Consensus-Based Distributed Support Vector Machines

Article *in* Journal of Machine Learning Research · May 2010

Source: DBLP

CITATIONS
410

READS
538

3 authors:

Pedro A. Forero
Navy's Space and Naval Warfare Systems Command
30 PUBLICATIONS **1,029** CITATIONS
SEE PROFILE

Alfonso Cano
King Juan Carlos University
34 PUBLICATIONS **1,776** CITATIONS
SEE PROFILE

G.B. Giannakis
University of Minnesota Twin Cities
1,249 PUBLICATIONS **70,953** CITATIONS
SEE PROFILE

Consensus-Based Distributed Linear Support Vector Machines

Pedro A. Forero, Alfonso Cano, Georgios B. Giannakis
University of Minnesota
Dept. of Electrical and Computer Engineering
Minneapolis, MN 55455, USA
{forer002, alfonso, georgios}@umn.edu

ABSTRACT

This paper develops algorithms to train linear support vector machines (SVMs) when training data are distributed across different nodes and their communication to a centralized node is prohibited due to, for example, communication overhead or privacy reasons. To accomplish this goal, the centralized linear SVM problem is cast as the solution of coupled decentralized convex optimization subproblems with consensus constraints on the parameters defining the classifier. Using the method of multipliers, distributed training algorithms are derived that do not exchange elements from the training set among nodes. The communications overhead of the novel approach is fixed and fully determined by the topology of the network instead of being determined by the size of the training sets as it is the case for existing incremental approaches. An online algorithm where data arrive sequentially to the nodes is also developed. Simulated tests illustrate the performance of the algorithms.

Categories and Subject Descriptors

C.2.4 [**Distributed systems**]: [Distributed applications]; I.2.6 [**Learning**]: [Induction]; G.1.6 [**Optimization**]: [Convex Programming]

General Terms

Algorithms

Keywords

Support Vector Machines, Sensor Networks, Optimization

1. INTRODUCTION

SVM classifiers have been successfully employed in applications ranging from medical imaging and biometric classification to speech and handwriting recognition [8, 17, 12]. Based on concepts from statistical learning theory, SVMs seek the maximum-margin linear classifier based on a training set comprising multidimensional data with corresponding classification labels [21, 20]. Distributed learning problems involving SVMs arise when training data are acquired by different nodes and their communication to a central processing unit or fusion center (FC) is costly or prohibited due to, for example, the size of local training data sets or privacy reasons. In the important case of wireless sensor networks (WSNs), nodes are battery operated; thus, transferring all data to an FC is discouraged due to stringent power constraints imposed on individual nodes.

Training an SVM requires solving a quadratic optimization problem of dimensionality given by the cardinality of the training set. The SVM solution is described by a subset of elements from the training set named support vectors (SVs). Whenever new data are to be classified, the classification decision made by the SVM is based solely on the SVs. Not surprisingly, various works dealing with distributed formulations of the SVM algorithm fully rely on SVs as a mean to percolate local information throughout the network. Distributed (D-) SVMs whereby each node broadcasts SVs, obtained from the local training set, to neighboring nodes were explored in [18, 9, 10]. Other related works, deal with parallel SVMs to handle large training sets [4, 5, 13]. In this case, the global training set is broken to smaller subsets, each to train an SVM independently in a sequential or parallel manner.

The novel approach pursued in the present paper trains an SVM in a *fully distributed* fashion. The centralized SVM problem is cast as the solution of coupled decentralized convex optimization subproblems with consensus constraints imposed on the parameters defining the classifier. Using the alternating direction method of multipliers (ADMoM) of [2], distributed training algorithms based solely on node-to-node message exchanges are developed. The derived distributed approaches feature:

- **Scalability and reduced communication overhead**. Unlike centralized approaches, whereby nodes communicate training samples to an FC, the distributed approach relies on *in-network* processing with information exchanges among single-hop neighboring nodes only. This keeps the communication overhead per node at an af-

IPSN'10, April 12–16, 2010, Stockholm, Sweden.

fordable level within its neighborhood, even if the network expands to cover a larger geographical area. In centralized approaches however, nodes will consume increased resources to reach the FC. Moreover, the novel distributed algorithms do not exchange SVs or elements among local training sets, and entail fixed inter-node communication overhead per iteration regardless of the size of local training sets as compared to, e.g., [18].

- **Robustness to isolated point(s) of failure**. In centralized scenarios, if the FC fails, the learning task fails altogether – a critical issue in tactical applications such as target classification. In contrast, if a single node fails while the network remains connected, the proposed algorithms will converge to a classifier trained using the data from nodes that remain operational. Even if the network becomes disconnected, the proposed algorithms will stay operational with performance dependent on the training samples per connected sub-network.

- **Fully decentralized network operation**. Alternative distributed approaches include incremental and parallel SVMs. Incremental cyclic SV message passing approaches need to identify a Hamiltonian cycle (going through all nodes once) in the network [18, 9]. This is needed not only in the deployment stage, but also every time a node fails. However, Hamiltonian cycles do not always exist, and if they do, finding them is an NP-hard task [19]. On the other hand, parallel SVM implementations assume full (any-to-any) network connectivity, and require a central unit defining how SVs from intermediate stages/nodes are combined, along with predefined inter-node communication protocols and specific conditions on the local training sets; see, e.g., [4, 5, 13].

- **Convergence guarantees to centralized SVM performance**. The derived DSVM algorithm is *provably convergent* to the centralized SVM classifier, as if all distributed samples were available centrally, regardless of the local training set available per node and the network topology, as long as the network is connected.

- **Robustness to noisy inter-node communications and privacy preservation**. The novel distributed classification scheme is robust even when noise is present among inter-node exchanges. Such noise is due to, e.g., vector quantization, additive Gaussian channel disturbances (in wireless links), or Laplacian noise added when transmitting samples to guarantee data privacy [7].

General notational conventions are as follows. Upper (lower) bold face letters are used for matrices (column vectors); $(\cdot)^T$ denotes matrix and vector transposition; the ji-th entry of a matrix (j-th entry of a vector) is denoted by $[\cdot]_{j,i}$ ($[\cdot]_j$); $\text{diag}(x_1,\ldots,x_N)$ denotes a diagonal matrix with $x_1,\ldots,x_N$ on its main diagonal; $|\cdot|$ denotes set cardinality; $\succeq$ ($\preceq$) element-wise $\geq$ ($\leq$); $\{\cdot\}$ a set of variables with appropriate elements; $\|\cdot\|$ the Euclidean norm; $\mathbf{1}_j$ ($\mathbf{0}_j$) a vector of all ones (zeros) of size N_j; $\mathbf{I}_M$ stands for the $M \times M$ identity matrix; $\mathbb{E}\{\cdot\}$ denotes expected value; and $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ the multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

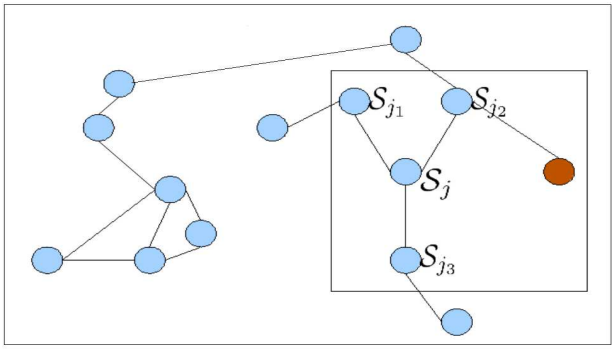

Figure 1: Network example. Nodes are represented by colored circles.

2. PRELIMINARIES

With reference to Figure 1, consider a network with J nodes modeled by an undirected graph $\mathcal{G}(\mathcal{J}, \mathcal{E})$ with vertices $\mathcal{J} := \{1,\ldots,J\}$ representing nodes and edges $\mathcal{E}$ describing links among communicating nodes. Node $j \in \mathcal{J}$ only communicates with nodes in its one-hop neighborhood (ball) $\mathcal{B}_j \subseteq \mathcal{J}$. The graph $\mathcal{G}$ is assumed connected, i.e., any two nodes in $\mathcal{G}$ are connected by a (perhaps multihop) path in $\mathcal{G}$. No other requirements on $\mathcal{G}$ are postulated, thus e.g. $\mathcal{G}$ can contain cycles. At every node $j \in \mathcal{J}$, a labeled training set $\mathcal{S}_j := \{(\mathbf{x}_{jn}, y_{jn}) : n = 1,\ldots,N_j\}$ of size N_j is available, where $\mathbf{x}_{jn} \in \mathcal{X}$ denotes a $p \times 1$ feature vector belonging to the input space $\mathcal{X} \subset \mathbb{R}^p$, and $y_{jn} \in \mathcal{Y} := \{-1, 1\}$ denotes its corresponding class label.[1]

Given $\mathcal{S}_j$ per node j, the objective is to find a maximum-margin linear discriminant function $g(\mathbf{x})$ in a distributed fashion to classify any new feature vector $\mathbf{x}$ to one of the two classes $\{-1, 1\}$, without communicating $\mathcal{S}_j$ to other nodes $j' \neq j$. Potential application scenarios are outlined next.

Example 1: *(Environmental monitoring using WSNs)* Suppose that a set of wireless sensors is deployed to construct a model for predicting the presence of a particular pollutant at a given point in time and space. Through self-localization, sensor j can determine its position $\mathbf{r}_j := [r_{j1},\ r_{j2},\ r_{j3}]^T$ in an area of interest. Through sensing, node j can determine the presence of a particular pollutant at location $\mathbf{r}_j$ and time t_n; an event denoted by $p_{jn} \in \{1, -1\}$, where $p_{jn} = 1$ indicates the presence of the pollutant and $p_{jn} = -1$ its absence. With $\mathcal{S}_j = \{([\mathbf{r}_j^T, t_n]^T, p_{jn}) : n = 1,\ldots,N_j\}$ available at sensor j, the objective is to determine the presence of a pollutant at any point $\mathbf{r}$ and time t. Since sensors are deployed to cover a geographical area, remote sensors may be unable to send $\mathcal{S}_j$ to an FC. Because sensors are only allowed to communicate

[1]Although not included in this model, K-ary alphabets $\mathcal{Y}$ with $K > 2$ can be considered as well.

with one-hop neighbors, percolating $\mathcal{S}_j$ throughout the network becomes prohibitive as the network size or the size of $\mathcal{S}_j$ grow.

Example 2: *(Underwater acoustic sensor networks)* Consider a set of J battery operated nodes with multiple sensors deployed for underwater surveillance [1]. The J nodes form a connected network whose representing graph $\mathcal{G}$ has size $J-1$. During the training phase, each node collects a training set $\mathcal{S}_j$ comprised by acoustic, magnetic, and mechanical observations (vibration). The training sets $\mathcal{S}_j$ are collected in-situ due to the dependency of the observations on the underwater location of the node, e.g., shallow or deep water. The goal of the underwater acoustic network is to classify small underwater unmanned vehicles, divers and submarines. Due to power constraints direct communication between every node and an FC is discouraged. Furthermore, low transmission bandwidth and sensitivity to multipath and fading hinder incremental communication schemes of local datasets to the FC.

