Probabilistic Graphical Models 1 (2000) 1-48 Submitted 4/00; Published 10/00

d-VMP: Distributed Variational Message Passing

Editor:

Abstract

Motivated by a real-world financial dataset, we propose a distributed variational message passing scheme for learning conjugate exponential models. We show that the method can be seen as a projected natural gradient ascent algorithm, and it therefore has good convergence properties. This is supported experimentally, where we show that the approach is robust wrt. common problems like imbalanced data, heavy-tailed empirical distributions, and a high degree of missing values. The scheme is based on map-reduce operations, and utilizes the memory management of modern big data frameworks like Flink to obtain a time-efficient and scalable implementation. The proposed algorithm compares favourably to stochastic variational inference both in terms of speed and quality of the learned models. 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.

1. Introduction

This paper is motivated by a real-life dataset from a Spanish financial institution. The long-term goal is to learn a generative model for the data-set, with a dual view towards monitoring the set of customers as a whole and making predictions for single customers (e.g., finding the probability that a given customer will default on his obligations within a certain time-frame). The dataset has some complicating characteristics: *i*) The number of missing observations is high, with a degree of missingness typically in the range from 30% to 90% for key variables. *ii*) The distributions for a number of key variables, e.g., *balance*, are zero-inflated and have heavy tails (see Figure 3 (left) for an example). *iii*) The dataset is imbalanced in the sense that there are few customers from important customer groups (like defaulters). *iv*) The dataset is of considerable size, hence a flexible model structure will have to be combined with efficient approximate inference and learning techniques.

Stochastic approximation theory (Robbins and Monro, 1951; Kushner and Yin, 1997) has been one of the main tools employed for scaling variational inference algorithms (Welling and 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. The SVI algorithm iteratively updates the model parameters based on subsampled data batches. However, even though the algorithm learns the parameters consistently, it neither estimates all the local hidden variables of the model (i.e., the latent variables describing a customer not in the current data-batch) nor does it generate the full evidence lower bound that can be useful, e.g., for model comparison tasks or monitoring convergence.[1]

A key assumption for the SVI algorithm's scalability 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 for the financial dataset because the noise level (in terms of variation in the gradient between batches) is too large. To see the problem, consider a model, which contains a mixture of Gaussians to capture the balance of defaulting clients (which represents a tiny but relevant group of clients). A mini-batch of a few hundred samples[2] will result in very few observations of defaulters, even fewer with an observed value of the variable in question, and only rarely will an observation come from the tail of the distribution.

An immediate solution to this problem can be to increase the batch-size, but to this end it is worth noticing that if the full dataset is stored using a distributed file system like hdfs (Dean and Ghemawat, 2008), the stochastic approaches are confronted with an additional problem: While data sub-sampling can be performed in parallel, the batches must be sent through the network to the master node, where the main computations take place. Such data traffic is the main bootle-neck of a computer cluster, and this prevents the use of big data batches. One could also envision solving this problem by better exploiting the hardware available in the cluster, and following the distributed approach of Broderick et al. (2013). However, that solution strategy gives rise to new subtle problems when combining the local posteriors (Campbell and How, 2014). As a consequence, if the original data set contains hundred of millions of samples, the subsampled batches will always represent only a tiny part of the data, and the result (as we will see in Section 4 where we report on our experiments) is a poor model fit.

Our solution is therefore rather to define a distributed and scalable version of the *variational message passing* (VMP) scheme (Winn and Bishop, 2005) for approximate Bayesian parameter learning, which, as a bi-product of the developments, leads to increased understanding of VMP as a projected natural gradient ascent algorithm. We show empirically that our approach is able to capture the peculiarities of the financial dataset and, at the same time, provide better and faster convergence results compared to the SVI algorithm. We analyze the scalability of the algorithm using a model with more than a billion nodes (and approximately 75% latent variables) running on a computer cluster with 128 processing units.

