{old}} - d_1\alpha_1^{\text{old}} k_{1i} - d_2\alpha_2^{\text{old}} k_{2i}, \\ u_i &= \sum_{j=3}^{2l} d_j \alpha_j^{\text{old}} = b^{\text{old}} - d_1\alpha_1^{\text{old}} - d_2\alpha_2^{\text{old}}, \end{aligned} \tag{12}$$

and the variables with "old" superscripts indicate values at the end of the previous iteration. W_{constant} are terms that do not depend on either α_1 or α_2.

Each step will find the maximum along the line defined by the linear equality in (10). That linear equality constraint can be expressed as

$$\alpha_1^{\text{new}} + s\alpha_2^{\text{new}} = \alpha_1^{\text{old}} + s\alpha_2^{\text{old}} = r. \tag{13}$$

The objective function can be expressed in terms of α_2 alone,

$$\begin{aligned} W = {} & c_1(r - s\alpha_2) + c_2\alpha_2 - \frac{1}{2}k_{11}(r - s\alpha_2)^2 - \frac{1}{2}k_{22}\alpha_2^2 \\ & - sk_{12}(r - s\alpha_2)\alpha_2 - d_1(r - s\alpha_2)v_1 - d_2\alpha_2 v_2 \\ & - \frac{1}{2}(r - s\alpha_2)^2 - \frac{1}{2}\alpha_2^2 - s(r - s\alpha_2)\alpha_2 \\ & - d_1(r - s\alpha_2)u_1 - d_2\alpha_2 u_2 + W_{\text{constant}}. \end{aligned} \tag{14}$$

The stationary point of the objective function is at

$$\begin{aligned} \frac{dW}{d\alpha_2} = {} & -(k_{11} + k_{22} - 2k_{12})\alpha_2 + s(k_{11} - k_{12})r \\ & + d_2(v_1 - v_2) + d_2(u_1 - u_2) - c_1 s + c_2 = 0. \end{aligned} \tag{15}$$

If the second derivate along the linear equality constraint is positive, then the maximum of the objective function can be expressed as

$$\begin{aligned} \alpha_2^{\text{new}}(k_{11} + k_{22} - 2k_{12}) = {} & s(k_{11} - k_{12})r + d_2(v_1 - v_2) \\ & + d_2(u_1 - u_2) - c_1 s + c_2. \end{aligned} \tag{16}$$

Expanding the equations for r, u, and v yields

$$\begin{aligned} & \alpha_2^{\text{new}}(k_{11} + k_{22} - 2k_{12}) \\ & \quad = \alpha_2^{\text{old}}(k_{11} + k_{22} - 2k_{12}) \\ & \qquad + d_2\big((f(\vec{x}_1) - f(\vec{x}_2)) - c_1 d_1 + c_2 d_2\big). \end{aligned} \tag{17}$$

Then

$$\begin{aligned} \alpha_2^{\text{new}} &= \alpha_2^{\text{old}} - \frac{d_2(E_1 - E_2)}{\eta}, \quad E_i = f^{\text{old}}(\vec{x}_i - c_i d_i), \\ \eta &= 2k_{12} - k_{11} - k_{22}. \end{aligned} \tag{18}$$

Then the following bounds apply to α_2:

(i) if $y_1^0 = y_2^0$: $L = \max(0, \alpha_1^{\text{old}} + \alpha_2^{\text{old}} - C)$, $H = \min(C, \alpha_1^{\text{old}} + \alpha_2^{\text{old}})$,

(ii) if $y_1^0 \neq y_2^0$: $L = \max(0, \alpha_2^{\text{old}} - \alpha_1^{\text{old}})$, $H = \min(C, C + \alpha_2^{\text{old}} - \alpha_1^{\text{old}})$.

By solving (8) for Lagrange multipliers α, b can be computed as (10). After each step, b is recomputed, so that the KKT conditions are fulfilled for the optimization problem.

4. EXPERIMENTAL RESULTS

The RSVR algorithm is tested against the standard SVR training with chunking algorithm and against Smola's SMO method on a series of benchmarks. The RSVR, SMO, and chunking are all written in C++, using Microsoft's Visual C++ 6.0 compiler. Joachims' package SVM$^{\text{light}}$[1]

[1] SVM$^{\text{light}}$ is available at http://download.joachims.org/svm_light/v2.01/svm_light.tar.gz.