If $\{\mathcal{S}_j\}_{j=1}^J$ were available at an FC, then the global optimal variables $\mathbf{w}^*$ and b^* describing the *centralized* maximum-margin linear discriminant function $g^*(\mathbf{x}) = \mathbf{x}^T\mathbf{w}^*+b^*$ could be obtained by solving the following convex optimization problem; see e.g., [20, Ch. 7]

$$\begin{aligned}\min_{\mathbf{w},b,\{\xi_{jn}\}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{j=1}^{J}\sum_{n=1}^{N_j}\xi_{jn} \\ \text{s.t.} \quad & y_{jn}(\mathbf{w}^T\mathbf{x}_{jn}+b) \geq 1-\xi_{jn} \quad \forall j\in\mathcal{J},\, n=1,\ldots,N_j \\ & \xi_{jn}\geq 0 \qquad\qquad\qquad\quad \forall j\in\mathcal{J},\, n=1,\ldots,N_j\end{aligned} \tag{1}$$

where the slack variables ξ_{jn} account for non-linearly separable training sets, and C is a tunable positive scalar.

The objective here is to develop a *distributed* solver to the centralized problem in (1) without exchanging or transmitting local training sets among nodes or to an FC, while guaranteeing similar performance to the one of a centralized equivalent SVM. Although sequential approaches to construct local SVM estimates are possible, the size of the information exchanges required might be excessive, especially if the number of SVs per node is large [10, 18]. Exchanging all local SVs among all nodes in the network several times is necessary for existing distributed classifiers to approach the optimal centralized solution. Moreover, sequential schemes require a Hamiltonian cycle to be identified in the network, thereby guaranteeing the minimum communication overhead possible. Computing such a cycle is an NP-hard task and in most cases a sub-optimal cycle is used at the expense of increased communication overhead [19]. In other situations communicating SVs directly might be prohibited because of the sensitivity of the information bore.

3. DISTRIBUTED LINEAR SVM

This section presents a reformulation of the maximum-margin linear classifier problem in (1) that is amenable to distributed implementation. Let $\mathbf{w}_j$ and b_j denote local auxiliary variables defining a local linear discriminant function $g_j(\mathbf{x}) = \mathbf{w}_j^T\mathbf{x}+b_j$ at node j. Variables $\{\mathbf{w}_j\}_{j=1}^J$ and $\{b_j\}_{j=1}^J$ can be seen as the solution of the following *consensus-based* optimization problem (cf. (1))

$$\begin{aligned}\min_{\{\mathbf{w}_j,b_j,\xi_{jn}\}} \quad & \frac{1}{2}\sum_{j=1}^{J}\|\mathbf{w}_j\|^2 + JC\sum_{j=1}^{J}\sum_{n=1}^{N_j}\xi_{jn} \\ \text{s.t.} \quad & y_{jn}(\mathbf{w}_j^T\mathbf{x}_{jn}+b_j)\geq 1-\xi_{jn} \quad \forall j\in\mathcal{J},\, n=1,\ldots,N_j \\ & \xi_{jn}\geq 0 \qquad\qquad\qquad\quad \forall j\in\mathcal{J},\, n=1,\ldots,N_j \\ & \mathbf{w}_j=\mathbf{w}_i,\; b_j=b_i \qquad \forall j\in\mathcal{J},\, \forall i\in\mathcal{B}_j.\end{aligned} \tag{2}$$

For connected graphs, the equality constraints introduced in (2) enforce agreement on $\mathbf{w}_j$ and b_j across nodes, i.e., $\mathbf{w}_1^* = \ldots = \mathbf{w}_J^*$ and $b_1^* = \ldots = b_J^*$ for the optimal solution $\{\mathbf{w}_j^*, b_j^*\}_{j=1}^J$ of (2). A factor J is introduced to the cost function of (2) to guarantee that $\mathbf{w}_1^* = \ldots = \mathbf{w}_J^* = \mathbf{w}^*$ and $b_1^* = \ldots = b_J^* = b^*$, where $\mathbf{w}^*$ and b^* are the optimal solvers of (1), thus guaranteeing that problems (1) and (2) are equivalent.

For notational brevity, define the augmented vector $\mathbf{v}_j := [\mathbf{w}_j^T,\; b_j]^T$, the feature matrix $\mathbf{X}_j := [[\mathbf{x}_{j1},\ldots,\mathbf{x}_{jN_j}]^T, \mathbf{1}_j]$, the diagonal label matrix $\mathbf{Y}_j := \text{diag}(y_{j1},\ldots,y_{jN_j})$ and the collection of slack variables $\boldsymbol{\xi}_j := [\xi_{j1},\ldots,\xi_{jN_j}]^T$. With these definitions, problem (2) can be rewritten as

$$\begin{aligned}\min_{\{\mathbf{v}_j,\boldsymbol{\xi}_j,\boldsymbol{\omega}_{ji}\}} \quad & \frac{1}{2}\sum_{j=1}^{J}\mathbf{v}_j^T(\mathbf{I}_{p+1}-\boldsymbol{\Pi}_{p+1})\mathbf{v}_j + JC\sum_{j=1}^{J}\mathbf{1}_j^T\boldsymbol{\xi}_j \\ \text{s.t.} \quad & \mathbf{Y}_j\mathbf{X}_j\mathbf{v}_j \succeq \mathbf{1}_j-\boldsymbol{\xi}_j \qquad \forall j\in\mathcal{J} \\ & \boldsymbol{\xi}_j\succeq\mathbf{0}_j \qquad\qquad\quad \forall j\in\mathcal{J} \\ & \mathbf{v}_j=\boldsymbol{\omega}_{ji},\; \boldsymbol{\omega}_{ji}=\mathbf{v}_i \qquad \forall j\in\mathcal{J},\, \forall i\in\mathcal{B}_j\end{aligned} \tag{3}$$

where $\boldsymbol{\Pi}_{p+1}$ is a $(p+1)\times(p+1)$ matrix with zeros everywhere except for the $(p+1,p+1)$-st entry, that is given by $[\boldsymbol{\Pi}_{p+1}]_{(p+1),(p+1)} = 1$; and $\{\boldsymbol{\omega}_{ji}\}$ denotes a set of auxiliary variables used to decouple the variables $\mathbf{v}_j$ between neighboring nodes while maintaining the network's ability to reach global consensus. Let $\boldsymbol{\alpha}_{ji1}$ and $\boldsymbol{\alpha}_{ji2}$ denote the Lagrange multipliers corresponding to the equality constraints $\mathbf{v}_j = \boldsymbol{\omega}_{ji}$ and $\boldsymbol{\omega}_{ji} = \mathbf{v}_i$, respectively. The surrogate augmented Lagrangian of (3) is given by

$$\begin{aligned}\mathcal{L}(\{\mathbf{v}_j\},\{\boldsymbol{\xi}_j\},\{\boldsymbol{\omega}_{ji}\},\{\boldsymbol{\alpha}_{jik}\}) &= \frac{1}{2}\sum_{j=1}^{J}\mathbf{v}_j^T(\mathbf{I}_{p+1}-\boldsymbol{\Pi}_{p+1})\mathbf{v}_j \\ &+ JC\sum_{j=1}^{J}\mathbf{1}_j^T\boldsymbol{\xi}_j + \sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\boldsymbol{\alpha}_{ji1}^T(\mathbf{v}_j-\boldsymbol{\omega}_{ji}) + \sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\boldsymbol{\alpha}_{ji2}^T(\boldsymbol{\omega}_{ji}-\mathbf{v}_i) \\ &+ \frac{\eta}{2}\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\|\mathbf{v}_j-\boldsymbol{\omega}_{ji}\|^2 + \frac{\eta}{2}\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\|\boldsymbol{\omega}_{ji}-\mathbf{v}_i\|^2\end{aligned} \tag{4}$$

where η is a positive tuning parameter. Note that the set of constraints $\mathcal{W} := \{\mathbf{Y}_j\mathbf{X}_j\mathbf{v}_j \succeq \mathbf{1}_j-\boldsymbol{\xi}_j,\; \boldsymbol{\xi}_j\succeq\mathbf{0}_j\}$ was not included in the Lagrangian $\mathcal{L}$, thus the name surrogate. The

augmented terms $||\mathbf{v}_j - \boldsymbol{\omega}_{ji}||^2$ and $||\boldsymbol{\omega}_{ji} - \mathbf{v}_i||^2$ are included to further penalize the equality constraints in (3); see, e.g., [2, Ch. 3]. The role of these extra terms is two-fold: (a) they effect *strict* convexity of the cost with respect to $\mathbf{v}_j$ and $\boldsymbol{\omega}_{ji}$, and thus ensure convergence to the unique optimum of the global cost (whenever possible), even when the local costs are convex but not strictly so; and (b) through the scalar η, they allow one to tradeoff speed of convergence for steady-state approximation error [2, Ch. 3]

Based on [2, Ch. 3], the ADMoM-based DSVM (MoM-DSVM) minimizes (4) cyclically with respect to (w.r.t.) one set of variables while keeping all other variables fixed, and performs gradient ascent updates on the multipliers $\boldsymbol{\alpha}_{jik}$. The resulting ADMoM iterations are summarized by the following lemma.

Lemma 1. *The ADMoM iterations corresponding to* (3) *are given by*

$$\{\mathbf{v}_j(t+1), \boldsymbol{\xi}_j(t+1)\} = \underset{\{\mathbf{v}_j, \boldsymbol{\xi}_j\} \in \mathcal{W}}{\arg\min} \mathcal{L}\left(\{\mathbf{v}_j\}, \{\boldsymbol{\xi}_j\}, \{\boldsymbol{\omega}_{ji}(t)\}, \{\boldsymbol{\alpha}_{jik}(t)\}\right) \quad (5a)$$

$$\{\boldsymbol{\omega}_{ji}(t+1)\} = \underset{\{\boldsymbol{\omega}_{ji}\}}{\arg\min} \mathcal{L}\left(\{\mathbf{v}_j(t+1)\}, \{\boldsymbol{\xi}_j(t+1)\}, \{\boldsymbol{\omega}_{ji}\}, \{\boldsymbol{\alpha}_{jik}(t)\}\right) \quad (5b)$$

$$\boldsymbol{\alpha}_{ji1}(t+1) = \boldsymbol{\alpha}_{ji1}(t) + \eta[\mathbf{v}_j(t+1) - \boldsymbol{\omega}_{ji}(t+1)] \quad \forall j \in \mathcal{J},\ \forall i \in \mathcal{B}_j \quad (5c)$$

$$\boldsymbol{\alpha}_{ji2}(t+1) = \boldsymbol{\alpha}_{ji2}(t) + \eta[\boldsymbol{\omega}_{ji}(t+1) - \mathbf{v}_i(t+1)] \quad \forall j \in \mathcal{J},\ \forall i \in \mathcal{B}_j. \quad (5d)$$

PROOF. See Appendix A. □

Note that update (5b) can be computed in closed form due to the inclusion of the quadratic penalty terms in (4). With a suitable initialization, the number of iterates in (5a)-(5d) can be reduced, as established in the following lemma.

