Distributed Probabilistic Inferencing in Sensor Networks using Variational Approximation

Sourav Mukherjee [1] Hillol Kargupta [2]

Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County.

Abstract

This paper considers the problem of distributed inferencing in a sensor network. It particularly explores the probabilistic inferencing problem in the context of a distributed Boltzmann machine-based framework for monitoring the network. The paper offers a variational mean-field approach to develop communication-efficient local algorithm for Variational Inferencing in Distributed Environments (VIDE). It compares the performance of the proposed approximate variational technique with respect to the exact and centralized techniques. It shows that the VIDE offers a much more communication-efficient solution at very little cost in terms of the accuracy. It also offers experimental results in order to substantiate the scalability of the proposed algorithm.

1 Introduction

Probabilistic inferencing lies at the core of a wide range of sensor network applications, such as *target tracking*, *target location*, and *distributed sensor calibration.* Probabilistic inferencing can be viewed as the task of computing the posterior probability distribution of a set of hidden variables, given a set of visible variables (also known as evidence variables or observed variables), and a probability model for the underlying joint distribution of the hidden and the visible variables [3].

In this paper, we consider the problem of inferencing in a heterogeneously distributed environment where each visible variable is measured in a different node, and each node is required to compute the probability distribution

[1] sourav1@umbc.edu
[2] hillol@cs.umbc.edu, corresponding author; also affiliated to Agnik, LLC.

of the hidden variables, given the visible variables. The problem of exact inferencing, even when the data is centralized, is intractable, and variational methods provide a deterministic approximation technique used in such inferencing [6,7]. This paper explores the possibility of variational approximation in distributed computation for inferencing in a distributed Boltzmann machine. The paper presents an algorithm, VIDE (Variational Inference in Distributed Environment), which can perform probabilistic inferencing in heterogeneously distributed environments using variational approximation.

The rest of this paper is organized as follows. Section 2 motivates and formally defines the problem of distributed inferencing in the context of sensor network applications. Section 3 offers a brief survey of existing work that is relevant. Section 4 shows how the inferencing problem can be posed as a variational optimization problem. Section 5 presents our algorithm, VIDE, which extends the variational approach to the heterogeneously distributed environment. It shows that variational inference in a heterogeneously distributed system reduces to a distributed average computation, followed by a purely local phase of computation. Section 6 analyzes the communication complexity of VIDE. It also presents an analysis of the energy consumption of VIDE. Section 7 presents the experimental results. Finally, Section 8 concludes this paper.

2 Motivation and Problem Definition

A large number of sensor network applications involve probabilistic inferencing. This section presents some applications, which motivate the distributed probabilistic inferencing problem. Next, it presents a formal definition on the distributed inferencing problem.

Consider the problem of *target classification* in a sensor network . An object is present in a field of sensor networks, and several attributes of that object are being measured by the sensors. The goal is to classify the object. If the measured attributes are considered to be visible variables, and the class label to be a hidden variable, then the target classification problem requires an estimation of the probability distribution of the hidden variable given the visible variables.

A similar problem is that of *target localization* in sensor networks. Several attributes of a possibly mobile object are being measured by sensor networks. The task is to estimate the position of the object. For example, in a battlefield where acoustic sensor networks have been deployed, we may want to estimate the position where a fire-arm was just triggered. Again, we can model the coordinates of the object as hidden variables, and the sensor network measurements as the visible variables. For that, we may need to know the probability distribution of the hidden variables given the visible variables.

As a third example, we can consider the *distributed sensor calibration* problem [5,3]. Let us suppose that some variable (e.g. temperature) is being measured by a set of sensors. The measurements taken by the individual sensors may have biases, depending on the measurement technique used. However, if we have a probability model involving the true temperatures, the measured temperatures, and the biases, then we can compute the posterior probability of the true temperatures, given the measured temperatures.

All the above example are special cases of the following abstract problem:

Probabilistic Inferencing Problem. Let $\mathbf{X} = (\mathbf{X}_h, \mathbf{X}_v)$ be a random vector, where $\mathbf{X}_h$ and $\mathbf{X}_v$ consist of the hidden and the visible attributes respectively. We are also given a model for the joint probability, $P(\mathbf{x}_h, \mathbf{x}_v|\theta)$, where θ is the parameter vector for the model. We wish to compute the posterior probability of the hidden variables, given the visible variables, that is:

$$P(\mathbf{x}_h|\mathbf{x}_v;\theta) = \frac{P(\mathbf{x}_h, \mathbf{x}_v|\theta)}{P(\mathbf{x}_v|\theta)}$$

The probability model is usually expressed as a graphical model. We shall see, in the next section, that performing exact inferencing in a graphical model is an intractable problem, and so we need effective approximation techniques to solve the problem. We shall also see, later, that variational methods provide a deterministic approximation technique to such problems.

Another aspect common to all the above scenarios is that each node in the network measures a distinct visible random variable, that is, the data is completely heterogeneously distributed. The naïve way to perform inferencing in such a system would be to centralize all the data at one site and then run an inferencing algorithm. However, sensor nodes have limited battery life, and communication accounts for the majority of the power consumption in sensor nodes. This makes the above approach impracticable. Moreover, the system may need to take decisions based on the inference, within real-time constraints. In that case, the limitations in communication bandwidth is likely to restrict the system's capability to react within real-time constraints. Thus, we need a way to compute the desired distribution with minimum communication. This motivates the definition of the distributed probabilistic inferencing problem:

Distributed Probabilistic Inferencing Problem. Formally, let $\mathbf{X}_v = \{X_v^{(1)}, X_v^{(2)}, ..., X_v^{(p)}\}$ such that $X_v^{(i)}$ is measured at node N_i of a network. Our problem is to estimate $P(\mathbf{x}_h|\mathbf{x}_v;\theta)$ while minimizing the communication between the nodes of the sensor network.

As mentioned earlier, the probability distribution of $\mathbf{X}$ is usually expressed as a graphical model. A graphical model consists of a set of nodes, representing the random variables, and a set of edges, representing dependencies between pairs of random variables [7].

For concreteness, we shall consider a special kind of graphical model, known as the Boltzmann Machine. The reasons for selecting Boltzmann machines in this study are two-fold:

(1) The properties of Boltzmann Machines as probability models have been studied extensively in the literature (see, for example, [29]).
(2) Boltzmann Machines have been shown to be very useful tools in pattern recognition problems. For example, Boltzmann Machines have been used in recognition of handwritten digits [30]. The usefulness of such models in face-recognition has been demonstrated in [31]. Boltzmann Machines have also been used in biometric authentication through fingerprint recognition, as described in [32].

Thus, the Boltzmann Machine constitutes a graphical model that is both well understood, and applicable to pattern recognition problems. The present work addresses the problem of performing inference in such a graphical model, when each visible variable is measured or detected at a separate node.

The paper also frequently uses the term "local algorithms" throughout this paper. Let us explain what exactly that means in this paper. Consider a network represented by an abstract tuple $G =< V, E, C_{V,t}, \mathbb{D}_V, P_E >$. V and E represent the vertex and edge sets of the undirected network-graph; $C_{V,t}$ represents the set of all states $C_{v,t}$ of a vertex $v \in V$ at time t. $\mathbb{D}_V = \{\mathbb{D}_1, \mathbb{D}_2, \cdots \mathbb{D}_N\}$ and P_E represents properties of every edge is E. The *distance* ($\text{dist}_G(u,v)$) between a pair of nodes u and v is the length of the shortest path between those vertices. The α-neighborhood of a node $u \in V$ is defined as, $\Gamma_\alpha(u,G) = \{v|\text{dist}_G(u,v) \leq \alpha\}$. Let each node $v \in V$ stores a data set X_v. An α-local query by some vertex v is a query whose response can be computed using some function $f(X_\alpha(v))$ where $X_\alpha(v) = \{X_v|v \in \Gamma_\alpha(v,V)\}$. An algorithm is called α-local if it never requires computation of a β-local query such that $\beta > \alpha$. In the rest of this paper, the term *local algorithm* will be loosely used for implying an *α-local algorithm* where α is either a small constant or a slowly growing function with respect to the network parameters such as the number of nodes. The next section presents a review of the related work done in this field.

3 Related Work

As mentioned earlier, target tracking, target localization etc. are important sensor network applications which constitute special examples of the distributed inferencing problem; in recent times, they have been studied extensively (see, for example, [1], [2]). The distributed sensor calibration problem, which is another example of the distributed inferencing problem, is described in [5,3].

