Journal of Machine Learning Research 11 (2010) 1663-1707 Submitted 6/09; Revised 1/10; Published 5/10

Consensus-Based Distributed Support Vector Machines

Pedro A. Forero FORER002@UMN.EDU
Alfonso Cano ALFONSO@UMN.EDU
Georgios B. Giannakis GEORGIOS@UMN.EDU
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, MN 55455, USA

Editor: Sathiya Keerthi

Abstract

This paper develops algorithms to train support vector machines when training data are distributed across different nodes, and their communication to a centralized processing unit is prohibited due to, for example, communication complexity, scalability, or privacy reasons. To accomplish this goal, the centralized linear SVM problem is cast as a set of decentralized convex optimization subproblems (one per node) with consensus constraints on the wanted classifier parameters. Using the alternating direction method of multipliers, fully distributed training algorithms are obtained without exchanging training data among nodes. Different from existing incremental approaches, the overhead associated with inter-node communications is fixed and solely dependent on the network topology rather than the size of the training sets available per node. Important generalizations to train nonlinear SVMs in a distributed fashion are also developed along with sequential variants capable of online processing. Simulated tests illustrate the performance of the novel algorithms.[1]

Keywords: support vector machine, distributed optimization, distributed data mining, distributed learning, sensor networks

1. Introduction

Problems calling for *distributed learning* solutions include those arising when training data are acquired by different nodes, and their communication to a central processing unit, often referred to as fusion center (FC), is costly or even discouraged due to, for example, scalability, communication overhead, or privacy reasons. Indeed, in applications involving wireless sensor networks (WSNs) with battery-operated nodes, transferring each sensor's data to the FC maybe prohibited due to power limitations. In other cases, nodes gathering sensitive or private information needed to design the classifier may not be willing to share their training data.

For *centralized learning* on the other hand, the merits of support vector machines (SVMs) have been well documented in various supervised classification tasks emerging in applications such as medical imaging, bio-informatics, speech, and handwriting recognition, to name a few (Vapnik, 1998; Schölkopf and Smola, 2002; El-Naqa et al., 2002; Liang et al., 2007; Ganapathiraju et al., 2004; Li, 2005; Markowska-Kaczmar and Kubacki, 2005). Centralized SVMs are maximum-margin

1. Prepared 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.

linear classifiers designed based on a centrally available training set comprising multidimensional data with corresponding classification labels. Training an SVM requires solving a quadratic optimization problem of dimensionality dependent on the cardinality of the training set. The resulting linear SVM discriminant depends on a subset of elements from the training set, known as support vectors (SVs). Application settings better suited for nonlinear discriminants have been also considered by mapping vectors at the classifier's input to a higher dimensional space, where linear classification is performed. In either linear or nonlinear SVMs designed with centrally available training data, the decision on new data to be classified is based solely on the SVs.

For this reason, recent designs of *distributed SVM* classifiers rely on SVs obtained from local training sets (Flouri et al., 2006, 2008; Lu et al., 2008). These SVs obtained locally per node are incrementally passed on to neighboring nodes, and further processed at the FC to obtain a discriminant function approaching the centralized one obtained as if all training sets were centrally available. Convergence of the *incremental* distributed (D) SVM to the centralized SVM requires multiple SV exchanges between the nodes and the FC (Flouri et al., 2006); see also Flouri et al. (2008), where convergence of a gossip-based DSVM is guaranteed when classes are linearly separable. Without updating local SVs through node-FC exchanges, DSVM schemes can approximate but not ensure the performance of centralized SVM classifiers (Navia-Vazquez et al., 2006).

Another class of DSVMs deals with *parallel* designs of centralized SVMs—a direction well motivated when training sets are prohibitively large (Chang et al., 2007; Do and Poulet, 2006; Graf et al., 2005; Bordes et al., 2005). Partial SVMs obtained using small training subsets are combined at a central processor. These parallel designs can be applied to distributed networked nodes, only if a central unit is available to judiciously combine partial SVs from intermediate stages. Moreover, convergence to the centralized SVM is generally not guaranteed for any partitioning of the aggregate data set (Graf et al., 2005; Bordes et al., 2005).

The novel approach pursued in the present paper trains an SVM in a *fully distributed* fashion that does not require a central processing unit. The centralized SVM problem is cast as a set of coupled decentralized convex optimization subproblems with consensus constraints imposed on the desired classifier parameters. Using the alternating direction method of multipliers (ADMoM), see, for example, Bertsekas and Tsitsiklis (1997), distributed training algorithms that are provably convergent to the centralized SVM are developed based solely on message exchanges among neighboring nodes. Compared to existing alternatives, the novel DSVM classifier offers the following distinct features.

- **Scalability and reduced communication overhead**. Compared to approaches having distributed nodes communicate training samples to an FC, the DSVM approach here relies on *in-network* processing with messages exchanged only among single-hop neighboring nodes. This keeps the communication overhead per node at an affordable level within its neighborhood, even when the network scales to cover a larger geographical area. In FC-based approaches however, nodes consume increased resources to reach the FC as the coverage area grows. Different from, for example, Lu et al. (2008), and without exchanging SVs, the novel DSVM incurs a fixed overhead for inter-node communications per iteration regardless of the size of the local training sets.

- **Robustness to isolated point(s) of failure**. If the FC fails, an FC-based SVM design will fail 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 algorithm will converge

to a classifier trained using the data of nodes that remain operational. But even if the network becomes disconnected, the proposed algorithm will stay operational with performance dependent on the number of training samples per connected sub-network.

- **Fully decentralized network operation**. Alternative distributed approaches include incremental and parallel SVMs. Incremental passing of local SVs requires identification of a Hamiltonian cycle (going through all nodes once) in the network (Lu et al., 2008; Flouri et al., 2006). And 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 (Papadimitriou, 2006). 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; see, for example, Chang et al. (2007), Do and Poulet (2006) and Graf et al. (2005).

- **Convergence guarantees to centralized SVM performance**. For linear SVMs, the novel DSVM algorithm is *provably convergent* to a centralized SVM classifier, as if all distributed samples were available centrally. For nonlinear SVMs, it converges to the solution of a modified cost function whereby nodes agree on the classification decision for a subset of points. If those points correspond to a classification query, the network "agrees on" the classification of these points with performance identical to the centralized one. For other classification queries, nodes provide classification results that closely approximate the centralized SVM.

- **Robustness to noisy inter-node communications and privacy preservation**. The novel DSVM scheme is robust to various sources of disturbance that maybe present in message exchanges. Those can be due to, for example, quantization errors, additive Gaussian receiver noise, or, Laplacian noise intentionally added for privacy (Dwork et al., 2006; Chaudhuri and Monteleoni, 2008).

The rest of this paper is organized as follows. To provide context, Section 2 outlines the centralized linear and nonlinear SVM designs. Section 3 deals with the novel fully distributed linear SVM algorithm. Section 3.1 is devoted to an online DSVM algorithm for synchronous and asynchronous updates, while Section 4 generalizes the DSVM formulation to allow for nonlinear classifiers using kernels. Finally, Sections 5 and 6 present numerical results and concluding remarks.

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 $[\cdot]_{ji}$ ($[\cdot]_j$); diag($\mathbf{x}$) denotes a diagonal matrix with $\mathbf{x}$ on its main diagonal; diag$\{\cdot\}$ is a (block) diagonal matrix with the elements in $\{\cdot\}$ on its diagonal; $|\cdot|$ denotes set cardinality; $\succeq$ ($\preceq$) element-wise $\geq$ ($\leq$); $\{\cdot\}$ ($[\cdot]$) a set (matrix) of variables with appropriate elements (entries); $\|\cdot\|$ 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 $\mathcal{N}(m, \Sigma)$ for the multivariate Gaussian distribution with mean m, and covariance matrix Σ.

2. Preliminaries and Problem Statement

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 com-

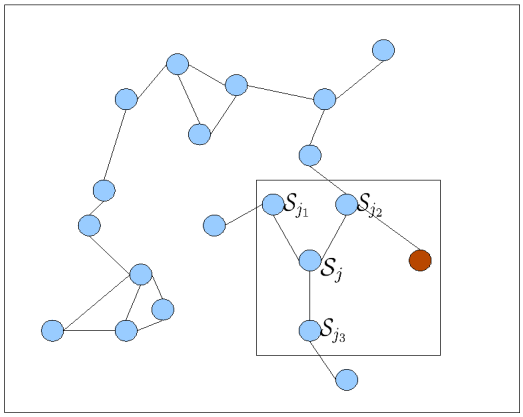

Figure 1: Network example where connectivity among nodes, represented by colored circles, is denoted by a line joining them.

municating 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, that is, any two nodes in $\mathcal{G}$ are connected by a (perhaps multihop) path in $\mathcal{G}$. Notice that nodes do not have to be fully connected (any-to-any), and $\mathcal{G}$ is allowed to 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 $\mathbf{x}_{jn} \in \mathcal{X}$ is a $p \times 1$ data vector belonging to the input space $\mathcal{X} \subseteq \mathbb{R}^p$, and $y_{jn} \in \mathcal{Y} := \{-1, 1\}$ denotes its corresponding class label.[2]

Given $\mathcal{S}_j$ per node j, the *goal* is to find a maximum-margin linear discriminant function $g(\mathbf{x})$ in a distributed fashion, and thus enable each node to classify any new input 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 include but are not limited to the following ones.

Example 1 *(Wireless sensor networks).* Consider a set of wireless sensors deployed to infer the presence or absence of a pollutant over a geographical area at any time t_n. Sensor j measures and forms a local binary decision variable $y_{jn} \in \{1, -1\}$, where $y_{jn} = 1(-1)$ indicates presence (absence) of the pollutant at the position vector $\mathbf{x}_j := [x_{j1},\ x_{j2},\ x_{j3}]^T$. (Each sensor knows its position $\mathbf{x}_j$ using existing self-localization algorithms Langendoen and Reijers, 2003.) The goal is to have each low-cost sensor improve the performance of local detection achieved based on $\mathcal{S}_j = \{([\mathbf{x}_j^T, t_n]^T, y_{jn}) : n = 1, \ldots, N_j\}$, and through collaboration with other sensors approach the global performance attainable if each sensor had available all other sensors data. Stringent power limitations prevent sensor j to send its set $\mathcal{S}_j$ to all other sensors or to an FC, if the latter is available. If these sensors are acoustic and are deployed to classify underwater unmanned vehicles, divers or submarines, then low transmission bandwidth and multipath effects further discourage incremental communication of the local data sets to an FC (Akyildiz et al., 2005).

Example 2 *(Distributed medical databases).* Suppose that $\mathcal{S}_j$ are patient data records stored at a hospital j. Each $\mathbf{x}_{jn}$ here contains patient descriptors (e.g., age, sex or blood pressure), and y_{jn} is a particular diagnosis (e.g., the patient is diabetic or not). The objective is to automatically diagnose

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

(classify) a patient arriving at hospital j with descriptor $\mathbf{x}$, using all available data $\{\mathcal{S}_j\}_{j=1}^J$, rather than $\mathcal{S}_j$ alone. However, a nonchalant exchange of database entries $(\mathbf{x}_{jn}^T, y_{jn})$ can pose a privacy risk for the information exchanged. Moreover, a large percentage of medical information may require exchanging high resolution images. Thus, communicating and processing large amounts of high-dimensional medical data at an FC may be computationally prohibitive.

Example 3 *(Collaborative data mining).* Consider two different government agencies, a local agency A and a nation-wide agency B, with corresponding databases $\mathcal{S}_A$ and $\mathcal{S}_B$. Both agencies are willing to collaborate in order to classify jointly possible security threats. However, lower clearance level requirements at agency A prevents agency B from granting agency A open access to $\mathcal{S}_B$. Furthermore, even if an agreement granting temporary access to agency A were possible, databases $\mathcal{S}_A$ and $\mathcal{S}_B$ are confined to their current physical locations due to security policies.

If $\{\mathcal{S}_j\}_{j=1}^J$ were all centrally available at an FC, then the global 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 convex optimization problem; see, for example, Schölkopf and Smola (2002, Ch. 7)

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

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

Nonlinear discriminant functions $g(\mathbf{x})$ can also be found along the lines of (1) after mapping vectors $\mathbf{x}_{jn}$ to a higher dimensional space $\mathcal{H} \subseteq \mathbb{R}^P$, with $P > p$, via a nonlinear transformation $\phi : \mathcal{X} \to \mathcal{H}$. The generalized maximum-margin linear classifier in $\mathcal{H}$ is then obtained after replacing $\mathbf{x}_{jn}$ with $\phi(\mathbf{x}_{jn})$ in (1), and solving the following optimization problem

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

Problem (2) is typically tackled by solving its dual. Letting λ_{jn} denote the Lagrange multiplier corresponding to the constraint $y_{jn}(\mathbf{w}^T\phi(\mathbf{x}_{jn}) + b) \geq 1 - \xi_{jn}$, the dual problem of (2) is:

$$\begin{aligned} \max_{\{\lambda_{jn}\}} \quad & -\frac{1}{2}\sum_{j=1}^{J}\sum_{i=1}^{J}\sum_{n=1}^{N_j}\sum_{m=1}^{N_i}\lambda_{jn}\lambda_{im}y_{jn}y_{im}\phi^T(\mathbf{x}_{jn})\phi(\mathbf{x}_{im}) + \sum_{j=1}^{J}\sum_{n=1}^{N_j}\lambda_{jn} \\ \text{s.t.} \quad & \sum_{j=1}^{J}\sum_{n=1}^{N_j}\lambda_{jn}y_{jn} = 0 \\ & 0 \leq \lambda_{jn} \leq C \qquad\qquad \forall j \in \mathcal{J},\ n = 1, \ldots, N_j. \end{aligned} \tag{3}$$

Using the Lagrange multipliers λ_{jn}^* optimizing (3), and the Karush-Kuhn-Tucker (KKT) optimality conditions, the optimal classifier parameters can be expressed as

$$\begin{aligned} \mathbf{w}^* &= \sum_{j=1}^{J} \sum_{n=1}^{N_j} \lambda_{jn}^* y_{jn} \phi(\mathbf{x}_{jn}), \\ b^* &= y_{jn} - \mathbf{w}^{*T} \phi(\mathbf{x}_{jn}) \end{aligned} \tag{4}$$

with $\mathbf{x}_{jn}$ in (4) satisfying $\lambda_{jn}^* \in (0, C)$. Training vectors corresponding to non-zero λ_{jn}^*'s constitute the SVs. Once the SVs are identified, all other training vectors with $\lambda_{jn}^* = 0$ can be discarded since they do not contribute to $\mathbf{w}^*$. From this vantage point, SVs are the most informative elements of the training set. Solving (3) does not require knowledge of ϕ but only inner product values $\phi^T(\mathbf{x}_{jn})\phi(\mathbf{x}_{im}) := K(\mathbf{x}_{jn}, \mathbf{x}_{im})$, which can be computed through a pre-selected positive semi-definite kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$; see, for example, Schölkopf and Smola (2002, Ch. 2). Although not explicitly given, the optimal slack variables ξ_{jn}^* can be found through the KKT conditions of (2) in terms of λ_{jn}^* (Schölkopf and Smola, 2002). The optimal discriminant function can be also expressed in terms of kernels as

$$g^*(\mathbf{x}) = \sum_{j=1}^{J} \sum_{n=1}^{N_j} \lambda_{jn}^* y_{jn} K(\mathbf{x}_{jn}, \mathbf{x}) + b^* \tag{5}$$

where $b^* = y_{jn} - \sum_{i=1}^{J} \sum_{m=1}^{N_i} \lambda_{im}^* y_{im} K(\mathbf{x}_{im}, \mathbf{x}_{jn})$ for any SV $\mathbf{x}_{jn}$ with $\lambda_{jn}^* \in (0, C)$. This so-called kernel trick allows finding maximum-margin linear classifiers in higher dimensional spaces without explicitly operating in such spaces (Schölkopf and Smola, 2002).

The objective here is to develop *fully distributed* solvers of the centralized problems in (1) and (2) while guaranteeing performance approaching that of a centralized equivalent SVM. Although incremental solvers are possible, the size of information exchanges required might be excessive, especially if the number of SVs per node is large (Flouri et al., 2008; Lu et al., 2008). Recall that exchanging all local SVs among all nodes in the network several times is necessary for incremental DSVMs to approach the optimal centralized solution. Moreover, incremental schemes require a Hamiltonian cycle in the network to be identified in order to minimize the communication overhead. 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. In other situations, communicating SVs directly might be prohibited because of the sensitivity of the information bore, as already mentioned in Examples 2 and 3.

3. Distributed Linear Support Vector Machine

This section presents a reformulation of the maximum-margin linear classifier problem in (1) to an equivalent distributed form, which can be solved using the alternating direction method of multipliers (ADMoM) outlined in Appendix A. (For detailed exposition of the ADMoM, see, for example, Bertsekas and Tsitsiklis, 1997.)

To this end, consider replacing the common (coupling) variables $(\mathbf{w}, b)$ in (1) with auxiliary per-node variables $\{(\mathbf{w}_j, b_j)\}_{j=1}^{J}$, and adding consensus constraints to force these variables to agree across neighboring nodes. With proper scaling of the cost by J, the proposed *consensus-based*

reformulation of (1) becomes

$$\begin{aligned} \min_{\{\mathbf{w}_j, b_j, \xi_{jn}\}} \quad & \frac{1}{2}\sum_{j=1}^{J} \|\mathbf{w}_j\|^2 + JC\sum_{j=1}^{J}\sum_{n=1}^{N_j} \xi_{jn} \\ \text{s.t.} \quad & y_{jn}(\mathbf{w}_j^T \mathbf{x}_{jn} + b_j) \geq 1 - \xi_{jn} \qquad \forall j \in \mathcal{J},\ n = 1,\ldots,N_j \\ & \xi_{jn} \geq 0 \qquad \forall j \in \mathcal{J},\ n = 1,\ldots,N_j \\ & \mathbf{w}_j = \mathbf{w}_i,\ b_j = b_i \qquad \forall j \in \mathcal{J},\ i \in \mathcal{B}_j. \end{aligned} \tag{6}$$

From a high-level view, problem (6) can be solved in a distributed fashion because each node j can optimize only the j-dependent terms of the cost, and also meet all the consensus constraints $\mathbf{w}_j = \mathbf{w}_i,\ b_j = b_i$, by exchanging messages only with nodes i in its neighborhood $\mathcal{B}_j$. What is more, network connectivity ensures that consensus in neighborhoods enables network-wide consensus. And thus, as the ensuing lemma asserts, solving (6) is equivalent to solving (1) so long as the network remains connected.

Lemma 1 *If* $\{(\mathbf{w}_j, b_j)\}_{j=1}^{J}$ *denotes a feasible solution of* (6)*, and the graph* $\mathcal{G}$ *is connected, then problems* (1) *and* (6) *are equivalent, that is,* $\mathbf{w}_j = \mathbf{w}$ *and* $b_j = b\ \forall j = 1,\ldots,J$*, where* $(\mathbf{w}, b)$ *is a feasible solution of* (1)*.*

Proof See Appendix B. ■

To specify how (6) can be solved using the ADMoM, define for notational brevity the augmented vector $\mathbf{v}_j := [\mathbf{w}_j^T,\ b_j]^T$, the augmented 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 vector of slack variables $\xi_j := [\xi_{j1},\ldots,\xi_{jN_j}]^T$. With these definitions, it follows readily that $\mathbf{w}_j = (\mathbf{I}_{p+1} - \Pi_{p+1})\mathbf{v}_j$, where Π_{p+1} is a $(p+1)\times(p+1)$ matrix with zeros everywhere except for the $(p+1, p+1)$-st entry, given by $[\Pi_{p+1}]_{(p+1)(p+1)} = 1$. Thus, problem (6) can be rewritten as

$$\begin{aligned} \min_{\{\mathbf{v}_j, \xi_j, \omega_{ji}\}} \quad & \frac{1}{2}\sum_{j=1}^{J} \mathbf{v}_j^T(\mathbf{I}_{p+1} - \Pi_{p+1})\mathbf{v}_j + JC\sum_{j=1}^{J} \mathbf{1}_j^T \xi_j \\ \text{s.t.} \quad & \mathbf{Y}_j\mathbf{X}_j\mathbf{v}_j \succeq \mathbf{1}_j - \xi_j \qquad \forall j \in \mathcal{J} \\ & \xi_j \succeq \mathbf{0}_j \qquad \forall j \in \mathcal{J} \\ & \mathbf{v}_j = \omega_{ji},\ \omega_{ji} = \mathbf{v}_i \qquad \forall j \in \mathcal{J},\ \forall i \in \mathcal{B}_j \end{aligned} \tag{7}$$

where the redundant variables $\{\omega_{ji}\}$ will turn out to facilitate the decoupling of the classifier parameters $\mathbf{v}_j$ at node j from those of their neighbors at neighbors $i \in \mathcal{B}_j$.

