

# Distributed Parallel Support Vector Machines in Strongly Connected Networks

Yumao Lu, Vwani Roychowdhury, Lieven Vandenberghe

**Abstract**— We propose a distributed parallel support vector machine (DPSVM) training mechanism in a configurable network environment for distributed data mining. The basic idea is to exchange support vectors among a strongly connected network (SCN) so that multiple servers may work concurrently on distributed data set with limited communication cost and fast training speed. The percentage of servers that can work in parallel and the communication overhead may be adjusted through network configuration. The proposed algorithm further speeds up through online implementation and synchronization. We prove that the global optimal classifier can be achieved iteratively over a strongly connected network. Experiments on a real world data set show that the computing time scales well with the size of the training data for most networks. Numerical results show that a randomly generated SCN may achieve better performance than the state of the art method, Cascade SVM, in terms of total training time.

## I. INTRODUCTION

Distributed classification is necessary if a centralized system is infeasible due to geographical, physical and computational constraints. The objectives of distributed data mining usually include robustness to changes in the network topology [1], efficient representation for high-dimensional and massive data sets, least synchronization and communication, least duplication, better load balancing and good decision precision and certainty.

The classification problem is an important problem in data mining. The support vector machine (SVM), which implements the principle of structural error minimization [2], [3], is one of the most popular classification algorithms in many fields [4], [5], [6].

SVMs are ideally suited for a framework where different sites can potentially exchange only a small number of training vectors. For a data set at a particular site, the support vectors (SVs) are the natural representatives of discriminant information of the database *and* the optimal solution to a local classifier. Distributed and parallel support vector machines, which emphasize global optimality, draw more and more attentions in recent years.

Current parallel support vector machines (PSVM) include matrix multi-coloring successive overrelaxation (SOR) method [7], [8] and variable projection method (VPM) in sequential minimal optimization (SMO) [9]. These methods are excellent in terms of speed. However, they all need centralized access to the training data and therefore cannot be used in distributed classification applications. In summary, the current popular parallel methods are not distributed.

On the other hand, current distributed supported vector machines (DSVM) are not taking enough advantage of parallel

computing. The main obstacle is that the more servers work concurrently, the more data transmitted, causing excessive data accumulation that slows down the training process. Syed et al. proposed the first distributed support vector machine (DSVM) algorithm that finds SVs locally and processes them altogether in a central processing center in 1999 [10]. Their solution, however, is not global optimal. Caragea et al. in 2005 improved this algorithm by allowing the data processing center to send support vectors back to the distributed data source and iteratively achieve the global optimum [11]. This model is slow due to extensive data accumulation in each site [12]. Navia-Vazquez et al. proposed distributed semiparametric support vector machine to reduce the communication cost by transmitting a function of subset of support vectors,  $R$ . Their algorithm, however, is suboptimal depending on the definition of  $R$ . In order to accelerate DSVM, Graf et al. in 2004 had an algorithm that implemented distributed processors into cascade top-down network topology, namely Cascade SVM [13]. The bottom node of the network is the central processing center. To the best of our knowledge, the Cascade SVM is the fastest DSVM that is globally optimal. This is the first time that network topology was taken into consideration to accelerate the distributed training process. However, there is no comparison to other network topologies. It is not clear either that their distributed SVM may converge in a more general network.

In this paper, we propose a distributed parallel support vector machine (DPSVM) for distributed data classification in a general network configuration, namely strongly connected network. The objective of the distributed classification problem is to classify distributed data, i.e., determine a single global classifier, by judiciously sampling and distributing subsets of data among the various sites. Ideally, a single site or server should not end up storing the complete data set; instead, the different sites should exchange a minimum number of data samples, and together converge to a single global solution in an iterative fashion.

The basic idea of DPSVM is to exchange SVs over a strongly connected network and update (instead of recompute) the local solutions iteratively, based on the SVs that a particular site receives from its neighbors. We prove that our algorithm converges to a globally optimal classifier (at every site) for arbitrarily distributed data over a strongly connected network. Recall that a strongly connected network is a directed graph where there is a directed path between any pair of nodes; the strongly connected property makes sure that the critical constraints are shared across all the nodes/sites in the network, and each site converges to the globally optimal solution.

Although this proof does not guarantee the convergence speed, Lu and Roychowdhury in 2006 proved that a similar parallel SVM has the provably fastest average convergence rate among all decomposition algorithms if each site randomly selects training vectors that follows a carefully designed distribution [14].

Practically, the DPSVM achieves the fast convergence time. The proposed algorithm is analyzed under a variety of network topologies as well as other configurations such as network size, online and offline implementation etc., which have dramatic impact on the performance in terms of convergence speed and data accumulation. We show that the efficiency of DPSVM and the overall communication overhead can be improved by controlling the network sparsity. Specifically, a randomly generated strongly connected network with a sparsity constraint outperforms the Cascade SVM, that is reported to be the fastest distributed SVM algorithm to the best of our knowledge.

This paper is organized as follows. We present our distributed classification algorithm in the next section followed by the proof of the global convergence in Section III. We introduce the detailed algorithm implementation in Section IV. The performance study is given in Section V. We conclude in Section VI.

## II. DISTRIBUTED SUPPORT VECTOR MACHINE

### A. Problem

We consider the following problem: the training data are arbitrarily distributed in  $L$  sites. Each site is a node within a strongly connected network, defined as follows.

*Definition 1:* A *Strongly connected network (DCN)* is a directed network in which it is possible to reach any node starting from any other node by traversing edges in the direction(s) in which they point.

Differing from a weakly connected network, which becomes a connected (undirected) graph if all of its directed edges are replaced with undirected edges, a DCN makes sure each node in the work can be accessed from any other nodes. This is an important property required by our convergence proof (See the proof of Theorem 1).

There are  $N_l$  training vectors in site  $l$  and  $N$  training vectors in all sites, where  $N_l, \forall l$  can be an arbitrary integer, such that  $\sum_{l=1}^L N_l = N$ . Each training vector is denoted by  $z_i$   $i = 1, \dots, N$  where  $z_i \in \mathbf{R}^n$ , and  $y_i \in \{+1, -1\}$  is its label.

The global SVM problem, *i.e.*, the traditional centralized problem is briefly summarized in the next section.

### B. Support Vector Machine

In a SVM training problem, we are seeking a hyperplane to separate a set of positively and negatively labeled training data. The hyperplane is defined by  $w^T z + b = 0$ , where the parameter  $w \in \mathbf{R}^n$  is a vector orthogonal to the hyperplane and  $b \in \mathbf{R}$  is the bias. The decision function is the hyperplane classifier

$$H(x) = \text{sign}(w^T z + b).$$

The hyperplane is designed such that  $y_i(w^T z_i + b) \geq 1$ . The margin is defined by the distance of the two parallel

hyperplanes  $w^T z + b = 1$  and  $w^T z + b = -1$ , *i.e.*  $2/\|w\|_2$ . The margin is related to the generalization of the classifier [2]. One may easily transform the decision function to

$$H(x) = \text{sign}(g^T x). \quad (1)$$

where vector  $g = [w; b] \in \mathbf{R}^{n+1}$  and  $x = [z; 1] \in \mathbf{R}^{n+1}$ . The benefit of this transformation is to simplify the formulation so that the bias  $b$  does not need to be calculated separately. For general linear nonseparable problems, a set of slack variables  $\xi_i$ 's is introduced. The SVM training problem for the nonseparable problem is defined as follows:

$$\begin{aligned} & \text{minimize} && (1/2)g^T x + \gamma \mathbf{1}^T \xi \\ & \text{subject to} && y_i g^T \phi(g_i) \geq 1 - \xi_i, \quad i = 1, \dots, N \\ & && \xi \geq 0 \end{aligned} \quad (2)$$

where  $\mathbf{1}$  is a vector of ones,  $\phi$  is a lifting function and the scalar  $\gamma$  is called regularization parameter that is usually empirically selected to reduce the testing error rate.

