Game-Theoretic Defense of Adversarial Distributed Support Vector Machines

RUI ZHANG
QUANYAN ZHU

With a large number of sensors and control units in networked systems, distributed support vector machines (DSVMs) play a fundamental role in scalable and efficient multi-sensor classification and prediction tasks. However, DSVMs are vulnerable to adversaries who can modify and generate data to deceive the system to misclassification and misprediction. This work aims to design defense strategies for DSVM learner against a potential adversary. We establish a game-theoretic framework to capture the conflicting interests between the DSVM learner and the attacker. The Nash equilibrium of the game allows predicting the outcome of learning algorithms in adversarial environments, and enhancing the resilience of the machine learning through dynamic distributed learning algorithms. We show that the DSVM learner is less vulnerable when he uses a balanced network with fewer nodes and higher degree. We also show that adding more training samples is an efficient defense strategy against an attacker. We present secure and resilient DSVM algorithms with verification method and rejection method, and show their resiliency against adversary with numerical experiments.

Manuscript received May 15, 2017; released for publication July 1, 2017.

Refereeing of this contribution was handled by Huimin Chen.

Authors' address: Department of Electrical and Computer Engineering, New York University, Brooklyn, NY, 11201 USA (E-mail: {rz885,qz494}@nyu.edu).

This research is partially supported by a DHS grant through Critical Infrastructure Resilience Institute (CIRI) and grants SES-1541164, and ECCS-1550000 from National Science Foundation (NSF).

A preliminary version of this work has been presented at the 18th International Conference on Information fusion, Washington, D.C., 2015 [34].

1557-6418/19/$17.00 © 2019 JAIF

1. INTRODUCTION

Support Vector Machines (SVMs) have been widely used in multi-sensor data fusion problems, such as motor fault detection [1], land cover classification [30], and gas prediction [39]. In these applications, a fusion center is required to collect data from each sensor and train the SVM classifier. However, the computations in the fusion center and its communications with sensors become costly when the size of data and network becomes large [11].

To solve the large-scale data fusion problems, several methods have been developed to speed up SVMs. For example, in [28], Tsang et al. have introduced an approximation method to scale up SVMs. In [10], Dong et al. have presented an efficient SVM algorithm using parallel optimization. These methods only speed up the computations in the fusion center, but the data transmissions between fusion center and sensors still require a significant amount of time and channel usages.

Efficiency is not the only drawback of the centralized SVM using fusion center. Sensors that collect sensitive or private information to design the classifier may not be willing to share their training data [13]. Moreover, a compromised fusion center attacked by an adversary may give erroneous information to all the sensors in the network. Furthermore, compromised sensors may also provide misleading information to the fusion center, and consequently affect uncompromised sensors [6].

Distributed support vector machines (DSVMs) draw attentions recently as it does not require a fusion center to process data collections and computations [13, 23, 29]. Each node in the network solves decentralized sub-problems themselves using their own data, and only a small amount of data is transferred between nodes, which makes DSVMs more efficient and private than the centralized counterpart.

However, DSVMs are also vulnerable. For example, misleading information can be spread to the whole network, and the large number of nodes and complex connections in a network makes it harder to detect and track the source of the incorrect information [5]. Moreover, even though we can find the compromised nodes, an adversary can attack other nodes and spread misleading information.

Thus, it is important to design secure and resilient distributed support vector machines algorithms against potential attacks from an adversary. In this paper, we focus on a consensus based DSVM algorithm where SVM problem is captured by a set of decentralized convex optimization sub-problems with consensus constraints on their decision variables [13, 34]. We aim to design defense strategies against potential attacks by analyzing the equilibrium of the game-theoretic model between a DSVM learner and an attacker.

In our previous work [34], we have built a game-theoretic framework to capture the conflict of interests between the DSVM learner and the attacker who can modify the training data. In the two-person nonzero-sum game, the learner aims to decentralize the computations over a network of nodes and minimize the error with an effort to minimize misclassification, while the attacker seeks to modify strategically the training data and maximize the error constrained by its computational capabilities.

The game formulation of the security problem enables a formal analysis of the impact of the DSVM algorithm in adversarial environments. The Nash equilibrium of the game enables the prediction of the outcome, and yields an optimal response strategy to the adversary behaviors. The game framework also provides a theoretic basis for developing dynamic learning algorithms that will enhance the security and the resilience of DSVMs.

In this paper, we propose several defense strategies for a DSVM learner against a potential attacker, and we show the effectiveness of our defense strategies using numerical experiments. The major contributions of this work are multi-fold.

Firstly, we capture the attacker's objective and constrained capabilities in a game-theoretic framework, and develop a nonzero-sum game to model the strategic interactions between an attacker and a learner with a set of nodes. We then fully characterize the Nash equilibrium by showing the strategic equivalence between the original nonzerosum game and a zero-sum game.

Secondly, we develop secure and resilient distributed algorithms based on alternating direction method of multipliers (ADMoM) [4]. Each node communicates with its neighboring nodes, and updates its decision strategically in response to adversarial environments. We present a summary of numerical results in [34].

Lastly, we present four defense strategies against potential attackers. The first defense strategy is to use balanced networks with fewer nodes and higher degrees. In the second defense strategy, we show that adding training samples to compromised nodes can reduce the vulnerability of the learning system. Adding samples to uncompromised nodes at the beginning of the training process also makes the learner less vulnerable. The third defense strategy is to use verification method where each node verifies its received data, and only accepts reasonable information from neighboring nodes to prevent misleading or illegitimate information sent to uncompromised nodes. The fourth defense strategy is to use rejection method where each node rejects unacceptable updates. Thus, not only misleading information is kept from affecting uncompromised nodes, but also wrong updates could be prevented in compromised nodes.

1.1. Related Works

Our work intersects the research areas on data fusion, machine learning, cyber security and machine learning. Machine learning tools have been used to tackle data fusion problems, e.g., [9, 16, 31]. However, machine learning systems can be insecure [2]. For example, in [17], Huang et al. have shown that SpamBayes and PCA-based network anomaly detection are vulnerable to causative attacks. In [3], Biggio et al. have shown that popular classification algorithms can be evaded even if the attacker has limited knowledge of learner's system.

With distributed machine learning tools developed for solving large-scale multi-sensor data fusion problems, each sensor solves sub-problems themselves and transmits information with neighboring sensors [24]. However, cyber security becomes another problem as an attacker may launch malicious cyber attacks to the data fusion networks [7]. Thus, it is important for the machine learning learner to analyze the equilibrium of the game with an adversary and design defense strategies against potential attacks.

Game theory is a natural tool to address this problem. It has been used in the study of the security of machine learning. For example, in [21], Liu et al. have modeled the interaction between a learner and an attacker as a two-person sequential noncooperative Stackelberg game. In [19], Kantarcioglu et al. have used game theory to analyze the equilibrium behavior of adversarial learning.

Game theory has also been used widely in cyber security as it provides mathematical tools for modeling situations of conflicts and predicting the behaviors of the attacker and defender in network security [22, 38, 40–43]. For example, in [26], Shen et al. have built an adaptive Markov game model to infer possible cyber attack patterns. In [18], Jiang et al. have presented an attack prediction and optimal active defense method using a stochastic game.

With game theory, we are able to analyze the game between a distributed machine learning learner with an adversary in a network, and further design defense strategies for the learner against the attacker. In our work, we focus on a class of consensus-based distributed support vector machines algorithms [13]. We assume that the attacker has the ability to modify training data to achieve his objectives.

In our previous works [33–37], we have built a game-theoretic model to capture the conflicts between a DSVM learner and an adversary who can modify training data or labels, and we have solved the game-theoretic problem with ADMoM [4]. In this work, we further analyze the equilibrium behaviors, and design defense strategies for DSVMs against potential attacks. We use numerical experiments to verify the effectiveness of our strategies.

1.2. Organization of the Paper

The rest of this paper is organized as follows. Section 2 outlines the consensus-based distributed support vector machines. In Section 3, we establish game-theoretic models for the learner and the attacker. Section 4 deals with the distributed and dynamic algorithms for the learner and the attacker. Section 5 summarizes our previous numerical experiments. Section 6 presents four different defense strategies and their corresponding numerical experiments. Section 7 provides conclusion remarks.

1.3. Summary of Notations

Notations in this paper are summarized as follows. Boldface letters are used for matrices (column vectors); $(\cdot)^T$ denotes matrix and vector transposition; $(\cdot)^{(t)}$ denotes values at step t; $[\cdot]_{vu}$ denotes the vuth entry of a matrix; $\mathrm{diag}(\mathbf{X})$ is the diagonal matrix with $\mathbf{X}$ on its main diagonal; $\|\cdot\|$ is the norm of the matrix or vector; $\mathcal{V}$ denotes the set of nodes in a network; $\mathcal{B}_v$ denotes the set of neighboring nodes of node v; $\mathcal{U}$ denotes the action set used by the attacker.

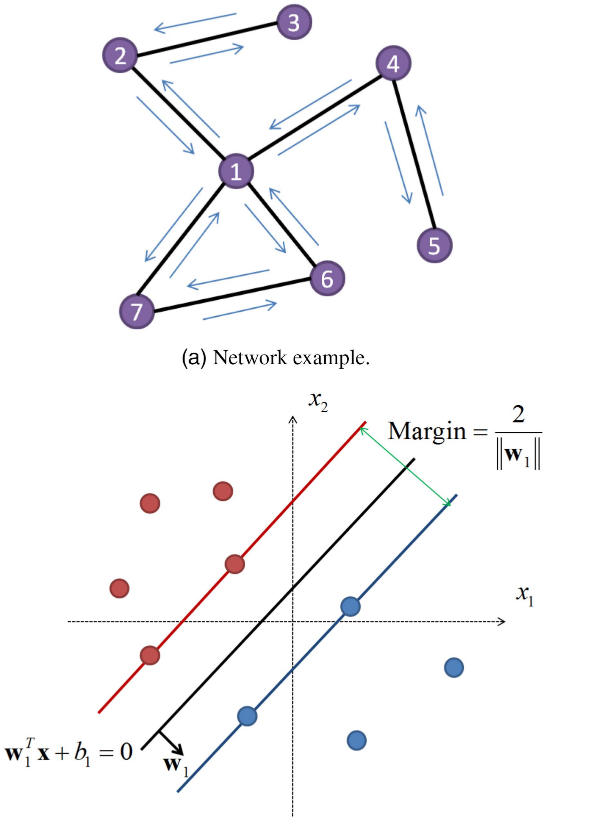
2. DISTRIBUTED SUPPORT VECTOR MACHINES

In this section, we present a distributed support vector machines learner in the network modeled by an undirected graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ with $\mathcal{V} := \{1,\ldots,V\}$ representing the set of nodes, and $\mathcal{E}$ representing the set of links between nodes. Node $v \in \mathcal{V}$ communicates only with his neighboring nodes $\mathcal{B}_v \subseteq \mathcal{V}$. Note that without loss of generality, graph $\mathcal{G}$ is assumed to be connected; in other words, any two nodes in graph $\mathcal{G}$ are connected by a path. However, nodes in $\mathcal{G}$ do not have to be fully connected, which means that nodes are not required to directly connect to all the other nodes in the network. The network can contain cycles. At every node $v \in \mathcal{V}$, a labelled training set $\mathcal{D}_v := \{(\mathbf{x}_{vn}, y_{vn}) : n = 1,\ldots,N_v\}$ of size N_v is available, where $\mathbf{x}_{vn} \in \mathbb{R}^p$ represents a p-dimensional pattern, and they are divided into two groups with labels $y_{vn} \in \{+1,-1\}$. Examples of a network of distributed nodes are illustrated in Fig. 1(a).

The goal of the learner is to design DSVM algorithms for each node in the network based on its local training data $\mathcal{D}_v$, so that each node has the ability to give new input $\mathbf{x}$ a label of $+1$ or -1 without communicating $\mathcal{D}_v$ to other nodes $v' \neq v$. To achieve this, the learner aims to find local maximum-margin linear discriminant functions $g_v(\mathbf{x}) = \mathbf{x}^T\mathbf{w}_v^* + b_v^*$ at every node $v \in \mathcal{V}$ with the consensus constraints $\mathbf{w}_1^* = \mathbf{w}_2^* = \cdots = \mathbf{w}_V^*$, $b_1^* = b_2^* = \cdots = b_V^*$, forcing all the local variables $\{\mathbf{w}_v^*, b_v^*\}$ to agree across neighboring nodes. Variables $\mathbf{w}_v^*$ and b_v^* of the local discriminant functions $g_v(\mathbf{x})$ can be obtained by solving the following convex optimization problem [13]:

$$\min_{\{\mathbf{w}_v,b_v,\{\xi_{vn}\}\}} \frac{1}{2}\sum_{v\in\mathcal{V}}\|\mathbf{w}_v\|_2^2 + VC_l\sum_{v\in\mathcal{V}}\sum_{n=1}^{N_v}\xi_{vn}$$

$$\begin{array}{lll} & \mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn} + b_v) \geq 1 - \xi_{vn}, & \forall v \in \mathcal{V},\ n = 1,\ldots,N_v; \\ \text{s.t.} & \xi_{vn} \geq 0, & \forall v \in \mathcal{V},\ n = 1,\ldots,N_v; \\ & \mathbf{w}_v = \mathbf{w}_u,\ b_v = b_u, & \forall v \in \mathcal{V},\ u \in \mathcal{B}_v. \end{array} \tag{1}$$


3
2
4
1
5
6
7
x_2
x_1
Margin $= \frac{2}{\|\mathbf{w}_1\|}$
$\mathbf{w}_1^T\mathbf{x} + b_1 = 0$
$\mathbf{w}_1$

(a) Network example.

(b) SVMs at node 1.

Fig. 1. Network example. (a) There are 7 nodes in this network. (b) Each node contains a labelled training set $\mathcal{D}_v := \{(\mathbf{x}_{vn}, y_{vn}) : n = 1,\ldots,N_v\}$. Each node can communicate with its neighbors. In each node, the learner aims to find the best linear discriminant line (Black solid line).

In the above problem, slack variables ξ_{vn} account for non-linearly separable training sets. C_l is a tunable positive scalar for the learner.

