Distributed Learning of Predictive Structures From Multiple Tasks Over Networks

Junhao Hua, Chunguang Li, *Senior Member, IEEE*, and Hui-Liang Shen

***Abstract*—This paper is concerned with the problem of distributed multitask learning over networks, which aims to simultaneously infer multiple node-specific parameter vectors in a collaborative manner. Most of the existing works on the distributed multitask problem modeled the task relatedness by assuming some similarities of parameter vectors in an explicit way. In this paper, we implicitly model the similarity of parameter vectors by assuming that the parameter vectors share a common low-dimensional predictive structure on hypothesis spaces, which is learned using the available data in networks. A distributed structure learning algorithm for the in-network cooperative estimation problem is then derived based on the block coordinate descent method integrated with the inexact alternating direction method of multipliers technique. Simulations on both synthetic and real-world datasets are given to verify the effectiveness of the proposed algorithm. In the case that each node shares a common predictive subspace, it is demonstrated that the proposed multitask algorithm outperforms the noncooperative learning algorithm. Moreover, the use of the inexact approach can significantly reduce the communication bandwidth and still provide the same optimal solution as the corresponding centralized approach.**

***Index Terms*—Alternating direction method of multipliers (ADMM), data-driven, distributed estimation, multitask learning, wireless sensor network.**

I. Introduction

SENSOR networks are composed of a large number of small low-power drives (nodes) distributedly deployed in resource-limited environments to execute some tasks, such as detection, tracking, and object classification [1], [2]. In recent years, the problem of distributed estimation over sensor networks, where the nodes cooperatively estimate some parameter of interest using local noisy measurements and information obtained from one-hop neighbors, has attracted significant attention [3]–[6]. Compared with the centralized estimation, the distributed estimation scheme does not need a powerful fusion center, so it is more flexible and provides robustness to node and/or link failures. In the fully distributed scheme, each node only communicates with its one-hop neighbors, which will help save energy and bandwidth. Due to these merits, distributed estimation has been used in a wide range of fields, including environmental monitoring [7], industrial automation [8], and military surveillance [9]. A plenty of algorithms have been proposed for distributed estimation over networks, such as consensus-based [10], diffusion-based [3], and alternating direction method of multipliers (ADMM)-based algorithms [11].

The previous studies on distributed estimation problems have intensively focused on the scenarios where all nodes in a network collaboratively estimate a single parameter vector [12]–[14]. The problems of this type are referred to as single-task problems. Recently, a handful of studies extend this setting to the multitask problems [15]–[17]. In this setting, multiple optimum parameter vectors to be inferred simultaneously by the network are different but related. In fact, in many important real-world in-network applications including pattern classification and regression, the tasks among nodes may be different and the number of available measurements for each task is quite limited. If the network learns each task independently, the estimation performance of the resulting predictors may be very poor. Alternatively, by learning these multiple tasks simultaneously, the relatedness of tasks among nodes can be exploited to improve estimation accuracy.

Previous works [15], [16] on distributed multitask estimation have modeled the task relatedness through assuming that the optimum parameters at two connected nodes are close to each other in an Euclidean norm. However, this assumption cannot be always satisfied in practical applications. In machine learning community, a plenty of works focus on learning a low-dimensional feature representation from multiple tasks [18]–[21]. This framework has been successfully applied in several machine learning applications, such as natural language tagging [18], optical character recognition [20], and automated images annotation [21]. In this paper, we focus on developing a distributed multitask learning (dMTL) algorithm based on this framework for sensor networks.

Specifically, we model the task relatedness by assuming that multiple tasks among all nodes share a low-dimensional predictive structure on hypothesis spaces [18], [21]. This assumption works in many practical applications. For example, suppose that each node in a network has its own image classification task, although the natural scene images obtained from different nodes

Manuscript received January 30, 2016; revised May 19, 2016; accepted June 11, 2016. Date of publication July 7, 2016; date of current version April 10, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61571392, Grant 61371160, and Grant 61471320, in part by the National Program for Special Support of Eminent Professionals, and in part by the Fundamental Research Funds for the Central Universities under Grant 2016QNA5004. *(Corresponding author: Chunguang Li.)*

The authors are with the College of Information Science and Electronic Engineering, Zhejiang University, and also with the Zhejiang Provincial Key Laboratory of Information Processing, Communication and Networking, Hangzhou 310027, China (e-mail: huajh@zju.edu.cn; cgli@zju.edu.cn; shenhl@zju.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2016.2588463

may belong to different categories, the underlying image patterns for all nodes may have similar feature representations in the form of a linear combination of certain bases. With this assumption, we expect that by cooperatively learning these tasks over a network, we can find some shared low-dimensional structures with highest predictive power to improve estimation and generalization performance of each single task. In the distributed estimation context, structure learning from multiple tasks has not been well studied yet. One relevant study [17] considers the online multitask learning where the node hypothesis spaces partially overlap. However, it assumes that the overlap of hypothesis subspaces is known in advance, which limits its applicability.

In this paper, we consider the situation where the shared predictive structure is unknown, and we learn it only using the available data from all nodes. The aim of this paper is to enhance estimation performance in a multitask network by learning the common predictive structure with distributed strategies based on local data and information exchange between one-hop neighboring nodes. Specifically, we formulate a distributed estimation framework for learning common predictive structures from multiple tasks in a sensor network, and then derive a computationally inexpensive and energy-saving dMTL algorithm for this framework based on the block coordinate descent (BCD) method integrating with the inexact ADMM iterations.

The rest of this paper is organized as follows. In Section II, we formulate the problem of distributed multitask estimation over networks. In Sections III and IV, we propose the dMTL algorithm, and present the detailed computations of subproblems, respectively. In Section V, numerical simulations are presented to illustrate the effectiveness and advantages of the proposed algorithm. Finally, conclusion is drawn in Section VI.

II. Problem Formulation

A. Network Model

We consider a connected sensor network consisting of N nodes distributed over a geographic region. We use graphs to represent networks. The considered undirected graph without a self-loop $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $\mathcal{V} = \{1, 2, \ldots, N\}$ and a set of edges $\mathcal{E}$, where each edge $(k, l) \in \mathcal{E}$ connects an unordered pair of distinct nodes. For each node $k \in \mathcal{V}$, let $\mathcal{N}_k = \{l | (k, l) \in \mathcal{E}, k \neq l\}$ be the set of its neighboring nodes (excluding node k itself). Each node k collects/stores a small number (N_k) of measurements $\{(\boldsymbol{x}_{k,i}, \boldsymbol{y}_{k,i})\}_{i=1}^{N_k} \subset \mathbb{R}^p \times \mathbb{R}$ with $\boldsymbol{x}_{k,i}$ denoting an input vector and $\boldsymbol{y}_{k,i}$ denoting corresponding scalar output.

The task of each node k is to seek a predictor f_k that maps each input vector $\boldsymbol{x}_{k,i}$ to the corresponding output $\boldsymbol{y}_{k,i}$. For simplicity, we employ linear prediction models for all tasks. At each node k, a linear predictor is defined as

$$f_k(\boldsymbol{x}) := \boldsymbol{u}_k^T \boldsymbol{x} \tag{1}$$

where $\boldsymbol{u}_k \in \mathbb{R}^p$ is a weight vector. Thus, the task of each node k becomes to seek the optimum parameter vector $\boldsymbol{u}_k$ using the input and output data $\{(\boldsymbol{x}_{k,i}, \boldsymbol{y}_{k,i})\}$.

B. Formulation of dMTL

As discussed in Section I, previous works on distributed estimation intensively studied the problem of the single task learning, in which all nodes in a network collaboratively estimate a common parameter vector, i.e., $\boldsymbol{u}_k = \boldsymbol{u}^* \forall k$. In contrast, the problem considered here assumes that the tasks across all nodes are different, i.e., $\boldsymbol{u}_k \neq \boldsymbol{u}_j \forall k, j \in \mathcal{V}$, but related to some extent. We learn these tasks simultaneously and exploit the task relatedness to enhance the estimation performance of each node.

