A Local Scalable Distributed Expectation Maximization Algorithm for Large Peer-to-Peer Networks

Kanishka Bhaduri
MCT Inc., NASA Ames Research Center
MS 269-2, Moffett Field, CA 94035
Kanishka.Bhaduri-1@nasa.gov

Ashok N. Srivastava
NASA Ames Research Center
MS 269-2, Moffett Field, CA 94035
Ashok.N.Srivastava@nasa.gov

***Abstract*—This paper describes a local and distributed expectation maximization algorithm for learning parameters of Gaussian mixture models (GMM) in large peer-to-peer (P2P) environments. The algorithm can be used for a variety of well-known data mining tasks in distributed environments such as clustering, anomaly detection, target tracking, and density estimation to name a few, necessary for many emerging P2P applications in bioinformatics, webmining and sensor networks. Centralizing all or some of the data to build global models is impractical in such P2P environments because of the large number of data sources, the asynchronous nature of the P2P networks, and dynamic nature of the data/network. The proposed algorithm takes a two-step approach. In the monitoring phase, the algorithm checks if the model 'quality' is acceptable by using an efficient local algorithm. This is then used as a feedback loop to sample data from the network and rebuild the GMM when it is outdated. We present thorough experimental results to verify our theoretical claims.**

***Keywords*-peer-to-peer; local algorithms; expectation maximization**

I. Introduction

Expectation Maximization (EM) is a powerful statistical and data mining tool which can be used for a variety of tasks such as clustering, estimating parameters from the data in the presence of hidden variables, anomaly detection, target tracking and more. In 1977, Dempster *et al.* [1] presented the seminal work on EM and its application for estimating the parameters of a Gaussian Mixture Model (GMM). The authors showed that given a sample of data, there is a two step process which can estimate certain unknown parameters of the data by judiciously choosing hidden variables. The classical EM algorithm is well-understood and produces satisfactory estimates of the parameters when the data is centralized.

However, there exist emerging technologies where the data is not located at a central location but rather distributed across a large network of nodes or machines connected by an underlying communication infrastructure. The next generation Peer-to-Peer (P2P) networks such as Gnutella, BitTorrents, and the sensor networks offer some examples. A scalable and distributed EM algorithm developed for such networks can be deployed for a variety of tasks such as collaborative target tracking, clustering, density estimation, data compaction, and surveillance in wireless sensor networks [2][3].

To solve this problem, in this paper we develop an asynchronous P2P distributed (**PeDEM**) algorithm for monitoring and subsequent reactive updating of a GMM model. Our algorithm is provably correct *i.e.* given all the data, our algorithm will converge to the same result produced by a similar centralized algorithm. The algorithmic framework is *local*, in the sense that the computation and communication load at each node is independent of the size or the number of nodes of the network. This guarantees high scalability of the algorithm to possibly millions of nodes. The proposed methodology takes a two-step approach for building and maintaining GMM parameters in P2P networks. The first step is the *monitoring phase* in which, given an arbitrary estimate of the GMM parameters, our algorithm checks if they are valid with respect to the current data within user-specified thresholds. If not, this algorithm raises a flag, whereby we employ a convergecast-broadcast technique to rebuild the model parameters. This step is known as the *computation phase*. The specific contributions of this paper are as follows: (1) To the best of the authors' knowledge this is one of the first attempts on developing a completely asynchronous and local algorithm for *monitoring* the GMM parameters in P2P networks. (2) Besides this direct contribution, this paper shows how second order statistics can be directly monitored in a P2P network.

In the next section we present the work related to this research.

II. Related Work

Work related to this research can be subdivided into two major areas — distributed EM algorithms and computation in large distributed systems.

Distributed EM Algorithms: In standard EM algorithm, the task is to estimate some unknown parameters from the given data in the presence of some *unobserved* or *hidden* variables. Dempster *et al.* [1] proposed an iterative technique alternating between the **E**-step and **M**-step that solves this estimation problem. Most of the effort has been focused on efficiently computing the updates in the M-step (which are essentially averages) in a distributed fashion. Nowak [3] proposed a distributed EM (DEM) algorithm in which a ring topology is overlaid over the original network encompassing all the nodes and the updates are passed from one node to the next in sequence. Newscast EM [4], proposed by Kowalczyk and Vlassis uses gossip-style distributed computation to compute the parameters of the M-step. Using deterministic averaging technique, Gu [2] proposed an EM algorithm for GMM which uses graph Laplacian for asymptotic convergence. Wolfe *et al.* [5] developed a fully distributed EM algorithm based on MapReduce and junction tree topology. Unlike these algorithms, the technique proposed in this paper is local and hence offers excellent scalability.

Data Mining in Large Distributed (P2P) Systems: Several algorithms have been proposed for data mining in large P2P systems. *Probabilistic* algorithms such as the P2P k-Means algorithm by Bandyopadhyay *et al.* [6], the newscast model by Kowalczyk *et al.* [7], distributed inner product identification by Das *et al.* [8] compute the results within some error bounds. *Deterministic* algorithms produce exactly the same results compared to a centralized algorithm *e.g.* [9] and [10]. *Local* algorithms for P2P data mining include the majority voting and protocol developed by Wolff and Schuster [11], outlier detection [12], meta-classification [13], decision trees [14] and the generic local algorithms [15][16].

III. Preliminaries

In this section we present some background material necessary to understand our **PeDEM** algorithm that we have developed.

A. Expectation Maximization

EM [1] is an iterative optimization technique to estimate some unknown parameters Θ given some data $\mathbf{U}$. It is also assumed that there are some *hidden* variables $\mathbf{J}$. The EM algorithm iteratively alternates between two steps to maximize the posterior probability distribution of the parameters Θ given $\mathbf{U}$:

(1) **E-Step**: estimate the *E*xpected value of $\mathbf{J}$ given Θ, $\mathbf{U}$.

(2) **M-Step**: re-estimate Θ to *M*aximize the likelihood of $\mathbf{U}$, given the estimates of $\mathbf{J}$ found in the previous E-step.

For GMM, a multidimensional Gaussian mixture for a random vector $\overrightarrow{x} \in \mathbb{R}^d$ is defined as the weighted combination:

$$p(\overrightarrow{x}) = \sum_{s=1}^{k} \pi_s p(\overrightarrow{x}|s)$$

of k Gaussian densities where the s-th density is given by

$$p(\overrightarrow{x}|s) = \frac{1}{(2\pi)^{d/2}|\mathbf{C}_s|^{1/2}} exp\left[-(\overrightarrow{x} - \overrightarrow{\mu_s})^{\mathrm{T}}\mathbf{C}_s{}^{-1}(\overrightarrow{x} - \overrightarrow{\mu_s})/2\right]$$

each parameterized by its mean vector $\overrightarrow{\mu_s} = [\mu_{s.1}\mu_{s.2}\ldots\mu_{s.d}]^{\mathrm{T}}$ and covariance matrix $\mathbf{C}_s$=$(x - \mu_s)(x - \mu_s)^{\mathrm{T}}$. $\pi_s = p(s)$ is the prior probability that the s-th density generates a data point. Given n multi-dimensional samples $\mathbf{X} = \{\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}\}$, the task is to estimate the set of parameters by maximizing the log-likelihood of the parameters given the data:

$$\mathcal{L}(\Theta|\mathbf{X}) = \log \prod_{a=1}^{n} p(\overrightarrow{x_a}|\Theta) = \sum_{a=1}^{n} \log\left(\sum_{s=1}^{k} \pi_s \mathcal{N}(\overrightarrow{x_a}; \overrightarrow{\mu_s}, \mathbf{C}_s)\right)$$

Using EM for GMM, the E-step and the M-step can be written as:

E-step (estimate the contribution of each point):

$$q_{s,a} = \frac{\pi_s \mathcal{N}(\overrightarrow{x_a}; \overrightarrow{\mu_s}, \mathbf{C}_s)}{\sum_{r=1}^{k} \pi_r \mathcal{N}(\overrightarrow{x_a}; \overrightarrow{\mu_r}, \mathbf{C}_r)} \quad (1)$$

M-step (recompute the parameters):

$$\pi_s = \frac{\sum_{a=1}^{n} q_{s,a}}{n} \quad (2)$$

$$\overrightarrow{\mu_s} = \frac{\sum_{a=1}^{n} q_{s,a}\overrightarrow{x_a}}{\sum_{a=1}^{n} q_{s,a}} \quad (3)$$

$$\mathbf{C}_s = \frac{\sum_{a=1}^{n} q_{s,a}(\overrightarrow{x_a} - \overrightarrow{\mu}_s)(\overrightarrow{x_a} - \overrightarrow{\mu}_s)^{\mathrm{T}}}{\sum_{a=1}^{n} q_{s,a}} \quad (4)$$

where $\mathcal{N}(\overrightarrow{x_a}; \overrightarrow{\mu_s}, \mathbf{C}_s)$ denotes the pdf of a normal distribution with input $\overrightarrow{x_a}$, mean $\overrightarrow{\mu_s}$ and covariance $\mathbf{C}_s$. Note that the above computation needs to be carried out for all the k Gaussian components.

