Learning Graphical Models from a Distributed Stream

Yu Zhang #[1], Srikanta Tirthapura #[2], Graham Cormode *

# *Electrical and Computer Engineering Department, Iowa State University*
[1] yuz1988@iastate.edu [2] snt@iastate.edu
* *University of Warwick,* g.cormode@warwick.ac.uk

***Abstract*—A current challenge for data management systems is to support the construction and maintenance of machine learning models over data that is large, multi-dimensional, and evolving. While systems that could support these tasks are emerging, the need to scale to distributed, streaming data requires new models and algorithms. In this setting, as well as computational scalability and model accuracy, we also need to minimize the amount of communication between distributed processors, which is the chief component of latency.**

We study Bayesian Networks, the workhorse of graphical models, and present a communication-efficient method for continuously learning and maintaining a Bayesian network model over data that is arriving as a distributed stream partitioned across multiple processors. We show a strategy for maintaining model parameters that leads to an exponential reduction in communication when compared with baseline approaches to maintain the exact MLE (maximum likelihood estimation). Meanwhile, our strategy provides similar prediction errors for the target distribution and for classification tasks.

I. Introduction

With the increasing need for large scale data analysis, distributed machine learning [?] has grown in importance in recent years, leading to the development of platforms such as Spark MLlib [?], Tensorflow [?] and Graphlab [?]. Raw data is described by a large number of interrelated variables, and an important task is to describe the joint distribution over these variables, allowing inferences and predictions to be made. For example, consider a large-scale sensor network where each sensor is observing events in its local area (say, vehicles across a highway network; or pollution levels within a city). There can be many factors associated with each event, such as duration, scale, surrounding environmental conditions and many other features collected by the sensor. However, directly modeling the full joint distribution of all these features may be infeasible, since the complexity of such a model grows exponentially with the number of variables. For instance, the complexity of a model with n variables, each taking one of J values is $O(J^n)$ parameters. The most common way to tame this complexity has to use a graphical model to compactly encode conditional dependencies among variables in the data, and so reduce the number of parameters.

We focus on *Bayesian Networks*, a general and widely used class of graphical models. A Bayesian network can be represented as a directed acyclic graph (DAG), where each node represents a variable and an edge directed from one node to another represents a conditional dependency between the corresponding variables. Bayesian networks have found applications in numerous domains, such as decision making [?], [?] and cybersecurity [?], [?]. In these domains, new training examples can arrive online, and it is important to incorporate new data into the model as it arrives. For instance, in malware classification, as more data observed, the Bayesian network can be adjusted in an online manner to better classify future inputs as either benign or malicious.

While a graphical model can help in reducing complexity, the number of parameters in such a model can still be quite high, and tracking each parameter independently is expensive, especially in a distributed system that sends a message for each update. The key insight in our work is that it is not necessary to log every event in real time; rather, we can aggregate information, and only update the model when there is a substantial change in the inferred model. This still allows us to continuously maintain the model, but with substantially reduced communication. In order to give strong approximation guarantees for this approach, we delve deeper into the construction of Bayesian Networks.

The fundamental task in building a Bayesian Network is to estimate the conditional probability distribution (CPD) of a variable given the values assigned to its parents. Once the CPDs of different variables are known, the joint distribution can be derived over any subset of variables using the chain rule [?]. To estimate the CPDs from empirical data, we use the maximum likelihood estimation (MLE) principle. The CPD of each event can be obtained by the ratio of the prevalence of that event versus the parent event (for independent variables, we obtain the single variable distribution). Thus the central task is to obtain accurate counts of different subsets of events. Following the above discussion, the problem has a tantalizingly clear solution: to materialize the needed frequency counts in order to estimate the CPDs accurately.

Modern data analysis systems deal with massive, dynamic, and distributed data sources, such as network traffic monitors and large-scale sensor networks. On such sources, the simple solution of centralizing all data would incur a very high network communication cost. As observed in prior work on distributed stream monitoring [?], [?], it is important to minimize network communication in order to scale to streams of higher velocity and variety. Our technical challenge is to design a scheme that can accurately maintain the Bayesian Network model, while minimizing the communication incurred.

We formalize the problem using the continuous distributed stream monitoring model [?]. There are many sites, each receiving an individual stream of observations i.e. we assume the data is horizontally partitioned. A separate coordinator node, which receives no input itself, interacts with the sites to collaboratively maintain a model over the union of all data seen so far, and also answers queries. This model captures many of the difficulties that arise in learning tasks in big data systems – data is large, streaming in, and distributed over many sites; and models need to be maintained in a timely manner allowing for real-time responses.

Our work makes extensive use of a primitive called a *distributed counter*, which enables accurate event counting without triggering a message for each event. We first show a basic monitoring scheme that uses distributed counters independently for each variable in the model. However, our strongest results arise when we provide a deeper technical analysis of how the counts combine, to give tighter accuracy guarantees with a lower communication cost. The resulting exponential improvements in the worst-case cost for this task are matched by dramatic reductions observed in practice. In more detail, our contributions are as follows:

Contributions. We present the first communication-efficient algorithms that continuously maintain a graphical model over distributed data streams.

— Our algorithms maintain an accurate approximation of the Maximum Likelihood Estimate (MLE) using communication cost that is only *logarithmic* in the number of distributed observations. This is in contrast with the approach that maintains an exact MLE using a communication cost *linear* in the number of observations.

— Our communication-efficient algorithms provide a provable guarantee that the model maintained is close to the MLE model given current observations, in a precise sense (Sections **??**, **??**).

— We present three algorithms, in increasing order of ability to capture model parameters, BASELINE, UNIFORM, and NONUNIFORM in Section **??**. All three follow a similar outline, and differ in how they allocate internal parameters that control the accuracy of approximation of different quantities. NONUNIFORM has the most involved analysis (but is straightforward to implement) to handle the case when the sizes of the CPDs of different random variables may be very different from each other. Section **??** shows how these algorithms apply to typical machine learning tasks such as classification.

— We present an evaluation, both using simulations as well as implementation over a cluster, in showing that on a stream of a few million distributed training examples, our methods resulted in an improvement of 100-1000x in communication cost over the maintenance of exact MLEs, while providing estimates of joint probability with nearly the same accuracy as obtained by exact MLEs.

This provides a method for communication-efficient maintenance of a graphical model over distributed, streaming data. Prior works on maintaining a graphical model have considered efficiency in terms of space (memory) and time, but these costs tend to be secondary when compared to the communication cost in a distributed system. Our method is built on the careful combination of multiple technical pieces. Since the overall joint distribution is formed by composing many CPDs, we divide the maximum "error budget" among the different parameters within the different CPDs so that (a) the error of the joint distribution is within the desired budget, and (b) the communication cost is as small as possible. We pose this as a convex optimization problem and use its solution to parameterize the algorithms for distributed counters. The next advance is to leverage concentration bounds to argue that the aggregate behavior of the approximate model consisting of multiple random variables (each estimating a parameter of a CPD) is concentrated within a small range. As a result, the dependence of the communication cost on the number of variables n can be brought down from $O(n)$ to $O(\sqrt{n})$.

II. PRIOR AND RELATED WORK

Many recent works are devoted to designing algorithms with efficient communication in distributed machine learning. Balcan *et al.* [?] were perhaps the first to give formal consideration to this problem, based on the model of PAC (Probably Approximately Correct) learning. They showed lower bounds and algorithms for the non-streaming case, where k parties each hold parts of the input, and want to collaborate to compute a model. We call this "the static distributed model". Daumé *et al.* [?] considered a distributed version of the classification problem: training data points are assigned labels, and the goal is to build a model to predict labels for new examples. Algorithms are also proposed in the static distributed model, where the classifiers are linear separators (hyperplanes) allowing either no or small error. Most recently, Chen *et al.* [?] considered spectral graph clustering, and showed that the trivial approach of centralizing all data can only be beaten when a broadcast model of communication is allowed.

In the direction of lower bounds, Zhang *et al.* [?] considered the computation of statistical estimators in the static distributed model, and show communication lower bounds for minimizing the expected squared error, based on information theory. Phillips *et al.* [?] show lower bounds using communication complexity arguments via the "number in hand" model. Various functions related to machine learning models are shown to be "hard" i.e., require large amounts of communication in the distributed model .

Some previous works have extended sketching techniques to the problem of streaming estimation of parameters of a Bayesian network. McGregor and Vu [?] gave sketch-based algorithms to measure whether given data was "consistent" with a prescribed model i.e. they compare the empirical probabilities in the full joint distribution with those that arise from fitting the same data into a particular Bayesian network. They also provide a streaming algorithm that finds a good degree-one Bayesian network (i.e. when the graph is a tree). Kveton *et al.* [?] adapt sketches to allow estimation of parameters for models that have very high-cardinality variables. However, neither of these methods consider the distributed setting.

The continuous distributed monitoring model has been well studied in the data management and algorithms communities, but there has been limited work on machine learning problems in this model. A survey of the model and basic results is given in [**?**]. Efficient distributed counting is one of the first problems studied in this model [**?**], and subsequently refined [**?**], [**?**]. The strongest theoretical results on this problem are randomized algorithms due to Huang *et al.* [**?**]. Generic techniques are introduced and studied by Sharfman *et al.* [**?**]. Some problems studied in this model include clustering [**?**], anomaly detection [**?**], entropy computation [**?**] and sampling [**?**].

III. Preliminaries

Let $\mathbb{P}\left[E\right]$ denote the probability of event E. For random variable X, let $\mathrm{dom}(X)$ denote the domain of X. We use $\mathbb{P}\left[x\right]$ as a shorthand for $\mathbb{P}\left[X = x\right]$ when the random variable is clear from the context. For a set of random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ let $\mathbb{P}\left[X_1, \ldots, X_n\right]$ or $\mathbb{P}\left[\mathcal{X}\right]$ denote the joint distribution over $\mathcal{X}$. Let $\mathrm{dom}(\mathcal{X})$ denote the set of all possible assignments to $\mathcal{X}$.

