Probabilistic Methods for Divide-and-Conquer Distributed Learning

Abstract

A new probabilitistic extension of the Alternating Direction Method of Multipliers (ADMM) is developed, termed pADMM. This method is motivated by the goal of performing regularized empirical risk minimization in a distributed manner, for learning problems at massive data scale. The proposed pADMM shares statistical strength across the distributed nodes, it explicitly addresses small-sample bias on a given distributed node, and inference is performed with an efficient generalized expectation-maximization (gEM) approach. Encouraging results are demonstrated for matrix completion and logistic regression at massive scale, with comparisons made to several alternatives.

1. Introduction

Access to data at massive scale has proliferated recently. A significant machine learning challenge concerns development of methods that efficiently model and learn from data at this scale, while retaining analysis flexibility and sophistication.

Many statistical learning problems are formulated in terms of regularized empirical risk minimization (Hastie et al., 2001), and it is desirable to efficiently extend such methods to data at massive scale. When the size of the data is too large to be stored on a single machine, or at least too large to keep in a single localized memory, one popular solution is to store and process the data in a distributed manner. Consequently, the focus of this work is to study distributed learning algorithms (Bertsekas, 1989) for empirical risk minimization problems.

Divide-and-conquer approaches are intuitively appealing for designing distributed learning algorithms. In this setting the data are assumed partitioned into blocks, with each block assigned to one node of a cluster of computing nodes. Each node obtains a local estimate of the global model parameters, based its own data block; these local estimates are communicated to a central node, where a consensus estimate is manifested for the global parameters.

Divide-and-conquer approaches have been studied for a number of problems within a distributed setting, including the Bootstrap (Kleiner et al., 2012), matrix factorization (Mackey et al., 2011), logistic regression (Mann et al., 2009), kernel ridge regression (Zhang et al., 2013), and general smooth convex optimization problems (Zhang et al., 2012). The optimality of the estimate obtained from the divide-and-conquer approach crucially depends on the quality of the local estimators, which in turn depends on the information contained in each block, *e.g.,* the size of data block (Zhang et al., 2012; 2013). However, complicated models that require large-scale data to learn may have at least subsets of local parameters that are poorly estimated using only one block of the data; it is therefore of interest to infer the confidence of each block-dependent estimate. As the number of blocks (nodes) increases, this issue may become more severe, as the amount of data on a given node decreases, and hence block-dependent parameter uncertainty may rise. Additionally, block-specific bias may exist, caused by random partitioning of data into blocks; this is called the "small-sample bias" problem, studied in (Zhang et al., 2012; Scott et al., 2013).

To address the above challenges, we propose a series of probabilistic methods that model the uncertainty caused by the random partitions, facilitating the sharing of statistical strength among data blocks and nodes. We develop a new method of integrating the small-sample bias as part of the model. We also design a generalized Expectation-Maximization (gEM) algorithm for the proposed probabilistic methods, based on which we show that the probabilistic algorithms inherit similar computational merits of their well-studied non-probabilistic counterparts in the large-scale setting, to which we compare.

A series of algoithms are developed in this paper, summarized here for reader convenience. In Section 2.2 we discuss the Average Mixture (AVGM) method for distributed computing. While widely studied, this approach can be sensitive to node-dependent variation in the data considered, particularly for large-scale distributed problems. To account for parameter uncertainty on each computing node, in Section 3.1 we explicitly infer node-dependent *distributions* on the model parameters, with AVGM generalized

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

to a Maximum Entropy Mixture (MEM) method. While MEM accounts for node-dependent uncertainty, there is no sharing of statistical strength among the nodes, and therefore in Section 3.2 a Hierarchical Model (HM) is developed. In the HM approach the priors on node-dependent parameters are characterized in terms of a new shared global parameter. This global parameter is accounted for throughout learning, sharing statistical strength across the nodes. Finally, in Section 3.3 the node-dependent distributions in the HM approach are required to have the same expectation of model parameters, yielding a new probabilitistic extension of the Alternating Direction Method of Multipliers (ADMM), termed pADMM. We show that pADMM represents an extension of several related distributed-computing methods, addressing small-sample bias, but with greater flexibility and generalization than existing approaches (*e.g.,* the hierarchical probabilistic setting avoids model tuning and the need for cross-validation, which can be expensive). We demonstrate in Section 6, with multiple model and data types, that the proposed pADMM is an attractive new approach for analysis of massive data within an regularized empirical risk minimization formulation, from the standpoints of both accuracy and speed.

2. Preliminaries

2.1. Problem formulation

In the following we consider a general regularized empirical risk minimization problem of the form $\min_\theta \ \sum_{n\in\Omega} l(y_n;\theta) + h(\theta)$, where θ is the parameter of interest, which could be a vector or a matrix; $l(y_n;\theta)$ denotes the loss function, which could be non-convex; $h(\theta)$ denotes the regularization function, which could be non-smooth; and $\mathcal{Y} = \{y_n, n \in \Omega\}$ denotes the observed dataset with y_n representing the n^{th} instance and Ω representing the dataset's index set. In a large-scale setting, the size of the dataset $|\mathcal{Y}|$ becomes too large for a single machine to store all of the data, or at least to keep the data in memory. In this work we focus on the case for which $\mathcal{Y}$ is stored in a distributed manner by a cluster of computing nodes connected through a network. More concretely, we assume $\mathcal{Y}$ is (randomly) partitioned into B blocks, where B may equal to the number of computing nodes in a cluster. We denote each block as $\mathcal{Y}_b = \{y_n, n \in \Omega_b\}$, with Ω_b representing the b^{th} data block's index set. In a distributed setting, the regularized empirical risk minimization problem is formulated as

$$\min_\theta f(\theta) = \sum_{b=1}^{B} l(\mathcal{Y}_b;\theta) + h(\theta) \qquad (1)$$

where $l(\mathcal{Y}_b;\theta) = \sum_{n\in\Omega_b} l(y_n;\theta)$ is the loss function for data block b. In the following we focus on distributed learning algorithms for solving (1), and $\hat{\theta} = \arg\min_\theta f(\theta)$ will be used to represent a (typically) local-optimal solution.

