Probabilistic Methods for Distributed Learning

by

XianXing Zhang

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

_____
Lawrence Carin, Supervisor

_____
David B. Dunson

_____
Arthur Calderbank

_____
Guillermo Sapiro

_____
Katherine Heller

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of Duke University
2014

Abstract
(Engineering)

Probabilistic Methods for Distributed Learning

by

XianXing Zhang

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

_____
Lawrence Carin, Supervisor

_____
David B. Dunson

_____
Arthur Calderbank

_____
Guillermo Sapiro

_____
Katherine Heller

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering
in the Graduate School of Duke University
2014

Abstract

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 [15]. To scale this method to big data that are becoming commonplace in various applications, it is desirable to efficiently extend empirical risk minimization to a large-scale setting. 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 dissertation is to study distributed learning algorithms [3] for empirical risk minimization problems.

Toward this end we propose a series of probabilistic methods for divide-and-conquer distributed learning, with these methods accounting for an increasing set of challenges. The basic Maximum Entropy Mixture (MEM) method is first proposed, to model uncertainty caused by randomly partitioning the data across computing nodes. We then develop a hierarchical extension to MEM, termed hMEM, facilitating sharing of statistical strength among data blocks. Finally, to addresses small-sample bias, we impose the constraint that the mean of inferred parameters is the same across all data blocks, yielding a hierarchical MEM with expectation constraint (termed hecMEM). Computations are performed with a generalized Expectation-

Maximization algorithm. The hecMEM method achieves state-of-the-art results for distributed matrix completion and logistic regression at massive scale, with comparisons made to MEM, hMEM and several alternative approaches.

To my family

Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

Symbols

$\mathbb{X}$	denotes a operator, *e.g.,* $\mathbb{E}$ denotes the expectation operator.
$\mathcal{X}$	denotes a distribution, *e.g.,* Gaussian distribution with mean μ and precision γ is denoted as $\mathcal{N}(\mu, \gamma)$.
$\mathbf{X}$	denotes a matrix.
X_{ij}	denotes the (i, j)th entry of matrix $\mathbf{X}$.
$\boldsymbol{x}$	denotes a vector.
$\boldsymbol{x}_i$	denotes the ith entry of vector $\boldsymbol{x}$

Abbreviations

AVGM	denotes the average mixture algorithm [45].
sAVGM	denotes the subsampled average mixture algorithm [45].
ADMM	denotes the alternating direction methods of multipliers [7].
VB	denotes the variational Bayesian inference algorithm.
CG	denotes the conjugate gradient algorithm
L-BFGS	denotes the limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm.
CD	denotes the coordinate descent algorithm
ALS	denotes the alternating least square algorithm for matrix factorization
AUC	denotes the area under curve evaluation metric for classification tasks.

RMSE	denotes the rooted mean square error evaluation metric for matrix completion.
MEM	denotes the maximum entropy mixture algorithm proposed in this dissertation.
hMEM	denotes the hierarchical maximum entropy mixture algorithm proposed in this dissertation.
hecMEM	denotes the hierarchical maximum entropy mixture algorithm with equality constraint proposed in this dissertation.

Acknowledgements

I am thankful to my advisor, Professor Lawrence Carin, who guided me tirelessly throughout my PhD, while giving the maximum degree of freedom to me to choose research topics that interest me. Larry is a brilliant researcher who is always able to push ideas one level further, provide the input needed to finish a task, and share his experience about the research process. I was especially fortunate to work with him and benefit from his perspectives. I am deeply indebted to Larry for his extensive editing of papers that comprise most portion of this thesis, as well as all of my published research papers.

The work in this dissertation is the result of collaboration with many other people. Deepak Agarwal and Bo Long initiated the probabilistic distributed learning project, and provided many helpful insights and discussion throughout the whole process. In addition to my interactions with Deepak and Bo, the other collaborators from LinkedIn and Yahoo! Labs, both past and present, have played a pivotal role in my graduate studies. In particular, I would like to thank Bee-Chung Chen, Liang Zhang, Ajit Singh, Romer Rosales, Yi Chang, Anlei Dong, Nathan Liu, Roger Jie Luo, who were always there when I need help.

I have had the honour of working with Professor David Dunson from Department of Statistical Science, who was also my Master thesis's advisor and has provided me with invaluable guidance on my work in Bayesian nonparametrics and insight into the fields of Statistics. Other faculty members from Duke University, Arthur Calderbank,

Guillermo Sapiro, Sayan Mukherjee, Katherine Heller, Barbara Engelhardt, among others, provided me indispensable help through a massive number of emails and courses they offer for graduate students.

I am also extremely lucky to work with a group of talented young researchers in Larry's research lab, both past and present. Xuejun Liao, Lihan He, Chunping Wang, Chenghui Cai, Lu Ren, Haojun Chen, John Paisley, Eric Wang, Minhua Chen, Bo Chen and Mingyuan Zhou, who gave me all-around support during my early PhD studies, and my long-time officemates and friends- Lingbo Li, Zhengming Xing, David Carlson, Kyle Ulrich, Miao Liu, Yingjian Wang, Wenzhao Lian- provided stimulating conversations and tolerated my incessant interruptions.

On a more personal note, I would like to thank my family for their support during my five-year adventure tens of thousands away from home. Finally, this endeavour would have been infinitely more challenging without the love and support of Lihong Mo.

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 [15]. To scale this method to big data that are becoming commonplace in various applications, it is desirable to efficiently extend empirical risk minimization to a distributed-computing setting. We develop a series of increasingly more sophisticated probabilistic algorithms to address this challenge. 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 [3] 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 parameters. Such approaches have been studied for a number of problems within a distributed setting, including the Bootstrap [21], matrix factorization [25], logistic regression [26], kernel ridge regression [46], and general smooth

convex optimization problems [45]. 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, number of parameters, etc. [45, 46]. It is therefore of interest to estimate the uncertainty 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 [45, 33] in the context of distributed learning. For scenarios where the number of parameters increase with observations (*e.g.,* , matrix factorization), the impact of such bias is significant and must be addressed for statistical efficiency. Non-convex learning also presents another big challenge for divide-and-conquer approaches; since parameter estimation from different blocks could be at different local optimal points, it is difficult to infer a global optimal estimation from them.

We propose a series of probabilistic methods that model the uncertainty caused by the random partitions, and mitigate the small sample size bias by sharing 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 algorithms are developed in this paper, summarized here for reader convenience. In Section 1.2 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 2.1 we explicitly infer node-dependent *distributions* on the model parameters, with AVGM generalized 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 2.2 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 2.9 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 hecMEM. We show that hecMEM 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 hecMEM 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.

1

Preliminaries

1.1 Problem formulation

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; $h(\theta)$ denotes the regularization function; and $\mathcal{Y} = \{y_n, n \in \Omega\}$ denotes the observed data 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 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

$$\min_\theta f(\theta) = \sum_{b=1}^{B} l(\mathcal{Y}_b;\theta) + h(\theta) \tag{1.1}$$

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 (1.1), and $\hat{\theta} = \arg\min_\theta f(\theta)$ will be used to represent the optimal solution.

1.2 Average mixture based algorithms

One of the well studied distributed learning algorithms for (1.1) is the average mixture (AVGM) method [26, 50, 45]. The basic idea of AVGM is to divide the data into blocks as stated in Section 1.1, 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

$$\begin{aligned} \hat{\theta}_b &= \arg\min_{\theta_b} f(\theta_b), \\ \hat{\theta} &= \frac{\sum_{b=1}^{B} w_b \hat{\theta}_b}{\sum_{b=1}^{B} w_b} \end{aligned} \tag{1.2}$$

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

$$f(\theta_b) = l(\mathcal{Y}_b;\theta_b) + \frac{h(\theta_b)}{B} \tag{1.3}$$

and w_b is a weight parameter, commonly set to $1/B$ [26, 50] or proportional to $|\mathcal{Y}_n|$ in [45].

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 [45], statistical optimality is retained for AVGM given enough information per block, *e.g.,* when $|\mathcal{Y}_b|$ is not too small. However, high dimensional 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 [45, 33].

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.

2

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 general 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.