As in the centralized case, problem (7) will be solved through its dual. Toward this objective, let α_{ji1} (α_{ji2}) denote the Lagrange multipliers corresponding to the constraint $\mathbf{v}_j = \omega_{ji}$ (respectively $\omega_{ji} = \mathbf{v}_i$), and consider what we term surrogate augmented Lagrangian function

$$\begin{aligned} \mathcal{L}(\{\mathbf{v}_j\},\{\xi_j\},\{\omega_{ji}\},\{\alpha_{jik}\}) = & \frac{1}{2}\sum_{j=1}^{J} \mathbf{v}_j^T(\mathbf{I}_{p+1} - \Pi_{p+1})\mathbf{v}_j + JC\sum_{j=1}^{J}\mathbf{1}_j^T\xi_j + \sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\alpha_{ji1}^T(\mathbf{v}_j - \omega_{ji}) \\ & + \sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\alpha_{ji2}^T(\omega_{ji} - \mathbf{v}_i) + \frac{\eta}{2}\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\|\mathbf{v}_j - \omega_{ji}\|^2 + \frac{\eta}{2}\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\|\omega_{ji} - \mathbf{v}_i\|^2 \end{aligned} \tag{8}$$

where the adjective "surrogate" is used because $\mathcal{L}$ does not include the set of constraints $\mathcal{W} := \{\mathbf{Y}_j\mathbf{X}_j\mathbf{v}_j \succeq \mathbf{1}_j - \xi_j,\ \xi_j \succeq \mathbf{0}_j\}$, and the adjective "augmented" because $\mathcal{L}$ includes two quadratic terms (scaled by the tuning constant $\eta > 0$) to further regularize the equality constraints in (7). The role of these quadratic terms $||\mathbf{v}_j - \omega_{ji}||^2$ and $||\omega_{ji} - \mathbf{v}_i||^2$ is twofold: (a) they effect *strict* convexity of $\mathcal{L}$ with respect to (w.r.t.) ω_{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 trade off speed of convergence for steady-state approximation error (Bertsekas and Tsitsiklis, 1997, Ch. 3).

Consider now solving (7) iteratively by minimizing $\mathcal{L}$ in a cyclic fashion with respect to one set of variables while keeping all other variables fixed. The multipliers $\{\alpha_{ji1}, \alpha_{ji2}\}$ must be also updated per iteration using gradient ascent. The iterations required per node j are summarized in the following lemma.

Lemma 2 *The distributed iterations solving* (7) *are*

$$\{\mathbf{v}_j(t+1), \xi_j(t+1)\} = \arg \min_{\{\mathbf{v}_j, \xi_j\} \in \mathcal{W}} \mathcal{L}(\{\mathbf{v}_j\}, \{\xi_j\}, \{\omega_{ji}(t)\}, \{\alpha_{jik}(t)\}), \tag{9}$$

$$\{\omega_{ji}(t+1)\} = \arg \min_{\{\omega_{ji}\}} \mathcal{L}(\{\mathbf{v}_j(t+1)\}, \{\xi_j(t+1)\}, \{\omega_{ji}\}, \{\alpha_{jik}(t)\}), \tag{10}$$

$$\alpha_{ji1}(t+1) = \alpha_{ji1}(t) + \eta(\mathbf{v}_j(t+1) - \omega_{ji}(t+1)) \qquad \forall\, j \in \mathcal{J},\ \forall i \in \mathcal{B}_j, \tag{11}$$

$$\alpha_{ji2}(t+1) = \alpha_{ji2}(t) + \eta(\omega_{ji}(t+1) - \mathbf{v}_i(t+1)) \qquad \forall\, j \in \mathcal{J},\ \forall i \in \mathcal{B}_j. \tag{12}$$

and correspond to the ADMoM solver reviewed in Appendix A.

Proof See Appendix C. ■

Lemma 2 links the proposed DSVM design with the convergent ADMoM solver, and thus ensures convergence of the novel MoM-DSVM to the centralized SVM classifier. However, for the particular problem at hand it is possible to simplify iterations (9)-(12). Indeed, simple inspection of (8) confirms that with all other variables fixed, the cost in (10) is linear-quadratic in ω_{ji}; hence, $\omega_{ji}(t+1)$ can be found in closed form per iteration, and the resulting closed-form expression can be substituted back to eliminate ω_{ji} from $\mathcal{L}$. Furthermore, Appendix D shows that the two sets of multipliers α_{ji1} and α_{ji2} can be combined into one set α_j after appropriate initialization of the iterations (11) and (12), as asserted by the following lemma.

Lemma 3 *Selecting* $\alpha_{ji1}(0) = \alpha_{ji2}(0) = 0_{(p+1)\times 1}$ *as initialization* $\forall\, j \in \mathcal{J},\ \forall i \in \mathcal{B}_j$, *iterations* (9)-(12) *reduce to*

$$\{\mathbf{v}_j(t+1), \xi_j(t+1)\} = \arg \min_{\{\mathbf{v}_j, \xi_j\} \in \mathcal{W}} \mathcal{L}'(\{\mathbf{v}_j\}, \{\xi_j\}, \{\mathbf{v}_j(t)\}, \{\alpha_j(t)\}), \tag{13}$$

$$\alpha_j(t+1) = \alpha_j(t) + \frac{\eta}{2} \sum_{i \in \mathcal{B}_j} [\mathbf{v}_j(t+1) - \mathbf{v}_i(t+1)] \quad \forall\, j \in \mathcal{J} \tag{14}$$

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

$$\mathcal{L}'(\{\mathbf{v}_j\},\{\xi_j\},\{\mathbf{v}_j(t)\},\{\alpha_j(t)\}) = \frac{1}{2}\sum_{j=1}^{J}\mathbf{v}_j^T(\mathbf{I}_{p+1}-\Pi_{p+1})\mathbf{v}_j + JC\sum_{j=1}^{J}\mathbf{1}_j^T\xi_j + 2\sum_{j=1}^{J}\alpha_j^T(t)\mathbf{v}_j + \eta\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\left\|\mathbf{v}_j - \frac{1}{2}[\mathbf{v}_j(t)+\mathbf{v}_i(t)]\right\|^2. \quad (15)$$

Proof See Appendix D. ■

The optimization problem in (13) involves the reduced Lagrangian $\mathcal{L}'$ in (15), which is linear-quadratic in $\mathbf{v}_j$ and ξ_j. In addition, the constraint set $\mathcal{W}$ is linear in these variables. To solve this constrained minimization problem through its dual, let $\lambda_j := [\lambda_{j1},\ldots,\lambda_{jN_j}]^T$ denote Lagrange multipliers per node corresponding to the constraints $\mathbf{Y}_j\mathbf{X}_j\mathbf{v}_j \succeq \mathbf{1}_j - \xi_j$. Solving the dual of (13) yields the optimal λ_j at iteration $t+1$, namely $\lambda_j(t+1)$, as a function of $\mathbf{v}_j(t)$ and $\alpha_j(t)$; while the KKT conditions provide expressions for $\mathbf{v}_j(t+1)$ as a function of $\alpha_j(t)$, and the optimal dual variables $\lambda_j(t+1)$. Notwithstanding, the resultant iterations are decoupled across nodes. These iterations and the associated convergence guarantees can be summarized as follows.

Proposition 1 *Consider the per node iterates* $\lambda_j(t)$, $\mathbf{v}_j(t)$ *and* $\alpha_j(t)$, *given by*

$$\lambda_j(t+1) = \arg\max_{\lambda_j:\,\mathbf{0}_j\preceq\lambda_j\preceq JC\mathbf{1}_j} -\frac{1}{2}\lambda_j^T\mathbf{Y}_j\mathbf{X}_j\mathbf{U}_j^{-1}\mathbf{X}_j^T\mathbf{Y}_j\lambda_j + \left(\mathbf{1}_j + \mathbf{Y}_j\mathbf{X}_j\mathbf{U}_j^{-1}\mathbf{f}_j(t)\right)^T\lambda_j, \quad (16)$$

$$\mathbf{v}_j(t+1) = \mathbf{U}_j^{-1}\left[\mathbf{X}_j^T\mathbf{Y}_j\lambda_j(t+1) - \mathbf{f}_j(t)\right], \quad (17)$$

$$\alpha_j(t+1) = \alpha_j(t) + \frac{\eta}{2}\sum_{i\in\mathcal{B}_j}[\mathbf{v}_j(t+1) - \mathbf{v}_i(t+1)] \quad (18)$$

where $\mathbf{U}_j := (1+2\eta|\mathcal{B}_j|)\mathbf{I}_{p+1} - \Pi_{p+1}$, $\mathbf{f}_j(t) := 2\alpha_j(t) - \eta\sum_{i\in\mathcal{B}_j}[\mathbf{v}_j(t)+\mathbf{v}_i(t)]$, $\eta > 0$, *and arbitrary initialization vectors* $\lambda_j(0)$, $\mathbf{v}_j(0)$, *and* $\alpha_j(0) = \mathbf{0}_{(p+1)\times 1}$. *The iterate* $\mathbf{v}_j(t)$ *converges to the solution of* (7), *call it* $\mathbf{v}^*$, *as* $t\to\infty$; *that is,* $\lim_{t\to\infty}\mathbf{v}_j(t) = \mathbf{v}^*$.

Proof See Appendix E. ■

Similar to the centralized SVM algorithm, if $[\lambda_j(t)]_n \neq 0$, then $[\mathbf{x}_{jn}^T, 1]^T$ is an SV. Finding $\lambda_j(t+1)$ as in (16) requires solving a quadratic optimization problem similar to the one that a centralized SVM would solve, for example, via a gradient projection algorithm or an interior point method; see for example, Schölkopf and Smola (2002, Ch. 6). However, the number of variables involved in (16) per iteration per node is considerably smaller when compared to its centralized counterpart, namely N_j versus $\sum_{j=1}^{J}N_j$. Also, the optimal local slack variables ξ_j^* can be found via the KKT conditions for (13).

The ADMoM-based DSVM (MoM-DSVM) iterations (16)-(18) are summarized as Algorithm 1, and are illustrated in Figure 2. All nodes have available 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. Every node then updates its local $(p+1)\times 1$ estimates $\mathbf{v}_j(t)$ and $\alpha_j(t)$; and the $N_j\times 1$ vector $\lambda_j(t)$.

At iteration $t+1$, node j computes vector $\mathbf{f}_j(t)$ locally to obtain its local $\lambda_j(t+1)$ via (16). Vector $\lambda_j(t+1)$ together with the local training set $\mathcal{S}_j$ are used at node j to compute $\mathbf{v}_j(t+1)$ via (17). 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 $t+1$ resumes when every node updates its local $\alpha_j(t+1)$ vector via (18). 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

$$g_j^{(t)}(\mathbf{x}) = [\mathbf{x}^T, 1]\mathbf{v}_j(t) \tag{19}$$

which from Proposition 1 is guaranteed to converge to the same solution across all nodes as $t \to \infty$. Simulated tests in Section 5 will demonstrate that after a few iterations the classification performance of (19) outperforms that of the local discriminant function obtained based on the local training set alone. The effect of η on the convergence rate of MoM-DSVM will be tested numerically in Section 5.

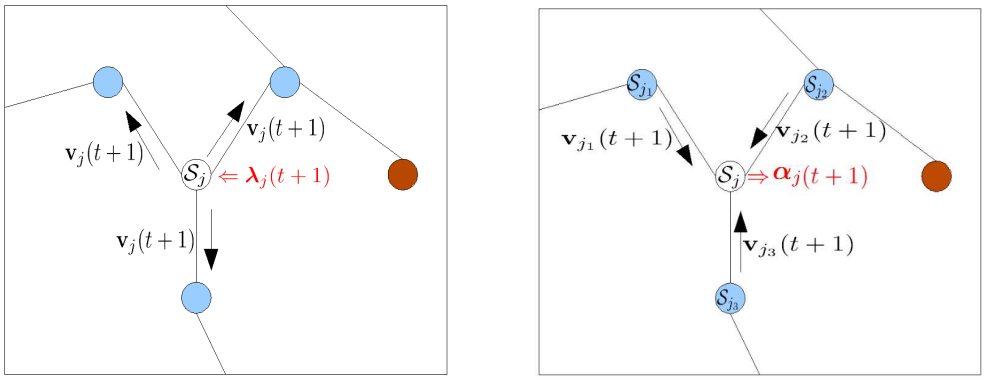

Figure 2: Visualization of iterations (16)-(18): (left) every node $j \in \mathcal{J}$ computes $\lambda_j(t+1)$ to obtain $\mathbf{v}_j(t+1)$, and then broadcasts $\mathbf{v}_j(t+1)$ to all neighbors $i \in \mathcal{B}_j$; (right) once every node $j \in \mathcal{J}$ has received $\mathbf{v}_i(t+1)$ from all $i \in \mathcal{B}_j$, it computes $\alpha_j(t+1)$.

Remark 1 *The messages exchanged among neighboring nodes in the MoM-DSVM algorithm correspond to local estimates $\mathbf{v}_j(t)$, which together with the local multiplier vectors $\alpha_j(t)$, convey sufficient information about the local training sets to achieve consensus globally. Per iteration and per node a message of* fixed size $(p+1)\times 1$ *is broadcasted (vectors α_j are not exchanged among nodes.) This is to be contrasted with incremental DSVM algorithms in, for example, Lu et al. (2008), Flouri et al. (2006) and Flouri et al. (2008), where the size of the messages exchanged between neighboring nodes depends on the number of SVs found at each incremental step. Although the SVs of each training set may be few, the overall number of SVs may remain large, thus consuming considerable power when transmitting SVs from one node to the next.*

Remark 2 *Real networks are prone to node failures, for example, sensors in a WSN may run out of battery during operation. Thanks to its fully decentralized mode of operation, the novel MoM-DSVM algorithm guarantees that the remaining nodes in the network will reach consensus as long as the*

Algorithm 1 MoM-DSVM

Require: Randomly initialize $\mathbf{v}_j(0)$, and $\alpha_j(0) = \mathbf{0}_{(p+1)\times 1}$ for every $j \in \mathcal{J}$

for $t = 0, 1, 2, \ldots$ **do**
  for all $j \in \mathcal{J}$ **do**
    Compute $\lambda_j(t+1)$ via (16).
    Compute $\mathbf{v}_j(t+1)$ via (17).
  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 $\alpha_j(t+1)$ via (18).
  end for
end for

node that fails, say $j_o \in \mathcal{J}$, does not correspond to a cut-vertex *of $\mathcal{G}$. In this case, the operational network graph $\mathcal{G}_o := \mathcal{G} - j_o$ remains connected, and thus surviving nodes can percolate information throughout $\mathcal{G}_o$. Of course, $\mathcal{S}_{j_o}$ will not participate in training the SVM. If j_o is a* cut-vertex *of $\mathcal{G}$, the algorithm will remain operational in each connected component of the resulting sub-graph $\mathcal{G}_o$, reaching consensus among nodes in each of the connected components.*

3.1 Online Distributed Support Vector Machine

In many distributed learning tasks data arrive sequentially, and possibly asynchronously. In addition, the processes to be learned may change with time. In such cases, training examples need to be added or removed from each local training set $\mathcal{S}_j$. Training sets of increasing of decreasing size can be expressed in terms of *time-varying* augmented data matrices $\mathbf{X}_j(t)$, and corresponding label matrices $\mathbf{Y}_j(t)$. An online version of DSVM is thus well motivated when a new training example $\mathbf{x}_{jn}(t)$ along with its label $y_{jn}(t)$ acquired at time t are incorporated into $\mathbf{X}_j(t)$ and $\mathbf{Y}_j(t)$, respectively. The corresponding modified iterations are given by (cf. (16)-(18))

$$\begin{aligned}\lambda_j(t+1) &= \underset{\lambda_j:\, \mathbf{0}_j(t+1)\preceq\lambda_j\preceq JC\mathbf{1}_j(t+1)}{\arg\max} -\frac{1}{2}\lambda_j^T\mathbf{Y}_j(t+1)\mathbf{X}_j(t+1)\mathbf{U}_j^{-1}\mathbf{X}_j(t+1)^T\mathbf{Y}_j(t+1)\lambda_j \\ &\qquad + \left(\mathbf{1}_j - \mathbf{Y}_j(t+1)\mathbf{X}_j(t+1)\mathbf{U}_j^{-1}\mathbf{f}_j(t)\right)^T\lambda_j, \qquad (20)\\ \mathbf{v}_j(t+1) &= \mathbf{U}_j^{-1}\left(\mathbf{X}_j(t+1)^T\mathbf{Y}_j(t+1)\lambda_j(t+1) - 2\alpha_j(t) + \eta\sum_{i\in\mathcal{B}_j}[\mathbf{v}_j(t)+\mathbf{v}_i(t)]\right), \qquad (21)\\ \alpha_j(t+1) &= \alpha_j(t) + \frac{\eta}{2}\sum_{i\in\mathcal{B}_j}[\mathbf{v}_j(t+1) - \mathbf{v}_i(t+1)]. \qquad (22)\end{aligned}$$

Note that the dimensionality of λ_j must vary to accommodate the variable number of $\mathcal{S}_j$ elements at every time instant t. The online MoM-DSVM classifier is summarized as Algorithm 2. For this algorithm to run, no conditions need to be imposed on how the sets $\mathcal{S}_j(t)$ increase or decrease. Their changes can be asynchronous and may comprise multiple training examples at once. In principle,

Algorithm 2 Online MoM-DSVM

Require: Randomly initialize $\mathbf{v}_j(0)$, and $\alpha_j(0) = \mathbf{0}_{(p+1)\times 1}$ for every $j \in \mathcal{J}$.

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 $\lambda_j(t+1)$ via (20).
    Compute $\mathbf{v}_j(t+1)$ via (21).
  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 $\alpha_j(t+1)$ via (22).
  end for
end for

the parameters η and C can also become time-dependent. The effect of these parameters will be discussed in Section 5.

Intuitively speaking, if the training sets remain invariant across a sufficient number of time instants, $\mathbf{v}_j(t)$ will closely track the optimal linear classifier. Rigorous convergence analysis of Algorithm 2 for any given rate of change of the training set goes beyond the scope of this work. Simulations will however demonstrate that the modified iterations in (20)-(22) are able to track changes in the training sets even when these occur at every time instant t.

Remark 3 *Compared to existing* centralized *online SVM alternatives in, for example, Cauwenberghs and Poggio (2000) and Fung and Mangasarian (2002), the online MoM-DSVM algorithm of this section allows 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 can return a partially trained model constructed with examples available to the network at any given time. Likewise, elements of the training sets can be removed without having to restart the MoM-DSVM algorithm. This feature also allows adapting MoM-DSVM to jointly operate with algorithms that account for concept drift (Klinkenberg and Joachims, 2000). In the classification context, concept drift defines a change in the true classification boundaries between classes. In general, accounting for concept drift requires two main steps, which can be easily handled by the online MoM-DSVM: (i) acquisition of updated elements in the training set that better describe the current concept; and (ii) removal of outdated elements from the training set.*

4. Distributed Nonlinear Support Vector Machine

In Section 3, problem (1) was reformulated to allow all nodes to consent on $\mathbf{v}^*$. However, applying an identical reformulation to the nonlinear classification problem in (2) would require updates in

(16)-(18) to be carried in $\mathcal{H}$. As the dimensionality P of $\mathcal{H}$ increases, the local computation and communication complexities become increasingly prohibitive.