2.2. Average mixture based algorithms

One of the well studied distributed learning algorithms for (1) is the average mixture (AVGM) method (Mann et al., 2009; Zinkevich et al., 2010; Zhang et al., 2012). The basic idea of AVGM is to divide the data into blocks as stated in Section 2.1, and each block b is characterized by local parameter θ_b; the estimate of θ_b, denoted $\hat{\theta}_b$, is computed based on $\mathcal{Y}_b$. The estimate of global parameter θ is obtained by taking a (weighted) average over the local estimates $\hat{\theta}_b$. AVGM is summarized as

$$\hat{\theta}_b = \arg\min_{\theta_b} f(\theta_b), \quad \hat{\theta} = \textstyle\sum_{b=1}^{B} w_b\hat{\theta}_b / \sum_{b=1}^{B} w_b \qquad (2)$$

where $f(\theta_b)$ is the local objective function

$$f(\theta_b) = l(\mathcal{Y}_b;\theta_b) + \tfrac{1}{B}h(\theta_b) \qquad (3)$$

and w_b is a weight parameter, commonly set to $1/B$ (Mann et al., 2009; Zinkevich et al., 2010) or proportional to $|\mathcal{Y}_n|$ in (Zhang et al., 2012).

AVGM is categorized as one type of "divide-and-conquer" method, where the data are divided into blocks with a local estimate computed for each data block; the global estimate is obtained from a consensus of the local estimates. As theoretically and empirically demonstrated in (Zhang et al., 2012), statistical optimality is retained for AVGM given enough information per block, *e.g.,* when $|\mathcal{Y}_b|$ is not too small. However, almost by definition, complicated models that require large-scale data to learn may have at least subsets of local parameters θ_b that are poorly estimated based only on $\mathcal{Y}_b$, and therefore it is of interest to infer the confidence of each block-dependent estimate. For a fixed-size dataset, as the number of blocks B increases this issue may be exacerbated, as the amount of data on a given node decreases, and hence block-dependent parameter uncertainty may rise. Besides, block-specific bias may exist; this may be caused by the random partitioning of data into blocks, which is called the "small-sample bias" problem as studied in (Zhang et al., 2012; Scott et al., 2013). To address these challenges, in the following sections we focus on developing a series of probabilistic distributed methods as complements to their non-probabilistic counterparts. The AVGM approach discussed above provides the basic intuition and motivation, but the proposed models infer uncertainty of estimates within each block, and account for this when making overall estimates based on data from all B blocks.

3. Probabilistic Divide-and-Conquer Methods

In the proposed probabilistic methods, instead of finding a point estimate θ_b for each block b, we model θ_b as a latent (unobserved) random variable and consider a more gen-

eral problem of finding a distribution $q(\theta_b)$ over all possible configurations of it. In the following we first discuss modeling frameworks built on this idea, and then discuss the corresponding learning algorithm that enables the probabilistic methods to share similar computational merits of their well-studied non-probabilistic counterparts in the large-scale setting.

3.1. Modeling the uncertainty

Perhaps the most straightforward way of formulating an empirical-risk-minimization problem with respect to a distribution is via the maximum entropy principle, which is well-studied in the natural language processing literature (Berger et al., 1996). Under the maximum entropy principle, the problem becomes one of finding a distribution $\hat{q}(\theta_b)$, such that (i) we minimize the convex combination of local objective functions (3) of block b with respect to $\hat{q}(\theta_b)$, denoted as $\mathbb{E}_{\hat{q}(\theta_b)}[f(\theta_b)] = \int f(\theta_b)\hat{q}(\theta_b)d\theta_b$; and (ii) at the same time we maximize the entropy of $\hat{q}(\theta_b)$, denoted $\mathbb{H}[\hat{q}(\theta_b)] = -\int \hat{q}(\theta_b)\log\hat{q}(\theta_b)d\theta$. The problem is formulated as

$$\hat{q}(\theta_b) = \arg\min_{q(\theta_b)} -\mathbb{H}[q(\theta_b)] + \mathbb{E}_{q(\theta_b)}[f(\theta_b)] \quad (4)$$

After obtaining $\hat{q}(\theta_b)$, $\hat{\theta}$ is estimated as a (weighted) average of the mean of $\hat{q}(\theta_b)$, similar to AVGM:

$$\hat{\theta} = \textstyle\sum_{b=1}^{B} w_b \mathbb{E}_{\hat{q}(\theta_b)}[\theta_b] / \sum_{b=1}^{B} w_b \quad (5)$$

In the following we refer to (4) and (5) as the Maximum Entropy Mixture (MEM) method.

Concerning (4), by maximizing $\mathbb{H}[q(\theta_b)]$, we seek a rich/diverse set of possible parameters θ_b with which to estimate the global parameter θ, with the probability of θ_b represented by $q(\theta_b)$. However, we simultaneously want this rich set of parameters to fit the model well, in that the expected value of $f(\theta_b)$ with respect to $q(\theta_b)$ is small. Note that based on the estimated set of distributions $\{\hat{q}(\theta_1), \ldots, \hat{q}(\theta_B)\}$, we can assess the quality of the estimator $\hat{\theta}$, *e.g.,* calculating the credible/confidence interval of $\hat{\theta}$, similar to the Bootstrap-based methods in (Kleiner et al., 2012). To relate our method to AVGM, here we focus on the case for which a point estimate of $\hat{\theta}$ is of interest, taken as a (weighted) average of the mean of the parameters θ_b, as in (5).

After some simple algebra, the optimal solution for (4) has the form

$$\hat{q}(\theta_b) = \tfrac{1}{Z} e^{-f(\theta_b)} \quad (6)$$

where $Z = \int e^{-f(\theta_b)} d\theta_b$ is the normalization constant. Note that AVGM actually finds the maximum likelihood estimate of (6) with respect to θ_b, which suggests that AVGM is a special case of MEM.

