Scaling up Bayesian variational inference using distributed computing clusters

Andrés R. Masegosa[a,1,*], Ana M. Martinez[b,1], Helge Langseth[a], Thomas D. Nielsen[b], Antonio Salmerón[c], Darío Ramos-López[c], Anders L. Madsen[d,b]

[a] *Department of Computer and Information Science, The Norwegian University of Science and Technology, Norway*
[b] *Department of Computer Science, Aalborg University, Denmark*
[c] *Department of Mathematics, University of Almería, Spain*
[d] *HUGIN EXPERT A/S, Aalborg, Denmark*

Abstract

In this paper we present an approach for scaling up Bayesian learning using variational methods by exploiting distributed computing clusters managed by modern big data processing tools like Apache Spark or Apache Flink, which efficiently support iterative map-reduce operations. Our approach is defined as a distributed projected natural gradient ascent algorithm, has excellent convergence properties, and covers a wide range of conjugate exponential family models. We evaluate the proposed algorithm on three real-world datasets from different domains (the Pubmed abstracts dataset, a GPS trajectory dataset, and a financial dataset) and using several models (LDA, factor analysis, mixture of Gaussians and linear regression models). Our approach compares favourably to stochastic variational inference and streaming variational Bayes, two of the main current proposals for scaling up variational methods. For the scalability analysis, we evaluate our approach over a network with more than one billion nodes and approx. 75% latent variables using a computer cluster with 128 processing units (AWS). The proposed methods are released as part of an open-source toolbox for scalable probabilistic machine learning (`http://www.amidsttoolbox.com`) Masegosa et al. (2017).

Keywords: Probabilistic Graphical Models, Conjugate Exponential Family, Scalable Bayesian learning, Variational inference, Apache Flink

1. Introduction

The development of computing hardware was one of the key factors driving the spread of the Bayesian statistics (Bernardo & Smith, 2006). The advent of

*Corresponding author
Email address: `andresmasegosa@ual.es` (Andrés R. Masegosa)
[1] These two authors are considered as first authors and contributed equally to this work.

Monte Carlo methods and powerful computing units made the application of Bayesian inference feasible in a wide range of fields (Doucet et al., 2001; Chen et al., 2012). Recently, the emergence of large datasets raised a new challenge for Bayesian statistics because the previously proposed (approximate) methods were hardly able to deal with datasets involving hundred of thousands or millions of data samples (Hoffman et al., 2013). One of the main lines of research for scaling up Bayesian inference has been based on the combination of variational approaches and stochastic approximation theory (Sato, 2001; Hoffman et al., 2013; Foulds et al., 2013; Khan et al., 2015), while other approaches have also proposed the use of parallel computing architectures as a way to scale up Bayesian inference (Broderick et al., 2013).

Big data processing technologies have quickly evolved over the last years (Hashem et al., 2015). Most of these technologies provide a simple API which allows, using few lines of code, to easily manage, process, and query large datasets by seamlessly controlling a large number of distributed computing units based on commodity hardware (Carbone et al., 2015; Meng et al., 2016). They are built using sophisticated memory management schemes to cache the data in the main memory of the computing units, which greatly speeds up the iterative processing of the data. Apache Spark[2] and Apache Flink[3] are probably the most well-known of these big data processing tools.

In this paper we exploit these recent advances in big data processing on distributed computing clusters to define a novel distributed and scalable variational inference scheme. More precisely, our approach is built on a novel interpretation of an existing variational inference approach for conjugate exponential models, the so-called *variational message passing* (VMP) algorithm (Winn & Bishop, 2005), which is shown to behave as a projected natural gradient ascent algorithm (Hoffman et al., 2013; Luo & Tseng, 1993). Using this interpretation of VMP, we propose two alternative distributed schemes with different convergence properties. We empirically evaluate our approach on different latent variable models (LDA, Factor Analysis, Mixture of Gaussians and Linear Regression) over different real-world datasets (Pubmed abstracts, GPS trajectory, and real-world financial dataset). We compare our results with stochastic variational inference (SVI) (Hoffman et al., 2013), and streaming variational Bayes (SVB) (Broderick et al., 2013) and we show that our methods converge quicker and to better solutions than these alternatives. We analyze the scalability of our approach using a model with more than one billion nodes (and approximately 75% latent variables) running on a computer cluster with 128 processing units.

Contributions: We present a novel approach for scaling up variational methods, enabling the methods to exploit modern distributed computing technologies. For this purpose, we cast VMP as a projected natural gradient ascent algorithm, which gives a theoretical and practical foundation for the parallelization of variational methods over general conjugate exponential family models.

[2] `http://spark.apache.org`
[3] `https://flink.apache.org`

When compared to SVI and SVB, we find that our approach scales better and provides superior solutions, is defined for a broader class of models, and has the ability to produce important quantities like the posterior over all latent variables and the full evidence lower bound. The proposed methods are released as part of an open-source toolbox for scalable probabilistic machine learning (`http://www.amidsttoolbox.com`) (Masegosa et al., 2017, 2016a; Cabañas et al., 2016).

2. Preliminaries

2.1. Models

In this paper, we focus on conjugate exponential Bayesian network models for performing Bayesian learning on iid. data. To simplify the presentation and discussion in the paper, we shall focus on model structures of the form shown in Figure 1, although the proposed algorithm also applies to more general types of models; we shall return to this issue at the end of this section. The model in Figure 1 includes observable variables $\mathbf{X} = \mathbf{X}_{1:N}$, a vector $\boldsymbol{\theta} = \theta_{1:M}$ of global hidden variables (or parameters), a vector of local hidden variables $\boldsymbol{H} = \boldsymbol{H}_{1:N}$, and a vector of fixed parameters denoted by $\boldsymbol{\alpha}$. Note that $\mathbf{X}_i$ and $\boldsymbol{H}_i$ are themselves collections of variables. We shall use $\mathcal{D} = \mathbf{d}_{1:N}$ to denote the available data, i.e., the observed values of $\mathbf{X}$.

With the conditional distributions in the model belonging to the exponential family, we have that all distributions are of the following form

$$\ln p(X|\mathrm{pa}(X)) \quad = \quad \ln h_X + \boldsymbol{\eta}_X^p(\mathrm{pa}(X))^T \boldsymbol{s}_X(X) - A_X(\boldsymbol{\eta}_X^p(\mathrm{pa}(X))), \quad (1)$$

where $\mathrm{pa}(X)$ denotes the parents of X in the directed acyclic graph of the induced Bayesian network model. The scalar functions h_X and $A_X(\cdot)$ are the base measure and the log-normalizer, respectively; the vector functions $\boldsymbol{\eta}_X^p(\cdot)$ and $\boldsymbol{s}_X(\cdot)$ are the *natural parameters* and the *sufficient statistics* vectors, respectively. The subscript X means that the associated functional forms may be different for the different factors of the model, but we may remove the subscript when clear from the context. Similarly, we will sometimes omit the superscript p when it is clear which distribution the natural parameters $\boldsymbol{\eta}_X$ belong to.

By also requiring that the distributions are conjugate, we have that the posterior distribution for each variable in the model has the same functional form as its prior distribution. Consequently, learning (i.e. conditioning the model on observations) only changes the values of the parameters of the model rather than the functional form of the distributions. This can be achieved by expressing the functional form of $p(X|\mathrm{pa}(X))$ in terms of the sufficient statistics $\boldsymbol{s}_Z(Z)$ of any of the parents $Z \in \mathrm{pa}(X)$ of X:

$$\begin{aligned} \ln p(X|\mathrm{pa}(X)) \quad &= \quad \ln h_Z + \boldsymbol{\eta}_{XZ}(X, \mathrm{co}_Z(X))^T \boldsymbol{s}_Z(Z) \\ &- \quad A_Z(\boldsymbol{\eta}_{XZ}(X, \mathrm{co}_Z(X))), \end{aligned} \quad (2)$$

where $\mathrm{co}_Z(X)$ denotes the coparents of Z with respect to X, i.e., $\mathrm{co}_Z(X) = \mathrm{pa}(X) \setminus \{Z\}$.

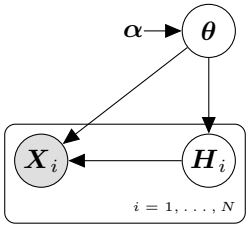


Figure 1: A plate model representation of the probabilistic models covered by d-VMP.

2.2. Variational message passing

Variational Message Passing (VMP) is an algorithm for performing variational inference over general models belonging to the conjugate exponential family (Winn & Bishop, 2005). Variational inference is a deterministic technique for finding a tractable posterior distribution, denoted by q, which approximates the Bayesian posterior, $p(\boldsymbol{\theta}, \boldsymbol{H}|\mathcal{D})$, that is often intractable to compute. More specifically, by letting $\mathcal{Q}$ be a set of possible approximations of this posterior, variational inference solves the following optimization problem for any model in the conjugate exponential family:

$$\min_{q(\boldsymbol{\theta},\boldsymbol{H})\in\mathcal{Q}} KL(q(\boldsymbol{\theta}, \boldsymbol{H})|p(\boldsymbol{\theta}, \boldsymbol{H}|\mathcal{D})), \tag{3}$$

where KL denotes the Kullback-Leibler divergence between two probability distributions.

In the *mean field variational* approach the approximation family $\mathcal{Q}$ is assumed to fully factorize:

$$q(\boldsymbol{\theta}, \boldsymbol{H}) = \prod_{k=1}^{M} q(\theta_k) \prod_{i=1}^{N} \prod_{j=1}^{J} q(H_{i,j}), \tag{4}$$

where J is the number of local hidden variables, which is assumed fixed for all $i = 1, \ldots, N$. For a conjugate exponential model, each of the components $q(\theta_k)$ will belong to the same exponential family as the prior $p(\theta_k|\text{pa}(\theta_k))$; the same holds true for the local variables $H_{i,j}$. We can therefore represent the variational posteriors by their *natural parameter* vectors, which are denoted by $\boldsymbol{\eta}_{\theta_k}$. Again, subscripts will be removed when clear from the context.

Under the same framework, we also consider generalized mean-field approximations (Winn & Bishop, 2005), where some parameters may be grouped together in a new multidimensional parameter vector with the aim of capturing richer variational posteriors at the expense of having a more complex variational family. For the sake of simplicity, we will treat this multidimensional parameters as a single parameter. The assumption is that expectations under this multidimensional distributions can also be computed in closed form.