Our approach to mitigate this well-known "curse of dimensionality" is to enforce consensus of the local discriminants g_j^* on a subspace of reduced rank $L < P$. To this end, we project the consensus constraints corresponding to (2) and consider the optimization problem (cf. (6))

$$\begin{aligned} \min_{\{\mathbf{w}_j, b_j, \xi_j\}} \quad & \frac{1}{2}\sum_{j=1}^{J}\|\mathbf{w}_j\|^2 + JC\sum_{j=1}^{J}\mathbf{1}_j^T\xi_j \\ \text{s.t.} \quad & \mathbf{Y}_j(\Phi(\mathbf{X}_j)\mathbf{w}_j + \mathbf{1}_j b_j) \succeq \mathbf{1}_j - \xi_j \quad \forall j \in \mathcal{J} \\ & \xi_j \succeq \mathbf{0}_j \quad \forall j \in \mathcal{J} \\ & \mathbf{G}\mathbf{w}_j = \mathbf{G}\mathbf{w}_i \quad \forall j \in \mathcal{J}, i \in \mathcal{B}_j \\ & b_j = b_i \quad \forall j \in \mathcal{J}, i \in \mathcal{B}_j \end{aligned} \tag{23}$$

where $\Phi(\mathbf{X}_j) := [\phi(\mathbf{x}_{j1}), \ldots, \phi(\mathbf{x}_{jN_j})]^T$, and $\mathbf{G} := [\phi(\chi_1), \ldots, \phi(\chi_L)]^T$ is a fat $L \times P$ matrix common to all nodes with preselected vectors $\{\chi_l\}_{l=1}^L$ specifying its rows. Each $\chi_l \in \mathcal{X}$ corresponds to a $\phi(\chi_l) \in \mathcal{H}$, which at the optimal solution $\{\mathbf{w}_j^*, b_j^*\}_{j=1}^J$ of (23), satisfies $\phi^T(\chi_l)\mathbf{w}_1^* = \cdots = \phi^T(\chi_l)\mathbf{w}_J^* = \phi^T(\chi_l)\mathbf{w}^*$. The projected constraints $\{\mathbf{G}\mathbf{w}_j = \mathbf{G}\mathbf{w}_i\}$ along with $\{b_j = b_i\}$ force all nodes to agree on the value of the local discriminant functions $g_j^*(\chi_l)$ at the vectors $\{\chi_l\}_{l=1}^L$, but not necessarily for all $\mathbf{x} \in \mathcal{X}$. This is the price paid for reducing the computational complexity of (23) to an affordable level. Clearly, the choice of vectors $\{\chi_l\}_{l=1}^L$, their number L, and the local training sets $\mathcal{S}_j$ determine how similar the local discriminant functions g_j^* are. If $\mathbf{G} = \mathbf{I}_P$, then (23) reduces to (6), and $g_1^*(\mathbf{x}) = \ldots = g_J^*(\mathbf{x}) = g^*(\mathbf{x})$, $\forall \mathbf{x} \in \mathcal{X}$, but the high dimensionality challenge appears. At the end of this section, we will provide different design choices for $\{\chi_l\}_{l=1}^L$, and test them via simulations in Section 5.

Because the cost in (23) is strictly convex w.r.t. $\mathbf{w}_j$, it guarantees that the set of optimal vectors $\{\mathbf{w}_j^*\}$ is unique even when $\mathbf{G}$ is a 'fat' matrix ($L < P$) and/or ill-conditioned (Bertsekas, 1999, Prop. 5.2.1). As in (2), having $\{\mathbf{w}_j^*\}$ known is of limited use, since the mapping ϕ may be unknown, or if known, evaluating vectors $\phi(\mathbf{x})$ may entail an excessive computational cost. Fortunately, the resulting discriminant function $g_j^*(\mathbf{x})$ admits a reduced-complexity solution because it can be expressed in terms of kernels, as shown by the following theorem.

Theorem 4 *For every positive semi-definite kernel $K(\cdot,\cdot)$, the discriminant functions $g_j^*(\mathbf{x}) = \phi^T(\mathbf{x})\mathbf{w}_j^* + b_j^*$ with $\{\mathbf{w}_j^*, b_j^*\}$ denoting the optimal solution of* (23)*, can be written as*

$$g_j^*(\mathbf{x}) = \sum_{n=1}^{N_j} a_{jn}^* K(\mathbf{x}, \mathbf{x}_{jn}) + \sum_{l=1}^{L} c_{jl}^* K(\mathbf{x}, \chi_l) + b_j^*, \ \forall j \in \mathcal{J} \tag{24}$$

where $\{a_{jn}^*\}$ and $\{c_{jl}^*\}$ are real-valued scalar coefficients.

Proof See Appendix F. ■

The space of functions g_j described by (24) is fully determined by the span of the kernel function $K(\cdot,\cdot)$ centered at training vectors $\{\mathbf{x}_{jn}, n = 1, \ldots, N_j\}$ per node, and also at the vectors $\{\chi_l\}_{l=1}^L$ which are common to all nodes. Thus, similarity of the discriminant functions g_j^* across nodes is

naturally constrained by the corresponding $\mathcal{S}_j$. Theorem 1 also reveals the effect of $\{\chi_l\}_{l=1}^L$ on the $\{g_j^*\}$. By introducing vectors $\{\chi_l\}_{l=1}^L$ common to all nodes, a subset of basis functions common to all local functional spaces is introduced a fortiori. Coefficients a_{jn}^* and c_{jl}^* are found so that all local discriminants g_j^* agree on their values at points $\{\chi_l\}_{l=1}^L$. Intuitively, at every node these coefficients summarize the global information available to the network.

Theorem 1 is an existence result whereby each nonlinear discriminant function g_j^* is expressible in terms of $\mathcal{S}_j$ and $\{\chi_l\}_{l=1}^L$. However, finding the coefficients a_{jn}^*, c_{jl}^* and b_j^* in a distributed fashion remains an issue. Next, it is shown that these coefficients can be obtained iteratively by applying the ADMoM solver to (23).

Similar to (7), introduce auxiliary variables $\{\omega_{ji}\}$ ($\{\zeta_{ji}\}$) to decouple the constraints $\mathbf{G}\mathbf{w}_j = \mathbf{G}\mathbf{w}_i$ ($b_j = b_i$) across nodes, and α_{jik} (β_{jik}) denote the corresponding Lagrange multipliers (cf. (8)). The surrogate augmented Lagrangian for problem (23) is then

$$\begin{aligned}\mathcal{L}(\{\mathbf{w}_j\},\{\xi_j\},\{\omega_{ji}\},\{\alpha_{jik}\},\{\zeta_{ji}\},\{\beta_{jik}\}) &= \frac{1}{2}\sum_{j=1}^J \|\mathbf{w}_j\|^2 + JC\sum_{j=1}^J \mathbf{1}_j^T\xi_j + \sum_{j=1}^J\sum_{i\in\mathcal{B}_j}\alpha_{ji1}^T(\mathbf{G}\mathbf{w}_j-\omega_{ji}) \\ &+ \sum_{j=1}^J\sum_{i\in\mathcal{B}_j}\alpha_{ji2}^T(\omega_{ji}-\mathbf{G}\mathbf{w}_i) + \sum_{j=1}^J\sum_{i\in\mathcal{B}_j}\beta_{ji1}(b_j-\zeta_{ji}) + \sum_{j=1}^J\sum_{i\in\mathcal{B}_j}\beta_{ji2}(\zeta_{ji}-b_i) \\ &+ \frac{\eta}{2}\sum_{j=1}^J\sum_{i\in\mathcal{B}_j}\|\mathbf{G}\mathbf{w}_j-\omega_{ji}\|^2 + \frac{\eta}{2}\sum_{j=1}^J\sum_{i\in\mathcal{B}_j}\|\omega_{ji}-\mathbf{G}\mathbf{w}_i\|^2 + \frac{\eta}{2}\sum_{j=1}^J\sum_{i\in\mathcal{B}_j}\|b_j-\zeta_{ji}\|^2 + \frac{\eta}{2}\sum_{j=1}^J\sum_{i\in\mathcal{B}_j}\|\zeta_{ji}-b_i\|^2.\end{aligned}$$

Following the steps of Lemma 2, and with $\{\alpha_{jik}\}$ and $\{\beta_{jik}\}$ initialized at zero, the ADMoM iterations take the form

$$\{\mathbf{w}_j(t+1), b_j(t+1), \xi_j(t+1)\} = \arg\min_{\{\mathbf{w}_j,b_j,\xi_j\}\in\mathcal{W}} \mathcal{L}'(\{\mathbf{w}_j\},\{b_j\},\{\xi_j\},\{\alpha_j(t)\},\{\beta_j(t)\}), \tag{25}$$

$$\alpha_j(t+1) = \alpha_j(t) + \frac{\eta}{2}\sum_{i\in\mathcal{B}_j}\mathbf{G}[\mathbf{w}_j(t+1) - \mathbf{w}_i(t+1)], \tag{26}$$

$$\beta_j(t+1) = \beta_j(t) + \frac{\eta}{2}\sum_{i\in\mathcal{B}_j}[b_j(t+1) - b_i(t+1)] \tag{27}$$

where $\mathcal{L}'$ is defined similar to (8), $\alpha_j(t)$ as in Lemma 3, and $\beta_j(t) := \sum_{i\in\mathcal{B}_j}\beta_{ji1}(t)$. The ADMoM iterations (25)-(27) will not be explicitly solved since iterates $\mathbf{w}_j(t)$ lie in the high-dimensional space $\mathcal{H}$. Nevertheless, our objective is not to find $\mathbf{w}_j^*$, but rather the discriminant function $g_j^*(\mathbf{x})$. To this end, let $\Gamma := [\chi_1,\ldots,\chi_L]^T$, and define the kernel matrices with entries

$$[\mathbf{K}(\mathbf{X}_j,\mathbf{X}_j)]_{n,m} := K(\mathbf{x}_{jn},\mathbf{x}_{jm}), \tag{28}$$

$$[\mathbf{K}(\mathbf{X}_j,\Gamma)]_{n,l} := K(\mathbf{x}_{jn},\chi_l), \tag{29}$$

$$[\mathbf{K}(\Gamma,\Gamma)]_{l,l'} := K(\chi_l,\chi_{l'}). \tag{30}$$

From Theorem 1 it follows that each local update $g_j^{(t)}(\mathbf{x}) = \phi^T(\mathbf{x})\mathbf{w}_j(t) + b_j(t)$ admits per iteration t a solution expressed in terms of kernels. The latter is specified by the coefficients $\{a_{jn}(t)\}$, $\{c_{jl}(t)\}$ and $\{b_j(t)\}$ that can be obtained in closed form, as shown in the next proposition.

Proposition 2 *Let $\lambda_j := [\lambda_{j1},\ldots,\lambda_{jN_j}]^T$ denote the Lagrange multiplier corresponding to the constraint $\mathbf{Y}_j(\Phi(\mathbf{X}_j)\mathbf{w}_j+\mathbf{1}_j b_j)\succeq \mathbf{1}_j-\xi_j$, and $\widetilde{\mathbf{w}}_j(t) := \mathbf{G}\mathbf{w}_j(t)$. The local discriminant function $g_j^{(t)}(\mathbf{x})$ at iteration t is*

$$g_j^{(t)}(\mathbf{x}) = \sum_{n=1}^{N_j} a_{jn}(t)K(\mathbf{x},\mathbf{x}_{jn}) + \sum_{l=1}^{L} c_{jl}(t)K(\mathbf{x},\chi_l) + b_j(t) \tag{31}$$

where $\mathbf{a}_j(t) := [a_{j1}(t),\ldots,a_{jN_j}(t)]^T$, $\mathbf{c}_j(t) := [c_{j1}(t),\ldots,c_{jL}(t)]^T$, and $b_j(t)$ are given by

$$\mathbf{a}_j(t) := \mathbf{Y}_j\lambda_j(t), \tag{32}$$

$$\mathbf{c}_j(t) := 2\eta|\mathcal{B}_j|\widetilde{\mathbf{U}}_j^{-1}\left[\mathbf{K}(\Gamma,\Gamma)\mathbf{f}_j(t)-\mathbf{K}(\Gamma,\mathbf{X}_j)\mathbf{Y}_j\lambda_j(t)\right]-\widetilde{\mathbf{f}}_j(t), \tag{33}$$

$$b_j(t) := \frac{1}{2\eta|\mathcal{B}_j|}\left[\mathbf{1}_j^T\mathbf{Y}_j\lambda_j(t)-h_j(t)\right] \tag{34}$$

with $\lambda_j(t)$ denoting the vector multiplier update available at iteration t, $\widetilde{\mathbf{U}}_j := \mathbf{I}_L+2\eta|\mathcal{B}_j|\mathbf{K}(\Gamma,\Gamma)$, $\widetilde{\mathbf{f}}_j(t) := 2\alpha_j(t)-\eta\sum_{i\in\mathcal{B}_j}[\widetilde{\mathbf{w}}_j(t)+\widetilde{\mathbf{w}}_i(t)]$ and $h_j(t) := 2\beta_j(t)-\eta\sum_{i\in\mathcal{B}_j}[b_j(t)+b_i(t)]$.

Proof See Appendix G. ■

Proposition 2 asserts that in order to find $\mathbf{a}_j(t)$, $\mathbf{c}_j(t)$ and $b_j(t)$ in (32), (33) and (34), it suffices to obtain $\lambda_j(t)$, $\widetilde{\mathbf{w}}_j(t)$, $b_j(t)$, $\alpha_j(t)$, and $\beta_j(t)$. Note that finding the $L\times 1$ vector $\widetilde{\mathbf{w}}_j(t)$ from $\mathbf{w}_j(t)$ incurs complexity of order $O(L)$. The next proposition shows how to iteratively update $\lambda_j(t)$, $\widetilde{\mathbf{w}}_j(t)$, $b_j(t)$, $\alpha_j(t)$, and $\beta_j(t)$ in a distributed fashion.

Proposition 3 *The iterates $\lambda_j(t)$, $\widetilde{\mathbf{w}}_j(t)$, $b_j(t)$, $\alpha_j(t)$ and $\beta_j(t)$ can be obtained as*

$$\begin{aligned}\lambda_j(t+1) = \arg\max_{\lambda_j:\,\mathbf{0}_j\preceq\lambda_j\preceq JC\mathbf{1}_j} & -\frac{1}{2}\lambda_j^T\mathbf{Y}_j\left(\mathbf{K}(\mathbf{X}_j,\mathbf{X}_j)-\widetilde{\mathbf{K}}(\mathbf{X}_j,\mathbf{X}_j)+\frac{\mathbf{1}_j\mathbf{1}_j^T}{2\eta|\mathcal{B}_j|}\right)\mathbf{Y}_j\lambda_j+\mathbf{1}_j^T\lambda_j \\ & -\left(\widetilde{\mathbf{f}}_j^T(t)\left(\mathbf{K}(\Gamma,\mathbf{X}_j)-\widetilde{\mathbf{K}}(\Gamma,\mathbf{X}_j)\right)+h_j(t)\frac{\mathbf{1}^T}{2\eta|\mathcal{B}_j|}\right)\mathbf{Y}_j\lambda_j,\end{aligned} \tag{35}$$

$$\widetilde{\mathbf{w}}_j(t+1) = \left[\mathbf{K}(\Gamma,\mathbf{X}_j)-\widetilde{\mathbf{K}}(\Gamma,\mathbf{X}_j)\right]\mathbf{Y}_j\lambda_j(t+1)-\left[\mathbf{K}(\Gamma,\Gamma)-\widetilde{\mathbf{K}}(\Gamma,\Gamma)\right]\widetilde{\mathbf{f}}_j(t), \tag{36}$$

$$b_j(t+1) = \frac{1}{2\eta|\mathcal{B}_j|}\left[\mathbf{1}_j^T\mathbf{Y}_j\lambda_j(t+1)-h_j(t)\right], \tag{37}$$

$$\alpha_j(t+1) = \alpha_j(t)+\frac{\eta}{2}\sum_{i\in\mathcal{B}_j}[\widetilde{\mathbf{w}}_j(t+1)-\widetilde{\mathbf{w}}_i(t+1)], \tag{38}$$

$$\beta_j(t+1) = \beta_j(t)+\frac{\eta}{2}\sum_{i\in\mathcal{B}_j}[b_j(t+1)-b_i(t+1)] \tag{39}$$

where $\widetilde{\mathbf{K}}(\mathbf{Z},\mathbf{Z}') := 2\eta|\mathcal{B}_j|\mathbf{K}(\mathbf{Z},\Gamma)\widetilde{\mathbf{U}}_j^{-1}\mathbf{K}(\Gamma,\mathbf{Z}')$. With arbitrary initialization $\lambda_j(0)$, $\widetilde{\mathbf{w}}_j(0)$, and $b_j(0)$; and $\alpha_j(0)=\mathbf{0}_{L\times 1}$ and $\beta_j(0)=0$, the iterates $\{a_{jn}(t)\}$, $\{c_{jl}(t)\}$ and $\{b_j(t)\}$ in (32), (33) and (34) converge to $\{a_{jn}^*\}$, $\{c_{jl}^*\}$ and $\{b_j^*\}$ in (24), as $t\to\infty$, $\forall\, j\in\mathcal{J}, n=1,\ldots,N_j$, and $l=1,\ldots,L$.

Proof See Appendix H. ■

Algorithm 3 MoM-NDSVM

Require: Randomly initialize $\widetilde{\mathbf{w}}_j(0)$ and $b_j(0)$; and $\alpha_j(0) = \mathbf{0}_{L\times 1}$ and $\beta_j(0) = 0$ for every $j \in \mathcal{J}$.

for $t = 0, 1, 2, \ldots$ **do**
  for all $j \in \mathcal{J}$ **do**
    Compute $\lambda_j(t+1)$ via (35).
    Compute $\widetilde{\mathbf{w}}_j(t+1)$ via (36).
    Compute $b_j(t+1)$ via (37).
  end for
  for all $j \in \mathcal{J}$ **do**
    Broadcast $\widetilde{\mathbf{w}}_j(t+1)$ and $b_j(t+1)$ to all neighbors $i \in \mathcal{B}_j$.
  end for
  for all $j \in \mathcal{J}$ **do**
    Compute $\alpha_j(t+1)$ via (38).
    Compute $\beta_j(t+1)$ via (39).
    Compute $\mathbf{a}_j(t)$, $\mathbf{c}_j(t)$ and $b_j(t)$ via (32), (33) and (34), respectively.
  end for
end for

The iterations comprising the ADMoM-based non-linear DSVM (MoM-NDSVM) are summarized as Algorithm 3. It is important to stress that Algorithm 3 starts by having all nodes agree on the common quantities Γ, JC, η, and $K(\cdot,\cdot)$. Also, each node computes its local kernel matrices as in (28)-(30), which remain unchanged throughout. Subsequently, Algorithm 3 runs in a manner analogous to Algorithm 1, with the difference that every node communicates an $(L+1)\times 1$ vector (instead of $(p+1)\times 1$) for its neighbors to receive $\widetilde{\mathbf{w}}_j(t)$ and $b_j(t)$.

4.1 On the Optimality of NDSVM and the Selection of Common Vectors

By construction, Algorithm 3 produces local discriminant functions whose predictions for $\{\chi_l\}_{l=1}^L$ are the same for all nodes in the network; that is, $g_1^*(\chi_l) = \ldots = g_J^*(\chi_l) = g^*(\chi_l)$ for $l = 1, \ldots, L$, where $g^*(\chi_l) = \phi^T(\chi_l)\mathbf{w}^* + b^*$, and $\{\mathbf{w}^*, b^*\}$ are the optimal solution of the centralized problem (2). Viewing $\{\chi_l\}_{l=1}^L$ as a classification query, the proposed MoM-NDSVM algorithm can be implemented as follows. Having this query presented at any node j entailing a set of unlabeled vectors $\{\chi_l\}_{l=1}^L$, the novel scheme first percolates $\{\chi_l\}_{l=1}^L$ throughout the network.[3] Problem (23) is subsequently solved in a distributed fashion using Algorithm 3. Notice that in this procedure no database information is shared.

Although optimal in the sense of being convergent to its centralized counterpart, the algorithm just described needs to be run for every new classification query. Alternatively, one can envision procedures to find discriminant functions in a distributed fashion that classify new queries without having to re-run the distributed algorithm. The key is to pre-select a *fixed* set $\{\chi_l\}_{l=1}^L$ for which g^* in (5) is (approximately) equivalent to g_j^* in (24) for all $j \in \mathcal{J}$. From Theorem 1, we know that all local functions g_j^* share a common space spanned by the χ_l-induced kernels $\{K(\cdot,\chi_l)\}$. If the space $\mathcal{H}$ where g^* lies is finite dimensional, for example, when adopting linear or polynomial

3. Percolating $\{\chi\}_{l=1}^L$ in a distributed fashion through the network can be carried in a finite number of iterations at most equal to the diameter of the network.

kernels in (5), one can always find a finite-size set $\{\chi_l\}_{l=1}^L$ such that the space spanned by the set of kernels $\{K(\cdot,\chi_l)\}$ contains $\mathcal{H}$, and thus $g_1^*(\mathbf{x}) = \ldots = g_J^*(\mathbf{x}) = g^*(\mathbf{x})\ \forall \mathbf{x} \in \mathcal{X}$ (Predd et al., 2006). Indeed, when using linear kernels, the MoM-NDSVM developed here boils down to the MoM-DSVM developed in Section 3 for a suitable finite-size set $\{\chi_l\}_{l=1}^L$.

In general, however, the space spanned by $\{K(\cdot,\chi_l)\}$ may have lower dimensionality than $\mathcal{H}$; thus, local functions g_j^* do not coincide at every point. In this case, MoM-NDSVM finds local approximations to the centralized g^* which accommodate information available to all nodes. The degree of approximation depends on the choice of $\{\chi_l\}_{l=1}^L$. In what follows, we describe two alternatives to constructing such a set $\{\chi_l\}_{l=1}^L$.