2.1 MEM: 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 [2].

Under the maximum entropy principle [2], the problem becomes finding a distribution $q(\theta_b)$ such that (i) we minimize the convex combination of local objective functions (1.3) of block b with respect to $q(\theta_b)$, denoted as $\mathbb{E}_{q(\theta_b)}[f(\theta_b)] = \int f(\theta_b)q(\theta_b)d\theta_b$; and (ii) at the same time we maximize the entropy of $q(\theta_b)$, denoted $\mathbb{H}[q(\theta_b)] =$

$-\int q(\theta_b)\log q(\theta_b)d\theta$. The problem is formulated as

$$\hat{q}(\theta_b) = \arg\min_{q(\theta_b)} -\mathbb{H}[q(\theta_b)] + \mathbb{E}_{q(\theta_b)}[f(\theta_b)] \tag{2.1}$$

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:

$$\hat{\theta} = \frac{\sum_{b=1}^{B} w_b \mathbb{E}_{\hat{q}(\theta_b)}[\theta_b]}{\sum_{b=1}^{B} w_b} \tag{2.2}$$

In the following we refer to (2.1) and (2.2) as the Maximum Entropy Mixture (MEM) method.

Concerning (2.1), 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 [21]. 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 (2.2).

After some simple algebra, the optimal solution for (2.1) has the form

$$\hat{q}(\theta_b) = \frac{1}{Z} e^{-f(\theta_b)} \tag{2.3}$$

where $Z = \int e^{-f(\theta_b)} d\theta_b$ is the normalization constant. Note that AVGM actually finds the maximum likelihood estimate of (2.3) with respect to θ_b, which suggests that AVGM is a special case of MEM.

2.2 hMEM: Improved sharing of statistical strength

Both MEM and AVGM only use data from their own block, *i.e.,* θ_b only depends on $\mathcal{Y}_b$. Consequently, there is no sharing of information across the multiple blocks, until one performs the final averaging across blocks. We therefore propose a hierarchical extension (called hMEM), that maintains measures of uncertainty on any given node, while also sharing information from the data itself across nodes.

A common feature of AVGM and MEM is that the local estimators only use data from the their own block, *i.e.,* θ_b only depends on $\mathcal{Y}_b$. While MEM seeks to model the uncertainty in the block-dependent estimates, there is no sharing of information across the multiple blocks, until one performs the final averaging across blocks. We therefore propose a hierarchical extension (called hMEM), that maintains measures of uncertainty on any given node, while also sharing information from the data itself across nodes (improved sharing of statistical strength).

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 2, 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, hMEM models θ_b as a latent random variable, with interest in finding the distribution $q(\theta_b)$. hMEM places the 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 given θ. To incorporate the prior and hierarchical information into the empirical risk minimiza-

tion 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 hMEM representation is

$$\min_{\theta, q(\cdot)} \sum_{b=1}^{B} f(q(\theta_b), \theta) + h(\theta) \tag{2.4}$$

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

$$f(q(\theta_b), \theta) = \mathbb{D}[q(\theta_b)||p(\theta_b|\theta)] + \mathbb{E}_{q(\theta_b)}[l(\mathcal{Y}_b; \theta_b)] \tag{2.5}$$

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 hMEM formulation naturally leads to a generalized Expectation-Maximization (gEM) algorithm[1] to estimate θ and $q(\theta_b)$.

The gEM algorithm proceeds by iteratively applying two steps: the E-step finds a distribution $q(\theta_b)$ that minimizes objective function (2.4) while fixing θ, and evaluat-

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

ing the expectation and KL-divergence in (2.5) with respect to $q(\theta_b)$; and the M-step finds an estimate of θ that minimizes (2.4) 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 has the following optimal solution:

$$q^{k+1}(\theta_b) = \frac{1}{Z} p(\theta_b|\theta^k) e^{-l(\mathcal{Y}_b;\theta_b)} \tag{2.6}$$

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

$$\min_{\theta} \sum_{b=1}^{B} \mathbb{E}_{q^{k+1}(\theta_b)}[-\log p(\theta_b|\theta)] + h(\theta) \tag{2.7}$$

Note that minimizing (2.5) 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 [44] shows that the Bayesian posterior density $p(\theta_b|\mathcal{Y}_b, \theta)$ can equivalently be found by solving the following minimization problem:

$$\min_{q(\theta_b)} \mathbb{D}[q(\theta_b)||p(\theta_b|\theta)] + \mathbb{E}_{q(\theta_b)}[-\log p(\mathcal{Y}_b|\theta_b)] \tag{2.8}$$

By comparing (2.5) with (2.8), 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 (2.6) 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 do not have a probabilistic counterpart, *e.g.,* the hinge loss function for max-margin methods.

Algorithm 1 hecMEM

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 $q^{k+1}(\theta_b)$ based on (2.13).
    Update λ_b^{k+1} based on (2.14).
    Update γ_b^{k+1} based on (2.17).
  end for
  M-step:
  Update θ^{k+1} by solving (2.15) and ρ^{k+1} based on (2.17)
end for

Related frameworks to (2.4) and (2.5) are found in the literature for different applications. Some are known as the minimum relative entropy method [17, 48] for max-margin based discriminative learning tasks, or the posterior regularization method [13] and constrained Bayesian inference [22] for incorporating constraints into posterior inference.

2.3 hecMEM: Accounting for the local bias

As discussed at the beginning of Section 2, 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 [45, 33]. Similar ideas may be applied to the MEM method.

To further mitigate the local bias that may be manifested within hMEM, we 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

$$\begin{aligned} \min_{\theta, q(\cdot)} \quad & \textstyle\sum_{b=1}^{B} f(q(\theta_b), \theta) + h(\theta) \\ \text{s.t.} \quad & \mathbb{E}_{q(\theta_b)}[\theta_b] = \theta, \ b = 1, \ldots, B \end{aligned} \tag{2.9}$$

where $f(q(\theta_b), \theta)$ is the local objective function defined in (2.5). This leads to the principal algorithmic contribution of this paper, a hierarchical expectation-constrained maximum entropy method, denoted hecMEM. We observe that MEM, hMEM and hecMEM are different forms of a class of probabilistic distributed computing methods. MEM is the basic algorithm, in which data are not shared across the B blocks; hMEM introduces a hierarchical extension, which allows data sharing; and finally hecMEM further constrains the hMEM, imposing a constraint of consistency in the expected value of parameters across the B blocks.

For the special case of Gaussian priors, we show below that the hecMEM construction yields a *probabilistic* form of the widely studied Alternating Direction Method of Multipliers (ADMM) [7]. Assuming $p(\theta_b|\theta)$ is Gaussian:

$$p(\theta_b|\theta) = \mathcal{N}(\theta_b|\theta, (\rho\gamma_b)^{-1}\mathbf{I}) \tag{2.10}$$

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.

We apply the gEM algorithm discussed in Section 2.2 to hecMEM, for the aforementioned Gaussian prior. With (2.10), the local objective function $f(q(\theta_b), \theta)$ becomes (with constants independent of $q(\theta_b)$ and θ ignored):

$$f(q(\theta_b), \theta) = \mathbb{E}_{q(\theta_b)}\left[\frac{\rho\gamma_b}{2}||\theta_b - \theta||_2^2 + l(\mathcal{Y}_b; \theta_b)\right] \tag{2.11}$$

We have the following Lagrangian for each local optimization problem

$$\min_{q(\theta_b),\lambda_b} \; f(q(\theta_b),\theta) + \lambda_b \cdot (\mathbb{E}_{q(\theta_b)}[\theta_b] - \theta) \tag{2.12}$$

where $f(q(\theta_b),\theta)$ is defined in (2.11), λ_b is the Lagrange dual variable, and $\cdot$ represents the dot product. The dual of (2.12) is readily derived, and in the E-step we propose to solve problem (2.12) 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:

$$q^{k+1}(\theta_b) = \frac{1}{Z} e^{-l(\mathcal{Y}_b;\theta_b) - \frac{\rho\gamma_b}{2}||\theta_b - \theta^k + \mu_b^k||_2^2} \tag{2.13}$$

