Decentralized Semi-supervised Learning over Multitask Graphs

Maha Issa[†], Roula Nassif[†‡], Elsa Rizk[§], Ali H. Sayed[§]
[†]American University of Beirut, Lebanon
Email: mgi03@mail.aub.edu
[‡]University Côte d'Azur, I3S Laboratory, CNRS, France
Email: roula.nassif@unice.fr
[§]Ecole Polytechnique Fédérale de Lausanne, Switzerland
Email: {elsa.rizk, ali.sayed}@epfl.ch

***Abstract*—In network semi-supervised learning problems, only a subset of the network nodes is able to access the data labeling. This paper formulates a decentralized optimization problem where agents have individual decision rules to estimate, subject to the condition that neighboring agents (classifiers) are more likely to have similar labels. To promote such relationships, we propose to add to the aggregate sum of individual costs a graph regularization term that allows to penalize the differences between the labels at neighboring agents. Streaming data is assumed, and therefore, the *stochastic* (sub-)gradient method is used to solve the regularized problem. We provide conditions that guarantee the stability and convergence of the proposed algorithm. Simulation results show that collaboration among neighboring agents leads to better classification results by decreasing the probability of error and by improving the convergence rate.**

***Index Terms*—Decentralized learning and classification, semi-supervised learning over graphs, graph regularization.**

I. Introduction

Decentralized inference over graphs has received considerable attention over the past two decades. With some exceptions, the large majority of these works focuses on *consensus* and *diffusion* optimization by considering variations of the standard optimization problem [1]–[3]:

$$w^o = \underset{w}{\operatorname{argmin}} \sum_{k=1}^{N} J_k(w), \tag{1}$$

where N is the total number of nodes in the network and $J_k(w)$, which depends on an M-dimensional parameter vector $w \in \mathbb{R}^M$, represents a private cost function at agent k. Assuming that agents can communicate over a network represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, N\}$ denotes the set of nodes and $\mathcal{E}$ denotes the set of edges, the agents in a decentralized learning setting seek to estimate w^o in (1) collaboratively by performing local computations and by exchanging estimates with their neighbors. The neighborhood of agent k is denoted by $\mathcal{N}_k$; it consists of all agents that are connected to k by an edge.

Unlike traditional distributed environments, devices (also referred to as agents or clients) in modern machine learning applications generate data in a highly heterogeneous and distributed manner (since the data on a given device is based on the usage of the device by a particular client). Such heterogeneous statistical networks require more complex models and flexible algorithms than single-task implementations since their agents may need to estimate and track distinct, though related, tasks (or objectives) simultaneously. Previous efforts in this direction attempted to solve *variations* of the following regularized multitask learning formulation [4]:

$$\mathcal{W}^\star = \arg\min_{\mathcal{W}} \sum_{k=1}^{N} J_k(w_k) + \eta \mathcal{R}(w_1, \ldots, w_N), \tag{2}$$

where $\mathcal{W} = \text{col}\{w_1, \ldots, w_N\}$ denotes the collection of parameter vectors from across the network, $w_k \in \mathbb{R}^M$ is the parameter vector or the task at agent k, $\eta > 0$ is a regularization strength, and $\mathcal{R}(\cdot)$ is a regularization function promoting the relationships between the tasks. Examples of multitask regularizers include graph Laplacian regularization to promote graph signal smoothness [5], [6], network Lasso regularizer to promote graph clustering [7], and ℓ_1-norm sparsity-based regularizers to promote graph piecewise constant transitions [8], [9].

In this paper, we focus on decentralized learning over heterogeneous statistical graphs–see Fig. 1. We assume that at each time instant i, agent k is collecting an $M_k \times 1$ feature vector $\boldsymbol{h}_{k,i} \in \mathbb{R}^{M_k}$, which corresponds to a collection of observed attributes in a classification problem. We are interested in a heterogeneous system setting as in [10], [11] where agents might have access to different types and numbers of attributes or features. For instance, in weather sensor networks, some sensors might observe measurements related to wind speed, temperature, etc., while others might not have access to wind speed measurements or might observe another set of features. Such heterogeneities can lead to "different" learning abilities across the network. In settings where some agents observe a large number of (relevant) data while others observe a small number of (relevant) features, agents with limited learning abilities can benefit considerably from cooperating with their neighbors since they are more likely to observe similar labels. For instance, in weather forecasting applications [5], if it is raining in a given city on a given day, it is more likely

The work of R. Nassif was supported in part by ANR JCJC grant ANR-22-CE23-0015-01 (CEDRO project) and AUB URB grant.

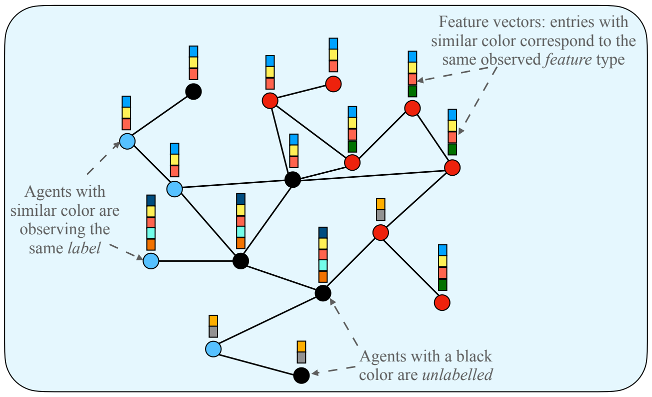
Feature vectors: entries with similar color correspond to the same observed *feature* type
Agents with similar color are observing the same *label*
Agents with a black color are *unlabelled*

Fig. 1: Illustration of the various sources of heterogeneity in the data acquisition processes.

that rain is also occurring in adjacent cities on this same day. However, problem formulation (2) might not lead to a meaningful cooperation rule since, in the heterogeneous setting of Fig. 1, instead of promoting the relationships between the tasks $\{w_k\}$, we need to promote the relationships between the labels $\{\gamma_k\}$. Besides feature types and sizes, we assume another source of heterogeneity in the data acquisition process. Particularly, we assume that some agents might have access to unlabeled data points preventing them from learning their classification rules at a given time instant. This scenario arises in applications where labeling at some agents is costly and requires human assistance, or in applications where some agents are not willing to use their own labels in the training process due to privacy concerns.

