Probabilistic Alternating Direction Method of Multipliers

Abstract

1. Introduction

And in sum, the main contribution of this work is that we formulated the distributed learning problem into a series of probabilistic frameworks in Section 3. And importantly, with the proposed generalized Expectation-Maximization (gEM) algorithm, we show that the probabilistic methods share similar computational merits of their well-studied non-probabilistic counterparts in the large-scale setting.

2. Preliminaries

2.1. Problem formulation

In the following we consider a general regularized empirical risk minimization problem of the form $\min_\theta \sum_{n\in\Omega} l(y_n;\theta) + h(\theta)$, here 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 when $\mathcal{Y}$ is stored distributedly by a cluster of computing nodes connected through network. More concretely, we assume $\mathcal{Y}$ is partitioned into B blocks, where B may equal to the number of computing nodes in a cluster, and 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 the distributed setting, the regularized empirical risk minimization problem can be formulated as

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

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 will focus on distributed learning algorithms solving (1), and $\hat{\theta} = \arg\min_\theta f(\theta)$ will be used to represent the (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 augment a local parameter θ_b for each block b, and the estimate of θ_b, denoted as $\hat{\theta}_b$, is computed based on $\mathcal{Y}_b$. Finally the estimate of global parameter θ is obtained by taking a (weighted) average over local estimates$\hat{\theta}_b$. Mathematically AVGM can be summarized as

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

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

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

and w_b is some weight parameter commonly set to be $1/B$ (Mann et al., 2009; Zinkevich et al., 2010) or proportional to $|\mathcal{Y}_n|$ in (Zhang et al., 2012). The AVGM is one kind of the "divide-and-conquer" methods, where the data are divided into blocks with local estimate computed for each data block, then the global estimate is obtained from a consensus of the local estimates. In the following we will focus on developing methods with similar spirit.

3. Probabilistic Frameworks for Distributed Learning

In the AVGM type of algorithm, to obtain a quality empirical minimizer $\hat{\theta}$ one needs enough information in each data block b to estimate θ_b as demonstrated in (Zhang et al., 2012), however, almost by definition models that require large-scale dataset will have at least subsets of local parameters θ_b that cannot be accurately estimated only based on $\mathcal{Y}_b$. Besides, the partition of $\mathcal{Y}$ into blocks $\{\mathcal{Y}_1, \ldots, \mathcal{Y}_B\}$ may also introduce extra uncertainty and bias within each block that need to be considered when designing a distributed learning algorithm.

In this work we propose a probabilistic approach to address the aforementioned problems. The core idea is that instead of finding a point estimate θ_b for each block b, we consider a more general problem where we are interested in finding

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

a distribution $q(\theta_b)$ over all possible configurations of the local parameters θ_b. In the following we will first develop a series of frameworks based this idea and then discuss the corresponding inference problems.

3.1. From a maximum entropy perspective

Perhaps the most straightforward way of formulating a empirical risk minimization problem with respect to a distribution is via the maximum entropy principle, where the problem becomes finding a distribution $\hat{q}(\theta_b)$, such that (i) the convex combination of local objective functions (3) of block b with respect to $\hat{q}(\theta_b)$, which is denoted as $\mathbb{E}_{\hat{q}(\theta_b)}[f(\theta_b)] = \int f(\theta_b)\hat{q}(\theta_b)d\theta_b$, is minimized, and (ii) the entropy of $\hat{q}(\theta_b)$, which is denoted as $\mathbb{H}[\hat{q}(\theta_b)] = -\int \hat{q}(\theta_b)\log\hat{q}(\theta_b)d\theta$, is maximized. After obtaining $\hat{q}(\theta_b)$, $\hat{\theta}$ is estimated as a (weighted) average of the mean of $\hat{q}(\theta_b)$. Mathematically, the problem is formulated as:

$$\begin{aligned} \hat{q}(\theta_b) &= \arg\min_{q(\theta_b)} -\mathbb{H}[q(\theta_b)] + \mathbb{E}_{q(\theta_b)}[f(\theta_b)] \\ \hat{\theta} &= \textstyle\sum_{b=1}^{B} w_b \mathbb{E}_{\hat{q}(\theta_b)}[\theta_b] / \sum_{b=1}^{B} w_b \end{aligned} \tag{4}$$

