A DISTRIBUTED CONSTRAINED-FORM SUPPORT VECTOR MACHINE

François D. Côté, Ioannis N. Psaromiligkos, and Warren J. Gross

Department of Electrical and Computer Engineering, McGill University, Montreal, Canada

ABSTRACT

Despite the importance of distributed learning, few fully distributed support vector machines exist. In this paper, not only do we provide a fully distributed nonlinear SVM; we propose the first distributed constrained-form SVM. In the fully distributed context, a dataset is distributed among networked agents that cannot divulge their data, let alone centralize the data, and can only communicate with their neighbors in the network. Our strategy is based on two algorithms: the Douglas-Rachford algorithm and the projection-gradient method. We validate our approach by demonstrating through simulations that it can train a classifier that agrees closely with the centralized solution.

***Index Terms*—** Optimization, classification, kernel method.

1. INTRODUCTION

Machine learning provides the tools for assimilating patterns in data. Originally, these tools were designed to work on a single machine, with a single dataset. But the reality today is that datasets are often made up of several subsets, distributed among networked agents, and centralizing the data is impossible. Distributed learning, therefore, has become an important field of research [1].

Although there is a vast body of work on distributed algorithms, there is a glaring gap in it: *fully* distributed support vector machines are rare. In fact, only three exist [2]–[4]. This scarcity is not due to disinterest—the SVM is the staple of machine learning—but to technical challenges. Training an SVM, particularly a nonlinear one, in a fully distributed way, is a "notoriously complex problem" [4].

In this paper, our goal is to develop a fully distributed nonlinear constrained-form SVM. Training a classifier is usually done by solving a penalized problem, but it is also possible to solve a constrained problem. Both problems are equivalent, yet there is strong motivation for solving the constrained problem: the constraint parameter is easier to determine than the penalization parameter [5].

Our objective is not to compare the constrained form to the penalized form. The question of constrained vs. penalized form is well documented [6], [7]. The penalized form is usually chosen, as it yields an easier problem, but at the cost of an expensive cross validation for determining the parameter. In contrast, the constraint parameter, which corresponds to the noise level of the data, may already be known in practice. Therefore, an approach that solves the harder, constrained problem, may be at a significant advantage.

1.1. Related work and contributions

To the best of our knowledge, no prior work exists on a distributed constrained-form SVM, only on the penalized form. The state of the art is due to Forero et al. [2]. Their SVM is nonlinear. Other algorithms are due to Wang et al. [3] and Scardapane et al. [4]. The classifier of Wang et al. is nonlinear, but their algorithm must be rerun whenever a new object enters the classifier. The approach of Scardapane et al., while semisupervised, remains for linear classification. Other distributed algorithms exist, but they are not fully distributed: they require a central processor or all-to-all communications. A survey of these approaches is found in all three sources cited in this paragraph.

Our approach differs from the approach of Forero et al. in several respects. In addition to considering the constrained form of the SVM, we provide an algorithm in which no step is left as an optimization problem; we allow for overlap in the distributed data; we use a more general dimensionality reduction technique by allowing for different sets of arbitrary objects across the network; and we depart from the customary alternating direction method of multipliers (ADMM).

Our contributions can be summarized as follows:

- We provide a distributed constrained-form SVM, Algorithm 1.
- We kernelize our approach. See Algorithm 2.
- We propose an efficient implementation of our kernel method. This is Algorithm 3.
- The implementation is based on an epigraphical projection, for which we offer a closed-form expression. See Proposition 1.

In this paper, our focus is on presenting the development of our approach. For the sake of brevity, we omit technical proofs.

1.2. Problem statement

Consider a network of m agents, each interested in learning how to tell apart two kinds of objects. Suppose that these objects belong to a set $\mathcal{X}$ and a label of ± 1 indicates to which class an object belongs. Together, the agents have amassed a dataset of ℓ labeled examples,

$$(x_1, y_1),\ \ldots, (x_\ell, y_\ell).$$

This dataset is known in the network only collectively, as a union of possibly overlapping subsets. Even if each agent only knows some of the examples, it must train a classifier as if it knew all of them.