Lemma 2. *If the iterates $\boldsymbol{\alpha}_{ji1}(t)$ and $\boldsymbol{\alpha}_{ji2}(t)$ are initialized to zero $\forall (j,i) \in \mathcal{E}$; i.e., $\boldsymbol{\alpha}_{ji1}(0) = \boldsymbol{\alpha}_{ji2}(0) = \mathbf{0}$, then iterations* (5a)-(5d) *are equivalent to*

$$\{\mathbf{v}_j(t+1), \boldsymbol{\xi}_j(t+1)\} = \underset{\{\mathbf{v}_j, \boldsymbol{\xi}_j\} \in \mathcal{W}}{\arg\min} \mathcal{L}'(\{\mathbf{v}_j\}, \{\boldsymbol{\xi}_j\}, \{\mathbf{v}_j(t)\}, \{\boldsymbol{\alpha}_j(t)\}) \quad (6a)$$

$$\boldsymbol{\alpha}_j(t+1) = \boldsymbol{\alpha}_j(t) + \frac{\eta}{2} \sum_{i \in \mathcal{B}_j} [\mathbf{v}_j(t+1) - \mathbf{v}_i(t+1)], \quad \forall j \in \mathcal{J} \quad (6b)$$

where $\boldsymbol{\alpha}_j(t) := \sum_{i \in \mathcal{B}_j} \boldsymbol{\alpha}_{ji1}(t)$, and $\mathcal{L}'$ is given by

$$\mathcal{L}'(\{\mathbf{v}_j\}, \{\boldsymbol{\xi}_j\}, \{\mathbf{v}_j(t)\}, \{\boldsymbol{\alpha}_j(t)\}) = \frac{1}{2} \sum_{j=1}^{J} \mathbf{v}_j^T (\mathbf{I}_{p+1} - \boldsymbol{\Pi}_{p+1}) \mathbf{v}_j + JC \sum_{j=1}^{J} \mathbf{1}_j^T \boldsymbol{\xi}_j + 2 \sum_{j=1}^{J} \boldsymbol{\alpha}_j^T(t) \mathbf{v}_j + \eta \sum_{j=1}^{J} \sum_{i \in \mathcal{B}_j} \left\| \mathbf{v}_j - \frac{1}{2}[\mathbf{v}_j(t) + \mathbf{v}_i(t)] \right\|^2. \quad (7)$$

PROOF. See Appendix B. □

Iteration (6a) solves a quadratic optimization problem with linear inequality constraints. Let $\boldsymbol{\lambda}_j := [\lambda_{j1}, \ldots, \lambda_{jN_j}]^T$ and $\boldsymbol{\gamma}_j := [\gamma_{j1}, \ldots, \gamma_{jN_j}]^T$ denote Lagrange multipliers per node corresponding to the inequality constraints $\mathbf{Y}_j \mathbf{X}_j \mathbf{v}_j \succeq \mathbf{1}_j - \boldsymbol{\xi}_j$ and $\boldsymbol{\xi}_j \succeq \mathbf{0}_j$, respectively. The dual problem of (6a) provides expressions for the optimal $\boldsymbol{\lambda}_j$ at iteration $t+1$, namely $\boldsymbol{\lambda}_j(t+1)$, as a function of $\mathbf{v}_j(t)$ and local aggregate Lagrange multipliers $\boldsymbol{\alpha}_j(t)$. Likewise, the Karush-Kuhn-Tucker (KKT) conditions provide expressions for $\mathbf{v}_j(t+1)$ as a function of $\boldsymbol{\alpha}_j(t)$ and the optimal dual variables $\boldsymbol{\lambda}_j(t+1)$. Interestingly, iterations are decoupled across nodes. The resulting iterations and the convergence results are established in the following proposition.

Proposition 1. *With t indexing iterations, consider the iterates $\boldsymbol{\lambda}_j(t)$, $\mathbf{v}_j(t)$ and $\boldsymbol{\alpha}_j(t)$, given by*

$$\boldsymbol{\lambda}_j(t+1) = \underset{\boldsymbol{\lambda}_j:\ \mathbf{0}_j \preceq \boldsymbol{\lambda}_j \preceq JC\mathbf{1}_j}{\arg\max} -\frac{1}{2} \boldsymbol{\lambda}_j^T \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{X}_j^T \mathbf{Y}_j \boldsymbol{\lambda}_j + \left(\mathbf{1}_j - \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{f}_j(t)\right)^T \boldsymbol{\lambda}_j \quad (8a)$$

$$\mathbf{v}_j(t+1) = \mathbf{U}_j^{-1} \left(\mathbf{X}_j^T \mathbf{Y}_j \boldsymbol{\lambda}_j(t+1) + \mathbf{f}_j(t)\right) \quad (8b)$$

$$\boldsymbol{\alpha}_j(t+1) = \boldsymbol{\alpha}_j(t) + \frac{\eta}{2} \sum_{i \in \mathcal{B}_j} [\mathbf{v}_j(t+1) - \mathbf{v}_i(t+1)] \quad (8c)$$

where $\mathbf{U}_j := (1 + 2\eta|\mathcal{B}_j|)\mathbf{I}_{p+1} - \boldsymbol{\Pi}_{p+1}$, $\mathbf{f}_j(t) := -2\boldsymbol{\alpha}_j(t) + \eta \sum_{i \in \mathcal{B}_j} [\mathbf{v}_j(t) + \mathbf{v}_i(t)]$, $\eta > 0$, and arbitrary initialization vectors $\boldsymbol{\lambda}_j(0)$, $\mathbf{v}_j(0)$ and $\boldsymbol{\alpha}_j(0)$. The iterate $\mathbf{v}_j(t)$ converges to the solution of (3)*, call it $\mathbf{v}^*$, as $t \to \infty$; i.e., $\lim_{t \to \infty} \mathbf{v}_j(t) = \mathbf{v}^*$.*

PROOF. See Appendix C. □

The iterate $\boldsymbol{\alpha}_j(t)$ captures the aggregate constraint violation between the local estimates $\mathbf{v}_j(t)$ and $\mathbf{v}_i(t)$ $\forall i \in \mathcal{B}_j$ (cf. (3)); and η trades off speed of convergence for accuracy of the solution after a finite number of iterations. Also, the local optimal slack variables $\boldsymbol{\xi}_j^*$ can be found via the KKT conditions for (5a). Similar to the centralized SVM algorithm, if $[\boldsymbol{\lambda}_j(t)]_n \neq 0$, then $[\mathbf{x}_{jn}^T, 1]^T$ is an SV. Finding $\boldsymbol{\lambda}_j(t+1)$ as in (8a) requires solving a quadratic optimization problem similar to the one that a centralized SVM would solve, e.g., via gradient projection algorithms or interior point methods; see e.g., [20, Ch. 6]. However, the number of variables involved in (8a) per iteration per node is considerably smaller when compared to its centralized counterpart, namely N_j versus $\sum_{j=1}^{J} N_j$.

The MoM-DSVM algorithm is summarized as Algorithm 1 and is illustrated in Figure 2. At the beginning all nodes agree on the values JC and η. Also, every node computes its local $N_j \times N_j$ matrix $\mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{X}_j^T \mathbf{Y}_j$ which remains unchanged throughout the entire algorithm. At $t = 0$, local $\mathbf{v}_j$ and $\boldsymbol{\alpha}_j$ are randomly initialized. Every node tracks its local $(p+1) \times 1$ estimates $\mathbf{v}_j(t)$ and $\boldsymbol{\alpha}_j(t)$; and $N_j \times 1$ vector $\boldsymbol{\lambda}_j(t)$. At iteration $t+1$, node j computes vector $\mathbf{f}_j(t)$ locally and finds its local $\boldsymbol{\lambda}_j(t+1)$ via (8a). Vector $\boldsymbol{\lambda}_j(t+1)$ together with the local training set $\mathcal{S}_j$ are used at node j to compute $\mathbf{v}_j(t+1)$

Algorithm 1 MoM-DSVM

Require: Initialize $\mathbf{v}_j(0)$ and $\boldsymbol{\alpha}_j(0)$, $\forall j \in \mathcal{J}$.

for $t = 0, 1, 2, \ldots$ **do**
  for all $j \in \mathcal{J}$ **do**
    Compute $\boldsymbol{\lambda}_j(t+1)$ via (8a).
    Compute $\mathbf{v}_j(t+1)$ via (8b).
  end for
  for all $j \in \mathcal{J}$ **do**
    Broadcast $\mathbf{v}_j(t+1)$ to all neighbors $i \in \mathcal{B}_j$.
  end for
  for all $j \in \mathcal{J}$ **do**
    Compute $\boldsymbol{\alpha}_j(t+1)$ via (8c).
  end for
end for

via (8b). Next, node j broadcasts its newly updated local estimates $\mathbf{v}_j(t+1)$ to all its one-hop neighbors $i \in \mathcal{B}_j$. Iteration $t+1$ concludes when every node updates its local $\boldsymbol{\alpha}_j(t+1)$ vector via (8c). The algorithm stops whenever the local cost $R_j(t) := \frac{1}{2}\mathbf{v}_j^T(t)(\mathbf{I}_{p+1} - \mathbf{\Pi}_{p+1})\mathbf{v}_j(t) + JC\mathbf{1}_j^T\boldsymbol{\xi}_j(t)$ changes below a prescribed threshold θ_j; i.e., whenever $|R_j(t_i) - R_j(t_{i-1})| < \tau_j \; \forall j \in \mathcal{J}$. To decide whether to stop the algorithm in a distributed fashion, sensors evaluate their stopping criterion at fixed time instants $t_0, t_1, t_2, \ldots$. At any of time, say t_i, each node j checks whether $|R_j(t_i) - R_j(t_{i-1})| < \tau_j$, and a 0-1 flag is broadcasted to all $i \in \mathcal{B}_j$ indicating whether the condition is satisfied. Neighbors update their flags by multiplying them with the received one. In a number of iterations equal at most to the diameter of $\mathcal{G}$, all nodes will agree on either 0 (at least one node did not satisfy the stopping condition); or, 1 (all nodes satisfied the stopping condition) in which case the algorithm stops.

Finally, note that at any given iteration t of the algorithm, each node j can evaluate its own local discriminant function $g_j^{(t)}(\mathbf{x})$ for any vector $\mathbf{x} \in \mathcal{X}$ as

$$g_j^{(t)}(\mathbf{x}) = [\mathbf{x}^T, 1]\mathbf{v}_j(t) \tag{9}$$

which from Proposition 1 will converge to the same solution across all nodes as $t \to \infty$. Simulated tests in Section 4 demonstrate that after a few iterations the classification performance of (9) outperforms that of the local discriminant function obtained by relying on the local training set alone.

Remark 1. The messages exchanged between neighboring nodes in the MoM-DSVM algorithm correspond to local estimates $\mathbf{v}_j(t)$ which together with the local aggregate Lagrange multipliers $\boldsymbol{\alpha}_j(t)$ encode sufficient information about the local training sets to achieve consensus globally. Furthermore, the size of the messages being communicated among neighboring nodes remains fixed. Per iteration and per node a message of size $(p+1) \times 1$ is broadcasted (vectors $\boldsymbol{\alpha}_j$ are not exchanged among nodes). This is to be contrasted with incremental SVM algorithms, see e.g. [18, 9, 10], where the size of the messages exchanged between neighboring nodes depends on the number of SVs found at each incremental step. Although the SVs are sparse in the training sets, the number of SVs may remain large causing excessive power usage when transmitting SVs from one node to the next.