Distributed Data Mining (DDM) has recently emerged as an extremely important area of research. Distributed Data Mining is concerned with analysis of data in distributed environments, while paying careful attention to issues related to computation, communication, storage, and human-computer interaction. The data can be distributed *homogeneously*, where all sites share the same schema, but each site stores attribute values of a different set of tuples. Alternatively, the data can be distributed *heterogeneously*, where all sites store the same set of tuples, but each site stores a different set of attributes. In the latter scenario, each site must store some key attribute, such that the join of the local data-sources is lossless. Detailed surveys of Distributed Data Mining algorithms and techniques have been presented in [22–24]. Some of the common data-analysis tasks include association rule mining, clustering, classification, kernel density estimation and so on. A sigficiant amount of research has been done in the recent years to develop techniques for such data analysis, suitable for distributed environments. For example, an algorithm for distributed association rule mining in peer-to-peer systems has been presented in [25]. K-means clustering has been extended to the distributed scenario in [26]. Techniques for performing non-parametric density estimation over homogeneously distributed data have been studied and experimentally evaluated in [27]. The problem addressed in the present paper, probabilistic inferencing in distributed environments, is a problem in DDM, because it involves performing inferencing in a database that is distributed in a completely heterogeneous way, with each node storing only a single visible variable.

The applicability of graphical models and belief propagation to distributed inferencing in sensor networks has been studied in [12,13,4,3]. The underlying probability distribution is represented as a graph, and inferencing proceeds as a sequence of message-passing operations between nodes of the graph [14]. In [4], such belief propagation based inferencing has been applied to scenarios with discrete random variables; in [13], it has been applied to multivariate Gaussian distributions.

Graphical models can be of two types: directed and undirected. In an undirected graphical model, the joint probability of all the random variables is given by the (normalized) product of local *potential functions*, where each potential function corresponds to a clique in the graph. A representative of the exact inferencing algorithms is the *junction-tree* algorithm [11] which operates on an undirected graphical model; if the original model is directed, then it converts it to an undirected model by an operation called *moralization*. The moralized graph is then *triangulated*, such that there are no 4-cycles without a chord. Then, the cliques of the graph can be arranged in a junction-tree data structure. It is important, that if two cliques have a node in common, then they assign the same marginal probability to that node (this is called the *local consistency property*). Maintenance of local consistency requires marginalization and rescaling the clique potentials. But, as observed in [7], this can take time that is exponential in the size of the largest clique in the network. Let, for example, the nodes in a particular clique be $X_1, X_2, ..., X_n$, and let $\Psi(x_1, x_2, ..., x_n)$ be the corresponding clique potential. Then, for any $X_1 = x_1$, the marginal probability of X_1 would be given by $\sum_{x_2,x_3,...,x_n} \Psi(x_1, x_2, ..., x_n)$ which takes time $O(2^{n-1})$. This suggests that any practical solution to the inferencing problem must resort to approximation techniques. Variational approximation is a deterministic approximation technique that has been shown to be suitable for this problem [6,7].

Boltzmann machines form an important and widely researched class of graphical models [15]. In fact, they were the first explicitly stochastic networks for which a learning rule was developed [16], [19] Traditionally, inferencing and learning in Boltzmann machines was done using Gibbs sampling; later, mean field approximations were applied [17]. Another interesting approach to the problem is *decimation*, [18], which is efficient in only certain subclasses of Boltzmann machines, such as Boltzmann trees and chains. Boltzmann machines have been applied in many pattern recognition problems, such as recognition of handwritten digits [30], face recognition [31], and fingerprint recognition [32]. The present work deals with the scenario where each visible variable in the Boltzmann machine is measured at a different node of the network, which implies that a distributed computation is necessary to perform the inferencing.

Approximation algorithms often provide practical solutions to problems that are intractable. Approximation algorithms can be classified into two broad categories: probabilistic approximation algorithms (or randomized algorithms) [34] and deterministic approximation algorithms. In a probabilistic approximation algorithm, the output of a problem instance depends on both the input and the output of a psuedo-random number generator. Deterministic approximation techniques, on the other hand, do not have any element of randomness involved. Variational approximation constitutes an important technique that falls under this category. Usually, the variational approach to solving a problem involves defining an objective function that is optimized at the solution to the problem, and then searching for an optimum of that function. Often, this search is done in a restricted subset of the solution space, which makes the technique computationally tractable. A typical example is the Finite Element Method (FEM). Suppose we wish to solve a differential equation, for which deriving a closed-form solution is difficult. As described in [7], the variational approach to solving this problem involves defining a variational objective function, and then searching for a solution that optimizes this objective function. However, since it is not practicable to search in the space of all functions, we consider only those functions that can be expressed as linear combinations of a finite set of basis functions. Thus the problem reduces to evaluation of the coefficients of these basis functions, which is a search problem in finite dimensional space.

Jordan, Jaakkola et al [6], [7] have developed the framework for applying variational optimization to graphical models, where exact inferencing is intractable. They have addressed problems like inferencing, parameter estimation using Expectation Maximization (EM) and Bayesian parameter estimation. However, their work assumes that the data is centralized.

The standard variational mean-field approximation for inferencing in Boltzmann machines is to assume that the hidden variables are independant, given the visible variables (this is described formally in the next section). This paper demonstrates the applicability of this approach in the distributed scenario.

Paskin et al [3] have developed a junction-tree based framework for performing exact inferencing in distributed graphical models. However, they do not adopt a variational approach.

Vlassis et al [9] have applied the variational EM approach to compute the parameters of a mixture of Gaussian distribution. In [10], they have shown that the variational formulation, when applied to the homogeneously distributed scenario, reduces to a distributed average computation problem. We, too, have used distributed average computation as an essential part of our algorithm; however, we are concerned with the heterogeneously distributed scenario, and the graphical model we choose is the Boltzmann machine.

This paper demonstrates that the variational approximation technique is applicable to the problem of probabilistic inferencing in heterogeneously distributed environments. In particular, it makes the following contributions:

- It extends the variational formulation of the probabilistic inferencing problem to the heterogeneously distributed scenario.
- Based on this formulation, it presents a deterministic approximation algorithm, VIDE (Variational Inferencing in Distributed Environments), for solving this problem.
- It presents theoretical analysis and experimental evaluation of the accuracy and communication cost of VIDE, and establishes VIDE as a communication-efficient technique for solving the distributed inferencing problem.

In the next section, we consider the problem of probabilistic inferencing in a Boltzmann machine, and its variational formulation.

4 Background

In this section, we first briefly review the variational method as a deterministic approximation technique for solving problems, applicable when finding the exact solution is too expensive. Then, we present the problem of inferencing in a Boltzmann machine, and explains why solving it naïvely is intractable. Finally, we present the variational mean-field technique for solving the inferencing problem.

4.1 The Variational Approximation Technique

In this section, we develop the basic ideas of variational approximation. To illustrate the technique, we shall use a simple problem: linear regression. The variational technique, as applied to this problem, is explained in detail in [7]; this section presents a brief summary of that discussion.

Illustrative Example: Linear Regression Consider a linear function $f : \mathbb{R}^d \to \mathbb{R}$. The task is to find f, given the values of $y_i = f(\mathbf{x_i})$, for $1 \leq i \leq n$ and $\{\mathbf{x_i} : 1 \leq i \leq n\} \subset \mathbb{R}^d$. Since f is linear, the problem reduces to finding the weight vector $\mathbf{w} \in \mathbb{R}^d$ such that $y = f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$.

The exact solution $\mathbf{w}^*$ to this problem is given by

$$\mathbf{w}^* = \mathbf{C}^{-1}\mathbf{b}$$

where

$$\mathbf{C} = \sum_{i=1}^{n} \mathbf{x_i}\mathbf{x_i}^T$$

and

$$\mathbf{b} = \sum_{i=1}^{n} y_i\mathbf{x_i}$$

Although, in this problem, we have a closed form expression for the exact solution, computing the exact solution involves computing the inverse of $\mathbf{C}$, which can become expensive, if the dimensionality d is large.

The variational method attempts find a solution to a given problem by optimizing an objective function.Depending on the nature of the objective function, and on the technique used to optimize it, the solution obtained using the variational method may be exact or approximate.

For the linear regression problem, let us define the following objective function:

$$J(\mathbf{w}) = \frac{1}{2}(\mathbf{w}^* - \mathbf{w})^T \mathbf{C}(\mathbf{w}^* - \mathbf{w})$$

which is the distance between the approximate solution $\mathbf{w}$ and the exact solution $\mathbf{w}^*$ weighted by the matrix $\mathbf{C}$.

Thus, $J(\mathbf{w}) \geq 0$ and $J(\mathbf{w}) = 0 \Leftrightarrow \mathbf{w} = \mathbf{w}^*$.

Thus, the variational formulation of this problem is simply:

Minimize:

$$J(\mathbf{w}) = \frac{1}{2}(\mathbf{w}^* - \mathbf{w})^T \mathbf{C}(\mathbf{w}^* - \mathbf{w})$$

Although the expression for $J(\mathbf{w})$ involves $\mathbf{w}^*$, which we do not know, we can still use the fact that $w^* = \mathbf{C}^{-1}\mathbf{b}$, to prove that:

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{b}^T \mathbf{C}^{-1}\mathbf{b} - \mathbf{w}^T\mathbf{b} + \frac{1}{2}\mathbf{w}^T\mathbf{C}\mathbf{w}$$

which is an expression for $J(\mathbf{w})$ that does not contain $\mathbf{w}^*$. Finally, the above expression indicates that minimizing $J(\mathbf{w})$ is equivalent to minimizing the following function:

$$\tilde{J}(\mathbf{w}) = -\mathbf{w}^T\mathbf{b} + \frac{1}{2}\mathbf{w}^T\mathbf{C}\mathbf{w}$$

This variational formulation is very useful. In this example, the objective function $\tilde{J}(\mathbf{w})$ is quadratic in $\mathbf{w}$ and hence, convex (this, however, is not the case in many other problems where the variational approximation is applied). As a result, a simple gradient descent will converge to the global minimum.