Training an SVM consists of determining a decision function. This function is parameterized by a vector w in a real Hilbert space $\mathcal{H}$ and a number b in $\mathbb{R}$. It is defined through the inner product in $\mathcal{H}$ and a fixed mapping $\phi\colon \mathcal{X} \to \mathcal{H}$ by

$$h(x; w, b) = \langle \phi(x), w \rangle + b, \quad x \in \mathcal{X}, \tag{1}$$

and its sign predicts the label of an object x. For the constrained-form SVM, the task is to find h such that w has the smallest possible norm (induced by the inner product) and the classification score, given by the value of h, has a total hinge loss (over the dataset) of at most ϵ. In other words, finding h involves solving the following problem, equivalent to the soft-margin formulation of the SVM [8]:

$$\min_{(w,b)\in\mathcal{H}\times\mathbb{R}} \ \|w\| \quad \text{s.t.} \quad \sum_{k=1}^{\ell} \max\{0, 1 - y_k h(x_k; w, b)\} \leq \epsilon. \tag{2}$$

Thus, the objective is

$$\text{find } h, \text{ as defined by (1), such that } (w, b) \text{ solves (2).} \tag{3}$$

To expound on the context, we make some assumptions:

1. Problem (2) has a solution.
2. The network of agents is connected.
3. There exists a (w, b) such that the inequality in (2) holds strictly.
4. Each agent knows m, ϵ, and its number of neighbors, as well as other parameters needed for the algorithms.
5. For each training example that each agent knows, the agent knows how many agents know the example.

By communicating only with its neighbors in the network, and never sharing its part of the data, every agent must arrive at the same solution for h. That is our objective: to develop a fully distributed algorithm for solving (3).

2. DEVELOPMENT

In this section, we present the development of our approach. We draw on our previous work on distributed optimization [9].

2.1. Data representation

Let us number the agents 1 to m and comment on Assumption 5. Because of this assumption, agent i can express its part of the data as a vector a_i in $\mathbb{R}^\ell$ and an operator $A_i : \mathbb{R}^\ell \to \mathcal{H}$. The vector is defined entrywise: its kth entry, $[a_i]_k$, is equal to y_k divided by the number of agents that know (x_k, y_k), if agent i itself knows the labeled example; and 0, otherwise. The operator is defined by

$$A_i u = \sum_{k=1}^{\ell} [a_i]_k [u]_k \phi(x_k), \quad u \in \mathbb{R}^\ell.$$

The significance of this data representation is that we can express the constraint in (2) in terms of a_i and the adjoint of A_i,

$$\sum_{k=1}^{\ell} \max\Big\{0, 1 - \Big[\sum_{i=1}^{m} (b a_i + A_i^* w)\Big]_k\Big\} \leq \epsilon, \tag{4}$$

evoking the data portrayal in [9, Sec. 1.1].

2.2. Constraint separation

Developing a distributed algorithm for solving (2) requires that we divide (4) among the agents. To do so, we assign auxiliary variables to the agents. Let $\mathcal{N}_i$ denote the set comprising agent i's neighbors. To each agent i, we assign, for each j in $\mathcal{N}_i$, a variable Δv_{ij} in $\mathbb{R}^\ell$. Let (w_i, b_i) be agent i's copy of (w, b). By using [9, Prop. 2] and recalling Assumption 2, we can prove that a necessary and sufficient condition for (4) to hold is that for each agent i,

$$\sum_{k=1}^{\ell} \max\Big\{0, \frac{1}{m} - \Big[b_i a_i + A_i^* w_i + \sum_{j \in \mathcal{N}_i} \Delta v_{ij}\Big]_k\Big\} \leq \frac{\epsilon}{m}, \tag{5}$$

and for every pair $\{i, j\}$ of neighboring agents, $\Delta v_{ij} = -\Delta v_{ji}$.

2.3. Problem reformulation

It is also useful to reformulate (2) in a way that accounts for the structure of the network. This time, we assign not only variables, but also functions to the agents. To each agent i, we assign a function $f_{1,i}$, and for each j in $\mathcal{N}_i$, a variable (w_{ij}, b_{ij}) in $\mathcal{H} \times \mathbb{R}$ and a function $f_{2,ij}$. The function $f_{1,i}$ takes a value of $+\infty$, unless (w_{ij}, b_{ij}) is the same, (w_i, b_i), for every j in $\mathcal{N}_i$, and (5) holds, in which case, the function takes a value of $\|w_i\|^2$. The function $f_{2,ij}$ takes a value of 0, if $(w_{ij}, b_{ij}) = (w_{ji}, b_{ji})$ and $\Delta v_{ij} = -\Delta v_{ji}$; and $+\infty$, otherwise.