In [18], the researchers proposed the alternating structure optimization algorithm for learning predictive functional structures from multiple related tasks. It learns all N predictors $\{f_1, \ldots, f_N\}$ simultaneously by exploiting a shared feature space in a simple linear form of low-dimensional feature map Θ across the N tasks. Following this approach, we express the prediction function f_k as

$$f_k(\boldsymbol{x}) = \boldsymbol{u}_k^T \boldsymbol{x} = \boldsymbol{w}_k^T \boldsymbol{x} + \boldsymbol{v}_k^T \Theta \boldsymbol{x} = (\boldsymbol{w}_k + \Theta^T \boldsymbol{v}_k)^T \boldsymbol{x} \tag{2}$$

where $\boldsymbol{w}_k \in \mathbb{R}^p$, $\boldsymbol{v}_k \in \mathbb{R}^h$ $(h < p)$ are weight vectors specific for each node, and $\Theta \in \mathbb{R}^{h \times p}$ is the common structure parameter with orthonormal constraint

$$\Theta \Theta^T = \mathbf{I}_h. \tag{3}$$

To find predictors $\{\hat{f}_1, \ldots, \hat{f}_N\}$ simultaneously, we could minimize a joint empirical risk function using the measurements $\{(\boldsymbol{x}_{k,i}, \boldsymbol{y}_{k,i})\}_{i=1}^{N_k}, k \in \mathcal{V}$. Formally, we define a local empirical loss function at node k as

$$L_k(\boldsymbol{u}_k) = \frac{1}{N_k} \sum_{i=1}^{N_k} L(\boldsymbol{u}_k^T \boldsymbol{x}_{k,i}, \boldsymbol{y}_{k,i}) \tag{4}$$

where the loss function $L(\cdot)$ is assumed to be convex and smooth for simplicity. The regularization method is then applied to encode our belief on what values the structure parameter Θ and model parameter vectors $\{\boldsymbol{u}_k\}$ are. There are many possible choices of regularization functions, such as l_2-norm [18], $l_{2,1}$-norm [19], and elastic net [22]. We use the regularization function proposed in [21] for our formulation. Mathematically, the optimization for the multitask learning problem is formulated as

$$\begin{aligned} \min_{\{\boldsymbol{u}_k, \boldsymbol{v}_k\}, \Theta} \quad & \sum_{k=1}^{N} \Big(L_k(\boldsymbol{u}_k) + g_k(\boldsymbol{u}_k, \boldsymbol{v}_k, \Theta) \Big) \\ \text{s.t.} \quad & \Theta \Theta^T = \mathbf{I}_h \end{aligned} \tag{5}$$

where the regularization function $g_k(\boldsymbol{u}_k, \boldsymbol{v}_k, \Theta)$ is defined as

$$g_k(\boldsymbol{u}_k, \boldsymbol{v}_k, \Theta) = \alpha_k \|\boldsymbol{u}_k - \Theta^T \boldsymbol{v}_k\|^2 + \beta_k \|\boldsymbol{u}_k\|^2. \tag{6}$$

The first component $\|\boldsymbol{u}_k - \Theta^T \boldsymbol{v}_k\|^2 = \|\boldsymbol{w}_k\|^2$ represents the task relatedness among N tasks, and the second component $\|\boldsymbol{u}_k\|^2$ controls the complexity of the predictor function f_k. The prespecified coefficients α_k, β_k indicate the importance of the corresponding regularization components, respectively. For simplicity, we use the same $\beta = \beta_1 = \cdots = \beta_N > 0$ and $\alpha = \alpha_1 = \cdots = \alpha_N > 0$ for all tasks.

C. Convex Relaxation

Note that the formulation (5) is nonconvex due to its orthonormal constraint and the regularization term. Following [21], we convert the nonconvex formulation (5) into a relaxed convex one, which admits a globally optimal solution.

We first consider the orthonormal constraint in (5), which is nonconvex. We define $\mathcal{M}_e = \{M \in \mathbb{R}^{p\times p} | M = \Theta^T\Theta, \Theta\Theta^T = \mathbf{I}_h, \Theta \in \mathbb{R}^{h\times p}\}$. It is known that the convex hull of $\mathcal{M}_e$ can be precisely expressed as the convex set [23]

$$\mathcal{M} = \{M \in \mathbb{R}^{p\times p} | \mathrm{tr}(M) = h, \mathbf{0} \preceq M \preceq \mathbf{I}_p\}. \tag{7}$$

Each element in $\mathcal{M}_e$ is referred to as an extreme point of $\mathcal{M}$, and $\mathcal{M}$ is the smallest convex set that contain $\mathcal{M}_e$.

We next transform the regularization term. With fixed $\boldsymbol{u}_k$ and Θ, it can be verified that the optimal value $\boldsymbol{v}_k^*$ in the problem (5) is achieved at

$$\boldsymbol{v}_k^* = \Theta\boldsymbol{u}_k = \arg\min_{\boldsymbol{v}_k} g_k(\boldsymbol{u}_k, \boldsymbol{v}_k, \Theta). \tag{8}$$

By eliminating $\boldsymbol{v}_k$, we rewrite the regularization term (6) as

$$\begin{aligned} G_k^0(\boldsymbol{u}_k, \Theta) &= g_k(\boldsymbol{u}_k, \boldsymbol{v}_k^*, \Theta) \\ &= \alpha\boldsymbol{u}_k^T\big((1+\eta)\mathbf{I}_p - \Theta^T\Theta\big)\boldsymbol{u}_k \end{aligned} \tag{9}$$

where $\eta := \beta/\alpha > 0$.

Moreover, using the fact that $(1+\eta)\mathbf{I}_p - \Theta^T\Theta = \eta(1+\eta)(\eta\mathbf{I}_p + \Theta^T\Theta)^{-1}$, which can be verified using the matrix inversion lemma [24] and the orthonormal constraint in (5), we first reformulate $G_k^0(\boldsymbol{u}_k, \Theta)$ into an equivalent form given by

$$G_k^1(\boldsymbol{u}_k, \Theta) = \alpha\eta(1+\eta)\boldsymbol{u}_k^T(\eta\mathbf{I}_p + \Theta^T\Theta)^{-1}\boldsymbol{u}_k. \tag{10}$$

We then make a relaxation by replacing $\Theta^T\Theta$ in (10) with $M \in \mathcal{M}$ in (7) and get a new regularization function

$$G_k(\boldsymbol{u}_k, M) := \alpha\eta(1+\eta)\boldsymbol{u}_k^T(\eta\mathbf{I}_p + M)^{-1}\boldsymbol{u}_k. \tag{11}$$

It follows from [25, Th. 3.1] that $G_k(\boldsymbol{u}_k, M)$ is jointly convex in $\boldsymbol{u}_k$ and M.

Thus, the nonconvex problem (5) is converted into a relaxed convex optimization problem

$$\min_{\{\boldsymbol{u}_k\}, M\in\mathcal{M}} \mathcal{R}(\{\boldsymbol{u}_k\}, M) \tag{12}$$

where the objective function $\mathcal{R} : \mathbb{R}^{p\times N} \times \mathbb{S}_+^p \to \mathbb{R}$ is

$$\mathcal{R}(\{\boldsymbol{u}_k\}, M) = \sum_{k=1}^N \Big(L_k(\boldsymbol{u}_k) + G_k(\boldsymbol{u}_k, M)\Big). \tag{13}$$

The optimization problem (12) has a larger feasible domain set compared to the problem (5), since the construction $M = \Theta^T\Theta$ is guaranteed to be feasible in (12), while a specific M may not be decomposed to $\Theta^T\Theta$ such that Θ is feasible in (5). Nevertheless, the optimal Θ of (5) can be approximated using the first h eigenvectors of the optimal M computed from the problem (12).

In the next section, we use the BCD method to minimize the objective function, whose convergence is guaranteed by the condition that the objective function is jointly strictly convex in $\{\boldsymbol{u}_k\}$ and M. We present a further modification of the objective function (13) by introducing a small perturbation, which is similar with that in [19] and [25]. The perturbed objective function $\mathcal{R}_\epsilon : \mathbb{R}^{p\times N} \times \mathbb{S}_+^p \to \mathbb{R}$ is given by

$$\mathcal{R}_\epsilon(\{\boldsymbol{u}_k\}, M) = \sum_{k=1}^N L_k(\boldsymbol{u}_k) + G_\epsilon(\{\boldsymbol{u}_k\}, M) \tag{14}$$

where the regularization term is defined as

$$\begin{aligned} G_\epsilon(\{\boldsymbol{u}_k\}, M) := \ & \alpha\eta(1+\eta)\mathrm{tr}\Big((\epsilon\mathbf{I}_p + \sum_{k=1}^N \boldsymbol{u}_k\boldsymbol{u}_k^T) \\ & \times (\eta\mathbf{I}_p + M)^{-1}\Big) \end{aligned} \tag{15}$$

and the small parameter $\epsilon > 0$. The objective function $\mathcal{R}_\epsilon$ approaches $\mathcal{R}$ as $\epsilon \to 0$. The function $f(M) = \mathrm{tr}((\eta\mathbf{I}_p + M)^{-1})$ is strictly convex, since $\eta\mathbf{I}_p + M$ is positive definite and $f(X) = \mathrm{tr}(X^{-1})$ is strictly convex on positive definite matrices [24]. Based on this and following the fact that $G_k(\cdot,\cdot)$ is jointly convex in $\boldsymbol{u}_k$ and M and strictly convex in $\boldsymbol{u}_k$, we conclude that the perturbed function G_ϵ is jointly strictly convex in $\{\boldsymbol{u}_k\}$ and M. Therefore, for any convex smooth loss function $L_k(\cdot)$, the objective function $\mathcal{R}_\epsilon(\cdot,\cdot)$ has a unique minimizer.