- **Grid-based designs**. Consider every entry k of the training vectors $\{\mathbf{x}_{jn}\}$, and form the intervals $I_k := [x_k^{\min}, x_k^{\max}]$, $k = 1,\ldots,p$, where $x_k^{\min} := \min_{j\in\mathcal{J},\, n=1,\ldots,N_j}[\mathbf{x}_{jn}]_k$ and $x_k^{\max} := \max_{j\in\mathcal{J},\, n=1,\ldots,N_j}[\mathbf{x}_{jn}]_k$. Take for convenience $L = M^p$, and partition uniformly each I_k to obtain a set of M equidistant points $Q_k := \{q_{k1},\ldots,q_{kM}\}$. The set $\{\chi_l\}_{l=1}^L$ can be formed by taking all M^p possible vectors with entries drawn from the Cartesian product $Q_1 \times \ldots, \times Q_p$. One possible set we use for generating the $\{\chi_l\}_{l=1}^L$ vectors is obtained by selecting the k-th entry of the l-th vector as $[\chi_l]_k = q_{k,\left(\frac{l}{M^{k-1}} \bmod M\right)+1}$, where $l = 1,\ldots,M^p$ and $k = 1,\ldots,p$. In this case, MoM-NDSVM performs a global consensus step on the entry-wise maxima and minima of the training vectors $\{\mathbf{x}_{jn}\}$. Global consensus on the entry-wise maxima and minima can be computed exactly in a finite number of iterations equal to at most the diameter of the graph $\mathcal{G}$.

- **Random designs**. Once again, we consider every entry k of the training vectors $\{\mathbf{x}_{jn}\}$. MoM-NDSVM starts by performing a consensus step on the entry-wise maxima and minima of the local training vectors $\{\mathbf{x}_{jn}\}$. The set $\{\chi_l\}_{l=1}^L$ is formed by drawing elements χ_l randomly from a uniform p-dimensional distribution with extreme points per entry given by the extreme points $x_k^{\min}$ and $x_k^{\max}$, $k = 1,\ldots,p$. To agree on the set $\{\chi_l\}_{l=1}^L$, all nodes in the network are assumed to share a common seed used to initialize the random sampling algorithms.

As mentioned earlier, the number of points L affects how close local functions are to each other as well as to the centralized one. The choice of L also depends on the kernel used, prior knowledge of the discriminant function, and the available local training data N_j. Increasing L guarantees that local functions will be asymptotically close to each other regardless of N_j; however, the communication cost and computational complexity per node will increase according to L [cf. Algorithm 3]. On the other hand, a small L reduces the communication overhead at the price of increasing the disagreement among the g_j^*'s. This trade-off will be further explored in the ensuing section through simulated tests.

5. Numerical Simulations

In this section, we analyze the performance of both MoM-DSVM and MoM-NDSVM algorithms using different networks with synthetic and real-world training sets. Although we focus on the binary classification case, it is worth remembering that K-ary classification problems with $K > 2$ can be solved via binary classification schemes, for example, by using one versus all classifiers, or all versus all classifiers (Duda et al., 2002, Ch. 5).

5.1 Linear Classifiers

In this section, we present experiments on synthetic and real data to illustrate the performance of our distributed method for training linear SVMs.

5.1.1 TEST CASE 1: SYNTHETIC TRAINING SET

Consider a randomly generated network with $J = 30$ nodes. The network is connected with algebraic connectivity 0.0448 and average degree per node 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 two-dimensional Gaussian distribution with common covariance matrix $\Sigma = [1,\ 0;\ 0,\ 2]$, and mean vectors $m_1 = [-1,\ -1]^T$ and $m_2 = [1,\ 1]^T$, respectively. Each local training set $\mathcal{S}_j$ consists of $N_j = N = 10$ labeled examples and was generated by: (i) randomly choosing class $\mathcal{C}_k$, $k = 1,2$; and, (ii) randomly generating a labeled example $(\mathbf{x}_{jn}^T, y_{jn} = \mathcal{C}_k)$ with $\mathbf{x}_{jn} \sim \mathcal{N}(m_k, \Sigma)$. Thus, the global training set contains $JN = 300$ training 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, drawn as in (i) and (ii), is used to evaluate the generalization performance of the classifiers. The Bayes optimal classifier for this 2-class problem is linear (Duda et al., 2002, Ch. 2), with risk $\mathbf{R}_{\text{Bayes}} = 0.1103$. The empirical risk of the centralized SVM in (1) is defined as

$$\mathbf{R}_{\text{emp}}^{\text{central}} := \frac{1}{N_T} \sum_{n=1}^{N_T} \frac{1}{2} |\widetilde{y}_n - \hat{y}_n|$$

where $\hat{y}_n$ is the predicted label for $\widetilde{\mathbf{x}}_n$. The average empirical risk of the MoM-DSVM algorithm as a function of the number of iterations is defined as

$$\mathbf{R}_{\text{emp}}(t) := \frac{1}{JN_T} \sum_{j=1}^{J} \sum_{n=1}^{N_T} \frac{1}{2} |\widetilde{y}_n - \hat{y}_{jn}(t)| \tag{40}$$

where $\hat{y}_{jn}(t)$ is the label prediction at iteration t and node j for $\widetilde{\mathbf{x}}_n$, $n = 1,\ldots,N_T$ using the SVM parameters in $\mathbf{v}_j(t)$. The average empirical risk of the local SVMs across nodes $\mathbf{R}_{\text{emp}}^{\text{local}}$ is defined as in (40) with $\hat{y}_{jn}$ found using only locally-trained SVMs.

Figure 3 (left) 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 are performed with randomly drawn local training and test sets per run. The centralized and local empirical risks with $C = 10$ are included for comparison. The average local prediction performance is also evaluated. Clearly, 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. To further visualize this test case, Figure 3 (right) shows the global training set, along with the linear discriminant functions found by the centralized SVM and the MoM-DSVM at two different nodes after 400 iterations with $JC = 20$ and $\eta = 10$. Local SVM results for two different nodes are also included for comparison.

5.1.2 TEST CASE 2: MNIST TRAINING SET

Here, the MoM-DSVM is tested on the MNIST database of handwritten images (Lecun et al., 1998). The MNIST database 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

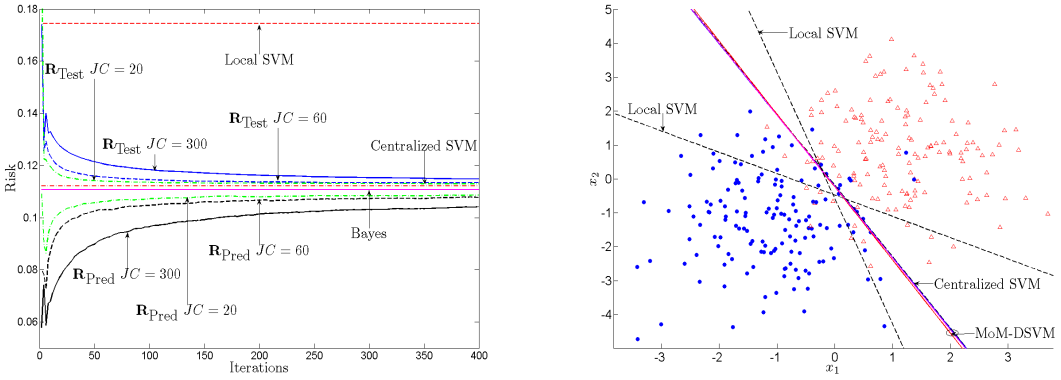

Figure 3: Evolution of the test error ($\mathbf{R}_{\text{Test}}$) and prediction error ($\mathbf{R}_{\text{Pred}}$) of MoM-DSVM for a two-class problem using synthetic data and a network with $J = 30$ nodes. Centralized SVM performance and average local SVMs performance are also plot for comparison (left). Decision boundary comparison among MoM-DSVM, centralized SVM and local SVM results for synthetic data generated from two Gaussian classes (right).

experiment each image is vectorized to a 784 by 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, that is, there is no preprocessing of the data. For simulations, we consider two randomly generated networks with $J = 25$, algebraic connectivity 3.2425, and average degree per node 12.80; and $J = 50$ nodes, algebraic connectivity 1.3961, and average degree per node 15.92. The training set is equally partitioned across nodes, thus every node in the network with $J = 25$ has $N_j = 472$ training vectors, and every node in the network with $J = 50$ has $N_j = 236$ samples. The distribution of samples across nodes influences the training phase of MoM-DSVM. For example, if data per node are biased toward one particular class, then the training phase may require more iterations to percolate appropriate information across the network. In the simulations, we consider the two extreme cases: (i) training data are evenly distributed across nodes, that is, every node has the same number of examples from digit 2 and from digit 9; and, (ii) highly biased local data, that is, every node has data corresponding to a single digit; thus, a local binary classifier cannot be constructed.

The figures in this section correspond to one run of the MoM-DSVM for a network with noiseless communication links. Figure 4 shows the evolution of the test error for the network with 25 nodes and highly biased local data. Likewise, Figure 5 shows the evolution of the test error for the network with 50 nodes and highly biased local data. Different values for the penalties JC and η were used to illustrate their effect on both the classification performance and the convergence rate of MoM-DSVM. The parameter JC controls the final performance of the classifier; but for a finite number of iterations, η also influences the final performance of the classifier. Larger values of η may be desirable; however, if η is too large, the algorithm first focuses on reaching consensus across nodes disregarding the classification performance. Although, MoM-DSVM is guaranteed to converge for all η, a very large choice for η may hinder the convergence rate.

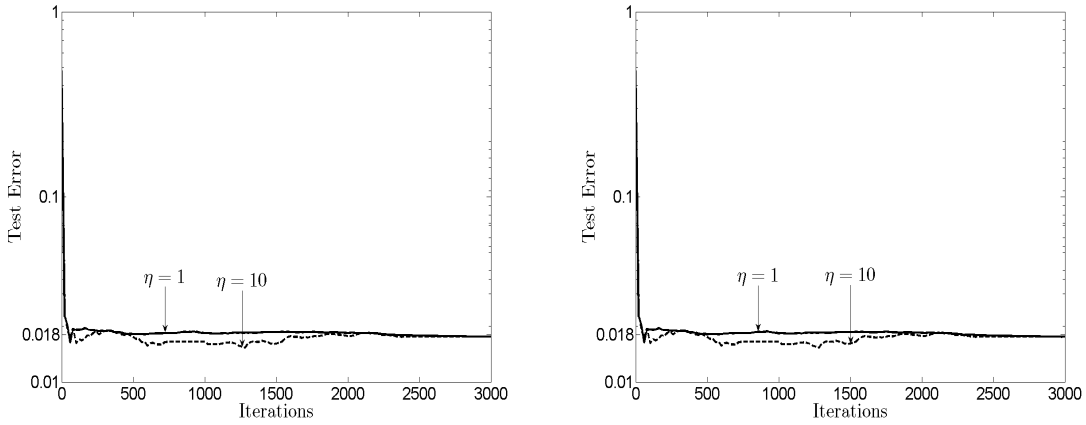

Figure 4: Evolution of the test error $(\mathbf{R}_{\text{Test}})$ of MoM-DSVM, with penalty coefficients $JC = 1$ (left) and $JC = 5$ (right), for a two-class problem using digits 2 and 9 from the MNIST data set unevenly distributed across nodes, and a network with $J = 25$ nodes.

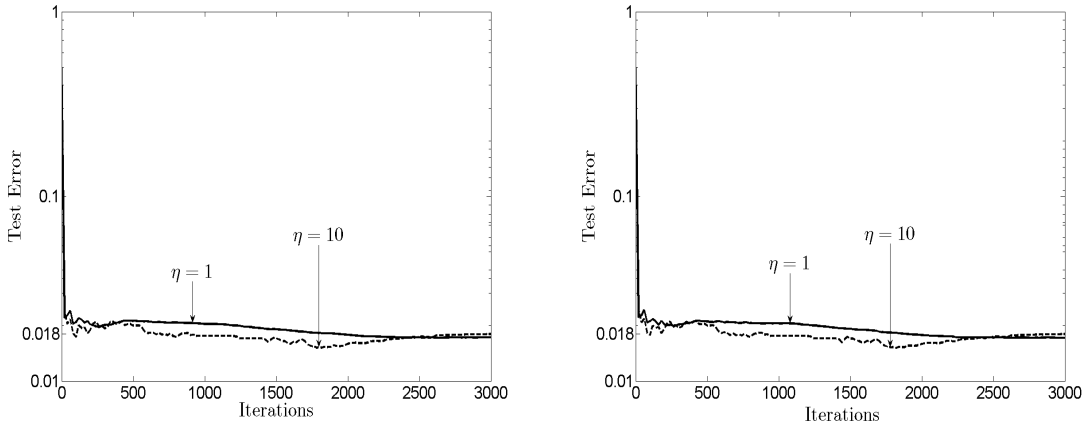

Figure 5: Evolution of the test error $(\mathbf{R}_{\text{Test}})$ of MoM-DSVM, with penalty coefficients $JC = 1$ (left) and $JC = 5$ (right), for a two-class problem using digits 2 and 9 from the MNIST data set unevenly distributed across nodes, and a network with $J = 50$ nodes.

Next, the dispersion of the solutions after 3,000 iterations for different values of η is tested. For our experiment, dispersion refers to how similar are the local $\mathbf{v}_j(t)$ at every node. The mean-squared error (MSE) of the solution across nodes is defined as $\Delta(t) := \frac{1}{J}\sum_{j=1}^{J} ||\mathbf{v}_j(t) - \overline{\mathbf{v}}(t)||^2$ where $\overline{\mathbf{v}}(t) := \frac{1}{J}\sum_{j=1}^{J} \mathbf{v}_j(t)$. Table 1 shows $\Delta(t)$ at $t = 3,000$ for different values of η and JC. Note that larger values of η lead to smaller dispersion in the solution; however, as illustrated in Figure 5, they do not imply faster convergence rates.

	$\Delta(t)$ at $t = 3,000$			
	$J = 25$		$J = 50$	
η	$JC = 1$	$JC = 5$	$JC = 1$	$JC = 5$
1	3.1849×10^{-7}	3.1870×10^{-7}	2.3749×10^{-7}	2.3227×10^{-7}
5	1.5591×10^{-8}	1.4760×10^{-8}	2.6613×10^{-8}	2.6646×10^{-8}
10	2.9280×10^{-10}	2.9112×10^{-10}	3.6028×10^{-9}	3.6207×10^{-9}

Table 1: MSE $\Delta(t)$ with MNIST data set and biased local data for different values of η and JC.

Consider next data that are evenly distributed across nodes. The MNIST training set is partitioned across nodes ensuring that every node has an equal number of examples from digit 2 and digit 9. Figure 6 shows the evolution of the test error for the network with $J = 25$ nodes and Figure 7 shows the evolution of the test error for the network with $J = 50$ nodes for different values for the penalties JC and η. In this case, local classifiers achieve low test error after one iteration of the MoM-DSVM. In subsequent iterations MoM-DSVM forces all local classifiers to consent, but the test error does not decrease monotonically across iterations. This variation is small, ranging between 0.015 and 0.02, since all local classifiers already have low test error. Both Figures 6 and 7 show that between iterations 500 and $2,000$, the global test reaches a minimum value, then it increases and converges to a larger value. This non-monotonic behavior can be attributed to the fact that the MoM-DSVM iterates are not guaranteed to be monotonic. Moreover, before consensus is reached across all nodes the test error at any given node and iteration index does not necessarily need to be greater than the centralized one.

It is also worth noticing the resemblance of the curves in the left and right panels of Figures 4, 5, 6 and 7. Although the test error is nearly identical for $JC = 1$ and $JC = 5$, this does not imply that the $\mathbf{v}_j(t)$ are nearly identical across iterations. Furthermore, the insensitivity w.r.t. small changes in JC reveals that in order to affect the classifier performance, the parameter JC must vary in the order of J. Relating the distributed setting with its centralized counterpart, it follows that with, for example, $J = 25$ a change in JC from 1 to 5 in the distributed setup of (6), corresponds to a change in C from 0.04 to 0.20 for the centralized setting of (1). Such a small change in C explains why the classification performance of the equivalent centralized scenarios is nearly identical as reflected in the figures.

In both biased and evenly distributed data, after a few iterations, MoM-DSVM yields an average performance close to the optimal one. It is also interesting to note that in the biased data case, nodes alone cannot construct an approximate classifier since they do not have samples from both classes. If an incremental approach were used it would need at least one full cycle through the network to enable construction of local estimators per node.

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 8. It is clear that in this case a larger η improves the convergence rate. Also, note that after a 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. Note that the communication cost of MoM-DSVM can be easily computed at any iteration in terms of the number of scalars transmitted across the network.

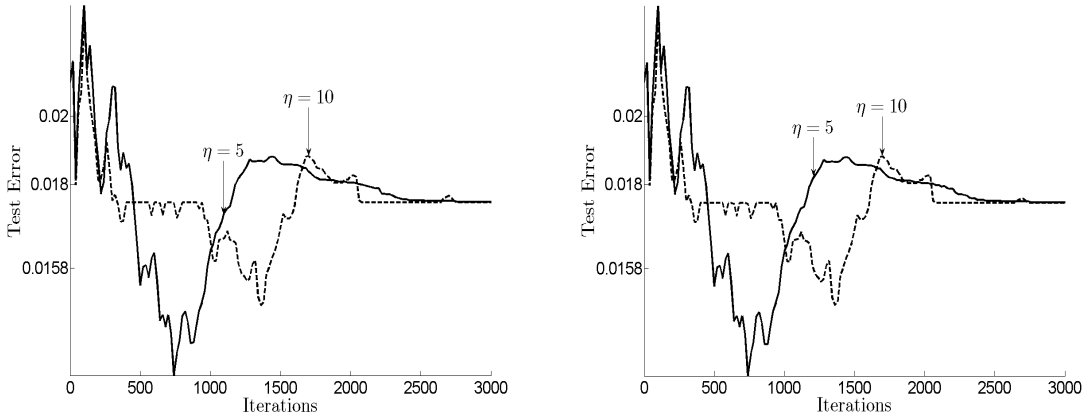

Figure 6: Evolution of the test error $(\mathbf{R}_{\text{Test}})$ of MoM-DSVM, with penalty coefficients $JC = 1$ (left) and $JC = 5$ (right), for a two-class problem using digits 2 and 9 from the MNIST data set evenly distributed across nodes, and a network with $J = 25$ nodes.

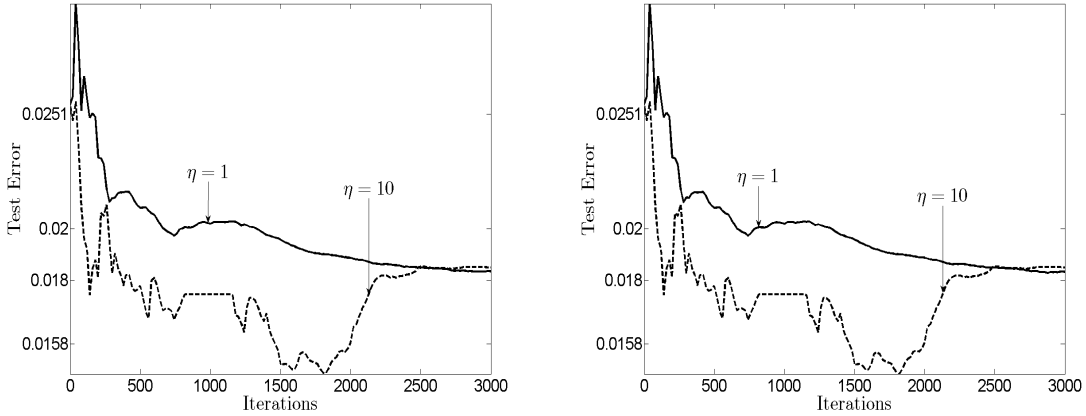

Figure 7: Evolution of the test error $(\mathbf{R}_{\text{Test}})$ of MoM-DSVM, with penalty coefficients $JC = 1$ (left) and $JC = 5$ (right), for a two-class problem using digits 2 and 9 from the MNIST data set evenly distributed across nodes, and a network with $J = 50$ nodes.

For the MNIST data set, the total communication cost up to iteration t is $785Jt$ scalars (cf. Section 3).

