

# Parallel Randomized Support Vector Machine

Yumao Lu and Vwani Roychowdhury

University of California, Los Angeles, CA 90095, USA

**Abstract.** A parallel support vector machine based on randomized sampling technique is proposed in this paper. We modeled a new LP-type problem so that it works for general linear-nonseparable SVM training problems unlike the previous work [2]. A unique priority based sampling mechanism is used so that we can prove an average convergence rate that is so far the fastest bounded convergence rate to the best of our knowledge. The numerical results on synthesized data and a real geometric database show that our algorithm has good scalability.

## 1 Introduction

Sampling theory has a long successful history in optimization [6, 1]. The application to the SVM training problem is first proposed by Balcazar et al. in 2001 [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 arbitrary small regularization factor  $\gamma$  ( $D$  and  $1/k$  in [2] and [3]). They also stated that there were number of implementation difficulties so that no relevant results could be provided [3].

We model a LP-type problem such that the general linear nonseparable problem can be covered by our randomized support vector machine (RSVM). In order to take advantage of distributed computing facilities, we proposed a novel parallel randomized SVM (PRSVM) in which multiple working sets can be worked on simultaneously. The basic idea of the PRSVM 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 [7, 10] that lacks of a convergence bound, our algorithm, the PRSVM, on average, converges to the global optimum classifier/regressor in less than  $(6\delta \ln(N + 6r(C - 1)\delta)/C$  iterations, where  $\delta$  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 for a working set. Since the RSVM is a special case of PRSVM, our proof naturally works for the RSVM. Note that, when  $C = 1$ , our result reduces to Balcazar's bound [3].

This paper is organized as follows. The support vector machine is introduced and formulated in the next section. Then, we present the parallel randomized support vector machine algorithm. The theoretical global convergence is given in the fourth section followed by a presentation of a successful application. We conclude our result in Section 6.

## 2 Support Vector Machine and Randomized Sampling

We prepare fundamentals and basic notations on SVM and randomized sampling technique 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  $i = 1, \dots, N$  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 [12]. The support vector machine (SVM) 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  $\mu_i, i = 1, \dots, N$  are introduced. The SVM problem is defined as follows:

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

where the scalar  $\gamma$  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  $X$  as

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

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

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

A nonlinear kernel function can be used for nonlinear separation of the training data. In that case, the gram matrix  $Z Z^T$  is replaced by a kernel matrix  $k(x, \tilde{x}) \in \mathbf{R}^{N \times N}$ . Our PRSVM that is described in the following section can be kernelized and therefore is able to keep the full advantages of the SVM.

### 2.2 The Sampling Lemma, LP-Type Problem and KKT Condition

An abstract problem is denoted by  $(\mathcal{S}, \phi)$ . Let  $\mathcal{X}$  be the set of training vector. That is, each element of  $\mathcal{X}$  is a row vector of the matrix  $X$ . Throughout this paper, we use *CALLIGRAPHIC* style letters to denote sets of the row vectors of a matrix denoted by the same letter with *italian* style. Here,  $\phi$  is a mapping from a given subset  $\mathcal{X}_{\mathcal{R}}$  of  $\mathcal{X}$  to the local solution of problem (1) with constraints corresponding to  $X_{\mathcal{R}}$  and  $\mathcal{S}$  is of size  $N$ . Define

$$\begin{aligned} \mathcal{V}(\mathcal{R}) & := \{s \in \mathcal{S} \setminus \mathcal{R} \mid \phi(\mathcal{R} \cup \{s\}) \neq \phi(\mathcal{R})\}, \\ \mathcal{E}(\mathcal{R}) & := \{s \in \mathcal{R} \mid \phi(\mathcal{R} \setminus \{s\}) \neq \phi(\mathcal{R})\}. \end{aligned}$$

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

$$s \text{ violates } \mathcal{R} \Leftrightarrow s \text{ is extreme in } \mathcal{R} \cup \{s\}.$$

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

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

Gartner proved the following sampling lemma [9]:

**Lemma 1.** (*Sampling Lemma*). For  $0 \leq r < N$ ,

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

**Proof.** By definitions, we have

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

where  $[\cdot]$  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.  $\square$

The problem  $(\mathcal{S}, \phi)$  is said to be a LP-type problem if  $\phi$  is monotone and local (see Definition 3.1 in [9]). Balcazar proved that the problem (1) is a LP-type problem [2]. So is the problem (2). We use the same definitions given by [9] to define the *basis* and *combinatorial dimension* as follows. For any  $\mathcal{R} \subseteq \mathcal{S}$ , a *basis* of  $\mathcal{R}$  is a inclusion-minimal subset  $\mathcal{B} \subseteq \mathcal{R}$  with  $\phi(\mathcal{B}) = \phi(\mathcal{R})$ . The *combinatorial dimension* of  $(\mathcal{S}, \phi)$ , denoted by  $\delta$ , is the size of a largest basis of  $\mathcal{S}$ . For a LP-type problem  $(\mathcal{S}, \phi)$  with combinatorial dimension  $\delta$ , the sampling lemma yields

$$v_r \leq \delta \frac{N-r}{r+1}. \quad (3)$$

This follows that  $|\mathcal{E}(\mathcal{R})| \leq \delta$ .