III. Distributed Multitask Learning

Before presenting the distributed algorithm for the problem (12) in the considered network, we would like to first present the centralized algorithm, in which the data from every node can be gathered in a fusion center, and the computation is performed in the fusion center. After that, we propose the distributed algorithm of the multitask learning problem over networks.

A. Centralized Multitask Algorithm

Based on the BCD method [26], we alternatively minimize the perturbed objective function $\mathcal{R}_\epsilon$ with respect to $\{\boldsymbol{u}_k\}$ and M, and the algorithm can be written as the following two steps

$$\begin{aligned} \boldsymbol{u}_k^t &= \arg\min_{\boldsymbol{u}_k} \mathcal{R}_\epsilon(\{\boldsymbol{u}_k\}, M^{t-1}) \quad \forall k \in \mathcal{V} \\ &= \arg\min_{\boldsymbol{u}_k} L_k(\boldsymbol{u}_k) + G_k(\boldsymbol{u}_k, M^{t-1}) \quad \forall k \in \mathcal{V} \end{aligned} \tag{16a}$$

$$\begin{aligned} M^t &= \arg\min_{M\in\mathcal{M}} \mathcal{R}_\epsilon(\{\boldsymbol{u}_k^t\}, M) \\ &= \arg\min_{M\in\mathcal{M}} G_\epsilon(\{\boldsymbol{u}_k^t\}, M) \\ &= \arg\min_{M\in\mathcal{M}} \mathrm{tr}\Big((\epsilon\mathbf{I}_p + \sum_{k=1}^N \boldsymbol{u}_k^t\boldsymbol{u}_k^{t,T})(\eta\mathbf{I}_p + M)^{-1}\Big) \end{aligned} \tag{16b}$$

where $t = 1, 2, \ldots$ is an iteration step.

We now present some convergence properties of (16). First, by construction, we observed that the values of the objective are nonincreasing, that is,

$$\mathcal{R}_\epsilon(\{\boldsymbol{u}_k^{t-1}\}, M^{t-1}) \geq \mathcal{R}_\epsilon(\{\boldsymbol{u}_k^t\}, M^{t-1}) \geq \mathcal{R}_\epsilon(\{\boldsymbol{u}_k^t\}, M^t).$$

These values are bounded, since $L_k(\cdot)$ is bounded from below, and thus, the iterates of the objective function converge. Moreover, the sequences $\{(\{\boldsymbol{u}_k^t\}, M^t)\}$ also converge as stated in the following theorem.

Theorem 1: For every $\epsilon > 0$, let $\{(\{\boldsymbol{u}_k^t\}, M^t)\}$ be the sequence generated by (16a) and (16b). Then, the whole sequence $\{(\{\boldsymbol{u}_k^t\}, M^t)\}$ converges to a unique global minimizer of $\mathcal{R}_\epsilon$ subject to the constraints $M \in \mathcal{M}$.

Based on [26, Proposition 2.7.1] and the joint strictly convex property of the objective function $\mathcal{R}_\epsilon$, Theorem 1 can be easily

proved. From the definition of $\mathcal{R}_\epsilon$, it is clear that the minimal value of $\mathcal{R}_\epsilon$ decreases with the decreasing of ϵ and converges to the minimal value of $\mathcal{R}$ as $\epsilon \to 0$. Moreover, since $\mathcal{R}_\epsilon$ is continuous in ϵ, $\{\boldsymbol{u}_k\}$ and M, the minimizer $\{(\{\boldsymbol{u}_k^*\}, M^*)\}$ of $\mathcal{R}_\epsilon$ converges to the set of the minimizers of $\mathcal{R}$ as $\epsilon \to 0$. To ensure the joint strict convexity of $\mathcal{R}_\epsilon$, meanwhile, keeping the minimizer of $\mathcal{R}_\epsilon$ close to that of $\mathcal{R}$, we set ϵ to an extremely small value in the following.

B. dMTL Framework

In this section, we extend the centralized algorithm (16) to the distributed scenario where every node in the network only communicates with its one-hop neighboring nodes. The main difficulty in doing this is that the updating of the common structure parameter M in (16b) needs all parameter vectors $\{\boldsymbol{u}_k\}$, which is nontrivial in a fully distributed network. Consider replacing the common structure parameter M in (14) with auxiliary per-node variables $\{M_k\}_{k=1}^N$, thus we need to make an agreement on the parameters $\{M_k\}$ among all nodes.

We propose a computationally inexpensive and energy saving strategy to deal with this problem. The key observation is that the computation of (16b) only depends on an intermediate quantity

$$\bar{Z} := \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{u}_k \boldsymbol{u}_k^T \in \mathbb{R}^{p\times p} \tag{17}$$

which is an average of all local quantities $\{\boldsymbol{u}_k \boldsymbol{u}_k^T\}_{k=1}^N$. If a cyclic path through all the nodes could be found, then an incremental algorithm can be derived by collecting all local quantities through the path. However, it is not practical in a low-cost networked system, since the communication resources are limited and exploring the network topology is hard and expensive. Note that the optimal value of the following optimization problem is equal to $\bar{Z}$ in (17):

$$Z^* = \arg\min_{Z} \sum_{k=1}^{N} \|Z - \boldsymbol{u}_k \boldsymbol{u}_k^T\|_F^2 \tag{18}$$

where $\|\cdot\|_F$ is the Frobenius norm. This can be reformulated to an equivalent distributed form. Let us define a set of per-node intermediate quantities $\{Z_k\}_{k=1}^N$, and add consensus constraints to force these variables to agree across neighboring nodes. Thus, the optimization problem in (18) is recast as the consensus-based distributed problem

$$\begin{aligned} \min_{\{Z_k\},\{W_{kj}\}} \quad & \sum_{k=1}^{N} \|Z_k - \boldsymbol{u}_k \boldsymbol{u}_k^T\|_F^2 \\ \text{s.t.} \quad & Z_k = W_{kj}, W_{kj} = Z_j \quad \forall k \in \mathcal{V}, j \in \mathcal{N}_k \end{aligned} \tag{19}$$

where the auxiliary variables $W_{kj} \in \mathbb{R}^{p\times p}$ decouple local variables Z_k at node k from those of their neighbors $Z_j, j \in \mathcal{N}_k$. This formulation is equivalent with (18) as stated in the following lemma [27], [29].

Lemma 1: If the graph $\mathcal{G}$ is connected, the problem (18) and (19) are equivalent, that is, $Z_k^* = Z^*, \forall k \in \mathcal{V}$, where $\{Z_k^*\}$ is the solution of (19) and Z^* is the solution of (18).

Proof: Consider any two nodes $k_0, k_l \in \mathcal{V}$. Since the network is connected, there exists a path $\{k_0 k_1, \ldots, k_{l-1} k_l\}$ connecting nodes k_0 and k_l. Because $k_{i+1} \in \mathcal{N}_{k_i}$ for $i = 0, 1, \ldots, l-1$, we have $\{Z_{k_i} = W_{k_i k_{i+1}}, W_{k_i k_{i+1}} = Z_{k_{i+1}}\}$. It is immediate that $Z_{k_0} = Z_{k_1} = \cdots = Z_{k_l}$. Since k_0, k_l are arbitrary, it follows that $Z_1 = Z_2 = \cdots = Z_N$. As any feasible solution of (19) satisfies $Z_1 = Z_2 = \cdots = Z_N = Z$, problem (19) becomes (18). ■