In the next few sections we shift our focus to distributed computation of these parameters and discuss some assumptions and necessary background material.

B. Notations and Assumptions

Let $V = \{P_1, \ldots, P_p\}$ be a set of peers connected via an underlying communication infrastructure. The set of P_i's neighbors, Γ_i, is known to P_i. Each peer communicates with its immediate neighbors (one hop neighbors) only. At time t, let $\mathcal{G}$ denote a collection of data tuples which have been generated from k Gaussian densities having unknown parameters and unknown mixing probabilities. The tuples are horizontally distributed over a large (undirected) network of machines (peers). The local data of peer P_i at time t is $S_i = [\overrightarrow{x_{i,1}}, \overrightarrow{x_{i,2}}, \ldots, \overrightarrow{x_{i,m_i}}]$, where $\overrightarrow{x_{i,j}} = [x_{i,j.1}x_{i,j.2}\ldots x_{i,j.d}]^{\mathrm{T}} \in \mathbb{R}^d$. Here m_i denotes the number of data tuples at P_i and d denotes the dimensionality of the data. The **global input** is denoted by $\mathcal{G} = \bigcup_{i=1,\ldots,p} S_i$.

We assume that communication among neighboring peers is reliable and ordered. These assumptions can be imposed using heartbeat mechanisms or retransmissions proposed elsewhere [15]. Furthermore, it is assumed that data sent from P_i to P_j is never sent back to P_i. One way of ensuring this is to assume that communication takes place over a communication tree – an assumption we make here.

C. Problem Formulation in P2P Scenario

When all the data is available at a central location, the update equations for the iterative EM algorithm are given by Equations 1–4. However, in the distributed setup, all the data is not available at a central location. Therefore, for any peer P_i, the log-likelihood per data point and the update equations for the EM algorithm, can be written as:

$$\overline{\mathcal{L}}(\Theta|\mathcal{G}) = \frac{\sum_{i=1}^{p}\sum_{a=1}^{m_i}\log\left(\sum_{s=1}^{k}\pi_s\mathcal{N}(\overrightarrow{x_{i,a}};\overrightarrow{\mu_s},\mathbf{C}_s)\right)}{\sum_{i=1}^{p}m_i} \tag{5}$$

E-step:

$$q_{i,s,a} = \frac{\pi_s\mathcal{N}\left(\overrightarrow{x_{i,a}};\overrightarrow{\mu_s},\mathbf{C}_s\right)}{\sum_{r=1}^{k}\pi_r\mathcal{N}\left(\overrightarrow{x_{i,a}};\overrightarrow{\mu_r},\mathbf{C}_r\right)} \tag{6}$$

M-step:

$$\pi_s = \frac{\sum_{i=1}^{p}\sum_{a=1}^{m_i}q_{i,s,a}}{\sum_{i=1}^{p}m_i} \tag{7}$$

$$\overrightarrow{\mu_s} = \frac{\sum_{i=1}^{p}\sum_{a=1}^{m_i}q_{i,s,a}\overrightarrow{x_{i,a}}}{\sum_{i=1}^{p}\sum_{a=1}^{m_i}q_{i,s,a}} \tag{8}$$

$$\mathbf{C}_s = \frac{\sum_{i=1}^{p}\sum_{a=1}^{m_i}q_{i,s,a}(\overrightarrow{x_{i,a}}-\overrightarrow{\mu}_s)(\overrightarrow{x_{i,a}}-\overrightarrow{\mu}_s)^{\mathrm{T}}}{\sum_{i=1}^{p}\sum_{a=1}^{m_i}q_{i,s,a}} \tag{9}$$

where the sum is taken over all peers' data. Note that computation in the E-step is entirely local to a peer. However, for the log-likelihood and the M-step, a peer needs information from all the nodes in the network in order to recompute the parameters. In this paper, we consider a monitoring version of this problem: *Given a time-varying data set and pre-computed initial values of these parameters (built from a centralized or sampled data) to all peers, do these parameters describe the union of all the data held by all the peers in terms of an appropriately low log-likelihood function?*

Our goal is to develop a framework under which each peer (1) checks if the current parameters of the GMM are up-to-date with respect to $\mathcal{G}$, and (2) recomputes the models whenever deemed unfit. For a centralized EM with static data, convergence occurs when the difference of the log-likelihood in two subsequent iterations becomes a constant. However, for our distributed setup, since we are considering dynamic data, we relax this criteria and consider a solution to be admissible when it is within a user-defined threshold ϵ of its true value. For the monitoring problem, let $\widehat{\Theta} = \{\widehat{\pi_s},\widehat{\overrightarrow{\mu_s}},\widehat{\mathbf{C}_s},\dots\}$, denote the parameters that were calculated offline based on some past data, and disseminated to all the peers. The monitoring problem is to check (1) if $\overline{\mathcal{L}}(\widehat{\Theta}|\mathcal{G}) < \epsilon$, or (2) if these parameters are valid with respect to the current data of all the peers. Below is a formal problem definition.

Problem Definition: Given a time varying dataset S_i, user-defined thresholds ϵ, ϵ_1, ϵ_2, and ϵ_3, and pre-computed estimates $\widehat{\Theta}$, for each Gaussian distribution, the monitoring problem is to check if:

— either $\overline{\mathcal{L}}(\widehat{\Theta}|\mathcal{G}) < \epsilon$, or

— $\left[|\pi_s - \widehat{\pi_s}| < \epsilon_1, \left|\left|\overrightarrow{\mu_s} - \widehat{\overrightarrow{\mu_s}}\right|\right|^2 < \epsilon_2, \left|\|\mathbf{C_s}\|_F - \widehat{\mathbf{C}_s}\right| < \epsilon_3\right] \forall s$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. One needs to either threshold the log-likelihood of the data or monitor each parameter separately. We discuss the tradeoff of these situations later.

D. Monitoring Functions

As a building block of **PeDEM**, we use an efficient, provably correct, and local algorithm for monitoring functions of average vectors in $\mathbb{R}^d$, where the vectors are distributed in a P2P network.

Peers communicate with one another by sending sets of points in $\mathbb{R}^d$ or statistics as defined later in this section. Let $X_{i,j}$ denote the last set of points sent by peer P_i to P_j. Assuming reliable messaging, once a message is delivered both P_i and P_j know $X_{i,j}$ and $X_{j,i}$. There are three sets of vectors which are crucial to the monitoring algorithm. The **knowledge** of P_i is $\mathcal{K}_i = S_i \bigcup_{P_j\in\Gamma_i} X_{j,i}$. $\mathcal{K}_i$ can also be initialized using combinations of vectors defined on S_i (instead of only S_i) as we will present in the next section. The **agreement** of P_i with any of its neighbors P_j is $\mathcal{A}_{i,j} = X_{i,j} \cup X_{j,i}$. The **withheld knowledge** of P_i with respect to a neighbor P_j is $\mathcal{W}_{i,j} = \mathcal{K}_i \setminus \mathcal{A}_{i,j}$.

In the next section we present a theorem which shows how we can convert this monitoring problem into a geometric problem for an efficient solution. For this we need to split the domain into convex regions since the stopping condition we describe later (Theorem IV.1) relies on this. The following definition states the properties of these convex regions.

Definition III.1. *A collection of non-overlapping regions $\mathcal{R}_\mathcal{F} = \{R_1, R_2, \dots, R_\ell, T\}$ is a **cover of region** $\mathbb{R}^d$, invariant with respect to a function $\mathcal{F} : \mathbb{R}^d \to \mathbb{O}$ (where $\mathbb{O}$ is an arbitrary range), if (1) every $R_i \in \mathcal{R}_\mathcal{F}$ (except T) is convex, (2) $\mathcal{F}$ is invariant in R_i* i.e., *$\forall(x,y) \in R_i, \mathcal{F}(x) = \mathcal{F}(y)$, and (3) $\mathcal{T}$ denotes the area of the domain, not encompassed by $\bigcup_{i=1}^{\ell} R_i$, known as the* tie *region.*

Finally, for any $\overrightarrow{x} \in \mathbb{R}^d$, let $\mathcal{R}_\mathcal{F}(\overrightarrow{x})$ denote the first region in $\mathcal{R}_\mathcal{F}$ which includes $\overrightarrow{x}$. The precise specification of the convex regions will depend on the definition of $\mathcal{F}$. Since these sets can be arbitrarily large, below we define sufficient statistics on each set which are far more efficient to communicate.