To solve the minimization problem in Equation (3), the variational approach exploits the transformation

$$\ln P(\mathcal{D}) = \mathcal{L}(q(\boldsymbol{\theta}, \boldsymbol{H})) + KL(q(\boldsymbol{\theta}, \boldsymbol{H})|p(\boldsymbol{\theta}, \boldsymbol{H}|\mathcal{D})).$$

Note that $\mathcal{L}(\cdot)$ is a *lower bound* of $\ln P(\mathcal{D})$, since $KL(\cdot, \cdot)$ is always non-negative. As the factor $\ln P(\mathcal{D})$ is constant w.r.t. to q, minimizing the KL term is equivalent to maximizing this lower bound. Variational methods maximize the lower bound by applying a coordinate ascent approach that iteratively updates the individual variational distributions while holding the others fixed (Winn & Bishop, 2005).

Updating a variational distribution essentially involves calculating the variational expectation of the logarithm of the original conditional distributions of the model. VMP exploits the fact that this operation can be done efficiently and in closed form when the distributions involved are conjugate-exponential (Beal, 2003). Moreover, the operations can be done locally, which means that updating the variational distributions of a variable X only involves variables in the Markov blanket of X:

$$\boldsymbol{\eta}_X^q = E_q(\boldsymbol{\eta}_X^p(\mathrm{pa}(X))) + \sum_{Y \in \mathrm{ch}(X)} E_q(\boldsymbol{\eta}_{XY}^p(Y, \mathrm{co}_X(Y))), \tag{5}$$

where E_q denotes the expectation wrt. q.

The natural parameter vectors $\boldsymbol{\eta}_X^p$ and $\boldsymbol{\eta}_{XY}^p$ are multi-linear functions wrt. the natural statistics vectors of the variables on which they depend (Winn & Bishop, 2005). This means that $E_q(\boldsymbol{\eta}_X^q(pa(X))) = \boldsymbol{\eta}_X^q(E_q(pa(X)))$ and $E_q(\boldsymbol{\eta}_{XY}^p(Y, \mathrm{co}_X(Y))) = \boldsymbol{\eta}_{XY}[E_q(Y), E_q(\mathrm{co}_X(Y))]$, and due to the mean-field approximation we can calculate the required expectations independently for each of the sufficient statistics vectors involved. With a slight abuse of notation we can therefore rewrite Equation (5) as

$$\begin{aligned} \boldsymbol{\eta}_X^q &= \boldsymbol{\eta}_X^p(\{E_q(\boldsymbol{s}(Z))|Z \in \mathrm{pa}(X)\}) \\ &+ \sum_{Y \in \mathrm{ch}(X)} \boldsymbol{\eta}_{XY}^p(\{E_q(\boldsymbol{s}(Y))\} \cup \{E_q(\boldsymbol{s}(Z))|Z \in \mathrm{co}_X(Y)\}). \end{aligned} \tag{6}$$

From this expression, the coordinate ascent algorithm can be formulated as a message passing scheme. The message sent from a parent node X to a child node Y is the expectation of the natural statistics vector of X wrt. q, and the message from a child Y to a parent X is based on the messages Y has received from the co-parents of X:

$$\boldsymbol{m}_{X \to Y} = E_q(\boldsymbol{s}(X)) \tag{7}$$

$$\boldsymbol{m}_{Y \to X} = \boldsymbol{\eta}_{XY}^p(\{E_q(\boldsymbol{s}(Y))\} \cup \{E_q(\boldsymbol{s}(Z))|Z \in \mathrm{co}_X(Y)\}). \tag{8}$$

Based on the messages specified above, we see that once X has received messages from all its neighbors, its updated variational distribution is given by

$$\boldsymbol{\eta}_X^q = \boldsymbol{\eta}_X^p(\{\boldsymbol{m}_{Z \to X}|Z \in \mathrm{pa}(X)\}) + \sum_{Y \in \mathrm{ch}(X)} \boldsymbol{m}_{Y \to X}; \tag{9}$$

the implied expectations can be calculated based on the equality $E_{q_X}(\boldsymbol{s}(X)) = \nabla_{\boldsymbol{\eta}_X} A_X(\boldsymbol{\eta}_X)$ (Casella & Berger, 2001).

2.3. Big data frameworks and iterative algorithms

In this paper, we assume the data to be analyzed is placed on a distributed computing cluster managed by a big data framework. A big data framework consists of a set of tools to implement and execute complex algorithms in parallel on a cluster of networked machines. The data is stored in a distributed fashion, for which big data frameworks provide an abstraction layer which relieves programmers from having to manage the parallel execution of their algorithms on the clusters, including fault-handling of the individual machines. The MapReduce framework (Dean & Ghemawat, 2008) (as, e.g., implemented in the open-source software Hadoop) was the first and most well-known of these frameworks.

In recent years a new generation of big data frameworks has emerged aiming at addressing some of the deficiencies of these initial frameworks. One the them is the inefficient coverage given by MapReduce/Hadoop to iterative algorithms. These kinds of algorithms, quite common in the field of machine learning and data mining, have to repeatedly process (iterate) over the whole dataset, applying one or more functions until a convergence criterion is met (Chu et al., 2007). With MapReduce/Hadoop, the data has to be loaded from the hard disk to main memory at each iteration, a process which is one of the main bottlenecks of a computing cluster.

New big data frameworks, such as Spark (Zaharia et al., 2010) or Flink (Alexandrov et al., 2014), address this problem by implementing different caching strategies, which allow the data to be kept on the slaves' main memory during different iterations. This has shown to provide an increase in performance of several orders of magnitude (Meng et al., 2015).

3. Related work

Stochastic approximation theory (Robbins & Monro, 1951; Kushner & Yin, 1997) has been one of the main tools employed for scaling variational inference algorithms (Welling & Teh, 2011; Foulds et al., 2013; Khan et al., 2015) over the last few years, with the *Stochastic Variational Inference* (SVI) algorithm (Hoffman et al., 2013) being the most prominent approach. Roughly speaking, the following steps are made in a sequential manner after initializing the parameters: First, a small batch of data is subsampled from the original dataset, and the variational problem is then solved given the data-batch and using the latest version of the parameters. Thereafter, the parameters are updated using a closed form equation; exactly how to do the update has attracted considerable research effort (Duchi et al., 2011; Mandt & Blei, 2014; Khan et al., 2015). The updated parameters are used for solving the variational optimization problem as the next iteration is run with a new batch of data.

A key assumption for the SVI algorithm's efficiency is that each batch of data gives an unbiased albeit noisy estimate of the true gradient of the objective function. While this is necessarily true when the data for a batch is selected

randomly, the algorithm is a poor fit when the dataset contains rare (but relevant) subpopulations which may be completely underrepresented in the sampled mini-batches, as shown by Masegosa et al. (2016b).

Another of SVI's limitations is its restrictive assumption regarding the model family. SVI assumes that models are in the conjugate exponential family and have complete conjugate conditionals (Hoffman et al., 2013). This model family is less general than the one covered by our approach (see Section 2.1). We can accommodate models with arbitrarily complex dependency structures among the local hidden variables, the observed nodes, and the global parameters. Observable nodes can also have missing values. As an example, the model presented in (Borchani et al., 2015b) (a dynamic classification model with a global hidden variable on top of the predictive variables) fits within our model family, but not within the family supported by SVI, as the global parameters of the dynamic model do not fully factorize into univariate normal-(inverse-)gamma distributions. Another relevant limitation is that SVI cannot work with full mean field variational posteriors, as SVI assumes that the variational approximation family only factorizes at the local level,

$$q(\boldsymbol{\theta}, \boldsymbol{H}) = q(\boldsymbol{\theta}) \prod_i q(\boldsymbol{H}_i), \tag{10}$$

which prevents the use of SVI with models with a large number of global parameters (e.g. a linear regression model with many regressor variables implies that $p(\boldsymbol{\theta})$ follows a multivariate normal-Wishart distribution, which is quadratic in the number of covariates). In addition to this, SVI does not estimate all the local hidden variables of the model (e.g. for customer data, they would contain customer specific insights), nor does it generate the full evidence lower bound that can be useful, e.g., for model comparison tasks or for monitoring convergence.

Broderick et al. (2013) envisions an alternative approach, which tries to exploit multiple computing units for parallel processing, the so-called Streaming Variational Bayes (SVB) method. It is based on a straightforward adaptation of Bayes theorem,

$$p(\boldsymbol{\theta}, \boldsymbol{H}|\mathcal{D}_{[1]}, \ldots, \mathcal{D}_{[m]}) \propto \frac{p(\boldsymbol{\theta}, \boldsymbol{H}_{[1]}|\mathcal{D}_{[1]})}{p(\boldsymbol{\theta}, \boldsymbol{H}_{[1]})} \cdots \frac{p(\boldsymbol{\theta}, \boldsymbol{H}_{[m]}|\mathcal{D}_{[m]})}{p(\boldsymbol{\theta}, \boldsymbol{H}_{[m]})} p(\boldsymbol{\theta}, \boldsymbol{H}), \tag{11}$$

where $\mathcal{D}_{[k]}$ and $\boldsymbol{H}_{[k]}$ refers to the distributed data stored at the k-th slave of the cluster and their corresponding local hidden variables, respectively.

The above expression shows that the global Bayesian posterior, $p(\boldsymbol{\theta}, \boldsymbol{H}|\mathcal{D}_{[1]}, \ldots, \mathcal{D}_{[m]})$, can be built by independently combining the local posteriors, $p(\boldsymbol{\theta}, \boldsymbol{H}_{[h]}|\mathcal{D}_{[k]})$. The SVB approach is based on building local variational approximations $q_{Local,[k]}(\boldsymbol{\theta}, \boldsymbol{H}_{[k]})$ to the exact (intractable) local posteriors $p(\boldsymbol{\theta}, \boldsymbol{H}_{[k]}|\mathcal{D}_{[k]})$ and, then, combining them following Equation (11) to obtain a global approximate posterior, $q_{Global}(\boldsymbol{\theta}, \boldsymbol{H})$. When the model is conjugate exponential, the quotient operations of Equation (11) can be computed in closed form as the difference between the respective natural parameters. Hence, this combination

rule can be easily implemented as follows,

$$\boldsymbol{\eta}^{q}_{Global} = \boldsymbol{\eta}^{p}_{prior} + \sum_{k=1}^{m} (\boldsymbol{\eta}^{q}_{Local,[k]} - \boldsymbol{\eta}^{p}_{prior}). \tag{12}$$

But, unless the local variational approximations provide an exact approximation, i.e., $KL(q_{Local,[k]}, p) = 0$, the SVB approach does not yield an optimal variational solution,

$$q_{Global} \neq \min_{q(\boldsymbol{\theta},\boldsymbol{H})\in\mathcal{Q}} KL(q(\boldsymbol{\theta},\boldsymbol{H})|p(\boldsymbol{\theta},\boldsymbol{H}|\mathcal{D})). \tag{13}$$

Furthermore, SVB does not even guarantee convergence to a stationary point of the lower bound function, unlike SVI. In addition to this, Campbell & How (2014) point out that the SVB approach could eventually give rise to new subtle problems when combining the local posteriors due to parameter unidentifiability issues.

