Distributed Variational Bayesian Algorithms Over Sensor Networks

Junhao Hua and Chunguang Li, *Senior Member, IEEE*

***Abstract*—Distributed inference/estimation in Bayesian framework in the context of sensor networks has recently received much attention due to its broad applicability. The variational Bayesian (VB) algorithm is a technique for approximating intractable integrals arising in Bayesian inference. In this paper, we propose two novel distributed VB algorithms for general Bayesian inference problem, which can be applied to a very general class of conjugate-exponential models. In the first approach, the global natural parameters at each node are optimized using a stochastic natural gradient that utilizes the Riemannian geometry of the approximation space, followed by an information diffusion step for cooperation with the neighbors. In the second method, a constrained optimization formulation for distributed estimation is established in natural parameter space and solved by alternating direction method of multipliers (ADMM). An application of the distributed inference/estimation of a Bayesian Gaussian mixture model is then presented, to evaluate the effectiveness of the proposed algorithms. Simulations on both synthetic and real datasets demonstrate that the proposed algorithms have excellent performance, which are almost as good as the corresponding centralized VB algorithm relying on all data available in a fusion center.**

***Index Terms*—Alternating direction method of multipliers, distributed algorithm, stochastic natural gradient, variational Bayes, wireless sensor network (WSN).**

I. Introduction

WIRELESS sensor networks consist of an amount of spatially distributed nodes/agents that have limited communication capabilities due to energy and bandwidth constraints. Such networks are well-suited to perform decentralized information processing and inference tasks [1]–[5]. Distributed approach performs inference/estimation tasks locally at each node using its local data and the information obtained from its one-hop neighbors. Compared with the centralized approach, it does not need a powerful fusion center. So it is more flexible and provides robustness to node and/or link failures in a network, in addition to saving communication resources and energy. In view of this, many distributed inference/estimation algorithms over networks have been proposed, such as consensus-based [6], [7], diffusion-based [8]–[11] and randomized gossip-based algorithms [12], [13]. With the development of modern wireless sensor networks and the extension of their application areas [14], [15], the observations are getting larger and more complex. Therefore, it is urgent to develop advanced in-network distributed algorithms for data analysis.

The Bayesian modeling provides us with an elegant approach to analyze massive data and its hidden structure. Several studies use probabilistic graphical models for distributed Bayesian inference. In [16], a distributed architecture using message passing (or belief propagation) on a junction tree was presented, which is constructed by the minimum spanning tree algorithm. But exploring the junction tree itself is expensive in low-cost networked systems. Combined with the belief propagation (BP), an in-network variational message passing framework was proposed for Markov random fields in [17]. However, the convergence can not be guaranteed in loopy graphs and it is intractable for complex models with non-Gaussian continuous variables. For tractability, the nonparametric BP was developed [18], [19], which combines the ideas from Monte Carlo and particle filtering for modeling uncertainty. However, the sampling-based technique is not suitable for large and/or distributed datasets due to the heavy computational cost.

The statistical inference tasks in the Bayesian framework often suffer from the computational intractability of posterior beliefs [20], [21]. To deal with it, one of the most successful methods in practice is the variational Bayesian (VB) approximation [22], [23]. In recent years, several scalable VB algorithms for massive and streaming data that arises in large scale applications were developed [24]–[26]. However, none of these algorithms are suitable for the networked systems. For example, the authors in [26] improve the scalability of variational inference for latent Dirichlet allocation in the MapReduce framework, but it needs a reducer (fusion center) and multiple mappers (nodes). The goal of the present paper is to design fully distributed variational Bayesian algorithms that can perform almost as well as the centralized VB.

Several previous studies have developed distributed VB algorithms for specific problems, especially for distributed density estimation using the Gaussian mixture model. In [27], an approach, which uses a cyclic path to incrementally collect all local quantities calculated at every node for the global estimation, was proposed. This approach needs a prior knowledge of the network topology and is not robust to node and/or link failures. In [28], [29], the distributed averaging strategies [30] were

Manuscript received September 29, 2014; revised May 19, 2015 and September 13, 2015; accepted October 16, 2015. Date of publication October 26, 2015; date of current version January 11, 2016. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Min Dong. This work was supported by the National Natural Science Foundation of China (Grant Nos. 61171153, 61571392, and 61471320) and the National Program for Special Support of Eminent Professionals. *(Corresponding author: Chunguang Li.)*

The authors are with the Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: cgli@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/TSP.2015.2493979

adopted to make a consensus among all nodes. In every VB step, each node exchanges information and repeats the averaging iteration many times, which exhausts the communication resources and energy. What's more, these algorithms are not extensible because they need to design local/global quantities carefully according to the specific model.

In this paper, we develop two robust variational algorithms for general Bayesian inference in a networked system. The first is based on stochastic optimization and distributed averaging, called as the distributed stochastic variational Bayesian algorithm (dSVB). This approach is motivated by the earlier development on the stochastic VB [25], which can only deal with the problem in the centralized scheme. We assume that the data model belongs to conjugate exponential families and the observed data is independent and identically distributed. Thus the global lower bound (objective function) is decomposed by a set of local lower bounds which are optimized in the natural parameter space. The stochastic gradient approximation is then adopted for local calculation, followed by a combination with its neighbors to diffuse information over the entire network. Note that the natural gradient [31] is adopted in our approach, since the parameter space of a distribution is Riemannian rather than Euclidean. The benefit of the stochastic gradient is that it can gradually learn information from data through the VB procedure with only one iteration in each VB step, which greatly reduces the communication cost while holding high accuracy. Furthermore, the quantity to be transmitted among neighbors is the natural parameter vector, which provides a general form of the message. Therefore, our algorithm is very general and can be automatically derived.

The second novel distributed variational Bayesian algorithm is based on the alternating direction method of multipliers (dVB-ADMM). As a simple but powerful optimization algorithm, the ADMM has been extended and developed for distributed convex optimization in recent years [32]–[34]. It is a very robust algorithm and few assumptions are needed for the convergence. To the best of our knowledge, the ADMM technique has not been applied to the VB algorithm for distributed inference/estimation. In this paper, the distributed VB algorithm is derived by solving a constrained optimization problem. The original variational objective function is equivalently transformed into a simple convex function with respect to the natural parameter vector. Importantly, the variational equality constraints of distributions, which are hard to be measured in the Riemannian space, are equivalently replaced by the equality constraints of their natural parameter vector using the Euclidean metric. A modified ADMM technique is then applied to solving this optimization problem in the natural parameter space. The message to be transmitted is also the natural parameter vector of a global distribution, which has much lower dimension and smaller size than the raw data.

In order to evaluate the effectiveness of the proposed algorithms, examples on distributed clustering and density estimation using Gaussian mixture model are presented. Numerical simulations on both synthetic and real-world datasets demonstrate that the proposed distributed approaches can perform almost as well as the corresponding centralized one and outperform the non-cooperation VB and non-stochastic-gradient based distributed VB algorithm. Furthermore, the dVB-ADMM converges faster than the dSVB.

The main contributions of this paper are summarized as follows.

- We propose a general distributed VB framework for conjugate-exponential models over a network.
- We integrate the stochastic natural gradient with the alternating iterative procedure over a network and propose the distributed stochastic VB algorithm.
- We establish a constrained optimization formulation for distributed inference in natural parameter space and develop a fast distributed VB algorithm based on ADMM.
- We solve the distributed inference/estimation of a Bayesian Gaussian mixture model in WSNs based on the dSVB and dVB-ADMM, respectively.

The rest of the paper is organized as follows. Section II states the problem and briefly reviews the traditional VB methods. Section III presents the general distributed Bayesian framework and then proposes the dSVB and the dVB-ADMM algorithms. An application on the distributed inference/estimation of a Bayesian Gaussian mixture model is then presented in Section IV. Section V provides detailed simulation results. Finally, conclusions are drawn in Section VI.

Notation

In this paper, we use boldface letters for matrices (column vectors). The superscript transposition $(\cdot)^T$ denotes transposition, and $[\cdot]_{ij}([\cdot]_i)$ denotes the ij-entry of a matrix (i-entry of a vector). The operator $\mathbb{E}[\cdot]$ denotes expectation, and $|\cdot|$ denotes the determinant of a matrix or absolute value in case of a scalar. Moreover, $\mathrm{Tr}(\cdot)$ denotes the trace operator, $\nabla(\cdot)$ stands for the vector differential operator, $\mathcal{N}(\cdot)$ is the Gaussian distribution, $\mathrm{Dir}(\cdot)$ is the Dirichlet distribution, $\mathcal{W}(\cdot)$ is the Wishart distribution, $\mathcal{NW}(\cdot)$ is normal-Wishart distribution, and $\mathrm{Mult}(\cdot)$ is the multinomial distributions. Finally, $\Gamma(\cdot)$ is the Gamma function, and $\varphi(\cdot)$ is the Digamma function. Other notations will be given if necessary.

II. Problem Statement and Preliminaries

We consider a sensor network consisting of N agents 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 $(i, j) \in \mathcal{E}$ connects an unordered pair of distinct nodes. For each node $i \in \mathcal{V}$, let $\mathcal{N}_i = \{j | (i, j) \in \mathcal{E}\}$ be a set of neighbors of node i (excluding node i itself).

Let us denote the observed dataset by $\boldsymbol{x}$. The data is collected by the N nodes of the network. Each node i has N_i D-dimension measurements $\boldsymbol{x}_i = \{\boldsymbol{x}_{ij}, j = 1, 2, \ldots, N_i\}$, and the full observed data is made up of measurements $\boldsymbol{x} = \{\boldsymbol{x}_i, i = 1, 2, \ldots, N\}$. Suppose the observed data is independent and identically distributed and produced by a generative model whose form is given. Generally, a generative model with unknown parameters often consists of a set of latent variables. In particular, we use $\boldsymbol{y}$ to denote the local latent variables and each node has a set of local latent variables $\boldsymbol{y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$. For notational convenience, we treat both

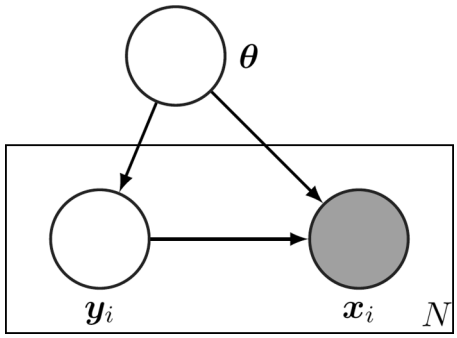

Fig. 1. A graphical model with observations $\{\boldsymbol{x}_i\}$, local latent variables $\{\boldsymbol{y}_i\}$ and global parameters $\boldsymbol{\theta}$. Circles represent random variables, Filled-in shapes indicate observed data and arrows describe conditional dependencies between variables. The N represents the repetition of the variables in the plate. (Though not pictured, each variable may be a collection of multiple random variables).

the unknown parameters and the global latent variables as the model parameters, denoted as $\boldsymbol{\theta}$. We group model parameters and latent variables as "unobserved variables", denoted as $\boldsymbol{z} = \{\boldsymbol{\theta}, \boldsymbol{y}\}$. We assume that the ith observation $\boldsymbol{x}_i$ and the ith local variable $\boldsymbol{y}_i$ are conditionally independent, given parameters $\boldsymbol{\theta}$, of all other observations and local latent variables. The graphical model in Fig. 1 captures the conditional dependencies among local latent variables, parameters and observed variables.

Given the form of the generative model, VB is to approximate the posterior of the unobserved variables, $P(\boldsymbol{z}|\boldsymbol{x})$, by a more tractable distribution, $Q(\boldsymbol{z})$. It is found by minimizing the Kullback-Leibler (KL) divergence between these two distributions [35],

$$\begin{aligned}\mathrm{KL}(Q(\boldsymbol{z})||P(\boldsymbol{z}|\boldsymbol{x})) &= \int Q(\boldsymbol{z}) \log \frac{Q(\boldsymbol{z})}{P(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} \\ &= -\mathbb{E}_Q[\log \frac{P(\boldsymbol{z},\boldsymbol{x})}{Q(\boldsymbol{z})}] + \log P(\boldsymbol{x}) \\ &= -\mathcal{L}(Q(\boldsymbol{z})) + \log P(\boldsymbol{x}), \end{aligned} \tag{1}$$

the lower bound $\mathcal{L}(Q)$ for the log evidence, $\log P(\boldsymbol{x})$, can be rewritten as a summation of an energy term and an entropy term (variational free energy),

$$\mathcal{L}(Q(\boldsymbol{z})) = \mathbb{E}_Q[\log P(\boldsymbol{z},\boldsymbol{x})] + \mathbb{H}[Q(\boldsymbol{z})]. \tag{2}$$

Minimizing the KL divergence is equivalent to maximizing the variational free energy since the log evidence is fixed with respect to the variational distribution Q. Thus, the inference task is presented as an optimization problem. In order to make it tractable, this problem is then "relaxed".

The first relaxation is to use the naive mean field theory [36], which limits the optimization to be optimized in a subset of distributions that are relatively easy to characterize. Specifically, it assumes the variational posterior of the unobserved variables can be factorized over some partitions $\boldsymbol{z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M\}$,

$$Q(\boldsymbol{z}) = \prod_{m=1}^{M} q_m(\boldsymbol{z}_m). \tag{3}$$

Each partition $\boldsymbol{z}_m$ has its own variational distribution $q_m(\boldsymbol{z}_m)$. In other words, each partition of the unobserved variables is mutually independent given the data. With the mean field assumption, the lower bound can be further decomposed into a suitable form, which is given in the following lemma.

Lemma 1: For each probability distribution, q_m, the variational free energy (2) can be written as

$$\mathcal{L}(q_1, \ldots, q_M) = -KL(q_m||q_m^*) + \mathbb{H}[q_{-m}] + \ln C. \tag{4}$$

In (4),

$$q_m^*(\boldsymbol{z}_m) \triangleq \frac{1}{C} \exp \mathbb{E}_{q_{-m}}[\ln P(\boldsymbol{z},\boldsymbol{x})], \tag{5}$$

where C is a normalizing constant, and $q_{-m}(\boldsymbol{z}_{-m})$ is the joint distribution of $\boldsymbol{z}_{-m}$ ($\boldsymbol{z}_{-m} \triangleq \boldsymbol{z} \setminus \boldsymbol{z}_m$ denotes all unobserved variables except $\boldsymbol{z}_m$).