To solve Problem (1), we first define $\mathbf{r}_v := [\mathbf{w}_v^T, b_v]^T$, the augmented matrix $\mathbf{X}_v := [(\mathbf{x}_{v1},\ldots,\mathbf{x}_{vN_v})^T, \mathbf{1}_v]$, the diagonal label matrix $\mathbf{Y}_v := \mathrm{diag}([y_{v1},\ldots,y_{vN_v}])$, and the vector of slack variables $\xi_v := [\xi_{v1},\ldots,\xi_{vN_v}]^T$. With these definitions, it follows readily that $\mathbf{w}_v = (\mathbf{I}_{p+1} - \Pi_{p+1})\mathbf{r}_v$, where Π_{p+1} is a $(p+1)\times(p+1)$ matrix with zeros everywhere except for the $(p+1,p+1)$st entry, given by $[\Pi_{p+1}]_{(p+1)(p+1)} = 1$. Thus, Problem (1) can be rewritten as

$$\min_{\{\mathbf{r}_v,\xi_v,\omega_{vu}\}} \frac{1}{2}\sum_{v\in\mathcal{V}}\mathbf{r}_v^T(\mathbf{I}_{p+1} - \Pi_{p+1})\mathbf{r}_v + VC_l\sum_{v\in\mathcal{V}}\mathbf{1}_v^T\xi_v$$

$$\begin{array}{llll} & \mathbf{Y}_v\mathbf{X}_v\mathbf{r}_v \geq \mathbf{1}_v - \boldsymbol{\xi}_v, & \forall v \in \mathcal{V}; & \text{(2a)} \\ \text{s.t.} & \boldsymbol{\xi}_v \geq \mathbf{0}_v, & \forall v \in \mathcal{V}; & \text{(2b)} \\ & \mathbf{r}_v = \omega_{vu},\ \omega_{vu} = \mathbf{r}_u, & \forall v \in \mathcal{V},\ \forall u \in \mathcal{B}_v. & \text{(2c)} \end{array} \tag{2}$$

Note that ω_{vu} is used to decompose the decision variable $\mathbf{r}_v$ to its neighbors $\mathbf{r}_u$, where $u \in \mathcal{B}_v$. Problem (2) is a min-problem with matrix form coming from Problem (1).

With alternating direction method of multipliers [4], Problem (2) can be solved distributedly in the following lemma [13],

LEMMA 1 *With arbitrary initialization* $\mathbf{r}_v^{(0)}$, $\lambda_v^{(0)}$, $\omega_{vu}^{(0)}$ *and* $\alpha_v^{(0)} = \mathbf{0}_{(p+1)\times 1}$, *the iterations per node are given by:*

$$\lambda_v^{(t+1)} \in \arg\max_{\mathbf{0}\leq\lambda_v\leq VC_l\mathbf{1}_v} -\tfrac{1}{2}\lambda_v^T\mathbf{Y}_v\mathbf{X}_v\mathbf{U}_v^{-1}\mathbf{X}_v^T\mathbf{Y}_v\lambda_v + (\mathbf{1}_v + \mathbf{Y}_v\mathbf{X}_v\mathbf{U}_v^{-1}\mathbf{f}_v^{(t)})^T\lambda_v, \tag{3}$$

$$\mathbf{r}_v^{(t+1)} = \mathbf{U}_v^{-1}(\mathbf{X}_v^T\mathbf{Y}_v\lambda_v^{(t+1)} - \mathbf{f}_v^{(t)}), \tag{4}$$

$$\omega_{vu}^{(t+1)} = \tfrac{1}{2}(\mathbf{r}_v^{(t+1)} + \mathbf{r}_u^{(t+1)}), \tag{5}$$

$$\alpha_v^{(t+1)} = \alpha_v^{(t)} + \frac{\eta}{2}\sum_{u\in\mathcal{B}_v}[\mathbf{r}_v^{(t+1)} - \mathbf{r}_u^{(t+1)}], \tag{6}$$

where $\mathbf{U}_v = (\mathbf{I}_{p+1} - \Pi_{p+1}) + 2\eta|\mathcal{B}_v|\mathbf{I}_{p+1}$, $\mathbf{f}_v^{(t)} = 2\alpha_v^{(t)} - 2\eta \times \sum_{u\in\mathcal{B}_v}\omega_{vu}^{(t)}$.

The proof of Lemma 1 can be found in [13]. Iteration (3) is a quadratic programming problem. λ_v are the Lagrange multipliers per node corresponding to constraint (2a). Iteration (4) computes the decision variables $\mathbf{r}_v$, note that the inverse of $\mathbf{U}_v$ always exists and easy to solve. Iteration (5) yields the consensus variables ω_{vu}. Iteration (6) computes α_v, e.g., the Lagrange multipliers corresponding to the consensus constraint (2c). Iterations (3)–(6) are summarized into Algorithm 1. Note that at any given iteration t of the algorithm, each node $v \in \mathcal{V}$ computes its own local discriminant function $g_v^{(t)}(\mathbf{x})$ for any vector $\mathbf{x}$ as

$$g_v^{(t)}(\mathbf{x}) = [\mathbf{x}^T, 1]\mathbf{r}_v^{(t)}. \tag{7}$$

ALGORITHM 1: *ADMoM-DSVM*

```
Randomly initialize $\mathbf{r}_v^{(0)}$, $\lambda_v^{(0)}$, $\omega_{vu}^{(0)}$ and $\alpha_v^{(0)} = \mathbf{0}_{(p+1)\times 1}$.
for $t = 0, 1, 2, \ldots$ do
    for all $v \in \mathcal{V}$ do
        Compute $\lambda_v^{(t+1)}$ via (3).
        Compute $\mathbf{r}_v^{(t+1)}$ via (4).
    end for
    for all $v \in \mathcal{V}$ do
        Broadcast $\mathbf{r}_v^{(t+1)}$ to all neighbors $u \in \mathcal{B}_v$.
    end for
    for all $v \in \mathcal{V}$ do
        Compute $\omega_{vu}^{(t+1)}$ via (5).
        Compute $\alpha_v^{(t+1)}$ via (6).
    end for
end for
```

Algorithm 1 solves the DSVM problem using ADMoM technique. It is a fully decentralized network operation, and it does not require exchanging training data or the value of decision functions, which meets the reduced communication overhead and privacy preservation requirements at the same time. However, information transmitted in the network not only helps improve the performance of each node, but also increases the damages from the attacker, as the misleading information can be spread to every node. To design a secure and resilient DSVM algorithm, we first build the attack model to capture the attacker's intentions of breaking the training process of the learner.

3. DISTRIBUTED SUPPORT VECTOR MACHINES WITH ADVERSARY

In this section, we present the game-theoretic framework of a DSVM learner and an attacker who takes over a set of nodes with the aim of breaking the training process of the learner. We assume that the attacker has a complete knowledge of the learner's Problem (1) by Kerckhoffs's principle: the enemy knows the system [25], which enables us to anticipate the interactions of the learner and the attacker in a worst-case scenario. Moreover, the attacker can easily acquire the complete knowledge of the learning systems in reality, for example, by node capture attacks [27] and computer worms [8], an attacker can compromise the whole network through connections between neighboring nodes, and thus obtain the private and sensitive information of the learner.

To achieve the malicious goal, the attacker takes over a set of nodes $\mathcal{V}_a := \{1, \ldots, V_a\}$ and changes $\mathbf{x}_{vn}$ into

$$\hat{\mathbf{x}}_{vn} = \mathbf{x}_{vn} - \delta_{vn},$$

where $\delta_{vn} \in \mathcal{U}_v$, and $\mathcal{U}_v$ is the attacker's action set at node v. Note that we use $\mathcal{V}_l = \{1, \ldots, V_l\}$ to represent nodes without the attacker. $V = V_a + V_l$ and $\mathcal{V} = \mathcal{V}_l \cup \mathcal{V}_a$. A node in the network is either under attack or not under attack. An example of the impact of the attacker on the learner is shown in Fig. 2. This type of attacks represents a challenge for the learner. On the one hand, the learner will find the incorrect classifiers at the compromised nodes, and communications in the network may lead to unanticipated results as misleading information from compromised nodes can be spread to and then used by uncompromised nodes. On the other hand, it is difficult for the learner to identify modified data, and furthermore, in distributed settings, the learner may not even be able to detect which nodes are under attack. Potential real world examples of the attackers are discussed as follows.

EXAMPLE 1 (Air pollution detection) [20].

Consider an air pollution detection system which uses DSVM as the classifiers to determine whether certain areas have air pollution. An attacker can modify the training data of the certain areas to let the system fail to recognize the air pollution. Moreover, the attacker can even modify other areas' training data to achieve his goal, since misleading information can be spread among the whole system by the communications between neighboring nodes. However, with a large amount of training data and areas, the learner will fail to detect the compromised data and areas, and the results of the air pollution detection system will be untrustworthy.

Example 2 (Distributed medical databases) [13].

Suppose several medical centers aim to find classifiers together on a certain disease using DSVM. An attacker can modify the training data of one medical center, which affects not only the compromised medical center, but also the uncompromised medical centers, as the misleading information can be spread among the network. As a result, all the medical centers might give inaccurate diagnosis on the disease. To find out the compromised training data, the learner is required to examine all the training data from all the medical centers, which is costly and sometimes even unrealistic.

Now Problem (1) changes to,

$$\min_{\{\mathbf{w}_v,b_v,\{\xi_{vn}\}\}} \frac{1}{2}\sum_{v\in\mathcal{V}}\|\mathbf{w}_v\|_2^2 + VC_l\sum_{v\in\mathcal{V}}\sum_{n=1}^{N_v}\xi_{vn}$$

s.t.

$$\begin{array}{ll}
y_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)\geq 1-\xi_{vn}, & \forall v\in\mathcal{V}_l,\ n=1,\ldots,N_v;\\
y_{vn}(\mathbf{w}_v^T\hat{\mathbf{x}}_{vn}+b_v)\geq 1-\xi_{vn}, & \forall v\in\mathcal{V}_a,\ n=1,\ldots,N_v;\\
\xi_{vn}\geq 0, & \forall v\in\mathcal{V},\ n=1,\ldots,N_v;\\
\mathbf{w}_v=\mathbf{w}_u,\ b_v=b_u, & \forall v\in\mathcal{V},\ u\in\mathcal{B}_v.
\end{array} \tag{8}$$

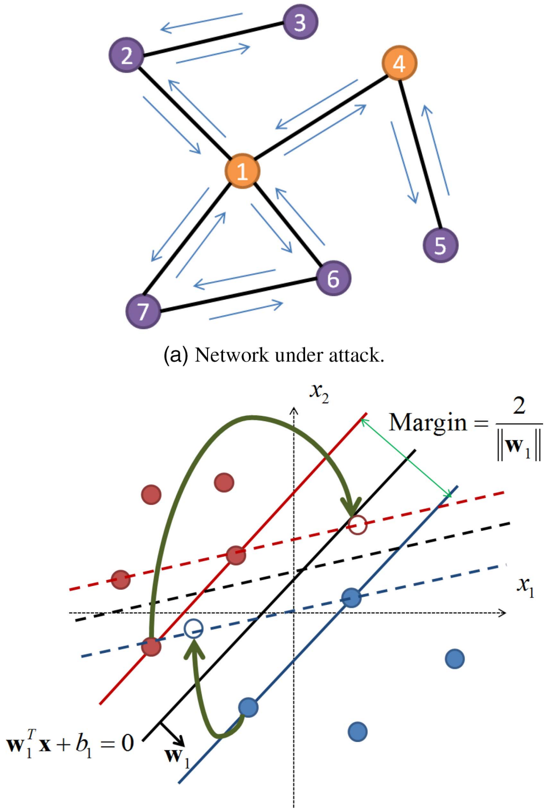
By minimizing the objective function in Problem (8), the learner can obtain the optimal variables $\{\mathbf{w}_v, b_v\}$, which can be used to build up the discriminant function to classify the testing data. The attacker, on the other hand, aims to find an optimal way to modify the data using variables $\{\delta_{vn}\}$ to maximize the same objective function. The behavior of the attacker can thus be captured as follows:

$$\max_{\{\delta_{vn}\}} \frac{1}{2}\sum_{v\in\mathcal{V}}\|\mathbf{w}_v\|_2^2 + VC_l\sum_{v\in\mathcal{V}}\sum_{n=1}^{N_v}\xi_{vn} - C_a\sum_{v\in\mathcal{V}_a}\sum_{n=1}^{N_v}\|\delta_{vn}\|_0$$

s.t.

$$\begin{array}{ll}
y_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)\geq 1-\xi_{vn}, & \forall v\in\mathcal{V}_l,\ n=1,\ldots,N_v;\\
y_{vn}(\mathbf{w}_v^T(\mathbf{x}_{vn}-\delta_{vn})+b_v)\geq 1-\xi_{vn}, & \forall v\in\mathcal{V}_a,\ n=1,\ldots,N_v;\\
\xi_{vn}\geq 0, & \forall v\in\mathcal{V},\ n=1,\ldots,N_v;\\
\mathbf{w}_v=\mathbf{w}_u,\ b_v=b_u, & \forall v\in\mathcal{V},\ u\in\mathcal{B}_v;\\
\delta_{vn}\in\mathcal{U}_v, & \forall v\in\mathcal{V}_a.
\end{array} \tag{9}$$

In above problem, the term $C_a\sum_{v\in\mathcal{V}_a}\sum_{n=1}^{N_v}\|\delta_{vn}\|_0$ represents the cost function for the attacker. l_0-norm is defined as $\|x\|_0 := |\{i : x_i \neq 0\}|$, i.e., a total number of nonzero elements in a vector. Here, we use the l_0-norm to denote the number of elements which are changed by the attacker. The objective function with l_0-norm captures the fact that the attacker aims to make the largest impact on the learner by changing the least number of elements. Constraint $\delta_{vn}\in\mathcal{U}_v$ indicates the action set of the attacker. In this paper, we use the following form of $\mathcal{U}_v$:


3
2
4
1
5
6
7
x_2
$\text{Margin} = \frac{2}{\|\mathbf{w}_1\|}$
x_1
$\mathbf{w}_1^T\mathbf{x}+b_1=0$
$\mathbf{w}_1$

(a) Network under attack.

(b) SVMs at compromised node 1.