3.2. Sharing statistical strength

One common feature of the AVGM and MEM methods is that the local estimators only use data from the their own block, *i.e.,* θ_b only depends on $\mathcal{Y}_b$, and is independent of data from other blocks. This strategy leads to communication-efficient distributed algorithms, because different computing nodes only need to communicate once when estimating the global parameter θ. However, as discussed at the beginning of Section 3, the information contained in one single block may not be sufficient to obtain an accurate estimate, and it could be helpful to share the statistical strength across data blocks when estimating the local parameters.

So motivated, we propose a method that facilitates the sharing of statistical strength among data blocks, through hierarchical modelling and Bayesian integration of prior information. Similar to MEM, we model θ_b as a latent random variable, with interest in finding the distribution $q(\theta_b)$. Toward that end we build a hierarchical model, placing prior $p(\theta_b|\theta)$ over each latent random variable θ_b, such that θ_b is dependent on the global parameter θ, but is conditionally independent of all other latent variables variables given θ. To incorporate the prior and hierarchical information into the empirical risk minimization framework, together with the convex combination of local objective functions, the distance between $q(\theta_b)$ and the prior distribution $p(\theta_b|\theta)$ is minimized, with the distance between distributions measured by the Kullback-Leibler (KL) divergence. The proposed hierarchical model (referred to as HM in the following sections) is summarized as follows

$$\min_{\theta, q(\cdot)} \textstyle\sum_{b=1}^{B} f(q(\theta_b), \theta) + h(\theta) \quad (7)$$

where $q(\cdot) = \{q(\theta_1), \ldots, q(\theta_B)\}$ and the local objective function is defined as

$$f(q(\theta_b), \theta) = \mathbb{D}[q(\theta_b)||p(\theta_b|\theta)] + \mathbb{E}_{q(\theta_b)}[l(\mathcal{Y}_b; \theta_b)] \quad (8)$$

where $\mathbb{D}[q(\theta_b)||p(\theta_b|\theta)] = \int q(\theta_b)\log\frac{q(\theta_b)}{p(\theta_b|\theta)}d\theta_b$ is the KL divergence between $q(\theta_b)$ and $p(\theta_b|\theta)$.

The same (global) parameter θ is used for all $p(\theta_b|\theta)$, and therefore data from all B blocks contribute toward estimating θ. Further, the global θ contributes toward learning each of the block-dependent θ_b, via $p(\theta_b|\theta)$. The integrated sharing of statistical strength associated with HM distinguishes it from AVGM and MEM (for each of which global averaging is only done *after* performing independent local computations).

Through the introduced hierarchy between data block $\mathcal{Y}_b$, θ_b and θ, the induced conditional independence suggests that $\mathcal{Y}_b$ and θ_b still is stored independent of other blocks in a distributed fashion, while the dependency between all latent variables $\{\theta_b\}_{b=1\ldots B}$ and global parameter θ via

prior distribution $p(\theta_b|\theta)$ facilitates the sharing of statistical strength across different blocks. The HM formulation naturally leads to a generalized Expectation-Maximization (gEM) algorithm[1] to estimate θ and $q(\theta_b)$ within the same framework, instead of two independent stages as in AVGM and MEM.

The gEM algorithm proceeds by iteratively applying two steps: the E-step finds a distribution $q(\theta_b)$ that minimizes objective function (7) while fixing θ, and evaluating the expectation and KL-divergence in (8) with respect to $q(\theta_b)$; and the M-step finds an estimate of θ that minimizes (7) while fixing $q(\theta_b)$. More concretely, in the k^{th} iteration, the E-step updates $q^{k+1}(\theta_b)$ by minimizing $f(q(\theta_b), \theta^k)$, which is shown has the following optimal solution:

$$q^{k+1}(\theta_b) = \tfrac{1}{Z} p(\theta_b|\theta^k) e^{-l(\mathcal{Y}_b;\theta_b)} \tag{9}$$

where $Z = \int_{\theta_b} p(\theta_b|\theta^k) e^{-l(\mathcal{Y}_b;\theta_b)} d\theta_b$ is the normalization constant. Given $q^{k+1}(\theta_b)$, the M-step consists of updating θ^{k+1} by solving the following problem

$$\min_\theta \ \sum_{b=1}^B \mathbb{E}_{q^{k+1}(\theta_b)}[-\log p(\theta_b|\theta)] + h(\theta) \tag{10}$$

Note that minimizing (8) with respect to $q(\theta_b)$ in the E-step is closely related to Bayes theorem: $p(\theta_b|\mathcal{Y}_b, \theta) = \frac{p(\theta_b|\theta)p(\mathcal{Y}_b|\theta_b)}{\int p(\theta_b|\theta)p(\mathcal{Y}_b|\theta_b)d\theta_b}$, with $p(\theta_b|\theta)$ and $p(\mathcal{Y}_b|\theta_b)$ representing the prior distribution density and likelihood function, respectively. Zellner (1988) shows that the Bayesian posterior density $p(\theta_b|\mathcal{Y}_b, \theta)$ can equivalently be found by solving the following minimization problem:

$$\min_{q(\theta_b)} \ \mathbb{D}[q(\theta_b)||p(\theta_b|\theta)] + \mathbb{E}_{q(\theta_b)}[-\log p(\mathcal{Y}_b|\theta_b)] \tag{11}$$

By comparing (8) with (11), we see that if $p(\mathcal{Y}_b|\theta_b) \propto e^{-l(\mathcal{Y}_b;\theta_b)}$, *i.e.,* when $l(\mathcal{Y}_b;\theta_b)$ is interpreted as a negtive log-likelihood function, then $q^{k+1}(\theta_b)$ in (9) is exactly the posterior distribution $p(\theta_b|\mathcal{Y}_b, \theta^k)$, *i.e.,* $q^{k+1}(\theta_b) = p(\theta_b|\mathcal{Y}_b, \theta^k)$. In this case, the gEM algorithm reduces to the classic EM algorithm for statistical models. In many scenarios the loss function $l(\mathcal{Y}_b;\theta_b)$ does have a probabilistic interpretation, *e.g.,* squared ℓ_2 and ℓ_1 loss functions are proportional to the Gaussian and Laplace negative log-likelihood function, respectively; however, some important loss functions don't have a probabilistic counterpart, *e.g.,* the hinge loss function for max-margin methods.