Definition 1: A Bayesian network $\mathcal{G} = (\mathcal{X}, \mathcal{E})$ is a directed acyclic graph with a set of nodes $\mathcal{X} = \{X_1, \ldots, X_n\}$ and edges $\mathcal{E}$. Each X_i represents a random variable. For $i \in [1, n]$, let $\mathsf{par}\left(X_i\right)$ denote the set of parents of X_i and $\mathrm{NonDescendants}\left(X_i\right)$ denote the variables that are not descendants of X_i. The random variables obey the following condition: for each $i \in [1, n]$, X_i is conditionally independent of $\mathrm{NonDescendants}\left(X_i\right)$, given $\mathsf{par}\left(X_i\right)$.
For $i = 1 \ldots n$, let J_i denote the size of $\mathrm{dom}(X_i)$ and K_i the size of $\mathrm{dom}(\mathsf{par}\left(X_i\right))$.

Conditional Probability Distribution. Given a Bayesian Network on $\mathcal{X}$, the joint distribution can be factorized as:

$$\mathbb{P}\left[\mathcal{X}\right] = \textstyle\prod_{i=1}^{n} \mathbb{P}\left[X_i \mid \mathsf{par}\left(X_i\right)\right] \tag{1}$$

For each i, $\mathbb{P}\left[X_i \mid \mathsf{par}\left(X_i\right)\right]$ is called the *conditional probability distribution (CPD)* of X_i. Let θ_i denote the CPD of X_i and $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_n\}$ the set of CPDs of all variables.

Given training data $\mathcal{D}$, we are interested in obtaining the **maximum likelihood estimate (MLE) of $\boldsymbol{\theta}$.** Suppose that $\mathcal{D}$ contains m instances $\xi[1], \ldots, \xi[m]$. Let $L(\boldsymbol{\theta} \mid \mathcal{D})$, the likelihood function of $\boldsymbol{\theta}$ given the dataset $\mathcal{D}$, be equal to the probability for dataset observed given those parameters.

$$L(\boldsymbol{\theta} \mid \mathcal{D}) = \mathbb{P}\left[\mathcal{D} \mid \boldsymbol{\theta}\right]$$

Let $L_i(\theta_i \mid \mathcal{D})$ denote the likelihood function for θ_i. The likelihood function of $\boldsymbol{\theta}$ can be decomposed as a product of independent local likelihood functions.

$$L(\boldsymbol{\theta} \mid \mathcal{D}) = \textstyle\prod_{i=1}^{n} L_i(\theta_i \mid \mathcal{D})$$

Let $\hat{\boldsymbol{\theta}}$ denote the value of $\boldsymbol{\theta}$ that maximizes the likelihood function, $\hat{\boldsymbol{\theta}}$ is also known as the Maximum Likelihoood Estimation (MLE) of $\boldsymbol{\theta}$. Similarly, let $\hat{\theta}_i$ denote the value of θ_i that maximizes $L_i(\theta_i \mid \mathcal{D})$.

*Lemma 1 ([**?**, proposition* 17.1*]):* Consider a Bayesian Network with given structure $\mathcal{G}$ and training dataset $\mathcal{D}$. Suppose for all $i \neq j$, θ_i and θ_j are independent. For each $i \in [1, n]$, if $\hat{\theta}_i$ maximizes the likelihood function $L_i(\theta_i : \mathcal{D})$, then $\hat{\boldsymbol{\theta}} = \{\hat{\theta}_1, \ldots, \hat{\theta}_n\}$ maximizes $L(\boldsymbol{\theta} : \mathcal{D})$.

Local CPD Estimation. In this work, we consider categorical random variables, so that the CPD of each variable X_i can be represented as a table, each entry is the probability $\mathbb{P}_i\left[x_i \mid \boldsymbol{x}_i^{par}\right]$ where x_i is the value of X_i and $x_i \in \mathrm{dom}(X_i)$, $\boldsymbol{x}_i^{par}$ is the vector of values on the dimensions corresponding to $\mathsf{par}\left(X_i\right)$ and $\boldsymbol{x}_i^{par} \in \mathrm{dom}(\mathsf{par}\left(X_i\right))$.

We can handle continuous valued variables by appropriate discretization, for example through applying a histogram, with bucket boundaries determined by domain knowledge, or found by estimation on a random sample.

*Lemma 2 ([**?**, Section* 17.2.3*]):* Given a training dataset $\mathcal{D}$, the maximum likelihood estimation (MLE) for θ_i is $\hat{\theta}_i(x_i \mid \boldsymbol{x}_i^{par}) = \frac{F_i(x_i, \boldsymbol{x}_i^{par})}{F_i(\boldsymbol{x}_i^{par})}$ where $F_i(x_i, \boldsymbol{x}_i^{par})$ is the number of events $(X_i = x_i, \mathsf{par}\left(X_i\right) = \boldsymbol{x}_i^{par})$ in $\mathcal{D}$, $F_i(\boldsymbol{x}_i^{par})$ is the number of events $(\mathsf{par}\left(X_i\right) = \boldsymbol{x}_i^{par})$ in $\mathcal{D}$.

From Lemma **??**, a solution that maximizes the local likelihood functions also maximizes the joint likelihood function. We further have that the MLE is an accurate estimate of the ground truth when the training dataset is sufficiently large.

*Lemma 3 ([**?**, Corollary* 17.3*]):* Given a Bayesian Network $\mathcal{G}$ on $\mathcal{X}$, let P^* denote the ground truth joint distribution consistent with $\mathcal{G}$ and $\hat{P}$ the joint distribution using MLE. Suppose $\mathbb{P}_i\left[x_i \mid \boldsymbol{x}_i^{par}\right] \geq \lambda$ for all $i, x_i, \boldsymbol{x}_i^{par}$. If $m \geq \frac{(1+\epsilon)^2}{2\lambda^{2(d+1)}\epsilon^2} \log \frac{nJ^{d+1}}{\delta}$ then $\mathbb{P}\left[e^{-n\epsilon} \leq \frac{\hat{P}}{P^*} \leq e^{n\epsilon}\right] > 1 - \delta$, where $J = \max_{i=1}^{n} J_i$, and $d = \max_{i=1}^{n} \left|\mathsf{par}\left(X_i\right)\right|$.

Approximate Distributed Counters. We make use of a randomized algorithm to continuously track counter values in the distributed monitoring model, due to [**?**].

*Lemma 4 ([**?**]):* Consider a distributed system with k sites. Given $0 < \epsilon < 1$, for $k \leq \frac{1}{\epsilon^2}$, there is a randomized distributed algorithm $\textsc{DistCounter}\left(\epsilon, \delta\right)$ that continuously maintains a distributed counter $\mathcal{A}$ with the property that $\mathbb{E}\left[\mathcal{A}\right] = \mathcal{C}$ and $\mathbf{Var}\left[\mathcal{A}\right] \leq (\epsilon\mathcal{C})^2$, where $\mathcal{C}$ is the exact value being counted. The communication cost is $O\left(\frac{\sqrt{k}}{\epsilon} \cdot \log T\right)$ messages, where T is the maximum value of $\mathcal{C}$. The algorithm uses $O(\log T)$ space at each site and $O(1)$ processing time per instance received.

Our Objective: Approximation to the MLE. Given a continuously changing data stream, exact maintenance of the MLE of the joint distribution is expensive communication-wise, since it requires the exact maintenance of multiple distributed counters, each of which may be incremented by many distributed processors. Hence, we consider the following notion of approximation to the MLE.

Definition 2: Consider a Bayesian Network $\mathcal{G}$ on $\mathcal{X}$. Let $\hat{P}\left[\cdot\right]$ denote the MLE of the joint distribution of $\mathcal{X}$. Given approximation factor $0 < \epsilon < 1$, an ϵ-approximation to the MLE is a joint probability distribution $\tilde{P}\left[\cdot\right]$ such that, for any assignment of values $\boldsymbol{x}$ to $\mathcal{X}$, $e^{-\epsilon} \leq \frac{\tilde{P}(\boldsymbol{x})}{\hat{P}(\boldsymbol{x})} \leq e^{\epsilon}$. Given an additional parameter $0 < \delta < 1$, a distribution $\tilde{P}$ is an (ϵ, δ)-approximation to MLE if it is an ϵ-approximation to the MLE with probability at least $1 - \delta$.

Our goal is to maintain a distribution $\tilde{P}$ that is an (ϵ, δ)-approximation to the MLE, given all data observed so far, in the distributed continuous model.

The task of choosing the graph $\mathcal{G}$ with which to model the data (i.e. which edges are present in the network and which are not) is also an important one, but one that we treat as orthogonal to our focus in this work. For data of moderate dimensionality, we may assume that the graph structure is provided by a domain expert, based on known structure and independence within the data. Otherwise, the graph structure can be learned offline based on a suitable sample of the data. The question of learning graph models "live" as data arrives, is a challenging one that we postpone to future work.

IV. Distributed Streaming MLE Approximation

Continuous maintenance of the MLE requires continuous maintenance of a number of counters, to track the different (empirical) conditional probability distributions.

For each $x_i \in \mathrm{dom}(X_i)$ and $\boldsymbol{x}_i^{par} \in \mathrm{dom}(\mathsf{par}\,(X_i))$, let $\mathcal{C}_i(\boldsymbol{x}_i^{par})$ be the counter that tracks the number of events $(\mathsf{par}\,(X_i) = \boldsymbol{x}_i^{par})$, and let $\mathcal{C}_i(x_i, \boldsymbol{x}_i^{par})$ be the counter that tracks the number of events $(X_i = x_i, \mathsf{par}\,(X_i) = \boldsymbol{x}_i^{par})$. When clear from the context, we use the counter to also denote its value when queried. Consider any input vector $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$. For $1 \leq i \leq n$, let $\boldsymbol{x}_i^{par}$ denote the projection of vector $\boldsymbol{x}$ on the dimensions corresponding to $\mathsf{par}\,(X_i)$. Based on Equation **??** and Lemma **??**, the empirical joint probability $\hat{P}\,[\boldsymbol{x}]$ can be factorized as:

$$\hat{P}\,[\boldsymbol{x}] = \prod_{i=1}^{n} \frac{\mathcal{C}_i(x_i, \boldsymbol{x}_i^{par})}{\mathcal{C}_i(\boldsymbol{x}_i^{par})} \qquad (2)$$

A. Strawman: Using Exact Counters

A simple solution to maintain parameters is to maintain each counter $\mathcal{C}_i(\cdot)$ and $\mathcal{C}_i(\cdot, \cdot)$ exactly at all times, at the coordinator. With this approach, the coordinator always has the MLE of the joint distribution, but the communication cost quickly becomes the bottleneck of the whole system. Each time an event is received at a site, the site tells the coordinator to update the centralizing parameters θ immediately, essentially losing any benefit of distributed processing.