Lemma 1 is a general result whose proof can be found in many literatures such as [35]. Using the calculus of variations, it can be easily shown that the "best" distribution for q_m is q_m^*, which depends on the other distributions q_{-m}. The VB alternately updates each variational posterior using

$$q_m = q_m^*, \forall m = 1, 2, \ldots, M, \tag{6}$$

with the other fixed. However, the VB update (5) can be intractable since the expectation is in fact an integral which usually has no analytical solution. Usually, the optimization is further relaxed by approximating the variational distribution to be optimized in the conjugate exponential families of distributions, a broad class of distributions that have been extensively studied in the statistics literature [37], [38]. Specifically, we assume the distributions of variables conditioned on their parents, as presented in Fig. 1, are drawn from the exponential family and are conjugate with respect to the distributions over these parents variables.

A density function in the exponential families can be written in the canonical form

$$Q(\boldsymbol{z}) = h(\boldsymbol{z}) \exp\{\boldsymbol{\phi}^T \boldsymbol{u}(\boldsymbol{z}) - A(\boldsymbol{\phi})\}, \tag{7}$$

where $\boldsymbol{u}(\boldsymbol{z})$ is a collection of functions of $\boldsymbol{z}$, known as natural sufficient statistics, $\boldsymbol{\phi}$ is an associated (natural) vector of canonical parameters, and $A(\boldsymbol{\phi})$ acts as a normalization function ensuring that the distribution integrates to unity for any given setting of the parameters. The natural parameter vector $\boldsymbol{\phi}$ belongs to the set (natural parameter space)

$$\Omega := \{\boldsymbol{\phi} | A(\boldsymbol{\phi}) < +\infty\}, \tag{8}$$

where the log partition function $A(\cdot)$ is a convex function of $\boldsymbol{\phi}$, and the domain Ω is a convex set [38]. We restrict our attention to regular exponential families for which the domain Ω is an open set.

Lemma 2: In the conjugate-exponential families, the (5) can be reparameterized as

$$q_m^*(\boldsymbol{z}_m) = h(\boldsymbol{z}_m) \exp\{\boldsymbol{\phi}_m^{*T} \boldsymbol{u}(\boldsymbol{z}_m) - A(\boldsymbol{\phi}_m^*)\}, \tag{9}$$

where the natural parameter vector $\boldsymbol{\phi}_m^*$ is a function of expectations of related natural sufficient statistics.

Lemma 2 is also a general result. Use the properties of conjugate-exponential families, the variational message passing (VMP) algorithm is formed to apply variational inference into a Bayesian network. Following the VMP framework, Lemma 2 can be easily derived [39].

Remark 1: The expectation and covariance of the natural sufficient statistics $\boldsymbol{u}(\boldsymbol{z}_m)$ can be derived by

$$\mathbb{E}[\boldsymbol{u}(\boldsymbol{z})] = \nabla_{\boldsymbol{\phi}} A(\boldsymbol{\phi}), \tag{10a}$$

$$\mathbb{E}\left[(\boldsymbol{u}(\boldsymbol{z}) - \mathbb{E}[\boldsymbol{u}(\boldsymbol{z})])(\boldsymbol{u}(\boldsymbol{z}) - \mathbb{E}[\boldsymbol{u}(\boldsymbol{z})])^T\right] = \nabla^2_{\boldsymbol{\phi}} A(\boldsymbol{\phi}), \tag{10b}$$

which guarantee the computational tractability of the calculation of the natural parameter vector $\boldsymbol{\phi}^*_m$ in Lemma 2.

Since the form of variational distributions is known in prior and stays unchanged in the iterations, the "best" distribution given in (9) can be totally determined and represented by its natural parameter vector $\boldsymbol{\phi}^*_m$. Therefore, the variational update (6) can be simply written as

$$\boldsymbol{\phi}_m = \boldsymbol{\phi}^*_m, \forall m = 1, \ldots, M. \tag{11}$$

It tells us that the variational problem (4) can be optimized directly through its natural parameters in the parameter space rather than the variational distribution in the probability space. We will take advantage of this fact in developing distributed VB methods below.

We can write (11) in a more familiar way. The variational Bayesian algorithm alternates between maximizing the lower bound with respect to the variational distribution of latent variables $\boldsymbol{y}$ and that of the model parameters $\boldsymbol{\theta}$. Therefore, the VB procedure (11) is like that of the expectation-maximization (EM) algorithm [40], consisting of two iterative steps

$$\text{VBE}: \boldsymbol{\phi}^*_y = \arg\max_{\boldsymbol{\phi}_y} \mathcal{L}(\boldsymbol{\phi}_y, \boldsymbol{\phi}^*_\theta), \tag{12a}$$

$$\text{VBM}: \boldsymbol{\phi}^*_\theta = \arg\max_{\boldsymbol{\phi}_\theta} \mathcal{L}(\boldsymbol{\phi}^*_y, \boldsymbol{\phi}_\theta). \tag{12b}$$

In the next section, we propose two novel algorithms to solve the optimization problems in the VB procedure (12) in a distributed fashion.

III. Distributed Variational Bayesian Algorithms

As mentioned in the Introduction, the total observed data $\boldsymbol{x}$ is not fully accessible in a single node, which makes the distributed inference problem difficult. Fortunately, the joint distribution $P(\boldsymbol{z}, \boldsymbol{x})$ in the objective function (2) can be decomposed under the conditional independence assumptions. It can be written as the product of the conditional distributions, in which the likelihood of the observed data at each node is separated, as shown below

$$\begin{aligned} P(\boldsymbol{z}, \boldsymbol{x}) &= P(\{\boldsymbol{y}_i\}, \boldsymbol{\theta}, \boldsymbol{x}) = P(\boldsymbol{\theta}) \prod_i^N P(\boldsymbol{x}_i|\boldsymbol{y}_i, \boldsymbol{\theta}) P(\boldsymbol{y}_i|\boldsymbol{\theta}) \\ &= \left(P(\boldsymbol{\theta}) \prod_i^N P(\boldsymbol{y}_i|\boldsymbol{\theta})\right) \prod_i^N P(\boldsymbol{x}_i|\boldsymbol{y}_i, \boldsymbol{\theta}) \\ &= P(\boldsymbol{z}) \prod_i^N P(\boldsymbol{x}_i|\boldsymbol{z}), \end{aligned} \tag{13}$$

where $P(\boldsymbol{x}_i|\boldsymbol{y}_i, \boldsymbol{\theta})$ is replaced by $P(\boldsymbol{x}_i|\boldsymbol{z})$ in the third equation, for notational convenience. Therefore, with a multiplication and division step, the global lower bound $\mathcal{L}(Q)$ is replaced by an average of the local lower bounds,

$$\begin{aligned} \mathcal{L}(Q(\boldsymbol{z})) &= \mathbb{E}_Q[\log P(\boldsymbol{z}, \boldsymbol{x})] + \mathbb{H}[Q(\boldsymbol{z})] \\ &= \frac{1}{N} \sum_{i=1}^N N\mathbb{E}_Q[\log P(\boldsymbol{x}_i|\boldsymbol{z})] + \mathbb{E}_Q[\frac{P(\boldsymbol{z})}{Q(\boldsymbol{z})}] \\ &= \frac{1}{N} \sum_{i=1}^N \mathcal{L}_i(Q(\boldsymbol{z})), \end{aligned} \tag{14}$$

where

$$\mathcal{L}_i(Q) \triangleq \mathbb{E}_Q[\log \frac{P(\boldsymbol{x}_i|\boldsymbol{z})^N P(\boldsymbol{z})}{Q(\boldsymbol{z})}]. \tag{15}$$

The quantity $\mathcal{L}_i(Q)$ is the lower bound for the log evidence of the observed data $\{\boldsymbol{x}_i\}_N$ at node i, where $\{\cdot\}_N$ means that the observed data $\boldsymbol{x}_i$ is replicated N times. Namely,

$$\mathcal{L}_i(Q) \leq \log E_{P(\boldsymbol{z})}[P(\boldsymbol{x}_i|\boldsymbol{z})^N] = \log P(\{\boldsymbol{x}_i\}_N).$$

The first inequality is derived from Jensen's inequality considering the concavity of the logarithmic function. The equality holds if and only if $Q(\boldsymbol{z}) \sim P(\boldsymbol{x}_i|\boldsymbol{z})^N P(\boldsymbol{z})$. Namely, $Q(\boldsymbol{z})$ is exactly the posterior of the unobserved data $\boldsymbol{z}$ given the replicated observed data $\{\boldsymbol{x}_i\}_N$. Although the lower bound can be decomposed, we can not independently maximize the local lower bound at each node to reach a global optimum. A key observation is that the global lower bound for the log evidence of the full observed data is definitely less than or equal to the averaged lower bound for the log evidence of the local replicated observed data over all nodes. Mathematically,

$$\begin{aligned} \max_Q \mathcal{L}(Q) &= \frac{1}{N} \sum_{i=1}^N \mathcal{L}_i(Q^*) \\ &\leq \frac{1}{N} \sum_{i=1}^N \mathcal{L}_i(Q_i^*) \\ &= \frac{1}{N} \sum_{i=1}^N \max_Q \mathcal{L}_i(Q) \end{aligned} \tag{16}$$

where Q^* is the optimal variational distribution maximizing the global lower bound $\mathcal{L}(Q)$, and Q_i^* is the one maximizing the local lower bound for the log evidence $\log P(\{\boldsymbol{x}_i\}_N)$. The equality in the second line holds if and only if Q^* is also the optimal solution for all the local lower bounds, which is not always the case. So we can not find the optimal solution by individually maximizing the local lower bound at each node, as done in the third line, and a distributed approach has to be designed to solve this problem.

As mentioned in the previous section, the lower bound can be directly optimized in the space of natural parameters. The latent variables are local variables, and we denote the natural parameter vectors of latent variables at each node by $\{\boldsymbol{\phi}_{y_i}, i = 1, \ldots, N\}$. By replacing the lower bound with the decomposed version (14), the VB procedure is rewritten as

$$\text{VBE}: \boldsymbol{\phi}^*_{y_i} = \arg\max_{\boldsymbol{\phi}_{y_i}} \mathcal{L}_i(\boldsymbol{\phi}_{y_i}, \boldsymbol{\phi}^*_\theta), \forall\, i = 1, \ldots, N, \tag{17a}$$

$$\text{VBM}: \boldsymbol{\phi}^*_\theta = \arg\max_{\boldsymbol{\phi}_\theta} \sum_{i=1}^N \mathcal{L}_i(\boldsymbol{\phi}^*_{y_i}, \boldsymbol{\phi}_\theta). \tag{17b}$$

Given the global natural parameters $\boldsymbol{\phi}_\theta^*$, the VBE step (17a) can be solved individually at each node. The solution can be directly found using Lemma 1 and Lemma 2 with a slight modification that replaces $\boldsymbol{x}$ with $\{\boldsymbol{x}_i\}_N$. However, the VBM step can not be solved straightforwardly. In the next two subsections, two approaches are presented for distributedly computing (17b).

A. Distributed Stochastic VB

Let's define a set of local natural parameter vectors $\{\boldsymbol{\phi}_{\theta,i}, i = 1,\ldots,N\}$ for each node. According to Lemma 1 and Lemma 2, the "best" variational distribution for the global model parameters at each node with that of the latent variables fixed is $q_{\theta,i}^*$, and its natural parameter vector is $\boldsymbol{\phi}_{\theta,i}^*$. In other words, the local lower bound $\mathcal{L}_i$ is maximized at $\boldsymbol{\phi}_{\theta,i}^*$ with $\boldsymbol{\phi}_{y_i}^*$ fixed, namely

$$\boldsymbol{\phi}_{\theta,i}^* = \arg\max_{\boldsymbol{\phi}_\theta} \mathcal{L}_i(\boldsymbol{\phi}_{y_i}^*, \boldsymbol{\phi}_\theta). \tag{18}$$

Then the solution for (17b) is found by taking the derivative of $\mathcal{L}_i$ with respect to $\boldsymbol{\phi}_\theta$,

$$\begin{aligned}
&\sum_{i=1}^N \frac{\partial}{\partial \boldsymbol{\phi}_\theta} \mathcal{L}_i(\boldsymbol{\phi}_{y_i}^*, \boldsymbol{\phi}_\theta) = -\sum_{i=1}^N \frac{\partial}{\partial \boldsymbol{\phi}_\theta} \mathrm{KL}(q_\theta \| q_{\theta,i}^*) \\
&= -\sum_{i=1}^N \frac{\partial}{\partial \boldsymbol{\phi}_\theta} \left((\boldsymbol{\phi}_\theta - \boldsymbol{\phi}_{\theta,i}^*)^T \mathbb{E}_\theta[\boldsymbol{u}(\theta)] - A(\boldsymbol{\phi}_\theta) + A(\boldsymbol{\phi}_{\theta,i}^*) \right) \\
&= -\sum_{i=1}^N \frac{\partial}{\partial \boldsymbol{\phi}_\theta} \left((\boldsymbol{\phi}_\theta - \boldsymbol{\phi}_{\theta,i}^*)^T \nabla_{\boldsymbol{\phi}_\theta} A(\boldsymbol{\phi}_\theta) - A(\boldsymbol{\phi}_\theta) \right) \\
&= -\sum_{i=1}^N \nabla_{\boldsymbol{\phi}_\theta}^2 A(\boldsymbol{\phi}_\theta)(\boldsymbol{\phi}_\theta - \boldsymbol{\phi}_{\theta,i}^*)
\end{aligned} \tag{19}$$

where the first, second and third equalities are derived from (4), (9) and (10a), respectively. Set the partial derivative to zero, we obtain the solution for the VBM step (17b)

$$\boldsymbol{\phi}_\theta^* = \frac{1}{N} \sum_{i=1}^N \boldsymbol{\phi}_{\theta,i}^*. \tag{20}$$

It is an average of all the local optimal natural parameters calculated at each node. If there exists a fusion center, which can receive all local optimums from all nodes, then a centralized VB is obtained. If a cyclic path through all the nodes could be found, then an incremental VB algorithm can be derived. However, neither can be applied in a low-cost networked system, since the communication resources are limited and exploring the network topology is hard and expensive.

