Online EM-based distributed estimation in sensor networks with faulty nodes

Pere Giménez-Febrer*, Alba Pagès-Zamora* and Roberto López-Valcarce†
*SPCOM Group, Universitat Politècnica de Catalunya-Barcelona Tech, Spain
†GPSC, Universidade de Vigo, Spain

***Abstract*—This paper focuses on the problem of the distributed estimation of a parameter vector based on noisy observations regularly acquired by the nodes of a wireless sensor network and assuming that some of the nodes have faulty sensors. We propose two online schemes, both centralized and distributed, based on the Expectation-Maximization (EM) algorithm. These algorithms are able to identify and disregard the faulty nodes, and provide a refined estimate of the parameters each time instant after a new set of observations is acquired. Simulation results demonstrate that the centralized versions of the proposed online algorithms attain the same estimation error as the centralized batch EM, whereas the distributed versions come very close to matching the batch EM.**

***Index terms*— online expectation-maximization algorithms, distributed estimation, sensor networks.**

I. Introduction

WIRELESS sensor networks (WSNs) have become increasingly popular in the recent years thanks to their potential to be used in a wide range of applications and provide robust solutions at a low cost. One common application of WSNs is the estimation of parameters from observations acquired by the sensors, which can be performed in a distributed manner in order to reduce the overall power consumption and increase the robustness of the system against node failures.

Besides a critical failure rendering the node unusable, a malfunctioning sensor can also impact the performance of the WSN since it corrupts the observations made at the node. Thus, it is crucial to identify which nodes have faulty sensors and discard their data. Sensor failures can be modelled as a hidden variable that determines whether the node is sensing the desired signal plus noise, or just noise [1]. One algorithm that encompasses the estimation of a parameter and the estimation of hidden variables is the Expectation-Maximization (EM) algorithm. The EM algorithm iteratively maximizes the likelihood of the data given the parameters, while at the same time estimates the value of the hidden variables that best fit the observed data. Distributed implementations of the EM algorithm are especially well suited for sensor networks and have been proposed in the literature. In [2], the summary statistics required to calculate the parameters are cycled through the network and updated at each node with its local information. Other consensus-based approaches have been proposed in [3], [4], [5], and [6] for the Gaussian mixture problem. In these works, the nodes sought to identify and reach a network-wide consensus on the parameters of a Gaussian Mixture by exchanging information only with their close neighbors. Similarly, [1] proposes a distributed diffusion-based EM algorithm for sensor networks with faulty nodes.

The EM algorithm operates in batch mode by processing the complete dataset at each iteration to update the parameter estimates, which can be computationally intensive for large datasets. Moreover, when the dataset becomes slowly available over time, such as when processing a data stream, operating in batch mode requires waiting until all the data have been collected before running the algorithm. In order to reduce the latency and computational load, online EM algorithms process a single data item, or a subset, and provide an updated estimate of the parameters at each iteration. For instance, the incremental EM proposed in [7] selects at each iteration one observation at random from the complete dataset and updates the estimates. However, even though it processes one item at a time, this algorithm still requires the complete data to be available. In [8], an online EM was proposed that replaces the maximization step by a gradient step evaluated at the newly acquired observation. Other online EM algorithms, such as the ones proposed in [9] and [10], perform a stochastic approximation by combining noisy observations of the log-likelihood function via a Robbins-Monro procedure. While all these works are centralized algorithms, a distributed version of [10] was proposed in [11] for asynchronous WSNs.

Existing online EM algorithms in the literature build on the assumption that observations are independent, which cannot be assumed in our case of WSNs with faulty nodes since the measurements taken by the same node depend on a single hidden variable. In this paper, we propose two online EM algorithms that take into account the dependence between observations, and their distributed implementations for sensor networks with faulty nodes.

The paper is organized as follows. Section I details the signal model for the WSN. Section II introduces the centralized Batch EM algorithm. In Section III, we review the classical online EM algorithms and present two new algorithms. Section IV introduces the distributed versions of the two algorithms and, finally, Section V includes the simulation results.

II. Signal model

Consider a sensor network formed by N nodes collecting observations at each iteration time $t \geq 1$ denoted by

$$y_i(t) = a_i x + w_i(t), \qquad i = 1, \ldots, N, \tag{1}$$

where $y_i(t) \in \mathbb{R}$, x is the parameter of interest, $\{a_i, \forall i\} = \{0, 1\}$ are independent identically distributed (iid) Bernoulli

This work is supported by the "Ministerio de Economía y Competitividad" of the Spanish Government and ERDF funds (TEC2013-41315-R DISNET, TEC2013-47020-C2-1-R COMPASS and TEC2015-69648-REDC Red COMONSENS), the Catalan Government (2014 SGR 60 AGAUR) and the Galician Government (AtlantTIC, GRC2013/009, R2014/037).