Set Statistics: For each set, define two statistics: (1) the *average* which is the average of all the points in the respective sets (*e.g.* $\overline{S_i}$, $\overline{\mathcal{K}_i}$, $\overline{\mathcal{A}_{i,j}}$, $\overline{\mathcal{W}_{i,j}}$, $\overline{X_{i,j}}$, $\overline{X_{j,i}}$ and $\overline{\mathcal{G}}$), and (2) the *weights* of the sets denoted by $\omega(S_i)$, $\omega(X_{i,j})$, $\omega(X_{j,i})$, $\omega(\mathcal{K}_i)$, $\omega(\mathcal{A}_{i,j})$, $\omega(\mathcal{W}_{i,j})$, and $\omega(\mathcal{G})$. Each peer communicates these two statistics for each set. We can write the following expressions for the weights and the average vectors of each set:

Knowledge

— $\omega(\mathcal{K}_i) = \omega(S_i) + \sum_{P_j\in\Gamma_i}\omega(X_{j,i})$

— $\overline{\mathcal{K}_i} = \frac{\omega(S_i)}{\omega(\mathcal{K}_i)}\overline{S_i} + \sum_{P_j\in\Gamma_i}\frac{\omega(X_{j,i})}{\omega(\mathcal{K}_i)}\overline{X_{j,i}}$

Agreement

— $\omega(\mathcal{A}_{i,j}) = \omega(X_{i,j}) + \omega(X_{j,i})$

— $\overline{\mathcal{A}_{i,j}} = \frac{\omega(X_{i,j})}{\omega(\mathcal{A}_{i,j})}\overline{X_{i,j}} + \frac{\omega(X_{j,i})}{\omega(\mathcal{A}_{i,j})}\overline{X_{j,i}}$

Withheld

— $\omega(\mathcal{W}_{i,j}) = \omega(\mathcal{K}_i) - \omega(\mathcal{A}_{i,j})$

— $\overline{\mathcal{W}_{i,j}} = \frac{\omega(\mathcal{K}_i)}{\omega(\mathcal{W}_{i,j})}\overline{\mathcal{K}_i} - \frac{\omega(\mathcal{A}_{i,j})}{\omega(\mathcal{W}_{i,j})}\overline{\mathcal{A}_{i,j}}$

Note that these computations are local to a peer. The general methodology for computing $\mathcal{F}(\overline{\mathcal{G}})$ requires us to cover the domain of $\mathcal{F}$ using non-overlapping convex regions. In the next section we present a stopping condition for the peers to converge to the correct result.

IV. Globally Correct Termination Criteria

The goal of the monitoring algorithm is to raise a flag whenever the estimates of the parameters are no longer valid with respect to $\mathcal{G}$. The EM monitoring algorithm guarantees eventual correctness: once computation terminates, each peer computes the correct result compared to a centralized setting. The following theorem allows a peer to stop sending messages and achieve a correct termination state *i.e.* decide if $\mathcal{F}(\overline{\mathcal{G}}) > \epsilon$ or $< \epsilon$ solely based on $\overline{\mathcal{K}_i}$, $\overline{\mathcal{A}_{i,j}}$, and $\overline{\mathcal{W}_{i,j}}$.

Theorem IV.1. [Termination Criteria] *Let $P_1, \ldots, P_n$ be a set of peers connected to each other via a spanning tree $G(V, E)$. Let $\mathcal{G}$, $\mathcal{K}_i$, $\mathcal{A}_{i,j}$, and $\mathcal{W}_{i,j}$ be as defined in the previous section. Let R be any region in $\mathcal{R}_{\mathcal{F}}$. If at time t no messages traverse the network, and for each P_i, $\overline{\mathcal{K}_i} \in R$ and for every $P_j \in \Gamma_i$, $\overline{\mathcal{A}_{i,j}} \in R$ and either $\overline{\mathcal{W}_{i,j}} \in R$ or $\mathcal{W}_{i,j} = \emptyset$, then $\overline{\mathcal{G}} \in R$.*

Proof: Consider a network $G(V, E)$. Let us select a leaf peer P_i for which $\overline{\mathcal{K}_i} \in R$ and for every $P_j \in \Gamma_i$, $\overline{\mathcal{A}_{i,j}} \in R$ and either $\overline{\mathcal{W}_{i,j}} \in R$ or $\mathcal{W}_{i,j} = \emptyset$. Let us eliminate P_i by sending all of its withheld knowledge $\mathcal{W}_{i,j}$ to $P_j \in \Gamma_i$. Now, the new knowledge of P_j becomes $\mathcal{K}_j^{'} = \mathcal{K}_j \cup \mathcal{W}_{i,j}$. Since G is a tree, $\mathcal{K}_j \cap \mathcal{W}_{i,j} = \emptyset$. Therefore, for some $\eta \in [0, 1]$, $\overline{\mathcal{K}_j^{'}} = \eta \cdot \overline{\mathcal{K}_j} + (1 - \eta) \cdot \overline{\mathcal{W}_{i,j}}$. Since both $\overline{\mathcal{K}_j}$ and $\overline{\mathcal{W}_{i,j}} \in R$, $\overline{\mathcal{K}_j^{'}}$ in R too. Using a similar argument on the withheld knowledge of P_j, we can show that it also belongs to R as a result of this unification. Continuing, we will be left with one peer P_1 whose $\overline{\mathcal{K}_1} \in R$. Also under the said unification, $\mathcal{K}_1 = \mathcal{G}$ since no data was lost. Thus, $\overline{\mathcal{G}} \in R$. ■

The above theorem allows a peer to stop the communication and output $\mathcal{F}(\overline{\mathcal{K}_i})$ which will eventually become equal to $\mathcal{F}(\overline{\mathcal{G}})$. A peer can avoid communication even if its local data changes or the network changes as long as the result of the theorem is satisfied. Indeed, if the result of the theorem holds for every peer, and all messages have been delivered, then Theorem IV.1 guarantees this is the correct solution. Otherwise, if there exists one peer P_z for which the condition does not hold, then either of these two things will happen: (1) a message will eventually be received by P_z or, (2) P_z will send a message. In either of these two cases, $\overline{\mathcal{K}_z}$ will change thereby guaranteeing globally correct convergence.

V. Monitoring GMM Parameters

We present the monitoring of the log-likelihood of the data and the three parameters given in Equations 7–9 in the next few sections. Note that in practice we need to monitor either the log-likelihood of the data or the GMM parameters.

A. Monitoring log-likelihood

Monitoring the average log likelihood of the global data is equivalent to checking if the following quantity is less ϵ: $\overline{\mathcal{L}}(\widehat{\Theta}|\mathcal{G}) = \frac{\sum_{i=1}^{p} \mathcal{L}_i(\widehat{\Theta}|S_i)}{\sum_{i=1}^{p} m_i} < \epsilon$, where $\mathcal{L}_i(\widehat{\Theta}|S_i)$ is the log-likelihood of all the points available at peer P_i. Thus each peer has a number in $\mathbb{R}$, and the goal is to check if the average of those numbers is greater than ϵ. This can be done using the framework presented in Section III-D. Each peer checks if its knowledge is in $[0, \epsilon)$ or (ϵ, ∞). We discuss the details of how this is done in Section V. Therefore, for monitoring $\overline{\mathcal{L}}(\widehat{\Theta}|\mathcal{G})$, the following initializations need to be carried out:

(1) $M^{\mathcal{L}}.S_i = \left\{ \log\left(\sum_{s=1}^{k} \pi_s \mathcal{N}(\overrightarrow{x_{i,1}}; \overrightarrow{\mu_s}, \mathbf{C}_s) \right), \ldots \right\}$

(2) $\mathcal{R}_{\mathcal{F}} = \{[0, \epsilon), (\epsilon, \infty)\}$.

In this case $\mathcal{T} = \{\epsilon\}$. Since monitoring π_s is the same as $\overline{\mathcal{L}}(\widehat{\Theta}|\mathcal{G})$, we do not present it here due to shortage of space.