4. Distributed optimization of the lower bound

In this section, we show how we can perform distributed optimization of the lower bound function $\mathcal{L}$ using the same kind of messages as in regular VMP, but with a changed scheduling of these messages.

VMP optimizes the function $\mathcal{L}$ using the coordinate ascent method. For the kind of models we are considering (see Section 2.1), the lower bound function $\mathcal{L}$ decomposes as follows:

$$\mathcal{L}(q(\boldsymbol{\theta}), q(\boldsymbol{H})) = \mathcal{L}_{\boldsymbol{\theta}}(q(\boldsymbol{\theta})) + \sum_{n} \mathcal{L}_n(q(\boldsymbol{H}_n), q(\boldsymbol{\theta})),$$

where $\mathcal{L}_{\boldsymbol{\theta}}(q(\boldsymbol{\theta})) = E_q(\ln p(\boldsymbol{\theta})) - \sum_k E_q(\ln q(\boldsymbol{\theta}_k))$ and $\mathcal{L}_n = E_q(\ln p(\mathbf{d}_n, \boldsymbol{H}_n|\boldsymbol{\theta})) - E_q(\ln q(\boldsymbol{H}_n))$.

The optimization of the $\mathcal{L}$ function can be partly distributed by exploiting the fact that messages to the local variables in $\boldsymbol{H}_n$ do not depend on messages from other local variables $\boldsymbol{H}_{n'}$ for $n \neq n'$ and, consequently, if we keep $q(\boldsymbol{\theta})$ fixed we can maximize in parallel each of the $\mathcal{L}_n$ functions. In other words, the solution of the maximization problem

$$q^{(t+1)}(\boldsymbol{H}) = \arg\max_{q(\boldsymbol{H})} \mathcal{L}(q(\boldsymbol{H}), q^{(t)}(\boldsymbol{\theta})) \tag{14}$$

decomposes into the following independent maximization problems, which can be solved in parallel,

$$q^{(t+1)}(\boldsymbol{H}_n) = \arg\max_{q(\boldsymbol{H}_n)} \mathcal{L}_n(q(\boldsymbol{H}_n), q^{(t)}(\boldsymbol{\theta})). \tag{15}$$

If we consider Figure 2, which shows an unfolded model of the probabilistic models covered by d-VMP in a cluster with 3 slaves, the above procedure would

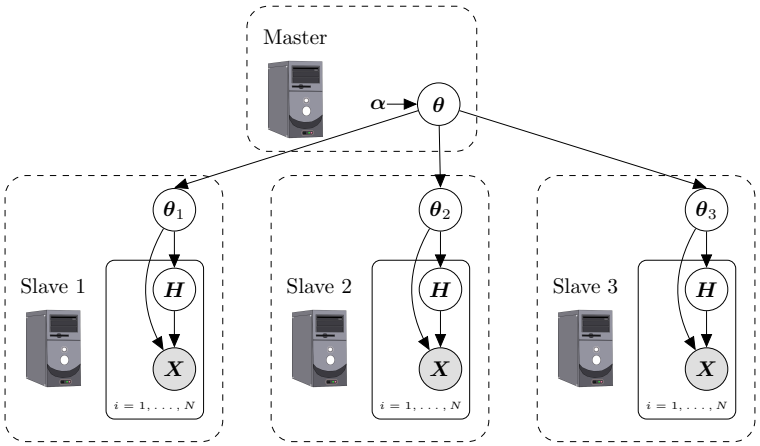


Figure 2: An unfolded model of the probabilistic models covered by d-VMP in a cluster with 3 slaves.

entail that the master node broadcasts to the slave nodes the current posterior over the global parameters $q^{(t)}(\boldsymbol{\theta})$. Each slave then solves in parallel the local optimization problem of Equation (15) using VMP by keeping fixed its local copy of the posterior over the global parameters $q^{(t)}(\boldsymbol{\theta})$ and sending messages between the local hidden variables until convergence of the local lower bound, $\mathcal{L}_n$, for data samples $\mathbf{d}_n$ locally stored at the slave.

The next step in the coordinate ascent method is

$$q^{(t+1)}(\boldsymbol{\theta}) = \arg\max_{q(\boldsymbol{\theta})} \mathcal{L}(q^{(t)}(\boldsymbol{H}), q(\boldsymbol{\theta})). \tag{16}$$

This step could, at first glance, be solved as follows: after solving Equation (15) each slave computes the local messages from $\boldsymbol{H}_n$ and $\boldsymbol{d}_n$ to $\boldsymbol{\theta}$ and sends them back to the master node. The master node then collects all the messages from the slaves and proceeds with the updating of the global parameters $\boldsymbol{\theta}$.

For the SVI's model family (see Equation (10)), it is assumed that Equation (16) can be solved in closed form because it employs a generalized mean-field approximation that does not factorize over the global parameters. This gives rise to a straightforward distributed algorithm[4]. Unfortunately, this last step is not immediately applicable for the general kinds of models we are considering. The difficulty is that the global parameters may be directly coupled through the VMP updating rules (see Equation (9)) and they can therefore not be updated independently of each other; recall that two variables are coupled if they appear in each others' Markov blankets.

[4]This approach was briefly discussed in Hoffman et al. (2013), page 1314.

Example 1. *Consider the following data generating model,* $y^{(i)} \sim \mathcal{N}(\beta_0 + \sum_{j=1}^{n} \beta_j x_j^{(i)}, \gamma)$*, where* $\beta_j \sim \mathcal{N}(0,1)$ *and* $\gamma \sim \Gamma(1,1)$*, and a full mean-field variational approximation. The variational posterior over* β_j *is computed using the messages coming from its children, i.e*

$$\eta^q_{\beta_j} = \eta^p_{prior} + \sum_i \boldsymbol{m}_{y^{(i)} \to \beta_j}.$$

The message $\boldsymbol{m}_{y^{(i)} \to \beta_j}$ *depends on the co-parents of* y *with respect to* β_j*, i.e, it depends on the moment parameters of the variational posteriors of* $q(\gamma)$ *and* $q(\beta_k)$*,* $k \neq j$ *(see Equation* (9)*). This means that all the variational posteriors for* β_i *cannot be updated independently and in parallel, because each of these posteriors would try to accommodate the observations and, in effect, could potentially over-compensate and fail to converge.*

We can overcome the above difficulty by only making joint updates of subsets of decoupled global parameters. More specifically, we propose to partition the global parameters $\boldsymbol{\theta}$ into R (non-disjoint) sets, denoted by $\mathcal{P} = \{\mathcal{I}_1, \ldots, \mathcal{I}_R\}$, so that if $\theta_i, \theta_j \in \mathcal{I}_s$ then $\theta_j \notin \text{mb}(\theta_i)$, where $\text{mb}(\cdot)$ denotes the Markov blanket of a node. Clearly, this condition does not define a unique partitioning as one may start by assigning each parameter to its own partition, i.e., $\mathcal{P} = \{\{\theta\} | \theta \in \boldsymbol{\theta}\}$, and then generate other partitionings by merging subsets whose union is also admissible. As we shall see later, for the sake of computational efficiency we generally seek to keep R as small as possible.

Example 2. *For the model given in Example 1, we have* $n+2$ *global parameters, one for each* β_j *coefficient (including* β_0*) and another one for* γ*. For this model there are* $n+2$ *partitions, because all global parameters belong to each other's Markov blanket, i.e.* $R = n+2$ *and* $|\mathcal{I}_j| = 1$*,* $\forall j \in \{1, \ldots, R\}$*.*

Equivalently, the messages to a parameter $\theta_i \in \mathcal{I}_s$ are only independent of the posterior q over the other parameters $\theta_j \in \mathcal{I}_s$, for $j \neq i$. Hence, when $\overline{\mathcal{I}_s} = \boldsymbol{\theta} \setminus \mathcal{I}_s$,

$$q^{(t+1)}_{\mathcal{I}_s} = \arg\max_{q_{\mathcal{I}_s}} \mathcal{L}(q^{(t)}_{\boldsymbol{H}}, q_{\mathcal{I}_s}, q^{(t)}_{\overline{\mathcal{I}_s}}), \tag{17}$$

i.e. we do not jointly update global parameters belonging to the same Markov blanket.

This approach leads to the distributed optimization method in Algorithm 1, which we call naive d-VMP. The shortcoming of this algorithm is that we need to iterate over all parameter partitions in $\mathcal{P}$ and, before updating the posterior over the global parameters in each partition, we need to recompute the posterior over the local hidden variables. In the next section, we show how we can skip the outer-most loop over the different partitions of the global variables in Algorithm 1.

Naive d-VMP
Initialize $q(\boldsymbol{\theta})$ and $q(\boldsymbol{H})$;
do
  for $s = 1, \ldots, R$ **do**
    for $\boldsymbol{H}_n \in \boldsymbol{H}$ **do in parallel**
      do
        for *each:* $\boldsymbol{H}_{n,j} \in H_n$ **do**
          Update $q(H_{n,j})$ using Equation (9);
        end
      until $\mathcal{L}_{\boldsymbol{H}_n}$ *converges*;
      Send messages to the Master node.
    end
    Collect messages at the Master node.
    for ***each:*** $\theta \in \mathcal{I}_s$ **do**
      Update $q(\theta)$ using Equation (9);
    end
  end
until $\mathcal{L}$ *converges*;

Algorithm 1: Naive d-VMP algorithm.