Thus, the key step in solving a computationally difficult problem by the variational method, involves defining an objective function, which is optimized at the exact solution. Having defined this function, the problem then reduces to a search problem that aims to find an approximate optimal point of this function.

In the next section, we formally define the problem of probabilistic inference in a Boltzmann machine; we argue that exact solution to this problem takes exponential time. Then, we demonstrate how the variational method is used to obtain an approximate solution.

4.2 Boltzmann Machine

Consider a random vector $\mathbf{X}$, comprised of binary random variables having range $\{0, 1\}$ (we are interested in finite-state distributions; thus a $k-$ary random variable can be expressed as a finite sequence of binary random variables, and so assuming the random variables to be binary does not lead to any loss of generality). Some of the components of $\mathbf{X}$ are visible, while others are hidden. Let $\mathbf{X}_h$ be the vector containing only the hidden components, and let $\mathbf{X}_v$ be the vector containing only the visible components, such that $\mathbf{X} = (\mathbf{X}_h, \mathbf{X}_v)$.

We assume that the distribution of $\mathbf{X}$ is a Boltzmann machine with parameter vector θ. That is,

$$P(\mathbf{x}|\theta) = \frac{1}{Z} \exp(-\Phi(\mathbf{x}; \theta))$$

where

$$\Phi(\mathbf{x}; \theta) = -(\sum_{i<j} \theta_{i,j} x_i x_j + \sum_i \theta_{i0} x_i)$$

is the *energy function*, and

$$Z = \sum_{\mathbf{x}} \exp(-\Phi(\mathbf{x}; \theta))$$

is the normalizing constant, called the *partition function.*

The basic inferencing problem is to compute the probability distribution of the hidden variables, given the visible variables,

$$P(\mathbf{x}_h|\mathbf{x}_v;\theta) = \frac{P(\mathbf{x}_h,\mathbf{x}_v|\theta)}{P(\mathbf{x}_v|\theta)}$$

Actually, if the distribution $P(\mathbf{x}|\theta)$ is a Boltzmann machine, then the distribution $P(\mathbf{x}_h|\mathbf{x}_v;\theta)$ is also a Boltzmann machine, given by

$$P(\mathbf{x}_h|\mathbf{x}_v;\theta) = \frac{1}{Z_c}\exp(-\Phi_c(\mathbf{x}_h;\theta))$$

where

$$\Phi_c(\mathbf{x}_h;\theta) = -(\sum_{i<j;\, x_i,x_j\in\mathbf{x}_h} \theta_{i,j}x_ix_j + \sum_{x_i\in\mathbf{x}_h} \theta^c_{i0}x_i)$$

such that the indices i,j vary only over the hidden variables, and

$$\theta^c_{i0} = \theta_{i0} + \sum_{x_j\in\mathbf{x}_v} \theta_{i,j}x_j$$

This is because, if, for any i,j, both X_i, X_j are hidden, then they continue to contribute the quadratic exponent $\theta_{i,j}x_ix_j$ in the numerator. Otherwise, if, say X_i is hidden and X_j is visible, then x_j becomes a constant, so the exponent becomes linear in x_i; this explains why $\theta^c_{i,0}$ has a contribution $\theta_{i,j}x_j$, when X_j is visible. Finally, when both X_i, X_j are visible, then the exponential of $\theta_{i,j}x_ix_j$ becomes a constant, and is included in the overall normalization constant Z_c.

The normalization constant, Z_c is, then, the *partition function* of this new Boltzmann machine. Exact computation of the *partition function*, however, is intractable because

$$Z_c = \sum_{\mathbf{x}_h} \exp(-\Phi_c(\mathbf{x}_h;\theta))$$

which is a summation over all possible values of the hidden variable vector $\mathbf{x}_h \in \{0,1\}^H$, where H is the number of hidden variables. The number of terms in this summation is,therefore, 2^H. Since this is exponential in the number of hidden variables, we need approximate techniques to estimate the distribution.

4.3 Variational Mean-Field Approximation

Variational methods try to approximate $P(\mathbf{x}_h|\mathbf{x}_v;\theta)$ by a distribution $Q(\mathbf{x}_h)$ which is computationally tractable. In particular, the mean-field approximation assumes that Q factorizes completely across the hidden variables,

$$Q(\mathbf{x}_h) = \prod_i Q_i(x_i)$$

To quantify the accuracy of the approximation, we use the following standard objective function [8,6,7] (which is to be maximized):

$$J(Q) = -\sum_{\mathbf{x}_h} Q(\mathbf{x}_h) \ln \frac{Q(\mathbf{x}_h)}{\exp(-\Phi_c(\mathbf{x}_h;\theta))}$$

Now if we define each of the individual marginal distributions Q_i to be a binary distribution with mean μ_i, that is,

$$Q_i(x_i) = \mu_i^{x_i}(1-\mu_i)^{1-x_i}$$

then the parameters μ_i that maximize $J(Q)$ are given by the set of *mean field equations* (one for each μ_i):

$$\mu_i = \sigma(\sum_{j>i} \theta_{i,j}\mu_j + \theta_{i0}^c)$$

where $\sigma(z) = 1/(1+e^{-z})$ is the logistic function.

The following section presents our algorithm, VIDE, for solving this problem in a distributed setting, such as in a sensor network environment.

5 Variational Inferencing in Distributed Environments (VIDE)

Consider the case when $\mathbf{X}_v = \{X_v^{(1)}, X_v^{(2)}, ..., X_v^{(p)}\}$ such that the i^{th} visible variable, $X_v^{(i)}$, is measured at node N_i of a network. This corresponds a vertical partitioning of $\mathbf{X}_v$, in which, at any instant of time, the i^{th} attribute of $\mathbf{x}_v$, $x_v^{(i)}$ is stored in the i^{th} node N_i. Also, each node knows the parameters of the distribution θ. Given that at any instant of time, node N_i measures $X_v^{(i)} = x_v^{(i)}$, we want each node to be able to compute the distribution $P(\mathbf{x}_h|\mathbf{x}_v;\theta)$ by computing the solution vector μ to the mean field equations. We propose an algorithm, VIDE, described in algorithm 1 which solves this problem in a distributed manner.

Algorithm 1 $VIDE$ for node k

```
Input: The following are the arguments to $VIDE$:
    (1) The value of the visible variable $X_v^{(k)}$ measured at this node.
    (2) The value of $\theta_{i,j}$ for every pair of random variables $X_i, X_j$
    (3) The value of $\theta_{i0}$ for every random variable $X_i$.
Output: The mean $\mu_i$ of the marginal distribution under the variational approximation, for every hidden variable $X_i$.

    Let $p$ be the number of nodes in the network.
 1: for each hidden variable $X_i$ do
 2:     $\psi_i \leftarrow Distributed\_Average(\{\theta_{i,j}x_j : x_j \in \mathbf{x}_v\})$
 3:     $\theta^c_{i0} \leftarrow \theta_{i0} + p \cdot \psi_i$
 4:     Also, initialize $\mu_i \leftarrow \frac{1}{2}$
 5: end for
 6: while $\mu$ has changed significantly in the last iteration do
 7:     for each hidden variable $X_i$ do
 8:         $\mu_i \leftarrow \sigma(\sum_{j>i} \mu_j + \theta^c_{i0})$
 9:     end for
10: end while
11: return $\mu$
```

The computation can be broken down into two phases, which must be performed by *each* node:

(1) Calculate, for each i,

$$\theta^c_{i0} = \theta_{i0} + \sum_{x_j \in \mathbf{x}_v} \theta_{i,j} x_j$$

(2) Iteratively, find a fixed point solution for the mean-field equations, as

$$\mu_i^{(t+1)} = \sigma(\sum_{j>i} \theta_{i,j}\mu_j^{(t)} + \theta^c_{i0})$$

where t denotes the number of iterations. This can be done in a *purely local fashion*, as each node already knows θ^c_{i0} (for every i), and all the other parameters in the above equation are stored locally.

The crux of the computation, therefore, lies in computing sums of the form $s = \sum_{i=1}^{p} s_i$ where each s_i is stored in a different node. Now, if we assume the network (*not* the data it measures) is static, then we can write the sum as

$$s = p\bar{s}$$

where

$$\bar{s} = \frac{1}{p} \sum_{i=1}^{p} s_i$$

is the arithmetic average of $\{s_i\}_{i=1}^{p}$.

Thus we can compute the sum in a distributed fashion by computing the corresponding average, and then multiplying it by the number of sites.

If the network is dynamic, however, then we will also need a distributed algorithm for estimating the network size. Fortunately, algorithms for estimating network size from local information exist [20]. However, in our experiments, we have considered only static networks.