Based on the distributed formulation (19), we now present a distributed algorithm in replace of the centralized one (16) to minimize the perturbed objective function $\mathcal{R}_\epsilon$. The proposed distributed BCD algorithm executes the following three steps iteratively until convergence:

$$\boldsymbol{u}_k^t = \arg\min_{\boldsymbol{u}_k} L_k(\boldsymbol{u}_k) + G_k(\boldsymbol{u}_k, M_k^{t-1}) \quad \forall k \in \mathcal{V} \tag{20a}$$

$$\begin{aligned} \{Z_k^t\} = & \arg\min_{\{Z_k\},\{W_{kj}\}} \sum_{k=1}^{N} \|Z_k - \boldsymbol{u}_k^t \boldsymbol{u}_k^{t,T}\|_F^2 \\ \text{s.t.} \quad Z_k = & W_{kj}, W_{kj} = Z_j \quad \forall k \in \mathcal{V}, j \in \mathcal{N}_k \end{aligned} \tag{20b}$$

$$\begin{aligned} M_k^t = & \arg\min_{M_k} \operatorname{tr}\big((\epsilon \mathbf{I}_p + Z_k^t)(\eta \mathbf{I}_p + M_k)^{-1}\big) \\ \text{s.t.} \quad \operatorname{tr}(M_k) = & h, \mathbf{0} \preceq M_k \preceq \mathbf{I}_p \quad \forall k \in \mathcal{V}. \end{aligned} \tag{20c}$$

Note that (20a) and (20c) are carried out locally at each node, and the computation of (20b) needs the collaborations among neighboring nodes and hence costs communication resources. In detail, at step (20a), every node k learns its parameter vector $\boldsymbol{u}_k$ based on its own structure parameter M_k. After that, each node k collaboratively computes its own intermediate quantity Z_k by exchanging information with its neighboring nodes. If the solution of (20b) is exact, all intermediate quantities $\{Z_k\}$ are equal to $\bar{Z}$ in (17) as stated in Lemma 1. Thus, the optimums of $\{M_k\}$ at step (20c) among all nodes are also equal. Therefore, we can obtain the minimizer of $\mathcal{R}_\epsilon$ via the distributed procedure (20).

However, at every BCD round, if we compute the solution of (20b) exactly, it may lead to communication overhead, since it may need many iterations to reach the exact one. Therefore, to save energy and bandwidth, we would like to compute the inexact but good enough solution of (20b), which will be presented and discussed in the next section.

IV. Computation of Subproblems of Distributed Multitask Algorithm

We now present the detailed algorithms to solve the minimization subproblems (20a), (20b), and (20c). Note that we have used the subscript t to denote the iteration step (outer iteration) of the BCD iterations. If iterative methods are applied to solving the subproblems at iteration t, we use the subscript l to denote the lth iteration (inner iteration) and omit t for notational simplicity.

Using previous estimates $\{M_k^{t-1}\}$, the computation of (20a) is carried out locally at every node. We apply accelerated gradient descent (AGD) methods [28] to solving (20a). For any convex smooth loss function $L_k(\cdot)$, the objective function in (20a) is strictly convex, and hence, the corresponding

Algorithm 1: Solving (20a) via the APG Method.

Input: Given M_k^{t-1} and $\boldsymbol{u}_k^{(0)} = \boldsymbol{u}_k^{(-1)} = \mathbf{0}$.

for $l = 1, 2, \ldots,$ **do**
  τ_l is a fixed step size or determined by line search.
  $\boldsymbol{u}_k^{(l-\frac{1}{2})} = \boldsymbol{u}_k^{(l-1)} + \frac{l-2}{l+1}(\boldsymbol{u}_k^{(l-1)} - \boldsymbol{u}_k^{(l-2)})$.
  $\boldsymbol{u}_k^{(l)} = \boldsymbol{u}_k^{(l-\frac{1}{2})} - 2\tau_l\alpha\big((1+\eta)\mathbf{I}_p - M_k^{t-1}\big)\boldsymbol{u}_k^{(l-\frac{1}{2})}$
    $-\tau_l \nabla_{\boldsymbol{u}_k} L_k(\boldsymbol{u}_k)\big|_{\boldsymbol{u}_k^{(l-\frac{1}{2})}}$.
  if the stopping criterion is satisfied **then**
    exit the loop
  end if
end for

Output: $\boldsymbol{u}_k^t = \boldsymbol{u}_k^{(l)}$.

optimization problem admits a unique minimizer. The detailed AGD algorithm for (20a) at node k is given by Algorithm 1.

A. Distributed Computation of Z_k in (20b)

We could use a dual-decomposition method for the separated objective function (20b). However, using the augmented Lagrangian can bring robustness to the dual ascent method [11], so we use this method below. Let Ω_{kj1} (Ω_{kj2}) denotes the Lagrange multiplier corresponding to the constraint $Z_k = W_{kj}$ (respectively $W_{kj} = Z_j$), and we construct the augmented Lagrangian function for the problem (20b) as follows:

$$\begin{aligned}\mathcal{L}_\rho(\{Z_k\},\{W_{kj}\},\{\Omega_{kjl}\}) = \sum_{k=1}^{N}\Big(&\|Z_k - \boldsymbol{u}_k^t \boldsymbol{u}_k^{t,T}\|_F^2 \\ &+ \rho\sum_{j\in\mathcal{N}_k}\|Z_k - W_{kj} + \Omega_{kj1}/\rho\|_F^2 \\ &+ \rho\sum_{j\in\mathcal{N}_k}\|W_{kj} - Z_j + \Omega_{kj2}/\rho\|_F^2\Big)\end{aligned}$$

where $\rho > 0$ is a penalty parameter. The ADMM [11] is then applied to solving the problem (20b) in a cyclic fashion by minimizing $\mathcal{L}_\rho$ with respect to the local variables $\{Z_k\}$ and auxiliary variables $\{W_{kj}\}$, followed by a gradient ascent step over the dual variables $\{\Omega_{kj1}, \Omega_{kj2}\}$. Following similar procedure as [29], we obtain the iterations required per node k for solving (20b)

$$\{Z_k^{(l)}\} = \arg\min_{\{Z_k\}} \mathcal{L}_\rho(\{Z_k\},\{W_{kj}^{(l-1)}\},\{\Omega_{kjl}^{(l-1)}\}) \quad (21a)$$

$$W_{kj}^{(l)} = \frac{1}{2\rho}(\Omega_{kj1}^{(l-1)} - \Omega_{kj2}^{(l-1)}) + \frac{1}{2}(Z_k^{(l-1)} + Z_j^{(l-1)}) \quad (21b)$$

$$\Omega_{kj1}^{(l)} = \Omega_{kj1}^{(l-1)} + \rho(Z_k^{(l)} - W_{kj}^{(l)}) \quad (21c)$$

$$\Omega_{kj2}^{(l)} = \Omega_{kj2}^{(l-1)} + \rho(W_{kj}^{(l)} - Z_j^{(l)}) \quad \forall k \in \mathcal{V}, j \in \mathcal{N}_k. \quad (21d)$$

The convergence of the ADMM method ensures the convergence of the distributed iterations (21a)–(21d) to the optimal value of (20b). Moreover, iterations (21a)–(21d) can be further simplified, as stated in the following lemma.

Lemma 2: Initializing all the Lagrange multipliers to zeros at every node, iterations (21a)–(21d) reduce to

$$Z_k^{(l)} = \frac{\boldsymbol{u}_k^t \boldsymbol{u}_k^{t,T} - 2\Omega_k^{(l-1)} + \rho\sum_{j\in\mathcal{N}_k}(Z_k^{(l-1)} + Z_j^{(l-1)})}{1 + 2\rho|\mathcal{N}_k|} \quad (22a)$$

$$\Omega_k^{(l)} = \Omega_k^{(l-1)} + \rho/2\sum_{j\in\mathcal{N}_k}(Z_k^{(l)} - Z_j^{(l)}) \quad \forall k \in \mathcal{V} \quad (22b)$$

where $|\mathcal{N}_k|$ is the number of neighboring nodes of node k, and $\Omega_k^{(l)} := \sum_{j\in\mathcal{N}_k}\Omega_{kj1}^{(l)}, \forall k \in \mathcal{V}$ are the scaled local aggregate Lagrange multipliers.

