Front. Electr. Electron. Eng. 2012, 7(2): 216–223
DOI 10.1007/s11460-011-0172-9

RESEARCH ARTICLE

Hua XU, Yun WEN, Jixiong WANG

A fast-convergence distributed support vector machine in small-scale strongly connected networks

Abstract In this paper, a fast-convergence distributed support vector machine (FDSVM) algorithm is proposed, aiming at efficiently solving the problem of distributed SVM training. Rather than exchanging information only among immediate neighbor sites, the proposed FDSVM employs a deterministic gossip protocol-based communication policy to accelerate diffusing information around the network, in which each site communicates with others in a flooding and iterative manner. This communication policy significantly reduces the total number of iterations, thus further speeding up the convergence of the algorithm. In addition, the proposed algorithm is proved to converge to the global optimum in finite steps over an arbitrary strongly connected network (SCN). Experiments on various benchmark data sets show that the proposed FDSVM consistently outperforms the related state-of-the-art approach for most networks, especially in the ring network, in terms of the total training time.

Keywords support vector machine, message passing interface, distributed computing, parallel computing, convergence, speedup

1 Introduction

As a powerful learning algorithm based on the statistical learning theory [1], support vector machine (SVM) recently receives increasing popularity in the community of data mining, for it delivers impressive generalization performance in a wide variety of machine learning problems, such as text categorization [2], face recognition [3], bioinformatics [4], and the like. However, the computation and storage requirements of SVMs increase rapidly with the number of training vectors, rendering many problems of practical interest out of their reach. Various approaches have been proposed over the last decade, aiming at improving the efficiency of SVM.

Traditional researches focus on accelerating the sequential algorithm of SVM [5–7]. One approach, namely, chunking, divides the training data into smaller subsets that are optimized iteratively until the global optimum is reached [6,7]. Sequential Minimal Optimization (SMO), which reduces the chunk size to two vectors, is the most popular algorithm of this kind [7]. Another optimization strategy, namely, shrinking, consists in eliminating non-support vectors earlier in the training process. The SVM^{light} proposed in Ref. [5] incorporates the shrinking technique with caching of the kernel data, achieving considerable savings in computation cost.

In attempts to obtain better efficiency, parallel computing techniques are adopted to exploit the parallelism in the sequential algorithm of SVM, which have been proved more suitable than chunking and shrinking approaches, especially on large-scale data sets [8,9]. A parallel implementation of SVM^{light} utilizes a novel technique, namely, variable projection method, to solve the partitioned subproblems [9]. Cao et al. in Ref. [8] identified the most costly operations in a modified SMO algorithm [10] and then parallelize these tasks to deliver a great speedup. There also exist some variations of the standard SVM that are better suited for parallelization [11,12]. Nevertheless, parallel implementations of SVM usually require a centralized access to the training data set, while the dependencies among operations within a single iteration impose limits on the potential functional parallelism.

More recently, a distributed computing framework is incorporated to break through the limits of current parallelization mechanisms. Since the support vectors (SVs) naturally represents the discriminative information of a data set, SVMs are inherently suited for a distributed framework where different sites can potentially exchange

Received April 28, 2011; accepted July 15, 2011

Hua XU (✉), Yun WEN, Jixiong WANG
State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
E-mail: xuhua@mail.tsinghua.edu.cn

only a small portion of training data, i.e., the local SVs. Syed et al. proposed the first distributed SVM algorithm that finds SVs locally and processes them altogether in a central processing site; the solution of which, however, is not globally optimal [13]. In Ref. [14], Graf et al. developed a cascade SVM that guarantees the global optimum, in which multiple processors for training SVM are organized as a binary cascade architecture. Wang and Jia improved upon the cascade SVM by designing a delicate "multi-trifurcate cascade" (MTC) architecture [15]. The MTC-SVM utilizes the multiplexing technique to process data in a pipelining manner, which is claimed to obtain fast feedback and high utilization ratio of computing nodes. Both the cascade SVM and the MTC-SVM, however, are constrained to a specific network topology. Lu et al. proposed a distributed parallel SVM (DPSVM) in a general network configuration, namely, strongly connected network (SCN), in which multiple nodes exchange local SVs with the immediate neighbors and together converge to a single global optimum in an iterative fashion [16]. A most recent work puts forward a distributed SVM in ad hoc sensor networks, in which a dynamic consensus algorithm is used to evaluate the global nonlinear classifier [17]. Their algorithm, nevertheless, gives only a sub-optimal solution. To the best of our knowledge, the DPSVM in Ref. [16] is the fastest DSVM that not only guarantees the global optimum but also puts few restrictions on the network topology configuration.

