A distributed EM algorithm for large peer-to-peer networks

Wojtek Kowalczyk \textsuperscript{a}  \quad Nikos Vlassis \textsuperscript{b}

\textsuperscript{a} Dept. of Computer Science, Vrije Universiteit Amsterdam
\textsuperscript{b} Informatics Institute, University of Amsterdam

Abstract

We propose a new distributed algorithm for Gaussian mixture learning: Newscast EM. The algorithm can run on large peer-to-peer networks in which each node observes a local quantity and can communicate with other nodes in a random, anonymous fashion. The main difference with the standard (centralized) EM algorithm is the M-step which is implemented in a decentralized manner: random pairs of nodes exchange their local parameter estimates and combine them appropriately. We provide theoretical evidence and demonstrate experimentally that only a few exchange cycles (e.g., 30) are sufficient in each M-step to guarantee convergence to the maximum likelihood estimates, and this holds independently of the size of the network (which could be in the order of million nodes).

1 Introduction

Recent advances in embedded systems technology have led to the development of large-scale distributed systems that can be used for applications like remote monitoring of environmental data. A sensor network (SN), for instance, is a collection of a large number of computing devices, each characterized by limited sensing, processing, and communication capabilities. Each node in the network measures some environmental property (e.g., temperature), and the collective of measurements from all nodes provides useful information about the state of the environment where the SN is being deployed. In many applications, the ultimate objective is to optimize a cost function that may depend on all measured data [10].

Although local perception at each node is cheap, efficiently extracting global information from the data (and potentially optimizing a cost function) is a challenging problem: typical computational requirements are minimal memory usage, high speed, and guaranteed solution quality, while typical network requirements are minimal communication, node anonymity, and fault tolerance. A centralized solution in which a single node collects all measurements is clearly inappropriate for networks of thousands or even millions of nodes (e.g., sensor dust). Instead, research is focussed on decentralized, in-network protocols that compute and disseminate aggregates of the data, resulting in minimal communication and therefore low energy consumption [1]. A very attractive approach to such distributed data
aggregation—which we adopt in this paper—is newscast computing [4, 5, 6] that involves very rapid, epidemic-style, data dissemination.

Many decentralized optimization algorithms for SNs [1, 9, 10, 3] rely on the use of a fixed routing tree over the network which allows for message passing between neighboring nodes. The distributed EM algorithm of [9], for instance, utilizes a routing path over which sufficient statistics are propagated from one node to the other in a fixed order. Using a fixed routing tree has several disadvantages: each node must have a name, fault tolerance may be difficult to achieve, while for optimization problems a full traversal of the tree may be needed in each step, with cost at least linear in the network size.

In this paper we propose a decentralized algorithm for Gaussian mixture learning of the distributed set of measurements. Contrary to most of the algorithms mentioned above, the proposed algorithm, which we call Newscast EM (NEM, and a variant of it NEMO), exhibits the following attractive properties: (i) it requires no routing tree, (ii) each node may be anonymous, and (iii) no synchronization is needed. In NEMO each node in the network maintains a local estimate of the parameters of the mixture, and updates them based on data aggregates that are asynchronously disseminated over the network. The choice of which nodes should communicate in each step is done by simple randomization over the set of (currently known) nodes. A similar algorithm (that involves averaging local rewards in a multiagent MDP task) was recently proposed in [7], but with no results on the rate of convergence.

2 Gaussian mixtures and the EM algorithm

A $k$-component Gaussian mixture for a random vector $x$ in $\mathbb{R}^d$ is defined as the convex combination

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

of $k$ Gaussian densities $p(x|s) = (2\pi)^{-d/2}|C_s|^{-1/2} \exp[-(x-m_s)^\top C_s^{-1}(x-m_s)/2]$, each parameterized by its mean $m_s$ and covariance matrix $C_s$. The components of the mixture are indexed by the random variable $s$ that takes values from 1 to $k$, and $\pi_s = p(s)$ defines a discrete prior distribution over the components. Given a set $\{x_1, \ldots, x_n\}$ of independent and identically distributed samples from $p(x)$, the learning task is to estimate the parameter vector $\theta = \{\pi_s, m_s, C_s\}_{s=1}^k$ of the $k$ components that maximizes the log-likelihood function $L(\theta) = \sum_{i=1}^n \log p(x_i; \theta)$. Throughout we assume that the likelihood function is bounded from above (e.g., by placing appropriate bounds on the components covariance matrices).