Contributions: We cast VMP as a projected natural gradient ascent algorithm, which gives a theoretical and practical foundation for parallelization of learning in models with a large set of global parameters. When compared to SVI, we find that our approach is more robust (see Section 4), 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.

1. Exactly how to do the update has attracted considerable research attention (Duchi et al., 2011; Mandt and Blei, 2014; Khan et al., 2015).
2. This is a typical batch-size in the literature; values from 10 to 1000 were for instance used by Hoffman et al. (2013).

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 in Figure 1 (left), 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 (left) 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

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

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:

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

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

This model family is more general than the one covered by SVI (Hoffman et al., 2013). Our approach 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., 2015a) (a dynamic classification model with a global hidden variable on top of the predictive variables) fits within this model family, but not within the family supported by SVI. Moreover, as will be discussed below, we also allow for factorized posterior approximations over the global parameters, which is also not supported by SVI (Hoffman et al., 2013).

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 and Bishop, 2005).

Variational inference is a deterministic technique for finding tractable posterior distributions, 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:[3]

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

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.

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

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

where $\mathcal{L}(\cdot)$ 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, minimizing the KL term is equivalent to maximizing the lower bound. Variational methods maximizes this lower bound by applying a coordinate ascent that iteratively updates the individual variational distributions while holding the others fixed (Winn and 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:

$$\boldsymbol{\eta}^q_X = E_q(\boldsymbol{\eta}^p_X(\text{pa}(X))) + \sum_{Y\in\text{ch}(X)} E_q(\boldsymbol{\eta}^p_{XY}(Y, \text{co}_X(Y))). \tag{4}$$

The natural parameter vectors $\boldsymbol{\eta}^p_X$ and $\boldsymbol{\eta}^p_{XY}$ are multi-linear functions wrt. the natural statistics vectors of the variables on which they depend (Winn and Bishop, 2005). This means that we can move the expectations inside $\boldsymbol{\eta}^p_X(\text{pa}(X))$ and $\boldsymbol{\eta}^p_{XY}(Y, \text{co}_X(Y))$, 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 (4) as

$$\boldsymbol{\eta}^q_X = \boldsymbol{\eta}^p_X(\{E_q(\boldsymbol{s}(Z))|Z \in \text{pa}(X)\}) + \sum_{Y\in\text{ch}(X)} \boldsymbol{\eta}^p_{XY}(\{E_q(\boldsymbol{s}(Y))\} \cup \{E_q(\boldsymbol{s}(Z))|Z \in \text{co}_X(Y)\}).$$

3. Although more general mean-field approximations can also be used (Winn and Bishop, 2005).

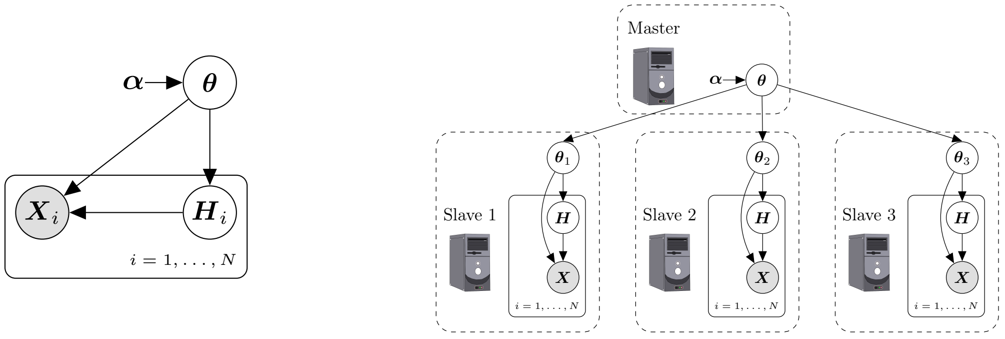


Figure 1: Probabilistic Models covered by d-VMP: (left) A plate model representation. (right) An unfolded model in a cluster with 3 slaves.

