REVIEW

INFORMATION SCIENCE

Big Learning with Bayesian methods

Jun Zhu*, Jianfei Chen, Wenbo Hu and Bo Zhang

ABSTRACT

The explosive growth in data volume and the availability of cheap computing resources have sparked increasing interest in Big learning, an emerging subfield that studies scalable machine learning algorithms, systems and applications with Big Data. Bayesian methods represent one important class of statistical methods for machine learning, with substantial recent developments on adaptive, flexible and scalable Bayesian learning. This article provides a survey of the recent advances in Big learning with Bayesian methods, termed Big Bayesian Learning, including non-parametric Bayesian methods for adaptively inferring model complexity, regularized Bayesian inference for improving the flexibility via posterior regularization, and scalable algorithms and systems based on stochastic subsampling and distributed computing for dealing with large-scale applications. We also provide various new perspectives on the large-scale Bayesian modeling and inference.

Keywords: Big Bayesian Learning, Bayesian non-parametrics, regularized Bayesian inference, scalable algorithms

INTRODUCTION

We live in an era of Big Data, where science, engineering and technology are producing massive data streams, with petabyte and exabyte scales becoming increasingly common [1–3]. Besides the explosive growth in volume, Big Data also has high velocity, high variety and high uncertainty. These complex data streams require ever-increasing processing speeds, economical storage and timely response for decision making in highly uncertain environments, and have raised various challenges to conventional data analysis [4].

With the primary goal of building intelligent systems that automatically improve from experiences, machine learning (ML) is becoming an increasingly important field to tackle the Big Data challenges [5], with an emerging field of 'Big Learning', which covers theories, algorithms and systems on addressing Big Data problems.

Big Learning challenges

In Big Data era, ML needs to deal with the challenges of learning from complex situations with 'large' N, 'large' P, 'large' L and 'large' M, where N is the data size, P is the feature dimension, L is the number of

tasks and M is the model size. Given that N is obvious, we explain the other factors below.

Large P: with the development of Internet, data sets with ultrahigh dimensionality have emerged, such as the spam filtering data with trillion features [6] and the even higher dimensional feature space via explicit kernel mapping [7]. Note that whether a learning problem is high-dimensional depends on the ratio between P and N. Many scientific problems with $P \gg N$ impose great challenges on learning, calling for effective regularization techniques to avoid overfitting and select salient features [4].

Large L: many tasks involve classifying text or images into tens of thousands or millions of categories. For example, the ImageNet [8] database consists of more than 14 millions of web images from 21 thousands of concepts, while with the goal of providing on average 1000 images for each of 100+ thousands of concepts (or synsets) in WordNet; and the LSHTC text classification challenge 2014 aims to classify Wikipedia documents into one of 325 056 categories [9]. Often, these categories are organized in a graph, e.g. the tree structure in ImageNet and the DAG (directed acyclic graph) structure in LSHTC, which can be explored for better learning [10,11].

TNList Lab, State Key Lab for Intelligent Technology and Systems, CBICR Center, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

*Corresponding author. E-mail: dcszj@mail.tsinghua.edu.cn

Large *M*: with the availability of massive data, models with millions or billions of parameters are becoming common. Significant progress has been made on learning deep models, which have multiple layers of non-linearities allowing them to extract multigrained representations of data, with successful applications in computer vision, speech recognition and natural language processing. Such models include neural networks [12], autoencoders [13,14] and probabilistic generative models [15,16].

Big Bayesian learning

Though Bayesian methods have been widely used in ML and many other areas, skepticism often arises when we talking about Bayesian methods for Big Data [17]. Practitioners also criticize that Bayesian methods are often too slow for even small-scaled problems, owing to many factors such as the non-conjugacy models with intractable integrals. Nevertheless, Bayesian methods have several advantages on dealing with the following.

(i) Uncertainty: our world is an uncertain place because of physical randomness, incomplete knowledge, ambiguities and contradictions. Bayesian methods provide a principled theory for combining prior knowledge and uncertain evidence to make sophisticated inference of hidden factors and predictions.

(ii) Flexibility: Bayesian methods are conceptually simple and flexible. Hierarchical Bayesian modeling offers a flexible tool for characterizing uncertainty, missing values, latent structures and more. Moreover, regularized Bayesian inference (RegBayes) [18] further augments the flexibility by introducing an extra dimension (i.e. a posterior regularization term) to incorporate domain knowledge or to optimize a learning objective. Finally, there exist very flexible algorithms (e.g. Markov Chain Monte Carlo (MCMC)) to perform posterior inference.

(iii) Adaptivity: the dynamics and uncertainty of Big Data require that our models should be adaptive when the learning scenarios change. Non-parametric Bayesian (NPB) methods provide elegant tools to deal with situations in which phenomena continue to emerge as data are collected [19]. Moreover, the Bayesian updating rule and its variants are sequential in nature and suitable for dealing with Big Data streams.

(iv) Overfitting: although the data volume grows exponentially, the predictive information grows slower than the amount of Shannon information [20], while our models are becoming increasingly large by leveraging powerful computers, such as the deep networks with billions of parameters. It implies that our models are increasing their capacity faster than the amount of information that we need to fill them with, therefore causing serious overfitting problems that call for effective regularization [21].

Therefore, Bayesian methods are becoming increasingly relevant in the Big Data era [22] to protect high-capacity models against overfitting, and to allow models adaptively updating their capacity. However, the application of Bayesian methods to Big Data problems runs into a computational bottleneck that needs to be addressed with new (approximate) inference methods. This article aims to provide a literature survey of the recent advances in Big Learning with Bayesian methods, including the basic concepts of Bayesian inference, NPB methods, RegBayes, scalable inference algorithms and systems based on stochastic subsampling and distributed computing. It is useful to note that our review is no way exhaustive. We select the materials to make it self-contained and technically rigorous. As data analysis is becoming an essential function in many scientific and engineering areas, this article should be of broad interest to the audiences who are dealing with data, especially those who are using statistical tools.

BASICS OF BAYESIAN METHODS

The general blueprint of Bayesian data analysis [23] is that a Bayesian model expresses a generative process of the data that includes hidden variables, under some statistical assumptions. The process specifies a joint probability distribution of the hidden and observed random variables. Given a set of observed data, data analysis is performed by 'posterior inference', which computes the conditional distribution of the hidden variables given the observed data. This section reviews the basic concepts and algorithms of Bayesian inference.