Proof: All the Lagrange multipliers are initialized to zeros, i.e., $\Omega_{kj1}^{(0)} = \Omega_{kj2}^{(0)} = \mathbf{0}_{p\times p}$. Substituting (21b) into (21c) and (21d), we have

$$\Omega_{kj1}^{(l)} = \Omega_{kj2}^{(l)} = \frac{1}{2}(\Omega_{kj1}^{(l-1)} + \Omega_{kj2}^{(l-1)}) + \frac{\rho}{2}(Z_k^{(l)} - Z_j^{(l)}). \quad (23)$$

By mathematical induction, we know that $\Omega_{kj1}^{(l)} = -\Omega_{jk1}^{(l)}, \forall k \in \mathcal{V}, j \in \mathcal{N}_k$. Therefore, the auxiliary variables $W_{kj}^{(l)}$ can be expressed as

$$W_{kj}^{(l)} = (Z_k^{(l)} + Z_j^{(l)})/2. \quad (24)$$

By substituting (24) into (23), we obtain the iteration (22b) for the scaled local aggregate Lagrange multiplier Ω_k. Taking the derivative of the Lagrangian $\mathcal{L}_\rho$ with respect to Z_k, setting the derivative to zero, and substituting (24) into the resulting equation, we obtain the closed-form solution (22a) for the optimization problem (21a). ■

Note that the proposed distributed algorithm has a double-loop structure, including the outer BCD iteration (20) and the inner ADMM iteration (22). At every inner loop l, the computation of (22) at each node k requires quantities $\{Z_j\}_{j\in\mathcal{N}_k}$ of its neighboring nodes and hence the communication among neighbors is needed. However, at every outer loop t, if the inner loop iterations run multiple times till convergence, it may lead to excessive communication overhead.

An inexact approach: To save energy resources and reduce communication costs, alternatively, we introduce an inexact approach for computing (20b). First, the number of the inner iteration is restricted to be a fixed small number $L > 0$. Second, the initial states of variables at current inner iteration are set equal to the last states at the previous inner iteration, i.e.,

$$Z_k^{t,(0)} = Z_k^{t-1,(L)}, \ \Omega_k^{t,(0)} = \Omega_k^{t-1,(L)} \quad \forall k \in \mathcal{V}. \quad (25)$$

Third, the iterative result $Z_k^{t,(L)}$ after L iterations is projected onto a positive semidefinite cone for feasibility. For clarity, the proposed inexact approach for solving (20b) is summarized in Algorithm 2.

Remark 1: We can treat this approach as an online setting of the ADMM, in which $\{Z_k\}$ are reevaluated after new "data points" $\{\boldsymbol{u}_k^t\}$ are revealed in every BCD round. We will show that the inexact intermediate quantities $\{Z_k^t\}$ obtained after a small number of iterations can also result in good enough estimates of $\{M_k^t\}$ and $\{\boldsymbol{u}_k^{t+1}\}$, and hence it saves communication resources greatly compared with that using exact minimization.

Algorithm 2: Solving (20b) via the Inexact ADMM Method.

Input: Node k has computed the local parameter $\boldsymbol{u}_k^t$. Setting $Z_k^{t,(0)} = Z_k^{t-1,(L)}$ and $\Omega_k^{t,(0)} = \Omega_k^{t-1,(L)}, \forall k \in \mathcal{V}$.

1: Set the penalty parameter ρ appropriately.
2: **for** $l \leftarrow 1, 2, \ldots, L$ **do**
3: **for all** $k = 1, \ldots, N$ **do**
4: Compute $Z_k^{t,(l)}$ via (22a).
5: Broadcast $Z_k^{t,(l)}$ to all neighbors in $\mathcal{N}_k$.
6: **end for**
7: **for all** $k = 1, \ldots, N$ **do**
8: Compute $\boldsymbol{\Omega}_k^{t,(l)}$ via (22b).
9: **end for**
10: **end for**

Output: $Z_k^t = \text{Proj}_{\mathbb{S}_+^p}(Z_k^{t,(L)}), \Omega_k^t = \Omega_k^{t,(L)}, \forall k \in \mathcal{V}$.

As data points $\{\boldsymbol{u}_k^t\}$ getting stabilized with the BCD procedure, the inexact quantities $\{Z_k^t\}$ among all nodes will gradually approach the exact one and the optimal minimizer can be obtained asymptotically.

B. Local Computation of M_k in (20c)

Since $\epsilon \mathbf{I}_p + Z_k^t$ and $(\eta \mathbf{I}_p + M_k)^{-1}$ are positive definite, the objective function in (20c) is strictly convex, and hence it admits a unique minimizer, which can be obtained via solving an eigenvalue optimization problem summarized in Theorem 2 (subscript k is omit for notational simplicity).

Theorem 2: Given an arbitrary symmetric matrix $Z \in \mathbb{R}^{p \times p}$ in (20c), let $Z = P \Sigma P^T$ be its eigendecomposition, where $P \in \mathbb{R}^{p \times p}$ is orthogonal, and $\Sigma = \text{diag}(\gamma_1, \ldots, \gamma_p) \in \mathbb{R}^{p \times p}$ is diagonal with the eigenvalues on its main diagonal. Let $\Sigma = \text{diag}(\sigma_1^*, \ldots, \sigma_p^*) \in \mathbb{R}^{p \times p}$, where $\{\sigma_i^*\}_{i=1}^p$ is the optimal solution to the following convex optimization problem:

$$\begin{aligned} \min_{\{\sigma_i\}} \quad & \sum_{i=1}^{p} \frac{\gamma_i + \epsilon}{\eta + \sigma_i} \\ \text{s.t.} \quad & \sum_{i=1}^{p} \sigma_i = h, 0 \le \sigma_i \le 1. \end{aligned} \tag{26}$$

Then, the global minimizer to (20c) is given by $M^* = P \Sigma P^T$, and the objective value in (26) is equal to that in (20c).

The proof of Theorem 2 is omitted since it is an existing result in [21]. The problem (26) can be solved using a linear time algorithm [21].

For clarity, the main steps for distributed learning of predictive structures from multiple tasks over sensor networks are presented in Algorithm 3. Due to space limitations, we do not provide the theoretical analysis.

C. Communication Complexity Analysis

We provide an analysis of the communication complexity of the distributed algorithm. In each outer iteration, each node transmits local estimates to its neighbors L times. Let T denote the total number of the outer iteration, then node k needs to transmit $O(TL|\mathcal{N}_k|)$ messages in total, where $|\mathcal{N}_k|$ is the number of its neighbors. Note that the less the inner iteration is, the more the outer iteration is needed to get convergence. In other words, the small value of L will lead to the large value of T. From Algorithm 3, we see that a large number of the outer iteration will lead to a large computational cost. In the following simulations, we experimentally study the communication cost with different number of the inner iteration L.

Algorithm 3: The Distributed Multitask Learning (dMTL).

Input: Each node k has a small number (N_k) of the labelled data $(\boldsymbol{x}_k, \boldsymbol{y}_k)$.

1: **for** $t \leftarrow 1, 2, \ldots$ **do**
2: **for all** $k = 1, \ldots, N$ **do**
3: Compute $\boldsymbol{u}_k^t$ in (20a) via Algorithm 1.
4: **end for**
5: Distributedly compute $\{Z_k^t\}$ in (20b) via Algorithm2.
6: **for all** $k = 1, \ldots, N$ **do**
7: Compute M_k^t in (20c) using Theorem 2.
8: **end for**
9: **end for**

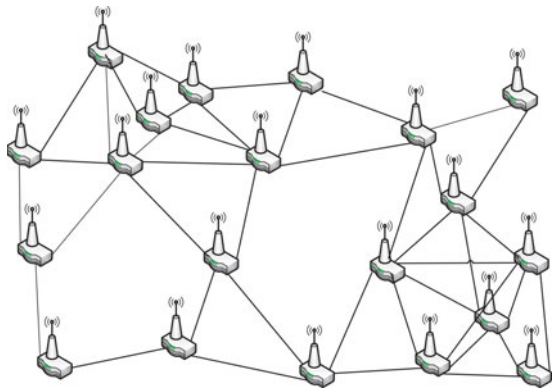


Fig. 1. Network connection.

V. Simulations

In this section, the performance of the proposed dMTL algorithm is evaluated via numerical simulations on both synthetic and real-world datasets.

We first examine kinds of properties of the dMTL on the synthetic data. We consider the case of regression and a randomly generated sensor network with 20 nodes. The nodes are randomly placed in a 2.5×2.5 square, and the communication distance is taken as 0.8, as shown in Fig. 1. The parameter $\boldsymbol{u}_k$ of the node (task) k is selected from an r-dimensional zero-mean Gaussian distribution $\mathcal{N}(\mathbf{0}, \Sigma)$, where the covariance matrix Σ is drawn from an $r \times r$ dimensional Wishart distribution $\mathcal{W}(r, r\mathbf{I})$ with r degrees of freedom. To these r-dimensional $\boldsymbol{u}_k$'s, we add $p - r$ irrelevant dimensions which are exactly zero, i.e., $\boldsymbol{u}_k = [\boldsymbol{u}_k^T, \mathbf{0}_{p-r}^T]^T$. By construction, the multiple tasks among all nodes share an r-dimensional predictive subspace. Note that we regenerate the covariance

matrix Σ and the corresponding parameters $\{\boldsymbol{u}_k\}$ for each independent simulation. The outputs $\boldsymbol{y}_{k,i}$ are computed from a noisy linear function, $\boldsymbol{y}_{k,i} = \boldsymbol{u}_k^T \boldsymbol{x}_{k,i} + \nu, i = 1, 2, \ldots, N_k$, where ν is zero-mean Gaussian noise with standard deviation equal to 0.05. The input data $\{\boldsymbol{x}_{k,i}\}$ is randomly generated uniformly from $[0, 1]^p$. For simplicity, we assume that the numbers of training samples among all nodes are equal. Unless otherwise specified, the relevant dimension is set as $r = 5$, and the feature dimension is set as $p = 20$ in the following simulations. Moreover, the mean-square-error loss function is used in this regression problem.