5.1.3 TEST CASE 3: SEQUENTIAL OPERATION

Consider 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 following fashion: at $t = 0$ each node has available one labeled training example drawn from the class distributions $\mathcal{C}_1$ and $\mathcal{C}_2$

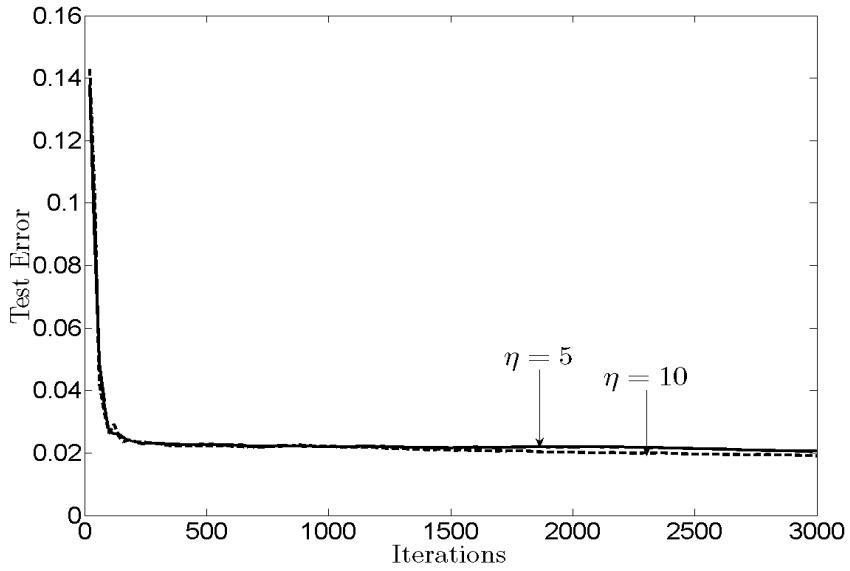

Figure 8: Evolution of the test error ($\mathbf{R}_{\text{Test}}$) of MoM-DSVM, with penalty coefficients $JC = 1$, for a two-class problem using digits 2 and 9 from the MNIST data set unevenly distributed across nodes, and a network with ring topology and $J = 25$ nodes.

described in Test Case 1. From $t = 0$ to $t = 19$, each node acquires a new labeled training example per iteration from this same distribution. From $t = 20$ 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 two-dimensional Gaussian distribution with covariance matrix $\Sigma_1 = [1,\ 0;\ 0,\ 2]$, and mean vector $m_1 = [-1,\ 5]^T$. 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 C_2 remains unchanged. During these iterations, we remove the training examples from $\mathcal{C}_1$ that were generated during the interval $t = 0$ to $t = 19$, one per iteration. From $t = 120$ to $t = 299$ nodes do not acquire new labeled training examples. From iteration $t = 300$ to $t = 499$, we include 8 new training examples per node and per iteration drawn only from class $\mathcal{C}_1$ with the same class-conditional distribution as the one used at the beginning of the algorithm $t = 0$. Finally, at iteration $t = 500$ all labeled training samples drawn from $t = 300$ to $t = 499$ are removed at each node at once, returning to the global data set available prior to iteration $t = 300$. The algorithm continues without any further change in the training set until convergence.

Figure 10 illustrates the tracking capabilities of the online MoM-DSVM scheme for different values of η. A total of 100 Monte Carlo runs were performed. The figure of merit in this case is $\mathcal{V}(t) := \frac{1}{J}\sum_{j=1}^{J} \left\|\mathbf{v}_j(t) - \mathbf{v}_c(t)\right\|$, where $\mathbf{v}_c(t)$ contains the coefficients of the centralized SVM using the training set available at time t. The peaks in Figure 10 correspond to the changes described in our experiment. 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 hindering the final performance of the algorithm. Figure 9 shows snapshots, for a single run of MoM-DSVM and $\eta = 30$, of the global training set

and local discriminant functions at different iterations. The solid training examples correspond to the current global SVs found by the online MoM-DSVM algorithm.

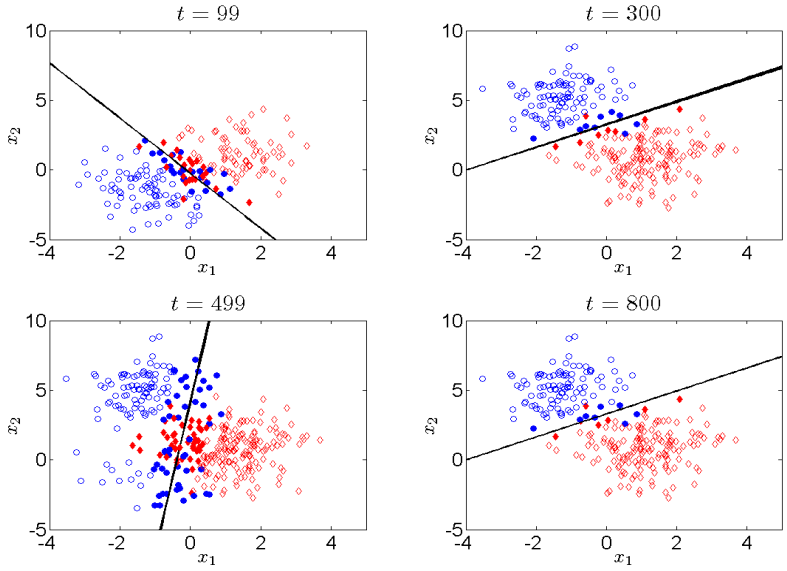

Figure 9: Snapshots of the global training data set and local linear discriminant $g_j^{(t)}(\mathbf{x})$ obtained with MoM-DSVM at all nodes for a synthetic training set evolving in time.

5.1.4 TEST CASE 4: COMMUNICATION COST COMPARISON

In this section, a comparison with the incremental SVM (ISVM) approach in Lu et al. (2008) is presented. The network with $J = 30$ nodes is considered again, where each node j has available a local training set with $N_j = N = 20$ with training vectors 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 $JC = 20$. The network topology is a ring; thus, ISVM entails no extra overhead due to inter-node communications. 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. For MoM-DSVM, this cost is fixed per iteration and equal to $3J$ scalars; recall that per iteration every node broadcasts $\mathbf{v}_j(t)$ to its neighborhood (cf. Algorithm 1). The ISVM approach locally trains an SVM and passes its local SVs to the next node in the cycle; the algorithm continues traversing the network until no SVs are shared among neighboring nodes. Thus, the communication cost per iteration depends on the number of SVs found at each node, that is, $3 \times \{\#\text{ of SVs at node } j\}$. A contingency strategy to prevent SVs from being transmitted multiple times by the same node as well as to prevent repetition of training set elements at individual nodes is run in parallel with the ISVM algorithm.

Figure 11 depicts the cumulative communication 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

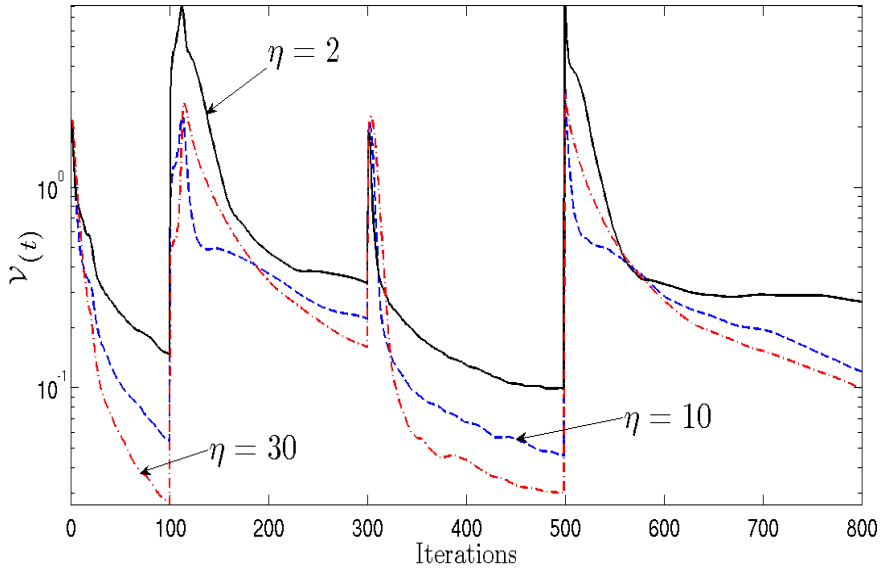

Figure 10: Average error $\mathcal{V}(t)$ of MoM-DSVM for a synthetic training set evolving in time evaluated for various values of η. The peaks correspond to iteration indexes where the local training sets were modified.

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 thus, in the context of WSNs (see Example 1), longer battery life for individual nodes.

5.2 Nonlinear Classifier

In this section, we present experiments on synthetic and real data to illustrate the performance of our distributed method for training nonlinear SVMs.

5.2.1 TEST CASE 5: SYNTHETIC TRAINING SET

Consider the same network as in Test Case 1. Each node acquires labeled training examples from two different equiprobable classes $\mathcal{C}_1$ and $\mathcal{C}_2$. Class $\mathcal{C}_1$ contains now examples from a two-dimensional Gaussian distribution with covariance matrix $\Sigma_1 = [0.6, 0; 0, 0.4]$, and mean vector $m_1 = [0,0]^T$. Class $\mathcal{C}_2$ is a mixture of Gaussian distributions with mixing parameters $\pi_{21} = 0.3$ and $\pi_{22} = 0.7$; mean vectors $m_2 = [-1,-1]^T$ and $m_3 = [2,2]^T$; and, equal covariance matrix Σ. The optimal Bayes classifier here is clearly nonlinear.

We generate a matrix Γ with rows taken from a uniform two-dimensional grid of L points. The extreme values of the grid are chosen equal to the extreme points of the global training set. Local training sets are of size $N_j = 10\ \forall j \in \mathcal{J}$, and are generated from the distributions described in the previous paragraph. Each node uses its local training sets as well as the matrix Γ to build the local

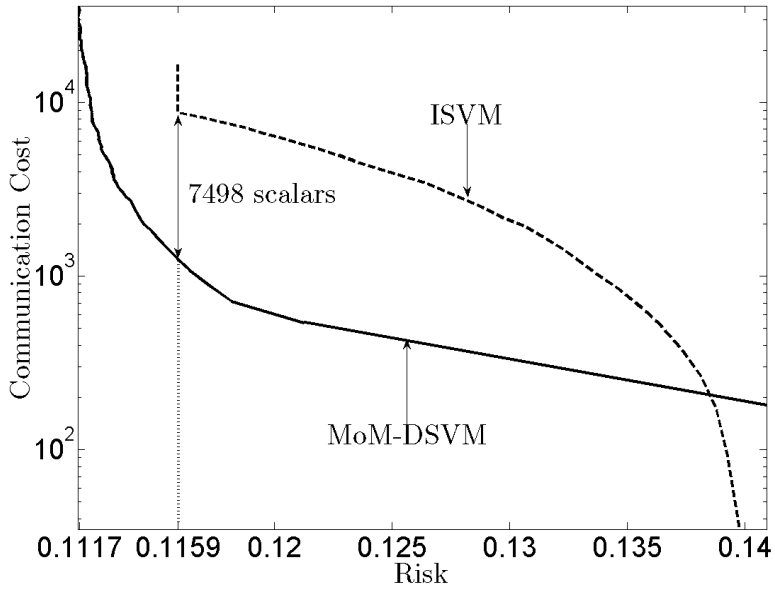

Figure 11: Communication cost, measured in terms of the number of scalars transmitted, of MoM-DSVM and ISVM for a network with ring topology and $J = 30$.

classifier, as described in (24). A Gaussian kernel with $\sigma^2 = 0.9$ and $\eta = 10$ was employed to construct a global nonlinear classifier. Figure 12 (left) shows the classification performance on the points $\{\chi_l\}_{l=1}^{L}$; that is, the classification performance when the testing set is given by $\{(\chi_l, y_l) : l = 1, \ldots, L\}$, where y_l indicates the class from which χ_l was originally drawn. For comparison, we have also included the Bayes risk, the centralized SVM empirical risk, and the local SVM risk. As expected, the classification performance of the distributed classifier approaches that of the centralized one.

Figure 12 (right) illustrates the performance of MoM-NDSVM on a randomly-generated test set of size $N_T = 600$ for various choices of L and JC. Matrix Γ was taken from a uniform two-dimensional grid of L points as before. A total of 500 Monte Carlo runs were performed. Clearly, the asymptotic performance of MoM-NDSVM rapidly outperforms the average performance of a locally-trained SVM and closely converges to the centralized SVM for larger values of L with all other parameters fixed. However, it is worth observing that the choice of the parameter JC also influences the performance. Large values for JC promote reduced number of prediction errors on the training set (possibly) leading to over-fitting. Various strategies, such as cross validation, can be implemented to select optimal values for both JC and σ^2 at the expense of training with MoM-NDSVM multiple times. To visualize the results, Figures 13 and 14 depicts the form of the discriminant function for several values of L at 6 different nodes in the network. Centralized and local discriminant functions are also included as benchmarks. Even though the nodes do not exactly agree on the final form of $g_j(\mathbf{x})$ at all points, their classification performance closely converges to the one obtained by the centralized SVM benchmark.

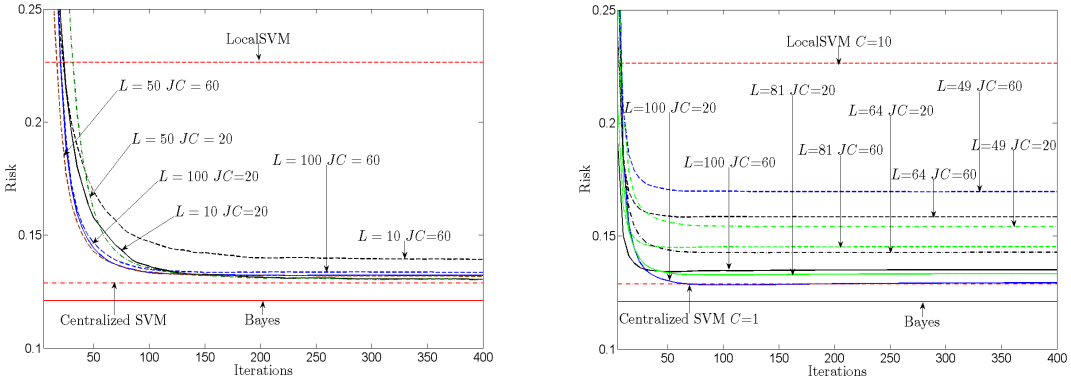

Figure 12: Evolution of prediction error $(\mathbf{R}_{\text{Pred}})$, where matrix Γ is considered a classification query of size L (left); and test error $(\mathbf{R}_{\text{Test}})$, where matrix Γ is constructed as a random grid with L points (right), for MoM-NDSVM applied to a two-class problem using synthetic data and a network with $J = 30$ nodes.

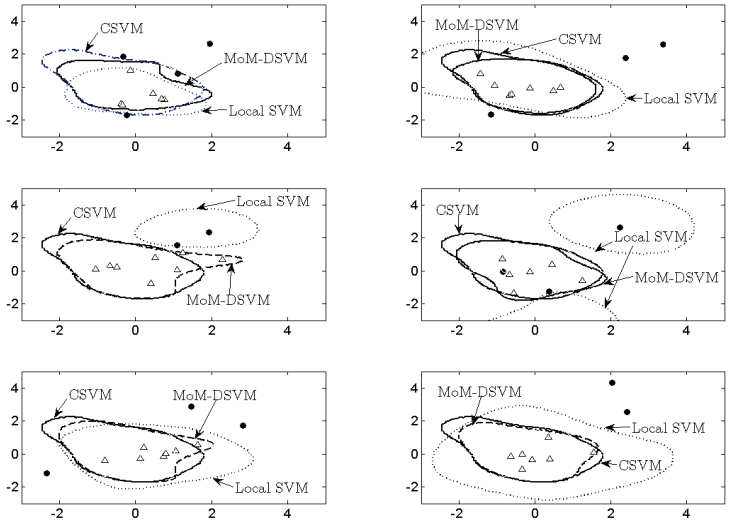

Figure 13: Comparison of the discriminant functions found by a centralized SVM, local SVMs, and the MoM-NDSVM algorithm at 6 different nodes of a network with $J = 30$ using synthetic data. A penalty term $JC = 20$ and a random grid with $L = 100$ were used.

5.2.2 TEST CASE 6: UCI TRAINING SETS

Four data sets from the UCI repository have been chosen to test our MoM-NDSVM algorithm: Iris, Wine, Puma Indians Diabetes, and Parkinsons (Asuncion and Newman, 2007). A brief description

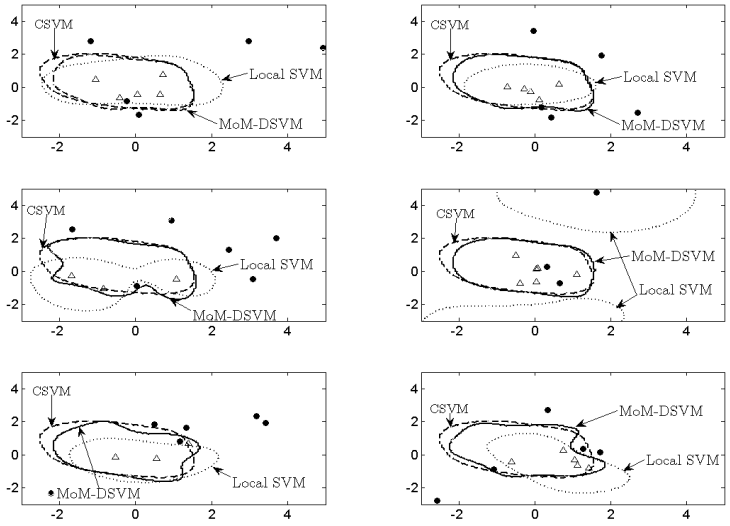

Figure 14: Comparison of the discriminant functions found by a centralized SVM, local SVMs, and the MoM-NDSVM algorithm at 6 different nodes of a network with $J = 30$ using synthetic data. A penalty term $JC = 60$ and a random grid with $L = 100$ were used.

Data set	Classes	Dim. Features	Size	Train. Set	Test Set
Iris	3	4	150	12	40
Wine	3	13	178	12	40
Diabetes	2	8	768	50	200
Parkinsons	2	23	197	12	40

Table 2: UCI data sets

of each of the data sets is shown in Table 2. Examples from the data sets are randomly split among $J = 5$ nodes in a fully-connected network. Focusing on the binary classification problem, only classes 2 and 3 from the Iris data set and classes 1 and 2 from the Wine data set are used. For simulation purposes, each local training set as well as the testing set have the same number of examples from each class.

Table 3 compares performance of the classifiers constructed via MoM-NDSVM with the average performance of the 5 local classifiers trained with local training sets only, and with the one of a centralized SVM trained with the training set available to the whole network. A total of 100 Monte Carlo runs were performed per data set, where both training and testing sets were drawn randomly per run. The MoM-NDSVM parameters JC and η were chosen via cross-validation for every training set as in Hastie et al. (2009, Ch. 7). Gaussian kernels as in the previous section were used. The local and centralized SVMs were trained using the Spider toolbox for MATLAB (Weston et al., 2006). To evaluate the local performance of the classifiers, each node trains a local SVM and its performance is compared with the one obtained via MoM-NDSVM. For each training set we

Data set	Local	Centralized	MoM-NDSVM $L = 150$	MoM-NDSVM $L = 300$	Class. Query
Iris	8.39%	4.43%	5.15%	5.26%	4.28%
Wine	15.71%	6.17%	7.37%	7.33%	6.60%
Diabetes	34.52%	24.40%	29.69%	28.92%	23.76%
Parkinsons 1	33.76%	18.45%	30.14%	31.13%	18.56%
Parkinsons 2	34.78%	18.86%	23.60%	24.05%	20.28%

Table 3: UCI data sets centralized versus local versus distributed performance comparison for $t = 1,000$. Parkinsons 2 is the normalized Parkinsons training set.

explore two cases: (i) local classifiers at each node; and (ii) Γ as a classification query. Figure 15 plots the training evolution of MoM-NDSVM for the Puma Indians Diabetes data set.

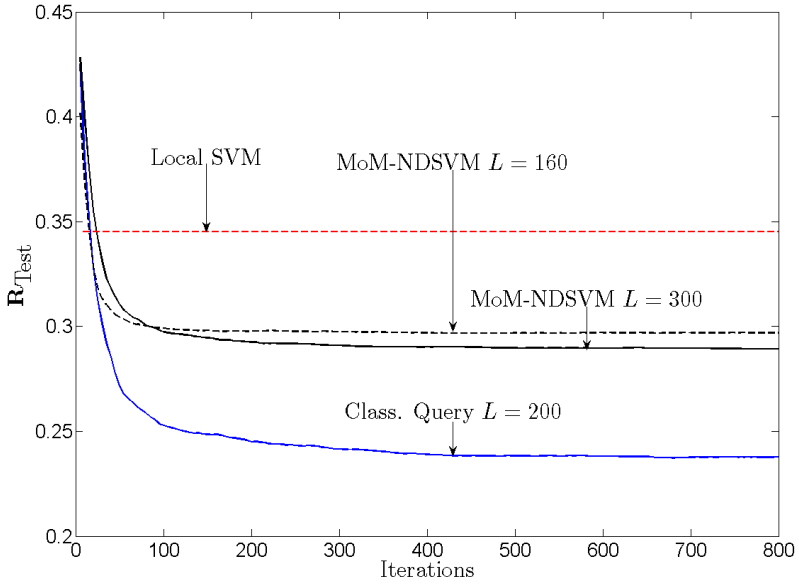

Figure 15: Evolution of test error ($\mathbf{R}_{\text{Test}}$) for Puma Indians Diabetes data set taken from the UCI repository, and a fully connected network with $J = 5$ nodes.