In the following we refer (4) as Maximum Entropy Mixture (MEM) method. The intuitive meaning of (4) can be understood as by maximizing the entropy (minimizing the negative entropy) of $q(\theta_b)$, $q(\theta_b)$ is chosen to be the distribution that makes the least claim to being informed beyond minimizing the convex combination of local objective functions $\mathbb{E}_{q(\theta_b)}[f(\theta_b)]$. The maximum entropy principle is widely used in natural language processing literature. 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 $\hat{\theta}$ as discussed in (Kleiner et al., 2012), but in order to relate our method to the AVGM method, for the MEM method we only consider the case where the point estimate of $\hat{\theta}$ only of interest, which is taken to be a (weighted) average of the mean of the parameters θ_b as in (4).

It can be shown that the optimal solution for (4) has the following simple form

$$\hat{q}(\theta_b) = \frac{1}{Z} e^{-f(\theta_b)} \tag{5}$$

here $Z = \int e^{-f(\theta_b)} d\theta_b$ is the normalization constant. Note that when $\hat{q}(\theta_b)$ is constrained to be a Dirac delta distribution with point mass at $\hat{\theta}_b$, *e.g.,* $\hat{q}(\theta_b) = \delta_{\hat{\theta}_b}(\theta_b)$, some simple algebra will show that $\hat{\theta}_b$ corresponds to the empirical local minimizer of AVGM method in (2), which suggests that the AVGM method based solution is a special case of the MEM method. Another way to look at it is that the empirical minimizer of AVGM in (2) is actually the maximum likelihood estimator of distribution $\hat{q}(\theta_b)$ in (5).

3.2. From a hierarchical modelling perspective

One common feature of the AVGM and MEM methods is that, since the whole dataset $\mathcal{Y}$ is partitioned into blocks at different computing nodes, each computing node b computes the local estimate only uses data in the block b, *i.e.,* only depends on $\mathcal{Y}_b$, and is independent of data from other blocks. Although this strategy leads to communication-efficient algorithms as different computing nodes only needs to communicates once (Zhang et al., 2012), as discussed in the beginning of Section 3 that the information contained in a single block may not be sufficient to obtain accurate local estimators, it may be helpful to share the statistical strength of data blocks among different computing nodes, while not increasing too much communication cost among the computing nodes.

In the next we propose a method that facilitates the sharing of statistical strength among data blocks of different computing nodes in a distributed setting, through hierarchical modelling and Bayesian integration of prior information. In this framework we model θ_b as latent (unobserved) random variable, with prior distribution $p(\theta_b|\theta)$, *i.e.,* the local latent variable θ_b is dependent on the global parameter θ *a priori*, and the latent random variables θ_b is conditionally independent of each other given θ. Similar to MEM, we are interested in finding a distribution $q(\theta_b)$ over the latent random variable θ_b by minimizing a convex combination of local objective functions, but different than MEM here we make additional assumption on $q(\theta_b)$, such that 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 divergence. The proposed hierarchical model (referred to as HM in the following sections) can be formulated as follows

$$\min_{\theta, q(\cdot)} \ \textstyle\sum_{b=1}^{B} f(q(\theta_b), \theta) + h(\theta) \tag{6}$$

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

$$f(q(\theta_b), \theta) = \mathbb{D}[q(\theta_b)||p(\theta_b|\theta)] + \mathbb{E}_{q(\theta_b)}[l(\mathcal{Y}_b; \theta_b)] \tag{7}$$

where $\mathbb{D}[q(\theta_b)||p(\theta_b|\theta)] = \int q(\theta_b)\log\frac{q(\theta_b)}{p(\theta_b|\theta)}d\theta_b$ denotes the Kullback-Leibler (KL) divergence between $q(\theta_b)$ and $p(\theta_b|\theta)$. We can recover the previous MEM method by placing an uninformative (possibly improper) prior on θ_b.