Let $\mathcal{E}$ denote the set that comprises every unordered pair of agents that are neighbors. To solve (2), the agents can collaboratively solve (cf. [9, eq. (4)])

$$\min \sum_{i=1}^{m} f_{1,i} + \sum_{\{i,j\} \in \mathcal{E}} f_{2,ij} \quad \text{over all the variables.} \tag{6}$$

At the optimal values of all the variables, not only is (w_i, b_i) the same for every agent i; it also solves (2).

If $\mathcal{H}$ is a finite coordinate space, then the *distributed scaled Douglas-Rachford algorithm* [9, Algorithm 3] provides a distributed procedure for solving (6) as well as a guarantee on convergence. (This approach requires invertible matrices as parameters. For simplicity, we use identity matrices.) The result of applying this procedure to (6) is Algorithm 1.

Because of Assumption 3, we can show that [9, Prop. 1] applies, establishing the convergence of Algorithm 1.

2.4. Dimensionality reduction

In deriving Algorithm 1, we assumed that $\mathcal{H}$ was finite dimensional. To make the algorithm applicable to the infinite-dimensional case, we employ an approximation based on the dimensionality reduction strategy of Forero et al. [2].

Suppose that every pair $\{i, j\}$ of neighboring agents share ℓ_{ij} arbitrary objects, $\tilde{x}_{ij,k} \in \mathcal{X}$ for $k = 1, \ldots, \ell_{ij}$. For ease of notation,

Algorithm 1 Distributed constrained-form SVM

This algorithm provides each agent i with a sequence $(w_{i,0}, b_{i,0})$, $(w_{i,1}, b_{i,1}), \ldots$ that converges, when $\mathcal{H}$ is a finite coordinate space, to the solution to (2).

The first step is to initialize the network with two real numbers, $\gamma > 0$ and $\lambda \in (0, 2)$, and to let each agent i choose, for each j in $\mathcal{N}_i$, three elements, $z_{1,ij,0} \in \mathcal{H}$, $z_{2,ij,0} \in \mathbb{R}$, and $z_{3,ij,0} \in \mathbb{R}^\ell$. Then, in parallel with the other agents, each agent repeats the following steps. After n iterations, agent i

1. receives $z_{1,ji,n}$, $z_{2,ji,n}$, and $z_{3,ji,n}$ from each neighbor j;
2. finds the element (w_i, b_i) in $\mathcal{H} \times \mathbb{R}$ and the family $(\Delta v_{ij})_{j \in \mathcal{N}_i}$ of vectors in $\mathbb{R}^\ell$ that together minimize

$$\gamma \|w_i\|^2 + \frac{1}{2} \sum_{j \in \mathcal{N}_i} \big(\|w_i - z_{1,ji,n}\|^2 + \big(b_i - z_{2,ji,n}\big)^2 + \|\Delta v_{ij} + z_{3,ji,n}\|^2\big)$$

   subject to (5), and assigns the minimizers to $(w_{i,n}, b_{i,n})$ and $(\Delta v_{ij,n})_{j \in \mathcal{N}_i}$;
3. updates, for each j in $\mathcal{N}_i$, three elements,

$$z_{1,ij,n+1} = z_{1,ij,n} + \lambda\Big(w_{i,n} - \frac{1}{2}(z_{1,ij,n} + z_{1,ji,n})\Big), \tag{7}$$

$$z_{2,ij,n+1} = z_{2,ij,n} + \lambda\Big(b_{i,n} - \frac{1}{2}(z_{2,ij,n} + z_{2,ji,n})\Big), \text{ and} \tag{8}$$

$$z_{3,ij,n+1} = z_{3,ij,n} + \lambda\Big(\Delta v_{ij,n} - \frac{1}{2}(z_{3,ij,n} - z_{3,ji,n})\Big), \tag{9}$$

   and sends them to neighbor j.

let $\ell_{ij} = \ell_{ji}$ and $\tilde{x}_{ij,k} = \tilde{x}_{ji,k}$. By using these objects, agent i can form, for each j in $\mathcal{N}_i$, an operator $R_{ij}\colon \mathbb{R}^{\ell_{ij}} \to \mathcal{H}$ defined by

$$R_{ij}u = \sum_{k=1}^{\ell_{ij}} [u]_k \phi(\tilde{x}_{ij,k}), \quad u \in \mathbb{R}^{\ell_{ij}}.$$

This operator allows us to modify Algorithm 1 so that the agents only need to exchange finite-dimensional vectors.