Then, we are able to relate the definitions of the extremes, violators and the basis to our general SVM training problem (1) or (2). For any local solution  $\theta^p$  or  $\alpha^p$  of problem  $(\mathcal{X}_p, \phi)$ , 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:

$$Z\theta^* \geq \mathbf{1} - \mu^*, \quad \mu^* \geq 0, \quad 0 \leq \alpha^* \leq \gamma \mathbf{1},$$

$$\theta^* = Z^T \alpha^*, (\gamma - \alpha_i^*) \mu_i^* = 0, i = 1, \dots, N.$$

Since the  $\mu_i$  and  $\alpha_i$  for the training vector  $x_i$  is always 0 for  $x_i \in \mathcal{X} \setminus \mathcal{X}_p$ , 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  $(\mathcal{X}_p, \phi)$ . The size of the largest basis,  $\delta$  is naturally the largest number of support vectors for all subproblems  $(\mathcal{X}_p, \phi)$ ,  $\mathcal{X}_p \subseteq \mathcal{X}$ . For separable problems,  $\delta$  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  $\delta$  before we actually solve the problem. What we can do is to set a sufficiently large number to bound  $\delta$  from above.

### 3 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  $p = 1, 2, \dots, C$ . We also assume that each working site contains  $r$  training vectors, where  $r \geq 6\delta^2$  and  $\delta$  denotes the combinatorial dimension of the SVM problem.

Define a function  $u(\cdot)$  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  $|\mathcal{W}| = u(\mathcal{X})$ . 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 (PR SVM) 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**  $t = 1, 2, \dots$

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  $r \geq 6\delta^2$ );  
 $\mathcal{S}^{p+1} := \mathcal{S}^p \setminus \mathcal{W}_p$ ;  
**End For**
2. Each site with priority  $p$ ,  $p \leq C$  solves the local partial problem and record the solution  $\theta^p$ . Send this solution to all other sites  $q$ ,  $q \neq p$ .

3. Each site with priority  $q$ ,  $q = 1, \dots, C + 1$ , checks the solution  $\theta^p$  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,

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

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  $\theta^p$ .

The priority setting of working sets actually defines the order of sampling. The highest priority server gets the first sampled batch of data, lower one gets the second batch and so on. This kind of sequential behavior is designed to help define 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 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 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,  $\delta$ , 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 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 nonseparable problems to an equivalent separable problems by assuming an arbitrarily small  $\gamma$ . 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.

#### 4 Proof of the Average Convergence Rate

We prove the average number of iterations executed in our algorithm, PRSVM, 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 tradition RSVM becomes a special case of our PRSVM.

**Theorem 1.** *For general SVM training problem the average number of iterations executed in the PRSVM 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 the bound of the number of successful updates. Let  $\mathcal{V}_p$  denote the set of violators from site with priority  $q \geq p$  for the solution  $\theta^p$ . 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

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

Therefore, at each successful update, we have

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

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

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

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  $\delta$  successful updates. That is, after  $k$  successful updates,  $u(\{x_i\}) \geq 2^{k/\delta}$ .

Therefore, we have

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

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 (3), we have

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

By Markov equality, we have

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

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 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  $\delta$  is bounded by the number of support vectors.  $\square$

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

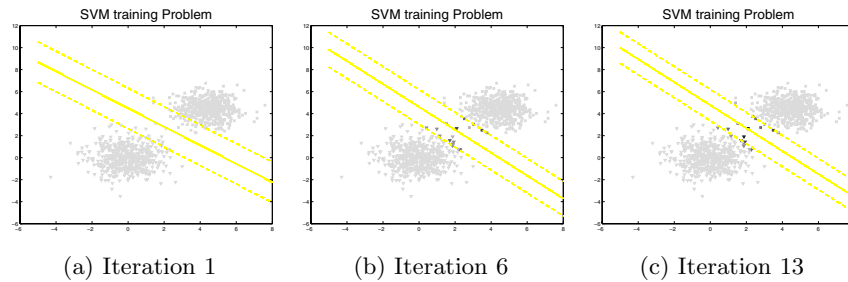
## 5 Simulations and Applications

We analysis our PRSVM by using synthesized data and a real-world geographic information system (GIS) database.

Through out this section, the machine we used has a Pentium IV 2.26G CPU and 512M RAM. The operation system is Windows XP. The SVM<sup>light</sup> [11] 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.

### 5.1 Synthesized Demonstration

We demonstrate our RSVM (reduced PRSVM when  $C = 1$ ) training procedure by using a synthesized two-dimensional training data set. This data set consists of 1000 data points: 500 positive and 500 negative. Each class is generated from an independent Gaussian distribution. Random noise is added.



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

We set the sample size  $r$  to be 100 and the regularization factor  $\gamma$  to be 0.2. The RSVM converges in 13 iteration. In order to demonstrate the weighting procedure, we choose three iterations (iteration 1, 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. Fig. 1 shows that how those "important" points stand out and get higher and higher probability to be sampled.

