# Secure and Resilient Distributed Machine Learning Under Adversarial Environments

Rui Zhang and Quanyan Zhu

***Abstract*—With a large number of sensors and control units in networked systems, the decentralized computing algorithms play a key role in scalable and efficient data processing for detection and estimation. The well-known algorithms are vulnerable to adversaries who can modify and generate data to deceive the system to misclassify or misestimate the information from the distributed data processing. This work aims to develop secure, resilient and distributed machine learning algorithms under adversarial environment. We establish a game-theoretic framework to capture the conflicting interests between the adversary and a set of distributed data processing units. The Nash equilibrium of the game allows predicting the outcome of learning algorithms in adversarial environment, and enhancing the resilience of the machine learning through dynamic distributed learning algorithms. We use Spambase Dataset to illustrate and corroborate our results.**

## I. Introduction

Machine learning algorithms have been widely used in multiple application domains including spam filtering, pattern recognition, search engines and multimedia. However, well-known algorithms such as Kalman filtering [1], SVMs [2] and PCAs [3], are generally vulnerable to adversaries who can modify and generate data to deceive the system to misclassify or misestimate the information from the distributed data processing. For example, an attacker can strategically craft training data for the spam filter to increase its misclassification rates and evade spam detectors [4]. In addition, an attacker can change the measurement of a subset of sensors to mislead the fusion center to a wrong conclusion of the environment [5] or stealthily bypass the alert system [6].

The security of the machine learning algorithms will be exacerbated when the computations become decentralized across a large-scale network of distributed sensor or control units. An adversary can launch a number of cyber attacks, such as node capture [7] or replication attack [8], to compromise and take over nodes in the network. Despite the fact that distributed computation provides scalable and efficient data processing, the vulnerability of machine learning algorithms on each node will lead to unanticipated consequence at a larger scale.

Hence, it is imperative to address these security concerns by developing secure and resilient distributed learning algorithms. In this paper, we focus on a class of support vector machines algorithms, and aim to establish a game-theoretic framework to capture the conflicting interests between the adversary and a set of distributed data processing units. 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 of misclassification, while an attacker seeks to strategically modify the training data and maximize the error constrained by its computational capabilities.

R. Zhang and Q. Zhu are with Department of Electrical and Computer Engineering, New York University, Brooklyn, NY, 11201 E-mail:{rz885,qz494}@nyu.edu

The game formulation of the security problem enables a formal analysis of the impact of the machine learning algorithm in adversarial environment. The Nash equilibrium of the game allows the prediction of the outcome, and yields optimal response strategies 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 distributed SVMs.

Our work intersects the research areas on game theory, cyber security and machine learning. This research is closely related to the distributed support vector machines proposed in [9]. We leverage a similar consensus-based approach to develop decentralized machine learning algorithms. Our work is also related to a recent body of work on the security of machine learning, e.g. [10]–[13]. In [10], Barreno et. al. presents a taxonomy of attacks on machine learning algorithms. In [13], Liu et. al. introduces a Stackelberg game to model interaction between the adversary and the learner. The major focus on their work is on centralized machine learning tools. In this work, we extend the security framework to a distributed framework in which the security issue is aggravated by the distributed nature of the system, and the constraints from the network topology.

The major contribution of this work can be summarized as follows:

1) We capture the attacker's objective and constrained capabilities in a game-theoretic framework, and develop a non-zero-sum game to model the strategic interactions between an attacker and a learner with a distributed set of nodes.
2) We fully characterize the Nash equilibrium by showing the strategic equivalence between the original nonzero-sum game and a zero-sum game.
3) We develop secure and resilient distributed algorithms based on alternating direction method of multipliers (ADMoM) [15]. Each node communicates with its neighboring nodes, and updates its decision strategically in response to adversarial environment.
4) We demonstrate that network topology plays an important role in resilience to adversary behaviors. Networks with less nodes and higher connectivity are shown to be more resilient.

The rest of this paper is organized as follows. Section 2 outlines the distributed support vector designs. 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. Finally, Section

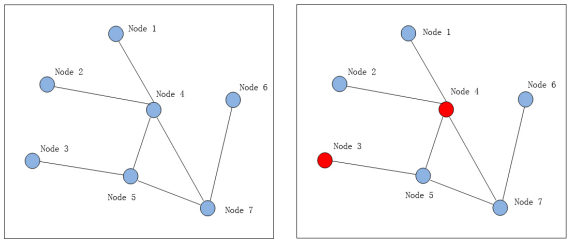
5 and Section 6 present numerical results and concluding remarks.

### A. Summary of Notations

Notions in this paper are summarized as follows. Boldface letters are used for matrices (column vectors); $(\cdot)^T$ denotes matrix and vector transposition; $[\cdot]_{vu}$ denotes the $vu$-th entry of a matrix; $diag(\mathbf{x})$ is the diagonal matrix with $\mathbf{x}$ on its main diagonal; $\|\cdot\|$ is the norm of the matrix or vector; $\mathcal{U}$ denotes the action set which is used by the attacker.

## II. Preliminaries and Problem Statement

In this section, we present a two-player machine learning game in a distributed network involving a learner and an attacker to capture the strategic interactions between them. The network is modeled by an undirected graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ with $\mathcal{V} := \{1,...,V\}$ representing the set of nodes, and $\mathcal{E}$ representing the set of links between nodes. Node $v \in \mathcal{V}$ only communicates 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 $\mathcal{D}_v := \{(\mathbf{x}_{vn}, y_{vn}) : n = 1,...,N_v\}$ 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). An attacker can compromise a set of nodes in the network and modify their training dataset. Fig. 1(b) shows two nodes, 3 and 4, are controlled by the attacker.




(a) Network example without an attacker (b) Network example with an attacker

Fig. 1. Network example: There are 7 nodes in this network. Each node contains a labelled training set $\mathcal{D}_v := \{(\mathbf{x}_{vn}, y_{vn}) : n = 1,...,N_v\}$. Node 4 can communicate with its 4 neighbors: node 1, 2, 5 and 7. An attacker can take over a subset of the nodes in the network as seen in (b). The compromised nodes are marked in red.