We modify Algorithm 1 as follows: we make $z_{1,ij,n}$ a vector in $\mathbb{R}^{\ell_{ij}}$; we replace $\|w_i - z_{1,ji,n}\|^2$ in Step 2 with $\|R_{ij}^* w_i - z_{1,ji,n}\|^2$, modifying the objective function; and we replace (7) with

$$z_{1,ij,n+1} = z_{1,ij,n} + \lambda\Big(r_{ij,n} - \frac{1}{2}(z_{1,ij,n} + z_{1,ji,n})\Big), \qquad (10)$$

where $r_{ij,n} = R_{ij}^* w_{i,n}$.

With these modifications, we can prove that convergence is still a guarantee, but to an approximate solution, possibly different for each agent. As the number of common objects increases, the better the approximation—a behavior corroborated by Forero et al. [2].

2.5. Kernel evaluations

If $\mathcal{H}$ is infinite dimensional, then directly applying ϕ is impractical. In fact, the agents may not even know ϕ. Rather, they may know a function $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $K(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$ [10, Ch. 2]. If ϕ appears only in inner products, then K makes working in $\mathcal{H}$ simple. The most popular choice is the Gaussian kernel, for which

$$K(x_1, x_2) = \exp\big(-\|x_1 - x_2\|^2 / C\big), \quad C > 0.$$

In the remainder of this paper, we assume that the agents know a kernel K instead of ϕ.

In developing a kernel method for our approach, we will come across several matrices whose entries depend on K. Table 1 provides a description of these matrices.

2.6. Duality

Because the agents know K and not ϕ, they must determine h using K. Duality allows us to do this.

Let η_i be a vector in $\mathbb{R}^\ell$. If

$$[\eta_i]_k \geq 0 \quad \text{and} \quad [\eta_i]_k \geq \frac{1}{m} - \Big[b_i a_i + A_i^* w_i + \sum_{j \in \mathcal{N}_i} \Delta v_{ij}\Big]_k \quad \forall k$$

and

$$\sum_{k=1}^{\ell} [\eta_i]_k \leq \frac{\epsilon}{m},$$

then (5) must be true. To express the problem in Step 2 of Algorithm 1, which now includes the variable η_i, we use duality. Let us introduce dual variables μ_i and ξ_i in $\mathbb{R}^\ell$ and ν_i in $\mathbb{R}$ such that

$$[\mu_i]_k \geq 0 \quad \text{and} \quad [\xi_i]_k \geq 0 \quad \forall k \quad \text{and} \quad \nu_i \geq 0.$$

The Lagrangian is given by the modified objective function described in Section 2.4 with the following added to it:

$$2\gamma\Big(-\mu_i^T\Big(-\frac{1}{m}\mathbf{1} + b_i a_i + A_i^* w_i + \sum_{j \in \mathcal{N}_i} \Delta v_{ij} + \eta_i\Big) + \nu_i\Big(\mathbf{1}^T \eta_i - \frac{\epsilon}{m}\Big) - \xi_i^T \eta_i\Big),$$

where the superscript T denotes the transpose, $\mathbf{1}$ indicates the vector of all ones, and the factor of 2γ is for mathematical convenience.

Matrix	Equal to	Size	Entry (k, k')
G_i	$A_i^* A_i$	$\ell \times \ell$	$[a_i]_k [a_i]_{k'} K(x_k, x_{k'})$
M_{ij}	$R_{ij}^* A_i$	$\ell_{ij} \times \ell$	$[a_i]_{k'} K(\tilde{x}_{ij,k}, x_{k'})$
$\tilde{G}_{ijj'}$	$R_{ij}^* R_{ij'}$	$\ell_{ij} \times \ell_{ij'}$	$K(\tilde{x}_{ij,k}, \tilde{x}_{ij',k'})$

Block matrix	Description
$\tilde{M}_i$	Blocks indexed by $\mathcal{N}_i \times \mathcal{N}_i$. Block (j, j') is $\tilde{G}_{ijj'}$.
$\tilde{M}_{ij}$	This is block column j of $\tilde{M}_i$.
M_i	Stacked blocks indexed by $\mathcal{N}_i$. Block j is M_{ij}.