The problem of labeling a partially labeled graph has already been considered in the machine learning community, with the so-called *semi-supervised learning on graphs* [12]–[15]. Within this community, the graph represents the similarities between data points or feature vectors, and closer data points tend to have similar class labels. Several centralized strategies have been derived under the assumption that all data are available beforehand. In [12] and [13], graph Laplacian regularization is used to solve the partially labeled datasets classification problem, while in [14] a label propagation algorithm is derived. Non-smooth graph-based regularization is used in [16] to encourage the sparsity of the differences between the log odd ratios at neighboring nodes. In settings where agents have access to streaming data and are only allowed to exchange information with other agents over a communication graph, new strategies for jointly learning the agents decision rules and performing label propagation must be developed.

This paper focuses on decentralized semi-supervised multitask learning in streaming and heterogeneous data acquisition settings (as illustrated in Fig. 1). Each agent k is interested in estimating its own decision rule w_k^o under the prior information that labels $\{\gamma_k\}$ at neighboring agents are more likely to be the same. To promote the labels' relationships, we propose to add to the cost function $J_k(w_k)$ a regularization term consisting of a weighted sum of ℓ_1-norm (or squared ℓ_2-norm) of the differences between the labels. The proposed learning algorithm is presented in Sec. II and its stability is studied in Sec. III. The simulation results are presented and discussed in Sec. IV. The conclusion is presented in Sec. V.

II. Decentralized semi-supervised multitask learning

In this work, we assume that each agent k is interested in solving an online binary classification problem. We first assume the supervised setting where, at every time instant i, agent k is observing a feature vector $\boldsymbol{h}_{k,i} \in \mathbb{R}^{M_k}$ and the corresponding class label $\boldsymbol{\gamma}_k(i) = \pm 1$. The objective at agent k is to construct a classifier to predict the label $\boldsymbol{\gamma}_k$ based on the knowledge of the feature vector $\boldsymbol{h}_k$. To that end, agent k can in principle use a logistic regression machine [2], [17]–[19] that seeks an $M_k \times 1$ vector w_k^o, such that the predicted label is $\widehat{\boldsymbol{\gamma}}_k(i) = \text{sign}(\boldsymbol{h}_{k,i}^\top w_k^o)$ and

$$w_k^o \triangleq \arg\min_{w_k} J_k(w_k), \tag{3}$$

with

$$J_k(w_k) \triangleq \mathbb{E} \ln\left(1 + e^{-\boldsymbol{\gamma}_k(i)\boldsymbol{h}_{k,i}^\top w_k}\right) + \frac{\rho}{2}\|w_k\|^2, \tag{4}$$

where the expectation is computed over the distribution of the random data $\{\boldsymbol{h}_{k,i}, \boldsymbol{\gamma}_k(i)\}$ and where ρ is a positive regularization parameter. Note that, while logistic regression costs are considered to illustrate the semi-supervised multitask learning problem, the decentralized approach developed in the section can be applied to a wide class of individual costs. For generalization purposes, we conduct in Sec. III a mean-square-error analysis of the proposed approach for general costs satisfying Assumption 2 further ahead. We let A denote the $N \times N$ symmetric adjacency matrix, which describes the graph structure and the connections. The (k, ℓ)-th entry $a_{k\ell} \geq 0$ of A reflects the strength of the relation between nodes k and ℓ. For instance, if node ℓ is connected to node k (i.e., $\ell \in \mathcal{N}_k$) and is more likely to observe the same label as node k, then the weights $a_{k\ell}$ and $a_{\ell k}$ are more likely to be large. If there is no edge connecting nodes k and ℓ, then $a_{k\ell} = 0$.

In heterogeneous data acquisition settings (such as the ones depicted in Fig. 1), agents with a small number of (relevant) features are probably not able to make decisions on their own. Additionally, even if the number of observed features is enough, some agents may be affected by noise, and others may not have access to their true labels, preventing them from properly estimating their decision rules. These facts motivate us to seek a meaningful cooperation rule that allows agents with limited learning abilities to benefit from the learning process of their neighbors. To that end, two regularization functions will be investigated in the following.

Label Lasso regularization: If we let:

$$\widehat{\boldsymbol{\Gamma}}_i = \text{col}\{\widehat{\boldsymbol{\gamma}}_1(i), \ldots, \widehat{\boldsymbol{\gamma}}_N(i)\}, \tag{5}$$

denote the collection of predicted labels from across the network, we can derive the cooperation rule by formulating a regularized multitask optimization problem that employs the total variation of $\widehat{\mathbf{\Gamma}}_i$ [20]:

$$\text{TV} = \sum_{(k,\ell)\in\mathcal{E}} a_{k\ell}\mathbb{E}|\widehat{\gamma}_k(i) - \widehat{\gamma}_\ell(i)|, \quad (6)$$

as a regularizer, where the expectation is computed over the distribution of the random variables $\{\widehat{\gamma}_k(i)\}$. In settings where neighboring nodes are more likely to be observing similar labels, a small total variation is expected over the graph. Intuitively, given that the weights are non-negative, the total variation (6) is expected to be small if nodes with a large weight $a_{k\ell}$ on the edge connecting them have similar label values $\{\widehat{\gamma}_k(i), \widehat{\gamma}_\ell(i)\}$. The non-smooth regularizer (6) is suitable for clustered-network applications where agents are decomposed into clusters, and within each cluster, agents are observing the same label.