Distributed averaging consensus approach has been proposed to solve this kind of problem [30], [41]. However, a key weakness of this approach is that too many iterations are needed to reach a consensus in each VB step. A diffusion-based EM algorithm is proposed in [42] for distributed estimation of Gaussian mixtures, in which each node diffuses its local statistics with its neighbors only once per EM step (one-step averaging). One might think that we can borrow this simple one-step averaging idea to develop a distributed VB, and approximate (20) using $\boldsymbol{\phi}_{\theta,i}^t = \frac{1}{|\mathcal{N}_i|+1} \sum_{j\in\mathcal{N}_i\cup\{i\}} \boldsymbol{\phi}_{\theta,j}^{*,t}, \forall i = 1,\ldots,N$, where t is a time instant and $|\mathcal{N}_i|$ is the degree of node i (i.e., the number of neighbors of the node i). However, it can not provide a good result when the local data is imbalanced. Since the best value of $\boldsymbol{\phi}_{\theta,i}^t$ in this procedure is totally determined by its own and its neighbors' local latent variables $\{\boldsymbol{\phi}_{y_j}^t\}_{j\in\mathcal{N}_i\cup\{i\}}$, which is not a representative of the whole set of latent variables $\{\boldsymbol{\phi}_{y_i}^t\}_{i=1}^N$ when the local data is imbalanced.

We propose a gradient-based method for distributed estimation. Instead of using the first-order condition in the VBM step to yield a local optimum $\boldsymbol{\phi}_{\theta,i}^*$ at each node, we use a stochastic gradient method [25], [43], followed by a diffusion procedure [8] to get the global natural parameters (20) gradually and approximately. We denote an intermediate quantity estimated by a gradient ascent step at node i as $\boldsymbol{\varphi}_{\theta,i}$. After computing this intermediate quantity, a simple combination step is followed. Namely, for each time t, the update equations at node i are

$$\boldsymbol{\varphi}_{\theta,i}^t = \boldsymbol{\phi}_{\theta,i}^{t-1} + \eta_t \tilde{\nabla}_{\boldsymbol{\phi}_\theta} \mathcal{L}_i(\boldsymbol{\phi}_{y_i}^*, \boldsymbol{\phi}_{\theta,i}^{t-1}), \tag{21a}$$

$$\boldsymbol{\phi}_{\theta,i}^t = \sum_{j\in\mathcal{N}_i\cup\{i\}} w_{ij} \boldsymbol{\varphi}_{\theta,j}^t, \tag{21b}$$

where $\tilde{\nabla}_{\boldsymbol{\phi}_\theta} \mathcal{L}_i$ denotes the natural gradient [31] in a Riemannian space (explained in following paragraphs), η_t is the step size satisfying [43]

$$\sum \eta_t = \infty; \sum \eta_t^2 < \infty, \tag{22}$$

and $\{w_{ij}\}$ are non-negative weights satisfying

$$\sum_{j=1}^N w_{ij} = 1, w_{ij} = 0 \text{ if } j \notin \mathcal{N}_i \cup \{i\}. \tag{23}$$

There are many possible rules for choosing the weights $\{w_{ij}\}$, such as the Metropolis, the Laplacian and the nearest neighbors rules [6], [30], [44]. In addition, we can also use some strategies to optimize the combination weights [9], [45].

This approach is motivated by the diffusion LMS algorithm [8], [46], which addresses a distributed linear estimation problem in a cooperative fashion. Though the steady-state performance of the diffusion cooperation scheme has been well studied for the linear estimation problem [8], theoretical performance analysis of the proposed algorithm (21) is hard since the local lower bound $\mathcal{L}_i$ is more complex. Instead of providing theoretical analysis, we show that the procedure (21) can be interpreted as a distributed implementation of the stochastic variational inference [25]. Each node runs a gradient ascent step (21a) using only the local data, which is similar to the stochastic approximation based on the subsample [25]. The combination step (21b), which diffuses all local estimates over the entire network, can be considered as a procedure gradually collecting global (all local) sufficient statistics (since $\boldsymbol{\phi}_{\theta,i}$ is a function of expectations of related natural sufficient statistics as shown in Lemma 2) with the iterations of the VB procedure (17). Compared with the one-step averaging approach, our gradient-based approach takes the previous estimate $\boldsymbol{\phi}_{\theta,i}^{t-1}$ into account in (21a), which is a result obtained by diffusing all estimates among nodes over the entire network. Therefore, the convergence point of the gradient ascent procedure (21a) is not a solution of the local objective function $\mathcal{L}_i$ but that of the global objective function (17b).

This approach is based on the natural gradient [31] rather than the standard gradient. The natural gradient of a function

accounts for the information geometry [47] of its parameter space, using a Riemannian metric to adjust the direction of the standard gradient. In variational inference, the natural gradients have been used for nonlinear state space models [48], Bayesian mixtures [49] and latent Dirichlet allocation [25]. The Riemannian metric rather than the Euclidean metric is used in this paper, since the latter can not properly scale the gradient under the manifold of a probability distribution. As we know, different parameters of a distribution have different roles, such as location, shape and scale, and the effect of one parameter can be mutually influenced by the other parameters. For example [25], the distributions $\mathcal{N}(0, 10^5)$ and $\mathcal{N}(10, 10^5)$ are almost indistinguishable, and the Euclidean distance between means of those two distributions is 10. In contrast, the distributions $\mathcal{N}(0, 0.01)$ and $\mathcal{N}(0.1, 0.01)$ barely overlap, but this is not reflected in the Euclidean distance between their mean parameters, which is only 0.1. It is showed in [31] that the parameter space of a distribution has a Riemannian metric structure. In this case, the natural gradient can give the steepest direction.

In a Riemannian space of parameters, the steepest ascent direction of the objective function $\mathcal{L}_i(\boldsymbol{\phi}^*_{y_i}, \boldsymbol{\phi}_\theta)$ for $\boldsymbol{\phi}_\theta$ with $\boldsymbol{\phi}^*_{y_i}$ fixed is given by

$$\tilde{\nabla}_{\boldsymbol{\phi}_\theta} \mathcal{L}_i(\boldsymbol{\phi}^*_{y_i}, \boldsymbol{\phi}_\theta) = G^{-1}(\boldsymbol{\phi}_\theta) \nabla_{\boldsymbol{\phi}_\theta} \mathcal{L}_i(\boldsymbol{\phi}^*_{y_i}, \boldsymbol{\phi}_\theta), \tag{24}$$

where G^{-1} is the inverse of the Riemannian metric and $\nabla_{\boldsymbol{\phi}_\theta} \mathcal{L}_i$ is the standard gradient, which is given by

$$\nabla_{\boldsymbol{\phi}_\theta} \mathcal{L}_i = \nabla^2_{\boldsymbol{\phi}_\theta} A(\boldsymbol{\phi}_\theta)(\boldsymbol{\phi}_\theta - \boldsymbol{\phi}^*_{\theta,i}). \tag{25}$$

The metric $G(\boldsymbol{\phi}_\theta)$ is the Fisher information matrix of a distribution [50]. Since $\boldsymbol{\phi}_\theta$ is a natural parameter of an exponential family distribution $P(\boldsymbol{\theta}|\boldsymbol{\phi}_\theta)$, the Fisher metric $G(\boldsymbol{\phi}_\theta)$ defined by $P(\boldsymbol{\theta}|\boldsymbol{\phi}_\theta)$ is the second derivative of its log partition function $A(\boldsymbol{\phi}_\theta)$. Mathematically,

$$\begin{aligned} G(\boldsymbol{\phi}_\theta) &= \mathbb{E}\left[(\nabla_{\boldsymbol{\phi}_\theta} \log P(\boldsymbol{\theta}|\boldsymbol{\phi}_\theta))(\nabla_{\boldsymbol{\phi}_\theta} \log P(\boldsymbol{\theta}|\boldsymbol{\phi}_\theta))^T\right] \\ &= \mathbb{E}\left[(\boldsymbol{u}(\boldsymbol{\theta}) - \mathbb{E}[\boldsymbol{u}(\boldsymbol{\theta})])(\boldsymbol{u}(\boldsymbol{\theta}) - \mathbb{E}[\boldsymbol{u}(\boldsymbol{\theta})])^T\right] \\ &= \nabla^2_{\boldsymbol{\phi}_\theta} A(\boldsymbol{\phi}_\theta), \end{aligned}$$

where the third equality is derived from (10b). The natural gradient of $\mathcal{L}_i(\boldsymbol{\phi}^*_{y_i}, \boldsymbol{\phi}_\theta)$ w.r.t. $\boldsymbol{\phi}_\theta$ is then simplified as

$$\begin{aligned} \tilde{\nabla}_{\boldsymbol{\phi}_\theta} \mathcal{L}_i(\boldsymbol{\phi}^*_{y_i}, \boldsymbol{\phi}_\theta) &= \nabla^{-2}_{\boldsymbol{\phi}_\theta} A(\boldsymbol{\phi}_\theta) \nabla_{\boldsymbol{\phi}_\theta} \mathcal{L}_i(\boldsymbol{\phi}^*_{y_i}, \boldsymbol{\phi}^{t-1}_{\theta,i}) \\ &= \boldsymbol{\phi}^*_{\theta,i} - \boldsymbol{\phi}^{t-1}_{\theta,i}. \end{aligned} \tag{26}$$

Substituting it into (21a), we obtain the procedure for computing the global natural parameters at each node i

$$\varphi^t_{\theta,i} = \boldsymbol{\phi}^{t-1}_{\theta,i} + \eta_t(\boldsymbol{\phi}^{*,t}_{\theta,i} - \boldsymbol{\phi}^{t-1}_{\theta,i}), \tag{27a}$$

$$\boldsymbol{\phi}^t_{\theta,i} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} \varphi^t_{\theta,j}. \tag{27b}$$

At time instant t, each node calculates its intermediate quantity $\varphi^t_{\theta,i}$ using its local data, then transfers it to its neighbors and also receives messages $\{\boldsymbol{\varphi}^t_{\theta,j}, j \in \mathcal{N}_i\}$ from the neighbors. Since usually the size and dimension of the intermediate quantities are much smaller than the raw data, it saves the communication resources and energy to a great extent. Rather than reaching a consensus in each VB step, the network diffuses the information along with the VB iterations so that (27) only needs to iterate once in a single VB step. To further analyze this process, we rewrite (27) into a single update:

$$\boldsymbol{\phi}^t_{\theta,i} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} \boldsymbol{\phi}^{t-1}_{\theta,j} + \eta_t \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} (\boldsymbol{\phi}^{*,t}_{\theta,j} - \boldsymbol{\phi}^{t-1}_{\theta,j}). \tag{28}$$

The role of the first term in (28) is to diffuse information over the entire network. The second term gradually updates estimate using the local data and the information received from its neighbors. The step size η_t is a trade-off between the diffusion speed (the first term) and the learning speed (the second term). Although the residuals in the second term would not strictly be eliminated, a sufficiently small step size η_t ensures a small steady state error. But a small step size also decreases the convergence speed. Conversely, a large step size improves the rate of convergence but might lead to instability.

Remark 2 (On the Selection of the Step Size η_t): We suggest to use a time-varying step size for the natural gradient,

$$\eta_t = \frac{1}{d_0 + \tau t}, 1 \leq d_0, 0 < \tau < 1, \tag{29}$$

which satisfies the conditions (22). In (29), the forgetting rate τ controls the decreasing speed of the step size and the parameter d_0 down-weights early iterations [25]. In Section V, we empirically fix $d_0 = 1$. We explore a variety of forgetting rates, and numerical simulations show that $\tau \in [0.1, 0.3]$ could give a good performance for kinds of applications/problems.

For clarity, the distributed stochastic variational Bayesian algorithm (dSVB) is summarized in Algorithm 1.

Algorithm 1: The dSVB algorithm

Require: Node i observes data $\boldsymbol{x}_i$. The natural parameters are initialized using non-informative priors.

Set the tuning parameter τ appropriately.
for $t \leftarrow 1, 2, \ldots$ **do** ▷t: time step
  for all $i = 1, \ldots, N$ **do**
    $\boldsymbol{\phi}^{*,t}_{y_i} = \arg\max_{\boldsymbol{\phi}_{y_i}} \mathcal{L}_i(\boldsymbol{\phi}_{y_i}, \boldsymbol{\phi}^{t-1}_{\theta,i})$. ▷VBE
    $\boldsymbol{\phi}^{*,t}_{\theta,i} = \arg\max_{\boldsymbol{\phi}_\theta} \mathcal{L}_i(\boldsymbol{\phi}^{*,t}_{y_i}, \boldsymbol{\phi}_\theta)$.
    Compute $\boldsymbol{\varphi}^t_{\theta,i}$ via (27a). ▷ Natural gradient
    Broadcast $\boldsymbol{\varphi}^t_{\theta,i}$ to all neighbors in $\mathcal{N}_i$.
  end for
  for all $i = 1, \ldots, N$ **do**
    Compute $\boldsymbol{\phi}^t_{\theta,i}$ via (27b). ▷ Combination
  end for
end for

B. Distributed VB Based on ADMM

In this subsection, we reformulate the VBM step as a constrained minimization problem, and then we use a well-studied optimization technique, the alternating direction method of multipliers (ADMM) [32], to solve it. By adding the consensus constraints that local variables $\{\boldsymbol{\phi}_{\theta,i}\}$ agree upon with its neighbors, the maximization problem (17b) of the VBM step can be formulated as,

$$\begin{aligned}\min_{\{\boldsymbol{\phi}_{\theta,i}\},\{\boldsymbol{\varphi}_{\theta,i,j}\}} & -\sum_{i=1}^{N}\mathcal{L}_i(\boldsymbol{\phi}^*_{y_i},\boldsymbol{\phi}_{\theta,i})\\ \text{s.t.}\quad & \boldsymbol{\phi}_{\theta,i}=\boldsymbol{\varphi}_{\theta,i,j}, i=1,\ldots,N, j\in\mathcal{N}_i\\ \text{s.t.}\quad & \boldsymbol{\varphi}_{\theta,i,j}=\boldsymbol{\phi}_{\theta,j}, i=1,\ldots,N, j\in\mathcal{N}_i, \end{aligned}\tag{30}$$

where the auxiliary variables $\boldsymbol{\varphi}_{\theta,i,j}$ decouple local variables $\boldsymbol{\phi}_{\theta,i}$ at node i from those of their neighbors $j\in\mathcal{N}_i$. With the assumption that the network remains connected, the consensus constraints guarantee that problem (17b) and (30) are equivalent. Since there exists a path between any two nodes in the network, the consensus constraints imply that the local variables in these two nodes are equal. Since the pair of nodes is arbitrary, any feasible solution for (30) is also a solution for (17b).