Lemma 5: If exact counters are used to maintain the MLE of a Bayesian network on n variables in the distributed monitoring model, the total communication cost to continuously maintain the model over m event observations is $O(mn)$, spread across m messages of size n.

B. Master Algorithms Using Approximate Counters

The major issue with using exact counters to maintain the MLE is the communication cost, which increases linearly with the number of events received from the stream. We describe a set of "master" algorithms that we use to approximately track statistics, leading to a reduced communication cost, yet maintaining an approximation of the MLE. In successive sections we tune their parameters and analysis to improve their behavior. In Section **??**, we describe the Baseline algorithm which divides the error budget uniformly and pessimistically across all variables. Section **??** gives the Uniform approach, which keeps the uniform allocation, but uses an improved randomized analysis. Finally, the NonUniform algorithm in Section **??** adjusts the error budget allocation to account for the cardinalities of different variables.

Algorithm 1: Init(n, epsfnA, epsfnB)

```
/* Initialization of Distributed Counters.                    */
Input: $n$ is the number of variables. epsfnA and epsfnB
       are parameters of initialization functions provided
       by specific algorithms.
foreach $i$ from $1$ to $n$ do
    foreach $x_i \in \mathrm{dom}(X_i)$, $\boldsymbol{x}_i^{par} \in \mathrm{dom}(\mathsf{par}\,(X_i))$ do
        $\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par}) \leftarrow$ DistCounter(epsfnA($i$), $\delta$)
    foreach $\boldsymbol{x}_i^{par} \in \mathrm{dom}(\mathsf{par}\,(X_i))$ do
        $\mathcal{A}_i(\boldsymbol{x}_i^{par}) \leftarrow$ DistCounter(epsfnB($i$), $\delta$)
```

Algorithm 2: Update($\boldsymbol{x}$)

```
/* Called by a site upon receiving a new event                */
Input: $\boldsymbol{x} = \langle x_1, \ldots, x_d \rangle$ is an observation.
foreach $i$ from $1$ to $n$ do
    Increment $\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})$
    Increment $\mathcal{A}_i(\boldsymbol{x}_i^{par})$
```

These algorithms build on top of approximate distributed counters (Lemma **??**), denoted by $\mathcal{A}$. At any point, the coordinator can answer a query over the joint distribution by using the outputs of the approximate counters, rather than the exact values of the counters (which it no longer has access to). We have the following objective:

Definition 3 (MLE Tracking Problem): Given $0 < \epsilon < 1$, for $i \in [1, n]$, we seek to maintain distributed counters $\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})$ and $\mathcal{A}_i(\boldsymbol{x}_i^{par})$ such that for any data input vector $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, we have

$$e^{-\epsilon} \leq \frac{\tilde{P}(\boldsymbol{x})}{\hat{P}(\boldsymbol{x})} = \prod_{i=1}^{n} \left(\frac{\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})}{\mathcal{C}_i(x_i, \boldsymbol{x}_i^{par})} \cdot \frac{\mathcal{C}_i(\boldsymbol{x_i})}{\mathcal{A}_i(\boldsymbol{x}_i^{par})} \right) \leq e^{\epsilon}$$

Our general approach is as follows. Each algorithm initializes a set of distributed counters (DistCounter in Algorithm **??**). Once a new event is received, we update the two counters associated with the CPD for each variable (Algorithm **??**). A query is processed as in Algorithm **??** by probing the approximate CPDs. The different algorithms are specified based on how they set the error parameters for the distributed counters, captured in the functions epsfnA and epsfnB.

C. Baseline *Algorithm Using Approximate Counters*

Our first approach Baseline, sets the error parameter of each counter $\mathcal{A}(\cdot)$ and $\mathcal{A}(\cdot, \cdot)$ to a value $\frac{\epsilon}{3n}$, which is small enough so that the overall error in estimating the MLE is

Algorithm 3: QUERY($\boldsymbol{x}$)

```
/* Used to query the joint probability distribution.     */
Input: $\boldsymbol{x} = \langle x_1, \ldots, x_d \rangle$ is an input vector
Output: Estimated Probability $\tilde{P}[\boldsymbol{x}]$
foreach $i$ from $1$ to $n$ do
    $p_i \leftarrow \frac{\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})}{\mathcal{A}_i(\boldsymbol{x}_i^{par})}$
Return $\prod_{i=1}^n p_i$
```

within desired bounds. In other words, BASELINE configures Algorithm **??** with $\texttt{epsfnA}(i) = \texttt{epsfnB}(i) = \frac{\epsilon}{3n}$. Our analysis makes use of the following standard fact.

Fact 1: For $0 < \epsilon < 1$ and $n \in \mathbb{Z}^+$, when $\alpha \leq \frac{\epsilon}{3n}$, $\left(\frac{1+\alpha}{1-\alpha}\right)^n \leq e^{\epsilon}$ and $\left(\frac{1-\alpha}{1+\alpha}\right)^n \geq e^{-\epsilon}$

Lemma 6: Given $0 < \epsilon, \delta < 1$ and a Bayesian Network with n variables, the BASELINE algorithm maintains the parameters of the Bayesian Network such that at any point, it is an (ϵ, δ)-approximation to the MLE. The total communication cost across m training observations is $O\left(\frac{n^2 J^{d+1}\sqrt{k}}{\epsilon} \cdot \log \frac{1}{\delta} \cdot \log m\right)$ messages, where J is the maximum domain cardinality for any variable X_i, d is the maximum number of parents for any variable and k is the number of sites.

Proof: We analyze the ratio

$$\frac{\tilde{P}(\boldsymbol{x})}{\hat{P}(\boldsymbol{x})} = \prod_{i=1}^n \frac{\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})}{\mathcal{A}_i(\boldsymbol{x}_i^{par})} \cdot \frac{\mathcal{C}_i(\boldsymbol{x}_i^{par})}{\mathcal{C}_i(x_i, \boldsymbol{x}_i^{par})}$$

We provide a probabilistic analysis based on applying Chebyshev's inequality to the variance of each counter (Lemma **??**). We first choose a particular value of the counter's parameter ϵ', which is proportional to ϵ/n. By appealing to the union bound, we have that each counter $\mathcal{A}_i()$ is in the range $(1 \pm \frac{\epsilon}{3n}) \cdot \mathcal{C}_i()$ with probability at least $1 - \delta$. The worst case is when $\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par}) = \left(1 - \frac{\epsilon}{3n}\right) \cdot \mathcal{C}_i(x_i, \boldsymbol{x}_i^{par})$ and $\mathcal{A}_i(\boldsymbol{x}_i^{par}) = (1 + \frac{\epsilon}{3n}) \cdot \mathcal{C}_i(\boldsymbol{x}_i^{par})$, i.e each counter takes on an extreme value within its confidence interval. In this case, $\frac{\tilde{P}(\boldsymbol{x})}{\hat{P}(\boldsymbol{x})}$ takes on the minimum value. Using Fact **??**, we get $\frac{\tilde{P}(\boldsymbol{x})}{\hat{P}(\boldsymbol{x})} \geq \left(\frac{1-\frac{\epsilon}{3n}}{1+\frac{\epsilon}{3n}}\right)^n \geq e^{-\epsilon}$. Symmetrically, we have $\frac{\tilde{P}(\boldsymbol{x})}{\hat{P}(\boldsymbol{x})} \leq e^{\epsilon}$ when we make pessimistic assumptions in the other direction.

Using Lemma **??**, the communication cost for each distributed counter is $O\left(\frac{n\sqrt{k}}{\epsilon} \cdot \log \frac{1}{\delta} \cdot \log m\right)$ messages. For each $i \in [1, n]$, there are at most J^{d+1} counters $\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})$ and at most J^d counters $\mathcal{A}_i(\boldsymbol{x}_i^{par})$ for all $x_i \in \mathrm{dom}(X_i)$ and $\boldsymbol{x}_i^{par} \in \mathrm{dom}(\mathsf{par}(X_i))$. So the total communication cost is $O\left(\frac{n^2 J^{d+1}\sqrt{k}}{\epsilon} \cdot \log \frac{1}{\delta} \cdot \log m\right)$ messages. ∎

D. UNIFORM*: Improved Uniform Approximate Counters*

The approach in BASELINE is overly pessimistic: it assumes that all errors may fall in precisely the worst possible direction. Since the counter algorithms are unbiased and random, we can provide a more refined statistical analysis and still obtain our desired guarantee with less communication.

Recall that the randomized counter algorithm in Lemma **??** can be shown to have the following properties:

- Each distributed counter is unbiased, $\mathbb{E}[\mathcal{A}] = \mathcal{C}$.
- The variance of counter is bounded, $\mathbf{Var}[\mathcal{A}] \leq (\epsilon' \mathcal{C})^2$, where ϵ' is the error parameter used in $\mathcal{A}$.

Hence the product of multiple distributed counters is also unbiased, and we can also bound the variance of the product.

Our UNIFORM algorithm initializes its state using Algorithm **??** with $\texttt{epsfnA}(i) = \texttt{epsfnB}(i) = \frac{\epsilon}{16\sqrt{n}}$. We prove its properties after first stating a useful fact.

Fact 2: When $0 < x < 0.3$, $e^x < 1 + 2x$ and $e^{-2x} < 1 - x$.

Lemma 7: Given input vector $\boldsymbol{x} = \langle x_1, \ldots, x_d \rangle$, let $F = \prod_{i=1}^n \mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})$ and $f = \prod_{i=1}^n \mathcal{C}_i(x_i, \boldsymbol{x}_i^{par})$. With Algorithm UNIFORM, $\mathbb{E}[F] = f$ and $\mathbf{Var}[F] \leq \frac{\epsilon^2}{128} \cdot f^2$.