Graph Laplacian regularization: This regularizer is suitable for applications where the smoothness of the signal w.r.t. the underlying graph must be promoted. To that end, if we let $L = \text{diag}\{A\mathbb{1}_N\} - A$ denote the Laplacian matrix, then the smoothness of the graph signal $\widehat{\mathbf{\Gamma}}_i$ can be measured in terms of the quadratic form of the graph Laplacian [5], [21]:

$$\mathcal{S} = \mathbb{E}\left[\widehat{\mathbf{\Gamma}}_i^\top L\widehat{\mathbf{\Gamma}}_i\right] = \frac{1}{2}\sum_{(k,\ell)\in\mathcal{E}} a_{k\ell}\mathbb{E}\left(\widehat{\gamma}_k(i) - \widehat{\gamma}_\ell(i)\right)^2, \quad (7)$$

The smaller $\mathcal{S}$ is, the smoother $\widehat{\mathbf{\Gamma}}_i$ on the graph is.

Now given the above possibilities for regularization, and motivated by the previous discussion, we propose to solve the following optimization problem at each agent k:

$$w_k^\star \triangleq \arg\min_{w_k} J_k(w_k) + r_k\big(w_k, \{w_\ell\}\big), \quad (8)$$

with

$$r_k\big(w_k, \{w_\ell\}\big) \triangleq \eta \sum_{\ell\in\mathcal{N}_k} a_{k\ell}\mathbb{E}f(\widehat{\gamma}_k(i) - \widehat{\gamma}_\ell(i)), \quad (9)$$

where the expectation is computed over the distribution of the random variables $\{\widehat{\gamma}_k(i)\}$ and where η is a positive regularization parameter that ensures a tradeoff between fidelity to the measurements and the prior information on the relationships between the labels. The function $f : \mathbb{R} \to \mathbb{R}$ can be selected as the absolute value function $f(x) = |x|$ in sparsity promoting settings such as (6) and as the quadratic function $f(x) = \frac{1}{2}x^2$ in smoothness promoting settings such as (7). Since the function $f(\cdot)$ can be non-differentiable, we shall employ the sub-gradient approach [19], [22] to solve problem (8). When logistic regression is employed, the predicted label is expressed as $\widehat{\gamma}_k(i) = \text{sign}(\boldsymbol{h}_{k,i}^\top w_k)$. For mathematical tractability, we shall replace the non-smooth $\text{sign}(\cdot)$ function by a smooth approximation given by the hyperbolic tangent function $\tanh(\cdot)$ [23]. Therefore, instead of solving (8), agent k solves the following optimization problem:

$$\arg\min_{w_k} J_k(w_k) + \widetilde{r}_k\big(w_k, \{w_\ell\}\big), \quad (10)$$

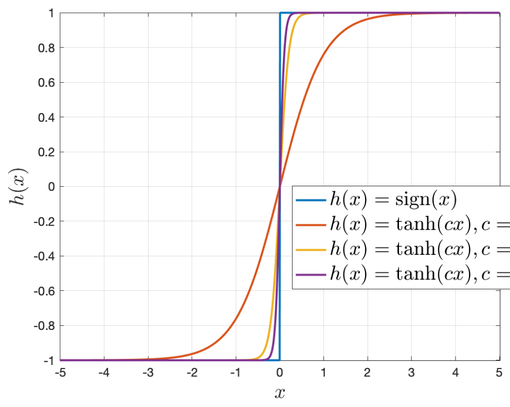

Fig. 2: Sign function and its hyperbolic tangent approximations.

where:

$$\widetilde{r}_k\big(w_k, \{w_\ell\}\big) \triangleq \eta\sum_{\ell\in\mathcal{N}_k} a_{k\ell}\mathbb{E}f(\tanh(c\boldsymbol{h}_{k,i}^\top w_k) - \tanh(c\boldsymbol{h}_{\ell,i}^\top w_\ell)), \quad (11)$$

and the constant c controls the slope of the $\tanh(\cdot)$ function–see Fig. 2.

Since the distribution of the random data $\{\boldsymbol{h}_{k,i}, \boldsymbol{\gamma}_k(i)\}$ is unknown, agent k will need to learn directly from the observed data by replacing the true gradient (and subgradient) vectors by their stochastic approximations. By using the gradient (and sub-gradient) vectors of the loss functions as an approximation, we arrive at Algorithm 1 for solving (10). Agent k starts with $w_{k,-1}$, an initial random guess for w_k^o in (3). The semi-supervised setting is taken into account through the binary variable $\boldsymbol{\epsilon}_k(i)$ that is equal to 1 if agent k is able to access its true label at iteration i, and 0 otherwise. At every iteration i, agent k receives the feature vector $\boldsymbol{h}_{k,i}$. It also receives the class label $\boldsymbol{\gamma}_k(i)$ when $\boldsymbol{\epsilon}_k(i) = 1$. Then, it performs two steps–see Algorithm 1. In the first step (15), which is referred to as the *self-learning* step, agent k updates its estimate $\boldsymbol{w}_{k,i-1}$ by stepping in the opposite direction of the stochastic approximation for the gradient of the risk $J_k(\cdot)$ defined in (4), scaled by a small positive step-size μ [2]. We employ the following approximation:

$$\widehat{\nabla_{w_k} J}_k(w_k) = -\frac{\boldsymbol{\gamma}_k(i)\boldsymbol{h}_{k,i}\, e^{-\boldsymbol{\gamma}_k(i)\boldsymbol{h}_{k,i}^\top w_k}}{1 + e^{-\boldsymbol{\gamma}_k(i)\boldsymbol{h}_{k,i}^\top w_k}} + \rho w_k. \quad (12)$$