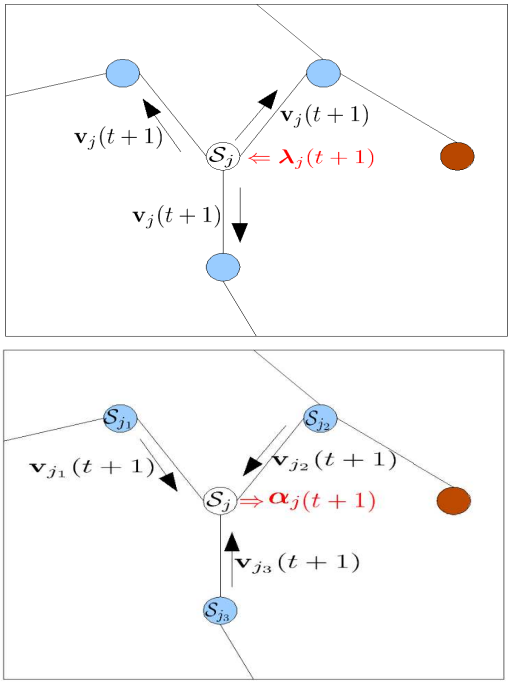

Figure 2: Iterations (8a)-(8c) **as described in Algorithm 1: (top) every node** $j \in \mathcal{J}$ **computes** $\boldsymbol{\lambda}_j(t+1)$ **to obtain** $\mathbf{v}_j(t+1)$**, then every node** j **broadcasts** $\mathbf{v}_j(t+1)$ **to all neighbors** $i \in \mathcal{B}_j$**; (bottom) once every node** $j \in \mathcal{J}$ **has received** $\mathbf{v}_i(t+1)$ **from all** $i \in \mathcal{B}_j$**, every node** j **updates** $\boldsymbol{\alpha}_j(t+1)$**.**

Remark 2. Algorithm 1 requires each node j to wait for all neighboring information $\mathbf{v}_i(t+1)$, $i \in \mathcal{B}_j$, before updating $\boldsymbol{\alpha}_j(t+1)$ in (8c); thus, it is a synchronous algorithm. Although an asynchronous implementation is desirable, no convergence guarantees or delay tolerance bounds for AD-MoM are available at this time [2]. However, there are related asynchronous schemes derived for simplified cost functions that deal with asynchronous updates due to link failures and packet delays [14], which could prove helpful to obtain an asynchronous version of MoM-DSVM.

3.1 Online Distributed SVM

In many distributed sensing tasks data arrive sequentially, and possibly asynchronously. In addition, the observed phenomena may change with time. In such cases, training examples need to be added or removed from each local training set $\mathcal{S}_j$. Incremental and decremental training sets are expressed in terms of *time-varying* augmented features matrix $\mathbf{X}_j(t)$, and corresponding label matrices $\mathbf{Y}_j(t)$. An online version of MoM-DSVM is thus well motivated when a new training example $\mathbf{x}_{jn}$ with corresponding label y_{jn} acquired at time t is incorporated into $\mathbf{X}_j(t)$ and $\mathbf{Y}_j(t)$, respectively. This is possible since convergence is guaranteed for any initial value of

$\boldsymbol{\lambda}_j(t)$, $\mathbf{v}_j(t)$ and $\boldsymbol{\alpha}_j(t)$. Upon substituting $\mathbf{X}_j(t)$ and $\mathbf{Y}_j(t)$ in (8a)-(8c), the corresponding modified iterations are given by

$$\boldsymbol{\lambda}_j(t+1) = \underset{\boldsymbol{\lambda}_j:\, \mathbf{0}_j(t+1) \preceq \boldsymbol{\lambda}_j \preceq JC\mathbf{1}_j(t+1)}{\arg\max} -\frac{1}{2}\boldsymbol{\lambda}_j^T \mathbf{Y}_j(t+1) \times \mathbf{X}_j(t+1)\mathbf{U}_j^{-1}\mathbf{X}_j(t+1)^T\mathbf{Y}_j(t+1)\boldsymbol{\lambda}_j + \left[\mathbf{1}_j(t+1) - \mathbf{Y}_j(t+1)\mathbf{X}_j(t+1)\mathbf{U}_j^{-1}\mathbf{f}_j(t)\right]^T \boldsymbol{\lambda}_j \quad (10a)$$

$$\mathbf{v}_j(t+1) = \mathbf{U}_j^{-1}\{\mathbf{X}_j(t+1)^T\mathbf{Y}_j(t+1)\boldsymbol{\lambda}_j(t+1) - 2\boldsymbol{\alpha}_j(t) + \eta\sum_{i\in\mathcal{B}_j}[\mathbf{v}_j(t) + \mathbf{v}_i(t)]\} \quad (10b)$$

$$\boldsymbol{\alpha}_j(t+1) = \boldsymbol{\alpha}_j(t) + \frac{\eta}{2}\sum_{i\in\mathcal{B}_j}[\mathbf{v}_j(t+1) - \mathbf{v}_i(t+1)]. \quad (10c)$$

Note that the dimensionality of $\boldsymbol{\lambda}_j$ must be modified to accommodate the variable number of elements in $\mathcal{S}_j$ at every time instant t, which also explains why the notation $\mathbf{0}_j(t+1)$ and $\mathbf{1}_j(t+1)$ is used in (10a). The online MoM-DSVM algorithm is summarized as Algorithm 2. For this algorithm to run, no conditions on the increments and decrements are imposed over $\mathcal{S}_j(t)$. They can be asynchronous and may comprise multiple training examples at once. In principle however, the penalty parameter η as well as JC can become time-dependent. Intuitively, if the training sets remain invariant across a sufficient number of time instants, $\mathbf{v}_j(t)$ will closely track the optimal maximum-margin linear classifier. Moreover, simulated test demonstrate that the modified iterations in (10a)-(10c) are able to track changes in the training set even when these occur at every time instant t.

Remark 3. Compared to existing *centralized* online SVM alternatives in e.g., [3, 11], the online MoM-DSVM algorithm allows for seamless integration of both distributed and online processing. Nodes with training sets available at initialization and nodes that are acquiring their training sets online can be integrated to jointly find the maximum-margin linear classifier. Furthermore, whenever needed, the online MoM-DSVM algorithm can return a partially trained model constructed with the examples available to the network at a given time. Likewise, elements of the training sets can be removed without having to restart the MoM-DSVM algorithm. This feature allows, e.g., a joint implementation with algorithms that aim to account for *concept drift* [15].

Algorithm 2 Online MoM-DSVM

Require: Initialize $\mathbf{v}_j(0)$ and $\boldsymbol{\alpha}_j(0)$ $\forall j \in \mathcal{J}$.
for $t = 0, 1, 2, \ldots$ **do**
  for all $j \in \mathcal{J}$ **do**
    Update $\mathbf{Y}_j(t+1)\mathbf{X}_j(t+1)\mathbf{U}_j^{-1}\mathbf{X}_j(t+1)^T\mathbf{Y}_j(t+1)$.
    Compute $\boldsymbol{\lambda}_j(t+1)$ via (10a).
    Compute $\mathbf{v}_j(t+1)$ via (10b).
  end for
  for all $j \in \mathcal{J}$ **do**
    Broadcast $\mathbf{v}_j(t+1)$ to all neighbors $i \in \mathcal{B}_j$.
  end for
  for all $j \in \mathcal{J}$ **do**
    Compute $\boldsymbol{\alpha}_j(t+1)$ via (10c).
  end for
end for

4. NUMERICAL SIMULATIONS

4.1 Test Case 1: Synthetic Data

Consider a randomly generated network with $J = 30$ nodes, algebraic connectivity 0.0448 and average degree per sensor 3.267. Each node acquires labeled training examples from two different classes $\mathcal{C}_1$ and $\mathcal{C}_2$ with corresponding labels $y_1 = 1$ and $y_2 = -1$. Classes $\mathcal{C}_1$ and $\mathcal{C}_2$ are equiprobable and consist of random vectors drawn from a 2-dimensional Gaussian distribution with common covariance matrix $\boldsymbol{\Sigma} = [1,\ 0;\ 0,\ 2]$ and mean vectors $\boldsymbol{\mu}_1 = [-1,\ -1]^T$ and $\boldsymbol{\mu}_2 = [1,\ 1]^T$, respectively. Each local training set $\mathcal{S}_j$ consists of $N_j = N = 10$ labeled examples. Likewise, a test set $\mathcal{S}_{\text{Test}} := \{(\widetilde{\mathbf{x}}_n, \widetilde{y}_n),\ n = 1, \ldots, N_T\}$ with $N_T = 600$ examples is used to evaluate the generalization performance of the classifiers. The Bayes optimal classifier for this 2-class problem is linear [6, Ch. 2], with risk $\mathbf{R}_{\text{Bayes}} = 0.1103$. The average empirical risk of the MoM-DSVM algorithm as a function of the number of iterations is defined as

$$\mathbf{R}_{\text{emp}}(t) := \frac{1}{JN_T}\sum_{j=1}^{J}\sum_{n=1}^{N_T}\frac{1}{2}|\widetilde{y}_n - \widehat{y}_{jn}(t)| \quad (11)$$

where $\widehat{y}_{jn}(t)$ is the prediction at node j and iteration t for $\widetilde{\mathbf{x}}_n$, $n = 1, \ldots, N_T$ using $\mathbf{v}_j(t)$. The average empirical risk of the local SVMs across nodes $\mathbf{R}_{\text{emp}}^{\text{local}}$ is defined as in (11) with $\widehat{y}_{jn}$ found using only locally-trained SVMs.

Figure 3 (top) depicts the risk of the MoM-DSVM algorithm as a function of the number of iterations t for different values of JC. In this test, $\eta = 10$ and a total of 500 Monte Carlo runs were performed with local training sets and the test set randomly drawn for each Monte Carlo run. The risk of the MoM-DSVM algorithm reduces as the number of iterations increases, quickly outperforming local-based predictions and approaching that of the centralized benchmark. The convergence rate of MoM-DSVM can be tuned by modifying the parameters JC and η. To further visualize this test case, Figure 3 (bottom) shows the global training set, and the linear discriminant functions found by a centralized SVM, MoM-DSVM and local SVMs at two nodes with $JC = 20$.