Related frameworks to (7) and (8) are found in the literature for different applications. Some are known as the minimum relative entropy method (Jaakkola et al., 1999; Zhu et al., 2012) for max-margin based discriminative learning tasks, or the posterior regularization method (Ganchev et al., 2010) and constrained Bayesian inference (Koyejo & Ghosh, 2013) for incorporating constraints into posterior inference.

[1]Termed a *generalized* EM because some loss functions $l(\mathcal{Y}_b;\theta_b)$ can't be interpreted as a likelihood function, as discussed below.

Algorithm 1 pADMM

Initialize: $\theta^0, \rho^0, \{q^0(\theta_b), \lambda_b^0, \gamma_b^0\}_{b=1\ldots B}$
for $k = 0, 1, 2, \ldots$ **do**
  E-step:
  for $b = 1$ **to** B in parallel **do**
    Update $q^{k+1}(\theta_b), \lambda_b^{k+1}$ based on (16), (17).
    Update γ_b^{k+1} based on (20).
  end for
  M-step:
  Update θ^{k+1} by solving (18) and ρ^{k+1} based on (20)
end for

3.3. Modelling the local bias

As discussed at the beginning of Section 3, in the large-scale setting, the dataset is (pre-)partitioned into blocks, and some blocks may provide unreliable or biased estimates, particularly as the amount of data in each block diminishes (*e.g.,* as one scales to a large number of nodes). In AVGM, this problem is addressed with the bootstrap/jackknife bias-correction method, as a post-processing step, where the local estimates are shifted to avoid an accumulation of biases in the global estimate (Zhang et al., 2012; Scott et al., 2013). Similar ideas may be applied to the MEM method.

In this section we explicitly model the local bias as integrated within the HM method. The basic idea is to constrain the latent random variables θ_b to be consistent with each other as well as the global parameter θ in expectation, which is equivalent to constraining $q(\theta_b)$ to share the same mean across all blocks. Note that although the means of $q(\theta_b)$ are now fixed to be the same, these distributions may still be flexible enough to allow block-dependent variability. The optimization problem becomes

$$\begin{array}{ll} \min_{\theta, q(\cdot)} & \sum_{b=1}^B f(q(\theta_b), \theta) + h(\theta) \\ \text{subject to} & \mathbb{E}_{q(\theta_b)}[\theta_b] = \theta, \ b = 1, \ldots, B \end{array} \tag{12}$$

where $f(q(\theta_b), \theta)$ is the local objective function defined in (8). This specialization of the HM approach is termed pADMM, for probabilistic Alternating Direction Method of Multipliers; we will show below that pADMM is closely related to Alternating Direction Method of Multipliers (ADMM). In order to make this approach clear, in the following we focus on a special case when $p(\theta_b|\theta)$ is Gaussian:

$$p(\theta_b|\theta) = \mathcal{N}(\theta_b|\theta, (\rho\gamma_b)^{-1}\mathbf{I}) \tag{13}$$

with mean θ and isotropic covariance matrix $(\rho\gamma_b)^{-1}\mathbf{I}$, where $\mathbf{I}$ denotes the identity matrix. In the following ρ and

γ_b will be referred to as the global and local precision parameter, respectively. Later in this section we will show ρ and γ_b are learned from data, which empirically will lead to faster convergence of the algorithm, and thus less communication between computing nodes.

In the next we apply the gEM algorithm discussed in Section 3.2 to pADMM. With the Gaussian assumption in (13), the local objective function $f(q(\theta_b), \theta)$ becomes (with constants independent of $q(\theta_b)$ and θ ignored):

$$f(q(\theta_b), \theta) = \mathbb{E}_{q(\theta_b)} \left[\frac{\rho\gamma_b}{2} ||\theta_b - \theta||_2^2 + l(\mathcal{Y}_b; \theta_b) \right] \quad (14)$$

We have the following Lagrangian for each local optimization problem

$$\min_{q(\theta_b),\lambda_b} \; f(q(\theta_b), \theta) + \lambda_b \cdot (\mathbb{E}_{q(\theta_b)}[\theta_b] - \theta) \quad (15)$$

where $f(q(\theta_b), \theta)$ is defined in (14), λ_b is the Lagrange dual variable, and $\cdot$ represents the dot product. The dual of (15) is readily derived, and in the E-step we propose to solve problem (15) via dual ascent, *e.g.,* the dual problem is solved using gradient ascent. With fixed dual variable and global parameter, the updating equation of $q(\theta_b)$ is:

$$q^{k+1}(\theta_b) = \tfrac{1}{Z} e^{-l(\mathcal{Y}_b;\theta_b) - \frac{\rho\gamma_b}{2}||\theta_b - \theta^k + \mu_b^k||_2^2} \quad (16)$$

where $Z = \int e^{-l(\mathcal{Y}_b;\theta_b) - \frac{\rho\gamma_b}{2}||\theta_b - \theta^k + \mu_b^k||_2^2} d\theta_b$ is the normalization constant. $\mu_b^k = \frac{1}{\rho\gamma_b}\lambda_b^k$ is the scaled dual variable, which acts in a similar role to the Bootstrap adjustment in (Zhang et al., 2012), and pADMM reduces to HM by fixing $\lambda_b^k = 0$. Inspired by ADMM (discussed in Section 5.1), the dual variable λ_b is updated using gradient ascent with step size $\rho\gamma_b$:

$$\lambda_b^{k+1} = \lambda_b^k + \rho\gamma_b(\mathbb{E}_{q^{k+1}(\theta_b)}[\theta_b] - \theta^k) \quad (17)$$