Maximization of the data log-likelihood $L(\theta)$ can be carried out by the EM algorithm [2] which can be seen as iteratively maximizing a lower bound of the data log-likelihood [8]. This bound $\mathcal{F}(\theta, Q)$ is a function of the current mixture parameters $\theta$ and a factorized distribution $Q = \prod_{i=1}^n q_i(s)$, where each ‘responsibility’ $q_i(s)$ corresponds to a data point $x_i$ and defines an arbitrary discrete distribution
over $s$. This lower bound is given by:

$$\mathcal{F}(\theta, Q) = \sum_{i=1}^{n} \sum_{s=1}^{k} q_i(s) \left[ \log p(x_i, s; \theta) - \log q_i(s) \right].$$

(2)

In the E-step of the EM algorithm, the responsibilities $q_i(s)$ assigned to the points $x_i$ equal the Bayes posteriors given the parameters found in the previous step, i.e., $q_i(s) = p(s|x_i)$. In the M-step, we solve for the unknown parameters of the mixture by maximizing $\mathcal{F}$. This yields the following solutions:

$$\pi_s = \frac{\sum_i q_i(s)}{n}, \quad m_s = \frac{\sum_i q_i(s)x_i}{n\pi_s}, \quad C_s = \frac{\sum_i q_i(s)x_ix_i^\top}{n\pi_s} - m_sm_s^\top,$$

(3)

where the sums run over all data points ($i = 1, \ldots, n$).

Note that the main operation of the M-step is averaging: $\pi_s$ is the average of $q_i(s)$, $m_s$ is the average of products $q_i(s)x_i$ (divided by $\pi_s$), and the covariance matrix $C_s$ is the average of matrices $q_i(s)x_ix_i^\top$ (divided by $\pi_s$ and decreased by $m_sm_s^\top$). This observation is essential for the proposed algorithm, as we will shortly see.

3 Newscast Computing and Averaging

The proposed algorithm for learning Gaussian mixtures relies on the use of the newscast model of computation [4]. Very informally, a newscast network consists of large number of anonymous nodes that exchange data and process them. The network operates in cycles: in every cycle each node $i$ contacts a randomly selected node $j$, and then the nodes exchange their data and perform some application-specific computations. The underlying newscast protocol involves dynamic exchange of (i) caches with addresses of network members (membership management), and (ii) application data (information dissemination). The protocol is very robust, scalable, and simple to implement—its Java implementation is only a few kB of code and can run on small network-enabled computing devices such as mobile phones, PDA’s, or sensors.

An application of the newscast protocol is to find the average of numbers that are distributed along the network nodes [5]. Suppose that $n$ nodes of a network store values $v_1, \ldots, v_n$ (one value per node). To compute their average, the nodes operate as follows: Initially every node $i$ takes $x_i = v_i$ as the local estimate of the average. Then, in every cycle, whenever two nodes $i$ and $j$ exchange their estimates $x_i$ and $x_j$ (i.e., $x_i$ is sent to $j$ and $x_j$ is sent to $i$), they update their local estimates to $(x_i + x_j)/2$. After $n$ such exchanges, the variance of the original set is on average reduced by at least $1/e \approx 0.368$ as we show next.

**Lemma 3.1.** Let $X$ be a set of $n$ points in $\mathbb{R}$, with mean $m$ and variance $\sigma^2$, and suppose we select two points from $X$ uniformly at random and replace them with their average. The variance of $X$ is then reduced on the average by $1 - 1/n$. 
Proof. Let \( X = \{ x_1, \ldots, x_n \} \) with \( m = \frac{1}{n} \sum_{k=1}^{n} x_k \) and \( \sigma^2 = \frac{1}{n} \sum_{k=1}^{n} (x_k - m)^2 \), and let \( I \) and \( J \) be two iid random variables that are uniform in \( \{ 1, \ldots, n \} \). For any realization \( (i, j) \) of \( (I, J) \), the points \( i \) and \( j \) are selected and replaced by \( x'_i = \frac{1}{2}(x_i + x_j) \). Note that the mean \( m \) does not change, while for the new variance \( \sigma^2_{ij} \) holds: 
\[
\sigma^2_{ij} = n\sigma^2 - (x_i - m)^2 - (x_j - m)^2 + 2\left(\frac{x_i + x_j}{2} - m\right)^2 = n\sigma^2 - \frac{1}{2}(x_i - m)^2 - \frac{1}{2}(x_j - m)^2 + (x_i - m)(x_j - m).
\]
Taking expectations over \( I \) and \( J \) gives: 
\[
E[\sigma^2_{ij}] = \frac{1}{2} \left( \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma^2_{ij} \right) = (1 - 1/n)\sigma^2.
\]

The above lemma applies when each pair of nodes is chosen uniformly at random, but in practice each node \( i \) initiates a single contact with a random node \( j \) within each cycle (note that a node may be contacted many times by other nodes). The equivalent of achieving fully random pair selection procedure is to have each node wait for a time interval that is randomly drawn from the exponential distribution [6], in which case the conditions of the above lemma apply. Clearly, if the waiting time goes to zero, for \( n \) exchanges holds:
\[
\sigma^2_{t+1} \leq (1 - 1/n)^n \sigma^2 \approx \frac{\sigma^2}{\varepsilon}.
\]