The ADMM technique can not be used for solving (30) in a straightforward manner. The objective function in (30) is a function of a variational distribution, whose variational parameters have the Riemannian metric structure, as discussed in the previous subsection. In contrast, the equality constraints and quadratic penalty terms in the standard method of multipliers are in fact using the Euclidean metric. One possible way for solving this inconsistence is to replace the penalty term with a more general deviation penalty, whose parameter space also has the Riemannian character, such as the one derived from a Bregman divergence [51]. Unfortunately, to the best of our knowledge, there is no proof of the convergence of ADMM with nonquadratic penalty terms available currently.

Therefore, we turn our attention to designing an objective function using the Euclidean metric that is equivalent to (30). Using the results derived in (26), it is easy to show that the natural gradient of (30) with respect to $\boldsymbol{\phi}_{\theta,i}$ is $\boldsymbol{\phi}_{\theta,i}-\boldsymbol{\phi}^*_{\theta,i}$. This implies that the following relation holds for all i,

$$\nabla_{\boldsymbol{\phi}_{\theta,i}}\left(\frac{1}{2}\|\boldsymbol{\phi}_{\theta,i}-\boldsymbol{\phi}^*_{\theta,i}\|_F^2\right)=-\tilde{\nabla}_{\boldsymbol{\phi}_{\theta,i}}\mathcal{L}_i(\boldsymbol{\phi}^*_{y_i},\boldsymbol{\phi}_{\theta,i}),$$

where $\|\cdot\|_F$ is the Frobenius norm. Consequently, the original problem (30) can be recast as the following formulation,

$$\begin{aligned}\min_{\boldsymbol{\phi}_{\theta,i},\boldsymbol{\varphi}_{\theta,i,j}} & \frac{1}{2}\sum_{i=1}^{N}\|\boldsymbol{\phi}_{\theta,i}-\boldsymbol{\phi}^*_{\theta,i}\|_F^2\\ \text{s.t.}\quad & \boldsymbol{\phi}_{\theta,i}=\boldsymbol{\varphi}_{\theta,i,j}, i=1,\ldots,N, j\in\mathcal{N}_i,\\ \text{s.t.}\quad & \boldsymbol{\varphi}_{\theta,i,j}=\boldsymbol{\phi}_{\theta,j}, i=1,\ldots,N, j\in\mathcal{N}_i. \end{aligned}\tag{31}$$

We could use a dual decomposition method for the separated objective function (31). However, using the augmented Lagrangian can bring robustness to the dual ascent method [32], so we use this method below. Let $\boldsymbol{\lambda}_{ij1}$ and $\boldsymbol{\lambda}_{ij2}$ denote the Lagrange multipliers corresponding to the constraints $\boldsymbol{\phi}_{\theta,i}=\boldsymbol{\varphi}_{\theta,i,j}$ and $\boldsymbol{\varphi}_{\theta,i,j}=\boldsymbol{\phi}_{\theta,j}$, the augmented Lagrangian function is formulated as

$$\begin{aligned}&\mathcal{L}_\rho(\{\boldsymbol{\phi}_{\theta,i}\},\{\boldsymbol{\varphi}_{\theta,i,j}\},\{\boldsymbol{\lambda}_{ij1},\boldsymbol{\lambda}_{ij2}\})\\ &=\frac{1}{2}\sum_{i=1}^{N}\|\boldsymbol{\phi}_{\theta,i}-\boldsymbol{\phi}^*_{\theta,i}\|_F^2\\ &\quad+\sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i}\Big(\mathrm{Tr}\big(\boldsymbol{\lambda}_{ij1}^T(\boldsymbol{\phi}_{\theta,i}-\boldsymbol{\varphi}_{\theta,i,j})\big)\\ &\quad+\mathrm{Tr}\big(\boldsymbol{\lambda}_{ij2}^T(\boldsymbol{\varphi}_{\theta,i,j}-\boldsymbol{\phi}_{\theta,j})\big)\Big)\\ &\quad+\frac{\rho}{2}\sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i}\Big(\|\boldsymbol{\phi}_{\theta,i}-\boldsymbol{\varphi}_{\theta,i,j}\|_F^2+\|\boldsymbol{\varphi}_{\theta,i,j}-\boldsymbol{\phi}_{\theta,j})\|_F^2\Big), \end{aligned}\tag{32}$$

where $\rho>0$ is a penalty parameter. The ADMM solves (32) in a cyclic fashion by minimizing $\mathcal{L}_\rho$ with respect to the local variables $\{\boldsymbol{\phi}_{\theta,i}\}$ and auxiliary variables $\{\boldsymbol{\varphi}_{\theta,i,j}\}$, followed by a gradient ascent step over the dual variables $\{\boldsymbol{\lambda}_{ij1},\boldsymbol{\lambda}_{ij2}\}$. Taking the derivative of the Lagrangian (32) with respect to each variable, and setting the derivative to zero, we get a closed-form solution $\forall i=1,\ldots,N,\forall j\in\mathcal{N}_i$,

$$\boldsymbol{\phi}^t_{\theta,i}=\frac{\boldsymbol{\phi}^{*,t}_{\theta,i}+\sum_{j\in\mathcal{N}_i}\left(\boldsymbol{\lambda}^{t-1}_{ji2}-\boldsymbol{\lambda}^{t-1}_{ij1}+\rho(\boldsymbol{\varphi}^{t-1}_{\theta,i,j}+\boldsymbol{\varphi}^{t-1}_{\theta,j,i})\right)}{1+2\rho N_i},\tag{33a}$$

$$\boldsymbol{\varphi}^t_{\theta,i,j}=\frac{1}{2\rho}(\boldsymbol{\lambda}^{t-1}_{ij1}-\boldsymbol{\lambda}^{t-1}_{ij2})+\frac{1}{2}(\boldsymbol{\phi}^t_{\theta,i}+\boldsymbol{\phi}^t_{\theta,j}),\tag{33b}$$

$$\boldsymbol{\lambda}^t_{ij1}=\boldsymbol{\lambda}^{t-1}_{ij1}+\rho(\boldsymbol{\phi}^t_{\theta,i}-\boldsymbol{\varphi}^t_{\theta,i,j}),\tag{33c}$$

$$\boldsymbol{\lambda}^t_{ij2}=\boldsymbol{\lambda}^{t-1}_{ij2}+\rho(\boldsymbol{\varphi}^t_{\theta,i,j}-\boldsymbol{\phi}^t_{\theta,j}).\tag{33d}$$

Substituting (33b) into (33c) and (33d), we have

$$\boldsymbol{\lambda}^t_{ij1}=\frac{1}{2}(\boldsymbol{\lambda}^{t-1}_{ij1}+\boldsymbol{\lambda}^{t-1}_{ij2})+\frac{\rho}{2}(\boldsymbol{\phi}^t_i-\boldsymbol{\phi}^t_j),\tag{34a}$$

$$\boldsymbol{\lambda}^t_{ij2}=\frac{1}{2}(\boldsymbol{\lambda}^{t-1}_{ij1}+\boldsymbol{\lambda}^{t-1}_{ij2})+\frac{\rho}{2}(\boldsymbol{\phi}^t_i-\boldsymbol{\phi}^t_j).\tag{34b}$$

All the Lagrange multipliers are initialized to zeros at each node. By mathematical induction, we know that $\boldsymbol{\lambda}^t_{ij1}=\boldsymbol{\lambda}^t_{ij2}$ and $\boldsymbol{\lambda}^t_{ij1}=-\boldsymbol{\lambda}^t_{ji1},\forall i=1,\ldots,N,j\in\mathcal{N}_i$ for any time instant t. Therefore, the auxiliary variable $\boldsymbol{\varphi}^t_{\theta,i,j}$ can be expressed as

$$\boldsymbol{\varphi}^t_{\theta,i,j}=\frac{1}{2}(\boldsymbol{\phi}^t_{\theta,i}+\boldsymbol{\phi}^t_{\theta,j}).\tag{35}$$

Substituting it into (33a), we have

$$\boldsymbol{\phi}^t_{\theta,i}=\frac{\boldsymbol{\phi}^{*,t}_{\theta,i}-2\boldsymbol{\lambda}^{t-1}_i+\rho\sum_{j\in\mathcal{N}_i}(\boldsymbol{\phi}^{t-1}_{\theta,i}+\boldsymbol{\phi}^{t-1}_{\theta,j})}{1+2\rho N_i},\tag{36}$$

where $\boldsymbol{\lambda}^t_i\triangleq\sum_{j\in\mathcal{N}_i}\boldsymbol{\lambda}^t_{ij1}$ denotes a local aggregate Lagrange multiplier. By substituting (35) into (34a), we obtain an iteration equation for the local aggregate Lagrange multiplier

$$\boldsymbol{\lambda}^t_i=\boldsymbol{\lambda}^{t-1}_i+\rho/2\sum_{j\in\mathcal{N}_i}(\boldsymbol{\phi}^t_{\theta,i}-\boldsymbol{\phi}^t_{\theta,j}).\tag{37}$$

Thus, the auxiliary variables $\{\boldsymbol{\varphi}_{\theta,i,j}\}$ now is eliminated. Alternating between (36) and (37) for all nodes solves the optimization in the VBM step. Thus, we solve the problem (17) in a distributed fashion.

We restrict the ADMM procedure (33) to running only one time in each VBM step in order to save communication resources and energy. However, a numerical issue will arise, which will not happen when the ADMM runs multiple times until it converges. Note that unlike the procedure (27a), (27b) in the dSVB approach, the update (36) is not a convex operation because the sign in front of the multipliers $\boldsymbol{\lambda}_i^t$ is negative. As a result, the updated natural parameters $\boldsymbol{\phi}_{\theta,i}^t$ may not belong to the convex set Ω defined in (8) though $\boldsymbol{\phi}_{\theta,i}^{t-1}$ is in the convex set. In other words, the log partition $A(\boldsymbol{\phi}_{\theta,i}^t)$ may become infinite. For example, the covariance matrix in a normal distribution may become negative definite.

To handle with this issue, a possible solution is to use the projected gradient algorithm [52]. Using this method, the procedure of minimizing the augmented Lagrangian function (32) w.r.t. $\boldsymbol{\phi}_{\theta,i}$ subject to the constraint $\boldsymbol{\phi}_\theta \in \Omega$ can be written as two steps

$$\hat{\boldsymbol{\phi}}_{\theta,i}^t = \frac{\boldsymbol{\phi}_{\theta,i}^{*,t} - 2\boldsymbol{\lambda}_i^{t-1} + \rho \sum_{j\in\mathcal{N}_i}(\boldsymbol{\phi}_{\theta,i}^{t-1} + \boldsymbol{\phi}_{\theta,j}^{t-1})}{1 + 2\rho N_i}, \quad (38a)$$

$$\boldsymbol{\phi}_{\theta,i}^t = \arg\min_{\boldsymbol{\phi}_\theta\in\Omega} \|\boldsymbol{\phi}_\theta - \hat{\boldsymbol{\phi}}_{\theta,i}^t\|_F^2, \quad (38b)$$

where the first step is exactly the same as (36), and the second step is the projection of $\hat{\boldsymbol{\phi}}_{\theta,i}^t$ onto set Ω. Unfortunately, this extra projection step may make the optimization variable $\boldsymbol{\phi}_{\theta,i}^t$ far away from the optimal value (20) when the point $\hat{\boldsymbol{\phi}}_{\theta,i}^t$ is far away from the domain Ω. Hence, we do not simply use this method.

Instead, we use a trick to handle with this numerical issue. We introduce a time-varying parameter κ_t to control the evolution of the dual variables $\boldsymbol{\lambda}_i$. By replacing the step size $\rho/2$ with $\kappa_t\rho/2$, the new gradient ascent iteration for the multipliers becomes

$$\boldsymbol{\lambda}_i^t = \boldsymbol{\lambda}_i^{t-1} + \kappa_t\rho/2 \sum_{j\in\mathcal{N}_i}(\boldsymbol{\phi}_{\theta,i}^t - \boldsymbol{\phi}_{\theta,j}^t). \quad (39)$$

Note that the local optimums $\{\boldsymbol{\phi}_{\theta,i}^*\}$ among nodes could be very different, and the residuals (i.e., dual variables) among nodes could be very large at the very beginning stage of the ADMM iterations. Thus the difference between $\boldsymbol{\lambda}_i^t$ and $\boldsymbol{\lambda}_i^{t-1}$ could be very large. The fact that the sign in front of $\boldsymbol{\lambda}_i^t$ is negative in (36) may make the point $\boldsymbol{\phi}_{\theta,i}^t$ not in the convex set Ω after the step (36). So we set a small value for κ_t at the very beginning stage to ensure $\boldsymbol{\phi}_{\theta,i}^t$ being in the interior of the set Ω (or being not in the interior but close to it, hence the projection step (38b) can be applied), and gradually increase it until it reaches 1. We use the following simple equation for updating the time-varying scalar factor,

$$\kappa_t = 1 - \frac{1}{(1+\xi t)^2}, 0 < \xi < 1, \quad (40)$$

where the parameter ξ controls the increasing speed. Although the optimal value of ξ may depend on the observations, numerical simulations show that a small value of ξ is usually a good choice. We set $\xi = 0.05$ in the following examples.