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:

$$\boldsymbol{m}_{X\rightarrow Y} = E_q(\boldsymbol{s}(X)) \qquad \boldsymbol{m}_{Y\rightarrow X} = \boldsymbol{\eta}^p_{XY}(\{E_q(\boldsymbol{s}(Y))\} \cup \{E_q(\boldsymbol{s}(Z)) | Z \in \mathrm{co}_X(Y)\}). \quad (5)$$

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

$$\boldsymbol{\eta}^q_X = \boldsymbol{\eta}^p_X(\{\boldsymbol{m}_{Z\rightarrow X} | Z \in \mathrm{pa}(X)\}) + \sum_{Y\in \mathrm{ch}(X)} \boldsymbol{m}_{Y\rightarrow X};$$

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

3. Distributed VMP (d-VMP)

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

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

where $\mathcal{L}_{\boldsymbol{\theta}}(q(\boldsymbol{\theta})) = E_q(\ln p(\boldsymbol{\theta})) - E_q(\ln q(\boldsymbol{\theta}))$ 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

$$q^{(t+1)}(\boldsymbol{H}) = \arg\max_{q(\boldsymbol{H})} \mathcal{L}(q(\boldsymbol{H}), q^{(t)}(\boldsymbol{\theta})) \tag{6}$$

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

$$q^{(t+1)}(\boldsymbol{H}_n) = \arg\max_{q(\boldsymbol{H}_n)} \mathcal{L}_n(q(\boldsymbol{H}_n), q^{(t)}(\boldsymbol{\theta})). \tag{7}$$

If we consider Figure 1 (right), the above procedure would 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 (7) 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,

$$q^{(t+1)}(\boldsymbol{\theta}) = \arg\max_{q(\boldsymbol{\theta})} \mathcal{L}(q^{(t)}(\boldsymbol{H}), q(\boldsymbol{\theta})) \tag{8}$$

could, at first glance, be solved as follows: after solving Equation (7) 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 $\boldsymbol{\theta}$. 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 (5)) 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. As an example, consider the model $Y^{(j)} = \sum_{i=1}^{n} \beta_i x_i^{(j)} + \epsilon^{(j)}$, where $\beta_i, \epsilon^{(j)} \sim \mathcal{N}(0,1)$. Assuming that we make an observation over $\boldsymbol{Y}, \boldsymbol{x}$, then by updating all the variational posteriors over the β_is independently and in parallel, each of these posteriors would try to accommodate the observations and, in effect, potentially over-compensate and fail to converge.

There are two immediate solutions to the above problem. The first is to update the $q(\theta_i)$s sequentially. That is, after each update of a global parameter θ_i, we recompute the local messages from $\boldsymbol{H}_n$ and $\boldsymbol{d}_n$ to θ' using the last updated $q(\theta_i)$. With this approach we handle parameter coupling by always updating global parameters in light of the other parameter updates that have been performed. Unfortunately, with this approach we need to iterate over all global parameters, and for each iteration we have to recompute all the messages in the local models. For models with many global parameters (e.g. having a large n in the linear regression model above), the algorithm would incur a big overhead. Another solution would be to employ a generalized mean-field approximation that does not factorize over the global parameters. This would be in the spirit of the SVI algorithm (Hoffman et al., 2013), but, unfortunately, this approach is also prohibitive for models with a large number of (coupled) global parameters.

In order to overcome these difficulties, we instead propose a gradient ascent-based algorithm, which we derive from the observation that VMP can be considered a projected natural gradient ascent algorithm.