Of the several algorithms for distributed average computation, we choose the A1 algorithm proposed by Mehyar et al.[28]. This can be considered as one implementation of the *Distributed_Average* function used by VIDE.

Distributed Average Computation. In this section, we describe the distributed averaging algorithm used by VIDE. Each node k stores a scalar, z_k, and an estimate y_k of the global average. It is the aim of the algorithm to achieve the condition: $y_k \approx \frac{1}{p}\sum_{i=1}^{p} z_i$. Algorithm 2 describes the algorithm in brief.

Intuitively, the algorithm works as follows. Each node k starts by setting its average estimate to the value it stores, i.e., $y_k = z_k$. At each time step, it communicates its current estimate of the average with some immediate neighbor; then, these two neighbors adjust their estimates so that the two estimates are now closer, and such that their sum is conserved. Thus, after a sufficient number of such updates have occurred in the network, the average estimates of all the nodes will be very close to one another. Since their sum will still be the sum of all the values stored in the network (sum of the average estimates is conserved in an update), these estimates will be close to the true average across the network. For a more detailed study of the algorithm, the reader is referred to [28].

It is clear, from lines 1 and 2 of algorithm 1 that VIDE needs to invoke the distributed averaging routine once for each hidden variable $X_h^{(i)}$. Conceptually, the A1 algorithm runs forever at each node. In the implementation of VIDE, however, the the averaging process terminates when a predetermined maximum number of messages have been passed in the network.

Algorithm 2 $A1\_Average(\{z_i\}_{i=1}^{p})$ for node k

```
Input: The following are the inputs to the algorithm:
      (1) $z_k$, the value stored in node $k$, and,
      (2) $\gamma_k$, the step-size parameter used by node $k$ for averaging.
Output: $y_k \approx \frac{1}{p}\sum_{i=1}^{p} z_i$
 1: Initialize: $y_k = z_k$
 2: while true do
 3:   while Every neighbor $i$ such that $i < k$ has not updated $k$ do
 4:     if such a neighbor $i$ initiates update with node $k$ then
 5:       if $k$ is already executing an update with some other neighbor $j$
          then
 6:         Send negative acknowledgement (NACK) to node $i$. A NACK
            message will cause this update operation to fail, and will have no
            effect on the state of the network.
 7:       else
 8:         $Increment = \gamma_k(y_i - y_k)$
 9:         Update for node $k$:
                 $y_k = y_k + Increment$
10:         Send $Increment$ to node $i$.
11:         Update for node $i$:
                 $y_i = y_i - Increment$
12:       end if
13:     end if
14:   end while
15:   Now sequentially initiate update with every neighbor $m$ such that $m >$
      $k$.
16: end while
```

In the next section, we present a theoretical analysis of the accuracy of VIDE, as a function of the communication complexity.

6 Analysis

In this section, we present error bounds for VIDE, and show that the bounds become exponentially tighter, as the number of messages exchanged increases. Then, we derive an expression for the energy consumption of VIDE, which is important from a sensor network viewpoint.

Throughout this analysis, we assume that H is the number of hidden variables, and V is the number of visible variables.

The accuracy of VIDE will be affected by the following factors:

(1) The error in computing the parameters $\{\theta_{i0}^c\}_{i=0}^H$ using the distributed averaging algorithm. As more messages are exchanged, this error becomes smaller.
(2) The error in computing $\{\mu_i\}_{i=0}^H$ by solving the mean-field equations, as the result of the error in $\{\theta_{i0}^c\}_{i=0}^H$.

First, we consider the error due to the distributed averaging algorithm. Then we see how this affects the error in VIDE's results.

6.1 *Error in Distributed Averaging*

Our error analysis will be based on the analysis presented in [28]. Let us assume that there is a network with V nodes, and each node i stores a scalar z_i. Each note wishes to estimate $\bar{z} = \sum_{i=1}^V z_i$. Let $y_i(t)$ be the estimate of this average, at node i and time t. Initially, $y_i(0) = z_i$ for all i, and the conservation property of the algorithm guarantees that $\bar{y}(t) = \sum_{i=0}^V y_i(t) = \bar{z}$ for all times t.

To quantify the convergence properties, the authors of [28] use the following metric:

$$P(t) = \sum_{1 \le i < j \le V} |y_i(t) - y_j(t)|$$

Thus, at time t, $P(t)$ is the sum of the difference in the average estimates between every pair of nodes. The minimum value of $P(t)$ is 0, which happens when every node has exactly the same estimate for the average.

It is shown in the paper that given any time t, there exists $t' > t$ such that there has been at least one update in each link, during the time interval $[t, t']$. Further,

$$P(t') \le (1 - \frac{8\gamma^*}{V^2})P(t)$$

where $\gamma^* = min_k min\{\gamma_k, 1 - \gamma_k\}$.

The authors use this result to demonstrate that

$$\lim_{t \to \infty} P(t) = 0$$

We will take this analysis one step further, and present a convergence rate.

First, we assume that there exists $T > 0$ such that within any time interval of length T, there is guaranteed to be at least one update along each link. Such a restriction is desirable in any practical implementation of the eventual update assumption.

Then, the time interval $[0, t]$ contains at least $\lfloor \frac{t}{T} \rfloor$ subintervals of length T. applying the above result, we get

$$P(t) \leq P(0)(1 - 8\frac{\gamma^*}{V^2})^{\lfloor \frac{t}{T} \rfloor}$$

We use this to prove the following result.

Theorem 6.1 *Let U be the number of updates that occur in the time interval $[0, t]$, and let L be the total number of links in the network. Then,*

$$|y_i(t) - \bar{z}| \leq P(0)(1 - 8\frac{\gamma^*}{V^2})^{\lfloor \frac{U}{LT} \rfloor}$$

Proof Note that since U is the number of updates that occured in the interval $[0, t]$, then U is proportional to the number of messages exchanged. Also, since the number of links in the network is L then $U \leq tL$ which implies

$$P(t) \leq P(0)(1 - 8\frac{\gamma^*}{V^2})^{\lfloor \frac{U}{LT} \rfloor}$$

Finally, if we assume, without loss of generality, that

$$y_1(t) \leq y_2(t) \leq ... \leq y_V(t)$$

then it is easy to see that, for all i, $|y_i(t) - \bar{y}(t)| \leq |y_1(t) - y_V(t)|$ and that

$$|y_1(t) - y_V(t)| = |\sum_{i=1}^{V-1}(y_i(t) - y_{i+1}(t))|$$

$$\leq \sum_{i=1}^{V-1} |y_i(t) - y_{i+1}(t)| \leq P(t)$$

This gives us the convergence rate of the averaging algorithm:

$$|y_i(t) - \bar{z}| \leq P(0)(1 - 8\frac{\gamma^*}{V^2})^{\lfloor \frac{U}{LT} \rfloor}$$

which completes the proof. $\square$

In VIDE, we compute each $\theta^c_{i,0}$ using the above algorithm (here, i is an index that refers to the i^{th} hidden variable), and so the above convergence rate applies to each such $\theta^c_{i,0}$.

In the next section we show how the error in solving the mean field equations depends on the error in computing these average estimates.

6.2 Error in Solving the Mean Field Equations

After the distributed averaging phase, each node will have its estimate of $\theta^c_{i,0}$, corresponding to the i^{th} hidden variable $X_h^{(i)}$. In the subsequent phase, each node solves the system of mean-field equations, to estimate the mean μ_i of each hidden variable $X_h^{(i)}$. This is purely local computation; the error in this computation originates from:

(1) The error in $\theta^c_{i,0}$ for each hidden variable $X_h^{(i)}$,
(2) The error tolerance ϵ such that the iterative algorithm:

$$\mu_i^{(t+1)} = \sigma(\sum_{j>i} \theta_{i,j}\mu_j^{(t)} + \theta^c_{i0})$$

stops when $||\mu^{(t+1)} - \mu^{(t)}|| < \epsilon$.

.

Out of these, the second factor does not, in any way, affect the communication complexity of our algorithm. Smaller values of ϵ will lead to more iterations of a purely local computation. Hence, we will concentrate only on the first factor, and assume that given $\{\theta^c_{i,0}\}_{i=0}^H$, the mean field equations are solved exactly.