### 5.2 Application in a Geographic Information System Database

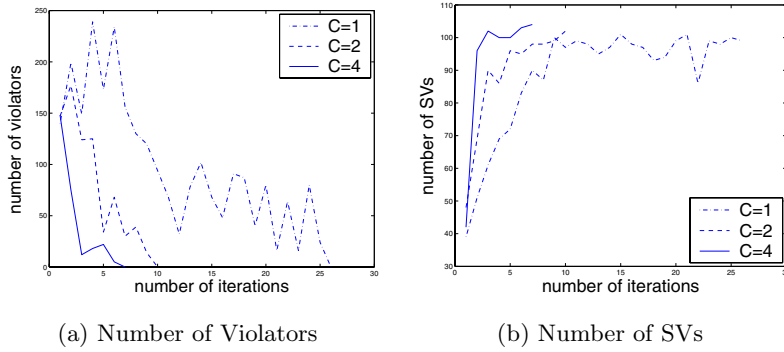
We select covtype, a geographic information system database, from the UCI Repository of machine learning databases as our PRSVM applications [5]. The covtype database consists of 581,012 instances. There are 12 measures but 54 columns of data: 10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables [4]. There are totally 7 classes. We scale all quantitative variables to  $[0,1]$  and keep binary variable unchanged. We select 287831 training vectors and use our PRSVM to classify class 4 against the rest. This is a very suitable database for testing PRSVM since the database has huge number of training data and the number of SVs is limited.

We set the size of working size  $r$  to be 60000, the regularization factor  $\gamma$  to be 0.2. We try three cases with  $C = 1$ ,  $C = 2$  and  $C = 4$  and compare the learning time with the  $\text{SVM}^{\text{light}}$  in Table 1. The results show that our implementation of RSVM and PRSVM achieves comparable result with the reported fastest algorithm  $\text{SVM}^{\text{light}}$ , though they cannot beat  $\text{SVM}^{\text{light}}$  in terms of computing speed for now. However, the lack of a theoretical convergence bound makes  $\text{SVM}^{\text{light}}$  not always preferable.

**Table 1.** Algorithm performance comparison of SVM<sup>light</sup>, RSVM and PRSVM

Algorithm	C	Number of Iterations	Learning Time (CPU Seconds)
SVM <sup>light</sup>	1	-	11.7
RSVM	1	27	47.32
PRSVM	2	10	20.81
	4	7	15.52

We plot the number of violators and support vectors (extremes) in each iterations in Fig. 2 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.



**Fig. 2.** 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 will find the support vectors much faster than that with less servers.

## 6 Conclusions

The proposed PRSVM has the following advantages over previous works. It is able to solve general nonseparable SVM training problems. This is achieved by using KKT condition as the criterion of identifying violators and extremes. Second, our algorithm supports multiple working sets that may work parallel. Multiple working sets have more freedom than normal gradient based parallel algorithms since no synchronization and no special solver is required. Our PRSVM also has a provable and fast average convergence bound. Last, our numerical results show that multiple working sets have scalable computing advantage. The provable convergence bound and scalable results make our algorithm more preferable in some applications.

Further research is going to be conducted to accelerate the performance of the PRSVM. Intuitively, the weighting mechanism may be able to be improved so that the initial iterations play a more determinant role.

## References

1. Ilan Adler and Ron Shamir. A randomized scheme for speeding up algorithms for linear and convex programming with high constraints-to-variable ratio. *Mathematical Programming*, 1993.
2. Jose Balcazar, Yang Dai, Junichi Tanaka, and Osamu Watanabe. Provably fast training algorithm for support vector machines. *Proceedings of First IEEE International Conference on Data Mining (ICDM01)*, 2001.
3. Jose Balcazar, Yang Dai, and Osamu Watanabe. Provably fast support vector regression using random sampling. *Proceedings of SIAM Workshop on Discrete Mathematics and Data Mining*, April 2001.
4. Jock A. Blackard and Denis J. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover type from cartographic variables. *Computer and Electronics in Agriculture*, 24, 1999.
5. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
6. Kenneth L. Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *Proceeding of 29th IEEE Symposium on Foundations of Computer Science (FOCS'88)*, 1988.
7. Ronan Collobert, Samy Bengio, and Yoshua Bengio. A parallel mixture of svms for very large scale problems. In *Neural Information Processing Systems*, pages 633–640, 2001.
8. Tatjana Eitrich and Bruno Lang. Shared memory parallel support vector machine learning. Technical report, ZAM Publications on Parallel Applications, 2005.
9. Bernd Gartner and Emo Welzl. A simple sampling lemma: Analysis and applications in geometric optimization. *Proceeding of the 16th Annual ACM Symposium on Computational Geometry (SCG)*, 2000.
10. Hans Peter Graf, Eric Cosatto, Leon Bottou, Igor Dourdanovic, and Vladimir Vapnik. Parallel support vector machine: The cascade svm. In *Advances in Neural Information Processing Systems*, 2005.
11. Thorsten Joachims. Making large-scale svm learning practical. *Advances in Kernel Methods - Support Vector Learning*, pages 169–184, 1998.
12. Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.