Note that through the introduced hierarchy between data block $\mathcal{Y}_b$, latent random variable θ_b and global parameter θ, the HM method maintains the independence among different data blocks, *i.e.,* they still can be stored in a distributed fashion, while facilitating the sharing of statistical strength among latent random variables in different blocks, as they are dependent on the same global parameter θ through prior distribution $p(\theta_b|\theta)$. Besides, the formulation of HM naturally leads us to a generalized Expectation-Maximization

(gEM) algorithm to estimate θ and $q(\theta_b)$ iteratively within the same framework, instead of two separate steps as in AVGM and MEM, as we discuss in the following. The gEM algorithm proceeds by iteratively applying two steps, the E-step finds a distribution $q(\theta_b)$ that minimizes objective function (6) given a fixed θ, and the M-steps finds an estimator θ that minimizes (6), based on the integrations (*i.e.,* the expectation and KL-divergence) in (6) calculated via $q(\theta_b)$ returned from the E-step. More concretely in the E-step we solves the optimization problem $q^{k+1}(\theta_b) = \arg\min_{q(\theta_b)} f(q(\theta_b), \theta^k)$, where the superscript k indexes the iteration, and $q^{k+1}(\theta_b)$ can be shown has the following optimal solution:

$$q^{k+1}(\theta_b) = \frac{1}{Z} p(\theta_b|\theta^k) e^{-l(\mathcal{Y}_b;\theta_b)} \tag{8}$$

where $Z = \int_{\theta_b} p(\theta_b|\theta^k) e^{-l(\mathcal{Y}_b;\theta_b)} d\theta_b$ is the normalization constant.

Before we move on to discuss the M-step, note that minimizing objective function (7) with respect to $q(\theta_b)$ is closely related to the Bayes theorem: $p(\theta_b|\mathcal{Y}_b, \theta^k) = \frac{p(\theta_b|\theta^k)p(\mathcal{Y}_b|\theta_b)}{\int p(\theta_b|\theta^k)p(\mathcal{Y}_b|\theta_b)d\theta_b}$ with $p(\theta_b|\theta^k)$ and $p(\mathcal{Y}_b|\theta_b)$ representing the prior distribution density and likelihood function respectively. Zellner (1988) shows that the Bayesian posterior density $p(\theta_b|\mathcal{Y}_b, \theta^k)$ can equivalently be found by solving the following minimization problem:

$$\min_{q(\theta_b)} \ \mathbb{D}[q(\theta_b)||p(\theta_b|\theta^k)] + \mathbb{E}_{q(\theta_b)}[-\log p(\mathcal{Y}_b|\theta_b)] \tag{9}$$

By comparing (7) against (9), we see that if $p(\mathcal{Y}_b|\theta_b) \propto e^{-l(\mathcal{Y}_b;\theta_b)}$, *i.e.,* when $l(\mathcal{Y}_b;\theta_b)$ can be interpreted as a negative log-likelihood function, then $q^{k+1}(\theta_b)$ in (8) 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 E-step in our gEM method reduces to the E-step in the classic EM algorithms. 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 doesn't have a probabilistic counterpart, *e.g.,* the hinge loss function for max-margin methods. As a result, in the following we will call the proposed algorithm gEM, as it can be applied to a broader class of models.

Now given $q^{k+1}(\theta_b)$, the M-step consists of updating θ^{k+1} by solving the following problem

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

In sum, the proposed gEM algorithm iterates over the following two steps, (i) each computing node estimates distribution $q^{k+1}(\theta_b)$ by (8) independently given the global parameter θ^k, and (ii) the information about distribution $q^{k+1}(\theta_b)$ are gathered to estimate the global parameter by solving (10), and then scatters the updated θ^{k+1} to each computing node to refine their estimate on $q^{k+2}(\theta_b)$.

Related framework of (6) can be found in the literature known as the minimum relative entropy method (Jaakkola et al., 1999; Zhu et al., 2012) for discriminative learning tasks, or the posterior regularization method (Ganchev et al., 2010) for incorporating constraint to posterior inference and low rank multi-task learning (Koyejo & Ghosh, 2013).

3.3. Small-sample bias problem

As discussed in the beginning of Section 3, in the large-scale setting $\mathcal{Y}$ is partitioned into blocks $\{\mathcal{Y}_1, \ldots, \mathcal{Y}_B\}$, and consequently the bias that is absent in $\mathcal{Y}$ may present in some of the blocks $\mathcal{Y}_b$, which may lead to biased local estimator. This so-called small-sample bias problem is studied in (Zhang et al., 2012; Scott et al., 2013), which has shown that the small-sample bias is a critical issue that affects the performance of distributed learning algorithms especially with the increasing number of partitions B. Such problem is addressed with the bootstrap/jackknife bias correction method in the AVGM setting as a post-processing step (Zhang et al., 2012; Scott et al., 2013), where the local estimate needs to be shifted to avoid an accumulation of small sample biases in the global estimate. And similar idea can be straightforwardly applied to the MEM method discussed in Section 3.1.