5. Distributed VMP (d-VMP)

5.1. VMP as a projected natural gradient ascent algorithm

In this section we extend the analysis carried out by Sato (2001); Hoffman et al. (2013) to general conjugate exponential models and show how the message passing scheme of VMP can be interpreted as a projected natural gradient ascent algorithm (Luo & Tseng, 1993). For any variable X in the model, the lower bound $\mathcal{L}$ with respect to $q(X)$ can be expressed as

$$\begin{aligned}\mathcal{L}(q(X)) &= E_q(\ln p(X|\mathrm{pa}(X))) - E_q(\ln q(X)) \\ &+ \sum_{Y \in \mathrm{ch}(X)} E_q(\ln p(Y|X, \mathrm{co}_X(Y)) + \text{const.}\end{aligned}$$

Using the conjugacy properties of the exponential models and the equality $E_q(\boldsymbol{s}(X)) = \nabla_{\boldsymbol{\eta}_X} A_X(\boldsymbol{\eta}_X)$, we can rewrite the above equation in terms of $\boldsymbol{\eta}_X$, the natural parameters of $q(X)$:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\eta}_X) &= E_q(\boldsymbol{\eta}_X^p(\mathrm{pa}(X)))\nabla_{\boldsymbol{\eta}_X} A_X - \boldsymbol{\eta}_X^T \nabla_{\boldsymbol{\eta}_X} A_X + A_X \\ &+ \sum_{Y \in \mathrm{ch}(X)} E_q(\boldsymbol{\eta}_{XY}^p(Y, \mathrm{co}_X(Y)))^T \nabla_{\boldsymbol{\eta}_X} A_X + \text{const.} \qquad (18)\end{aligned}$$

where A_X implicitly takes $\boldsymbol{\eta}_X$ as argument. We can now derive the gradient of $\mathcal{L}$ with respect to $\boldsymbol{\eta}_X$ based on Equation (9):

$$\nabla_{\boldsymbol{\eta}_X} \mathcal{L} = \nabla^2_{\boldsymbol{\eta}_X} A_X^{\mathrm{T}}(\boldsymbol{\eta}_X)\Big(\boldsymbol{\eta}_X^p(\{\boldsymbol{m}_{Z \to X} | Z \in \mathrm{pa}(X)\}) + \sum_{Y \in \mathrm{ch}(X)} \boldsymbol{m}_{Y \to X} - \boldsymbol{\eta}_X\Big). \qquad (19)$$

As pointed out by Sato (2001), Equation (19) can be used to compute the *natural gradient* of $\mathcal{L}$, denoted $\hat{\nabla}\mathcal{L}$, by premultiplying the gradient of $\mathcal{L}$ by the

inverse of the Fisher information matrix of $q(X)$, denoted by $G(\boldsymbol{\eta}_X)$ (which acts as a Riemannian metric over the parameter space of the statistical model),

$$\hat{\nabla}_{\boldsymbol{\eta}_X}\mathcal{L}(\boldsymbol{\eta}_X) \doteq G(\boldsymbol{\eta}_X)^{-1}\nabla_{\boldsymbol{\eta}_X}\mathcal{L}(\boldsymbol{\eta}_X).$$

For the exponential family, this Fisher information matrix corresponds to the Hessian of the log-normalizer, $G(\boldsymbol{\eta}_X) = \nabla^2_{\boldsymbol{\eta}_X} A_X(\boldsymbol{\eta}_X)$. Consequently, the natural gradient of $\mathcal{L}$ can simply be computed as

$$\hat{\nabla}_{\boldsymbol{\eta}_X}\mathcal{L} = \boldsymbol{\eta}^p_X(\{\boldsymbol{m}_{Z\to X}|Z \in \mathrm{pa}(X)\}) + \sum_{Y\in\mathrm{ch}(X)} \boldsymbol{m}_{Y\to X} - \boldsymbol{\eta}_X. \tag{20}$$

In light of the above equation, VMP can be seen as a gradient ascent method moving in orthogonal directions across the natural gradient with steps of length one,

$$\boldsymbol{\eta}^{(t+1)}_X = \boldsymbol{\eta}^{(t)}_X + \hat{\nabla}_{\boldsymbol{\eta}_X}\mathcal{L}(\boldsymbol{\eta}^{(t)}) \;=\; \boldsymbol{\eta}^p_X(\{\boldsymbol{m}^{(t)}_{Z\to X}|Z \in \mathrm{pa}(X)\}) + \sum_{Y\in\mathrm{ch}(X)} \boldsymbol{m}^{(t)}_{Y\to X}.$$

The above updating scheme corresponds to a *projected natural gradient ascent algorithm* (Luo & Tseng, 1993) by restating the above equation as follows,

$$\boldsymbol{\eta}^{(t+1)}_X \;=\; \boldsymbol{\eta}^{(t)}_X + \rho_{X,t}[\hat{\nabla}_{\boldsymbol{\eta}}\mathcal{L}(\boldsymbol{\eta}^{(t)})]^+_X \tag{21}$$

where, abusing notation, $[\cdot]^+_X$ denotes the orthogonal projection onto the coordinates of the natural parameters $\boldsymbol{\eta}_X$, and $\rho_{X,t}$ denotes the sequence of learning rates for the coordinate X.

In the case of VMP, $\rho_{X,t}$ is always equal to 1. This can be seen by combining Equations (20) and (21) while fixing $\rho_{X,t} = 1$. With this operation we get the VMP updating equation detailed in Equation (5). In this sense, $\rho_{X,t} = 1$ is an optimal value because in every step we reach the maximum of $\mathcal{L}$ over the X coordinate. Iterating over all the coordinates guarantees the convergence of the projected gradient ascent algorithm of Equation (21) to a stationary point of the function $\mathcal{L}$ (Luo & Tseng, 1993).

5.2. d-VMP as a distributed projected natural gradient ascent algorithm

In light of the above derivation, we present a new distributed optimization algorithm (d-VMP), detailed in Algorithm 2, that is expressed as a projected natural gradient ascent algorithm performing *parallel block coordinate updates* (Luo & Tseng, 1993). These parallel block coordinate updates address the aforementioned global parameters coupling problem (cf. Section 4), which we previously solved by independently updating the non-coupled global parameters. Intuitively, this new method groups in a single block all coupled global parameters to perform safe distributed updates. As a result we have a gradient ascent algorithm where the learning rates are no longer known to have optimal values at 1, but rather need to be adjusted. We have employed a simple *backtracking line search method* (Boyd & Vandenberghe, 2004) for automatically fixing the learning rates.

Distributed VMP
Initialize $q(\theta)$ and $q(\mathbf{H})$;
do
  for $\mathbf{H}_n \in \mathbf{H}$ **do in parallel**
    do
      for *each:* $\mathbf{H}_{n,j} \in H_n$ **do**
        Update $q(H_{n,j})$ using Equation (9);
      end
    until $\mathcal{L}_{\mathbf{H}_n}$ *converges*;
  end
  Collect and combine messages at the master node.
  for $k = 1 \dots S$ **do in parallel**
    Compute $\eta_{\mathcal{J}_k}^{(t+1)}$ following Equation (22).
  end
until $\mathcal{L}$ *converges*;

Algorithm 2: The d-VMP algorithm.

The d-VMP algorithm starts by defining a disjoint partitioning of the global parameters, denoted by $\mathcal{T} = \{\mathcal{J}_1, \dots, \mathcal{J}_S\}$, by using the following condition:

Definition 1. *If the variational distribution over θ_i and θ_j factorizes and θ_i belongs to the Markov blanket of θ_j in the probabilistic model (i.e. $\theta_i \in \text{mb}(\theta_j)$) then θ_i and θ_j belong to the same set $\mathcal{J}_k$* [5].

Note that this partitioning is unique (except for the trivial re-labelling of the sets $\mathcal{J}_k$). Note also that for the SVI's models (see Equation (10)) there is a single $q(\boldsymbol{\theta})$ distribution covering all the global parameters, hence for those models our approach simplifies to a partitioning with a single set covering all the global parameters. The cardinality of this set is equal to one because, as we pointed out in Section 2.2, we assume there is a single multidimensional parameter (i.e. defined as the Cartesian product of all the individual global parameters).

Example 3. *For the model examined in Examples 1 and 2, all the global parameters defining the conditional distribution between the predictive variable and the regressors belong to each other's Markov blanket (they are all parents of the same $y^{(j)}$ nodes). Thus, we have one partition including $n + 2$ parameters (the regression coefficients, the intercept, and the variance). In addition, we also have the parameters defining the univariate Gaussian distributions of the regressors, which leads to n additional partitions, each including two parameters, one for the mean and one for the variance of the Gaussian distribution. In conclusion, we have $S = n + 1$ partitions, one of size $n + 2$, while the others are of size 2.*

Other models have different partitions. For example, a factor analysis model (see Section 6.1) with k hidden factors and n observable variables would have $S = k + n$ partitions. The first k partitions contain the parameters defining the Gaussian probability density for each of the hidden factors, and therefore hold 2

[5] Note that if $\theta_i \in \text{mb}(\theta_j)$ then $\theta_j \in \text{mb}(\theta_i)$. Also note that if $\theta_i \in \text{mb}(\theta_z)$ and $\theta_j \in \text{mb}(\theta_z)$, then θ_i and θ_j belong to the same partition $\mathcal{J}_k$ even though $\theta_i \notin \text{mb}(\theta_j)$.

Model	S	$\lvert\mathcal{J}_i\rvert$	Model	S	$\lvert\mathcal{J}_i\rvert$
Linear Regression	$n+1$	$\{2, n+2\}$	Factor analysis	$n+k$	$\{2, k+2\}$
LDA	Z	$\{1\}$	Mix. of Gaussians	$k \cdot n + 1$	$\{2, 1\}$

Table 1: Number and size of the partitions for some latent variable models with n number of observable variables, k hidden variables for the Factor Analyzer, Z topics for the LDA and k mixutures components for the Mixture of Gaussian model. See Example 3 for more details.

parameters (mean and variance). The other n partitions contain the parameters defining the conditional probability of each observable variable given the hidden factors. Hence, each of these partitions contains $k+2$ parameters (the $k+1$ β-coefficients and the variance). Table 1 gives the number and the size of the partitions for some latent variable models.