3.2 VMP as a projected natural gradient ascent algorithm

In this section we extend the analysis carried out in (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 and Tseng, 1993). For any variable X in the model, the lower bound $\mathcal{L}$ with respect to $q(X)$ can be expressed as

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

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 the natural parameters of $q(X)$:

$$\mathcal{L}(\boldsymbol{\eta}_X) = E_q(\boldsymbol{\eta}_X^p(\text{pa}(X)))\nabla_{\boldsymbol{\eta}_X} A_X - \boldsymbol{\eta}_X^T \nabla_{\boldsymbol{\eta}_X} A_X + A_X + \sum_{Y\in \text{ch}(X)} E_q(\boldsymbol{\eta}_{XY}^p(Y, \text{co}_X(Y)))^T \nabla_{\boldsymbol{\eta}_X} A_X;$$

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 (5):

$$\nabla_{\boldsymbol{\eta}_X}\mathcal{L} = \nabla^2_{\boldsymbol{\eta}_X} A_X^{\mathrm{T}}(\boldsymbol{\eta}_X)\Big(\boldsymbol{\eta}_X^p(\{\boldsymbol{m}_{Z\to X}|Z\in \text{pa}(X)\}) + \sum_{Y\in \text{ch}(X)} \boldsymbol{m}_{Y\to X} - \boldsymbol{\eta}_X\Big). \tag{9}$$

As pointed out in (Sato, 2001), Equation (9) 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

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

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

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

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

$$\boldsymbol{\eta}_X^{(t+1)} \;=\; \boldsymbol{\eta}_X^{(t)} + \rho_{X,t}[\hat{\nabla}_{\boldsymbol{\eta}}\mathcal{L}(\boldsymbol{\eta}^{(t)})]_X^+ \tag{10}$$

where $[\cdot]_X^+$ denotes the orthogonal projection onto the coordinate X and $\rho_{X,t}$ denotes the sequence of learning rates for the coordinate X, which in case of VMP is always equal to 1. This is also the optimal value because in every step we reach the maximum of $\mathcal{L}$ over this coordinate. Iterating over all the coordinates guarantees the convergence of the projected gradient ascent algorithm of Equation (10) to a stationary point of the function $\mathcal{L}$ (Luo and Tseng, 1993).

3.3 d-VMP as a distributed projected natural gradient ascent algorithm

In light of the above derivation, we can devise a distributed optimization algorithm, detailed in Algorithm 1, by making *block coordinate updates* (Luo and Tseng, 1993). These blocks will prevent the need for iterating over all the global parameters (cf. Section 3.1).

First we define a disjoint partitioning of the global parameters, denoted by $\mathcal{R} = \{\mathcal{J}_1, \ldots, \mathcal{J}_S\}$, in such a way that we obtain the maximum number of partitions under the constraint that if two variables appear in each others' Markov blankets, then they belong to the same partition. That is, the partitions correspond to the connected components of the global parameters in the induced moral graph restricted to these parameters. Note that the partitioning is unique.

With this partitioning, the distributed VMP algorithm in Algorithm 1 follows immediately from Section 3.2. At the Master node we now iterate over all the partitions in $\mathcal{R}$ following the updating scheme of the projected gradient ascent method (Luo and Tseng, 1993) in Equation (10). If the learning rates are properly adjusted, at each iteration we always increase the lower bound because we move in the direction of the (projected) gradient. Note that we can have different learning rates for the different blocks/partitions of the parameters.

For those partitions with a unique parameter, $|\mathcal{J}_s| = 1$, we can set the learning rate $\rho_{s,t}$ to 1, since we will then move to the maximum of $\mathcal{L}$ for the projected coordinate (see Section 3.2). In this case the learning rate can be considered optimal in a *greedy search* sense. For other partitions, we also recommend using a learning rate of 1, based on recent theoretical analyses of natural gradient methods (Martens, 2014) and our experimental evaluation, which shows a stable behavior under this settings. It is important to emphasize, though, that for non-singleton partitions, the algorithm share convergence properties with general gradient-based algorithms and convergence is therefore generally dependent on the learning rate being employed.

```
Distributed VMP
Initialize $q(\theta)$ and $q(\mathbf{H})$;
do
    for in parallel: $\mathbf{H}_n \in \mathbf{H}$ do
        do
            for each: $\mathbf{H}_{n,j} \in H_n$ do
                Update $q(H_{n,j})$ using Equation (5);
            end
        until $\mathcal{L}_{\mathbf{H}_n}$ converges;
    end
    Collect and combine messages at the master node.
    for $r = 1 \ldots S$ do
        $\eta^{(t+1)}_{\mathcal{J}_r} = \eta^{(t)}_{\mathcal{J}_r} + \rho_{r,t}[\hat{\nabla}_\eta \mathcal{L}(\eta^{(t)})]^{+}_{\mathcal{J}_r}$
    end
until $\mathcal{L}$ converges;
```