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 [45], and hecMEM reduces to hMEM by fixing $\lambda_b^k = 0$. Inspired by ADMM [7], the dual variable λ_b is updated using gradient ascent with step size $\rho\gamma_b$:

$$\lambda_b^{k+1} = \lambda_b^k + \rho\gamma_b(\mathbb{E}_{q^{k+1}(\theta_b)}[\theta_b] - \theta^k) \tag{2.14}$$

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:

$$\min_{\theta} \; \sum_{b=1}^{B} \frac{\rho\gamma_b}{2} ||\theta - \mathbb{E}_{q^{k+1}(\theta_b)}[\theta_b] + \mu_b^{k+1}||_2^2 + h(\theta) \tag{2.15}$$

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 (2.15) is generally solved efficiently by proximal gradient methods [29]. 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 $0 \leq \alpha \leq 1$ and $\beta \geq 0$, the update for θ^{k+1} is

$$\theta^{k+1} = \frac{S(\sum_{b=1}^{B} \rho\gamma_b \mathbb{E}_{q^{k+1}}[\theta_b], \beta\alpha)}{\sum_{b=1}^{B} \rho\gamma_b + \beta(1-\alpha)} \tag{2.16}$$

where $S(a,b) = \text{sign}(a)(|a| - b)_+$ is the soft thresholding operator [12]. Note that (2.16) resembles the weighted average strategy in (1.2), 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 $\rho\gamma_b$. Next we propose to update ρ and γ_b in the M-step, done by minimizing (2.9) 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

$$\begin{aligned} \rho^{k+1} &= \frac{BP}{\sum_{b=1}^{B} \gamma_b^k r^{k+1}} \\ \gamma_b^{k+1} &= \frac{P}{\rho^{k+1} r^{k+1}} \end{aligned} \tag{2.17}$$

where P denotes the dimension of θ_b. The hecMEM method is summarized in Algorithm 1. Note that by setting $\mu_b = 0$, hecMEM reduces to hMEM. Also note that, under the Gaussian assumption (2.10) for the hMEM 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 [4] when only a point esimate of θ_b is of interest.

2.4 Full Bayesian treatment

Although the above proposed methods, *i.e.,* MEM, hMEM and hecMEM, are motivated by the optimization problem (1.1), 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 (2.8), the gEM algorithms proposed for hMEM and hecMEM 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 (1.1) is replaced by a prior distribution, posterior inference is performed on θ, and then we have a fully Bayesian model. [33, 27] represent two recent works in this direc-

tion, although their frameworks are both based on the AVGM setting. The proposed hecMEM 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.

3

Practical Issues of the gEM Algorithm

One important issue in making gEM scale 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 [36] 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 2.4, gEM becomes the variational Bayesian inference algorithm [5].

As discussed in Sections 2.1-2.9, for MEM, hMEM and hecMEM the distribution $q(\theta_b)$ of interest is found by minimizing the local objective function (2.5), 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 [5]. 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 methods are well studied for various types of loss functions and likelihood functions, a few examples include [48] for hinge loss function, [20] for the logistic/soft-max loss function, and [19, 37] 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 [36]. Mean field based variational methods are found in many Bayesian hierarchical models with latent variables and intractable posterior distributions [36].

By restricting $q(\theta_b) = \prod_i q(\theta_{bi})$ to be fully factorized across its components, the variational method 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 [12, 8, 42, 40].

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.

4

Related Distributed Learning Methods

4.1 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 [7]. Here we summarize key aspects of Alternating Direction Method of Multipliers (ADMM) [7], to highlight connections to hecMEM with a Gaussian prior.

Note that we can rewrite (1.1) into the following equivalent problem:

$$\begin{aligned} &\min_{\theta,\theta_b} \ \sum_{b=1}^{B} l(\mathcal{Y}_b, \theta_b) + h(\theta) \\ &\text{s.t. } \theta_b - \theta = 0, \ b = 1, \ldots, B \end{aligned} \tag{4.1}$$

The Alternating Direction Method of Multipliers (ADMM) [7] formulation for the problem (4.1) is derived directly from the following augmented Lagrangian

$$L_\rho(\{\theta_b, \lambda_b\}_{b=1}^{B}, \theta) = \sum_{b=1}^{B} L_\rho(\theta_b, \lambda_b, \theta) + h(\theta) \tag{4.2}$$

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

$$L_\rho(\theta_b, \lambda_b, \theta) = l(\mathcal{Y}_b, \theta_b) + \lambda_b \cdot (\theta_b - \theta) + \frac{\rho}{2}||\theta_b - \theta||_2^2 \tag{4.3}$$

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:

$$\theta_b^{k+1} = \arg\min_{\theta_b} \; L_\gamma(\theta_b, \lambda_b^k, \theta^k) \tag{4.4}$$

$$\lambda_b^{k+1} = \lambda_b^k + \rho(\theta_b^{k+1} - \theta^k) \tag{4.5}$$

$$\theta^{k+1} = \arg\min_{\theta} \; \sum_{b=1}^{B} \frac{\rho}{2}||\theta_b^{k+1} - \theta + \mu_b^{k+1}||_2^2 + h(\theta) \tag{4.6}$$

Note that in the gradient ascent update in (4.5), the step size is chosen to be ρ. The motivation behind this choice is explained by Boyd *el al.* [7], that by using ρ as the step size, the iterate $\theta_b^{k+1}, \lambda_b^{k+1}$ in (4.4 - 4.5) 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 (4.4) - (4.6) will yield optimal solution $\hat{\theta}$.

Now comparing the update equations for ADMM in (4.4) - (4.6) with that of hecMEM (2.11), (2.13) - (2.15), we see that the gEM algorithm for hecMEM may be thought of as a probabilistic version of ADMM. On the other hand, ADMM may be thought of as a way of finding a point estimate for hecMEM, which again resembles the connections between AVGM and MEM, and hMEM and penalty methods. Note that carefully tuning the parameter ρ is important for 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 hecMEM, by contrast, ρ is updated automatically, as in (2.17). In hecMEM the block-specific precision parameter γ_b is modelled explicitly and also updated in (2.17), which leads to faster convergence as will be showed empirically through experiments. Note that for ADMM modelling the block-specific parameters will unfortunately increase the number of parameters to tune.

4.2 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 (1.1), 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-gradient computation. The distributed sub-gradient method is guaranteed to find the (local) optimal solution, however, it requires relatively many iterations before convergence, which requires frequent communication 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.

5

Two examples

5.1 Distributed ℓ_1-regularized logistic regression

In logistic regression 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 (1.1) is

$$l(\mathbb{Y}_b; \theta) = \sum_{n \in \Omega_b} \log(1 + \exp(-l_n(x_n \cdot \theta))), \tag{5.1}$$

and the regularization function is

$$h(\theta) = \mu ||\theta||_1, \tag{5.2}$$

where μ is the regularization parameter, which is fixed to $\mu = 1$ through all the experiment for simplicity. We assume the prior distribution $p(\theta_b|\theta)$ is Gaussian as in (2.10) for hMEM and hecMEM.

Note that the main problem of applying the gEM algorithm here is $q(\theta_b)$ not analytic, as the normalization constant Z in (2.3) for MEM, (2.6) for hMEM, and (2.13) for hecMEM are all intractable. To address this problem, we propose to apply

the Laplace variational method [37] to approximate $l(\mathbb{Y}_b;\theta)$ into a quadratic function, based on which $q(\theta_b)$ is a Gaussian distribution. However, when the dimension of θ_b is high, inverting the covariance matrix of $q(\theta_b)$ becomes computationally prohibitive. To make gEM scalable, we further apply the mean-field variational method and restrict $q(\theta)$ to take a fully factorized form, such that $q(\theta_b) = \prod_{j=1}^{P} q(\theta_{bj})$ with P denoting the number of features. As a result, the E-step proceeds by updating each scalar based density $q(\theta_{bj})$ in turn, avoiding any matrix related computations.