Bayes' theorem

At the core of Bayesian methods is Bayes' theorem (a.k.a Bayes' rule). Let $\boldsymbol{\Theta}$ be the model parameters and $\mathcal{D}$ be the given data set. The Bayesian posterior distribution is

$$p(\boldsymbol{\Theta}|\mathcal{D}) = \frac{p_0(\boldsymbol{\Theta})p(\mathcal{D}|\boldsymbol{\Theta})}{p(\mathcal{D})}, \qquad (1)$$

where $p_0(\cdot)$ is a prior distribution, chosen before seeing any data; $p(\mathcal{D}|\boldsymbol{\Theta})$ is the assumed likelihood model; and $p(\mathcal{D}) = \int p_0(\boldsymbol{\Theta}) p(\mathcal{D}|\boldsymbol{\Theta}) \mathrm{d}\boldsymbol{\Theta}$ is the marginal likelihood (or evidence), often involving an intractable integration problem that requires approximate inference as detailed below. The year 2013 marks the 250th anniversary of Thomas Bayes' essay on how humans can sequentially learn from experience, steadily updating their beliefs as more data become available [24].

A useful variational formulation of Bayes' rule is

$$\min_{q(\boldsymbol{\Theta})\in\mathcal{P}} \mathrm{KL}(q(\boldsymbol{\Theta})\|p_0(\boldsymbol{\Theta})) - \mathbb{E}_q[\log p(\mathcal{D}|\boldsymbol{\Theta})], \tag{2}$$

where $\mathcal{P}$ is the space of all distributions that make the objective well-defined. It can be shown that the optimum solution to (2) is identical to the Bayesian posterior. In fact, if we add the constant term $\log p(\mathcal{D})$, the problem is equivalent to minimizing the KL-divergence between $q(\boldsymbol{\Theta})$ and the Bayesian posterior $p(\boldsymbol{\Theta}|\mathcal{D})$, which is non-negative and takes 0 if and only if q equals to $p(\boldsymbol{\Theta}|\mathcal{D})$. The variational interpretation is significant in two aspects: (i) it provides a basis for variational Bayes methods; and (ii) it provides a starting point to make Bayesian methods more flexible by incorporating a rich set of posterior constraints. We will make these clear soon later.

It is noteworthy that $q(\boldsymbol{\Theta})$ represents the density of a general post-data posterior in the sense of [25, pp.15] not necessarily corresponding to a Bayesian posterior induced by Bayes' rule. As we shall see in Section Regularized Bayesian inference, when we introduce additional constraints, the post-data posterior $q(\boldsymbol{\Theta})$ is different from the Bayesian posterior $p(\boldsymbol{\Theta}|\mathcal{D})$, and moreover, it could even not be obtainable by the conventional Bayesian inference via Bayes' rule. In the sequel, in order to distinguish $q(\cdot)$ from the Bayesian posterior, we will call it post-data posterior. The optimization formulation in (ii) implies that Bayes' rule is an information projection procedure that projects a prior density to a post-data posterior by taking account of the observed data. In general, Bayes's rule is a special case of the principle of minimum information [25,26].

Bayesian methods in ML

Bayesian statistics has been applied to almost every ML task ranging from the single-variate regression/classification to the structured output predictions and to the unsupervised/semi-supervised learning scenarios [27]. In essence however, there are several basic tasks that we briefly review below.

Prediction: after training, Bayesian models make predictions using the distribution:

$$p(\mathbf{x}|\mathcal{D}) = \int p(\mathbf{x}, \boldsymbol{\Theta}|\mathcal{D}) \mathrm{d}\boldsymbol{\Theta}$$
$$= \int p(\mathbf{x}|\boldsymbol{\Theta}, \mathcal{D}) p(\boldsymbol{\Theta}|\mathcal{D}) \mathrm{d}\boldsymbol{\Theta}, \quad (3)$$

where $p(\mathbf{x}|\boldsymbol{\Theta}, \mathcal{D})$ is often simplified as $p(\mathbf{x}|\boldsymbol{\Theta})$ due to the i.i.d assumption of the data when the model is given. Since the integral is taken over the posterior distribution, the training data are considered.

Model selection: model selection is a fundamental problem in statistics and ML [28]. Let $\mathbf{M}$ be a family of models where each model is indexed by a set of parameters $\boldsymbol{\Theta}$. Then, the marginal likelihood of the model family (or model evidence) is

$$p(\mathcal{D}|\mathbf{M}) = \int p(\mathcal{D}|\boldsymbol{\Theta}) p(\boldsymbol{\Theta}|\mathbf{M}) \mathrm{d}\boldsymbol{\Theta}, \quad (4)$$

where $p(\boldsymbol{\Theta}|\mathbf{M})$ is often assumed to be uniform if no strong prior exists.

For two different model families $\mathbf{M}_1$ and $\mathbf{M}_2$, the ratio of model evidences $\kappa = \frac{p(\mathcal{D}|\mathbf{M}_1)}{p(\mathcal{D}|\mathbf{M}_2)}$ is called Bayes factor [29]. The advantage of using Bayes factors for model selection is that it automatically and naturally includes a penalty for including too much model structure [27, chap 3]. Thus it guards against overfitting. For models where an explicit version of the likelihood is not available or too costly to evaluate approximate Bayesian computation can be used for model selection in a Bayesian framework [30,31] while with the caveat that approximate-Bayesian estimates of Bayes factors are often biased [32].

Approximate Bayesian inference

Though conceptually simple Bayesian inference has computational difficulties, which arise from the intractability of high-dimensional integrals as involved in the posterior and in Eqs (3, 4). These are typically not only analytically intractable but also difficult to obtain numerically. Common practice resorts to approximate methods, which can be grouped into two categories (Both maximum likelihood estimation (MLE), $\hat{\boldsymbol{\Theta}}_{\mathrm{MLE}} = \mathrm{argmax}_{\boldsymbol{\Theta}} p(\mathcal{D}|\boldsymbol{\Theta})$, and maximum a posterior estimation (MAP), $\dot{\boldsymbol{\Theta}}_{\mathrm{MAP}} = \mathrm{argmax}_{\boldsymbol{\Theta}} p_0(\boldsymbol{\Theta}) p(\mathcal{D}|\boldsymbol{\Theta})$, can be seen as the third type of approximation methods to do Bayesian inference. We omit them since they examine only a single point, and so can neglect the potentially large distributions in the integrals.)—variational methods and MC methods.

Variational Bayesian methods

Variational methods have a long history in physics, statistics, control theory and economics. In ML, variational formulations appear naturally in regularization theory, maximum entropy estimates and approximate inference in graphical models. We refer the readers to the seminal book [33] and the nice short overview [34] for more details. A variational method basically consists of two parts:

(i) cast the problems as some optimization problems;

(ii) find an approximate solution when the exact solution is not feasible.

