Roula Nassif, Stefan Vlaski, Cédric Richard,
Jie Chen, and Ali H. Sayed

Multitask Learning Over Graphs

An approach for distributed, streaming machine learning

©ISTOCKPHOTO.COM/HAMSTER3D

Digital Object Identifier 10.1109/MSP.2020.2966273
Date of current version: 28 April 2020

The problem of simultaneously learning several related tasks has received considerable attention in several domains, especially in machine learning, with the so-called multitask learning (MTL) problem, or learning to learn problem [1], [2]. MTL is an approach to inductive transfer learning (using what is learned for one problem to assist with another problem), and it helps improve generalization performance relative to learning each task separately by using the domain information contained in the training signals of related tasks as an inductive bias. Several strategies have been derived within this community under the assumption that all data are available beforehand at a fusion center.

However, recent years have witnessed an increasing ability to collect data in a distributed and streamed manner. This requires the design of new strategies for jointly learning multiple tasks from streaming data over distributed (or networked) systems. This article provides an overview of multitask strategies for learning and adaptation over networks. The working hypothesis for these strategies is that agents are allowed to cooperate with each other to learn distinct, though related, tasks. The article shows how cooperation steers the network-limiting point and how different cooperation rules allow the promotion of different task-relatedness models. It also explains how and when cooperation over multitask networks outperforms noncooperative strategies.

Multitask network models

Consider a networked system consisting of a collection of N autonomous agents (sensors, classifiers, and so on) distributed over some geographic area and connected through a topology. The neighborhood of agent k is denoted by $\mathcal{N}_k$; it consists of all agents that are connected to k by an edge [Figure 1(a)]. A real-valued, strongly convex, and differentiable cost $J_k(w_k)$ is associated with each agent k. The objective (or the task) at agent k is to estimate the parameter vector, w_k^o, of size $M_k \times 1$ that minimizes $J_k(w_k)$, namely,

$$w_k^o \triangleq \operatorname*{argmin}_{w_k} J_k(w_k). \tag{1}$$

Depending on how the minimizers across the agents relate to each other, we distinguish between three categories of networks.

- *Single-task network*: All costs $J_k(w_k)$ are minimized at the same location w^o, particularly, $w_k^o = w^o$ for all k [Figure 1(a)].
- *Clustered multitask network*: The N agents are grouped into Q clusters $\mathcal{C}_q$ $(q = 1, \ldots, Q)$, and, within each cluster $\mathcal{C}_q$, all the costs are minimized at the same location $w_{\mathcal{C}_q}^o$; specifically, $w_k^o = w_{\mathcal{C}_q}^o$ for all $k \in \mathcal{C}_q$ [Figure 1(b)]. Similarities or relationships may exist among the distinct minimizers $\{w_{\mathcal{C}_q}^o\}$.
- *Multitask network*: The individual costs are minimized at distinct, though related, locations $\{w_k^o\}$ [Figure 1(c)].

Each agent k can solve (1) on its own. However, since the objectives across the network relate to each other, it is expected that by properly promoting these relationships, one may improve the network performance. In other words, it is expected that through cooperation among the agents, one may improve the network performance. One important question is how to design cooperative strategies that can lead to better performance versus noncooperative methods, where each agent attempts to determine w_k^o on its own. This article explains how MTL over graphs addresses this question.

Prior to MTL over graphs, there were many works in the machine learning literature that considered the simultaneous learning of multiple related tasks [1]–[7]. MTL has been shown, both empirically and theoretically, to improve performance relative to the traditional approach of learning each task separately. Several task-relatedness models are considered, depending on the machine learning application. For example, in [1] and [5], the functions to be learned are assumed to share a common underlying representation. In [6], it is assumed that the tasks are close to each other in some Hilbert space. Probability-based approaches, where a probability model capturing the relations between tasks is estimated simultaneously with functions corresponding to each task, have also been discussed [3]. In addition, graph-based approaches, where the relations between tasks are captured by an underlying graph, are also described in the literature [4], [7].

MTL has been shown, both empirically and theoretically, to improve performance relative to the traditional approach of learning each task separately.

All of these works assume that all data are available beforehand at a fusion center and propose batch-mode methods for solving multitask problems. Other existing works, such as [8], investigate a distributed data setting. However, most of these undertakings still require an architecture consisting of workers along with a master, where agents perform local computations and then send intermediate results to the master for further processing. Such solution methods are not fully distributed, which limits their range of practical applications.

This article focuses on fully distributed solutions that avoid the need for central data aggregation or processing and, instead, rely on local computations and communication exchanges among neighborhoods. In addition to providing distributed implementations, the solutions presented in this article learn continuously from streaming data. We begin by describing a class of noncooperative solutions that are able to respond in real time to streaming data. Then, we explain how these solutions can be extended to handle MTL over graphs.

Noncooperative learning under streaming data

Throughout this article, there is an explicit assumption that agents operate in the streaming data setting, that is, that each agent k receives at each time instant i one instantaneous realization $\boldsymbol{x}_{k,i}$ of a random data $\boldsymbol{x}_k$. The goal of agent k is to estimate the vector w_k^o that minimizes its risk function $J_k(w_k) \triangleq \mathbb{E}_{\boldsymbol{x}_k} Q_k(w_k; \boldsymbol{x}_k)$, defined in terms of some loss function $Q_k(\cdot)$. The expectation is computed over the distribution of the data $\boldsymbol{x}_k$. Agent k is particularly interested in solving the problem in the stochastic setting when the distribution of the data is generally unknown. This means that the risks $J_k(\cdot)$ and their gradients $\nabla_{w_k} J_k(\cdot)$ are unknown. As such, approximate gradient vectors $\widehat{\nabla_{w_k} J_k}(\cdot)$ will need to be employed,

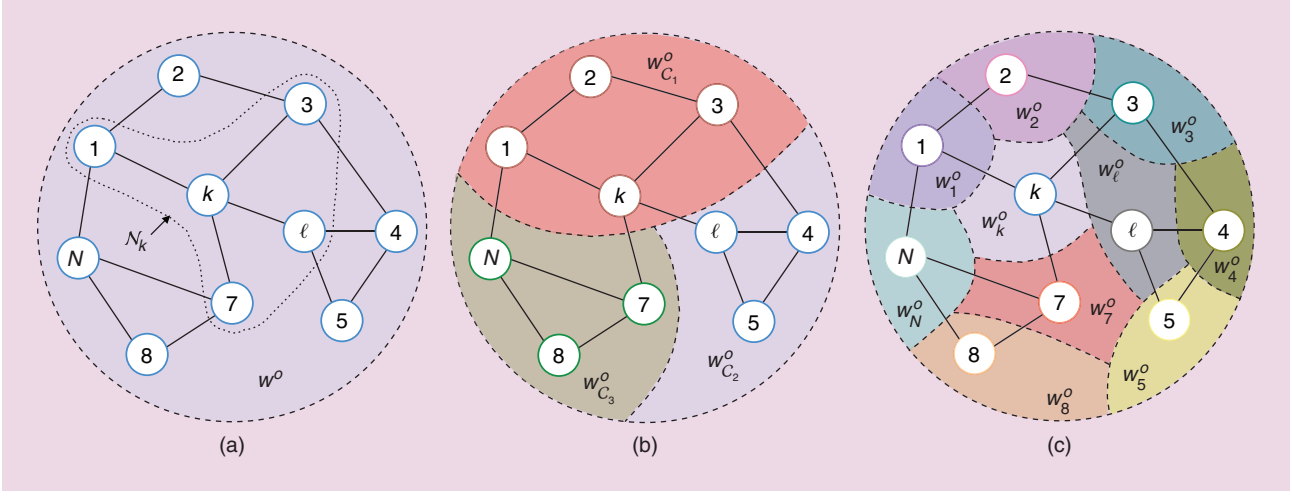

FIGURE 1. The network models: (a) single-task, (b) clustered multitask, and (c) multitask networks.

which leads to the following stochastic gradient algorithm for solving (1):

$$\boldsymbol{w}_{k,i} = \boldsymbol{w}_{k,i-1} - \mu\widehat{\nabla_{w_k} J_k}(\boldsymbol{w}_{k,i-1}), \tag{2}$$

where $\boldsymbol{w}_{k,i}$ is the estimate of w_k^o at iteration i and $\mu > 0$ is a small step-size parameter. Resorting to the instantaneous realization $\boldsymbol{x}_{k,i}$ of the random data $\boldsymbol{x}_k$, a common construction in the stochastic approximation theory is to employ the following gradient approximation at iteration i:

$$\widehat{\nabla_{w_k} J_k}(w_k) = \nabla_{w_k} Q_k(w_k; \boldsymbol{x}_{k,i}). \tag{3}$$

Therefore, we focus on stochastic gradient algorithms, which are powerful iterative procedures for solving (1) in the streaming data. They enable continuous learning and adaptation in response to drifts in the location of the minimizers due to changes in the costs. We illustrate (2) and (3) by considering scenarios from machine learning and adaptive filter theory.