B. Monitoring $\overrightarrow{\mu_s}$

Following a similar argument, monitoring $\overrightarrow{\mu_s}$ is equivalent to thresholding the following quantity:

$$Err(\overrightarrow{\mu_s}) \quad = \quad \left|\left|\overrightarrow{\mu_s} - \widehat{\overrightarrow{\mu}}_s\right|\right| = \left|\left| \frac{\sum_{i=1}^{p} \sum_{a=1}^{m_i} q_{i,s,a} \left[\overrightarrow{x_{i,a}} - \widehat{\overrightarrow{\mu}}_s \right]}{\sum_{i=1}^{p} \sum_{a=1}^{m_i} q_{i,s,a}} \right|\right| < \epsilon_2$$

where the error vector is the average of $q_{i,s,a} \left[\overrightarrow{x_{i,a}} - \widehat{\overrightarrow{\mu}}_s \right]$ across all the peers. However the average in this case is not taken with respect to the number of tuples in the dataset S_i, but rather over all the $q_{i,s,a}$'s. As a result, we set $|S_i| = \sum_{a=1}^{m_i} q_{i,s,a}$. Moreover, for this problem in $\mathbb{R}^2$, the geometric interpretation to the monitoring problem is to check if the L2-norm of the vector difference between $\overrightarrow{\mu_s} - \widehat{\overrightarrow{\mu}}_s$ lies inside a circle of radius ϵ_2. Geometrically, the area in which $Err(\overrightarrow{\mu_s}) < \epsilon_2$, is inside the sphere and hence is already convex in $\mathbb{R}^d$. However, outside of the sphere is not convex. Hence random tangent lines are drawn on the surface of the sphere by choosing points $\widehat{u_1}, \ldots, \widehat{u_r}$ on the sphere (the same points across all peers). Each of these half-spaces is convex. To check if an arbitrary point $\overrightarrow{z}$ is inside the sphere, a peer simply checks if $\|\overrightarrow{z}\| < \epsilon_2$. To check if it is outside, a peer selects the first point $\widehat{u_i}$ such that $\overrightarrow{z} \cdot \widehat{u_i} > \epsilon_2$. The following denotes the initialization necessary for this instance of the problem M^{μ_s}: $M^{\mu_s}.\overline{S_i} = \frac{\sum_{a=1}^{m_i} q_{i,s,m_i} \left[\overrightarrow{x_{i,a}} - \widehat{\overrightarrow{\mu}}_s \right]}{\sum_{a=1}^{m_i} q_{i,s,a}}$,

$M^{\mu_s}.\omega(S_i) = \sum_{a=1}^{m_i} q_{i,s,a}$,

$\mathcal{R}_{\mathcal{F}} = \underbrace{\{\overrightarrow{z} \in \mathbb{R}^d : ||\overrightarrow{z}|| < \epsilon_2\}}_{R_{in}} \bigcup_{i=1}^{r} \underbrace{\{\overrightarrow{z} \in \mathbb{R}^d : \overrightarrow{z} \cdot \widehat{u_i} > \epsilon_2\}}_{R_1,\ldots,R_r}$.

In this case $\mathcal{T}$ denotes the area between the polygon formed by the half-spaces and the circle.

C. Monitoring C_s

The last parameter that we need to monitor is the covariance matrix $\mathbf{C}_s$. A natural extension of the L2-norm in this case is the Frobenius norm. Let $\overrightarrow{y_{i,a}} = \overrightarrow{x_{i,a}} - \overrightarrow{\mu_s}$. It can be shown that,

$$\|\mathbf{C}_s\|_F^2 \quad \leq \quad \left(\frac{\sum_{i=1}^{p} \sum_{a=1}^{m_i} q_{i,s,a} \left\{ y_{i,a.1}^2 + \cdots + y_{i,a.d}^2 \right\}}{\sum_{i=1}^{p} \sum_{a=1}^{m_i} q_{i,s,a}} \right)^2$$

By taking the square root and re-substituting $\overrightarrow{y_{i,a}}$, we get,

$$C_s^n \quad = \quad \sum_{k=1}^{d} \left\{ \frac{\sum_{i=1}^{p} \sum_{a=1}^{m_i} q_{i,s,a} x_{i,a.k}^2}{\sum_{i=1}^{p} \sum_{a=1}^{m_i} q_{i,s,a}} - \left(\frac{\sum_{i=1}^{p} \sum_{a=1}^{m_i} q_{i,s,a} x_{i,a.k}}{\sum_{i=1}^{p} \sum_{a=1}^{m_i} q_{i,s,a}} \right)^2 \right\}$$

We need to check if $\|\mathbf{C}_s\|_F - \widehat{C_s} < \epsilon_3$ *i.e.* $\|\mathbf{C}_s\|_F < \widehat{C_s} + \epsilon_3$. Since $\|\mathbf{C}_s\|_F$ is not a convex function, but C_s^n is, we monitor the latter one instead. Note that, $C_s^n < \epsilon_3 \Rightarrow \|\mathbf{C}_s\|_F < \epsilon_3$. $C_s^n > \epsilon_3 \not\Rightarrow \|\mathbf{C}_s\|_F > \epsilon_3$. Therefore, using C_s^n for thresholding is more conservative: in the worst case we will have more false alerts for building new models but no false dismissals.

Let $Err(C_s^n) = C_s^n - \widehat{C_s}$. Let $g : \mathbb{R}^{2d} \rightarrow \mathbb{R}$ be defined as follows: $\forall (s_1, \ldots, s_{2d}) \in \mathbb{R}$, $g(s_1, \ldots, s_{2d}) = \sum_{i=1}^{d} s_i - \sum_{i=d+1}^{2d} s_i^2 - \widehat{C_s} - \epsilon_3$. We have the following key result:

$$Err(C_s^n) < \epsilon_3 \Leftrightarrow$$

$$g \left(\frac{\sum_{i=1}^{p} \sum_{a=1}^{m_i} q_{i,s,a} \left(x_{i,a.1}^2, \ldots, x_{i,a.d}^2, x_{i,a.1}, \ldots, x_{i,a.d} \right)}{\sum_{i=1}^{p} \sum_{a=1}^{m_i} q_{i,s,a}} \right) < 0.$$

Each peer can locally compute the 2-d dimensional vector $q_{i,s,a} \left(x_{i,a.1}^2, \ldots, x_{i,a.d}^2, x_{i,a.1}, \ldots, x_{i,a.d} \right)$. Then the goal is zero-thresholding g applied to the average of local vectors. Taking the Hessian of $-g$ it can be easily shown that $-g$ is convex. The outside of g can be decomposed into convex regions using tangent lines placed at random points $\widehat{u_1}, \ldots, \widehat{u_z}$ on g. Checking if $g(\overline{\mathcal{K}_i}) < \epsilon_3$ is equivalent to checking if $\overline{\mathcal{K}_i}$ lies inside g. If not, we find the first point $\widehat{u_i}$ such that $\overline{\mathcal{K}_i} \cdot \widehat{u_i} > \|\widehat{u_i}\|$. We then apply the theorem for half space defined by $\widehat{u_i}$.

Now, since $Err(C_s^n)$ can be both positive or negative, we need to check if $|Err(C_s^n)| < \epsilon_3$. Therefore, we need two monitoring instances denoted by $M_1^{C_s}$ and $M_2^{C_s}$. The following denotes the datasets and convex regions for this monitoring problem.

$M_1^{C_s}.\overline{S_i} = \frac{\sum_{a=1}^{m_i} q_{i,s,a} \left(x_{i,a.1}^2, \ldots, x_{i,a.d}^2, x_{i,a.1}, \ldots, x_{i,a.1} \right)}{\sum_{a=1}^{m_i} q_{i,s,a}}$,
$M_1^{C_s}.\omega(S_i) = \sum_{a=1}^{m_i} q_{i,s,a}$,
$M_2^{C_s}.\overline{S_i} = -M_1^{C_s}.\overline{S_i}$, $M_2^{C_s}.\omega(S_i) = \sum_{a=1}^{m_i} q_{i,s,a}$
$\mathcal{R}_{\mathcal{F}} = \left\{ \overrightarrow{z} \in \mathbb{R}^{2d} : g(z) < 0 \bigcup_{i=1}^{z} \overrightarrow{z} \cdot \widehat{u_i} > \|\widehat{u_i}\| \right\}$.

D. Algorithm

Having discussed each of the monitoring problems, we are now in a position to present the algorithms for monitoring the parameters. Using log-likelihood, we need to instantiate only one monitoring problem. However, using the parameters themselves, we need to monitor the prior, mean, and covariance for each Gaussian $s \in \{1, \ldots, k\}$.