Note that the variable $\boldsymbol{\epsilon}_k(i)$ is added to this step since the gradient approximation cannot be computed when the label is not observed (i.e., when $\boldsymbol{\epsilon}_k(i) = 0$). In the second step (16), which is referred to as the *social learning* step, agent k collaborates with its neighbors $\ell \in \mathcal{N}_k$ by stepping in the opposite direction of the stochastic approximation of the subgradient (or gradient) vector of the regularizer $\widetilde{r}_k\big(\cdot, \{\boldsymbol{\psi}_{\ell,i}\}\big)$ defined in (11). Note that, in the sparse case (i.e., when $f(x) = |x|$), we employ the

Algorithm 1: Logistic semi-supervised learning over multitask graphs

```
Initialize: $w_{k,-1}$ for every agent $k$;
for every iteration $i \geq 0$ do
    for every agent $k$ do
        collect the feature vector $\boldsymbol{h}_{k,i}$;
        set $\boldsymbol{\epsilon}_k(i)$ to 1 if a label $\boldsymbol{\gamma}_k(i)$ is available, and to 0 otherwise;
            $\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} - \mu\boldsymbol{\epsilon}_k(i)\widehat{\nabla_{w_k} J}_k(\boldsymbol{w}_{k,i-1})$   (15)
            $\boldsymbol{w}_{k,i} = \boldsymbol{\psi}_{k,i} - \mu\,\widehat{\partial_{w_k}\widetilde{r}_k}\big(\boldsymbol{\psi}_{k,i},\{\boldsymbol{\psi}_{\ell,i}\}_{\ell\in\mathcal{N}_k}\big)$   (16)
        where in the sparse case:
            $\widehat{\partial_{w_k}\widetilde{r}_k}\left(\boldsymbol{\psi}_{k,i},\{\boldsymbol{\psi}_{\ell,i}\}_{\ell\in\mathcal{N}_k}\right) = \eta\sum_{\ell\in\mathcal{N}_k} a_{k\ell}c\boldsymbol{h}_{k,i}\left(1-\tanh^2\left(c\boldsymbol{h}_{k,i}^\top\boldsymbol{\psi}_{k,i}\right)\right)\times \text{sign}\left(\tanh\left(c\boldsymbol{h}_{k,i}^\top\boldsymbol{\psi}_{k,i}\right)-\tanh\left(c\boldsymbol{h}_{\ell,i}^\top\boldsymbol{\psi}_{\ell,i}\right)\right)$   (17)
        and in the smooth case:
            $\widehat{\partial_{w_k}\widetilde{r}_k}\left(\boldsymbol{\psi}_{k,i},\{\boldsymbol{\psi}_{\ell,i}\}_{\ell\in\mathcal{N}_k}\right) = \eta\sum_{\ell\in\mathcal{N}_k} a_{k\ell}c\boldsymbol{h}_{k,i}\left(1-\tanh^2\left(c\boldsymbol{h}_{k,i}^\top\boldsymbol{\psi}_{k,i}\right)\right)\times \left(\tanh\left(c\boldsymbol{h}_{k,i}^\top\boldsymbol{\psi}_{k,i}\right)-\tanh\left(c\boldsymbol{h}_{\ell,i}^\top\boldsymbol{\psi}_{\ell,i}\right)\right)$   (18)
    end
end
```

following subgradient approximation [24, p. 42]:

$$\widehat{\partial_{w_k}\widetilde{r}_k}(w_k,w_\ell)=\eta\sum_{\ell\in\mathcal{N}_k} a_{k\ell}c\boldsymbol{h}_{k,i}\left(1-\left(\tanh\left(c\boldsymbol{h}_{k,i}^\top w_k\right)\right)^2\right)\times \text{sign}\Big(\tanh\left(c\boldsymbol{h}_{k,i}^\top w_k\right)-\tanh\left(c\boldsymbol{h}_{\ell,i}^\top w_\ell\right)\Big), \tag{13}$$

which results in the combination step (17) in Alg. 1, and in the smooth case (i.e., when $f(x)=\frac{1}{2}x^2$), we employ the following gradient approximation:

$$\widehat{\partial_{w_k}\widetilde{r}_k}(w_k,w_\ell)=\eta\sum_{\ell\in\mathcal{N}_k} a_{k\ell}c\boldsymbol{h}_{k,i}\left(1-\left(\tanh\left(c\boldsymbol{h}_{k,i}^\top w_k\right)\right)^2\right)\times \Big(\tanh\left(c\boldsymbol{h}_{k,i}^\top w_k\right)-\tanh\left(c\boldsymbol{h}_{\ell,i}^\top w_\ell\right)\Big), \tag{14}$$

which results in the combination step (18) in Alg. 1. According to (17) and (18), agent k needs to collect from its neighbors $\ell\in\mathcal{N}_k$ the *scalar* values $\big\{\tanh\left(c\boldsymbol{h}_{\ell,i}^\top\boldsymbol{\psi}_{\ell,i}\right)\big\}$ to perform step (16). These values can be interpreted as the intermediate predictions of the labels at neighboring agents.