Example 1: Logistic regression network

Let $\gamma_k(i) = \pm 1$ be a streaming sequence of (class) binary random variables, and let $\boldsymbol{h}_{k,i}$ be the corresponding streaming sequence of $M_k \times 1$ real random (feature) vectors with $R_{h,k} = \mathbb{E}\boldsymbol{h}_{k,i}\boldsymbol{h}_{k,i}^{\mathsf{T}} > 0$. The processes $\{\gamma_k(i), \boldsymbol{h}_{k,i}\}$ are assumed to be wide-sense stationary. In these problems, agent k seeks to estimate the vector w_k^o that minimizes the regularized logistic risk function [9]:

$$J_k(w_k) = \mathbb{E}\ln(1 + e^{-\gamma_k(i)\boldsymbol{h}_{k,i}^{\mathsf{T}} w_k}) + \frac{\rho}{2}\|w_k\|^2, \tag{4}$$

where $\rho > 0$ is a regularization parameter. Once w_k^o is found, $\hat{\gamma}_k(i) = \text{sign}(\boldsymbol{h}_{k,i}^{\mathsf{T}} w_k^o)$ can then be used as a decision rule to classify new features. Using approximation (3), we obtain the following stochastic-gradient algorithm for minimizing (4):

$$\boldsymbol{w}_{k,i} = (1 - \mu\rho)\boldsymbol{w}_{k,i-1} + \mu\gamma_k(i)\boldsymbol{h}_{k,i}\left(\frac{1}{1 + e^{\gamma_k(i)\boldsymbol{h}_{k,i}^{\mathsf{T}}\boldsymbol{w}_{k,i-1}}}\right). \tag{5}$$

□

Example 2: Mean-square-error network

In such networks, each agent is subjected to streaming data $\{\boldsymbol{d}_k(i), \boldsymbol{u}_{k,i}\}$ that are assumed to satisfy a linear regression model,

$$\boldsymbol{d}_k(i) = \boldsymbol{u}_{k,i}^{\mathsf{T}} w_k^o + \boldsymbol{v}_k(i), \tag{6}$$

for some unknown $M_k \times 1$ vector w_k^o to be estimated by agent k with $\boldsymbol{v}_k(i)$ denoting a zero-mean measurement noise. For these networks, the risk function takes the form of a mean-square-error (MSE) cost [10]:

$$J_k(w_k) = \frac{1}{2}\mathbb{E}(\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}^{\mathsf{T}} w_k)^2, \tag{7}$$

which is minimized at w_k^o. The processes $\{\boldsymbol{u}_{k,i}, \boldsymbol{v}_k(i)\}$ are zero-mean jointly wide-sense stationary with 1) $\mathbb{E}\boldsymbol{u}_{k,i}\boldsymbol{u}_{\ell,i}^{\mathsf{T}} = R_{u,k} > 0$ if $k = \ell$ and zero otherwise and 2) $\mathbb{E}\boldsymbol{v}_k(i)\boldsymbol{v}_\ell(i) = \sigma_{v,k}^2$ if $k = \ell$ and zero otherwise, and 3) $\boldsymbol{u}_{k,i}$ and $\boldsymbol{v}_k(j)$ are independent of each other. Using approximation (3), we obtain the following stochastic-gradient algorithm:

$$\boldsymbol{w}_{k,i} = \boldsymbol{w}_{k,i-1} + \mu\boldsymbol{u}_{k,i}(\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}^{\mathsf{T}}\boldsymbol{w}_{k,i-1}), \tag{8}$$

which is the well-known least-mean-squares algorithm [11]. □

The use of the approximate gradient $\widehat{\nabla_{wk} J_k}(\cdot)$ instead of the true gradient $\nabla_{wk} J_k(\cdot)$ in (2) introduces perturbations into the operation of the gradient descent iteration. This perturbation is referred to as the *gradient noise*, defined as $\boldsymbol{s}_{k,i}(w_k) \triangleq \nabla_{w_k} J_k(w_k) - \widehat{\nabla_{w_k} J_k}(w_k)$. The presence of this perturbation prevents the stochastic iterate $\boldsymbol{w}_{k,i}$ from converging almost surely to the minimizer w_k^o when constant step sizes are used. Some deterioration in performance will occur, and the iterate $\boldsymbol{w}_{k,i}$ will instead fluctuate close to w_k^o. It is common in the adaptive-filtering and stochastic-gradient-optimization literatures to assess the size of these fluctuations by measuring their steady-state mean-square value [9]–[11]. Therefore, we focus on highlighting the benefit of MTL on the network mean-square deviation (MSD), which is defined as the steady-state average variance value:

$$\text{MSD} \triangleq \lim_{i\to\infty} \frac{1}{N}\sum_{k=1}^{N} \mathbb{E}\|w_k^o - \boldsymbol{w}_{k,i}\|^2. \tag{9}$$

Consequently, when discussing theoretical performance results, and to avoid excessive technicalities, it is sufficient to focus on the MSE networks described in Example 2 and to assume that $R_{u,k} = R_u$ and $M_k = M$ for all k. In this way, the quality of the measurements, captured by the noise power $\sigma_{v,k}^2$, is allowed to vary across the network, with some agents collecting noisier data than other agents. Assuming uniform regressor covariance allows us to quantify the improvement in performance that results from cooperation without biasing the results by the statistical nature of the regression data at the agents.

Performance results under more general conditions, such as allowing for space-dependent covariances and lengths and for general second-order differentiable cost functions that are not necessarily quadratic, can also be found in [9, Ch. 3–4] for (2), in [12] for the strategy introduced in the "Multitask Estimation Under Smoothness" section, and in [13] for the strategies in the "Multitask Estimation Under Subspace Constraints" section. The MSD performance expressions in these works are derived under Lipschitz gradient vectors and Hessian matrices assumptions. The analyses in these works also allow recovery of the excess-risk metric at agent k, which is defined as $\text{ER}_k \triangleq \lim_{i\to\infty} \mathbb{E}(J_k(\boldsymbol{w}_{k,i}) - J_k(w_k^o))$; see, for example, [9, pp. 388–390]. Due to space limitations, we focus only on presenting MSD performance expressions.

Performance result 1

Consider an MSE network running the noncooperative algorithm (8). Assume further that $R_{u,k} = R_u$ and $M_k = M$ for all k. Under these assumptions, and for sufficiently small step sizes, the individual steady-state variance $\text{MSD}_k \triangleq \lim_{i\to\infty} \mathbb{E}\|w_k^o - w_{k,i}\|^2$, and the network MSD defined by (9) is given by [10]

$$\text{MSD}_k = \frac{\mu M}{2}\cdot\sigma_{v,k}^2, \qquad \text{MSD}^{\text{nc}} = \frac{\mu M}{2}\cdot\left(\frac{1}{N}\sum_{k=1}^{N}\sigma_{v,k}^2\right), \quad (10)$$

□

where the superscript "*nc*" is used to indicate that the MSD expression is for the noncooperative solution. First, observe that the performance is on the order of μ. The smaller μ is, the better the performance will be, but the slower the convergence toward w_k^o will be [9], [10]. [The same observation is valid for future expressions (17) and (34), with convergence to $\mathcal{W}^*$ in (17) instead.] Second, observe that agents with noisier data will perform worse than agents with cleaner data. However, since agents are observing data arising from similar or related models w_k^o, it is expected that an appropriate cooperation among agents can help enhance network performance.

Multitask-learning framework

Depending on the application, several task-relatedness models can be considered. For each, an appropriate convex optimization problem is solved in a distributed and adaptive manner. This results in different multitask strategies and, therefore, different cooperation rules among agents. Rather than describing each optimization problem in isolation, we begin by introducing a general problem, which allows us to recover various multitask strategies as special cases.

Let $\mathcal{W} \triangleq \text{col}\{w_1, \ldots, w_N\}$ denote the collection of parameter vectors from across the network. We consider the following global optimization problem for the multitask formulation:

$$\mathcal{W}^* = \underset{\mathcal{W}}{\text{argmin}} \; J^{\text{glob}}(\mathcal{W}) = \sum_{k=1}^{N} J_k(w_k) + \frac{\eta}{2}\mathcal{R}(\mathcal{W}), \quad \text{subject to } \mathcal{W} \in \Omega \quad (11)$$

where $\mathcal{R}(\cdot)$ is a convex regularization function promoting the relationships between the tasks, Ω is a closed convex set defining the feasible region of the parameter vectors, and $\eta > 0$ is a parameter controlling the importance of the regularization. The choice of the regularizer $\mathcal{R}(\cdot)$ and the set Ω depends on prior information about how the multitask models relate to each other. To illustrate how problem formulation (11) can be used, we consider the following two examples that are multitask oriented.

Example 3: Weather forecasting

Consider the network in Figure 2(a), consisting of $N = 139$ weather stations located across the United States and collecting daily measurements [12]. Let $\boldsymbol{h}_{k,i}$ denote the feature vector consisting of collected data (temperature, wind speed, dew point, and so on) at sensor k and day i, and let $\gamma_k(i)$ denote the corresponding binary variable associated with rain occurrence, that is, $\gamma_k(i) = 1$ if rain occurred, and $\gamma_k(i) = -1$ otherwise. The objective is to construct a classifier at each station to predict whether it will rain or not based on the knowledge of $\boldsymbol{h}_{k,i}$. To this end, each station can use an individual logistic regression machine similar to the one described in Example 1; in this case, the cost $J_k(w_k)$ in (11) takes the form (4).

