2011 IEEE International Workshop on Machine Learning for Signal Processing
September 18-21, 2011, Beijing, China

COLLABORATIVE LEARNING OF MIXTURE MODELS USING DIFFUSION ADAPTATION

Zaid J. Towfic, Jianshu Chen, and Ali H. Sayed

Department of Electrical Engineering
University of California, Los Angeles

ABSTRACT

In large ad-hoc networks, classification tasks such as spam filtering, multi-camera surveillance, and advertising have been traditionally implemented in a centralized manner by means of fusion centers. These centers receive and process the information that is collected from across the network. In this paper, we develop a decentralized adaptive strategy for information processing and apply it to the task of estimating the parameters of a Gaussian-mixture-model (GMM). The proposed technique employs adaptive diffusion algorithms that enable adaptation, learning, and cooperation at local levels. The simulation results illustrate how the proposed technique outperforms non-collaborative learning and is competitive against centralized solutions.

Index Terms— online-learning, Newton's method, diffusion, Expectation-Maximization, Gaussian-mixture-model, machine learning, distributed processing

1. INTRODUCTION

In large ad-hoc networks, classification tasks have traditionally been implemented in a centralized manner at fusion servers that retain and process information collected from across the network [1]. Server-based classification schemes can suffer from a handful of limitations. First, fusion-based solutions are centralized in nature and demand high levels of communication resources between the fusion center and the access points at which the data are collected. Second, in many classification problems, the amount of data that is available is abundant and continuous, which further compounds the communications requirements. Furthermore, in applications where the data are collected across nodes that are already distributed in space, it may be more convenient to process the data in a decentralized manner through local processing rather than rely on repeated data transfer to the central server. This last point is especially relevant when feature extraction mechanisms preserve privacy features, and transmissions over multiple hops may pose security risks and copyright issues. In some other applications, such as multi-camera tracking and surveillance, item suggestion and advertising, and social tagging and bookmarking, it is inherently preferable to process data in a distributed manner and in real-time. For all these reasons, we focus in this work on proposing decentralized strategies that are able to endow distributed agents with learning and adaptation abilities. These distributed techniques are able to deliver enhanced performance and to compete favorably with centralized solutions.

In a series of recent works, a family of diffusion-based adaptation strategies have been proposed to enable estimation, tracking, and detection over distributed networks [2–6]. The adaptive diffusion techniques have the distinct advantage of relying solely on in-network (local) processing to enable adaptation and learning in real-time. In this paper, we show how to apply the diffusion strategy to maximize the Expectation-Maximization (EM) [7] utility function. The EM algorithm is a popular technique that is used for both classification and clustering purposes [8, 9]. For example, EM algorithms have been applied to Gaussian Mixture Models (GMMs) in several applications including speaker identification and image retrieval [10, 11]. In recent works [12, 13], distributed EM algorithms were proposed. In [12], the algorithm estimates local sufficient statistics at each node and then shares information with neighbors to arrive at the global statistics through an average consensus filter. Subsequently, each node accomplishes the maximization step based on the estimated global statistics. In [13], two rounds of sharing are conducted: first, the local statistics are averaged between neighbors and, second, a new round of sharing communicates pre-estimates between neighbors. However, both of these approaches apply only when the mixture model is Gaussian, where the maximization can be solved analytically and in closed form.

In this paper, we develop a general distributed approach to solve the problem for general mixture models where the maximization step is solved by an adaptive diffusion process rather than in closed form. This adaptive diffusion procedure provides improved steady-state mean-square-error in simulation compared to [**?**, 13]. In the maximization step, instead of using a steepest-descent argument for optimization, we develop a Newton's recursion based on a diffusion adaptation strategy. Compared with consensus-based solutions, the diffusion strategy does not require different agents to converge to the same global statistics; the individual agents are allowed flexibility through adaptation and through their own assessment of local information. Diffusion strategies also enable recursive learning and adaptation. We test the proposed solution on the "Iris" dataset [14] and on a synthetic dataset generated from a GMM consisting of two components. The simulated results indicate that the mean-square-deviation of the estimated parameter is within 1dB of the centralized solution, over 3dB better than that of the algorithms proposed in [12, 13], and over 10dB better than non-cooperative schemes.

We use the following notation throughout the paper: random quantities are denoted in boldface while deterministic quantities are plain. Matrices are denoted by capital letters while vectors and scalars are denoted by small letters. No distinction is made between vectors and scalars, except for the context.

2. PROBLEM FORMULATION

Consider a connected network consisting of N nodes. Each node k has access to multiple i.i.d. vector realizations $\{\boldsymbol{x}_{k,1}, \dots, \boldsymbol{x}_{k,D}\}$. It is assumed that all nodes in the network observe data generated by the same probability density function (PDF). The PDF of the data is assumed to consist of a mixture of C component distributions as

Email:{ztowfic, jshchen, sayed}@ee.ucla.edu. This work was supported in part by NSF grants CCF-1011918, CCF-0942936, and ECS-0725441.