Algorithm 1: The d-VMP algorithm

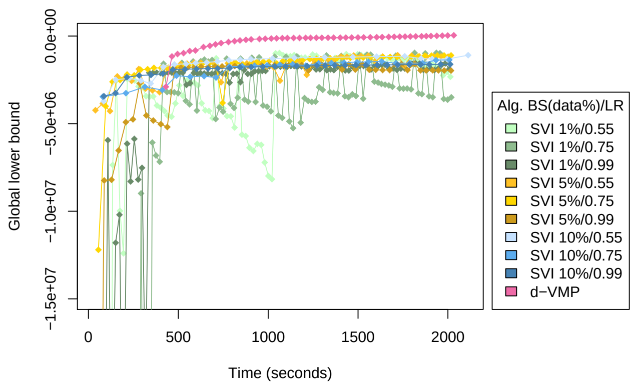
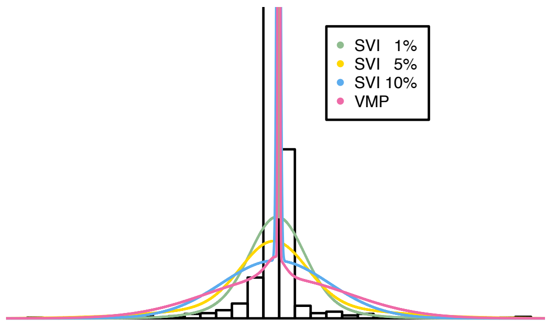


Figure 2: Lower bound values for d-VMP and SVI. Each point corresponds to a single iteration of each algorithm.



	BS (% data)	LR	Log-Likel.
SVI	1 %	0.55	-180902.87
		0.75	-298564.03
		0.99	-426979.52
	5 %	0.55	-177302.24
		0.75	-333264.16
		0.99	-628105.70
	10 %	0.55	-347035.22
		0.75	-397525.45
		0.99	-538087.13
d-VMP		1.0	67265.34

Figure 3: (Left) Resulting mixtures of learnt posteriors for one of the attributes of defaulter clients for d-VMP and SVI with different batch sizes. (Right) Results in terms of test marginal log-likelihood (after 2000 seconds of learning).

4. Experiments

Our experiments are divided into two parts. First, we compare d-VMP's capability of explaining unseen data to that of SVI. We then investigate the scalability of the approach in a distributed setting by performing inference in a model with more than 10^9 nodes.

4.1 Model fit to the data

As mentioned in the introduction, this paper is motivated by a real-life data set from a financial institution. The dataset contains millions of records of financial operations from millions of clients throughout several years. 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., 2015a,b) for further details. The dataset is highly imbalanced, for instance is the percentage of defaulting clients 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.

The model employed in the evaluation has a naive-Bayes like structure, extended with hidden variables. Attributes are continuous variables modelled with Gaussian distributions, and have the Bernoulli-distributed "Defaulter" attribute as a shared parent. Each attribute has a separate hidden binary variable as an extra parent, which is used to represent a local mixture distribution for that attribute. Finally, all attributes share an additional hidden binary parent used to model a Gaussian mixture at the global level. When the model is "unrolled" it contains more than 3.5 million nodes, out of which 75% are latent variables. Thus, the posterior to be approximated contains several million terms.

