Knowl Inf Syst (2008) 14:233–247
DOI 10.1007/s10115-007-0082-6

REGULAR PAPER

Parallel randomized sampling for support vector machine (SVM) and support vector regression (SVR)

Yumao Lu · Vwani Roychowdhury

Received: 12 April 2006 / Revised: 14 November 2006 / Accepted: 26 January 2007 /
Published online: 30 June 2007

Abstract A parallel randomized support vector machine (PRSVM) and a parallel randomized support vector regression (PRSVR) algorithm based on a randomized sampling technique are proposed in this paper. The proposed PRSVM and PRSVR have four major advantages over previous methods. (1) We prove that the proposed algorithms achieve an average convergence rate that is so far the fastest bounded convergence rate, among all SVM decomposition training algorithms to the best of our knowledge. The fast average convergence bound is achieved by a unique priority based sampling mechanism. (2) Unlike previous work (Provably fast training algorithm for support vector machines, 2001) the proposed algorithms work for general linear-nonseparable SVM and general non-linear SVR problems. This improvement is achieved by modeling new LP-type problems based on Karush–Kuhn–Tucker optimality conditions. (3) The proposed algorithms are the first parallel version of randomized sampling algorithms for SVM and SVR. Both the analytical convergence bound and the numerical results in a real application show that the proposed algorithm has good scalability. (4) We present demonstrations of the algorithms based on both synthetic data and data obtained from a real word application. Performance comparisons with $\text{SVM}^{\text{light}}$ show that the proposed algorithms may be efficiently implemented.

Keywords Randomized sampling · Support vector machine · Support vector regression · Parallel algorithm

Y. Lu (✉)
Yahoo! Inc., Applied Research, Sunnyvale, CA 94089, USA
e-mail: yumaol@yahoo-inc.com

V. Roychowdhury
Department of Electrical Engineering, University of California,
Los Angeles, CA, USA

1 Introduction

The support vector machine (SVM) has recently become one of the most popular methods for classification and regression in machine learning. The underlying training problem can be formulated as a quadratic programming (QP) problem that can be solved by standard optimization algorithms [12,27,28]. These algorithms, however, are only able to solve some small and medium size problems since they require the loading of the whole Gram matrix.

The first decomposition algorithm, that works on a small subset of training vectors in each iteration was first proposed by Osuna [24]. Several improved versions of decomposition algorithms, including the Chunking algorithm [23], sequential minimization optimization (SMO) algorithm [25], successive over-relaxation (SOR) algorithm [22] and their variations [11,14,20], have made it feasible to apply SVM for large scale training problems. The lack of a theoretical convergence bound of the decomposition algorithms for general SVM training problems, however, compromises the potential applicability of such a powerful tool set, especially for many real-time and on-line applications, where a convergence bound becomes critical [7]. This paper proposes a randomized sampling algorithm that has a provably fast convergence bound for general SVM and SVR training problems.

Sampling theory has a long successful history in optimization [1,9]. The application to the SVM training problem was first proposed by Balcazar et al. [2]. However, Balcazar assumed that the SVM training problem is a separable problem or a problem that can be transformed to an equivalent separable problem by assuming an arbitrarily small regularization factor γ (D and $1/k$ in Balcazar et al. [2] and Balcazar et al. [3]). They also stated that there were number of implementation difficulties so that no relevant experimental results could be provided [3].

We model new LP-type problems for SVM and SVR, respectively, such that the general linear nonseparable problem can be covered by our randomized support vector machine (RSVM) algorithm and the general nonlinear regression problem can be covered by our randomized support vector regression (RSVR) algorithm. In the LP-type problems, Karush–Kuhn–Tucker (KKT) optimality conditions are used as criteria to identify violators and extremes. The advantage of using KKT conditions over using classification errors proposed by Balcazar et al. [2] lies in the fact that the KKT conditions will be always satisfied when a global optimum is achieved, while classification errors may not vanish in linear nonseparable problems, even when a global optimum has been achieved.

Parallel learning is necessary if a centralized system is infeasible because of geographical, physical and computational reasons. It has been noted that scalable parallel algorithms are important for a machine learning algorithm to be successfully applied in large scale industrial applications [4,8,19,21,26]. In order to take advantage of distributed computing facilities, we proposed a novel parallel randomized SVM/SVR, in which multiple working sets can be worked on simultaneously. To the best of our knowledge, it is the very first attempt in extending randomized sampling technique to a parallel algorithm for SVM/SVR training problems.

The basic idea of the algorithm is to randomly shuffle the training vectors among a network based on a carefully designed priority and weighting mechanism, and to solve the multiple local problems simultaneously. Unlike the previous works on parallel SVM [11,18] that lacks a convergence bound, our algorithms on average, converges to the global optimum classifier/regressor in less than $(6\delta \ln(N+6r(C-1)\delta)/C$ iterations,

where δ denotes the underlying combinatorial dimension, N denotes the total number of training vector, C denotes the number of working sites, and r denotes the size of each working set. Since the RSVM/RSVR is a special case of PRSVM/PRSVR, our proof naturally works for the RSVM/RSVR. Note that, when $C = 1$, our result reduces to Balcazar's bound [3].

This paper is organized as follows. The support vector machine and support vector regression problems are introduced and formulated in Sect. 2, followed by our LP-type modeling in Sect. 3. We present the parallel randomized support vector machine and support vector regression (PRSVM/PRSVR) algorithms in Sect. 4. The theoretical global convergence bound is given in Sect. 5, followed by a presentation of demonstrations on synthetic data and data from a geographic information system application in Sect. 6. We discuss some interesting issues in Sect. 7 and conclude this paper in Sect. 8.