The performance of MoM-NDSVM for (i) depends heavily on the choice of Γ. To illustrate this point, the size of the local training sets per node has been chosen small when compared to the dimensionality of the feature space. Let $x_k^{\min}$ and $x_k^{\max}$ correspond to the smallest and largest values that the k-th feature can take. The row-elements of matrix Γ are chosen randomly and independently over the interval $[x_k^{\min}, x_k^{\max}]$ per component. Two different values of L were chosen to compare the performance of MoM-NDSVM. For small values of p, Γ can be chosen as a grid of M uniformly spaced points per dimension; therefore, $L = M^p$. The results summarized in Table 3 highlight

	$\eta = 10$		$\eta = 20$	
L	$p_0 = 49$	$p_0 = 81$	$p_0 = 49$	$p_0 = 81$
400	0.370%	1.246%	0.504%	1.448%
800	0.136%	0.242%	0.234%	0.654%

Table 4: Average MoM-NDSVM risk (Gaussian kernel) at iteration $t = 3,000$ for compressed MNIST data set with dimensionality p_0.

the fact that the classification performance at each node remains limited by the training examples available locally. However, in this extremely challenging case, collaboration among nodes improves the overall classification performance of the network.

Table 3 also illustrates how conditioning of the data together with the choice for the kernel function can impact the performance of MoM-NDSVM. In particular, its last two rows compare the classification performance achieved for the Parkinsons training set without normalization (Parkinsons 1) and with its features normalized to have maximum absolute size unity (Parkinsons 2). Although both centralized and local performance remain nearly unchanged, the MoM-NDSVM performance improves about 7% for both $L = 150$ and $L = 300$. An intuitive explanation follows from looking closer at the values of the features in the Parkinsons training set. Some features take values in the order of 10^2 while others take values in the order of 10^{-3}; thus, a Gaussian kernel that spans symmetrically along all directions is not the best kernel choice for this case. After normalization, a smaller number L of Gaussian kernels can be used to obtain a better representation of the decision surface. In conclusion, data across nodes must be preprocessed whenever possible to achieve a better trade off between classification performance and the computational complexity of MoM-NDSVM.

Note that the classification performance for case (ii) approaches the centralized SVM one. After a few iterations, the classification accuracy returned by the network surpasses that of locally trained SVMs. The speed of convergence might be hindered if L is chosen large. Fine tuning of η can achieve a desirable trade-off between speed of convergence and performance in terms of test error.

5.2.3 TEST CASE 7: MNIST TRAINING SET

Consider a network with $J = 25$, algebraic connectivity 3.2425, and average degree per node 12.8. Local training sets have been constructed based on the MNIST data set using digits 2 and 9 only. The nodes wish to train a nonlinear global classifier using a Gaussian kernel via MoM-NDSVM. Each node j has available a training set $\mathcal{S}_j$ with 472 examples from one class only, thus individual nodes cannot construct a classifier locally. The large size of the images in MNIST leads to an excessively large choice for $L = L_0 \gg 784$, hindering the convergence of MoM-NDSVM. Instead, each image has been compressed via principal component analysis (PCA) to vectors of dimensionality $p_0 < 784$. It was observed experimentally that after compression, the two classes become separable. Indeed, the centralized equivalent SVM yields test error zero. Figure 16 depicts the performance of MoM-NDSVM for various choices of η and p_0. Note that $\eta = 20$ leads to slower convergence of the average risk across the network. Table 4 summarizes the classification performance of MoM-NDSVM after 3,000 iterations. A larger value for L improves the average classification performance of the network. However, the number of iterations required for MoM-NDSVM to converge increases with L.

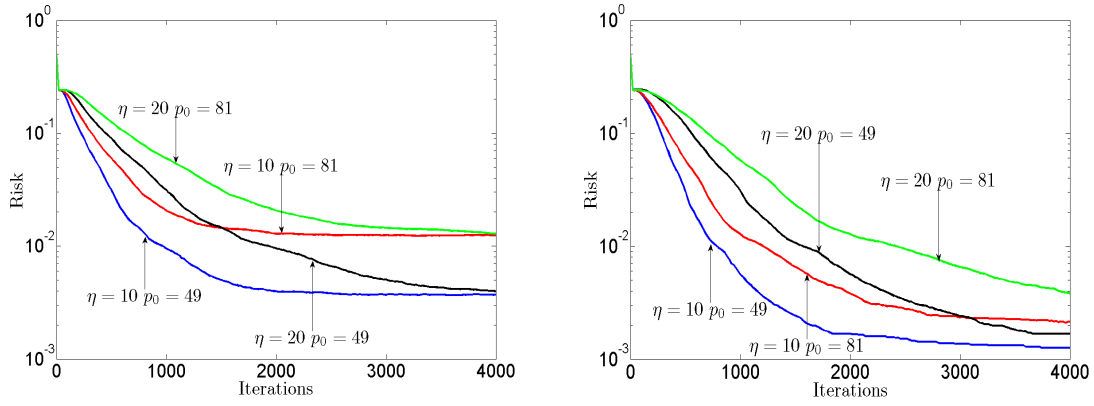

Figure 16: Average evolution of MoM-NDSVM risk (Gaussian kernel) for compressed MNIST data set for $L = 400$ (left), and $L = 800$ (right) in a network with $J = 25$ nodes. MNIST images have been compressed to dimensionality p_0 via PCA.

5.3 Noisy Inter-node Communications

This subsection presents robustness tests of the novel distributed classification scheme with noisy inter-node exchanges. Such noise is due to, for example, quantization error, additive Gaussian noise at the receiving ends, or, Laplacian noise intentionally added to transmitted samples in order to guarantee data privacy (Chaudhuri and Monteleoni, 2008; Dwork et al., 2006). Although focus is placed on MoM-DSVM, the results also carry over to MoM-NDSVM.

5.3.1 TEST CASE 8: MOM-DSVM WITH PERTURBED TRANSMISSIONS

In this setting, per iteration t, each node j purposely introduces a perturbation $\varepsilon_j(t)$ to the variable $\mathbf{v}_j(t)$ before transmission. Perturbed transmissions can be used to preserve data privacy (Dwork et al., 2006). Consider an eavesdropper accessing the noisy versions of $\mathbf{v}_j(t)$. The form and variance level Σ_j of the local perturbations $\varepsilon_j(t)$ can be adjusted per node to prevent the eavesdropper from learning $\mathcal{S}_j$. For instance, Dwork et al. (2006) suggests introducing zero-mean Laplacian random variable whose variance depends on the sensitivity of $\mathbf{v}_j(t)$ as a function of $\mathcal{S}_j$.

The MoM-DSVM iterations, with $JC = 5$ and $\eta = 10$, are modified by introducing local perturbations $\varepsilon_j(t)$ to $\mathbf{v}_j(t)$. Each $\varepsilon_j(t)$ is zero-mean Laplacian distributed and white across time and space, that is, $\mathbb{E}\{\varepsilon_j(t_1)\varepsilon_j^T(t_2)\} = 0$ if $t_1 \neq t_2$ and $\mathbb{E}\{\varepsilon_i(t)\varepsilon_j^T(t)\} = 0$ if $i \neq j$ $\forall i, j \in \mathcal{J}$. The resulting MoM-DSVM iterations are

$$\begin{aligned}
\lambda_j(t+1) &= \arg\max_{\lambda_j:\, \mathbf{0}_j \preceq \lambda_j \preceq JC\mathbf{1}_j} -\frac{1}{2}\lambda_j^T \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{X}_j^T \mathbf{Y}_j \lambda_j + \left(\mathbf{1}_j + \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{f}_j(t)\right)^T \lambda_j, \\
\mathbf{v}_j(t+1) &= \mathbf{U}_j^{-1}\left[\mathbf{X}_j^T \mathbf{Y}_j \lambda_j(t+1) - \mathbf{f}_j(t)\right], \\
\alpha_j(t+1) &= \alpha_j(t) + \frac{\eta}{2} \sum_{i \in \mathcal{B}_j} \left[(\mathbf{v}_j(t+1) + \varepsilon_j(t)) - (\mathbf{v}_i(t+1) + \varepsilon_i(t))\right]
\end{aligned}$$

where $\mathbf{U}_j = (1 + 2\eta|\mathcal{B}_j|)\mathbf{I}_{p+1} - \Pi_{p+1}$ and $\mathbf{f}_j(t) := 2\alpha_j(t) - \eta \sum_{i \in \mathcal{B}_j}[\mathbf{v}_j(t) + \varepsilon_j(t) + \mathbf{v}_i(t) + \varepsilon_i(t)]$ (cf. Proposition 1). In this case, MoM-DSVM operates in an analogous manner to Algorithm 1, differing

only in their broadcasting step. In the perturbed transmissions case, every node j broadcasts a perturbed vector $\mathbf{v}_j(t)+\varepsilon_j(t)$ (instead of $\mathbf{v}_j(t)$ alone) to its one-hop neighbors. Note that neighboring nodes $i \in \mathcal{B}_j$ only "see" the aggregate perturbed vector $\mathbf{v}_j(t)+\varepsilon_j(t)$ from node j.

Figure 17 illustrate the performance of MoM-DSVM after 100 Monte Carlo runs with perturbed transmissions for a network with $J = 8$ nodes, algebraic connectivity 0.4194, and average degree per node 2.5. Nodes collect observations from 2 classes $\mathcal{C}_1$ and $\mathcal{C}_2$, where $\mathcal{C}_1$ is $\mathcal{N}(m_1, \Sigma_1)$ with $m_1 = [0,\ 0]^T$, and $\Sigma_1 = [0.6,\ 0;\ 0,\ 0.4]$, and $\mathcal{C}_2$ is $\mathcal{N}(m_2, \Sigma_2)$ with $m_2 = [2,\ 2]^T$, and $\Sigma_2 = [1,\ 0;\ 0,\ 2]$. Each node collects an equal number of observations per class for a total of $N_j = N = 50$ observations. The noise $\varepsilon_j(t)$, inserted per transmission per node, has covariance matrix given by $\sigma^2\mathbf{I}_3$. 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 to 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.

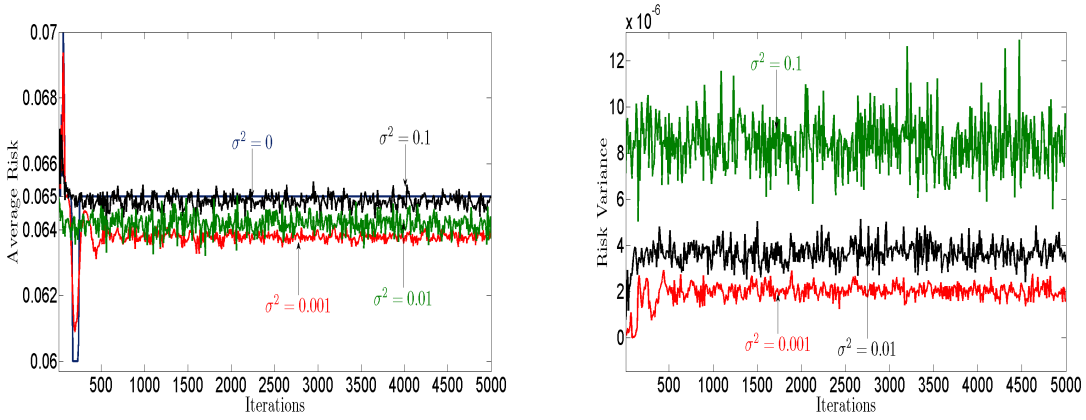

Figure 17: Average risk (left) and risk variance (right) for a network with $J = 8$, and a finite variance perturbation added to $\mathbf{v}_j(t)$ before it is broadcasted.

5.3.2 TEST CASE 9: NOISY COMMUNICATION LINKS

The MoM-DSVM is also robust to non-ideal inter-node links corrupted by additive noise due to, for example, quantization or additive Gaussian receiver noise. In this case, the noise is added at the receiver side. The MoM-DSVM must be modified to obtain a bounded variance on the estimates $\mathbf{v}_j(t)$, and the local Lagrange multipliers $\alpha_{ji}(t) := \alpha_{ji1}(t)$ must be exchanged among neighboring nodes; see Zhu et al. (2009) for similar approaches. Each communication link between node j and node $i \in \mathcal{B}_j$ introduces additive noise $\varepsilon^v_{ji}(t)$ ($\varepsilon^\alpha_{ji}(t)$) to the variable $\mathbf{v}_j(t)$ (α_{ji}). The perturbations $\{\varepsilon^v_{ji}(t)\}$ ($\{\varepsilon^\alpha_{ji}(t)\}$) are zero-mean random variables with covariance matrix Σ^v_{ji} (Σ^α_{ji}), white across

Algorithm 4 MoM-DSVM with noisy links

Require: Randomly initialize $\mathbf{v}_j(0)$, and $\alpha_{ji}(0) = \mathbf{0}_{(p+1)\times 1}\ \forall\ j \in \mathcal{J}\ \forall i \in \mathcal{B}_j$

for $t = 0, 1, 2, \ldots$ **do**
  for all $j \in \mathcal{J}$ **do**
    Compute $\lambda_j(t+1)$ via (41).
    Compute $\mathbf{v}_j(t+1)$ via (42).
  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 $\alpha_{ji}(t+1)$ via (43).
  end for
  for all $j \in \mathcal{J},\ i \in \mathcal{B}_j$ **do**
    Transmit $\alpha_{ji}(t+1)$ to $i \in \mathcal{B}_j$.
  end for
end for

time and space. The modified MoM-DSVM iterations are

$$\lambda_j(t+1) = \underset{\lambda_j:\, \mathbf{0}_j \preceq \lambda_j \preceq JC\mathbf{1}_j}{\arg\max} -\frac{1}{2}\lambda_j^T \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{X}_j^T \mathbf{Y}_j \lambda_j + \left(\mathbf{1}_j + \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{f}_j(t)\right)^T \lambda_j, \tag{41}$$

$$\mathbf{v}_j(t+1) = \mathbf{U}_j^{-1}\left[\mathbf{X}_j^T \mathbf{Y}_j \lambda_j(t+1) - \mathbf{f}_j(t)\right], \tag{42}$$

$$\alpha_{ji}(t+1) = \alpha_{ji}(t) + \frac{\eta}{2}\left[\mathbf{v}_j(t+1) - (\mathbf{v}_i(t+1) + \varepsilon_{ij}^v(t))\right] \tag{43}$$

where $\mathbf{f}_j(t) := \sum_{i\in\mathcal{B}_j}\left\{\alpha_{ji}(t) - (\alpha_{ij}(t) + \varepsilon_{ij}^{\alpha}(t)) - \eta[\mathbf{v}_j(t) + (\mathbf{v}_i(t) + \varepsilon_{ij}^v(t))]\right\}$. The resulting MoM-DSVM algorithm with noisy links is summarized as Algorithm 4.

The left panels of Figures 18 and 19 depict the average performance after 100 Monte Carlo runs of MoM-DSVM for the same network of Test Case 8. As seen, the variance of the estimates $\mathbf{v}_j(t)$ yielded by the modified MoM-DSVM algorithm remains bounded.

Incremental approaches are hindered by noisy communication links because noise added to the SVs perturbs and accumulates in the local training sets. In ISVM, SVs are bound to percolate across the network, and even to come back to the node where they originated. Due to the noise, however, nodes cannot recognize noisy feature vectors already in $\mathcal{S}_j$. This is problematic since the size of local problems being solved per node increases linearly with the size of the training set, thus requiring a heuristic size-control scheme. The right panels of Figures 18 and 19 show the performance of an ISVM for different levels of noise variance. Noise is added to the SVs and noisy labels are rounded to 1 or -1. Different from MoM-DSVM, the performance of ISVM quickly deteriorates, even for low noise levels since the average risk approaches 0.5 after a few iterations, which amounts to pure guessing of the binary classifier.

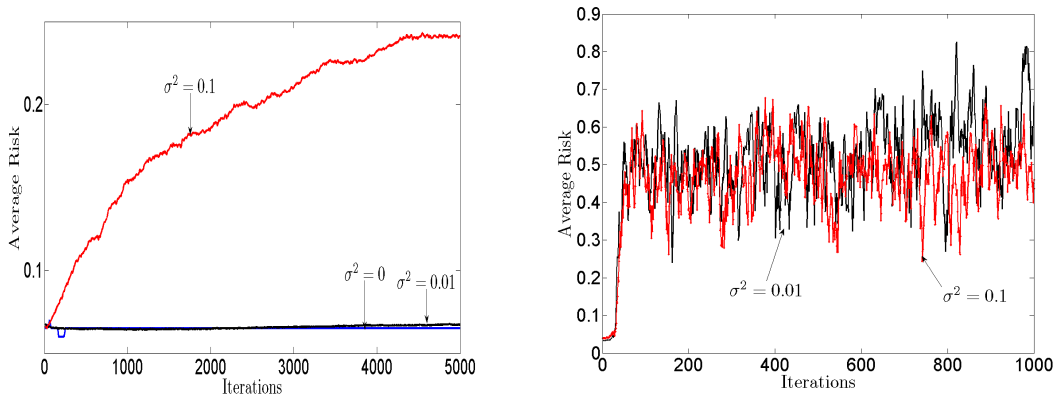

Figure 18: Average risk for a network with $J = 8$, and noisy communication links using a synthetic data set. MoM-DSVM (left) and incremental SVM approach (right).

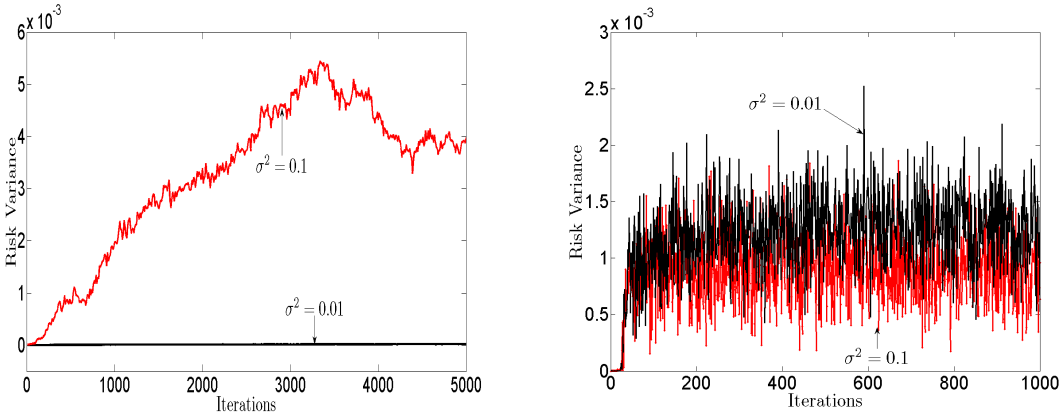

Figure 19: Risk variance for a network with $J = 8$ and noisy communication links corresponding to average risk in Figure 18. MoM-DSVM (left) and incremental SVM approach (right).

6. Conclusions

This work developed distributed SVM algorithms by reformulating the centralized SVM training problem into per-node separable sub-problems linked via consensus constraints, which can be solved using decentralized optimization tools. The novel algorithms are well suited for applications involving data that cannot be shared among collaborating nodes, which possibly operate under stringent resources, and may thus desire to reduce overhead of inter-node message exchanges.

Based on distributed training sets, the novel MoM-DSVM algorithm constructs a maximum-margin linear classifier iteratively. At every iteration, locally updated classifier vectors are exchanged among neighboring nodes. Convergence to the centralized linear SVM formulation is guaranteed. The approach lends itself naturally to online and asynchronous variants, which allow adaptation of the proposed DSVM to scenarios when elements of the local training sets become

available sequentially, or, when outdated elements need to be removed. Furthermore, the MoM-DSVM can be generalized to construct distributed nonlinear discriminant functions. The resulting iterative MoM-NDSVM algorithm is provably convergent to its centralized counterpart, and its complexity is kept at a manageable level by using the kernel trick. Local classifiers are limited by the span of their local training sets, and a set of basis common to all nodes.

Although not formally treated, the novel distributed classification algorithms can be readily extended to solve distributed support vector regression (DSVR) problems. The main characteristics of the present approach, such as its convexity, remain unchanged. Therefore, it is expected that linear and nonlinear estimators developed for MoM-DSVR, will enjoy convergence claims similar to those proved here for MoM-DSVM and MoM-NDSVM classifiers. To complement the distributed supervised classifiers introduced here, our current research deals with consensus-based distributed versions of the unsupervised k-means and expectation-maximization clustering algorithms.

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. Part of this work appears in *Proc. of 9th ACM/IEEE Conference on Information Processing in Sensor Networks*, Stockholm, Sweden, April 16-19, 2010.

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

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U. S. Government.