In this paper, we propose a fast-convergence distributed support vector machine (FDSVM) for distributed data classification in strongly connected networks, which is mainly motivated by the observation that the DPSVM experiences a severe degradation of performance under the environment of sparse networks, e.g., the ring network [16]. In general, the proposed FDSVM improves upon the original DPSVM to obtain faster convergence speed while retaining a globally optimal solution. The principle of the proposed algorithm consists in accelerating the information diffusion over the SCN by relaxing the communication constraints of exchanging the local SVs only between immediate neighbors. The updated communication policy allows each site to not only send its own local SVs but also forward newly received SVs until all the sites completely collect the total set of local SVs. Intuitively, this relaxation may increase the communication cost per iteration; however, the acceleration of information fusion will significantly reduce the number of iterations to converge, which in turn achieves substantial savings in the total computation cost. Empirical studies are carried out on various benchmark data sets with several network topology configurations. The experimental results indicate that the proposed approach outperforms the DPSVM for most networks, especially in the ring network, in terms of the total training time.

The remainder of the paper is organized as follows. The DPSVM in SCN is reviewed in the next section. Section 3 presents the algorithmic flow and implementation of the proposed FDSVM. The empirical study is given in Sect. 4, followed by some concluding remarks and suggestions for future research in Sect. 5.

2 Brief overview of DPSVM in SCN

2.1 Problem

Consider the following problem for distributed training SVM. The training data are assumed to be arbitrarily distributed among L sites within an SCN. An SCN is a directed network in which it is possible to reach any node starting from any other node by traversing the directed edges.

Suppose there are N_l training vectors in site l and N training vectors in all sites, where $\sum_{l=1}^{L} N_l = N$. Each training vector is denoted by z_i, $i = 1,2,...,N$, where $z_i \in \mathbb{R}^n$, and $y_i \in \{-1, +1\}$ is its label. As to a site l, its task is to seek a hyperplane to separate the positively and negatively labeled instances in the local training set N_l. The hyperplane is defined by $w^{\mathrm{T}}z + b = 0$, where $w \in \mathbb{R}^n$ is a vector orthogonal to the hyperplane, and $b \in \mathbb{R}$ is the bias. The decision function is thus defined as $H(x) = \mathrm{sgn}(w^{\mathrm{T}}z + b)$. To maximize the generalization ability of SVM, the algorithm seeks to maximize the margin of the classifier, which is further defined by the distance of the two parallel hyperplanes $w^{\mathrm{T}}z + b = 1$ and $w^{\mathrm{T}}z + b = -1$, i.e., $2/\|w\|^2$. For general linearly non-separable problems, a set of slack variables ξ_i is introduced so as to allow the misclassification of some training vectors. More details about the SVM classification problem are provided in Ref. [1].

2.2 Algorithm

For the convenience of description, some notations used throughout this paper are summarized in Table 1.

Specially, SVs are the training vectors that lie on the marginal boundary or are misclassified, which represent the discriminant information of the underlying

Table 1 Some notations used in the paper

symbol	definition
t	the number of an iteration in the computation process
l	the site number of a site in the target distributed system
L	total number of sites in the target distributed system
E	total number of directed links in the target distributed system
N	total training data set
N_{SV}	total SVs in the global training data set
$N^{l,t}$	local training set in site l at iteration t
IPS_l	the set of all the immediate precedent sites of site l
IDS_l	the set of all the immediate descendant sites of site l

classification problem. The basic idea of DPSVM lies in exchanging SVs among the immediate neighbor sites over an SCN and updating the local classifier for each site iteratively, until every site converges to a global optimal solution. Algorithm 1 presents the main steps of DPSVM.

Algorithm 1 Algorithmic flow of the original DPSVM

1. **Initialization**: The iteration number and local SV sets are reset, $t \leftarrow 0$, $SV^{l,t} \leftarrow \varnothing$, $l = 1,2,...,L$. The local training sets are initialized arbitrarily such that $\cup_{l=1}^{L} N^{l,0} = N$.
2. **repeat**
3. $t \leftarrow t+1$
4. **for** each site l, $l = 1,2,...,L$ **do**
5. Receive SVs from its immediate precedent sites IPS_l
6. Update the local training set, $N^{l,t} \leftarrow N^{l,t-1} \cup \{x_i : x_i \in SV^{m,t-1}, \forall m \in IPS_l\}$
7. Solve the problem on $N^{l,t}$, and record the optimal objective value $h^{l,t}$
8. Calculate the local SV set $SV^{l,t}$, and pass it to all sites in IDS_l
9. **end for**
10. **until** $h^{l,t} = h^{l,t-1}$ for all $l = 1,2,...,L$