The distributed VB algorithm based on ADMM (dVB-ADMM) is summarized in the Algorithm 2.

Algorithm 2: The dVB-ADMM algorithm

Require: Node i observes data $\boldsymbol{x}_i$. The natural parameters are initialized using non-informative priors.

Set the penalty parameter ρ appropriately.
Set $\boldsymbol{\lambda}_i = \mathbf{0}, \forall i$.
for $t \leftarrow 1, 2, \ldots$ **do** ▷t: time step
  for all $i = 1, \ldots, N$ **do**
    $\boldsymbol{\phi}_{y_i}^{*,t} = \arg\max_{\boldsymbol{\phi}_{y_i}} \mathcal{L}_i(\boldsymbol{\phi}_{y_i}, \boldsymbol{\phi}_{\theta,i}^{t-1})$. ▷ VBE
    $\boldsymbol{\phi}_{\theta,i}^{*,t} = \arg\max_{\boldsymbol{\phi}_\theta} \mathcal{L}_i(\boldsymbol{\phi}_{y_i}^{*,t}, \boldsymbol{\phi}_\theta)$.
    Compute $\boldsymbol{\phi}_{\theta,i}^t$ via (38). ▷ Primal update
    Broadcast $\boldsymbol{\phi}_{\theta,i}^t$ to all neighbors in $\mathcal{N}_i$.
  end for
  for all $i = 1, \ldots, N$ **do**
    Compute $\boldsymbol{\lambda}_i^t$ via (39). ▷ Dual update
  end for
end for

Remark 3 (On the Selection of the Penalty Parameter ρ): Unlike the step size η_t in the dSVB algorithm, the step size ρ in (37) is not necessarily to be time-varying and to satisfy the condition (22). So we can choose a fixed step size ρ. Numerical simulations in Section V will illustrate that ρ affects the convergence speed of the algorithm. To explain this, we refer to $\boldsymbol{r}_i^t = \sum_{j\in\mathcal{N}_i}(\boldsymbol{\phi}_i^t - \boldsymbol{\phi}_j^t)$ as the primal residual and $\boldsymbol{s}_i^t = \rho\sum_{j\in\mathcal{N}_i\cup i}(\boldsymbol{\phi}_j^t - \boldsymbol{\phi}_j^{t-1})$ as the dual residual at iteration t, which can be derived from the primal and dual feasibility conditions for problem (31). A large value of ρ leads to a large penalty on violations of the primal feasibility and so tends to produce a small primal residual. Conversely, a small value of ρ would produce a small dual residual, which in turn may induce a large primal residual. Numerical simulations in Section V suggest that a relatively small value of ρ is preferable.

Remark 4 (On the Convergence of the dVB-ADMM): Since the distributions here belong to the conjugate-exponential family, the space of natural parameters is always convex [38]. Thus, the "best" natural parameter $\boldsymbol{\phi}_m^*$ in Lemma 2 can always be found. In other words, the solutions of the VBE step (17a) and the local optimization problem (18) at each node exist and can be easily obtained. In the dVB-ADMM, the solution of (18), in obtaining which we uses the solution of (17a), is the local quantities $\{\boldsymbol{\phi}_{\theta,i}^{*,t}\}$ in (36). The VBE step (17a) and the local optimizing step (18) can be viewed as a part of the alternating procedure of the ADMM. As we know, the ADMM is proved to

converge in the context of distributed consensus problems [32], [53], [34]. So, with appropriate penalty parameter, the ADMM iterations (17a), (18), (36) and (39) can guarantee convergence.

IV. Distributed VB for Gaussian Mixture Models

In recent years, the mixture models over sensor networks have been studied and applied to distributed density estimation, distributed clustering, etc. [27], [41], [42]. A standard method for this inference/estimation problem is based on the EM algorithm under the maximum likelihood (ML) framework. However, ML is well-known for its tendency toward overfitting the data and its preference of complex models. A fully Bayesian treatment of mixture modelling can avoid overfitting by integrating out the parameters and identity the optimal structure of models by automatically penalizing the complex model with a lower posterior probability. Unfortunately, the computation of a posterior probability in a Bayesian mixture model is intractable. The VB method provides an analytical approximation solution for this problem [22], [23]. In this section, the proposed algorithms, the dSVB and dVB-ADMM, are applied to a Bayesian Gaussian mixture model (GMM).

Consider a general wireless sensor network with N nodes. Each node i has N_i D-dimension measurements $\boldsymbol{x}_{ij}(i = 1, 2, \ldots, N, j = 1, 2, \ldots, N_i)$. Due to the limitations on energy and communication resources, we can not collect all the data together, so we need to process the data distributedly. We assume the measurements are modeled by a mixture of Gaussians with K components. Each component is a Gaussian distribution with the mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\lambda}_k^{-1}$,

$$\mathcal{N}(\boldsymbol{x}_{ij}|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1}) = \frac{|\boldsymbol{\Lambda}_k|^{1/2}}{(2\pi)^{D/2}} e^{-\frac{1}{2}(\boldsymbol{x}_{ij}-\boldsymbol{\mu}_k)^T \boldsymbol{\Lambda}_k (\boldsymbol{x}_{ij}-\boldsymbol{\mu}_k)}.$$

The Gaussian mixture distribution for observation $\boldsymbol{x}_{ij}$ is

$$p(\boldsymbol{x}_{ij}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_{ij}|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1}), \tag{41}$$

where $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_K\}$, $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$ and $\boldsymbol{\Lambda} = \{\boldsymbol{\Lambda}_1, \ldots, \boldsymbol{\Lambda}_K\}$. The standard mixture distribution (41) does not belong to the exponential family and therefore cannot be used directly as a conditional distribution within a conjugate-exponential model. Instead, we introduce an additional discrete latent variable $\{\boldsymbol{y}_i\}$ for each node, which indicates from which component distribution each data point was drawn. Hence, the local distribution at each node can be written as

$$P(\{\boldsymbol{x}_i\}_N|\boldsymbol{y}_i, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_{j=1}^{N_i} \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{x}_{ij}|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})^{N \cdot y_{ijk}}, \tag{42}$$

where $\boldsymbol{y}_i = \{\boldsymbol{y}_{i1}, \ldots, \boldsymbol{y}_{iN_i}\}$ and $\boldsymbol{y}_{ij} = \{y_{ij1}, y_{ij2}, \ldots, y_{ijK}\}$.

The conjugate priors of the parameters and latent variables need to be specified. The prior of $\boldsymbol{y}_i$ at node i is a product of multinomials, conditional on the mixing coefficients, which are assigned a Dirichlet prior. The means are assigned multivariate Gaussian conjugate priors, conditional on the precision matrices (inverse covariance matrices), which are assigned Wishart priors. All priors are given by

$$P(\boldsymbol{y}_i|\boldsymbol{\pi}) = \prod_{j=1}^{N_i} \mathrm{Mult}(1, \boldsymbol{\pi}), \tag{43a}$$

$$P(\boldsymbol{\pi}) = \mathrm{Dir}(K, \alpha_0), \tag{43b}$$

$$P(\boldsymbol{\mu}|\boldsymbol{\Lambda}) = \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{\mu}_0, (\beta_0 \boldsymbol{\Lambda}_k)^{-1}), \tag{43c}$$

$$P(\boldsymbol{\Lambda}) = \prod_{k=1}^{K} \mathcal{W}(\boldsymbol{W}_0, \nu_0). \tag{43d}$$

With these priors, the generative model for each node i is then factorized using the conditional independence,

$$P(\{\boldsymbol{x}_i\}_N, \boldsymbol{y}_i, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = P(\{\boldsymbol{x}_i\}_N|\boldsymbol{y}_i, \boldsymbol{\mu}, \Lambda) P(\boldsymbol{y}_i|\boldsymbol{\pi}) P(\boldsymbol{\pi}) P(\boldsymbol{\mu}|\boldsymbol{\Lambda}) P(\boldsymbol{\Lambda}).$$

With the mean field assumption, the joint variational distribution of the unobserved variables is factorized as

$$q(\boldsymbol{y}_i, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = q(\boldsymbol{y}_i) q(\boldsymbol{\pi}) \prod_{k=1}^{K} q(\boldsymbol{\mu}_k|\boldsymbol{\Lambda}_k) q(\boldsymbol{\Lambda}_k).$$

Using the VB update (5), the "best" variational distribution can be simply derived. As a general result for a conjugate-exponential model, the form of each distribution is the same as its prior. Specifically, the local optimal variational distributions at node i (only using local data) are

$$q^*(\boldsymbol{y}_i) = \prod_{j=1}^{N_i} \mathrm{Mult}(1, r_{ij1}, \ldots, r_{ijK}), \tag{44a}$$

$$q^*(\boldsymbol{\pi}_i) = \mathrm{Dir}(\alpha_{i1}, \ldots, \alpha_{iK}), \tag{44b}$$

$$q^*(\boldsymbol{\mu}_{ik}, \boldsymbol{\Lambda}_{ik}) = \mathcal{N}(\boldsymbol{m}_{ik}, (\beta_{ik} \boldsymbol{\Lambda}_{ik})^{-1}) \mathcal{W}(W_{ik}, \nu_{ik}), \forall k, \tag{44c}$$

where the update equations for the hyperparameters are given in the Appendix A. The parameters of $q^*(\boldsymbol{y}_i)$ depend on the sufficient statistics of $q^*(\boldsymbol{\pi}_i)$ and $q^*(\boldsymbol{\mu}_{ik}, \boldsymbol{\Lambda}_{ik}), \forall k$, whose parameters in turn depend on the sufficient statistics of $q^*(\boldsymbol{y}_i)$.

To apply the dSVB and dVB-ADMM to a specific model, all we need to do is to derive the local optimum of global natural parameters (18). Note that the natural parameters can be viewed as a function of the hyperparameters, and the hyperparameters can be simply transformed to a natural parameters. It is not necessary to consider the latent variables $\{\boldsymbol{y}_i\}$ since they are local variables. As for the mixing coefficients $\{\boldsymbol{\pi}_i\}$, the natural parameter vector of the Dirichlet distribution is

$$\boldsymbol{\phi}_{\boldsymbol{\pi}_i} = [\alpha_{i1} - 1, \ldots, \alpha_{iK} - 1]^T.$$

The natural parameter vector of the normal-Wishart Distribution is given by

$$\boldsymbol{\phi}_{\boldsymbol{\mu}_{ik}, \boldsymbol{\Lambda}_{ik}} = \begin{bmatrix} \frac{\nu_{ik} - D}{2} \\ -\frac{1}{2}\boldsymbol{W}_{ik}^{-1} - \frac{\beta_{ik}}{2}\boldsymbol{m}_{ik}\boldsymbol{m}_{ik}^T \\ \beta_{ik}\boldsymbol{m}_{ik} \\ -\frac{1}{2}\beta_{ik} \end{bmatrix}, \forall k = 1, \ldots, K.$$

To simplify the notation and to keep notational consistency, we introduce a global natural parameter vector $\boldsymbol{\phi}_{\boldsymbol{\theta},i}$ for the joint

distribution $q^*(\boldsymbol{\pi}_i)\prod_{k=1}^{K} q^*(\boldsymbol{\mu}_{ik}, \boldsymbol{\Lambda}_{ik})$ (also in exponential family), defined as

$$\boldsymbol{\phi}_{\boldsymbol{\theta},i} = [\boldsymbol{\phi}_{\boldsymbol{\pi}_i}^T, \boldsymbol{\phi}_{\boldsymbol{\mu}_{i1},\boldsymbol{\Lambda}_{i1}}^T, \ldots, \boldsymbol{\phi}_{\boldsymbol{\mu}_{iK},\boldsymbol{\Lambda}_{iK}}^T]^T. \tag{45}$$

This global natural parameter vector is the message to be exchanged among nodes. As we see, it is very different from those in the literatures [27], [42], [41], who use the model parameters or sufficient statistics directly. In our method, the natural parameter vector is properly scaled, therefore we do not need to adjust them manually. Substituting the update equations (in Appendix A) of the hyperparameters into (45), we can see that these quantities are very close to the sufficient statistics, but the prior information are included in our framework. In fact, the natural parameter vector is a function of the expectation of related sufficient statistics, as illustrated in Lemma 2.

At the tth iteration of the distributed VB procedure at node i, the local optimal hyperparameters of the global distributions are denoted as $\{\alpha_{ik}^{*,t}\}$, $\{\boldsymbol{m}_{ik}^{*,t}\}$, $\{\boldsymbol{\beta}_{ik}^{*,t}\}$, $\{\boldsymbol{W}_{ik}^{*,t}\}$ and $\{\boldsymbol{\nu}_{ik}^{*,t}\}$ and summarized using a natural parameter vector $\boldsymbol{\phi}_{\boldsymbol{\theta},i}^{*,t}$ (corresponding to the quantity in (18)). Once the update equation of the natural parameter vector is derived and substituted into Algorithm 1 and Algorithm 2, the dSVB and dVB-ADMM algorithms for distributed inference/estimation of Gaussian mixtures are immediately obtained, respectively.

V. Experimental Results

In this section, the performance of the proposed VB algorithms for distributed inference/estimation of Gaussian mixture model is evaluated via numerical simulations on both synthetic and real-world datasets.