Let $\Delta\mu_i$ be the error in computing μ_i, the mean of the marginal distribution Q_i corresponding to the $i^t h$ variable $X_h^{(i)}$. Let $\Delta\theta^c_{i,0}$ be the error in computing $\theta^c_{i,0}$, incurred by the distributed averaging algorithm. Then, we have the following result:

Theorem 6.2 *If $\{|\Delta\theta^c_{i,0}|\}_{i=1}^H$ are small enough so that we can ignore error terms of order 2 and higher, then,*

$$|\Delta\mu_i| \leq \sum_{j=i+1}^{H} |\theta_{i,j}||\Delta\mu_j| + |\Delta\theta^c_{i,0}|$$

Proof We have,

$$\mu_i = \frac{1}{1 + \exp\left(-[\sum_{j=i+1}^H \theta_{i,j}\mu_j + \theta^c_{i,0}]\right)}$$

$$= f(\{\mu_j\}_{j=i+1}^H, \theta^c_{i,0})$$

Considering only first order errors, we have,

$$\Delta\mu_i = \sum_{j=i+1}^{H} \frac{\partial f}{\partial \mu_j}\Delta\mu_j + \frac{\partial f}{\partial \theta^c_{i,0}}\Delta\theta^c_{i,0}$$

Taking magnitude on both sides yeilds,

$$|\Delta\mu_i| = \left|\sum_{j=i+1}^{H} \frac{\partial f}{\partial \mu_j}\Delta\mu_j + \frac{\partial f}{\partial \theta^c_{i,0}}\Delta\theta^c_{i,0}\right|$$

$$\leq \sum_{j=i+1}^{H} \left|\frac{\partial f}{\partial \mu_j}\Delta\mu_j\right| + \left|\frac{\partial f}{\partial \theta^c_{i,0}}\Delta\theta^c_{i,0}\right|$$

Now, it is straight-forward to calculate these partial derivatives:

$$\frac{\partial f}{\partial \mu_j} = \theta_{i,j} \frac{\exp\left(-[\sum_{j=i+1}^{H} \theta_{i,j}\mu_j + \theta_{i,0}^c]\right)}{(1+\exp\left(-[\sum_{j=i+1}^{H} \theta_{i,j}\mu_j + \theta_{i,0}^c]\right))^2}$$

which gives,

$$\left|\frac{\partial f}{\partial \mu_j}\right| \leq |\theta_{i,j}|$$

and,

$$\frac{\partial f}{\partial \theta_{i,0}^c} = \frac{\exp\left(-[\sum_{j=i+1}^{H} \theta_{i,j}\mu_j + \theta_{i,0}^c]\right)}{(1+\exp\left(-[\sum_{j=i+1}^{H} \theta_{i,j}\mu_j + \theta_{i,0}^c]\right))^2}$$

which gives,

$$\left|\frac{\partial f}{\partial \theta_{i,0}^c}\right| \leq 1$$

Combining these inequalities, we get

$$|\Delta\mu_i| \leq \sum_{j=i+1}^{H} |\theta_{i,j}||\Delta\mu_j| + |\Delta\theta_{i,0}^c|$$

which proves the theorem. $\square$

Theorem 6.2 gives a recurrence relation for $\{\Delta\mu_i\}_{i=1}^{H}$, in terms of $\{\Delta\theta_{i,0}^c\}$. The following theorem provides the solution to this recurrence.

Theorem 6.3 *For* $0 \leq i \leq H-1$,

$$|\Delta\mu_{H-i}| \leq \sum_{k=H-i}^{H} |\Delta\theta_{k,0}^c| S(H-i,k)$$

where, $S(l,l) = 1$ for $1 \leq l \leq H$, and for $1 \leq l < m \leq H$ we have,

$$S(l,m) = \sum_{r=2}^{m-l+1} \sum_{l=i_1<i_2<...<i_r=m} |\theta_{i_1,i_2}||\theta_{i_2,i_3}|...|\theta_{i_{r-1}i_r}|$$

Proof We prove this statement by Mathematical Induction on i.

Basis $[i = 0]$ This follows directly from Theorem 6.2.

Induction Let us assume the induction hypothesis to be true for i, and then we prove it for $i + 1$.

Now, from Theorem 6.2

$$|\Delta\mu_{H-(i+1)}| \leq |\Delta\theta^c_{H-(i+1),0}| + \sum_{j=0}^{i} |\theta_{H-(i+1),H-j}||\Delta\mu_{H-j}|$$

$$\leq |\Delta\theta^c_{H-(i+1),0}| + \sum_{j=0}^{i} |\theta_{H-(i+1),H-j}|(\sum_{k=H-j}^{H} |\Delta\theta^c_{k,0}|S(H-j,k))$$

$$= |\Delta\theta^c_{H-(i+1),0}| + \sum_{j=0}^{i} \sum_{k=H-j}^{H} |\theta_{H-(i+1),H-j}||\Delta\theta^c_{k,0}|S(H-j,k)$$

$$= |\Delta\theta^c_{H-(i+1),0}| + \sum_{k=H-i}^{H} \sum_{j=H-k}^{i} |\theta_{H-(i+1),H-j}||\Delta\theta^c_{k,0}|S(H-j,k)$$

$$= |\Delta\theta^c_{H-(i+1),0}| + \sum_{k=H-i}^{H} |\Delta\theta^c_{k,0}| \sum_{j=H-k}^{i} |\theta_{H-(i+1),H-j}|S(H-j,k)$$

$$= |\Delta\theta^c_{H-(i+1),0}| + \sum_{k=H-i}^{H} |\Delta\theta^c_{k,0}|S(H-(i+1),k)$$

$$= \sum_{k=H-(i+1)}^{H} |\Delta\theta^c_{k,0}| S(H-(i+1), k)$$

which proves the induction hypothesis. $\Box$

If we extend the definition of $S(\cdot,\cdot)$ so that $S(l,m) = 0$ if $l > m$, then we can restate Theorem 6.3 as

$$|\Delta\mu_{H-i}| \leq \sum_{k=1}^{H} |\Delta\theta^c_{k,0}| S(H-i, k)$$

where $0 \leq i \leq H-1$. Equivalently we can write,

$$|\Delta\mu_i| \leq \sum_{k=1}^{H} |\Delta\theta^c_{k,0}| S(i, k)$$

where $1 \leq i \leq H$.

Theorem 6.3 expresses the error $|\Delta\mu_i|$ in the computing the mean μ_i of the marginal distribution of $X_h^{(i)}$. If we define the variational mean vector as $\mu = (\mu_1, \mu_2, ..., \mu_H)$, then the following theorem gives the error $||\Delta\mu||$ in computing the variational mean, as a function of the error in distributed averaging.

Theorem 6.4

$$||\Delta\mu|| \leq \sum_{k=1}^{H} |\Delta\theta^c_{k,0}| M(k)$$

where

$$M(k) = \sum_{i=1}^{H} S(i, k)$$

Proof Since

$$\mu = (\mu_1, \mu_2, ..., \mu_H)$$

we have

$$\Delta\mu = (\Delta\mu_1, \Delta\mu_2, ..., \Delta\mu_H)$$

Thus,

$$||\Delta\mu|| \le \sum_{i=1}^{H} |\Delta\mu_i|$$

$$\le \sum_{i=1}^{H}\sum_{k=1}^{H} |\Delta\theta_{k,0}^{c}| S(i,k)$$

$$= \sum_{k=1}^{H}\sum_{i=1}^{H} |\Delta\theta_{k,0}^{c}| S(i,k)$$

$$= \sum_{k=1}^{H} |\Delta\theta_{k,0}^{c}| (\sum_{i=1}^{H} S(i,k))$$

$$= \sum_{k=1}^{H} |\Delta\theta_{k,0}^{c}| M(k)$$

where $M(k) = \sum_{i=1}^{H} S(i,k)$. This completes the proof. $\square$

Finally, we can combine Theorem 6.1 and Theorem 6.4 to express the error in the variational mean vector, as a function of the number of updates in the network. The next theorem presents this result.

Theorem 6.5

$$||\Delta\mu|| \le V\beta(1 - \frac{8\gamma^*}{V^2})^{\lfloor \frac{U}{LT} \rfloor}$$

for some constant $\beta > 0$.

Proof From Theorem 6.1 it follows that for all $1 \le k \le H$,

$$|\Delta\theta_{k,0}^{c}| \le VP_k(0)(1 - \frac{8\gamma^*}{V^2})^{\lfloor \frac{U}{LT} \rfloor}$$

In the above inequality, V appears as a multiplicative constant because, computing $\{\theta_{k,0}^c\}_{k=1}^H$ involves computing sums, which, in turn are computed in a distributed fashion, by first calculating averages, and then multiplying by the network size, V.

Thus, from Theorem 6.1 and Theorem 6.4 we have,

$$||\Delta\mu|| \leq \sum_{k=1}^{H} |\Delta\theta_{k,0}^c| M(k)$$

$$= \sum_{k=1}^{H} V P_k(0)(1 - \frac{8\gamma^*}{V^2})^{\lfloor \frac{U}{LT} \rfloor} M(k)$$

$$= V\beta(1 - \frac{8\gamma^*}{V^2})^{\lfloor \frac{U}{LT} \rfloor}$$

where

$$\beta = \sum_{k=1}^{H} M(k) P_k(0)$$

This completes the proof. □

Our theoretical analysis establishes that the error decreases exponentially as the number of updates (and hence, the number of messages) increases.

Now we will derive an expression for the energy consumption of VIDE.

6.3 Energy Analysis

This section presents an analysis of the total amount of energy spent by the sensor nodes in VIDE. For simplicity, we shall make the following assumptions:

(1) Since a major part of a sensor node's energy is spent in communication, we shall focus only on communication, and ignore the othe factors, whose contribution to the total energy spent is usually much smaller. Also, we shall ignore the effects of interference.