Lu et al. in Ref. [16] first proved that the DPSVM satisfies both conditions of global lower bound and nondecreasing, which together ensure that the stopping criterion, $h^{l,t} = h^{l,t-1}$, $\forall l \in \{1,2,...,L\}$, can be met in finite number of iterations. Meanwhile, it is proved that the SV sets of the immediately adjacent sites are identical once upon the DPSVM converges. Given that each site in an SCN is accessible by every other site, $SV^{l,t}$ will converge to the SV set for the union of total training vectors from all the sites, N_{SV}, which guarantees that the final local solution of each site is globally optimal. For rigorous proofs of convergence to the global optimum, refer to Ref. [16].

2.3 Performance study

The DPSVM is tested in Ref. [16] with a broad range of parameter settings, including network configuration (size and topology), synchronization policy, online implementation option, and initial data distribution. The empirical study indicates that the DPSVM is robust within a wide variety of initial data distributions. In the meantime, the synchronous implementation always dominates the asynchronous one in terms of the total training time, while the online implementation is much faster than the offline one. On the other hand, the experimental results also reveal that network configuration has a dramatic impact on the performance of DPSVM. Taking into account both the size and topology of a network, a new metric, namely, the density of a network d, is defined as follows:

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

where E is the total number of the directed links, and L is the total number of sites in the distributed system. It is reported that the DPSVM scales well for most network topologies (including binary cascade, random sparse, random dense, and fully connected), with the exception of the ring network that is the sparsest SCN topological structure. In general, denser network topologies outperform sparser ones, while there is a remarkable decline of performance in terms of convergence speed over the ring network. The great degradation can be mostly attributed to the sharp increase in the total number of iterations, which is in turn determined by the low information exchange rate (in the ring network, the precedent site receives information from its descendant after at least $L-1$ iterations).

3 The proposed distributed SVM

3.1 Motivation

The algorithm proposed in this paper is mainly motivated by the observation that the DPSVM achieves state-of-the-art performance in network topologies with medium or high density while undergoing a severe degradation in the ring network that is representative of sparse networks. Thus, it is natural to consider how to obtain a comparative performance in the ring network compared with the denser network topologies.

First, let us take a closer look into the training time of DPSVM. In general, the total training time, T_{train} can be roughly divided into two main parts, T_{comp} and T_{comm}. The former stands for the computation cost of solving the local optimization problem in each site, while the latter stands for the communication cost of exchanging the local SVs over the SCN. The communication cost per iteration is proportional to the number of exchanged SVs. Given that the total number of SVs is limited compared with the total training data set, both the T_{comp} and T_{comm} are assumed to be roughly proportional to the total number of iterations. This assertion is in accordance with the experimental results plotted in Ref. [16], especially in both the ring network and the fully connected network. Therefore, the deficiency of DPSVM in the ring network could be mostly attributable to the sharp increase in the total number of iterations to convergence.

Why does DPSVM converge over the ring network in a much larger number of iterations? We will readily find that the information fusion in the ring network is much slower than that in the dense networks, e.g., the fully connected network. It takes at least $L-1$ iterations for a site in the ring network to receive information from its descendant site while only a single iteration in the fully connected network. Furthermore, it is the constraint of exchanging local SV sets only between neighbor sites that slows down the information fusion around the sparse networks, in which a site is connected to just a limited portion of the

whole sites in the distributed environment. By contrast, a site in the fully connected network is able to receive local SV sets from all the other sites in just a single iteration.

Thus, it is natural to guess that if the rate of information fusion over an arbitrary SCN equals that in the fully connected network, the DPSVM would converge in nearly the same number of iterations. This could be achieved by removing the constraints of exchanging information only between immediate neighbors, allowing each site to not only send its own local information but also forward the information received from neighbors iteratively, until a consistent state is reached. Since the strong connectivity of an SCN ensures that each site in the network can be accessed from any other site, the communication process under this updated policy will surely end up with every site's collecting total information from all the other sites in the network. This situation is consistent with what the DPSVM acts in the fully connected network. Given that the computation cost accounts for the main part of the total training time, although the new communication policy might increase the communication cost per iteration, the modified DPSVM would deliver the same comparative performance over an arbitrary SCN as compared with the fully connected network, for the total number of iterations would be remarkably reduced.

3.2 Algorithm flow

As stated above, the principle of the proposed FDSVM consists in the acceleration of fusing information across an arbitrary SCN, which improves upon the original DPSVM by relaxing the restrictions in communication within a single iteration. The new communication policy allows each site to send its own local SVs and forward the received SVs from its neighbors as well, until every site collects the total SVs from all the other sites.