Fig. 2. Network with attacker. (a) Node 1 and 4 are under attack. (b) In compromised node, for example, node 1, an attacker modifies the training data which leads to a wrong linear discriminant line (Black dotted line).

$$\mathcal{U}_v = \left\{(\delta_{v1},\ldots,\delta_{vN_v})\left|\sum_{n=1}^{N_v}\|\delta_{vn}\|_2^2\leq C_{v,\delta}\right.\right\},$$

which is related to the atomic action set $\mathcal{U}_{v0} = \{\delta_v \mid \|\delta_v\|_2^2 \leq C_{v,\delta}\}$. $C_{v,\delta}$ indicates the bound of the sum of the norm of all the changes at node v. A higher $C_{v,\delta}$ indicates that the attacker has a larger degree of freedom in changing the value $\mathbf{x}_{vn}$. Thus, training these data will lead to a higher risk for the learner. Notice that $C_{v,\delta}$ can vary at different nodes, and we use C_δ to represent the situation when $C_{v,\delta}$ are equal at every node. $\delta_v\in\mathbb{R}^p$ from the atomic action set has the same form with δ_{vn}, but δ_v and $(\delta_{v1},\ldots,\delta_{vN_v})$ are bounded by same $C_{v,\delta}$. Furthermore, the atomic action set $\mathcal{U}_{v0}$ has the following properties [32].

(P1) $\mathbf{0}\in\mathcal{U}_{v0}$;

(P2) For any $\mathbf{w}_0\in\mathbb{R}^p$:

$$\max_{\delta_v\in\mathcal{U}_{v0}}[\mathbf{w}_0^T\delta_v] = \max_{\delta'_v\in\mathcal{U}_{v0}}[-\mathbf{w}_0^T\delta'_v] < +\infty.$$

The first property (P1) states that the attacker can choose not to change the value of $\mathbf{x}_{vn}$. Property (P2) states that the atomic action set is bounded and symmetric. Here, "bounded" means that the attacker has the limit on the capability of changing $\mathbf{x}_{vn}$. It is reasonable since changing the value significantly will result in the evident detection of the learner.

For the learner, the learning process is to find the discriminant function which separates the training data into two classes with less error, and then use the discriminant function to classify testing data. Since the attacker has the ability to change the value of original data $\mathbf{x}_{vn} \in \mathcal{X}$ into $\hat{\mathbf{x}}_{vn} \in \hat{\mathcal{X}}$, the learner will find the discriminant function that separates the data in $\hat{\mathcal{X}}$ more accurate, rather than the data in $\mathcal{X}$. As a result, when using the discriminant function to classify the testing data $\mathbf{x} \in \mathcal{X}$, it will be prone to be misclassified.

Since the learner aims at a high classification accuracy, while the attacker seeks to lower the accuracy, we can capture the conflicting goals of the players as a two-person nonzero-sum game by combining Problem (8) and Problem (9) together. The solution to the game problem is described by Nash equilibrium, which yields the equilibrium strategies for both players, and predicts the outcome of machine learning in the adversarial environment. By comparing Problem (8) with Problem (9), we notice that they contain the same terms in their objective functions and the constraints in the two problems are uncoupled. As a result, the nonzero-sum game can be reformulated into a zero-sum game, which takes the minimax or max-min form as follows:

$$
\begin{aligned}
\min_{\{\mathbf{w}_v, b_v, \{\xi_{vn}\}\}} \max_{\{\delta_{vn}\}} & \frac{1}{2} \sum_{v \in \mathcal{V}} \|\mathbf{w}_v\|_2^2 + VC_l \sum_{v \in \mathcal{V}} \sum_{n=1}^{N_v} \xi_{vn} \\
& - C_a \sum_{v \in \mathcal{V}_a} \sum_{n=1}^{N_v} \|\delta_{vn}\|_0
\end{aligned}
$$

s.t.

$$
\begin{aligned}
& \mathrm{y}_{vn}(\mathbf{w}_v^T \mathbf{x}_{vn} + b_v) \ge 1 - \xi_{vn}, && \forall v \in \mathcal{V}_l,\ n = 1, \ldots, N_v; \\
& \mathrm{y}_{vn}(\mathbf{w}_v^T (\mathbf{x}_{vn} - \delta_{vn}) + b_v) \ge 1 - \xi_{vn}, && \forall v \in \mathcal{V}_a,\ n = 1, \ldots, N_v; \\
& \xi_{vn} \ge 0, && \forall v \in \mathcal{V},\ n = 1, \ldots, N_v; \\
& \mathbf{w}_v = \mathbf{w}_u, b_v = b_u, && \forall v \in \mathcal{V},\ u \in \mathcal{B}_v; \\
& \delta_{vn} \in \mathcal{U}_v, && \forall v \in \mathcal{V}_a.
\end{aligned}
\tag{10}
$$

Note that the first and fourth constraints only contribute to the minimization part of the problem, the fifth constraint only affects the maximization part. The second and third constraints contribute to both the minimization and the maximization part. The first term of the objective function is the inverse of the distance of margin. The second term is the sum of all the slack variables which captures the error penalties. On one hand, minimizing the objective function captures the trade-off between a larger margin and a small error penalty of the learner, while on the other hand, maximizing the objective function captures the trade-off between a large error penalty and a small cost of the attacker. As a result, solving Problem (10) can be understood as finding the saddle-point equilibrium of the zero-sum game between the attacker and the learner.

Definition 1 Let $\mathcal{S}_L$ and $\mathcal{S}_A$ be the action sets for the DSVM learner and the attacker, respectively. Notice that here $\mathcal{S}_A = \{\mathcal{U}_v\}_{v \in \mathcal{V}_a}$. Then, the strategy pair $(\{\mathbf{w}_v^*, b_v^*, \{\xi_{vn}^*\}\}, \{\delta_{vn}^*\})$ is a saddle-point equilibrium solution of the zero-sum game defined by the triple $G_z := \langle \{L, A\}, \{\mathcal{S}_L, \mathcal{S}_A\}, K \rangle$, if $K(\{\mathbf{w}_v^*, b_v^*, \{\xi_{vn}^*\}\}, \{\delta_{vn}\}) \le K(\{\mathbf{w}_v^*, b_v^*, \{\xi_{vn}^*\}\}, \{\delta_{vn}^*\}) \le K(\{\mathbf{w}_v, b_v, \{\xi_{vn}\}\}, \{\delta_{vn}^*\}), \forall v \in \mathcal{V}$, where K is the objective function of Problem (10).

Based on the property of the action set and atomic action set, Problem (10) can be further simplified as stated in the following lemma [34].

Lemma 2 *Assume that $\mathcal{U}_v$ is an action set with corresponding atomic action set $\mathcal{U}_{v0}$. Then, Problem (10) is equivalent to the following optimization problem:*

$$
\begin{aligned}
\min_{\{\mathbf{w}_v, b_v, \{\xi_{vn}\}\}} \max_{\{\delta_v\}} & \frac{1}{2} \sum_{v \in \mathcal{V}} \|\mathbf{w}_v\|_2^2 + VC_l \sum_{v \in \mathcal{V}} \sum_{n=1}^{N_v} \xi_{vn} \\
& + \sum_{v \in \mathcal{V}_a} (V_a C_l \mathbf{w}_v^T \delta_v - C_a \|\delta_v\|_0)
\end{aligned}
$$

s.t.

$$
\begin{aligned}
& \mathrm{y}_{vn}(\mathbf{w}_v^T \mathbf{x}_{vn} + b_v) \ge 1 - \xi_{vn}, && \forall v \in \mathcal{V},\ n = 1, \ldots, N_v; \\
& \xi_{vn} \ge 0, && \forall v \in \mathcal{V},\ n = 1, \ldots, N_v; \\
& \mathbf{w}_v = \mathbf{w}_u,\ b_v = b_u, && \forall v \in \mathcal{V},\ u \in \mathcal{B}_v; \\
& \delta_v \in \mathcal{U}_{v0}, && \forall v \in \mathcal{V}_a.
\end{aligned}
\tag{11}
$$

Proof See Appendix A.

In Problem (10), the second and third constraints are the coupled terms with the second term of the objective function. But in Problem (11), the only coupled term is $V_a C_l \mathbf{w}_v^T \delta_v$, which is linear in the decision variables of the attacker and the learner, respectively.

4. ADMOM-DSVM AND DISTRIBUTED ALGORITHM

In the previous section, we have combined Problem (8) for the learner with Problem (9) for the attacker into one minimax Problem (10), and have showed its equivalence to Problem (11). In this section, we develop iterative algorithms to find equilibrium solutions to Problem (11). Using a similar method in Section II, Problem (11) can be rewritten into matrix form as

$$\min_{\{\mathbf{r}_v,\xi_v,\omega_{vu}\}} \max_{\{\delta_v\}} \frac{1}{2}\sum_{v\in\mathcal{V}} \mathbf{r}_v^T(\mathbf{I}_{p+1}-\Pi_{p+1})\mathbf{r}_v + VC_l \sum_{v\in\mathcal{V}} \mathbf{1}_v^T \xi_v + \sum_{v\in\mathcal{V}_a}(V_aC_l\mathbf{r}_v^T(\mathbf{I}_{p+1}-\Pi_{p+1})\delta_v - C_a\|\delta_v\|_0)$$

s.t.

$$\begin{aligned}
\mathbf{Y}_v\mathbf{X}_v\mathbf{r}_v \geq \mathbf{1}_v - \boldsymbol{\xi}_v, &\quad \forall v\in\mathcal{V}; && (12a)\\
\boldsymbol{\xi}_v \geq \mathbf{0}_v, &\quad \forall v\in\mathcal{V}; && (12b)\\
\mathbf{r}_v = \omega_{vu}, \omega_{vu} = \mathbf{r}_u, &\quad \forall v\in\mathcal{V},\ \forall u\in\mathcal{B}_v; && (12c)\\
\delta_v \in \mathcal{U}_{v0}, &\quad \forall v\in\mathcal{V}_a. && (12d)
\end{aligned} \tag{12}$$

To solve problem (12), we use best response dynamics to construct the best response for the min-problem and max-problem separately. The min-problem and max-problem are archived by fixing $\{\delta_v\}$ and $\{\mathbf{r}_v\}$, respectively. With ADMoM [12], we can develop a method of solving Problem (12) in a distributed way as follows: The first step is that each node randomly picks an initial $\mathbf{r}_v^{(0)}$, $\delta_v^{(0)}$ and $\alpha_v = \mathbf{0}_{(p+1)\times 1}$, then solve the max-problem with $\{\mathbf{r}_v^{(0)}\}$, and obtain$\{\delta_v^{(1)}\}$. The next step is to solve the min-problem with $\{\delta_v^{(1)}\}$ and obtain $\{\mathbf{r}_v^{(1)}\}$, then we repeat solving the max-problem with $\{\mathbf{r}_v\}$ from the previous step and the min-problem with $\{\delta_v\}$ from the previous step until the pair $\{\mathbf{r}_v, \delta_v\}$ achieves convergence. The iterations of solving Problem (12) can be summarized as follows [34].

LEMMA 3 *With arbitrary initialization $\boldsymbol{\delta}_v^{(0)}$, $\mathbf{r}_v^{(0)}$, $\lambda_v^{(0)}$, $\omega_{vu}^{(0)}$ and $\alpha_v^{(0)} = \mathbf{0}_{(p+1)\times 1}$, the iterations per node are given by:*

$$\delta_v^{(t+1)} \in \arg\max_{\{\delta_v,s_v\}} V_aC_l\mathbf{r}_v^{(t)T}(\mathbf{I}_{p+1}-\Pi_{p+1})\delta_v - \mathbf{1}^T s_v$$

$$\text{s.t.}\quad \begin{aligned}
C_a\delta_v \leq s_v, &\quad \forall v\in\mathcal{V}_a;\\
C_a\delta_v \geq -s_v, &\quad \forall v\in\mathcal{V}_a;\\
\delta_v \in \mathcal{U}_{v0}, &\quad \forall v\in\mathcal{V}_a.
\end{aligned} \tag{13}$$

$$\lambda_v^{(t+1)} \in \arg\max_{\mathbf{0}\leq\boldsymbol{\lambda}_v\leq VC_l\mathbf{1}_v} -\tfrac{1}{2}\lambda_v^T\mathbf{Y}_v\mathbf{X}_v\mathbf{U}_v^{-1}\mathbf{X}_v^T\mathbf{Y}_v\lambda_v + (\mathbf{1}_v + \mathbf{Y}_v\mathbf{X}_v\mathbf{U}_v^{-1}\mathbf{f}_v^{(t)})^T\lambda_v, \tag{14}$$

$$\mathbf{r}_v^{(t+1)} = \mathbf{U}_v^{-1}(\mathbf{X}_v^T\mathbf{Y}_v\lambda_v^{(t+1)} - \mathbf{f}_v^{(t)}), \tag{15}$$

$$\omega_{vu}^{(t+1)} = \tfrac{1}{2}(\mathbf{r}_v^{(t+1)} + \mathbf{r}_u^{(t+1)}), \tag{16}$$

$$\alpha_v^{(t+1)} = \alpha_v^{(t)} + \frac{\eta}{2}\sum_{u\in\mathcal{B}_v}[\mathbf{r}_v^{(t+1)} - \mathbf{r}_u^{(t+1)}], \tag{17}$$

where $\mathbf{U}_v = (\mathbf{I}_{p+1}-\Pi_{p+1}) + 2\eta|\mathcal{B}_v|\mathbf{I}_{p+1}$, $\mathbf{f}_v^{(t)} = V_aC_l\delta_v^{(t)} + 2\alpha_v^{(t)} - 2\eta\sum_{u\in\mathcal{B}_v}\omega_{vu}^{(t)}$.

PROOF See Appendix B.

Iteration (13) corresponds to the attacker's Max-Problem (9), while iterations (14)–(17) correspond to the learner's Min-Problem (8). The Minimax Problem (11) is solved by iterating them together. Note that, iterations (14)–(17) differ from iterations (3)–(6) only in $\mathbf{f}_v$. In (14)–(17), $\mathbf{f}_v$ adds another term $V_aC_l\delta_v$ which captures the attacker's impact on the learner. Iterations (13)–(17) are summarized into Algorithm 2.