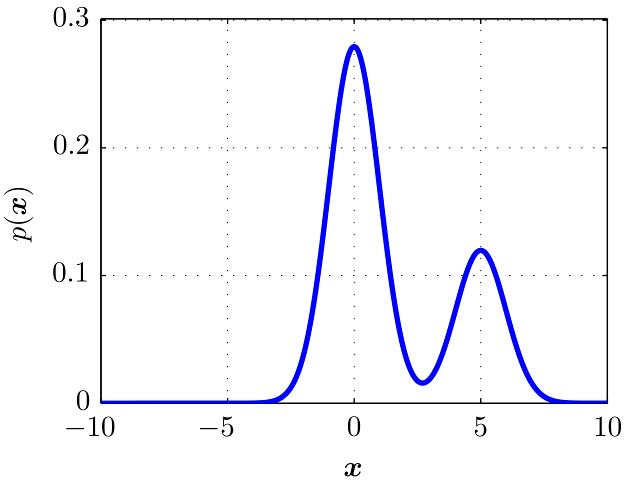

Fig. 1. Example of a PDF of a mixture model consisting of two Gaussian components with means 0 and 5, respectively, and unit variance. Instances are drawn from the zero-mean component 70% of the time.

follows:

$$p(\boldsymbol{x}) = \sum_{j=1}^{C} \alpha_j p(\boldsymbol{x}|z=j;\beta_j) \tag{1}$$

where α_j is the prior probability that the sample comes from class $z=j$, and β_j are parameters that define the jth component of the distribution. The coefficients α_j satisfy $\sum_{j=1}^{C}\alpha_j = 1$ and $\alpha_j \geq 0$ for all $j \in \{1,\ldots,C\}$. In this way, each instance $\boldsymbol{x}_{k,i}$ arises from a mixture of C component distributions each weighted by the prior α_j. One example of a distribution as in (1) would be to consider data that arises 70% of the time from a Gaussian distribution with mean 0 and variance 1, and 30% of the time from a Gaussian distribution with mean 5 and variance 1. In this case, we would set $\alpha_1 = 0.7$, $\alpha_2 = 0.3$, $\boldsymbol{x}|z=1 \sim \mathcal{N}(0,1)$, $\boldsymbol{x}|z=2 \sim \mathcal{N}(5,1)$, $\beta_1 = (\mu_1,\sigma_1^2) = (0,1)$, and $\beta_2 = (\mu_2,\sigma_2^2) = (5,1)$. The probability density function for this example is illustrated in Figure 1.

Our objective is to estimate the model parameter vector θ from all the data, where

$$\boxed{\theta = \left(\{\alpha_j\}_{j=1}^{C}, \{\beta_j\}_{j=1}^{C}\right)}$$

The standard method used in the literature to estimate such parameters is the expectation maximization (EM) algorithm [9]. The general EM algorithm introduces unobserved variables $\boldsymbol{z}_{k,i}$, which, together with the observed variables $\boldsymbol{x}_{k,i}$, form the extended or complete variables $\boldsymbol{y}_{k,i} = (\boldsymbol{x}_{k,i}, \boldsymbol{z}_{k,i})$. Then, at each iteration, the algorithm evaluates the expectation of the log-likelihood function of the complete data $\{\boldsymbol{y}_{k,i}\}$ conditioned on the observed samples $\{\boldsymbol{x}_{k,i}\}$ and the current estimate of the parameter θ of the mixture model. For the mixture model, the unobserved variable $\boldsymbol{z}_{k,i}$ usually denotes the mixture component from which the instance $\boldsymbol{x}_{k,i}$ is generated. It is known [7, 15] that the EM algorithm can make the likelihood function increase until it reaches a (local or global) maximum.

The EM algorithm can be applied by each node k individually to its own data. The argument in [9, pp.46] shows that, at each iteration, node k would maximize the following objective function:

$$\boxed{\begin{aligned} J_k^{\text{loc}}(\theta) = &\sum_{n=1}^{D}\sum_{z_{k,n}=1}^{C} p(z_{k,n}|\boldsymbol{x}_{k,n};\theta_{k,i-1}) \\ &\times \log\left(\alpha_{z_{k,n}} p(\boldsymbol{x}_{k,n}|z_{k,n};\theta)\right)\end{aligned}} \tag{2}$$

where $\theta_{k,i-1}$ is the node k's estimate of θ at time $i-1$, and the probability $p(z_{k,n}|\boldsymbol{x}_{k,n};\theta_{k,i-1})$ is evaluated by

$$\boxed{p(z_{k,n}|\boldsymbol{x}_{k,n};\theta_{k,i-1}) \triangleq \frac{p(\boldsymbol{x}_{k,n}|z_{k,n};\theta_{k,i-1})\alpha_{z_{k,n}}}{\sum\limits_{z_{k,n}} p(\boldsymbol{x}_{k,n}|z_{k,n};\theta_{k,i-1})\alpha_{z_{k,n}}}} \tag{3}$$

The EM algorithm can also be implemented at a fusion center, which is assumed to collect all data from the N nodes. Then, the global objective function becomes:

$$\boxed{J^{\text{glob}}(\theta) \triangleq \sum_{l=1}^{N} J_l^{\text{loc}}(\theta)} \tag{4}$$

with $\theta_{k,i-1}$ replaced by θ_{i-1} for all $k=1,\ldots,N$. There are many ways to maximize (4) iteratively such as using a steepest descent method or Newton's method. Here, we consider Newton's method, which takes the following form:

$$\begin{aligned}\theta_i &= \theta_{i-1} - \lambda\cdot[\nabla^2 J^{\text{glob}}(\theta_{i-1})]^{-1}\nabla J^{\text{glob}}(\theta_{i-1}) \\ &= \theta_{i-1} - \lambda\cdot\sum_{l=1}^{N}\left[\sum_{l=1}^{N}\nabla^2 J_l^{\text{loc}}(\theta_{i-1})\right]^{-1}\nabla J_l^{\text{loc}}(\theta_{i-1})\end{aligned} \tag{5}$$

where λ is some small step-size and $\nabla J_l^{\text{loc}}(\theta)$ and $\nabla^2 J_l^{\text{loc}}(\theta)$ are the gradient vector and the Hessian matrix of the local objective function $J_l^{\text{loc}}(\theta)$, respectively. After the new estimate θ_i is computed, it is sent back to all nodes, i.e., $\theta_{k,i} = \theta_i$.