In general, the proposed FDSVM works as follows. Each site within an SCN first trains the local SVM classifier via the local training data set and calculates the local SVs. After that, every site keeps sending the local SVs and forwarding the newly received SVs to its descendant sites. Meanwhile, each site keeps receiving the SVs from its precedent sites, until there is no newly arrived information. Then, the local training data set is updated by adding the total local SVs from all the other sites, and the local SVs are recalculated. The distributed computation process is repeated until the stopping criterion is satisfied, i.e., all the sites converge to the globally optimal solution. Algorithm 2 shows the main steps of the proposed FDSVM.

Algorithm 2 Algorithmic flow of the proposed FDSVM

1. **Initialization**: The iteration number and local SV sets are reset, $t \leftarrow 0$, $SV^{l,t} \leftarrow \varnothing$, $l = 1,2,...,L$. The local training sets are initialized arbitrarily such that $\cup_{l=1}^{L} N^{l,0} = N$.
2. **repeat**
3. $t \leftarrow t + 1$
4. **for** each site l, $l = 1,2,...,L$ **do**
5. Keep receiving SVs from its immediate precedent sites IPS_l, until there is no newly arrived information
6. Update the local training set, $N^{l,t} \leftarrow N^{l,t-1} \cup \{x_i : x_i \in SV^{m,t-1}, \forall m \in \{1,2,...,l-1,l+1,...,L\}\}$
7. Solve the problem on $N^{l,t}$, and record the optimal objective value $h^{l,t}$
8. Calculate the local SV set $SV^{l,t}$, and pass it to all sites in IDS_l
9. Forward the received SVs to all sites in IDS_l
10. **end for**
11. **until** $h^{l,t} = h^{l,t-1}$ for all $l = 1,2,...,L$

3.3 Communication strategy

Unlike the original DPSVM in which each site is restricted to exchange SVs only with its immediate neighbors via the directed links, the proposed FDSVM allows transmitting information between any two sites over an SCN. Intuitively, this relaxation will accelerate fusing information across the SCN; however, it will also increase the communication cost per iteration. Thus, how to diffuse information effectively and efficiently is the key to the success of the proposed FDSVM.

Since each site has a unique piece of initial information, the problem of information fusion in the proposed FDSVM can be formalized as a consensus problem (also named as virtual synchronization in some literature [18]), in which all the sites should finally agree on a certain state, i.e., the full set of SVs in the proposed FDSVM. Gossip protocol, also named epidemic protocol [19], has long been recognized as a satisfactory solution to information dissemination in distributed systems, which is appealing for its robustness, simplicity, and scalability [20–22]. In this paper, a deterministic gossip protocol-based communication strategy is deployed.

In the view of a single site, the communication process mainly consists in spreading information in a flooding and iterative manner. Specifically, each site begins with sending its own local SVs to the sites in IDS_l while receiving the local SVs from the sites in IPS_l. After then, it keeps forwarding the received SVs to IDS_l until there is no newly arrived SVs from IPS_l. In the end, every site will completely collect the local SVs from all the other sites in the distributed system, rather than only those from its immediate precedent sites in the original DPSVM. Theoretically, the complexity of the proposed communication strategy is bounded by di, where di refers to the diameter of the network. Furthermore, the diameter of a network is defined as the length of the shortest path (i.e., the number of links in the path) between the furthest pair of nodes in the network.

3.4 Proof of convergence

Proposition 1 *The proposed FDSVM in an arbitrary SCN acts the same as the original DPSVM in the fully connected network with the same sites.*

Proof Without loss of generality, we only consider the site l at iteration t. In the proposed FDSVM, the site l will not start the computation process until it collects the local SVs of the previous iteration from all the other sites in the SCN. That is, the local training data set of the site l at iteration t can be denoted as

$$N^{l,t} \leftarrow N^{l,t-1} \cup \{x_i : x_i \in SV^{m,t-1}, \forall m \in \{1,2,...,l-1,l+1,...,L\}\}.$$

On the other hand, considering the original DPSVM in the fully connected network with the same sites, the site l starts processing its data upon receiving the local SVs of the previous iteration from all the immediate precedent sites, i.e., the sites in IPS_l, which means that the local training set can be defined as

$$N^{l,t} \leftarrow N^{l,t-1} \cup \{x_i : x_i \in SV^{m,t-1}, \forall m \in IPS_l\}.$$