4.2 Test Case 2: Sequential Data

Next, the performance of the online MoM-DSVM algorithm is tested in a network with $J = 10$ nodes, algebraic connectivity 0.3267, and average degree per node 2.80. Data from two classes arrive sequentially at each node in the network in the following fashion: at $t = 0$ each node has available one labeled training example drawn from the class distributions described in Test case 1. From $t = 0$ to $t = 19$, each node acquires a new labeled training example per iteration from the distribution described in Test case 1. From $t = 20$

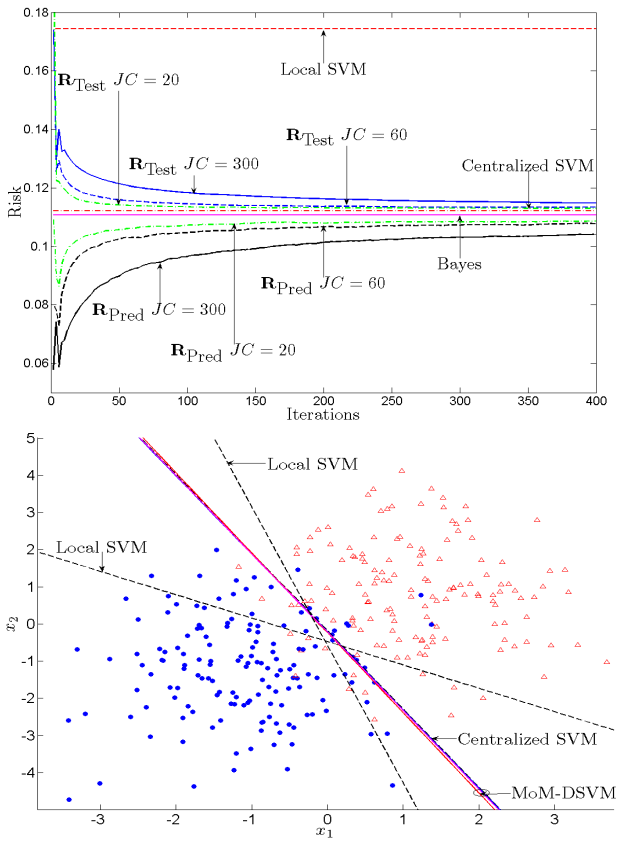

Figure 3: MoM-DSVM: (top) test error $(\mathbf{R}_{\text{Test}})$ and prediction error $(\mathbf{R}_{\text{Pred}})$ for a two-class synthetic problem solved via MoM-DSVM; (bottom) centralized and local SVM results for two Gaussian classes.

to $t = 99$ no new training example is acquired. After iteration $t = 99$, the distribution from which training examples in class $\mathcal{C}_1$ were generated changes to a 2-dimensional Gaussian distribution with covariance matrix $\mathbf{\Sigma}_1 = [1,\ 0;\ 0,\ 2]$ and mean vector $\boldsymbol{\mu}_1 = [-1,\ 5]^T$. From $t = 100$ to $t = 119$, each node acquires a new labeled training example per iteration using the new class-conditional distribution of $\mathcal{C}_1$ while the class-conditional distribution of $\mathcal{C}_2$ remains unchanged. During these iterations, the training examples from $\mathcal{C}_1$ generated during the interval $t = 0$ to $t = 19$ are removed, one per iteration. From $t = 120$ to $t = 299$ nodes do not acquire new labeled training examples. From iteration $t = 300$ to $t = 499$, 8 new training examples are included per node drawn only from class $\mathcal{C}_1$ with the same class-conditional distribution as the one used at the beginning of the algorithm. Finally, at iteration $t = 500$ all labeled training samples drawn from $t = 300$ to $t = 499$ are removed from each node at once, returning to the global data set available prior to iteration $t = 300$. The algorithm continues without any further changes in the local training sets until convergence.

Figure 4 (top) illustrates the tracking capabilities of the online MoM-DSVM scheme for different values of η and 100 Monte Carlo runs performed per test. The figure of merit in this case is $\mathcal{V}(t) := \frac{1}{J}\sum_{j=1}^{J} \|\mathbf{v}_j(t) - \mathbf{v}_c(t)\|$, where $\mathbf{v}_c(t)$ are the coefficients of the centralized SVM using the training set available at time t. Note that the peaks in Figure 4 (top) correspond to the changes of local training sets. MoM-DSVM rapidly adapts the coefficients after the local training sets are modified. Clearly, the parameter η can be tuned to control the speed with which MoM-DSVM adapts. Notice that a large η may cause over-damping effects. Figure 4 (bottom) shows snapshots, for a single run, of the MoM-DSVM results, the global training set and the local discriminant functions at different iterations for $\eta = 30$. The color-filled training examples correspond to the current global SVs found by the online MoM-DSVM algorithm.

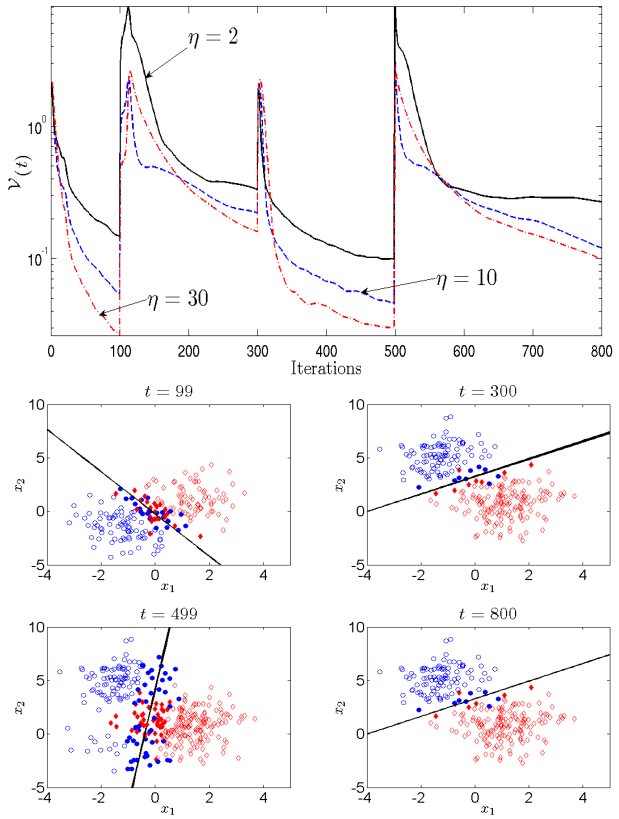

Figure 4: Online MoM-DSVM: (top) average error $\mathcal{V}(t)$ for various values of η; (bottom) snapshots of the global training data set and local $g_j(\mathbf{x})$ at all nodes.

4.3 Test Case 3: MNIST Dataset

In this case, MoM-DSVM is tested on the MNIST data-base of handwritten images [16]. The MNIST data base contains images of digits 0 to 9. All images are of size 28 by 28 pixels. We consider the binary problem of classifying digit 2 versus digit 9 using a linear classifier. For this experiment each image is vectorized to a 784×1 vector. In particular, we take 5,900 training samples per digit, and a test set of 1,000 samples per digit. Both training and test sets used are as given by the MNIST database, i.e., there is no preprocessing of the data. The centralized performance of a linear classifier trained

with all observations was 0.018, obtained with "The Spider" [22], a machine learning toolbox. For simulations, we consider a randomly generated network with $J = 25$, algebraic connectivity 3.2425, and average degree per node 12.80. The training set is equally partitioned across nodes, thus every node in the network receives $N_j = 472$ samples. Note that the distribution of samples across nodes influences the training phase of MoM-DSVM. In particular, if data per node are biased toward one particular class, then, intuitively, the training phase may require more iterations to percolate appropriate information across the network. In the simulations, we consider the extreme case when every node has data corresponding to a single digit, thus a local binary classifier cannot be constructed. Even if an incremental approach were used, it would need at least one full cycle through the network to enable the construction of local estimators at every node.

Figure 5 (top) shows the evolution of the test error for the a network with $J = 25$ nodes and highly biased local data. Different values for the penalties η were used to illustrate their effect on both the classification performance and the convergence rate of MoM-DSVM. The penalty parameter JC determines the final performance of the classifier, however, the figures also reveal that for a finite number of iterations η also influences the final performance of the classifier. Larger values of η may be desirable, however, note that if η is too large, then MoM-DSVM focuses on reaching consensus across nodes disregarding the classification performance. Although our MoM-DSVM is guaranteed to converge for all η, a large value for η may hinder the convergence rate.

Finally, the effect of network connectivity on the performance of MoM-DSVM is explored. In this experiment, we consider a network with $J = 25$ nodes, ring topology and biased data distribution as before. The performance of MoM-DSVM is illustrated by Figure 5 (bottom). It is clear that in this case a larger η improves the convergence rate of the algorithm. Also, after few iterations the average performance of the classifier across the network is close to the optimal. In practice, a small reduction of performance over the centralized classifier may be acceptable in which case MoM-DSVM can stop after a small number of iterations. The communication cost of MoM-DSVM can be easily computed at any iteration in terms of the number of scalars transmitted across the network. For the MNIST data set the communication cost up to iteration t is $785Jt$ scalars (cf. Section 3).

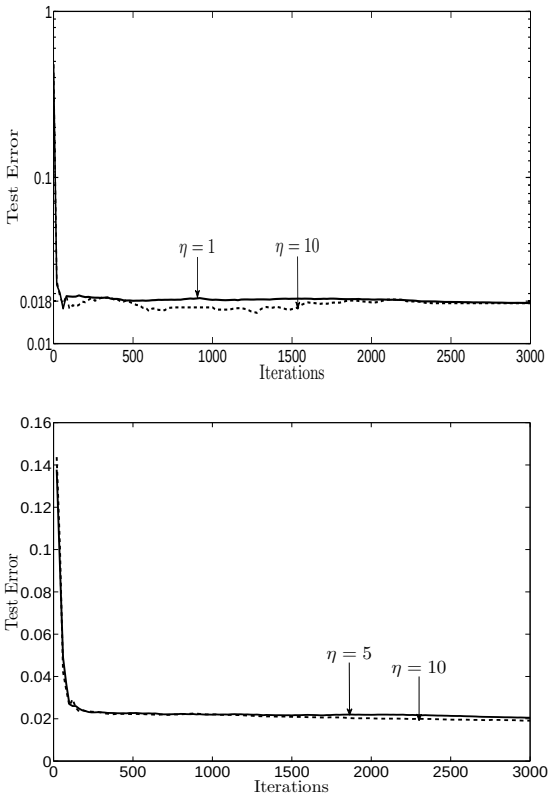

Figure 5: Evolution of the test error $(\mathbf{R}_{\mathbf{Test}})$ of MoM-DSVM for $J = 25$ nodes, a two-class problem using digits 2 and 9 from MNIST data set unevenly distributed across nodes: (top) $JC = 5$, and a random network topology; (bottom) $JC = 1$, and a ring network topology.

4.4 Test case 4: Communication Cost

A comparison with the incremental SVM (ISVM) approach in [18] was also explored. In this setting we consider the same network as in Test Case 1. Each node acquires a local training set with $N = 20$ observations generated as in Test Case 1. A global test set with $N_T = 1,200$ was used and 100 Monte Carlo runs were performed. The MoM-DSVM algorithm used $JC = 20$ and the local SVMs used $C = 10$. The network topology is a ring; thus, no overhead in the communication cost for ISVM is added. Nevertheless, in more general network topologies such overhead might dramatically increase the total communication cost incurred by ISVM. The communication cost is measured in terms of the number of scalars communicated per node.