ALGORITHM 2: *DSVM under attack*

```
Randomly initialize $\boldsymbol{\delta}_v^{(0)}, \mathbf{r}_v^{(0)}, \lambda_v^{(0)}, \omega_{vu}^{(0)}$ and $\alpha_v^{(0)} = \mathbf{0}_{(p+1)\times 1}$.
for $t = 0, 1, 2, \ldots$ do
    for all $v \in \mathcal{V}$ do
        Compute $\delta_v^{(t+1)}$ via (13).
    end for
    for all $v \in \mathcal{V}$ do
        Compute $\lambda_v^{(t+1)}$ via (14).
        Compute $\mathbf{r}_v^{(t+1)}$ via (15).
    end for
    for all $v \in \mathcal{V}$ do
        Broadcast $\mathbf{r}_v^{(t+1)}$ to all neighbors $u \in \mathcal{B}_v$.
    end for
    for all $v \in \mathcal{V}$ do
        Compute $\omega_{vu}^{(t+1)}$ via (16).
        Compute $\alpha_v^{(t+1)}$ via (17).
    end for
end for
```

Algorithm 2 solves the Minimax Problem (11) using ADMoM technique. It is a fully distributed algorithm which only requires transmitting $\mathbf{r}_v$ between each nodes. The attacker's behavior is captured as calculating δ_v by solving the linear programming Problem (13) with the learner's decision variable $\mathbf{r}_v$. The learner's behavior is captured as computing (14)–(17) with δ_v from the attacker. Since the learner transmits $\mathbf{r}_v$ to each neighboring nodes, misleading information will eventually spread in the whole network, which leads to misclassifications in all nodes.

5. NUMERICAL RESULTS

In this section, we summarize numerical results of DSVM under adversarial environments. We use empirical risk to measure the performance of DSVM. The empirical risk at node v at step t is defined as follows:

$$\mathbf{R}_v^{(t)} := \frac{1}{\tilde{N}_v}\sum_{n=1}^{\tilde{N}_v}\frac{1}{2}|\tilde{y}_{vn} - \hat{y}_{vn}^{(t)}|, \tag{18}$$

where $\tilde{y}_{vn}$ is the true label; $\hat{y}_{vn}^{(t)}$ is the predicted label; and $\tilde{N}_v$ represents the number of testing samples in node v. The empirical risk (18) sums over the number of misclassified samples in node v, and then divides it by the number of all testing samples in node v. Notice that testing samples can vary for different nodes. In order to measure the global performance, we use the global empirical risk defined as follows:

$$\mathbf{R}_G^{(t)} := \frac{1}{\tilde{N}}\sum_{v\in\mathcal{V}}\sum_{n=1}^{\tilde{N}_v}\frac{1}{2}|\tilde{y}_{vn} - \hat{y}_{vn}^{(t)}|, \tag{19}$$

where $\tilde{N} = \sum_{v\in\mathcal{V}} \tilde{N}_v$, representing the total number of testing samples. Clearly, a higher global empirical risk shows that there are more testing samples being misclassified, i.e., a worse performance of DSVM. We use the first experiment to illustrate the significant impact of the attacker.

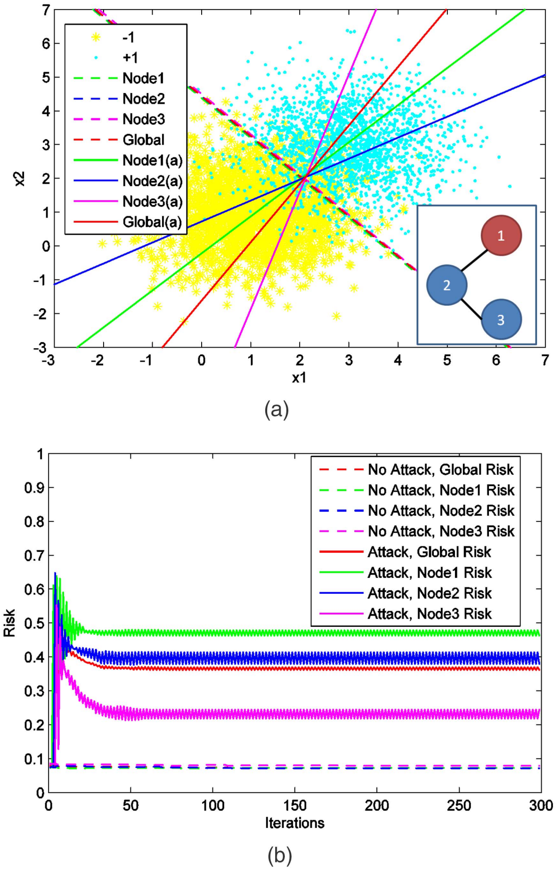
Consider a network with 3 nodes, which can be seen at the bottom right corner of Fig. 3(a). Each node contains 80 training samples and 1000 testing samples from the same global training dataset, which is shown as points and stars in Fig. 3(a). Yellow stars and magenta points are labelled as -1 and $+1$, respectively. They are generated from two-dimensional Gaussian distributions with mean vectors $[1,1]$ and $[3,3]$, with the same covariance matrix $[1,0;0,1]$. The learner has the ability $C_l = 1$ and $\eta = 1$. The attacker has the atomic action set parameter $C_{1,\delta} = 9{,}000{,}000$, and the cost parameter $C_a = 1$. The attacker only attacks Node 1 and the attack starts from the beginning of the training process. Numerical results are shown in Fig. 3(b). Notice that the risks when there is an attacker are much higher than the risks when there is no attacker, which indicates that the attacker has a significant impact on the learner. Also, we can conclude that the risks at the node under attack are much higher than the risks in nodes without attack, but both of them are higher than the risks when there is no attacker in the network. This shows that the attacker has the ability to affect uncompromised nodes through network connections. We can also observe from Fig. 3(a) that the solid lines, which represent the situation when there is an attacker, cannot separate yellow stars and magenta points.

It is clear that the attacker can cause disastrous results for the learner. In our previous work [34], we have shown that results of the game between the DSVM learner and the attacker are affected by both the attacker's ability and the network topologies. We summarize our previous numerical results from [34] in the following observations.

Observation 1

The attacker's ability is captured by four measures, i.e., (i) the time t for the attacker to take an action, (ii) the atomic action set parameter $C_{v,\delta}$, (iii) the cost parameter C_a, and (iv) the number of compromised nodes $|\mathcal{V}_a|$. The impact of them is summarized as follows.

- *The time t for an attacker to take an action does not affect the equilibrium risks.*
- *A larger $C_{v,\delta}$ increases the equilibrium risk, as a larger $C_{v,\delta}$ indicates that the attacker can make a larger modification on training data.*
- *A larger C_a decreases the equilibrium risk, as a larger C_a restricts the attacker's actions to make changes.*



(a)

(b)

Fig. 3. Evolution of the empirical risks of ADMoM-DSVM with an attacker at a network with 3 nodes shown at the bottom right corner of figure (a). The attacker only attacks red node 1 from the beginning of the training process. Training data and testing data are generated from two Gaussian classes. Dotted lines and solid lines show the results when there is no attacker and there is an attacker, respectively. Different colors represent risks or discriminant lines of different nodes.

- *A larger number of compromised nodes $|\mathcal{V}_a|$ increases the equilibrium risk as attacking more nodes gives the attacker access to modify more training samples.*

Observation 2 *Denote the degree of node v as $|\mathcal{B}_v|/(|\mathcal{V}|-1)$ and the degree of a network as the average degree of all the nodes. The impact of network topologies are summarized as follows.*

- *Networks with higher degrees and fewer nodes are less vulnerable to attackers.*
- *Balanced networks, i.e., nodes in these networks have the same number of neighboring nodes, are more secure than unbalanced networks.*

Notice that here we assume that each node in the network contains the same number of training samples.

Observation 3 *For a specified network, assuming that all the nodes contain the same number of training samples, the impact of a node is summarized as follows.*

- *Nodes with higher degrees are more vulnerable, i.e., attacking nodes with higher degrees leads to a higher global equilibrium risk.*

- *Attacking nodes with lower degrees can lead to a higher global equilibrium risk if the network contains nodes with higher degrees, comparing to networks without high degree nodes but has the same average degree.*

Observations 1, 2 and 3 summarize our previous numerical experiments in [34]. From Observation 1, the attacker makes a larger impact when he has a higher capability, such as, he has a larger $C_{v,\delta}$ and a smaller C_a, or he can attack more nodes. From Observation 2, on the one hand, the attacker can choose to attack unbalanced networks with lower degrees and more nodes to make a more significant impact on the learner, on the other hand, the learner should select balanced networks with higher degrees and fewer nodes to reduce potential damages from attacker. From Observation 3, the attacker benefits more from attacking nodes with higher degrees, while the learner should avoid using high degree nodes. These observations provide both players the strategies to make a larger impact on the other ones. In the following subsections, we present in detail how the attacker and the learner can find better strategies against each other.

5.1. Attacker's Strategies

Consider that a DSVM learner operates training data on an unbalanced network. We assume that the attacker knows the learner's algorithm and the network topology. We also assume that the attacker has the ability to attack any nodes in this network with $\sum_{v=1}^{V_a} C_{v,\delta} \leq C_{V_a,\delta}$, i.e., a total sum of all changed values in the network should be bounded by $C_{V_a,\delta}$. Notice that bounded $C_{V_a,\delta}$ represents a trade-off between attacking more nodes V_a and attacking each nodes with larger $C_{v,\delta}$. Since attacking different nodes leads to different global equilibrium risks, and the attacker prefers higher risks, there exists an optimal strategy of selecting $\mathcal{V}_a$ and $\{C_{v,\delta}\}_{v\in\mathcal{V}_a}$ for the attacker which has the highest equilibrium global risk with a bounded $C_{V_a,\delta}$. The optimal strategy can be found by solving the following problem:

$$
\begin{aligned}
&\max_{\{\mathcal{V}_a, C_{v,\delta}\}} \min_{\{\mathbf{w}_v, b_v, \{\xi_{vn}\}\}} \max_{\{\delta_{vn}\}} \frac{1}{2}\sum_{v\in\mathcal{V}} \|\mathbf{w}_v\|_2^2 + VC_l \sum_{v\in\mathcal{V}}\sum_{n=1}^{N_v} \xi_{vn} \\
&\qquad - C_a \sum_{v\in\mathcal{V}_a}\sum_{n=1}^{N_v} \|\delta_{vn}\|_0 - \sum_{v\in\mathcal{V}_a} h_v \\
&\text{s.t.} \\
&\mathrm{y}_{vn}(\mathbf{w}_v^T \mathbf{x}_{vn} + b_v) \geq 1 - \xi_{vn}, \qquad \forall v \in \mathcal{V}_l,\ n = 1,\ldots,N_v; \\
&\mathrm{y}_{vn}(\mathbf{w}_v^T (\mathbf{x}_{vn} - \delta_{vn}) + b_v) \geq 1 - \xi_{vn}, \quad \forall v \in \mathcal{V}_a,\ n = 1,\ldots,N_v; \\
&\xi_{vn} \geq 0, \qquad \forall v \in \mathcal{V},\ n = 1,\ldots,N_v; \\
&\mathbf{w}_v = \mathbf{w}_u,\ b_v = b_u, \qquad \forall v \in \mathcal{V},\ u \in \mathcal{B}_v; \\
&\delta_{vn} \in \mathcal{U}_v,\ \text{i.e.,}\ \textstyle\sum_{n=1}^{N_v} \|\delta_{vn}\|_2^2 \leq C_{v,\delta}, \quad \forall v \in \mathcal{V}_a; \\
&\textstyle\sum_{v=1}^{V_a} C_{v,\delta} \leq C_{V_a,\delta}.
\end{aligned}
\tag{20}
$$

Note that Problem (20) extends Problem (10) by maximizing over variables $\mathcal{V}_a$ and $\{C_{v,\delta}\}$ with a new constraint $\sum_{v}^{V_a} C_{v,\delta} \leq C_{V_a,\delta}$ that captures a bound on the attacker's ability. The last term h_v in the objective function represents the cost of attacking node v.

Problem (20) is based on the assumption that the attacker has the knowledge of the learner's algorithm and the network topology. The learner aims to minimize the classification errors in Problem (10), while the attacker aims to maximize those errors. In Problem (20), the attacker has two components to maximize. Maximizing over $\{\delta_{vn}\}$ is the same as in Problem (10). Maximizing over $\mathcal{V}_a$ and $\{C_{v,\delta}\}$ indicates the objective of the attacker to maximize the equilibrium risk of the original game with a bounded $C_{V_a,\delta}$ and a cost h_v. By solving Problem (20), the attacker can find the optimal strategy of $\mathcal{V}_a$ and $\{C_{v,\delta}\}_{v\in\mathcal{V}_a}$, which has the maximized equilibrium risk.

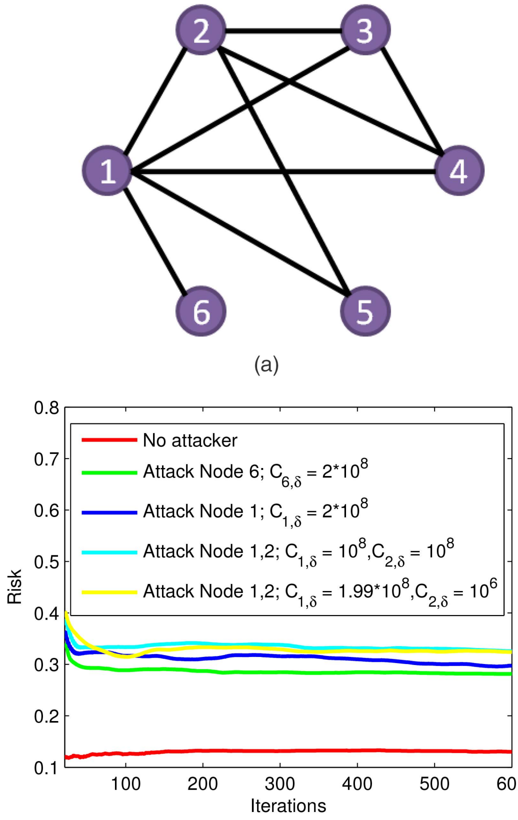
However, solving Problem (20) can be a challenge as the decision variables $\mathcal{V}_a$ and $C_{v,\delta}$ are coupled with the decisions of the learner and the attacker. The attacker is still able to make a larger impact on the learner by Observation 1, 2 and 3. For example, instead of randomly picking nodes to attack and assigning $C_{v,\delta}$, the attacker can strategically attack high degree nodes, which leads to a higher risk from our observations. One numerical example is shown in Fig. 4.