Algorithm (5) is not distributed: it requires access to all local data across all nodes. In the next section, we develop a distributed strategy based on the adaptive diffusion strategies of [2, 4]

3. DISTRIBUTED OPTIMIZATION VIA DIFFUSION

First, we rewrite the global objective function in the following form [4]:

$$J^{\text{glob}}(\theta) = J_k^{\text{loc}}(\theta) + \sum_{l\neq k} J_l^{\text{loc}}(\theta) \tag{6}$$

Assume $J_l^{\text{loc}}(\theta)$ is second-order differentiable, and that there exists a θ_l^{loc} that optimizes $J_l^{\text{loc}}(\theta)$. Then, $J_l^{\text{loc}}(\theta)$ $(l \neq k)$ can be approximated, via a second order Taylor series expansion around θ_l^{loc}, as:

$$\begin{aligned} J_l^{\text{loc}}(\theta) \approx &J_l^{\text{loc}}(\theta_l^{\text{loc}}) + \nabla J_l^{\text{loc}}(\theta_l^{\text{loc}})^\top(\theta - \theta_l^{\text{loc}}) \\ &+ \frac{1}{2}(\theta-\theta_l^{\text{loc}})^\top \nabla^2 J_l^{\text{loc}}(\theta_l^{\text{loc}})(\theta-\theta_l^{\text{loc}}) \\ = &\left\|\theta - \theta_l^{\text{loc}}\right\|_{\Gamma_l}^2 + \text{constant}\end{aligned} \tag{7}$$

where $\Gamma_l \triangleq \nabla^2 J_l^{\text{loc}}(\theta_l^{\text{loc}})/2$. In the last step, we used the fact that $\nabla J_l^{\text{loc}}(\theta_l^{\text{loc}}) = 0$, because θ_l^{loc} optimizes $J_l^{\text{loc}}(\theta)$. Furthermore, since $J_l^{\mathrm{loc}}(\theta_l^{\mathrm{loc}})$ is independent of the optimization variable θ, we treat it as a constant and drop it from now on. Then, the objective function in (4) can be replaced by the equivalent objective function:

$$J^{\mathrm{glob}'}(\theta) = J_k^{\mathrm{loc}}(\theta) + \sum_{l\neq k} \left\|\theta - \theta_l^{\mathrm{loc}}\right\|_{\Gamma_l}^2 \tag{8}$$

Optimizing the above objective function at every node k still requires the nodes to have access to global information, namely, the local estimates θ_l^{loc}, and the matrices Γ_l at the other nodes. However, (8) provides insights into motivating a useful distributed implementation. Specifically, we use (8) to motivate a local cost functions at the individual nodes that serve as approximations to (8). To do so, we first confine the sum in (8) to be over the neighborhood of node k, i.e., $\mathcal{N}_k/\{k\}$. Then, we replace the local optimal θ_l^{loc} with an intermediate estimate for it that will be available at node l, denoted by θ_l. In this way, each node k can proceed to minimize the modified objective function:

$$J_k^{\mathrm{dist}}(\theta) = J_k^{\mathrm{loc}}(\theta) + \sum_{l\in\mathcal{N}_k/\{k\}} \left\|\theta - \theta_l^{\mathrm{loc}}\right\|_{\Gamma_l}^2 \tag{9}$$

We could proceed here and develop a Newton's method to optimize (9) in a manner similar to (5). However, in order to adapt to the case in which the unknown parameter θ is a matrix in the next section, we will go through a quadratic approximation argument to motivate Newton's method [16].

Suppose we have an intermediate estimate $\theta_{k,i-1}$ of the unknown θ at node k. Let us introduce a small perturbation δ to $\theta_{k,i-1}$, and perform a second-order Taylor series expansion of $J_k^{\mathrm{dist}}(\theta_{k,i-1}+\delta)$ around $\theta_{k,i-1}$. Then, by (9), we get

$$\begin{aligned} &J_k^{\mathrm{dist}}(\theta_{k,i-1}+\delta)\\ &\quad\approx J_k^{\mathrm{loc}}(\theta_{k,i-1}) + \nabla_\theta J_k^{\mathrm{loc}}(\theta_{k,i-1})^\top\delta + \frac{1}{2}\delta^\top\nabla_\theta^2 J_k^{\mathrm{loc}}(\theta_{k,i-1})\delta\\ &\qquad + \sum_{l\in\mathcal{N}_k/\{k\}} \|\theta_{k,i-1}+\delta-\theta_l\|_{\Gamma_l}^2 \end{aligned} \tag{10}$$

The main idea is to approximate $J_k^{\mathrm{dist}}(\theta_{k,i-1}+\delta)$ by a quadratic function centered around $\theta_{k,i-1}$, and then pick the value of δ that optimizes the quadratic approximation. This happens when the gradient with respect to δ is zero. Taking the gradient of above approximated objective funtion with respect to δ, we get

$$\begin{aligned} \nabla_\delta J_k^{\mathrm{dist}}(\theta_{k,i-1}+\delta) \approx{}& \nabla_\theta J_k^{\mathrm{loc}}(\theta_{k,i-1}) + \nabla_\theta^2 J_k^{\mathrm{loc}}(\theta_{k,i-1})\delta\\ &+2\sum_{l\in\mathcal{N}_k/\{k\}} \Gamma_l(\theta_{k,i-1}+\delta-\theta_l) \end{aligned}$$