In order to use Theorem IV.1 for developing a monitoring algorithm, the following steps must be followed: (1) specify the input to the algorithm (*i.e.* S_i), and (2) specify the cover *i.e.* $\mathcal{R}_{\mathcal{F}}$. For each of the monitoring problems, these are already specified in the previous sections. Algorithm 1 presents the pseudo-code for monitoring $\overline{\mathcal{L}}(\widehat{\Theta}|\mathcal{G})$. The other computations can be performed in a similar fashion.

```
Input: $\epsilon$, $\mathcal{R}_{\mathcal{F}}$, $S_i$, $\Gamma_i$, $L$ and $\widehat{\Theta}$
Output: Set
$flag^{\mathcal{L}} = \begin{cases} 1 & \text{if } M^{\mathcal{L}}.\overline{\mathcal{K}_i} > \epsilon \\ 0 & \text{otherwise} \end{cases}$
Initialization: Initialize $M^{\mathcal{L}}$
On any Event:
begin
    forall $P_j \in \Gamma_i$ do
        if $\mathcal{R}_{\mathcal{F}}\left(M^{\mathcal{L}}.\overline{\mathcal{K}_i}\right) = T$ then
            $M^{\mathcal{L}}.\omega\left(X_{i,j}\right) \leftarrow M^{\mathcal{L}}.\omega\left(\mathcal{K}_i\right) - M^{\mathcal{L}}.\omega\left(X_{j,i}\right)$;
            $M^{\mathcal{L}}.\overline{X_{i,j}} \leftarrow \frac{M^{\mathcal{L}}.\omega(\mathcal{K}_i) M^{\mathcal{L}}.\overline{\mathcal{K}_i} - M^{\mathcal{L}}.\omega\left(X_{j,i}\right) M^{\mathcal{L}}.\overline{X_{j,i}}}{M^{\mathcal{L}}.\omega\left(X_{j,i}\right)}$;
        end
        if $\left(M^{\mathcal{L}}.\overline{\mathcal{A}_{i,j}} \notin \mathcal{R}_{\mathcal{F}}\left(M^{\mathcal{L}}.\overline{\mathcal{K}_i}\right)\right)$
        $\bigvee \left(M^{\mathcal{L}}.\overline{\mathcal{W}_{i,j}} \notin \mathcal{R}_{\mathcal{F}}\left(M^{\mathcal{L}}.\overline{\mathcal{K}_i}\right)\right)$
        $\bigvee \left(M^{\mathcal{L}}.\omega\left(\mathcal{W}_{i,j}\right) = 0 \bigwedge M^{\mathcal{L}}.\overline{\mathcal{A}_{i,j}} \neq M^{\mathcal{L}}.\overline{\mathcal{K}_i}\right)$ then
            Compute new $M^{\mathcal{L}}.\omega\left(X_{j,i}\right)$ and $M^{\mathcal{L}}.\overline{X_{j,i}}$ such
            that $M^{\mathcal{L}}.\overline{\mathcal{A}_{j,i}}, M^{\mathcal{L}}.\overline{\mathcal{W}_{j,i}} \in \mathcal{R}_{\mathcal{F}}(M^{\mathcal{L}}.\overline{\mathcal{K}_i})$
        end
        if $CurrTime - LastMsgSent > L$ then
            SendMsg($M^{\mathcal{L}}.\overline{X_{i,j}}, M^{\mathcal{L}}.\omega(X_{i,j})$) to $P_j$
        end
        else Wait $(L - (CurrTime - LastMsgSent))$
        time and check conditions again
    end
end
On MessageRecvd$\left(\overline{X}, \omega(X)\right)$ from $P_j$:
begin
    $M^{\mathcal{L}}.\overline{X_{j,i}} \leftarrow \overline{X}$;
    $M^{\mathcal{L}}.\omega\left(X_{j,i}\right) \leftarrow \omega(X)$;
    Recompute $M^{\mathcal{L}}.\overline{\mathcal{K}_i}, M^{\mathcal{L}}.\overline{\mathcal{A}_{i,j}}, M^{\mathcal{L}}.\overline{\mathcal{W}_{i,j}}$;
end
```

Algorithm 1: Pseudo code for monitoring $\overline{\mathcal{L}}(\widehat{\Theta}|\mathcal{G})$ for any peer P_i.

For any peer P_i, the input to the algorithm are ϵ, $\mathcal{R}_{\mathcal{F}}$, S_i, Γ_i, L and $\widehat{\Theta}$ (we describe L later). The output of each monitoring instance is a flag which is set if the corresponding $\overline{\mathcal{K}_i}$ exceeds the threshold *e.g.* if $M_1^{\pi_s}.\overline{\mathcal{K}_i} > \epsilon_1$ or $\left\|M^{\mu_s}.\overline{\mathcal{K}_i}\right\| > \epsilon_2$ or $g\left(M_1^{C_s}.\overline{\mathcal{K}_i}\right) > 0$. In the initialization phase, it initializes its local statistics $\overline{\mathcal{K}_i}$, $\overline{\mathcal{A}_{i,j}}$ and $\overline{\mathcal{W}_{i,j}}$ according to the equations in Section III-D. The algorithm is entirely event driven. Events can be one of the following: a change in local data S_i, message received or a change in the set of neighbors Γ_i. If one of these things happen, a peer checks if the conditions of Theorem IV.1 are satisfied. First peer P_i finds the active region: the region $R \in \mathcal{R}_{\mathcal{F}}$ in which $\overline{\mathcal{K}_i}$ lies *i.e.* $R = \mathcal{R}_{\mathcal{F}}(\overline{\mathcal{K}_i})$. If, $R = T$, *i.e.* the knowledge lies in the tie region, the condition of the theorem does not guarantee a solution and hence the only correct solution is flooding all of its data. On the contrary, if $\forall\ P_j \in \Gamma_i$, both $\overline{\mathcal{A}_{i,j}}, \overline{\mathcal{W}_{i,j}} \in R$, P_i does nothing and can rely on the result of the theorem for correctness. If $\overline{\mathcal{A}_{i,j}} \notin R$ or $\overline{\mathcal{W}_{i,j}} \notin R$, the result of the theorem dictates P_i to send a message to P_j. Other than these two cases, a peer need not send any message even if its local data has changed.

Message sending is performed by the **SendMsg** method. When $R = T$, the peer has to flood whatever knowledge it has. Thus it sets $\overline{X_{i,j}}$ and $\omega(X_{i,j})$ equals to its knowledge minus what it had received from P_j previously. It then sends this to P_j. However, when $R \neq T$, a peer can refrain from sending all data. This technique of sending all the data has adverse effects on the communication in a dynamic data scenario. This is because if a peer communicates all of its data, and the data changes again later, the change is far more noisy than the original data. So we always set $\overline{X_{i,j}}$ and $\omega(X_{i,j})$ such that some data is retained while still maintaining the conditions of the theorem. We do this by checking with an exponentially decreasing set of values of $\omega(\mathcal{W}_{i,j})$ until either all $\overline{\mathcal{K}_i}$, $\overline{\mathcal{A}_{i,j}}$ and $\overline{\mathcal{W}_{i,j}} \in R$, or $\omega(\mathcal{W}_{i,j})$=0. If the latter happens, it means that a peer cannot have any withheld knowledge and it has to send all of its knowledge in order to satisfy the conditions of the theorem. Lastly, when P_i receives a message ($\overline{X}$ and $\omega(X)$) from P_j, it sets $\overline{X_{j,i}} \leftarrow \overline{X}$ and $\omega(X_{j,i}) \leftarrow \omega(X)$ and checks the conditions of Theorem IV.1 again.

To prevent message explosion, in our event-based system we employ a "leaky bucket" mechanism which ensures that no two messages are sent in a period shorter than a constant L. This technique is not new but has been used earlier [14]. Note that this mechanism does not enforce synchronization or affect correctness; at most it might delay convergence. We explore its effect thoroughly in our experiments. Next we discuss the correctness of **PeDEM**.

Correctness and Complexity Analysis: For **PeDEM**, computation will continue for each node unless one of the following happens: (1) for every node, $\overline{\mathcal{K}_i} = \overline{\mathcal{G}}$ or, (2) for every P_i and every neighbor P_j, $\overline{\mathcal{K}_i}$, $\overline{\mathcal{A}_{i,j}}$,and $\overline{\mathcal{W}_{i,j}} \in R$. In the former case, obviously $\mathcal{F}(\overline{\mathcal{K}_i}) = \mathcal{F}(\overline{\mathcal{G}})$. In the latter case, Theorem IV.1 dictates that $\overline{\mathcal{G}} \in R$. Therefore, in either of the cases $\mathcal{F}(\overline{\mathcal{K}_i}) = \mathcal{F}(\overline{\mathcal{G}})$.