Similar analysis shows that if in each cycle each node initiates exactly one contact, then the variance is reduced on the average by a factor \( 1/(2\sqrt{\varepsilon}) \approx 0.3033 \) (see the same paper for experimental results demonstrating the above bounds). Thus \( m \) cycles of the averaging algorithm reduce the original variance \( \sigma^2 \) of the data set \( \{ x_1, \ldots, x_n \} \) to \( \sigma^2/(2\sqrt{\varepsilon})^m \). The fact that the averaging algorithm reduces variance exponentially fast means that the nodes learn the average very quickly. Indeed, using Chebyshev’s inequality \( P(\{ x - \mu \} \geq \varepsilon) \leq \sigma^2/\varepsilon^2 \) we can see that for any \( \varepsilon \) the probability of making a mistake bigger than \( \varepsilon \) is dropping exponentially fast with the number of cycles \( m \).

Note that with increasing number of cycles \( m \), the values of the set \( X \) are more and more normally distributed. Indeed, elements of \( X \) can be viewed as realizations of a random variable that is expressed as an arithmetic average of \( 2^m \) random variables (due to successive averaging operations), so the Central Limit Theorem applies. Therefore, to make sure that the errors of approximations are smaller than \( \varepsilon \) it is sufficient to choose a number of cycles \( m \) in such a way that the standard deviation is a few times smaller than \( \varepsilon \). For instance, if the initial variance of \( X \) is 1 then we need about 23, 42, or 61 cycles to reduce possible errors to \( 10^{-5} \), \( 10^{-10} \), or \( 10^{-15} \), respectively. Note that the \( m \) vs. \( \varepsilon \) trade-off does not depend on the number of nodes or the shape of the initial distribution of data.

4 The Newscast EM algorithm

The Newscast EM (NEM) algorithm is a direct application of the averaging protocol described above for estimating the parameters of a Gaussian mixture using (3). Suppose that single data records \( x_i \) are distributed over the nodes of a network (one record per node). Each node \( i \) maintains a local estimate of the parameters of each mixture component \( s \), denoted by \( \pi'_i, m'_i, C'_i = \bar{C}'_i - m'_i m'_i \top \), as well as
a responsibility \( q_i(s) \). Each node initializes its \( q_i(s) \) arbitrarily and then all nodes start the process of finding the averages of \( q_i(s), q_i(s)x_i, \) and \( q_i(s)x_ix_i^T \), over all \( i \)’s, as required in (3).

The complete NEM algorithm is as follows:

1. **Initialization.** Each node \( i \) sets \( q_i(s) \) to some random positive value and then normalizes all \( q_i(s) \) to sum to 1 over all \( s \). Moreover it sets its local estimates for component \( s \) as follows:
   \[
   \pi_i^s = q_i(s), \quad m_i^s = x_i, \quad \tilde{C}_i^s = x_ix_i^T.
   \]

2. **M-step.** The newscast averaging process is iterated for \( m \) cycles (e.g., \( m = 30 \)). In each cycle, each node \( i \) selects a node \( j \) at random and the following operations take place (for each component \( s \)):
   \[
   \begin{align*}
   \pi_i^{s(new)} &= \pi_j^{s(new)} = \frac{\pi_i^s + \pi_j^s}{2}, \\
   m_i^{s(new)} &= m_j^{s(new)} = \frac{\pi_i^s m_i^s + \pi_j^s m_j^s}{\pi_i^s + \pi_j^s}, \\
   \tilde{C}_i^{s(new)} &= \tilde{C}_j^{s(new)} = \frac{\pi_i^s \tilde{C}_i^s + \pi_j^s \tilde{C}_j^s}{\pi_i^s + \pi_j^s}.
   \end{align*}
   \]

3. **E-step.** Each node calculates new responsibilities \( q_i(s) = p(s|x_i) \) using the estimates \( \pi_i^s, m_i^s, \) and \( C_i^s = \tilde{C}_i^s - m_i^s m_i^{sT} \). Note that these updates are fully local.

4. **Go to step 2,** unless a stop criterion is met (involving relative changes of the parameter estimates).

In total there are \( k[1 + d + d(d + 1)/2] \) averages to be found, where \( k \) denotes the number of components and \( d \) the dimensionality of the data. The algorithm proceeds as the normal EM algorithm, with the M-step implemented in a distributed manner: nodes continuously exchange their local parameter estimates until convergence. Then each node recalculates local posteriors \( q_i(s) \) in the E-step, a new M-step starts, and so on.