Before proceeding, it should be noted that graph regularization was previously considered in [10] to promote the labels relationships in a decentralized context. The current work, however, differs from [10] in four ways. First, the graph Laplacian regularization (7) was applied in [10] to the inner products $\{\boldsymbol{h}_{k,i}^\top w_k\}$ instead of the *predicted labels* $\{\widehat{\boldsymbol{\gamma}}_k(i)\approx\tanh(c\boldsymbol{h}_{k,i}^\top w_k)\}$. Second, the *semi-supervised* setting was not considered in [10] where it was assumed that all agents have access to their true labels during the learning process. Third, the *non-smooth* network Lasso regularizer was not considered in [10]. Finally, instead of using the rule $\text{sign}(\boldsymbol{h}_{k,i}^\top\boldsymbol{w}_{k,i})$ for *label prediction*, the approach in [10] proposed to set the final label as the average of the neighborhood predicted labels.

III. Stability analysis

We will now examine the mean-square-error stability of Alg. 1 under the following assumption on the random variable $\boldsymbol{\epsilon}_k(i)$.

Assumption 1 *The (semi-supervised modeling) variable $\boldsymbol{\epsilon}_k(i)$ is assumed to be Bernoulli randomly distributed, i.e.:*

$$\boldsymbol{\epsilon}_k(i)=\begin{cases}1, & \textit{with probability}\, q_k,\\ 0, & \textit{with probability}\, 1-q_k,\end{cases} \tag{19}$$

with $0<q_k\leq 1$. It is also assumed that $\boldsymbol{\epsilon}_k(i)$ is independent of all the other random mechanisms in the networked system.

For each node k, we define the error vector $\widetilde{\boldsymbol{w}}_{k,i}$ and the intermediate error vector $\widetilde{\boldsymbol{\psi}}_{k,i}$ as

$$\widetilde{\boldsymbol{w}}_{k,i}\triangleq w_k^\circ-\boldsymbol{w}_{k,i},\qquad \widetilde{\boldsymbol{\psi}}_{k,i}\triangleq w_k^\circ-\boldsymbol{\psi}_{k,i}, \tag{20}$$

respectively. We also define the gradient noise vector $\boldsymbol{s}_{k,i}(\cdot)$ as the difference between the true gradient and its approximation at iteration i, namely,

$$\boldsymbol{s}_{k,i}(w)=\nabla_{w_k}J_k(w)-\widehat{\nabla_{w_k}J}_k(w). \tag{21}$$

Observe that the gradient approximation can be expressed as:

$$\widehat{\nabla_{w_k}J}_k(\boldsymbol{w}_{k,i-1})=\nabla_{w_k}J_k(\boldsymbol{w}_{k,i-1})-\boldsymbol{s}_{k,i}(\boldsymbol{w}_{k,i-1}). \tag{22}$$

Now, by using the mean-value theorem for real arguments [2], we can write:

$$\nabla_{w_k}J_k(\boldsymbol{w}_{k,i-1})-\nabla_{w_k}J_k(w_k^\circ)=-\boldsymbol{H}_{k,i-1}\widetilde{\boldsymbol{w}}_{k,i-1}, \tag{23}$$

where

$$\boldsymbol{H}_{k,i-1}\triangleq\int_0^1\nabla_{w_k}^2J_k(w_k^\circ-t\widetilde{\boldsymbol{w}}_{k,i-1})dt. \tag{24}$$

By combining (22) and (23), and by subtracting w_k° from both sides of (15) and (16), we obtain:

$$\widetilde{\boldsymbol{\psi}}_{k,i}=(I_{M_k}-\mu\boldsymbol{\epsilon}_k(i)\boldsymbol{H}_{k,i-1})\widetilde{\boldsymbol{w}}_{k,i-1}+\mu\boldsymbol{\epsilon}_k(i)\nabla_{w_k}J_k(w_k^\circ) -\mu\boldsymbol{\epsilon}_k(i)\boldsymbol{s}_{k,i}(\boldsymbol{w}_{k,i-1}), \tag{25}$$

$$\widetilde{\boldsymbol{w}}_{k,i}=\widetilde{\boldsymbol{\psi}}_{k,i}+\mu\widehat{\partial_{w_k}\widetilde{r}_k}\big(\boldsymbol{\psi}_{k,i},\{\boldsymbol{\psi}_{\ell,i}\}\big). \tag{26}$$

Let $\widetilde{\boldsymbol{w}}_i=\text{col}\{\widetilde{\boldsymbol{w}}_{k,i}\}_{k=1}^N$ denote the network error vector. Substituting (25) into (26), we get

$$\widetilde{\boldsymbol{w}}_i=\boldsymbol{\mathcal{B}}_{i-1}\widetilde{\boldsymbol{w}}_{i-1}+\mu\,\boldsymbol{\mathcal{E}}_ib-\mu\,\boldsymbol{\mathcal{E}}_i\boldsymbol{s}_i+\mu\boldsymbol{r}_i, \tag{27}$$

where

$$\mathcal{H}_{i-1} \triangleq \text{diag}\{\boldsymbol{H}_{k,i-1}\}_{k=1}^{N}, \quad (28)$$

$$\mathcal{B}_{i-1} \triangleq I - \mu\,\mathcal{E}_i\mathcal{H}_{i-1}, \quad (29)$$

$$b \triangleq \text{col}\{\nabla_{w_k} J_k(w_k^{\circ})\}_{k=1}^{N}, \quad (30)$$

$$\mathcal{E}_i \triangleq \text{diag}\{\boldsymbol{\epsilon}_k(i) I_{M_k}\}_{k=1}^{N}, \quad (31)$$

$$\boldsymbol{s}_i \triangleq \text{col}\{\boldsymbol{s}_{k,i}(\boldsymbol{w}_{k,i-1})\}_{k=1}^{N}, \quad (32)$$

$$\boldsymbol{r}_i \triangleq \text{col}\left\{\widehat{\partial_{w_k}\widetilde{r}_k}(\boldsymbol{\psi}_{k,i}, \{\boldsymbol{\psi}_{\ell,i}\})\right\}_{k=1}^{N}. \quad (33)$$

In the following, and for generalization purposes, we analyze Alg. 1 for twice differentiable costs $\{J_k(\cdot)\}$ and gradient noise processes $\{\boldsymbol{s}_{k,i}(\cdot)\}$ satisfying the following assumptions, which are commonly used in the literature [2]. It can be shown that these assumptions are satisfied by the logistic regression costs in (4) and by the gradient approximation (12).