The goal of the learner is to design distributed SVM algorithms for each node in the network based on its local training data. To achieve this, the learner aims to find a maximum-margin linear discriminant function $g_v(\mathbf{x}) = \mathbf{x}^T\mathbf{w}_v^* + b_v^*$ 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. This approach enables each node to classify any new input vector $\mathbf{x}$ to one of the two classes $\{+1,-1\}$ without communicating $\mathcal{D}_v$ to other nodes $v' \neq v$. 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:

$$\begin{array}{ll} \min\limits_{\{\mathbf{w}_v, b_v, \{\xi_{vn}\}\}} & \frac{1}{2}\sum\limits_{v=1}^{V}\|\mathbf{w}_v\|_2^2 + VC_l\sum\limits_{v=1}^{V}\sum\limits_{n=1}^{N_v}\xi_{vn} \\ \text{s.t.} & \mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn} + b_v) \geq 1 - \xi_{vn}, \quad \forall v \in \mathcal{V}, n = 1,...,N_v; \\ & \xi_{vn} \geq 0, \qquad\qquad\qquad\quad \forall v \in \mathcal{V}, n = 1,...,N_v; \\ & \mathbf{w}_v = \mathbf{w}_u, b_v = b_u, \qquad\quad \forall v \in \mathcal{V}, u \in \mathcal{B}_v. \end{array} \tag{1}$$

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

On the other hand, an attacker takes over a set of nodes with the aim of breaking the training process of learner. We assume that the attacker has the complete knowledge of the learner's problem (1), and he can modify the value $\mathbf{x}_{vn}$ of the node $v$ into $\widehat{\mathbf{x}}_{vn} = \mathbf{x}_{vn} - \delta_{vn}$, where $\delta_{vn} \in \mathcal{U}$, and $\mathcal{U}$ is the attacker's action set which is described in detail in Definition 1 and 2, [14].

**Definition 1** *An atomic action set $\mathcal{U}_0 \subseteq \mathbb{R}^p$ has the following two properties.*

(I) $\mathbf{0} \in \mathcal{U}_0$;

(II) For any $\mathbf{w}_0 \in \mathbb{R}^p$ :
$\max\limits_{\delta\in\mathcal{U}_0}\left[\mathbf{w}_0^T\delta\right] = \max\limits_{\delta'\in\mathcal{U}_0}\left[-\mathbf{w}_0^T\delta'\right] < +\infty.$

The first property states that the attacker can choose not to change the value of $\mathbf{x}_{vn}$. The second property states that the atomic action set is bounded and symmetric. Here, "bounded" means that the attacker has the limit on changing $\mathbf{x}_{vn}$. It is reasonable since changing the value too much will result in the evident recognition by the learner. In particular, all norm balls and ellipsoids centered at the origin are atomic action sets. Furthermore, we further specify a class of sublinear aggregated action sets for the the attacker, which will be convenient to characterize the attacker's behavior.

**Definition 2** *A set $\mathcal{U} \subseteq \mathbb{R}^{p\times n}$ is a sublinear aggregated action set of an atomic action set $\mathcal{U}_0$, if $\mathcal{U}^- \subseteq \mathcal{U} \subseteq \mathcal{U}^+$, where*

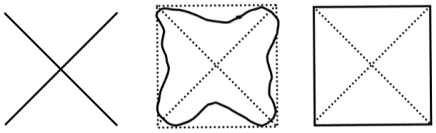
$$\mathcal{U}^- \triangleq \bigcup_{t=1}^{n}\mathcal{U}_t^-, \mathcal{U}_t^- \triangleq \left\{(\delta_1,...,\delta_n) \;\middle|\; \begin{array}{l}\delta_t \in \mathcal{U}_0; \\ \delta_i = \mathbf{0}, i \neq t.\end{array}\right\};$$

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

Figure 2 provides an example of sublinear aggregated action sets. Without loss of generality, the sublinear aggregated action set we used in this paper takes the form of

$$\mathcal{U} = \left\{(\delta_1,...,\delta_n) \;\middle|\; \sum_{i=1}^{n}\|\delta_i\| \leq C_\delta\right\}$$

, which has the atomic action set $\mathcal{U}_0 = \{\delta \,|\|\delta\| \leq C_\delta\}$.



(a) $\mathcal{U}^-$ (b) $\mathcal{U}$ (c) $\mathcal{U}^+$

Fig. 2. Illustration of a sublinear aggregated action set $\mathcal{U}$

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 $\widehat{\mathbf{x}}_{vn} \in \widehat{\mathcal{X}}$, the learner will find the discriminant function that separates data in $\widehat{\mathcal{X}}$ more accurate, rather than 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 high accuracy, the attacker seeks to lower the accuracy, we will capture the conflicting goals of the players in a game-theoretic framework.

## III. Distributed Support Vector Machines with Adversary

Before stating the game problem, we first reformulate the learner's problem (1) into an equivalent problem as follows:

$$\begin{aligned}\min_{\{\mathbf{w}_v,b_v\}} \tfrac{1}{2}\sum_{v=1}^{V}\|\mathbf{w}_v\|^2 \\ + VC_l\sum_{v=1}^{V}\sum_{n=1}^{N_v}\left[1-\mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)\right]_+ \\ \text{s.t.}\quad \mathbf{w}_v=\mathbf{w}_u,\quad b_v=b_u,\quad \forall v\in\mathcal{V}, u\in B_v.\end{aligned} \tag{2}$$

In the above problem, the term $\left[1-\mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)\right]_+ := \max[1-\mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v),0]$ is the hinge loss function, which captures the constraints $\mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)\geq 1-\xi_{vn}$ and $\xi_{vn}\geq 0$ which are related to $\xi_{vn}$ in problem (1).

Optimization problem (2) is formed by the learner who seeks to find the maximum-margin linear discriminant function. Since an attacker takes over a set of nodes $\mathcal{V}_a := \{1,...,V_a\}$ and changes $\mathbf{x}_{vn}$ into $\widehat{\mathbf{x}}_{vn}=\mathbf{x}_{vn}-\delta_{vn}$. We use $\mathcal{V}_l=\{1,...,V_l\}$ to represent nodes without the attacker. Note that, $V=V_a+V_l$ and $\mathcal{V}=\mathcal{V}_l\cup\mathcal{V}_a$. The behavior of the learner can be captured by the following optimization problem:

$$\begin{aligned}\min_{\{\mathbf{w}_v,b_v\}} \tfrac{1}{2}\sum_{v=1}^{V}\|\mathbf{w}_v\|^2 \\ + V_lC_l\sum_{v=1}^{V_l}\sum_{n=1}^{N_v}\left[1-\mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)\right]_+ \\ + V_aC_l\sum_{v=1}^{V_a}\sum_{n=1}^{N_v}\left[1-\mathrm{y}_{vn}(\mathbf{w}_v^T(\mathbf{x}_{vn}-\delta_{vn})+b_v)\right]_+ \\ \text{s.t.}\quad \mathbf{w}_v=\mathbf{w}_u,\quad b_v=b_u,\quad \forall v\in\mathcal{V}, u\in\mathcal{B}_v.\end{aligned} \tag{3}$$