Setting $\nabla_\delta J_k^{\mathrm{dist}}(\theta_{k,i-1}+\delta) = 0$ and solving for δ, the optimum update step is found to be:

$$\begin{aligned} \delta &\approx -H_i^{-1}\left[\nabla_\theta J_k^{\mathrm{loc}}(\theta_{k,i-1}) + 2\sum_{l\in\mathcal{N}_k/\{k\}} \Gamma_l(\theta_{k,i-1}-\theta_l)\right]\\ &= -H_i^{-1}\left[\nabla_\theta J_k^{\mathrm{loc}}(\theta_{k,i-1}) + \sum_{l\in\mathcal{N}_k/\{k\}} \nabla_\theta^2 J_l^{\mathrm{loc}}(\theta_l)(\theta_{k,i-1}-\theta_l)\right] \end{aligned} \tag{11}$$

where

$$\begin{aligned} H_i &\triangleq \nabla_\theta^2 J_k^{\mathrm{loc}}(\theta_{k,i-1}) + 2\sum_{l\in\mathcal{N}_k/\{k\}} \Gamma_l\\ &= \nabla_\theta^2 J_k^{\mathrm{loc}}(\theta_{k,i-1}) + \sum_{l\in\mathcal{N}_k/\{k\}} \nabla_\theta^2 J_l^{\mathrm{loc}}(\theta_l) \end{aligned} \tag{12}$$

Next, we assume the Hessian matrices $\nabla_\theta^2 J_l^{\mathrm{loc}}(\theta)$ do not deviate from each other significantly for different l. We expect this approximation to be reasonable since all nodes sample data according to the same distribution, yielding similar Hessians. Thus,

$$\nabla_\theta^2 J_l^{\mathrm{loc}}(\theta_l) \approx b_{k,l}H_i, \quad \nabla_\theta^2 J_k^{\mathrm{loc}}(\theta_{k,i-1}) \approx b_{k,k}H_i$$

From (12), this means that

$$H_i \approx \sum_{l\in\mathcal{N}_k} b_{k,l}H_i \Rightarrow \sum_{l\in\mathcal{N}_k} b_{k,l} = 1 \tag{13}$$

where $b_{l,k}$ is some nonnegative scalar that scales the neighborhood Hessian to yield the local objective function's Hessian. Then, (11) simplifies to

$$\begin{aligned} \delta \approx{}& -b_{k,k}\left[\nabla_\theta^2 J_k^{\mathrm{loc}}(\theta_{k,i-1})\right]^{-1}\nabla_\theta J_k^{\mathrm{loc}}(\theta_{k,i-1})\\ &-\sum_{l\in\mathcal{N}_k/\{k\}} b_{k,l}(\theta_{k,i-1}-\theta_l) \end{aligned}$$

and the recursive update equation can be written as

$$\begin{aligned} \theta_{k,i} ={}& \theta_{k,i-1} - \lambda b_{k,k}\left[\nabla_\theta^2 J_k^{\mathrm{loc}}(\theta_{k,i-1})\right]^{-1}\nabla_\theta J_k^{\mathrm{loc}}(\theta_{k,i-1})\\ &-\nu_k\sum_{l\in\mathcal{N}_k/\{k\}} b_{k,l}(\theta_{k,i-1}-\theta_l) \end{aligned} \tag{14}$$

where λ and ν_k are small step-sizes associated with the Newton-step. We can accomplish the update (14) in two steps by generating an intermediate estimate $\psi_{k,i}$ as follows:

$$\psi_{k,i} = \theta_{k,i-1} - \lambda b_{k,k}\left[\nabla_\theta^2 J_k^{\mathrm{loc}}(\theta_{k,i-1})\right]^{-1}\nabla_\theta J_k^{\mathrm{loc}}(\theta_{k,i-1}) \tag{15}$$

$$\theta_{k,i} = \psi_{k,i} - \nu_k\sum_{l\in\mathcal{N}_k/\{k\}} b_{k,l}(\theta_{k,i-1}-\theta_l) \tag{16}$$

We further replace θ_l in (16) by the intermediate estimate that is available at node l at time i, namely, $\psi_{l,i}$. We also replace $\theta_{k,i-1}$ in (16) by $\psi_{k,i}$. Then, (16) leads to

$$\begin{aligned} \theta_{k,i} &= \psi_{k,i} - \nu_k\sum_{l\in\mathcal{N}_k/\{k\}} b_{k,l}(\psi_{k,i}-\psi_{l,i})\\ &= (1-\nu_k+\nu_k b_{k,k})\psi_{k,i} + \nu_k\sum_{l\in\mathcal{N}_k/\{k\}} b_{k,l}\psi_{l,i} \end{aligned} \tag{17}$$

Let us introduce the following coefficients

$$a_{k,k} = 1-\nu_k+\nu_k b_{k,k} \text{ and } a_{l,k} = \nu_k b_{l,k} \text{ for } l\neq k$$

Assume A is the matrix whose $\{l,k\}$-entry is $a_{l,k}$, then it satisfies:

$$\boxed{a_{l,k} = 0 \text{ if } l\notin\mathcal{N}_k, \mathbb{1}^T A = \mathbb{1}^T} \tag{18}$$