For Bayes' rule we have provided a variational formulation in (2) which is equivalent to minimizing the KL-divergence between the variational distribution $q(\boldsymbol{\Theta})$ and the target posterior $p(\boldsymbol{\Theta}|\mathcal{D})$. We can also show that the negative of the objective in (2) is a lower bound of the evidence (i.e. log-likelihood):

$$\log p(\mathcal{D}) \geq \mathbb{E}_q[\log p(\boldsymbol{\Theta}, \mathcal{D})] - \mathbb{E}_q[\log q(\boldsymbol{\Theta})]. \tag{5}$$

Then, variational Bayesian methods maximize the evidence lower bound (ELBO):

$$\max_{q \in \mathcal{P}} \mathbb{E}_q[\log p(\boldsymbol{\Theta}, \mathcal{D})] - \mathbb{E}_q[\log q(\boldsymbol{\Theta})], \tag{6}$$

whose solution is the target posterior if no assumptions are made.

However, in many cases it is intractable to calculate the target posterior. Therefore, to simplify the optimization, the variational distribution is often assumed to be in some parametric family, e.g. $q_{\phi}(\boldsymbol{\Theta})$, and has some mean-field representation:

$$q_{\phi}(\boldsymbol{\Theta}) = \prod_{i=1}^{M} q_{\phi_i}(\boldsymbol{\Theta}_i), \tag{7}$$

where the set $\{\boldsymbol{\Theta}_i\}_{i=1}^{M}$ represents a partition of $\boldsymbol{\Theta}$. Then, the problem transforms to find the best parameters $\hat{\boldsymbol{\phi}}$ that maximize the ELBO, which can be solved with numerical optimization methods. For example, with the factorization assumption, coordinate descent is often used to iteratively solve for ϕ_i until reaching some local optimum. Once a variational approximation q^* is found, the Bayesian integrals can be approximated by replacing $p(\boldsymbol{\Theta}|\mathcal{D})$ by q^*. In many cases, the model $\boldsymbol{\Theta}$ consists of parameters $\boldsymbol{\theta}$ and hidden variables $\mathbf{h}$. Then, if we make the (structured) mean-field assumption that $q(\boldsymbol{\theta}, \mathbf{h}) = q(\boldsymbol{\theta})q(\mathbf{h})$, the variational problem can be solved by a variational Bayesian EM algorithm [35] which alternately updates $q(\mathbf{h})$ at the variational Bayesian E-step and updates $q(\boldsymbol{\theta})$ at the variational Bayesian M-step.

MC methods

MC methods represent a diverse class of algorithms that rely on repeated random sampling to compute the solution to problems whose solution space is too large to explore systematically or whose systemic behavior is too complex to model. The basic idea of MC methods is to draw a set of i.i.d samples $\{\boldsymbol{\Theta}_i\}_{i=1}^{N}$ from a target distribution $p(\boldsymbol{\Theta})$ and use the empirical distribution $\hat{p}(\cdot) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\boldsymbol{\Theta}_i}(\cdot)$, to approximate the target distribution, where $\delta_{\boldsymbol{\Theta}_i}(\cdot)$ is the delta-Dirac mass located at $\boldsymbol{\Theta}_i$. Consider the common operation on calculating the expectation of some function ϕ with respect to a given distribution. Let $p(\boldsymbol{\Theta}) = \tilde{p}(\boldsymbol{\Theta})/Z$ be the density of a probability distribution, where $\tilde{p}(\boldsymbol{\Theta})$ is the unnormalized version that can be computed pointwise up to a normalizing constant Z. The expectation of interest is

$$I = \int \phi(\boldsymbol{\Theta})p(\boldsymbol{\Theta})\mathrm{d}\boldsymbol{\Theta}. \tag{8}$$

Replacing $p(\cdot)$ by $\hat{p}(\cdot)$, we get the unbiased MC estimate of this quantity:

$$\hat{I}_{\mathrm{MC}} = \frac{1}{N}\sum_{i=1}^{N} \phi(\boldsymbol{\Theta}_i). \tag{9}$$

Asymptotically, when $N \to \infty$, the estimate $\hat{I}_{\mathrm{MC}}$ will almost surely converge to I by the strong law of large numbers. In practice, however, we often cannot sample from p directly. Many methods have been developed, such as rejection sampling and importance sampling, which however often suffer from severe limitations in high dimensional spaces. We refer the readers to the book [36] and the review article [37] for details. Below we introduce MCMC a very general and powerful framework that allows sampling from a broad family of distributions and scales well with the dimensionality of the sample space. More importantly many advances have been made on scalable MCMC methods for Big Data, which will be discussed later.

An MCMC method constructs an ergodic p-stationary Markov chain sequentially. Once the chain has converged (i.e. finishing the burn-in phase), we can use the samples to estimate I. The Metropolis-Hastings algorithm [38,39] constructs such a chain by using the following rule to transit from the current state $\boldsymbol{\Theta}_t$ to the next state $\boldsymbol{\Theta}_{t+1}$:

(i) draw a candidate state $\boldsymbol{\Theta}'$ from a proposal distribution $q(\boldsymbol{\Theta}|\boldsymbol{\Theta}_t)$;

(ii) compute the acceptance probability:

$$A(\mathbf{\Theta}', \mathbf{\Theta}_t) \triangleq \min\left(1, \frac{\tilde{p}(\mathbf{\Theta}')q(\mathbf{\Theta}_t|\mathbf{\Theta}')}{\tilde{p}(\mathbf{\Theta}_t)q(\mathbf{\Theta}'|\mathbf{\Theta}_t)}\right);$$

$$(10)$$

(iii) draw $\gamma \sim$ Uniform$[0, 1]$. If $\gamma < A(\mathbf{\Theta}', \mathbf{\Theta}_t)$ set $\mathbf{\Theta}_{t+1} \leftarrow \mathbf{\Theta}'$, otherwise set $\mathbf{\Theta}_{t+1} \leftarrow \mathbf{\Theta}_t$.

Note that for Bayesian models, each MCMC step involves an evaluation of the full likelihood to get the (unnormalized) posterior $\tilde{p}(\mathbf{\Theta})$, which can be prohibitive for Big Learning with massive data sets. We will revisit this problem later.

One special type of MCMC methods is the Gibbs sampling [40] which iteratively draws samples from local conditionals. Let $\mathbf{\Theta}$ be a M-dimensional vector. The standard Gibbs sampler performs the following steps to get a new sample $\mathbf{\Theta}^{(t+1)}$:

(i) draw a sample $\theta_1^{(t+1)} \sim p(\theta_1|\theta_2^{(t)}, \cdots, \theta_M^{(t)})$;
(ii) for $j = 2: M - 1$, draw a sample

$$\theta_j^{(t+1)} \sim p(\theta_j|\theta_1^{(t+1)}, \cdots, \theta_{j-1}^{(t+1)}, \theta_{j+1}^t \cdots, \theta_M^t);$$

(iii) draw a sample $\theta_M^{(t+1)} \sim p(\theta_M|\theta_1^{(t+1)}, \cdots, \theta_{M-1}^{(t+1)})$.