Definition 1 has a direct implication on the functional form of the gradient of $\mathcal{L}$. To formally define this implication we introduce the following notation: given a partition $\mathcal{T}$, we denote by $\boldsymbol{\eta}_{\mathcal{J}_l}$ the natural parameters defining the variational posteriors over the global parameters $\boldsymbol{\theta}_k$ included in $\mathcal{J}_l$.

Lemma 1. *Given a partition $\mathcal{T}$ satisfying Definition 1, we have that for any two natural parameters $\eta_i \in \boldsymbol{\eta}_{\mathcal{J}_k}$ and $\eta_j \in \boldsymbol{\eta}_{\mathcal{J}_l}$ with $k \neq l$, the corresponding Hessian entry of $\mathcal{L}$ is always zero, i.e. $\forall \eta \quad \frac{\partial^2 \mathcal{L}}{\partial \eta_i \partial \eta_j}(\eta) = 0$.*

Proof. The result derives from the observation that when θ_i and θ_j belong to different partitions, then the variational posterior over θ_i and θ_j factorizes and θ_j does not belong to the Markov blanket of θ_i. In consequence, as shown in Equation (18), the functional terms of $\mathcal{L}(\boldsymbol{\eta}_{\theta_i})$ depending of $\boldsymbol{\eta}_{\theta_j}$ are absorbed in the constant term. □

According to Lemma 1, the functional form of the gradient of $\mathcal{L}$ with respect to the natural parameters in $\boldsymbol{\eta}_{\mathcal{J}_k}$ does not depend on the natural parameters in any other partition $\boldsymbol{\eta}_{\mathcal{J}_l}$ with $l \neq k$.

Given a parameter partition $\mathcal{T}$, the d-VMP algorithm presented as Algorithm 2 follows immediately from the equations in Section 5.1. At the Master node we now iterate over all the partitions in $\mathcal{T}$ following the updating scheme of the projected gradient ascent method (see Equation (21)) (Luo & Tseng, 1993) with block updates. More precisely, for each $\mathcal{J}_k \in \mathcal{T}$,

$$\eta_{\mathcal{J}_k}^{(t+1)} = \eta_{\mathcal{J}_k}^{(t)} + \rho_{k,t}[\hat{\nabla}_{\eta}\mathcal{L}(\eta^{(t)})]_{\mathcal{J}_k}^{+}. \tag{22}$$

However, while the block updates are performed sequentially in the original projected gradient descent algorithm, they are now performed in parallel (see the last for-loop in Algorithm 2). The following result ensures that block projected gradients can be computed in parallel, simply because they do not depend on each other:

Theorem 1. *Let $\mathcal{T}$ be a partitioning satisfying Definition 1. Under the updating scheme given by Equation* (22) *in Algorithm 2, it holds that*

$$[\nabla_\eta \mathcal{L}(\eta_{\mathcal{J}_1}^{(t)}, \ldots, \eta_{\mathcal{J}_k}^{(t)}, \ldots, \eta_{\mathcal{J}_S}^{(t)})]_{\mathcal{J}_k}^{+} = [\nabla_\eta \mathcal{L}(\eta_{\mathcal{J}_1}^{(t+1)}, \ldots, \eta_{\mathcal{J}_k}^{(t)}, \ldots, \eta_{\mathcal{J}_S}^{(t+1)})]_{\mathcal{J}_k}^{+}.$$

Proof. The equality follows directly from the conditions given in Lemma 1, which states that the functional form of the partial derivatives of the natural parameters associated to one partition does not depend of the natural parameters associated to any of the other partitions. □

There are also some special cases where the learning rates can be safely fixed before-hand, and do not need any tuning:

Theorem 2. *Let $\mathcal{T}$ be a partitioning satisfying the conditions of Definition 1, and where the cardinality of each partitions is equal to* 1*, i.e. $\forall k\ |\mathcal{J}_k| = 1$. Then, fixing $\rho_{k,t} = 1$ guarantees that Algorithm 2 converges to a stationary point of $\mathcal{L}$.*

Proof. To prove this result it suffices to prove that each iteration of the algorithm results in an increase of $\mathcal{L}$. This is achieved by iteratively solving the maximization steps detailed in Equation (14) and Equation (16). The maximization step of Equation (14) is correctly solved in the parallel for-loop by exploiting the decomposition given in Equation (15). The maximization step of Equation (16) can be alternatively expressed as

$$\arg \max_{\eta_{\mathcal{J}_1}, \ldots, \eta_{\mathcal{J}_S}} \mathcal{L}(\eta_{\mathcal{J}_1}, \ldots, \eta_{\mathcal{J}_S}, \eta_{\mathbf{H}}), \tag{23}$$

where $\eta_{\mathbf{H}}$ refers to the natural parameters of the variational posterior over the local hidden variables $\mathbf{H}$. Due to the Lemma 1, the maximization problem in Equation (23) decomposes into independent maximization problems,

$$\arg \max_{\eta_{\mathcal{J}_k}} \mathcal{L}(\eta_{\mathcal{J}_1}, \ldots, \eta_{\mathcal{J}_S}, \eta_{\mathbf{H}}). \tag{24}$$

This follows because the functional form of the partial derivative with respect to natural parameters in $\eta_{\mathcal{J}_k}$ does not depend of the other natural parameters. Because the partitions $\mathcal{J}_k$ only contain a single global parameter, the maximization problem of Equation (24) can be solved in closed form when performing the updating step given in Equation (21) with a step size of length 1. □

Notice that the above case applies to all the models covered by the SVI algorithm. Note also that in the above case d-VMP and naive-dVMP are identical, as stated formally below:

Theorem 3. *Let $\mathcal{T}$ be a partitioning satisfying the conditions given in Definition 1, and assume that the cardinality of each partitions is equal to* 1*, i.e. $\forall k\ |\mathcal{J}_k| = 1$. Then, the naive d-VMP algorithm (Algorithm 1) and the d-VMP algorithm (Algorithm 2) are identical.*

Proof. It follows from the assumptions of Theorem 3 and from Definition 1 that all the global parameters have empty Markov blankets (excluding the global parameter itself). In consequence, there is a single partitioning $\mathcal{P} = \{\mathcal{I}_1\}$ for the naive d-VMP algorithm, where $\mathcal{I}_1$ covers all the global parameters, and consequently $R = 1$. Additionally, due to Theorem 2 the updating equation of the naive d-VMP algorithm in Equation (9) is equal to the updating equation of the d-VMP algorithm in Equation (22) when $\forall k \ |\mathcal{J}_k| = 1$. □

6. Experiments

In this section, we evaluate and compare d-VMP using three datasets coming from different domains and using four different probabilistic models. The goals of the experiments are to determine how well the models represent new data as measured by the likelihood of an unseen test dataset, and to evaluate how quickly the different methods optimize the $\mathcal{L}$ function. Details on the evaluation are given in Section 6.2. Our experiments are divided into several parts. First, we compare d-VMP to naive d-VMP and SVB with models that cannot be learnt using SVI, unless intractable variational posteriors are used (see Section 3). Next, we choose models that can be handled by SVI. As commented in the previous section, for these models there are no differences between d-VMP and naive d-VMP, and so the comparison is made between d-VMP, SVI, and SVB. Finally, we investigate the scalability of the approach in a distributed setting by performing inference in a model with more than 10^9 nodes.

6.1. Experimental settings

We use three different real-life datasets in our experiments coming from three different domains:

Financial dataset: Our original financial dataset contains millions of records of financial operations from millions of clients collected over several years at the BCC financial group[6]. Due to confidentiality reasons, we only have direct access to a representative sub-sample of 55.000 clients. The dataset contains 33 features that describe the financial status of the clients, see (Borchani et al., 2015b,a) for further details. Each of the clients is additionally labeled as defaulter or non-defaulter depending of whether the client defaulted in one of her/his debts during the last two years. As expected, the dataset is highly imbalanced, that is, the percentage of defaulting clients is significantly smaller than that of non-defaulting ones. Additionally, the distributions of most of the attributes are zero-inflated, multi-modal, and typically have from 30 to 90 percent missing values.

GPS-trajectories dataset: The GPS trajectory dataset (Zheng et al., 2008, 2009, 2010) consists of a set of $17,621$ trajectories, each of which is represented by a sequence of time-stamped points with information about latitude,

[6] `https://www.bcc.es/`

longitude (we have omitted altitude), and day of the week for 182 users over a period of about 5 years. In total, it involves more than 4.5 million GPS measurements. We preprocessed this dataset with the aim of defining trajectories covering a set of GPS measurements. More precisely, we sampled 1 out of every 10 GPS measurements and defined trajectories as tuples of 25 subsamples measurements. In that way, every instance in the preprocessed dataset is a trajectory covering a group of 25 consecutive GPS measurements coming from the same user.

NSF Research Awards Abstracts: This dataset contains a text collection, in a bag-of-words representation, of abstracts describing NSF awards for basic research over the period from 1990 to 2003. There are more than 128 thousand documents/abstracts, 30799 words in the vocabulary and more than 10 million words in total (Lichman, 2013).

For evaluating the d-VMP algorithm on the three datasets above, we use four different probabilistic models. The models vary slightly from dataset to dataset, to better accommodate their different characteristics; the most expressive representative from each model class is shown in Figure 3. This graphical description is given in terms of plate notation. In the four models all attributes are continuous variables modeled with Gaussian distributions.

Linear regression (LR): We tested this model on the financial and GPS datasets. In this model, we set the "total credit amount" as the dependent variable Y for the the financial dataset. For the GPS dataset, the dependent variable is the longitude coordinate of the last GPS measurement of the trajectory defined by the instance. In both cases, there is a clear correlation between the dependent variables and the rest of the attributes. In this case, the mean field approximation consists of a set of univariate Gaussian distributions for the regressor coefficients and a Gamma distribution for modeling the variance of the dependent variable. In the financial dataset, the unrolled model contains more than 1.2 million nodes, of which more than 0.7 million are latent[7] due to the high percentage of missing data. The unrolled model for the GPS dataset contains more than 1.9 million nodes, of which 150 of them are latent.

Factor Analyzer (FA): We tested this model on the financial and GPS datasets. In this model, there is a set of five Gaussian distributed latent variables serving as parents of all the attributes. For the financial dataset, these attributes also have the Bernoulli-distributed "Defaulter" variable as a shared parent (in effect we learn separate FA models for the defaulting and non-defaulting clients), as shown in Figure 3(b). Similarly, the mean field approximation consists of a set of univariate Gaussian distributions for the regressor coefficients between the observable and the hidden factors as well as Gamma distributions modeling the variance of each observable variable. In the financial dataset, the unrolled model contains more than 1.3 million nodes, and more than 0.8 million of them are latent. In the GPS dataset, the unrolled model contains more than 2.1

[7]Latent nodes include global parameters, local hidden variables, and variables with missing observations.

million nodes; more than 190 thousand are latent.