2 Support vector machine and support vector regression

We introduce fundamentals and basic notations on SVM and SVR in this section.

2.1 Support vector machine

Let us first consider a simple linear separation problem. We are seeking a hyperplane to separate a set of positively and negatively labeled training data. The hyperplane is defined by $w^T x_i - b = 0$ with parameter $w \in \mathbf{R}^m$ and $b \in \mathbf{R}$ such that $y_i(w^T x_i - b) > 1$ for $i = 1, \ldots, N$ where $x_i \in \mathbf{R}^m$ is a training data point and $y_i \in \{+1, -1\}$ denotes the class of the vector x_i. The margin is defined by the distance of the two parallel hyperplanes $w^T x - b = 1$ and $w^T x - b = -1$, i.e., $2/||w||_2$. The margin is related to the generalization of the classifier [28]. The support vector machine (SVM) training problem is in fact a quadratic programming problem, which maximizes the margin over the parameters of the linear classifier. For general nonseparable problems, a set of slack variables $\mu_i, i = 1, \ldots, N$ are introduced. The SVM training problem problem is defined as follows:

$$\begin{array}{ll} \text{minimize} & (1/2)w^T w + \gamma \mathbf{1}^T \mu \\ \text{subject to} & y_i(w^T x_i - b) \geq 1 - \mu_i, \quad i = 1, \ldots, N \\ & \mu \geq 0 \end{array} \tag{1}$$

where the scalar γ is usually empirically selected to reduce the testing error rate. To simplify notations, we define $v_i = (x_i, -1)$, $\theta = (w, b)$, and a matrix Z as

$$Z = [(y_1 v_1)\ (y_2 v_2)\ \ldots\ (y_N v_N)]^T.$$

The dual of problem (1) is shown as follows:

$$\begin{array}{ll} \text{maximize} & -(1/2)\alpha^T Z Z^T \alpha + \mathbf{1}^T \alpha \\ \text{subject to} & 0 \leq \alpha \leq \gamma \mathbf{1}. \end{array} \tag{2}$$

A nonlinear kernel function can be used for nonlinear separation of the training data. In that case, the gram matrix ZZ^T is replaced by a kernel matrix $K \in \mathbf{R}^{N \times N}$. Our PRSVM that is described in the Sect. 4 can be kernelized and therefore is able to keep the full advantages of the SVM.

2.2 Support vector regression

Support vector regression is a robust function estimation method. The basic idea is to minimize a pre-defined risk function over the parameters of the regressor.

We, again, start from the estimation of a linear function

$$f(x) = w^T x + b, \tag{3}$$

based on independent and identically distributed (IID) training samples

$$(x_1, y_1), \ldots, (x_N, y_N)$$

where $x_i \in \mathbf{R}^m$, $y_i \in \mathbf{R}$, $w \in \mathbf{R}^m$, $b \in \mathbf{R}$ and $f : \mathbf{R}^m \to \mathbf{R}$.

To preserve the sparse property of the solution, Vapnik used the ϵ-insensitive loss function

$$c(x, y, f(x)) = |y_i - f(x_i)|_\epsilon \equiv \max\{0, |y - f(x)| - \epsilon\},$$

which does not penalize errors below the tolerance ϵ [28].

To minimize a regularized ϵ-insensitive loss function, we have

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} w^T w + \gamma \mathbf{1}^T (\xi + \tilde{\xi}) \\ \text{subject to} & w^T x + b - y \le \epsilon + \xi \\ & y - w^T x - b \le \epsilon + \tilde{\xi} \\ & \xi, \tilde{\xi} \ge 0, \end{array} \tag{4}$$

where $\xi \in \mathbf{R}^N$ and $\tilde{\xi} \in \mathbf{R}^N$ are the allowed error "above" and "below" the margin boundary, $||w||^2$ is the regularization term and γ is an empirically selected scalar used to balance these two terms.

Recall that $v_i = (x_i, -1)$, $\theta = (w, b)$, and a matrix Z as

$$Z = [(y_1 v_1)\ (y_2 v_2)\ \ldots\ (y_N v_N)]^T.$$

The corresponding dual formulation has the form

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} (\tilde{\alpha} - \alpha)^T Z Z^T (\tilde{\alpha} - \alpha) - y^T (\tilde{\alpha} - \alpha) + \epsilon \mathbf{1}^T (\tilde{\alpha} + \alpha) \\ \text{subject to} & \mathbf{1}^T (\tilde{\alpha} - \alpha) = 0 \\ & 0 \le \tilde{\alpha}, \alpha \le \gamma. \end{array} \tag{5}$$

where the dual variable $\tilde{\alpha}, \alpha \in \mathbf{R}^N$. A *support vector* for SVR is defined as the training vectors x_i such that the corresponding optimal dual variable

$$\tilde{\alpha}_i - \alpha_i \neq 0.$$

For convenience, we define vector $\upsilon \in \mathbf{R}^N$ such that

$$\upsilon = \tilde{\alpha} - \alpha.$$

Similarly, a nonlinear kernel function can also be used for general nonlinear regression. In that case, the gram matrix ZZ^T is replaced by a kernel matrix $K \in \mathbf{R}^{N \times N}$. Our PRSVR (a unified algorithm for regression problems with PRSVM) that is described in the Sect. 4 works for general kernelized nonlinear support vector regression problems.

3 Randomized sampling

We model two LP-type problems for support vector machines and support regression respectively in this section. The modeling procedure helps in understanding the main algorithm proposed in the following section.