One issue with MCMC methods is that the convergence rate can be prohibitively slow even for conventional applications. Extensive efforts have been spent to improve the convergence rates. For example, hybrid MC methods explore gradient information to improve the mixing rates when the model parameters are continuous, with representative examples of Langevin dynamics and Hamiltonian dynamics [41]. Other improvements include population-based MCMC methods [42] and annealing methods [43] that can sometimes handle distributions with multiple modes. Another useful technique to develop simpler or more efficient MCMC methods is data augmentation [44–46] which introduces auxiliary variables to transform marginal dependency into a set of conditional independencies (CI). For Gibbs samplers blockwise Gibbs sampling and partially collapsed Gibbs (PCG) sampling [47] often improve the convergence. A PCG sampler is as simple as an ordinary Gibbs sampler but often improves the convergence by replacing some of the conditional distributions of an ordinary Gibbs sampler with conditional distributions of some marginal distributions.

FAQ

Common questions regarding Bayesian methods are as follows.

Q: Why should I use Bayesian methods?

A: There are many reasons for choosing Bayesian methods, as discussed in the Introduction. A formal theoretical argument is provided by the classic de Finitti theorem, which states that: If $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$ are infinitely exchangeable, then for any N

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \int \left(\prod_{i=1}^{N} p(\mathbf{x}_i|\boldsymbol{\theta})\right) dP(\boldsymbol{\theta})$$

$$(11)$$

for some random variable $\boldsymbol{\theta}$ and probability measure P. The infinite exchangeability is an often satisfied property. For example, any i.i.d data are infinitely exchangeable. Moreover, the data whose ordering information is not informative is also infinitely exchangeable, e.g. the commonly used bag-of-words representation of documents [48] and images [49].

Q: How should I choose the prior?

A: There are two schools of thought namely objective Bayes and subjective Bayes. For objective Bayes, an improper non-informative prior (e.g. the Jeffreys prior [50] and the maximum-entropy prior [51]) is used to capture ignorance which admits good frequentist properties. In contrast subjective Bayesian methods embrace the influence of priors. A prior may have some parameters λ. Since it is often difficult to elicit an honest prior, e.g. setting the true value of λ, two practical methods are often used. One is hierarchical Bayesian methods, which assume a hyper-prior on λ and define the prior as a marginal distribution:

$$p_0(\mathbf{\Theta}) = \int p_0(\mathbf{\Theta}|\lambda)p(\lambda)d\lambda. \qquad (12)$$

Though $p(\lambda)$ may have hyper-parameters as well, it is commonly believed that these parameters will have a weak influence as long as they are far from the likelihood model, thus can be fixed at some convenient values or put another layer of hyper-prior.

Another method is 'empirical' Bayes, which adopts a data-driven estimate $\hat{\lambda}$ and uses $p_0(\mathbf{\Theta}|\hat{\lambda})$ as the prior. Empirical Bayes can be seen as an approximation to the hierarchical approach, where $p(\lambda)$ is approximated by a delta-Dirac mass $\delta_{\hat{\lambda}}(\lambda)$. One common choice is maximum marginal likelihood estimate, that is, $\hat{\lambda} = \text{argmax}_\lambda p(\mathcal{D}|\lambda)$. Empirical Bayes has been applied in many problems, including variable section [52] and non-parametric Bayesian methods [53]. Recent progress has been made on characterizing the conditions when empirical Bayes merges with the Bayesian inference [54] as well as the convergence rates of empirical Bayes methods [55].
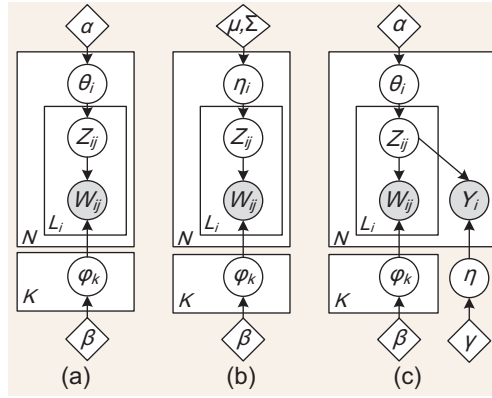
Figure 1. Graphical models of (a) LDA [48], (b) logistic-normal topic model [56], and (c) supervised LDA.

In practice another important consideration is the tradeoff between model capacity and computational cost. If a prior is conjugate to the likelihood the posterior inference will be relatively simpler in terms of computation and memory demands, as the posterior belongs to the same family as the prior.

Example 1. Dirichlet-Multinomial Conjugate Pair. Let $\mathbf{x} \in \{0, 1\}^V$ be a one-hot representation of a discrete variable with V possible values. It is easy to verify that for the multinomial likelihood, $p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{k=1}^{V} \theta_k^{x_k}$, the conjugate prior is a Dirichlet distribution, $p_0(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \mathrm{Dir}(\boldsymbol{\alpha}) = \frac{1}{Z} \prod_{k=1}^{V} \theta_k^{\alpha_k - 1}$, where $\boldsymbol{\alpha}$ is the hyper-parameter and Z is the normalization factor. In fact, the posterior distribution is $\mathrm{Dir}(\boldsymbol{\alpha} + \mathbf{x})$.

A popular Bayesian model that explores such conjugacy is latent Dirichlet allocation (LDA) [48] as illustrated in Fig. 1a (All the figures are drawn by the authors with full copyright.). LDA posits that each document $\mathbf{w}_i$ is an admixture of a set of K topics, of which each topic $\boldsymbol{\psi}_k$ is a unigram distribution over a given vocabulary. The generative process is as follows:

(i) draw K topics $\boldsymbol{\psi}_k \sim \mathrm{Dir}(\boldsymbol{\beta})$,
(ii) for each document $i \in [N]$:
  (a) draw a topic mixing vector $\boldsymbol{\theta}_i \sim \mathrm{Dir}(\boldsymbol{\alpha})$,
  (b) for each word $j \in [L_i]$ in document i:
    (1) draw a topic assignment $z_{ij} \sim \mathrm{Multi}(\boldsymbol{\theta}_i)$,
    (2) draw a word $w_{ij} \sim \mathrm{Multi}(\boldsymbol{\psi}_{z_{ij}})$.

LDA has been popular in many applications. However, a conjugate prior can be restrictive. For example, the Dirichlet distribution does not impose correlation between different parameters, except the normalization constraint. In order to obtain more flexible models, a non-conjugate prior can be chosen.