Define the Newton step of the the local objective function as

$$\boxed{\delta_{k,i-1}^{\mathrm{loc}} \triangleq -\left[\nabla^2 J_k^{\mathrm{loc}}(\theta_{k,i-1})\right]^{-1}\nabla J_k^{\mathrm{loc}}(\theta_{k,i-1})} \tag{19}$$

Then, the algorithm (15) and (17) can be expressed as:

$$\boxed{\begin{aligned}\psi_{k,i} &= \theta_{k,i-1} + \gamma_k \delta^{\text{loc}}_{k,i-1}\\ \theta_{k,i} &= \sum_{l\in\mathcal{N}_k} a_{l,k}\psi_{l,i}\end{aligned}} \tag{20}$$

where γ_k is some small step-size and $a_{l,k}$ are nonnegative combination coefficients used to combine the shared estimates. In the next section, we will derive the specific algorithm for estimating the multivariate Gaussian-mixture-model.

4. APPLICATION TO GAUSSIAN-MIXTURE-MODELS

In this section, we use (20) to derive update equations for the parameters of a multivariate GMM. In this case, the distribution of an instance vector $\boldsymbol{x}_{k,n}$ is described by (1), where the component distribution is given by

$$p(\boldsymbol{x}_{k,n}|z_{k,n}=j) \triangleq \frac{\exp\left\{-\frac{1}{2}\left\|\boldsymbol{x}_{k,n}-\mu_j\right\|^2_{\Sigma_j^{-1}}\right\}}{(2\pi)^{\frac{M}{2}}\det\left(\Sigma_j\right)^{\frac{1}{2}}} \tag{21}$$

where M is the length of the feature vector $\boldsymbol{x}_{k,n}$. Furthermore, μ_j and Σ_j are the mean vector and covariance matrix of the j-th Gaussian component, respectively.

Then,we can substitute (21) into (2), (4), (19), and (20) to derive the specific diffusion Newton method for EM. Before that, we first simplify the notation. Let us replace $z_{k,n}$ with the dummy index variable j as this will not change equation (2), and hence replace $\alpha_{z_{k,n}}$ with α_j. Then, (2) becomes:

$$\boxed{\begin{aligned}J_k^{\text{loc}}(\theta) = \sum_{n=1}^{D}\sum_{j=1}^{C} p(j|\boldsymbol{x}_{k,n};\theta_{k,i-1})\cdot\\ \left[\log(\alpha_j)+\log(p(\boldsymbol{x}_{k,n}|j;\theta))\right]\end{aligned}}$$

It is now possible to derive the Newton step (19) for the parameter θ, defined as:

$$\theta = \left(\{\mu_j\},\{\Sigma_j^{-1}\},\{\alpha_j\}\right) \tag{22}$$

which then completes the algorithm. Ideally, we can derive the recursion for θ jointly according to (19). However, this will complicate the algorithm. Hence, our strategy here is to derive the recursion for each of $\{\mu_j\}$, $\{\Sigma_j\}$, and $\{\alpha_j\}$ separately while fixing the other two. Afterwards, the three recursions will operate jointly to estimate θ.

4.1. Diffusion adaptation for $\{\mu_j\}$

We derive the Newton-step for the mean vectors $\{\mu_j\}_{j=1}^{C}$ directly from the definition in (19), while fixing $\{\Sigma_j\}$ and $\{\alpha_j\}$. We note that the gradient and the Hessian can respectively be written as:

$$\nabla_{\mu_j} J_k^{\text{loc}}(\{\mu_j\}) = \sum_{n=1}^{D} p(j|\boldsymbol{x}_{k,n};\theta_{k,i-1})\Sigma_j^{-1}(\boldsymbol{x}_{k,n}-\mu_j) \tag{23}$$

$$\nabla^2_{\mu_j} J_k^{\text{loc}}(\{\mu_j\}) = -\sum_{n=1}^{D} p(j|\boldsymbol{x}_{k,n};\theta_{k,i-1})\Sigma_j^{-1},\ j=1,\ldots,C$$

Then, the Newton step evaluated at

$$\boxed{\theta_{k,i-1} = (\{\mu_{k,i-1,j}\},\{\Sigma_{k,i-1,j}\},\{\alpha_{k,i-1,j}\})}$$

can be written as:

$$\boxed{\delta^{\mu,\text{loc}}_{k,i-1,j} = \frac{\sum_{n=1}^{D} p(j|\boldsymbol{x}_{k,n};\theta_{k,i-1})\left(\boldsymbol{x}_{k,n}-\mu_{k,i-1,j}\right)}{\sum_{n=1}^{D} p(j|\boldsymbol{x}_{k,n};\theta_{k,i-1})}} \tag{24}$$

It is worth noting here that Newton's method in this application has lower complexity than a steepest-descend solution for the same problem. This can be seen by comparing (23) with (24) and noting that it consists of a matrix-vector multiplication. The multiplication by the inverse of the Hessian reduces the complexity to vector subtraction. The main reason for this simplicity is that Newton's method uses a quadratic approximation to derive the optimal update, and the log-likelihood function for $\{\mu_j\}$ is exactly quadratic.