Assumption 2 *The Hessian matrix function $\nabla^2_{w_k} J_k(w_k)$ is bounded from below and above as follows:*

$$0 < \lambda_{k,\min} I_{M_k} \leq \nabla^2_{w_k} J_k(w_k) \leq \lambda_{k,\max} I_{M_k}, \quad (34)$$

where $\lambda_{k,\min}$ and $\lambda_{k,\max}$ are the smallest and largest eigenvalues of the Hessian matrix at agent k, respectively.

Assumption 3 *The gradient noise process defined in (21) is assumed to satisfy the following conditions for $1 \leq k \leq N$:*

$$\mathbb{E}[\boldsymbol{s}_{k,i}(\boldsymbol{w}_k) \mid \boldsymbol{\mathcal{F}}_{i-1}] = 0, \quad (35)$$

$$\mathbb{E}[\|\boldsymbol{s}_{k,i}(\boldsymbol{w}_k)\|^2 \mid \boldsymbol{\mathcal{F}}_{i-1}] \leq \beta_k^2\|\boldsymbol{w}_k\|^2 + \sigma^2_{s,k}, \quad (36)$$

for some $\beta_k^2 \geq 0$, $\sigma^2_{s,k} \geq 0$, and where $\boldsymbol{\mathcal{F}}_{i-1}$ denotes the collection of the past iterates $\{\boldsymbol{w}_{k,j} \mid \forall\, k = 1, ..., N \text{ and } j \leq i-1\}$.

Theorem 1 *Assuming that the components of the feature vectors $\boldsymbol{h}_{k,i}$ are bounded, and under Assumptions 1, 2, and 3, if the step-size parameter μ satisfies:*

$$\mu < \frac{2\min_{1\leq k\leq N} q_k\lambda_k}{\left(\min_{1\leq k\leq N} q_k\lambda_k\right)^2 + \beta^2_{q,\max}}, \quad (37)$$

where:

$$\lambda_k \triangleq \begin{cases} \lambda_{k,\min}, & \text{if } |1-\mu q_k\lambda_{k,\min}| \geq |1-\mu q_k\lambda_{k,\max}| \\ \lambda_{k,\max}, & \text{otherwise} \end{cases}, \quad (38)$$

$$\beta^2_{q,\max} \triangleq \max_{1\leq k\leq N} q_k\overline{\beta}_k^2, \quad (39)$$

with $\overline{\beta}_k^2 = 2\beta_k^2$, then in the limit it holds that:

$$\limsup_{i\to\infty}\ \mathbb{E}\|\widetilde{\boldsymbol{w}}_i\|^2 \leq \mathcal{O}(\mu) + \mathcal{O}(\mu\eta^2) + \mathcal{O}(\eta). \quad (40)$$

That is, for large i, and for a small enough μ, Alg. 1 is stable and converges in the mean-square-error sense.

Proof. The proof is omitted due to space limitations. The arguments are along the lines developed in [2], [5], [8], [25] for multitask [5], [8] and single-task [2], [25] learning with proper adjustments to handle the semi-supervised scenario. ■

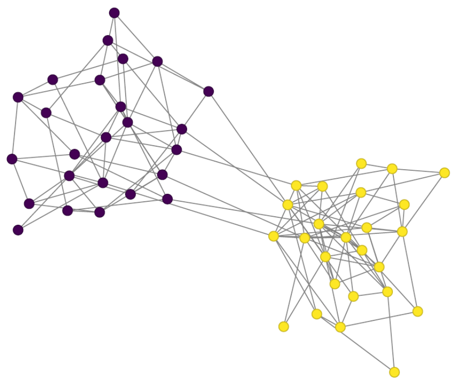
Fig. 3: Clustered network structure (agents with the same color observe the same label).

IV. Experimental results

A. Synthetic data experiments

Two experiments are conducted with the network structure generated using the Stochastic Block Model [26], and the graph parameters chosen such that $N = 50$ nodes, divided equally into two clusters. The probability of connection between nodes in the same cluster is set to 0.2, while the probability of inter-cluster connection is 0.01. The edge weights are random numbers, where the intra-cluster edge weights follow a normal distribution $\mathcal{N}(2, 0.2)$ while the inter-cluster edge weights are small random numbers between 0 and 0.001. The resulting network is illustrated in Fig. 3. At every iteration, the true labels $\boldsymbol{\gamma}_k(i)$ are generated by a random choice between 1 and -1 for agents in the first cluster, and the opposite label is assigned for agents in the second cluster. Agents observe different numbers of features with the number of features M_k at agent k chosen randomly from a discrete set of values ranging between 1 and 5. The m-th entry of the feature vector $\boldsymbol{h}_{k,i}$ is generated according to the following equation:

$$[\boldsymbol{h}_{k,i}]_m = \boldsymbol{\gamma}_k(i).\boldsymbol{e}_k(i) + \boldsymbol{v}_k(i), \quad (41)$$

where $\boldsymbol{e}_k(i)$ and $\boldsymbol{v}_k(i)$ are randomly generated from the Gaussian distributions $\mathcal{N}(m, 0.5)$ and $\mathcal{N}(0, 1)$, respectively. We set μ, c, and ρ equal to 0.01, 10, and 0.05, respectively.