Table 1. Description of matrices.

Let $\mu_{i,n}$ and $\nu_{i,n}$ denote the optimal values of the dual variables μ_i and ν_i. At the optimal values of all the variables, the gradient of the Lagrangian with respect to (w_i, b_i), $(\Delta v_{ij})_{j \in \mathcal{N}_i}$, and η_i must be zero. From this fact, we can show that

$$\Delta v_{ij,n} = 2\gamma \mu_{i,n} - z_{3,ji,n} \qquad (11)$$

and

$$b_{i,n} = \frac{1}{m_i}\Big(2\gamma a_i^T \mu_{i,n} + \sum_{j \in \mathcal{N}_i} z_{2,ji,n}\Big), \qquad (12)$$

where m_i denotes the cardinality of $\mathcal{N}_i$. Let the family $(z_{1,ji,n})_{j \in \mathcal{N}_i}$ be in the form of a stacked vector, and let I denote the identity matrix. By defining

$$J_{i,\gamma} = (\tilde{M}_i + 2\gamma I)^{-1}$$

and

$$\tilde{\mu}_{i,n} = \frac{1}{2\gamma}\big(I - J_{i,\gamma}\tilde{M}_i\big)(z_{1,ji,n})_{j \in \mathcal{N}_i} - J_{i,\gamma} M_i \mu_{i,n}, \qquad (13)$$

we can show that

$$r_{ij,n} = M_{ij}\mu_{i,n} + \tilde{M}_{ij}^T \tilde{\mu}_{i,n}. \qquad (14)$$

By defining

$$p_{i,n} = \frac{1}{m_i} \sum_{j \in \mathcal{N}_i} (z_{2,ji,n} a_i - m_i z_{3,ji,n}) - \frac{1}{m}\mathbf{1} + \frac{1}{2\gamma} M_i^T (I - \tilde{M}_i J_{i,\gamma})(z_{1,ji,n})_{j \in \mathcal{N}_i} \qquad (15)$$

and

$$H_i = \gamma m_i I + \frac{\gamma}{m_i} a_i a_i^T + \frac{1}{2}(G_i - M_i^T J_{i,\gamma} M_i),$$

we can show that $(\mu_{i,n}, \nu_{i,n})$ corresponds to the solution to

$$\min_{(\mu_i, \nu_i) \in \mathcal{S}} \ \mu_i^T H_i \mu_i + \mu_i^T p_{i,n} + \nu_i \frac{\epsilon}{m}, \qquad (16)$$

where $\mathcal{S}$ is an epigraph that can be expressed as

$$\{(\mu, \nu) \in \mathbb{R}^\ell \times \mathbb{R}\colon 0 \leq [\mu]_k \leq \nu \ \ \forall k\}.$$

From the results in this section, we can also use the kernel to express the approximation of h provided by our approach:

$$h_{i,n}(x) = \sum_{k=1}^{\ell} [a_i]_k [\mu_{i,n}]_k K(x, x_k) + \sum_{j \in \mathcal{N}_i} \sum_{k=1}^{\ell_{ij}} [\tilde{\mu}_{i,n}]_k K(x, \tilde{x}_{ij,k}) + b_{i,n}, \quad x \in \mathcal{X}. \qquad (17)$$

Through duality, we have described the modified algorithm using kernel evaluations. This constitutes our main result, Algorithm 2.

Algorithm 2 Kernelized distributed constrained-form SVM

By carrying out the following steps, each agent i obtains successive approximations, $h_{i,0}$, $h_{i,1}$, ..., of the function h that solves (3).

The first step is to initialize the network with two real numbers, $\gamma > 0$ and $\lambda \in (0, 2)$, and to let each agent i choose, for each j in $\mathcal{N}_i$, three elements, $z_{1,ij,0} \in \mathbb{R}^{\ell_{ij}}$, $z_{2,ij,0} \in \mathbb{R}$, and $z_{3,ij,0} \in \mathbb{R}^{\ell}$. Then, in parallel with the other agents, each agent repeats the following steps. After n iterations, agent i

1. receives $z_{1,ji,n}$, $z_{2,ji,n}$, and $z_{3,ji,n}$ from each neighbor j;
2. computes (15), solves (16), evaluates (12) and (13), and obtains $h_{i,n}$ from (17);
3. computes (11) and (14), and updates, for each j in $\mathcal{N}_i$, three elements, $z_{1,ij,n+1}$, $z_{2,ij,n+1}$, and $z_{3,ij,n+1}$, via (10), (8), and (9), and sends these three elements to neighbor j.