Given the full connectivity of the network, the site l has a two-way link to any other site, i.e., $IPS_l = \{1,2,...,l-1,l+1,...,L\}$, which indicates that every site in the proposed FDSVM has the same local training set at the same iteration as that in the original DPSVM. Since the result of SVM algorithm is only determined by the status of the training data set, it is safely concluded that the proposed FDSVM in an arbitrary SCN acts the same as the original DPSVM in the fully connected network with the same sites. □

Theorem 1 *The proposed FDSVM over an arbitrary SCN converges to the global optimal solution in finite steps.*

Proof Proposition 1 shows that the proposed FDSVM in an arbitrary SCN acts the same as the original DPSVM in the fully connected network with the same sites. Given that the original DPSVM over an SCN has been proved to converge to the global optimal solution in finite steps [16] while the fully connected network is clearly an SCN, we could conclude that the proposed FDSVM over an arbitrary SCN converges to the global optimal solution in finite steps. □

In addition, the proof above also reveals that the proposed FDSVM is expected to have similar performance with the original DPSVM in the fully connected network, which is to be validated in Sect. 4.

3.5 Implementation

Lu et al. in Ref. [16] have demonstrated that their proposed DPSVM has a suite of implementation options. These implementation options have dramatic impact on performance in terms of training time and communication overhead. Taking into consideration the recommendations made in Ref. [16], we choose the implementation options of our proposed FDSVM as follows.

Network configuration includes two main measures of a network, size, and topology. Limited by the number of severs available, the proposed FDSVM is implemented in a distributed system with only four sites. However, as pointed in Ref. [16], it is no longer very meaningful to have more than N/N_{SV} sites, where N_{SV} denotes the number of SVs for the global problems. In addition, with the number of the distributed sites increasing, the efficiency of parallel implementation declines, in which the metric of efficiency is defined as the obtained speedup divided by the number of sites available. Thus, the distributed system with four sites is thought to be a rational tradeoff between the cost and utilization of the computation resource. On the other hand, network topology is also a key issue in the distributed algorithm implementation. Three network topology configurations are considered in our proposed FDSVM, including a ring network, a random dense network, and a fully connected network, all of which are depicted in Figs. 1(a), 1(b), and 1(c), respectively.

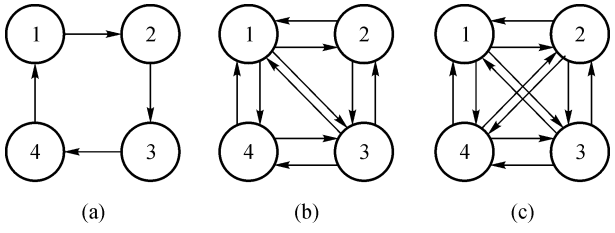

Fig. 1 Three network configurations used in implementation of the proposed FDSVM. (a) Ring network, d = 0.33; (b) random dense network, d = 0.83; (c) fully connected network, d = 1.0

Timing of local computation also plays an important role on the effect of data accumulation, furthermore, training speed. We adopt the synchronous strategy in the proposed FDSVM, in which each site will not start processing local data until it has completely collected the SVs from all the other sites. The synchronous strategy indicates that each site may need to wait for all the other sites to finish processing. To minimize the gap among the computation cost of all the sites, we uniformly distribute the total training data around all the sites in the initialization phase.

4 Experimental details

4.1 Test bed

To verify the advantages of our algorithm, we implement both the proposed FDSVM and the DPSVM in Java with MPJ Express[1)], which is the Java version of Message

1) MPJ Express Project. http://mpj-express.org/

Passing Interface (MPI). The LibSVM[VERSION 2.9] is adopted as our local SVM solver, in which a slack factor is introduced, while the penalty factor parameter is set to the default value, 10.[2] We test our proposed algorithm over the data sets provided by University of California, Irvine, at the website of LibSVM[3], including a1a, a2a, a3a, a4a, a5a, a6a, a7a, a8a, and a9a, which are derived from the famous UCI repository broadly cited in the field of machine learning[4] (the detailed characteristics of these benchmark data sets are shown in Table 2). Throughout this section, we use Intel Core 2 2.13 GHz processor with 1 GB RAM to solve local SVM problem while the distributed system is connected by 100 Mb/s LAN.

Table 2 Benchmark data sets used in comparison study

data set	#training vectors	#features	#classes
a1a	1605	123	2
a2a	2265	123	2
a3a	3185	123	2
a4a	4781	123	2
a5a	6414	123	2
a6a	11220	123	2
a7a	16100	123	2
a8a	22696	123	2
a9a	32561	123	2

4.2 Closer investigation of DPSVM

We first implement the original DPSVM and take a closer look into the composition of its running time. The DPSVM is run in the ring network using all the benchmark data sets listed in Table 2. The running time is plotted in Fig. 2, in which the total training time is split into three different parts, including computation cost, communication overhead, and cost of all the other operations (file manipulation, initialization, and termination of the parallel run-time environment, etc.).