Finally, given $q^{k+1}(\theta_b)$ and λ_b^{k+1} obtained from the E-step for each local block b, the M-step updates θ^{k+1} by solving the following minimization problem:

$$\min_\theta \; \textstyle\sum_{b=1}^B \frac{\rho\gamma_b}{2} ||\theta - \mathbb{E}_{q^{k+1}(\theta_b)}[\theta_b] + \mu_b^{k+1}||_2^2 + h(\theta) \quad (18)$$

where $\mathbb{E}_{q^{k+1}(\theta_b)}[\theta_b]$ is simply the mean of $q^{k+1}(\theta_b)$, and recall that μ_b^{k+1} is the scaled dual variable. An optimization problem of form (18) generally solved efficiently by proximal gradient methods (Parikh & Boyd, 2013). As an example, when $h(\theta) = \beta(\alpha||\theta||_1 + (1-\alpha)||\theta||_2^2)$ corresponding to a convex combination of the ℓ_1 and ℓ_2 regularizer with $0 \le \alpha \le 1$ and $\beta \ge 0$, the update for θ^{k+1} is

$$\theta^{k+1} = \frac{S(\sum_{b=1}^B \rho\gamma_b \mathbb{E}_{q^{k+1}}[\theta_b], \beta\alpha)}{\sum_{b=1}^B \rho\gamma_b + \beta(1-\alpha)} \quad (19)$$

where $S(a, b) = \text{sign}(a)(|a| - b)_+$ is the soft thresholding operator (Friedman et al., 2010). Note that (19) resembles the weighted average strategy in (2), however, instead of fixing w_b to $1/B$ or proportional to $|\Omega_b|$, the weight corresponds to the product of global and local precision parameters $\rho\gamma_b$. In the next we propose to update ρ and γ_b in the M-step, done by minimizing (12) with respect to ρ and γ_b in turn. Denoting the expected squared residual as $r^{k+1} = \mathbb{E}_{q^{k+1}(\theta_b)}[||\theta_b^{k+1} - \theta^{k+1} + \mu_b^{k+1}||_2^2]$, the update equations for ρ^{k+1} and γ_b^{k+1} is

$$\rho^{k+1} = \tfrac{BP}{\sum_{b=1}^B \gamma_b^k r^{k+1}}, \quad \gamma_b^{k+1} = \tfrac{P}{\rho^{k+1} r^{k+1}} \quad (20)$$

where P denotes the dimension of θ_b. The pADMM method is summarized in Algorithm 1; note that by setting $\mu_b = 0$, pADMM reduces to HM. Note that, under the Gaussian assumption (13) for the HM method with γ_b^k fixed and with ρ^k set to be an increasing sequence, *e.g.,* $\rho^1 < \rho^2 < \ldots$ and $\lim_{k\to\infty} \rho^k = \infty$, it reduces to the penalty methods (Bertsekas & Tsitsiklis, 1996) when only a point esimate of θ_b is of interest.

3.4. Full Bayesian treatment

Although the above proposed methods, *i.e.,* MEM, HM and pADMM, are motivated by the optimization problem (1), they can naturally be extended to a fully Bayesian setting (rather than making a point estimate for θ, we estimate its posterior distribution). As shown by (11), the gEM algorithms proposed for HM and pADMM reduce to classic EM algorithms when $l(\mathcal{Y}_b; \theta_b)$ corresponds to a negative log-likelihood function. Based on this observation, if the global parameter θ is also modeled as a random variable, and the regularization function $h(\theta)$ in (1) is replaced by a prior distribution, posterior inference is performed on θ, and then we have a fully Bayesian model. (Scott et al., 2013; Neiswanger et al., 2013) represent two recent works in this direction, although their frameworks are both based on the AVGM setting. The proposed pADMM may be viewed as a Bayesian model for the node-dependent parameters θ_b, with posterior distributions approximated as $q(\theta_b)$; the global θ may be viewed as a hyperparameter, for which a point estimate is employed.

4. Practical Issues of the gEM Algorithm

One important issue in making gEM applicable for large-scale data is to evaluate the expectations efficiently in the E-step, *e.g.,* $\mathbb{E}_{q(\theta_b)}[\theta_b]$, which in turn generally requires an analytic expression for the distribution $q(\theta_b)$, or at least the ability to draw samples from it. Given the expectations found in the E-step, efficient optimization algorithms, *e.g.,* the proximal gradient method or stochastic gradient based methods, become applicable for the M-step. Consequently, the first key technical component in our implementation is adoption of variational methods (Wainwright & Jordan, 2008) in the E-step, which transforms the gEM

algorithm to a variational EM algorithm; if we extend our method to be fully Bayesian as discussed in Section 3.4, gEM becomes the variational Bayesian inference algorithm (Bishop, 2006).

As discussed in Sections 3.1-3.3, for MEM, HM and pADMM the distribution $q(\theta_b)$ of interest is found by minimizing the local objective function (8), which could be problematic when the expectation $\mathbb{E}_{q(\theta_b)}[l(\mathcal{Y}_b;\theta_b)]$ is not analytic. Two types of variational methods are applied here, commonly referred to as local and global variational methods in the literature (Bishop, 2006). Local variational methods first find an upper bound of the loss function $\tilde{l}(\mathcal{Y}_b;\theta_b;\xi_b) \geq l(\mathcal{Y}_b;\theta_b)$, then the expectation of the upper bound $\mathbb{E}_{q(\theta_b)}[\tilde{l}(\mathcal{Y}_b;\theta_b;\xi_b)]$ is minimized to find $q(\theta_b)$, with ξ_b representing some variational parameters that is optimized to tighten the gap between $\tilde{l}(\mathcal{Y}_b;\theta_b;\xi_b)$ and $l(\mathcal{Y}_b;\theta_b)$. Local variational method are well studied for various types of loss functions and likelihood functions, a few examples include (Zhu et al., 2012) for hinge loss function, (Jaakkola & Jordan, 2000; Khan et al., 2010) for the logistic/soft-max loss function, and (Khan et al., 2013; Wang & Blei, 2013) for general non-conjugate likelihood functions.