Consider the learner operates on a network shown in Fig. 4(a). We assume that the attacker can only attack 2 nodes with the bound $C_{V_a,\delta} = 2 \times 10^8$, and the cost of attacking node v, i.e., h_v are the same for every node. A naive attacker may randomly attack 1 node with $C_{v,\delta} = 2 \times 10^8$. However, a smart attacker will choose 2 nodes with higher degrees, and by modifying the value of $C_{v,\delta}$ in both nodes, he can make a larger impact on the learner. Numerical results are shown in Fig. 4(b).

From Fig. 4, the attacker has four different strategies, (i) the attacker only attacks Node 6, (ii) the attacker only attacks Node 1, (iii) the attacker attacks Node 1,2 with balanced ability, and (iv) the attacker attacks Node 1,2 with unbalanced ability. We can see that when the attacker choose Strategy (iii), the risk is the highest. However, if we take the cost of attacking different nodes into consideration, this strategy may not be the best as attacking 2 nodes may cost too much. But from the example, we can see that Observations 1, 2 and 3 provide us a way to find a better strategy for the attacker. They also provide us tools of finding better strategies for the learner.

5.2. Learner's Strategies

A DSVM learner aims to find the best discriminant functions with the least classification errors. Since an attacker will increase the classification errors, a better strategy of the learner is to reduce the attacker's impact as much as possible. In this section, we assume that the learner is trying to find the strategy of network topology that has a smallest risk with potential attacks.



(a)

(b)

Fig. 4. Evolution of moving average of global empirical risks of ADMoM-DSVM with the attacker on Spam dataset [14]. Each node contains 40 training samples. Attacker has four strategies with same $C_{\mathcal{V}_a,\delta} = 2\times 10^8$ and $C_a = 0.01$.

We assume that the learner has the ability to select any kinds of network topologies and assign any number of training samples in each node. The learner's strategy can be found by solving the following problem,

$$\min_{\{\mathcal{V},\mathcal{B}_v,N_v\}}\ \min_{\{\mathbf{w}_v,b_v,\{\xi_{vn}\}\}}\ \max_{\{\delta_{vn}\}}\ \frac{1}{2}\sum_{v\in\mathcal{V}}\|\mathbf{w}_v\|_2^2 + VC_l\sum_{v\in\mathcal{V}}\sum_{n=1}^{N_v}\xi_{vn} - C_a\sum_{v\in\mathcal{V}_a}\sum_{n=1}^{N_v}\|\delta_{vn}\|_0 - \sum_{v\in\mathcal{V}}T_v(N_v) - \sum_{v\in\mathcal{V}}B_v(\mathcal{B}_v)$$

s.t.

$$\begin{array}{ll}
y_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)\geq 1-\xi_{vn}, & \forall v\in\mathcal{V}_l,\ n=1,\ldots,N_v;\\
y_{vn}(\mathbf{w}_v^T(\mathbf{x}_{vn}-\delta_{vn})+b_v)\geq 1-\xi_{vn}, & \forall v\in\mathcal{V}_a,\ n=1,\ldots,N_v;\\
\xi_{vn}\geq 0, & \forall v\in\mathcal{V},\ n=1,\ldots,N_v;\\
\mathbf{w}_v=\mathbf{w}_u,\ b_v=b_u, & \forall v\in\mathcal{V},\ u\in\mathcal{B}_v;\\
\delta_{vn}\in\mathcal{U}_v,\ \text{i.e., } \sum_{n=1}^{N_v}\|\delta_{vn}\|_2^2\leq C_{v,\delta}, & \forall v\in\mathcal{V}_a.
\end{array} \tag{21}$$

Note that Problem (21) extends Problem (10) by minimizing over variables $\mathcal{V}$, $\mathcal{B}_v$ and N_v with new costs $T_v(N_v)$ and $B_v(\mathcal{B}_v)$. $T_v(N_v)$ represents the cost of training N_v samples in node v, $B_v(\mathcal{B}_v)$ represents the cost of sending information from node v to his neighboring nodes $u\in\mathcal{B}_v$. Problem (21) can be understood as the learner's objective of minimizing equilibrium risk of the game with potential attacks by finding the best network topology $\mathcal{V}$, $\mathcal{B}_v$ and training samples' assignments N_v.

Solving Problem (21) can be a challenge as $\mathcal{V}$, $\mathcal{B}_v$ and N_v are coupled with the decisions of the learner and the attacker. But the learner can benefit from Observation 1, 2 and 3. For example, the learner should select a balanced network with fewer nodes and higher degree, which has a smaller equilibrium risk. However, in reality, the learner may not be able to modify network topologies as the connections between nodes can be fixed, or it may not be possible to add connections between nodes. Thus, to reduce the impact of the attacker, the learner requires actionable defense strategies.

In the following sections, we present four different defense strategies, and we verify their effectiveness with numerical experiments.

6. DSVM DEFENSE STRATEGIES

In this section, we present four defense strategies (DSs) for the DSVM learner. We show their effectiveness with numerical experiments.

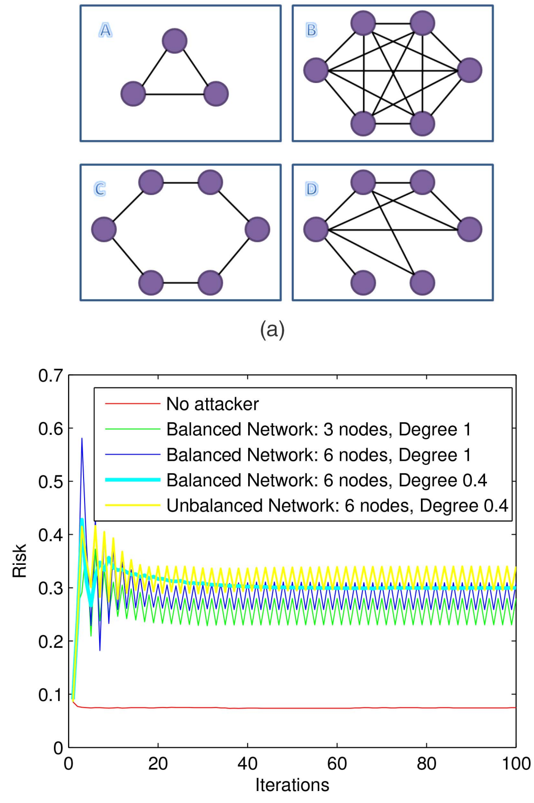
6.1. DSVM Defense Strategy 1: Selecting Network Topology

DS 1 for the learner is to find a network topology that has a smaller risk when there is an attacker. From the last section, the learner can find the network topology by solving Problem (21). However, Problem (21) is difficult to solve. But we are still able to find a secure network topology using Observation 2 and 3. The network topology should be close to a balanced network with fewer nodes and a higher degree. A numerical experiment is shown in Fig. 5.

Consider that a DSVM learner trains 300 samples, and he aims to select a secure network topology from four topologies shown in Fig. 5(a). DS 1 indicates that we should select network A or B as network A has the smallest number of nodes among all the networks, and network B has the highest degree among networks B,C,D. Numerical results in Fig. 5(b) show that DS 1 has smaller risks.

Though selecting a network with fewer nodes reduces the vulnerability of the learner, but each node is required to train more training samples, which takes more time and memory usages. In addition, the learner may not have the ability to select a proper network topology as most networks are fixed. Moreover, improving the degree of the network may not be always applicable as adding connections between nodes is costly. Thus DS 1 is suitable for cases when the network connections are convenient to modify.

Consider the application in which several wireless temperature sensors in the building aim to decide whether to open their air conditioners or not. Since a large building may have hundreds of sensors and the



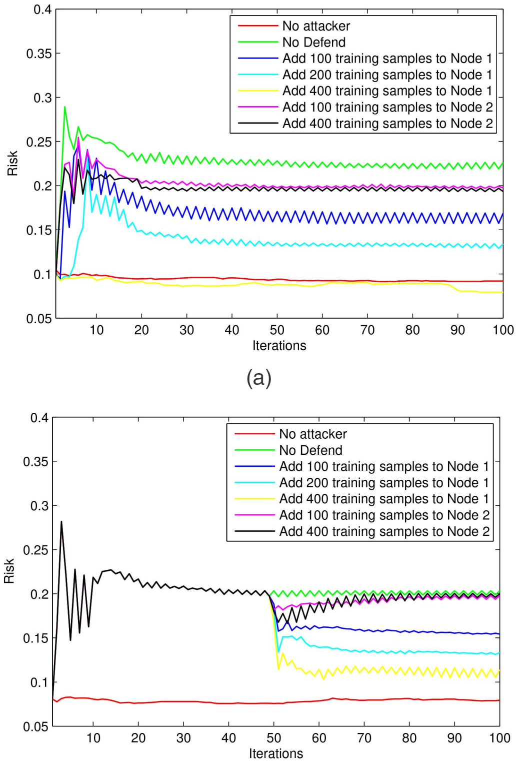
(b)

Fig. 5. Evolution of the global empirical risks of ADMoM-DSVM with an attacker on a random dataset. The learner has four options of network topologies which are shown in figure (a). Topology A is a balanced network with 3 nodes and degree 1, each node in this network contains 80 training samples. Network B is a balanced network with 6 nodes and degree 1. Network C is a balanced network with 6 nodes and degree 0.4. Network D is an unbalanced network with 6 nodes and degree 0.4. Each node in network B,C and D contains 40 training samples. Attacker attacks 1 node in network A, but he attacks 2 nodes in network B,C,D, so the attacker can modify the same number of training samples in different network topologies. The attacker has $C_{v,\delta} = 5 \times 10^5$ and $C_a = 0.01$.

temperatures are always changing with time, centralized classifications may take a significant amount of time to collect, transmit, and process the data. DSVMs can be used here as each sensor operates on its own data, and only a small amount of information is transmitted between sensors. But if there is an attacker who has the ability to modify the training data in several sensors, then the sensors in the building will lead to wrong decisions. In this case, wireless temperature sensors can adapt and modify their network topology. Thus, a secure strategy here is to use DS 1 to create a balanced network with fewer sensors and a higher average degree.

6.2. DSVM Defense Strategy 2: Adding Training Samples

Since the attacker is limited to making modifications on the training data, a higher volume of training data will decrease the ratio of incorrect data at a node. As long as most of the data are correct, the learner can find the discriminant function with small classification errors. Thus adding more training samples becomes a reasonable defense strategy. Numerical experiments are shown in Fig. 6.



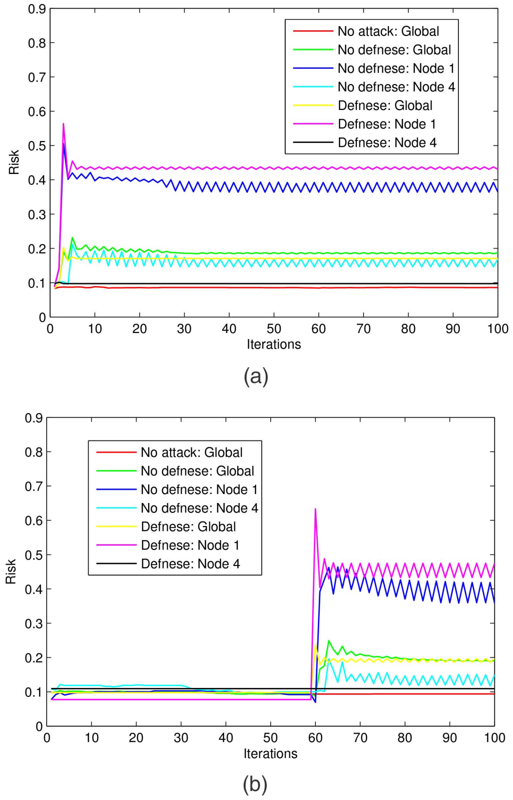
(a)

(b)

Fig. 6. Evolution of global empirical risks of ADMoM-DSVM with an attacker at a balanced network with 6 nodes and degree 0.4 on random dataset, which is shown in Fig. 5(a) as Network C. Each node contains 40 training samples. Attacker only attacks node 1 with $C_{1,\delta} = 10^6$ and $C_a = 0.01$. (a) Defense starts from step 0. (b) Defense starts from step 50.

From Fig. 6, when we add training samples to network, the risk is lower. Thus adding training samples is a proper defense strategy. Note that more samples we add, the lower the risk will be. Adding training samples to compromised nodes turns out to be more efficient than adding to uncompromised nodes. However, training more samples requires more time and memory usages, which sacrifices efficiency. Thus, DS 2 is a trade-off between efficiency and security.

DS 2 is suitable for the case when the learner cannot change the network topology, but the size of training data is sufficiently large and each node has a strong computing capability. For example, consider an application where several environmental stations plan to detect whether some areas are under pollution with a wired communication network. DSVMs are suitable to process a large amount of data computations and transmissions. However, if an attacker modifies the training data, environmental stations may lead to misdetection. In this case, DS 1 may not be applicable as the wired connections between each station are fixed. However, since each station can collect enough training data and



(a)

(b)

Fig. 7. Evolution of the empirical risks of ADMoM-DSVM with the attacker at a balanced network with 4 nodes degree 0.4 on random dataset. Each node contains 60 training samples. Attacker only attacks Node 1 with $C_{1,\delta} = 10^5$ and $C_a = 0.01$. $\tau = 0.1$. (a) Attack starts from step 0. (b) Attack starts from step 60.

has a higher computation capability, DS 2 is more appropriate and the learner can add more training samples to each node to make the training process more secure. Note that using more samples requires additional time to train and more spaces to store data.

6.3. DSVM Defense Strategy 3: Verification Method

DS 1 suggests that the learner uses a balanced network with fewer nodes and a higher degree. However, using fewer nodes requires that each node trains more training samples, which sacrifices the efficiency. Increasing the degree of the network requires creating more connections between nodes, which are usually not applicable as building new lines may incur a high cost. DS 2 indicates that adding more training samples can reduce the vulnerability of the network, which also sacrifices the efficiency. Thus, both DS 1 and DS 2 have their limitations on securing a training process. In this section, we present a verification method that reduces the vulnerability without modifying the network topology or adding training samples.