However, it is expected that the decision rules $\{w_k^o\}$ at neighboring stations would be similar since they are collecting features arising from similar statistical distributions. Moreover, the strength of similarity is expected to be inversely proportional to the physical distance between the stations. This gives rise to a weighted graph (with the closest nodes connected by edges), and one may expect to improve the network performance by promoting the smoothness of $\{w_k^o\}$ with respect to the underlying graph. The simplest possible term that encourages smoothness is the graph Laplacian regularizer $S(\mathcal{W})$, defined by (13). By choosing $\mathcal{R}(\mathcal{W}) = S(\mathcal{W})$ and $\Omega = \mathbb{R}^{MN}$ in (11), one arrives at a multitask formulation for the weather forecasting application that takes into account the smoothness prior over the graph. This formulation and other possible formulations are solved in the "Regularized Multitask Estimation" and "Multitask Estimation Under Subspace Constraints" sections. □

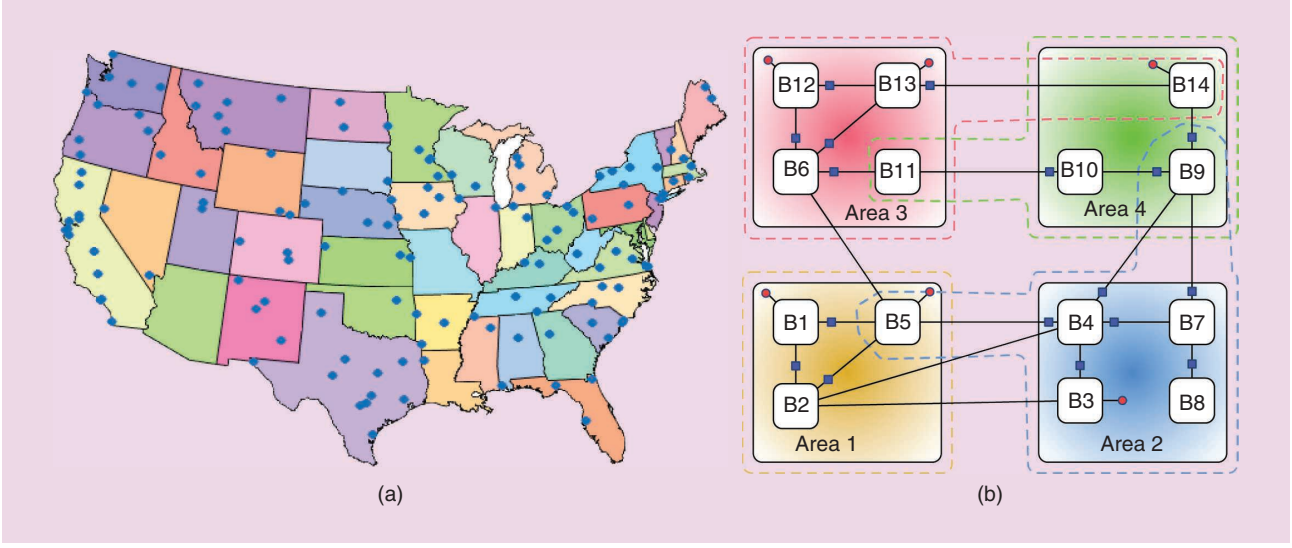

FIGURE 2. Examples of multitask applications include (a) weather forecasting and (b) distributed power-system monitoring.

Example 4: Power-system-state monitoring

Figure 2(b) illustrates an IEEE 14-bus power-monitoring system partitioned into four areas, where each area comprises a subset of buses supervised by its own control center [14]. The local state vectors (bus voltages) to be estimated at neighboring areas may partially overlap, as the areas are interconnected. This is because each control center collects measurements related to the voltages across its local buses and voltages across the interconnection between neighboring centers. For example, area 2 supervises buses 3, 4, 7, and 8. Since it collects current readings on lines 4 and 5 and 7 and 9, its state vector extends to buses 5 (supervised by area 1) and 9 (supervised by area 4). In other words, if we let w^n denote the state of bus n, then the cost $J_2(\cdot)$ at area 2 will depend on the extended parameter vector $w_2 = \text{col}\{w^3, w^4, w^5, w^7, w^8, w^9\}$.

However, since the parameter vectors at areas 1 and 4 will be $w_1 = \text{col}\{w^1, w^2, w^5\}$ and $w_4 = \text{col}\{w^9, w^{10}, w^{11}, w^{14}\}$, respectively, consensus needs to be reached on the variable w^5 between areas 2 and 1, and on the variable w^9 between areas 2 and 4, while minimizing the individual cost $J_2(w_2)$, penalizing deviation from data models of the form $y_k = H_k w_k + v_k$, where H_k is the measurement matrix, and v_k is a zero-mean noise. Thus, distributed-power-state estimation can be formulated as (11) with $\mathcal{R}(\mathcal{W}) = 0$, whereas the constraint set Ω, in this case, should be selected to promote consensus over the overlapped variables. In the "Multitask Estimation With Overlapping Parameter Vectors" section, we explain how such problems can be solved. □

Returning to (11), observe that even though the aggregate cost $\sum_{k=1}^{N} J_k(w_k)$ is separable in w_k, the cooperation between agents is necessary due to the coupling between tasks through the regularization and constraint. When solving problem (11), agent k will be responsible for estimating w_k^* (the kth subvector of $\mathcal{W}^* = \text{col}\{w_1^*, \ldots, w_N^*\}$), which is generally different from w_k^o in (1), the actual objective at agent k. However, it is expected that accurate prior information will allow the designer to choose the regularizer $\mathcal{R}(\cdot)$, the set Ω, and the strength η in a way that minimizes the distance between w_k^* and w_k^o.

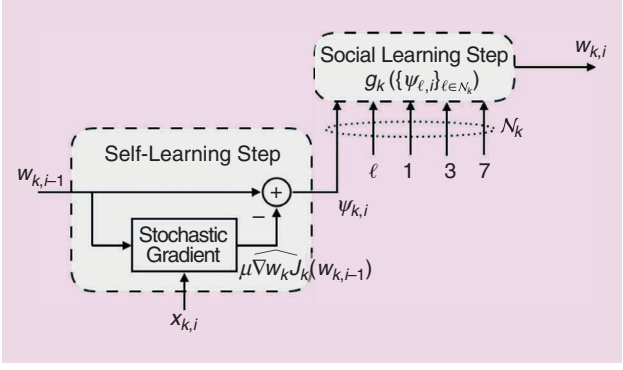

FIGURE 3. A common diagram for the multitask strategies described in this article. The structure involves two main steps: self-learning (12a) and social learning (12b).

In MTL, regularization is widely used to promote task relationships.

Although some existing works use primal-dual methods [15] to solve multitask estimation problems, we limit our exposition to the class of primal techniques (based on propagating and estimating the primal variable) that employ stochastic-gradient iterations. Extensive studies in the literature have shown that small step sizes enable these strategies to learn well in streaming data settings. Due to the separability property of $\sum_{k=1}^{N} J_k(w_k)$, the multitask algorithms described in the sequel will have a common structure given by

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} - \mu\widehat{\nabla_{w_k} J_k}(\boldsymbol{w}_{k,i-1}) \tag{12a}$$

$$\boldsymbol{w}_{k,i} = g_k(\{\boldsymbol{\psi}_{\ell,i}\}_{\ell\in\mathcal{N}_k}). \tag{12b}$$

The first step, (12a), corresponds to the stochastic gradient step on the individual cost $J_k(\cdot)$. We refer to this step as the *self-learning step* (Figure 3).

Compared with the noncooperative strategy (2), the result of the gradient descent step is $\boldsymbol{\psi}_{k,i}$, an intermediate estimate of w_k^o at iteration i. This step is followed by a social learning step (12b), which uses some function $g_k(\cdot)$ of the neighborhood iterates. As we will see in the following sections, the form of this function depends on the regularizer $\eta\mathcal{R}(\cdot)$ and the set Ω in (11), both of which allow promotion of the prior information on how the tasks w_k^o are related. The result of this second step is $\boldsymbol{w}_{k,i}$, the estimate of w_k^o, defined by (1), at iteration i. Since we are interested in a distributed setting, agents during social learning are allowed to collect estimators only from their local neighborhood $\mathcal{N}_k$ (Figure 3).

Next, we show how the formulation, (11), and the social learning step, (12b), specialize for regularized ("Regularized Multitask Estimation" section) and subspace-constrained ("Multitask Estimation Under Subspace Constraints" section) conditions. Due to space limitations, the clustered multitask formulation is considered in the extended version [16] of this article.

Regularized multitask estimation

In this section, we focus on the regularization term $\mathcal{R}(\mathcal{W})$ in (11) and its implications for learning dynamics. In MTL, regularization is widely used to promote task relationships. In most network applications, the underlying graph structure contains information about the relatedness among neighboring tasks. As such, when considering graph-based MTL applications, incorporating the graph structure into the regularization term is a reasonable and natural step. The smoothness model (under which the tasks are similar at neighboring vertices, with the strength of similarity specified by the weight between them) will play a central role in our discussion. This smoothness property is often observed in real-world applications (see, e.g., Example 3) and is rich enough to convey the main ideas behind MTL, as we see in the next section. We examine two main questions: 1) How should graph-based priors be incorporated into the regularizer? and 2) How does the resulting MTL algorithm behave?