By minimizing the objective function in problem (3), 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:

$$\begin{aligned}\max_{\{\delta_{vn}\}} \tfrac{1}{2}\sum_{v=1}^{V}\|\mathbf{w}_v\|^2 \\ + V_lC_l\sum_{v=1}^{V_l}\sum_{n=1}^{N_v}\left[1-\mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)\right]_+ \\ + V_aC_l\sum_{v=1}^{V_a}\sum_{n=1}^{N_v}\left[1-\mathrm{y}_{vn}(\mathbf{w}_v^T(\mathbf{x}_{vn}-\delta_{vn})+b_v)\right]_+ \\ - C_a\sum_{v=1}^{V_a}\sum_{n=1}^{N_v}\|\delta_{vn}\|_0. \\ \text{s.t.}\ (\delta_{v1},...,\delta_{vN_v})\in\mathcal{U}_v,\quad \forall v\in\mathcal{V}_a.\end{aligned} \tag{4}$$

In above problem, the term $C_a\sum_{v=1}^{V_a}\sum_{n=1}^{N_v}\|\delta_{vn}\|_0$ 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 find a minimum set of training data in a node.

The problem (3) and problem (4) can constitute a two-person nonzero-sum game between an attacker and a learner. The solution to the game problem is often 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 (3) with problem (4), 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 maximin form as follows:

$$\begin{aligned}\min_{\{\mathbf{w}_v,b_v\}}\max_{\{\delta_{vn}\}} K\left(\{\mathbf{w}_v,b_v\},\{\delta_{vn}\}\right)\triangleq \tfrac{1}{2}\sum_{v=1}^{V}\|\mathbf{w}_v\|^2 \\ + V_lC_l\sum_{v=1}^{V_l}\sum_{n=1}^{N_v}\left[1-\mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)\right]_+ \\ + V_aC_l\sum_{v=1}^{V_a}\sum_{n=1}^{N_v}\left[1-\mathrm{y}_{vn}(\mathbf{w}_v^T(\mathbf{x}_{vn}-\delta_{vn})+b_v)\right]_+ \\ - C_a\sum_{v=1}^{V_a}\sum_{n=1}^{N_v}\|\delta_{vn}\|_0 \\ \text{s.t.}\quad \mathbf{w}_v=\mathbf{w}_u, b_v=b_u, \qquad \forall v\in\mathcal{V}, u\in\mathcal{B}_v; \\ (\delta_{v1},...,\delta_{vN_v})\in\mathcal{U}_v, \qquad \forall v\in\mathcal{V}_a.\end{aligned} \tag{5}$$

Note that there are two sets of constraints: The first one $\mathbf{w_v}=\mathbf{w_u}, b_v=b_u, \forall v\in\mathcal{V}, u\in\mathcal{B}_v$, only contributes to the minimization part of the problem, while the second one $(\delta_{v1},\cdots,\delta_{vN_v})\in\mathcal{U}_v, \forall v\in\mathcal{V}_a$, only affects the maximization part. The first term of $K\left(\{\mathbf{w}_v,b_v\},\{\delta_{vn}\}\right)$ is the inverse of the distance of margin. The second term is the error penalty of nodes without attacker. The third term is the error penalty of nodes with attacker, and the last term is the cost function for the attacker. 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 larger error penalty and a small cost of the attacker. As a

result, solving problem (5) can be understood as finding the saddle point of the zero-sum game between the attacker and the learner.

Based on the property of sublinear aggregated action set and atomic action set, Problem (5) can be further simplified as stated in the following Proposition 1.

**Proposition 1** *Assume $\mathcal{U}_v$ is a sublinear aggregated action set with corresponding atomic action set $\mathcal{U}_{v0}$. Then, problem (5) is equivalent to the following optimization problem:*

$$\begin{aligned}\min_{\{\mathbf{w}_v,b_v\}}\max_{\{\delta_v\}}\ &\tfrac{1}{2}\sum_{v=1}^{V}\|\mathbf{w}_v\|^2+VC_l\sum_{v=1}^{V}\sum_{n=1}^{N_v}\xi_{vn}\\ &+\sum_{v=1}^{V_a}\left(V_aC_l\mathbf{w}_v^T\delta_v-C_a\|\delta_v\|_0\right)\\ \text{s.t.}\quad &\mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)\geq 1-\xi_{vn}, && \forall v\in\mathcal{V}, n=1,...,N_v;\\ &\xi_{vn}\geq 0, && \forall v\in\mathcal{V}, n=1,...,N_v;\\ &\mathbf{w}_v=\mathbf{w}_u, b_v=b_u, && \forall v\in\mathcal{V}, u\in\mathcal{B}_v;\\ &\delta_v\in\mathcal{U}_{v0}, && \forall v\in\mathcal{V}_a.\end{aligned} \tag{6}$$

**Proof**. See Appendix A.

## IV. ADMoM-DSVM and Distributed Algorithm

In the previous section, we have combined problem (3) for the learner with problem (4) for the attacker into one min-max problem (5), and showed its equivalence to problem (6). In this section, we will develop iterative algorithms to solve problem (6).

Firstly, we define $\mathbf{r}_v := [\mathbf{w}_v^T, b_v]^T$, the augmented matrix $\mathbf{X}_v := [(\mathbf{x}_{v1},...,\mathbf{x}_{vN_v})^T, \mathbf{1}_v]$, the diagonal label matrix $\mathbf{Y}_v := diag([y_{v1},...,y_{vN_v}])$, and the vector of slack variables $\xi_v := [\xi_{v1},....,\xi_{vN_v}]^T$. With these definitions, it follows readily that $\mathbf{w}_v = (\mathbf{I}_{p+1}-\Pi_{p+1})\mathbf{r}_v$, where $\Pi_{p+1}$ 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 (6) can be rewritten as

$$\begin{aligned}\min_{\{\mathbf{r}_v,\xi_v,\omega_{vu}\}}\max_{\{\delta_v\}}\ &\tfrac{1}{2}\sum_{v=1}^{V}\mathbf{r}_v^T(\mathbf{I}_{p+1}-\Pi_{p+1})\mathbf{r}_v+VC_l\sum_{v=1}^{V}\mathbf{1}_v^T\xi_v\\ &+\sum_{v=1}^{V_a}\left(V_aC_l\mathbf{r}_v^T(\mathbf{I}_{p+1}-\Pi_{p+1})\delta_v-C_a\|\delta_v\|_0\right)\\ \text{s.t.}\quad &\mathbf{Y}_v\mathbf{X}_v\mathbf{r}_v\geq\mathbf{1}_v-\xi_v, && \forall v\in\mathcal{V};\\ &\xi_v\geq\mathbf{0}_v, && \forall v\in\mathcal{V};\\ &\mathbf{r}_v=\omega_{vu},\omega_{vu}=\mathbf{r}_u, && \forall v\in\mathcal{V},\forall u\in\mathcal{B}_u;\\ &\delta_v\in\mathcal{U}_{v0}, && \forall v\in\mathcal{V}_a.\end{aligned} \tag{7}$$