In ADMoM-DSVM Algorithm 1, each node in the network receives $\mathbf{r}_u$ from his neighboring nodes and it also sends his $\mathbf{r}_v$ to his neighboring nodes at each step. Since $\mathbf{r}_u$ from neighboring nodes of node v contributes to the updates of $\mathbf{r}_v$, a wrong $\mathbf{r}_u$ can lead to an incorrect update of $\mathbf{r}_v$. As a result, if node v is protected from receiving wrong $\mathbf{r}_u$ from compromised nodes, it can have a correct discriminant function.

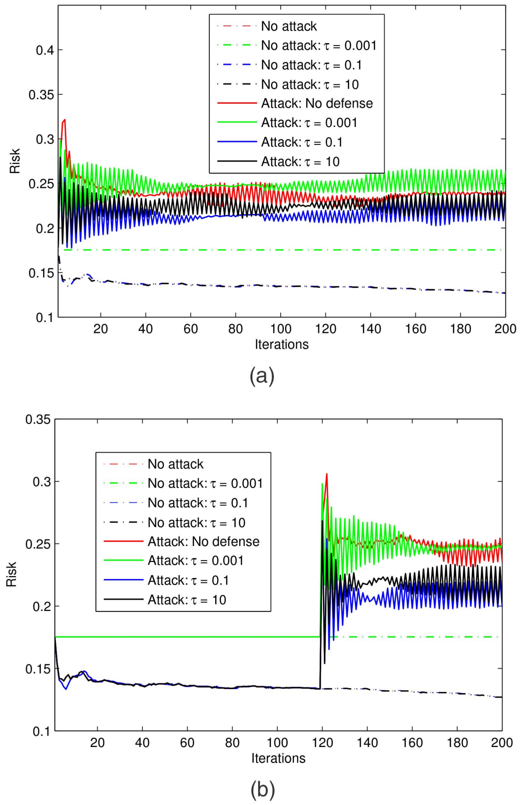
Recall DSVM Problem (2), note that consensus constraints $\mathbf{r}_v = \omega_{vu}, \omega_{vu} = \mathbf{r}_u$ force all the local decision variables $\mathbf{r}_v$ to agree with each other. Thus, $\mathbf{r}_1^{(t)} \approx \cdots \approx \mathbf{r}_V^{(t)}$ should hold for every step t during the training process. Thus, if $\mathbf{r}_v$ violates this, then the learner can tell that node v is under attack. With Algorithm 1, if node v finds $\mathbf{r}_u$ is significantly different from $\mathbf{r}_v$, then he will reject using $\mathbf{r}_u$ to update himself. We call this method as the verification method. The ADMoM-DSVM algorithm with verification method can be summarized as Algorithm 3.

ALGORITHM 3: *DSVM with Verification*

```
Randomly initialize $\mathbf{r}_v^{(0)}$, $\lambda_v^{(0)}$, $\omega_{vu}^{(0)}$, set $\alpha_v^{(0)} = \mathbf{0}_{(p+1)\times 1}$, set $\widehat{\mathcal{B}_v} = \mathcal{B}_v$.
 1: for $t = 0, 1, 2, \ldots$ do
 2:   for all $v \in \mathcal{V}$ do
 3:     Compute $\lambda_v^{(t+1)}$ via (3) with $\widehat{\mathcal{B}_v}$.
 4:     Compute $\mathbf{r}_v^{(t+1)}$ via (4) with $\widehat{\mathcal{B}_v}$.
 5:   end for
 6:   for all $v \in \mathcal{V}$ do
 7:     Broadcast $\mathbf{r}_v^{(t+1)}$ to all neighbors $u \in \mathcal{B}_v$.
 8:   end for
 9:   for all $v \in \mathcal{V}$ do
10:     Set $\hat{\mathcal{B}}_v = \emptyset$.
11:     for all $u \in \mathcal{B}_v$ do
12:       if $\left| 1 - \frac{\|\mathbf{r}_u^{(t+1)}\|_2}{\|\mathbf{r}_v^{(t+1)}\|_2} \right| < \tau$
13:         Set $u \in \widehat{\mathcal{B}_v}$.
14:       end if
15:     end for
16:   end for
17:   for all $v \in \mathcal{V}$ do
18:     Compute $\omega_{vu}^{(t+1)}$ via (5) with $\widehat{\mathcal{B}_v}$.
19:     Compute $\alpha_v^{(t+1)}$ via (6) with $\widehat{\mathcal{B}_v}$.
20:   end for
21: end for
```

Algorithm 3 differs from Algorithm 1 in the verification method. Each node computes with information only from trusted neighboring nodes $u \in \hat{\mathcal{B}}_v$. The verification method is based on the inequality in step 12 of Algorithm 3. τ indicates the tolerance of indifference from $\mathbf{r}_u$ to $\mathbf{r}_v$, and $\tau \geq 0$. When τ is close to 0, node v is very sensitive to the information from other nodes, and it only uses $\mathbf{r}_u$ that is very close to $\mathbf{r}_v$. Numerical experiments are shown in Fig. 7 and Fig. 8.

We can see from Fig. 7 that the global risk has decreased when there is a verification method. Note that in uncompromised node 4, the risk is close to the risk when there is no attacker, while in compromised node 1, the risk is higher than the risk when there is



(a)

(b)

Fig. 8. Evolution of the global empirical risks of ADMoM-DSVM with the attacker at a balanced network with 4 nodes degree 0.4 on Spambase dataset [14]. Each node contains 60 training samples. Attacker only attacks node 1 with $C_{1,\delta} = 10^6$ and $C_a = 0.01$. (a) Attack starts from step 0. (b) Attack starts from step 120.

no defense. This indicates that, though the verification method protects uncompromised nodes from receiving misleading information, it also prevents compromised nodes from receiving correct information.

Fig. 8 compares the global risks when the learner uses different τ. We can see that when $\tau = 10$, the risk is higher than the risk when $\tau = 0.1$, thus some of the misleading information is still able to be spread in the network. When $\tau = 0.001$, we can see that the risk is even higher than the risk when there is no defense. Also note that when there is no attacker, the risk of DSVM with $\tau = 0.001$ does not converge to the risk of normal DSVM. This indicates that when τ is close to 0, the misleading information cannot be spread to other nodes, but the useful information is also forbidden to transmit. Thus DS 3 requires a proper selection of τ.

DS 3 is suitable for the case when training data are used in a large network. Since it is difficult for the attacker to attack many nodes at the same time, for a network with a large number of nodes, all the uncompromised nodes can be kept from being affected by the compromised nodes. Moreover, the learner can distinguish compromised nodes by their high local classification risks, and thus, without revoking the training process and retraining all the data in every node, the learner is able to maintain the resilience of the training process by deleting or correcting the compromised nodes. Comparing to DS 1 and 2, DS 3 does not sacrifice efficiency to maintain security, but the compromised nodes may result in worse performances.

6.4. DSVM Defense Strategy 4: Rejection Method

DSs 1, 2 and 3 have shown that with selecting proper network topologies, adding training samples and verification method, DSVM learner can be less vulnerable to attacks. However, DSs 1 and 2 will sacrifice efficiency. In DS 3, compromised nodes may result in worse performances. In this section, we present the rejection method where each node rejects unreasonable updates. With the rejection method, once there is an attacker, the iteration will terminate to prevent further damages caused by the attacker.

The rejection method relies on a combined residual, which measures both the primal and dual error simultaneously:

$$J^{(t+1)} = \eta \sum_{v\in\mathcal{V}} \sum_{u\in B_v} \|\omega_{vu}^{(t+1)} - \omega_{vu}^{(t)}\|_2^2 + \frac{2}{\eta} \sum_{v\in\mathcal{V}} \|\alpha_v^{(t+1)} - \alpha_v^{(t)}\|_2^2. \tag{22}$$

Note that the combined residual contains two terms. The first term measures the dual residual. The second term measures the primal residual. The combined residual has the following lemma [15].

LEMMA 4 *Iterations (3)–(6) satisfy that $J^{(t+1)} \leq J^{(t)}$, which can also be rewritten as:*

$$\begin{aligned} &\eta \sum_{v\in\mathcal{V}} \sum_{u\in B_v} \|\omega_{vu}^{(t+1)} - \omega_{vu}^{(t)}\|_2^2 + \frac{2}{\eta} \sum_{v\in\mathcal{V}} \|\alpha_v^{(t+1)} - \alpha_v^{(t)}\|_2^2 \\ &\qquad \leq \eta \sum_{v\in\mathcal{V}} \sum_{u\in B_v} \|\omega_{vu}^{(t)} - \omega_{vu}^{(t-1)}\|_2^2 + \frac{2}{\eta} \sum_{v\in\mathcal{V}} \|\alpha_v^{(t)} - \alpha_v^{(t-1)}\|_2^2. \end{aligned} \tag{23}$$

A proof of Lemma 4 can be found in [15]. Lemma 4 indicates that the combined residual always decreases over time. Since the attacker aims to break the training process, this inequality will not be satisfied when there is an attacker. Note that computing Inequality (23) requires ω_{vu} and α_v from every node, which can be achieved by a fusion center in centralized machine learning problems. However, since the learner uses a fully distributed network without a fusion center, we decentralize Inequality (23) into $|\mathcal{V}|$ distributed inequalities, for $v \in \mathcal{V}$:

$$\begin{aligned} &\eta \sum_{u\in B_v} \|\omega_{vu}^{(t+1)} - \omega_{vu}^{(t)}\|_2^2 + \frac{2}{\eta} \|\alpha_v^{(t+1)} - \alpha_v^{(t)}\|_2^2 \\ &\qquad \leq \eta \sum_{u\in B_v} \|\omega_{vu}^{(t)} - \omega_{vu}^{(t-1)}\|_2^2 + \frac{2}{\eta} \|\alpha_v^{(t)} - \alpha_v^{(t-1)}\|_2^2. \end{aligned} \tag{24}$$

Note that there is no guarantee that Inequality (24) holds based on Inequality (23). As a result, we relax the distributed inequality with a parameter $\rho > 1$, which is summarized in the following proposition.

PROPOSITION 1 *Iterations (3)–(6) satisfy that* $J_v^{(t+1)} \leq \rho J_v^{(t)}$, *where*

$$J_v^{(t)} = \eta \sum_{u \in B_v} \|\omega_{vu}^{(t)} - \omega_{vu}^{(t-1)}\|_2^2 + \frac{2}{\eta}\|\alpha_v^{(t)} - \alpha_v^{(t-1)}\|_2^2. \tag{25}$$

PROOF

Let us assume that $J_v^{(t+1)} \leq \rho J_v^{(t)}$ does not hold for $v = v_0$, we have $J_{v_0}^{(t+1)} > \rho J_{v_0}^{(t)}$ and $J_{v \neq v_0}^{(t+1)} \leq \rho J_{v \neq v_0}^{(t)}$. As a result, $J_{v_0}^{(t+1)} > \rho^{(t+1)} J_{v_0}^{(0)}$ which increases exponentially with $\rho > 1$. Since $J_v^{(t)}$ is always larger than 0, Inequality (23) will be violated eventually. Proposition 1 holds.

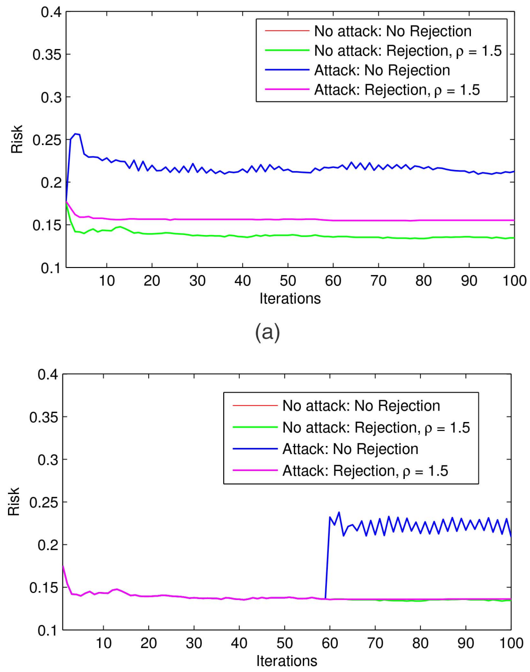
With the inequality in Proposition 1, the new DSVM algorithm with rejection method can be summarized into Algorithm 4. In Algorithm 4, if the inequality at Step 15 is satisfied, the current update will be rejected. $J_v^{(0)}$ should be set to be sufficiently large to pass the first rejection test. Numerical experiments are shown in Fig. 9, Fig. 10, and Fig. 11.

ALGORITHM 4: *DSVM with Rejection*

Randomly initialize $\mathbf{r}_v^{(0)}$, $\lambda_v^{(0)}$, $\omega_{vu}^{(0)}$ and $\alpha_v^{(0)} = \mathbf{0}_{(p+1)\times 1}$, set $J_v^{(0)}$ very large.

```
 1: for $t = 0, 1, 2, \ldots$ do
 2:     for all $v \in \mathcal{V}$ do
 3:         Compute $\lambda_v^{(t+1)}$ via (3).
 4:         Compute $\mathbf{r}_v^{(t+1)}$ via (4).
 5:     end for
 6:     for all $v \in \mathcal{V}$ do
 7:         Broadcast $\mathbf{r}_v^{(t+1)}$ to all neighbors $u \in \mathcal{B}_v$.
 8:     end for
 9:     for all $v \in \mathcal{V}$ do
10:         Compute $\omega_{vu}^{(t+1)}$ via (5).
11:         Compute $\alpha_v^{(t+1)}$ via (6).
12:         Compute $J_v^{(t+1)}$ via (25).
13:     end for
14:     for all $v \in \mathcal{V}$ do
15:         if $J_v^{(t+1)} > \rho J_v^{(t)}$
16:             $\lambda_v^{(t+1)} = \lambda_v^{(t)}$, $\mathbf{r}_v^{(t+1)} = \mathbf{r}_v^{(t)}$,
17:             $\alpha_v^{(t+1)} = \alpha_v^{(t)}$, $\omega_{vu}^{(t+1)} = \omega_v^{(t)}$,
18:             $J_v^{(t+1)} = J_v^{(t)}$.
19:         end if
20:     end for
21: end for
```

From Fig. 9, we can see that the DSVM algorithm with rejection method has a lower risk than the normal algorithm when there is an attacker. And it has the same performance when there is no attacker, which indicates that when $\rho = 1.5$, rejection method does not affect the training process. Fig. 10 and Fig. 11 show the results when $\rho = 1$ and $\rho = 100$, respectively. We can see from Fig. 10 that when $\rho = 1$, the risk is lower when there is an attacker, but convergence slows down when there is no attacker. We can see from Fig. 11 that when $\rho = 100$, the risk with rejection method is even higher than the risk of the standard algorithm, because wrong updates can still be treated as a correct update and accumulates as iteration goes.