3.1 The sampling lemma and LP-type problem

Before presenting our LP-type models, we introduce the fundamentals and notations in randomized sampling theory.

Let $\mathcal{X}$ be the set of training vectors. That is, each element of $\mathcal{X}$ is a row vector of the matrix X. Throughout this paper, we use $\mathcal{CALLIGRAPHIC}$ style letters to denote sets of the row vectors of a matrix, which itself is denoted by the same letter in *italics*. An abstract problem is denoted by $(\mathcal{X}, \phi)$. Here, ϕ is a mapping from a given subset $\mathcal{X}_\mathcal{R}$ of $\mathcal{X}$ to the solution of problem (1) with constraints corresponding to X_R and $\mathcal{X}$ is of size N. Such a problem, where the constraints correspond to only a subset of all the training vectors will be referred to as a *local problem* from now on. In our approach, we will divide the training vectors into multiple subsets (with repetitions, if needed) or working sets, and SVM/SVR solutions will be found for each local problem. Define

$$\begin{aligned}\mathcal{V}(\mathcal{X}) &:= \{x \in \mathcal{X}\backslash\mathcal{X}_\mathcal{R} | \phi(\mathcal{X}_\mathcal{R} \cup \{x\}) \neq \phi(\mathcal{X}_\mathcal{R})\},\\ \mathcal{E}(\mathcal{X}) &:= \{x \in \mathcal{X}_\mathcal{R} | \phi(\mathcal{X}_\mathcal{R}\backslash\{x\}) \neq \phi(\mathcal{X}_\mathcal{R})\}.\end{aligned}$$

The elements of $\mathcal{V}(\mathcal{X}_\mathcal{R})$ are called violators of $\mathcal{X}_\mathcal{R}$ and the elements of $\mathcal{E}(\mathcal{X}_\mathcal{R})$ are called extremes in $\mathcal{X}_\mathcal{R}$. By definition, we have

$$x \text{ violates } \mathcal{X}_\mathcal{R} \;\Leftrightarrow\; \text{x is extreme in } \mathcal{X}_\mathcal{R} \cup \{x\}.$$

For a random sample $\mathcal{X}_\mathcal{R}$ of size r, we consider the expected values

$$\begin{aligned}v_r &:= E_{|\mathcal{X}_\mathcal{R}|=r}(|\mathcal{V}_\mathcal{R}|)\\ e_r &:= E_{|\mathcal{X}_\mathcal{R}|=r}(|\mathcal{E}_\mathcal{R}|)\end{aligned}$$

Gartner proved the following sampling lemma [17]:

Lemma 3.1 *(Sampling Lemma). For* $0 \leq r < N$,

$$\frac{v_r}{N-r} = \frac{e_{r+1}}{r+1}.$$

Proof By definitions, we have

$$\begin{aligned}\binom{N}{r} v_r &= \sum_{\mathcal{X}_\mathcal{R}} \sum_{x \in \mathcal{X}\backslash\mathcal{X}} [x \text{ violates } \mathcal{X}_\mathcal{R}]\\ &= \sum_{\mathcal{X}_\mathcal{R}} \sum_{x \in \mathcal{X}\backslash\mathcal{X}_\mathcal{R}} [x \text{ is extreme in } \mathcal{X}_\mathcal{R} \cup \{x\}]\\ &= \sum_{\mathcal{Q}} \sum_{x \in \mathcal{Q}} [x \text{ is extreme in } \mathcal{Q}]\\ &= \binom{N}{r+1} e_{r+1},\end{aligned}$$

where [.] is the indicator variable for the event in brackets and the last row follows the fact that the set $\mathcal{Q}$ has $r+1$ elements. The Lemma immediately follows.

The problem $(\mathcal{X}, \phi)$ is said to be a LP-type problem if ϕ is monotone and local (see Definition 3.1 in [17]). Balcazar et al. [2] proved that the problem (1) is a LP-type

problem . So is the problem (2). Similarly, problems (4) and (5) are also LP-type problems. We use the same definitions given by [17] to define the *basis* and *combinatorial dimension* as follows. For any $\mathcal{X}_\mathcal{R} \subseteq \mathcal{X}$, a *basis* of $\mathcal{X}_\mathcal{R}$ is an inclusion-minimal subset $\mathcal{B} \subseteq \mathcal{X}_\mathcal{R}$ with $\phi(\mathcal{B}) = \phi(\mathcal{X}_\mathcal{R})$. The *combinatorial dimension* of ($\mathcal{X}$,ϕ), denoted by δ, is the size of a largest basis of $\mathcal{X}$. For a LP-type problem ($\mathcal{X}$,ϕ) with combinatorial dimension δ, the sampling lemma yields

$$v_r \leq \delta \frac{N-r}{r+1}. \tag{6}$$

This follows from the fact that $|\mathcal{E}(\mathcal{X}_\mathcal{R})| \leq \delta$.