For comparison, we also simulate the centralized multitask learning (cMTL) algorithm, which alternately performs (16a) and (16b), the corresponding noncooperative (non-coop) learning algorithm, which estimates each of tasks at each node independently using standard ridge regression, and the diffusion LMS for multitask networks (MT-LMS) proposed in [15], which promotes the similarity of parameter vectors via their distance among neighboring nodes. Note that the ridge regression is equivalent to (5) with $\alpha = 0$, and its tradeoff parameter β is determined via the cross validation.

All results below are averaged over 100 independent Monte Carlo simulations with randomly generated samples. We employ the mean-square deviation (MSD) of estimated parameter errors as the performance measure.

A. Convergence Study

We first check the convergence of the dMTL with the inexact approach, in which the number of inner iterations L is fixed at 3, 6, and 9, respectively. For the dMTL and the cMTL, we heuristically set the dimension of subspace h as 5 and then determined the parameters $\alpha = 10^{-2}$ and $\eta = 10^{-2}$. We set the step size of the MT-LMS as 0.2 and set its regularization parameter as 10^{-3}. We evaluated the MSD performance of the dMTL with different number of training samples per node, and two of them ($N_k = 20, 40$) are shown in Fig. 2, compared with the cMTL, the noncooperative, and the MT-LMS algorithms. The MSD learning curves are obtained by averaging all local MSDs among nodes. Since the MT-LMS is an online algorithm, whose iteration procedure is not the same as the BCD of the dMTL, we draw a line to represent its steady-state MSD obtained after $10\,000$ iterations. Since the noncooperative algorithm has no iteration procedure, we also draw a line to represent its steady-state MSD. We observed that in all cases, the MSD of the dMTL can converge to that of the cMTL even with a very few times of inner iterations. We also observe that the dMTL gets converged faster as the number of inner iterations L increases or the number of the training samples per node increases.

Furthermore, by learning multiple tasks simultaneously through cooperating with neighboring nodes in a multitask network, the dMTL improves the estimation accuracy of each task compared with the noncooperative algorithm, and this improvement is quite big when the number of the training samples per node is small. To further verify this point, we evaluated the steady-state MSD performance of the dMTL with different training sample sizes. As shown in Fig. 3, the steady-state MSD

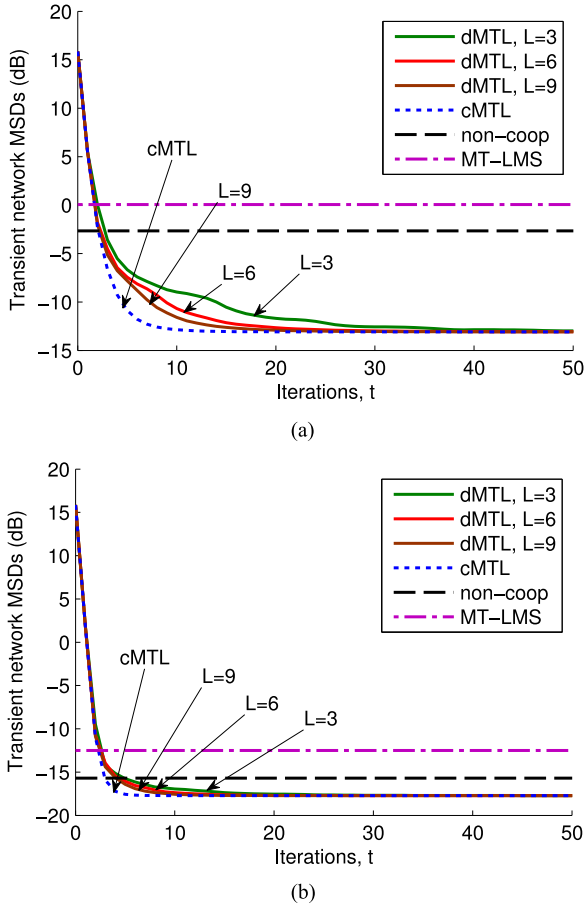


Fig. 2. Transient network MSDs of the dMTL with different number of inner iterations ($L = 3, 6, 9$), compared with the cMTL, the non-coop, and the MT-LMS algorithms. (a) $N_k = 20$, (b) $N_k = 40$.

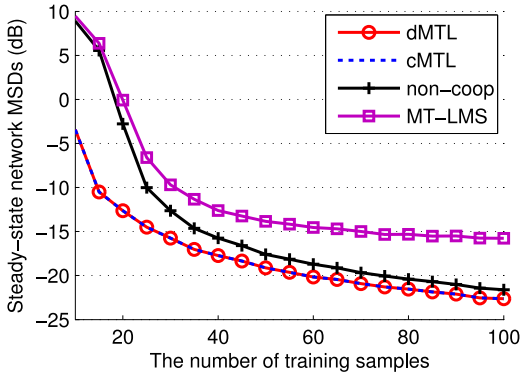


Fig. 3. Steady-state network MSDs versus the number of training samples, compared with the cMTL, the non-coop, and the MT-LMS algorithms.

of the dMTL gradually decreases as the training sample size increases, and the dMTL can always achieve better performance than the noncooperative and the MT-LMS algorithms. Especially when the training sample size is small, the dMTL makes use of the relationship among multiple tasks and achieves much better performance. We fixed the sample size as $N_k = 20$ in the following simulations. From the above results, we see that the

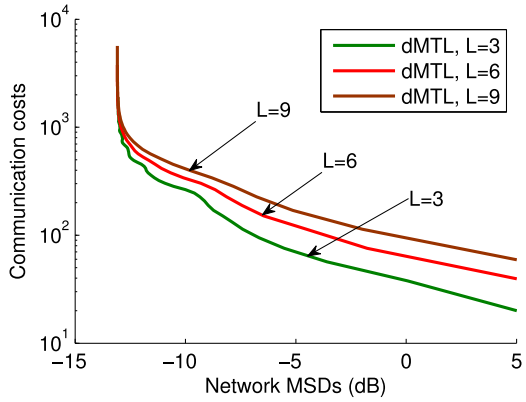


Fig. 4. Communication costs versus the network MSDs with different number of inner iteration L.

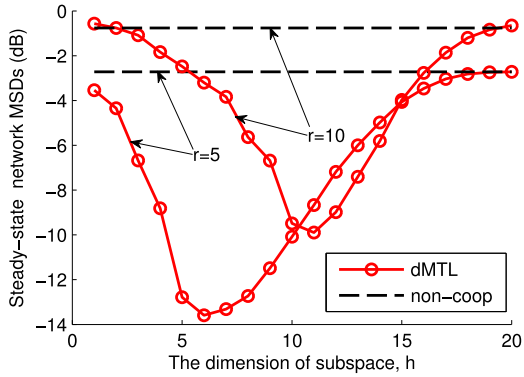


Fig. 5. Steady-state MSDs versus the dimension of subspace h, as the number of relevant variables r simultaneously changes.

MT-LMS algorithm performs even worse than the noncooperative algorithm. Note that the MT-LMS is an online algorithm so it has extra misadjustment error, and it is designed for estimation problem with parameter similarity characterized by clusters and distances, so it is not suitable for this experiment. Moreover, since the cMTL has the same steady-state performance as the dMTL, we omit both the MT-LMS and cMTL in the following steady-state simulations.

B. Communication Cost of the dMTL

We have analyzed the communication complexity of the dMTL in the previous section. In this section, we experimentally study the communication cost of the dMTL with different number of inner iteration L. We use the same parameter settings as previous experiment ($N_k = 20$). At node k, the communication cost after t outer iteration is $tL|\mathcal{N}_k|$. We compute the network communication cost by averaging all costs among all nodes over networks. As shown in Fig. 4, to achieve the same network MSD, the dMTL with the small value of L costs less communication resources than those with the large values of L. Thus, in practical applications, we can choose an appropriate value of L to balance the communication cost and computation time. We fixed $L = 6$ in the following simulations.