Appendix A. The Alternating Direction Method of Multipliers

The ADMoM is a distributed optimization algorithm solving the following problem

$$\begin{aligned} \min_{\mathbf{v}} \quad & F_1(\mathbf{v}) + F_2(\mathbf{A}\mathbf{v}) \\ \text{s.t.} \quad & \mathbf{v} \in \mathcal{P}_1,\ \mathbf{A}\mathbf{v} \in \mathcal{P}_2 \end{aligned} \tag{44}$$

where $F_1 : \mathbb{R}^{p_1} \to \mathbb{R}$ and $F_2 : \mathbb{R}^{p_2} \to \mathbb{R}$ are convex functions, $\mathbf{A}$ is a $p_2 \times p_1$ matrix, while $\mathcal{P}_1 \subset \mathbb{R}^{p_1}$ and $\mathcal{P}_2 \subset \mathbb{R}^{p_2}$ denote non-empty polyhedral sets.

Upon introducing the auxiliary variable $\omega \in \mathbb{R}^{p_2}$, ADMoM solves the separable problem

$$\begin{aligned} \min_{\mathbf{v},\omega} \quad & F_1(\mathbf{v}) + F_2(\omega) \\ \text{s.t.} \quad & \mathbf{A}\mathbf{v} = \omega \\ & \mathbf{v} \in \mathcal{P}_1,\ \omega \in \mathcal{P}_2 \end{aligned} \tag{45}$$

which is clearly equivalent to (44). With $\alpha \in \mathbb{R}^{p_2}$ denoting the Lagrange multiplier corresponding to the constraint $\mathbf{Av} = \omega$, the augmented Lagrangian corresponding to (45) is

$$\mathcal{L}(\mathbf{v}, \omega, \alpha) = F_1(\mathbf{v}) + F_2(\omega) + \alpha^T(\mathbf{Av} - \omega) + \frac{\eta}{2}\|\mathbf{Av} - \omega\|^2 \tag{46}$$

where the parameter $\eta > 0$ controls the impact of the constraint violation in (45). The ADMoM minimizes alternately $\mathcal{L}$ in (46) w.r.t. the primal variable $\mathbf{v}$, then w.r.t. the auxiliary variable ω, and after each cycle it uses these iterates to update the multiplier. Specifically, with t denoting iteration index, the ADMoM iterates at $t+1$ are given by

$$\mathbf{v}(t+1) = \arg\min_{\mathbf{v}\in\mathcal{P}_1} \mathcal{L}(\mathbf{v}, \omega(t), \alpha(t)), \tag{47}$$

$$\omega(t+1) = \arg\min_{\omega\in\mathcal{P}_2} \mathcal{L}(\mathbf{v}(t+1), \omega, \alpha(t)), \tag{48}$$

$$\alpha(t+1) = \alpha(t) + \eta\left(\mathbf{Av}(t+1) - \omega(t+1)\right). \tag{49}$$

Thanks to the auxiliary variable ω, each of the optimization problems (47) and (48) can be run separately, possibly by different processors. The following proposition states the main claim regarding convergence of the ADMoM iterates, and its proof can be found in Bertsekas and Tsitsiklis (1997, Ch. 3, Proposition 4.2).

Proposition 4 *Assume that the optimal solution set v^* of* (44) *is non-empty, and either $\mathcal{P}_1$ is bounded, or, $\mathbf{A}^T\mathbf{A}$ is nonsingular. Then, a sequence $\{\mathbf{v}(t), \omega(t), \alpha(t)\}$ generated by the iterations* (47)-(49) *is bounded, and every limit point of $\{\mathbf{v}(t)\}$ is an optimal solution of* (44)*. Furthermore, $\{\alpha(t)\}$ converges to an optimal solution α^* of the dual problem [cf.* (45)*]*

$$\min_{\alpha\in\mathbb{R}^{p_2}} \quad H_1(\alpha) + H_2(\alpha)$$

where for all $\alpha \in \mathbb{R}^{p_2}$

$$H_1(\alpha) := \inf_{\mathbf{v}\in\mathcal{P}_1} [F_1(\mathbf{v}) + \alpha^T\mathbf{Av}],$$
$$H_2(\alpha) := \inf_{\omega\in\mathcal{P}_2} [F_2(\omega) - \alpha^T\omega] .$$

Appendix B. Proof of Lemma 1

First, the equality constraints $\{\mathbf{w}_j = \mathbf{w}_i\}$ and $\{b_j = b_i\}$ will be shown equivalent to $\mathbf{w}_1 = \cdots = \mathbf{w}_J$ and $b_1 = \cdots = b_J$, respectively, for any feasible solution of (6). Consider any two nodes j_0 and j_k both in $\mathcal{J}$. Since the network is connected, there exists a path $\{j_0 j_1 \ldots j_{k-1} j_k\}$ of length at least one, which connects nodes j_0 and j_k. Because $j_{\ell+1} \in \mathcal{B}_\ell$ for $\ell = 0, 1, \ldots, k-1$, it is immediate that $\mathbf{w}_{j_\ell} = \mathbf{w}_{j_1} = \cdots = \mathbf{w}_{j_{k-1}} = \mathbf{w}_{j_k}$. Since j_ℓ, $j_k \in \mathcal{J}$ are arbitrary, it follows readily that $\mathbf{w}_1 = \cdots = \mathbf{w}_J$. A similar argument leads to $b_1 = \cdots = b_J$.

As any feasible solution of (6) satisfies $\mathbf{w}_1 = \cdots = \mathbf{w}_J = \mathbf{w}$ and $b_1 = \ldots = b_J = b$, problem (6) becomes

$$\begin{aligned} \min_{\mathbf{w},b,\{\xi_j\}} \quad & J\left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{j=1}^{J}\mathbf{1}_j^T\xi_j\right) \\ \text{s.t.} \quad & \mathbf{Y}_j(\mathbf{X}_j\mathbf{w} + b\mathbf{1}_j) \succeq \mathbf{1}_j - \xi_j \quad \forall j \in \mathcal{J} \\ & \xi_j \succeq \mathbf{0}_j \end{aligned} \tag{50}$$

which is equivalent to (1), since the constant J can be dropped from the cost function in (50).

Appendix C. Proof of Lemma 2

The objective here is to cast (7) in the form of (45), and thus show that iterations (9)-(12) correspond to the ADMoM iterations (47)-(49) in Appendix A. First, it will be shown that the set of consensus constraints in (7), namely $\{\mathbf{v}_j = \omega_{ji},\ \omega_{ji} = \mathbf{v}_i\ :\ \forall j \in \mathcal{J},\ \forall i \in \mathcal{B}_j\}$, can be written as the equality constraint $\mathbf{A}\mathbf{v} = \omega$ in (45). With this objective in mind, consider listing the constraints $\mathbf{v}_j = \omega_{ji}$ across nodes $j \in \mathcal{J}$ and neighbors $i \in \mathcal{B}_j$ as follows

$$\begin{aligned} \{\mathbf{v}_1 &= \omega_{1i}\}_{i\in\mathcal{B}_1} \\ &\vdots \\ \{\mathbf{v}_J &= \omega_{Ji}\}_{i\in\mathcal{B}_J}. \end{aligned} \tag{51}$$

Since for every $\mathbf{v}_j$ there are $|\mathcal{B}_j|$ constraints, the total number of equalities in (51) is $\sum_{j=1}^{J}|\mathcal{B}_j| = 2|\mathcal{E}|$, where $|\mathcal{E}|$ is the number of edges in the network. The factor 2 in $2|\mathcal{E}|$ is because for every edge $j \leftrightarrow i$ there are two constraints, namely $\mathbf{v}_j = \omega_{ji}$ and $\mathbf{v}_i = \omega_{ij}$.

The set of equalities in (51) can be written in matrix-vector form as

$$\begin{matrix} |\mathcal{B}_1| \text{ vectors} \\ \\ |\mathcal{B}_J| \text{ vectors} \end{matrix} \left\{ \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_J \\ \vdots \\ \mathbf{v}_J \end{bmatrix} \right. = \underbrace{\begin{bmatrix} \underbrace{\begin{bmatrix} \mathbf{I}_{p+1} \\ \vdots \\ \mathbf{I}_{p+1} \end{bmatrix}}_{\mathbf{A}_1:=} & & \\ & \ddots & \\ & & \underbrace{\begin{bmatrix} \mathbf{I}_{p+1} \\ \vdots \\ \mathbf{I}_{p+1} \end{bmatrix}}_{\mathbf{A}_J:=} \end{bmatrix}}_{\mathbf{A}':=} \underbrace{\begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_J \end{bmatrix}}_{\mathbf{v}:=} = \underbrace{\begin{bmatrix} \{\omega_{1i}\}_{i\in\mathcal{B}_1} \\ \vdots \\ \{\omega_{Ji}\}_{i\in\mathcal{B}_J} \end{bmatrix}}_{\omega':=} \tag{52}$$

where $\mathbf{A}'\mathbf{v}$ replicates $\mathbf{v}$ in accordance with the left-hand-side (l.h.s.) of (51). Matrix $\mathbf{A}'$ in (52) is block-diagonal with block entries $\mathbf{A}_j := [\mathbf{I}_{p+1}, \ldots, \mathbf{I}_{p+1}]^T$ containing $|\mathcal{B}_j|$ identity matrices of size $(p+1)\times(p+1)$. Since $\mathbf{v}_j$ and ω_{ji} are $(p+1)\times 1$ vectors, $\mathbf{v}$ has size $(p+1)J \times 1$, and ω' has size $2(p+1)|\mathcal{E}| \times 1$.

Equation (52) shows that the constraints of the form $\mathbf{v}_j = \omega_{ji}$ can be compactly written as

$$\mathbf{A}'\mathbf{v} = \omega'. \tag{53}$$

Consider now the remaining constraints, which are of the form $\omega_{ji} = \mathbf{v}_i$, and can be listed explicitly as [cf. (51)]

$$\begin{aligned} \{\mathbf{v}_1 &= \omega_{j1}\}_{j\in\mathcal{B}_1} \\ &\vdots \\ \{\mathbf{v}_J &= \omega_{jJ}\}_{j\in\mathcal{B}_J}. \end{aligned} \tag{54}$$

Since the l.h.s. of (54) coincides with the l.h.s. of (51), the set of equations in (54) can be likewise written as [cf. (53)]

$$\mathbf{A}'\mathbf{v} = \omega'' \tag{55}$$

where now $\omega'' := [\{\omega_{j1}^T\}_{j\in\mathcal{B}_1},\ldots,\{\omega_{jJ}^T\}_{j\in\mathcal{B}_J}]^T$. Notice that ω'' is a permuted version of ω', since it can be obtained from ω' by replacing vector ω_{ji} in ω' with vector ω_{ij}. Hence, using a $2|\mathcal{E}|\times 2|\mathcal{E}|$ permutation matrix $\mathbf{E}$ and letting $\otimes$ denote the Kronecker product, ω'' can be related to ω' as

$$\omega'' = (\mathbf{E}\otimes\mathbf{I}_{p+1})\omega' \tag{56}$$

where $\mathbf{E} := [\{\mathbf{e}_{1i}\}_{i\in\mathcal{B}_1},\ldots,\{\mathbf{e}_{Ji}\}_{i\in\mathcal{B}_J}]$, and $\mathbf{e}_{ji}$ is a $2|\mathcal{E}|\times 1$ indicator vector given by

$$\mathbf{e}_{ji} := \left[\begin{array}{c} \mathbf{e}_1^{(ji)} \\ \vdots \\ \mathbf{e}_J^{(ji)} \end{array}\right]$$

with sub-blocks $\mathbf{e}_{i'}^{(ji)} := [\{\delta(j-j',i-i')\}_{j'\in\mathcal{B}_{i'}}]^T$, and $\delta(\cdot,\cdot)$ denoting Kronecker's delta function. Intuitively, $\mathbf{e}_{ji}$ identifies with a one the position where ω_{ji} in ω' is to be re-allocated in ω''.

Substituting (56) into (55) yields

$$\mathbf{A}'\mathbf{v} = (\mathbf{E}\otimes\mathbf{I}_{p+1})\omega'. \tag{57}$$

Concatenating (53) and (57) one arrives at

$$\mathbf{A}\mathbf{v} = \mathbf{E}'\omega' \tag{58}$$

where

$$\mathbf{A} := \left[\begin{array}{c} \mathbf{A}' \\ \mathbf{A}' \end{array}\right] \quad \text{and} \quad \mathbf{E}' := \left[\begin{array}{c} \mathbf{I}_{2(p+1)|\mathcal{E}|} \\ \mathbf{E}\otimes\mathbf{I}_{p+1} \end{array}\right]. \tag{59}$$

Using (58), problem (7) can be re-written as

$$\begin{array}{ll} \min\limits_{\mathbf{v},\omega,\{\xi_j\}} & \frac{1}{2}\sum\limits_{j=1}^{J}\mathbf{v}_j^T(\mathbf{I}_{p+1}-\Pi_{p+1})\mathbf{v}_j + JC\sum\limits_{j=1}^{J}\mathbf{1}_j^T\xi_j \\ \text{s.t.} & \mathbf{Y}_j\mathbf{X}_j\mathbf{v}_j \succeq \mathbf{1}_j - \xi_j \qquad \forall j\in\mathcal{J} \\ & \xi_j \succeq \mathbf{0}_j \qquad \forall j\in\mathcal{J} \\ & \mathbf{A}\mathbf{v} = \mathbf{E}'\omega'. \end{array} \tag{60}$$

It is known that the slack variables $\{\xi_j\}$ can be eliminated by introducing the hinge loss function $\ell(y,[\mathbf{x}^T,1]\mathbf{v}) := \max\{0,1-y[\mathbf{x}^T,1]\mathbf{v}\}$ (Schölkopf and Smola, 2002), which reduces (60) to its equivalent form

$$\begin{array}{ll} \min\limits_{\mathbf{v},\omega} & \frac{1}{2}\sum\limits_{j=1}^{J}\mathbf{v}_j^T(\mathbf{I}_{p+1}-\Pi_{p+1})\mathbf{v}_j + JC\sum\limits_{j=1}^{J}\sum\limits_{n=1}^{N_j}\ell(y_{jn},[\mathbf{x}_{jn}^T,1]\mathbf{v}_j) \\ \text{s.t.} & \mathbf{A}\mathbf{v} = \mathbf{E}'\omega'. \end{array} \tag{61}$$

Comparing the latter with (45), it follows readily that (61), which is equivalent to (7), belongs to the ADMoM-solvable class since (61) is subsumed by (45) with the special choices

$$\begin{aligned} F_1(\mathbf{v}) &:= \frac{1}{2}\sum_{j=1}^{J}\mathbf{v}_j^T(\mathbf{I}_{p+1}-\Pi_{p+1})\mathbf{v}_j + JC\sum_{j=1}^{J}\sum_{n=1}^{N_j}\ell(y_{jn},[\mathbf{x}_{jn}^T,1]\mathbf{v}_j), \\ F_2(\omega) &:= 0, \\ \mathcal{P}_1 &:= \mathbb{R}^{(p+1)J}, \\ \mathcal{P}_2 &:= \{\omega\in\mathbb{R}^{4(p+1)|\mathcal{E}|}\,|\,\omega=\mathbf{E}'\omega' \text{ for some } \omega'\in\mathbb{R}^{2(p+1)|\mathcal{E}|}\} \end{aligned} \tag{62}$$

where $\omega := \mathbf{E}'\omega'$ is now placed in the constraint set $\mathcal{P}_2$.

So far it has been proved that the problem in (7) can be cast as (45). The ADMoM iterations for (45) are (47)-(49), with corresponding iterates $\mathbf{v}(t)$, $\omega(t)$ and $\alpha(t)$. Given the constraint set $\mathcal{P}_2$ in (62), for every iterate $\omega(t)$ there exists a unique $\omega'(t)$ satisfying $\omega(t)=\mathbf{E}'\omega'(t)$, due to the fact that $\mathbf{E}'$ in (59) is full column rank. Hence, $\omega(t)$ can be replaced by $\mathbf{E}'\omega'(t)$ in iterations (47)-(49). Iteration (9) then follows by re-introducing the slack variables $\{\xi_j(t)\}$. Iteration (10) follows because now ω' is unconstrained. Finally, iterations (11) and (12) follow from (49) by splitting $\alpha(t)$ into appropriate sub-groups of vectors $\{\alpha_{ji1}(t)\}$ and $\{\alpha_{ji2}(t)\}$, respectively.

Appendix D. Proof of Lemma 3

The goal of this appendix is to show that iterations (9)-(12) reduce to (13)-(14). To start, notice that the cost in (10) is linear-quadratic w.r.t. ω_{ji}. Thus, setting the derivative of $\mathcal{L}$ w.r.t. ω_{ji} equal to zero, $\omega_{ji}(t+1)$ can be found in closed form as

$$\omega_{ji}(t+1)=\frac{1}{2\eta}(\alpha_{ji1}(t)-\alpha_{ji2}(t))+\frac{1}{2}(\mathbf{v}_j(t+1)+\mathbf{v}_i(t+1)). \tag{63}$$

Substituting (63) into (11) and (12), yields

$$\alpha_{ji1}(t+1) = \frac{1}{2}(\alpha_{ji1}(t)+\alpha_{ji2}(t))+\frac{\eta}{2}(\mathbf{v}_j(t+1)-\mathbf{v}_i(t+1)), \tag{64}$$

$$\alpha_{ji2}(t+1) = \frac{1}{2}(\alpha_{ji1}(t)+\alpha_{ji2}(t))+\frac{\eta}{2}(\mathbf{v}_j(t+1)-\mathbf{v}_i(t+1)). \tag{65}$$

Suppose now that $\alpha_{ji1}(t)$ and $\alpha_{ji2}(t)$ are initialized identically to zero at every node j; that is, $\alpha_{ji1}(0)=\alpha_{ji2}(0)=\mathbf{0}_{(p+1)\times 1}\ \forall j\in\mathcal{J}$ and $\forall i\in\mathcal{B}_j$. From (64) and (65), it follows easily that $\alpha_{ji1}(1)=\alpha_{ji2}(1)$. Similarly, if $\alpha_{ji1}(t-1)=\alpha_{ji2}(t-1)$, then by induction $\alpha_{ji1}(t)=\alpha_{ji2}(t)$. Thus, only one set of multipliers, say $\{\alpha_{ji1}\}$, needs to be stored and updated per node j.

Upon substituting $\omega_{ji}(t+1)=(1/2)(\mathbf{v}_j(t+1)+\mathbf{v}_i(t+1))$ into the objective function of (9) and using $\alpha_{ji1}(t)=\alpha_{ji2}(t)$, one obtains

$$\begin{aligned} \mathcal{L}'(\{\mathbf{v}_j\},\{\xi_j\},\{\mathbf{v}_j(t)\},\{\alpha_{ji1}(t)\}) &= \frac{1}{2}\sum_{j=1}^{J}\mathbf{v}_j^T(\mathbf{I}_{p+1}-\Pi_{p+1})\mathbf{v}_j + JC\sum_{j=1}^{J}\mathbf{1}_j^T\xi_j \\ &+\sum_{j=1}\sum_{i\in\mathcal{B}_j}\alpha_{ji1}^T(t)(\mathbf{v}_j-\mathbf{v}_i)+\frac{\eta}{2}\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\left\|\mathbf{v}_j-\frac{1}{2}[\mathbf{v}_j(t)+\mathbf{v}_i(t)]\right\|^2+\frac{\eta}{2}\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\left\|\mathbf{v}_i-\frac{1}{2}[\mathbf{v}_j(t)+\mathbf{v}_i(t)]\right\|^2. \end{aligned} \tag{66}$$

The first double sum in the right hand side (r.h.s.) of (66) can be rewritten as

$$\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\alpha_{ji1}^T(t)(\mathbf{v}_j-\mathbf{v}_i)=\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\mathbf{v}_j^T(\alpha_{ji1}(t)-\alpha_{ij1}(t))$$
$$=2\sum_{j=1}^{J}\mathbf{v}_j^T\sum_{i\in\mathcal{B}_j}\alpha_{ji1}(t) \tag{67}$$

where the first equality follows because $\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\alpha_{ji1}^T(t)\mathbf{v}_i=\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\alpha_{ij1}^T(t)\mathbf{v}_j$. Intuitively, the r.h.s. computes the sum by fixing a node j and adding the inner products of $\mathbf{v}_j$ with the incoming Lagrange multipliers $\alpha_{ij1}(t)$; while the left hand side performs the same sum by fixing a node j and adding the inner products of outgoing Lagrange multipliers $\alpha_{ji1}(t)$ and the corresponding $\mathbf{v}_i$ neighbors. The second equality on (67) holds because the all-zero initialization of the multipliers implies that $\alpha_{ji1}(t)=-\alpha_{ij1}(t)\ \forall t$ [cf. from (64)-(65)]. Likewise, the second and third double sums in the r.h.s. of (66) can be simplified to

$$\frac{\eta}{2}\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\left[\left\|\mathbf{v}_j-\frac{1}{2}[\mathbf{v}_j(t)+\mathbf{v}_i(t)]\right\|^2+\left\|\mathbf{v}_i-\frac{1}{2}[\mathbf{v}_j(t)+\mathbf{v}_i(t)]\right\|^2\right]=\eta\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\left\|\mathbf{v}_j-\frac{1}{2}[\mathbf{v}_j(t)+\mathbf{v}_i(t)]\right\|^2. \tag{68}$$

Lemma 3 follows after substituting (67) and (68) into (66), and defining $\alpha_j(t):=\sum_{i\in\mathcal{B}_j}\alpha_{ji1}(t)$.