(a)

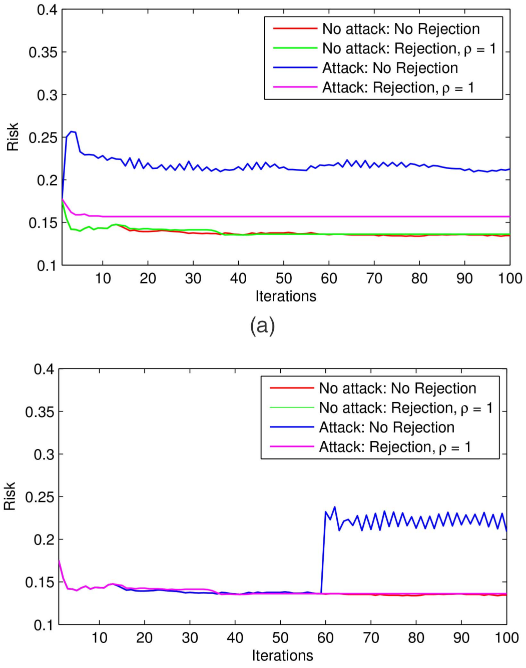
(b)

Fig. 9. Evolution of the empirical risks of ADMoM-DSVM Rejection with the attacker at a balanced network with 4 nodes of degree 0.4 on Spambase dataset [14]. Each node has 60 training samples. The attacker only attacks 1 node with $C_{1,\delta} = 10^5$ and $C_a = 0.01$. The rejection method has $\rho = 1.5$. (a) Attack starts from step 0. (b) Attack starts from step 60.

From the numerical experiments, the value of ρ is important to the rejection method. A smaller ρ may slow down the convergence of the DSVM algorithm without attacker, a larger ρ does not prevent attacks. With a properly selected ρ, the training process becomes less vulnerable to attackers.

DS 4 is suitable for a wide range of applications as wrong updates will be rejected. Comparing to DSs 1 and 2, DS 4 does not sacrifice efficiency. Comparing to DS 3, compromised nodes in DS 4 has been kept from being further damaged by the attacker. One possible drawback of DS 4 is that it may require insights of the problem to find a proper ρ.

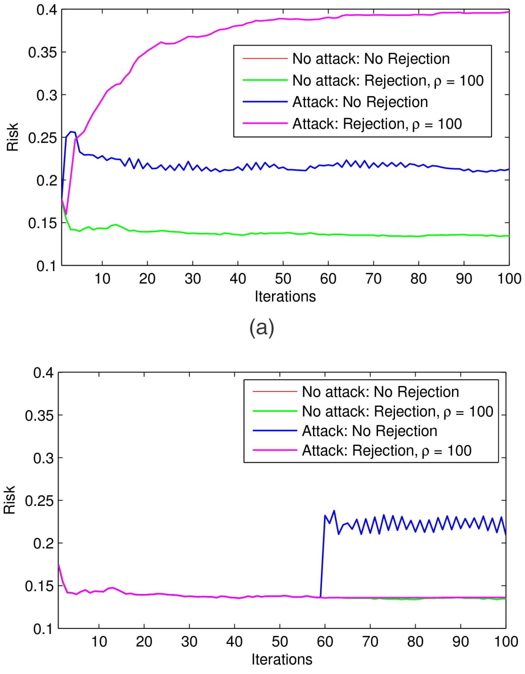
Each defense strategy is suitable for a different scenario and applications. The choice of defense strategies will depend on the applications and the constraints on the defender's actions. Though four defense strategies have their own advantages and disadvantages, a combination of all the defense strategies can be used to secure the training process of the learner.



(a)

(b)

Fig. 10. Evolution of the empirical risks of ADMoM-DSVM Rejection with the attacker at a balanced network with 4 nodes of degree 0.4 on Spambase dataset [14]. Each node has 60 training samples. The attacker only attacks 1 node with $C_{1,\delta} = 10^5$ and $C_a = 0.01$. The rejection method has $\rho = 1$. (a) Attack starts from step 0. (b) Attack starts from step 60.



(a)

(b)

Fig. 11. Evolution of the empirical risks of ADMoM-DSVM Rejection with the attacker at a balanced network with 4 nodes of degree 0.4 on Spambase dataset [14]. Each node has 60 training samples. The attacker only attacks 1 node with $C_{1,\delta} = 10^5$ and $C_a = 0.01$. The rejection method has $\rho = 100$. (a) Attack starts from step 0. (b) Attack starts from step 60.

7. CONCLUSION

Distributed support vector machines are ubiquitous but inherently vulnerable to adversaries. This paper has investigated defense strategies of DSVM against potential attackers. We have established a game-theoretic framework to capture the strategic interactions between an attacker and a learner with a network of distributed nodes. We have shown that the nonzero-sum game is strategically equivalent to a zero-sum game, and hence, its equilibrium can be characterized by a saddle-point equilibrium solution to a minimax problem. By using the technique of ADMoM, we have developed secure and resilient algorithms that can respond to the adversarial environment. We have shown that a balanced network with fewer nodes and a higher degree is less vulnerable to the attacker. Moreover, adding more training samples has been proved to reduce the vulnerability of the system. We have shown that verification method where each node verifies information from neighboring nodes can protect uncompromised nodes from receiving misleading information, but compromised nodes are also prevented from receiving correct information. We have shown that rejection method where each node rejects unreasonable updates can stop global training process from deterioration, thus wrong information is thwarted from affecting the system. One direction of future works is to extend the current framework to investigate nonlinear DSVM and other machine learning algorithms.

APPENDIX A: PROOF OF LEMMA 2

A detailed proof of Lemma 2 can be found in our previous work [34]. By using hinge loss function, we reformulate Problem (10) into the following problem:

$$\min_{\{\mathbf{w}_v, b_v\}} \max_{\{\delta_{vn}\}} \frac{1}{2} \sum_{v \in \mathcal{V}} \|\mathbf{w}_v\|_2^2$$

$$+ V_l C_l \sum_{v \in \mathcal{V}_l} \sum_{n=1}^{N_v} [1 - \mathrm{y}_{vn}(\mathbf{w}_v^T \mathbf{x}_{vn} + b_v)]_+$$

$$+ V_a C_l \sum_{v \in \mathcal{V}_a} \sum_{n=1}^{N_v} [1 - \mathrm{y}_{vn}(\mathbf{w}_v^T (\mathbf{x}_{vn} - \delta_{vn}) + b_v)]_+$$

$$- C_a \sum_{v \in \mathcal{V}_a} \sum_{n=1}^{N_v} \|\delta_{vn}\|_0$$

s.t.

$$\begin{aligned} &\mathbf{w}_v = \mathbf{w}_u,\ b_v = b_u, \quad \forall v \in \mathcal{V},\ u \in \mathcal{B}_v; \\ &(\delta_{v1}, \ldots, \delta_{vN_v}) \in \mathcal{U}_v, \quad \forall v \in \mathcal{V}_a. \end{aligned} \tag{26}$$

Similarly, Problem (11) can be reformulated into the following problem:

$$
\begin{aligned}
\min_{\{\mathbf{w}_v,b_v\}} \max_{\{\delta_v\}} & \frac{1}{2}\sum_{v\in\mathcal{V}}\|\mathbf{w}_v\|^2 \\
& + V_l C_l \sum_{v\in\mathcal{V}_l}\sum_{n=1}^{N_v}[1-\mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)]_+ \\
& + V_a C_l \sum_{v\in\mathcal{V}_a}\sum_{n=1}^{N_v}[1-\mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)]_+ \\
& + \sum_{v\in\mathcal{V}_a}(V_a C_l \mathbf{w}_v^T\delta_v - C_a\|\delta_v\|_0) \\
\text{s.t.} \quad & \mathbf{w}_v=\mathbf{w}_u,\ b_v=b_u,\quad \forall v\in\mathcal{V},\ u\in\mathcal{B}_v; \\
& \delta_v\in\mathcal{U}_{v0},\qquad\qquad\quad \forall v\in\mathcal{V}_a.
\end{aligned} \tag{27}
$$

As a result, we only need to prove that problem (26) is equivalent to problem (27). Since both of problems are min-max problems with the same variables, we only need to prove that we minimize the same maximization problem. Moreover, since $\{\delta_{vn}\}$ is independent in the maximization part of (26), and δ_v is independent in the maximization part of (27), we can separate maximization problem into V_a sub-maximization problems, and solving the sub-problems is equivalent to solving the global maximization problem. As a result, we only need to show that the following sub-problem

$$
\begin{aligned}
\max_{\{\delta_{vn}\}\in\mathcal{U}_v} S(\{\delta_{vn}\}) \triangleq\ & V_a C_l \sum_{n=1}^{N_v}[1-\mathrm{y}_{vn}(\mathbf{w}_v^T(\mathbf{x}_{vn}-\delta_{vn})+b_v)]_+ \\
& - C_a\sum_{n=1}^{N_v}\|\delta_{vn}\|_0
\end{aligned} \tag{28}
$$

is equivalent to the following sub-problem

$$
\begin{aligned}
\max_{\delta_v\in\mathcal{U}_{v0}} & V_a C_l \sum_{n=1}^{N_v}[1-\mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)]_+ \\
& + V_a C_l \mathbf{w}_v^T\delta_v - C_a\|\delta_v\|_0.
\end{aligned} \tag{29}
$$

We adopt the similar proof in [32], recall the properties of sublinear aggregated action set, $\mathcal{U}_v^-\subseteq\mathcal{U}_v\subseteq\mathcal{U}_v^+$, where

$$
\mathcal{U}^-\triangleq\bigcup_{t=1}^{n}\mathcal{U}_t^-,\ \mathcal{U}_t^-\triangleq\left\{(\delta_1,\ldots,\delta_n)\,\middle|\,\begin{array}{l}\delta_t\in\mathcal{U}_0;\\ \delta_i=\mathbf{0},\ i\neq t.\end{array}\right\};
$$

$$
\mathcal{U}^+\triangleq\left\{(\alpha_1\delta_1,\ldots,\alpha_n\delta_n)\,\middle|\,\begin{array}{l}\sum_{i=1}^{n}\alpha_i=1;\ \alpha_i\geq 0,\\ \delta_i\in\mathcal{U}_0,\ i=1,\ldots,n\end{array}\right\}.
$$

Hence, fixing any $(\mathbf{w}_v,b_v)\in\mathbb{R}^{p+1}$, we have the following inequalities:

$$
\max_{\{\delta_{vn}\}\in\mathcal{U}_v^-} S(\{\delta_{vn}\}) \leq \max_{\{\delta_{vn}\}\in\mathcal{U}_v} S(\{\delta_{vn}\}) \leq \max_{\{\delta_{vn}\}\in\mathcal{U}_v^+} S(\{\delta_{vn}\}) \tag{30}
$$

We can show that (29) is no larger than the leftmost term and no smaller than the rightmost term [34]. Thus, the equivalence between (28) and (29) holds. Hence, Lemma 2 holds.

APPENDIX B: PROOF OF LEMMA 3

We use best response dynamics to construct the best response for the min-problem and max-problem separately. The min-problem and max-problem are achieved by fixing $\{\mathbf{r}_v,\xi_v\}$ and $\{\delta_v\}$, respectively. For fixed $\{\mathbf{r}_v^*,\xi_v^*\}$,

$$
\begin{aligned}
& \delta_v^*\in\arg\max_{\{\delta_v\}}\sum_{v\in\mathcal{V}_a}(V_a C_l \mathbf{r}_v^{*T}(\mathbf{I}_{p+1}-\Pi_{p+1})\delta_v - C_a\|\delta_v\|_0) \\
& \text{s.t.}\quad \delta_v\in\mathcal{U}_{v0},\ \forall v\in\mathcal{V}_a.
\end{aligned} \tag{31}
$$

We relax l_0 norm to l_1 norm to represent the cost function of the attacker. By writing the dual form of the l_1 norm, we arrive at

$$
\begin{aligned}
\delta_v^*\in\arg & \max_{\{\delta_v,s_v\}} V_a C_l \mathbf{r}_v^{*T}(\mathbf{I}_{p+1}-\Pi_{p+1})\delta_v - \mathbf{1}^T s_v \\
& C_a\delta_v\leq s_v, \\
\text{s.t.}\quad & C_a\delta_v\geq -s_v, \\
& \delta_v\in\mathcal{U}_{v0}.
\end{aligned} \tag{32}
$$

For fixed $\{\delta_v^*\}$, we have

$$
\begin{aligned}
\min_{\{\mathbf{r}_v,\omega_{vu},\xi_v\}} & \frac{1}{2}\sum_{v\in\mathcal{V}}\mathbf{r}_v^T(\mathbf{I}_{p+1}-\Pi_{p+1})\mathbf{r}_v \\
& + V_a C_l \sum_{v\in\mathcal{V}_a}\mathbf{r}_v^T(\mathbf{I}_{p+1}-\Pi_{p+1})\delta_v^* + V C_l \sum_{v\in\mathcal{V}}\mathbf{1}_v^T\boldsymbol{\xi}_v \\
& \mathbf{Y}_v\mathbf{X}_v\mathbf{r}_v\geq\mathbf{1}_v-\boldsymbol{\xi}_v,\qquad \forall v\in\mathcal{V}; \\
\text{s.t.}\quad & \boldsymbol{\xi}_v\geq\mathbf{0}_v,\qquad\qquad\quad \forall v\in\mathcal{V}; \\
& \mathbf{r}_v=\omega_{vu},\ \omega_{vu}=\mathbf{r}_u,\quad \forall v\in\mathcal{V},\ \forall u\in\mathcal{B}_u.
\end{aligned} \tag{33}
$$

Note that term $-C_a\|\delta_v^*\|_0$ is removed since it does not play a role in the minimization problem. Based on (32) and (33), we have the method of solving Problem (12) as follows, first step is that we randomly pick initial $\{\mathbf{r}_v^{(0)},\delta_v^{(0)}\}$, and then we solve Max-problem (32) with $\{\mathbf{r}_v^{(0)}\}$ to obtain $\{\delta_v^{(1)}\}$. In next step, we solve Min-problem (33) to obtain $\{\mathbf{r}_v^{(1)}\}$ with $\{\delta_v^{(1)}\}$ from the previous step. We repeat solving the max-problem with $\{\mathbf{r}_v^{(t-1)}\}$ and solving the min-problem with $\{\delta_v^{(t)}\}$ until convergence. Furthermore, we use the alternating direction method of multipliers (ADMoM) to solve Problem (33).