The corresponding dual of the problem (2) is shown as follows:

$$\begin{aligned} & \text{maximize} && -(1/2)\alpha^T Q \alpha + \mathbf{1}^T \alpha \\ & \text{subject to} && 0 \leq \alpha \leq \gamma \mathbf{1} \end{aligned} \quad (3)$$

where the Gram matrix  $Q$  has the component  $Q_{ij} = y_i y_j \phi(x_i)^T \phi(x_j)$ .

It is common to replace  $\phi(x)^T \phi(\tilde{x})$  by a kernel function  $F(x, \tilde{x})$  such that  $Q_{ij} = K_{ij} y_i y_j$  and  $K_{ij} = F(x_i, x_j)$ , where  $K \in \mathbf{R}^{N \times N}$  is the so-called kernel matrix. The nonlinear kernel may lift the dimension of training vectors to a higher dimension so that they can be separated linearly. If the kernel matrix  $K$  is positive definite, problem (3) is guaranteed to be strictly convex. There are several popular choices of nonlinear kernel functions: inhomogeneous polynomial kernels

$$F(x, \tilde{x}) = x^T \tilde{x} + 1,$$

and gaussian kernels

$$F(x, \tilde{x}) = \exp\left(-\frac{\|x - \tilde{x}\|^2}{2\sigma^2}\right).$$

Correspondingly we have the decision function of the form

$$H(x) = \text{sign}\left(\sum_{i=1}^N y_i \alpha_i F(x_i, x)\right).$$

For the general SVM optimization problem, the complementary slackness condition has the form

$$\alpha_i [y_i (g^T x_i) - 1 + \xi_i] = 0 \text{ for all } i = 1, \dots, N, \quad (4)$$

in the optimum. Therefore, support vectors are the vectors that lie in between the margin boundary (including those on the boundary) and that are misclassified.

### C. Efficient Information Carrier: Support Vectors

Support vectors (SVs), namely the training data that have non-zero  $\alpha$  values, lie physically on the margin or are misclassified. Those vectors carry all the classification information of the local data set. Exchanging support vectors, instead of

moving all the data, is a natural way to fuse information among distributed sites.

Support vectors (SVs) are a natural representation of the discriminant information of the underlying classification problem. The basic idea of the distributed support vector machines is to exchange SVs over a strongly connected network and update the local solutions for each site iteratively. We exploit the idea of exchanging SVs in a deterministic way instead of a randomized way [14] in our distributed learning algorithms. Our algorithm is based on the observation that the number of support vectors (SVs) may be very limited for the local classification problems. We prove that our algorithm converges to a global optimal classifier for an arbitrarily distributed database across a strongly connected network.

#### D. Algorithm

The distributed support vector machines (DPSVM) algorithm works as follows. Each site within a strongly connected network classifies subsets of training data locally via SVM and passes the calculated SVs to its descendant sites and receives SVs from its ancestor sites and recalculates the SVs and passes them to its descendant sites and so on. The algorithm of DPSVM consists of the following steps.

*Initialization:* We initialize the algorithm with iteration  $t = 0$  and local support vector set in site  $l$  at iteration  $t$ ,  $V^{t,l} = \emptyset$ ,  $\forall l, t = 0$ . The training set at iteration  $t$  in site  $l$  is denoted by  $S^{t,l}$ . The  $S^{0,l}$  is initialized arbitrarily such that  $\cup_{l=1}^L S^{0,l} = S$ , where  $S$  denotes the total sample space.

*Iteration:* Each iteration consists of the following steps.

- 1)  $t := t + 1$ .
- 2) For any site  $l$ ,  $l = 1, \dots, L$ , once it receives SVs from its ancestor sites, we repeat the following steps.
  - a) Merge training vectors  $X_l^{\text{add}} = \{x_i : x_i \in V^{t,m}, \forall m \in \text{UPS}_l, x_i \notin S^{t-1,l}\}$ , where  $\text{UPS}_l$  is the set of all immediate ancestor sites of site  $l$ , to the current training sets  $S^{t,l}$  of the current site  $l$ .
  - b) Solve the problem (3). Record the optimal objective value  $h^{t,l}$  and the solution  $\alpha^{t,l}$ .
  - c) Find the set  $V^{t,l} = \{x_i : \alpha_i^{t,l} > 0\}$  and pass them to all immediate descendant sites.
- 3) If  $h^{t,l} = h^{t-1,l}$  for all  $l$ , stop; otherwise, go to step 2.

Every site starts to work once it receives SVs from its ancestor sites. In step 2(b), we start solving the newly formed problem from the best solution currently available and update this solution locally. We use SVM<sup>light</sup> [15] as our local solver. The numerical results show that the computing speed can be dramatically increased by applying the available best solution  $\alpha^{t-1}$  as the initial starting point in Step 1(b). We are going to discuss this issue later in the paper.

To demonstrate the convergence, we randomly generate 200 independent two-dimensional data sampled from two independent Gaussian distributions. We randomly distributed all the data to 5 sites. We assume the 5 site forms a ring network. The SVM problem is solved locally, and pass the support vectors to their descendant sites. The DPSVM converges in 4 iterations and the local results are shown in the Fig. 1. One may observe

the decreasing of the local margins and they converge to the global optimal classifier.

We prove the global convergence in Section III.

### III. PROOF OF CONVERGENCE

We prove our algorithm, DPSVM, converges to the global optimal classifier in finite steps in this section.

*Lemma 1: Global Lower Bound:* Let  $h^*$  be the global optimum value, which is the optimum value of the following problem

$$\begin{aligned} & \text{maximize} && -(1/2)\alpha^T Q \alpha + \mathbf{1}^T \alpha \\ & \text{subject to} && 0 \leq \alpha \leq \gamma \mathbf{1}. \end{aligned} \quad (5)$$

where  $Q$  is the Gram matrix for all training samples. Then,

$$h^{t,l} \leq h^* \quad \forall t, l.$$

*Proof.* Define  $\tilde{\alpha}^{t,l}$  as follows

$$\tilde{\alpha}^{t,l} = \begin{cases} \alpha^{t,l}, & \text{if } i \in S^{t,l} \\ 0, & \text{otherwise} \end{cases}$$

Then the proof follows immediately by the fact that the solutions  $\tilde{\alpha}^{t,l} \forall t, l$  are always a feasible solution to the global problem (5) with objective value  $h^{t,l}$ .  $\blacktriangleleft$

Lemma 1 shows that each solution of the subproblem serves as a lower bound for the global SVM problem.