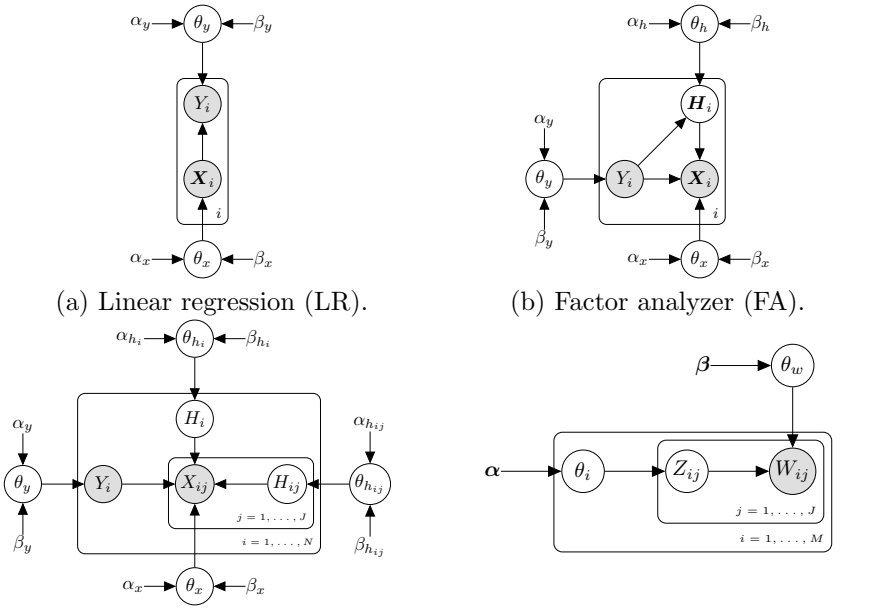
Mixture of Gaussian (MoG): We tested this model on the financial and GPS datasets. Here, all observed variables share a hidden multinomial parent with five states used to model a Gaussian mixture at the "global level". In the financial dataset, the observed variables also have the "Defaulter" variable as a shared parent. Additionally, each attribute has a separate hidden binary variable as an extra parent, which is used to represent a local mixture distribution for that attribute (accounting for frequent zero-inflated distributions). For the GPS-trajectory dataset, the MoG does not have the individual hidden variables for the attributes and the "day of the week" is used as shared parent. In this case, the mean field approximation consists of a set of Dirichlet distributions for modeling the parameters of the multinomial hidden variables and a set of joint Normal-Gamma distributions to model different mixture components. Instead of these joint Normal-Gamma distributions we use independent Normal and Gamma distributions as in the FA model. We did that in order to be able to compare with SVI, which restricts the family of mean-field approximations (see Section 3). In the financial dataset, the unrolled model contains more than 2.4 million nodes, and more than 1.8 million of them are latent. The unrolled model for GPS dataset contains more than 1.9 million nodes and more than 38 thousand of them are latent.

Latent Dirichlet Allocation (Blei et al., 2003) (LDA): We tested this model on the NSF Research Awards Abstract dataset. Fig. 3 (d) shows the LDA-like graphical model used, where J is the number of words, M the number of documents, W represents the set of words, Z the set of topics, θ_i the topic Dirichlet distribution for each document, θ_w a Dirichlet distribution over the set of words, and α the prior distribution over[8] θ_z. A set of ten topics were used in this experiments with standard hyperparameters (i.e. $\boldsymbol{\alpha} = (0.1 \dots 0.1)$ and $\boldsymbol{\beta} = (0.01 \dots 0.01)$). We use the standard mean field approximation based on Dirichlet distributions modeling the words per topic distributions. The unrolled model contains more than 20 million nodes, and more than 10 million of them are latent.

6.2. Evaluation settings

In a first step, we split the above datasets into a training and test dataset containing two thirds and one third of the data, respectively. We then compare how well the $\mathcal{L}$ function, i.e., the variational marginal log-likelihood of the training-data, is optimized and how well each model explains the unseen test data. For the latter, we measure the marginal log-likelihood of the data. For the former comparison, we measure at different times steps the value of the optimized $\mathcal{L}$ function stopping when a fixed time budget is consumed by each method. In the case of SVI, we explicitly compute the current value of the

[8] We implemented a slightly simplified (and more efficient) version, which assumes that all the occurrences of a given word inside a single document belongs to the same topic.



(a) Linear regression (LR). (b) Factor analyzer (FA).

(c) Mixture of Gaussians (MoG). (d) Latent Dirichlet Allocation (LDA)

Figure 3: Graphical models used in the experiments.

$\mathcal{L}$ function and its time-stamp after every ten mini-batch update[9]. For both versions of d-VMP, we measure the $\mathcal{L}$ function and its time-stamp every global iteration of the outer loop (see Algorithms 1 and 2). By plotting these values we can compare how quickly the different methods optimize the $\mathcal{L}$ function. SVB does not allow this evaluation because it only produces one parameter estimate. We detail later how we use SVB in our evaluation.

Another important point when evaluating variational methods is the initialization of the variational parameters. Usually, a random initialization is prescribed, but it is known that heuristics can be utilized (Hoffman et al., 2013). In our case, we use the SVB output as an initial point for the d-VMP and SVI algorithms[10]. The aim of this setting is to simplify the experimental evaluation by including a simple way to compare with SVB. By starting with the SVB solution, we can directly compare with it. Random initializations were also evaluated but produced worse overall results. We fixed a mini-batch size of 1000 data instances for all the methods. In the case of LDA models, we took a subset of documents containing no more than 1000 words.

With respect to the d-VMP learning rates, we use a single learning rate

[9]The SVI algorithm does not provide this value. We have to use the d-VMP distributed approach to compute it. The associated computation time is excluded from the analysis.

[10]The time for learning the SVB as initial point is also included in the times reported below.

for all partitions and employ a simple *backtracking line search method* (Boyd & Vandenberghe, 2004) starting with a learning rate of 1 (according to recent theoretical analyses of natural gradient methods (Martens, 2014) this learning rate is typically close to optimal) and a discounting factor of 0.5. This setting guarantees convergence to a stationary point of the lower bound function. It also shows a stable convergence behavior in our experimental evaluation. This contrasts with SVI, which requires hand-tuning of the learning rates.

The experiments were run on a Linux computer with 32 computing units and 64GB of RAM in order to speed up the results. For the LDA experiments we resorted to using a 16 node cluster through the Amazon Web Services (more information about the AWS setting is given in Section 6.5).

6.3. Naive d-VMP, d-VMP, and SVB

As commented above, in this section we evaluate the performance difference between naive d-VMP, d-VMP, and SVB. We use the FA and LR models, which cannot be easily handled by SVI. Therefore only the financial and the GPS-trajectory datasets are considered here.

Figures 4 and 5 show the results of these comparisons in terms of $\mathcal{L}$ (left y-axis). We also plot the ratio between the results at different iterations and the $\mathcal{L}$ value at the starting point (right y-axis), that corresponds to the SVB solution. Theses ratios indicate the degree of improvement of naive d-VMP and d-VMP over SVB. In all cases except for Figure 5 (b), where the differences are very small, d-VMP reports significantly higher $\mathcal{L}$ value than naive d-VMP. In particular, in Figure 5 (a) we can appreciate the magnitude of the differences. In this case, naive d-VMP becomes roughly 50 times better than SVB, whereas d-VMP is in the order of 300 times better.

In Table 2 we provide the marginal log-likelihood over the test datasets for each of the models. As can be seen, we obtain results consistent with those obtained when analyzing the log-likelihood over the training set.

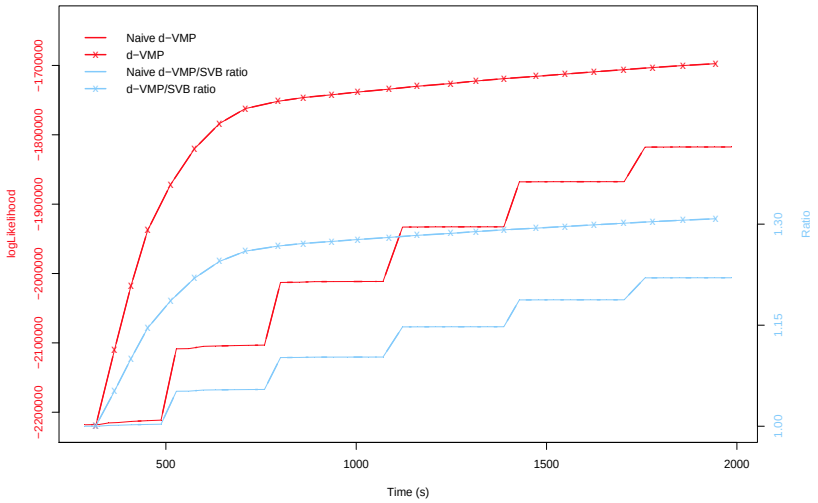
6.4. d-VMP, SVI and SVB

For the performance comparison between d-VMP and SVI we only considered the MoG and LDA models, because, as mentioned above, SVI cannot cover the FA and LR models unless computationally costly multidimensional variational approximations are used.

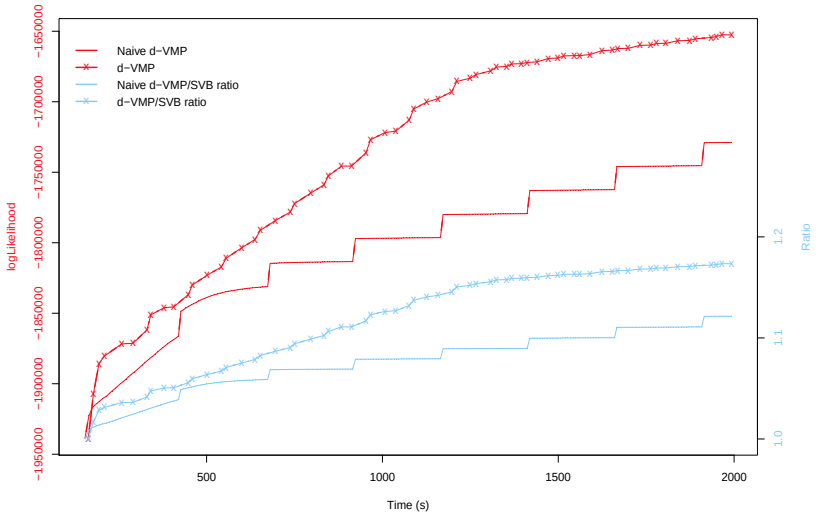
Figures 6 (a)-(c) display the comparisons between d-VMP and SVI for the MoG and LDA models. In the case of SVI, we evaluated the effect of different schedules for the learning rate by using $\rho_t = (1 + t)^{-\tau}$ and letting τ take the values 0.55, 0.75, and 0.99.