In this section we study the small-sample bias problem by explicitly model it as an integrated part of the HM method discussed in Section 3.2. As we will show later that such consideration will empirically reduce the number of iterations needed to reach a (local) optimal solution, which is important for distributed learning algorithms as less number of iterations means less communications between the computing nodes. The basic idea of our method is to constrain the mean of local distribution $q(\theta_b)$ to be consistent with each other, while only allowing the curvature to vary across different blocks. Mathematically this idea can be formulated as follows

$$\begin{aligned} \min_{\theta, q(\cdot)} \quad & \textstyle\sum_{b=1}^{B} f(q(\theta_b), \theta) + h(\theta) \\ \text{subject to} \quad & \mathbb{E}_{q(\theta_b)}[\theta_b] = \theta, \ b = 1, \ldots, B \end{aligned} \tag{11}$$

where $f(q(\theta_b), \theta)$ is the local objective function defined same as in (7). We name this method pADMM, which stands for probabilistic Alternating Direction Method of Multipliers, as we will show later that (11) is closely related to the Alternating Direction Method of Multipliers (ADMM), which is gathering much attentions in a variety of applications in recent years, and a comprehensive survey on ADMM, especially its application on distributed learning, can be found in (Boyd et al., 2010). In order to make

the link of (11) and ADMM more transparent, in the following we will focus on discussing a concrete case

$$p(\theta_b|\theta) = \mathcal{N}(\theta_b|\theta, \gamma_b^{-1}\mathbf{I}) \tag{12}$$

such that the prior distribution is Gaussian with mean θ and isotropic covariance matrix $\gamma_b^{-1}\mathbf{I}$, where $\mathbf{I}$ is the identity matrix of size $P \times P$ with P denotes the dimension of parameter θ and θ_b. In the following γ_b will be referred to as precision parameter, which intuitively models the curvature of the prior distribution. Note that we also could model the precision parameter as a global parameter, *e.g.,* $\gamma_b = \gamma, \forall b$ as well, however, we wish to allow each block to has the freedom to express it's own characteristic by having different precision parameters, and empirically we found such block-specific precision parameters will lead to faster convergence of the algorithm, and thus less communications between computing nodes.

In the next we apply the gEM algorithm discussed in Section 3.2 to pADMM. Based on the Gaussian assumption in (12), the local objective function $f(q(\theta_b), \theta)$ in (11) becomes (with constants independent of $q(\theta_b)$ and θ ignored)

$$f(q(\theta_b), \theta) = \mathbb{E}_{q(\theta_b)}\left[\frac{\gamma_b}{2}||\theta_b - \theta||_2^2 + l(\mathcal{Y}_b; \theta_b)\right] \tag{13}$$

In the E-step, given the global parameter estimator $\hat{\theta}$, we have the Lagrangian for each local optimization problem as follows

$$\min_{q(\theta_b), \lambda_b} \; f(q(\theta_b), \theta) + \lambda_b \cdot (\mathbb{E}_{q(\theta_b)}[\theta_b] - \theta) \tag{14}$$

Where $f(q(\theta_b), \theta)$ is defined in (13) and λ_b is the Lagrange dual variable and $\cdot$ represents the dot product. The dual of (14) can be readily derived, and we propose to solve problem (14) via dual ascent, *e.g.,* the dual problem is solved using gradient ascent. The updating equation of $q(\theta_b)$ and λ_b has the following form

$$q^{k+1}(\theta_b) = \frac{1}{Z} e^{-l(\mathcal{Y}_b;\theta_b) - \frac{\gamma_b}{2}||\theta_b - \theta^k + \mu_b^k||_2^2} \tag{15}$$

where $Z = \int e^{-l(\mathcal{Y}_b;\theta_b) - \frac{\gamma_b}{2}||\theta_b - \theta^k + \mu_b^k||_2^2} d\theta_b$ is the normalization constant, and $\mu_b^k = \frac{1}{\gamma_b}\lambda_b^k$ is the scaled dual variable, which intuitively models the bias/adjustment of the local estimates, and pADMM reduces to HM by fixing $\lambda_b^k = 0$, *i.e.,* no modelling of the small-sample bias. The dual variable λ_b is updated using gradient ascent with step size γ_b (the motivation to choose γ_b as step size is inspired by ADMM as we will discuss later):

$$\lambda_b^{k+1} = \lambda_b^k + \gamma_b(\mathbb{E}_{q^{k+1}(\theta_b)}[\theta_b] - \theta^k) \tag{16}$$