*Lemma 2: Nondecreasing:* The objective value for any site, say site  $l$ , at iteration  $t$  is always greater than or equal to the maximum of the objective of the same site in the last iteration and the maximal objective values of its immediate ancestor sites, site  $m$ ,  $\forall m \in \text{UPS}_l$ , from which site  $l$  receives SVs in the current iteration. That is,

$$h^{t,l} \geq \max\{h^{t-1,l}, \max_{m \in \text{UPS}_l} \{h^{t,m}\}\}$$

*Proof.* Assume  $t > 1$  without losing generality. Let us first assume site  $l$  receives SVs only from one of its ancestor site, say site  $m$ . The solutions corresponding to the objectives  $h^{t,l}$ ,  $h^{t-1,l}$  and  $h^{t,m}$  are  $\alpha^{t,l}$ ,  $\alpha^{t-1,l}$  and  $\alpha^{t,m}$ . Denote the nonzero part of  $\alpha^{t,m}$  by  $\alpha_1$ . The corresponding Gram matrix is denoted by  $Q_1$ . Therefore,

$$h^{t,m} = -\alpha_1^T Q_1 \alpha_1 + \mathbf{1}^T \alpha_1.$$

Define  $\alpha_2 = \alpha^{t-1,l}$  and the corresponding Gram matrix to be  $Q_2$ . We have

$$h^{t-1,l} = -\alpha_2^T Q_2 \alpha_2 + \mathbf{1}^T \alpha_2.$$

Since some SVs corresponding to  $\alpha_1$  may be the same as some of those corresponding to  $\alpha_2$ , we reorder and rewrite  $\alpha_1$  and  $\alpha_2$  as follows:

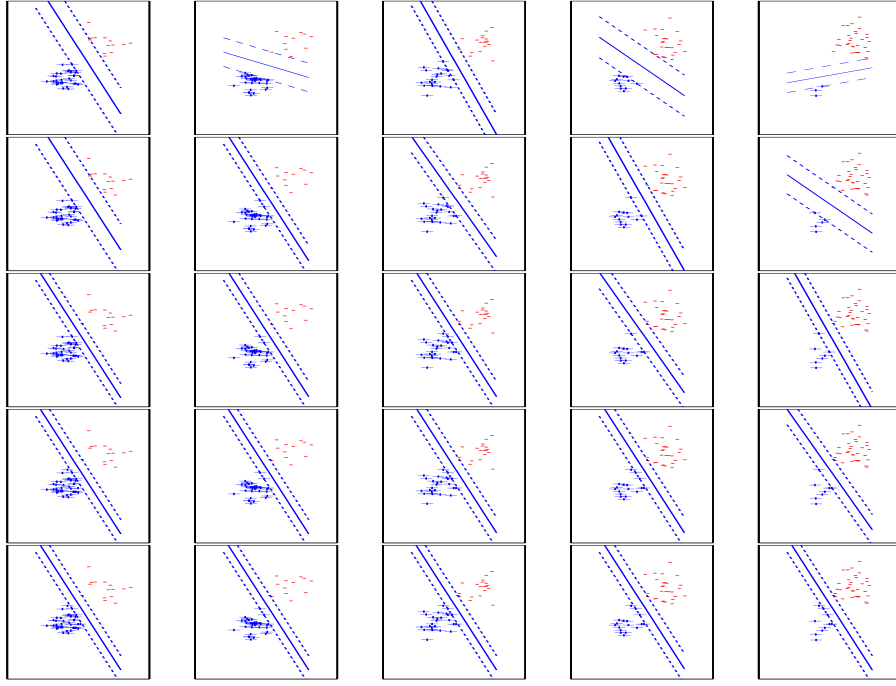
$$\begin{aligned} \alpha_1 &= [\alpha_1^a; \alpha_1^b], \\ \alpha_2 &= [\alpha_2^b; \alpha_2^c]. \end{aligned}$$

So, the newly formed problem for site  $l$  at iteration  $t$  has the Gram matrix

$$Q = \begin{bmatrix} Q_{aa} & Q_{ab} & Q_{ac} \\ Q_{ba} & Q_{bb} & Q_{bc} \\ Q_{ca} & Q_{cb} & Q_{cc} \end{bmatrix}$$

and we have

$$Q_1 = \begin{bmatrix} Q_{aa} & Q_{ab} \\ Q_{ba} & Q_{bb} \end{bmatrix}$$



**Fig. 1:** Demonstration of data distribution in DPSVM iterations. Distribution of training data in 5 sites before iterations begins (the first row) and in iteration 1 to 4 (the 2nd to the 5th rows). This figure shows how all local classifiers converge to the global optimum classifier.

and

$$Q_2 = \begin{bmatrix} Q_{bb} & Q_{bc} \\ Q_{cb} & Q_{cc} \end{bmatrix}$$

where  $Q_{aa}$ ,  $Q_{bb}$  and  $Q_{cc}$  are the Gram matrixes corresponding to  $\alpha_1^a$ ,  $\alpha_1^b$  (or  $\alpha_2^b$ ) and  $\alpha_2^c$ . So The newly formed problem can be written as

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

Note that  $[\alpha_1; 0; 0; \dots; 0]$  and  $[0; 0; \dots; \alpha_2]$  are both feasible solution of the above maximization problem with the objective value  $h^{t-1,l}$  and  $h^{t,m}$  respectively. Therefore, the optimum value  $h^{t,l}$  satisfies

$$h^{t,l} \geq \max\{h^{t-1,l}, h^{t,m}\}.$$

Now we remove the assumption that site  $l$  only receives SVs from site  $m$ . Since  $m$  is selected arbitrarily from  $l$ 's ancestor sites and adding more samples cannot decrease the optimal objective value (following the same argument in Lemma 1), we have that

$$h^{t,l} \geq \max\{h^{t-1,l}, \max_{m \in \text{UPS}_l} \{h^{t,m}\}\}.$$

¶

Lemma 2 shows that the objective value of the subproblem in our DPSVM algorithm monotonically increases in each iteration.

*Corollary 1:* The stopping criterion,  $h^{t,l} = h^{t-1,l}$  for all  $l$  and  $t > T$ , can be satisfied in finite number of iterations.

*Proof.* By Lemma 2, adding SVs from the adjacent site with higher objective value results in increase of the objective value of the newly formed problem. Lemma 1 shows that

every optimal objective value for each site at each iteration is bounded by  $h^*$ . Since the data points are limited and no duplicated data are allowed,  $h^{t,l} = h^{t-1,l}$  for all  $l$  and  $t > T$  can be satisfied in finite number of steps. ¶

Before proving our theorem, we have to prove a key proposition first.

*Proposition 1:* If any Gram matrix formed by any training sample data sets are positive definite, and  $h^{t,l} = h^{t-1,l} = h^{t-2,l} = \dots = h^{t-l,m_l}$  where  $l-1, \dots, l-m_l$  are the ancestor sites from which site  $l$  receives SVs at the current iteration, then the support vectors sets  $V^{t-1,l}$ ,  $V^{t,m}$ ,  $\forall m \in \text{UPS}_l$  and  $V^{t,l}$  are equal and the solution corresponding to the SVs from either of the above site is the optimal solution for the SVM training problem for the union of the training vectors of site  $l$  and its immediate ancestor  $m$ ,  $\forall m \in \text{UPS}_l$  at iteration  $t$ .

