Asynchronous Distributed Learning from Constraints[*]

Francesco Farina, Stefano Melacci, Andrea Garulli, and Antonio Giannitrapani

Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche,
Università di Siena, Siena, Italy.

Abstract

In this paper, the extension of the framework of Learning from Constraints (LfC) to a distributed setting where multiple parties, connected over the network, contribute to the learning process is studied. LfC relies on the generic notion of "constraint" to inject knowledge into the learning problem and, due to its generality, it deals with possibly nonconvex constraints, enforced either in a hard or soft way. Motivated by recent progresses in the field of distributed and constrained nonconvex optimization, we apply the (distributed) Asynchronous Method of Multipliers (ASYMM) to LfC. The study shows that such a method allows us to support scenarios where selected constraints (i.e., knowledge), data, and outcomes of the learning process can be locally stored in each computational node without being shared with the rest of the network, opening the road to further investigations into privacy-preserving LfC. Constraints act as a bridge between what is shared over the net and what is private to each node and no central authority is required. We demonstrate the applicability of these ideas in two distributed real-world settings in the context of digit recognition and document classification.

1 Introduction

The generic framework of Learning from Constraints (LfC) [1–3] reframes the learning process in a context that is described by a collection of constraints. Such constraints are the mean that is used to inject knowledge into the learning process and they represent different aspects of the task at hand. The goal of LfC is to learn a vector function f (classifier, regressor, etc.) by solving a constrained optimization problem where f is required to maximize some regularity conditions in the space to which it belongs [2]. Different types of knowledge can be exploited in LfC, including the ones that are represented using First-Order Logic (FOL) formulas [4]. For example, knowledge on the relationships among classes [5,6], on the interactions among different tasks [7], and on labeled regions of the input space [8], can be easily converted into constraints and embedded in the LfC learning problem (including point-wise constrains $f(x) - y = 0$ on supervised pairs (x, y)). The strength of LfC is more evident when using semi-supervised data, thus enforcing constraints also on unsupervised samples. Depending on the type of knowledge, constraints can be convex or non-convex, enforced in a soft or hard way [3].

To the best of our knowledge, LfC has always been conceived as a *centralized* framework, where constraints (i.e., knowledge), data, and the learned predictors are all handled within the same computational unit. This paper studies the extension of LfC to the *distributed* setting, where multiple computational nodes, connected over the network, contribute to the learning process. This setting is inspired by the nowadays organization of data and knowledge, where it is extremely common to participate to communities over the net, sharing some resources (e.g., public photos on social networks), keeping other local (e.g., private pictures taken with a personal smartphone, saved on the cloud), and having the need of developing (and eventually sharing) customized or more robust services that might benefit both from private data and public data taken from the net (e.g., a recognizer of pictures of a custom type). Our goal consists in formulating a distributed implementation of LfC with a generic structure that covers the described setting and that could be further extended emphasizing more specific aspects, such as the ones related to privacy-preserving methods [9–12]. The generality of LfC prevents the direct application of many distributed optimization approaches (see [13] and references therein), since we need to support hard, soft, convex and nonconvex constraints. There has been several recent progresses in the field of distributed constrained optimization [14–17], and the Asynchronous Method of Multipliers (ASYMM) [18,19] offers the capability of dealing with convex and nonconvex constraints that are locally defined in computational nodes. Moreover, ASYMM has been proved to be equivalent to a centralized instance of the Method of Multipliers [20], thus inheriting the properties of its centralized counterpart.

It is worth observing that recently there has been an increased interest in distributed learning scenarios (see, e.g., [21–24]). Specific frameworks have been studied, like the one of federated learning [25,26], and several algorithms have been proposed [27–29]. However, distributed learning is usually intended in the sense of *learning from distributed datasets*, and central servers are required to perform at least a part of the learning process. Conversely, in this paper we consider a scenario in which not only data, but also knowledge is distributed in the network. Moreover, we exploit a fully distributed architecture, in which no central computational unit is required.

The main contribution of this work is to tailor the ASYMM algorithm to the aforementioned LfC distributed setting, showing how constraints can be used as a bridge between shared and private resources. As a proof-of-concept, the model is applied to two real-world problems: digit image classification and document classification. In both cases, we consider semi-supervised data, constraints on supervised examples, and constraints devised from FOL formulas. The results show that, in this distributed setting, FOL-based constraints improve the quality of the private classifiers, and local and shared constraints are asymptotically fulfilled.

2 Learning from constraints

In the framework of LfC we consider the problem of finding the most suitable vector function $f := [f_1, \ldots, f_F] \in \mathcal{F}$ subject to a set of constraints that models the available knowledge on the considered problem. $\mathcal{F}$ is a space of functions from $\mathcal{X} \subset \mathbb{R}^d$ (being d the dimensionality of the input data) to $\mathbb{R}^F$ where a regularity measure is defined, and each f_i is referred to as "task function" (for example, a classifier of a certain class). It is pretty common to enforce *point-wise* constraints, i.e., constraints applied to f evaluated on a given collection of data points $\mathcal{D}$, and to consider both the *bilateral* and/or *unilateral* cases, that we denote by

$$\Phi\left(f \mid \mathcal{D}\right) = 0, \quad \check{\Phi}\left(f \mid \mathcal{D}\right) \leq 0, \tag{1}$$