3.2 LP-type problem modeling for SVM

Now, we are ready to relate the definitions of the extremes, violators and the basis to our general SVM training problem (1) or (2). For any local solution θ^p or α^p of problem ($\mathcal{X}_p$, ϕ), the basis is the support vector set, $\mathcal{SV}_p$. The violators of the local solutions will be the vectors that violate the Karush–Kuhn–Tucker (KKT) necessary and sufficient optimality conditions. The KKT conditions for the problem (1) and (2) are listed as follows:

$$\begin{aligned} &Z\theta^\star \geq \mathbf{1} - \mu^\star,\ \mu^\star \geq 0,\ 0 \leq \alpha^\star \leq \gamma\mathbf{1}, \\ &\theta^\star = Z^T\alpha^\star,\ (\gamma - \alpha_i^\star)\mu_i^\star = 0, \quad i = 1, \ldots, N. \end{aligned}$$

Since the μ_i and α_i for the training vector x_i is always 0 for $x_i \in \mathcal{X}\backslash\mathcal{X}_p$, the only condition needed to be tested is

$${\theta^p}^T z_i \geq 1$$

or

$${\alpha^p}^T Z_p z_i \geq 1.$$

Any training vector that violates the above condition is called a violator to ($\mathcal{X}_p$, ϕ). The size of the largest basis, δ is naturally the largest number of support vectors for all subproblems ($\mathcal{X}_p$, ϕ), $\mathcal{X}_p \subseteq \mathcal{X}$. For separable problems, δ is bounded by one plus the lifted dimension, *i.e.*, $\delta \leq n + 1$. For general nonseparable problems, we do not know the bound for δ before we actually solve the problem. What we can do is to set a sufficiently large number to bound δ from above.

3.3 LP-type problem modeling for SVR

The similar technique can be applied to SVR training problems. For any local solution θ^p or υ^p of problem ($\mathcal{X}_p$, ϕ), the basis is the support vector set, $\mathcal{SV}_p$. The violators of the local solutions will be the vectors that violate the KKT necessary and sufficient optimality conditions. The KKT conditions for the problem (4) and (5) are listed as follows:

$$\begin{aligned} &|Z\theta^\star - y| \leq \epsilon^\star + \max\{\tilde{\mu}^\star, \mu^\star\},\ \tilde{\mu}^\star, \mu^\star \geq 0,\ \mathbf{1}\upsilon^\star = 0,\ \theta^\star = Z^T\upsilon^\star, \\ &\max\{0, |(ZZ^T\upsilon^\star)_i - y_i| - \epsilon\}\upsilon_i^\star = 0, \quad i = 1, \ldots, N, \end{aligned}$$

where $(.)_i$ denotes the i-th component of the vector in bracket.

Since the μ_i, $\tilde{\mu}_i$ and υ_i for the training vector x_i is always 0 for $x_i \in \mathcal{X}\backslash\mathcal{X}_p$, the only condition needed to be tested is

$$|(Z\theta^\star)_i - y_i| \leq \epsilon$$

or

$$|(ZZ^T\upsilon^\star)_i - y_i| \leq \epsilon.$$

Any training vector that violates the above condition is called a violator to $(\mathcal{X}_p, \phi)$. The size of the largest basis, δ is naturally the largest number of support vectors for all subproblems $(\mathcal{X}_p, \phi)$, $\mathcal{X}_p \subseteq \mathcal{X}$.

4 Algorithm

We consider the following problem: the training data are distributed in $C + 1$ sites, where there are C working sets and 1 nonworking set. Each working site is assigned a priority number $p = 1, 2, \ldots, C$. We also assume that each working site contains r training vectors, where $r \geq 6\delta^2$ and δ denotes the combinatorial dimension of the underlying SVM/SVR problem.

Define a function $u(.)$ to record the number of copies of elements of a training set. For training set $\mathcal{X}$, we define a set $\mathcal{W}$ such that $\mathcal{W}$ contains the virtually duplicated copies of the training vectors. We have $|W| = u(\mathcal{X})$. We also define the virtual set $\mathcal{W}_p$ corresponding to training set $\mathcal{X}_p$ at site p.

Our parallel randomized support vector machine/regression (PRSVM/PRSVR) algorithm works as follows.

Initialization

Training vectors $\mathcal{X}$ are randomly distributed to $C + 1$ sites. Assign priorities to all sites such that each site gets a unique priority number. Set $u(\{x_i\}) = 1, \ \forall i$. Hence, $u(\mathcal{X}) = N$. We have $|\mathcal{X}_p| = |\mathcal{W}_p|$ for all p. Set $t = 0$.

Iteration

Each iteration consists of the following steps.

Repeat for $t = 1, 2, \ldots$

1. Randomly distribute the training vectors over the working sites according to $u(\mathcal{X})$ as follows. Let $\mathcal{S}^1 = \mathcal{W}$.
   For $p = 1 : C$
   Choose r training vectors, $\mathcal{W}_p$ from $\mathcal{S}^p$ uniformly (and make sure $r \geq 6\delta^2$);
   $\mathcal{S}^{p+1} := \mathcal{S}^p\backslash\mathcal{W}_p$;
   End For
2. Each site with priority p, $p \leq C$ solves the local partial problem and record the solution θ^p. Send this solution to all other sites q, $q \neq p$.
3. Each site with priority q, $q = 1, \ldots, C + 1$, checks the solution θ^p from site with higher priority $p, p < q$. Define $\mathcal{V}_{q,p}$ to be the training vectors in the site with priority q that violate the KKT condition corresponding to solution (w^p, b^p), $q \neq p$.

That is,

$$\mathcal{V}_{q,p} := \{x_i | \theta^{p^T}([x_i; 1]) y_i < 1, x_i \in \mathcal{X}_q, x_i \notin \mathcal{X}_p\}$$

for classification problems or

$$\mathcal{V}_{q,p} := \{x_i | |\theta^{p^T}([x_i; 1] - y_i| > \epsilon, x_i \in \mathcal{X}_q, x_i \notin \mathcal{X}_p\}$$

for regression problems.

4. **If** $\sum_{q=p+1}^{C+1} u(\mathcal{V}_{q,p}) \leq |\mathcal{S}^p|/(3\delta)$ **then** $u(\{x_i\}) = 2u(\{x_i\})$, for all $x_i \in \mathcal{V}_{q,p}$, $\forall q \neq p$, $\forall p$;

until $\cup_{q \neq p} \mathcal{V}_{q,p} = \emptyset$ for some p.