*Proof.* Note that at time  $t+1$ , for site  $l$ , it is the same to update its data set by simultaneously adding SVs from site  $m$ ,  $\forall m \in \text{UPS}_l$  as to update the data set by sequentially adding SVs from those ancestor sites. That is to say, we only need to show the result in the case that site  $l$  only receives SVs from one such site, say site  $l-1$ . Recall the notation used in the proof of Lemma:  $h^{t,l-1} = -\alpha_1^T Q_1 \alpha_1 + \mathbf{1}^T \alpha_1$ ,  $h^{t-1,l} = -\alpha_2^T Q_2 \alpha_2 + \mathbf{1}^T \alpha_2$ ,  $h^{t,l} = -\alpha^T Q \alpha + \mathbf{1}^T \alpha$ .  $\alpha_1 = [\alpha_1^a; \alpha_1^b]$ ,  $\alpha_2 = [\alpha_2^b; \alpha_2^c]$ .  $\alpha = [\alpha^a; \alpha^b; \alpha^c]$ , where  $\alpha_1$  is the nonzero part of  $\alpha^{t,l-1}$ ,  $\alpha_2 = \alpha^{t-1,l}$  and  $\alpha = \alpha^{t,l}$ . The newly formed problem for site  $l$  at iteration  $t$

has the Gram matrix  $Q = \begin{bmatrix} Q_{aa} & Q_{ab} & Q_{ac} \\ Q_{ba} & Q_{bb} & Q_{bc} \\ Q_{ca} & Q_{cb} & Q_{cc} \end{bmatrix}$  and we have

$Q_1 = \begin{bmatrix} Q_{aa} & Q_{ab} \\ Q_{ba} & Q_{bb} \end{bmatrix}$  and  $Q_2 = \begin{bmatrix} Q_{bb} & Q_{bc} \\ Q_{cb} & Q_{cc} \end{bmatrix}$ . Note that  $[\alpha_1; 0; \dots; 0]$  and  $[0; \dots; 0; \alpha_2]$  are both feasible for the global optimal problem (5) and the optimal solution of this problem is  $\alpha$ . Since,  $h^{t-1,l} = h^{t,l-1} = h^{t,l}$ , by uniqueness of optimal solution for strictly convex problem, we have  $\alpha^a = \alpha_1^a = 0$ ,  $\alpha^b = \alpha_1^b = \alpha_2^b$  and  $\alpha^c = \alpha_2^c = 0$ . Since  $\alpha^a$  is nonzero, we have

$$\{\alpha_1^a\} = \emptyset.$$

This already shows that the support vectors sets  $V^{t-1,l}$ ,  $V^{t,l-1}$  and  $V^{t,l}$  are identical. Then we prove that  $[0; \dots; 0; \alpha]$  is the optimal solution for the SVM problem with union of the training vector from site  $l$  and  $m$ . The Karush-Kuhn-Tucker (KKT) conditions for the problem (5) can be stated as follows:

$$\begin{aligned} Q_i \alpha &\geq 1 && \text{if } \alpha_i = 0 \\ Q_i \alpha &\leq 1 && \text{if } \alpha_i = \gamma \\ Q_i \alpha &= 1 && \text{if } 0 < \alpha_i < \gamma, \end{aligned}$$

where  $Q_i$  is the  $i$ -th row of the  $Q$  corresponding to  $\alpha_i$ . Since  $[0; 0; \dots; 0; \alpha_1]$ ,  $\alpha_2$  and  $\alpha$  satisfy the KKT condition of the problem  $P^{t,m}$ ,  $P^{t-1,l}$  and  $P^{t,l}$  where  $P^{t,l}$  denotes the SVM problem at iteration  $t$  for site  $l$ . By simple algebra,  $[0; \dots; 0; \alpha_b; 0; \dots; 0]$  satisfies the KKT condition of the SVM problem with union training vectors. Therefore, the  $[0; \dots; 0; \alpha_b; 0; \dots; 0]$  is the optimal solution for the SVM problem with union of the training vector from site  $l$  and  $m$ . By sequentially adding SVs from  $m$ ,  $\forall m \in \text{UPS}_l$ , we may repeat the above argument so that our proposition is true.  $\blacklozenge$

*Theorem 1:* Distributed SVM over a strongly connected network converges to the global optimal classifier in finite steps.

*Proof.* The Corollary 1 shows that the DPSVM converges in finite steps. Proposition 1 shows that when the DPSVM converges, the support vectors of each sites that are immediately adjacent are identical. Therefore, if site  $i$  can be assessed by site  $j$ , they have identical support vectors upon convergence. Since the network is strongly connected, all sites can be accessed by all other sites. Therefore, the support vectors of all sites over the network are identical. Let  $V^*$  denotes the converged support vector set. The solution defined by

$$\alpha_i^* = \begin{cases} \alpha^{t,l}, & \text{if } x_i \in V^* \\ 0, & \text{otherwise} \end{cases}$$

is always a feasible solution for the global SVM problem (5). By Proposition 1,  $V^*$  is also the support vector set for the union of the training samples from one site and its ancestor sites and the corresponding solution is optimal for the SVM with the union of the training samples from the those sites. As each site is accessible by all other sites by strongly connected networks,  $V^*$  is the support vector set for the union of the training samples from all sites. Therefore, the solution  $\alpha^*$  is global optimum.  $\blacklozenge$

#### IV. ALGORITHM IMPLEMENTATION

The proposed algorithm has an array of implementation options, including training parameter selection, network topology configuration, sequential/parallel and online/off-line implementation. Those implementation options have dramatic

impact on classification accuracy and performance in terms of training time and communication overhead.

##### A. Global Parameter Estimation

One may note that the algorithm requires that all sites use an identical training parameter  $\gamma$ , which balances training error and margin, such that the final converged solution is guaranteed to be the global optimum. In a centralized method, such parameters are usually tuned empirically through cross-validations. In distributed data mining, such tuning may not be feasible given partially available training data.

We, in this paper, propose an intuitive method to generate a parameter  $\gamma$  that has good performance to the global optimization problem by using locally available partial training data and statistics of the global training data. The training parameter  $\gamma$  may be estimated as follows.

$$\gamma = \gamma_l \frac{N\sigma_l^2}{N_l\sigma^2} \quad (6)$$

where  $\gamma_l$  is a good parameter selected based on data in site  $l$ , and  $\sigma_l$  and  $\sigma$  are defined as follows.

$$\sigma_l^2 = \frac{1}{N_l} \sum_{i \in l} K(x_i, x_i) - 2K(x_i, o_l) + K(o_l, o_l)$$

and

$$\sigma^2 = \frac{1}{N} \sum_{\forall i} K(x_i, x_i) - 2K(x_i, o) + K(o, o)$$

where  $o$  is the center of all training vectors and  $o_l$  is the center of the training vectors at site  $l$ .

Equation (6) can be justified qualitatively as follows. By formula (2), if  $\gamma$  is larger, the empirical error is weighted higher in the objective. That is equivalent to say, the corresponding prior is weaker for a larger  $\gamma$ . Some researchers choose  $N/\sigma^2$  to be a default value of  $\gamma$  [16]. Therefore, after determining a local parameter  $\gamma_l$  at site  $l$  by cross-validation or other methods, one may expect (6) to be a good parameter for the overall problem. Our experiment results confirm such expectation.

In the MNIST digit image classification discussed in the Section V, we classify digit 0 from digit 2. Suppose the 11881 training vectors are distributed into 15 sites. One may estimate parameter  $\gamma$  from the global optimization problem with a local optimal parameter  $\gamma_i$  at site  $i$  by (6). We use 1000 vectors as a validation set for parameter tuning and another 1000 vectors as test set. Both are sampled from MNIST test set. We first search for an local optimal  $\gamma_1$  at site 1 where there are 807 training vectors. Table I gives the test error over the validation set using different setting of  $\gamma$ . Therefore, the optimal local parameter  $\gamma_1 = 2^{-7}$ . Given the global statistics  $\sigma^2 = 109.49$  and the local statistics  $\sigma^2 = 110.36$ , we could estimate a  $\hat{\gamma}$  as

$$\hat{\gamma} = \gamma_l \frac{N\sigma_l^2}{N_l\sigma^2} = 2^{-7} \frac{11881 \times 110.36}{807 \times 109.49} = 0.117.$$