Experiment 1: This experiment is conducted by assuming that, within each cluster, only one node is observing a label (i.e., its probability q_k is set to 1, while the probability of the remaining nodes is 0). Every agent k is tested on a separate validation set of $J = 100$ samples at every iteration. Figure 4 reports the validation loss calculated according to:

$$\text{VL}(i) = \frac{1}{2N}\sum_{k=1}^{N}\left(\frac{1}{J}\sum_{j=1}^{J}|\boldsymbol{\gamma}_k(j) - \text{sign}(\boldsymbol{h}_{k,j}^\top\boldsymbol{w}_{k,i})|\right). \quad (42)$$

The results are averaged over 20 Monte-Carlo runs. For $\eta = 0$ (non-cooperative case), the validation error remains constant at approximately 0.48 during the learning process due to the fact that uninformed nodes are not able to learn on their own

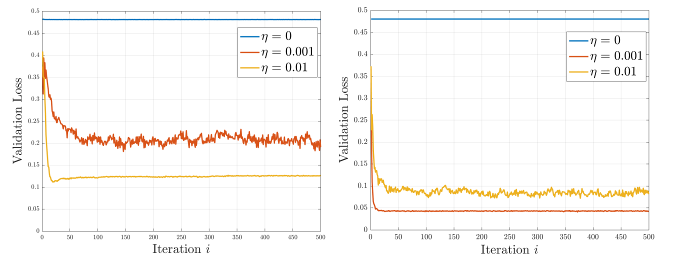

Fig. 4: Experiment 1. *(Left)* Sparsity promoting regularizer ($f(x) = |x|$). *(Right)* Smooth graph regularizer ($f(x) = \frac{1}{2}x^2$).

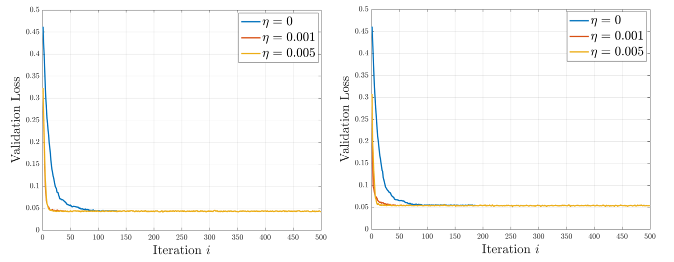

Fig. 5: Experiment 2. *(Left)* Sparsity promoting regularizer ($f(x) = |x|$). *(Right)* Smooth graph regularizer ($f(x) = \frac{1}{2}x^2$).

since their true labels are not accessible. However, through cooperation (by setting $\eta = \{0.001, 0.01\}$), the validation error decreases considerably in both, ℓ_1- and squared ℓ_2-norm, settings.

Experiment 2: This experiment is conducted by assuming that each agent in the network is observing its true label with a probability $q_k = 0.1$. The results are shown in Fig. 5. In a random sampling setting, no significant difference is observed between the steady-state validation losses of the non-cooperative ($\eta = 0$) and the cooperative ($\eta = \{0.001, 0.01\}$) implementations. Nevertheless, a key observation is that co-operation improves the convergence rate.

B. Real data experiments

We test Alg. 1 on the weather dataset considered in [5]. The dataset corresponds to a collection of daily measurements (mean temperature, mean wind speed, rain or snow occurence, etc.) taken from 2004 to 2017 at 139 weather stations located around the continental United States. The objective at each weather station is to predict whether it will rain (or snow) or not based on five daily collected measurements (mean temperature, mean dew point, mean visibility, mean wind speed, and maximum sustained wind speed). We construct an undirected weighted graph of $N = 139$ nodes by using the same approach as in [5]. To model the heterogeneous setting, the feature vector $\boldsymbol{h}_{k,i}$ at agent k is either composed of $M_k = 5$ entries corresponding to the five daily collected measurements, or composed of $M_k = 4$ entries corresponding to four randomly selected measurements out of five. The label $\gamma_k(i)$ is equal to 1 if rain or snow happened at station k and day i, and -1 otherwise. The dataset is split into a training set used to learn the decision rule w_k^o, and a test set from which $\widehat{\gamma}_k(i)$ are generated for performance evaluation. The training set comprises daily weather data recorded at the stations in the interval $2004 - 2012$ (a total number of 3288 days) and the test set contains data recorded in the interval $2012 - 2017$ (a total number of $J = 1826$ days) [5]. The performance of the network classifiers is measured by evaluating the testing error defined as:

$$\frac{1}{2N}\sum_{k=1}^{N}\left(\frac{1}{J}\sum_{j=1}^{J}|\gamma_k(j) - \text{sign}(\boldsymbol{h}_{k,j}^\top \boldsymbol{w}_{k,\infty})|\right), \qquad (43)$$

where $\boldsymbol{w}_{k,\infty}$ is the average of the last 200 iterates generated by Alg. 1 at agent k [5]. We set μ, c, and ρ equal to 3×10^{-4}, 10, and 2×10^{-5}, respectively.

Experiment 1: This experiment is conducted by assuming that only 7 nodes out of 139 are observing a label (i.e., their probability q_k is set to 1, while the probability of the remaining nodes is set to 0) – see Fig. 6 for illustration. We report in Tables I and II the testing error (43) for several values of η when the sparsity (Table I) and the smoothness (Table II) promoting regularizers are used.

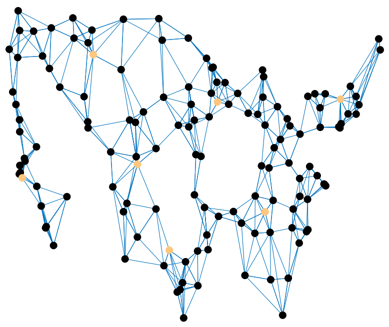

Fig. 6: Illustration of the network consisting of 139 weather stations. Nodes with a black color are unlabelled.