Return the solution θ^p.

The priority setting of working sets actually defines the order of sampling. The highest priority server gets the first batch of sampled data, lower one gets the second batch and so on. This kind of sequential behavior is designed to help defining violators and extremes clearly under a multiple working site configuration.

Step 2 involves a merging procedure. If $u(\{x_i\})$ copies of vector x_i are sampled to a working set $\mathcal{W}_p$, only one copy of x_i is included in the optimization problem $(\mathcal{X}_p, \phi)$ that we are solving, while we record this number of copies as a weight of this training vector.

The merging procedure has two properties:

Property 4.1 *A training vector that is not in working set $\mathcal{X}_p$ must not be a violator of the problem $(\mathcal{X}_p, \phi)$ if one or more copies of this vector are included in the working set $\mathcal{X}_p$. That is, $x_i \notin \mathcal{V}(\mathcal{X}_p)$, if $x_i \in \mathcal{X}_p$.*

Property 4.2 *If multiple copies of a vector x_i are sampled to a working set $\mathcal{X}_p$, none of those of vectors can be the extreme of the problem $(\mathcal{X}_p, \phi)$. That is, $x_i \notin \mathcal{E}(\mathcal{X}_p)$ if $u(\{x_i\}) > 1$ at site p.*

The above two properties follow immediately by definitions of violators and extremes.

One may note that the merging procedure actually constructs an abstract problem $(\mathcal{W}_p, \phi')$ such that $\phi'(\mathcal{W}_p) = \phi(\mathcal{X}_p)$. By definition, $(\mathcal{W}_p, \phi')$ is a LP-type problem and has the same combinatorial dimension, δ, as the problem $(\mathcal{X}_p, \phi)$. If the set of violators of $(\mathcal{X}_p, \phi)$ is $\mathcal{V}_p$, the number of violators of $(\mathcal{W}_p, \phi')$ is $u(\mathcal{V}_p)$.

Step 4 plays the key role in this algorithm. It says that if the number of violators of the LP-type problem $(\mathcal{W}_p, \phi')$ is not too large, we double the weights of the violators of $(\mathcal{W}_p, \phi')$ in all sites. Otherwise, we keep the weights untouched since the violators already have enough weights to be sampled to a working site.

One may note when $C = 1$, the PRSVM/PRSVR is reduced to the RSVM. However, our RSVM is different from the randomized support vector machine training algorithm in [2] in several ways. First, our RSVM is capable of solving general nonseparable problems, while Balcazar's method has to transfer a nonseparable problem to an equivalent separable problem by assuming an arbitrarily small γ. Second, our RSVM merges examples after sampling them. Duplicated examples are not allowed in the optimization steps. Third, we test the KKT conditions to identify a violator instead of identifying a misclassified point. In our RSVM, a correctly classified example may also be a violator if this example violates the KKT condition.

5 Proof of the average convergence rate

We prove the average number of iterations executed in our algorithm, PRSVM/PRSVR, is bounded by $(6\delta/C)\ln(N+6r(C-1)\delta)$ in this section. This proof is a generalization of the one given in [2]. The result of the traditional RSVM becomes a special case of our PRSVM.

Theorem 5.1 *For general SVM training problem the average number of iterations executed in the PRSVM/PRSVR algorithm is bounded by* $(6\delta/C)\ln(N+6r(C-1)\delta)$.

Proof We consider an update to be successful if the if-condition in the step 4 holds in an iteration. One iteration has C updates, successful or not.

We first show a bound on the number of successful updates. Let $\mathcal{V}_p$ denote the set of violators from site with priority $q \geq p$ for the solution θ^p. By this definition, we have

$$u(\mathcal{V}_p) = \sum_{q=p+1}^{C+1} u(\mathcal{V}_{q,p})$$

Since the if-condition holds, we have

$$\sum_{q=p+1}^{C+1} u(\mathcal{V}_{q,p}) \leq u(S^p)/(3\delta) \leq u(\mathcal{X})/(3\delta).$$

By noting that the total number of training vectors including duplicated ones in each working sites is always r for any iterations, we have

$$\sum_{q=1}^{p-1} u(\mathcal{V}_{q,p}) \leq r(p-1) \leq r(C-1)$$

and

$$\begin{aligned}\sum_{q\neq p} u(V_{q,p}) &= \sum_{q=p+1}^{C+1} u(\mathcal{V}_{q,p}) + \sum_{q=1}^{p-1} u(\mathcal{V}_{q,p}) \\ &= u(\mathcal{V}_p) + \sum_{q=1}^{p-1} u(\mathcal{V}_{q,p})\end{aligned}$$

Therefore, at each successful update, we have

$$u_k(\mathcal{X}) \leq u_{k-1}(\mathcal{X})(1+\frac{1}{3\delta}) + 2r(C-1).$$

where k denotes the number of successful updates. Since $u_0(\mathcal{X}) = N$, after k successful updates, we have

$$\begin{aligned}u_k(\mathcal{X}) &\leq N(1+\tfrac{1}{3\delta})^k + 2r(C-1)3\delta[(1+\tfrac{1}{3\delta})^k - 1] \\ &< (N+6r(C-1)\delta)(1+\tfrac{1}{3\delta})^k\end{aligned}$$

Let $\mathcal{X}_0$ be the set of support vectors of the original problem (1) or (2). At each successful iterations, some x_i of $\mathcal{X}_0$ must not be in $\mathcal{X}_p$. Hence, $u(\{x_i\})$ gets doubled.