Multitask estimation under smoothness

We assume that a symmetric, weighted adjacency matrix C is associated with the connected graph illustrated in Figure 1(c). If there is an edge connecting agents k and ℓ, then $[C]_{k\ell} = c_{k\ell} > 0$ reflects the strength of the relation between k and ℓ; otherwise, $[C]_{k\ell} = 0$. These weights are usually dictated by the physics of the problem at hand (see, for example, [17] and [18, Ch. 4] for graph construction methods). We introduce the graph Laplacian L, which is a differential operator defined as $L = \text{diag}\{C\mathbb{1}_N\} - C$. Assuming that the tasks have the same length, that is, $M_k = M\ \forall k$, the smoothness of $\mathcal{W}$ over the graph is measured in terms of a quadratic form of the Laplacian [19]:

$$S(\mathcal{W}) = \mathcal{W}^{\top}\mathcal{L}\mathcal{W} = \frac{1}{2}\sum_{k=1}^{N}\sum_{\ell\in\mathcal{N}_k} c_{k\ell}\|w_k - w_\ell\|^2, \tag{13}$$

where $\mathcal{L} = L \otimes I_M$ is an extended form of the graph Laplacian (defined in terms of the Kronecker product operator $\otimes$). The smaller $S(\mathcal{W})$ is, the smoother the signal $\mathcal{W}$ on the graph is. Given that the weights are nonnegative, $S(\mathcal{W})$ shows that $\mathcal{W}$ is smooth if nodes with a large $c_{k\ell}$ on the edge connecting them have similar weight values $\{w_k, w_\ell\}$. Therefore, to enforce the prior belief that the target signal $\mathcal{W}^o = \text{col}\{w_1^o, \ldots, w_N^o\}$ is smooth with respect to the underlying weighted graph, in (11), one may choose

$$\mathcal{R}(\mathcal{W}) = S(\mathcal{W}), \quad \text{and} \quad \Omega = \mathbb{R}^{MN}. \tag{14}$$

Under this choice, the stochastic gradient algorithm for solving (11) takes the following form:

$$\mathcal{W}_i = \boldsymbol{\psi}_i - \mu\eta\mathcal{L}\mathcal{W}_{i-1}, \tag{15}$$

where $\mathcal{W}_i$ is the estimate of $\mathcal{W}^*$ at instant i, and $\boldsymbol{\psi}_i = \text{col}\{\boldsymbol{\psi}_{k,i}\}_{k=1}^N$ is the vector collecting the intermediate estimates $\boldsymbol{\psi}_{k,i}$ in (12a) from across all agents. Since we expect $\boldsymbol{\psi}_i$ to be an improved estimate compared to $\mathcal{W}_{i-1}$, we propose replacing $\mathcal{W}_{i-1}$ in (15) with $\boldsymbol{\psi}_i$. By doing so, we obtain (12) with the social learning step given by

$$\boldsymbol{w}_{k,i} = \boldsymbol{\psi}_{k,i} - \mu\eta\sum_{\ell\in\mathcal{N}_k} c_{k\ell}(\boldsymbol{\psi}_{k,i} - \boldsymbol{\psi}_{\ell,i}). \tag{16}$$

The substitution of $\mathcal{W}_{i-1}$ with $\boldsymbol{\psi}_i$ is reminiscent of incremental-type arguments in gradient-descent algorithms [20]. Analyses in the context of adaptation over networks show that substitutions of this type lead to enhanced network stability since they allow the stability of the agents to be preserved after cooperation (see, e.g., [10, p. 160] for details). Regarding (16), it follows that, when the spectral radius of the combination matrix $I - \mu\eta\mathcal{L}$ is equal to one, sufficiently small step sizes ensuring the individual agents' stability will also ensure the network stability [12]. [In this article, a network is said to be stable if the MSE $(1/N)\|\mathcal{W}^* - \mathcal{W}_i\|^2$ converges asymptotically to a bounded region of the order of the step-size.]

Understanding this bias-variance tradeoff is critical for understanding the behavior of regularized multitask algorithms.

Proximal-based approaches are also proposed in [21] to solve multitask problems under smoothness. However, these approaches require the evaluation of the proximal operator defined by (28) of the risk $Q_k(\cdot)$ at each iteration i, which can be computationally expensive.

Bias-variance tradeoff

We next consider the interesting question of whether MTL is beneficial compared to noncooperation. The answer to this inquiry requires 1) studying the performance of (12) relative to the actual agents objectives $\{w_k^o\}$ and then 2) examining when the multitask implementation (12) can lead to enhanced performance in comparison to the noncooperative solution (2).

Algorithm (12) was studied in detail in [12]. It was shown that the network MSD defined by (9) is mainly influenced by the sum of two factors, as explained later. The first factor is the steady-state variance of (12) with respect to the regularized solution $\mathcal{W}^*$ in (11), namely, $\lim_{i\to\infty}(1/N)\mathbb{E}\|\mathcal{W}^* - \boldsymbol{\mathcal{W}}_i\|^2$. The second is the bias or the average distance between the regularized solution $\mathcal{W}^*$ and the unregularized $\mathcal{W}^o$, namely, $(1/N)\|\mathcal{W}^o - \mathcal{W}^*\|^2$. By increasing the regularization strength η, the variance term is more likely to decrease, while the bias term is more likely to increase. Understanding this bias-variance tradeoff is critical for understanding the behavior of regularized multitask algorithms.

Therefore, we describe the bias-variance behavior of (16) by considering the expressions derived in [12]. These expressions are useful for illustrating the concept of MTL. Instead of involving the vertex domain given by the entries $\{c_{k\ell}\}$ of the adjacency matrix, these expressions involve the graph spectral information defined by the eigendecomposition of the Laplacian L. Because the Laplacian is a real symmetric matrix, it has a complete set of orthonormal eigenvectors; we denote them by $\{v_1, \ldots, v_N\}$. For convenience, we order the set of real, nonnegative eigenvalues of L as $0 = \lambda_1 < \lambda_2 \le \ldots \le \lambda_N$, where, since the network is connected, there is only one zero eigenvalue with corresponding eigenvector $v_1 = (1/\sqrt{N})\mathbb{1}_N$ [22]. Therefore, the Laplacian can be decomposed as $L = V\Lambda V^{\top}$, where $\Lambda = \text{diag}\{\lambda_1, \ldots, \lambda_N\}$ and $V = [v_1, \ldots, v_N]$.

Performance result 2

Consider an MSE network running the multitask algorithm (12) with the second step given by (16). Assume further that $R_{u,k} = R_u\ \forall k$ and that $\rho(I - \mu\eta L) \le 1$. Under these assumptions and for sufficiently small step sizes and smooth signal $\mathcal{W}^o$, it is shown that [12]

$$\lim_{i\to\infty}\frac{1}{N}\mathbb{E}\|\mathcal{W}^* - \boldsymbol{\mathcal{W}}_i\|^2 \approx \sum_{m=1}^{N}\varphi(\lambda_m), \tag{17}$$

where

$$\varphi(\lambda_m) = \frac{\mu}{2N}\left(\sum_{k=1}^{N}[v_m]_k^2\sigma_{v,k}^2\right)\left(\sum_{q=1}^{M}\frac{\lambda_{u,q}}{\lambda_{u,q} + \eta\lambda_m}\right), \tag{18}$$

with $\lambda_{u,q}$, the qth eigenvalue of R_u, and $[v_m]_k$, the kth entry of the eigenvector v_m. For the bias term, it can be shown that [12]:

$$\|\mathcal{W}^o - \mathcal{W}^*\|^2 = \sum_{m=2}^{N} \zeta(\lambda_m), \tag{19}$$

where

$$\zeta(\lambda_m) = \left\| \eta\lambda_m (R_u + \eta\lambda_m I_M)^{-1} \bar{w}_m^o \right\|^2, \tag{20}$$

with $\bar{w}_m^o = (v_m^\top \otimes I_M)\mathcal{W}^o$, the mth subvector of $\bar{\mathcal{W}}^o = (V^\top \otimes I_M)\mathcal{W}^o$ corresponding to the eigenvalue λ_m. □

The steady-state variance (17) consists of the summation of N terms $\varphi(\lambda_m)$, each corresponding to an eigenvalue λ_m of the Laplacian. The first, $\varphi(\lambda_1 = 0)$, is independent of the regularization strength η. The remaining terms, $\varphi(\lambda_m \neq 0)$, decrease when η increases. Therefore, when η increases, the variance decreases. From (19), we observe that the bias tends to increase by increasing the regularization strength η. However, an interesting fact arises for smooth $\mathcal{W}^o$. We rewrite the regularizer in (13) as

$$S(\mathcal{W}^o) = (\bar{\mathcal{W}}^o)^\top (\Lambda \otimes I_M)\bar{\mathcal{W}}^o = \sum_{m=2}^{N} \lambda_m \|\bar{w}_m^o\|^2, \tag{21}$$

where we used the fact that $\lambda_1 = 0$. Intuitively, given that $\lambda_m > 0$ for $m = 2, \ldots, N$, (21) shows that $\mathcal{W}^o$ is considered to be smooth over the graph if $\|\bar{w}_m^o\|^2$ corresponding to large eigenvalue λ_m is very small. Therefore, a smooth $\mathcal{W}^o$ is mainly contained in $[0, \lambda_c]$, that is, $\|\bar{w}_m^o\|^2 \approx 0$ if $\lambda_m > \lambda_c$, and the smoother $\mathcal{W}^o$ is, the smaller λ_c will be.

In this case, the effective sum in (19) is over the first $c \ll N$ terms (corresponding to small eigenvalues λ_m) instead of N terms. We conclude that as long as $\mathcal{W}^o$ is sufficiently smooth, moderate regularization strengths η in the range $(0, \infty)$ exist such that the decrease in variance at these values of η will dominate the increase in bias. In other words, the MSD at these values of η will be less than the MSD at $\eta = 0$, which corresponds to the noncooperative mode of operation.

From (16), the social learning step following from the Laplacian regularization term (13) involves a single communication step at every stochastic gradient update. When multiple steps are allowed, it is reasonable to expect that performance can be improved. In the following section, we show how such solution can be designed.