2

respectively. Notice that Φ and $\check{\Phi}$ compactly indicate vectors of constraints[1]. We consider $\mathcal{D}$ to be partitioned into a collection of points for which a label $y \in Y$ is known and a set of unlabeled points, respectively collected in $\hat{\mathcal{D}}$ and $\tilde{\mathcal{D}}$,

$$\mathcal{D} = \hat{\mathcal{D}} \cup \tilde{\mathcal{D}} \ . \tag{2}$$

A popular category of constraints that is frequently exploited in LfC is given by polynomials derived from First-Order Logic formulas [4] . In particular, each task function f_i is assumed to implement the activation in $[0, 1]$ of a predicate that describes a property of the considered environment, and FOL formulas represent relationships among such properties, i.e., among the tasks in f. FOL formulas are then converted into numerical constraints using Triangular Norms (T-Norms, [30]), special binary functions that generalize the conjunction operator $f_i \wedge f_j$. For example, we might know that, in the considered environment, $f_1(x) \wedge f_2(x) \Rightarrow f_3(x), \forall x \in \mathcal{D}$. This information is converted into a bilateral constraint of Eq. (1), that in the case of the product T-Norm is $f_1(x) f_2(x)(1 - f_3(x)) = 0$, and applied to all the data points of $\mathcal{D}$ (see [4] for more examples). In this paper, we assume f to be a generic neural network. As regularity measure we use the squared norm of the weights, leading to the popular weight decay term (for simplicity, we avoid reporting this term in the following equations).

Depending on the nature of the constraints, it could be necessary to enforce some of them in a *hard* way, and others in a *soft* manner [3]. While supervisions on examples are generally subject to noise (suggesting a penalty-based soft enforcement), there might be structural or environment-related conditions that must be enforced in a *hard* way. In the rest of the paper, we will use the notation $\Phi, \check{\Phi}$ to refer to those constraints that must be enforced in a hard way, while ψ indicates the sum of the penalty functions associated to the soft constraints[2]. Moreover, the choices of both the form of the constraints of (1) and of the form of f usually end up in generating constraints that are *nonconvex* with respect to the model parameters that are subject of optimization. This consideration holds even more strongly when we select f to be a generic neural net.

3 Distributed framework

When moving to the distributed setting, we consider N computational nodes connected over a network, whose underlying connectivity structure can be represented through an undirected and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of node-to-node connections. Nodes might have some specific requirements in terms of the resources they want to keep *private* (local) and the ones they want to *share* with the other nodes. We distinguish among three types of resources: *data* (i.e, the available data points), *knowledge* (i.e. constraints), and *predictors* (the outcome of the learning process, i.e., the task functions in f). Figure 1 illustrates the distributed framework with privacy conditions, where we can distinguish the node-private resources and the shared ones, accessible by all the nodes. From the notation point of view, the subscript i indicates a private resource of the i-th node, while the superscript s is used to refer to shared resources.

More formally, for each node i we consider some private data $\mathcal{D}_i$, private knowledge $\Phi_i, \check{\Phi}_i, \psi_i$, and some private predictors modeled with a vector function $p_i(x, w_i)$, where w_i are

---

[1]For simplicity, all the constraints are applied to the same $\mathcal{D}$, but our approach also holds when different constraints operate on different data. We will sometimes replace the vector f with an explicit list of functions.

[2]The definitions of ψ can include weighting terms to give different importance to the different soft constraints.
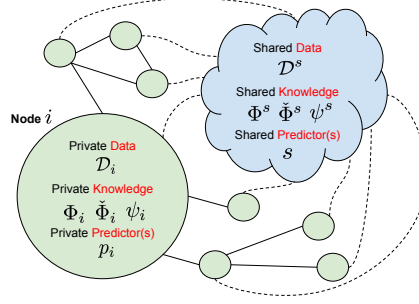
FIGURE 1: An illustrative view of the distributed framework and of the notation used in the paper. The graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by the greenish nodes (belonging to $\mathcal{V}$) and the solid connections between them (belonging to $\mathcal{E}$). The blue cloud only includes shared elements, and it is accessible by all the nodes of the graph.

the learnable parameters of the neural network. Similarly, for the whole network we define shared data $\mathcal{D}^s$, shared knowledge $\Phi^s, \check{\Phi}^s, \psi^s$, and some shared predictors implemented by the vector function $s(x, w^s)$, with parameters w^s. Then, the merged collection of all the data (Eq. (2)) and the set of all the model parameters (shared and private) are

$$\mathcal{D} = \left(\bigcup_{i=1}^{N} \mathcal{D}_i \right) \cup \mathcal{D}^s, \qquad\qquad w = \left(\bigcup_{i=1}^{N} w_i \right) \cup w^s . \tag{3}$$

The centralized optimization problem we have to solve is

$$
\begin{aligned}
\underset{w}{\text{minimize}} \quad & \sum_{i=1}^{N} \psi_i \left(p_i, s \mid \mathcal{D}_i \cup \mathcal{D}^s \right) + \psi^s \left(s \mid \mathcal{D} \right) \\
\text{subject to} \quad & \Phi_i \left(p_i, s \mid \mathcal{D}_i \cup \mathcal{D}^s \right) = 0, \quad \forall i = 1, \dots, N \\
& \check{\Phi}_i \left(p_i, s \mid \mathcal{D}_i \cup \mathcal{D}^s \right) \leq 0, \quad \forall i = 1, \dots, N \\
& \Phi^s \left(s \mid \mathcal{D} \right) = 0, \\
& \check{\Phi}^s \left(s \mid \mathcal{D} \right) \leq 0
\end{aligned}
\tag{4}
$$

where constraints involve the just introduced vector functions $p_i \in \mathcal{F}$, $i = 1, \dots, N$, and $s \in \mathcal{F}$. Notice that the private constraints can involve both private and shared predictors, thus bridging shared and local resources. Due to their shared nature, constraints $\Phi^s, \check{\Phi}^s, \psi^s$ can be enforced in all the available data.