Since, $|\mathcal{X}_0| \leq \delta$, there is some x_i in $\mathcal{X}_0$ that gets doubled at least once every δ successful updates. That is, after k successful updates, $u(\{x_i\}) \geq 2^{k/\delta}$.

Therefore, we have

$$2^{\frac{k}{\delta}} \leq u(\mathcal{X}) \leq (N + 6r(C-1)\delta)\left(1 + \frac{1}{3\delta}\right)^k.$$

By simple algebra, we have

$$k \leq 3\delta \ln(N + 6r(C-1)\delta).$$

That is, the algorithm terminates within less than $3\delta \ln(N + 6r(C-1)\delta)$ successful updates.

The rest is to prove that the probability of a successful update is higher than one half. By sampling lemma, the bound (6), we have

$$\begin{aligned} \text{Exp}(u(\mathcal{V}_p)) &\leq \frac{(u(\mathcal{S}^p) - r)\delta}{r+1} \\ &< \frac{u(\mathcal{S}^p)}{6\delta} \end{aligned}$$

By Markov equality, we have

$$\begin{aligned} &\text{Pro}\{u(\mathcal{V}_p) \leq \tfrac{u(\mathcal{S}^p)}{3\delta}\} \\ \geq\ &\text{Pro}\{u(\mathcal{V}_p) \leq 2\text{Exp}(u(\mathcal{V}_p))\} \\ \geq\ &\tfrac{1}{2}. \end{aligned}$$

This implies that the expected number of updates is at most twice as large as the number of successful updates, *i.e.*, $K \leq 6\delta \ln(N + 6r(C-1)\delta)$, where K denotes the total number of updates. Note that, at the end of each iteration, we have

$$K = Ct.$$

Therefore, the PRSVM/PRSVR algorithm guarantees, on average, within $(6\delta/C)\ln(N + 6r(C-1)\delta)$ steps, that all the support vectors are contained by one of the C working sites. For separable problems, we have $\delta \leq n + 1$. For general nonseparable problems, we have δ is bounded by the number of support vectors. □

The bound of average convergence rate $(6\delta/C)\ln(N + 6r(C-1)\delta)$ clearly shows the linear scalability if $N >> \delta$. This can be true if the number of support vector is very limited.

6 Simulations and applications

We analyze our PRSVM and PRSVR by using synthetic data and a real-world geographic information system (GIS) database.

Throughout this section, the machine we used has a Pentium IV 2.26G CPU and 512M RAM. The operating system is Windows XP. The SVM$^{\text{light}}$ [20] version 6.01 was used as the local SVM solver. Parallel computing is virtually simulated in a single machine. Therefore, we ignore any communication overhead.

6.1 Demonstrations using synthetic data

We demonstrate our RSVM and RSVR (reduced PRSVM and PRSVR when $C = 1$) in two experiments with synthesized training data.

In one experiment, a total number of 1,000 two-dimensional training vectors are generated. This data set consists of 500 positive and 500 negative labeled vectors. Each class is generated from an independent Gaussian distribution with added random Gaussian noise.

We set the sample size r to be 100 and the regularization factor γ to be 0.2. The RSVM converges in 13 iterations. In order to demonstrate the weighting procedure, we choose three iterations (iteration 0 (initial state), iteration 6 and iteration 13) and plot the weights of the training vectors in Fig. 1. The darker a point appears, the higher weight the training sample has.

The RSVR (reduced PRSVR when $C = 1$) training procedure is demonstrated by using a synthesized one-dimensional training data set. This data set consists of 5,000 data points generated by a sigmoid function with 20% additive Gaussian noise.

In this experiment, we choose the sample size r to be 2,000, the tube width ϵ to be 0.2 and the regularization factor γ to be 0.2. A unit-variance Gaussian kernel is selected for fitting the sigmoid curve. The RSVR algorithm converges in 11 iterations. To demonstrate the weighting procedure, we choose three iterations (iteration 0, iteration 5 and iteration 11) and plot the weights of the training vectors in Fig. 2. In this figure, vectors with higher weights are plotted with darker and larger markers.

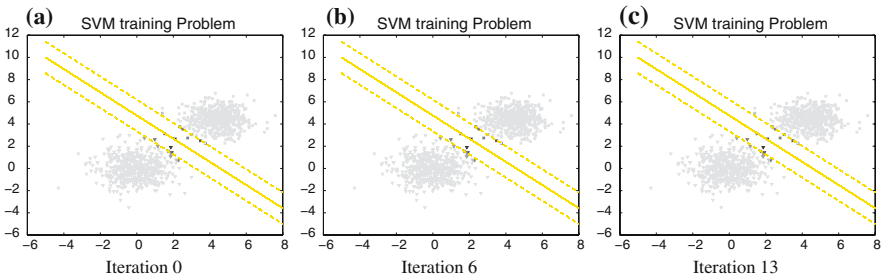

Fig. 1 Weights of training vectors in iterations. *Darker points* denote higher weights

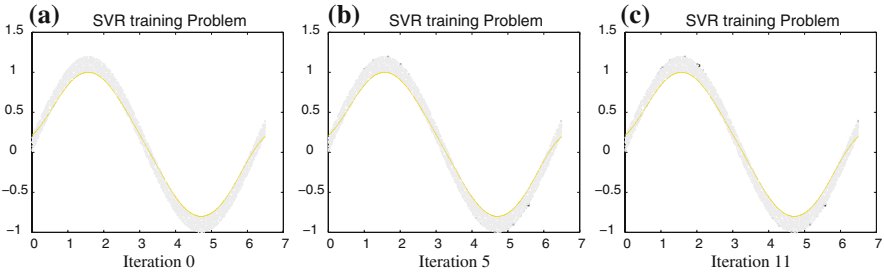

Fig. 2 Weights of training vectors in iterations. Vectors with higher weights are plotted with *darker* and *larger markers*

Table 1 Algorithm performance comparison of SVM^{light}, RSVM and PRSVM

Algorithm	C	Number of Iterations	Learning time (CPU Seconds)
SVM^{light}	1	–	11.7
RSVM	1	27	47.32
PRSVM	2	10	20.81
	4	7	15.52

Figs. 1 and 2 show how those "important" points stand out and get higher and higher probability to be sampled in the training process of randomized support vector machine and randomized support vector regression respectively.