5.2 Distributed low-rank matrix completion

In our notation, $\mathcal{Y} = \{y_{mn}, (m,n) \in \Omega\}$ corresponds to the observed entries from a matrix of dimension $M \times N$, and Ω is a subset of $\{1,\ldots,M\} \otimes \{1,\ldots,N\}$ denoting the indexes of the observed entries, where $\otimes$ is the Cartesian product. Likewise, $\theta \in \mathbb{R}^{M\times N}$ denotes the model parameters of interest. We consider the quadratic loss function $l(\mathcal{Y};\theta) = \sum_{(m,n)\in\Omega}(y_{mn} - \theta_{mn})^2$, and the nuclear norm regularization function $h(\theta)$ [30]. Following recent work [32, 31, 40], we assume θ has rank at most K, consequently θ can be explicitly written as UV' where $U \in \mathbb{R}^{M\times K}$ and $V \in \mathbb{R}^{N\times K}$ are row and column factor matrices; V' denotes the transpose of V. This approximation transforms low-rank matrix completion to a non-convex problem, with objective function

$$f(U,V) = \sum_{(m,n)\in\Omega} (y_{mn} - u_m v_n')^2 + h(U,V) \tag{5.3}$$

where $u_m \in \mathbb{R}^{1\times K}$ denotes the m^{th} row of U, $h(U,V) = \mu(||U||_F^2 + ||V||_F^2)$, μ is the regularization parameter and $||U||_F^2 = \sum_m \sum_k u_{mk}^2$ is the Frobenius norm.

In the large-scale setting, we randomly partition the observed entries $\mathcal{Y}$ and its index set Ω into B blocks. This is done by first randomly [31] partitioning the row index into B^r sets, with each set denoted as Ω_i^r. Column index sets Ω_j^c, $j = 1 \ldots B^c$,

are formed likewise. Given Ω_i^r and Ω_j^c, Ω is conformably partitioned into $B = B^r B^c$ blocks, such that $\Omega_{ij} = \Omega_i^r \otimes \Omega_j^c$ represents the index set of the entries belong to block (i, j), where each block is now indexed by a pair of integers.

For each block (i, j), the augmented row latent random variables are denoted as $\mathcal{U}_{ij} = \{u_{ijm} : m \in \Omega_i^r\}$, where $u_{ijm} \in \mathbb{R}^{1\times K}$ is dependent on global parameter u_m *a priori*. The prior distribution is written as $p(\mathcal{U}_{ij}|U(\Omega_i^r)) = \prod_{m\in\Omega_i^r} p(u_{ijm}|u_m)$, where $U(\Omega_i^r) = \{u_m : m \in \Omega_i^r\}$ and $p(u_{ijm}|u_m) = \mathcal{N}(u_m, (\rho\gamma_{ij})^{-1}\mathbf{I})$ is assumed to be Gaussian as in (2.10). Column latent random variables $\mathcal{V}_{ij} = \{v_{ijn} : n \in \Omega_j^c\}$ and their prior distribution $p(\mathcal{V}_{ij}|V(\Omega_j^c))$ are defined likewise.

In the proposed probabilistic methods the distribution $q(\mathcal{U}_{ij}, \mathcal{V}_{ij})$ is of interest. The hecMEM formulation of the low-rank matrix completion problem is

$$\begin{aligned} \min_{U,V,q(\cdot)} \ & \sum_{i=1}^{B^r}\sum_{j=1}^{B^c} f(q(\mathcal{U}_{ij}, \mathcal{V}_{ij}), U, V) + h(U, V) \\ \text{s.t. } & \mathbb{E}_{q(\mathcal{U}_{ij},\mathcal{V}_{ij})}[\mathcal{U}_{ij}] = U(\Omega_i^r),\ 1 \le i \le B^r \\ & \mathbb{E}_{q(\mathcal{U}_{ij},\mathcal{V}_{ij})}[\mathcal{V}_{ij}] = V(\Omega_j^c),\ 1 \le j \le B^c \end{aligned} \tag{5.4}$$

where the local objective function $f(q(\mathcal{U}_{ij}, \mathcal{V}_{ij}), U, V)$ is

$$\begin{aligned} f(q(\mathcal{U}_{ij}, \mathcal{V}_{ij}), U, V) &= \mathbb{E}_{q(\mathcal{U}_{ij},\mathcal{V}_{ij})}[l(\mathcal{Y}_{ij}, \mathcal{U}_{ij}, \mathcal{V}_{ij})] \\ &+\mathbb{D}[q(\mathcal{U}_{ij}, \mathcal{V}_{ij})||p(\mathcal{U}_{ij}|U(\Omega_i^r))p(\mathcal{V}_{ij}|V(\Omega_j^c))] \end{aligned} \tag{5.5}$$

where $l(\mathcal{Y}_{ij}, \mathcal{U}_{ij}, \mathcal{V}_{ij}) = \sum_{(m,n)\in\Omega_{ij}} (y_{mn} - u_{ijm}v'_{ijn})^2$ is the local loss function.

The expectation in (5.5) is not analytic, and to address this problem we follow [24] and make the mean-field assumption, restricting $q(\mathcal{U}_{ij}, \mathcal{V}_{ij}) = q(\mathcal{U}_{ij})q(\mathcal{V}_{ij})$. However, the variational Bayesian inference algorithm in [24] involves computationally expensive matrix inversions in each iteration. Here we further assume $q(\mathcal{U}_{ij})$ and

$q(\mathcal{V}_{ij})$ to be fully factorized:

$$
\begin{aligned}
q(\mathcal{U}_{ij}) &= \prod_{m \in \Omega_i^r} \prod_{k=1}^{K} q(u_{ijmk}) \\
q(\mathcal{V}_{ij}) &= \prod_{n \in \Omega_j^c} \prod_{k=1}^{K} q(v_{ijnk})
\end{aligned}
\tag{5.6}
$$

Note that as discussed in Section 3, (5.6) may lead to less accurate approximation of $q(\mathcal{U}_{ij}, \mathcal{V}_{ij})$, however, with (5.6) all computations in the gEM algorithm now involve only scalars, which is efficient and scalable.

Note that although we have focused on (5.3), an alternative formulation can also be applied to the hecMEM formulation in (5.4) - (5.5), *e.g.,* $l(\mathcal{Y}_{ij}, \mathcal{U}_i, \mathcal{V}_j)$ could be the hinge loss function [35, 38, 39], or $h(U, V)$ could be the max-norm regularization [35, 23].

6

Empirical Study

We solve two challenging problems with the proposed methods, utilizing the variational gEM algorithm described above: non-smooth ℓ_1-regularized logistic regression, and non-convex low-rank matrix completion.

6.1 Experimental setting

The distributed statistical optimization experimental environment is setup by the Spark cluster computing framework of version 0.8.1[1]. 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 hundreds of computing nodes. With Spark we conduct experiments on both multi-core and distributed environments.

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 m2.xlarge

[1] `http://spark.incubator.apache.org`

Table 6.1: Statistics for each classification dataset

	P	$\|\Omega\|$	$\|\Omega^{\text{Test}}\|$	size
KDDCup2010	7,272,193	19,264,097	748,401	5GB
Ads Click	100,000	170,000,000	40,000,000	100 GB

instance with 4 virtual CPUs and 17 Gigabytes memory [2]. For industrially large datasets, 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, and the source code used to conduct the experiments below will be made publicly available.

6.2 Datasets for sparse logistic regression

We consider two real-world datasets for experiment. The first dataset was originally used for the KDDCup2010 competition, and the version we use is prepared by the winner team [3] [41]. After filtering out infrequent features that occur less than 5 times across the training examples, the dataset contains 7,272,193 features and 19,264,097 training samples. The KDDCup2010 dataset is sparse, as on average each sample only uses 29 out of 7,272,193 features. The second dataset consists of 60 days of advertising event logs sampled for a major social network site. Both the training and test sets consist of about 170 and 40 million 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 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. Some important statistics for each dataset are summarized in Table 6.1.