Graph spectral regularization

The main observation behind the introduction of this regularizer is that a smooth $\mathcal{W}^o$ over a graph exhibits a special structure in the graph spectral domain (mainly contained in $[0, \lambda_c]$, that is, $\|\bar{w}_m^o\|^2 \approx 0$ if $\lambda_m > \lambda_c$) [23]. Graph spectral regularization is used to more thoroughly leverage the spectral information and improve the multitask network performance. In this case, the network will aim at solving (11), with $\Omega = \mathbb{R}^{MN}$ and $\mathcal{R}(\cdot)$ properly selected to promote the prior information available on the structure of $\mathcal{W}^o$ in the graph spectral domain. The following class of regularization functionals on graphs can be used for this purpose [23], [24]:

$$\mathcal{R}(\mathcal{W}) = \mathcal{W}^\top r(\mathcal{L})\mathcal{W} = \mathcal{W}^\top (r(L) \otimes I_M)\mathcal{W}, \tag{22}$$

where $r(\cdot)$ is some well-defined nonnegative function on the spectrum $\sigma(L) = \{\lambda_1, \ldots, \lambda_N\}$ of L, and $r(L)$ is the corresponding matrix function defined as [25, p. 3]:

$$r(L) = V r(\Lambda) V^\top = \sum_{m=1}^{N} r(\lambda_m) v_m v_m^\top. \tag{23}$$

Construction (22) uses the Laplacian as a means to design regularization operators. Requiring a positive semidefinite regularizer $r(L)$ imposes the constraint $r(\lambda) \geq 0$ for all $\lambda \in \sigma(L)$. Substituting (23) into (22), we obtain the following [compare with the regularizer in (21) to see how an extra degree of freedom is introduced in the multitask network design]:

$$\mathcal{R}(\mathcal{W}) = \bar{\mathcal{W}}^\top (r(\Lambda) \otimes I_M)\bar{\mathcal{W}} = \sum_{m=1}^{N} r(\lambda_m) \|\bar{w}_m\|^2, \tag{24}$$

where $\bar{\mathcal{W}} = (V^\top \otimes I_M)\mathcal{W}$ and $\bar{w}_m = (v_m^\top \otimes I_M)\mathcal{W}$. The regularization in (24) promotes a particular structure in the graph spectral domain. It strongly penalizes $\|\bar{w}_m\|^2$, for which the corresponding $r(\lambda_m)$ is large. Thus, one prefers $r(\lambda_m)$ to be large for those $\|\bar{w}_m\|^2$ that are small, and vice versa. From the discussion following (21), it is clear that, under smoothness, the function $r(\lambda)$ must be chosen to be monotonically increasing in λ. One typical choice is $r(\lambda) = \lambda^S$ with $S \geq 1$. Example 5 illustrates, for instance, the benefit of using λ^3 instead of λ.

Assuming the regularizer $r(L)$ in (22) can be written as an Sth-degree polynomial of the Laplacian L, that is, $r(L) = \sum_{s=0}^{S} \beta_s L^s$ for some constants $\{\beta_s\}$ [or, equivalently, $r(\lambda) = \sum_{s=0}^{S} \beta_s \lambda^s$], and following similar arguments that led to (16), one arrives at the following social step (12b) [24]:

$$\begin{cases} \boldsymbol{\psi}_{k,i}^s = \beta_{S-s}\boldsymbol{\psi}_{k,i} + \sum\limits_{\ell \in \mathcal{N}_k} c_{k\ell}(\boldsymbol{\psi}_{k,i}^{s-1} - \boldsymbol{\psi}_{\ell,i}^{s-1}), \quad s = 1, \ldots, S \\ \boldsymbol{w}_{k,i} = \boldsymbol{\psi}_{k,i} - \mu\eta\boldsymbol{\psi}_{k,i}^S \end{cases}, \tag{25}$$

where $\boldsymbol{\psi}_{k,i}^0 = \beta_S \boldsymbol{\psi}_{k,i}$. It requires S communication steps. The resulting algorithm (25) is distributed, since, at each step, each agent is required to exchange information only locally with its neighbors. Since S communication steps are required, agent k ends up collecting information from its S-hop neighborhood.

For more general $r(\lambda)$ that are not necessarily polynomial in λ, one would like to benefit from the sparsity of the graph captured by L. As long as $r(L)$ can be approximated by some lower-order polynomial in L, say $r(L) \approx \sum_{s=0}^{S} \beta_s L^s$, distributed implementations of the form (25) are possible (see [24]). Problems of this type have already been considered in graph filter designs [26], [27].

For instance, in [26], Shuman et al. propose locally approximating $r(\cdot)$ by a polynomial $\tilde{r}(\cdot)$ computed by truncating a shifted Chebyshev series expansion of $r(\cdot)$ on $[0, \lambda_N]$. When the regularizer $r(\cdot)$ is continuous, the Chebyshev approximation $\tilde{r}(\cdot)$ converges to it rapidly as S increases. When the

regularizer presents some discontinuities, polynomial approximation methods are not advised over adaptive networks, since accurate approximation would require a large-order S, and, consequently, a large number of communication steps at each iteration. Projection-based methods similar to the one described in the "Multitask Estimation Under Subspace Constraints" section can be useful in this case. For example, if the smooth signal $\mathcal{W}^o$ is contained only in $[0, \lambda_c]$, instead of using a discontinuous regularizer $r(\lambda)$ of the form $r(\lambda_m) = 0$ if $m < c$ and $\beta \gg 0$ otherwise, one may design a multitask network that is able to project onto the space spanned by the first c eigenvectors of the graph Laplacian.

Since the optimization problems in the "Multitask Estimation Under Smoothness" and "Graph Spectral Regularization" sections are the same, with $\mathcal{L}$ in (13) replaced by $r(\mathcal{L})$ in (22), the multitask strategy (25) will behave in a similar manner as (16). Particularly, the bias-variance tradeoff discussion continues to apply, and (17)–(20) continue to hold, with λ_m on the right-hand side of (18) and (20) replaced by the function $r(\lambda_m)$. By replacing λ_m on the RHS of (20) by $r(\lambda_m)$, one may directly observe the consequence of this regularizer on the bias term (19), which can now be made close to zero [by choosing in the smoothness case, for example, $r(\lambda_m) \approx 0$ if $\lambda_m \in [0, \lambda_c]$ and $\beta_m > 0$ otherwise].

Example 5: Graph spectral filtering

Consider the MSE network example in Figure 4 and assume uniform data profile, that is, $R_{u,k} = R_u$ and $\sigma^2_{v,k} = \sigma^2_v \ \forall k$. In Figure 4(a), we illustrate the entries of the tasks w^o_k, which are smooth over the underlying graph. In Figure 4(b), we illustrate the behavior of the algorithms previously described in the graph spectral domain. The top plot represents the behavior of the network output $\mathcal{W}^*$ for three different choices of regularizer $r(\lambda) = \{0, \lambda, \lambda^3\}$. The bottom plot represents the behavior of the steady-state variance (18), with the eigenvalue λ_m replaced by the function $r(\lambda_m)$.

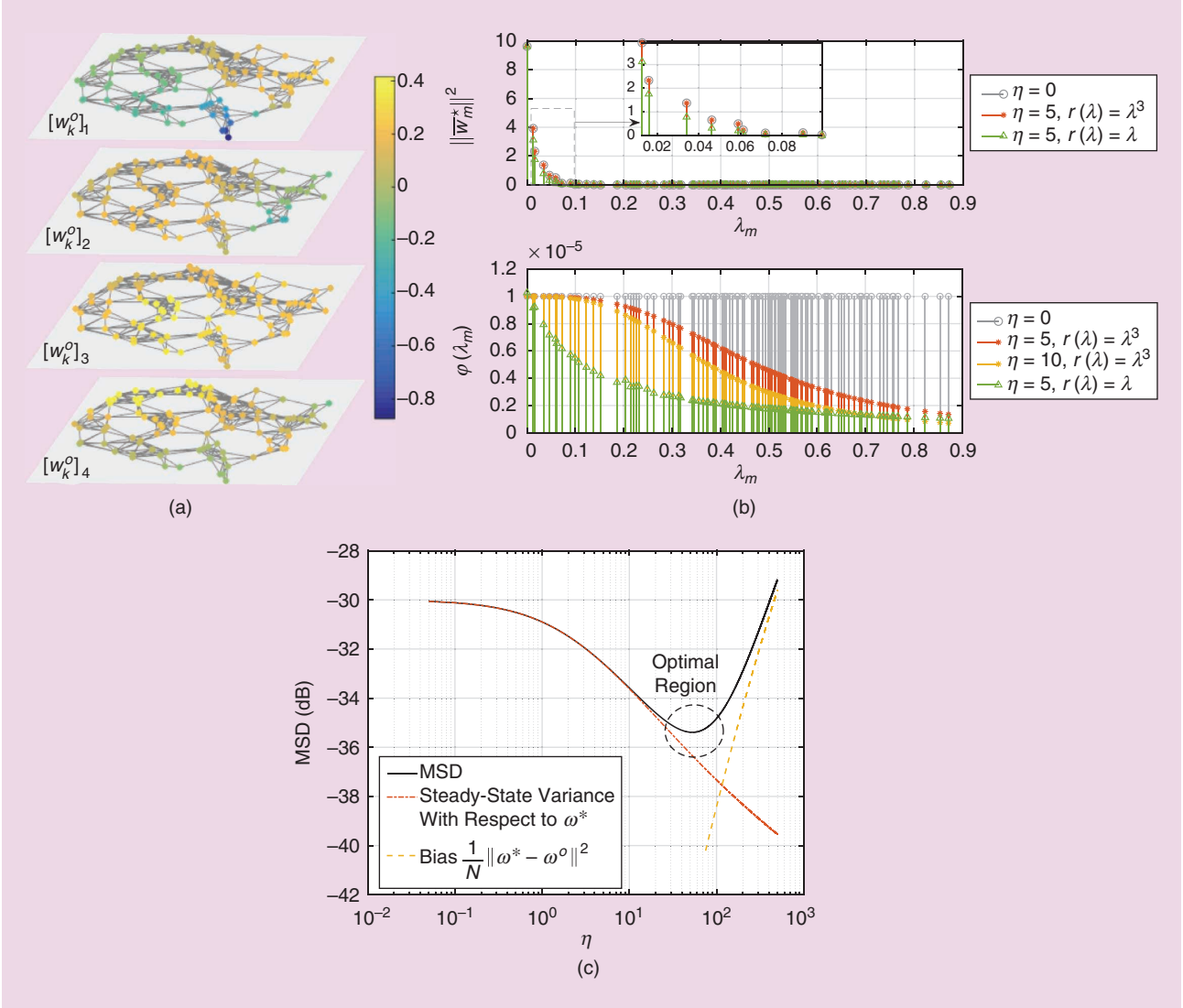

FIGURE 4. An illustrative example for spectral regularization. (a) The estimation under smoothness. (b) The behavior of (12) in the graph spectral domain with $\bar{w}^*_m = (v^\top_m \otimes I_M)\mathcal{W}^*$. (c) The bias-variance tradeoff for $r(\lambda) = \lambda^3$.