Proof: From Lemma **??**, for $i \in [1, n]$ we have

$$\mathbb{E}[\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})] = \mathcal{C}_i(x_i, \boldsymbol{x}_i^{par}).$$

Since all the distributed counters $\mathcal{A}_i(\cdot, \cdot)$ are independent, we have:

$$\mathbb{E}\left[\prod_{i=1}^n \mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})\right] = \prod_{i=1}^n \mathcal{C}_i(x_i, \boldsymbol{x}_i^{par})$$

This proves $\mathbb{E}[F] = f$. We next compute $\mathbb{E}\left[\mathcal{A}_i^2(x_i, \boldsymbol{x}_i^{par})\right]$,

$$\begin{aligned}\mathbb{E}\left[\mathcal{A}_i^2(x_i, \boldsymbol{x}_i^{par})\right] &= \mathbf{Var}[\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})] + (\mathbb{E}[\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})])^2 \\ &\leq (\texttt{epsfnA}(i) \cdot \mathcal{C}_i(x_i, \boldsymbol{x}_i^{par}))^2 + \mathcal{C}_i^2(x_i, \boldsymbol{x}_i^{par}) \\ &\leq \left(1 + \frac{\epsilon^2}{256n}\right) \cdot \mathcal{C}_i^2(x_i, \boldsymbol{x}_i^{par})\end{aligned}$$

By noting that different terms $\mathcal{A}_i^2(x_i, \boldsymbol{x}_i^{par})$ are independent:

$$\mathbb{E}\left[F^2\right] = \mathbb{E}\left[\left(\prod_{i=1}^n \mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})\right)^2\right] = \prod_{i=1}^n \mathbb{E}\left[\mathcal{A}_i^2(x_i, \boldsymbol{x}_i^{par})\right]$$

$$\leq \left(1 + \frac{\epsilon^2}{256n}\right)^n \cdot \prod_{i=1}^n \mathcal{C}_i^2(x_i, \boldsymbol{x}_i^{par}) \leq e^{\epsilon^2/256} \cdot f^2$$

Using Fact **??**, $\mathbb{E}\left[F^2\right] \leq e^{\epsilon^2/256} \cdot f^2 \leq \left(1 + \frac{\epsilon^2}{128}\right) \cdot f^2$

Since $\mathbb{E}[F] = f$, we calculate $\mathbf{Var}[F]$:

$$\mathbf{Var}[F] = \mathbb{E}\left[F^2\right] - (\mathbb{E}[F])^2 \leq \left(1 + \frac{\epsilon^2}{128}\right) \cdot f^2 - f^2 = \frac{\epsilon^2}{128} \cdot f^2$$

∎

Using Chebyshev's inequality, we can bound F.

Lemma 8: For $i \in [1, n]$, maintaining distributed counters $\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})$ with approximation factor $\frac{\epsilon}{16\sqrt{n}}$, gives $e^{-\frac{\epsilon}{2}} \leq \prod_{i=1}^n \frac{\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})}{\mathcal{C}_i(x_i, \boldsymbol{x}_i^{par})} \leq e^{\frac{\epsilon}{2}}$ with probability at least 7/8.

Proof: Using the Chebyshev inequality, with $\mathbb{E}[F] = f$

$$\mathbb{P}\left[|F - f| \leq \sqrt{8\mathbf{Var}[F]}\right] \geq \tfrac{7}{8}$$

From Lemma **??**, $\mathbf{Var}[F] \leq \frac{\epsilon^2}{128} \cdot f^2$, hence

$$\mathbb{P}\left[|F - f| \leq \tfrac{\epsilon f}{4}\right] \geq \tfrac{7}{8}$$

and so (via Fact **??**), $e^{-\frac{\epsilon}{2}} \leq \left(1-\frac{\epsilon}{4}\right) \leq \frac{F}{f} \leq \left(1+\frac{\epsilon}{4}\right) \leq e^{\frac{\epsilon}{2}}$ with probability at least $7/8$. ■

For the term $\frac{\mathcal{C}_i(\boldsymbol{x}_i^{par})}{\mathcal{A}_i(\boldsymbol{x}_i^{par})}$, we maintain distributed counters $\mathcal{A}_i(\boldsymbol{x}_i^{par})$ with approximation factor $\frac{\epsilon}{16\sqrt{n}}$. One subtlety here is that different variables, say X_i and X_j, $i \neq j$ can have $\mathsf{par}\,(X_i) = \mathsf{par}\,(X_j)$, so that $\prod_{i=1}^{n} \frac{\mathcal{C}_i(\boldsymbol{x}_i^{par})}{\mathcal{A}_i(\boldsymbol{x}_i^{par})}$ can have duplicate terms, arising from different i. This leads to terms in the product that are not independent of each other. To simplify such cases, for each $i \in [1, n]$, we maintain separate distributed counters $\mathcal{A}_i(\boldsymbol{x}_i^{par})$, so that when $\mathsf{par}\,(X_i) = \mathsf{par}\,(X_j)$, the counters $\mathcal{A}_i(\boldsymbol{x}_i^{par})$ and $\mathcal{A}_j(\boldsymbol{x}_j^{par})$ are independent of each other. Then, we can show the following lemma for counters $\mathcal{A}(\boldsymbol{x}_i^{par})$, which is derived in a manner similar to Lemma **??** and **??**. The proof is omitted.

Lemma 9: For $i \in [1, n]$, when we maintain distributed counters $\mathcal{A}_i(\boldsymbol{x}_i^{par})$ with approximation factor $\frac{\epsilon}{16\sqrt{n}}$, we have $e^{-\frac{\epsilon}{2}} \leq \prod_{i=1}^{n} \frac{\mathcal{C}_i(\boldsymbol{x}_i^{par})}{\mathcal{A}_i(\boldsymbol{x}_i^{par})} \leq e^{\frac{\epsilon}{2}}$ with probability at least $7/8$.

Combining these results, we obtain the following result about UNIFORM.

Theorem 1: Given $0 < \epsilon, \delta < 1$, UNIFORM algorithm continuously maintains an (ϵ, δ)-approximation to the MLE over the course of m observations. The communication cost over all observations is $O\left(\frac{n^{3/2} J^{d+1}\sqrt{k}}{\epsilon} \cdot \log\frac{1}{\delta} \cdot \log m\right)$ messages, where J is the maximum domain cardinality for any variable X_i, d is the maximum number of parents for a variable in the Bayesian network, and k is the number of sites.

Proof: Recall that our approximation ratio is given by

$$\frac{\tilde{P}(\boldsymbol{x})}{\hat{P}(\boldsymbol{x})} = \prod_{i=1}^{n} \frac{\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})}{\mathcal{A}_i(\boldsymbol{x}_i^{par})} \cdot \frac{\mathcal{C}_i(\boldsymbol{x}_i^{par})}{\mathcal{C}_i(x_i, \boldsymbol{x}_i^{par})}$$

Combining Lemmas **??** and **??**, we have

$$e^{-\epsilon} \leq \prod_{i=1}^{n} \frac{\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})}{\mathcal{C}_i(x_i, \boldsymbol{x}_i^{par})} \cdot \frac{\mathcal{C}_i(\boldsymbol{x}_i^{par})}{\mathcal{A}_i(\boldsymbol{x}_i^{par})} \leq e^{\epsilon}$$

with probability at least $3/4$, showing that the model that is maintained is an $(\epsilon, 1/4)$ approximation to the MLE. By taking the median of $O(\log\frac{1}{\delta})$ independent instances of the UNIFORM algorithm, we improve the error probability to δ.

The communication cost for each distributed counter is $O\left(\frac{\sqrt{nk}}{\epsilon} \cdot \log\frac{1}{\delta} \cdot \log m\right)$ messages. For each $i \in [1, n]$, there are at most J^{d+1} counters $\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})$ for all $x_i \in \mathrm{dom}(X_i)$ and $\boldsymbol{x}_i^{par} \in \mathrm{dom}(\mathsf{par}\,(X_i))$, and at most J^d counters $\mathcal{A}_i(\boldsymbol{x}_i^{par})$ for all $\boldsymbol{x}_i^{par} \in \mathrm{dom}(\mathsf{par}\,(X_i))$. So the total communication cost is $O\left(\frac{n^{3/2} J^{d+1}\sqrt{k}}{\epsilon} \cdot \log\frac{1}{\delta} \cdot \log m\right)$ messages. ■

E. Non-uniform Approximate Counters

In computing the communication cost of UNIFORM, we made the simplifying assumption that the domains of different variables are of the same size J, and each variable has the same number of parents d [1]. While this streamlines the analysis, it misses a chance to more tightly bound the communication by better adapting to the cost of parameter estimation. Our third algorithm, NONUNIFORM, has a more involved analysis by making more use of the information about the Bayesian Network.

We set the approximation parameters of distributed counters $\mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})$ and $\mathcal{A}_i(\boldsymbol{x}_i^{par})$ as a function of the values J_i (the cardinality of $\mathrm{dom}(X_i)$) and K_i (the cardinality of $\mathrm{dom}\,(\mathsf{par}\,(X_i))$). To find the settings that yield the best trade-offs, we express the total communication cost as a function of different J_is and K_is. Consider first the maintenance of the CPD for variable X_i, this uses counters of the form $\mathcal{A}_i(\cdot,\cdot)$. Using an approximation error of ν_i for these counters leads to a communication cost proportional to $\frac{J_i K_i}{\nu_i}$, since the number of such counters needed at X_i is $J_i K_i$. Thus, the total cost across all variables is $\sum_{i=1}^{n} \frac{J_i K_i}{\nu_i}$. In order to ensure correctness (approximation to the MLE), we consider the variance of our estimate of the joint probability distribution. Let $F = \prod_{i=1}^{n} \mathcal{A}_i(x_i, \boldsymbol{x}_i^{par})$ and $f = \prod_{i=1}^{n} \mathcal{C}_i(x_i, \boldsymbol{x}_i^{par})$.

$$\begin{aligned} \mathbb{E}\left[F^2\right] &= \textstyle\prod_{i=1}^{n}\left(1+\nu_i^2\right) \cdot f^2 \leq \prod_{i=1}^{n} e^{\nu_i^2} \cdot f^2 \\ &= e^{\left(\sum_{i=1}^{n}\nu_i^2\right)} \cdot f^2 \leq \left(1 + 2\textstyle\sum_{i=1}^{n}\nu_i^2\right) \cdot f^2 \end{aligned} \quad (3)$$