6.2 Application on a geographic information system database

We select **covtype**, a geographic information system database, from the UCI Repository of machine learning databases as our PRSVM applications [6]. The covtype database consists of 5,81,012 instances. There are 12 measures but 54 columns of data: 10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables [5]. There are totally seven classes. We scale all quantitative variables to [0,1] and keep binary variable unchanged. We select 2,87,831 training vectors and use our PRSVM to classify class 4 against the rest.

We set the size of working size r to be 60,000, the regularization factor γ to be 0.2. Since classification performance is not our major concern, we do not fine turn the parameter. A linear kernel is used so that the number of support vector can be limited. This application turns out to be a very suitable case for testing PRSVM since the database has huge number of training data and the number of SVs is relatively small.

We try three cases with $C = 1$, $C = 2$ and $C = 4$ and compare the learning time with the SVM^{light} in Table 1. The results show that our implementation of RSVM and PRSVM achieves comparable result with the reported fastest algorithm SVM^{light}, though they cannot beat SVM^{light} in terms of computing speed for now. However, the lack of a theoretical convergence bound makes SVM^{light} not always preferable.

We plot the number of violators and support vectors (extremes) in each iterations in Fig. 3 to compare the performance of different number of working sites. The results show the scalability of our method. The numerical results match the theoretical result very well.

7 Discussion

In modem learning theory, a robust classifier or regressor usually comes from a sparse solution [13]. The beauty of support vector machine lies in its strength of optimally determining a subset of training vectors, the support vectors, that define the classifier/regressor. Since the complete problem, involving all the data points, are often too big to handle, iterative decomposition methods, where only a subset of the data points are handled at any one iteration, have been proposed. However, mathematical programming sometimes lacks simplicity, and no convergence bound has been proved for such standard decomposition algorithms. On the other hand, boosting [15] and stochastic boosting [16] algorithms also seek to determine those critical

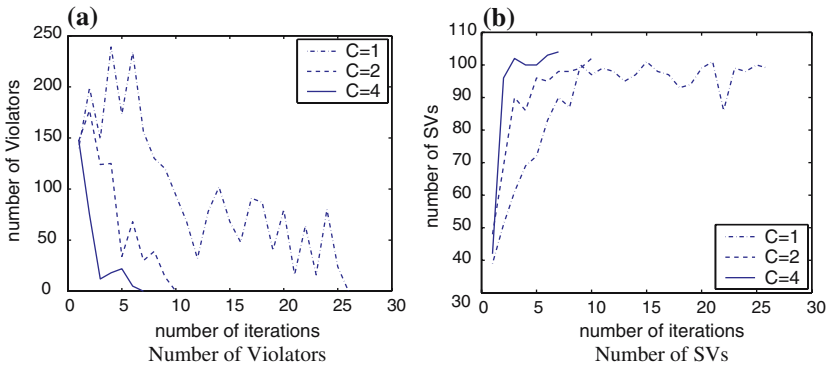

Fig. 3 Number of violators and SVs found in each iterations of PRSVM. This figure shows the effect of adding more servers. The system with more servers find the support vectors much faster than that with less servers

training samples iteratively. These algorithms are simple, but only locally optimal. The parallel randomized support vector machine is an algorithm that *combines* simple sampling/resampling techniques and mathematical programming together such that a global optimal solution is achieved with a simple convergence bound.

The proposed PRSVM/PRSVR algorithm takes advantage of the sparse solution of support vector machine. That is, the algorithm is suitable for applications with relatively limited numbers of support vectors. A well-defined problem, in which a robust solution exists, satisfies this condition. In Fig. 2, one may observe that the number of support vectors are limited even if a Gaussian kernel is used, which is able to lift the dimension of feature vectors to infinity.

We cannot deny, however, that the condition $r \geq 6\delta^2$ may be too restrictive and may cause some inefficiency in the actual performance. In some of our experiments (not presented in this paper), the actual size of a working set r can be substantially smaller than the bound of $6\delta^2$ and yet the algorithm converges much faster in terms of cpu seconds. But we have not found theoretical evidence to guarantee a convergence if the condition $r \geq 6\delta^2$ does not hold.

8 Conclusions

We have made four contributions in this paper. First, we propose a parallel randomized sampling algorithm that is able to solve general nonseparable SVM training problems. This is achieved by using KKT conditions as the criteria of identifying violators and extremes. Second, our algorithm supports multiple working sets that may work in parallel to take advantage of multiple computing facilities. Third, we prove that the PRSVM and PRSVR have a fast average convergence rate. Last, our numerical results show that multiple working sets have a scalable computing advantage. The provable convergence bound and scalable results hold the potential to make our algorithm the preferred on in many applications.

Further research is going to be conducted to improve the performance of the PRSVM/PRSVR. If we could relax the condition $r \geq 6\delta^2$, each working set then may contain less number of training samples so that the algorithm may become more efficient and suitable for additional applications that have insufficient training samples.