(2) We shall use the Rayleigh Fading model to calculate the energy requirements [33].

We begin by defining some parameters. Let N_0 be the noise power, Θ be the minimum value of singal to noise ratio for successful transmission, d_{max} be the transmission range of a node, p_r be the required minimum success probability of a message transmission, and α be the attenuation coefficient.

Then, the following theorem gives an expression for the minimum amount of energy required by VIDE to achieve a desired accuracy.

Theorem 6.6 *The total energy E_{VIDE} required by a sensor network performing VIDE is related to the error $\Delta\mu$ as follows:*

$$E_{VIDE} \geq 2HLT\frac{\log(|\Delta\mu|/V\beta)}{\log\epsilon_V}(\frac{\mathrm{d}_{max}^{\alpha}\Theta N_0}{-\ln p_r})$$

where

$$\epsilon_V = 1 - \frac{8\gamma^*}{V^2}$$

and the other symbols have the same meanings as in the preceding theorems.

Proof Theorem 6.5 shows that

$$|\Delta\mu| \leq V\beta\epsilon_V^{\lfloor\frac{U}{LT}\rfloor}$$

which can be rewritten as:

$$\lfloor\frac{U}{LT}\rfloor \geq \frac{\log(|\Delta\mu|/V\beta)}{\log\epsilon_V}$$

This gives a lower bound to the number of updates required:

$$U \geq LT\frac{\log(|\Delta\mu|/V\beta)}{\log\epsilon_V}$$

Now, in each update operation, the node that initiates the update sends its current average estimate. The receiver node updates its average, and sends back the amount by which it has changed its average. This is done for each hidden variable. Thus the total number of floating point numbers communicated during one update is $2HU$. We also consider this to be the total number of messages required for an update, assuming that one message is required per floating point number.

It is shown in [33] that for each unit of communication, the amount of energy spent in transmission is $E_L = \frac{\mathrm{d}_{max}^{\alpha}\Theta N_0}{-\ln p_r}$ (our notation slightly differs from that in [33]). Thus the total amount of energy spent by the sensor network is bounded by:

$$E_{VIDE} \geq 2HUE_L$$

That is,

$$E_{VIDE} \geq 2HLT\frac{\log(|\Delta\mu|/V\beta)}{\log \epsilon_V}(\frac{\mathrm{d}_{max}^{\alpha}\Theta N_0}{-\ln p_r})$$

which completes the proof. $\square$

Our communication analysis shows that as the number of messages exchanged in the network increases, the error drops exponentially.In the next section, we present experimental results that validate this claim.

7 Experimental Study

This section presents the results of our experimental evaluation of VIDE. It begins by defining the objectives of the experiments; then it describes the experimental setup, and finally, it presents the results.

7.1 Objectives

Clearly, the accuracy of the variational approximation depends of the accuracy of the distributed average computation. In our experiments, we wanted to see how close the distributed variational approximation was to the variational approximation done centrally.

Let μ^c be the solution vector to the mean field equations computed by the centralized variational approximation. That is, μ_i^c is the mean of the i^{th} hidden random variable under this approximation. Let, in the distributed scenario, the solution vector to the mean field equations computed by the j^{th} node be $\mu^{(j)}$. Then the relative error for node j is $err(j) = ||\mu^c - \mu^{(j)}||/||\mu^c||$. The maximum relative error, then, is $max\_err = \max_j\{err(j)\}$.

We wish to observe the rate of decrease of max_err as the number of messages per node increases in the network.

Next, we discuss the experimental setup.

7.2 Experimental Setup

In each experiment, we first generated the parameters of the Boltzmann machine, by selecting the value of each parameter in a uniformly random way, from the interval $[0, 1)$. We use these parameter values both for the centralized and the distributed inferencing. Ideally, we would want to learn the Boltzmann machine parameters from a training data set, and then perform testing. However, we have not yet addressed the problem of learning these parameters in a heterogeneously distributed environment. So we test only the inferencing capability of VIDE: given an arbitrary setting of the parameter values, we want to know how accurate VIDE is compared to centralized variational inferencing.

Then, we generate a data-set in which each vector represents and assignment of binary values to the visible variables. For each such vector, the solution to the mean-field equations is computed both in a centralized way, and in a distributed way, using VIDE. The maximum relative error max_err is then computed (as discussed above).

We have tested VIDE using both synthetic data sets, and real life time series data sets, as described below.

Synthetic Data Sets Synthetic data sets consist of uniformly randomly generated binary vectors with V dimensions, where V is the network size, and hence, the number of visible variables. The number of hidden variables, H, has been varied as follows:

- For $V = 100$, we have selected $H \in \{1, 2, 3, 4, 5, 10\}$.
- For $V = 200$, we have selected $H \in \{2, 4, 6, 8, 10, 20\}$.

- For $V = 500$, we have selected $H \in \{5, 10, 15, 20, 25\}$.

Real-life Time Series Data Sets The real-life time series data sets used in this paper have been downloaded from (`http://www.cs.ucr.edu/~eamonn/time_series_data/`). We have used two data sets, Adiac and 50words. In future, we would like to test VIDE with many more real life data sets.

These data-sets consist of vectors of real numbers. We have transformed these vectors to binary vectors using *local gradient based discretization*. If the value of an attribute of the vector increases from one time-instant to the next, then in the binary version of the vector, that attribute is assigned a value 1, otherwise it is assigned a value 0.

The hidden and visible variables were chosen as follows:

- **Adiac.** Each vector, in this dataset, has 177 attributes. We chose the left-most 100 attributes to be visible, and the next H to be hidden, where $H \in \{1, 2, 3, 4, 5, 10\}$. The remaining attributes were ignored.
- **50words.** Each vector, in this dataset, has 271 attributes. We chose the leftmost 200 attributes to be visible, and the next H to be hidden, where $H \in \{2, 4, 6, 8, 10, 20\}$. The remaining attributes were ignored.

We now present the results of our experiments.

7.3 Experimental Results

As mentioned earlier, we wish to measure the accuracy of VIDE as a function of communication bandwidth. In each of the plots that follow, the x-axis represents the number of messages communicated by each node, where we assume that it takes a single message to communicate a floating point number. The y-axis represents the maximum relative error, max_err, averaged over a specified number of trials (also mentioned in the figures).

First we present the results for synthetically generated test data.

Experiments with Synthetic Data The dependence of max_err on the number of messages per node, for networks of size $V = 100$, $V = 200$, and $V = 500$ are demonstrated in Figure 1, 2 and 3 respectively.

Experiments with Real-life Time Series Data The dependence of max_err on the number of messages per node, for networks of size $V = 100$ (correspond-

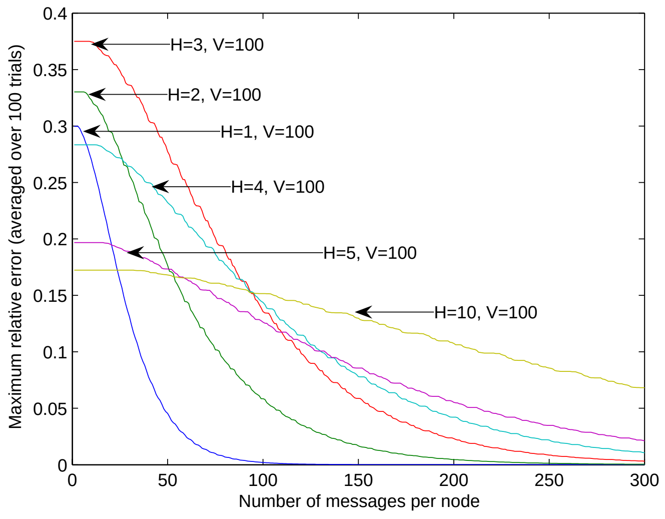

Fig. 1. Error vs. communication cost for V=100, using synthetic data.

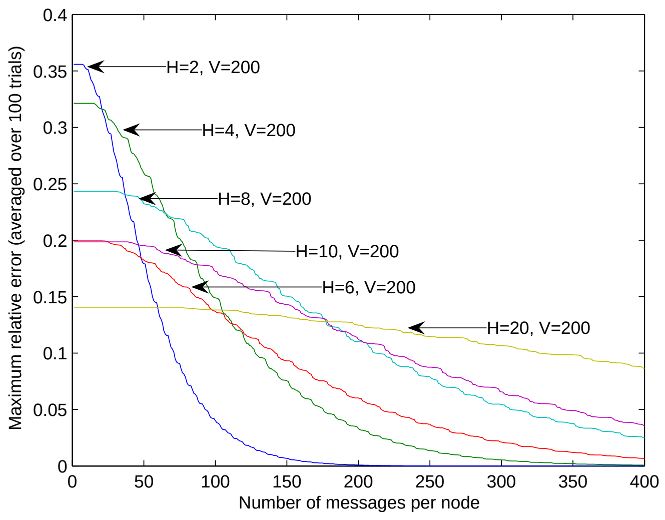

Fig. 2. Error vs. communication cost for V=200, using synthetic data.

ing to data set Adiac) and $V = 200$ (corresponding to data set 50words) are demonstrated in Figure 4 and 5 respectively.