With the alternating direction method of multipliers (ADMoM), the iterations of solving problem (6) are summarized as follows:

**Proposition 2** *With arbitrary initialization $\delta_v(0), \mathbf{r}_v(0), \lambda_v(0)$ and $\alpha_v(0)=\mathbf{0}_{(p+1)\times 1}$, the iterations per node are given by:*

$$\begin{aligned}\delta_v(t+1)\in\arg\max_{\{\delta_v,s_v\}}\ &V_aC_l\mathbf{r}_v^T(t)(\mathbf{I}_{p+1}-\Pi_{p+1})\delta_v\\ &-\mathbf{1}^Ts_v\\ \text{s.t.}\quad &C_a\delta_v\leq s_v, && \forall v\in\mathcal{V}_a;\\ &C_a\delta_v\geq -s_v, && \forall v\in\mathcal{V}_a;\\ &\delta_v\in\mathcal{U}_{v0}, && \forall v\in\mathcal{V}_a.\end{aligned} \tag{8}$$

$$\begin{aligned}&\lambda_v(t+1)\\ &\in\arg\max_{\mathbf{0}\leq\lambda_v\leq VC_l\mathbf{1}_v}\ -\tfrac{1}{2}\lambda_v^T\mathbf{Y}_v\mathbf{X}_v\mathbf{U}_v^{-1}\mathbf{X}_v^T\mathbf{Y}_v\lambda_v\\ &\qquad+\left(\mathbf{1}_v+\mathbf{Y}_v\mathbf{X}_v\mathbf{U}_v^{-1}\mathbf{f}_v(t)\right)^T\lambda_v,\end{aligned} \tag{9}$$

$$\mathbf{r}_v(t+1)=\mathbf{U}_v^{-1}\left(\mathbf{X}_v^T\mathbf{Y}_v\lambda_v(t+1)-\mathbf{f}_v(t)\right), \tag{10}$$

$$\alpha_v(t+1)=\alpha_v(t)+\frac{\eta}{2}\sum_{u\in\mathcal{B}_v}[\mathbf{r}_v(t+1)-\mathbf{r}_u(t+1)], \tag{11}$$

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

**Proof**. See Appendix B.

Iterations (8) - (11) are summarized into Algorithm 1 below. Note that at any given iteration $t$ of the algorithm, each node $v\in\mathcal{V}$ evaluates its own local discriminant function $g_v^{(t)}(\mathbf{x})$ for any vector $\mathbf{x}$ as

$$g_v^{(t)}(\mathbf{x})=[\mathbf{x}^T,1]\mathbf{r}_v(t) \tag{12}$$

**Algorithm 1**

Randomly initialize $\delta_v(0), \mathbf{r}_v(0), \lambda_v(0)$ and $\alpha_v(0)=\mathbf{0}_{(p+1)\times 1}$ for every $v\in\mathcal{V}$.

**for** $t=0,1,2,...$ **do**
  **for all** $v\in\mathcal{V}$ **do**
    Compute $\delta_v(t+1)$ via (8).
  **end for**
  **for all** $v\in\mathcal{V}$ **do**
    Compute $\lambda_v(t+1)$ via (9).
    Compute $\mathbf{r}_v(t+1)$ via (10).
  **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 $\alpha_v(t+1)$ via (11).
  **end for**
**end for**

Algorithm 1 solves the min-max 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. The nature of the iterative algorithms also provides resiliency to the distributed machine learning algorithms. It provides mechanisms for each node to respond to its neighbors and the adversarial behaviors in real time. When unanticipated events occur, the algorithm will be able to automatically respond and self-configure in an optimal way.

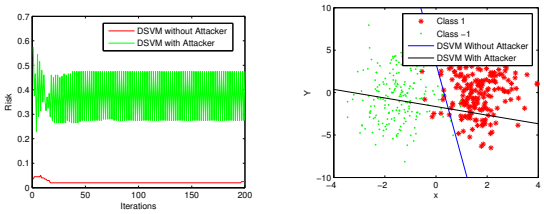
## V. Numerical Experiments

In this section, we analyze the performance of DSVM with attacker changing the training data. We use average empirical risk as a metric to assess the accuracy of the learning, which is defined as follows:

$$\mathbf{R}_{emp}(t):=\frac{1}{N_{Test}}\sum_{v=1}^{V}\sum_{n=1}^{N_{vT}}\frac{1}{2}\left|\widetilde{y}_{vn}-\widehat{y}_{vn}(t)\right|, \tag{13}$$

where $\widetilde{y}_{vn}$ is the true label, $\widehat{y}_{vn}(t)$ is the predicted label. Clearly, a high average empirical risk indicates a bad performance.

Figure 3(a) depicts the risk of the ADMoM-DSVM in the game with the attacker. Here, we use a fully connected network with 4 nodes. Each node contains 50 labeled 2-dimension training samples and 50 testing samples from global training set which is shown in Figure 3(b). The attacker uses the atomic action set with $C_\delta = 100$ and $C_a = 1$. Clearly, Figure 3(a) shows that the attacker has a significant impact on DSVM since the risk is higher than the one for the DSVM without attacker. Figure 3(b) plots the result of the training samples. Blue line represents the discriminant functions found by the ADMoM-DSVM without an attacker, and the black one represents the situation with an attacker. Clearly, the black line does not separate green samples and red samples properly.




(a) (b)

Fig. 3. Evolution of the empirical errors of ADMoM-DSVM with the attacker. Training data and testing data are generated from two Gaussian classes.

It is obvious that the attacker can cause disastrous results for the learner. In next subsections, we will illustrate in detail how the attacker affects the training process with different values of $C_a$, $C_\delta$ and numbers of nodes he can take over. From the learner's aspect, we will study how the network connections affect the attacker's objective.

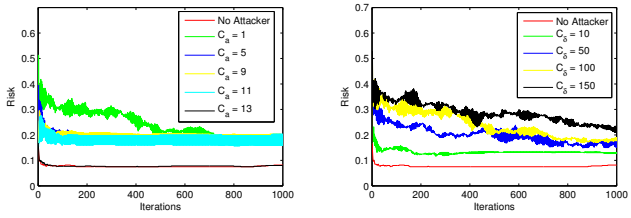
The data we used here is Spambase Data [16]. We use 600 samples for training and the other 600 samples for testing. Each sample has 57 dimensions, and the labels of each samples are either 1 or -1.