Let us define $w^s|_i$ as a local copy of w^s made by node i and $w|_i = w_i \cup w^s|_i$. Following the same intuition behind [18], we rewrite problem (4) in an equivalent form, exploiting the connectedness of $\mathcal{G}$,

$$
\begin{aligned}
\underset{w|_1, \dots, w|_N}{\text{minimize}} \quad & \sum_{i=1}^{N} \left(\psi_i \left(p_i, s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s \right) + \frac{1}{N} \psi^s \left(s|_i \mid \mathcal{D}^s \right) + \psi^s \left(s|_i \mid \mathcal{D}_i \right) \right) \\
\text{subject to} \quad & w^s|_i = w^s|_j, && \forall (i,j) \in \mathcal{E} \\
& \Phi_i \left(p_i, s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s \right) = 0, && \forall i \in \mathcal{V} \\
& \check{\Phi}_i \left(p_i, s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s \right) \leq 0, && \forall i \in \mathcal{V} \\
& \Phi^s \left(s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s \right) = 0, && \forall i \in \mathcal{V} \\
& \check{\Phi}^s \left(s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s \right) \leq 0 && \forall i \in \mathcal{V}
\end{aligned}
\tag{5}
$$

4

where $s|_i = s(x, w^s|_i)$. The first constraint ensures consistency of the local copies over $\mathcal{G}$, and the last two constraints (involving shared resources) are now replicated N times, splitting the private portion of the data. The objective function has been regrouped in order to be a summation over the index i, paying attention to differently weigh ψ^s when applied to shared or private data. This formulation of the problem can be more easily partitioned among the nodes and it helps in the application of a distributed optimization algorithms.

4 ASYMM algorithm

The Asynchronous Method of Multipliers (ASYMM) is a distributed optimization algorithm that has no central authorities, and that solves constrained optimization problems in which both local cost functions and constraints can be nonconvex. Thus, it is a well suited method for solving problem (4) in a distributed way, when rewritten in the form (5).

The idea behind ASYMM is rooted around the concept of computational units that wake up asynchronously at different time instants, perform some operations, and broadcast their local copies of *shared* parameters $w^s|_i$ (and some other variables) to their neighbors $\mathcal{N}_i$. We assume that each node keeps waking up indefinitely and the time interval between two consecutive awakenings is bounded for all nodes. Moreover, we assume for simplicity that it cannot happen for two nodes to be awake in the same time instant. When nodes i wakes up, it performs a gradient descent step on a *locally defined* augmented Lagrangian until every neighboring node matches a convergence criterion based on a node-defined tolerance ϵ_i. By doing so, the nodes collectively approach a stationary point of the entire augmented Lagrangian of the considered optimization problem. The convergence check on the augmented Lagrangian is performed by the nodes in a distributed way, using a logic-AND algorithm (see [18]). When a node gets aware of the convergence condition, it performs one ascent step on its local multiplier vector and it increases its penalty parameters. After a node has received the updated multipliers and penalty parameters associated to *shared* constraints from all its neighbors on $\mathcal{G}$, it starts over a new Lagrangian minimization. Under suitable technical assumptions, it can be shown that the computational units collectively converge to a local minimum of problem (4) which satisfies all the constraints. Moreover *private* resources are never passed over the net.

In order to devise a specialized version of ASYMM for problem (5), we need to introduce the corresponding augmented Lagrangian. Let ν_{ij} and ρ_{ij} be the multiplier vector and penalty parameter associated to the equality constraint $w^s|_i = w^s|_j$. We compactly define $\nu_i = [\nu_{ij}]_{j \in \mathcal{N}_i}$, $\rho_i = [\rho_{ij}]_{j \in \mathcal{N}_i}$. Similarly, let λ_i, λ_i^s and ϱ_i, ϱ_i^s (resp. μ_i, μ_i^s and ζ_i, ζ_i^s) be the multiplier and penalty parameter associated to the equality (resp. inequality) constraint of node i, where the superscript s denotes the association with the local copy of the shared constraints. Moreover, let $w = [w|_1; ...; w|_N]$, and denote by $p = [\rho_i, \varrho_i, \zeta_i]_{i \in \mathcal{V}}$ the vector stacking all the penalty parameters; $\nu = [\nu_i]_{i \in \mathcal{V}}$, $\lambda = [\lambda_i, \lambda_i^s]_{i \in \mathcal{V}}$ and $\mu = [\mu_i, \mu_i^s]_{i \in \mathcal{V}}$ be the vectors stacking the corresponding multipliers, and, consistently, let $\theta = [\nu; \lambda; \mu]$. Finally, in order to present the next equations in a simpler way, let us define two parametric functions with a compact notation: $q_c(a, b) = \frac{1}{2c}\left(\max\{0, a + cb\}^2 - a^2\right)$ and $v_c(a, b) = a^\top b + \frac{c}{2}\|b\|^2$, where the max operator is to be intended component-wise. Then, the augmented Lagrangian

associated to (5) is

$$\mathcal{L}_p(w, \theta) = \sum_{i=1}^{N} \left\{ \psi_i(p_i, s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s) + \frac{1}{N} \psi^s(s|_i \mid \mathcal{D}^s) \right.$$
$$+ \psi^s(s|_i \mid \mathcal{D}_i) + \sum_{j \in \mathcal{N}_i} v_{\rho_{ij}}(\nu_{ij}, w^s|_i - w^s|_j) +$$
$$+ v_{\varrho_i}(\lambda_i, \Phi_i(p_i, s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s)) +$$
$$+ v_{\varrho_i^s}(\lambda_i^s, \Phi^s(s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s)) +$$
$$+ \mathbb{1}^\top q_{\zeta_i}(\mu_i, \check{\Phi}_i(p_i, s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s)) +$$
$$\left. + \mathbb{1}^\top q_{\zeta_i^s}(\mu_i^s, \check{\Phi}^s(s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s)) \right\}, \tag{6}$$

where $\mathbb{1}$ is a (column) vector of ones.