4.2. Diffusion adaptation for $\{\Sigma_j^{-1}\}$

In order to derive the Newton step for the matrix Σ_j^{-1}, we use the same approach used in Sec. 3. Specifically, we add a small perturbation to $\{\Sigma_{k,i-1,j}\}$ in the objective function, and optimize for the perturbation after expanding the objective function up to second order:

$$\begin{aligned}&J_k^{\text{loc}}(\Sigma^{-1}_{k,i-1,j}+\Delta)\\ &=\sum_{n=1}^{D} p(j|\boldsymbol{x}_{k,n};\theta_{k,i-1})\cdot\Big[\log(\alpha_{k,i-1,j})-\frac{M}{2}\log(2\pi)+\\ &\frac{1}{2}\log\det(\Sigma^{-1}_{k,i-1,j}+\Delta)-\frac{1}{2}\left\|\boldsymbol{x}_{k,n}-\mu_{k,i-1,j}\right\|^2_{\Sigma^{-1}_{k,i-1,j}+\Delta}\\ &+\text{constant}\Big] &(25)\\ &\approx\sum_{n=1}^{D} p(j|\boldsymbol{x}_{k,n};\theta_{k,i-1})\cdot\Big[\log(\alpha_{k,i-1,j})-\frac{M}{2}\log(2\pi)+\\ &\frac{1}{2}\log\det(\Sigma^{-1}_{k,i-1,j})+\frac{1}{2}\mathsf{Tr}(\Sigma_{k,i-1,j}\Delta)-\\ &\frac{1}{4}\mathsf{Tr}(\Sigma_{k,i-1,j}\Delta\Sigma_{k,i-1,j}\Delta)-\frac{1}{2}\left\|\boldsymbol{x}_{k,n}-\mu_{k,i-1,j}\right\|^2_{\Sigma^{-1}_{k,i-1,j}+\Delta}\\ &+\text{constant}\Big] &(26)\end{aligned}$$

where we used the second-order Taylor series expansion of $\log\det(X)$ given in [16]:

$$\log\det(X+\Delta)\approx\log\det(X)+\mathsf{Tr}(X\Delta)-\frac{1}{2}\mathsf{Tr}(X\Delta X\Delta)$$

Next, just as in Sec. 3, we need to evaluate the gradient of $J_k^{\text{loc}}(\Sigma^{-1}_{k,i-1,j}+\Delta)$ with respect to Δ, set it to zero, and solve for the optimal Δ. Using the gradient properties:

$$\nabla_X \mathsf{Tr}(AX)=A,\quad \nabla_X\mathsf{Tr}(AXAX)=2AXA$$

for symmetric matrices A and X, we can compute the gradient of $J_k^{\text{loc}}(\Sigma^{-1}_{k,i-1,m}+\Delta)$ with respect to Δ as:

$$\begin{aligned}&\nabla_\Delta J_k^{\text{loc}}(\Sigma^{-1}_{k,i-1,j}+\Delta)\\ &\approx\sum_{n=1}^{D} p(j|\boldsymbol{x}_{k,n};\theta_{k,i-1})\Big[\frac{1}{2}\Sigma_{k,i-1,j}-\frac{1}{2}\Sigma_{k,i-1,j}\Delta\Sigma_{k,i-1,j}-\\ &\frac{1}{2}(\boldsymbol{x}_{k,n}-\mu_{k,i-1,j})(\boldsymbol{x}_{k,n}-\mu_{k,i-1,j})^\top\Big]\end{aligned}$$

Setting the gradient to zero and solving for Δ yields the Newton step in (27).

$$\boxed{\Delta_{k,i-1}^{\Sigma^{-1},\text{loc}} = \Sigma_{k,i-1,j}^{-1} - \Sigma_{k,i-1,j}^{-1} \frac{\sum_{n=1}^{D} p(j|\boldsymbol{x}_{k,n};\theta_{k,i-1})(\boldsymbol{x}_{k,n}-\mu_{k,i-1,j})(\boldsymbol{x}_{k,n}-\mu_{k,i-1,j})^\top}{\sum_{n=1}^{D} p(j|\boldsymbol{x}_{k,n};\theta_{k,i-1})} \Sigma_{k,i-1,j}^{-1}} \quad (27)$$

4.3. Diffusion adaptation for $\{\alpha_j\}$

We note that the recursion for $\{\alpha_j\}$ must preserve the fact that $\sum_j \alpha_j = 1$. We rewrite the optimization problem for $\alpha = \text{col}\{\alpha_1, \alpha_2, \ldots, \alpha_C\}$ as

$$\max \quad J_k^{\text{loc}}(\alpha) \quad (28)$$

$$\text{subject to} \quad \mathbb{1}^T\alpha = 1 \quad (29)$$

We follow the same approach as above where we add a step to the optimization variable and perform a second-order Taylor series approximation near α. The Newton step can then be found by solving the following system of equations [16]:

$$\begin{bmatrix} \nabla_\alpha^2 J_k^{\text{loc}}(\alpha) & \mathbb{1} \\ \mathbb{1}^T & 0 \end{bmatrix} \begin{bmatrix} \delta^{\alpha,\text{loc}} \\ w \end{bmatrix} = \begin{bmatrix} -\nabla_\alpha J_k^{\text{loc}}(\alpha) \\ 0 \end{bmatrix} \quad (30)$$

where w is the lagrange multiplier associated with the equality constraint and the gradient vector and Hessian matrix are defined as:

$$\nabla_\alpha J_k^{\text{loc}}(\alpha) = \left(\sum_{n=1}^{D} p(j|\boldsymbol{x}_{k,n};\theta_{k,i-1})\right) \text{col}\left\{\frac{1}{\alpha_1}, \ldots, \frac{1}{\alpha_C}\right\}$$

$$\nabla_\alpha^2 J_k^{\text{loc}}(\alpha) = \left(\sum_{n=1}^{D} p(j|\boldsymbol{x}_{k,n};\theta_{k,i-1})\right) \text{diag}\left\{\frac{-1}{\alpha_1^2}, \ldots, \frac{-1}{\alpha_C^2}\right\}$$