978-0-9928-6265-7/16/$31.00 ©2016 IEEE

random variables with probability $p \triangleq \Pr\{a_i = 1\}$, and $\{w_i(t), \forall i \forall t\}$ are iid zero-mean Gaussian with variance σ^2 and independent of $\{a_i, \forall i\}$. Thus, a value of $a_i = 1$ indicates that node i is sensing the parameter x corrupted by noise, whereas a value of $a_i = 0$ indicates a sensor failure such that node i measures only noise. Rewriting (1) for the entire network in vector form yields

$$\boldsymbol{y}(t) = \boldsymbol{a}x \otimes \mathbf{1}_t + \boldsymbol{\omega}(t),$$

where $\otimes$ denotes the Kronecker product, $\mathbf{1}_t$ is an all-ones column vector with t components,

$$\begin{aligned}\boldsymbol{y}(t) &= [\boldsymbol{y}_1^T(t), \ldots, \boldsymbol{y}_N^T(t)]^T,\\ \boldsymbol{a} &= [a_1, \ldots, a_N],\\ \boldsymbol{\omega}(t) &= [\boldsymbol{\omega}_1^T(t), \ldots, \boldsymbol{\omega}_N^T(t)]^T,\end{aligned}$$

and $\{\forall i, \forall t\}$ we have $\boldsymbol{y}_i(t) = [y_i(1), \ldots, y_i(t)]^T$, and $\boldsymbol{\omega}_i(t) = [w_i(1), \ldots, w_i(t)]^T$. Note that observations from different nodes are iid but those from the same node are dependent. The probability density function of $\boldsymbol{y}(t)$ is

$$f(\boldsymbol{y}(t) \,|\, \boldsymbol{\theta}) = \prod_{i=1}^{N} \Bigg[\frac{p}{(2\pi\sigma^2)^{\frac{t}{2}}} \prod_{l=1}^{t} \exp\left(-\frac{(y_i(l) - x)^2}{2\sigma^2}\right) + \frac{1-p}{(2\pi\sigma^2)^{\frac{t}{2}}} \prod_{l=1}^{t} \exp\left(-\frac{y_i^2(l)}{2\sigma^2}\right) \Bigg],$$

where $\boldsymbol{\theta} = [x, \sigma^2]^T$. The parameter to be estimated is x, whereas σ^2 is regarded as an unknown nuisance parameter; the a priori probability p is assumed known throughout. Since closed-form maximization of $f(\boldsymbol{y}(t) \,|\, \boldsymbol{\theta})$ is not possible, we resort to the EM algorithm to determine which nodes are non-faulty and estimate x.

III. Centralized Batch EM

In this section we introduce the standard centralized EM algorithm [12], to which we will refer as Batch EM, assuming that each node acquires observations until iteration $t = L$ observations and the NL observations are available at a central location. Provided that there are enough observations, this iterative algorithm obtains the maximum likelihood estimate of the parameters of interest $\boldsymbol{\theta}$ in the presence of unobserved data. We regard vector $\boldsymbol{y}(L) = [\boldsymbol{y}_1(L), \ldots, \boldsymbol{y}_N(L)]^T$ as the incomplete observation, and $\{\boldsymbol{y}(L), \boldsymbol{a}\}$ as the complete one. The Batch EM algorithm calculates the estimates of $\boldsymbol{\theta}$ by successively executing the following two steps at each iteration k:

1) *E-step:* given an estimate $\hat{\boldsymbol{\theta}}_k = [\hat{x}_k, \hat{\sigma}_k^2]^T$, compute the conditional expectation

$$\begin{aligned}Q(\tilde{\boldsymbol{\theta}}\,;\, \hat{\boldsymbol{\theta}}_k) &= \mathbb{E}_{\boldsymbol{a}}\{\log f(\boldsymbol{y}(L), \boldsymbol{a} \,|\, \tilde{\boldsymbol{\theta}}) \,|\, \hat{\boldsymbol{\theta}}_k, \boldsymbol{y}(L)\}\\ &= \sum_{i=1}^{N} \mathbb{E}_{a_i}\{\log f(\boldsymbol{y}_i(L), a_i \,|\, \tilde{\boldsymbol{\theta}}) \,|\, \hat{\boldsymbol{\theta}}_k, \boldsymbol{y}_i(L)\},\end{aligned} \tag{2}$$

where $\tilde{\boldsymbol{\theta}} = [\tilde{x}, \tilde{\sigma}^2]^T$ denotes a trial value of $\boldsymbol{\theta}$.