In order to collectively minimize (6), nodes in ASYMM need to compute a *local* augmented Lagrangian. The local augmented Lagrangian for node i groups all the terms in (6) depending on $w|_i$ and it is defined as

$$\tilde{\mathcal{L}}_{p_{\mathcal{N}_i}}(w_i, w^s|_{\mathcal{N}_i}, \theta_{\mathcal{N}_i}) = \psi_i(p_i, s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s) + \frac{1}{N} \psi^s(s|_i \mid \mathcal{D}^s) +$$
$$+ \psi^s(s|_i \mid \mathcal{D}_i) + \sum_{j \in \mathcal{N}_i} v_{\rho_{ij}}(\nu_{ij}, w^s|_i - w^s|_j) +$$
$$+ v_{\varrho_i}(\lambda_i, \Phi_i(p_i, s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s)) +$$
$$+ v_{\varrho_i^s}(\lambda_i^s, \Phi^s(s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s)) +$$
$$+ \mathbb{1}^\top q_{\zeta_i}(\mu_i, \check{\Phi}_i(p_i, s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s)) +$$
$$+ \mathbb{1}^\top q_{\zeta_i^s}(\mu_i^s, \check{\Phi}^s(s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s))) \tag{7}$$

where $w^s|_{\mathcal{N}_i} = [w^s|_j]_{j \in \mathcal{N}_i \cup \{i\}}$, $\theta_{\mathcal{N}_i} = [\lambda_i, \lambda_i^s, \mu_i, \mu_i^s, \nu_i, [\nu_{ji}]_{j \in \mathcal{N}_i}]$, and $p_{\mathcal{N}_i} = [\varrho_i, \varrho_i^s, \zeta_i, \zeta_i^s, \rho_i, [\rho_{ji}]_{j \in \mathcal{N}_i}]$.

Finally, we define a local binary matrix $S_i \in \{0, 1\}^{d_G \times d_i}$ for each node, where d_G is the graph diameter and $d_i = |\mathcal{N}_i| + 1$. Such a matrix is used to perform the distributed logic-AND algorithm, which is a building block of ASYMM (see [18]).

Given the above definitions, the ASYMM algorithm applied to problem (5) is reported in Algorithm 1, being $\alpha_i > 0$ the stepsize selected by node i.

In [18], it has been shown that the distributed Algorithm 1 is equivalent to a centralized version of the Method of Multipliers in which the primal update is carried out by means of a *block-coordinate gradient descent* on the augmented Lagrangian (e.g., see [31] for a survey on coordinate descent algorithms). Specifically, there exists a sequence of (centralized) block-coordinate gradient descent steps that returns the same sequence of estimates $w^s|_i$ as those computed by ASYMM. This equivalence property implies that ASYMM inherits all the convergence properties of the centralized block-coordinate Method of Multipliers. In particular, under technical assumptions on the local augmented Lagrangian, similar to those adopted in the centralized case (see [20]), the estimates $w^s|_i$ generated by ASYMM converge to a local minimum. A key point to establish this result is to bound the norm of the gradient of the augmented Lagrangian (5) by a function of the local tolerances ϵ_i employed in Algorithm 1. The interested reader is referred to [18] for a thorough theoretical analysis of ASYMM.

Algorithm 1 ASYMM

Initialization: $w|_i$, θ_i, $\mathcal{N}_i$, p_i, $S_i = \mathbf{0}_{d_G \times d_i}$, $M_{done} = 0$.

AWAKE

    if $\prod_{b=1}^{d_i} S_i[d_G, b] \neq 1$ **and not** M_{done} **then**

        $w^s|_i \leftarrow w^s|_i - \alpha_i \nabla_{w^s|_i} \tilde{\mathcal{L}}_{p_{\mathcal{N}_i}}(w_i, w^s|_{\mathcal{N}_i}, \theta_{\mathcal{N}_i})$

        if $\|\nabla_{w^s|_i} \tilde{\mathcal{L}}_{p_{\mathcal{N}_i}}(w_i, w^s|_{\mathcal{N}_i}, \theta_{\mathcal{N}_i})\| \leq \epsilon_i$ **then** $S_i[1, d_i] \leftarrow 1$

        $S_i[l, d_i] \leftarrow \prod_{b=1}^{d_i} S_i[l-1, b]$ for $l = 2, ..., d_G$

        BROADCAST $w^s|_i$, $S_i[:, d_i]$ to all $j \in \mathcal{N}_i$

    if $\prod_{b=1}^{d_i} S_i[d_G, b] = 1$ **and not** M_{done} **then**

        $\nu_{ij} \leftarrow \nu_{ij} + \rho_{ij}(w^s|_i - w^s|_j)$ for $j \in \mathcal{N}_i$

        $\lambda_i \leftarrow \lambda_i + \varrho_i \Phi_i(p_i, s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s)$

        $\lambda_i^s \leftarrow \lambda_i^s + \varrho_i^s \Phi^s(s|_i \mid \mathcal{D}^s)$

        $\mu_i \leftarrow \max\{0, \mu_i + \zeta_i \check{\Phi}_i(p_i, s|_i \mid \mathcal{D}_i \cup \mathcal{D}^s)\}$

        $\mu_i^s \leftarrow \max\{0, \mu_i^s + \zeta_i^s \check{\Phi}^s(s|_i \mid \mathcal{D}^s)\}$

        update ϱ_i, ζ_i and ρ_i

        $M_{done} \leftarrow 1$

        BROADCAST ν_{ij}, ρ_{ij} to $j \in \mathcal{N}_i$

IDLE

    If $S_j[:, d_j]$ received from $j \in \mathcal{N}_i$ and not already received some new ν_{ji} **then** $S_i[l, j_i] \leftarrow S_j[l, d_j]$ for $l = 1, ..., d_G$

    if ν_{ji} and ρ_{ji} received from $j \in \mathcal{N}_i$ set $S_i[d_G, :] \leftarrow 1$

    if $w^s|_j^{new}$ received from $j \in \mathcal{N}_i$, update $w^s|_j \leftarrow w^s|_j^{new}$

    if M_{done} **and** ν_{ji} received from all $j \in \mathcal{N}_i$ **then**

        $M_{done} \leftarrow 0$, $S_i \leftarrow \mathbf{0}_{d_G \times d_i}$, update ϵ_i
---

predicate	polynomial form
a $\Rightarrow$ b	$a(1 - b) = 0$
$\neg$ (a $\wedge$ b)	$ab = 0$
(a $\wedge$ b) $\Rightarrow$ c	$ab(1 - c) = 0$
a $\underline{\vee}$ b	$a + b - 1 = 0, ab = 0$

5 Experiments

We evaluate the numerical application of our approach to two different distributed environments, focussing on digit recognition and document classification, respectively.