The Newton step can then be shown to be

$$\boxed{\begin{aligned}\delta_{k,i-1}^{\alpha,\text{loc}} = &\left[I - \frac{\left[\nabla_\alpha^2 J_k^{\text{loc}}(\alpha_{k,i-1})\right]^{-1}\mathbb{1}\mathbb{1}^T}{\mathbb{1}^T\left[\nabla_\alpha^2 J_k^{\text{loc}}(\alpha_{k,i-1})\right]^{-1}\mathbb{1}}\right] \times \\ &\left(-\left[\nabla_\alpha^2 J_k^{\text{loc}}(\alpha_{k,i-1})\right]^{-1}\nabla_\alpha J_k^{\text{loc}}(\alpha_{k,i-1})\right)\end{aligned}} \quad (31)$$

and $\left[\nabla_\alpha^2 J_k^{\text{loc}}(\alpha_{k,i-1})\right]^{-1}$ can be computed easily since the Hessian is diagonal. We note here, however, that it is possible in this formulation for $\alpha_{k,i-1,m}$ to become negative as we have not incorporated the inequality constraint $\alpha_{k,i-1,j} \geq 0$. For this reason, we project $\alpha_{k,i-1,m}$ onto the non-negative orthant after each computation. Hence, by (20), the recursion for $\{\alpha_j\}$ becomes:

$$\boxed{\begin{aligned}\psi_{k,i,j}^{\alpha} &= \max\left(\alpha_{k,i-1,j} + \gamma_k \delta_{k,i-1}^{\alpha,\text{loc}}(j), 0\right) \\ \alpha_{k,i,j} &= \sum_{l\in\mathcal{N}_k} a_{l,k}\psi_{l,i,j}^{\alpha}\end{aligned}} \quad (32)$$

where $\delta_{k,i-1}^{\alpha,\text{loc}}(j)$ is the jth element of $\delta_{k,i-1}^{\alpha,\text{loc}}$ in (31).

5. SIMULATION RESULTS

We evaluate our mixture model estimation method by first estimating a known distribution. This allows us to establish some performance results and compare the centralized and non-cooperative solutions against our algorithm, as well as compare our algorithm to the ones proposed in [12] and [13]. Following this, we will evaluate the performance of the algorithm on the popular "Iris" dataset [14]. For all simulations, the combination coefficients $a_{l,k}$ in (20) are chosen as $a_{l,k} = {}^1/_{|\mathcal{N}_k|}$ where $|\mathcal{N}_k|$ is the size of node k's neighborhood.

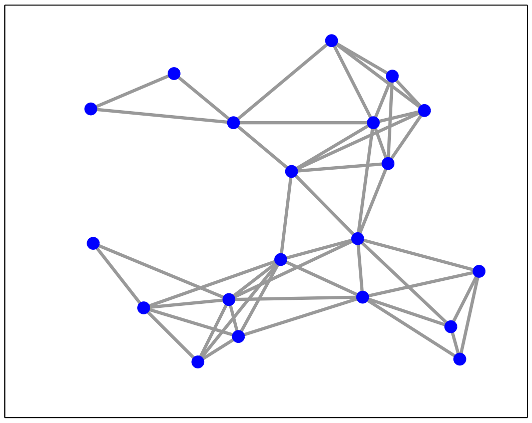

Fig. 2. A network used in the estimation of the mixture-model.

5.1. The "Two Gaussians" Data Set

In this section, we attempt to estimate the parameters of a 2-D GMM. The mixture consists of two Gaussian components with weights $\alpha_1 = 7/10$, $\alpha_2 = 3/10$ and means $\mu_1 = 0\mathbb{1}$ and $\mu_2 = 4\mathbb{1}$. The covariance matrices of the two components are defined to be $\Sigma_c = cI$ for $c = \{1, 2\}$. We simulate $N = 20$ nodes in a single-component network. An example of such network is illustrated in Figure 2. Each node receives a collection of $D = 100$ feature vectors $\{\boldsymbol{x}_{k,1}, \ldots, \boldsymbol{x}_{k,D}\}$ and performs Algorithm (20) to estimate the model parameters. The received feature vectors are corrupted by white Gaussian noise whose variance varies between the different nodes. The variance of the noise is chosen to be uniformly random in the range $[0, 1)$. A step-size of $\gamma_k = 0.05$ is chosen for all nodes. The mean-square-deviation (MSD) is defined as

$$\text{MSD}_\theta = \mathsf{E}\left[\|\hat{\theta} - \theta\|_2^2\right] \quad (33)$$

where $\|\cdot\|_2$ is the Euclidean norm. The MSD is evaluated and used for comparison purposes between the different algorithms. The above expectation is evaluated over different experiments and the different nodes in the network. Due to lack of space, we only display the MSDs of μ_1 and μ_2 in Figure 3.

We compare our algorithm with the consensus-based method proposed in [12]. We include 100 consensus iterations in order for the discrete consensus filter to converge. We chose $\eta = {}^1/_N$ since the condition for convergence of the discrete consensus filter in [12] is that $\eta \leq {}^1/_{d_{\max}}$ where $d_{\max}$ is the maximum degree. In our comparison to the algorithm proposed in [13], we include a single round of averaging in the D-step and M-step respectively.