The results indicate that the computation cost accounts for the main portion of the total training time, especially with the number of training vectors increasing. Taking the benchmark of a6a, a medium-scale data set, as an example, the computation cost amounts to 72.34% of the total running time, while the communication overhead takes up only a comparatively low percentage, i.e., 16.88%. Thus, we would preferably consider the optimization of computation cost, even at the expense of sacrificing the communication efficiency, which will more remarkably reduce the total training time.

On the other hand, the original DPSVM is implemented over the three different networks shown in Fig. 1 using all

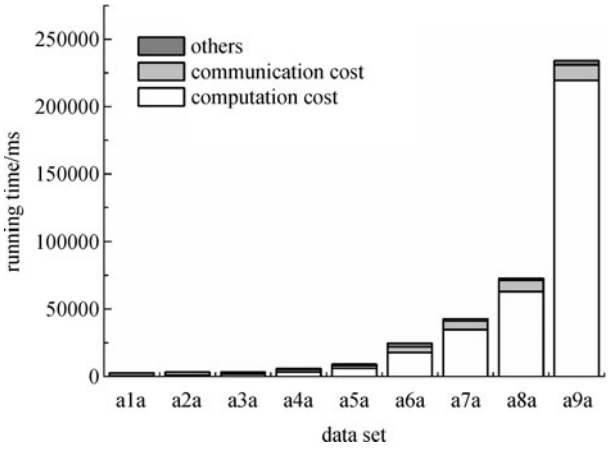

Fig. 2 Composition of original DPSVM's running time in ring network

the benchmark data sets. The results are shown in Fig. 3, which cohere with the conclusion made in Ref. [16] that the convergence of DPSVM slows down with decreasing network density.

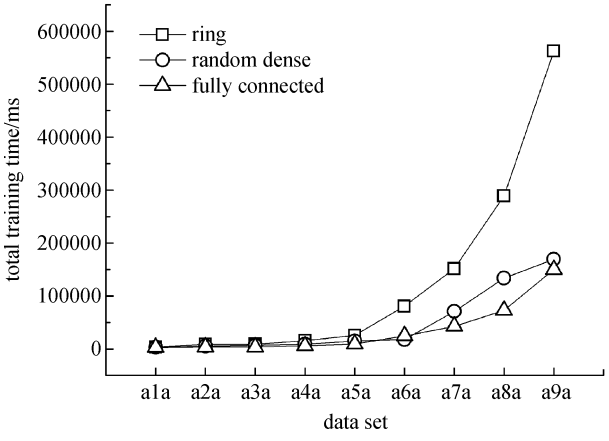

Fig. 3 Comparison of performance of original DPSVM over different networks

4.3 Performance comparison between DPSVM and FDSVM

In this section, the proposed FDSVM is compared with the DPSVM over the three different networks depicted in Fig. 1. It is noticeable that both algorithms do not stop until convergence, which indicates that the final results of both algorithms over each benchmark data set are identical. We first record the total training time in terms of elapsed CPU seconds. The results over three different network configurations are plotted in Figs. 4(a), 4(b), and 4(c),

2) Chang C C, Lin C J. LibSVM: A library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm/
3) LibSVM data: Classification (binary class). http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
4) Frank A, Asuncion A. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

respectively.

The proposed FDSVM consistently outperforms the DPSVM in the ring network and the random dense network, which is in accordance with our assertion made in Sect. 3.1 that the proposed algorithm would have advantages over the original DPSVM in terms of convergence speed. In the fully connected network, both algorithms obtain similar performance, which empirically validates the deduction made in Sect. 3.5 that the proposed FDSVM acts the same as the DPSVM over the fully connected network. To sum up, the gap between the performance of both algorithms becomes more remarkable with the density of the network decreasing.