From Lemma **??**, to bound the error of the joint distribution, we want that $\mathbb{E}\left[F^2\right] \leq \left(1+\frac{\epsilon^2}{128}\right) \cdot f^2$ which can be ensured by providing the following condition is satisfied,

$$\textstyle\sum_{i=1}^{n} \nu_i^2 \leq \epsilon^2/256 \quad (4)$$

Thus, the problem is to find values of $\nu_1, \ldots, \nu_n$ to minimize communication while satisfying this constraint. That is,

$$\text{Minimize} \quad \textstyle\sum_{i=1}^{n} \frac{J_i K_i}{\nu_i} \quad \text{subject to} \quad \sum_{i=1}^{n} \nu_i^2 = \frac{\epsilon^2}{256} \quad (5)$$

Using the Lagrange Multiplier Method, let $\mathcal{L} = \sum_{i=1}^{n} \frac{J_i K_i}{\nu_i} + \lambda\left(\nu_i^2 - \frac{\epsilon^2}{256}\right)$, we must satisfy:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \nu_1} = -\frac{J_1 K_1}{\nu_1^2} + 2\lambda\nu_1 = 0 \\ \frac{\partial \mathcal{L}}{\partial \nu_2} = -\frac{J_2 K_2}{\nu_2^2} + 2\lambda\nu_2 = 0 \\ \quad \vdots \\ \frac{\partial \mathcal{L}}{\partial \nu_n} = -\frac{J_n K_n}{\nu_n^2} + 2\lambda\nu_n = 0 \\ \sum_{i=1}^{n} \nu_i^2 = \frac{\epsilon^2}{256} \end{cases} \quad (6)$$

Solving the above equations, the optimal parameters are:

$$\nu_i = \frac{(J_i K_i)^{1/3}\epsilon}{16\alpha}, \quad \text{where} \quad \alpha = \left(\textstyle\sum_{i=1}^{n} (J_i K_i)^{2/3}\right)^{1/2} \quad (7)$$

Next we consider the distributed counters $\mathcal{A}(\cdot)$. For each $i \in [1, n]$ and each $\boldsymbol{x}_i^{par} \in \mathrm{dom}(\mathsf{par}\,(X_i))$, we maintain $\mathcal{A}_i(\boldsymbol{x}_i^{par})$ independently and ignore the shared parents as we did in the Section **??**. Let μ_i denote the approximation factor for $\mathcal{A}_i(\boldsymbol{x}_i^{par})$, the communication cost for counter $\mathcal{A}_i(\boldsymbol{x}_i^{par})$ is proportional to $\sum_{i=1}^{n} \frac{K_i}{\mu_i}$ and the restriction due to bounding the error of joint distribution is $\sum_{i=1}^{n} \mu_i^2 \leq \frac{\epsilon^2}{256}$. Similarly to

[1] Note that these assumptions were only used to determine the communication cost, and do not affect the correctness of the algorithm.

above, the solution via the Lagrange multiplier method is

$$\mu_i = \frac{K_i^{1/3}\epsilon}{16\beta}, \quad \text{where} \quad \beta = \Big(\sum_{i=1}^{n} K_i^{2/3}\Big)^{1/2} \tag{8}$$

Setting $\texttt{epsfnA}(i) = \nu_i$ as in (**??**) and $\texttt{epsfnB}(i) = \mu_i$ as in (**??**) in Algorithm **??** gives our NonUniform algorithm.

Theorem 2: Given $0 < \epsilon, \delta < 1$, NonUniform continuously maintains an (ϵ, δ)-approximation to the MLE given m training observations. The communication cost over all observations is $O\left(\Gamma \cdot \frac{\sqrt{k}}{\epsilon} \cdot \log\frac{1}{\delta} \cdot \log m\right)$ messages, where $\Gamma = \left(\sum_{i=1}^{n}(J_iK_i)^{2/3}\right)^{3/2} + \left(\sum_{i=1}^{n} K_i^{2/3}\right)^{3/2}$

The correctness of NonUniform follows from Conditions **??** and **??** which together constrain the variance of the variable F. The communication cost is obtained by substituting the values of ν_i and μ_i into expressions for the communication cost, as in the proof of Theorem **??**. We omit the detailed proof due to space constraints.

Comparison between Uniform and NonUniform. It is clear that NonUniform should be at least as good as Uniform (in the limit), since it optimizes over more information. We now show cases where they separate. First, note that Uniform and NonUniform have the same dependence on k, ϵ, δ, and m. So to compare the two algorithms, we focus on their dependence on the J_is and K_is. Consider a case when all but one of the n variables are binary valued, and variable X_1 can take one of J different values, for some $J \gg 1$. Further, suppose that (1) the network was a tree so that d, the maximum number of parents of a node is 1, and (2) X_1 was a leaf in the tree, so that $K_i = 1$ for all nodes X_i. The communication bound for Uniform by Theorem **??** is $O(n^{1.5}J^2)$, while the bound for NonUniform by Theorem **??** is $O((n + J^{2/3})^{1.5}) = O(\max\{n^{1.5}, J\})$. In this case, our analysis argues that NonUniform provides a much smaller communication cost than Uniform. However, such 'unbalanced' models may be uncommon in practice; our experimental study (Section **??**) shows that while the cost of NonUniform is often lower, the margin is not large.

V. Special Cases and Extensions

Section **??** showed that NonUniform has the tightest bounds on communication cost to maintain an approximation to the MLE. In this section, we apply NonUniform to networks with special structure, such as Tree-Structured Network and Naïve Bayes, as well as to a classification problem.

Tree Structured Network. When the Bayesian Network is structured as a tree, each node has exactly one parent, except for the single root[2]. The following result is a consequence of Theorem **??** specialized to a tree, by noting that each set $\mathsf{par}(X_i)$ is of size 1, we let $J_{\mathsf{par}(i)}$ denote K_i, the cardinality of $\mathsf{par}(X_i)$.

Lemma 10: Given $0 < \epsilon, \delta < 1$ and a tree-structured network with n variables, Algorithm NonUniform can continuously maintain an (ϵ, δ)-approximation to the MLE incurring communication cost $O(\Gamma \cdot \frac{\sqrt{k}}{\epsilon} \cdot \log\frac{1}{\delta} \cdot \log m)$ messages. where $\Gamma = (\sum_{i=1}^{n}(J_iJ_{\mathsf{par}(i)})^{2/3})^{3/2} + (\sum_{i=1}^{n} J_{\mathsf{par}(i)}^{2/3})^{3/2}$. For the case when $J_i = J$ for all i, this reduces to $\Gamma = O(n^{1.5}J^2)$.

Naïve Bayes: The `Naïve Bayes` model is perhaps the most commonly used graphical model, especially in tasks such as classification. The graphical model of `Naïve Bayes` is a two-layer tree where we assume the root is node 1.

Specializing the NonUniform algorithm for the case of `Naïve Bayes`, we use results (**??**) and (**??**). For each node X_i with $i \in [2, n]$, $K_i = J_1$. Hence, we have the approximation factors $\texttt{epsfnA}(i) = \nu_i$ and $\texttt{epsfnB}(i) = \mu_i$ as follows.

$$\nu_i = \tfrac{\epsilon}{16} J_i^{1/3} \Big/ \big(\textstyle\sum_{i=2}^{n} J_i^{2/3}\big)^{1/2}, \qquad \mu_i = \tfrac{\epsilon}{16\sqrt{n}} \tag{9}$$

Due to space constraints, we omit further details of this case, which can be found in the long version of the paper.

Lemma 11: Given $0 < \epsilon, \delta < 1$ and a `Naïve Bayes` model with n variables, there is an algorithm that continuously maintains an (ϵ, δ)-approximation to the MLE, incurring a communication cost $O\left(\frac{\sqrt{k}}{\epsilon} \cdot J_1 \cdot \left(\sum_{i=2}^{n} J_i^{2/3}\right)^{3/2} \cdot \log\frac{1}{\delta} \cdot \log m\right)$ messages over m distributed observations. In the case when all J_i are equal to J, this expression is $O\left(\frac{n^{3/2}\sqrt{k}}{\epsilon} \cdot J^2 \cdot \log\frac{1}{\delta} \cdot \log m\right)$.

Classification: Thus far, our goal has been to estimate probabilities of joint distributions of random variables. We now present an application of these techniques to the task of classification. In classification, we are given some evidence e, the objective is to find an assignment to a subset of random variables Y, given e. The usual way to do this is to find the assignment that maximizes the probability, given e. That is, $\text{Class}(\boldsymbol{Y} \mid e) = \arg\max_y \mathbb{P}[y \mid e]$. We are interested in an approximate version of the above formulation, given by:

Definition 4: Given a Bayesian Network $\mathcal{G}$, let Y denote the set of variables whose values need to be assigned, and ϵ denote an error parameter. For any evidence e, we say that $\boldsymbol{b}$ solves the approximate Bayesian Classification with ϵ error if

$$\hat{P}[Y = \boldsymbol{b} \mid e] \geq (1 - \epsilon) \cdot \max_y \hat{P}[Y = y \mid e].$$

In other words, we want to find the assignment to the set of variables Y with conditional probability close to the maximum, if not equal to the maximum.

Lemma 12: Given evidence e and set of variables Y, if $e^{-\epsilon/4} \leq \frac{\hat{P}[\mathcal{X}]}{\hat{P}[\mathcal{X}]} \leq e^{\epsilon/4}$, then we can find assignment $\boldsymbol{b}$ that solves the approximate Bayesian Classification problem with ϵ error.

We defer the proof to the full version of the paper [**?**].

Theorem 3: There is an algorithm for Bayesian Classification (Definition **??**), with communication $O\left(\Gamma \cdot \frac{\sqrt{k}}{\epsilon} \cdot \log\frac{1}{\delta} \cdot \log m\right)$ messages over m distributed observations, where $\Gamma = \left(\sum_{i=1}^{n}(J_iK_i)^{2/3}\right)^{3/2} + \left(\sum_{i=1}^{n} K_i^{2/3}\right)^{3/2}$.

[2] We assume that the graph is connected, but this can be easily generalized for the case of a forest.

TABLE I
BAYESIAN NETWORKS USED IN THE EXPERIMENTS.