2) *M-step:* obtain the estimate for the next iteration as

$$\hat{\boldsymbol{\theta}}_{k+1} = \arg\max_{\tilde{\boldsymbol{\theta}}} Q(\tilde{\boldsymbol{\theta}}\,;\, \hat{\boldsymbol{\theta}}_k).$$

The E-step calculates the a posteriori conditional probability $\hat{a}_i(k) \triangleq \Pr\left\{a_i = 1 \,|\, \hat{\boldsymbol{\theta}}_k, \boldsymbol{y}_i(L)\right\}$ for each i as

$$\hat{a}_i(k) = \frac{p \prod_{l=1}^{L} \exp\left(-\frac{(y_i(l) - \hat{x}_k)^2}{2\hat{\sigma}_k^2}\right)}{p \prod_{l=1}^{L} \exp\left(-\frac{(y_i(l) - \hat{x}_k)^2}{2\hat{\sigma}_k^2}\right) + (1-p) \prod_{l=1}^{L} \exp\left(-\frac{y_i^2(l)}{2\hat{\sigma}_k^2}\right)}.$$

After $\{\hat{a}_i(k)\ \forall i\}$ are calculated, (2) becomes

$$\begin{aligned}Q(\tilde{\boldsymbol{\theta}}\,;\, \hat{\boldsymbol{\theta}}_k) = &-\frac{NL}{2}\log 2\pi\tilde{\sigma}^2 - \frac{1}{2\tilde{\sigma}^2}\sum_{i=1}^{N}\sum_{l=1}^{L}\hat{a}_i(k)(y_i(l) - \tilde{x})^2\\ &- \frac{1}{2\tilde{\sigma}^2}\sum_{i=1}^{N}\sum_{l=1}^{L}\hat{a}_i(k)y_i^2(l).\end{aligned} \tag{3}$$

The M-step then maximizes (3) with respect to $\boldsymbol{\theta}$ and yields the following estimates for iteration $k+1$:

$$\begin{aligned}\hat{x}_{k+1} &= \frac{\sum_{i=1}^{N}\hat{a}_i(k)\sum_{l=1}^{L}y_i(l)}{L\sum_{i=1}^{N}\hat{a}_i(k)},\\ \hat{\sigma}_{k+1}^2 &= \frac{\sum_{i=1}^{N}\sum_{l=1}^{L}(y_i^2(l) - 2\hat{a}_i(k)y_i(l)\hat{x}_{k+1} + \hat{a}_i(k)\hat{x}_{k+1}^2)}{NL}.\end{aligned}$$

IV. Centralized Online EM

In a centralized online setting, the nodes send their newly acquired observations at iteration t to a central location where they are immediately processed instead of waiting until all L observations have been collected. In this section we propose two online versions of the centralized Batch EM algorithm which do not require the complete dataset to be available at each iteration, and review the classical online EM algorithms in the literature.

A. Recursive online EM

The recursive online EM (ROEM) algorithm we propose takes into account the dependence between the observations, and recalculates the conditional expectation for all the observations every time a new set of N observations is received. Similar to Batch EM, ROEM executes the E-step and M-Step, with the difference that N new observations are added at each iteration instead of waiting until all the observations are collected. The online steps are

1) *E-step:* given an estimate $\hat{\boldsymbol{\theta}}_t = [\hat{x}_t, \hat{\sigma}_t^2]^T$ at iteration t, compute the conditional expectation

$$Q_t(\tilde{\boldsymbol{\theta}}\,;\, \hat{\boldsymbol{\theta}}_t) = \mathbb{E}_{\boldsymbol{a}}\{\log f(\boldsymbol{y}(t), \boldsymbol{a} \,|\, \tilde{\boldsymbol{\theta}}) \,|\, \hat{\boldsymbol{\theta}}_t, \boldsymbol{y}(t)\}. \tag{4}$$

2) *M-step:* obtain the estimate for the next iteration as

$$\hat{\boldsymbol{\theta}}_{t+1} = \arg\max_{\tilde{\boldsymbol{\theta}}} Q_t(\tilde{\boldsymbol{\theta}}\,;\, \hat{\boldsymbol{\theta}}_t).$$

The E-step calculates the a posteriori conditional probability $\hat{a}_i(t) \triangleq \Pr\left\{a_i = 1 \,|\, \hat{\boldsymbol{\theta}}_t, \boldsymbol{y}_i(t)\right\}$ for each i as

$$\hat{a}_i(t) = \frac{p \prod_{l=1}^{t} \exp\left(-\frac{(y_i(l) - \hat{x}_t)^2}{2\hat{\sigma}_t^2}\right)}{p \prod_{l=1}^{t} \exp\left(-\frac{(y_i(l) - \hat{x}_t)^2}{2\hat{\sigma}_t^2}\right) + (1-p) \prod_{l=1}^{t} \exp\left(-\frac{y_i^2(l)}{2\hat{\sigma}_t^2}\right)}. \tag{5}$$