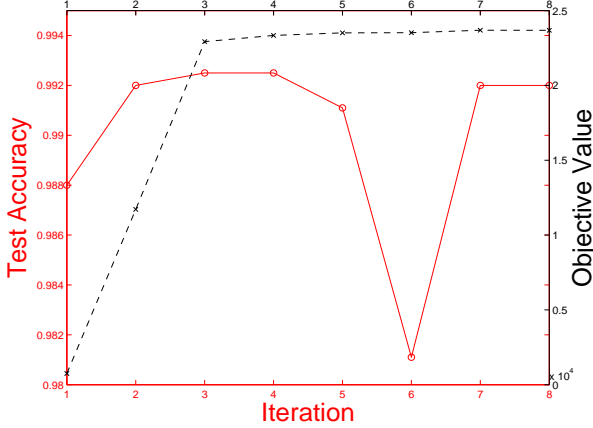
To justify the estimated parameter  $\hat{\gamma}$ , we enumerate several possible value of global parameter  $\gamma$ , train a model based on all training vectors and obtain the test error, summarized in Table II. One may observe the best  $\gamma^* = 2^{-3} = 0.125$ . Using

**TABLE I:** Validation Error at Site 1

$\gamma_1$	$2^{-9}$	$2^{-7}$	$2^{-5}$	$2^{-3}$	$2^{-1}$	2	$2^3$	$2^5$
Validation Error	0.9851	<b>0.9876</b>	0.9846	0.9811	0.9811	0.9811	0.9811	0.9811

**TABLE II:** Test Error for the Global Training Problem

$\gamma$	$2^{-9}$	$2^{-7}$	$2^{-5}$	$2^{-3}$	$2^{-1}$	2	$2^3$	$2^5$
Test Error	0.9911	0.9911	0.9911	<b>0.9920</b>	0.9906	0.9886	0.9886	0.9841

**Fig. 2:** Test Accuracy and Objective Value in Iterations

this error In this specific application,  $\hat{\gamma}$  is a good estimation of the true  $\gamma^*$ .

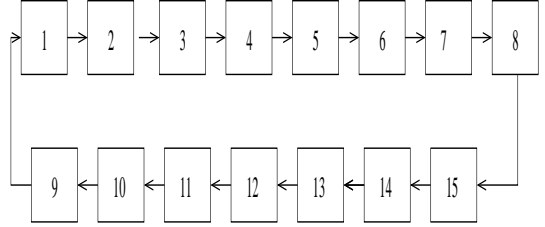
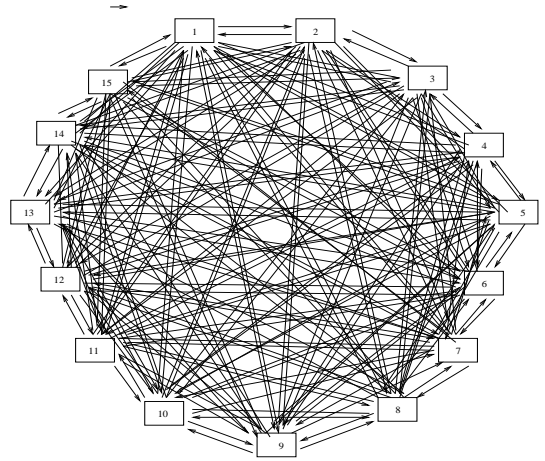
With the estimated  $\gamma^*$ , we plot the test accuracy and objective values of site 1 in each iteration of our DPSVM algorithm in Figure 2. The network we choose has 7 sites and a random sparse topology. One may observe that the objective value (of the primal problem) is monotonously increasing while the test accuracy is improving, which may not be monotonic. The power of DPSVM is to get global optimal classifier in a local site without transmitting all data into one site.

### B. Network Configuration

Network configuration has a dramatic impact on performance of our DPSVM algorithm. Size and topology of a network determine the number of sites that may work concurrently and the frequency a site receives new training data from other sites.

The size of network is restricted by the number of servers available and number of data center distributed. The larger a network is, the more server that may work parallel, however, the more communication overhead caused as well.

One may note that, upon convergence, each site must at least contain the whole set of support vectors, therefore, it is no longer very meaningful to have more than  $N/N_{SV}$  sites while  $N_{SV}$  denotes the number of support vectors for the global problems. Actually, Lu and Roychowdhury in 2006 showed that the number of training vector must be bounded below in order that a randomized distributed algorithm has a provably fast convergence rate [14]. Our experiments show

**Fig. 3:** Diagram: a ring network.**Fig. 4:** Diagram: a fully connected network.

that the training speed is in general faster in a larger size of a network given the network size is limited.

Network topology configuration is a key issue in implementing the distributed algorithms. Let us first consider two extreme cases: a ring network (see Figure 3 as an example) and a fully connected network (see Figure 4 as an example). A ring network is the sparsest strongly connected network while the fully connected network is the densest one. The denser a network is, the more frequently the information exchange occurs. Graf et al. in 2004 proposed a special network, namely cascade SVM, and demonstrated that the training speed in such network outperform other methods [13]. The cascade SVM is a special case of strongly connected network. Figure 5 gives a typical example.

We may define a connectivity matrix  $C$  such that

$$C(k, l) = \begin{cases} 1, & \text{if node } k \text{ has an edge pointing to node } l \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

Without losing generality, the connectivity matrix for a ring

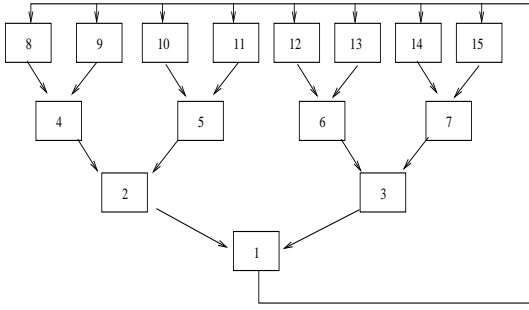


Fig. 5: Diagram: a cascade network.

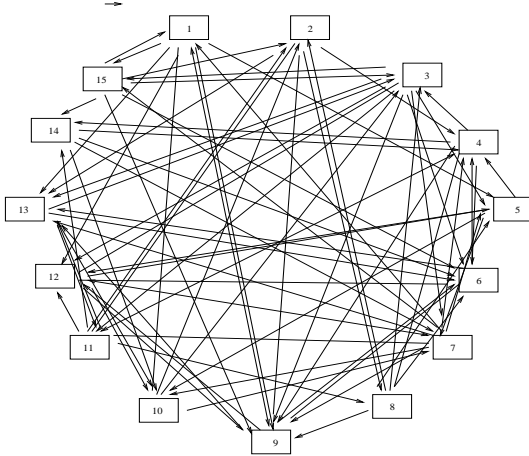


Fig. 6: Diagram: a random strongly connected network.

network has the following form

$$C_{rr}(k, l) = \begin{cases} 1, & \text{if } l = k + 1, k < L \text{ or } k = L, l = 1 \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

A fully connected network has the corresponding connectivity matrix

$$C_f(k, l) = \begin{cases} 0, & \text{if } k = l \\ 1, & \text{otherwise} \end{cases} \quad (9)$$