C. Subspace Selection

In this section, we study the effect of the dimension of the shared feature subspace h on the estimation performance of the dMTL. We fixed the dimension of feature space $p = 20$, and test two cases that the value of the relevant dimension r is taken as 5 and 10, respectively. We set the parameters $\alpha = 10^{-2}$, and $\eta = 10^{-2}$. For each case, we record the obtained steady-state MSDs of estimated parameter errors with different dimensions of the subspace ($h < p$). The result is presented in Fig. 5. In both cases, we observe that when h gets close to the relevant dimension $r = 5$ (or $r = 10$), the dMTL has a relatively lower MSD. It is reasonable since the relevant dimension r is exactly the dimension of the subspace by construction. In practice, it is good enough to choose $h = [5, 8]$ when $r = 5$, and choose $h = [10, 12]$ when $r = 10$. We also observe that when h gets close to the feature dimension $p = 20$, the dMTL has the similar performance with the noncooperative algorithm. The reason is that when $h = p$, the common subspace becomes the original predictive space, and no extra information can be extracted from the multitask learning framework. We set $r = 5$ and $h = 5$ in the following simulations.

D. Sensitivity Study

In this section, we study the effect of the parameters α and η on the estimation performance of the dMTL. Recall that $\eta = \beta/\alpha$, where the regularization parameters α and β are used to tradeoff the importance of two regularization components in (6). We first set $\eta = 10^{-3}, 10^{-2}, 10^{-1}$, respectively; meanwhile, we vary α in the range $[10^{-6}, 10^{0}]$. As shown in Fig. 6 (top), the dMTL can achieve relatively better performance if α is set to some value close to 10^{-2} no matter what value of η is. Too small or too large value of α will result in relatively bad performance. Nevertheless, we can choose an appropriate value of α via the simple cross validation.

In the second experiment of this part, we set $\alpha = 10^{-3}, 10^{-2}, 10^{-1}$, respectively; meanwhile, we vary η in the range $[10^{-8}, 10^{2}]$. As shown in Fig. 6 (bottom), the dMTL is not very sensitive to the parameter η. The dMTL can achieve better performance when η is taken a relatively small value (over a very wide range $[10^{-8}, 10^{-2}]$) no matter what value of α is. We observe similar results on other simulations. Note that η is the ratio of β to α, a small value of η is corresponding to a small proportion of the second component of the regularization function (6). In the case that the number of training samples is limited ($N_k = 20$ in this experiment), the multitask framework works and the first component of the regularization (6) plays a major role. Empirically, we set $\eta = 10^{-2}$ in the following simulation.

E. Impact of the Number of Tasks (Nodes)

To test the effect of the number of jointly learned tasks, we used the dMTL with $N = 10, 20, 100$ nodes (tasks). The density of networks remains unchanged. To ensure this, the communication distance still remains 0.8 and the square, in which the nodes are randomly placed, is proportionally zoomed in and out. In this experiment, we set $r = 5$, $h = 5$, $\alpha = 10^{-2}$, and $\eta = 10^{-2}$; meanwhile, we vary the feature dimension p in

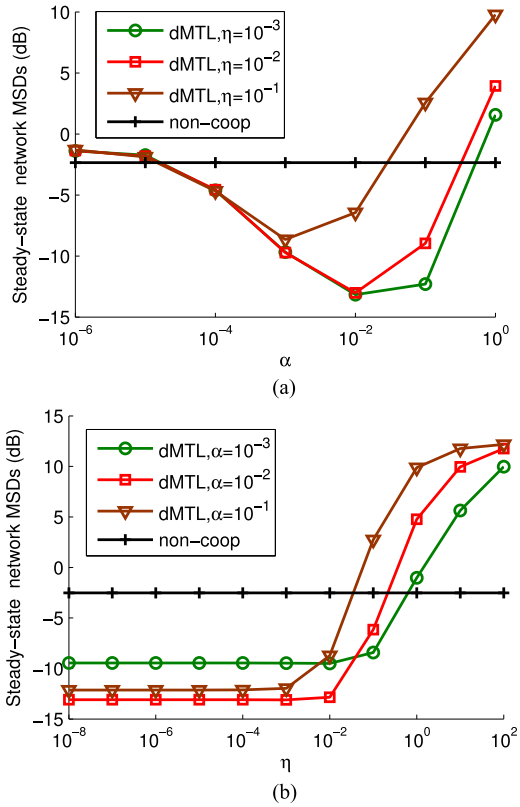


Fig. 6. Sensitivity study. (a) Steady-state MSDs versus the parameter α in the dMTL; (b) steady-state MSDs versus the parameter η in the dMTL.

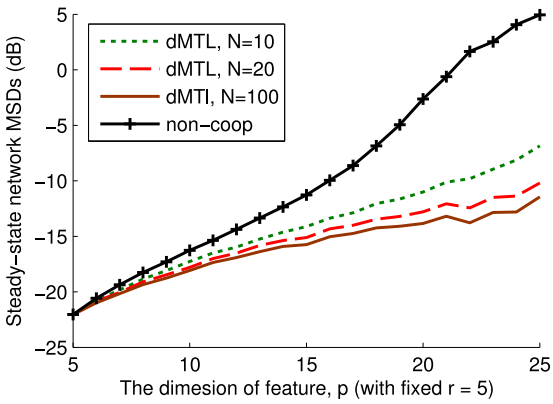


Fig. 7. Steady-state MSDs versus the dimension of feature space p with the fixed $r = 5$, as the number of tasks N simultaneously changes.

the range $[5, 25]$. For $N = 10$ and 20, the performance metrics of the dMTL are averaged over randomly selected subsets of the 100 tasks, so that our estimates have comparable variance. Fig. 7 shows experimental results. With the increase of the number of tasks, the performance of the dMTL improves. The reason is that by learning more tasks simultaneously, more structure information of the parameter space can be extracted from multiple tasks and the common subspace can be estimated more accurately. Furthermore, as the number of irrelevant dimension ($p - r = [0, 20]$) increases, the performance of the dMTL gets a bit worse but its advantage over the noncooperative algorithm

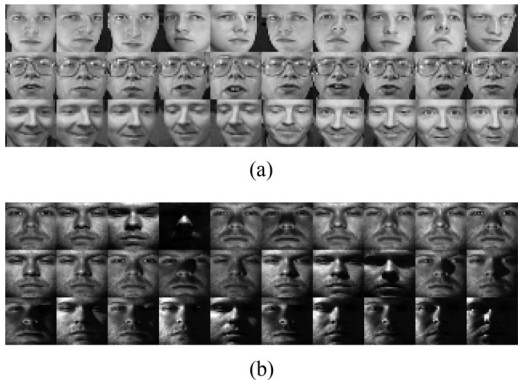


Fig. 8. Sample images from (a) ORL face database and (b) Extended Yale-B face database.

becomes more and more obvious. To explain this, note that with the increasing of the number of irrelevant variables, the noncooperative algorithm needs more training samples per node to obtain an acceptable estimate. Meanwhile, the dMTL makes use of the structure information from multiple tasks and can cover the shortage of training samples in some extent.

F. Classification of Real Data

In this section, we examine the dMTL on the two real-world datasets. In many practical image processing applications in a sensor network, the sensor node equipped with tiny cameras often has a small number of image samples. Thus, the image classification task executed at a single node may has a poor performance. In this case, a multitask network can be used to improve the performance of the classifier at each node. To simulate the above scenario, we use the proposed dMTL algorithm for cooperatively performing the image classification tasks in a multitask network using two real-world datasets: Olivetti Research Laboratory (ORL) database[1] and Extended Yale Face-B database.[2] The ORL database of Faces contains ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions, and facial details. The extended Yale-B face database contains 16 128 images of 38 human subjects under 9 poses and 64 illumination conditions. We choose the frontal pose and use all the images under different illuminations, thus we get 64 images per individual. Some sample images are shown in Fig. 8. All images were manually aligned, cropped, and then resized to 32×32 pixels, with 256 gray levels per pixel. Each image is represented by a 1024-dimensional vector in the image space. We employ the misclassification rate (test error rate) as the performance measure. Moreover, the modified Huber loss function is used in this classification problem.