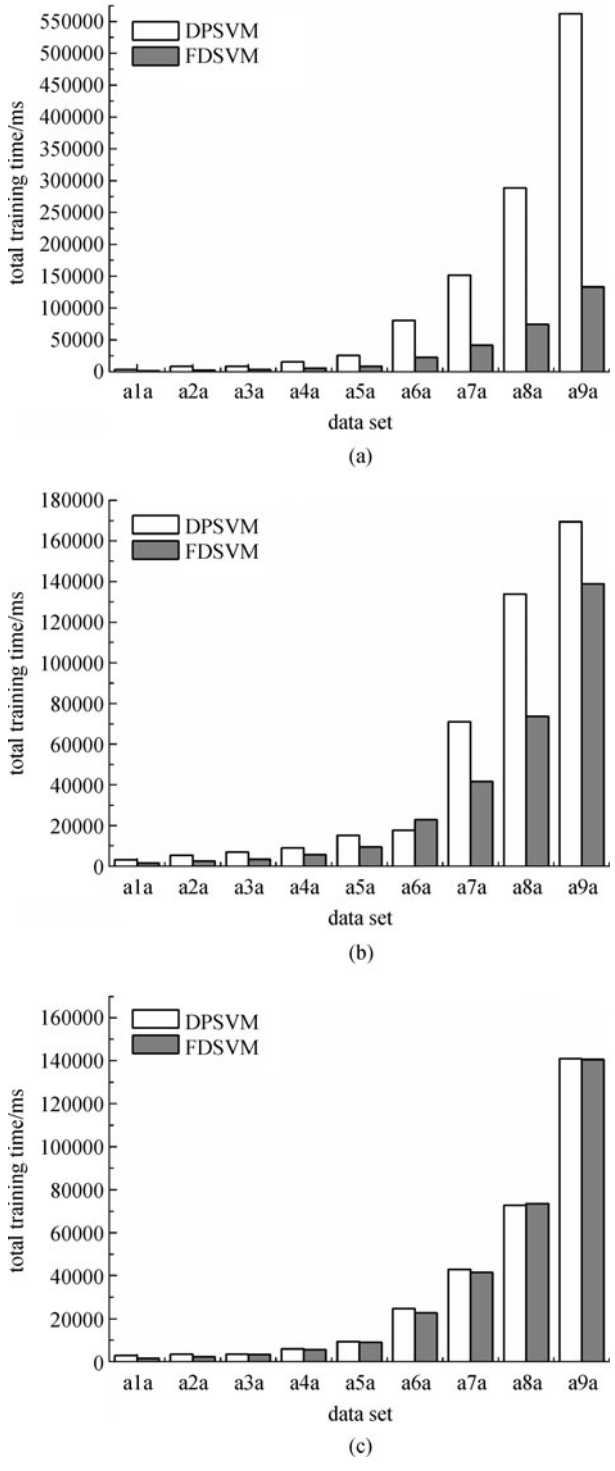

Fig. 4 Comparison of performance of the proposed FDSVM and DPSVM with three different network configurations. (a) Ring network; (b) random dense network; (c) fully connected network

Furthermore, we take a closer investigation into the improvement of performance, in which the running results are recorded in terms of the total number of iterations to convergence. The results are presented in Table 3, which explain the performance gap between the two algorithms well. The proposed FDSVM takes only two iterations to converge in all the three networks, the number of which is equivalent to that of the DPSVM in the fully connected network. This could attribute to the communication policy used in the FDSVM, which achieves the same rate of information fusion over an arbitrary SCN as that of the DPSVM in the fully connected network.

Table 3 Convergence speed of FDSVM and DPSVM in terms of total number of iterations

data set	ring network		random dense network		fully connected network	
	DPSVM	FDSVM	DPSVM	FDSVM	DPSVM	FDSVM
a1a	5	2	3	2	2	2
a2a	5	2	3	2	2	2
a3a	5	2	3	2	2	2
a4a	5	2	3	2	2	2
a5a	5	2	3	2	2	2
a6a	5	2	3	2	2	2
a7a	5	2	3	2	2	2
a8a	5	2	3	2	2	2
a9a	5	2	3	2	2	2

5 Conclusion and future study

In this paper, a distributed algorithm, namely, FDSVM, is proposed for distributed training SVM, which mainly improves upon the original DPSVM [16] in attempts to obtain faster convergence speed. The inspiration of the proposed FDSVM mostly comes from the observation that the DPSVM experiences a severe deficiency in the ring network due to a sharp increase in the number of iterations to converge. Rather than simply exchanging information among immediate neighbor sites, a deterministic gossip protocol-based communication policy is deployed to accelerate diffusing information around the network, aiming at reducing the number of iterations upon convergence. This improvement significantly decreases the computation cost, which accounts for the main portion of the total training time. Besides, the proposed FDSVM

over an arbitrary SCN is proved to converge to the global optimum in finite steps. Empirical study conducted on various benchmark data sets shows that the proposed FDSVM consistently outperforms the DPSVM for most networks, especially in sparse networks, e.g., the ring network, in terms of the total training time.