Determining the communication complexity of local algorithms in dynamic environments is still an open research issue. Researches have proposed definitions of locality [15][8]. For **PeDEM**, the worst case communication complexity is bounded by $O(|V|^2)$ when $\overline{\mathcal{G}} = \epsilon$. Since this does not happen often in practice, local algorithms exhibit good scalability (in most cases independent of $|V|$) as shown in our experiments.

In the next section we describe how we can use this monitoring algorithm in closed loop.

VI. Computing New Models

The monitoring algorithm presented in the previous section generates an alert whenever $\mathcal{F}(\overline{\mathcal{K}_i})$ goes outside ϵ. Once this happens, we need to build new models. Building global models in a distributed environment is communication intensive. Here we rely on the outcome of our correct and efficient local algorithm to generate a trigger dictating the need for re-building the model. Given enough time to converge, the correctness of our monitoring algorithm ensures that even simple techniques such as best-effort sampling from the network may be sufficient to produce good results. If the model is not satisfactory, the underlying monitoring algorithm would eventually indicate this and a new model building will be triggered.

The idea of model computation in the network is very simple. Peers engage in a convergecast-broadcast process.

```
Input: $\epsilon$, $\epsilon_1$, $\epsilon_2$, $\epsilon_3$, $\mathcal{R}_{\mathcal{F}}$, $S_i$, $\Gamma_i$, $L$, $B$, $\widehat{\Theta}$ and $\tau$
Output: New model $\widehat{\Theta}$
Initialization: Initialize vectors; Set
$LastDataAlert \leftarrow \infty$;$Datasent \leftarrow false$;
On Receiving a message:
begin
    (MsgType, Recvd_$P_j$) ← MessageRecvdFrom($P_j$)
    if $MsgType = Monitoring\_Msg$ then
        Update Monitoring Algorithm;
    end
    if $MsgType = Pattern\_Msg$ then
        Update models;Forward new models to $\Gamma_i$;
        $Datasent = false$;Restart PeDEM;
    end
    if $MsgType = Dataset\_Msg$ then
        $NumRecvd$ = Count_num_recvd();
        $Recvd\_Dataset = Recvd\_Dataset \bigcup Recvd\_P_j$;
        if $NumRecvd$=$\Gamma_i - 1$ then
            $flag$=Output_Monitoring_Algorithm();
            if $Datasent = false \bigwedge flag = 1$ then
                if $CurrTime - LastDataAlert > \tau$ then
                    $D$=Sample($S_i$, $Recvd\_Dataset$, $B$);
                    $Datasent = true$;
                    Send $D$ to remaining neighbor;
                end
                else LastDataAlert=CurrTime;Check back in
                $\tau$ time;
            end
            if $flag$=0 then $LastDataAlert \leftarrow \infty$
        end
        if $NumRecvd$=$\Gamma_i$ then
            $D$=Sample($S_i$, $Recvd\_Dataset$, $B$);
            $NewModel$=EM($D$);
            Forward $NewModel$ to all neighbors;
            Datasent=$false$;
            Restart Monitoring Algorithm;
        end
    end
end
On an event:
Run Monitoring Algorithm;
flag=Output_Monitoring_Algorithm();
if flag=1 and $P_j$=IsLeaf() then
    Execute the same conditions as MsgType=Dataset_Msg;
end
```

Algorithm 2: Pseudo code for **PeDEM** Algorithm.

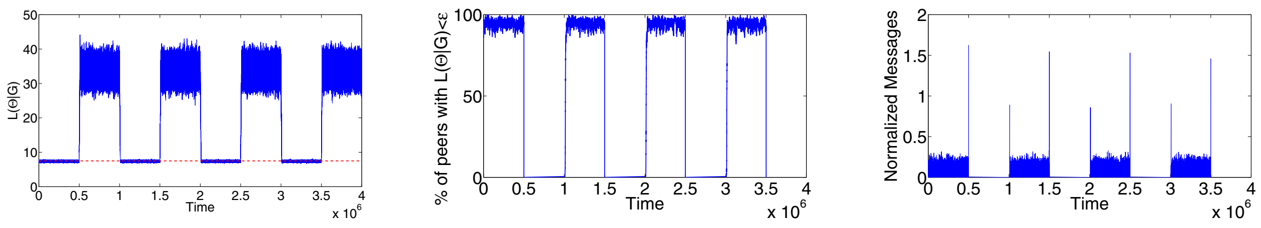

Figure 1. Plot of typical experiments. Each experiment is run for several epochs. Quality is measured as the percentage of peers correctly detecting an alert. Cost is measured during the entire experiment and during stationary phases. Last 80% of the time refers to stationary phase to ignore transitional effects. The dotted line in Figure 1(a) refers to ϵ=7.0.

The monitoring algorithm raises a flag whenever the model is a misfit with respect to the global data. If this happens for any peer, it first waits for a specific amount of time which we call the alert mitigation time τ to see if the alert is indeed due to a data change or random noise. If the alert exists even after τ units of time, the peer checks if it has received data from all its neighbors except one. If yes, it generates a sample of user-defined size B from its own data and each of its children weighing each point inversely as the size of its subtree such that each point has an equal probability of being included in the sample. It then sends the sample to its parent and marks its state as convergecast. Whenever a peer receives data from all peers, it becomes the root of the convergecast tree and employs a centralized EM algorithm to build new model parameters. It then sends these models to itself and marks its state as broadcast. Whenever a peer P_j gets new models from P_i, it forwards those to all its neighbors in Γ_j except P_i. P_j then moves from the convergecast to the broadcast phase. Because we do not impose the root of the tree, it may so happen that two peers get all the data simultaneously. We break the tie in such scenarios using the id of the nodes. For the same convergecast round, assuming $i > j$, if both P_i and P_j build new models, P_i is allowed to propagate the models in the network. Algorithm 2 presents the pseudo code of **PeDEM**. As shown, there are three types of messages: $Monitoring\_Msg$, $Pattern\_Msg$ and $Dataset\_Msg$. The monitoring message is passed to the underlying monitoring algorithm. The pattern message is received as part of the broadcast round while the datasets are received when the peer engages in the convergecast round.

VII. Experimental Results

We demonstrate the performance of **PeDEM** on a synthetic data generator and a real-world dataset (see Section VII-C). Our implementation of the algorithm was done in Java using the DDMT[1] toolkit developed at UMBC. For the topology, we used the BRITE topology generator[2]. We experimented with the *Barabasi Albert (BA)* model since it generates edge delays (in millisec) simulating the internet. We convert the edge delays to simulator ticks for time measurement since wall time is meaningless when simulating thousands of peers on a single PC. For all our experiments the average edge delay is 1100 simulator ticks. On top of each network generated by BRITE, we overlay a communication tree.

[1]http://www.umbc.edu/ddm/Software/DDMT/
[2]http://www.cs.bu.edu/brite/

For synthetic data experiments, the input data of a peer is a set of vectors in $\mathbb{R}^d$ generated according to multi-dimensional GMM. More specifically, for a given experiment, we fix the prior probabilities, the means and the covariance matrices. Whenever a peer needs an additional data point, it first selects a Gaussian s with corresponding probability π_s and then generates a Gaussian vector in $\mathbb{R}^d$ with mean and covariance $\overrightarrow{\mu_s}, \mathbf{C}_s$. The prior, means and the covariances are changed randomly at controlled intervals to create an epoch change.

The two quantities of measurement are the quality of the results and the cost. When the monitoring algorithm is used merely as a detector (open loop mode), quality refers to the percentage of the peers which correctly detect a change in the GMM model using (i) $\overline{\mathcal{L}}(\widehat{\Theta}|\mathcal{G})$, (ii) mean, and (iii) covariance monitoring. When **PeDEM** is used for rebuilding the model (closed loop), quality refers to the average log-likelihood over the entire experiment compared against a centralized execution. We plot both the mean and standard deviation for 10 independent trials (two columns for each measurement in Table I). Due to the dynamic nature, the cost of our algorithm, referred to as normalized messages, denotes the number of messages exchanged per peer per unit of leaky bucket. For a broadcast-based algorithm, its normalized message is 2, assuming 2 neighbors per peer on average. We report both the stationary and overall messages. We also report, where appropriate, messages required for convergecast and broadcast of the model.