Example 2. Logistic-normal prior. A logistic-normal distribution [57] provides one way to impose correlation structure among the multiple dimensions of $\boldsymbol{\theta}$. It is defined as follows:

$$\boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{\mu}\Sigma), \ \theta_k = \frac{e^{\eta_k}}{\sum_j e^{\eta_j}}. \qquad (13)$$

This prior has been used to develop correlated topic models (or logistic-normal topic models) [56] which can infer the correlation structure among topics. However, the flexibility pays cost on computation, needing scalable algorithms to learn large topic graphs [58].

BIG BAYESIAN LEARNING

Though much more emphasis in big Bayesian learning has been put on scalable algorithms and systems substantial advances have been made on 'adaptive' and 'flexible' Bayesian methods. This section reviews NPB methods for adaptively inferring model complexity and RegBayes for improving the flexibility via posterior regularization, while leaving the large part of scalable algorithms and systems to next sections.

NPB methods

For parametric Bayesian models, the parameter space is pre-specified. No matter how the data changes, the number of parameters is fixed. This restriction may cause limitations on model capacity, especially for Big Data applications, where it may be difficult or even counterproductive to fix the number of parameters a priori. For example, a Gaussian mixture model with a fixed number of clusters may fit the given data set well; however, it may be suboptimal to use the same number of clusters if more data comes under a slightly changed distribution. It would be ideal if the clustering models can figure out the unknown number of clusters automatically. Similar requirements on automatical model selection exist in feature representation learning [59] or factor analysis where we would like the models to automatically figure out the dimension of latent features (or factors) and maybe also the topological structure among features (or factors) at different abstraction levels [60].

NPB methods provide an elegant solution to such needs on automatic adaptation of model capacity when learning a single model. Such adaptivity is obtained by defining stochastic processes on rich measure spaces. Classical examples include Dirichlet process (DP) Indian buffet process (IBP) and Gaussian process (GP). Below we briefly review DP and IBP. We refer the readers to the articles [61–63] for a nice overview and the textbook [19] for a comprehensive treatment.
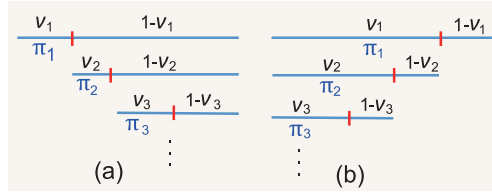
Figure 2. The stick-breaking process for: (a) DP; (b) IBP.

Dirichlet process

A DP defines the distribution of random measures. It was first developed in [64]. Specifically a DP is parameterized by a concentration parameter $\alpha > 0$ and a base distribution G_0 over a measure space Ω. A random variable drawn from a DP $G \sim \mathcal{DP}(\alpha, G_0)$, is itself a distribution over Ω. It was shown that the random distributions drawn from a DP are discrete almost surely, that is, they place the probability mass on a countably infinite collection of atoms, i.e.

$$G = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}, \qquad (14)$$

where θ_k is the value (or location) of the kth atom independently drawn from the base distribution G_0 and π_k is the probability assigned to the kth atom. Sethuraman [65] provided a constructive definition of π_k based on a stick-breaking process as illustrated in Fig. 2a. Consider a stick with unit length. We break the stick into an infinite number of segments π_k by the following process with $\nu_k \sim \text{Beta}(1, \alpha)$:

$$\pi_1 = \nu_1, \ \pi_k = \nu_k \prod_{j=1}^{k-1}(1 - \nu_j), \ k = 2, 3, \ldots, \infty.$$
$$(15)$$

That is, we first choose a beta variable ν_1 and break ν_1 of the stick. Then, for the remaining segment, we draw another beta variable and break off that proportion of the remainder of the stick. Such a representation of DP provides insights for developing variational approximate inference algorithms [66].

DP is closely related to the Chinese restaurant process (CRP) [67] which defines a distribution over infinite partitions of integers. CRP derives its name from a metaphor: Image a restaurant with an infinite number of tables and a sequence of customers entering the restaurant and sitting down. The first customer sits at the first table. For each of the subsequent customers she sits at each of the occupied tables with a probability proportional to the number of previous customers sitting there, and at the next unoccupied table with a probability proportional to α. In this process, the assignment of customers to tables defines a random partition. In fact,

if we repeatedly draw a set of samples from G, that is, $\theta_i \sim G$, $i \in [N]$, then it was shown that the joint distribution of $\theta_{1:N}$

$$p(\theta_1, \ldots, \theta_N | \alpha, G_0) = \int \left(\prod_{i=1}^{N} p(\theta_i | G) \right) \\ \times \, dP(G | \alpha, G_0)$$

exists a clustering property, that is, the θ_is will share repeated values with a non-zero probability. These shared values define a partition of the integers from 1 to N, and the distribution of this partition is a CRP with parameter α. Therefore, DP is the de Finetti mixing distribution of CRP.

Antoniak [68] first developed DP mixture models by adding a data generating step that is, $\mathbf{x}_i \sim p(\mathbf{x}|\theta_i)$, $i \in [N]$. Again, marginalizing out the random distribution G, the DP mixture reduces to a CRP mixture, which enjoys nice Gibbs sampling algorithms [69]. For DP mixtures a slice sampler [46] has been developed [70] which transforms the infinite sum in Eq. (14) into a finite sum conditioned on some uniformly distributed auxiliary variable.

Indian Buffet process

A mixture model assumes that each data is assigned to one single component. Latent factor models weaken this assumption by associating each data with some or all of the components. When the number of components is smaller than the feature dimension latent factor models provide dimensionality reduction. Popular examples include factor analysis, principal component analysis and independent component analysis. The general assumption of a latent factor model is that the observed data $\mathbf{x} \in \mathbb{R}^P$ is generated by a noisy weighted combination of latent factors, that is,

$$\mathbf{x}_i = W \mathbf{z}_i + \epsilon_i, \qquad (16)$$

where W is a $P \times K$ factor loading matrix, with element W_{mk} expressing how latent factor k influences the observation dimension m; $\mathbf{z}_i$ is a K-dimensional vector expressing the activity of each factor; and ϵ_i is a vector of independent noise terms (usually Gassian noise). In the above models, the number of factors K is assumed to be known. IBP [71] provides a NPB variant of latent factor models and it allows the number of factors to grow as more data are observed.

Consider binary factors for simplicity (Real-valued factors can be easily considered by defining $\mathbf{h}_i = \mathbf{z}_i \odot \boldsymbol{\mu}_i$ where the binary $\mathbf{z}_i$ are 0/1 masks to indicate whether a factor is active or not, and $\boldsymbol{\mu}_i$ are the values of the factors.). Putting the latent factors of N data points in a matrix Z, of which the ith row is

z_i. IBP defines a process over the space of binary matrixes with an unbounded number of columns. IBP derives its name from a similar metaphor as CRP. Image a buffet with an infinite number of dishes (factors) arranged in a line and a sequence of customers choosing the dishes. Let z_{ik} denote whether customer i chooses dish k. Then, the first customer chooses K_1 dishes, where $K_1 \sim \mathrm{Poisson}(\alpha)$; and the subsequent customer n (>1) chooses:

(i) each of the previously sampled dishes with probability m_k/n, where m_k is the number of customers who have chosen dish k;

(ii) K_i additional dishes, where $K_i \sim \mathrm{Poisson}(\alpha/n)$.