7.4 Remarks

Let us consider the experiment with synthetic data, with number of visible variables $V = 100$. As indicated by Figure 1, the maximum relative error falls exponentially, as the number of messages per node increases. This means that the behavior of centralized variational inference can be closely approximated even with a modest amount of communication.

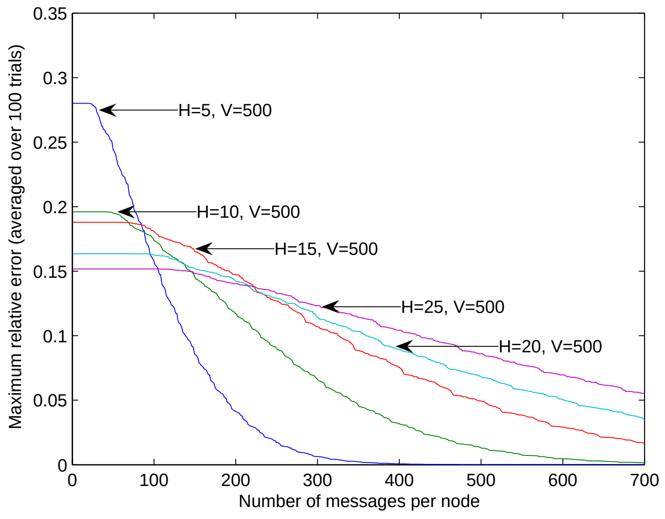

Fig. 3. Error vs. communication cost for V=500, using synthetic data.

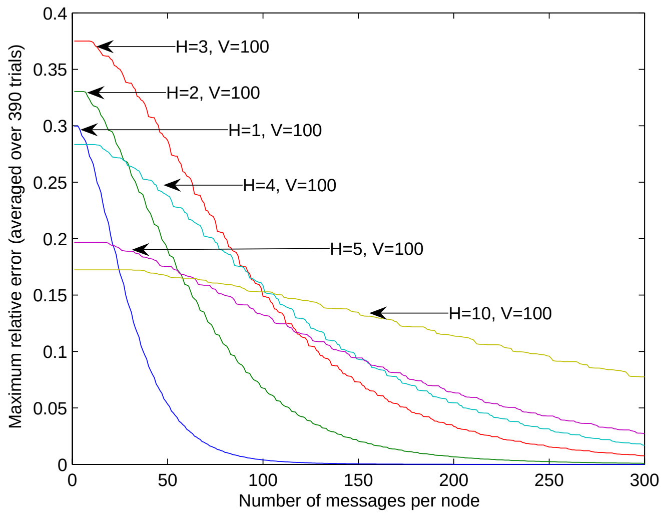

Fig. 4. Error vs. communication cost for V=100 using Adiac data set.

Another interesting parameter to note from Figure 1 is the rate at which the maximum relative error decreases as the number of messages per node increases. It is seen Figure 1 that this rate decreases, as the number of hidden variables, H, increases .At $H = 1$, the maximum relative error falls very sharply. But at $H = 10$, the rate of decrease is very low.

The reason for this is that, during the distributed average computation phase, the amount of data communicated between a pair of nodes, during each update operation, increases linearly with H. This means, for the same communication bandwidth allowance, the number of update operations will be higher, leading to greater accuracy, if H is lower. In particular, let U be the number of updates. Each pairwise update involves the exchange of $2H$ floating point numbers.

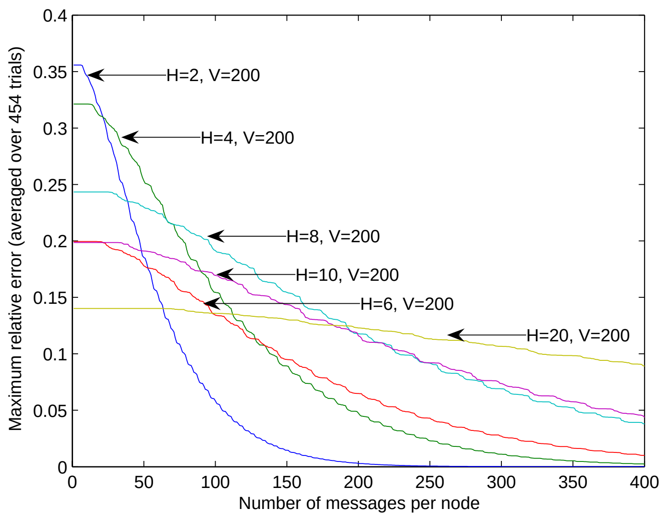

Fig. 5. Error vs. communication cost for V=200 using 50words data set.

Assuming that one message is required for communicating each floating point number, the total number of messages required, then, is $2HU$ (as shown in the proof of Theorem 6.6), and the number of messages per node is $n = 2HU/V$. We can rewrite this as

$$U = \frac{1}{2}n\frac{V}{H}$$

Thus, for the same number of messages per node, the number of updates is greater when the V/H is greater, that is, when H is smaller compared to V.

The above analyis also suggests that if we change V and H keeping their ratio constant, then the rate of decrease of the maximum relative error should be comparable.

This conjecture is strengthened by the results reported in Figure 2, Figure 3, and even by the experiments done with real-life data, as shown in Figure 4 and Figure 5. We see that the curves corresponding to $H = 1, V = 100$ (Figure 1), $H = 2, V = 200$ (Figure 2), $H = 5, V = 500$ (Figure 3), $H = 1, V = 100$ (Figure 4) and $H = 2, V = 200$ (Figure 5) all have the same shape. Moreover, they correspond to the same value of V/H which is 100. Other sets of curves having the same V/H have the same property: the rate of decrease of maximum relative error is similar.

We next present experimental results to demonstrate the scalability of VIDE.

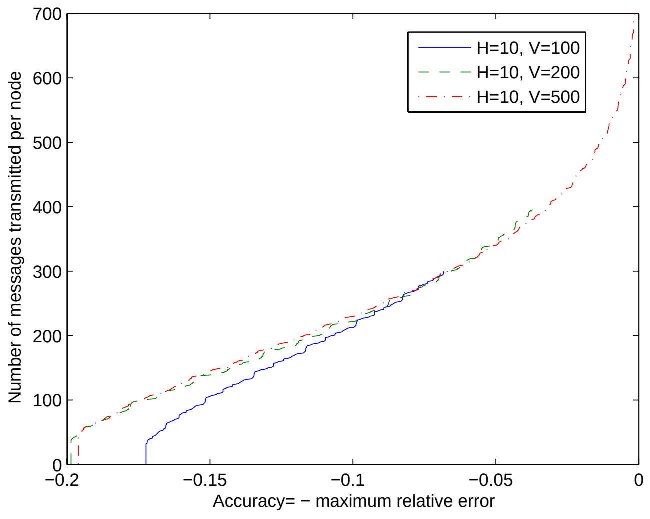

Fig. 6. Scalability of VIDE – experiments with synthetic data. Here H and V represent the number of hidden variables and the number of visible variables respectively. V is also the size of the network.

7.5 Experimental Results: Scalability

In this section, we investigate the scalability of VIDE. In order to evaluate the scalability of VIDE, we need to examine how the number of messages transmitted per node increases with network size, in order to achieve a given level of accuracy, for a given number of hidden variables. We consider scalability both for synthetic and real-life time-series data.

Scalability: Synthetic Data. The scalability of VIDE with synthetic data is demonstrated in Figure 6. In this figure, the x-axis denotes the desired level of accuracy. Since accuracy increases as the maximum relative error decreases, we define accuracy as the negative of the maximum relative error. In the y-axis, we plot the number of messages necessary to transmit per node, for various network sizes.

Remark. It is seen from Figure 6 that for reasonably high levels of accuracy, the plots for the three network sizes are very close to each other. This indicates that for a given accuracy level, the number of messages transmitted per node increases very slightly, as the network becomes larger. This demonstrates that VIDE is extremely scalable with synthetic data.

Next, we investigate the scalability of VIDE with real-life time-series data.

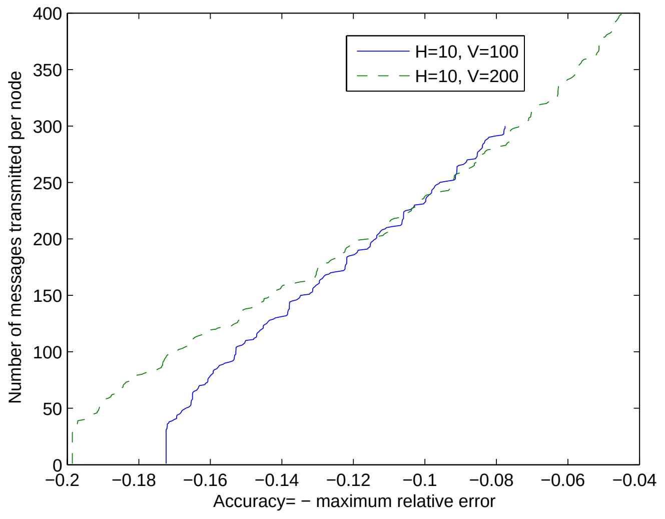

Fig. 7. Scalability of VIDE – experiments with real-life time-series data. Here H and V represent the number of hidden variables and the number of visible variables respectively. V is also the size of the network.