The regularizer $r(\lambda)=\lambda^3$ penalizes low eigenvalues less than $r(\lambda)=\lambda$ and, consequently, preserves all of the signal components $\bar{w}_m$; observe the graph low-pass filtering behavior [26], [27]. Small eigenvalues λ_m correspond to low frequencies, $\bar{\mathcal{W}}=(V^{\mathsf{T}}\otimes I_M)\mathcal{W}$ corresponds to the graph Fourier transform, and $\bar{w}_m=(v_m\otimes I_M)\mathcal{W}$ corresponds to the mth frequency content of $\mathcal{W}$. It can be shown that the mth frequency content of the output can be bounded as $\|\bar{w}_m^*\|\leq \lambda_{u,\max}/(\lambda_{u,\max}+\eta r(\lambda_m))\|\bar{w}_m^o\|$ in terms of the mth frequency content of the input $\mathcal{W}^o$, where $\lambda_{u,\max}$ is the maximum eigenvalue of R_u (see [12]).

Since $r(\lambda)$ is monotonically increasing in λ, for fixed η, as λ increases, the ratio decreases. Therefore, the network output $\mathcal{W}^*$ can be interpreted as the output of a low-pass graph filter applied to the signal $\mathcal{W}^o$. A similar behavior arises for the steady-state variance. For fixed η, as λ_m increases, the variance at the mth frequency, that is, $\varphi(\lambda_m)$, decreases, and for fixed λ_m, as η increases, $\varphi(\lambda_m)$ decreases. The regularizer $r(\lambda)$ controls the shape of the filter, and the strength η controls the sharpness. The noncooperative solution ($\eta=0$) corresponds to an all-pass graph filter. In Figure 4(c), we illustrate the bias-variance tradeoff in the case of $r(\lambda)=\lambda^3$. □

Returning to the diagram in Figure 3, we see that the self-learning step corresponds to the inference step, where agent k estimates w_k^o from streaming data $\boldsymbol{x}_{k,i}$, and that the social-learning step corresponds to the graph-filtering step, where the agents collaborate to perform spatial filtering and reduce the effect of the noise on the network MSD defined by (9). These steps are performed simultaneously. Therefore, MTL over networks allows the blending of real-time adaptation with graph (spatial) filtering.

MTL over networks allows the blending of real-time adaptation with graph (spatial) filtering.

Nonquadratic regularization

Nonquadratic regularization has been also discussed in the literature [15], [28], [29]. This scenario induces nonlinearities in the social learning step (12b). In this case, multitask algorithms are derived to solve problem (11) with

$$\Omega=\mathbb{R}^{MN} \quad \text{and} \quad \mathcal{R}(\mathcal{W})=\sum_{k=1}^{N}\sum_{\ell\in\mathcal{N}_k}\rho_{k\ell}h_{k\ell}(w_k,w_\ell), \tag{26}$$

where $h_{k\ell}:\mathbb{R}^M\times\mathbb{R}^M\to\mathbb{R}$ is a convex cost function associated with the link (k,ℓ). In general, this function is used to enforce some constraints on the pairs of variables across an edge. Equation (26), by allowing arbitrary distance measures $h_{k\ell}(\cdot,\cdot)$, is a generalization of the previously employed quadratic regularization. In fact, setting $h_{k\ell}(w_k,w_\ell)=\|w_k-w_\ell\|^2$ recovers (13). Examples of other typical choices are the ℓ_2-norm regularizer $h_{k\ell}(w_k,w_\ell)=\|w_k-w_\ell\|$ and the ℓ_1-norm regularizer $h_{k\ell}(w_k,w_\ell)=\|w_k-w_\ell\|_1$.

Instead of encouraging global smoothness, these sparsity-based regularizers can adapt to heterogeneity in the level of smoothness of the tasks w_k^o across nodes [30]. Such heterogeneity is observed, for instance, in the problem of predicting housing prices [29]. In this problem, the objective at each node (house) in a graph (where neighboring houses are connected by edges) is to learn the weights w_k^o of a regression model (examples of features include the number of bedrooms, square footage, and so on) to estimate the price. Due to location-based factors (such as distance to a highway) that are often unknown a priori and, therefore, cannot be incorporated as features, similar houses in different, though close, locations (neighbors) can have drastically different prices, that is, drastically different w_k^o.

The objective, in this case, is to encourage neighboring houses that share common models to cooperate without being influenced by the misleading information of neighbors sharing different models, that is, perform automatic clustering. To do so, Hallac et al. [29] propose solving the network Lasso problem [i.e., (11)] with ℓ_2-norm regularizer in (26). The rationale behind this choice is that the ℓ_2-norm regularizer encourages group sparsity, that is, consensus across an edge $w_k=w_\ell$.

On the other hand, the ℓ_1-norm regularizer is used in [28] to promote the prior that the parameter vectors at neighboring nodes have a large number of similar entries and a small number of distinct entries. The weight $\rho_{k\ell}\geq 0$ in (26) associated with the link (k,ℓ) aims at locally adjusting the regularization strength; it is usually dictated by the physics of the problem at hand. For primal adaptive techniques and due to the nondifferentiability of the regularizers, proximal gradient methods can be used to solve (11). Assuming $\rho_{k\ell}=\rho_{\ell k}$ and $h_{k\ell}(w_k,w_\ell)=\|w_k-w_\ell\|_1$, one may arrive at a multitask algorithm of the form (12) with the social learning step (12b) given by (see the derivations in [28])

$$\boldsymbol{w}_{k,i}=\text{prox}_{\eta\mu\tilde{g}_{k,i}}(\boldsymbol{\psi}_{k,i}), \tag{27}$$

where $\text{prox}_{\gamma g}(w')$ denotes the proximal operator of the function $g(w)$:

$$\text{prox}_{\gamma g}(w')=\operatorname*{argmin}_{w\in\mathbb{R}^M} g(w)+\frac{1}{2\gamma}\|w-w'\|^2, \tag{28}$$

and where the function $\tilde{g}_{k,i}:\mathbb{R}^M\to\mathbb{R}$ is given by $\tilde{g}_{k,i}(w_k)=\sum_{\ell\in\mathcal{N}_k}\rho_{k\ell}h_{k\ell}(w_k,\boldsymbol{\psi}_{\ell,i})$. The proximal operator in (27) must be evaluated at each iteration. For the weighted sum of the ℓ_1 regularizer, a closed-form expression can be found in [28].

Multitask estimation under subspace constraints

In addition to regularized-based algorithms, projection-based algorithms have received considerable attention in the literature of deterministic [20], [31]–[33] and stochastic [9], [10], [13], [34]–[36] optimization. The objective in this case is to design distributed networks that are able to project onto low-dimensional subspaces while minimizing the individual costs, that is, solve problems of (11) with [13], [31]

$$\mathcal{R}(\mathcal{W})=0, \quad \text{and} \quad \Omega=\text{Range}(\mathcal{U}), \tag{29}$$

where $\text{Range}(\cdot)$ denotes the range space operator, and $\mathcal{U}$ is an $M_t\times P$ full-column rank matrix with $M_t=\sum_{k=1}^{N}M_k$ and

$P \ll M_t$. As we explain, consensus-type problems are instances of this formulation. Also, multitask estimation under smoothness can benefit from this formulation: as explained previously in the "Graph Spectral Regularization" section, when the first c eigenvectors of the Laplacian are available, the designer can project onto Range($\mathcal{U}$) with $\mathcal{U} = [v_1, \ldots, v_c] \otimes I_M$ instead of using regularization.

Let $\mathcal{P}_{\mathcal{U}} = \mathcal{U}(\mathcal{U}^\top \mathcal{U})^{-1}\mathcal{U}^\top$ denote the projection onto the range space of $\mathcal{U}$. Assuming that the network topology and the signal subspace $\mathcal{U}$ are such that the following feasibility problem,

$$\begin{aligned} &\text{find} && \mathcal{A} \\ &\text{such that} && \mathcal{A}\mathcal{U} = \mathcal{U},\ \ \mathcal{U}^\top \mathcal{A} = \mathcal{U}^\top,\ \ \rho(\mathcal{A} - \mathcal{P}_{\mathcal{U}}) < 1, \\ & && [\mathcal{A}]_{k\ell} = 0, \ \ \text{if } \ell \notin \mathcal{N}_k \text{ and } \ell \neq k \end{aligned} \tag{30}$$

admits at least one solution, one may arrive at a multitask strategy of the form (12), with the social learning step (12b) given by [13]

$$\boldsymbol{w}_{k,i} = \sum_{\ell \in \mathcal{N}_k} A_{k\ell} \boldsymbol{\psi}_{\ell,i}, \tag{31}$$

where $A_{k\ell} = [\mathcal{A}]_{k\ell}$ is the (k, ℓ)th block (of size $M_k \times M_\ell$) of the $N \times N$ block matrix $\mathcal{A}$. A matrix $\mathcal{A}$ satisfying the constraints in (30) is semiconvergent [13], [31]. Particularly, it holds that

$$\lim_{i \to \infty} \mathcal{A}^i = \mathcal{P}_{\mathcal{U}}. \tag{32}$$

In single-task estimation, the agents are seeking a common minimizer w^o.