The ADMoM is a distributed optimization algorithm solving the following problem:

$$
\begin{aligned}
& \min_{\mathbf{r},\omega} f(\mathbf{r})+g(\omega) \\
& \text{s.t.}\quad \mathbf{M}\mathbf{r}=\omega,
\end{aligned} \tag{34}
$$

where f and g are convex functions [12].

The augmented Lagrangian corresponding to (34) is

$$L(\mathbf{r},\omega,\alpha) = f(\mathbf{r}) + g(\omega) + \alpha^T(\mathbf{Mr} - \omega) + \frac{\eta}{2}\|\mathbf{Mr} - \omega\|^2, \tag{35}$$

where α denotes the Lagrange multiplier.

Then, the ADMoM solves problem (34) by the update rules below:

$$\mathbf{r}^{(t+1)} \in \arg\min_{\mathbf{r}} L(\mathbf{r},\omega^{(t)},\alpha^{(t)}); \tag{36}$$

$$\omega^{(t+1)} \in \arg\min_{\omega} L(\mathbf{r}^{(t+1)},\omega,\alpha^{(t)}); \tag{37}$$

$$\alpha^{(t+1)} = \alpha^{(t)} + \eta(\mathbf{Mr}^{(t+1)} - \omega^{(t+1)}). \tag{38}$$

The objective here is to transform Problem (33) into the form of (34), and then we can solve Problem (33) by iterations (36), (37), and (38). We adopt a similar method in [13], which leads to the following result.

REMARK 1 Each node iterates $\lambda_v^{(t)}, \mathbf{r}_v^{(t)}$ and $\alpha_v^{(t)}$, given by

$$\begin{aligned}\lambda_v^{(t+1)} \in \arg\max_{\mathbf{0}\le\boldsymbol{\lambda}_v\le VC_l\mathbf{1}_v} & -\tfrac{1}{2}\lambda_v^T\mathbf{Y}_v\mathbf{X}_v\mathbf{U}_v^{-1}\mathbf{X}_v^T\mathbf{Y}_v\lambda_v \\ & + (\mathbf{1}_v + \mathbf{Y}_v\mathbf{X}_v\mathbf{U}_v^{-1}\mathbf{f}_v^{(t)})^T\lambda_v, \end{aligned} \tag{39}$$

$$\mathbf{r}_v^{(t+1)} = \mathbf{U}_v^{-1}(\mathbf{X}_v^T\mathbf{Y}_v\lambda_v^{(t+1)} - \mathbf{f}_v^{(t)}), \tag{40}$$

$$\omega_{vu}^{(t+1)} = \tfrac{1}{2}(\mathbf{r}_v^{(t+1)} + \mathbf{r}_u^{(t+1)}), \tag{41}$$

$$\alpha_v^{(t+1)} = \alpha_v^{(t)} + \frac{\eta}{2}\sum_{u\in\mathcal{B}_v}[\mathbf{r}_v^{(t+1)} - \mathbf{r}_u^{(t+1)}], \tag{42}$$

where $\mathbf{U}_v = (\mathbf{I}_{p+1} - \Pi_{p+1}) + 2\eta|\mathcal{B}_v|\mathbf{I}_{p+1}$, $\mathbf{f}_v^{(t)} = V_aC_l\delta_v^*$ $+2\alpha_v^{(t)} - 2\eta\sum_{u\in\mathcal{B}_v}\omega_{vu}^{(t)}$.

By combining the above remark with Problem (32), we can obtain Lemma 3.

REFERENCES

[1] T. P. Banerjee and S. Das
Multi-sensor data fusion using support vector machine for motor fault detection
Information Sciences, 217 (Dec. 2012), pp. 96–107.

[2] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar
Can machine learning be secure?
in Proceedings of the 2006 ACM Symposium on Information, computer and communications security, ACM, Mar. 2006, pp. 16–25.

[3] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli
Evasion attacks against machine learning at test time
in Machine Learning and Knowledge Discovery in Databases, Springer, Sep. 2013, pp. 387–402.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein
Distributed optimization and statistical learning via the alternating direction method of multipliers
Foundations and Trends® in Machine Learning, 3 (July 2011), pp. 1–122.

[5] H. Chan and A. Perrig
Security and privacy in sensor networks
Computer, 36 (Oct. 2003), pp. 103–105.

[6] R. Chen, J.-M. Park, and K. Bian
Robust distributed spectrum sensing in cognitive radio networks
in INFOCOM 2008. The 27th Conference on Computer Communications. IEEE, IEEE, Apr. 2008.

[7] X. Chen, K. Makki, K. Yen, and N. Pissinou
Sensor network security: a survey
IEEE Communications Surveys & Tutorials, 11 (Apr. 2009).

[8] Z. Chen, L. Gao, and K. Kwiat
Modeling the spread of active worms
in INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, vol. 3, IEEE, Mar. 2003, pp. 1890–1900.

[9] M. Dalponte, L. Bruzzone, and D. Gianelle
Fusion of hyperspectral and lidar remote sensing data for classification of complex forest areas
IEEE Transactions on Geoscience and Remote Sensing, 46 (May 2008), pp. 1416–1427.

[10] J.-x. Dong, A. Krzyzak, and C. Y. Suen
Fast svm training algorithm with decomposition on very large data sets
IEEE transactions on pattern analysis and machine intelligence, 27 (Apr. 2005), pp. 603–618.

[11] H. F. Durrant-Whyte, B. Rao, and H. Hu
Toward a fully decentralized architecture for multi-sensor data fusion
in Robotics and Automation, 1990. Proceedings, 1990 IEEE International Conference on, IEEE, May 1990, pp. 1331–1336.

[12] J. Eckstein and W. Yao
Augmented lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results
RUTCOR Research Reports, 32 (Dec. 2012), p. 3.

[13] P. A. Forero, A. Cano, and G. B. Giannakis
Consensus-based distributed support vector machines
Journal of Machine Learning Research, 11 (2010), pp. 1663–1707.

[14] A. Frank and A. Asuncion
UCI machine learning repository [http://archive. ics. uci. edu/ml]. irvine, ca: University of california
School of Information and Computer Science, 213 (2010).

[15] B. He and X. Yuan
On non-ergodic convergence rate of douglas-rachford alternating direction method of multipliers
Numerische Mathematik, 130 (July 2012), pp. 567–577.

[16] J. Hert, P. Willett, D. J. Wilton, P. Acklin, K. Azzaoui, E. Jacoby, and A. Schuffenhauer
New methods for ligand-based virtual screening: use of data fusion and machine learning to enhance the effectiveness of similarity searching
Journal of chemical information and modeling, 46 (Mar. 2006), pp. 462–470.

[17] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar
Adversarial machine learning
in Proceedings of the 4th ACM workshop on Security and artificial intelligence, ACM, Oct. 2011, pp. 43–58.

[18] W. Jiang, Z.-h. Tian, H.-l. Zhang, and X.-f. Song
A stochastic game theoretic approach to attack prediction and optimal active defense strategy decision
in Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on, IEEE, Apr. 2008, pp. 648–653.

[19] M. Kantarcioglu, B. Xi, and C. Clifton
A game theoretical framework for adversarial learning
in CERIAS 9th Annual Information Security Symposium, Citeseer, 2008.

[20] K. K. Khedo, R. Perseedoss, A. Mungur, et al.
A wireless sensor network air pollution monitoring system
arXiv preprint arXiv:1005.1737, (May 2010).

[21] W. Liu and S. Chawla
A game theoretical model for adversarial learning
in Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on, IEEE, Dec. 2009, pp. 25–30.

[22] M. H. Manshaei, Q. Zhu, T. Alpcan, T. Bacşar, and J.-P. Hubaux
Game theory meets network security and privacy
ACM Computing Surveys (CSUR), 45 (June 2013), p. 25.

[23] A. Navia-Vázquez and E. Parrado-Hernandez
Distributed support vector machines
IEEE Transactions on Neural Networks, 17 (July 2006), pp. 1091–1097.

[24] J. B. Predd, S. R. Kulkarni, and H. V. Poor
Distributed learning in wireless sensor networks
John Wiley & Sons: Chichester, UK, Oct. 2007.

[25] C. E. Shannon
Communication theory of secrecy systems
Bell Labs Technical Journal, 28 (Oct. 1949), pp. 656–715.

[26] D. Shen, G. Chen, E. Blasch, and G. Tadda
Adaptive markov game theoretic data fusion approach for cyber network defense
in Military Communications Conference, 2007. MILCOM 2007. IEEE, IEEE, Oct. 2007, pp. 1–7.

[27] P. Tague and R. Poovendran
Modeling node capture attacks in wireless sensor networks
in Communication, Control, and Computing, 2008 46th Annual Allerton Conference on, IEEE, Sep. 2008, pp. 1221–1224.

[28] I. W. Tsang, J. T. Kwok, and P.-M. Cheung
Core vector machines: Fast svm training on very large data sets
Journal of Machine Learning Research, 6 (2005), pp. 363–392.

[29] D. Wang and Y. Zhou
Distributed support vector machines: An overview
in Control and Decision Conference (CCDC), 2012 24th Chinese, IEEE, May 2012, pp. 3897–3901.

[30] B. Waske and J. A. Benediktsson
Fusion of support vector machines for classification of multisensor data
IEEE Transactions on Geoscience and Remote Sensing, 45 (Dec. 2007), pp. 3858–3866.

[31] G. Wu, Y. Wu, L. Jiao, Y.-F. Wang, and E. Y. Chang
Multi-camera spatio-temporal fusion and biased sequence-data learning for security surveillance
in Proceedings of the eleventh ACM international conference on Multimedia, ACM, Nov. 2003, pp. 528–538.

[32] H. Xu, C. Caramanis, and S. Mannor
Robustness and regularization of support vector machines
Journal of Machine Learning Research, 10 (2009), pp. 1485–1510.

[33] R. Zhang and Q. Zhu
A game-theoretic defense against data poisoning attacks in distributed support vector machines
in Decision and Control (CDC), 2017 IEEE 56th Annual Conference on, IEEE, Dec. 2017, pp. 4582–4587.

[34] ———
Secure and resilient distributed machine learning under adversarial environments
in Information Fusion (Fusion), 2015 18th International Conference on, IEEE, July 2015, pp. 644–651.

[35] ———
Student research highlight: Secure and resilient distributed machine learning under adversarial environments
IEEE Aerospace and Electronic Systems Magazine, 31 (Mar. 2016), pp. 34–36.

[36] ———
A game-theoretic analysis of label flipping attacks on distributed support vector machines
in Information Sciences and Systems (CISS), 2017 51st Annual Conference on, IEEE, Mar. 2017, pp. 1–6.

[37] ———
A game-theoretic approach to design secure and resilient distributed support vector machines
IEEE Transactions on Neural Networks and Learning Systems, (Mar. 2018).

[38] R. Zhang, Q. Zhu, and Y. Hayel
A bi-level game approach to attack-aware cyber insurance of computer networks
IEEE Journal on Selected Areas in Communications, 35 (Mar. 2017), pp. 779–794.

[39] X.-H. Zhao, W. Gang, K.-K. Zhao, and D.-J. Tan
On-line least squares support vector machine algorithm in gas prediction
Mining Science and Technology (China), 19 (Mar. 2009), pp. 194–198.

[40] Q. Zhu and T. Basar
Game-theoretic methods for robustness, security, and resilience of cyberphysical control systems: games-in-games principle for optimal cross-layer resilient control systems
IEEE control systems, 35 (Feb. 2015), pp. 46–65.

[41] Q. Zhu and T. Başar
Game-theoretic approach to feedback-driven multi-stage moving target defense
in International Conference on Decision and Game Theory for Security, Springer, Nov. 2013, pp. 246–263.

[42] Q. Zhu, H. Tembine, and T. Başar
Heterogeneous learning in zero-sum stochastic games with incomplete information
in Decision and Control (CDC), 2010 49th IEEE Conference on, IEEE, Dec. 2010, pp. 219–224.

[43] ———
Distributed strategic learning with application to network security
in American Control Conference (ACC), 2011, IEEE, June 2011, pp. 4057–4062.

Rui Zhang received the B.S. degree in optical information science and technology from Wuhan University in 2014, and the M.S. degree in electrical engineering from New York University in 2016, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering. He won the Runner Up Best Student Award at the International Conference on Information Fusion 2015. His research interests include cyber-insurance, network security, machine learning, optimal transport, and cyber-physical systems.

Quanyan Zhu received B. Eng. in Honors Electrical Engineering from McGill University in 2006, M.A.Sc. from University of Toronto in 2008, and Ph.D. from the University of Illinois at Urbana-Champaign (UIUC) in 2013. After stints at Princeton University, he is currently an assistant professor at the Department of Electrical and Computer Engineering, New York University. He is a recipient of many awards including NSERC Canada Graduate Scholarship (CGS), Mavis Future Faculty Fellowships, and NSERC Postdoctoral Fellowship (PDF). He spearheaded and chaired INFOCOM Workshop on Communications and Control on Smart Energy Systems (CCSES), and Midwest Workshop on Control and Game Theory (WCGT). His current research interests include Internet of things, cyber-physical systems, security and privacy, game theory, and system and control.