Scalability: Real-life time-series data. For real-life time series experiments, we select the same data sets, Adiac and 50words, as described earlier. The scalability of VIDE with real-life data is demonstrated in Figure 7. In this figure, the x-axis denotes the desired level of accuracy. Since accuracy increases as the maximum relative error decreases, we define accuracy as the negative of the maximum relative error. In the y-axis, we plot the number of messages necessary to transmit per node, for various network sizes. For $V = 100$, we use the Adiac dataset; for $V = 200$, we use the 50words dataset.

Remark. It is seen from Figure 7 that even when the network size is doubled, the number of messages transmitted per node, for a given level of accuracy, increases very slightly. This indicates that VIDE is scalable even with real-life data.

To summarize, the experimental results show that the communication requirements for VIDE are modest; even with limited communication-allowance, VIDE can achieve sufficiently accurate results. Moreover, VIDE is a scalable technique, because the communication requirements per node increases at a very slow rate, as the network becomes larger. Thus, our experimental study establishes VIDE as a deterministic approximation technique for performing distributed inferencing, in a communication-efficient and scalable manner.This seems to empirically back up the claim for local behavior. The next section concludes the paper.

8 Conclusion and Future Work

Distributed data mining problems are often challenging because of communication bandwidth limitations. A lot of contemporary research in this field aims at finding probabilistic approximation algorithms for solving these problems. This paper takes a different approach, and explores the applicability of deterministic approximations to distributed data mining problems. In particular, it demonstrates the applicability of a deterministic approximation technique, the variational method, to the problem of distributed inferencing in a graphical model. It extends the variational formulation of the probabilistic inferencing problem to the heterogeneously distributed scenario. Based on the variational formulation, it presents a deterministic approximation algorithm, VIDE, for distributed inferencing. The theoretical analysis and experimental results shown in this paper indicate that VIDE is a communication-efficient solution to the problem of probabilistic inferencing in heterogeneously distributed environments.

Natural extensions of this work would involve exploring whether other distributed data mining problems can be formulated using the variational optimization framework, and whether the variational method leads to communication efficient techniques for solving these problems. We would like to investigate these issues in the future.

Acknowledgment

The authors would like to thank partial support provided by the National Science Foundation Career Award, IIS-0093353.

The authors would also like to thank Dr. Eamonn Keogh for making available useful time series data sets at `http://www.cs.ucr.edu/~eamonn/time_series_data/`.

References

[1] C. Lin, W. Peng, Y. TSeng. Efficient In-Network Moving Object Tracking in Wireless Sensor Networks. *IEEE Transactions on Mobile Computing* (to appear).

[2] E.B. Ermis, V. Saligrama. Detection and Localization in Sensor Networks Using Distributed FDR. *Conference on Information Sciences and Systems (CISS)*, 2006 .

[3] M. Paskin, C. Guestrin, J. McFadden. A Robust Architecture for Inference in Distributed Sensor Networks. *International Conference on Information Processing in Sensor Networks (IPSN)*,2005.

[4] M.A. Paskin and C.E. Guestrin. Robust Probabilistic Inference in Distributed Systems. *Uncertainty in Artificial Intelligence 20*, 2004.

[5] V. Byckovskiy et al. A Collaborative Approach to In-Place Sensor Calibration. *International Conference on Information Processing in Sensor Networks (IPSN)*, 2003.

[6] M. Jordan et al. An Introduction To Variational Methods For Graphical Models. To appear in *Learning in Graphical Models*,Kluwer Academic Publishers.

[7] T. S. Jaakkola. Tutorial on Variational Approximation Methods. *Advanced Mean Field Methods - Theory and Practice* (Editors: D. Saad and M. Opper), MIT Press.

[8] D. McKay. Variational Methods. *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, 2003.

[9] J.J. Verbeek, N. Vlassis, J.R.J. Nunnink. A Variational EM Approach for Large-Scale Mixture Modelling. *Proceedings of* 8^{th} *Annual Conference of the Advanced School of Computing and Imaging, Heijen*, The Netherlands, June 2003.

[10] W. Kowalczyk and N. Vlassis. Newscast EM. *Advances in Neural Information Processing Systems 17*, MIT Press, 2005.

[11] F.V. Jensen. *An Introduction to Bayesian Networks.* London UCL Press, 1996.

[12] L. Chen, M. Wainwright, M. Cetin, A. Willsky. Data Association based on Optimization in Graphical Models with Application to Sensor Networks. *Mathematical and Computer Modelling, Special Issue on Optimization and Control for Military Applications*, to appear.

[13] C. Crick, A. Pfeffer. "Loopy Belief Propagation as a Basis for Communication with Sensor Networks. *Uncertainty in Artificial Intelligence 18*, August 2003.

[14] A. T. Ihler. Inference in Sensor Networks: Graphical Models and Particle Models. PhD Thesis.

[15] M. Jordan, C.M. Bishop. Neural Networks. *Computing Surveys* (in press).

[16] D.H. Ackley, G.E. Hinton, T.J. Sejnowski. A Learning Algorithm for Boltzmann Machines. *Cognitive Science 9*, 1985.

[17] C. Peterson, J.R. Anderson. A Mean Field Theory Learning Algorithm for Neural Networks. *Complex Systems 1*, 1987

[18] L.K. Saul, M.I. Jordan. Learning in Boltzmann Trees. *Neural Computation 6(6)*, 1994.

[19] S. M. Rüger. Decimatable Boltzmann Machines for Diagnosis: Efficient Learning and Inference. *IMACS World Congress*, 1997.

[20] K. Horowitz, D. Malkhi. Estimating Network Size from Local Information. *The Information Processing Letters Journal, 88(5):237-243*, December 2003.

[21] L. Lovasz. Random Walks on Graphs: A Survey. *Combinarotics, Paul Erdos is Eighty, 2:1-46*, 1993.

[22] H. Kargupta, I. Hamzaoglu, B. Stafford. Scalable, Distributed Data Mining Using an Agent-Based Architecture. *Proceedings of Knowledge Discovery and Data Mining*, Eds: D. Heckerman, H. Mannila, D. Pregibon and R. Uthurusamy, pp. 211 – 214, AAAI Press, 1997.

[23] H. Kargupta, B. Park, D. Hershberger, E. Johnson. Collective Data Mining: A New Perspective Toward Distributed Data Mining. *Advances in Distributed and Parallel Knowledge Discovery*, Eds: Hillol Kargupta and Philip Chan, MIT/AAAI Press, 1999.

[24] H. Kargupta K. Sivakumar. Existential Pleasures of Distributed Data Mining. *Data Mining: Next Generation Challenges and Future Directions*, Eds: H. Kargupta, A. Joshi, K. Sivakumar, Y. Yesha. AAAI/MIT Press, 2004.

[25] R. Wolff, A. Schuster. Association Rule Mining in Peer-to-peer Systems. *IEEE Transactions on Systems, Man and Cybernetics, Part B, 34(6):2426-2438*, December 2004.

[26] S. Dutta, C.Gianella, H. Kargupta. K-Means Clustering over Peer-to-peer Networks. 8^t*h International Workshop on High Performance and Distributed Mining, SIAM International Conference on Data Mining*, 2005.

[27] C. Giannella, H. Dutta, S. Mukherjee, H. Kargupta. Efficient Kernel Density Estimation Over Distributed Data. 9^{th} *International Workshop on High Performance and Distributed Mining, SIAM International Conference on Data Mining*, Bethesda, USA, April, 2006.

[28] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, R. Murray. Distributed Averaging on Peer-to-Peer Networks. *The Fourth International Symposium on Information Processing in Sensor Networks*, Los Angeles, California, USA, April 2005.

[29] J. J. Hopfield. Learning Algorithms and Probability Distributions in Feed-Forward and Feed-Back Networks. *Proceedings of the National Academy of Sciences of the United States of America, 84(23):8429-8433*, December 1987.

[30] G. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. Technical Report: GCNU TR 2000-004, 2000.

[31] Y. Teh, G. Hinton Rate-coded Restricted Boltzmann Machines for Face Recognition. *Advances in Neural Information Processing Systems 13*, MIT Press, Cambridge, MA, 2001.

[32] S. A. Zabidi, M. E. Salami. Design and Development of Intelligent Fingerprint-Based Security System *Knowledge-Based Intelligent Information and Engineering Systems, 3214/2004: 312-318*, Lecture Notes in Computer Science, Springer Berlin/ Heidelberg, October 2004.

[33] X. Liu, M. Haenggi. Performance Analysis of Rayleigh Fadin Ad Hoc Networks with Regular Topology. *Global Telecommunications Conference 5, 2005. GLOBECOM '05. IEEE* November-December 2005.

[34] R. Motwani and P. Raghavan. Randomized Algorithms. Cambridge University Press, New York (NY), 1995.