Then, the M-step gives the estimation of the parameters as

$$\hat{x}_{t+1} = \frac{\sum_{i=1}^{N} \hat{a}_i(t) \sum_{l=1}^{t} y_i(l)}{t \sum_{i=1}^{N} \hat{a}_i(t)}, \tag{6}$$

$$\hat{\sigma}^2_{t+1} = \frac{\sum_{i=1}^{N} \sum_{l=1}^{t} (y_i^2(l) - 2\hat{a}_i(t) y_i(l) \hat{x}_{t+1} + \hat{a}_i(t) \hat{x}^2_{t+1})}{Nt}. \tag{7}$$

The cost of computing (5), (6) and (7) can be easily reduced by substituting the summations by cumulative intermediate variables so that each observation is processed only once. Thus, let us define the online variables

$$s_1(t) = t \sum_{i=1}^{N} \hat{a}_i(t), \tag{8}$$

$$s_2(t) = \sum_{i=1}^{N} \hat{a}_i(t) d_{1,i}(t), \tag{9}$$

$$s_3(t) = \sum_{i=1}^{N} d_{2,i}(t), \tag{10}$$

where $d_{1,i}(t) = d_{1,i}(t-1) + y_i(t)$ and $d_{2,i}(t) = d_{2,i}(t-1) + y_i^2(t)$. Then, we can rewrite (5)-(7) in terms of the online variables as

$$\hat{a}_i(t) = \frac{1}{1 + \frac{1-p}{p} \exp\left(\frac{-2d_{1,i}(t)\hat{x}_t + t\hat{x}_t}{2\hat{\sigma}_t^2}\right)},$$

$$\hat{x}_{t+1} = \frac{s_2(t)}{s_1(t)},$$

$$\hat{\sigma}^2_{t+1} = \frac{s_3(t) - 2s_2(t)\hat{x}_{t+1} + s_1(t)\hat{x}^2_{t+1}}{Nt}.$$

Indeed, one can see that at each iteration t the ROEM algorithm executes the two steps of the Batch EM algorithm with Nt measurements but using as initial estimate $\hat{\boldsymbol{\theta}}_t$ the one obtained in the previous iteration, when only $N(t-1)$ measurements were available, and so on.

B. Online incremental Newton

An alternative formulation of the Batch EM algorithm which substitutes the M-step by a Newton update was proposed by Lange in [13]. By applying this approach to the ROEM algorithm in Section II-A, we readily obtain the online incremental Newton (OIN) algorithm. Thus, when a new set of observations are received, the OIN algorithm performs the E-step in (4) embedded in a gradient step. The estimates calculated at iteration t for the next iteration are obtained as

$$\hat{\boldsymbol{\theta}}_{t+1} = \hat{\boldsymbol{\theta}}_t + \gamma_t \boldsymbol{I}^{-1}(\hat{\boldsymbol{\theta}}_t) \mathbb{E}_{\boldsymbol{a}} \left\{ \nabla_{\hat{\boldsymbol{\theta}}_t} \log f(\boldsymbol{y}(t), \boldsymbol{a} \,|\, \tilde{\boldsymbol{\theta}}) \,|\, \boldsymbol{y}(t), \hat{\boldsymbol{\theta}}_t \right\},$$

where γ_t is a diminishing step size, $\nabla_{\hat{\boldsymbol{\theta}}_t} \log f(\boldsymbol{y}(t), \boldsymbol{a} \,|\, \tilde{\boldsymbol{\theta}})$ denotes the gradient with respect to $\tilde{\boldsymbol{\theta}}$ evaluated at $\hat{\boldsymbol{\theta}}_t$, and $\boldsymbol{I}^{-1}(\hat{\boldsymbol{\theta}}_t)$ is the inverse of Fisher's Information Matrix associated to a complete observation, defined as $\boldsymbol{I}(\hat{\boldsymbol{\theta}}_t) = -\mathbb{E}_{\boldsymbol{y}} \left\{ \nabla^2_{\tilde{\boldsymbol{\theta}}} \log f(\boldsymbol{y}(t), \boldsymbol{a}; \hat{\boldsymbol{\theta}}_t) \right\}$. After some straightforward algebra, the compact expressions for $\hat{x}_{t+1}$ and $\hat{\sigma}^2_{t+1}$ are

$$\hat{x}_{t+1} = \hat{x}_t + \frac{\gamma_t}{Ntp}(s_2(t) - s_1(t)\hat{x}_t), \tag{11}$$

$$\hat{\sigma}^2_{t+1} = \hat{\sigma}^2_t + \frac{\gamma_t}{2Nt} \left(-\hat{\sigma}^2_t Nt + s_3(t) - 2s_2(t)\hat{x}_t + s_1(t)\hat{x}^2_t \right), \tag{12}$$

with $s_1(t), s_2(t)$ and $s_3(t)$ defined in equations (8)-(10).