A typical cascade network in Figure 5 has the corresponding connectivity matrix

$$C_c(k, l) = \begin{cases} 1, & \text{if } l = 2^p, 2^{p+1} \leq k \leq 2^{p+1} + 1 \\ 1, & \text{if } l = 2^p + 2^q, \\ & 2^{p+1} + 2^{q+1} \leq k \leq 2^{p+1} + 2^{q+1} + 1, \\ & \forall q < p \\ 1 & \text{if } k = 1, 2^p \leq l \leq 2^{p+1} - 1, p = \log_2(N + 1) \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

we define a density of a directed network,  $d$  as follows

$$d = \frac{E}{L(L-1)} \quad (11)$$

where  $E$  is the number of directed edges and  $L$  is the number of nodes in the network. Since  $c(k, k) = 0, \forall k$  always holds in our network configurations, we have

$$\frac{1}{L-1} \leq d \leq 1. \quad (12)$$

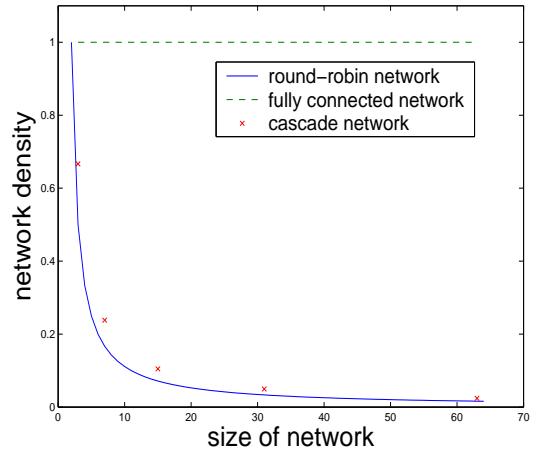


Fig. 7: Network density vs network size

The lower and upper bound of the density are achieved by the ring network and fully connected network respectively. That is,

$$d_{rr} = \frac{1}{L-1} \quad (13)$$

and

$$d_f = 1 \quad (14)$$

One may calculate the density of the cascade network as follows

$$d_c = \frac{2^p + 2^{p-1} - 2}{(2^p - 1)(2^p - 2)}, \forall p \geq 2. \quad (15)$$

We plot the network density against network size for the above three networks in Figure 7.

A random strongly connected network (RSCN) may have a topology shown in Figure 6. The network density of a RSCN may be anywhere between the solid line of ring network and dash line of the fully connected network in Fig. 7. Specifically, a RSCN in Fig. 6 has its connectivity matrix  $C_{ra} =$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (16)$$

This RSCN has 77 directed edges and 15 nodes. Its network density

$$d_{ra} = \frac{77}{210} = 0.37.$$

That said, it is denser than the cascade network of the same size. In our experiments, we try a variety of network

topologies. The results show that the cascade networks are not the "golden" density positions. A random network of the same size, such as the one in Fig. 6, may achieve better performance in terms of training speed.

### C. Synchronization

Other than network topology, timing of local computing also plays an important role on the effect of data accumulation and therefore the training speed. In this paper, we discuss two timing strategies: synchronous and asynchronous implementations.

One strategy is to process local data upon receiving support vectors from one site's all immediate ancestor sites. For example, in network defined in Figure 6, in second and later iteration, site 1 will start processing once having received data from site 7, 9 and 15, given it is currently idle. One may also observe this from the first column of (16). We call this strategy synchronous implementation as each server needs to wait for all of its immediately ancestor sites to finish processing.

Another strategy is called asynchronous implementation. In this strategy, each site processes its data upon receiving new data from any of its immediate ancestor sites. For example, in network defined in Figure 6, in second and later iteration, site 1 will start processing once having received data from either site 7, 9 or 15, given it is currently idle. Asynchronous implementation guarantee that all server are utilized as much as possible.

The advantage and disadvantage of both implementation strategies are empirically compared in Section V.

### D. Online Implementation

Since the DPSVM constructs a local problem by adding critical data, local SVM training problems in subsequent iterations are essentially problem adjustments which append variables and constraints based on the problem of the last iteration. Online algorithm is a popular solution for incremental learning problems. In this paper, we compare a simple online algorithm with an off-line algorithm.

In the simple online implementation, at each iteration, every site always uses current dual solution ( $\alpha$  values) as its initial  $\alpha$  values. The newly added training data will also bring a vector of non-zero  $\alpha$ 's. We use a slightly modified SVM<sup>light</sup> [15] as our local SVM solver, which may take an arbitrary  $\alpha$  vector as input. To avoid possible infinite loops due to numerical inaccuracies in the core QP-solver, a somewhat random working set is selected in the first iteration and we do it for every 100 iterations.

Our experiments in Section V show that this simple on-line implementation greatly improves the distributed training speed over the off-line implementation.

## V. PERFORMANCE STUDIES

We test our algorithm over a real-word data set: MNIST database of handwritten digits [17]. The digits have been size-normalized and centered in a fixed-size image. This database is a standard database online for researchers to compare their

training method because MNIST data is clean and doesn't needs any preprocessing. All digit images are centered in a 28x28 image. We vectorize each image to a 784 dimension vector. The problem is to classify digit 0 from digit 2, which involves 11811 training vectors. A linear kernel is applied in all experiments to simply the parameter tuning process.

We first introduce our terminology used in this section.

- $N$ : total number of training samples;
- $L$ : total number of distributed sites;
- $E$ : total number of directed edges in the network;
- $d$ : density of network;
- $N_{SV}$ : total number of global support vectors;
- $T$ : total number of DPSVM iterations;
- $\delta$ : total number of transmitted data per time;
- $\Delta$ : total number of transmitted vectors;
- $N_{l,t}$ : number of training samples in site  $l$  at iteration  $t$ ;
- $\max\{N_{l,t}\}$ : maximum number of training samples among all sites among all iterations;
- $e$ : elapsed CPU seconds of running DPSVM;
- $\varsigma$ : the standard deviation of initial training data distribution.

Throughout this section, the machines we use to solve local SVM problem are Pentium 4 3.2GHz with 1GB RAM.

### A. Effect of Initial Data Distribution

In real world applications, data may be distributed arbitrarily around the world. In certain applications, the distributed data may not be balanced or unbiased. For example, search engine companies usually need to classify Web pages for different markets, where the document data are usually stored locally due to both legal and technical constraints. Those data are highly unbalanced and biased. For example, China market may contain much less business pages than US market. In other applications, the data may be evenly distributed, intentionally or naturally. For example, Unites States Citizen and Immigration Services (USCIS) often distributes cases to their geographically distributed service centers to balance workloads.

To analyze the effect of initial data distribution, we randomly distribute the 11811 training data to a fully connected network with 15 sites. We do this several times with different settings of standard deviation of data distribution. When the data are distributed evenly, the standard deviation of the initial data distribution,  $\varsigma$  is approximately 0. When  $\varsigma$  is larger, the distributed more unbalanced. We run our DPSVM for  $\varsigma = 0, 305.7, 610.7$  respectively. We also test a special portional distribution where all data are initially distributed into 8 sites only and the rest 7 sites are left empty. This particular distribution has  $\varsigma = 766.9$ . In a four-layer binary cascade network, the data are initially distributed into the first layer under this distribution. The total communication cost  $\Delta$  and elapsed training time  $e$  against the initial data distribution are summarized in Table III. The results show that evenly distributed data may result in a fast convergence (less iteration and shorter cpu time) while the special portional distribution (8 sites of data and 7 empty sites) obviously achieves less communication overhead.

**TABLE III:** Performance over the initial data distribution

$\varsigma$	$T$	$\delta$	$\Delta$	$\max_{i,t}\{N_{it}\}$	$e$
Random Dense Network					
0	8	126	15191	1836	9.00
305.7	8	115	13897	2086	8.93
610.7	8	130	15363	2247	12.32
766.9*	6	102	12247	2097	16.92
Binary Cascade Network					
0	16	37	8842	1008	15.51
305.7	17	35	8977	1569	15.63
610.7	18	30	8120	1653	17.20
766.9*	18	26	6906	1731	16.43

\* : Training data are evenly distributed in 8 sites. The other 7 sites are empty before receiving any support vector from other sites.