Dataset	Number of Nodes	Number of Edges	Number of Parameters
ALARM [?]	37	46	509
HEPAR II [?]	70	123	1453
LINK [?]	724	1125	14211
MUNIN [?]	1041	1397	80592

Proof: We use NONUNIFORM to maintain distributed counters with error factor $\frac{\epsilon}{4}$. From Theorem ??, we have $e^{-\epsilon/4} \leq \frac{\hat{P}[\mathcal{X}]}{\tilde{P}[\mathcal{X}]} \leq e^{\epsilon/4}$ where $\mathcal{X}$ denote all the variables. Then from Lemma ??, we achieve our goal of approximate Bayesian Classification with ϵ error. ■

VI. EXPERIMENTAL EVALUATION

A. Setup and Implementation Details

We evaluate our algorithms via a simulated stream monitoring system, and a live implementation on a cluster. The distributed learning algorithm was implemented on a cluster on Amazon Web Services, where each machine in the cluster is an EC2 t2.micro instance. All the machines are located in the region: `us-east-2a`. Events (training data) arrive at sites, where each event is sent to a site chosen uniformly at random. Queries are posed at the coordinator.

Data: We use real-world Bayesian Networks from the repository at [?]. All have been used in prior studies [?], [?], [?], [?]. In our experiments, we assume the network topology prescribed, but learn model parameters from the training data. The size of the network size ranges from small ($20-60$ nodes) to large (> 1000 nodes). Table ?? provides an overview of the networks that we use. Here, the number of edges corresponds to the total number of conditional dependency relationship.

Training Data: For each network, we generate training data based on the ground truth for the parameters. To do this, we first generate a topological ordering of all vertices in the Bayesian Network (which is guaranteed to be acyclic), and then assign values to nodes (random variables) in this order, based on the known conditional probability distributions.

Testing Data: Our testing data consists of a number of queries, each one for the probability of a specific event. We measure the accuracy according to the ability of the trained network to accurately estimate the probabilities of different events. To do this, we generate 1000 events on the joint probability space represented by the Bayesian network, and estimate the probability of each event using the parameters that have been learnt by the distributed algorithm. Each event is chosen so that its ground truth probability is at least 0.01 – this is to rule out events that are highly unlikely, for which not enough data may be available to estimate the probabilities accurately.

Algorithms: We implemented four algorithms: EXACTMLE, BASELINE, UNIFORM, and NONUNIFORM. EXACTMLE is the strawman algorithm that uses exact counters so that each site informs the coordinator whenever it receives a new observation. This algorithm sends a message for each counter, so that the length of each message exchanged is approximately the same. The other three algorithms, BASELINE, UNIFORM, and NONUNIFORM, are as described in Sections ??, ??, and ?? respectively. For each of these algorithms, a message contains an update to the value of a single counter. For our implementation on the cluster, we optimized the message transmission by merging multiple updates into a single message as follows. Upon receiving an event, we merge the resulting updates for all counters into a single message to be sent out to the coordinator. For algorithms using the randomized counters, if there were no updates to any counter, then there is no message sent. This optimization was applied to all algorithms, and they benefit the *less efficient* algorithms, such as EXACTMLE and BASELINE the most.

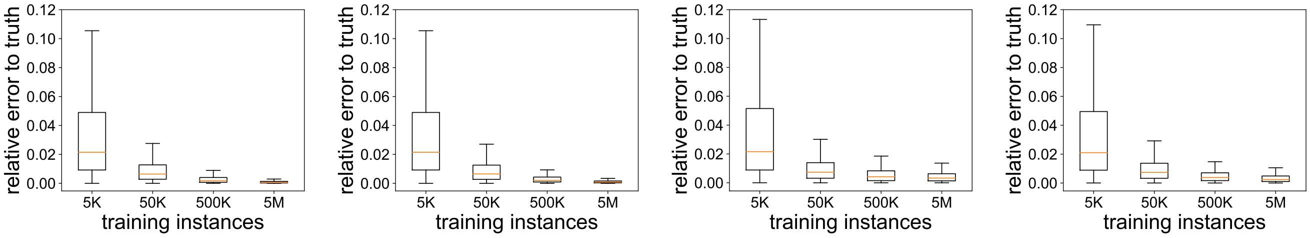
Metrics: We compute the probability for each testing event using the approximate model maintained by the distributed algorithm. We compare this with the ground truth probability for the testing event, derived from the ground truth model. For BASELINE, UNIFORM, and NONUNIFORM, we compare their results with those obtained by EXACTMLE, and report the median value from five independent runs. Unless otherwise specified, we set $\epsilon = 0.1$ and the number of sites to $k = 30$.

B. Results and Discussion

The error relative to the ground truth is the average error of the probability estimate returned by the model learnt by the algorithm, relative to the ground truth probability. Figures ?? and ?? respectively show this error as a function of the number of training instances, for the HEPAR II and LINK datasets respectively. As expected, for each algorithm, the median error decreases with an increase in the number of training instances, as can be seen by the middle quantile in the boxplot. The interquartile ranges also shrink with more training instances, showing that the variance of the error is also decreasing.

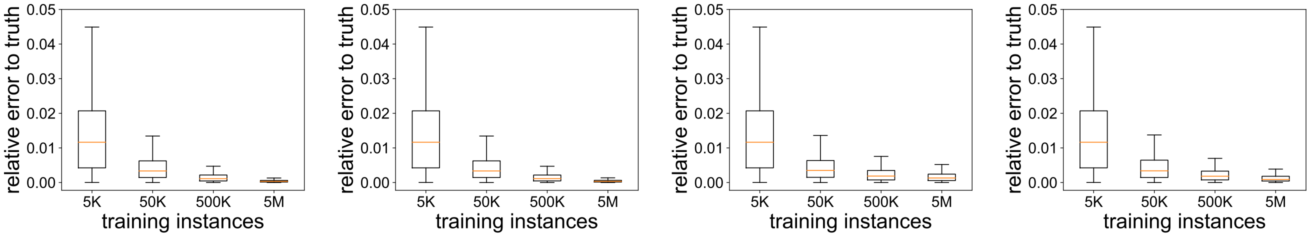
Figure ?? shows the mean relative error to the ground truth for each algorithm. We observe all the algorithms have similar performance when the number of training instances is small, say $5K$ and $50K$. When the number of training instances is large, EXACTMLE has the best accuracy, which is to be expected, since it computes the model parameters based on exact counters. BASELINE has the next best accuracy, closely followed by UNIFORM and NONUNIFORM, that show similar accuracy. Finally, all these algorithms achieve good accuracy results. For instance, after $5M$ examples, the error in estimated event probabilities is always less than one percent, for every algorithm.

The error relative to the MLE is the error of the probability estimate returned by the model learnt by the algorithm, relative to the model learnt using exact counters. The distribution of this error is shown to compare the UNIFORM and NONUNIFORM algorithms that use approximate counters, in Figures ??. We observe that the difference between these algorithms does not appear particularly large in these plots. The mean error for different algorithms is compared in Figure ??. We can consider the measured error as having two sources: (1) Statistical error, which is inherent due to training examples seen so far – this is captured by the error of the model learnt by



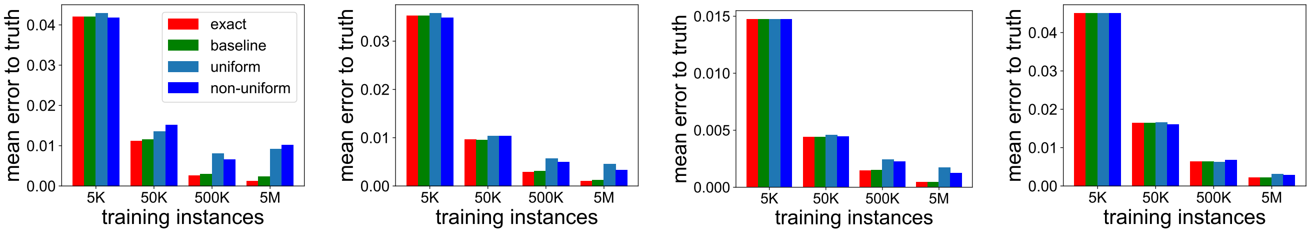
(a) Exact (b) Baseline (c) Uniform (d) Non-uniform

Fig. 1. Testing error (relative to the ground truth) vs. number of training instances. The dataset is HEPAR II.



(a) Exact (b) Baseline (c) Uniform (d) Non-uniform

Fig. 2. Testing error (relative to the ground truth) vs. number of training points. The dataset is LINK.



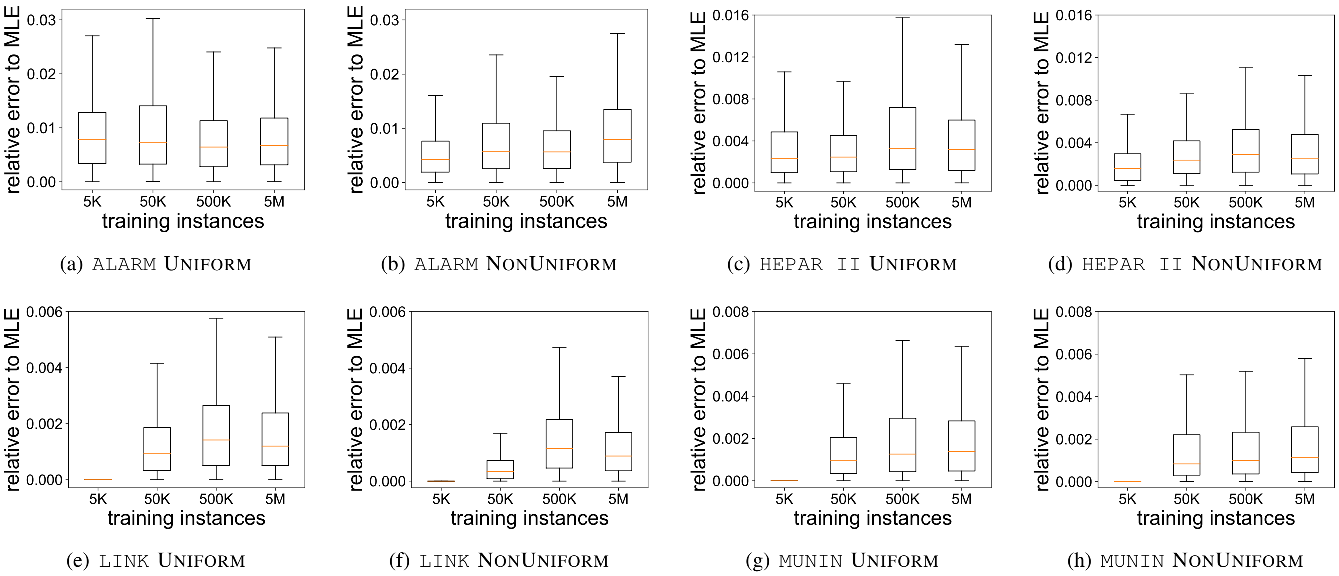
(a) ALARM (b) HEPAR II (c) LINK (d) MUNIN