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

[3] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets`

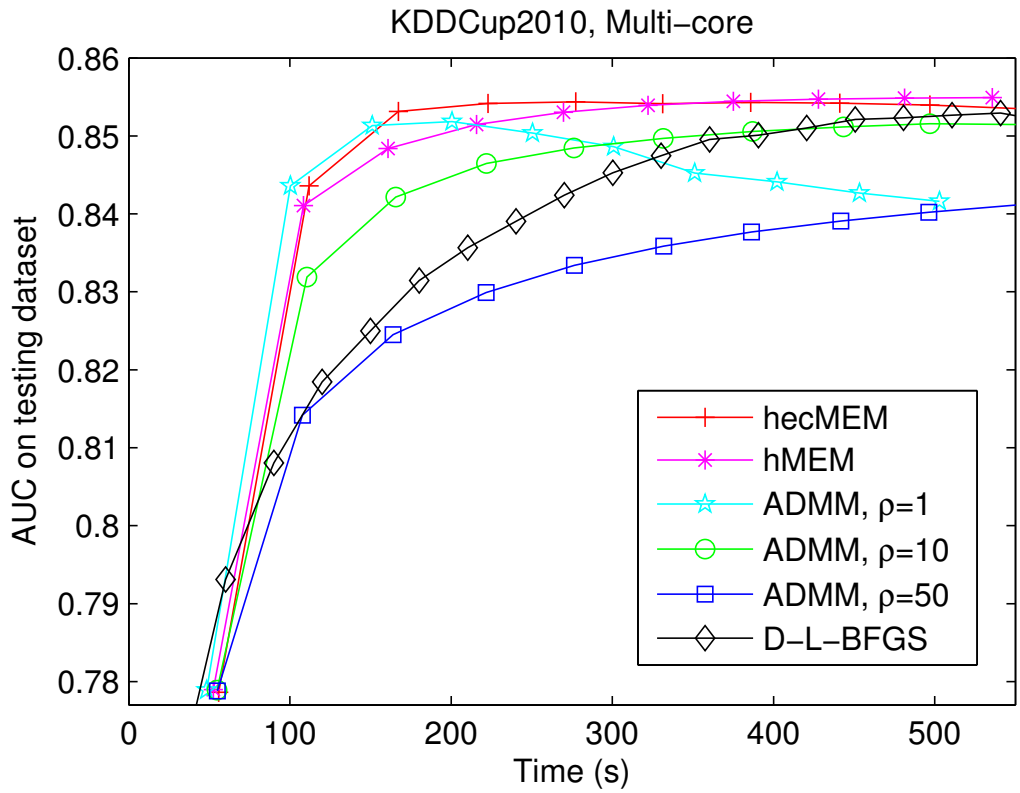

FIGURE 6.1: AUC versus computation time in the multi-core environment on the KDDCup2010 data.

6.3 Experimental results for sparse logistic regression

We compare three types of algorithms: (*i*) the proposed hMEM and hecMEM methods, (*ii*) ADMM for distributed logistic regression [7], and (*iii*) the L-BFGS algorithm for logistic regression implemented in a distributed fashion as discussed in Section 4.2. Good initialization helps convergence, and in the following we use MEM to provide an initialization for hMEM and hecMEM, while AVGM is used as the initialization of ADMM.

In the first experiment we compare hMEM, hecMEM, ADMM and the distributed L-BFGS (D-L-BFGS) in the multi-core setting with the KDDCup2010 dataset. Since there are 16 cores available as discussed in Section 6.1, for hMEM, hecMEM and ADMM we set the number of data blocks $B = 16$. Note that in a multi-core setting,

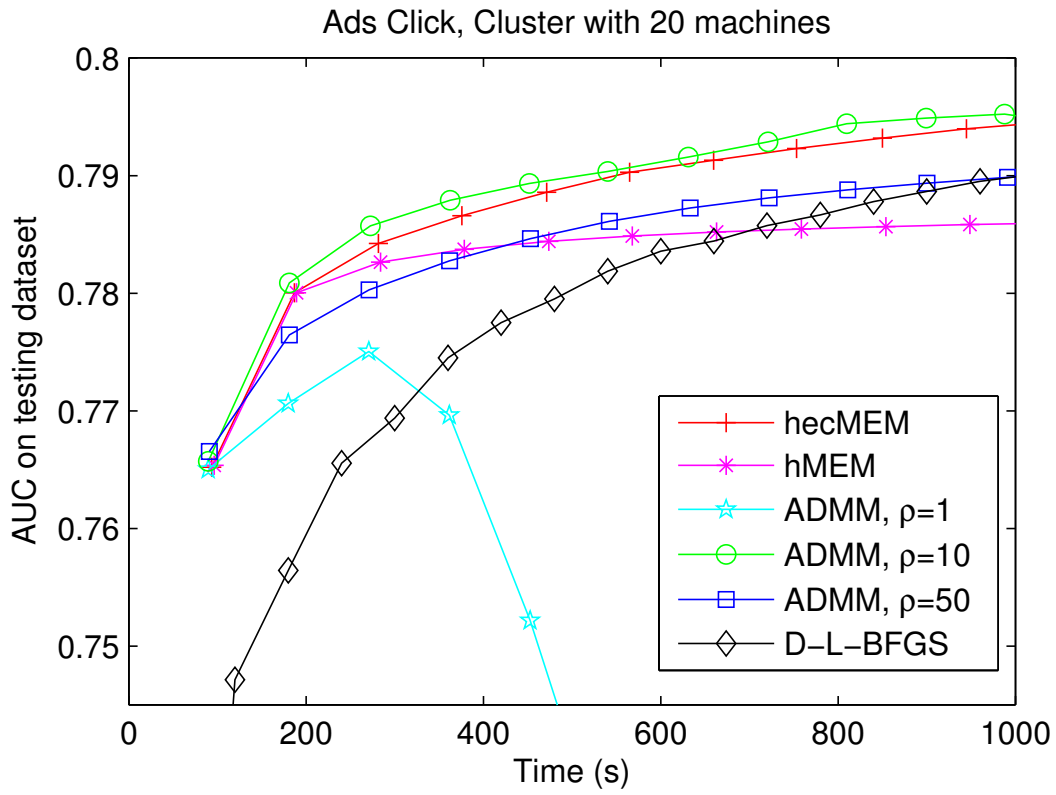

FIGURE 6.2: AUC versus computation time in a distributed environment with 20 computing nodes on the Ads click data

we can better understand the convergence of different methods when the delay caused by communication is negligible. The result is shown in Figure 6.1. We observe that both hMEM and hecMEM converge faster than the other methods, and we deduce this is because KDDCup2010 is a sparse dataset, and the probabilistic nature of hMEM and hecMEM handles the uncertainty caused by sparsity better than the non-probabilistic algorithms. We also observe that ADMM is moderately sensitive to its tuning parameter ρ for this dataset, as small ρ causes over-fitting while big ρ leads to slower convergence. Note that in preliminary experiments we compared ADMM with different ρs, and the representative examples $\rho = \{1, 10, 50\}$ are chosen for illustration.