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

$$\min_{\theta} \; \textstyle\sum_{b=1}^B \frac{\gamma_b}{2}||\theta - \mathbb{E}_{q^{k+1}(\theta_b)}[\theta_b] + \mu_b^{k+1}||_2^2 + h(\theta) \tag{17}$$

which can be solved by proximal gradient methods (Parikh & Boyd, 2013). As one example, when $h(\theta) = \beta(\alpha||\theta||_1 + (1-\alpha)||\theta||_2^2)$ corresponds the elastic-net regularizer with parameter $0 \leq \alpha \leq 1$ and $\beta \geq 0$, the updating equation for θ^{k+1} is

$$\theta^{k+1} = \frac{S\left(\sum_{b=1}^B \gamma_b \mathbb{E}_{q^{k+1}}[\theta_b], \beta\alpha\right)}{\sum_{b=1}^B \gamma_b + \beta(1-\alpha)} \tag{18}$$

Where $S(a, b) = \text{sign}(a)(|a| - b)_+$ is the soft thresholding operator (Friedman et al., 2010). Note that (18) 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 precision parameter γ_b. We also propose to update γ_b in the M-step by minimizing (11) with respect to γ_b, which leads to the following updating equation for the reciprocal of γ_b^{k+1}:

$$1/\gamma_b^{k+1} = \frac{1}{P}\mathbb{E}_{q^{k+1}(\theta_b)}[(\theta_b^{k+1} - \theta^{k+1} + \mu_b^{k+1})^2] \tag{19}$$

Where recall that P is the dimension of θ_b. Note that by setting $\mu_b = 0$, (19) reduces to the empirical Bayes estimate of γ_b for HM under the Gaussian assumption (12). As a side note, for the HM method if $\{\gamma_b^k\}$ is setting to be an increasing sequence, *e.g.,* $\gamma_b^1 < \gamma_b^2 < \ldots$ and letting γ_b^k converges to ∞, while constraining $q(\theta_b)$ to be a Dirac delta function, HM reduces to a class of the penalty methods (Bertsekas, 1989; Bertsekas & Tsitsiklis, 1996), which can also be used to solve distributed learning problems, although generally with slower convergence rate than ADMM (Bertsekas, 1989). To better understands pADMM, in the following we briefly review the well-studied and widely-used ADMM method, and demonstrates how pADMM relates to ADMM.

3.3.1. Distributed learning via ADMM

Here we only summarize important aspects of ADMM, a more detailed discussion can be found at (Boyd et al., 2010). First note that we can re-write (1) into the following equivalent optimization problem:

$$\begin{aligned} &\min_{\theta, \theta_b} \; \textstyle\sum_{b=1}^B l(\mathcal{Y}_b, \theta_b) + h(\theta) \\ &\text{subject to } \; \theta_b - \theta = 0, \; b = 1, \ldots, B \end{aligned} \tag{20}$$

The ADMM formulation for the problem (20) can be derived directly from the following augmented Lagrangian

$$L_\gamma(\{\theta_b, \lambda_b\}_{b=1}^B, \theta) = \textstyle\sum_{b=1}^B L_\gamma(\theta_b, \lambda_b, \theta) + h(\theta) \tag{21}$$

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

$$L_\gamma(\theta_b, \lambda_b, \theta) = l(\mathcal{Y}_b, \theta_b) + \lambda_b \cdot (\theta_b - \theta) + \frac{\gamma}{2}||\theta_b - \theta||_2^2 \quad (22)$$

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

$$\theta_b^{k+1} = \arg\min_{\theta_b} \; L_\gamma(\theta_b, \lambda_b^k, \theta^k) \quad (23)$$

$$\lambda_b^{k+1} = \lambda_b^k + \gamma(\theta_b^{k+1} - \theta^k) \quad (24)$$

$$\theta^{k+1} = \arg\min_{\theta} \; \sum_{b=1}^B \frac{\gamma}{2}||\theta_b^{k+1} - \theta + \mu_b^{k+1}||_2^2 + h(\theta) \quad (25)$$

Note that in the gradient ascent update in (24) the step size is chosen to be γ. The motivation behind this choice is explained by Boyd el al. (2010), that by using γ as the step size, the iterate $\theta_b^{k+1}, \lambda_b^{k+1}$ in (23 - 24) 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 (23 - 25) will yield optimal solution $\hat{\theta}$.