2.7. Implementation

Because we have to solve (16) at every iteration of Algorithm 2, it is important that we do so efficiently.

To solve the problem, we can use a projection-gradient method [11, Corollary 27.10, p. 406], computing, for $t = 0, 1, \ldots,$

$$(\mu_{i,n,t+1}, \nu_{i,n,t+1}) = \operatorname{proj}_{\mathcal{S}}\Big(\mu_{i,n,t} - \delta_i(2H_i\mu_{i,n,t} + p_{i,n}), \nu_{i,n,t} - \delta_i\frac{\epsilon}{m}\Big), \quad (18)$$

where δ_i is a real number in $(0, 2/L_i)$, and L_i—given by twice the spectral norm of H_i—is the Lipschitz constant of the gradient of the objective function in (16). We have found that two iterations of this method suffice to maintain the convergence of our approach provided that we initialize $(\mu_{i,n,0}, \nu_{i,n,0})$ to $(\mu_{i,n-1}, \nu_{i,n-1})$. We describe this subroutine in Algorithm 3.

To compute the projection, we can prove the following result, inspired by an epigraphical projection in the context of a centralized multiclass constrained-form SVM [5, Prop. 3.3].

Proposition 1. *Let u be a vector in $\mathbb{R}^{\ell}$, and let $\bar{u}$ denote u with entries sorted in ascending order. Let v be a real number. Define*

$$q_k = \max\{0, (v + [\bar{u}]_k + \cdots + [\bar{u}]_{\ell})/(\ell + 2 - k)\}, \quad k = 1, \ldots, \ell.$$

Then, at most one of the following inequalities is true:

$$q_1 \le [\bar{u}]_1, \quad [\bar{u}]_1 < q_2 \le [\bar{u}]_2, \quad \ldots, \quad [\bar{u}]_{\ell-1} < q_{\ell} \le [\bar{u}]_{\ell}.$$

Set $\nu' = q_k$ for the q_k such that the inequality is true; if none is true, set $\nu' = \max\{0, v\}$. The projection (μ, ν) of the point (u, v) on $\mathcal{S}$ is such that

$$[\mu]_k = \operatorname{median}\{0, [u]_k, \nu'\} \quad \forall k \quad \textit{and} \quad \nu = \nu'.$$

Algorithm 3 Implementation of Step 2 of Algorithm 2

During the initialization of Algorithm 2, each agent i chooses a point $(\mu_{i,-1}, \nu_{i,-1})$ in $\mathbb{R}^{\ell} \times \mathbb{R}$ and a real number δ_i in $(0, 2/L_i)$.

At Step 2, agent i

a) sets $(\mu_{i,n,0}, \nu_{i,n,0}) = (\mu_{i,n-1}, \nu_{i,n-1})$;
b) computes (15), then (18) for $t = 0$ and $t = 1$ using Proposition 1;
c) sets $(\mu_{i,n}, \nu_{i,n}) = (\mu_{i,n,2}, \nu_{i,n,2})$, computes (12) and (13), and obtains $h_{i,n}$ from (17).

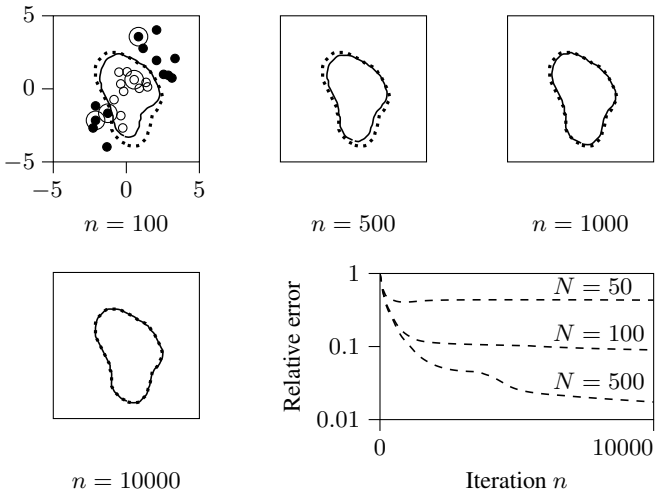

Fig. 1. Agent 1's decision boundary (——) closely agrees with the centralized one (·····) after 10000 iterations of Algorithm 2 when the agents share 500 points. As the number of these points increases (see - - -), the closer to the centralized result the agent can get.