The first two constraints in (30) state that the P columns of $\mathcal{U}$ are right and left eigenvectors of $\mathcal{A}$ associated with the eigenvalue of one. Together with these two constraints, the third constraint in (30) ensures that $\mathcal{A}$ has P eigenvalues at one and that all other eigenvalues are strictly less than one in magnitude. The last constraint in (30) corresponds to the sparsity constraint, which characterizes the network topology and ensures local exchange of information at each instant i.

Before explaining how some typical choices of $\mathcal{U}$ lead to well-studied distributed inference problems, we note that the distributed algorithm (31) has an attractive property: in the small-step-size regime, the iterations generated by (31) achieve the steady-state performance of the following gradient projection algorithm [13]:

$$\mathcal{W}_i = \mathcal{P}_{\mathcal{U}}\Big(\mathcal{W}_i - \mu \text{col}\{\widehat{\nabla_{w_k} J_k}(\boldsymbol{w}_{k,i-1})\}_{k=1}^N\Big), \tag{33}$$

which is centralized since, at each instant i, agent k needs to send its estimate $\boldsymbol{\psi}_{k,i}$ in (12a) to a fusion center, which performs the projection and then sends the result $\boldsymbol{w}_{k,i}$ back to the agent.

Performance result 3

Consider an MSE network running (12), with the social step (12b) given by (31), with $\mathcal{A} = [A_{k\ell}]$ satisfying the constraints in (30). Assume that the network is seeking $\mathcal{W}^o \in$ Range($\mathcal{U}$). Assume further that $\mathcal{U} = U \otimes I_M$, where $U = [u_1, \ldots, u_{\bar{P}}]$ is semiorthogonal, and that $R_{u,k} = R_u$ and $M_k = M$ for all k. Under these assumptions, and for sufficiently small step sizes, the network MSD defined by (9) is given by [13]:

$$\text{MSD} = \frac{\mu M}{2N} \sum_{m=1}^{\bar{P}} \left(\sum_{k=1}^{N} [u_m]_k^2 \sigma_{v,k}^2 \right). \tag{34}$$

The projection framework will not induce bias in the estimation. This is because $\mathcal{W}^o \in$ Range($\mathcal{U}$), and, therefore, the vector $\mathcal{W}^*$ in (11) is equal to $\mathcal{W}^o$, the network objective. Moreover, the benefit of cooperation can be readily seen by assuming uniform variances $\sigma_{v,k}^2 = \sigma_v^2$ for all k. In this case, comparing (34) with (10) in the noncooperative case, we conclude that $\text{MSD} = (\bar{P}/N)\text{MSD}^{\text{nc}}$, where $\bar{P}/N \ll 1$. Therefore, the cooperative strategy outperforms the noncooperative one by a factor of $N/\bar{P}$.

Single-task estimation

In single-task estimation, the agents are seeking a common minimizer w^o [Figure 1(a)]. This problem is encountered in many applications. Examples include target localization and distributed sensing (see, e.g., [10]). Single-task estimation can be recast in the form of (11), where $\mathcal{R}(\cdot)$ and Ω are chosen according to (29), with $\mathcal{U} = (1/\sqrt{N})(\mathbb{1}_N \otimes I_M)$. Several algorithms for solving such consensus-type problems have been proposed in the literature, including incremental [20], consensus [32], and diffusion [9], [10] strategies. Due to lack of space, we will describe only the class of diffusion strategies, which can be written in the form of (12), with the social step (12b) given by

$$\boldsymbol{w}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{k\ell} \boldsymbol{\psi}_{\ell,i}, \tag{35}$$

where $a_{k\ell}$ corresponds to the (k, ℓ)th entry of an $N \times N$ doubly stochastic matrix A satisfying

$$a_{k\ell} \geq 0, \ \sum_{\ell=1}^{N} a_{k\ell} = 1, \ \sum_{k=1}^{N} a_{k\ell} = 1, \text{ and } a_{k\ell} = 0 \text{ if } \ell \notin \mathcal{N}_k. \tag{36}$$

Several methods for locally selecting these combination coefficients have been proposed in the literature, such as the metropolis and Laplacian rules (see, e.g., [9]). Step (35) can be written in the form of (31), with $A_{k\ell} = a_{k\ell} I_M$ and $\mathcal{A} = A \otimes I_M$, and the resulting matrix $\mathcal{A}$ will satisfy the constraints in (30) over a strongly connected network.

Multitask estimation with overlapping parameter vectors

It is assumed that the individual costs $J_k(\cdot)$ depend on only a subset of the components of a global parameter vector $w = [w^1, \ldots, w^M]^\top \in \mathbb{R}^{M \times 1}$ [33]–[36]. This situation is observed in Example 4, where the network global parameter vector is $w = \text{col}\{w^n\}_{n=1}^{14}$ and where the states w_k, to be estimated at neighboring areas, partially overlap. It can be verified that this problem can also be recast in the form of (29) with $\mathcal{U}$ properly selected. To solve this consensus-type

problem, and motivated by the single-task diffusion strategies, researchers [34], [35] propose the following algorithm. Assume agent k is interested in estimating the entry w^n of w and let $\mathcal{N}_k^n$ denote the set of neighbors of k that are also involved in estimating w^n. To reach consensus on w^n, agent k assigns to its entry w^n a set of nonnegative coefficients $\{a_{k\ell}^n\}$ satisfying

$$a_{k\ell}^n = 0 \text{ if } \ell \notin \mathcal{N}_k^n, \quad \sum_{\ell \in \mathcal{N}_k^n} a_{k\ell}^n = 1, \quad \sum_{\ell \in \mathcal{N}_k^n} a_{\ell k}^n = 1 \tag{37}$$

and performs the following convex combination:

$$\boldsymbol{w}_{k,i}^n = \sum_{\ell \in \mathcal{N}_k^n} a_{k\ell}^n \boldsymbol{\psi}_{\ell,i}^n, \tag{38}$$

where $\boldsymbol{\psi}_{\ell,i}^n$ is the entry of the $M_\ell \times 1$ intermediate estimate $\boldsymbol{\psi}_{\ell,i}$ [obtained from (12a)] corresponding to the variable w^n, and $\boldsymbol{w}_{k,i}^n$ is the estimate of w^n at node k and instant i. It can also be verified that solution (38) can be written in the form of (31) with the block $A_{k\ell}$ properly selected.

For MSE networks, a recursive-least-squares approach is proposed in [36] to solve overlapping multitask estimation. In general, second-order gradient methods enjoy faster convergence rates than first-order methods at the expense of increasing the computational complexity.

Conclusions

In this article, we explained how prior knowledge about task relationships can be incorporated into the adaptation mechanism and how different priors yield different multitask strategies. Therefore, choosing the optimal strategy for a given problem is equivalent to selecting the task-relatedness model that best fits the underlying problem. Finding the most practically viable solution, then, further balances this model fit against computational and communication constraints.

There are several other aspects and strategies for MTL over graphs that were not covered in this article due to space limitations. For instance, we focused only on multitask networks endowed with parameter estimation tasks. However, distributed detection has also been considered from a multitask perspective (see, e.g., [37]). Online network clustering has been discussed as well. In that case, the objective is to design diffusion networks that are able to adapt their combination coefficients in (35) to exclude harmful neighbors sharing distinct tasks [38], [39]. Readers can refer to [40] for a list of other multitask-oriented works in the literature.

MTL over graphs is worth exploring further, as there are many potential ideas to develop. For instance, the expressions show the sensitivity of the results to the underlying graph structure. It would be useful to infer the entries $c_{k\ell}$ of the adjacency matrix in (16) simultaneously with the self-learning step (12a); this leads to learning the relations between the tasks simultaneously with the tasks. Automatically determining the optimal regularization strength η and allowing edge regularizers $h_{k\ell}(w_k, w_\ell)$ beyond just the ℓ_1 norm also constitute clear extensions. Finally, we believe that the number of MTL applications in distributed, streaming machine learning is vast and hope to witness increased utilization of the algorithms and theoretical results established in the domain of learning and adaptation over networks.

Choosing the optimal strategy for a given problem is equivalent to selecting the task-relatedness model that best fits the underlying problem.

Authors

Roula Nassif (roula.nassif@aub.edu.lb) received her Dipl.-Ing. and M.S. degrees in industrial control and electrical engineering in 2013 from Lebanese University, Beirut, and Compiègne University of Technology, France, respectively, and her Ph.D. degree in electrical and computer engineering in 2016 from the University of Côte d'Azur, Nice, France. She is an assistant professor at American University of Beirut (AUB), Lebanon. Prior to joining AUB, she was a postdoctoral scholar at the Adaptive Systems Laboratory, École Polytechnique Fédérale de Lausanne, Switzerland. Her current research interests include adaptation and learning over networks.

Stefan Vlaski (stefan.vlaski@epfl.ch) received his B.Sc. degree in electrical engineering from Technical University Darmstadt, Germany, in 2013 and his M.S. degree in electrical engineering and Ph.D. degree in electrical and computer engineering from the University of California, Los Angeles (UCLA) in 2014 and 2019, respectively. He is currently a postdoctoral researcher at the Adaptive Systems Laboratory, École Polytechnique Fédérale de Lausanne, Switzerland. He was a recipient of the German National Scholarship at Technical University Darmstadt and the Graduate Division Fellowship at UCLA. His research interests include machine learning, signal processing, and optimization with a particular emphasis on adaptive and decentralized solutions.