IBP plays the same role for latent factor models that CRP plays for mixture models, allowing an unbounded number of latent factors. Analogous to the role that DP is the de Finetti mixing distribution of CRP, the de Finetti mixing distribution underlying IBP is a Beta process [72]. IBP also admits a stick-breaking representation [73] as shown in Fig. 2b where the stick lengths are defined as:

$$\nu_k \sim \mathrm{Beta}(\alpha, 1), \ \pi_k = \prod_{j=1}^{k} \nu_j, \ k = 1, 2, \ldots, \infty.$$
(17)

Note that unlike the stick-breaking representation of DP, where the stick lengths sum to 1, the stick lengths here need not sum to 1. Such a representation has lead to the developments of MC [73] as well as variational approximation inference algorithms [74].

Gaussian process

Kernel machines (e.g. support vector machines) [75] represent an important class of methods in ML and has received extensive attention. GPs provide a principled practical probabilistic approach to learning in kernel machines. A GP is defined on the space of continuous functions [76]. In ML the prime use of GPs is to learn the unknown mapping function from inputs to outputs for supervised learning.

Take the simple linear regression model as an example. Let $\mathbf{x} \in \mathbb{R}^M$ be an input data point and $y \in \mathbb{R}$ be the output. A linear regression model is

$$f(\mathbf{x}) = \boldsymbol{\theta}^\top \phi(\mathbf{x}), \ y = f(\mathbf{x}) + \epsilon,$$

where $\phi(\mathbf{x})$ is a vector of features extracted from $\mathbf{x}$, and ϵ is an independent noise. For the Gaussian noise, e.g. $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, the likelihood of y conditioned on $\mathbf{x}$ is also a Gaussian, that is, $p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(f(\mathbf{x}), \sigma^2 I)$. Consider a Bayesian approach, where we put a zero-mean Gaussian prior,

$\boldsymbol{\theta} \sim \mathcal{N}(0, \Sigma)$. Given a set of training observations $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$. Let X be the $M \times N$ design matrix, and $\mathbf{y}$ be the vector of the targets. By Bayes' theorem, we can easily derive that the posterior is also a Gaussian distribution (see [76] for more details)

$$p(\boldsymbol{\theta}|X\mathbf{y}) = \mathcal{N}\left(\frac{1}{\sigma^2} A^{-1}\Phi\mathbf{y}, A^{-1}\right), \quad (18)$$

where $A^{-1} = \sigma^{-2}\Phi\Phi^\top + \Sigma^{-1}$ and $\Phi = \phi(X)$. For a test example $\mathbf{x}_*$, we can also derive that the distribution of the predictive value $f_* \triangleq f(\mathbf{x}_*)$ is also a Gaussian:

$$p(f_*|\mathbf{x}_*, X, \mathbf{y}) = \mathcal{N}\left(\frac{1}{\sigma^2}\phi_*^\top A^{-1}\Phi\mathbf{y}, \phi_*^\top A^{-1}\phi_*\right),$$
(19)

where $\phi_* \triangleq \phi(\mathbf{x}_*)$. In some equivalent form, the Gaussian mean and covariance only involve the inner products in input space. Therefore, the kernel trick can be explored in such models, which avoids the explicit evaluation of the feature vectors.

The above Bayesian linear regression model is a very simple example of GPs. In the most general form, GPs define a stochastic process over functions $f(\mathbf{x})$. A GP is characterized by a mean function $m(\mathbf{x})$ and a covariance function $\kappa(\mathbf{x}, \mathbf{x}')$, denoted by $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}'))$. Given any finite set of observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$, the function values (The function values are random variables due to the randomness of f.) $(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))$ follow a multivariate Gaussian distribution with mean $(m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n))$ and covariance $K : K(i, j) = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. The above definition with any finite collection of function values guarantee to define a stochastic process (i.e. GP), by examining the consistency requirement of the Kolmogorov extension theorem.

GPs have also been used in classification tasks, where the likelihood is often non-conjugate to the GP prior, therefore requiring approximate inference algorithms, including both variational and MC methods. Other research has considered GP LVM (GP-LVM) [77].

Extensions

To meet the flexibility and adaptivity requirements of Big Learning many recent advances have been made on developing sophisticated NPB methods for modeling various types of data, including grouped data, spatial data, time series and networks.

Hierarchical models are natural tools to describe grouped data, e.g. documents from different source domains. Hierarchical Dirichlet process (HDP) [78] and hierarchical Beta process [72] have

been developed allowing an infinite number of latent components to be shared by multiple domains. The work [60] presents a cascading IBP (CIBP) to learn the topological structure of multiple layers of latent features including the number of layers the number of hidden units at each layer, the connection structure between units at neighboring layers and the activation function of hidden units. The recent work [79] presents an extended CIBP process to generate connections between non-consecutive layers.

Another dimension of the extensions concerns modeling the dependencies between observations in a time series. For example DP has been used to develop the infinite hidden Markov models [80] which posit the same sequential structure as in the hidden Markov models, but allowing an infinite number of latent classes. In [78] it was shown that iHMM is a special case of HDP. The recent work [81] presents a max-margin training of iHMMs under the regularized Bayesian framework as will be reviewed shortly.

Finally, for spatial data, modeling dependency between nearby data points is important. Recent extensions of Bayesian non-parametric methods include the dependent DP [82] spatial DP [83] distance-dependent CRP [84] dependent IBP [85] and distance-dependent IBP [86]. For network data analysis (e.g. social networks biological networks and citation networks) recent extensions include the NPB relational latent feature models for link prediction [87,88] which adopt IBP to allow for an unbounded number of latent features and the non-parametric mixed membership stochastic block models for community discovery [89,90], which use HDP to allow mixed membership in an unbounded number of latent communities.

Regularized Bayesian inference

RegBayes [18] represents one recent advance that extends the scope of Bayesian methods on incorporating rich side information. Recall that the classic Bayes' theorem is equivalent to a variational optimization problem as in (2). RegBayes builds on this formulation and defines the generic optimization problem:

$$\min_{q(\boldsymbol{\Theta}) \in \mathcal{P}} \mathrm{KL}(q(\boldsymbol{\Theta}) \| p(\boldsymbol{\Theta}|\mathcal{D})) + c \cdot \Omega(q(\boldsymbol{\Theta}); \mathcal{D})$$

$$(20)$$

where $\Omega(q(\boldsymbol{\Theta}); \mathcal{D})$ is the 'posterior regularization' term; c is a non-negative regularization parameter; and $p(\boldsymbol{\Theta}|\mathcal{D})$ is the ordinary Bayesian posterior. Figure 3 provides a high-level comparison between

RegBayes and Bayes' rule. Several questions need to be answered in order to solve practical problems.