Figure 6 depicts the cumulative cost for MoM-DSVM and ISVM as a function of their classification performance. In this particular case and with the most favorable network topology for an incremental approach, we observe that MoM-DSVM achieves a comparable risk to ISVM with a smaller number of transmitted scalars. Specifically, to achieve a risk of 0.1159, MoM-DSVM communicates on average 1,260 scalars whereas ISVM communicates on average 8,758 scalars. MoM-DSVM can largely reduce the amount of communications throughout the network, a gain that translates directly to lower power consumption and longer battery life for individual nodes.

4.5 Test Case 5: Perturbed Transmissions

We next demonstrate that the novel distributed classification scheme is robust even when noise is present among internode exchanges. Specifically, in this test case we purposely add Laplacian noise $\boldsymbol{\epsilon}_j(t)$ to the transmitted variables $\mathbf{v}_j(t)$ at each iteration t and node j. Perturbed transmissions can be used to preserve the privacy of the local databases $\mathcal{S}_j$ in the

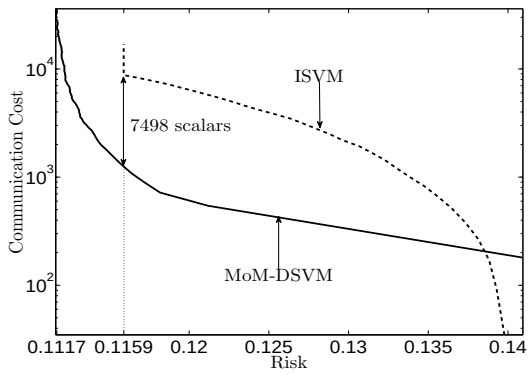

Figure 6: Communication cost for MoM-DSVM and ISVM for a WSN with $J = 30$, and a ring topology.

network against eavesdroppers [7]. The form and variance level $\mathbf{C}_j$ of the local perturbations $\boldsymbol{\epsilon}_j(t)$ can be adjusted per node to guarantee a prescribed level of privacy [7].

In this test, perturbations $\boldsymbol{\epsilon}_j(t)$ are zero-mean and white across time and space, i.e., $\mathbb{E}\{\boldsymbol{\epsilon}_j(t_1)\boldsymbol{\epsilon}_j^T(t_2)\} = \mathbf{0}_{p+1,p+1}$ if $t_1 \neq t_2$ and $\mathbb{E}\{\boldsymbol{\epsilon}_i(t)\boldsymbol{\epsilon}_j^T(t)\} = \mathbf{0}_{p+1,p+1}$ if $i \neq j\ \forall i, j \in \mathcal{J}$, where $\mathbf{0}_{p+1,p+1}$ denotes a $(p+1) \times (p+1)$ matrix of zeros. The resulting MoM-DSVM iterations are given by

$$\boldsymbol{\lambda}_j(t+1) = \underset{\boldsymbol{\lambda}_j:\, \mathbf{0}_j \preceq \boldsymbol{\lambda}_j \preceq JC\mathbf{1}_j}{\arg\max} \; -\frac{1}{2}\boldsymbol{\lambda}_j^T \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{X}_j^T \mathbf{Y}_j \boldsymbol{\lambda}_j + \left(\mathbf{1}_j - \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{f}_j(t)\right)^T \boldsymbol{\lambda}_j \quad (12a)$$

$$\mathbf{v}_j(t+1) = \mathbf{U}_j^{-1}\left[\mathbf{X}_j^T \mathbf{Y}_j \widetilde{\boldsymbol{\lambda}}_j(t+1) + \mathbf{f}_j(t)\right] \quad (12b)$$

$$\boldsymbol{\alpha}_j(t+1) = \boldsymbol{\alpha}_j(t) + \frac{\eta}{2}\sum_{i \in \mathcal{B}_j}[\mathbf{v}_j(t+1) + \boldsymbol{\epsilon}_j(t) - \mathbf{v}_i(t+1) - \boldsymbol{\epsilon}_i(t)] \quad (12c)$$

where $\mathbf{U}_j := (1+2\eta|\mathcal{B}_j|)\mathbf{I}_{p+1} - \boldsymbol{\Pi}_{p+1}$ and $\mathbf{f}_j(t) := -2\boldsymbol{\alpha}_j(t) + \eta\sum_{i \in \mathcal{B}_j}[\mathbf{v}_j(t) + \boldsymbol{\epsilon}_j(t) + \mathbf{v}_i(t) + \boldsymbol{\epsilon}_i(t)]$.

Figure 7 illustrates the performance of MoM-DSVM with perturbed transmissions for a network with $J = 8$ nodes, algebraic connectivity 0.4194, and average degree per node 2.5 after 100 Monte Carlo runs. Nodes collect observations from 2 classes $\mathcal{C}_1$ and $\mathcal{C}_2$, where $\mathcal{C}_1$ is $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ with $\boldsymbol{\mu}_1 = [0,\ 0]^T$, and $\boldsymbol{\Sigma}_1 = [0.6,\ 0;\ 0,\ 0.4]$, and $\mathcal{C}_2$ is $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ with $\boldsymbol{\mu}_2 = [2,\ 2]^T$, and $\boldsymbol{\Sigma}_1 = [1,\ 0;\ 0,\ 2]$. Each node collects an equal number of observations per class for a total of $N_j = N = 50$ observations. The noise $\boldsymbol{\epsilon}_j(t)$, inserted per transmission per node, has $\mathbf{C}_j = \sigma^2\mathbf{I}_3$. The optimal classifier is determined by $\mathbf{v}^* = [-1.29,\ -0.76,\ 1.78]^T$ which is the one obtained by MoM-DSVM with $\sigma^2 = 0$. Interestingly, the average risk in the presence of perturbed transmissions remains close the perturbation free risk. Even for a large perturbation $\sigma^2 = 1$, the average risk hovers around 0.1075. Furthermore, the risk variance remains small. Indeed, it can be shown that the proposed scheme yields estimates $\mathbf{v}_j(t)$ with bounded variance.

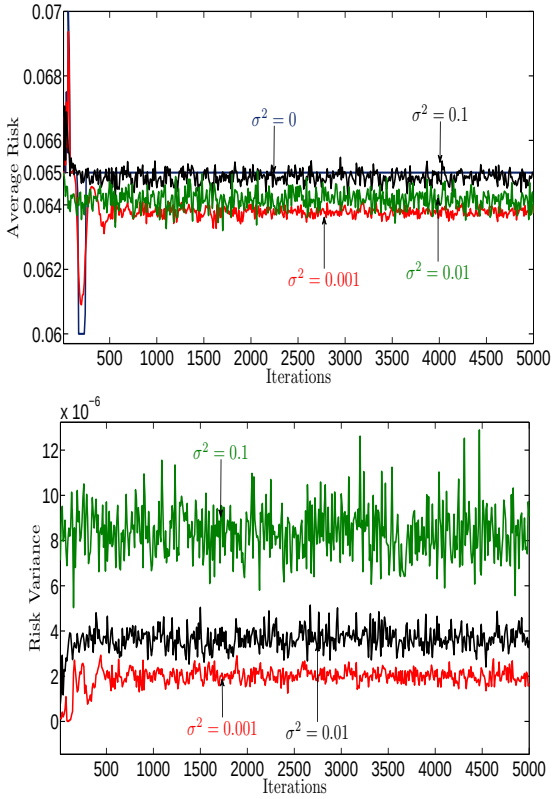

Figure 7: Average risk (top) and risk variance (bottom) for a network with $J = 8$ and a finite variance perturbation added to $\mathbf{v}_j(t)$ before it is broadcasted.

4.6 Test Case 6: Noisy Communication Links

The MoM-DSVM is also robust to non-ideal inter-node links due to, e.g., vector quantization or additive Gaussian noise present at the receiver side. To guarantee stability of the variables $\mathbf{v}_j(t)$, the MoM-DSVM must be modified, and the local Lagrange multipliers $\boldsymbol{\alpha}_{ji}(t) := \boldsymbol{\alpha}_{ji1}(t)$ must be exchanged among neighboring nodes. Let $\bar{\boldsymbol{\epsilon}}_{ij}(t)$ ($\boldsymbol{\epsilon}_{ij}(t)$) denote the additive noise present at the link between node $i \in \mathcal{B}_j$ and node j when transmitting variable $\mathbf{v}_j(t)$ ($\boldsymbol{\alpha}_{ji}(t)$). The modified MoM-DSVM iterations are now given by

$$\widetilde{\boldsymbol{\lambda}}_j(t+1) = \underset{\boldsymbol{\lambda}_j:\, \mathbf{0}_j \preceq \boldsymbol{\lambda}_j \preceq JC\mathbf{1}_j}{\arg\max} \; -\frac{1}{2}\boldsymbol{\lambda}_j^T \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{X}_j^T \mathbf{Y}_j \boldsymbol{\lambda}_j + \left(\mathbf{1}_j - \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{f}_j(t)\right)^T \boldsymbol{\lambda}_j \quad (13a)$$

$$\mathbf{v}_j(t+1) = \mathbf{U}_j^{-1}\left[\mathbf{X}_j^T \mathbf{Y}_j \widetilde{\boldsymbol{\lambda}}_j(t+1) + \mathbf{f}_j(t)\right] \quad (13b)$$

$$\boldsymbol{\alpha}_{ji}(t+1) = \boldsymbol{\alpha}_{ji}(t) + \frac{\eta}{2}[\mathbf{v}_j(t+1) - \mathbf{v}_i(t+1) - \bar{\boldsymbol{\epsilon}}_{ij}(t)] \quad (13c)$$

where

$$\mathbf{f}_j(t) = \sum_{i \in \mathcal{B}_j}\{-[\boldsymbol{\alpha}_{ji}(t) - \boldsymbol{\alpha}_{ij}(t) - \boldsymbol{\epsilon}_{ij}(t)] + \eta[\mathbf{v}_j(t) + \mathbf{v}_i(t) + \bar{\boldsymbol{\epsilon}}_{ij}(t)]\}. \quad (14)$$

Algorithm 3 MoM-DSVM with noisy links

Require: Initialize $\mathbf{v}_j(0)$ and $\boldsymbol{\alpha}_{ji}(0)$, $\forall\, j \in \mathcal{J}$, $\forall i \in \mathcal{B}_j$.

for $t = 0, 1, 2, \ldots$ **do**
  for all $j \in \mathcal{J}$ **do**
    Compute $\boldsymbol{\lambda}_j(t+1)$ via (13a).
    Compute $\mathbf{v}_j(t+1)$ via (13b).
  end for
  for all $j \in \mathcal{J}$ **do**
    Broadcast $\mathbf{v}_j(t+1)$ to all neighbors $i \in \mathcal{B}_j$.
  end for
  for all $j \in \mathcal{J},\ i \in \mathcal{B}_j$ **do**
    Compute $\boldsymbol{\alpha}_{ji}(t+1)$ via (13c).
  end for
  for all $j \in \mathcal{J},\ i \in \mathcal{B}_j$ **do**
    Transmit $\boldsymbol{\alpha}_{ji}(t+1)$ to $i \in \mathcal{B}_j$.
  end for
end for

Similar to Algorithm 1, Lagrange multiplier iterates are initizalized as $\boldsymbol{\alpha}_{ji}(0) = \mathbf{0}$. The resulting MoM-DSVM algorithm with noisy links is summarized as Algorithm 3.