A. Results: EM Monitoring

Figure 1 and Table I demonstrate the dependence of the monitoring algorithm on the following algorithm and system parameters: size of local dataset $|S_i|$; error thresholds ϵ, ϵ_1, ϵ_2, and ϵ_3; size of leaky bucket L; number of peers p; dimension of the problem d; and number of Gaussians k. In every experiment, we vary one of the parameters

Parameter	Values	Percentage of Correct Peers						Stationary Messages						Overall Messages					
		$\overline{\mathcal{L}}(\widehat{\Theta}\|\mathcal{G})$		Mean		Cov		$\overline{\mathcal{L}}(\widehat{\Theta}\|\mathcal{G})$		Mean		Cov		$\overline{\mathcal{L}}(\widehat{\Theta}\|\mathcal{G})$		Mean		Cov	
$\|S_i\|$	100	94.03	5.54	75.6	8.76	89.7	8.76	0.32	0.1	0.45	0.09	0.41	0.13	0.74	0.09	0.62	0.11	0.93	0.12
	200	97.7	5.1	85.5	8.71	93.7	8.99	0.26	0.09	0.18	0.06	0.34	0.09	0.46	0.08	0.53	0.1	0.61	0.09
	400	99.86	4.3	97.6	6.7	99.7	7.96	0.11	0.09	0.18	0.05	0.21	0.09	0.21	0.07	0.22	0.09	0.38	0.87
ϵ	6, 0.3, 5	56.5	5.98	67.9	10.1	73.2	9.18	1.23	0.09	0.68	0.1	0.61	0.16	1.67	0.09	0.78	0.16	1.21	0.19
	7, 0.5, 8	67.8	5.3	75.6	8.76	89.7	8.76	0.57	0.12	0.45	0.09	0.41	0.13	0.89	0.11	0.62	0.11	0.93	0.12
	9, 1.0, 10	97.9	5.32	99.5	5.35	94.63	6.14	0.21	0.11	0.21	0.06	0.35	0.07	0.34	0.1	0.56	0.08	0.86	0.09
L	200	95.6	6.34	76.4	7.32	90.6	6.54	0.43	0.10	0.56	0.12	0.47	0.08	0.84	0.09	0.71	0.12	1.14	0.09
	500	94.03	5.54	75.6	7.76	89.7	6.76	0.32	0.08	0.45	0.09	0.41	0.1	0.74	0.08	0.62	0.11	0.93	0.09
	1000	92.03	5.32	72.7	7.21	89.2	5.67	0.29	0.09	0.35	0.09	0.38	0.11	0.61	0.1	0.56	0.1	0.83	0.09
# Peers	500	94.03	5.54	75.6	8.76	89.7	8.76	0.32	0.11	0.45	0.11	0.41	0.1	0.74	0.11	0.62	0.1	0.93	0.1
	1000	94.12	6.71	75.4	6.45	89.3	6.31	0.32	0.11	0.44	0.09	0.41	0.1	0.73	0.08	0.61	0.11	0.92	0.11
	2000	94.09	4.46	75.2	6.45	89.4	6.33	0.31	0.11	0.43	0.09	0.41	0.09	0.74	0.09	0.62	0.11	0.92	0.2
# Dimension	2	96.07	5.12	81.34	7.21	94.3	7.32	0.29	0.12	0.41	0.12	0.4	0.11	0.67	0.12	0.52	0.13	0.78	0.13
	4	90.83	5.67	71.23	7.32	82.3	6.97	0.43	0.09	0.68	0.09	0.51	0.09	0.89	0.09	0.72	0.09	1.1	0.09
	7	65.34	5.98	60.78	6.32	61.2	6.98	0.93	0.098	1.23	0.09	0.9	0.1	1.23	0.01	1.45	0.09	1.57	0.09
# Gaussians	2	94.03	5.54	75.6	7.76	89.7	7.31	0.32	0.1	0.45	0.1	0.41	0.1	0.7	0.1	0.62	0.1	0.93	0.112
	3	93.05	6.31	72.17	6.51	86.71	7.59	0.33	0.09	0.47	0.12	0.42	0.09	0.77	0.09	0.63	0.09	0.94	0.11
	5	91.04	5.51	70.78	6.31	82.17	6.19	0.38	0.09	0.51	0.1	0.45	0.08	0.83	0.09	0.71	0.08	1.03	0.08

Table I
DEPENDENCY OF QUALITY AND COST ON THE DIFFERENT ALGORITHM PARAMETERS.

while keeping the others at their default values: $|S_i| = 100$, $\epsilon = 8.0$, $\epsilon_1 = 0.5$, $\epsilon_2 = 0.5$, $\epsilon_3 = 8.0$, L=500, p = 500, d=3, k=2. We choose ϵ based on a sample of data offline.

Figure 1 shows the data, quality and cost graph of a typical experiment. In all our experiments, we replace 10% of the data of each peer after every 1000 simulator ticks. On top of this, after every $5{\times}10^5$ ticks, we induce an epoch change by changing the distribution. For epochs 1, 3, 5,..., each peer is given the same model parameters $\widehat{\Theta}$ from which the data is generated. For other epochs, we change the data generator without changing $\widehat{\Theta}$ given to each peer. Hence $\overline{\mathcal{L}}(\widehat{\Theta}|\mathcal{G})$ for these epochs are low while for the others it is quite high as shown in Figure 1(a). Corresponding to this, Figure 1(b) reports the number of peers detecting $\overline{\mathcal{L}}(\widehat{\Theta}|\mathcal{G}) < \epsilon$ for each epoch. As is evident for the odd epochs, since threshold ϵ=7.0 passes through $\overline{\mathcal{L}}(\widehat{\Theta}|\mathcal{G})$, approximately 95% of the peers report the correct result. For the even epochs, the detection is much simpler since the $\overline{\mathcal{L}}(\widehat{\Theta}|\mathcal{G})$ is far away from ϵ. Figure 1(c) shows the corresponding communication cost which temporarily increases on every epoch change. In order to calculate the stationary cost, we consider the later 80% of each epoch.

The first row of the table depicts the quality and cost of **PeDEM** as size of the local dataset is increased from 100 to 400. As shown, the quality improves as $|S_i|$ is increased for all the monitoring instances. Both the stationary and overall communication overhead of the algorithm decrease as $|S_i|$ increases. Next we demonstrate the variation of quality and cost as the different ϵ-s vary. The three ϵ values at each row correspond to ϵ, ϵ_2 and ϵ_3 respectively. As expected, the quality improves as ϵ is increased for all the monitoring instances. Both the stationary and overall cost also decreases steadily with increasing ϵ. The reason is that smaller ϵ translates to a more difficult problem, thereby increasing the cost and deteriorating the quality. The variation with the size of the leaky bucket is shown next. The quality of the algorithm approximately remains constant as L is varied, but both the stationary and overall messages decrease with increasing L.

The last three parameters correspond to the scalability results. As shown, quality remains almost constant as the number of peers is increased. Also both the overall and stationary messages converge to a constant, independent of the number of nodes. This is because in a local algorithm, the resource consumption becomes independent of the size of the network with no change in quality. This results in excellent scalability of the algorithm. However, as shown, quality decreases linearly and cost increases linearly as the number of dimension is increased. This is can be attributed to an increase in variance in each dimension due to the covariance matrix.

Parameter	Values	Computed Model				Stationary Messages		Overall Messages		Data Rounds per epoch
		Distributed		Centralized		Avg	Std. dev	Avg	Std. dev	
τ	1000	8.4098	0.43	7.1098	0.21	0.015	0.009	0.0453	0.0075	3.5
	2000	8.4796	0.51	7.1098	0.21	0.024	0.007	0.0512	0.0061	2
	3000	8.5289	0.62	7.1098	0.21	0.029	0.004	0.0526	0.0051	1.5
	5000	9.1538	0.75	7.1098	0.21	0.0401	0.003	0.0647	0.0053	1
ϵ	6	7.456	0.58	7.1098	0.21	0.09	0.009	0.123	0.0075	5
	7	8.124	0.53	7.1098	0.21	0.089	0.007	0.101	0.0061	3.5
	8	8.479	0.51	7.1098	0.21	0.024	0.004	0.0512	0.0051	2.5
	9	8.7779	0.28	7.1098	0.21	0.021	0.003	0.0432	0.0053	2
Sample size	2000	9.8744	0.25	8.7098	0.46	0.056	0.009	0.079	0.0075	3
	5000	8.4796	0.33	7.1908	0.21	0.024	0.007	0.0512	0.0061	2
	7000	7.3898	0.21	6.51	0.19	0.023	0.004	0.051	0.0051	1.2
	10000	6.7124	0.28	5.92	0.15	0.021	0.003	0.0512	0.0053	1.2

Table II
PERFORMANCE OF PEDEM IN CLOSED LOOP EXPERIMENTS.

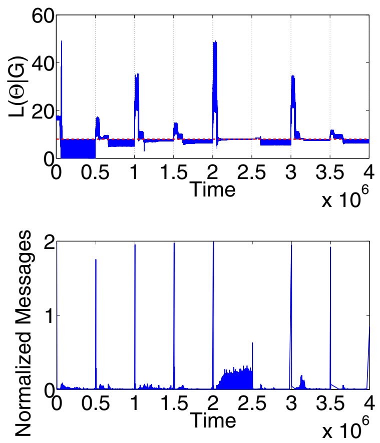

Figure 2. Typical run of PeDEM in closed loop.