References

1. Adler I, Shamir R (1993) A randomized scheme for speeding up algorithms for linear and convex programming with high constraints-to-variable ratio. Math Program 61(1–3): 39–52
2. Balcazar J, Dai Y, Tanaka J, Watanabe O (2001) Provably fast training algorithm for support vector machines. In: The 1st IEEE International Conference on Data Mining (ICDM01)
3. Balcazar J, Dai Y, Tanaka J, Watanabe O (2002) Provably fast training algorithm for support vector machines. Technical Reports on Mathematical and Computing Sciences TR-C160, Tokyo Institute of Technology, Tokyo
4. Bastiere A (1998) Methods for multisensor classification of aiborne targets integrating evidence theory. Aerospace Sci Technol 2: 401–411
5. Blackard JA, Dean DJ (1999) Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover type from cartographic variables. Comput Electronics Agric 24: 131–151
6. Blake CL, Merz C (1998) UCI repository of machine learning databases http://www.ics.uci.edu/∼mlearn/MLRepository.html
7. Cesa-Bianchi N (2004) Applications of regularized least squares to classification problems. In: Lecture Notes in Artificial Intelligence. Springer, Berlin, pp 14–18
8. Chellappa R, Zheng Q, Burlina P, Shekhar C, Eom KB (1997) On the positioning of multisensor imagery for exploitation and target recognition. In: Proceedings of the IEEE, vol 85, pp 120–138
9. Clarkson KL (1988) Las vegas algorithms for linear and integer programming when the dimension is small. In: The 29th IEEE symposium on foundations of computer science (FOCS'88)
10. Collobert R, Bengio S (2001) Svmtorch: Support vector machines for large-scale regress problems. J Mach Lear Res 1: 276–285
11. Collobert R, Bengio S, Bengio Y (2001) A parallel mixture of svms for very large scale problems. In:Neural Information Processing Systems, pp 633–640
12. Cortes C, Vapnik V (1995) Support-vector networks. Machine Learning 20: 273–297
13. Duda RO, Hart PE, Stork DG (2000) Pattern Classification. Wiley
14. Flake GW, Lawrence S (2002) Efficient svm regression training with smo. Mach Learn 46(1–3): 271–290
15. Freund Y, Schapire RE (1995) A decision-theoretic generalization of on-line learning and application to boosting. J Comput Syst Sci 55(1): 119–139
16. Friedman JH (2002) Stochastic gradient boosting. Comput Stat Data Anal 38(4): 367–378
17. Gartner B, Welzl E (2000) A simple sampling lemma: Analysis and applications in geometric optimization. In: Proceeding of the 16th annual ACM symposium on computational geometry (SCG)
18. Graf HP, Cosatto E, Bottou L, Dourdanovic I, Vapnik V (2005) Parallel support vector machine: The cascade svm. In: Advances in neural information processing systems
19. Jeon B, Landgrebe DA (1999) Decision fusion approach for multitemporal classification. IEEE Trans Geosci Remote Sens 37: 1227–1233
20. Joachims T (1998) Making large-scale svm learning practical. In: Advances in Kernel Methods - Support Vector Learning. MIT Press, Cambridge, pp 169–184
21. Li S, Kwok JT-Y, Tsang IW-H, Wang Y (1998) Fusion images with different focuses using support vector machines. IEEE Trans Neural Netw 15: 1555–1560
22. Mangasarian O, Musicant D (1999) Successive overrelaxation for support vector machines. IEEE Trans Neural Netw 10(5): 1032–1037
23. Osuna E, Freund R, Girosi F (1997) An improved training algorithm for support vector machines. In: IEEE workshop on neural networks for signal processing. pp 276–285
24. Osuna E, Freund R, Girosi F (1997) Support vector machine: training and applications, Artificial Intelligence Laboratory Memo No. 1602, Massachusetts Institute of Technology Massachusetts
25. Platt JC (1998) Sequenctial minimal optimization: a fast algorithm for training support vector machines, Technical Report MSR-TR-98-14, Microsof Research
26. Pohl C, Genderen JLV (1998) Mutisensor image fusion in remote sensing: Concepts, methods and applications. Int J Remote Sens 19: 823–854
27. Schmidt MS (1996) Identity speaks with support vector networks. In: The 28th symposium on the interface (INTERFACE-96), Sydney
28. Vapnik V (1995) The Nature of Statistical Learning Theory. Springer, New York

Authors Biography

Yumao Lu is currently a research scientist at Yahoo! Inc., Applied Research, Sunnyvale, California. He received a Ph. D. degree in Electrical Engineering at the University of California, Los Angeles in 2005, an M. Phil. degree in Industrial Engineering at the Hong Kong University of Science and Technology in 2001 and a B. Eng. degree in Automatic Control at the Huazhong University of Science and Technology, Wuhan, China in 1999. Dr. Lu's Research interests span the fields of machine learning, information retrieval, convex and non-convex optimization, data mining, bio-informatics and medical imaging.

Vwani Roychowdhury received the Ph.D. in electrical engineering from Stanford University in 1989. From 1991 to 1996, he was a faculty member with the School of electrical and Computer Engineering, Purdue University, where he was promoted to Associate Professor in 1995. In 1996, he joined the University of California, Los Angeles, where he has been a Professor of electrical engineering since 1998. Professor Roychowdhury's research interests span a wide range of topics that deal with information science, including advanced statistical processing and learning theory, models of computation, quantum and nanoelectronic computation, quantum information processing, fault-tolerant computation, combinatorics and information theory.