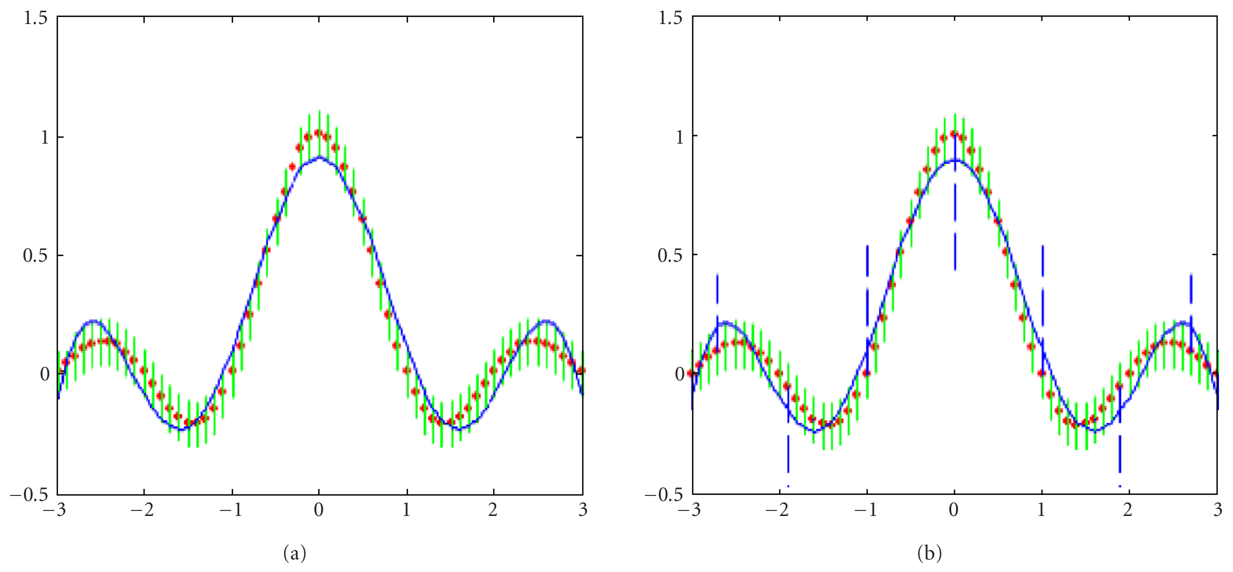

Figure 1: Approximation results of (a) Smola's SMO method and (b) RSVR method.

Table 1: Approximation effect of SVR using various methods.

Experiment	Time (s)	Number of SVs	Expectation of error	Variance of error
RSVR	0.337 ± 0.023	9 ± 1.26	0.045 ± 0.0009	0.0053 ± 0.0012
Smola's SMO	0.467 ± 0.049	9 ± 0.4	0.036 ± 0.0007	0.0048 ± 0.0021
Chunking	0.521 ± 0.031	9 ± 0	0.037 ± 0.0013	0.0052 ± 0.0019
SVM$^{\text{light}}$	0.497 ± 0.056	9 ± 0	0.032 ± 0.0011	0.0049 ± 0.0023

(version 2.01) with a default working set size of 10 is used to test the decomposition method. The CPU time of all algorithms is measured on an unloaded 633 MHz Celeron II processor running Windows 2000 professional.

The chunking algorithm uses the projected conjugate gradient algorithm as its QP solver, as suggested by Burges [1]. All algorithms use sparse dot product code and kernel caching. Both SMO and chunking share folded linear SVM code.

Experiment 1. In the first experiment, we consider the approximation of the sinc function $f(x) = (\sin \pi x)/\pi x$. Here we use the kernel $K(x_1, x_2) = \exp(-\|x_1 - x_2\|^2/\delta)$, $C = 100$ $\delta = 0.1$ and $\varepsilon = 0.1$. Figure 1 shows the approximated results of SMO method and RSVR method, respectively.