Now comparing updating equations for ADMM in (23 - 25) against that of for pADMM (13, 15 - 17), we see that the gEM algorithm for pADMM can be thought as a probabilistic version of ADMM, hence the name, and ADMM can be thought as a way of finding a point estimate for pADMM, with pADMM reduces to ADMM when $q(\theta_b) = \delta_{\hat{\theta}_b}(\theta_b)$ becomes a Dirac delta distribution. This also resembles the relationship between AVGM and MEM. Note that carefully tuning the parameter γ is one caveat of reaching fast convergence of ADMM, which is commonly done by cross-validation, which is particularly computationally expensive in the large-scale setting. However, as mentioned before the fast convergence is also particularly critical in the distributed learning scenario as less number of iterations means less expensive communication among computing nodes when the dimension of the parameter is high. Although the same problem exists for HM and pADMM, we will show empirically that by updating γ_b as in (19), pADMM mitigates the dilemma ADMM faces by having comparable or faster convergence against a well-tuned ADMM.

3.4. Full Bayesian treatment

Although the frameworks we proposed, *i.e.,* MEM, HM and pADMM, are motivated by the optimization problem (1), they can naturally be extended to full Bayesian frameworks. As discussed in Section 3.2 and equation (9), the gEM algorithm proposed for HM and pADMM corresponds to the EM algorithm for maximum likelihood estimation or maximum a posteriori probability estimate when $l(\mathcal{Y}_b; \theta_b)$ corresponds to a negative log-likelihood function. Besides that, if the global parameter θ also treated as a random variable and placed with a prior distribution instead of a regularization function $h(\theta)$ as in (1), such that the M-step is replaced by a Bayesian posterior inference procedure, then we have a full Bayesian model. However, this topic involves the design of efficient inference algorithm for Bayesian posterior computation and is beyond the scope of this work. (Scott et al., 2013; Neiswanger et al., 2013) represent two recent work in this direction, although their frameworks are both in the AVGM based setting.

4. Distributed (sub-)gradient method

We have been focused on the "divide-and-conquer" type of frameworks thus far, *e.g.,* AVGM, MEM, HM, ADMM and pADMM, 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. The distributed sub-gradient method for optimization problem (1) is straightforward, where for each iteration the sub-gradients $\partial l(\mathcal{Y}_b; \theta)$ are computed by in parallel 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. Each iteration of the distributed sub-gradient method can be readily expressed under the MapReduce paradigm (Dean & Ghemawat, 2004), however, one potential problem of the distributed sub-gradient method is that, it requires relatively many iterations to converge will requires constant communications among computing nodes in the cluster, which can be a drawback when the dimensionality of the data and the parameter is high, as communication can be prohibitively expensive. As a result, the AVGM and MEM based method represents one extreme of the distributed learning framework, where communication only happens once but the solution might not be optimal with finite number of data (although asymptotically optimal); and the distributed sub-gradient method is another extreme with frequent communications but is guaranteed to find the (local) optimal solution; at last ADMM and the proposed HM, ADMM methods lies in the middle of the two.

5. Practical Issues on the gEM Algorithm

In this section we first discuss some practical considerations for the proposed gEM algorithm, where one core issue in making the gEM practical in large-scale setting 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 analytical form of the distribution $q(\theta_b)$, or at least be able to draw samples from it. And given the expectations found in the E-step, efficient optimization algorithms, *e.g.,* the prox-

imal gradient method or stochastic gradient based methods, becomes applicable for the M-step. Consequently, the first key technical component in our implementation is the adoption of the variational methods (Wainwright & Jordan, 2008) in the E-step, as we briefly discuss in the following.