We compare our approach with Stochastic Variational Inference (SVI) (Hoffman et al., 2013). SVI uses products of joint Normal-Gamma distributions to model the q distribution over the global parameters. For the comparison we compute $\mathcal{L}$, the variational marginal log-likelihood of the training-data, for each of the sequential estimations of SVI after updating the posterior distributions over the global parameters of the model. Figure 2 shows the results of this comparison. We investigate the effect of SVI's batch size by using 1%, 5%, and

10% of the available data. We also consider the effect of different schedules for the learning rate by using $\rho_t = (1+t)^{-\tau}$ and let τ take the values 0.55, 0.75, and 0.99. SVI was randomly initialized, while d-VMP was initialized by running VMP locally over batch sizes with 1% of the data to calculate local updates to the global parameters, and use the aggregated result as the starting point.[4] The experiments were run on a Linux computer with 32 computing units and 64GB of RAM. It is clear that d-VMP quickly converges to significantly higher $\mathcal{L}$ values than SVI ever obtains. Furthermore, SVI suffers from the small batch sizes, and its behaviour is strongly affected by the learning rate.

The inferred models were also evaluated on a quarter of the data left aside as a test-set. Figure 3 (right) shows the marginal log-likelihood over that data. The results are in line with those reported in Figure 2, thus confirming that those results were not due to overfitting. The differences in the log-likehood values confirm that d-VMP learns better data representations. As an example, Figure 3 (left) shows the four-component mixture learned by SVI (with different batch sizes) and d-VMP for a particular attribute given defaulting clients. Only the central part of the distribution (which is centered around zero) is shown.[5] For all four cases there is a mixture component at zero with very small variance, which captures the zero-inflation. The remaining components collectively account for the rest of the density. Notice how d-VMP produces a density with higher variance than SVI, and thereby captures the tails of the distribution better. These results suggest that differences in test-set log-likelihood may partly be explained by the SVI-models under-estimating the variance, an effect we attribute to the subsampling approach.

4.2 Scalability

The financial data set that motivated our research contains millions of client operations recorded throughout several years, so it is essential to analyze the scalability of the proposed approach. The full data set is confidential, but the financial institution has therefore provided us with a script that generates 12 variables that are similarly distributed to the corresponding variables in the real data set. We produced a data set of 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 Amazon Web Services (AWS) to get access to a distributed computing environment of sufficient capacity. The AWS clusters are equipped with Hadoop distributions, on which we can conveniently run the Flink (`https://flink.apache.org/`) implementation of d-VMP.[6] 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 4 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 evident. 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.

4. Other batch sizes were evaluated giving rise to similar convergence schemes.
5. The distribution has a longer tail, which cannot be shown for confidentiality reasons.
6. A link to the source code will be including in the final version of the paper.