5.1 Digit Recognition

We consider a network composed by 10 nodes, indexed from 0 to 9. In the context of digit recognition, each node aims at learning to recognize a precise digit given its image x, where we assume that node i learns to recognize digit i. Notice that each node could also learn to recognize more than one digit. We consider only one digit per node for the sake of presentation. The i-th recognizer is a *private function* $p_i(x, w_i) \in [0, 1]$, and the i-th node has the use of *private data* $\mathcal{D}_i = \hat{\mathcal{D}}_i \cup \tilde{\mathcal{D}}_i$ composed of positive examples of such digit and negative examples of other digits (labeled with $y = 1$ and $y = 0$, respectively, and collected in $\hat{\mathcal{D}}_i$), and unsupervised examples (belonging to $\tilde{\mathcal{D}}_i$). No shared data are considered, so that $\mathcal{D} = \bigcup_{i=0}^{9} \mathcal{D}_i$. All the nodes of the network have access to a *shared function* with two scalar outputs $s(x, w^s) = [s_0(x, w^s), s_1(x, w^s)] \in [0, 1]^2$, that predicts whether x is even (first output) or odd (second output)[3].

Fitting the labeled examples in node i is a private soft-constraint, and it depends on p_i and $\hat{\mathcal{D}}_i$ only,

$$\psi_i(p_i \mid \hat{\mathcal{D}}_i) = \sum_{(x,y) \in \hat{\mathcal{D}}_i} (p_i(x, w_i) - y)^2 \; . \tag{8}$$

Due to the private nature of $\mathcal{D}_i$, each node has no information that it can directly use to learn s in discriminative way. However, each node has *private knowledge* about the fact that its associated digit is either even or odd, and all the nodes have access to the *shared knowledge* that s_0 and s_1 are mutually exclusive. Using FOL, we get the following universally quantified formulas (for the sake of simplicity, we skip the arguments of p_i and s),

$$p_i \Rightarrow s_0 \quad \text{for } i = 0, 2, 4, 6, 8$$
$$p_i \Rightarrow s_1 \quad \text{for } i = 1, 3, 5, 7, 9$$
$$s_0 \underline{\vee} s_1.$$

Using the product T-Norm (see Table 1), we convert the formulas into polynomial

---

[3]We used two outputs to emphasize the role of the mutual-exclusivity constraints that we will introduce shortly.

constraints that, following the notation of problem (5), become

$$\Phi_i(p_i, s|_i | \mathcal{D}_i) = p_i(x, w_i)(1 - s_0(x, w^s|_i)) = 0, \qquad \forall x \in \mathcal{D}_i, \ i = 0, 2, 4, 6, 8$$

$$\Phi_i(p_i, s|_i | \mathcal{D}_i) = p_i(x, w_i)(1 - s_1(x, w^s|_i)) = 0, \qquad \forall x \in \mathcal{D}_i, \ i = 1, 3, 5, 7, 9$$

$$\Phi^s(s|_i | \mathcal{D}_i) = \begin{bmatrix} s_0(x, w^s|_i) + s_1(x, w^s|_i) - 1 \\ s_0(x, w^s|_i) s_1(x, w^s|_i) \end{bmatrix} = 0, \qquad \forall x \in \mathcal{D}_i.$$

We selected the popular MNIST dataset to test our algorithm in the proposed scenario. MNIST consists of black and white images of handwritten digits of size 28×28 pixels. Each image is represented through a normalized-flattened vector $x \in [0, 1]^{784}$, in which each entry is a pixel intensity. The dataset comes divided in a training set and a test set, which consist of 60000 and 10000 labeled samples respectively.

In order to generate the private data $\mathcal{D}_i = \hat{\mathcal{D}}_i \cup \tilde{\mathcal{D}}_i$ we proceeded as follows. We randomly selected 6000 training examples, evenly distributed among classes, and we built each $\hat{\mathcal{D}}_i$ from them. In particular, we selected all the 600 images of digit i as positive examples, and 600 images of digits $\neq i$ as negative examples (evenly distributed among digits $\neq i$, and such that there is no overlap among the negative examples of the different $\hat{\mathcal{D}}_i$'s). The unsupervised sets $\tilde{\mathcal{D}}_i$, $i = 0, \ldots, 9$ were implemented by selecting the 54000 training examples not involved in the previous operations, and evenly assigning them to each $\hat{\mathcal{D}}_i$ with no overlap (keeping the original class distribution). We remark that this setting is different from the one that is commonly assumed in semi-supervised classification on the MNIST data, in which the same data (supervised and unsupervised) is shared by all the classifiers and in which only one class is predicted for each test example [32–34].