B. Results: EM Models

In this section we demonstrate the results of **PeDEM** in closed loop mode. As shown in Figure 2, whenever the distribution changes, $\overline{\mathcal{L}}(\widehat{\Theta}|\mathcal{G})$ exceeds ϵ (dotted line), thereby triggering model rebuilding. This reduces $\overline{\mathcal{L}}(\widehat{\Theta}|\mathcal{G})$ to less than ϵ and subsequently only the efficient monitoring algorithm operates. The monitoring cost increases after every epoch change and then decreases after new models are rebuilt.

We explore the effect of three parameters in this section: alert constant τ, ϵ and sample size of convergecast. Their default values are: $\tau = 2000$ (twice the average edge delay), $\epsilon = 7.0$ and sample size=5000. We compare **PeDEM** to a centralized EM algorithm having access to all the data. The results are shown in Table II in which we show (1) average and standard deviation of the quality (for both distributed and centralized scenarios), (2) same for both stationary and overall monitoring messages, and (3) finally the number of data rounds per epoch. From the table, we see that the average log-likelihood increases with increasing τ. This is because the model remains inaccurate for a longer time before it is rebuilt. For the centralized algorithm, its log-likelihood is independent of τ. For the cost, as expected, the number of convergecast rounds decrease as τ increases. The second set of results show that the average log-likelihood increases and cost decreases as ϵ is increased. Finally, increase in sample size improves quality and decreases cost due to more accurate models being built. Moreover, the log-likelihood of **PeDEM** and the centralized algorithm become closer as the sample size is increased. We can conclude that most of the error induced in **PeDEM** is as a result of the sample size and not due to the distributed computation.

Thus, in general **PeDEM** provides a moderate rate of false positives and an excellent rate of true positives. The models built are quite accurate with respect to centralization under moderately low communication overhead.

C. Application

Consider a sensor network deployed in a remote region of the forest to monitor changes in the forest cover over time. **PeDEM** can be efficiently used for this purpose by deploying it in the network. In the absence of real sensor network data, we have simulated the network in our simulator using the real-world forest cover dataset[3]. This dataset has 54 features — 44 binary and the rest categorical. The last column is the class label which can take values between 1 to 7 each representing a different type of forest cover. In order to use the dataset for detecting changes in forest cover using **PeDEM**, we performed the following preprocessing tasks: (1) We have only used tuples having categories 1, 2, 3 and 7. (2) We have grouped them such that categories 1 and 3 belong to the same set and 2 and 7 belong to the other; each set corresponds to one epoch. Each epoch denotes one forest cover type. (3) We removed the class information and all the binary attributes, leaving us with 10 attributes. (4) The dataset for each epoch consists of 240000 tuples. Initially the dataset was divided into 200 peers with each peer having 600 points. At 100,000, 200,000 and 300,000

[3]http://kdd.ics.uci.edu/databases/covertype/covertype.html

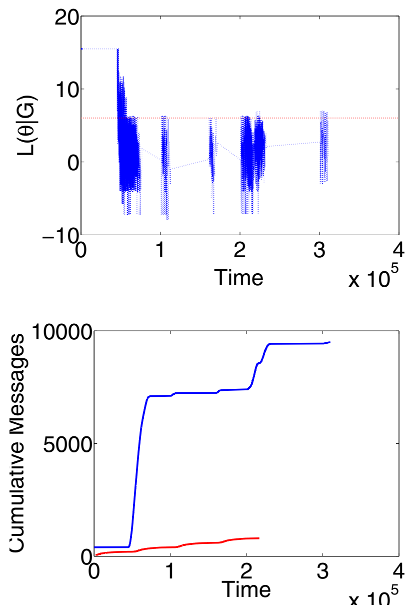

Figure 3. Quality and cost for forest covertype dataset.

clock ticks we replaced 10% of the peers points every 1000 clock ticks. Thus the entire dataset of each peer was replaced after 110,000, 210,000 and 310,000 clock ticks. We have fitted a 10-dimensional GMM having two gaussian components since each epoch consists of data from two separate classes. Figure 3 shows the results. The left figure shows the variation in $\overline{\mathcal{L}}(\widehat{\Theta}|\mathcal{G})$ vs. time with the red line showing ϵ. Whenever data changes, we see re-computation of the model (to reduce the log-likelihood). The right figure shows the cumulative monitoring messages (blue) and data messages (red). As seen, monitoring messages remain constant when the data do not change. We also directly compared the GMM model computed by **PeDEM** and that computed centrally and found that the models were very close. This result shows that **PeDEM** may allow scientists to track changes in the forest cover without centralizing all the data.

VIII. Conclusion

In this paper we have presented a local, asynchronous, distributed, and provably correct algorithm for monitoring the GMM parameters in a large P2P network. To the best of the authors' knowledge, this is the first EM algorithm specifically developed for such environments. The algorithm can seamlessly adapt to data and network changes. Besides direct contribution to monitoring GMM parameters, this paper also developed a technique for monitoring the covariance of the data. Our extensive experimental results support the theoretical claims.

Acknowledgements

This work was supported by the NASA Aviation Safety Program Intelligent Vehicle Health Management project.

References

[1] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. R. Stat. Soc., B*, vol. 39, no. 1, pp. 1–38, 1977.

[2] D. Gu, "Distributed EM Algorithm for Gaussian Mixtures in Sensor Networks," *IEEE TNN*, vol. 19, no. 7, pp. 1154–1166, 2008.

[3] R. D. Nowak, "Distributed EM Algorithms for Density Estimation and Clustering in Sensor Networks," *IEEE TSP*, vol. 51, no. 8, pp. 2245–2253, 2003.

[4] W. Kowalczyk and N. A. Vlassis, "Newscast EM," in *Proceedings of NIPS'04*, 2004, pp. 713–720.

[5] J. Wolfe, A. Haghighi, and D. Klein, "Fully Distributed EM for Very Large Datasets," in *Proceedings of ICML'08*, 2008, pp. 1184–1191.

[6] S. Bandyopadhyay, C. Giannella, U. Maulik, H. Kargupta, K. Liu, and S. Datta, "Clustering Distributed Data Streams in P2P Environments," *Inf. Sci.*, vol. 176, no. 14, pp. 1952–1985, 2006.

[7] W. Kowalczyk, M. Jelasity, and A. E. Eiben, "Towards Data Mining in Large and Fully Distributed Peer-to-Peer Overlay Networks," in *Proceedings of BNAIC*, 2003, pp. 203–210.

[8] K. Das, K. Bhaduri, K. Liu, and H. Kargupta, "Distributed Identification of Top-l Inner Product Elements & its Application in a P2P Network," *TKDE*, vol. 20, no. 4, pp. 475–488, 2008.

[9] N. Linial, "Locality in Distributed Graph Algorithms," *SIAM Journal of Computing*, vol. 21, pp. 193–201, 1992.

[10] Y. Afek, S. Kutten, and M. Yung, "The Local Detection Paradigm and Its Application to Self-Stabilization," *In Theoretical Computer Science*, vol. 186, no. 1–2, pp. 199–229, 1997.

[11] R. Wolff and A. Schuster, "Association Rule Mining in P2P Systems," *IEEE SMC-B*, vol. 34, no. 6, pp. 2426 – 2438, 2004.

[12] J. Branch, B. Szymanski, C. Giannella, R. Wolff, and H. Kargupta, "In-Network Outlier Detection in Wireless Sensor Networks," in *Proceedings of ICDCS'06*, 2006.

[13] P. Luo, H. Xiong, K. Lü, and Z. Shi, "Distributed Classification in P2P Networks," in *Proceedings of SIGKDD'07*, 2007, pp. 968–976.

[14] K. Bhaduri, R. Wolff, C. Giannella, and H. Kargupta, "Distributed Decision Tree Induction in P2P Systems," *Stat. Anal. and Data M.*, vol. 1, no. 2, pp. 85–103, 2008.

[15] R. Wolff, K. Bhaduri, and H. Kargupta, "A Generic Local Algorithm for Mining Data Streams in Large Distributed Systems," *TKDE*, vol. 21, no. 4, pp. 465–478, 2009.

[16] I. Sharfman, A. Schuster, and D. Keren, "A Geometric Approach to Monitoring Threshold Functions over Distributed Data Streams," in *Proceedings of SIGMOD'06*, 2006, pp. 301–312.