Similar observations are made in our second experiment, as shown in Figure

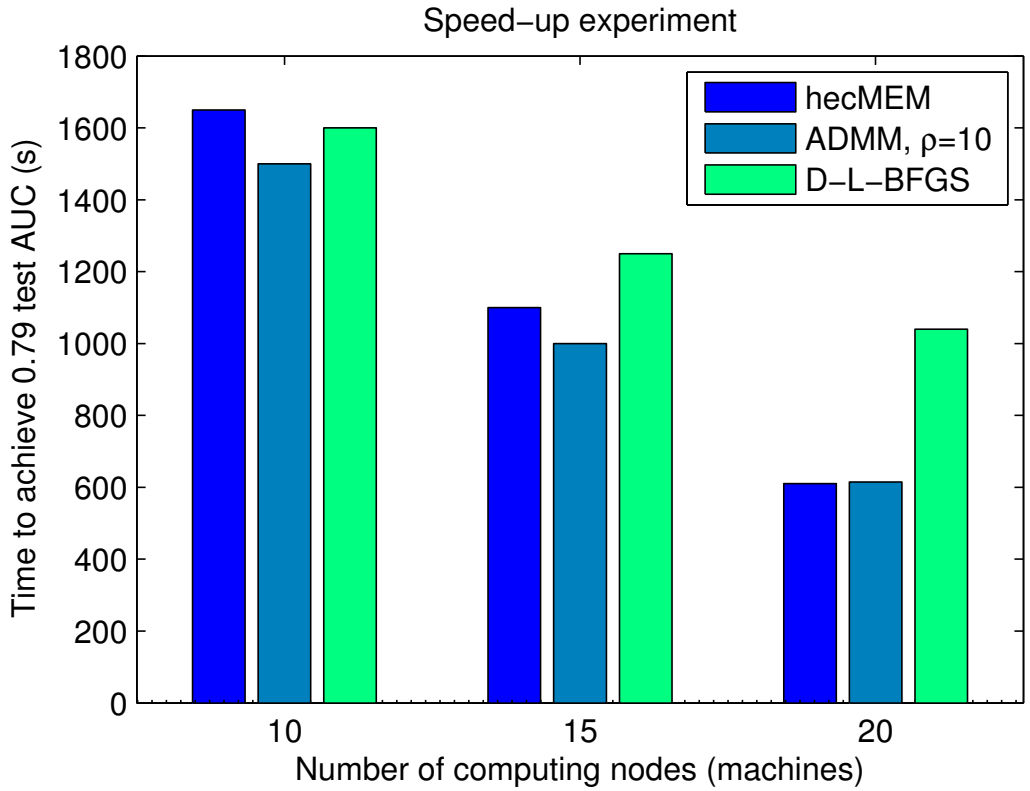

FIGURE 6.3: Speed-up comparison among hecMEM, ADMM and D-LBFGS on the Ads click data.

6.2, where we compare the four different algorithms in the distributed environment using the Ads click data. ADMM converges faster with a specific configuration of the tuning parameter ρ, as different configurations may either lead to a quick over-fitting or slower convergence. While hecMEM is more robust, as it updates the parameter ρ automatically, and reaches a comparable result of a well-tuned ADMM.

We also perform the speed-up experiment, which measures how much faster a distributed method is, in our case the time needed to reach an AUC of 0.79, when the number of computing nodes is increasing; in our case we consider from 10 to 20 machines. We observe that both hecMEM and ADMM have near linear speed-up, and the discrepancy between hecMEM and a well-tuned ADMM diminishing with the increase number of computing nodes, as the probabilistic formulation helps hecMEM

Table 6.2: Statistics for each collaborative filtering dataset

	M	N	$\lvert\Omega\rvert$	$\lvert\Omega^{\text{Test}}\rvert$
Netflix	2,649,429	17,770	99,072,112	1,408,395
Yahoo music	1,000,990	624,961	252,800,275	4,003,960
Synthetic	1,000,000	1,000,000	2,025,521,970	22,543,268

handle increased uncertainty caused by the increase in the number of blocks.

6.4 Datasets for low-rank matrix completion

We consider two public datasets, the Netflix dataset and Yahoo music rating (KDDCup2011) dataset, with the original training/test split retained for reproducibility. To test the scalability of the proposed methods, we generate a synthetic 1M $\times$ 1M matrix with rank 10 as described in [40]. Important statistics of each dataset is summarized in Table 6.2.

6.5 Experimental results for low-rank matrix completion

In the experiments we compare the proposed hMEM and hecMEM with ADMM with the low-rank matrix completion formulation. We also consider the CCD++ algorithm [40], shown to be effective compared to various other distributed matrix factorization framework, including distributed ALS [47], Hogwild! [28] and DSGS [14]. We also extend the CCD++ algorithm into a variational Bayesian setting by treating the factor matrices U, V as random variables with Gaussian priors, and learn $q(U, V)$ instead of U, V by assuming $q(U, V) = \prod_{m=1}^{M} \prod_{k=1}^{K} q(u_{mk}) \prod_{n=1}^{N} \prod_{k=1}^{K} q(v_{nk})$, and $q(u_{mk}), q(v_{nk})$ is updated the same fashion as u_{mk}, v_{nk} being updated in [40]. We term this method VB-CCD. We set $K = 30$ for the Netflix dataset and K=100 for the Yahoo music dataset, and the tuning parameters for CCD++ is set as in [40].

For the multi-core setting, the result on Netflix is shown in Figure 6.4, where for both hecMEM and ADMM the matrix is partitioned into 16 blocks, with $B^r = 16$

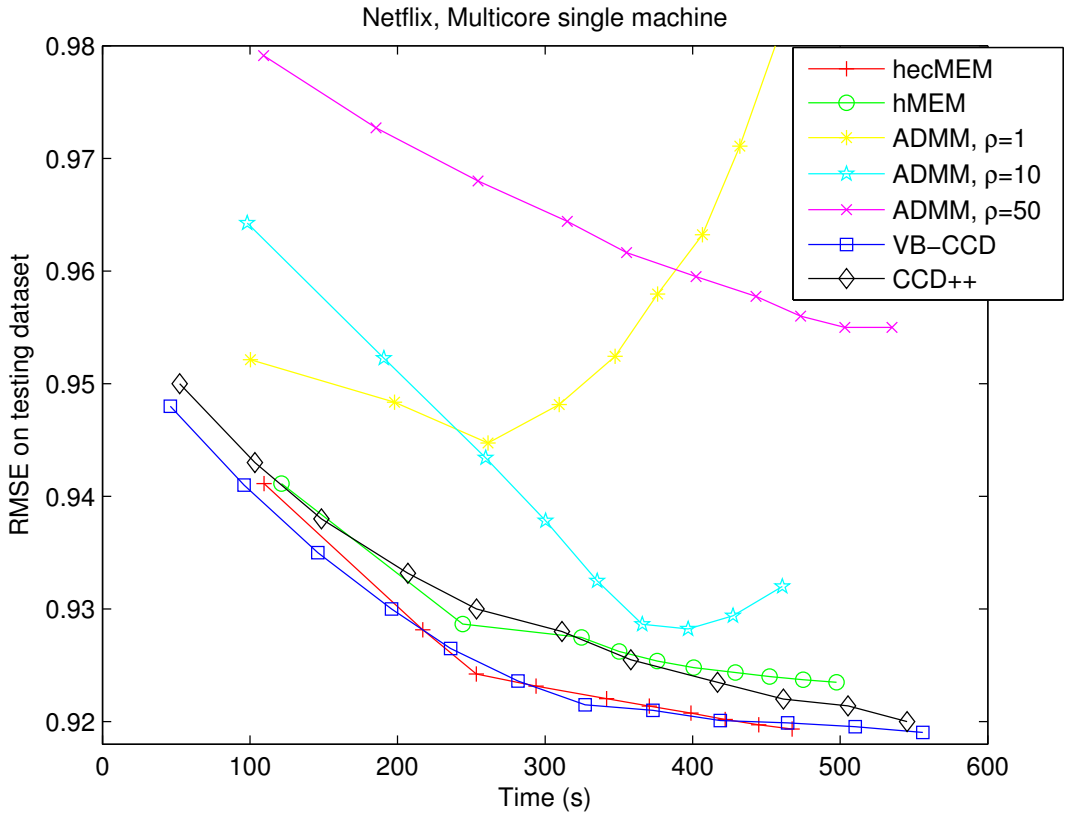

FIGURE 6.4: RMSE versus computation time in the multi-core environment on the Netflix data

and $B^c = 1$ because there are much more rows than columns. Note that ADMM-based methods with different tuning parameter ρ perform much worse than the other methods, as it either overfits quickly or is very slow to converge. This is likely partly because Netflix is already an extremely sparse dataset; by partitioning it into blocks, we only increase the degree of sparsity within each block, and the point estimates obtained from ADMM are over-confident and ignore uncertainty. Besides, the non-convex nature of the low-rank completion problem means that the local estimates of ADMM may find different local optimal, slowing convergence. On the other hand, hecMEM and VB-CCD takes all the possible configurations of local parameter into consideration and embraces the uncertainty, and converges faster