We modeled each function p_i, $i = 0, \ldots, 9$, s_0 and s_1 using a simple neural architecture, that is a Multi-Layer Perceptron (MLP) with a hidden layer of 300 units (tanh activation function) and an output unit with sigmoidal activation function. Then, we ran ASYMM by repeating 10 times the aforementioned data generation. Each run consists of 50000 total iterations (which means 5000 awakenings per node on average). After solving the optimization problem, the learned predictors were tested using the original MNIST test set. The mean and standard deviation of the obtained F1 scores[4] are reported in Table 2 (left column) for each predictor. While the results on each p_i confirm that each node is reaching its goal of learning a recognizer for digit i, we can also see how the system is learning to correctly (≈ 0.93) predict even and odd digits without having access to any example labeled as even or odd, but only using the hard constraints that are enforced on *private data* in a *distributed setting*. In order to experimentally verify the theoretical properties of ASYMM, we solved in a centralized way the same optimization problem considered in the presented scenario, by using the (centralized) Method of Multipliers. The results are reported in Table 2 (Centralized Semi-Supervised column) and are very close to those of the distributed implementation, the small discrepancies being due to the different orders in which block-coordinate descent steps are performed. In the distributed scenario, in order to give an idea of the role played by unsupervised data, the simulation has been repeated without using the unsupervised data, and the results are reported in the Table 2 (right column). We can see that the F1 scores of all the classifiers are lower than in the semi-supervised scenario, confirming that the distributed implementation positively exploits the unsupervised data. Finally, in order to see how the (hard) logic constraints are asymptotically satisfied, in Figure 2 (left), the average constraint violation (considering all the problem constraints) is reported over the evolution of one run of ASYMM, in logarithmic scale.

---

[4]The F1 score is a classification performance metrics which is typically defined in terms of precision (P) and recall (R) as $F1 = 2\frac{PR}{P+R}$.

TABLE 2: Digit classification problem, F1 score. Distributed and centralized optimization are compared. As a reference, the last column includes the results in the case in which only the supervised portion of the training data is used.

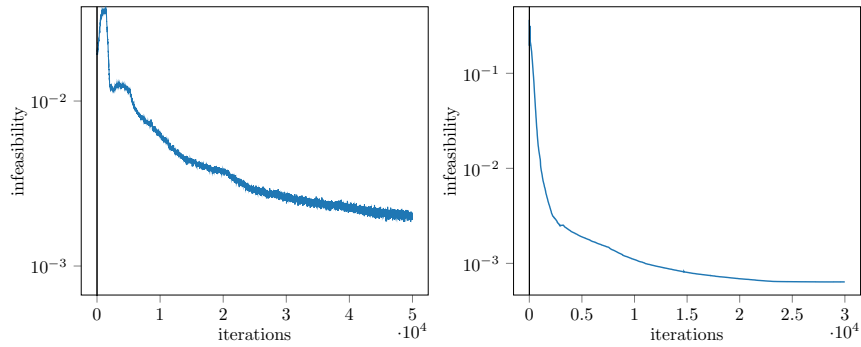| predictor	Distributed Semi-Supervised		Centralized Semi-Supervised		Distributed Supervised	
	mean	std	mean	std	mean	std
p_0	0.923	0.012	0.921	0.012	0.886	0.016
p_1	0.963	0.007	0.970	0.010	0.936	0.012
p_2	0.883	0.016	0.872	0.012	0.861	0.031
p_3	0.859	0.016	0.859	0.016	0.812	0.023
p_4	0.860	0.010	0.863	0.009	0.839	0.014
p_5	0.822	0.020	0.827	0.019	0.813	0.020
p_6	0.881	0.011	0.885	0.017	0.850	0.022
p_7	0.895	0.006	0.888	0.014	0.858	0.013
p_8	0.802	0.015	0.802	0.010	0.755	0.013
p_9	0.789	0.022	0.787	0.021	0.772	0.025
s_0	0.932	0.008	0.934	0.010	0.916	0.016
s_1	0.932	0.009	0.934	0.010	0.915	0.019

FIGURE 2: Average constraints violation over the evolution of the algorithm in (left) digit recognition and (right) text classification (in log scale).

local knowledge	aware nodes
$\neg$ (politics $\wedge$ wrestling)	2, 6
$\neg$ (politics $\wedge$ clothing)	2, 1
$\neg$ (politics $\wedge$ sport)	2, 5
$\neg$ (politics $\wedge$ running)	2, 3
$\neg$ (politics $\wedge$ shoes)	2, 4
wrestling $\Rightarrow$ sport	6, 5
(running $\wedge$ shoes) $\Rightarrow$ clothing	3, 4, 2
running $\Rightarrow$ sport	3, 5

5.2 Document Classification

In the second application, we consider the problem of document classification. We focus on a network of N nodes, each of them associated with a category of documents (N classes). Differently from the previous experiment, each node is assumed to have access to a *shared predictor* $s(x, w^s) = [s_1(x, w^s), \ldots, s_N(x, w^s)]$ with N outputs, that models the class-membership scores of an input document x, and that must be learned in a distributed setting. What makes this task more challenging is that each node i is associated to a unique document class, and it is equipped with a private dataset $\mathcal{D}_i = \hat{\mathcal{D}}_i \cup \tilde{\mathcal{D}}_i$ of (supervised) positive-only examples from the category associated to it (i.e. all the samples in $\hat{\mathcal{D}}_i$ have label $y = 1$), and a set of unlabeled documents ($\tilde{\mathcal{D}}_i$). Moreover, each node has some limited and incomplete *private knowledge* on how its document category is related to the other ones. The goal of the experiment is to make available to each node the N-class classifier s, without sharing private data, and learning from positive examples and constraints in a distributed setting.