The perturbations $\{\boldsymbol{\epsilon}_{ji}(t)\}$ ($\{\overline{\boldsymbol{\epsilon}}_{ji}(t)\}$) are zero-mean random variables with covariance matrix $\mathbf{C}_{ji}$ ($\overline{\mathbf{C}}_{ji}$), white across time and space. Fig. 8 shows the average performance of MoM-DSVM with noisy links (cf. Algorithm 3) after 100 Monte Carlo runs for the same network of Test Case 5. As seen, the variance of the estimates $\mathbf{v}_j(t)$ yielded by the modified MoM-DSVM algorithm remains bounded.

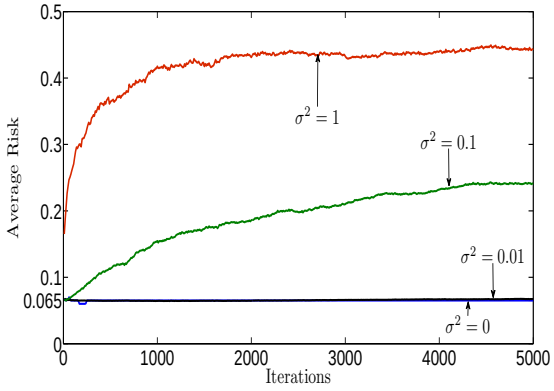

Figure 8: Average risk for a network with $J = 8$ and noisy communication links. Synthetic dataset.

5. CONCLUSIONS

This work developed distributed SVM algorithms capitalizing on a consensus-based formulation and using parallel optimization tools. The novel algorithms are well suited for applications that involve data that cannot be shared among nodes. Furthermore, power consumption is homogeneous across the network since MoM-DSVM maintains a fixed communication cost per iteration among neighboring nodes. The MoM-DSVM algorithm constructs a maximum-margin linear classifier using spatially distributed training sets. At every iteration, exchanges among neighboring nodes represent local linear classifier vectors. Convergence to the centralized linear SVM solution is guaranteed. An online and asynchronous version of the MoM-DSVM algorithm was also presented. This capability leads to possible integration of the DSVM algorithms in scenarios where elements of local training sets become available sequentially or need to be removed.

Extensions of this work to the nonlinear SVM case using kernels are possible. In this case, the resultant algorithms converge to the solution of a modified cost function whereby nodes agree on the classification decision on a subset of points. With reduced communication overhead and still maintaining data privacy, the preliminary classification tests confirm that the non-linear SVM closely approximates its centralized counterpart.

6. ACKNOWLEDGMENTS

This work was supported by NSF grants CCF 0830480 and CON 014658; and also through collaborative participation in the Communications and Networks Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

The authors also wish to thank Prof. A. Banerjee from the CS department at the University of Minnesota for his feedback on this and related topics.

APPENDIX

A. PROOF OF LEMMA 1

We want to show that iterations (5a)-(5d) correspond to the ADMoM iterations as in [2, Ch. 3]. First, define

$$\mathbf{v} := [\mathbf{v}_1^T, \ldots, \mathbf{v}_J^T, \boldsymbol{\xi}_1^T, \ldots, \boldsymbol{\xi}_J^T]^T \tag{15}$$

$$\widetilde{\boldsymbol{\omega}} := [\{\boldsymbol{\omega}_{1i}^T\}_{i\in\mathcal{B}_1}, \ldots, \{\boldsymbol{\omega}_{Ji}^T\}_{i\in\mathcal{B}_J}]^T \tag{16}$$

where $\mathbf{v}$ is a $((p+1)J + \sum_{j=1}^J N_j) \times 1$ vector and $\widetilde{\boldsymbol{\omega}}$ is a $(p+1)\sum_{j=1}^J |\mathcal{B}_j| \times 1$ vector. Moreover, let $\mathbf{A} := [\mathbf{A}_1, \mathbf{A}_2]$ where $\mathbf{A}_1 := \text{diag}\{\mathbf{A}_{11}, \ldots, \mathbf{A}_{1J}\}$ is a block diagonal matrix and $\mathbf{A}_{1j} := [\mathbf{I}_{p+1}, \ldots, \mathbf{I}_{p+1}]^T$ is a block matrix containing $2|\mathcal{B}_j|$ identity matrices; and, $\mathbf{A}_2$ is a $2(p+1)\sum_{j=1}^J |B_j| \times \sum_{j=1}^J N_j$ matrix of zeros. Next, define matrix $\widetilde{\mathbf{B}} := [\widetilde{\mathbf{B}}_1, \ldots, \widetilde{\mathbf{B}}_J]$ where the $J \times |\mathcal{B}_j|$ auxiliary matrix $\widetilde{\mathbf{B}}_j$ is defined as $\widetilde{\mathbf{B}}_j := [\{\boldsymbol{\chi}_{ji}\}_{i\in\mathcal{B}_j}]$; and, each $\boldsymbol{\chi}_{ji}$ is a $J \times 1$ indicator vector with a one at the j-th position, a one at the i-th position and zeros elsewhere. Matrix $\widetilde{\mathbf{B}}$ can be written in terms of its row entries as $\widetilde{\mathbf{B}} = [\mathbf{b}_1, \ldots, \mathbf{b}_J]^T$ where $\mathbf{b}_j$ is the transpose of the j-th row of $\widetilde{\mathbf{B}}$. A second auxiliary $2\sum_{j=1}^J |\mathcal{B}_j| \times \sum_{j=1}^J |\mathcal{B}_j|$ matrix $\overline{\mathbf{B}}$ is constructed block-wise as follows. For every j set $D = 2|\mathcal{B}_j|$ and visit every entry of $\mathbf{b}_j$ indexed by $h = 1, \ldots, \sum_{j=1}^J |\mathcal{B}_j|$. If $[\mathbf{b}_j]_h = 1$, then set the (j,h)-subblock

of $\overline{\mathbf{B}}$ to $\mathbf{e}_{jD}$, where $\mathbf{e}_{jD}$ is a $2|\mathcal{B}_j| \times 1$ vector with a one in the D-th entry and zeros elsewhere, update $D = D - 1$ and continue. Otherwise, set the (j,h)-subblock of $\overline{\mathbf{B}}$ to $\mathbf{0}_{2|\mathcal{B}_j|}$. Finally, let $\mathbf{B}$ be a $2(p+1)\sum_{j=1}^{J}|B_j| \times (p+1)\sum_{j=1}^{J}|\mathcal{B}_j|$ matrix. Using the previous definitions, we can define a new matrix $\mathbf{B}$ as $\mathbf{B} = \overline{\mathbf{B}} \otimes \mathbf{I}_{p+1}$ where $\otimes$ denotes the Kronecker product.

With these definitions, problem (3) can be written as

$$\begin{aligned} \min_{\mathbf{v},\boldsymbol{\omega}} \quad & G_1(\mathbf{v}) + G_2(\boldsymbol{\omega}) \\ \text{s.t.} \quad & \mathbf{A}\mathbf{v} = \boldsymbol{\omega} \\ & \mathbf{v} \in P_1,\ \boldsymbol{\omega} \in P_2 \end{aligned} \tag{17}$$

where $\boldsymbol{\omega} := \mathbf{B}\widetilde{\boldsymbol{\omega}}$, $G_1(\mathbf{v}) := \frac{1}{2}\sum_{j=1}^{J}\mathbf{v}_j^T(\mathbf{I}_{p+1} - \boldsymbol{\Pi}_{p+1})\mathbf{v}_j + JC\sum_{j=1}^{J}\mathbf{1}_j^T\boldsymbol{\xi}_j$, $G_2(\boldsymbol{\omega}) := 0$, $P_1 := \mathcal{W}_1 \times \ldots \times \mathcal{W}_J$ with $\mathcal{W}_j := \{\mathbf{v}_j, \boldsymbol{\xi}_j : \mathbf{Y}_j\mathbf{X}_j\mathbf{v}_j \succeq \mathbf{1}_j - \boldsymbol{\xi}_j, \boldsymbol{\xi}_j \succeq \mathbf{0}_j\}$, and $P_2 := \mathbb{R}^{p_2}$. Iterations (5a)-(5d) follow after applying the ADMoM steps in [2, Ch. 3] to (17).

B. PROOF OF LEMMA 2

Here, we establish that iterations (6a)-(6b) are equivalent to the ones in (5a)-(5d). Iteration (5b) corresponds to an unconstrained minimization problem over a quadratic cost function; thus, it accepts the closed-form solution for $\boldsymbol{\omega}_{ji}(t+1)$ as

$$\boldsymbol{\omega}_{ji}(t+1) = \frac{1}{2\eta}(\boldsymbol{\alpha}_{ji1}(t) - \boldsymbol{\alpha}_{ji2}(t)) + \frac{1}{2}(\mathbf{v}_j(t+1) + \mathbf{v}_i(t+1)). \tag{18}$$

Substituting (18) into (5c) and (5d), we obtain

$$\boldsymbol{\alpha}_{ji1}(t+1) = \frac{1}{2}(\boldsymbol{\alpha}_{ji1}(t) + \boldsymbol{\alpha}_{ji2}(t)) + \frac{\eta}{2}(\mathbf{v}_j(t+1) - \mathbf{v}_i(t+1)) \tag{19a}$$

$$\boldsymbol{\alpha}_{ji2}(t+1) = \frac{1}{2}(\boldsymbol{\alpha}_{ji1}(t) + \boldsymbol{\alpha}_{ji2}(t)) + \frac{\eta}{2}(\mathbf{v}_j(t+1) - \mathbf{v}_i(t+1)). \tag{19b}$$

Suppose that $\boldsymbol{\alpha}_{ji1}(t)$ and $\boldsymbol{\alpha}_{ji2}(t)$ are initialized to zero at every node j, i.e., $\boldsymbol{\alpha}_{ji1}(0) = \boldsymbol{\alpha}_{ji2}(0) = \mathbf{0}\ \forall j \in \mathcal{J}$ and $\forall i \in \mathcal{B}_j$. From (19a)-(19b), it follows that $\boldsymbol{\alpha}_{ji1}(1) = \boldsymbol{\alpha}_{ji2}(1)$. Similarly, if $\boldsymbol{\alpha}_{ji1}(t-1) = \boldsymbol{\alpha}_{ji2}(t-1)$, then by induction $\boldsymbol{\alpha}_{ji1}(t) = \boldsymbol{\alpha}_{ji2}(t)$. Thus, only one set of variables, say $\{\boldsymbol{\alpha}_{ji1}\}$, needs to be stored and updated per node j.