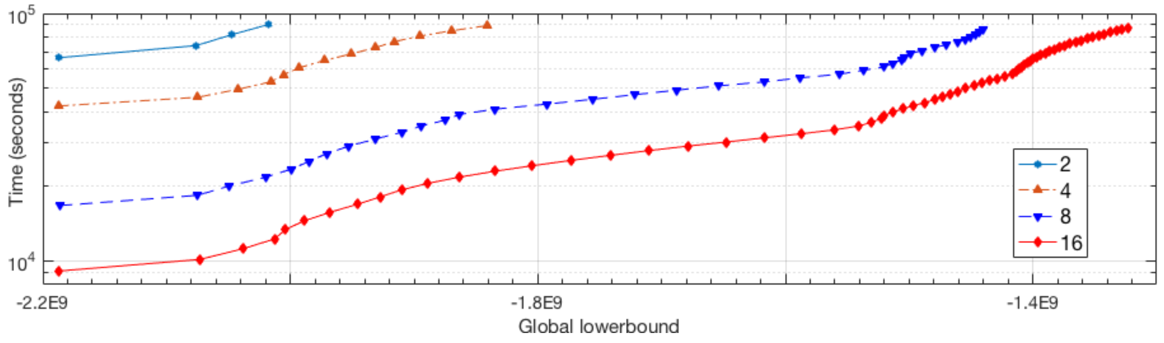


Figure 4: Time required to obtain approximations of the quality given by different global lower bounds when using 2, 4, 8 and 16 computational nodes. Each point corresponds to a single iteration of the algorithm. Note the log-scale on the y-axis.

In general, doubling the computational resources gives a speed-up factor of approximately 1.7.

5. 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 Flink to obtain a time-efficient and scalable implementation. Theoretical analysis of the algorithm supports the favorable learning behavior we have observed in practice.

The financial data-set that inspired this work is really a snap-shot of a data stream, and we plan to investigate the effects of replacing the underlying model structure with a dynamic model. 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.

We would also like to explore the potential of d-VMP for text data using LDA-like models. Here the high-dimensional multinomial distributions (size given by the size of the vocabulary) will have many states (i.e., words) with low probability. Therefore, stochastic approaches might experience problems when sub-sampling the data. Moreover, many large text corpora are stored in distributed environments, which also harms the use of these methods.

References

Matthew J. Beal. *Variational algorithms for approximate Bayesian inference*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

Hanen Borchani, Ana M. Martínez, Andrés Masegosa, Helge Langseth, Thomas D. Nielsen, Antonio Salmerón, Antonio Fernández, Anders L. Madsen, and Ramón Sáez. Modeling concept drift: A probabilistic graphical model based approach. In *Proc. of The Fourteenth Int. Symposium on IDA*, pages 72–83, 2015a.

Hanen Borchani, Ana M. Martínez, Andrés Masegosa, Helge Langseth, Thomas D. Nielsen, Antonio Salmerón, Antonio Fernández, Anders L. Madsen, and Ramón Sáez. Dynamic Bayesian modeling for risk prediction in credit operations. In *Proc. of SCAI*, 2015b.

Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C. Wilson, and Michael I. Jordan. Streaming variational Bayes. In *Advances in NIPS 26*, pages 1727–1735. 2013.

Trevor Campbell and Jonathan P. How. Approximate decentralized Bayesian inference. In *Proc. of the Thirtieth Conf. on UAI*, pages 102–111, 2014.

Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1):107, 2008. doi: 10.1145/1327452.1327492.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

James Foulds, Levi Boyles, Christopher DuBois, Padhraic Smyth, and Max Welling. Stochastic collapsed variational Bayesian inference for latent Dirichlet allocation. In *Proc. of the Int. Conf. on Knowledge Discovery and Data Mining*, pages 446–454. ACM, 2013.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

Mohammad Emtiyaz Khan, Reza Babanezhad, Wu Lin, Mark Schmidt, and Masashi Sugiyama. Convergence of proximal-gradient stochastic variational inference under non-decreasing step-size sequence. *arXiv preprint arXiv:1511.00146*, 2015.

Harold Joseph Kushner and G George Yin. *Stochastic approximation algorithms and applications*. Springer New York, 1997.

Zhi-Quan Luo and Paul Tseng. Error bounds and convergence analysis of feasible descent methods: A general approach. *Annals of Operations Research*, 46(1):157–178, 1993.

Stephan Mandt and David Blei. Smoothed gradients for stochastic variational inference. In *Advances in Neural Information Processing Systems*, pages 2438–2446, 2014.

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

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. doi: 10.1214/aoms/1177729586.

Masa-Aki Sato. Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681, 2001.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proc. of the Int. Conf. on Machine Learning (ICML-11)*, pages 681–688, 2011.

John M. Winn and Christopher M. Bishop. Variational message passing. *Journal of Machine Learning Research*, 6:661–694, 2005.