We examine the dMTL with different number of nodes (tasks) at $N = 10, 20, 40, 60, 80, 100$, respectively. The density of networks is kept the same as the previous experiment. We set $L = 6$, $h = 4$, $\eta = 10^{-3}$, and $\alpha = 10^{-3}$ for the ORL database ($\alpha = 10^{-2}$ for the Extended Yale-B database). For each simulation, each node performs a two-class image classification task,

[1] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
[2] http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html

TABLE I
PERFORMANCE COMPARISONS ON TWO REAL-WORLD DATASETS WITH DIFFERENCE NUMBER OF NODES (TASKS)

	Error Rates (mean $\pm$ std-dev%)			
	ORL face database		Extended Yale-B face	
N	non-coop	dMTL	non-coop	dMTL
10	15.13 ± 5.86	$\mathbf{14.39 \pm 5.81}$	8.10 ± 1.82	$\mathbf{6.37 \pm 1.82}$
20	15.27 ± 4.69	$\mathbf{14.55 \pm 4.67}$	7.89 ± 1.28	$\mathbf{6.41 \pm 1.16}$
40	15.04 ± 3.12	$\mathbf{14.30 \pm 3.08}$	7.96 ± 0.87	$\mathbf{6.30 \pm 0.86}$
60	15.02 ± 2.41	$\mathbf{14.25 \pm 2.37}$	8.02 ± 0.71	$\mathbf{6.47 \pm 0.61}$
80	15.15 ± 2.24	$\mathbf{14.41 \pm 2.23}$	8.04 ± 0.71	$\mathbf{6.39 \pm 0.58}$
100	15.31 ± 2.12	$\mathbf{14.59 \pm 2.11}$	8.01 ± 0.64	$\mathbf{6.39 \pm 0.47}$

in which two subjects are randomly selected from 40 (or 38) distinct subjects. We set the training and test ratio at 2:8 and record the averaged test error rate of the dMTL over all nodes (tasks). The final test error rate is averaged over 100 independent simulations. We compare the dMTL with the corresponding noncooperative (non-coop) learning algorithm, which performs each of tasks at each node independently using a linear classifier with modified Huber loss function and l_2-norm regularization. As shown in Table I, no matter how many tasks (nodes) are performed, the dMTL always has a relatively lower test error rate than the corresponding noncooperative algorithm on both of the two face databases. Especially on the Extended Yale-B database, the improvement is remarkable. Note that the classified subjects among all nodes are different, and, hence, the decision hyperplanes of the tasks are also different from each other. Thus, the overlap of hypothesis subspaces among all nodes may be quite small. Nevertheless, our proposed dMTL can still make use of the structure information to cover the shortage of the training samples and improves the generalization performance of the linear classifier at each node.

VI. CONCLUSION

Consider the case that each node in a network has different estimation task, and assuming all tasks share a common feature subspace, we formulated the dMTL framework. Based on the BCD method and the inexact ADMM technique, we derived a dMTL algorithm, which is computationally inexpensive and energy saving owing to the use of the inexact approach. Numerical simulations demonstrated that the proposed dMTL algorithm can converge to its centralized counterpart and outperforms the corresponding noncooperative learning algorithm.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[2] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *Proc. 46th IEEE Conf. Decision Control*, New Orleans, LA, USA, Dec. 2007, pp. 5492–5498.

[3] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3122–3136, Jul. 2008.

[4] I. D. Schizas, G. Mateos, and G. B. Giannakis, "Distributed LMS for consensus-based in-network adaptive processing," *IEEE Trans. Signal Process.*, vol. 57, no. 6, pp. 2365–2382, Jun. 2009.

[5] C. Li, P. Shen, Y. Liu, and Z. Zhang, "Diffusion information theoretic learning for distributed estimation over network," *IEEE Trans. Signal Process.*, vol. 61, no. 16, pp. 4011–4024, Aug. 2013.

[6] Y. Liu, C. Li, and Z. Zhang, "Diffusion sparse least-mean squares over networks," *IEEE Trans. Signal Process.*, vol. 60, no. 8, pp. 4480–4485, Aug. 2012.

[7] X. Cao, J. Chen, Y. Xiao, and Y. Sun, "Building-environment control with wireless sensor and actuator networks: Centralized versus distributed," *IEEE Trans. Ind. Electron.*, vol. 57, no. 11, pp. 3596–3605, Nov. 2010.

[8] J. Chen, X. Cao, P. Cheng, Y. Xiao, and Y. Sun, "Distributed collaborative control for industrial automation with wireless sensor and actuator networks," *IEEE Trans. Ind. Electron.*, vol. 57, no. 12, pp. 4219–4230, Dec. 2010.

[9] J. C. Chen, K. Yao, and R. E. Hudson, "Source localization and beamforming," *IEEE Signal Process. Mag.*, vol. 19, no. 2, pp. 30–39, Mar. 2002.

[10] R. Olfati-Saber, A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jul. 2011.

[12] A. G. Dimakis, S. Kar, J. M. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proc. IEEE*, vol. 98, no. 11, pp. 1847–1864, Nov. 2010.

[13] Z. Liu, Y. Liu, and C. Li, "Distributed sparse recursive least-squares over networks," *IEEE Trans. Signal Process.*, vol. 62, no. 6, pp. 1386–1395, Mar. 2014.

[14] P. Shen and C. Li, "Distributed information theoretic clustering," *IEEE Trans. Signal Process.*, vol. 62, no. 13, pp. 3442–3453, Jul. 2014.

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

[16] J. Chen, C. Richard, and A. H. Sayed, "Diffusion LMS over multitask networks," *IEEE Trans. Signal Process.*, vol. 63, no. 11, pp. 2733–2748, Jun. 2015.

[17] J. Chen, C. Richard, A. O. Hero, and A. H. Sayed, "Diffusion LMS for multitask problems with overlapping hypothesis subspaces," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process.*, Reims, France, Sep. 2014, pp. 1–6.

[18] R. K. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," *J. Mach. Learn. Res.*, vol. 6, pp. 1817–1853, Nov. 2005.

[19] A. Argyriou, T. Evgeniou, and M. Pontil, "Convex multi-task feature learning," *Mach. Learn.*, vol. 73, no. 3, pp. 243–272, Dec. 2008.

[20] G. Obozinski, B. Taskar, and M. I. Jordan, "Joint covariate selection and joint subspace selection for multiple classification problems," *J. Statist. Comput.*, vol. 20, no. 2, pp. 231–252, Apr. 2010.

[21] J. Chen, L. Tang, J. Liu, and J. Ye, "A convex formulation for learning a shared predictive structure from multiple tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 5, pp. 1025–1038, May 2013.

[22] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. Roy. Statist. Soc. B*, vol. 67, no. 2, pp. 301–320, Mar. 2005.

[23] M. L. Overton and R. S. Womersley, "Optimality conditions and duality theory for minimizing sums of the largest eigenvalues of symmetric matrices," *Math. Program.*, vol. 62, nos. 1–3, pp. 321–357, Feb. 1993.

[24] R. A. Horn and C. R. Johnson, *Matrix Analysis*. New York, NY, USA: Cambridge Univ. Press, Oct. 2012.

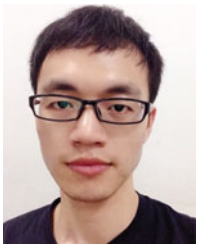
[25] A. Argyriou, M. Pontil, Y. Ying, and C. A. Micchelli, "A spectral regularization framework for multi-task structure learning," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2007, pp. 25–32.

[26] D. P. Bertsekas, *Nonlinear Programming*. Nashua, NH, USA: Athena Scientific, Apr. 2004.

[27] J. Hua and C. Li, "Distributed variational Bayesian algorithms over sensor networks," *IEEE Trans. Signal Process.*, vol. 64, no. 3, pp. 783–798, Feb. 2016.

[28] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87. New York, NY, USA: Springer Science & Business Media, 2004.

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



Junhao Hua received B.S. degrees in computer science and automation from Zhejiang University of Technology, Hangzhou, China, in 2013. He is currently working toward the Ph.D. degree in the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou.

His current research interests include statistical signal processing, Bayesian learning, and wireless sensor networks.

Chunguang Li (M'14–SM'14) received the M.S. degree in pattern recognition and intelligent systems and the Ph.D. degree in circuits and systems from the University of Electronic Science and Technology of China, Chengdu, China, in 2002 and 2004, respectively.

He is currently a Professor in the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China. His current research interests include statistical signal processing and wireless sensor networks.

Hui-Liang Shen received the B.Eng. and Ph.D. degrees in electronic engineering from Zhejiang University, Hangzhou, China, in 1996 and 2002, respectively.

He was a Research Associate and a Research Fellow with The Hong Kong Polytechnic University from 2001 to 2005. He is currently a Full Professor in the College of Information Science and Electronic Engineering, Zhejiang University. His research interests include color imaging, image processing, and three-dimensional computer vision.