Q: How to define the posterior regularization?

A: In general, posterior regularization can be any informative constraints that are expected to regularize the properties of the posterior distribution. It can be defined as the large-margin constraints to enforce a good prediction accuracy [91] or the logic constraints to incorporate expert knowledge [92] or the sparsity constraints [93]. The very recent work [94] presents an extension of RegBayes to reproducing kernel Hilbert space (RKHS) which provides a flexible framework to put regularization at the distribution level.

Example 3. Max-margin LDA. Following the paradigm of ordinary Bayes a supervised topic model is often defined by augmenting the likelihood model. For example, the supervised LDA (sLDA) [95] has a similar structure as LDA (see Fig. 1c) but with an additional likelihood $p(y_d|\mathbf{z}_d, \boldsymbol{\eta})$ to describe labels. Such a design can lead to an imbalanced combination of the word likelihood $p(\mathbf{w}_d|\mathbf{z}_d, \boldsymbol{\psi})$ and the label likelihood because a document often has tens or hundreds of words while only one label. The imbalance problem causes unsatisfactory prediction results [96].

To improve the discriminative power of supervised topic models the max-margin MedLDA has been developed, under the RegBayes framework. Consider binary classification for simplicity. In this case, we have $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i, \mathbf{z}_i, \boldsymbol{\psi}_k\}$. Let $f(\boldsymbol{\eta}, \mathbf{z}_i) = \boldsymbol{\eta}^\top \bar{\mathbf{z}}_i$ be the discriminant function(We ignore the offset for simplicity.), where $\bar{\mathbf{z}}_i$ is the average topic assignments, with $\bar{z}_i^k = \frac{1}{L_i} \sum_j \mathbb{I}(z_{ij} = k)$. The posterior regularization can be defined in two ways:

Averaging classifier. An averaging classifier makes predictions using the expected discriminant function, that is, $\hat{y}(q) = \mathrm{sign}(\mathbb{E}_q[f(\boldsymbol{\eta}, \mathbf{z})])$. Let $(x)_+ = \max(0, x)$. Then, the posterior regularization

$$\Omega^{\mathrm{Avg}}(q(\boldsymbol{\Theta}); \mathcal{D}) = \sum_{i=1}^{N} (1 - y_i \mathbb{E}_q[f(\boldsymbol{\eta}, \mathbf{z}_i)])_+$$

is an upper bound of the training error, therefore a good surrogate loss for learning. This strategy has been adopted in MedLDA [91].

Gibbs classifier. A Gibbs classifier (or stochastic classifier) randomly draws a sample $(\boldsymbol{\eta}\mathbf{z}_d)$ from the target posterior $q(\boldsymbol{\Theta})$ and makes predictions using the latent prediction rule, that is, $\hat{y}(\boldsymbol{\eta}, \mathbf{z}_i) = \mathrm{sign} f(\boldsymbol{\eta}, \mathbf{z}_i)$. Then, the posterior regularization is defined as:

$$\Omega^{\mathrm{Gibbs}}(q(\boldsymbol{\Theta}); \mathcal{D}) = \mathbb{E}_q \left[\sum_{i=1}^{N} (1 - y_i f(\boldsymbol{\eta}, \mathbf{z}_i))_+ \right].$$
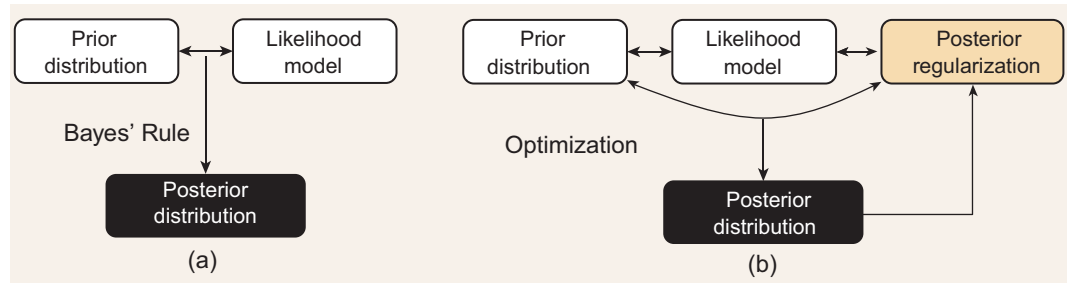
Figure 3. (a) Bayesian inference with the Bayes' rule; (b) RegBayes which solves an optimization problem with a posterior regularization term to incorporate rich side information.

This strategy has been adopted to develop Gibbs MedLDA [97].

The two strategies are closely related e.g. we can show that $\Omega^{\text{Gibbs}}(q(\boldsymbol{\Theta}))$ is an upper bound of $\Omega^{\text{Avg}}(q(\boldsymbol{\Theta}))$. The formulation with a Gibbs classifier can lead to a scalable Gibbs sampler by using data augmentation techniques [98]. If a logistic log-loss is adopted to define the posterior regularization an improved sLDA model can be developed to address the imbalance issue and lead to significantly more accurate predictions [96].

Q: What is the relationship between prior likelihood and posterior regularization?

A: Though the three parts are closely connected, there are some key differences. First, prior is chosen before seeing data, while both likelihood and posterior regularization depend on the data. Second, different from the likelihood, which is restricted to be a normalized distribution, no constraints are imposed on the posterior regularization. Therefore, posterior regularization is much more flexible than prior or likelihood. In fact, it can be shown that (i) putting constraints on priors is a special case of posterior regularization, where the regularization term does not depend on data; and (ii) RegBayes can be more flexible than standard Bayes' rule, that is, there exists some RegBayes posterior distributions that are not achievable by the Bayes' rule [18].

Q: How to solve the optimization problem?

A: The posterior regularization term affects the difficulty of solving problem (20). When the regularization term is a convex functional of $q(\boldsymbol{\Theta})$ which is common in many applications such as the above max-margin formulations, the optimal solution can be characterized in a general from via convex duality theory [18]. When the regularization term is non-convex a generalized representation theorem can also be derived, but requires more effects on dealing with the non-convexity [93].

SCALABLE ALGORITHMS

To deal with Big Data the posterior inference algorithms should be scalable. Significant advances have been made in two aspects: (i) using random sampling to do stochastic or online Bayesian inference; and (ii) using multi-core and multimachine architectures to do parallel and distributed Bayesian inference.