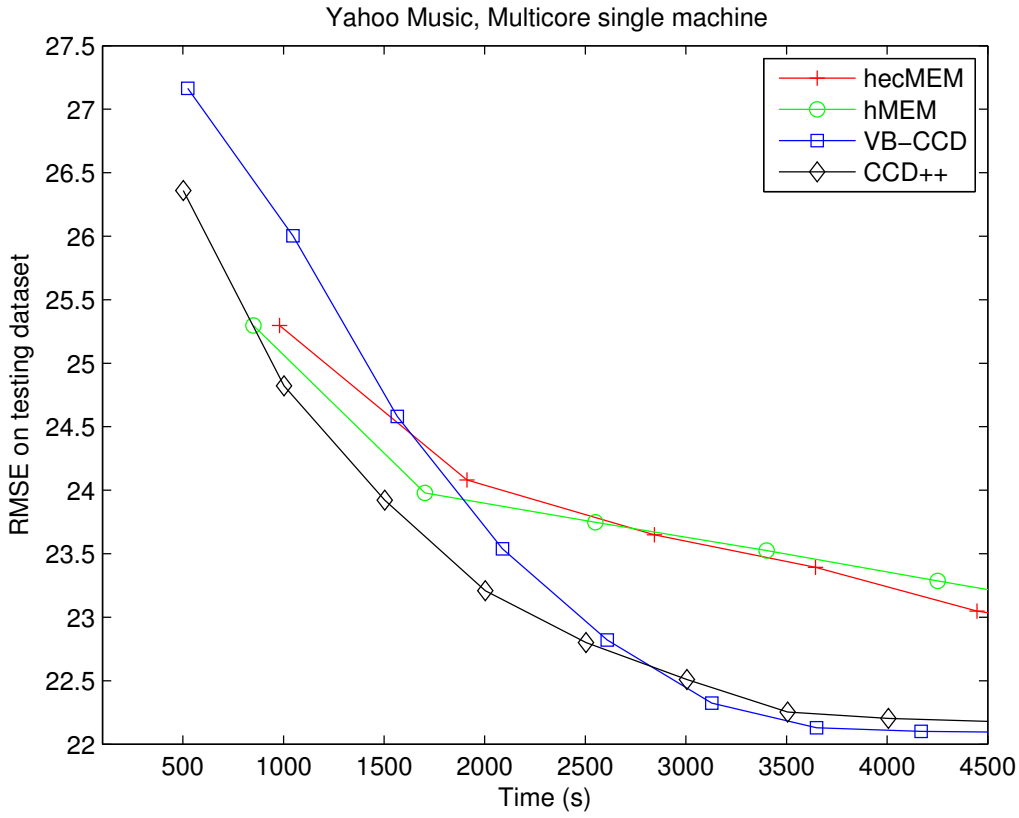

FIGURE 6.5: RMSE versus computation time in the multi-core environment on the Yahoo music data

than the rest methods when the communication cost is negligible. In Figure 6.5, the comparison is made based on the Yahoo music dataset, where the matrix is again partitioned into 16 blocks, with $B^r = 4$ and $B^c = 4$. Note that the ADMM result is not shown, as ADMM is again much slower than the other methods to achieve converge, or it over-fits quickly. In this case, hMEM and hecMEM converge slower than CCD++ and VB-CCD, and the loss of accuracy may be caused by the augmented latent random variables $\mathcal{U}_{ij}$ and $\mathcal{V}_{ij}$ with the mean-field approximation on each of them in Section 5.2.

We also make comparisons among methods in the distributed environment using the Yahoo music dataset, as shown in Figure 6.6 - 6.7. The distributed environment

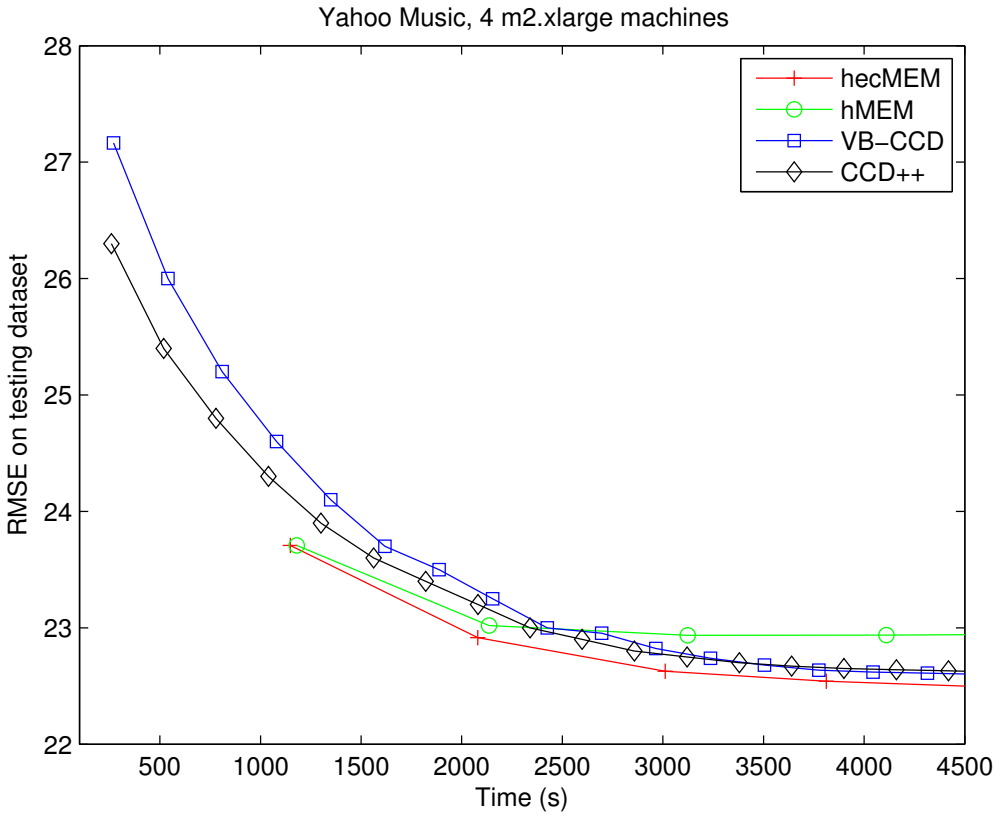

FIGURE 6.6: RMSE versus computation time in the distributed environment on the Yahoo music data with 4 m2.xlarge machines

on the left panel consists of 4 m2.xlarge machines with the matrix partitioned as $B^r = 2$ and $B^c = 2$, and the right panel consists of 9 m2.xlarge machines with $B^r = 3$ and $B^c = 3$. We observe that with the increased number of computing nodes, hecMEM's advantage becomes clearer. One possible explanation is that the CCD++ and VB-CCD take many short iterations, causing increased communication cost.

Finally, we conduct the speed-up experiment on the synthetic dataset, as shown in Figure 6.8. Consistent observations are made from Figure 6.8, with hecMEM being relatively slower to converge when the number of computing nodes is small, but it enjoys a near-linear speed-up with the increase of number of computing nodes,