A. Performance of the Distributed Stochastic VB Algorithm

We consider a randomly generated sensor network with 50 nodes. The nodes are randomly placed in a 3.5×3.5 square, and the communication distance is taken as 0.8. The constructed connected network has 144 edges, as shown in Fig. 2. The 2-dimensional observations are generated from the mixture of three Gaussian components ($K = 3$). The corresponding parameter settings are as follows

$$\begin{aligned}
&\boldsymbol{\pi} = (0.32, 0.45, 0.23),\\
&\mu_1 = (1.5,\ 3.5), \mu_2 = (4, 4), \mu_3 = (6.5, 4.5),\\
&\Sigma_1 = \Sigma_3 = \begin{bmatrix} 0.6, & 0.4 \\ 0.4, & 0.6 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 0.6, & -0.4 \\ -0.4, & 0.6 \end{bmatrix}.
\end{aligned}$$

Each node has 100 data observations available ($N_i = 100, i = 1, \ldots, 50$). In the first 15 nodes (node 1 to node 15), 80% observations come from the first Gaussian component and the other 20% observations are evenly from the other two Gaussian components. In the next 20 nodes (node 16 to node 35), 90% observations come from the second Gaussian component and the other 10% are observations evenly from the other two Gaussian components. In the last 15 nodes (node 36 to node 50), 60% observations come from the third Gaussian component and the other 40% observations are evenly from the other two Gaussian components.

The measure of performance used here is different from that used in the previous works [29], [34]. Firstly, we point out that the local/global free energy can not correctly assess the algorithm's performance, because the average of all local lower bounds, which is obtained by maximizing each local free energy independently, is always greater than or equal to that obtained by maximizing the global free energy, as shown in (16). Secondly, the simple mean squared error (MSE) of estimates can not evaluate the algorithm's performance well, since natural parameters differ greatly in magnitude. The one that has the largest magnitude will impact the MSE more than the others. The KL divergence is a good measure of the difference between two distributions. It can measure the information lost when using one distribution to approximate another. In general, we can not use the KL divergence as the measure of performance, since the true posterior is unknown. However, in this *synthetic* example, we can compute the ground truth posterior of model parameters $P(\boldsymbol{\theta}|\hat{\boldsymbol{\phi}}_\theta)$ in closed form based on the Bayes' theorem [22], since the ground truth observation model belongs to exponential families and it has a conjugate prior. Thus, we use the KL divergence between the joint variational distribution of model parameters $Q(\boldsymbol{\theta}|\boldsymbol{\phi}_{\theta,i})$ estimated at each node and the ground truth posterior $P(\boldsymbol{\theta}|\hat{\boldsymbol{\phi}}_\theta)$,

$$d(\boldsymbol{\phi}_{\theta,i}, \hat{\boldsymbol{\phi}}_\theta) = \mathrm{KL}\big(Q(\boldsymbol{\theta}|\boldsymbol{\phi}_{\theta,i})||P(\boldsymbol{\theta}|\hat{\boldsymbol{\phi}}_\theta)\big), \tag{46}$$

to assess the local performance, and the mean of all KL divergences to assess the global performance. Since Q and P are the same joint distribution (with different parameters) belonging to the exponential families, the KL divergence can be easily obtained in terms of a closed-form expression. The detailed computation of the KL divergence is given in Appendix B.

For comparison, we simulate the centralized VB algorithm (cVB) for the GMM, in which the global natural parameters (20) are computed by using all local quantities in a fusion center. We also simulate the non-stochastic-gradient based distributed VB algorithm (nsg-dVB), in which each node only uses its local optimum to diffuse the information with its neighbors.

There are two types of parameters needed to be determined for the dSVB. The first is the combination weight. In all of the following experiments, we simply assign it using the nearest neighbors rule [30],

$$w_{ij} \triangleq \begin{cases} \frac{1}{|\mathcal{N}_i|+1}, & \text{if } j \in \mathcal{N}_i \cup i \\ 0, & \text{elsewise} \end{cases} \tag{47}$$

where $|\mathcal{N}_i|$ denotes the degree of node i.

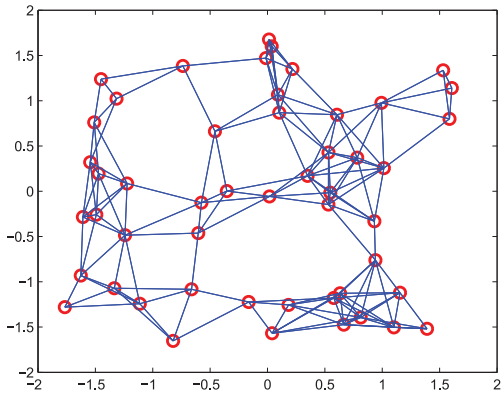

Fig. 2. Network connection.

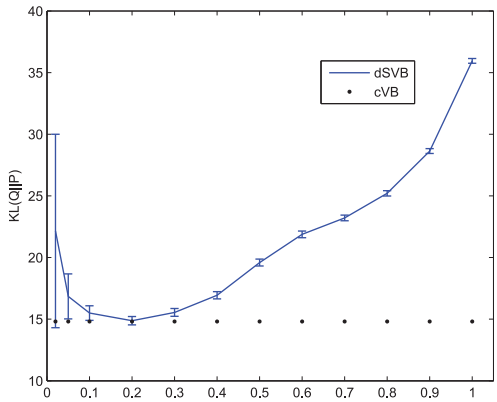

Fig. 3. Error bars comparison between centralized VB and distributed SVB for various values of τ.

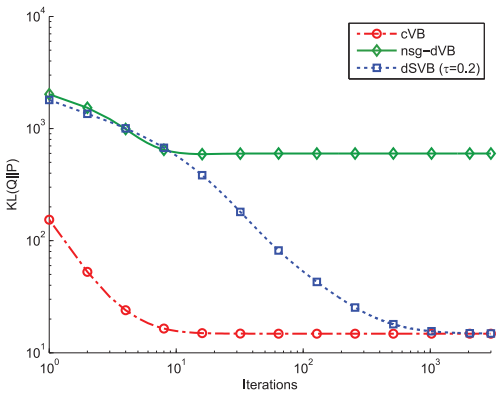

Fig. 4. The evolution of the mean cost (KL divergence) among all nodes of the dSVB($\tau = 0.2$), compared with the cVB and nsg-dVB.

The second is the forgetting rate τ for the step size η_t in (29). As analyzed in Section III-A, the choice of the forgetting rate τ is important for the performance of the dSVB. We next show how the performance depends on the value of τ numerically. Fig. 3 shows the means and the standard deviations of the cost (46) among all 50 nodes obtained after $t = 2000$ iterations for different values of the forgetting rate τ with the same initialization. For comparison, Fig. 3 also shows the cost of the centralized VB. As we see, the mean of the cost is approximately minimized in the interval $[0.1, 0.3]$ and making it either smaller or larger will lead to a higher cost. The standard deviation of the cost measures the difference of estimates among all nodes. This test reveals that the standard deviation goes down when the forgetting rate increases, because the step size η_t becomes smaller, so as the residuals in the second term of (28). Taking both the mean and standard deviation of the cost into account, we choose $\tau = 0.2$ in the following simulations.

In the second simulation, the performance of the dSVB is tested and compared with the cVB and nsg-dVB. Fig. 4 shows the evolution of the mean cost (KL divergence) with the iterations. The non-stochastic-gradient based VB (nsg-dVB) algorithm gets stuck at a local optimum and induces a very large bias, because the previous combination effect is eliminated by the VBE step and only the local information is utilized in every VBM step. Unlike the nsg-dVB, the dSVB improves the estimates gradually with the information diffused over the entire network. Finally the dSVB reaches a result as good as the centralized VB.

In Fig. 5, the final estimates among all nodes obtained after 3000 iterations (the dSVB converges after 1000 iterations in most cases) are compared with those obtained by the cVB and nsg-dSVB. It can be seen that the estimates obtained by the nsg-dVB are very different among all nodes. In contrast, the estimate obtained by the dSVB at each node is very close to that obtained by the centralized VB. It is worth pointing out that although the local observed data at each node is imbalanced, all local imbalanced data together is balanced. The dSVB scheme can maintain a balance between the information from local data and that from neighbors. As an example, Fig. 6 shows the contours of models estimated by different approaches at a randomly selected node. As we see, the nsg-dVB can not correctly estimate the mixture model, and it is still strongly impacted by the local imbalanced data. As for the dSVB, the estimated model is almost the same as the cVB's and the ground truth. The other nodes have similar results, which are not given due to space limitations.

B. Performance of the Distributed VB-ADMM Algorithm

In this subsection, the performance of the dVB-ADMM is tested using the same configuration established in the previous subsection.

The first experiment explores the convergence property of the dVB-ADMM with different values of the penalty parameter ρ. Fig. 7 reveals that a small value of ρ can give faster convergence. While larger values of ρ ensure that the natural parameters per node achieve the same value faster, since it produces smaller primal residuals $\{\boldsymbol{r}_i^t\}$, as we have analyzed in *Remark* 3. Note that if the value of ρ is too small, the primal residuals among nodes may become very large at the very beginning stage of VB iterations, which may give more chance to natural parameters to be out of the domain Ω. In this simulation, the covariance matrices of Gaussian components sometimes become negative definite when $\rho < 0.5$ (without projection). Therefore, the value of ρ should be small for the fast convergence speed but not too small. Unless otherwise specified, we choose $\rho = 0.5$ in the following experiments.

Next, we compare the performance of the dVB-ADMM with those of the cVB and dSVB. Fig. 8 shows that the dVB-ADMM outperforms the dSVB both in the convergence speed and the accuracy. Using the dSVB, the cost and differences among nodes decrease gradually and smoothly. While, in the dVB-ADMM, the estimate values and their differences among nodes fluctuate within a wide range at the beginning stage of iterations, but become very stable after a few hundreds of iterations. The dSVB gets converged after about 1000 iterations, while the dVB-ADMM only needs about 200 iterations to reach the same accuracy.

C. Evaluation of the Robustness of the Algorithms

In the above experiments, the distribution that draws the local sampled data from is very different for different nodes, but the number of the local sampled data points at each node is assumed to be the same. In this setting, both the dSVB and the dVB-ADMM approaches perform well. However, in many practical

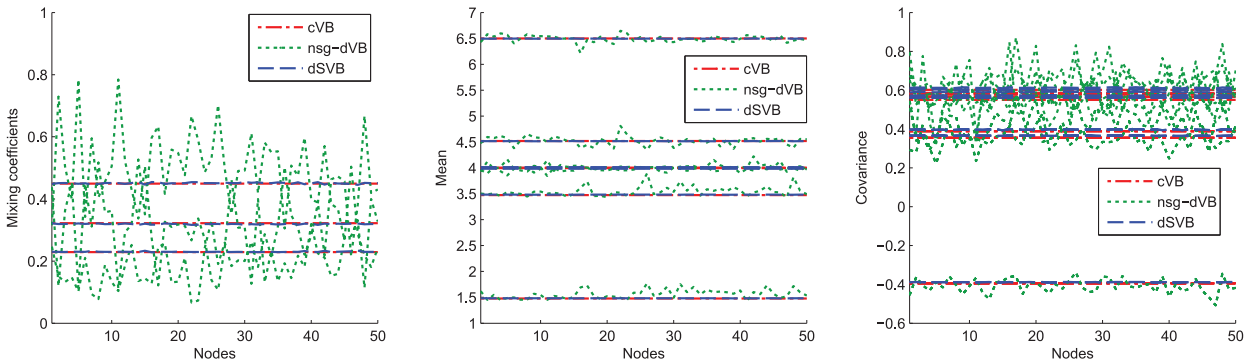

Fig. 5. The values of the estimated mixing coefficients (left), means (middle) and covariances (right) using the cVB, nsg-dVB and dSVB. The vector (means) and matrices (covariance) values are visualized by their entries for a better comparison in 2D coordinates.

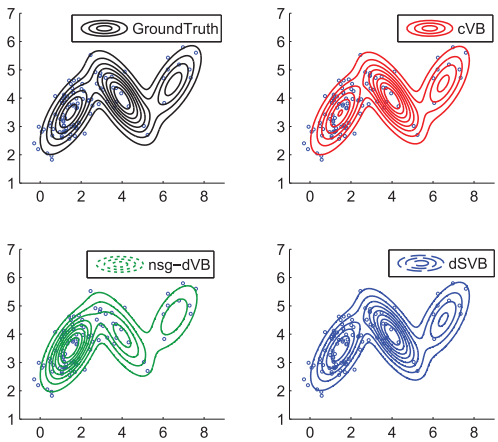

Fig. 6. The contours of the estimated mixture model at a randomly selected node (node 67) using the dSVB and nsg-dVB compared with the cVB's.

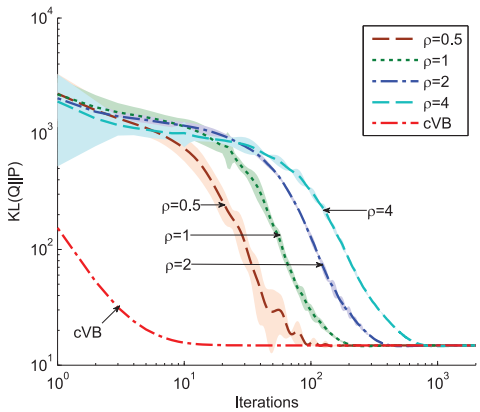

Fig. 7. The evolution of the means and standard deviations of the cost (KL divergence) of the dVB-ADMM algorithm with different penalty parameters ρ, compared with the cVB.

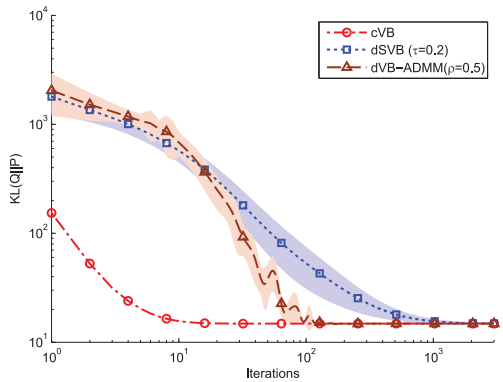

Fig. 8. The evolution of the mean and standard deviation of the cost (KL divergence) of the dVB-ADMM ($\rho = 0.5$), compared with the dSVB ($\tau = 0.2$) and the cVB.