### *A. Effect of Parameters $C_a$, $C_\delta$ and the Number of Nodes the Attacker Can Take Over*

The effect of the attacker depends to a great extent on parameters $C_a$ and $C_\delta$. $C_a$ represents the parameter of the cost function for the attacker. A larger $C_a$ means a larger cost for the attacker. $C_\delta$ is the parameter of the attacker's action set, which describes the fact that the norm of the values which the attacker has modified is bounded by $C_\delta$. In this simulation, we study the effect of the two parameters respectively, notice that here we assume that the attacker takes over all the nodes.

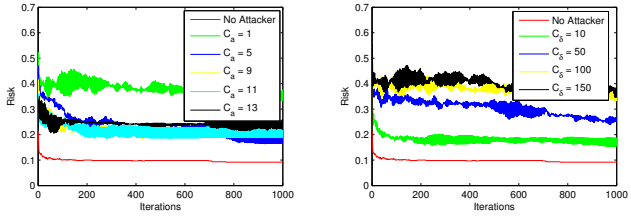
Figure 4(a) shows the results of different $C_a$ with $C_\delta = 100$, from the figure, we can see that there is a higher risk if the attacker has a lower $C_a$, and also it takes more iterations to converge. Another important observation is that, when $C_a$ is high enough, i.e., $C_a = 13$, the learner will achieve the similar results to the case where there is no attacker. Figure 4(b) shows the results of different $C_\delta$ with the same $C_a = 1$. It shows that a higher bound of the attacker's action set will account for a higher risk and lower convergence rate.

In the above experiments, an attacker is assumed to be able to take over all nodes in the network. In the next experiment, we study the case where the attacker's capability is limited, and he can only take over a subset of nodes in the network. We investigate how the number of compromised nodes by the attacker affects the classification results of the global network. Figure 7 shows the result of the experiment. From the figure, we can conclude that if the attacker has the ability of taking over more nodes, he will create a higher impact on the learner.



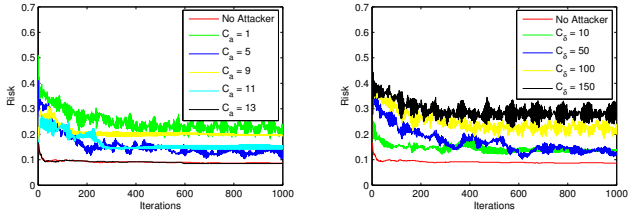

(a) (b)

Fig. 4. Evolution of the risk of ADMoM-DSVM, with different $C_a$ in (a) and different $C_\delta$ in (b). The network contains 10 nodes and it's fully connected.




(a) (b)

Fig. 5. Evolution of the risk of ADMoM-DSVM, with different $C_a$ in (a) and different $C_\delta$ in (b). The network contains 20 nodes and it's fully connected.




(a) (b)

Fig. 6. Evolution of the risk of ADMoM-DSVM, with different $C_a$ in (a) and different $C_\delta$ in (b). The network contains 10 nodes but each node has the connectivity of 0.33.

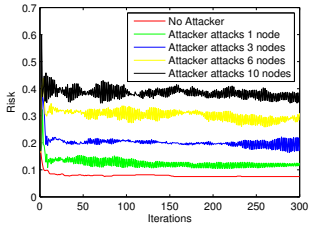



Fig. 7. Evolution of the risk under different numbers of nodes the attacker can take over.

### *B. Effect of the Network Connectivity*

In this section, we use a fully connected network with 10 nodes to study the effect of the network connectivity on the game between the learner and the attacker. Figure 5 shows the

results of classifying the same training and testing data in a fully connected network with 20 nodes. Compared to Figure 4, a network with more nodes is more vulnerable to the attacker when he can control every node in the network. Notice that when $C_a = 13$, our experiment shows that the attacker has no impact on the learner, but in the experiment with more nodes in network, the attacker has a significant impact on the learner.

Figure 6 shows the results of the situation when the network has less average connectivity. The connectivity being 0.33 indicates that the average of neighbors in each node is 3. In this experiment, we assume that all nodes have 3 neighbors. Note that with less connectivity, the risks of the network with an attacker converge faster but become higher. Hence we see that the convergence rate becomes slower with higher connectivity, and it will reduce the errors due to the attacker's behavior.

The last two experiments illustrate that the attacker will have more impact on the learner if the network contains more nodes or less connectivity, but all the experiments are conducted under the condition that the nodes in the network have the same level of connectivity. The next experiment will study the effect of nodes with heterogeneous connectivity in one network.

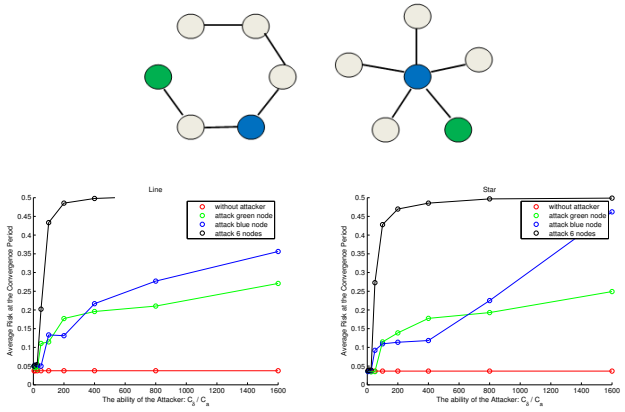



Fig. 8. Global Risk of ADMoM-DSVM in the game with an attacker in two different networks, $line$ and $star$. First row shows the network topologies; second row shows the result in either network. Clearly, nodes with more neighbors are more sensitive to the attacker.

Without losing generality, consider networks with same number of nodes and average connectivities, which is shown in Figure 8. Clearly, attacking 6 nodes will result in higher classification errors, but when the attacker can only take control of one node. Nodes with more neighbors (blue) turn out to be more sensitive to the capabilities of the attacker. In particular, as the attacker becomes more powerful, blue nodes will cause more damage than the green nodes with less neighbors.

## VI. Conclusion

Machine learning algorithms are ubiquitous but inherently vulnerable to adversaries. This paper has investigated the security issues of distributed support vector machines in an adversarial environment. 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 adversarial environment. Experimental results have shown that an attacker can have a significant impact on SVM if his capability and resources are sufficiently large. In addition, a network with a large number of nodes and low connectivity is less resilient than a network with higher connectivity. Hence, the network topology has a strong relation to the resiliency of the distributed SVM algorithm. One direction of future works is to develop a network design theory to form machine-learning networks that can achieve a desirable level of resiliency. In addition, we would also extend the current framework to investigate nonlinear distributed SVM, and other machine learning algorithms.