**TABLE IV:** Tested Network Configurations

L	Topology	E	d
3	ring	3	0.50
3	binary-cascade	4	0.67
3	fully connected	6	1.00
7	ring	7	0.17
7	binary-cascade	10	0.24
7	random sparse	14	0.33
7	random dense	33	0.79
7	fully connected	42	1.00
15	ring	15	0.07
15	binary-cascade	22	0.10
15	random sparse	54	0.26
15	random dense	159	0.76
15	fully connected	210	1.00

### B. Scalability on Network Topologies

We test our algorithm over an array of network configurations. Five network topologies are tested including ring networks, binary cascade networks, fully connected networks, random sparse networks and random dense networks. For the two types of random networks, we impose a constraint over the network density such that

$$d \leq 0.33$$

for random sparse networks and

$$d \geq 0.75$$

for random dense networks. Networks of 3, 7 and 15 sites are tested for ring, binary cascade and fully connected networks. Networks of 7 and 15 sites were tested for the two types of random generated networks. The number of sites, number of edges and the corresponding network densities of all tested networks are summarized in Table IV.

In this section, we apply online and synchronized implementation for each tested networks. The off-line and asynchronous implementation will be discussed later. We record the total training time in terms of elapsed CPU-seconds, the number of iterations, and the number of transmitted training vectors per site. The results are plotted in Fig. 8. The results show that our algorithm scales very well for all network topologies except for the ring network, given the size of the network is limited. In ring networks, however, one site's information reach all other sites only after as least  $L - 1$  iterations. The communication is therefore not efficient and the convergence becomes slow. The convergence over a variety of network topologies can be

observed from the number of iterations in Fig. 8 (b). One may also observe that a random dense network may outperform binary cascade SVM in terms of training time when the network size is not too small. In a small network, a fully-connected network has its advantage over other topologies. Fig 8 (c) shows the fact that the communication cost per site is not flat when size of the network is increasing. Therefore, the total communication overhead increase faster than the network grows. The reason is due to synchronization. That is, each site has to wait and accumulate data from all of its ancestor sites in each iteration. This issue will be further discussed in Section V-D.

### C. Effect of Online Implementation

We implement a simple online algorithm described in Section IV-D. To show its effect, we plot the computing time in each iteration at site 1 in Fig. 9. One may observe that in off-line implementation, the computing speed heavily depends on the absolute number of SVs. On the other side, the online computing time is more sensitive to the change of the number of SVs. The initial sharp increase of CPU seconds is due to the sharp increase of number of local SVs. The later decreasing of CPU seconds is due to the advantage of online implementation.

### D. Effect of Server Synchronization

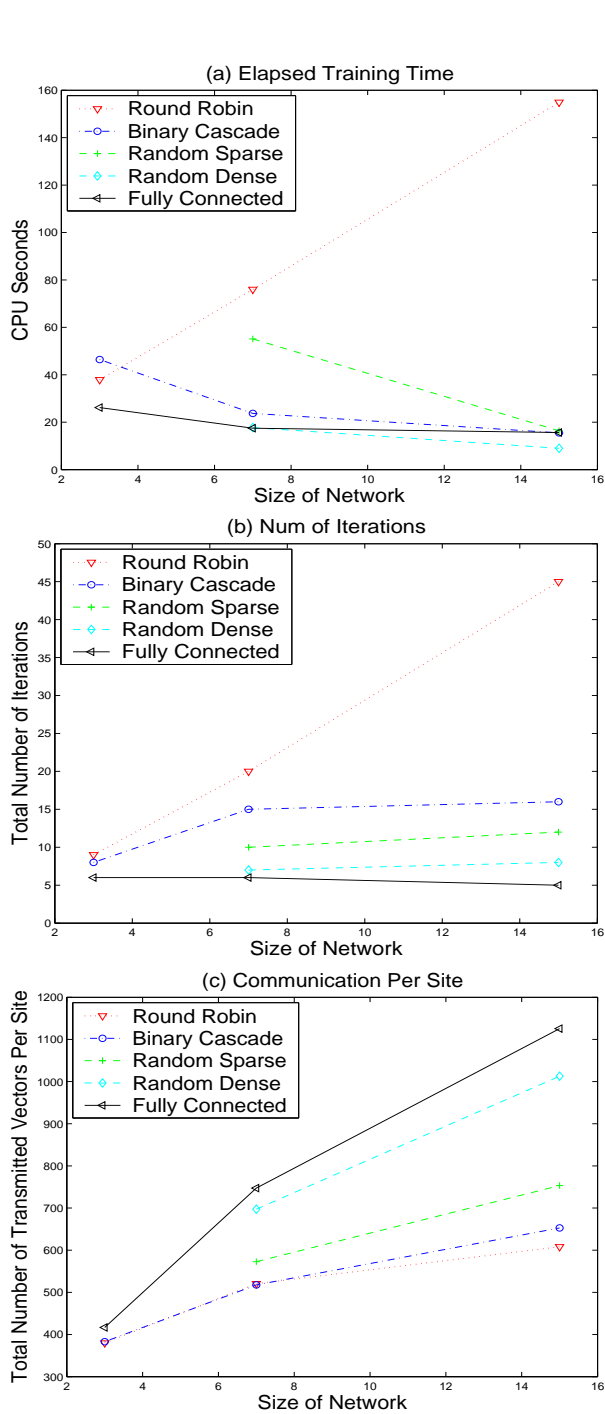
Recall that each site may start processing its local training vectors every time when it receives a batch of training data from one of its ancestor sites. We call it asynchronous implementation. Another implementation is to let each site wait until it receives new data from all of its ancestor sites. We call it synchronous implementation.

We implement the two strategies in a variety of networks. We compare their total training time and communication overhead for each of networks in Fig. 10 and Fig. 11 respectively. One may observe that, except for the cascade network, the synchronous implementation always dominates asynchronous one in terms of total training time. The reason that the difference of a synchronous and asynchronous is small in cascade networks is probably due to the fact that the topology of the cascade network already ensures certain synchronization.

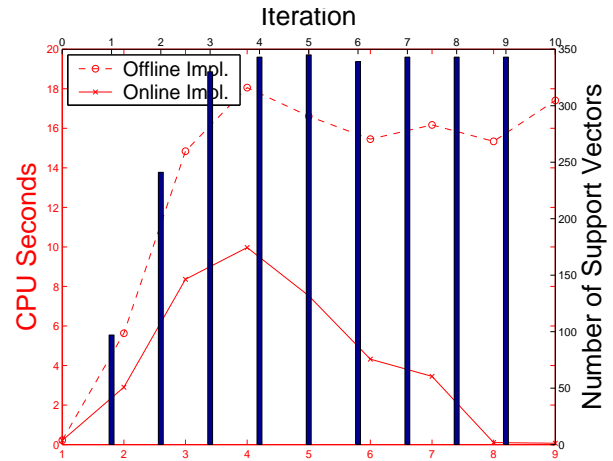
A more interesting result is in Fig. 11, where one may observe that the communication cost increases almost linearly with network size in asynchronous implementations and slower than synchronous implementations. The scenario can be explained as follows. In asynchronous implementation, since there are always sites that processes slower than other sites, the computing and communications are more frequent among those faster sites. The network that is effectively working is actually smaller since those slower sites contributes relatively less. As we know that a small network usually has longer processing time and less communication cost, the asynchronous implementation is similar as using a smaller network.