C. Classical algorithms

Online EM algorithms in the literature are based on stochastic approximation techniques [10], [9] or substitute the maximization step by a stochastic gradient step [8]. In these approaches, it is assumed that there is a new realization of a hidden variable for each new observation, and that the observations are iid. Thus, the conditional expectation being maximized is

$$Q(\tilde{\boldsymbol{\theta}}\,;\, \hat{\boldsymbol{\theta}}) = \frac{1}{NL} \sum_{l=1}^{NL} \mathbb{E}_{a(l)} \{ \log f(y(l), a(l) \,|\, \tilde{\boldsymbol{\theta}}) \,|\, \hat{\boldsymbol{\theta}}, y(l) \}, \tag{13}$$

where $y(l)$ denotes the l^{th} element of vector $\boldsymbol{y}(L)$, and $a(l)$ is the hidden variable associated with $y(l)$.

The online algorithm introduced by Cappé in [10] aims to finding a stationary point of (13) by solving the equation

$$\mathbb{E}_y \{ \mathbb{E}_a \{ \log f(y, a \,|\, \tilde{\boldsymbol{\theta}}) \mid \bar{\boldsymbol{\theta}}(Q(\tilde{\boldsymbol{\theta}}; \hat{\boldsymbol{\theta}})), \boldsymbol{y} \} \} = Q(\tilde{\boldsymbol{\theta}}; \hat{\boldsymbol{\theta}}),$$

where $\bar{\boldsymbol{\theta}}(Q(\tilde{\boldsymbol{\theta}}\,;\, \hat{\boldsymbol{\theta}})) = \arg\max_{\tilde{\boldsymbol{\theta}}} Q(\tilde{\boldsymbol{\theta}}\,;\, \hat{\boldsymbol{\theta}})$, by means of the following Robbins-Monro procedure

$$\hat{Q}_t(\tilde{\boldsymbol{\theta}}) = \hat{Q}_{t-1}(\tilde{\boldsymbol{\theta}}) \tag{14}$$
$$+ \gamma_t (\mathbb{E}_{a(t)} \{ \log f(y(t), a(t) \,|\, \tilde{\boldsymbol{\theta}}) \,|\, \bar{\boldsymbol{\theta}}(\hat{Q}_{t-1}(\tilde{\boldsymbol{\theta}})), y(t) \} - \hat{Q}_{t-1}(\tilde{\boldsymbol{\theta}})),$$

where γ_t is the step size and $\bar{\boldsymbol{\theta}}(\hat{Q}_{t-1}(\tilde{\boldsymbol{\theta}})) = \hat{\boldsymbol{\theta}}_t$.

The other classical approach to online EM algorithms was proposed by Titterington in [8] and maximizes (13) with the following recursion

$$\hat{\boldsymbol{\theta}}_{t+1} = \hat{\boldsymbol{\theta}}_t + \gamma_t \boldsymbol{I}^{-1}(\hat{\boldsymbol{\theta}}_t) \mathbb{E}_{a(t)} \{ \nabla_{\hat{\boldsymbol{\theta}}_t} \log f(y(t), a(t) \,|\, \tilde{\boldsymbol{\theta}}) | \hat{\boldsymbol{\theta}}_t, y(t) \}.$$

It is important to remark that, since the observations are assumed iid, both algorithms calculate the conditional expectation only taking into account the t^{th} observation. However, in the signal model described in Section II the observations made at a node are dependent and, hence, (2) cannot be decomposed into a sum of conditional expectations over l as in (13). Therefore, the Robbins-Monro procedure and the stochastic gradient, proposed in [10] and [8] respectively, are not adequate for our model. Since the calculating the conditional expectation conditioned on a single observation $y_i(t)$ is not reliable for our signal model, we propose the following adapted equations in order to provide a fair comparison in the simulations:

$$\hat{Q}_t(\tilde{\boldsymbol{\theta}}) = \hat{Q}_{t-1}(\tilde{\boldsymbol{\theta}})$$
$$+ \gamma_t \sum_{i=1}^{N} \mathbb{E}_{a_i} \{ \log f(y_i(t), a_i \,|\, \tilde{\boldsymbol{\theta}}) \,|\, \bar{\boldsymbol{\theta}}(\hat{Q}_t(\tilde{\boldsymbol{\theta}})), \boldsymbol{y}_i(t) \} - \hat{Q}_{t-1}(\tilde{\boldsymbol{\theta}}),$$

for Cappé's algorithm, and

$$\hat{\boldsymbol{\theta}}_{t+1} = \hat{\boldsymbol{\theta}}_t + \gamma_t \boldsymbol{I}^{-1}(\hat{\boldsymbol{\theta}}_t) \sum_{i=1}^{N} \mathbb{E}_{a_i} \{ \nabla_{\hat{\boldsymbol{\theta}}_t} \log f(y_i(t), a_i | \tilde{\boldsymbol{\theta}}) | \hat{\boldsymbol{\theta}}_t, \boldsymbol{y}_i(t) \}.$$

for Titterington's. Nevertheless, we observe the following significant differences that may impact the performance. Cappé's adapted version accumulates outdated expectations since they were calculated using less than t observations, whereas the cumulative variables $d_{1,i}$ and $d_{2,i}$ in ROEM do not depend on a_i. In the case of DOIN, Eqs. (11) and (12) calculate the full gradient with t observations instead of just one at the same computational cost.