Global variational methods, on the other hand, directly seek an approximation to the true underlying distribution over all random variables by restricting the range of $q(\theta_b)$ over which the optimization is performed, *e.g.,* the mean field methods restrict $q(\theta_b)$ to take factorized forms. Mean field based variational methods are found in many Bayesian hierarchical models with latent variables and intractable posterior distributions (Wainwright & Jordan, 2008). Besides their common usage, here we also propose to use mean field variational methods from a computational perspective, in that by restricting $q(\theta_b) = \prod_i q(\theta_{bi})$ to be fully factorized across its components, the variational methods proceeds by updating each scalar based density $q(\theta_{bi})$ in turn. Although such fully factorized constraint may lead to less accurate estimate of $q(\theta_b)$, it avoids computing high-dimensional quantities, *e.g.,* the covariance matrix of θ_b, and it is efficient and scalable in large-scale applications, which resembles some recent success of the coordinate descent algorithms (Friedman et al., 2010; Bradley et al., 2011; Yuan et al., 2012; Yu et al., 2013).

Although sampling-based methods, *e.g.,* MCMC and importance re-sampling, generally don't need to evaluate the normalization constant Z for $q(\theta_b)$ explicitly, and can readily be applied to our framework, the reason we prefer variational methods to sampling is two-fold: (i) It is generally more difficult to monitor the convergence of a sampling-based method relative to a deterministic variational method; and (ii) our experiments suggest that in the high-dimensional and large-scale setting, MCMC tends to take more iterations to reach a reasonable solution than variational method, and importance re-sampling tends to collapse to a single point.

5. Related Distributed Learning Methods

5.1. Distributed learning via ADMM

ADMM has generated much attention for a variety of applications recently; a comprehensive survey of ADMM, especially its application to distributed learning, is found in (Boyd et al., 2010). Here we only summarize important aspects of ADMM. First note that we can rewrite (1) into the following equivalent problem:

$$\begin{aligned} &\min_{\theta,\theta_b} \ \textstyle\sum_{b=1}^{B} l(\mathcal{Y}_b,\theta_b) + h(\theta) \\ &\text{subject to } \ \theta_b - \theta = 0, \ b = 1,\ldots,B \end{aligned} \tag{21}$$

The ADMM formulation for the problem (21) is derived directly from the following augmented Lagrangian

$$L_\rho(\{\theta_b,\lambda_b\}_{b=1}^B,\theta) = \textstyle\sum_{b=1}^{B} L_\rho(\theta_b,\lambda_b,\theta) + h(\theta) \tag{22}$$

Here $L_\rho(\theta_b,\lambda_b,\theta)$ is the local augmented Lagrangian:

$$L_\rho(\theta_b,\lambda_b,\theta) = l(\mathcal{Y}_b,\theta_b) + \lambda_b\cdot(\theta_b-\theta) + \tfrac{\rho}{2}||\theta_b-\theta||_2^2 \tag{23}$$

where $\rho \geq 0$ is the tuning parameter of the augmented Lagrangian. ADMM proceeds by iteratively updating θ_b, λ_b for each block b independently and θ:

$$\theta_b^{k+1} = \arg\min_{\theta_b} \ L_\gamma(\theta_b,\lambda_b^k,\theta^k) \tag{24}$$

$$\lambda_b^{k+1} = \lambda_b^k + \rho(\theta_b^{k+1} - \theta^k) \tag{25}$$

$$\theta^{k+1} = \arg\min_{\theta} \ \textstyle\sum_{b=1}^{B} \tfrac{\rho}{2}||\theta_b^{k+1}-\theta+\mu_b^{k+1}||_2^2 + h(\theta) \tag{26}$$

Note that in the gradient ascent update in (25) the step size is chosen to be ρ. The motivation behind this choice is explained by Boyd *el al.* (2010), that by using ρ as the step size, the iterate $\theta_b^{k+1}, \lambda_b^{k+1}$ in (24 - 25) is dual feasible, and as the ADMM proceeds the primal residual $\theta_b - \theta$ converges to zero, together with the dual feasible condition the procedure (24 - 26) will yield optimal solution $\hat{\theta}$.

Now comparing the update equations for ADMM in (24 - 26) with that of pADMM (14, 16 - 18), we see that the gEM algorithm for pADMM may be thought of as a probabilistic version of ADMM, hence the name. On the other hand, ADMM may be thought of as a way of finding a point estimate for pADMM, which again resembles the connections between AVGM and MEM, and HM and penalty methods. Note that carefully tuning the parameter ρ is one caveat of reaching fast convergence of ADMM, which is commonly done by cross-validation, which could be particularly computationally expensive in a large-scale setting, as one need perform multiple passes over the whole dataset. However,

as mentioned above, fast convergence is important for iterative algorithms in distributed learning, as fewer iterations means less expensive communication among computing nodes; this is particularly critical when the dimension of the parameter is high. In pADMM, by contrast, ρ is updated automatically, as in (20). Note that in pADMM the block-specific precision parameter γ_b is modelled explicitly and also updated in (20), which leads to faster convergence as will be showed empirically through experiments, while for ADMM modelling the block-specific parameters will unfortunately increase the number of parameters to tune.