In Figure 1b, we can also observe the action of Lagrange multipliers acting as forces (α_i, α_i^*) pulling and pushing the regression inside the ε-tube. These forces, however, can only be applied to the samples where the regression touches or even exceeds the predetermined tube. This directly accords with the illustration of the KKT-conditions, either the regression lies inside the tube (hence the conditions are satisfied with a margin), and Lagrange multipliers are 0, or the condition is exactly met and forces have to be applied to $\alpha_i \neq 0$ or $\alpha_i^* \neq 0$ to keep the constraints satisfied. This observation proves that the RSVR method can handle regression problems successfully.

In Table 1, we can see that the SVM trained with other various methods have nearly the same approximation accuracy. However, in this experiment, we can see that the testing accuracy of RSVR is little lower than traditional SVR.

Moreover, as the training efficiency is the main motivation of RSVR, we would like to discuss its different implementations and compare their training time with regular SVR.

Experiment 2. In order to compare the time consume of different training methods on massive data sets, we test these algorithms on three real-world data sets.

In this experiment, we adopt the same data sets used in [14]. In this experiment, we use the same kernel with $C = 3000$ and kernel parameters are shown in Table 2. Here we compare the programs on three different tasks that are stated as follows.

Kin

This data set represents a realistic simulation of the forward dynamics of an 8 link all-revolute robot arm. The task is to predict the distance of the end-effecter from a target, given

TABLE 2: Comparison on various data sets.

Data set	Training algorithms	Time (s)	Training set size	Number of SVs	Objective value of training error	Kernel parameter δ	ε
Kin	RSVR	3.15 ± 0.57	650	62 ± 10	0.65	100	0.5
	SMO	4.23 ± 0.31		62 ± 12			
	Chunking	4.74 ± 1.26		64 ± 8			
	SVM$^{\text{light}}$	5.42 ± 0.08		60 ± 7			
Sunspots	RSVR	23.36 ± 8.31	4000	388 ± 14	5.0	500	10.0
	SMO	76.18 ± 013.98		386 ± 7			
	Chunking	181.54 ± 16.75		387 ± 13			
	SVM$^{\text{light}}$	357.37 ± 15.44		387 ± 11			
Forest	RSVR	166.41 ± 29.37	20000	2534 ± 6	0.5	800	1.0
	SMO	582.3 ± 16.85		2532 ± 8			
	Chunking	1563.1 ± 54.6		2533 ± 5			
	SVM$^{\text{light}}$	1866.5 ± 46.7		2534 ± 5			

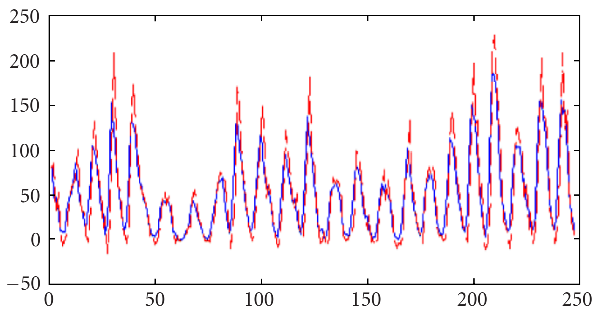

FIGURE 2: Comparison between real sunspot data (solid line) and predicted sunspot data (dashed line).

features like joint positions, twist angles, etc. The first data is of size 650.

Sunspots

Using a series representing the number of sunspots per day, we created one input/output pair for each day, the yearly average of the year starting the next day had to be predicted using the 12 previous yearly averages. This data set is of size 4000.

Forest

This data set is a classification task with 7 classes [14], where the first 20000 examples are used here. We transformed it into a regression task where the goal is to predict +1 for examples of class 2 and −1 for the other examples.

Table 2 illustrates the time consume, the training set size, and the number of support vectors for different training algorithms. In each data set, the objective values of training error are the same. Here we can see that with data set increase, the difference of training time among these training algorithms also increases greatly. When the size of data set reaches 20000, the training time needed by Chunking and SVM$^{\text{light}}$ is more than 11 times than that of RSVR. Here we define the training error to be the MSE over the training data set.