V. Distributed online EM

In this section we introduce the distributed versions of the ROEM and OIN algorithms. The distributed ROEM algorithm is based on the diffusion-based distributed EM proposed in [1], which enables the distributed computation of the parameters whenever they are expressed as operations between summations over i of variables available at the nodes. The ROEM fulfills this requirement since it is expressed in terms of the online variables (8)-(10).

Consider a sensor network formed by N nodes in which each node can only communicate with the neighbors within a given radius. The nodes lie on a connected undirected graph with weighted adjacency matrix $\boldsymbol{W} \in \mathbb{R}^{N\times N}$, in which an entry $\boldsymbol{W}_{ij}$ different from zero indicates that information flows from node j to node i. We assume that the matrix $\boldsymbol{W}$ meets the following conditions: $\boldsymbol{W}\mathbf{1} = \mathbf{1}$, $\mathbf{1}^T\boldsymbol{W} = \mathbf{1}^T$, and $\rho(\boldsymbol{W} - \frac{\mathbf{1}\mathbf{1}^T}{N}) < 1$, where $\rho(\cdot)$ is the spectral radius.

In the diffusion-based ROEM (DBROEM) algorithm, each node keeps a local copy of the online variables, which are updated when a new observation is taken at each iteration. The nodes combine their local copies with those of their neighbors so that the information is diffused through the network, and the nodes obtain an estimate of the global online variables. At each iteration, the nodes obtain the estimate of the global online variables as

$$\hat{s}_{1,i}(t) = \sum_{j=1}^{N} \boldsymbol{W}_{i,j}\left[(1-\alpha_t)\hat{s}_{1,j}(t-1) + \alpha_t t \hat{a}_j(t)\right],$$
$$\hat{s}_{2,i}(t) = \sum_{j=1}^{N} \boldsymbol{W}_{i,j}\left[(1-\alpha_t)\hat{s}_{2,j}(t-1) + \alpha_t \hat{a}_j(t) d_{1,j}(t)\right],$$
$$\hat{s}_{3,i}(t) = \sum_{j=1}^{N} \boldsymbol{W}_{i,j}\left[(1-\alpha_t)\hat{s}_{3,j}(t-1) + \alpha_t d_{2,j}(t)\right],$$

where α_t is a vanishing control parameter with values $0 \leq \alpha_t \leq 1$ that gradually increases the weight of the memory term $\hat{s}_{j,i}$, which diffuses the information throughout the network, while the weight of the innovation is decreased. Thus, in the event that the nodes stop acquiring new observations, or that $\alpha_t = 0$, the network will reach a consensus on the value of the global online variables. The local estimates of x and σ^2 are calculated by each node as

$$\hat{x}_i(t+1) = \frac{\hat{s}_{2,i}(t)}{\hat{s}_{1,i}(t)},$$
$$\hat{\sigma}_i^2(t+1) = \frac{\hat{s}_{3,i}(t) - 2\hat{s}_{2,i}(t)\hat{x}_i(t+1) + \hat{s}_{1,i}(t)\hat{x}_i(t+1)^2}{d_3(t)},$$

where $d_3(t) = d_3(t-1) + \alpha_t(t - d_3(t-1))$.

The proposed distributed OIN (DOIN) algorithm relies on the well-known adapt-then-combine (ATC) [14] scheme since the gradient term in (11) and (12) is also a summation over variables available at the nodes. In this approach, each node performs a local gradient update and the result is combined with the updates of its connected neighbors. The estimates at node i are given by

$$\hat{x}_i(t+1) = \sum_{j=1}^{N} \boldsymbol{W}_{i,j}\left[\hat{x}_j(t) - \frac{\gamma_t}{tp}(\hat{a}_j(t)(d_{1,j}(t) - t\hat{x}_j(t)))\right],$$
$$\hat{\sigma}_i^2(t+1) = \sum_{j=1}^{N} \boldsymbol{W}_{i,j}\left[\hat{\sigma}_j^2(t) - \frac{\gamma_t}{2}(-\hat{\sigma}_j^2(t) + \frac{1}{t}d_{2,j}(t) - \frac{2}{t}\hat{a}_j(t)\hat{x}_j(t)d_{1,j}(t) + \hat{a}_j(t)\hat{x}_j(t)^2)\right].$$

Since every node is minimizing the same cost function, the exchange of information between nodes accelerates the convergence to the minimum. Moreover, a consensus would also be reached if the nodes stopped acquiring new observations. The ATC scheme was also used in [11] to implement Cappé's algorithm in a distributed fashion. In this algorithm the nodes perform the local update with (14), using a number of sufficient statistics instead of the complete loglikelihood, and then average their updates with those of their neighbors.