5.2. Distributed (sub-)gradient method

We have focused on "divide-and-conquer" type frameworks thus far, and in this section we briefly discuss another type of distributed learning algorithm, the distributed sub-gradient method, which will serve as an important baseline when we compare different distributed algorithm empirically through experiments. For the distributed sub-gradient method for (1), in each iteration the sub-gradients $\partial l(\mathcal{Y}_b;\theta)$ are computed independently for each block b, and these separate sub-gradients are then summed up to compute the exact global sub-gradients $\sum_{b=1}^{B}\partial l(\mathcal{Y}_b;\theta)$, which are used to perform the optimization step and update the parameter θ received by all B blocks for the next iteration's sub-gradients computation. The distributed sub-gradient method is guaranteed to find the (local) optimal solution, however, it requires relatively many iterations before convergence, which in turn requires frequent communications among nodes in the cluster. On the other hand, the asymptotically optimal AVGM and MEM methods represent another extreme of the distributed learning framework, where communication only happens once, but the solution might not be optimal when each data block is insufficient.

6. Empirical Study

In this section we solve two challenging problems with the proposed methods and the variational gEM algorithm described above: the non-smooth ℓ_1-regularized logistic regression, and the non-convex low-rank matrix completion. In the following we first discuss the experimental setting.

6.1. Experimental setting

The distributed statistical optimization experimental environment is set up by the Spark cluster computing framework (Zaharia et al., 2010) of version 0.8.1[2]. Spark provides a fault-tolerant abstraction for in-memory fast iterative computing, that can run on either a single multi-core machine or cluster with up to hundreds of computing nodes. With Spark we conduct experiments on both multi-core and distributed environment. In the multi-core setting, we use a 16-core AMD Opteron 6212 processor with 50 Gigabytes memory. For the distributed setting, we build two types of clusters for dataset of different scales. For the moderately large dataset, we build the cluster with Amazon EC2 machines, where each computing node is a general purpose m1.xlarge instance with 4 virtual CPUs and 15 Gigabytes memory [3]. For industrially large dataset, the cluster consists of computing nodes each equipped with a 24-core Intel Xeon X5650 processor and 24 Gigabytes memory. All algorithms are implemented with Spark to make fair comparisons.

6.2. Distributed ℓ_1-regularized logistic regression

In logistic regression the each training data y_n contains two parts, $y_n = \{l_n, x_n\}$ where $l_n \in \{-1, +1\}$ represents the label and x_n represents the feature vector. The loss function in (1) is $l(\mathbb{Y}_b;\theta) = \sum_{n\in\Omega_b}\log(1+\exp(-l_n(x_n\cdot\theta)))$, and the regularization function is $h(\theta) = C||\theta||_1$, where C is a tuning parameter controls the sparsity of the parameter θ. And we assume the prior distribution $p(\theta_b|\theta)$ is Gaussian as in (13) for HM and pADMM.

In this case $q(\theta_b)$ doesn't readily have an analytical form, as the normalization constant Z in (6) for MEM, (9) for HM, and (16) for pADMM are all intractable. As mentioned in Section 4, we will use local and global variational methods here. For the local variational method, Laplace variational method (Lap) (Wang & Blei, 2013) and the Bohning bound based variational method (Bohn) (Khan et al., 2010) are used to obtain a quadratic approximation of $l(\mathbb{Y}_b;\theta)$, based on which $q(\theta_b)$ becomes tractable. For the global variational (mean-field) method, we assume $q(\theta_b) = \prod_{i=1}^{P} q(\theta_{bi})$ where P denotes the number of features. Jaakkola bound based method (Jaakkola & Jordan, 2000) are not adopted here because our preliminary results suggest that it doesn't work well with the mean-field assumption. We test the logistic regression on two datasets, one is the KDDCup2010 (bridge to algebra) dataset, which is commonly used as a benchmark dataset for binary classification, and is public available [4], and we filtered out features that occur less than 5 times across the training examples. The second dataset consists of 60 days of advertising event logs for a major social network site. We use the first 45 days of data as training and the last 15 days data as test. Both the training and test sets consist about 170 and 40 Millions of events. The features are extracted from user and ad campaign information, for instance, the ad campaign features include n-grams, categories, advertiser characteristics, among others. We use mutual information and the

[2] http://spark.incubator.apache.org

[3] http://aws.amazon.com/ec2/instance-types

[4] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

Table 1. Statistics and parameters for each dataset

	P	$\|\Omega\|$	$\|\bar{\Omega}\|$	nnz
KDDCup2010	4,156,973	19,264,097	748,401	562,968,283
Ads Click	100K	170M	40M	10B

minimum support criteria to do feature selection on these features and the two-way interactions between all pairs. Finally we obtain a feature set that contains about 100K binary features. The important statistics for each dataset is summarized in Table 6.2.

6.3. Distributed low-rank matrix completion

In our notation, $\mathcal{Y} = \{y_n, n \in \Omega\}$ corresponds to observed entries of a matrix $\mathbf{Y}$ of dimension $I \times J$, $n = (i, j)$ indexes each entry, and Ω is a subset of $\{1, \ldots, I\} \otimes \{1, \ldots, J\}$ denoting the indexes of the observed entries in $\mathbf{Y}$ with $\otimes$ denoting the Cartesian product. And similarly $\theta \in \mathbb{R}^{I \times J}$ denotes the model parameters of interest. The loss function is quadratic $l(\mathcal{Y}; \theta) = \sum_{(i,j)\in\Omega}(y_{i,j} - \theta_{i,j})^2$, and the regularization functions $h(\theta)$ we consider here are the nuclear norm and the γ_2-norm (also know as the max-norm) (Srebro et al., 2004). We assume θ has rank at most K, in which case θ can be explicitly written as UV' where $U \in \mathbb{R}^{I \times K}$ and $V \in \mathbb{R}^{J \times K}$. Such approximation transforms the low-rank matrix completion a non-convex problem, with objective function now can be written as

$$f(U, V) = \sum_{(i,j)\in\Omega}(y_{i,j} - U_{i\cdot}V'_{j\cdot})^2 + h(U) + h(V) \quad (27)$$