## Appendix A: Proof of Proposition 1

By using hinge loss function, we reformulate problem (6) into the following problem:

$$\begin{aligned}\min_{\{\mathbf{w}_v,b_v,\xi_{vn}\}}\ \max_{\{\delta_v\}}\ & \frac{1}{2}\sum_{v=1}^{V}\|\mathbf{w}_v\|^2 \\ &+ V_l C_l \sum_{v=1}^{V_l}\sum_{n=1}^{N_v}\left[1-\mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)\right]_+ \\ &+ V_a C_l \sum_{v=1}^{V_a}\sum_{n=1}^{N_v}\left[1-\mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)\right]_+ \\ &+ \sum_{v=1}^{V_a}\left(V_a C_l \mathbf{w}_v^T\delta_v - C_a\|\delta_v\|\right) \\ \text{s.t.}\quad & \mathbf{w}_v=\mathbf{w}_u, b_v=b_u, \quad \forall v\in\mathcal{V}, n=1,...,N_v; \\ & (\delta_{v1},...,\delta_{vN_v})\in\mathcal{U}_v, \qquad \forall v\in\mathcal{V}_a.\end{aligned} \tag{14}$$

As a result, we only need to prove that problem (5) is equivalent to problem (14). Since both of problems are min-max problems with the same variables, we only need to prove that we minimize the same maximization problem, as a result, we only need to show that the following problem

$$\begin{aligned}\max_{(\delta_{v1},...,\delta_{vN_v})\in\mathcal{U}_v}\ & V_a C_l \sum_{n=1}^{N_v}\left[1-\mathrm{y}_{vn}(\mathbf{w}_v^T(\mathbf{x}_{vn}-\delta_{vn})+b_v)\right]_+ \\ & - C_a\sum_{n=1}^{N_v}\|\delta_{vn}\|\end{aligned} \tag{15}$$

is equivalent to the following problem

$$\begin{aligned}\max_{\delta_v\in\mathcal{U}_{v0}}\ & V_a C_l \sum_{n=1}^{N_v}\left[1-\mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn}+b_v)\right]_+ \\ & + V_a C_l \mathbf{w}_v^T\delta_v - C_a\|\delta_v\|.\end{aligned} \tag{16}$$

Since $\delta_v$ is independent in (16), and $\{\delta_{vn}\}$ is independent in (15), we can separate maximization problem into $V_a$ sub-maximization problems, and solving the sub-problems is equivalent to solving the global maximization problem. Therefore, we only need to show the equivalence between the sub-problem.

We adopt the similar proof in [14], recall the definition of sublinear aggregated action set, $\mathcal{U}_v^- \subseteq \mathcal{U}_v \subseteq \mathcal{U}_v^+$. Hence, fixing any $(\mathbf{w}_v, b_v) \in \mathbb{R}^{n+1}$, we have the following inequalities:

$$\begin{aligned}
&\max_{(\delta_{v1},\ldots,\delta_{vN_v})\in\mathcal{U}_v^-} V_a C_l \sum_{n=1}^{N_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T(\mathbf{x}_{vn} - \delta_{vn}) + b_v)\right]_+ \\
&\quad - C_a \sum_{n=1}^{N_v} \|\delta_{vn}\| \\
\leq\ &\max_{(\delta_{v1},\ldots,\delta_{vN_v})\in\mathcal{U}_v} V_a C_l \sum_{n=1}^{N_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T(\mathbf{x}_{vn} - \delta_{vn}) + b_v)\right]_+ \\
&\quad - C_a \sum_{n=1}^{N_v} \|\delta_{vn}\| \\
\leq\ &\max_{(\delta_{v1},\ldots,\delta_{vN_v})\in\mathcal{U}_v^+} V_a C_l \sum_{n=1}^{N_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T(\mathbf{x}_{vn} - \delta_{vn}) + b_v)\right]_+ \\
&\quad - C_a \sum_{n=1}^{N_v} \|\delta_{vn}\| .
\end{aligned} \tag{17}$$

To prove the theorem, we show that (16) is no larger than the leftmost term and no smaller than the rightmost term. We first show that

$$\begin{aligned}
&\max_{\delta_v\in\mathcal{U}_{v0}} V_a C_l \sum_{n=1}^{N_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn} + b_v)\right]_+ \\
&\quad + V_a C_l \mathbf{w}_v^T\delta_v - C_a \|\delta_v\| \\
\leq\ &\max_{(\delta_{v1},\ldots,\delta_{vN_v})\in\mathcal{U}_v^-} V_a C_l \sum_{n=1}^{N_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T(\mathbf{x}_{vn} - \delta_{vn}) + b_v)\right]_+ \\
&\quad - C_a \sum_{n=1}^{N_v} \|\delta_{vn}\| .
\end{aligned} \tag{18}$$

As the samples $\{\mathbf{x}_{vn}, y_{vn}\}_{v=1}^{N_v}$ are not separable, there exists $t_v \in [1 : N_v]$ which satisfies that

$$\mathrm{y}_{t_v}({\mathbf{w}_v}^T\mathbf{x}_{t_v} + b_v) < 0. \tag{19}$$

Hence, recall the definition of sublinear aggregated action set, we have:

$$\begin{aligned}
&\max_{(\delta_{v1},\ldots,\delta_{vN_v})\in\mathcal{U}_{vt_v}^-} V_a C_l \sum_{n=1}^{N_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T(\mathbf{x}_{vn} - \delta_{vn}) + b_v)\right]_+ \\
&\quad - C_a \sum_{n=1}^{N_v} \|\delta_{vn}\| \\
=\ &\max_{\delta_{vt_v}\in\mathcal{U}_{v0}} V_a C_l \sum_{n\neq t_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn} + b_v)\right]_+ \\
&\quad + V_a C_l \left[1 - \mathrm{y}_{vt_v}(\mathbf{w}_v^T(\mathbf{x}_{vt_v} - \delta_{vt_v}) + b_v)\right]_+ \\
&\quad - C_a \|\delta_{vt_v}\| \\
=\ &\max_{\delta_{vt_v}\in\mathcal{U}_{v0}} V_a C_l \sum_{n\neq t_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn} + b_v)\right]_+ \\
&\quad + V_a C_l \left[1 - \mathrm{y}_{vt_v}(\mathbf{w}_v^T\mathbf{x}_{vt_v} + b_v)\right]_+ \\
&\quad + V_a C_l(\mathrm{y}_{vt_v}\mathbf{w}_v^T\delta_{vt_v}) - C_a \|\delta_{vt_v}\| \\
=\ &\max_{\delta_v\in\mathcal{U}_{v0}} V_a C_l \sum_{n=1}^{N_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn} + b_v)\right]_+ \\
&\quad + V_a C_l \mathbf{w}_v^T\delta_v - C_a \|\delta_v\| .
\end{aligned} \tag{20}$$

The second and third equalities hold because of the inequality (19) and $\max_{\delta_{t_v}\in\mathcal{U}_{v0}} (\mathrm{y}_{vt_v}{\mathbf{w}_v}^T\delta_{t_v})$ being non-negative (recall that $\mathbf{0} \in \mathcal{U}_{v0}$). Besides, we use $\delta_v$ to replace $\delta_{vt_v}$. Since $\mathcal{U}_{vt_v}^- \subseteq \mathcal{U}_v^-$, Inequality (18) holds.