Note that the weighted averages of the means \( m_i^s \) in (6) and the ‘covariances’ \( \tilde{C}_i^s \) in (7), with weights given by (5), are in fact unweighted averages of the corresponding products \( \pi_i^s m_i^s \) and \( \pi_i^s \tilde{C}_i^s \). Therefore, NEM converges exponentially fast to the correct component parameters, as explained in the previous section. The M-step finds all required parameters with a prespecified accuracy that is controlled by the number of newscast cycles \( m \); by setting \( m \) large enough we can guarantee that all nodes will have identical parameter estimates at convergence.

### 4.1 Newscast EM Optimized

A drawback of the NEM algorithm is that each node has to wait \( m \) cycles (the length of the M-step) before it can use the new parameter estimates in a new E-step. We propose here an improvement to the above algorithm, which we call Newscast EM Optimized (NEMO). The idea is the following: instead of waiting
until the M-step converges, after a small number of cycles (say, between 3 to 5) each node runs a local E-step and adjusts its responsibilities, and then propagates appropriate corrections through the network.

In particular, in NEMO the E-step changes as follows:

3. **Modified E-step.** Each node calculates new responsibilities

$$q_i'(s) = p(s|x_i)$$

using the local estimates $$\pi_i, m_i, \text{ and } C_i = C_i - m_i m_i^\top$$. Let $$q_i(s)$$ denote the previous responsibilities of node $$i$$. Then the node makes the following corrections:

$$\pi_i^{\text{new}}(s) = \pi_i + q_i'(s) - q_i(s) \quad (8)$$

$$m_i^{\text{new}}(s) = [m_i \pi_i + x_i (q_i'(s) - q_i(s))] / \pi_i^{\text{new}} \quad (9)$$

$$\tilde{C}_i^{\text{new}}(s) = [\tilde{C}_i \pi_i + x_i x_i^\top (q_i'(s) - q_i(s))] / \pi_i^{\text{new}} \quad (10)$$

The idea in the above updates is that a node corrects its local estimates after an E-step rather than restarting from scratch. In the M-step NEMO uses the newscast averaging algorithm that is applied to $$x_i$$’s and $$x_i x_i^\top$$’s with weights $$q_i(s)$$. After a small number of cycles (e.g., $$m = 5$$), each node re-estimates its local $$q_i(s)$$, calculates corrections, and updates its local estimates as shown above. Then the M-step is resumed. These corrections may increase the variance of the local estimates, but because model estimates are not reset after the E-step but only updated, the corresponding increase of variance is relatively small. Although we have not shown convergence theoretically, we have experimentally verified that convergence is achieved using a much smaller number of steps (3–5) compared to NEM (30–50).

5 **Results of Experiments**

To get an insight into the actual behavior of the presented algorithms we have implemented them and run numerous experiments using a newscast simulator. We used several artificially generated data sets–mixtures of 3-5 normal distributions–with the number of records varying between $$10^3$$ and $$10^6$$. To make fair comparisons we were using the same initial configurations.

The first thing we looked at was the variance reduction as a function of the number of cycles $$m$$. We noticed that the variance of weights ($$\pi_s$$) was dropping exactly as predicted by theoretical model, whereas the variance of means ($$m_s$$) was dropping much faster, especially in the first few cycles. This can be easily explained: calculation of means combines two averaging processes: averaging of weights and weighted averaging of original values. The same observation applies to the variance reduction rate for $$C_s$$.

Next, we looked at the role of the parameter $$m$$ on the accuracy of the final model that was measured by log-likelihood. Because each node ends up with its own, local model (which might be biased towards local data) we forced the network to iterate the averaging step 100 times after reaching the last cycle of the EM algorithm, so all the models were identical. We noticed that sporadically
smaller values of $m$ (say $m < 10$) led to a rapid improvement or deterioration of the final model. Apparently smaller values of $m$ introduce more variance into local estimates and this may pull the search process from local (or global) minima. For $m > 15$ there was no noticeable difference in model accuracy, see Figure 1. For more complicated data (more components, components closer to each other) the minimal values of $m$ from which changes were negligible were 20-30.

Finally, we looked at the differences between NEM and NEMO. It turned out that in most cases NEMO may use much smaller $m$ (5-10 times) than NEM to obtain the same accuracy. However, in some cases NEMO was not converging at all. It was caused by “corrections” that were causing weights to jump outside the $(0,1)$ interval.

6 Conclusions

We presented two algorithms for finding mixture models in a highly distributed manner: NEM and NEMO. The first algorithm directly mimics the classical (centralized) EM procedure. The main parameter—the number of averaging cycles per M-step—controls the accuracy of the estimates. In practice, 20-30 of such cycles guarantee the same results as the classical EM. This means that for every M-step each node has to exchange its local model 20-30 times, independently on the network size.

The NEMO algorithm reduces the data exchange effort by factor 5-10. Unfortunately, there is no guarantee of algorithm convergence. Our further research will focus on eliminating this drawback.
References