Fig. 3. Mean testing error (relative to the ground truth) vs. number of training points.

the exact counter, that is MLE relative to the ground truth, and (2) Approximation error, which is the difference between the model that we are tracking and the model learnt by using exact counters – this error arises due to our desire for efficiency of communication (i.e., trying to send fewer messages for counter maintenance). Our algorithms aim to control the approximation error, and this error is captured by the error relative to exact counter. We note from the plots that the error relative to exact counter remains approximately the same with increasing number of training points, for all three algorithms, BASELINE, UNIFORM, and NONUNIFORM. This is consistent with theoretical predictions since our algorithms only guarantee that these errors are less than a threshold (relative error ϵ), which does not decrease with increasing number of points. The error of NONUNIFORM is usually better than that of UNIFORM, by around 10% on average, varying based on the network used. We conclude that for the "typical" networks evaluated here, we do not observe dramatic differences between UNIFORM and NONUNIFORM. However, since NONUNIFORM is no more difficult to implement that UNIFORM then it is reasonable to prefer it.

Communication Cost vs. Stream Size for different algorithms is shown in Figure **??**. Note that the y-axis is in logarithmic scale. From this graph, we can observe that NONUNIFORM has the smallest communication cost in general, followed by UNIFORM. These two have a significantly smaller cost than BASELINE and EXACTMLE. The gap between EXACTMLE and NONUNIFORM increases as more training data arrives. For 5M training points, NONUNIFORM sends approximately 100 times fewer messages than EXACTMLE, while having almost the same accuracy when compared with the ground truth. This also shows that there is a concrete and tangible benefit using the improved analysis in UNIFORM and NONUNIFORM, in reducing the communication cost.

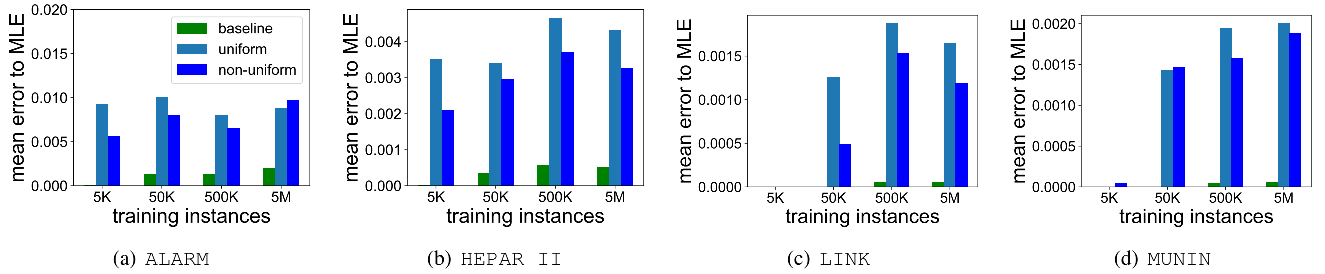
Performance on a Cluster: We measured the performance of the algorithm on a cluster of computers. The number of training instances is set to $500K$ and the number of sites (not including the coordinator) is varied from 2 to 10. We measure the total runtime as the (wallclock) time from the first to the last message received by the coordinator. Figure **??**



(a) ALARM UNIFORM (b) ALARM NONUNIFORM (c) HEPAR II UNIFORM (d) HEPAR II NONUNIFORM

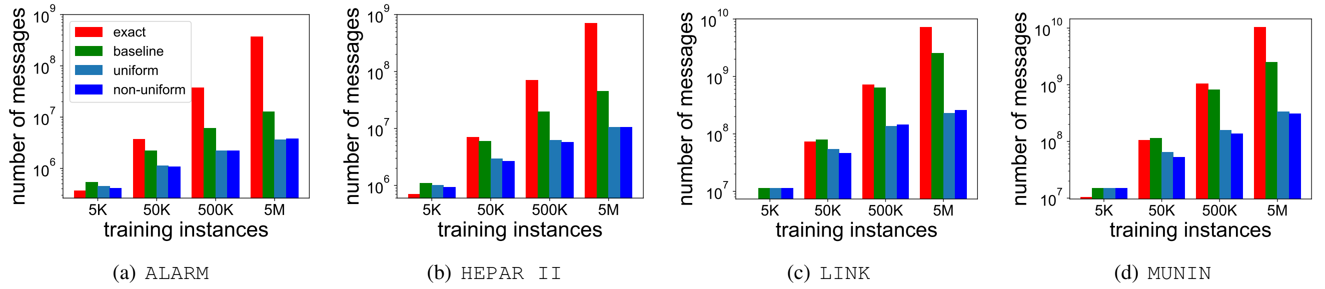
(e) LINK UNIFORM (f) LINK NONUNIFORM (g) MUNIN UNIFORM (h) MUNIN NONUNIFORM

Fig. 4. Testing error (relative to EXACTMLE) vs. number of training instances. The algorithm is UNIFORM and NONUNIFORM.



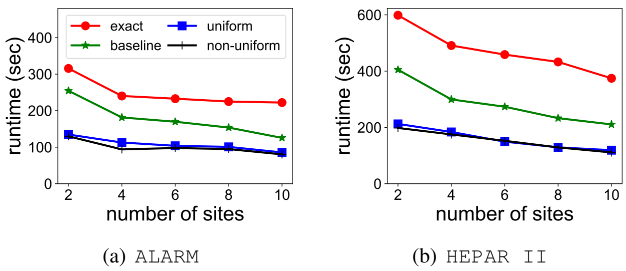
(a) ALARM (b) HEPAR II (c) LINK (d) MUNIN

Fig. 5. Mean testing error (relative to EXACTMLE) vs. number of training points.



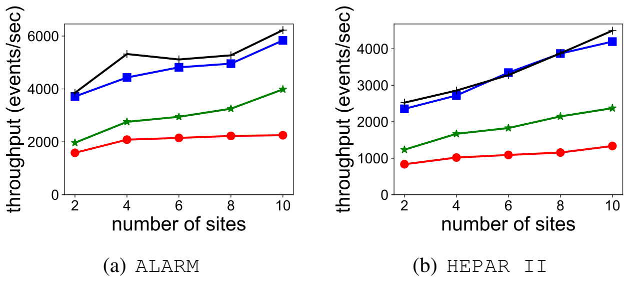
(a) ALARM (b) HEPAR II (c) LINK (d) MUNIN

Fig. 6. Communication cost vs. number of training instances.



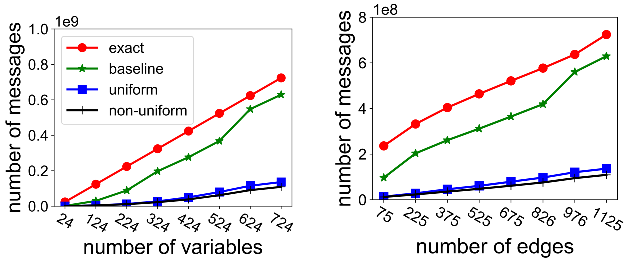
(a) ALARM (b) HEPAR II

Fig. 7. Training Runtime (on cluster) vs. the number of sites.



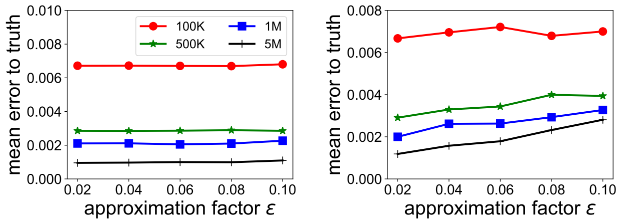
(a) ALARM (b) HEPAR II

Fig. 8. Throughput (on cluster) vs. the number of sites.



(a) varies on number of variables (b) varies on number of edges

Fig. 9. Sensitivity test as network scales (extending the LINK network).



(a) Baseline (b) Non-uniform

Fig. 10. HEPAR II mean error against ground truth vs. ϵ.

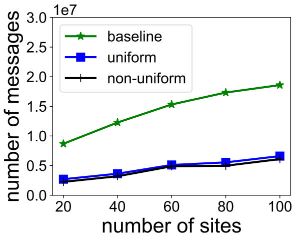


Fig. 11. Communication cost vs. number of sites, dataset is ALARM

shows the runtime of different algorithms. The Uniform and NonUniform algorithms have a significantly shorter runtime, about a half to a third that of ExactMLE, showing that they can accelerate the Bayesian Network training process. We note the following: (1) the difference between the runtime of NonUniform and ExactMLE over a network is not as large as the difference in the number of messages, since we are optimizing the number of messages by bundling many updates within a single message. (2) We can expect Uniform and NonUniform to perform even better relative to ExactMLE for streams with more training instances, since the number of messages sent increases only logarithmically for NonUniform, while it increases linearly for ExactMLE. We also plot the network throughput, defined as the average number of training points that the system can handle per second in Fig **??**. With more sites in the cluster, the network throughput increases, more so for the algorithms built on randomized counters.

Communication vs. Network Size: We test the communication cost under different sizes of networks, from small to large. To build realistic networks of different sizes, we start with the LINK network (which has 724 nodes and 1125 edges), and iteratively remove the sink nodes (outdegree of zero) one after another. This procedure generates eight different networks with $\{24, 124, 224, 324, 424, 524, 624, 724\}$ variables respectively. The communication cost for all the algorithms with $500K$ training instances is shown in Figure **??**. We observe that the number of messages of the ExactMLE algorithm increases linearly with the number of variables, as expected from our analysis. For the Uniform algorithm, even though the worst case bound on the number of messages is $O(n^{3/2})$), we see that the behavior appears closer to linear here. The number of messages sent is never more than a quarter that of the ExactMLE and Baseline algorithms. The NonUniform algorithm has slightly smaller communication cost than Uniform. Our experiments on varying the number of edges in the network shows a similar trend (Figure **??**).