In the following step, we prove that

$$\begin{aligned}
&\max_{(\delta_{v1},\ldots,\delta_{vN_v})\in\mathcal{U}_v^+} V_a C_l \sum_{n=1}^{N_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T(\mathbf{x}_{vn} - \delta_{vn}) + b_v)\right]_+ \\
&\quad - C_a \sum_{n=1}^{N_v} \|\delta_{vn}\| \\
\leq\ &\max_{\delta_v\in\mathcal{U}_{v0}} V_a C_l \sum_{n=1}^{N_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn} + b_v)\right]_+ \\
&\quad + V_a C_l \mathbf{w}_v^T\delta_v - C_a \|\delta_v\| .
\end{aligned} \tag{21}$$

Recall the definition of $\mathcal{U}^+$, we have:

$$\begin{aligned}
&\max_{(\delta_{v1},\ldots,\delta_{vN_v})\in\mathcal{U}_v^+} V_a C_l \sum_{n=1}^{N_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T(\mathbf{x}_{vn} - \delta_{vn}) + b_v)\right]_+ \\
&\quad - C_a \sum_{n=1}^{N_v} \|\delta_{vn}\| \\
=\ &\max_{\substack{\sum_{n=1}^{N_v}\alpha_{vn}=1;\\ \alpha_{vn}\geq 0;\widehat{\delta}_{vn}\in\mathcal{U}_{v0}}} V_a C_l \sum_{n=1}^{N_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T(\mathbf{x}_{vn} - \alpha_{vn}\widehat{\delta}_{vn}) + b_v)\right]_+ \\
&\quad - C_a \sum_{n=1}^{N_v} \|\alpha_{vn}\delta_{vn}\| \\
\leq\ &\max_{\substack{\sum_{n=1}^{N_v}\alpha_{vn}=1;\\ \alpha_{vn}\geq 0;\widehat{\delta}_{vn}\in\mathcal{U}_{v0}}} V_a C_l \sum_{n=1}^{N_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn} + b_v)\right]_+ \\
&\quad + \sum_{n=1}^{N_v} V_a C_l \alpha_{vn}\mathbf{w}_v^T\widehat{\delta}_{vn} - C_a \sum_{n=1}^{N_v} \|\alpha_{vn}\delta_{vn}\| \\
=\ &\max_{\substack{\sum_{n=1}^{N_v}\alpha_{vn}=1;\\ \alpha_{vn}\geq 0.}} \max_{\widehat{\delta}_{vn}\in\mathcal{U}_{v0}} V_a C_l \sum_{n=1}^{N_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn} + b_v)\right]_+ \\
&\quad + \alpha_{vn} \sum_{n=1}^{N_v} \left(V_a C_l \mathbf{w}_v^T\widehat{\delta}_{vn} - C_a \|\delta_{vn}\|\right) \\
=\ &\max_{\delta_v\in\mathcal{U}_{v0}} V_a C_l \sum_{n=1}^{N_v} \left[1 - \mathrm{y}_{vn}(\mathbf{w}_v^T\mathbf{x}_{vn} + b_v)\right]_+ \\
&\quad + V_a C_l \mathbf{w}_v^T\delta_v - C_a \|\delta_v\| .
\end{aligned} \tag{22}$$

Inequality (21) holds.

By combining the two steps, we can show the equivalence between (15) and (16). Hence, Proposition 1 holds.

## Appendix B: Proof of Proposition 2

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 $\{\mathbf{r}_v\}$ and $\{\delta_v\}$, respectively. For fixed $\{\mathbf{r}_v^*, \xi_v^*\}$,

$$\begin{aligned}
\delta_v^* \in \arg\max_{\{\delta_v\}} &\sum_{v=1}^{V_a} \left(V_a C_l {\mathbf{r}_\mathbf{v}^*}^T(\mathbf{I}_{p+1} - \Pi_{p+1})\delta_v\right) \\
\text{s.t.}\quad &\delta_v \in \mathcal{U}_{v0}, \quad \forall v \in \mathcal{V}_a.
\end{aligned} \tag{23}$$

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

$$\begin{aligned}\delta_v^* \in \arg\max_{\{\delta_v, s_v\}} & V_a C_l \mathbf{r}_{\mathbf{v}}^{*T}(\mathbf{I}_{p+1} - \Pi_{p+1})\delta_v \\ & - \mathbf{1}^T s_v \\ \text{s.t.} \quad & C_a\delta_v \le s_v, \qquad \forall v \in \mathcal{V}_a; \\ & C_a\delta_v \ge -s_v, \qquad \forall v \in \mathcal{V}_a; \\ & \delta_v \in \mathcal{U}_{v0}, \qquad \forall v \in \mathcal{V}_a.\end{aligned} \tag{24}$$

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

$$\begin{aligned}\min_{\{\mathbf{r}_v, \omega_{vu}, \xi_v\}} & \tfrac{1}{2}\sum_{v=1}^{V} \mathbf{r}_v^T(\mathbf{I}_{p+1} - \Pi_{p+1})\mathbf{r}_v \\ & + V_a C_l \sum_{v=1}^{V_a} \mathbf{r}_v^T(\mathbf{I}_{p+1} - \Pi_{p+1})\delta_v^* + VC_l \sum_{v=1}^{V} \mathbf{1}_v^T \xi_v \\ \text{s.t.} \quad & \mathbf{Y}_v\mathbf{X}_v\mathbf{r}_v \ge \mathbf{1}_v - \xi_v, \qquad \forall v \in \mathcal{V}; \\ & \xi_v \ge \mathbf{0}_v, \qquad \forall v \in \mathcal{V}; \\ & \mathbf{r}_v = \omega_{vu}, \omega_{vu} = \mathbf{r}_u, \quad \forall v \in \mathcal{V}, \forall u \in \mathcal{B}_u.\end{aligned} \tag{25}$$

Note that term $-C_a \|\delta_v^*\|$ is removed since it does not play a role in the maximization problem. Based on (24) and (25), we have the method of solving problem (7) as follows: The first step is to randomly pick an initial $\{\mathbf{r}_v(0), \delta_v(0)\}$, then solve max-problem (24) with $\{\mathbf{r}_v(0)\}$, and obtain$\{\delta_v(1)\}$, next step is to solve min-problem (25) with $\{\delta_v(1)\}$, and obtain $\{\mathbf{r}_v(1)\}$, then we repeat solving max-problem with $\{\mathbf{r}_v^*\}$ from the previous step and solving min-problem with $\{\delta_v^*\}$ from the previous step until the pair$\{\mathbf{r}_v, \delta_v\}$ achieves convergence. Furthermore, we use the alternating direction method of multipliers(ADMoM) to solve problem (25).