Appendix E. Proof of Proposition 1

Letting $\lambda_j:=[\lambda_{j1},\ldots,\lambda_{jN_j}]^T$ and $\mu_j:=[\mu_{j1},\ldots,\mu_{jN_j}]^T$ denote Lagrange multipliers associated with the constraints $\mathbf{Y}_j\mathbf{X}_j\mathbf{v}_j\succeq\mathbf{1}_j-\xi_j$ and $\xi_j\succeq\mathbf{0}_j$, respectively, the Lagrangian corresponding to (13) is given by

$$\begin{aligned}&\mathcal{L}''(\{\mathbf{v}_j\},\{\xi_j\},\{\lambda_j\},\{\mu_j\},\{\mathbf{v}_j(t)\},\{\alpha_j(t)\})\\&\quad=\frac{1}{2}\sum_{j=1}^{J}\mathbf{v}_j^T(\mathbf{I}_{p+1}-\Pi_{p+1})\mathbf{v}_j+JC\sum_{j=1}^{J}\mathbf{1}_j^T\xi_j-\sum_{j=1}^{J}\lambda_j^T(\mathbf{Y}_j\mathbf{X}_j\mathbf{v}_j-\mathbf{1}_j+\xi_j)\\&\quad-\sum_{j=1}^{J}\mu_j^T\xi_j+2\sum_{j=1}^{J}\alpha_j^T(t)\mathbf{v}_j+\eta\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\left\|\mathbf{v}_j-\frac{1}{2}[\mathbf{v}_j(t)+\mathbf{v}_i(t)]\right\|^2.\end{aligned} \tag{69}$$

The KKT conditions yield per iteration the primal and dual variables in (69) as follows

$$\mathbf{v}_j(t+1) = \mathbf{U}_j^{-1}\left(\mathbf{X}_j^T\mathbf{Y}_j\lambda_j(t+1)-2\alpha_j(t)+\eta\sum_{i\in\mathcal{B}_j}(\mathbf{v}_j(t)+\mathbf{v}_i(t))\right), \tag{70}$$

$$\mathbf{0}_j = JC\mathbf{1}_j-\lambda_j-\mu_j \tag{71}$$

where $\lambda_j(t+1)$ is the optimal Lagrange multiplier after iteration $t+1$, and the inverse of $\mathbf{U}_j:=(1+2\eta|\mathcal{B}_j|)\mathbf{I}_{p+1}-\Pi_{p+1}$ always exists.

The KKT conditions also require $\lambda_j\succeq\mathbf{0}_j$ and $\mu_j\succeq\mathbf{0}_j$, which allows (71) to be replaced by $\mathbf{0}_j\preceq\lambda_j\preceq JC\mathbf{1}_j$. To carry out the iteration (70) at every node, the optimal values $\lambda_j(t+1)$ of the

Lagrange multipliers λ_j are found by solving the Lagrange dual problem associated with (69). The pertinent dual function is given by

$$\mathcal{L}_\lambda(\{\lambda_j\}) = \sum_{j=1}^{J} -\frac{1}{2}\lambda_j^T \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{X}_j^T \mathbf{Y}_j \lambda_j + \left(\mathbf{1}_j - \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{f}_j(t)\right)^T \lambda_j \tag{72}$$

where $\mathbf{f}_j(t) := 2\alpha_j(t) - \eta \sum_{i\in\mathcal{B}_j}[\mathbf{v}_j(t) + \mathbf{v}_i(t)]$. Note that the Lagrange multipliers $\{\mu_j\}$ are not present in $\mathcal{L}_\lambda$. From (72), 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, $\lambda_j(t+1)$ is given by

$$\lambda_j(t+1) = \arg \max_{\lambda_j:\, \mathbf{0}_j \preceq \lambda_j \preceq JC\mathbf{1}_j} -\frac{1}{2}\lambda_j^T \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{X}_j^T \mathbf{Y}_j \lambda_j + \left(\mathbf{1}_j - \mathbf{Y}_j \mathbf{X}_j \mathbf{U}_j^{-1} \mathbf{f}_j(t)\right)^T \lambda_j \tag{73}$$

The dual variable update in (73) and the primal variable update in (70) coming from the KKT optimality, are precisely iterations (16) and (17) of Proposition 1, which together with (18) correspond to iterations (13) and (14). Lemma 3 shows that (13) and (14) are equivalent to (9)-(12). Lemma 2 establishes that (9)-(12) in turn correspond to the ADMoM iterations (47)-(49) of Appendix A. As stated in Proposition 4 in Appendix A, convergence of the ADMoM iterations (47)-(49) is guaranteed so long as: (i) $\mathcal{P}_1$ is bounded; or, (ii) $\mathbf{A}^T\mathbf{A}$ is nonsingular. Since for the problem at hand matrix $\mathbf{A}$ in (59) satisfies condition (ii), the iterates for (16)-(18) in Proposition 1 converge to the optimal solution of (7) for any $\eta > 0$.

Appendix F. Proof of Theorem 1

For simplicity, this theorem will be proved for purely linear discriminant functions $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. Consider the reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ of functions $g(\mathbf{x})$ with corresponding positive semi-definite kernel $K : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$, defined as

$$\mathcal{H} := \overline{\left\{ g(\cdot) = \sum_{n=1}^{N} \gamma_n K(\cdot, \mathbf{x}_n) :\ N \in \mathbb{N},\ \gamma_1, \ldots, \gamma_N \in \mathbb{R},\ \mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathcal{X} \right\}}$$

with $\overline{\mathcal{A}}$ denotes the completion of the set $\mathcal{A}$.

The parameter optimization problem (23) can be written in terms of the Hinge loss function $\ell(y, g(\mathbf{x})) := \max\{0, 1 - yg(\mathbf{x})\}$, and the RKHS-induced norm $\|g\|_{\mathcal{H}}^2$ as a regularized optimization problem to obtain, (see, e.g., Schölkopf and Smola, 2002)

$$\begin{aligned} \min_{\{g_j\in\mathcal{H}\}} \quad & \frac{1}{2}\sum_{j=1}^{J} \|g_j\|_{\mathcal{H}}^2 + JC \sum_{j=1}^{J}\sum_{n=1}^{N_j} \ell(y_{jn}, g_j(\mathbf{x}_{jn})) \\ \text{s.t. } & g_j(\chi_l) = g_i(\chi_l) \quad \forall j \in \mathcal{J}, i \in \mathcal{B}_j, l = 1, \ldots, L. \end{aligned} \tag{74}$$

Given the optimal Lagrange multipliers ς_{jil}^* for the constraints $\{g_j(\chi_l) = g_i(\chi_l)\}$, the solution $\{g_j^*\}$ of (74) can be obtained from its Lagrangian as

$$\{g_j^*\} = \arg \min_{\{g_j\in\mathcal{H}\}} \frac{1}{2}\sum_{j=1}^{J} \|g_j\|_{\mathcal{H}}^2 + JC \sum_{j=1}^{J}\sum_{n=1}^{N_j} \ell(y_{jn}, g_j(\mathbf{x}_{jn})) + \sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\sum_{l=1}^{L} \varsigma_{jil}^*(g_j(\chi_l) - g_i(\chi_l)). \tag{75}$$

Arguing as in (67), the last term in (75) can be written as

$$\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\sum_{l=1}^{L}\varsigma_{jil}^{*}(g_j(\chi_l)-g_i(\chi_l))=\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\sum_{l=1}^{L}(\varsigma_{jil}^{*}-\varsigma_{ijl}^{*})g_j(\chi_l)$$

thus rendering the cost in (75) separable across j. Hence, each g_j^* can be obtained per node as

$$g_j^{*}=\arg\min_{g_j\in\mathcal{H}}V(\mathbf{x}_{j1},\ldots,\mathbf{x}_{jN_j},y_{j1},\ldots,y_{jN_j},\{\chi_l\},g_j)+\frac{1}{2}\left\|g_j\right\|_{\mathcal{H}}^{2} \tag{76}$$

where $V(\mathbf{x}_{j1},\ldots,\mathbf{x}_{jN_j},y_{j1},\ldots,y_{jN_j},\{\chi_l\},g_j):=JC\sum_{n=1}^{N_j}\ell(y_{jn},g_j(\mathbf{x}_{jn}))+\sum_{i\in\mathcal{B}_j}\sum_{l=1}^{L}(\varsigma_{jil}^{*}-\varsigma_{ijl}^{*})g_j(\chi_l)$. Applying the Representer Theorem to (76) as in Wahba (1990) and Schölkopf and Smola (2002) one readily arrives at

$$g_j^{*}(\mathbf{x})=\sum_{n=1}^{N_j}a_{jn}^{*}K(\mathbf{x},\mathbf{x}_{jn})+\sum_{l=1}^{L}c_{jl}^{*}K(\mathbf{x},\chi_l). \tag{77}$$

Appendix G. Proof of Proposition 2

Recall that $\mu_j:=[\mu_{j1},\ldots,\mu_{jN_j}]^T$ denotes the Lagrange multiplier associated with the constraint $\xi_j\succeq\mathbf{0}_j$ (cf. Appendix E). The Lagrangian corresponding to (25) is given by

$$\begin{aligned}
&\mathcal{L}''(\{\mathbf{w}_j\},\{b_j\},\{\xi_j\},\{\lambda_j\},\{\mu_j\},\{\mathbf{w}_j(t)\},\{b_j(t)\},\{\alpha_j(t)\},\{\beta_j(t)\})\\
&\quad=\frac{1}{2}\sum_{j=1}^{J}\|\mathbf{w}_j\|^2+JC\sum_{j=1}^{J}\mathbf{1}_j^T\xi_j-\sum_{j=1}^{J}\lambda_j^T(\mathbf{Y}_j\Phi(\mathbf{X}_j)\mathbf{w}_j-\mathbf{1}_jb_j+\xi_j)\\
&\quad-\sum_{j=1}^{J}\mu_j^T\xi_j+2\sum_{j=1}^{J}\alpha_j^T(t)\mathbf{G}\mathbf{w}_j+2\sum_{j=1}^{J}\beta_j(t)b_j\\
&\quad+\eta\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\left\|\mathbf{G}\left[\mathbf{w}_j-\frac{1}{2}(\mathbf{w}_j(t)+\mathbf{w}_i(t))\right]\right\|^2+\eta\sum_{j=1}^{J}\sum_{i\in\mathcal{B}_j}\left\|b_j-\frac{1}{2}[b_j(t)+b_i(t)]\right\|^2.
\end{aligned}$$

From the KKT conditions for (25) it follows that

$$\mathbf{w}_j(t+1)=\widetilde{\mathbf{U}}_j^{-1}\left\{\Phi^T(\mathbf{X}_j)\mathbf{Y}_j\lambda_j(t+1)-\mathbf{G}^T\left[2\alpha_j(t)-\eta\sum_{i\in\mathcal{B}_j}\mathbf{G}(\mathbf{w}_j(t)+\mathbf{w}_i(t))\right]\right\}, \tag{78}$$

$$b_j(t+1)=\frac{1}{2\eta|\mathcal{B}_j|}\left[\mathbf{1}_j^T\mathbf{Y}_j\lambda_j(t+1)-2\beta_j(t)+\eta\sum_{i\in\mathcal{B}_j}(\beta_j(t)+\beta_i(t))\right] \tag{79}$$

where $\lambda_j(t+1)$ is the optimal Lagrange multiplier at iteration $t+1$, and $\widetilde{\mathbf{U}}_j:=\mathbf{I}_P+2\eta|\mathcal{B}_j|\mathbf{G}^T\mathbf{G}$. Using the Sherman-Morrison-Woodbury formula (Golub and Van Loan, 1996)

$$\widetilde{\mathbf{U}}_j^{-1}=\mathbf{I}_P-2\eta|\mathcal{B}_j|\mathbf{G}^T(\mathbf{I}_L+2\eta|\mathcal{B}_j|\mathbf{G}\mathbf{G}^T)^{-1}\mathbf{G}. \tag{80}$$

Substituting (80) into (78), left-multiplying by $\phi^T(\mathbf{x})$, and recalling that $\phi^T(\mathbf{x})\phi^T(\mathbf{x}')=K(\mathbf{x},\mathbf{x}')$, yields

$$\begin{aligned}
\phi^T(\mathbf{x})\mathbf{w}_j(t+1)=&\left(\mathbf{k}^T(\mathbf{x},\mathbf{X}_j)-2\eta|\mathcal{B}_j|\mathbf{k}^T(\mathbf{x},\Gamma)\widetilde{\mathbf{U}}_j^{-1}\mathbf{K}(\Gamma,\mathbf{X}_j)\right)\mathbf{Y}_j\lambda_j(t+1)\\
&-\left(\mathbf{k}^T(\mathbf{x},\Gamma)-2\eta|\mathcal{B}_j|\mathbf{k}^T(\mathbf{x},\Gamma)\widetilde{\mathbf{U}}_j^{-1}\mathbf{K}(\Gamma,\Gamma)\right)\widetilde{\mathbf{f}}_j(t+1)
\end{aligned} \tag{81}$$

where the entries of the kernel vector are $[\mathbf{k}(\mathbf{x},\mathbf{X}_j)]_n := K(\mathbf{x},\mathbf{x}_{jn})$ and $[\mathbf{k}(\mathbf{x},\Gamma)]_l := K(\mathbf{x},\chi_l)$.

Note that $g_j^{(t)}(\mathbf{x})$ in (31) follows from (77) and can be written as $g_j^{(t)}(\mathbf{x}) = \phi^T(\mathbf{x})\mathbf{w}_j(t) + b_j(t)$. Grouping terms in (81) that right-multiply $\mathbf{k}^T(\mathbf{x},\mathbf{X}_j)$ and those that right-multiply $\mathbf{k}^T(\mathbf{x},\Gamma)$, yields $\mathbf{a}_j(t)$ as in (32) and $\mathbf{c}_j(t)$ as in (33), respectively. Finally, $b_j(t)$ in (34) is given by (79).

Appendix H. Proof of Proposition 3

To obtain iteration (35), consider first the dual problem for (25), that is

$$\begin{aligned}\lambda_j(t+1) \;=\; \underset{\lambda_j:\,\mathbf{0}_j\preceq\lambda_j\preceq JC\mathbf{1}_j}{\arg\max} & -\frac{1}{2}\lambda_j^T\mathbf{Y}_j\left(\Phi(\mathbf{X}_j)\widetilde{\mathbf{U}}_j^{-1}\Phi^T(\mathbf{X}_j)+\frac{\mathbf{1}\mathbf{1}^T}{2\eta|\mathcal{B}_j|}\right)\mathbf{Y}_j\lambda_j \\ & +\left(\mathbf{1}_j-\mathbf{Y}_j\Phi(\mathbf{X}_j)\widetilde{\mathbf{U}}_j^{-1}\mathbf{G}^T\widetilde{\mathbf{f}}_j(t)-\frac{h_j(t)\mathbf{1}^T}{2\eta|\mathcal{B}_j|}\right)^T\lambda_j \end{aligned} \tag{82}$$

where $\widetilde{\mathbf{U}}_j^{-1}$ is given by (80), and $\widetilde{\mathbf{f}}_j(t)$ as in Proposition 2. Using (80), the term $\Phi(\mathbf{X}_j)\widetilde{\mathbf{U}}_j^{-1}\Phi^T(\mathbf{X}_j)$ can be written in terms of inner products, and summarized via kernels as

$$\Phi(\mathbf{X}_j)\widetilde{\mathbf{U}}_j^{-1}\Phi^T(\mathbf{X}_j) = \mathbf{K}(\mathbf{X}_j,\mathbf{X}_j) - 2\eta|\mathcal{B}_j|\mathbf{K}(\mathbf{X}_j,\Gamma)\widetilde{\mathbf{U}}_j^{-1}\mathbf{K}(\Gamma,\mathbf{X}_j). \tag{83}$$

Likewise, the term $\Phi(\mathbf{X}_j)\widetilde{\mathbf{U}}_j^{-1}\mathbf{G}^T\widetilde{\mathbf{f}}_j(t)$ can be expressed as

$$\Phi(\mathbf{X}_j)\widetilde{\mathbf{U}}_j^{-1}\mathbf{G}^T\widetilde{\mathbf{f}}_j(t) = \Big(\mathbf{K}(\mathbf{X}_j,\Gamma) - 2\eta|\mathcal{B}_j|\mathbf{K}(\mathbf{X}_j,\Gamma)\widetilde{\mathbf{U}}_j^{-1}\mathbf{K}(\Gamma,\Gamma)\Big)\widetilde{\mathbf{f}}_j(t). \tag{84}$$

Plugging (83) and (84) into (82), yields (35).

To obtain iteration (36), left-multiply $\mathbf{w}_j(t+1)$ in (78) by $\mathbf{G}$ to arrive at

$$\widetilde{\mathbf{w}}_j(t+1) \;=\; \mathbf{G}\widetilde{\mathbf{U}}_j^{-1}\Phi^T(\mathbf{X}_j)\mathbf{Y}_j\lambda_j - \mathbf{G}\widetilde{\mathbf{U}}_j^{-1}\mathbf{G}^T\left[2\alpha_j(t) - \eta\sum_{i\in\mathcal{B}_j}(\widetilde{\mathbf{w}}_j(t)+\widetilde{\mathbf{w}}_i(t))\right]. \tag{85}$$

The terms $\mathbf{G}\widetilde{\mathbf{U}}_j^{-1}\Phi^T(\mathbf{X}_j)$ and $\mathbf{G}\widetilde{\mathbf{U}}_j^{-1}\mathbf{G}^T$ can be respectively written as

$$\mathbf{G}\widetilde{\mathbf{U}}_j^{-1}\Phi^T(\mathbf{X}_j) = \mathbf{K}(\Gamma,\Gamma) - 2\eta|\mathcal{B}_j|\mathbf{K}(\Gamma,\Gamma)\widetilde{\mathbf{U}}_j^{-1}\mathbf{K}(\Gamma,\mathbf{X}_j) \tag{86}$$

and

$$\mathbf{G}\widetilde{\mathbf{U}}_j^{-1}\mathbf{G}^T = \mathbf{K}(\Gamma,\Gamma) - 2\eta|\mathcal{B}_j|\mathbf{K}(\Gamma,\Gamma)\widetilde{\mathbf{U}}_j^{-1}\mathbf{K}(\Gamma,\Gamma)\,. \tag{87}$$

Substituting (86) and (87) into (85), yields (36), and completes the proof of the proposition.

References

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

A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd ed., 1999.

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

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, Mar. 2005.

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.

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.

K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems*, 2008.

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.

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

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.

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

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

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

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

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

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

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.

T. Hastie, R. Tishbiranie, and J. Friedman. *The Elements of Statistical Learning*. Spinger, 2nd edition, 2009.

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, Jun. 29 - Jul. 2, 2000.

K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Comput. Netw.*, 43(4):499–518, 2003.

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

F. F. Li. Estimation of intelligibility from received arbitrary speech signals with support vector machine. In *Proc. of International Conference on Machine Learning and Cybernetics*, volume 6, pages 3755–3760, Guangzhou, China, Aug. 2005.

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

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.

U. Markowska-Kaczmar and P. Kubacki. Support vector machines in handwritten digits classification. In *Proc. of 5th International Conference on Intelligent Systems Design and Applications*, pages 406–411, Wroclaw, Poland, Sep. 8-11, 2005.

A. Navia-Vazquez, D. Gutierrez-Gonzalez, E. Parrado-Hernandez, and J. J. Navarro-Abellan. Distributed support vector machines. *IEEE Tran. on Neural Networks*, 17(4):1091–1097, Jul. 2006.

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

J. B. Predd, S. R. Kulkarni, and H. V. Poor. Distributed kernel regression: An algorithm for training collaboratively. *IEEE Trans. on Information Theory*, 55(4):1856–1871, Apr. 2009.

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

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

G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conf. Series in Applied Mathematics*. SIAM, Philadelphia, PA, USA, 1990.

J. Weston, A. Elisseeff, G. BakIr, and F. Sinz. The spider machine learning toolbox, 2006. URL `http://www.kyb.tuebingen.mpg.de/bs/people/spider/`.

H. Zhu, G. B. Giannakis, and A. Cano. Distributed in-network channel decoding. *IEEE Trans. on Signal Processing*, 57(10):3970–3983, Oct. 2009.