### E. Performance Comparison Between SVM and DPSVM

When merging all training data is infeasible due to physical or political reasons, one may apply the DPSVM that is able to

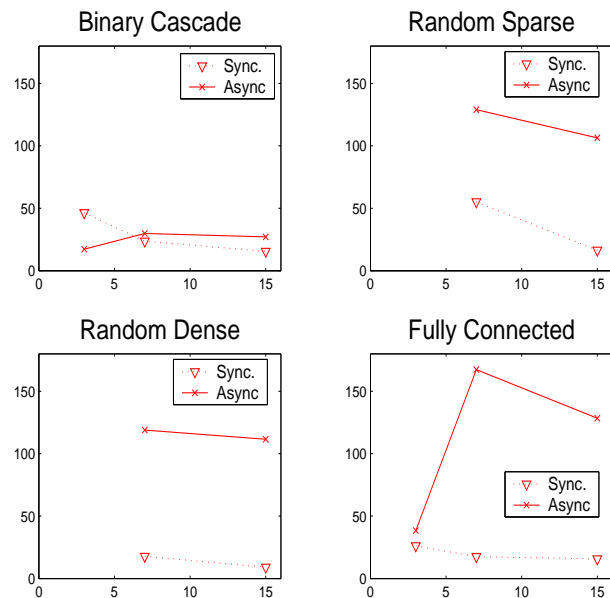


**Fig. 8:** Performance over Network Configurations. Five network topologies are compared. *Top.* The elapsed CPU seconds of training time over size of networks. *Middle.* The number of iterations over size of networks. *Bottom.* The total number of transmitted training vectors per site upon convergence over size of networks.



**Fig. 9:** Effect of the online implementation.

The number of support vectors and the computing time of site 1 per iteration DPSVM, implemented over a random sparse network with 15 sites. The solid line is the computing time per iteration for on-line implementation. The dashed line represents the computing time per iteration for off-line implementation. The bars represent the number of support vectors per iterations. The figures demonstrates that the computing time of off-line implementation also depends on the number of support vectors, while the online implementation does not.



**Fig. 10:** Training Time of Async. and Sync. Impl.

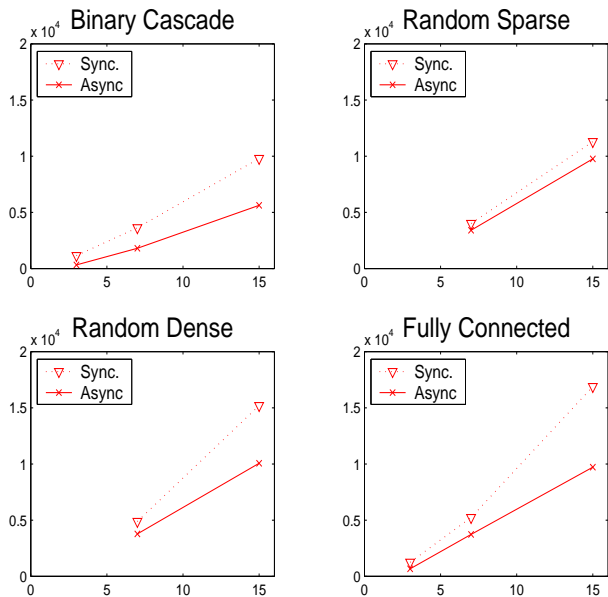


Fig. 11: Communication Overhead of Async. and Sync. Impl.

TABLE V: Tested Accuracy of DPSVM and Normal SVMs

L	DPSVM	Min (SVM)	Max (SVM)	Mean (SVM)
1	0.9920	0.9920	0.9920	0.9920
3	0.9920	0.9881	0.9916	0.9903
7	0.9920	0.9871	0.9911	0.9878
15	0.9920	0.9806	0.9891	0.9854

reach a global optimal solution or the normal support vector machine that works on local data only. We demonstrate that our DPSVM has the real advantage in terms of test accuracy over normal SVMs that are trained based on local training data.

We compare the DPSVM and SVM when MNIST data are randomly distributed into 3, 7 and 15 sites assuming that each site has the same number of training vectors. The training parameter is estimated following the approach introduced in Section IV-A. We compare the test accuracy of the DPSVM to the minimum, maximum and average value of the test accuracy of normal SVM among all sites. The results are summarized in Table V. The table shows expected phenomena: the less training are used, the worse the test accuracy is achieved. This is one of the motivations of the DPSVM: to reach global optimum and better test performance without aggregating all the data.

## VI. CONCLUSIONS AND FUTURE RESEARCH

The proposed distributed parallel support vector machine (DPSVM) training algorithm exploits a simple idea of partitioning training data and exchanging support vectors over a strongly connected network. The algorithm has been proved to converge to the global optimal classifier in finite steps. Experiments over a real-world database show that this algorithm is scalable and robust. The properties of this algorithm can be summarized as follows.

- The DPSVM algorithm is able to work on multiple arbitrarily partitioned working sets and achieve close to linear scalability if the size of network is not too large.
- Data accumulation during SVs exchange is limited if the overall SVs are limited. Communication cost is proportional to the number of SVs. Asynchronous implementation over a sparse network achieves the minimum data accumulation.
- The DPSVM algorithm is robust in terms of computing time and communication overhead to the initial distribution of the database. It is suitable for classification over arbitrary distributed databases as long as a network is denser than the ring network.
- In general, denser networks achieve less computing time while sparser networks achieve less data accumulation.
- On-line implementation is much faster. A fast on-line solver is critical for our DPSVM algorithm. This is also one of our future research directions.

## REFERENCES

- [1] Y. Zhu and X. R. Li, "Unified fusion rules for multisensor multihypothesis network decision systems," *IEEE Transactions on Neural Networks*, vol. 33, pp. 502–513, July 2003.
- [2] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer Verlag, 1995.
- [3] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.
- [4] H. Drucker, D. Wu, and V. N. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural Networks*, vol. 10, pp. 1048–1054, September 1999.
- [5] L. Cao and F. Tay, "Support vector machine with adaptive parameters in financial time series forecasting," *IEEE Transactions on Neural Networks*, vol. 14, pp. 1506–1518, November 2003.
- [6] S. Li, J. T. Kwok, I. W. Tsang, and Y. Wang, "Fusing images with different focuses using support vector machines," *IEEE Transactions on Neural Networks*, vol. 15, pp. 1555–1561, November 2004.
- [7] L. M. Adams and H. F. Jordan, "Is SOR color-blind?" *SIAM Journal on Scientific and Statistical Computing*, 1986.
- [8] U. Block, A. Frommer, and G. Mayer, "Block coloring schemes for the SOR method on local memory parallel computers," *parallel Computing*, 1990.
- [9] G. Zanghirati and L. Zanni, "A parallel solver for large quadratic programs in training support vector machines," *Parallel Computing*, vol. 29, pp. 535–551, November 2003.
- [10] N. Syed, H. Liu, and K. Sung, "Incremental learning with support vector machines," in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, San Diego, California, 1999.
- [11] C. Caragea, D. Caragea, and V. Honavar, "Learning support vector machine classifiers from distributed data sources," in *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI), Student Abstract and Poster Program*, Pittsburgh, Pennsylvania, 2005.
- [12] A. Navia-Vazquez, D. Gutierrez-Gonzalez, E. Parrado-Hernandez, and J. Navarro-Abellan, "Distributed support vector machines," *IEEE Transaction on Neural Networks*, vol. 17, pp. 1091–1097, 2006.
- [13] H. P. Graf, E. Cosatto, L. Bottou, I. Durdanovic, and V. Vapnik, "Parallel support vector machines: The cascade SVM," in *Proceedings of the Eighteenth Annual Conference on Neural Information Processing Systems (NIPS)*, Vancouver, Canada, 2004.
- [14] Y. Lu and V. Roychowdhury, "Parallel randomized support vector machine," *The 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2006)*, pp. 205–214, 2006.
- [15] T. Joachims, "Making large-scale SVM learning practical," *Advances in Kernel Methods - Support Vector Learning*, pp. 169–184, 1998.
- [16] —, "Svm-light support vector machine," 1998. [Online]. Available: <http://svmlight.joachims.org/>
- [17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.