Substituting $\boldsymbol{\omega}_{ji}(t+1) = \frac{1}{2}[\mathbf{v}_j(t+1) + \mathbf{v}_i(t+1)]$ in (4), we have that

$$\begin{aligned} \mathcal{L}'(\{\mathbf{v}_j\},\{\boldsymbol{\xi}_j\},\{\mathbf{v}_j(t)\},\{\boldsymbol{\alpha}_{ji1}(t)\}) &= JC\sum_{j=1}^{J}\mathbf{1}_j^T\boldsymbol{\xi}_j \\ &+ \frac{1}{2}\sum_{j=1}^{J}\mathbf{v}_j^T(\mathbf{I}_{p+1} - \boldsymbol{\Pi}_{p+1})\mathbf{v}_j + \sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\boldsymbol{\alpha}_{ji1}^T(t)(\mathbf{v}_j - \mathbf{v}_i) \\ &+ \frac{\eta}{2}\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\left\|\mathbf{v}_j - \frac{1}{2}[\mathbf{v}_j(t) + \mathbf{v}_i(t)]\right\|^2 \\ &+ \frac{\eta}{2}\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\left\|\mathbf{v}_i - \frac{1}{2}[\mathbf{v}_j(t) + \mathbf{v}_i(t)]\right\|^2. \end{aligned} \tag{20}$$

The third term in (20) can be rewritten as

$$\begin{aligned} \sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\boldsymbol{\alpha}_{ji1}^T(t)(\mathbf{v}_j - \mathbf{v}_i) &= \sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\mathbf{v}_j^T(\boldsymbol{\alpha}_{ji1}(t) - \boldsymbol{\alpha}_{ij1}(t)) \\ &= 2\sum_{j=1}^{J}\mathbf{v}_j^T\sum_{i\in\mathcal{B}_j}\boldsymbol{\alpha}_{ji1}(t) \end{aligned} \tag{21}$$

where the second equality comes from the fact that $\boldsymbol{\alpha}_{ji2}(t) = -\boldsymbol{\alpha}_{ij2}(t)\ \forall t$ (follows by induction from (19a)-(19b)). Likewise, the fourth and fifth terms in (20) can be rewritten as

$$\begin{aligned} \frac{\eta}{2}\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\left\{\left\|\mathbf{v}_j - \frac{1}{2}[\mathbf{v}_j(t) + \mathbf{v}_i(t)]\right\|^2 + \left\|\mathbf{v}_i - \frac{1}{2}[\mathbf{v}_j(t) + \mathbf{v}_i(t)]\right\|^2\right\} \\ = \eta\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\left\|\mathbf{v}_j - \frac{1}{2}[\mathbf{v}_j(t) + \mathbf{v}_i(t)]\right\|^2. \end{aligned} \tag{22}$$

Lemma 2 follows after substituting (21) and (22) into (20), and defining $\boldsymbol{\alpha}_j(t) := \sum_{i\in\mathcal{B}_j}\boldsymbol{\alpha}_{ji1}(t)$.

C. PROOF OF PROPOSITION 1

The Lagrangian for (6a) is given by

$$\begin{aligned} &\mathcal{L}''(\{\mathbf{v}_j\},\{\boldsymbol{\xi}_j\},\{\boldsymbol{\lambda}_j\},\{\boldsymbol{\gamma}_j\},\{\mathbf{v}_j(t)\},\{\boldsymbol{\alpha}_j(t)\}) \\ &= \frac{1}{2}\sum_{j=1}^{J}\mathbf{v}_j^T(\mathbf{I}_{p+1} - \boldsymbol{\Pi}_{p+1})\mathbf{v}_j + JC\sum_{j=1}^{J}\mathbf{1}_j^T\boldsymbol{\xi}_j \\ &\quad - \sum_{j=1}^{J}\boldsymbol{\lambda}_j^T(\mathbf{Y}_j\mathbf{X}_j\mathbf{v}_j - \mathbf{1}_j + \boldsymbol{\xi}_j) - \sum_{j=1}^{J}\boldsymbol{\gamma}_j^T\boldsymbol{\xi}_j \\ &\quad + 2\sum_{j=1}^{J}\boldsymbol{\alpha}_j^T(t)\mathbf{v}_j + \eta\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\left\|\mathbf{v}_j - \frac{1}{2}[\mathbf{v}_j(t) + \mathbf{v}_i(t)]\right\|^2. \end{aligned} \tag{23}$$

The KKT conditions provide expressions for both primal and dual variables in (23) as follows

$$\mathbf{v}_j(t+1) = \mathbf{U}_j^{-1}\left(\mathbf{X}_j^T\mathbf{Y}_j\boldsymbol{\lambda}_j(t+1) - 2\boldsymbol{\alpha}_j(t) + \eta\sum_{i\in\mathcal{B}_j}[\mathbf{v}_j(t) + \mathbf{v}_i(t)]\right) \tag{24}$$

$$\mathbf{0}_j = JC\mathbf{1}_j - \boldsymbol{\lambda}_j - \boldsymbol{\gamma}_j \tag{25}$$

where $\boldsymbol{\lambda}_j(t+1)$ is the optimal Lagrange multiplier after iteration $t+1$ and $\mathbf{U}_j := (1 + 2\eta|\mathcal{B}_j|)\mathbf{I}_{p+1} - \boldsymbol{\Pi}_{p+1}$. Notice that $\mathbf{U}_j^{-1}$ is always well defined.

The KKT conditions also require $\boldsymbol{\lambda}_j \succeq \mathbf{0}_j$ and $\boldsymbol{\gamma}_j \succeq \mathbf{0}_j$. Thus, (25) is summarized by $\mathbf{0}_j \preceq \boldsymbol{\lambda}_j \preceq JC\mathbf{1}_j$. To compute iteration (24) at every node, the optimal values $\boldsymbol{\lambda}_j(t+1)$ of the Lagrange multipliers $\boldsymbol{\lambda}_j$ are found by solving the Lagrange dual problem associated with (23). The Lagrange dual

function $\mathcal{L}_\lambda(\{\boldsymbol{\lambda}_j\})$ is

$$\mathcal{L}_\lambda(\{\boldsymbol{\lambda}_j\}) = -\frac{1}{2}\sum_{j=1}^{J} \boldsymbol{\lambda}_j^T \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{X}_j^T \mathbf{Y}_j \boldsymbol{\lambda}_j + \left(\mathbf{1}_j - \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{f}_j(t)\right)^T \boldsymbol{\lambda}_j \quad (26)$$

where $\mathbf{f}_j(t) := -2\boldsymbol{\alpha}_{ji1}(t) + \eta \sum_{i\in\mathcal{B}_j}[\mathbf{v}_j(t) + \mathbf{v}_i(t)]$. Note that the Lagrange multipliers $\boldsymbol{\gamma}_j$ are not present in the dual function. From (26), the Lagrange dual problem can be decoupled if each node j has access to the $\mathbf{v}_i(t)$ estimates of its neighboring nodes. Thus, $\boldsymbol{\lambda}_j(t+1)$ is given by

$$\boldsymbol{\lambda}_j(t+1) = \underset{\boldsymbol{\lambda}_j:\, \mathbf{0}_j \preceq \boldsymbol{\lambda}_j \preceq JC\mathbf{1}_j}{\arg\max} -\frac{1}{2}\boldsymbol{\lambda}_j^T \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{X}_j^T \mathbf{Y}_j \boldsymbol{\lambda}_j + \left(\mathbf{1}_j - \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{f}_j(t)\right)^T \boldsymbol{\lambda}_j. \quad (27)$$

Hence, the dual variable update in (27) and the KKT condition in (24) result in iterations (8a) and (8b).

So far we have proved that iterations (6a) and (6b) from Lemma 2 result in iterations (8a), (8b) and (8c) in Proposition 1. Iterations (6a) and (6b) are equivalent to the ones in (5a)-(5d) [cf. Appendix B], which result after applying the ADMoM of [2, Ch. 3]. Since the cost function in (3) is convex and its constraints comply with [2, Assumption 4.1, pg. 255], the sequences induced by iterations (5a)-(5d) converge to the optimal solution of (3) for any positive value of η, as shown in [2, Proposition 4.2, pg. 256], and so do (8a)-(8c) in Proposition 1.

D. REFERENCES

[1] I. F. Akyildiz, D. Pompili, and T. Melodia. Underwater acoustic sensor networks: Research challenges. *Ad Hoc Networks (Elsevier)*, 3(3):257–279, Mar. 2005.

[2] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[3] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Proc. of Neural Info. Processing Systems Conf.*, pages 409–415, Denver, CO, USA, Nov. 2000.

[4] E. Y. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, and H. Cui. PSVM: Parallelizing support vector machines on distributed computers. In *21st Neural Info. Processing Systems Conf.*, Vancouver, Canada, Dec. 3-6, 2007.

[5] T. Do and F. Poulet. Classifying one billion data with a new distributed SVM algorithm. In *Proc. of International Conf. on Research, Innovation and Vision for the Future*, pages 59–66, Ho Chi Minh City, Vietnam, Feb. 12-16 2006.

[6] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2nd edition, 2002.

[7] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proc. of 3rd Theory of Cryptography Conference*, pages 265–284, New York, NY, USA, Mar. 4-7 2006.

[8] I. El-Naqa, Y. Yang, M. Wernick, N. Galatsanos, and R. Nishikawa. A support vector machine approach for detection of microcalcifications. *IEEE Tran. on Medical Imaging*, 21(12):1552–1563, Dec. 2002.

[9] K. Flouri, B. Beferull-Lozano, and P. Tsakalides. Training a support-vector machine-based classifier in distributed sensor networks. In *Proc. of 14nd European Signal Processing Conf.*, Florence, Italy, Sep. 4-8 2006.

[10] K. Flouri, B. Beferull-Lozano, and P. Tsakalides. Distributed consensus algorithms for SVM training in wireless sensor networks. In *Proc. of 16nd European Signal Processing Conf.*, Laussane, Switzerland, Aug. 25-29 2008.

[11] G. Fung and O. L. Mangasarian. Incremental support vector machine classification. In *Proc. of the 2nd SIAM International Conf. on Data Mining*, pages 247–260, Arlington, VA, USA, Apr. 11-13 2002.

[12] A. Ganapathiraju, J. Hamaker, and J. Picone. Applications of support vector machines to speech recognition. *IEEE Tran. on Signal Processing*, 52(8):2348–2355, Aug. 2004.

[13] H. P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik. Parallel support vector machines: The cascade SVM. In *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005.

[14] H. Zhu, G. B. Giannakis, and A. Cano. Distributed In-Network Channel Decoding. *IEEE Tran. on Signal Processing*, 57(10):3970–3983, Oct. 2009.

[15] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *Proc. of the Seventeenth International Conf. on Machine Learning*, pages 487–494, Stanford, CA, USA, 2000.

[16] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, Nov. 1998.

[17] Y. Liang, M. Reyes, and J. Lee. Real-time detection of driver cognitive distraction using support vector machines. *IEEE Tran. on Intelligent Transportation Systems*, 8(2):340–350, Jun. 2007.

[18] Y. Lu, V. Roychowdhury, and L. Vandenberghe. Distributed parallel support vector machines in strongly connected networks. *IEEE Tran. on Neural Networks*, 19(7):1167–1178, Jul. 2008.

[19] C. H. Papadimitriou, *Computational Complexity.* Addison- Wesley, 1993.

[20] B. Schölkopf and A. Smola. *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[21] V. Vapnik. *Statistical Learning Theory*. Wiley, 1st edition, 1998.

[22] J. Weston, A. Elisseeff, G. BakIr, and F. Sinz. The spider machine learning toolbox, 2006.