VI. Simulation results

In this section we compare the performance of the following variants of the EM algorithm: centralized Batch EM, ROEM, the adapted versions of Cappé [10] (denoted by OEM1) and Titterington [8] (OEM2), OIN, DBROEM, DOIN, and the one given in [11] (DOEM1). We have simulated a sensor network with $N = 50$ nodes with a connectivity radius $r = 20$ randomly deployed in a 100×100 square. The algorithms have been run for $L = 1700$ iterations and $R = 6000$ realizations, with $x = 1$, $p = 0.6$ and a signal-to-noise ratio (SNR), defined as SNR $= \frac{x^2}{\sigma^2}$, ranging from -15 to 15 dB. The step sizes have been chosen so that each algorithm achieves the lowest estimation error. Vector $\boldsymbol{a}$ and the noise are randomized at each realization, whereas the position of the nodes and weight matrix, which has been generated with the Metropolis rule [15], remain static. As a performance metric we use the mean square error (MSE), defined as

$$\text{MSE}(t) = \frac{1}{R}\sum_{r=1}^{R}\frac{1}{N}\sum_{i=1}^{N}(\hat{x}_{i,r}(t) - x)^2,$$

where $\hat{x}_{i,r}(t)$ is the estimate of node i at iteration t and realization r.

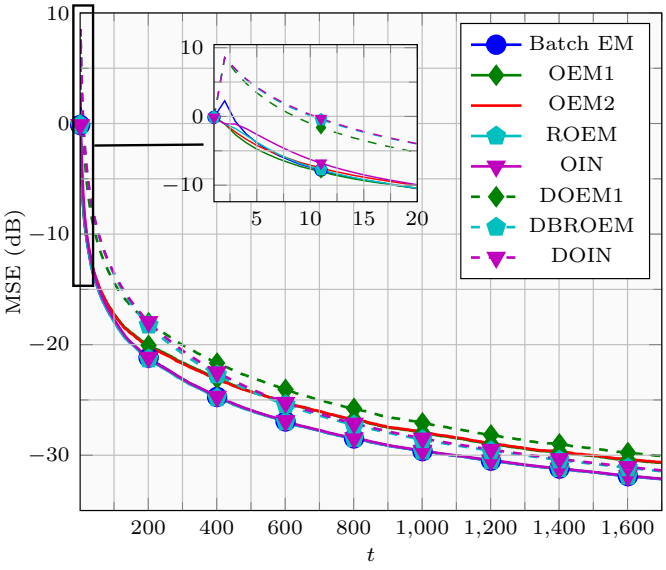

Figure 1. MSE vs. t for SNR= -15 dB.

Figure 1 shows the evolution of the MSE for all the algorithms over t for SNR=-15 dB. Note that, at each iteration, the line marked "Batch EM" shows the MSE of Batch EM run with t observations until convergence. We observe in the

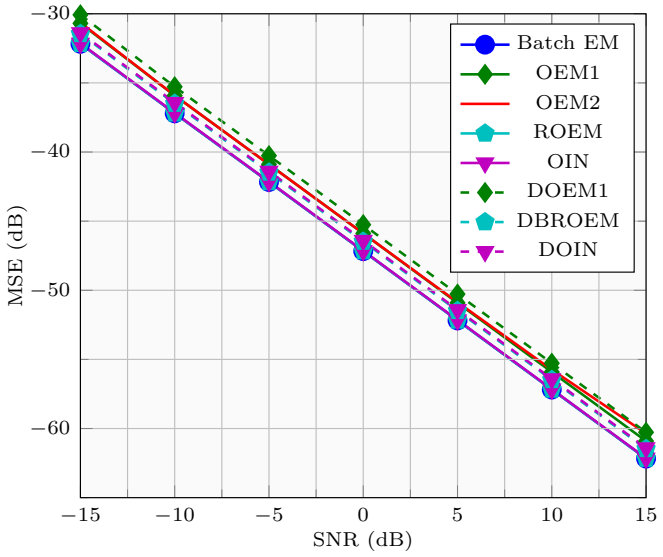

Figure 2. MSE vs. SNR after 1700 iterations.