Note that the nuclear norm and γ_2-norm on θ can be equivalent formulated in terms of U and V, such that the nuclear norm can be written as $h(U) = ||U||_F^2 := \sum_i \sum_k U_{ik}^2$, and γ_2-norm can be written as $h(U) = ||U||_{2,\infty}^2 := \max_i(\sum_k U_{ik}^2)$ (Recht et al., 2010). The optimization techniques for both norms that could be used in the M-step of the gEM algorithm can be found at (Salakhutdinov & Mnih, 2007; Lee et al., 2010). In the large-scale setting, we randomly partition the observed data $\mathcal{Y}$ into B blocks similar to the two-way random partition method in (Recht & Ré, 2013), with each block corresponds to a sub-matrix of size approximately $\frac{I}{B_r} \times \frac{J}{B_c}$, such that $B_r B_c = B$, and with parameters U and V conformally partitioned into B_r and B_c blocks respectively. Consequently, the pADMM formulation of the low-rank matrix completion problem is

$$\begin{aligned} &\min_{U,V,q(\cdot)} \ \sum_{b=1}^{B} f(U_b, V_b, U, V) + h(U) + h(V) \\ &\text{subject to } \mathbb{E}_{q(U_b)} = U_b, \ \mathbb{E}_{q(V_b)} = V_b \end{aligned} \quad (28)$$

Table 2. Comparison of memory and network usage for two types of matrix factorization algorithms

	network per epoch	memory per core
ALS/CD	$\mathcal{O}((M+N)KB)$	$\mathcal{O}(MK + NK + \frac{\|\Omega\|}{B})$
Proposed	$\mathcal{O}(MKB_c + NKB_r)$	$\mathcal{O}(\frac{MK}{B_r} + \frac{NK}{B_c} + \frac{\|\bar{\Omega}\|}{B})$

Table 3. Statistics for each dataset

	I	J	Ω	$\|\bar{\Omega}\|$
Netflix	2,649,429	17,770	99,072,112	1,408,395
KDDCup2011	1,000,990	624,961	252,800,275	4,003,960
Jumbo (Synthetic)	1,000,000	1,000,000	1,983,749,510	23,328,835

References

Berger, A. L., Della Pietra, V. J. Della, and Della Pietra, S. A. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1), 1996.

Bertsekas, D. *Parallel and Distributed Computation: Numerical Methods*. 1989.

Bertsekas, D. and Tsitsiklis, J. *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. 1996.

Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 2006.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 2010.

Bradley, J. K., Kyrola, A., Bickson, D., and Guestrin, C. Parallel coordinate descent for L1-regularized loss minimization. In *ICML*, 2011.

Friedman, J., Hastie, T., and Tibshirani, R. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 2010.

Ganchev, K., Graça, J., Gillenwater, J., and Taskar, B. Posterior regularization for structured latent variable models. *J. Mach. Learn. Res.*, 2010.

Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning*. 2001.

Jaakkola, T. S. and Jordan, M. I. Bayesian parameter estimation via variational methods. *Statistics and Computing*, 2000.

Jaakkola, T. S., Meila, M., and Jebara, T. Maximum entropy discrimination. In *NIPS*, 1999.

Khan, M. E., Bouchard, G., Marlin, B. M., and Murphy, K. P. Variational bounds for mixed-data factor analysis. In *NIPS*, 2010.

Khan, M. E., Aravkin, A., Friedlander, M., and Seeger, M. Fast dual variational inference for non-conjugate latent gaussian models. In *ICML*, 2013.

Kleiner, A., Talwalkar, A., Sarkar, P., and Jordan, M. I. The big data bootstrap. In *ICML*, 2012.

Koyejo, O. and Ghosh, J. Constrained bayesian inference for low rank multitask learning. In *UAI*, 2013.

Lee, J., Recht, B., Srebro, N., Salakhutdinov, R. R., and Tropp, J. A. Practical large-scale optimization for max-norm regularization. In *NIPS*, 2010.

Mackey, L., Talwalkar, A., and Jordan, M. I. Divide-and-conquer matrix factorization. In *NIPS*, 2011.

Mann, G., McDonald, R., Mohri, M., Silberman, N., and Walker, D. Efficient large-scale distributed training of conditional maximum entropy models. In *NIPS*, 2009.

Neiswanger, W., Wang, C., and Xing, E. P. Asymptotically exact, embarrassingly parallel MCMC. *arXiv:1311.4780*, 2013.

Parikh, N. and Boyd, S. Proximal algorithms. *Foundations and Trends in Optimization*, 2013.

Recht, B. and Ré, C. Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation*, 5:201–226, 2013.

Recht, B., Fazel, M., and Parrilo, P. Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

Salakhutdinov, R. and Mnih, A. Probabilistic matrix factorization. In *NIPS*, 2007.

Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H. A., George, E. I., and McCulloch, R. E. Bayes and Big data: The consensus Mnote Caril algorithm. In *Bayes 250*, 2013.

Srebro, N., Rennie, J., and Jaakkola, T. Maximum margin matrix factorization. In *NIPS*, 2004.

Wainwright, M. J. and Jordan, M. I. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 2008.

Wang, C. and Blei, D. M. Variational inference in nonconjugate models. *J. Mach. Learn. Res.*, 2013.

Yu, H.-F., Hsieh, C.-J., Si, S., and Dhillon, I. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *ICDM*, 2013.

Yuan, G.-X., Ho, C.-H., and Lin, C.-J. An improved glmnet for l1-regularized logistic regression and support vector machines. *J. Mach. Learn. Res.*, 2012.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. Spark: Cluster computing with working sets. In *HotCloud*, 2010.

Zellner, A. Optimal information processing and Bayes's theorem. *The American Statistician*, 1988.

Zhang, Y., Duchi, J., and Wainwright, M. Communication-efficient algorithms for statistical optimization. In *NIPS*, 2012.

Zhang, Y., Duchi, J., and Wainwright, M. Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates. *arXiv:1305.5029*, 2013.

Zhu, J., Ahmed, A., and Xing, E. P. Medlda: Maximum margin supervised topic models. *J. Mach. Learn. Res.*, 2012.

Zinkevich, M., Smola, A., Weimer, M., and Li, L. Parallelized stochastic gradient descent. In *NIPS*, 2010.