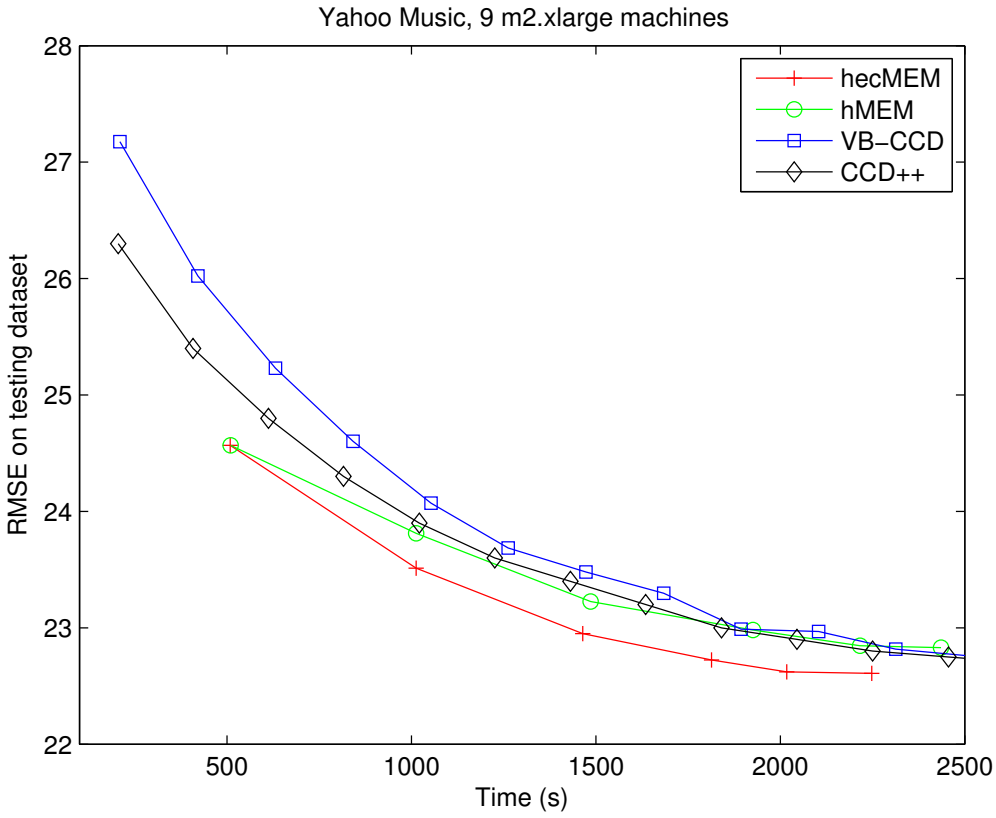

FIGURE 6.7: RMSE versus computation time in the distributed environment on the Yahoo music data with 9 m2.xlarge machines.

and finally becomes the first to achieve the targeted RMSE.

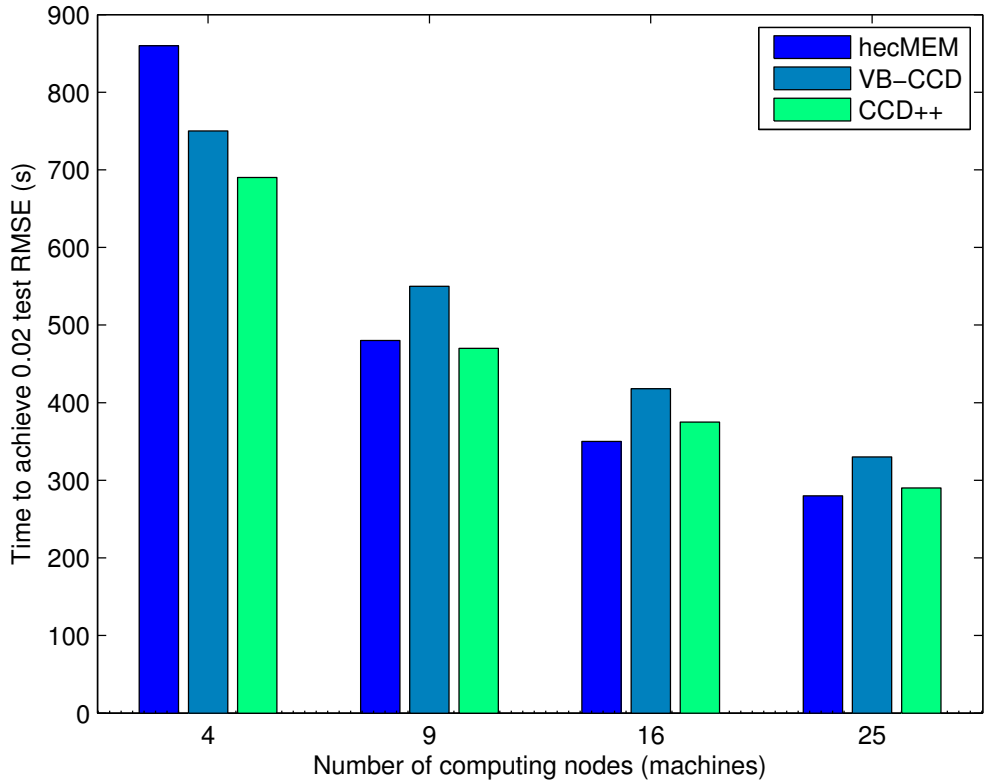

FIGURE 6.8: Speed-up comparison among hecMEM, VB-CCD and CCD++ on the synthetic data

7

Conclusions

In this dissertation a series of new probabilistic hierarchical models have been developed for analysis of massive-scale data within a general empirical-risk setting. We have demonstrated this formulation for logistic regression and matrix completion on massive-scale data, with encouraging results in comparisons to several of the leading methods in the literature.

The basic structure of the proposed method is well suited to other types of problems that have often been analyzed in a probabilistic setting. For example, it is of interest to extend the framework to topic modeling of massive document corpora (matrix of count data). For such problems it is anticipated that the details of the implementation will change, but the basic guiding principles will be retained: (*i*) account for uncertainty via probabilistic inference of parameters; (*ii*) impose a hierarchical model, to share statistical strength between data blocks; and (*iii*) impose consistency across the block-dependent estimates, via a constraint on the expectation or similar metric.

Bibliography

[1] O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *J. Mach. Learn. Res.*, 2008.

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

[3] D. Bertsekas. *Parallel and Distributed Computation: Numerical Methods*. 1989.

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

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

[6] L. Bottou. Online learning and stochastic approximation. *Online Learning and Neural Networks*, 1998.

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

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

[9] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In *NIPS*. 2007.

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

[11] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 2008.

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

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

[14] R. Gemulla, E. Nijkamp, P.J. Haas, and Y. Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. KDD '11, 2011.

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

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

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

[18] A. Juditsky, G. Lan, A.and Nemirovski, and A Shapiro. Stochastic approximation approach to stochastic programming. *SIAM J. Optim*, 2009.

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

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

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

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

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

[24] Y. J. Lim and Y. W. Teh. Variational Bayesian approach to movie rating prediction. In *KDDCup Workshop*, 2007.

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

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

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

[28] F. Niu, B. Recht, C. Ré, and S.J. Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, 2011.

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

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

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

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

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

[34] E. Sparks, A. Talwalkar, V. Smith, J. Kottalam, X. Pan, J. Gonzalez, M. Franklin, and M. I. Jordan adn T. Kraska. Mli: An api for distributed machine learning. In *ICDM*, 2013.

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

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

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

[38] M. Xu, J. Zhu, and B. Zhang. Bayesian nonparametric maximum margin matrix factorization for collaborative prediction. In *NIPS*, 2012.

[39] M. Xu, J. Zhu, and B. Zhang. Fast max-margin matrix factorization with data augmentation. In *ICML*, 2013.

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

[41] H.-F. Yu, H.-Y. Lo, et al. Feature engineering and classifier ensemble for kdd cup 2010. Proceedings of the KDD Cup 2010 Workshop, 2010.

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

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

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

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

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

[47] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the netflix prize. AAIM '08, 2008.

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

[49] M Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, 2003.

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

Biography

1. Name: XianXing Zhang
2. Birth date: January 2, 1987
3. Degrees earned
    - Duke University, Durham, NC, USA
        - Ph.D., Electrical and Computer Engineering, May, 2014 (expected)
        - M.Sc., Statistical Science, Fall, 2011
    - Nanjing University, Nanjing, Jiangsu, China
        - B.Sc., Computer Science, May, 2009
4. Publications
    - **XianXing Zhang**, Lawrence Carin. "Joint Modeling of a Matrix with Associated Text via Latent Binary Features". In NIPS 2012.
    - Lingbo Li, **XianXing Zhang**, Lawrence Carin. "Nested Dictionary Learning for Hierarchical Organization of Imagery and Text" . In UAI 2012.
    - **XianXing Zhang**, David B. Dunson, Lawrence Carin. "Hierarchical Topic Modeling for Analysis of Time-Evolving Personal Choices". In NIPS 2011.
    - **XianXing Zhang**, David B. Dunson, Lawrence Carin. "Tree-Structured Infinite Sparse Factor Model". In ICML 2011.
    - Derek Hao Hu, **XianXing Zhang**, Jie Yin, Vincent Wenchen Zheng, Qiang Yang. "Abnormal Activity Recognition based on HDP-HMM Models". In IJCAI 2009.
    - **XianXing Zhang**, Hua Liu, Yang Gao, Derek Hao Hu. "Detecting abnormal events via Hierarchical Dirichlet Processes" **Best Student Paper Award**. In PAKDD 2009.