It is clear that d-VMP quickly converges to significantly higher $\mathcal{L}$ values than SVI ever obtains. Furthermore, SVI is strongly affected by the learning rate in some cases. Masegosa et al. (2016b) provides a deeper analysis of the financial dataset, in particular discussing the underperformance of SVI.

In the case of LDA, SVI has an unfortunate behavior. The first iteration is associated with the solution provided by SVB; starting from this solution

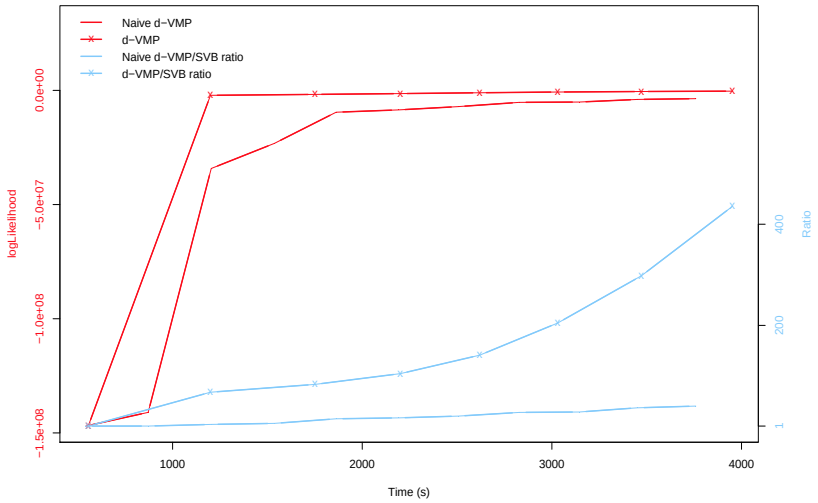


(a) Factor analyzer.

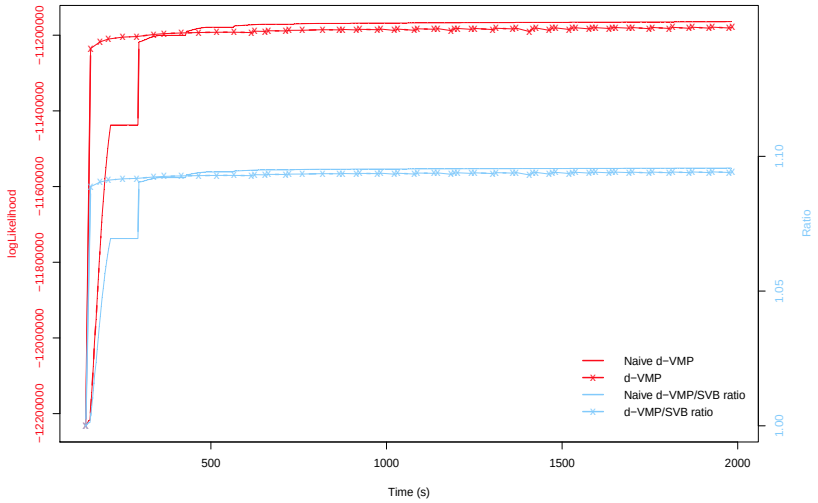


(b) Linear regression.

Figure 4: Comparison between naive d-VMP and d-VMP for the BCC data. Results are given in terms of the training marginal log-likelihood (left y-axis) and the marginal log-likelihood ratio wrt. SVB (right y-axis). SVB is considered the starting point for both algorithms.



(a) Factor analyzer



(b) Linear regression.

Figure 5: Comparison between naive d-VMP and d-VMP for GPS data. Results are given in terms of the training marginal log-likelihood (left y-axis) and the marginal log-likelihood ratio with SVB (right y-axis). SVB is considered the starting point for both algorithms. Although the lower bound for d-VMP in (a) seems to be constant due to the effect of the scale in the figure, when looking at the numbers the ELBO increases from $-1.5E06$ at 1500 sec to $-3.37E05$ at the end of the series. This increase is not immediately perceived in the ELBO series but it is clearly seen in the ratio series.

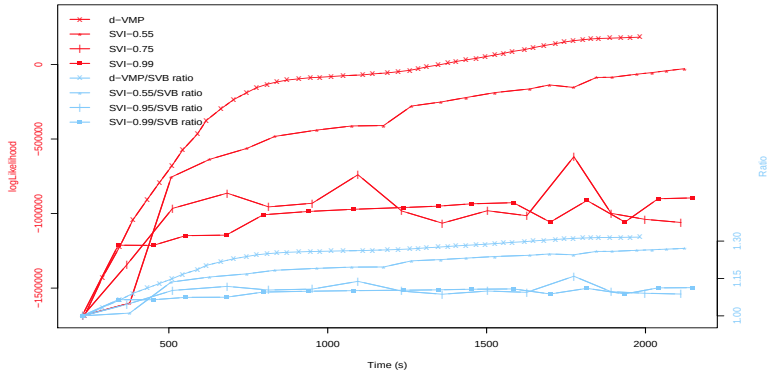
Dataset	Model	Algorithm	Test Marginal Log-likelihood
BCC	LR	naive d-VMP	-738942.3279
		d-VMP	**-714484.1839**
	FA	naive d-VMP	-753265.4537
		d-VMP	**-729342.414**
	MoG	d-VMP	**111517.9001**
		SVI-0.55	-83147.5018
		SVI-0.75	-214287.2947
		SVI-0.99	-318236.6455
GPS	LR	naive d-VMP	**-5584272.097**
		d-VMP	-36963837.4860
	FA	naive d-VMP	-22437872.2542
		d-VMP	**-14154391.0118**
	MoG	d-VMP	**244378.8553**
		SVI-0.55	53680.9002
		SVI-0.75	-53686.4863
		SVI-0.99	-220365.0224
Abstracts	LDA	d-VMP	**-2.58085216602e7**
		SVI-0.55	$-2.74148135769e7$
		SVI-0.75	$-2.72882443704e7$
		SVI-0.99	$-2.76996762555e7$

Table 2: Marginal log-likelihoods for the test sets. The maximum value within each group is highlighted in bold face. The reader should not be surprised by the positive values in the log-likelihood, as it is the result of values greater than 1 in the density functions of the Gaussian distributions.

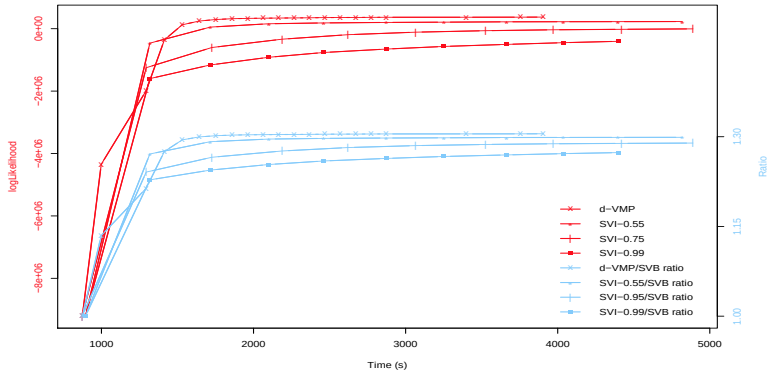
SVI updates the parameters but, ten iterations later, the solution provided by SVI has a lower quality than the initial one. Our explanation is the following: SVI moves along stochastic gradients, which are noisy estimates of the real gradients of $\mathcal{L}$. When subsampling a small batch of documents, only a tiny proportion of the words in the vocabulary are observed, and these words strongly influence the direction of the (stochastic) gradient (i.e. the pseudo-counts). At the first iterations, the weight of the local estimate of the parameter is much stronger than the weight of the previous estimate (recall that $\rho_0 = 1$, so the estimate given by SVB is *forgotten* in the SVI updating equation (Hoffman et al., 2013) [11]). When we evaluate this initial model over the whole training dataset, the documents with non-observed words in the initial mini-batches degrade the overall performance of the model.

Similarly, in Table 2 we provide the marginal log-likelhood over the test datasets for each of the models. As can be seen, the results obtained are consistent with those obtained for the training set.

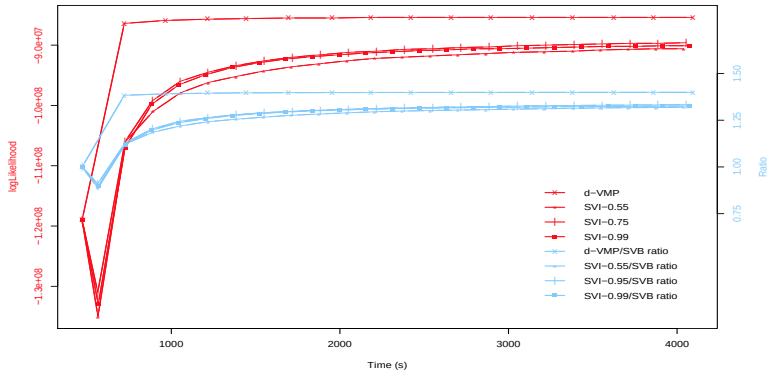
[11] We also started at $t = 1$ but the results were similar.



(a) Mixture of Gaussians: BCC data.



(b) Mixture of Gaussians: GPS data.



(c) Latent Dirichlet Allocation: NSF Research Awards Abstracts.

Figure 6: Performance comparisons between d-VMP and SVI. The results are given in terms of the training marginal log-likelihood and marginal log-likelihood ratio with SVB (starting point).