We implemented a network of $N = 6$ nodes, and picked the following 6 document categories: *clothing* (1), *politics* (2), *running* (3), *shoes* (4), *sport* (5) and *wrestling* (6), where the number indicates the node index associated to each of them. The private knowledge of each node is reported in Table 3, from which, using the polynomial forms in Table 1, the local constraints can be easily retrieved. As an example, following the described setup, node 4 has the use of positive examples of category 4 (*shoes*) and some other unlabeled data. It also knows how *shoes* is related with some other categories. In particular it knows that the following two relations hold: $\neg$ (*politics* $\wedge$ *shoes*) and (*running* $\wedge$ *shoes*) $\Rightarrow$ *clothing*. Following the rules of Table 1, the local constraint Φ_4 consists of:

$$\Phi_4(s|_4 \mid \mathcal{D}_4) = \begin{bmatrix} s_2(x, w^s|_4)s_4(x, w^s|_4) \\ (s_3(x, w^s|_4)s_4(x, w^s|_4))(1 - s_2(x, w^s|_4)) \end{bmatrix} = 0, \quad \forall x \in \mathcal{D}_4$$

The local objective function, instead, has the same form for all the nodes and is defined as

$$\psi_i(s_i|\hat{\mathcal{D}}_i) = \sum_{(x,y)\in\hat{\mathcal{D}}_i} (s_i(x, w^s|_i) - y)^2$$

The considered problem is in the form of (5) and, hence, can be solved by the ASYMM algorithm.

A collection of 5180 documents belonging to the selected categories has been obtained by crawling Wikipedia. We downloaded up to $1,000$ pages for each category, where roughly 50% of the pages were taken by exploring sub-categories (limiting the depth of the exploration,

and randomly deciding whether we should have considered a subcategory or not). Documents were represented by TF-IDF, on a dictionary of 10000 words. In this experiment, classes are not mutually exclusive. We marked 70% of the documents of each class as supervised samples, while the remaining 30% are marked as unsupervised. All the unsupervised data have been merged, randomized, and evenly assigned to the nodes (without overlap). We explored a transductive learning scenario, so the unlabeled data is also used to evaluate the quality of the learned classifiers.

We tested two types of architectures for the classifiers $s_1, \ldots, s_6$: neural networks without hidden layers (referred to as "single layer") and neural networks with one hidden layer (composed by 100 units with tanh activation). Both architectures share some common properties. Namely, in their output units they have sigmoidal activation functions and a fixed negative bias (-1), and we enforced a strong regularization (weight decay) to better cope with the selected setting (learning from positive examples). ASYMM has been run for 30000 total iterations (5000 awakenings per node on average) and the final results are reported in Table 4. The system is learning to classify the 6 classes, with some low-precision results due to the lack of large discriminative information (many false positives). One of the classifiers with the highest scores is about the *politics* class, since it is the only class which, by means of constraints, is known to be completely disjoint from the others. The classifiers that are more involved into constraints, such as *sport* and *running*, yield better results than the other ones. As a matter of fact, the other classifiers suffer from the small amount of knowledge which is injected in the system through the constraints. For example, the *clothing* classifier has no information to discriminate samples from the class *sport*. Introducing a hidden layer allows the system to develop strongly non-linear decision boundaries around the given positive examples, increasing the overall performance. As in the previous example, in order to corroborate the theoretical properties of ASYMM, we also provide the results obtained by solving the considered optimization problem in a centralized way (Table 4, last column - single layer case), once again almost equivalent to those provided by ASYMM.

In order to further evaluate the proposed distributed setting, we compare the results with those of a centralized approach in which the knowledge on the relationships between the 6 classifiers is not enforced through constraints, but by means of additional supervised data. In particular, we merged all the training sets of the 6 classifiers, and we enriched the supervision with additional labels that are coherent with the logic constraints of Table 3. For example, the data that are labeled as *running* or *wrestling* are also labeled as *sport* (due to constraints 6 and 8 of Table 3), while examples from the *politics* class also become negative examples of the class *sport* (due to constraint 3 of Table 3). We allowed all the classifiers to have access to these data (thus violating the privacy assumption we made in the distributed setting), so that each of them can exploit an augmented collection of supervised training examples with respect to the distributed case. We remark that now classifiers have also the use of negative examples. Then, we excluded the logic constraints and the unsupervised data from the optimization (unsupervised data is not used whenever we drop the logic constraints). Besides the two architectures considered in the set-up of Table 4, we also include the case in which the classifiers are modeled as RBF networks, in order to evaluate the effect of locally supported units in this learning problem. For each classifier we considered 1000 RBF neurons and another hidden layer consisting of 100 units with tanh activation (the centers of the RBF network were estimated on an out-of-sample small set of Wikipedia documents, and shared among the nodes). Results are reported in Table 5. The RBF network performs better on some classifiers, however the overall performances are lower than those obtained with the other two architectures.

By comparing the results reported in Table 5 and 4, it can be seen that the semi-supervised

TABLE 4: Precision (P), Recall (R) and F1 score on document classification, learning from positive examples, unsupervised data, and logic constraints.

predictor	Distributed 1 Hidden Layer			Distributed Single Layer			Centralized Single Layer		
	P	R	F1	P	R	F1	P	R	F1
s_1, clothing	0.366	0.942	0.527	0.344	0.893	0.497	0.345	0.892	0.498
s_2, politics	0.921	0.894	0.907	0.917	0.804	0.857	0.915	0.804	0.856
s_3, running	0.554	0.980	0.709	0.566	0.963	0.713	0.566	0.960	0.712
s_4, shoes	0.351	0.792	0.486	0.304	0.778	0.437	0.301	0.777	0.434
s_5, sport	0.793	0.992	0.940	0.784	0.991	0.875	0.782	0.989	0.873
s_6, wrestling	0.477	0.981	0.641	0.457	0.970	0.621	0.455	0.972	0.620

TABLE 5: Additional reference results in document classification. Predictors are independent, and they exploit a set of positive and also negative examples, obtained by artificially augmenting the original supervised portion of the training data with labels that are coherent with the logic constraints.

predictor	Standard Network 1 Hidden Layer			Standard Network Single Layer			RBF Network		
	P	R	F1	P	R	F1	P	R	F1
s_1, clothing	0.355	0.940	0.516	0.328	0.941	0.487	0.557	0.949	0.697
s_2, politics	0.923	0.929	0.921	0.937	0.907	0.922	0.882	0.888	0.882
s_3, running	0.491	0.990	0.657	0.490	0.980	0.653	0.487	0.996	0.654
s_4, shoes	0.244	0.781	0.371	0.248	0.722	0.369	0.135	0.861	0.231
s_5, sport	0.759	0.995	0.862	0.748	0.995	0.855	0.775	0.989	0.868
s_6, wrestling	0.412	0.980	0.581	0.395	0.983	0.563	0.504	0.981	0.660

scenario (in which we exploit only positive examples, logic constraints and unsupervised data) leads, in general, to slightly better scores than those obtained in the centralized setting with artificially generated labelings. This comparison emphasizes the quality of the proposed distributed setting and validates the idea of sharing knowledge by means of constraints. Finally, the average constraint violation along the evolution of ASYMM is reported in Figure 2 (right) for the single layer architecture, showing that, as expected, the violation vanishes as the algorithm proceeds.