The MSD is computed by averaging over 100 networks and random initializations around the true mean values. The initial covariance matrices are all assumed to be the identity matrix. The initial components weights are assumed to be uniform (i.e. $\alpha_c = {}^1/_2$ for

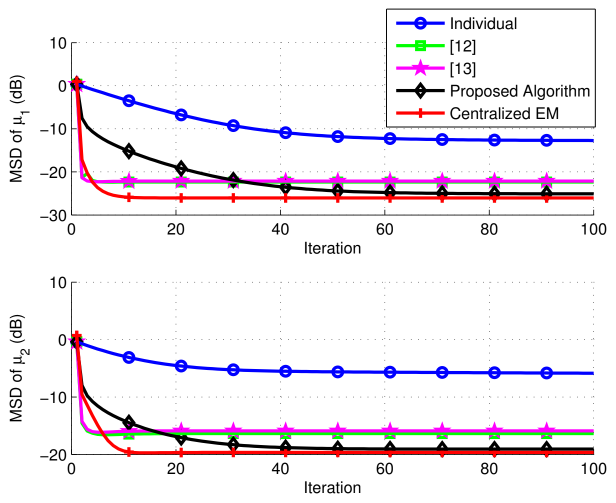

Fig. 3. MSD for the means of the distribution. MSD is averaged over 100 networks and random initializations around the true parameters.

$c = \{1, 2\}$). We note that the "Centralized EM" solution, the solution from [12], and the solution from [13] explicitly compute the the minimum of the expectation at each iteration for the Gaussian-mixture-model. This fact explains their faster convergence. In the proposed scheme, however, we use an iterative procedure to step in the direction of the minimum at every stage. While this may not be necessary in the Gaussian case, this iterative procedure is useful when it is not possible to compute the minimum analytically in closed form. In other words, the proposed scheme is not restricted to Gaussian mixture models, and it is seen to lead to lower mean-square-error even the Gaussian case. Additionally, the "Centralized EM" curve shows the performance of running EM at a centralized node where all the observations are available. This is not the case in the proposed work and [12, 13] where the raw data is not exchanged but only the estimates of the parameters of the generating distribution are exchanged within a node's neighborhood–no communication occurs past a single hop. "Individual" illustrates that if the nodes do not cooperate and rely only on their private observations, then the estimation of the distribution parameters suffers significantly.

5.2. The "Iris" Data Set

The Iris dataset contains 150 instances for three classes (evenly divided). We give the first 25 points of the data to each node as training. The training-data per node is corrupted by zero-mean white Gaussian noise with power σ_v^2 independently and uniformly distributed in the interval $(0, 4)$ for each node. Each class is fitted with a single mixture component since at the training state, the class labels are known to the classifiers. After the training stage, the entire noise-free dataset is used to generate the confusion matrix. The confusion matrix indicates what proportion of data points from class X are classified as class Y. Ideally, the confusion matrix of a classifier will be the identity matrix. Here we display the network average confusion matrix from non-cooperative and diffusion classifiers. We average the confusion matrices over 100 experiments and $N = 20$ nodes. The step-size for the algorithms is chosen to be $\gamma_k = 0.01$ and 500 iterations are used to ensure convergence. The classification for the diffusion-trained classifiers is done individually. The diffusion-based algorithm presented in this paper shows improvement in the classification of each of the three classes when compared to the non-cooperative classifier.

Table 1. Confusion matrix for non-cooperative classification. Cell $(1, 2)$ indicates a Setosa sample being mis-classified as Versicolor

	Setosa	Versicolor	Virginica
Setosa	0.9978	0.0004	0.0018
Versicolor	0.0063	0.8751	0.1187
Virginica	0	0.2563	0.7437

Table 2. Confusion matrix for diffusion-based training. Cell $(1, 2)$ indicates a Setosa sample being mis-classified as Versicolor

	Setosa	Versicolor	Virginica
Setosa	1	0	0
Versicolor	0	0.9784	0.0216
Virginica	0	0.1959	0.8041

6. REFERENCES

[1] V. V. Prakash and A. O'Donnell, "Fighting spam with reputation systems," *Queue*, vol. 3, pp. 36–41, November 2005.

[2] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7-2, pp. 3122–3136, July 2008.

[3] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1865–1877, May 2008.

[4] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1035–1048, March 2010.

[5] ——, "Diffusion strategies for distributed Kalman filtering and smoothing," *IEEE Transactions on Automatic Control*, vol. 55, no. 9, pp. 2069–2084, September 2010.

[6] ——, "Distributed detection over adaptive networks using diffusion adaptation," *Signal Processing, IEEE Transactions on*, vol. 59, no. 5, pp. 1917 –1932, May 2011.

[7] 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)*, vol. 39, no. 1, pp. 1–38, 1977.

[8] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[9] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 4th ed. Burlington, MA: Academic Press, 2008.

[10] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using gaussian mixture speaker models," *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 1, pp. 72–83, 1995.

[11] H. Permuter and J. Francos, "Gaussian mixture models of texture and colour for image database retrieval," in *Proc. IEEE ICASSP*, vol. 3, Hong Kong, April 2003, pp. 569–572.

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

[13] 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, June 2011.

[14] E. Anderson, "The irises of the gaspe peninsula," *Bulletin of the American Iris Society*, vol. 59, p. 25, 1935.

[15] C. Wu, "On the convergence properties of the EM algorithm," *The Annals of Statistics*, vol. 11, pp. 95–103, 1983.

[16] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.