Cédric Richard (cedric.richard@unice.fr) received his Dipl.-Ing. and M.S. degrees in 1994 and his Ph.D. degree in 1998, all in electrical and computer engineering from the Compiègne University of Technology, France. He is a full professor at Université Côte d'Azur, Nice, France. From 2010 to 2015, he was distinguished as a member of the Institut Universitaire de France. He is the author of more than 300 papers. He is the director-at-large of Region 8 (Europe, Middle East, Africa) of the IEEE Signal Processing Society (SPS) and a member of the Board of Governors of the SPS. He is also the director of the French Federal CNRS research association Information, Signal, Image, Vision. His current research interests include statistical signal processing and machine learning.

Jie Chen (dr.jie.chen@ieee.org) received his Dipl.-Ing. and M.S. degrees in 2009 from Troyes University of Technology (UTT), France, and from Xi'an Jiaotong University, respectively, both in information and telecommunication engineering. In 2013, he received his Ph.D. degree

from UTT. From 2013 to 2014, he was with Lagrange Laboratory, University of Nice Sophia Antipolis, France, and from 2014 to 2015, he was with the Department of Electrical Engineering and Computer Science at the University of Michigan. He is currently a professor at Northwestern Polytechnical University, Xi'an, China. He received the "Thousand Talents Plan (Youth Program)" Award in China. His current research interests include adaptive signal processing and distributed optimization.

Ali H. Sayed (ali.sayed@epfl.ch) is the dean of engineering at École Polytechnique Fédérale de Lausanne, Switzerland. He has served as distinguished professor and former chairman of electrical engineering at the University of California, Los Angeles. He is a member of the U.S. National Academy of Engineering and was recognized as a highly cited researcher. He is the author of more than 530 scholarly publications and six books, and his work has been recognized with several awards. His research interests include adaptation and learning, data and network sciences, distributed optimization, and statistical inference methods. He was the president of the IEEE Signal Processing Society from 2018 to 2019.

References

[1] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, July 1997. doi: 10.1023/A:1007379606734.

[2] S. Thrun and L. Pratt, *Learning to Learn*. Norwell, MA: Kluwer Academic Publishers, 1998.

[3] Y. Zhang and D. Yeung, "A convex formulation for learning task relationships in multi-task learning," in *Proc. 26th Conf. Uncertainty Artificial Intelligence*, Catalina Island, CA, July 2010, pp. 733–742.

[4] X. Chen, S. Kim, Q. Lin, J. G. Carbonell, and E. P. Xing, Graph-structured multitask regression and an efficient optimization method for general fused Lasso. May 2010. [Online]. Available: arXiv:1005.3579

[5] T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *Proc. 10th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, 2004, pp. 109–117. doi: 10.1145/1014052.1014067.

[6] L. Jacob, F. Bach, and J.-P. Vert, "Clustered multi-task learning: A convex formulation," in *Proc. 21st Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2008, pp. 745–752.

[7] T. Kato, H. Kashima, M. Sugiyama, and K. Asai, "Multi-task learning via conic programming," in *Proc. 20th Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2007, pp. 737–744.

[8] J. Wang, M. Kolar, and N. Srebro, "Distributed multi-task learning," in *Proc. Conf. Artificial Intelligence and Statistics*, Cadiz, Spain, 2016, pp. 751–760.

[9] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Found. Trends Mach. Learn.*, vol. 7, no. 4–5, pp. 311–801, 2014. doi: 10.1561/2200000051.

[10] A. H. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z. J. Towfic, "Diffusion strategies for adaptation and learning over networks," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 155–171, May 2013. doi: 10.1109/MSP.2012.2231991.

[11] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Upper Saddle River, NJ: Prentice Hall, 1985.

[12] R. Nassif, S. Vlaski, C. Richard, and A. H. Sayed, Learning over multitask graphs–Part I: Stability analysis. May 2018. [Online]. Available: arXiv:1805.08535

[13] R. Nassif, S. Vlaski, and A. H. Sayed, Adaptation and learning over networks under subspace constraints–Part II: Performance analysis. May 2019. [Online]. Available: arXiv:1906.12250

[14] V. Kekatos and G. B. Giannakis, "Distributed robust power system state estimation," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1617–1626, May 2013. doi: 10.1109/TPWRS.2012.2219629.

[15] A. Koppel, B. M. Sadler, and A. Ribeiro, "Proximity without consensus in online multiagent optimization," *IEEE Trans. Signal Process.*, vol. 65, no. 12, pp. 3062–3077, June 2017. doi: 10.1109/TSP.2017.2686368.

[16] R. Nassif, S. Vlaski, C. Richard, J. Chen, and A. H. Sayed, Multitask learning over graphs. 2019. [Online]. Available: arXiv:2001.02112

[17] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Comput. Netw.*, vol. 38, no. 4, pp. 393–422, 2002. doi: 10.1016/S1389-1286(01)00302-4.

[18] L. Grady and J. R. Polimeni, *Discrete Calculus*. Berlin: Springer-Verlag, 2010.

[19] D. Zhou and B. Schölkopf, "A regularization framework for learning from graph data," in *Proc. ICML Workshop Statistical Relational Learning and Connections Other Fields*, vol. 15, pp. 67–68, 2004.

[20] D. P. Bertsekas, "A new class of incremental gradient methods for least squares problems," *SIAM J. Optim.*, vol. 7, no. 4, pp. 913–926, Nov. 1997. doi: 10.1137/S1052623495287022.

[21] W. Wang, J. Wang, M. Kolar, and N. Srebro, Distributed stochastic multi-task learning with graph regularization. 2018. [Online]. Available: arXiv:1802.03830

[22] F. R. K. Chung, *Spectral Graph Theory*. Providence, RI: American Mathematical Society, 1997.

[23] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning Theory and Kernel Machines*, B. Schölkopf and M. K. Warmuth, Eds. Berlin: Springer-Verlag, 2003, pp. 144–158.

[24] R. Nassif, S. Vlaski, C. Richard, and A. H. Sayed, "A regularization framework for learning over multitask graphs," *IEEE Signal Process. Lett.*, vol. 26, no. 2, pp. 297–301, Feb. 2019. doi: 10.1109/LSP.2018.2889267.

[25] N. J. Higham, *Functions of Matrices: Theory and Computation*. Philadelphia: SIAM, 2008.

[26] D. I. Shuman, P. Vandergheynst, D. Kressner, and P. Frossard, "Distributed signal processing via Chebyshev polynomial approximation," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 4, pp. 736–751, Dec. 2018. doi: 10.1109/TSIPN.2018.2824239.

[27] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042–3054, 2014. doi: 10.1109/TSP.2014.2321121.

[28] 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, Dec. 2016. doi: 10.1109/TSP.2016.2601282.

[29] D. Hallac, J. Leskovec, and S. Boyd, "Network Lasso: Clustering and optimization in large graphs," in *Proc. 21th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Sydney, Australia, Aug. 2015, pp. 387–396. doi: 10.1145/2783258.2783313.

[30] Y.-X. Wang, J. Sharpnack, A. J. Smola, and R. J. Tibshirani, "Trend filtering on graphs," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 3651–3691, 2016.

[31] P. D. Lorenzo, S. Barbarossa, and S. Sardellitti, "Distributed signal recovery based on in-network subspace projections," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Brighton, U.K., May 2019, pp. 5242–5246. doi: 10.1109/ICASSP.2019.8682719.

[32] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, p. 48, Jan. 2009. doi: 10.1109/TAC.2008.2009515.

[33] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Püschel, "Distributed optimization with local domains: Applications in MPC and network flows," *IEEE Trans. Autom. Control*, vol. 60, no. 7, pp. 2004–2009, July 2015. doi: 10.1109/TAC.2014.2365686.

[34] S. A. Alghunaim and A. H. Sayed, "Distributed coupled multi-agent stochastic optimization," *IEEE Trans. Autom. Control*, vol. 65, no. 1, pp. 175–190, 2020. doi: 10.1109/TAC.2019.2906495.

[35] J. Plata-Chaves, N. Bogdanović, and K. Berberidis, "Distributed diffusion-based LMS for node-specific adaptive parameter estimation," *IEEE Trans. Signal Process.*, vol. 63, no. 13, pp. 3448–3460, 2015. doi: 10.1109/TSP.2015.2423256.

[36] A. K. Sahu, D. Jakovetić, and S. Kar, $\mathcal{CIRFE}$: A distributed random fields estimator," *IEEE Trans. Signal Process*, vol. 66, no. 18, pp. 4980–4995, Sept. 2018. doi: 10.1109/TSP.2018.2863646.

[37] F. K. Teklehaymanot, M. Muma, B. Béjar, P. Binder, A. Zoubir, and M. Vetterli, "Robust diffusion-based unsupervised object labelling in distributed camera networks," in *Proc. IEEE Africon 2015*, Sept. 2015, pp. 1–6. doi: 10.1109/AFRCON.2015.7331863.

[38] X. Zhao and A. H. Sayed, "Distributed clustering and learning over networks," *IEEE Trans. Signal Process.*, vol. 63, no. 13, pp. 3285–3300, July 2015. doi: 10.1109/TSP.2015.2415755.

[39] J. Chen, C. Richard, and A. H. Sayed, "Diffusion LMS over multitask networks," *IEEE Trans. Signal Process.*, vol. 63, no. 11, pp. 2733–2748, June 2015. doi: 10.1109/TSP.2015.2412918.

[40] J. Plata-Chaves, A. Bertrand, M. Moonen, S. Theodoridis, and A. M. Zoubir, "Heterogeneous and multitask wireless sensor networks," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 3, pp. 450–465, 2017. doi: 10.1109/JSTSP.2017.2676468.

SP