6 Conclusions

We proposed a distributed implementation of the framework of Learning from Constraints. We exploited the Asynchronous Method of Multipliers (ASYMM), and we implemented and evaluated a distributed setting where local (private) and shared resources (including constraints) are considered. Experiments were performed on distributed digit and document classification, confirming the quality of the proposed approach.

References

[1] M. Gori, *Machine Learning: A Constraint-based Approach.* Morgan Kaufmann, 2017.

[2] G. Gnecco, M. Gori, S. Melacci, and M. Sanguineti, "Foundations of support constraint machines," *Neural computation*, vol. 27, no. 2, pp. 388–480, 2015.

[3] ——, "Learning with mixed hard/soft point-wise constraints," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 9, pp. 2019–2032, 2015.

[4] M. Gori and S. Melacci, "Constraint verification with kernel machines," *IEEE transactions on neural networks and learning systems*, vol. 24, no. 5, pp. 825–831, 2013.

[5] M. Maggini, S. Melacci, and L. Sarti, "Learning from pairwise constraints by similarity neural networks," *Neural Networks*, vol. 26, pp. 141–158, 2012.

[6] S. Melacci and M. Gori, "Semi-supervised multiclass kernel machines with probabilistic constraints," in *Lecture Notes in Computer Science, vol. 6934*, R. Pirrone and F. Sorbello, Eds. Springer, 2011, pp. 21–32.

[7] S. Melacci, M. Maggini, and M. Gori, "Semi-supervised learning with constraints for multi-view object recognition," in *Lecture Notes in Computer Science, vol. 5769.* Springer, 2009, pp. 653–662.

[8] S. Melacci and M. Gori, "Learning with box kernels," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 2680–2692, 2013.

[9] M. J. Wainwright, M. I. Jordan, and J. C. Duchi, "Privacy aware learning," in *Advances in Neural Information Processing Systems*, 2012, pp. 1430–1438.

[10] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *Journal of Machine Learning Research*, vol. 12, no. Mar, pp. 1069–1109, 2011.

[11] A. Rajkumar and S. Agarwal, "A differentially private stochastic gradient descent algorithm for multiparty classification," in *Artificial Intelligence and Statistics*, 2012, pp. 933–941.

[12] R. Fierimonte, S. Scardapane, A. Uncini, and M. Panella, "Fully decentralized semi-supervised learning via privacy-preserving matrix completion," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 11, pp. 2699–2711, 2016.

[13] D. Yin, A. Pananjady, M. Lam, D. Papailiopoulos, K. Ramchandran, and P. Bartlett, "Gradient diversity: a key ingredient for scalable distributed learning," in *International Conference on Artificial Intelligence and Statistics*, 2018, pp. 1998–2007.

[14] H.-T. Wai, A. Scaglione, J. Lafond, and E. Moulines, "A projection-free decentralized algorithm for non-convex optimization," in *Signal and Information Processing (GlobalSIP), 2016 IEEE Global Conference on.* IEEE, 2016, pp. 475–479.

[15] P. Di Lorenzo and G. Scutari, "Next: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.

[16] T. Tatarenko and B. Touri, "Non-convex distributed optimization," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3744–3757, 2017.

[17] K. Margellos, A. Falsone, S. Garatti, and M. Prandini, "Distributed constrained optimization and consensus in uncertain networks via proximal minimization," *IEEE Transactions on Automatic Control*, vol. 63, no. 5, pp. 1372–1387, 2018.

[18] F. Farina, A. Garulli, A. Giannitrapani, and G. Notarstefano, "A distributed asynchronous method of multipliers for constrained nonconvex optimization," *Automatica*, vol. 103, pp. 243 – 253, 2019.

[19] ——, "Asynchronous distributed method of multipliers for constrained nonconvex optimization," in *2018 European Control Conference*, 2018.

[20] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods.* Academic press, 2014.

[21] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Advances in neural information processing systems*, 2012, pp. 1223–1231.

[22] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan, "Mlbase: A distributed machine-learning system." in *Cidr*, vol. 1, 2013, pp. 2–1.

[23] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, 2014, pp. 583–598.

[24] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu, "Petuum: A new platform for distributed machine learning on big data," *IEEE Transactions on Big Data*, vol. 1, no. 2, pp. 49–67, 2015.

[25] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[26] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[27] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[28] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: convergence analysis and network scaling," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, 2012.

[29] L. Georgopoulos and M. Hasler, "Distributed machine learning in networks by consensus," *Neurocomputing*, vol. 124, pp. 2–12, 2014.

[30] E. P. Klement, R. Mesiar, and E. Pap, *Triangular norms.* Springer Science & Business Media, 2013, vol. 8.

[31] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[32] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in neural information processing systems*, 2014, pp. 3581–3589.

[33] M. Wang, W. Fu, S. Hao, D. Tao, and X. Wu, "Scalable semi-supervised learning by efficient anchor graph regularization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1864–1877, 2016.

[34] T. Miyato, S.-i. Maeda, S. Ishii, and M. Koyama, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, 2018.