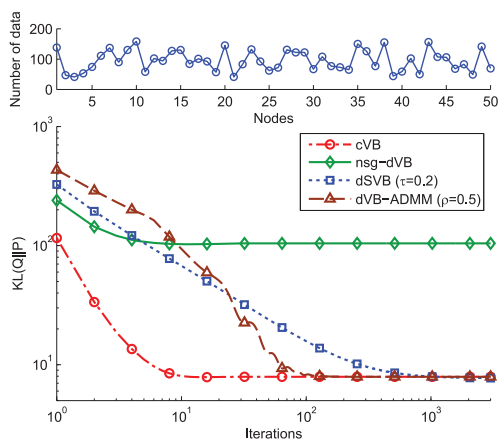

Fig. 9. The number of data at each node (top) and the average performance (bottom) of the dSVB and dVB-ADMM with the imbalanced data, compared with the nsg-dVB and cVB.

cases, the number of the sampled data point at each node could be very different. In order to test the robustness of the proposed algorithms, we evaluate their performance in this case. Furthermore, note that the number of nodes in different networks may be very different either, we will show that our approaches are scalable and can be applied into networks with different sizes.

1) The Case of the Unequal Data Sizes: In this experiment, we consider the case that the observed data sizes among different nodes are unequal. They are randomly selected from $40 \sim 160$, as shown in the top panel of Fig. 9. The distribution parameters are kept the same as those in the above simulations. All data samples are randomly generated from the whole Gaussian mixture model. In this setting, Fig. 9 illustrates that the dSVB and dVB-ADMM can also perform much better than

the nsg-dVB and almost as well as the centralized VB. The unbalancedness of the sample size has no significant impact on the performance. In fact, in the process of deriving the algorithms, we have never made any assumption about the number of the local data, and the natural parameter vector already carries this information. This again explains why the natural parameter vector is a good choice of the message exchanged among nodes.

2) The Case of Different Network Sizes: In the situation with different network sizes, the performance of both algorithms are evaluated here. The density of networks remains unchanged. To ensure this, the communication distance still remains 0.8 and the square, where the nodes are randomly placed, is proportionally zoomed in and out. Other settings are kept unchanged. We test various sizes of networks and three of them $(N = 30, 80, 100)$ are shown in Fig. 9. With the increase of the network size, the total number of iterations needed to get converged increases. It is inevitable because the local data is partial and the full information is distributed to all nodes. Nevertheless, the convergence can still be achieved and the performance is still good.

D. Clustering of Real Data

In this subsection, we examine the proposed algorithms for distributed clustering using GMM on three real-world datasets. We use the accuracy or the misclassification rate to measure the clustering performance. For comparison, except simulating the cVB and nsg-VB, we also simulate the non-cooperation VB algorithm (noncoop-VB), in which each node performs the VB without cooperating with its neighbors.

1) Atmosphere Data: To get an overall evaluation of the atmosphere quality, we can collect air samples distributedly using a WSN. The evaluation task is then taken by utilizing local computation and one-hop communication in the WSN. In this experiment, we use the real atmosphere data provided by [3]. A total of 1600 samples (including 830 clean air and 770 polluted air samples) are used, each with entries, the sulfur dioxide (SO_2), nitrogen dioxide (NO_2) and PM10. We use a WSN with 20 nodes, each with randomly allocated $N = 80$ measurements. The algebraic connectivity of the network is 0.24 and the average degree is 4.8. For this clustering task, the tuning parameters for the two proposed algorithms are set as $\tau = 0.2$ and $\rho = 1$, respectively. Table I shows the average accuracy and the number of misclassification samples for different algorithms over 300 independent Monte Carlo simulations with random initializations. Fig. 11 depicts the clustering results of different algorithms in one trial. From these results, we see that the numbers of the misclassification samples of the dSVB and the dVB-ADMM are much lower than that of the noncoop-VB and the nsg-dVB while very close to that of the cVB.

2) Ionosphere Data: To give an overall analysis of the ionosphere, a radar sensor network can be used to collect the radar returns from the ionosphere and cooperatively figure out which ones are "good" (showing evidence of some type of structure in the ionosphere) and which are "bad". To simulate this scenario, we perform the algorithms on the ionosphere dataset from the UCI learning repository [54]. This radar data was collected from a phased array of sixteen high-frequency antennas in Goose Bay, Labrador. There are 351 observations (including 225 "good" and 126 "bad" radar returns) with 34 continuous attributes. We use the same sensor network as that in the previous experiment. The algorithms' parameters are set as $\tau = 0.2$ and $\rho = 16$. The evaluation is performed by averaging over 300 independent Monte Carlo simulations with random initializations. For each simulation, 340 observations are randomly selected from the whole dataset and uniformly distributed to 20 nodes. As shown in Table II, the dSVB and the dVB-ADMM outperform the noncoop-VB and the nsg-dVB. Surprisingly, the dVB-ADMM can even get higher accuracy than the centralized VB.

3) COIL-20 Data: In many practical image processing applications, in order to obtain multi-aspect rich information, images are often acquired from many different positions in an environment. In this case, a WSN with sensor nodes equipped with tiny cameras can be used. To simulate the above scenario, we use the proposed algorithms for cooperatively solving an object

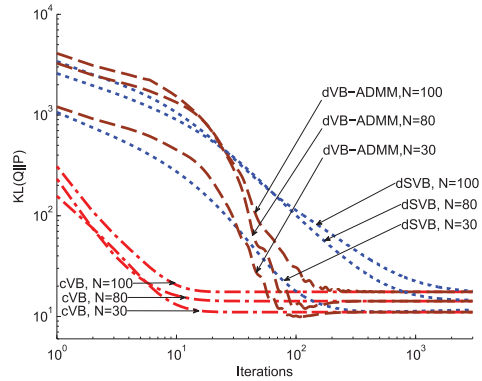

Fig. 10. The average performance of the dSVB and dSV-ADMM with different network sizes ($N = 30, 80, 100$). The tuning parameters are set as $\tau = 0.2$, $\rho = 0.5$ for all cases.

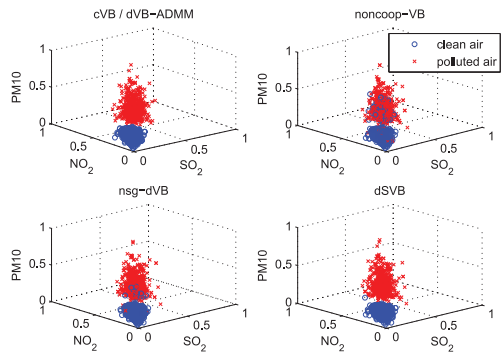

Fig. 11. Clustering results of different algorithms on the atmosphere dataset in one trial.

TABLE I
THE ACCURACY AND AVERAGE NUMBER OF MISCLASSIFICATION SAMPLES OF DIFFERENT ALGORITHMS ON THE ATMOSPHERE DATA

Algorithm	Accuracy (%)	# misclassification samples
cVB	**100.00**	**0.00**
noncoop-VB	89.75	164.06
nsg-dVB	98.99	16.15
dSVB	99.89	1.73
dVB-ADMM	99.99	0.03

TABLE II
THE ACCURACY AND AVERAGE NUMBER OF MISCLASSIFICATION SAMPLES OF DIFFERENT ALGORITHMS ON THE IONOSPHERE DATA

Algorithm	Accuracy (%)	# misclassification samples
cVB	82.08	60.94
noncoop-VB	59.65	137.19
nsg-dVB	64.89	119.36
dSVB	78.25	73.96
dVB-ADMM	**85.59**	**49.00**

Fig. 12. Sample images from the COIL-20 dataset.

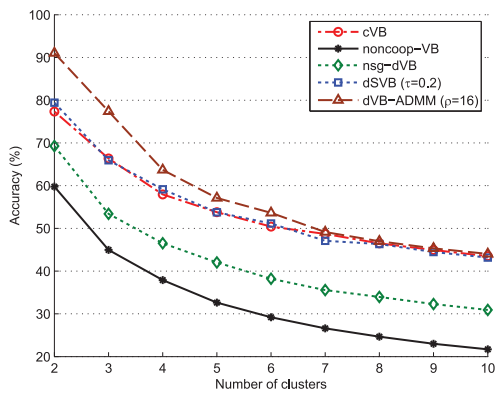

Fig. 13. Accuracy versus the number of clusters on the COIL-20 dataset.

classification problem using the COIL-20 image dataset [55]. It contains 20 objects. The images of each objects were taken 5 degrees apart as the object is rotated on a turntable and each object has 72 images. Some sample images are shown in Fig. 12. The size of each image is 32×32 pixels, with 256 gray levels per pixel. Thus, each image is represented by a 1,024-dimensional vector. To speed up iteration and to avoid the covariance matrix being singular, we apply PCA to reduce the dimension to 52, which keeps about 90 percent information according to the eigenvalues. We use a WSN with 10 nodes. The algebraic connectivity is 0.51 and the average degree per node is 3.0. For each given cluster number K (ranges from 2 to 10), 30 tests are conduced on randomly chosen clusters. For each test, the corresponding images are randomly grouped into 10 subsets and uniformly allocated to the 10 nodes. The tuning parameters are set as $\tau = 0.2$, $\rho = 16$.

The final performance scores for each clustering number K were computed by averaging the scores from 300 independent tests with random initializations. Fig. 13 shows the plots of the clustering performance versus the number of clusters. As we see, our proposed dSVB and dVB-ADMM algorithms perform almost as well as the centralized VB (cVB), and much better than the nsg-dVB and noncoop-VB. In some cases (when $K < 7$), the dVB-ADMM can achieve better clustering results than the centralized VB. The reason is that our distributed algorithms can be seen as the sparse and incremental variants of the VB algorithm [56], which might be less sensitive to initialization and on average they can find better local optimum more often than their centralized counterparts with random initializations. Thus, distributed algorithm with multiple parts of data might have more advantage to avoid obtaining a worse local optimum than centralized algorithm with the whole data. Similar phenomena have also been observed in other distributed clustering algorithms [3], [34].

VI. CONCLUSION

In this paper, two distributed variational algorithms are proposed for general Bayesian inference in a networked system. The variational problem is recast as an optimization problem in the natural parameter space. The variational free energy is decomposed as a set of local lower bounds. Each node runs the VBE step by maximizing the local lower bound with respect to the local latent variables. The VBM step is solved in two distributed schemes. In the dSVB scheme, a stochastic natural gradient is adopted for gradually improving the estimates with the local data and a combination step is used for the cooperation with neighbors. In the dVB-ADMM scheme, the variational optimization is redefined as a constrained minimization problem with a modified objective function. The ADMM technique is then used to solve this constrained optimization. In both schemes, each node only needs to exchange low dimensional natural parameters with its neighbors.

An application of the distributed inference/estimation of a Bayesian Gaussian mixture model is then presented, to evaluate the effectiveness of the proposed algorithms. The numerical simulations on both synthetic and real-world datasets demonstrate that the proposed algorithms outperform the non-stochastic-gradient based distributed VB (nsg-dVB) and non-cooperation VB (noncoop-VB) algorithms and they both can perform almost as well as the centralized VB (cVB). Both of the algorithms exhibit resilience to data imbalance and is applicable to networks with different sizes. The dVB-ADMM approach converges faster than the dSVB, but even the dSVB algorithm performs much better than the competing algorithms in the literature.

Further development includes expanding results to the non-conjugate exponential family and developing distributed algorithms for dynamic Bayesian networks.

APPENDIX

A. Distributed VB for Gaussian Mixtures

The hyperparameters for (44a), (44b) and (44c) are

$$r_{ijk} = \rho_{ijk} / \sum_{k=1}^{K} \rho_{ijk}, \boldsymbol{m}_{ik} = \frac{1}{\beta_{ik}}(\beta_0 \boldsymbol{\mu}_0 + R_{ik}\bar{\boldsymbol{x}}_{ik}),$$
$$\alpha_{ik} = \alpha_0 + R_{ik}, \beta_{ik} = \beta_0 + R_{ik}, \nu_{ik} = \nu_0 + R_{ik},$$
$$\boldsymbol{W}_{ik}^{-1} = \boldsymbol{W}_0^{-1} + R_{ik}\boldsymbol{S}_{ik} + \frac{\beta_0 R_{ik}}{\beta_0 + R_{ik}}(\bar{\boldsymbol{x}}_{ik} - \boldsymbol{\mu}_0)(\bar{\boldsymbol{x}}_{ik} - \boldsymbol{\mu}_0)^T,$$

where the sufficient statistics are given by

$$\begin{aligned}\ln \rho_{ijk} =& \mathbb{E}[\ln \pi_{ik}] + \frac{1}{2}\mathbb{E}[\ln |\boldsymbol{\Lambda}_{ik}|] - \frac{D}{2}\ln(2\pi)\\ &- \frac{1}{2}\ln \mathbb{E}_{\boldsymbol{\mu}_{ik},\boldsymbol{\Lambda}_{ik}}[(\boldsymbol{x}_{ij}-\boldsymbol{\mu}_{ik})^T\boldsymbol{\Lambda}_{ik}(\boldsymbol{x}_{ij}-\boldsymbol{\mu}_{ik})],\\ R_{ik} =& N\sum_{j=1}^{N_i} r_{ijk}, \bar{\boldsymbol{x}}_{ik} = \frac{N}{R_{ik}}\sum_{j=1}^{N_i} r_{ijk}\boldsymbol{x}_{ij},\\ \boldsymbol{S}_{ik} =& \frac{N}{R_{ik}}\sum_{j=1}^{N_i} r_{ijk}(\boldsymbol{x}_{ij}-\bar{\boldsymbol{x}}_{ik})(\boldsymbol{x}_{ij}-\bar{\boldsymbol{x}}_{ik})^T.\end{aligned}$$

The expectations in the above formulae are derived using (10a),

$$\begin{aligned}&\mathbb{E}[\ln \pi_{ik}] = \psi(\alpha_{ik}) - \psi\left(\sum_{i=1}^{K}\alpha_{ik}\right),\\ &\mathbb{E}[\ln |\boldsymbol{\Lambda}_{ik}|] = \sum_{j=1}^{D}\psi\left(\frac{\nu+1-j}{2}\right) + D\ln 2 + \ln|\boldsymbol{W}_{ik}|,\\ &\mathbb{E}_{\boldsymbol{\mu}_{ik},\boldsymbol{\Lambda}_{ik}}[(\boldsymbol{x}_{ij}-\boldsymbol{\mu}_{ik})^T\boldsymbol{\Lambda}_{ik}(\boldsymbol{x}_{ij}-\boldsymbol{\mu}_{ik})] = D\beta_{ik}^{-1}\\ &\qquad + \nu_{ik}(\boldsymbol{x}_{ij}-\boldsymbol{m}_{ik})^T\boldsymbol{W}_{ik}(\boldsymbol{x}_{ij}-\boldsymbol{m}_{ik}),\end{aligned}$$

where $\psi(\cdot)$ is digamma function.