In addition to the topology of the network, the size of the network is also an influential factor to the algorithm performance. However, the size exceeding N/N_{SV} is proved to be meaningless [16], for the communication and waiting cost introduced will offset the gain of parallelization. Our further research is mainly concerned with the incorporation of both functional parallelism and data parallelism, which is expected to improve the scalability.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant Nos. 60875073 and 61175110), the National Key Technology R&D Program of China (Grant No. 2009BAG12A08), and the National S&T Major Projects of China (Grant Nos. 2009ZX02001 and 2011ZX02101).

References

1. Vapnik V N. The Nature of Statistical Learning Theory. 2nd ed. New York: Springer, 2000
2. Lee C H, Yang H C. Construction of supervised and unsupervised learning systems for multilingual text categorization. Expert Systems with Applications, 2009, 36(2, Part 1): 2400–2410
3. Kim S K, Park Y J, Toh K A, Lee S. SVM-based feature extraction for face recognition. Pattern Recognition, 2010, 43(8): 2871–2881
4. Yu X, Cao J, Cai Y, Shi T, Li Y. Predicting rRNA-, RNA-, and DNA-binding proteins from primary structure with support vector machines. Journal of Theoretical Biology, 2006, 240(2): 175–184
5. Joachims T. Making large-scale support vector machine learning practical. In: Schölkopf B, Burges C J C, Smola A J, eds. Advances in Kernel Methods: Support Vector Learning. Cambridge, USA: MIT Press, 1999, 169–184
6. Osuna E, Freund R, Girosi F. Training support vector machines: An application to face detection. In: Proceedings of 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 1997, 130–136
7. Platt J C. Fast training of support vector machines using sequential minimal optimization. In: Schölkopf B, Burges C J C, Smola A J, eds. Advances in Kernel Methods: Support Vector Learning. Cambridge, USA: MIT Press, 1999, 185–208
8. Cao L J, Keerthi S S, Ong C J, Uvaraj P, Fu X J, Lee H P. Developing parallel sequential minimal optimization for fast training support vector machine. Neurocomputing, 2006, 70(1–3): 93–104
9. Zanghirati G, Zanni L. A parallel solver for large quadratic programs in training support vector machines. Parallel Computing, 2003, 29 (4): 535–551
10. Keerthi S S, Shevade S K, Bhattacharyya C, Murthy K R K. Improvements to Platt's SMO algorithm for SVM classifier design. Neural Computation, 2001, 13(3): 637–649
11. Dong J X, Krzyzak A, Suen C Y. A fast parallel optimization for training support vector machine. In: Proceedings of the 3rd International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM'03). Berlin: Springer-Verlag, 2003, 96–105
12. Jayadeva, Khemchandani R, Chandra S. Fast and robust learning through fuzzy linear proximal support vector machines. Neurocomputing, 2004, 61: 401–411
13. Syed N A, Liu H, Sung K K. Handling concept drifts in incremental learning with support vector machines. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99). New York: ACM, 1999, 317–321
14. Graf H P, Cosatto E, Bottou L, Durdanovic I, Vapnik V. Parallel support vector machines: The cascade SVM. In: Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2005, 521–528
15. Wang L, Jia H. A parallel training algorithm of support vector machines based on the MTC architecture. In: Proceedings of 2008 IEEE Conference on Cybernetics and Intelligent Systems. 2008, 274–278
16. Lu Y, Roychowdhury V, Vandenberghe L. Distributed parallel support vector machines in strongly connected networks. IEEE Transactions on Neural Networks, 2008, 19(7): 1167–1178
17. Wang D, Zheng J, Zhou Y, Li J. A scalable support vector machine for distributed classification in ad hoc sensor networks. Neurocomputing, 2010, 74(1–3): 394–400
18. Tanenbaum A S, van Steen M. Distributed Systems: Principles and Paradigms. Pearson Prentice Hall, 2007
19. Voulgaris S, van Steen M. An epidemic protocol for managing routing tables in very large peer-to-peer networks. In: Brunner M, Keller A, eds. Self-Managing Distributed Systems. Berlin: Springer, 2004, 299–308
20. Chen J Y, Pandurangan G. Optimal gossip-based aggregate computation. In: Proceedings of the 22nd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'10). New York: ACM, 2010, 124–133
21. Liben-Nowell D. Gossip is synteny: Incomplete gossip and an exact algorithm for syntenic distance. In: Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'01). Philadelphia, USA: Society for Industrial and Applied Mathematics, 2001, 177–185
22. Shah D. Gossip algorithms. Foundations and Trends in Networking, 2009, 3(1): 1–125