3. SIMULATIONS

In this section, we illustrate the effectiveness of our approach.

We consider a training dataset of 24 objects in $\mathbb{R}^2$ from two equiprobable classes. In one class, the objects arise from a normal distribution of mean $\left(\begin{smallmatrix}0\\0\end{smallmatrix}\right)$ and covariance $\left(\begin{smallmatrix}0.2 & 0\\0 & 0.3\end{smallmatrix}\right)$. In the other class, the objects arise from a mixture of two normal distributions, one with proportion 0.3 having mean $\left(\begin{smallmatrix}-2\\-2\end{smallmatrix}\right)$ and covariance $\left(\begin{smallmatrix}0.5 & 0\\0 & 1\end{smallmatrix}\right)$, and the other having mean $\left(\begin{smallmatrix}2\\2\end{smallmatrix}\right)$ and covariance $\left(\begin{smallmatrix}0.5 & 0\\0 & 2\end{smallmatrix}\right)$.

We consider a network of six agents as in [9, Sec. 4]. The agents each know a different subset of four labeled objects. In Fig. 1, for example, the circled points are those that agent 1 knows. All the agents know the same N arbitrary points whose entries are drawn uniformly between the minimum and maximum values of the corresponding coordinate in the dataset. We set ϵ, γ, and λ to 1 and δ_i to $1.99/L_i$. The agents use a Gaussian kernel with $C = 1.8$.

Fig. 1 illustrates the convergence of Algorithm 2 (with the implementation provided by Algorithm 3). It shows the relative error,

$$\sqrt{\frac{\|w_{1,n} - w^*\|^2 + (b_{1,n} - b^*)^2}{\|w^*\|^2 + (b^*)^2}},$$

between agent 1's approximation $(w_{1,n}, b_{1,n})$ and the result (w^*, b^*) obtained centrally (the relative error can be computed using kernel evaluations). The figure also shows the evolution of agent 1's decision boundary. Even if the agent only knows some of the training points, it obtains almost the same decision boundary as if it knew all of them.

4. CONCLUSION

In this paper, we have developed a distributed SVM algorithm. The machinery behind our approach consists of the Douglas-Rachford algorithm and the projection-gradient method. We have provided an implementation based on an efficient epigraphical projection.

Our study demonstrates that it is possible to train a nonlinear constrained-form SVM in a fully distributed way, and that despite the complexity of the problem, it is possible to train the classifier efficiently, using a sequence of closed-form steps.

5. REFERENCES

[1] D. Peteiro-Barral and B. Guijarro-Berdiñas, "A survey of methods for distributed machine learning," *Prog. Artif. Intell.*, vol. 2, no. 1, pp. 1–11, 2013.

[2] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *J. Mach. Learn. Res.*, vol. 11, pp. 1663–1707, 2010.

[3] D. Wang, J. Zheng, Y. Zhou, and J. Li, "A scalable support vector machine for distributed classification in ad hoc sensor networks," *Neurocomputing*, vol. 74, no. 1–3, pp. 394–400, 2010.

[4] S. Scardapane, R. Fierimonte, P. Di Lorenzo, M. Panella, and A. Uncini, "Distributed semi-supervised support vector machines," *Neural Net.*, vol. 80, pp. 43–52, 2016.

[5] G. Chierchia, N. Pustelnik, J.-C. Pesquet, and B. Pesquet-Popescu, "A proximal approach for sparse multiclass SVM," arXiv:1501.03669, 2015.

[6] L. Oneto, S. Ridella, and D. Anguita, "Tikhonov, Ivanov and Morozov regularization for support vector machine learning," *Mach. Learn.*, vol. 103, no. 1, pp. 103–136, 2016.

[7] D. Lorenz and N. Worliczek, "Necessary conditions for variational regularization schemes," *Inverse Probl.*, vol. 29, no. 7, 2013.

[8] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[9] F. D. Côté, I. N. Psaromiligkos, and W. J. Gross, "In-network linear regression with arbitrarily split data matrices," in *Proc. IEEE Global Conf. Signal and Information Processing*, 2016, pp. 580–584.

[10] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2002.

[11] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. New York: Springer, 2011.