B. KL Divergence for Exponential Family Distributions

We omit the subscript i, the node index, for notational simplicity. The joint variational distribution of model parameters in the synthetic example in Section V-A is

$$Q(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda}|\boldsymbol{\phi}_\theta) = q(\boldsymbol{\pi})\prod_{k=1}^{K} q(\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k),$$

and the corresponding ground truth posterior is

$$P(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda}|\hat{\boldsymbol{\phi}}_\theta) = P(\boldsymbol{\pi})\prod_{k=1}^{K} P(\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k),$$

where $q(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha})$, $P(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\hat{\boldsymbol{\alpha}})$ are Dirichlet distributions, and $q(\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k) = \mathcal{NW}(\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k|\boldsymbol{\phi}_k)$, $P(\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k) = \mathcal{NW}(\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k|\hat{\boldsymbol{\phi}}_k)$ are normal-Wishart distributions. Hence, the KL divergence between two distributions becomes

$$\begin{aligned}&d(\boldsymbol{\phi}_\theta,\hat{\boldsymbol{\phi}}_\theta)\\ &= \mathrm{KL}\big(Q(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda}|\boldsymbol{\phi}_\theta)||P(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda}|\hat{\boldsymbol{\phi}}_\theta)\big)\\ &= \mathrm{KL}(\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha})||\mathrm{Dir}(\boldsymbol{\pi}|\hat{\boldsymbol{\alpha}}))\\ &\quad + \sum_{k=1}^{K}\mathrm{KL}(\mathcal{NW}(\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k|\boldsymbol{\phi}_k)||\mathcal{NW}(\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k|\hat{\boldsymbol{\phi}}_k).\end{aligned}$$

The KL divergences in the above equation can be computed as follows.

1) Dirichlet Distribution: The KL divergence between two Dirichlet distributions is

$$\begin{aligned}&\mathrm{KL}(\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha})||\mathrm{Dir}(\boldsymbol{\pi}|\hat{\boldsymbol{\alpha}}))\\ &\qquad = \sum_{k=1}^{K}(\alpha_k-\hat{\alpha}_k)\mathbb{E}_{\boldsymbol{\alpha}}[\ln\pi_k] - \ln B(\boldsymbol{\alpha}) + \ln B(\hat{\boldsymbol{\alpha}}),\end{aligned}$$

where $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_K]^T$, $\boldsymbol{\pi} = [\boldsymbol{\pi}_1,\ldots,\boldsymbol{\pi}_K]^T$ and $B(\cdot)$ is the multinomial Beta function, which can be expressed as $B(\boldsymbol{\alpha}) = \prod_{k=1}^{K}\Gamma(\boldsymbol{\alpha}_k)/\Gamma(\sum_{k=1}^{K}\boldsymbol{\alpha}_k)$.

2) Normal-Wishart Distribution: The KL divergence between two normal-Wishart distributions is

$$\begin{aligned}&\mathrm{KL}(\mathcal{NW}(\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k|\boldsymbol{\phi}_k)||\mathcal{NW}(\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k|\hat{\boldsymbol{\phi}}_k)\\ &\quad = \mathrm{Tr}\left([\boldsymbol{\phi}_k-\hat{\boldsymbol{\phi}}_k]^T\begin{bmatrix}\mathbb{E}_{\boldsymbol{\phi}_k}[\ln|\boldsymbol{\Lambda}_k|]\\ \mathbb{E}_{\boldsymbol{\phi}_k}[\boldsymbol{\Lambda}_k]\\ \mathbb{E}_{\boldsymbol{\phi}_k}[\boldsymbol{\Lambda}_k\boldsymbol{\mu}_k]\\ \mathbb{E}_{\boldsymbol{\phi}_k}[\boldsymbol{\mu}_k^T\boldsymbol{\Lambda}_k\boldsymbol{\mu}_k]\end{bmatrix}\right) - A(\boldsymbol{\phi}_k) + A(\hat{\boldsymbol{\phi}}_k),\end{aligned}$$

where the natural parameter vector for normal-Wishart distribution is

$$\boldsymbol{\phi}_k = [\frac{\nu_k-D}{2}, -\frac{1}{2}\boldsymbol{W}_k^{-1} - \frac{\beta_k}{2}\boldsymbol{m}_k\boldsymbol{m}_k^T, \beta_k\boldsymbol{m}_k, -\frac{1}{2}\beta_k]^T,$$

and the sufficient statistics that have not been given in Appendix A include

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\phi}_k}[\boldsymbol{\Lambda}_k] &= \nu_k\boldsymbol{W}_k,\\ \mathbb{E}_{\boldsymbol{\phi}_k}[\boldsymbol{\Lambda}_k\boldsymbol{\mu}_k] &= \nu_k\boldsymbol{W}_k\boldsymbol{m}_k,\\ \mathbb{E}_{\boldsymbol{\phi}_k}[\boldsymbol{\mu}_k^T\boldsymbol{\Lambda}_k\boldsymbol{\mu}_k] &= D\beta_k^{-1} + \nu_k\boldsymbol{m}_k^T\boldsymbol{W}_k\boldsymbol{m}_k.\end{aligned}$$

In addition, the partition function $A(\cdot)$ for this distribution is

$$\begin{aligned}A(\boldsymbol{\phi}_k) = &-\frac{D}{2}\ln|\beta_k| + \frac{\nu_k}{2}\ln|\boldsymbol{W}_k| + \frac{\nu_k D}{2}\ln 2\\ &+\sum_{j=1}^{D}\ln\Gamma(\frac{\nu_k+1-j}{2}).\end{aligned}$$

Hence, the KL divergence between P and Q can be computed in terms of a closed-form expression using the above equations.

REFERENCES

[1] M. Taj and A. Cavallaro, "Distributed and decentralized multicamera tracking," *IEEE Signal Process. Mag.*, vol. 28, no. 3, pp. 46–58, 2011.

[2] 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, 2014.

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

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

[5] C. Li and H. Wang, "Distributed frequency estimation over sensor network," *IEEE Sensors J.*, vol. 15, no. 7, pp. 3973–3983, 2015.

[6] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[7] R. Olfati-Saber and J. S. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," in *Proc. IEEE Conf. Decision Control/Eur. Control Conf. (CDC-ECC '05)*, 2005, pp. 6698–6703.

[8] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1035–1048, 2010.

[9] N. Takahashi, I. Yamada, and A. H. Sayed, "Diffusion least-mean squares with adaptive combiners: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 58, no. 9, pp. 4795–4810, Sep. 2010.

[10] 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, 2012.

[11] 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, 2013.

[12] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[13] 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, 2010.
[14] 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, 2002.
[15] J. Predd, S. Kulkarni, and H. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 56–69, 2006.
[16] M. Paskin, C. Guestrin, and J. McFadden, "A robust architecture for distributed inference in sensor networks," in *Proc. 4th Int. Symp. Inf. Process. Sensor Netw., Ser. IPSN '05*, 2005, pp. 55–62.
[17] H. Dai, Y. Zhang, and J. Liu, "Structured variational methods for distributed inference in networked systems: Design and analysis," *IEEE Trans. Signal Process.*, vol. 61, no. 15, pp. 3827–3839, 2013.
[18] E. B. Sudderth, A. T. Ihler, M. Isard, W. T. Freeman, and A. S. Willsky, "Nonparametric belief propagation," *Commun. ACM*, vol. 53, no. 10, pp. 95–103, 2010.
[19] A. T. Ihler, J. W. Fisher, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 809–819, 2005.
[20] D. J. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural Comput.*, vol. 4, no. 3, pp. 448–472, 1992.
[21] G. F. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," *Mach. Learn.*, vol. 9, no. 4, pp. 309–347, 1992.
[22] V. Šmídl and A. Quinn, *The Variational Bayes Method in Signal Processing*. New York, NY, USA: Springer Science & Business Media, 2006.
[23] H. Attias, "Inferring parameters and structure of latent variable models by variational Bayes," in *Proc. 15th Conf. Uncertainty Artif. Intell.*, 1999, pp. 21–30.
[24] J. Wolfe, A. Haghighi, and D. Klein, "Fully distributed EM for very large datasets," in *Proc. ACM 25th Int. Conf. Mach. Learn.*, 2008, pp. 1184–1191.
[25] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 1303–1347, 2013.
[26] K. Zhai, J. Boyd-Graber, N. Asadi, and M. L. Alkhouja, "MR. LDA: A flexible large scale topic modeling package using variational inference in MapReduce," in *Proc. ACM 21st Int. Conf. World Wide Web, ser. WWW '12*, New York, NY, USA, 2012, pp. 879–888.
[27] B. Safarinejadian, M. B. Menhaj, and M. Karrari, "Distributed variational Bayesian algorithms for Gaussian mixtures in sensor networks," *Signal Process.*, vol. 90, no. 4, pp. 1197–1208, 2010.
[28] S. Mukherjee and H. Kargupta, "Distributed probabilistic inferencing in sensor networks using variational approximation," *J. Parallel Distrib. Comput.*, vol. 68, no. 1, pp. 78–92, 2008.
[29] B. Safarinejadian and M. B. Menhaj, "Distributed density estimation in sensor networks based on variational approximations," *Int. J. Syst. Sci.*, vol. 42, no. 9, pp. 1445–1457, 2011.
[30] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, vol. 53, no. 1, pp. 65–78, 2004.
[31] S.-I. Amari, "Natural gradient works efficiently in learning," *Neural Comput.*, vol. 10, no. 2, pp. 251–276, 1998.
[32] 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, 2011.
[33] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *J. Mach. Learn. Res.*, vol. 11, pp. 1663–1707, 2010.
[34] P. A. Forero, A. Cano, and G. B. Giannakis, "Distributed clustering using wireless sensor networks," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 4, pp. 707–724, 2011.
[35] M. J. Beal, "Variational algorithms for approximate Bayesian inference," Ph.D. dissertation, Univ. College London, London, U.K., 2003.
[36] L. K. Saul, T. Jaakkola, and M. I. Jordan, "Mean field theory for sigmoid belief networks," *J. Artif. Intell. Res.*, vol. 4, no. 61, p. 76, 1996.
[37] L. D. Brown, *Fundamentals of Statistical Exponential Families: With Applications in Statistical Decision Theory*. Hayward, CA, USA: Inst. Math. Statist., 1986.
[38] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Found. Trends Mach. Learn.*, vol. 1, no. 1-2, pp. 1–305, 2008.
[39] J. M. Winn and C. M. Bishop, "Variational message passing," in *J. Mach. Learn. Res.*, 2005, vol. 6, pp. 661–694.
[40] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc. Series B (Methodol.)*, pp. 1–38, 1977.
[41] D. Gu, "Distributed EM algorithm for Gaussian mixtures in sensor networks," *IEEE Trans. Neural Netw.*, vol. 19, no. 7, pp. 1154–1166, 2008.
[42] Y. Weng, W. Xiao, and L. Xie, "Diffusion-based EM algorithm for distributed estimation of Gaussian mixtures in wireless sensor networks," *Sensors*, vol. 11, no. 6, pp. 6297–6316, 2011.
[43] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, pp. 400–407, 1951.
[44] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 988–1001, 2003.
[45] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "Distributed Kalman filtering based on consensus strategies," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 4, pp. 622–633, 2008.
[46] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Process.*, vol. 55, no. 8, pp. 4064–4077, 2007.
[47] S.-I Amari, *Differential-Geometrical Methods in Statistics*. New York, NY, USA: Springer-Verlag, 1985.
[48] A. Honkela, M. Tornio, T. Raiko, and J. Karhunen, "Natural conjugate gradient in variational inference," in *Neural Information Processing*. New York, NY, USA: Springer, 2008, pp. 305–314.
[49] M.-A. Sato, "Online model selection based on the variational Bayes," *Neural Comput.*, vol. 13, no. 7, pp. 1649–1681, 2001.
[50] S.-I. Amari, "Differential geometry of curved exponential families-curvatures and information loss," *Ann. Statist.*, pp. 357–385, 1982.
[51] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh, "Clustering with Bregman divergences," *J. Mach. Learn. Res.*, vol. 6, pp. 1705–1749, 2005.
[52] Y. Nesterov, *Introductory Lectures on Convex Optimization*. New York, NY, USA: Springer Science & Business Media, 2004, vol. 87.
[53] T. Erseghe, D. Zennaro, E. Dall'Anese, and L. Vangelista, "Fast consensus by the alternating direction multipliers method," *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5523–5537, 2011.
[54] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker, "Classification of radar returns from the ionosphere using neural networks," *Johns Hopkins APL Tech. Dig*, vol. 10, pp. 262–266, 1989.
[55] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia Object Image Library (COIL-20)", Tech. Rep. CUCS-005-96, 1996 [Online]. Available: http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php, accessed Dec. 14, 2015.
[56] R. M. Neal and G. E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in *Learning in Graphical Models*. New York, NY, USA: Springer, 1998, pp. 355–368.

Junhao Hua received the B.S. degree both in computer science and automation from Zhejiang University of Technology, Hangzhou, China, in 2013.

Currently, he is pursuing the Ph.D. degree in the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China. His current research interests include statistical signal processing, Bayesian learning and wireless sensor network.

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.

Currently, he is a Professor with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China. His current research interests include statistical signal processing and wireless sensor network.