Experiment 3. In this experiment, we will use the RSVR trained by SMO to predict time series data set. Here we adopt Greenwich's sunspot data. The kernel parameters are $C = 3000$, $\delta = 500$, and $\varepsilon = 10$. We can also gain these data from Greenwich's homepage (http://science.msfc.nasa.gov/ssl/pad/solar/greenwch.htm). We use historic sunspot data to predict future sunspot data. Figure 2 shows the comparison between real sunspot data and predicted sunspot data. This illustrates that the SVM give good prediction to sunspot. This experiment proves that the RSVR trained by SMO algorithm can be used in practical problems successfully.

5. CONCLUSION

We have discussed the implementations of RSVR and its SMO fast training algorithm. Compared with Smola's SMO algorithm, we successfully reduce the variables from four to two. This reduces the complexity of training algorithm greatly and makes it easy to implement. Also we compare it with conventional SVR. Experiments indicate that in general the test accuracy of RSVR is little worse than that of the standard SVR. For the training time which is the main motivation of RSVR, we show that, based on the current implementation techniques, RSVR will be faster than regular SVR on large data set problems or some difficult cases with many support vectors. Therefore, for medium-size problems, standard SVR should be used, but for large problems, RSVR can effectively restrict the number of support vectors and can be an appealing alternate. Thus, for very large problems it is appropriate to try the RSVR first.

ACKNOWLEDGMENT

This work was supported by Chinese National Natural Science Foundation and Shanghai Bao Steel Co. (50174038, 30170274).

REFERENCES

[1] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[2] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Proc. IEEE Workshop on Neural Networks for Signal Processing VII*, J. Principe, L. Giles, N. Morgan, and E. Wilson, Eds., pp. 276–285, IEEE, Amelia Island, Fla, USA, September 1997.

[3] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods—Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds., pp. 185–208, MIT Press, Cambridge, Mass, USA, 1998.

[4] J. Platt, "Using sparseness and analytic QP to speed training of support vector machines," in *Advances in Neural Information Processing System*, M. S. Kearns, S. A. Solla, and D. A. Cohn, Eds., vol. 11, pp. 557–563, MIT Press, Cambridge, Mass, USA, 1999.

[5] S. Mukherjee, E. Osuna, and F. Girosi, "Nonlinear prediction of chaotic time series using support vector machines," in *Proc. IEEE Workshop on Neural Networks for Signal Processing VII*, pp. 511–520, Amelia Island, Fla, USA, September 1997.

[6] T. Friess, N. Cristianini, and C. Campbell, "The kernel-adatron: a fast and simple learning procedure for support vector machines," in *Proc. 15th International Conference in Machine Learning*, J. Shavlik, Ed., pp. 188–196, Morgan Kaufmann, San Francisco, Calif, USA, 1998.

[7] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," NeuroCOLT Tech. Rep. NC-TR-98-030, Royal Holloway College, University of London, London, UK, 1998.

[8] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design," *Neural Computation*, vol. 13, no. 3, pp. 637–649, 2001.

[9] G. W. Flake and S. Lawrence, "Efficient SVM regression training with SMO," *Machine Learning*, vol. 46, no. 1-3, pp. 271–290, 2002.

[10] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, NY, USA, 1995.

[11] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: an application to face detection," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 130–136, San Juan, Puerto Rico, June 1997.

[12] Y.-J. Lee and O. L. Mangasarian, "RSVM: reduced support vector machines," in *First SIAM International Conference on Data Mining*, pp. 350–366, Chicago, Ill, USA, April 2001.

[13] O. L. Mangasarian and D. R. Musicant, "Successive overrelaxation for support vector machines," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1032–1037, 1999.

[14] R. Collobert and S. Bengio, "SVMTorch: support vector machines for large-scale regression problems," *Journal of Machine Learning Research*, vol. 1, no. 2, pp. 143–160, 2001.

Quan Yong was born in 1976. He is a Ph.D. candidate at the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai. His current research interests include machine learning, and data mining.

Yang Jie was born in 1964. He is a professor and doctoral supervisor at the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai. His research interest areas are image processing, pattern recognition, and data mining and application. He is now supported by the National Natural Science Foundation of China.

Yao Lixiu was born in 1973. She is an Associate professor at the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai. Her current research interests include data mining techniques and their applications. She is now supported by the National Natural Science Foundation of China and BaoSteel Co.

Ye Chenzhou was born in 1974. He is a Ph.D. candidate at the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai. His current research interests include artificial intelligence and data mining.