As discussed in Section 3.1-3.3, for MEM, HM and pADMM the distribution $q(\theta_b)$ of interest is found by minimizing the local objective function (7), which could be problematic when the expectation $\mathbb{E}_{q(\theta_b)}[l(\mathcal{Y}_b;\theta_b)]$ has no analytical expression. Two types of variational methods, commonly referred to as local and global variational methods in the literature (Bishop, 2006), can be applied here. The local variational methods 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 can be 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 of different loss functions and Bayesian models, a few examples include (Zhu et al., 2012) for hinge loss function, (Jaakkola & Jordan, 2000; Khan et al., 2010) for logistic/soft-max loss function, and (Khan et al., 2013; Wang & Blei, 2013) for general non-conjugate models. The global variational methods restricts the range of $q(\theta)$ over which the optimization is performed, *e.g.,* the mean field methods restrict $q(\theta) = \prod_i q(\theta_i)$ to take factorized forms. Mean field based variational methods can be found in many Bayesian graphical models with latent variables, where the posterior distribution is intractable (Wainwright & Jordan, 2008). Here we propose to use mean field variational methods from a slightly different perspective, as we will discuss later that by restricting $q(\theta)$ to be fully factorized across its components, the variational methods proceeds the form of coordinate descent (CD) algorithms, which is shown to be efficient to find the (local) optimal solution in large-scale applications from the optimization viewpoint (Friedman et al., 2010; Bradley et al., 2011; Yuan et al., 2012; Yu et al., 2013). In both cases, the gEM algorithm reduces to the variational EM algorithm, and if we extend our method to be full Bayesian as discussed in Section 3.4, gEM corresponds to the variational Bayesian inference algorithm (Bishop, 2006; Wainwright & Jordan, 2008).

Although sampling based methods, *e.g.,* MCMC and importance re-sampling, generally doesn'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 method to sampling based method is twofold, (i) It is generally more difficult to monitor the convergence of a sampling based method than deterministic variational method; and (ii) our preliminary experiment suggests 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.

6. Empirical Study

With the variational gEM algorithm described above, we solve two challenging problems, the non-smooth ℓ_1-regularized logistic regression, and the non-convex low-rank matrix completion, with our proposed distributed learning methods. In the following we first introduce our experimental setting.

6.1. Experimental setting

MPI has no fault tolerance and doesn't scale very well to large clusters, Hadoop requires many disk read/write, which may lead to distract our attention on analysing the reason caused the different convergence behaviour of different methods, and Hadoop is too slow in iterative algorithms. (Not finished yet)

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 $x_n \in \mathbb{R}^P$ represents the feature vector with dimension P, and parameter $\theta \in \mathbb{R}^P$ also has the same dimension. 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 (12) for HM and pADMM.

In this case distribution $q(\theta_b)$ doesn't readily have an analytical form, as the normalization constant Z in (5) for MEM, (8) for HM, and (15) for pADMM are all intractable. And in this work we will use the Laplace variational method (Wang & Blei, 2013) and the Bohning bound based variational method (Khan et al., 2010) as they work favourably compared to other variational methods both computationally and efficaciously in our preliminary experiments.

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) (Sre-

Table 1. Comparison of memory and network usage for two types of matrix factorization algorithms

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

Table 2. Statistics and parameters for each dataset

	Netflix	KDDCup2011	Synthetic
M	2,649,429	1,000,990	1,000,000
N	17,770	624,961	1,000,000
$\lvert\Omega\rvert$	99,072,112	252,800,275	1,983,749,510
$\lvert\bar{\Omega}\rvert$	1,408,395	4,003,960	23,328,835
K	30	50	30
λ	0.05	1	0.001

bro 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

$$f(U,V) = \sum_{(i,j)\in\Omega}(y_{i,j} - U_{i\cdot}V'_{j\cdot})^2 + h(U) + h(V) \quad (26)$$

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

$$\begin{aligned} &\min_{U,V,q(\cdot)} \ \sum_{b=1}^{B} f(U_b, V_b, U, V) + h(U) + h(V) \\ &\text{subject to } \mathbb{E}_{q(U_b)} = U_b,\ \mathbb{E}_{q(V_b)} = V_b \end{aligned} \quad (27)$$

References

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.

Dean, J. and Ghemawat, S. Mapreduce: simplified data processing on large clusters. In *OSDI*, 2004.

Friedman, J. H., 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.

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.

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.

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.

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

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