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

$$\begin{aligned}\min_{\mathbf{r}} \ & f(\mathbf{r}) + g(\omega) \\ \text{s.t.} \quad & \mathbf{M}\mathbf{v} = \omega;\end{aligned} \tag{26}$$

where $f$ and $g$ are convex functions, $\mathbf{M}$ is a $p_2 \times p_1$ matrix. [15]

The augmented Lagrangian corresponding to (26) is

$$L(\mathbf{r}, \omega, \alpha) = f(\mathbf{r}) + g(\omega) + \alpha^T(\mathbf{M}\mathbf{v} - \omega) + \frac{\eta}{2}\|\mathbf{M}\mathbf{v} - \omega\|^2. \tag{27}$$

where $\alpha \in \mathbb{R}^{p_2}$ denotes the Lagrange multiplier.

Then, the ADMoM solves problem (26) by the update rules below:

$$\mathbf{r}(t+1) \in \arg\min_{\mathbf{r} \in \mathcal{P}_1} L(\mathbf{r}, \omega(t), \alpha(t)). \tag{28}$$

$$\omega(t+1) \in \arg\min_{\omega \in \mathcal{P}_2} L(\mathbf{r}(t+1), \omega, \alpha(t)). \tag{29}$$

$$\alpha(t+1) = \alpha(t) + \eta(\mathbf{M}\mathbf{v}(t+1) - \omega(t+1)). \tag{30}$$

The objective here is to transform problem (25) into the form of (26), then solve that by iterations used in (28), (29) and (30). We adopt the similar method in [9] to solve problem (25), which leads to the following result.

**Lemma** *Each node iterates $\lambda_v(t), \mathbf{r}_v(t)$ and $\alpha_v(t)$, given by*

$$\begin{aligned}\lambda_v(t+1) \in \arg\max_{\mathbf{0} \le \lambda_v \le VC_l\mathbf{1}_v} & -\tfrac{1}{2}\lambda_v^T\mathbf{Y}_v\mathbf{X}_v\mathbf{U}_v^{-1}\mathbf{X}_v^T\mathbf{Y}_v\lambda_v \\ & + (\mathbf{1}_v + \mathbf{Y}_v\mathbf{X}_v\mathbf{U}_v^{-1}\mathbf{f}_v(t))^T\lambda_v,\end{aligned} \tag{31}$$

$$\mathbf{r}_v(t+1) = \mathbf{U}_v^{-1}\left(\mathbf{X}_v^T\mathbf{Y}_v\lambda_v(t+1) - \mathbf{f}_v(t)\right), \tag{32}$$

$$\alpha_v(t+1) = \alpha_v(t) + \frac{\eta}{2}\sum_{u \in \mathcal{B}_v}[\mathbf{r}_v(t+1) - \mathbf{r}_u(t+1)], \tag{33}$$

*where* $\mathbf{U}_v = (\mathbf{I}_{p+1} - \Pi_{p+1}) + 2\eta|\mathcal{B}_v|\mathbf{I}_{p+1}, \mathbf{f}_v(t) = V_aC_l\delta_v^* + 2\alpha_v(t) - \eta\sum_{u \in \mathcal{B}_v}(\mathbf{r}_v(t) + \mathbf{r}_u(t)), \eta > 0$ .

By combining the above lemma with Problem (24), we obtain Proposition 2.

## REFERENCES


[1] Brown, Robert Grover, and Patrick YC Hwang. *Introduction to random signals and applied Kalman filtering*. Vol. 3. New York: Wiley, 1992.

[2] Suykens, Johan AK, and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters* 9, no. 3 (1999): 293-300.

[3] Wold, Svante, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems* 2, no. 1 (1987): 37-52.

[4] Meyer, Tony A., and Brendon Whateley. SpamBayes: Effective open-source, Bayesian based, email classification system. In *CEAS*. 2004.

[5] Kavitha, T., and D. Sridharan. Security vulnerabilities in wireless sensor networks: A survey. *Journal of information Assurance and Security* 5, no. 1 (2010): 31-44.

[6] Xie, Liang, Xinwen Zhang, Ashwin Chaugule, Trent Jaeger, and Sencun Zhu. Designing system-level defenses against cellphone malware. In *Reliable Distributed Systems, 2009. SRDS'09. 28th IEEE International Symposium on*, pp. 83-90. IEEE, 2009.

[7] Tague, Patrick, and Radha Poovendran. Modeling adaptive node capture attacks in multi-hop wireless networks. *Ad Hoc Networks* Ad Hoc Networks 5, no. 6 (2007): 801-814.

[8] Parno, Bryan, Adrian Perrig, and Virgil Gligor. Distributed detection of node replication attacks in sensor networks. In *Security and Privacy, 2005 IEEE Symposium on,* pp. 49-63. IEEE, 2005.

[9] Forero, Pedro A., Alfonso Cano, and Georgios B. Giannakis. Consensus-based distributed support vector machines. *The Journal of Machine Learning Research* 11 (2010): 1663-1707.

[10] Barreno, Marco, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. The security of machine learning. *Machine Learning* 81, no. 2 (2010): 121-148.

[11] Dalvi, Nilesh, Pedro Domingos, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining,* pp. 99-108. ACM, 2004.

[12] Kantarcioglu, Murat, Bowei Xi, and Chris Clifton. Classifier evaluation and attribute selection against active adversaries. *Data Mining and Knowledge Discovery* 22, no. 1-2 (2011): 291-335.

[13] Liu, Wei, and Sanjay Chawla. A game theoretical model for adversarial learning. In *Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on,* pp. 25-30. IEEE, 2009.

[14] Xu, Huan, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *The Journal of Machine Learning Research* 10 (2009): 1485-1510.

[15] Eckstein, Jonathan, and Yao Wang. Augmented Lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results. *RUTCOR Research Reports* 32 (2012).

[16] Lichman, Moshe. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2013.