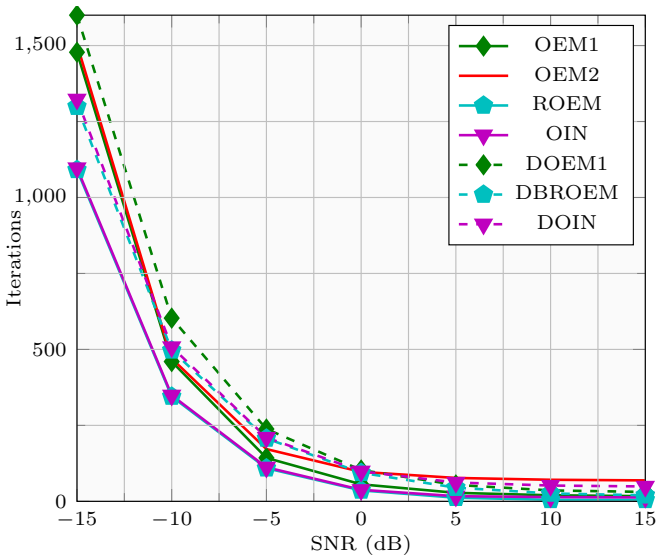

Figure 3. Iterations needed to reach an MSE of 10^{-3} at different SNR.

magnified view that ROEM and Batch EM merge after a few iterations. OIN also matches Batch EM, although it requires more iterations. We also observe that OEM1 and OEM2 perform at the same level but worse than the two proposed algorithms, which is due to the fact that the conditional expectations calculated for earlier observations are not updated with the information from the newly acquired observations. Among the distributed algorithms, DOIN and DBROEM have a very similar MSE through all the iterations, while the DOEM1 algorithm has it higher. Note that the distributed algorithms cannot converge to their centralized versions since the nodes are continuously acquiring new observations, and this new information requires several iterations to be diffused throughout the network. Still if the nodes were to stop acquiring new observations, DBROEM would be able to further reduce the gap with ROEM since, eventually, the information from all the observations would be globally available.

Figure 2 shows the final MSE after $L = 1700$ iterations for all the algorithms at different SNR. We observe that both ROEM and OIN coincide with Batch EM for the whole range of SNR. Still, as seen in Figure 1, the OIN algorithm requires more iterations to match Batch EM. Regarding the distributed algorithms, DBROEM and DOIN have the same MSE, which is below that of OEM1 and OEM2, and DOEM1 has the highest MSE.

Figure 3 shows the iterations and observations that each algorithm needs to reach an MSE of -30 dB at different SNR. Since a larger number of observations would result in a lower error for Batch EM, we see that the online algorithms need to acquire more observations at lower SNR to offset the rise in MSE due to the higher noise variance. We see that ROEM and OIN need the least observations and iterations, with ROEM having a slight edge for SNRs above 5 dB. On the other hand, DBROEM is the fastest of the distributed algorithms, requiring slightly fewer observations than DOIN for SNRs below 5 dB, and showing a more noticeable difference for SNRs above.

References

[1] S. S. Pereira, R. Lopez-Valcarce *et al.*, "A diffusion-based EM algorithm for distributed estimation in unreliable sensor networks," *Signal Processing Letters, IEEE*, vol. 20, no. 6, pp. 595–598, 2013.

[2] R. D. Nowak, "Distributed EM algorithms for density estimation and clustering in sensor networks," *Signal Processing, IEEE Transactions on*, vol. 51, no. 8, pp. 2245–2253, 2003.

[3] D. Gu, "Distributed EM algorithm for Gaussian mixtures in sensor networks," *Neural Networks, IEEE Transactions on*, vol. 19, no. 7, pp. 1154–1166, 2008.

[4] P. Forero, A. Cano, G. B. Giannakis *et al.*, "Consensus-based distributed expectation-maximization algorithm for density estimation and classification using wireless sensor networks," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 1989–1992.

[5] 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.

[6] S. Silva Pereira, R. Lopez-Valcarce *et al.*, "A diffusion-based distributed EM algorithm for density estimation in wireless sensor networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4449–4453.

[7] 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*. Springer, 1998, pp. 355–368.

[8] D. M. Titterington, "Recursive parameter estimation using incomplete data," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 257–267, 1984.

[9] M.-A. Sato and S. Ishii, "On-line EM algorithm for the normalized Gaussian network," *Neural computation*, vol. 12, no. 2, 2000.

[10] O. Cappé and E. Moulines, "On-line expectation–maximization algorithm for latent data models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 71, no. 3, pp. 593–613, 2009.

[11] G. Morral, P. Bianchi, and J. Jakubowicz, "On-line gossip-based distributed expectation maximization algorithm," in *Statistical Signal Processing Workshop (SSP), 2012 IEEE*. IEEE, 2012, pp. 305–308.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[13] K. Lange, "A quasi-Newton acceleration of the EM algorithm," *Statistica sinica*, vol. 5, no. 1, pp. 1–18, 1995.

[14] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *Signal Processing, IEEE Transactions on*, vol. 60, no. 8, pp. 4289–4305, 2012.

[15] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*. IEEE, 2005, pp. 63–70.