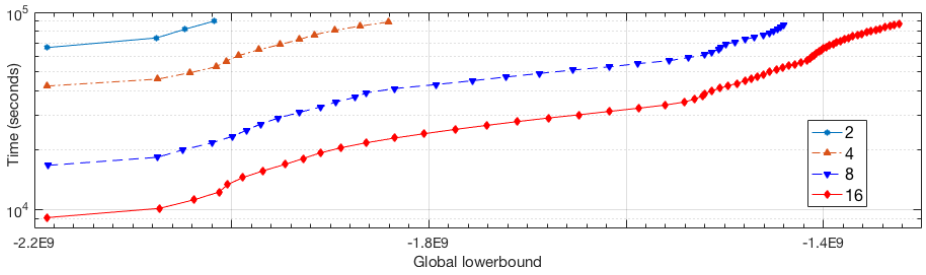


Figure 7: Time required to obtain approximations of the quality given by different global lower bounds when using 2, 4, 8, and 16 computational nodes. Each point corresponds to a single iteration of the algorithm. Note the log-scale on th y-axis.

6.5. Scalability

In the previous experiments we kept the dataset size and the computational resources fixed while making all the comparisons. However, as opposed to SVI, our approach can be scaled (by using more hardware) in case we need to process larger sample sizes or to speed up inference. In this section we evaluate the scalability of our approach.

As commented before, the original financial dataset contains millions of client operations recorded throughout several years. This dataset is confidential, so the financial institution has instead provided us with a script that generates samples over 12 variables that are distributed similarly to the corresponding variables in the real dataset. Using the script we produced a dataset with 42 million samples, which is used for the scalability test. Note that the resulting model contains more than *one billion* (10^9) nodes once it is "unrolled". More than 75% of the nodes are latent variables, leading to a posterior containing hundreds of millions of terms.

We used AWS to get access to a distributed computing environment of sufficient capacity to handle the model. The AWS clusters are equipped with Hadoop distributions, on which we can conveniently run the AMIDST toolbox (Masegosa et al., 2017). The toolbox was run on top of Apache Flink (`https://flink.apache.org/`). Cluster configurations of 2, 4, 8, and 16 nodes were employed. Each node contains 8 processing units, so the level of parallelization is between 16 and 128. Figure 7 displays the time required (wall-clock time; y-axis – note the log-scale) to get models of a given quality (measured by the variational marginal log-likelihood values; x-axis), using the four different computer clusters. The scalability of the d-VMP implementation is apparent. For example, less than 3.5 hours are required with 16 nodes to get a model of the same quality as produced by 2 nodes in 25 hours. In general, doubling the computational resources gives a speed-up factor of approximately 1.7.

7. Conclusions and future work

In this paper we have proposed d-VMP, a distributed variational message passing scheme for learning conjugate exponential models. Since d-VMP is an

iterative message passing scheme, it leverages from the memory management of modern big data frameworks like Apache Flink to obtain a time-efficient and scalable implementation. Theoretical analysis of the algorithm supports the favorable learning behavior we have observed in practice.

We plan to investigate the effects of replacing the underlying model structure with a dynamic model. Furthermore, high-speed data streams pose practical problems like limitations wrt. both computation time and available memory to store the streams. We are therefore interested in examining both theoretically and in practice how (potentially) inaccurate messages sent from the workers to the master node affect the robustness of the learned results.

Acknowledgements

This work was partly carried out as part of the AMIDST project. AMIDST has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 619209. Furthermore, this research has been partly funded by the Spanish Ministry of Economy and Competitiveness, through projects TIN2013-46638-C3-1-P, TIN2016-77902-C3-3-P and by ERDF funds. This paper is an extended version of (Masegosa et al., 2016b).

References

Alexandrov, A., Bergmann, R., Ewen, S., Freytag, J.-C., Hueske, F., Heise, A., Kao, O., Leich, M., Leser, U., Markl, V., Naumann, F., Peters, M., Rheinländer, A., Sax, M. J., Schelter, S., Höger, M., Tzoumas, K., & Warneke, D. (2014). The Stratosphere platform for big data analytics. *The Int. Journal on Very Large Data Bases*, *23*, 939–964.

Beal, M. J. (2003). *Variational algorithms for approximate Bayesian inference*. Ph.D. thesis Gatsby Computational Neuroscience Unit, University College London.

Bernardo, J. M., & Smith, A. F. (2006). *Bayesian theory*. John Wiley & Sons Canada, Limited.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of machine Learning research*, *3*, 993–1022.

Borchani, H., Martínez, A. M., Masegosa, A., Langseth, H., Nielsen, T. D., Salmerón, A., Fernández, A., Madsen, A. L., & Sáez, R. (2015a). Dynamic Bayesian modeling for risk prediction in credit operations. In *Proceedings of the 13th Scandinavian Conference on Artificial Intelligence* (pp. 17–26). IOS Press.

Borchani, H., Martínez, A. M., Masegosa, A., Langseth, H., Nielsen, T. D., Salmerón, A., Fernández, A., Madsen, A. L., & Sáez, R. (2015b). Modeling concept drift: A probabilistic graphical model based approach. In *Proc. of The Fourteenth Int. Symposium on IDA* (pp. 72–83). Springer International Publishing.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Broderick, T., Boyd, N., Wibisono, A., Wilson, A. C., & Jordan, M. I. (2013). Streaming variational Bayes. In *Advances in NIPS 26* (pp. 1727–1735). Curran Associates, Inc.

Cabañas, R., Martínez, A. M., Masegosa, A. R., Ramos-López, D., Samerón, A., Nielsen, T. D., Langseth, H., & Madsen, A. L. (2016). Financial data analysis with PGMs using AMIDST. In *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on* (pp. 1284–1287). IEEE.

Campbell, T., & How, J. P. (2014). Approximate decentralized Bayesian inference. In *Proc. of the Thirtieth Conf. on UAI* (pp. 102–111).

Carbone, P., Ewen, S., Haridi, S., Katsifodimos, A., Markl, V., & Tzoumas, K. (2015). Apache Flink: Stream and batch processing in a single engine. *Data Engineering*, (p. 28).

Casella, G., & Berger, R. (2001). *Statistical Inference*. Duxbury Resource Center.

Chen, M.-H., Shao, Q.-M., & Ibrahim, J. G. (2012). *Monte Carlo methods in Bayesian computation*. Springer Science & Business Media.

Chu, C.-T., Kim, S. K., Lin, Y.-A., Yu, Y., Bradski, G., Ng, A. Y., & Olukotun, K. (2007). Map-Reduce for Machine Learning on Multicore. *Advances in Neural Information Processing Systems 19*, (pp. 281–288).

Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, *51*, 107.

Doucet, A., De Freitas, N., & Gordon, N. (2001). An introduction to sequential Monte Carlo methods. In *Sequential Monte Carlo methods in practice* (pp. 3–14). Springer.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*, 2121–2159.

Foulds, J., Boyles, L., DuBois, C., Smyth, P., & Welling, M. (2013). Stochastic collapsed variational Bayesian inference for latent Dirichlet allocation. In *Proc. of the Int. Conf. on Knowledge Discovery and Data Mining* (pp. 446–454). ACM.

Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Khan, S. U. (2015). The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, *47*, 98–115.

Hoffman, M. D., Blei, D. M., Wang, C., & Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, *14*, 1303–1347.

Khan, M. E., Babanezhad, R., Lin, W., Schmidt, M., & Sugiyama, M. (2015). Convergence of proximal-gradient stochastic variational inference under non-decreasing step-size sequence. *arXiv preprint arXiv:1511.00146*, .

Kushner, H. J., & Yin, G. G. (1997). *Stochastic approximation algorithms and applications*. Springer New York.

Lichman, M. (2013). UCI machine learning repository. URL: `http://archive.ics.uci.edu/ml`.

Luo, Z.-Q., & Tseng, P. (1993). Error bounds and convergence analysis of feasible descent methods: A general approach. *Annals of Operations Research*, *46*, 157–178.

Mandt, S., & Blei, D. (2014). Smoothed gradients for stochastic variational inference. In *Advances in Neural Information Processing Systems* (pp. 2438–2446). MIT Press.

Martens, J. (2014). New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, .

Masegosa, A. R., Martínez, A. M., & Borchani, H. (2016a). Probabilistic graphical models on multi-core CPUs using Java 8. *IEEE Computational Intelligence Magazine*, *11*, 41–54.

Masegosa, A. R., Martínez, A. M., Langseth, H., Nielsen, T. D., Salmerón, A., Ramos-López, D., & Madsen, A. L. (2016b). d-VMP: Distributed variational message passing. In *PGM'2016. JMLR: Workshop and Conference Proceedings* (pp. 321–332). volume 52.

Masegosa, A. R., Martínez, A. M., Ramos-López, D., Cabañas, R., Salmerón, A., Nielsen, T. D., Langseth, H., & Madsen, A. L. (2017). AMIDST: a Java toolbox for scalable probabilistic machine learning. *arXiv preprint arXiv:1704.01427*, .

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., Xin, D., Xin, R., Franklin, M. J., Zadeh, R., Zaharia, M., & Talwalkar, A. (2015). MLlib: Machine learning in Apache Spark. *arXiv preprint arXiv:1505.06807*, .

Meng, X., Bradley, J., Yuvaz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S. et al. (2016). MLlib: Machine learning in Apache Spark. *JMLR*, *17*, 1–7.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, *22*, 400–407.

Sato, M.-A. (2001). Online model selection based on the variational Bayes. *Neural Computation*, *13*, 1649–1681.

Welling, M., & Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proc. of the Int. Conf. on Machine Learning (ICML-11)* (pp. 681–688).

Winn, J. M., & Bishop, C. M. (2005). Variational message passing. *Journal of Machine Learning Research*, *6*, 661–694.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. In *Proc. of the Second USENIX Conf. on Hot Topics in Cloud Computing* (pp. 1–7).

Zheng, Y., Li, Q., Chen, Y., Xie, X., & Ma, W.-Y. (2008). Understanding mobility based on GPS data. In *Proceedings of the 10th International Conference on Ubiquitous Computing* UbiComp '08 (pp. 312–321). New York, NY, USA: ACM.

Zheng, Y., Xie, X., & Ma, W.-Y. (2010). Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, *33*, 32–39.

Zheng, Y., Zhang, L., Xie, X., & Ma, W.-Y. (2009). Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th International Conference on World Wide Web* WWW '09 (pp. 791–800). New York, NY, USA: ACM.