η	0	0.01	0.1	0.4	1	10
Testing error	0.4899	0.4667	0.4349	**0.3919**	0.3941	0.3943

TABLE I: Experiment 1: Testing error when the sparsity promoting regularizer ($f(x) = |x|$) is used.

η	0	0.01	0.1	0.4	1	10
Testing error	0.4899	**0.394**	0.3978	0.3948	0.3941	0.3946

TABLE II: Experiment 1: Testing error when the smooth graph regularizer ($f(x) = \frac{1}{2}x^2$) is used.

The results show that, by promoting cooperation ($\eta > 0$), the testing error decreases. In the sparse case, the smallest testing error is obtained when $\eta = 0.4$, and in the smooth case, the smallest testing error is obtained when $\eta = 0.01$.

Experiment 2: In this experiment, each agent observes its label with a probability $q_k > 0 \ \forall k$. The results obtained using the sparsity promoting regularizer are summarized in Table III for several values of q_k. In this experiment, the smallest testing error is also obtained in a cooperative setting ($\eta \neq 0$).

η	0	0.01	0.1	0.5	1
Testing error ($q_k = 0.1$)	0.3867	0.382	**0.3778**	0.3876	0.3824
Testing error ($q_k = 0.4$)	0.3799	0.3775	**0.3772**	0.3805	0.3796
Testing error ($q_k = 0.8$)	0.3756	**0.3689**	0.372	0.3767	0.3801

TABLE III: Experiment 2: Testing error when the sparsity promoting regularizer ($f(x) = |x|$) is used.

V. Conclusion

In this study, we proposed to solve a network semi-supervised classification problem by adding a graph regularization term to the logistic loss function. This additional term is represented by the ℓ_1- or squared ℓ_2-norm of the differences between the labels to encourage their similarities at neighboring agents. A stochastic (sub-)gradient approach was proposed and studied in the mean-square-error sense. Experimental results were presented to illustrate the effectiveness of the approach.

References

[1] D. P. Bertsekas, "A new class of incremental gradient methods for least squares problems," *SIAM J. Optim.*, vol. 7, no. 4, 1997.

[2] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Found. Trends Mach. Learn.*, vol. 7, no. 4-5, pp. 311–801, 2014.

[3] A. Nedić, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Automat. Contr.*, vol. 54, no. 1, pp. 48–61, 2009.

[4] R. Nassif, S. Vlaski, C. Richard, J. Chen, and A. H. Sayed, "Multitask learning over graphs: An approach for distributed, streaming machine learning," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 14–25, 2020.

[5] R. Nassif, S. Vlaski, C. Richard, and A. H. Sayed, "Learning over multitask graphs—Part I: Stability analysis," *IEEE Open Journal of Signal Processing*, vol. 1, pp. 28–45, 2020.

[6] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Trans. Signal Process.*, vol. 62, no. 16, pp. 4129–4144, 2014.

[7] D. Hallac, J. Leskovec, and S. Boyd, "Network LASSO: Clustering and optimization in large graphs," in *Proc. ACM SIGKDD*, Sydney, Australia, 2015, pp. 387–396.

[8] R. Nassif, C. Richard, A. Ferrari, and A. H. Sayed, "Proximal multitask learning over networks with sparsity-inducing coregularization," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6329–6344, 2016.

[9] Y. Sarcheshmehpour, M. Leinonen, and A. Jung, "Federated learning from big data over networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process*, Toronto, Canada, 2021, pp. 3055–3059.

[10] E. Rizk, R. Nassif, and A. H. Sayed, "Network classifiers with output smoothing," arXiv:1911.04870, Oct. 2019.

[11] B. Ying, K. Yuan, and A. H. Sayed, "Supervised learning under distributed features," *IEEE Transactions on Signal Processing*, vol. 67, no. 4, pp. 977–992, Feb. 2019.

[12] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and semi-supervised learning on large graphs," in *Proc. Conf. Learning Theory*, Banff, Canada, 2004, pp. 624–638.

[13] R. K. Ando and T. Zhang, "Learning on graph with laplacian regularization," in *Proc. Adv. Neural Inf. Process. Syst.*, Cambridge, MA, 2006, pp. 25–32.

[14] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Technical report, Carnegie Mellon University, 2002.

[15] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, British Columbia, Canada, 2003, pp. 321–328.

[16] H. Ambos, N. Tran, and A. Jung, "Classifying big data over networks via the logistic network lasso," in *Proc. Asilomar Conf. Signals Syst. Comput.*, Pacific Grove, CA, 2018, pp. 855–858.

[17] D. W. Hosmer and S. Lemeshow, *Applied Logistic Regression*, 2nd ed. Wiley, NJ, 2000.

[18] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 4th ed. Academic Press, 2008.

[19] A. H. Sayed, *Inference and Learning from Data*. 3 vols., Cambridge University Press, 2023.

[20] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, 1992.

[21] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Sig. Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[22] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Springer, 2004.

[23] D. Ciulin, "About sign function and some extensions," in *Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering*, E. Khaled, Ed. Springer, Dordrecht, 2008, pp. 148–153.

[24] F. H. Clarke, *Optimization and Nonsmooth Analysis*. Wiley New York, 1983.

[25] P. Di Lorenzo and A. H. Sayed, "Sparse distributed learning based on diffusion adaptation," *IEEE Trans. Signal Process.*, vol. 61, no. 6, pp. 1419–1433, 2013.

[26] E. Abbe, "Community detection and stochastic block models: Recent developments," *J. Mach. Learn. Res.*, vol. 18, no. 177, pp. 1–86, 2018.