Accuracy vs. Approximation Factor: Figure **??** shows the testing error as a function of the parameter ϵ, and shows that the testing error increases with an increase in ϵ. For small values of ϵ, the testing error does not change significantly as ϵ changes. This is due to the fact that ϵ only controls the "approximation error", and in cases when the statistical error is large (i.e. small numbers of training instances), the approximation error is dwarfed by the statistical error, and the overall error is not sensitive to changes in ϵ.

Communication Cost vs. Number of Sites: Figure **??** shows the communication cost as the number of sites is varied, for the ALARM dataset. The plot shows that the number of messages increases sub-linearly with number of sites k.

Communication Cost of Uniform vs. NonUniform: Our results do not yet show a large difference in the communication cost of Uniform and NonUniform. The reason is that in the networks that we used, the cardinalities of all random variables were quite similar. In other words, for different $i \in [1, n]$, the J_is in Equation **??** and **??** have similar values, and so did the K_is, which makes the theoretical bounds for Uniform and NonUniform quite similar. To observe the communication efficiency of the non-uniform approximate counter, we generated a semi-synthetic Bayesian network NEW-ALARM based on the ALARM network (https://github.com/yuz1988/new-alarm). We keep the structure of the graph, but randomly choose 6 variables in the graph and increased the domain size of these to 20 (originally each variable took between $2-4$ distinct values). For this network, the communication cost of NonUniform was about 35 percent smaller than that of Uniform.

Classification: Finally, we show results on learning a Bayesian classifier for our data sets. For each testing instance, we first generate the values for all the variables (using the underlying

TABLE II
Error Rate for Bayesian Classification, 50K training instances

Dataset	ExactMLE	Baseline	Uniform	NonUniform
ALARM	0.056	0.055	0.053	0.066
HEPAR II	0.191	0.187	0.198	0.212
LINK	0.109	0.110	0.111	0.110
MUNIN	0.091	0.091	0.093	0.091

TABLE III
Communication cost (messages) to learn a Bayesian classifier

Dataset	ExactMLE	Baseline	Uniform	NonUniform
ALARM	$3.70 \cdot 10^6$	$4.07 \cdot 10^5$	$3.24 \cdot 10^5$	$3.23 \cdot 10^5$
HEPAR II	$7.00 \cdot 10^6$	$1.08 \cdot 10^6$	$7.59 \cdot 10^5$	$7.54 \cdot 10^5$
LINK	$7.24 \cdot 10^7$	$2.98 \cdot 10^7$	$8.22 \cdot 10^6$	$8.06 \cdot 10^6$
MUNIN	$1.04 \cdot 10^8$	$3.44 \cdot 10^7$	$1.13 \cdot 10^7$	$1.13 \cdot 10^7$

model), then randomly select one variable to predict, given the values of the remaining variables. We compare the true value and predicted value of the select variable and compute the error rate. Prediction error and communication cost for 50K examples and 1000 tests are shown in Tables **??** and **??** respectively. We note that even the ExactMLE algorithm has some prediction error relative to the ground truth, due to the statistical nature of the model. The error of the other algorithms, such as Uniform and NonUniform is very close to that of ExactMLE, but their communication cost is much smaller.

VII. Conclusion

We presented new distributed streaming algorithm to estimate the parameters of a Bayesian Network in the distributed monitoring model. Compared to approaches that maintain the exact MLE, our algorithms significantly reduce communication, while offering provable guarantees on the estimates of joint probability. Our experiments show that these algorithms indeed reduce communication and provide similar prediction errors as the MLE for estimation and classification tasks.

Directions for future work include: (1) to adapt our analysis when there is a more skewed distribution across different sites, (2) to consider time-decay models which gives higher weight to more recent stream instances, and (3) to learn the underlying graph "live" in an online fashion, as more data arrives.

Acknowledgement

The work of GC is supported in part by European Research Council grant ERC-2014-CoG 647557 and a Royal Society Wolfson Research Merit Award, and of YZ and ST are supported in part by the National Science Foundation through grants 1527541 and 1725702.

References

[1] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling Up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press, 2011.

[2] X. Meng, J. Bradley, B. Yavuz, E. Sparks, and et. al., "MLlib: Machine Learning in Apache Spark," *JMLR*, vol. 17, no. 1, pp. 1235–1241, 2016.

[3] M. A. et al., "Tensorflow: A system for large-scale machine learning," in *OSDI*, 2016, pp. 265–283.

[4] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: A framework for machine learning and data mining in the cloud," *PVLDB*, vol. 5, no. 8, pp. 716–727, Apr. 2012.

[5] Y. Wang and P. M. Djuric, "Sequential bayesian learning in linear networks with random decision making," in *ICASSP*, 2014, pp. 6404–6408.

[6] J. Aguilar, J. Torres, and K. Aguilar, "Autonomie decision making based on bayesian networks and ontologies," in *IJCNN*, 2016, pp. 3825–3832.

[7] P. Xie, J. H. Li, X. Ou, P. Liu, and R. Levy, "Using bayesian networks for cyber security analysis," in *IEEE/IFIP International Conference on Dependable Systems Networks*, 2010, pp. 211–220.

[8] D. Oyen, B. Anderson, and C. M. Anderson-Cook, "Bayesian networks with prior knowledge for malware phylogenetics," in *Artificial Intelligence for Cyber Security, AAAI Workshop*, 2016, pp. 185–192.

[9] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[10] I. Sharfman, A. Schuster, and D. Keren, "A geometric approach to monitoring threshold functions over distributed data streams," *ACM Trans. Database Syst.*, vol. 32, no. 4, p. 23, 2007.

[11] A. Shukla and Y. Simmhan, "Benchmarking distributed stream processing platforms for iot applications," in *Performance Evaluation and Benchmarking. Traditional - Big Data - Interest of Things TPCTC, Revised Selected Papers*, 2016, pp. 90–106.

[12] G. Cormode, S. Muthukrishnan, and K. Yi, "Algorithms for distributed functional monitoring," in *SODA*, 2008, pp. 21:1–21:20.

[13] M. Balcan, A. Blum, S. Fine, and Y. Mansour, "Distributed learning, communication complexity and privacy," in *PMLR*, 2012, pp. 26.1–26.22.

[14] H. D. III, J. M. Phillips, A. Saha, and S. Venkatasubramanian, "Protocols for learning classifiers on distributed data," in *AISTATS*, 2012, pp. 292–290.

[15] J. Chen, H. Sun, D. P. Woodruff, and Q. Zhang, "Communication-optimal distributed clustering," in *NIPS*, 2016, pp. 3720–3728.

[16] Y. Zhang, J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Information-theoretic lower bounds for distributed statistical estimation with communication constraints," in *NIPS*, 2013, pp. 2328–2336.

[17] J. M. Phillips, E. Verbin, and Q. Zhang, "Lower bounds for number-in-hand multiparty communication complexity, made easy," in *SODA*, 2012, pp. 486–501.

[18] A. McGregor and H. T. Vu, "Evaluating bayesian networks via data streams," in *COCOON*, 2015, pp. 731–743.

[19] B. Kveton, H. Bui, M. Ghavamzadeh, G. Theocharous, S. Muthukrishnan, and S. Sun, "Graphical model sketch," in *ECML PKDD*, 2016, pp. 81–97.

[20] G. Cormode, "The continuous distributed monitoring model," *SIGMOD Record*, vol. 42, no. 1, pp. 5–14, Mar. 2013.

[21] M. Dilman and D. Raz, "Efficient reactive monitoring," in *INFOCOM*, 2001, pp. 1012–1019 vol.2.

[22] R. Keralapura, G. Cormode, and J. Ramamirtham, "Communication-efficient distributed monitoring of thresholded counts," in *SIGMOD*, 2006, pp. 289–300.

[23] Z. Huang, K. Yi, and Q. Zhang, "Randomized algorithms for tracking distributed count, frequencies, and ranks," in *PODS*, 2012, pp. 295–306.

[24] G. Cormode, S. Muthukrishnan, and W. Zhuang, "Conquering the divide: Continuous clustering of distributed data streams," in *ICDE*, 2007, pp. 1036–1045.

[25] L. Huang, X. Nguyen, M. Garofalakis, J. Hellerstein, A. D. Joseph, M. Jordan, and N. Taft, "Communication-efficient online detection of network-wide anomalies," in *INFOCOM*, 2007, pp. 134–142.

[26] C. Arackaparambil, J. Brody, and A. Chakrabarti, "Functional monitoring without monotonicity," in *ICALP*, 2009, pp. 95–106.

[27] Y.-Y. Chung, S. Tirthapura, and D. P. Woodruff, "A simple message-optimal algorithm for random sampling from a distributed stream," *IEEE TKDE*, vol. 28, no. 6, pp. 1356–1368, Jun. 2016.

[28] Y. Zhang, S. Tirthapura, and G. Cormode, "Learning graphical models from a distributed stream," *CoRR*, vol. abs/1710.02103, 2017. [Online]. Available: http://arxiv.org/abs/1710.02103

[29] M. Scutari, "Bayesian network repository," http://www.bnlearn.com/bnrepository/, [Online; accessed May-01-2017].

[30] I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper, "The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks," in *Second European Conference on Artificial Intelligence in Medicine*, 1989, pp. 247–256.

[31] A. Onisko, "Probabilistic causal models in medicine: Application to diagnosis of liver disorders," Ph.D. dissertation, Polish Academy of Science, Mar. 2003.

[32] C. S. Jensen and A. Kong, "Blocking gibbs sampling for linkage analysis in large pedigrees with many loops," *The American Journal of Human Genetics*, vol. 65, no. 3, pp. 885 – 901, 1999.

[33] S. Andreassen, F. V. Jensen, S. K. Andersen, B. Falck, U. Kjærulff, M. Woldbye, A. R. Sørensen, A. Rosenfalck, and F. Jensen, "MUNIN

— an expert EMG assistant," in *Computer-Aided Electromyography and Expert Systems*, 1989.