Stochastic algorithms

In Big Learning, the intriguing results of [99] suggest that an algorithm as simple as stochastic gradient descent (SGD) can be optimally efficient in terms of 'number of bits learned per unit of computation'. For Bayesian models both stochastic variational and stochastic MC methods have been developed to explore the redundancy of data relative to a model by subsampling data examples for every update and reasoning about the uncertainty created in this process [22]. We overview each type in turn.

Stochastic variational methods

As we have stated in Section Variational Bayesian methods variational methods solve an optimization problem to find the best approximate distribution to the target posterior. When the variational distribution is characterized in some parametric form, this problem can be solved with SGD methods [100] or the adaptive SGD [101]. A SGD method randomly draws a subset B_t and updates the variational parameters using the estimated gradients that is

$$\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t + \epsilon_t \left(\nabla_{\boldsymbol{\phi}} \text{KL}(q \| p_0(\boldsymbol{\theta})) \right.$$
$$\left. - \nabla_{\boldsymbol{\phi}} \mathbb{E}_q [\log p(\mathcal{D}|\boldsymbol{\theta})] \right),$$

where the full data gradient is approximated as

$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_q [\log p(\mathcal{D}|\boldsymbol{\theta})] \approx \frac{N}{|B_t|} \sum_{i \in B_t} \nabla_{\boldsymbol{\phi}} \mathbb{E}_q [\log p(\mathbf{x}_i|\boldsymbol{\theta})],$$

(21)

and ϵ_t is a learning rate. If the noisy gradient is an unbiased estimate of the true gradient, the procedure is guaranteed to approach the optimal solution when the learning rate is appropriately set [102].

For Bayesian LVM we need to infer the latent variables when performing the updates. In general, we can group the latent variables into two categories—global variables and local variables. Global variables correspond to the model parameters $\boldsymbol{\theta}$ (e.g. the topics $\boldsymbol{\psi}$ in LDA), while local variables represent some hidden structures of the data (e.g. the topic assignments $\mathbf{z}$ in an LDA with the topic mixing proportions collapsed out). Figure 4 provides an illustration of such models and the stochastic variational inference, which consists of three steps:

(i) randomly draw a mini-batch B_t of data samples;
(ii) infer the local latent variables for each data in B_t;
(iii) update the global variables.

However, the standard gradients over the parameters $\boldsymbol{\phi}$ may not be the most informative direction (i.e. the steepest direction) to search for the distribution q. A better way is to use natural gradient [103] which is the steepest search direction in a Riemannian manifold space of probability distributions [104]. To reduce the efforts on hand-tuning the learning rate which often influences the performance much, the work [105] presents an adaptive learning rate while [106] adopts Bayesian optimization to search for good learning rates both leading to faster convergence. By borrowing the gradient averaging ideas from stochastic optimization [107] proposes to use smoothed gradients in stochastic variational inference to reduce the variance (by trading-off the bias). Stochastic variational inference methods have been studied for many Bayesian models such as LDA and HDP [104].

In many cases the ELBO and its gradient may be intractable to compute due to the intractability of the expectation over variational distributions. Two types of methods are commonly used to address this problem. First, another layer of variational bound is derived by introducing additional variational parameters. This has been used in many examples, such as the logistic-normal topic models [56] and sLDA [95]. For such methods it is important to develop tight variational bounds for specific models [108] which is still an active area. Another type of methods is to use MC estimates of the variational bound as well as its gradients. Recent work includes the stochastic approximation scheme with variance reduction [105,109] and the auto-encoding variational Bayes (AEVB) [110] that learns a neural network (a.k.a recognition model) to represent the variational distribution for continuous latent variables.

Consider the model with one layer of continuous latent variables $\mathbf{h}_i$ in Fig. 4a. Assume the variational distribution $q_{\boldsymbol{\phi}}(\boldsymbol{\Theta}) = q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \prod_{i=1}^{N} q_{\boldsymbol{\phi}}(\mathbf{h}_i|\mathbf{x}_i)$. Let $G_{\boldsymbol{\phi}}(\mathbf{xh}\boldsymbol{\theta}) = \log p(\mathbf{h}|\boldsymbol{\theta}) + \log p(\mathbf{x}|\mathbf{h}\boldsymbol{\theta}) - \log q_{\boldsymbol{\phi}}(\mathbf{h}|\mathbf{x})$. The ELBO in Eq. (2) can be written as

$$\mathcal{L}(\boldsymbol{\phi}; \mathcal{D}) = \mathbb{E}_q \Big[\log p_0(\boldsymbol{\theta}) + \sum_i G_{\boldsymbol{\phi}}(\mathbf{x}_i, \mathbf{h}_i, \boldsymbol{\theta}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \Big].$$

By using the equality $\nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\boldsymbol{\Theta}) = q_{\boldsymbol{\phi}}(\boldsymbol{\Theta}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\Theta})$, it can be shown that the gradient is

$$\nabla_{\boldsymbol{\phi}} \mathcal{L} = \mathbb{E}_q \Big[(\log p(\boldsymbol{\Theta}, \mathcal{D}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\Theta})) \times \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\Theta}) \Big].$$

A naive MC estimate of the gradient is

$$\nabla_{\boldsymbol{\phi}} \mathcal{L} \approx \frac{1}{L} \sum_{l=1}^{L} \Big[(\log p(\boldsymbol{\Theta}^l, \mathcal{D}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\Theta}^l)) \times \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\Theta}^l) \Big],$$

where $\boldsymbol{\Theta}^l \sim q_{\boldsymbol{\phi}}(\boldsymbol{\Theta})$. Note that the sampling and the gradient $\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\Theta}^l)$ only depend on the variational distribution, not the underlying model. However, the variance of such an estimate can be too large to be useful. In practice, effective variance reduction techniques are needed [105,109].

For continuous $\mathbf{h}$, a re-parameterization of the samples $\mathbf{h} \sim q_{\boldsymbol{\phi}}(\mathbf{h}|\mathbf{x})$ can be derived using a differentiable transformation $g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$ of a noise variable $\boldsymbol{\epsilon}$:

$$\mathbf{h} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}), \text{ where } \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}). \quad (22)$$

This is known as 'non-centered parameterization' (NCP) in statistics [111] while the original representation is known as 'centered parameterization' (CP). A similar NCP reparameterization exists for the continuous $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = f_{\boldsymbol{\phi}}(\boldsymbol{\zeta}), \text{ where } \boldsymbol{\zeta} \sim p(\boldsymbol{\zeta}). \quad (23)$$

Given a minibatch of data points B_t, we define $F_{\boldsymbol{\phi}}(\{\mathbf{x}_i, \mathbf{h}_i\}_{i \in B_t}, \boldsymbol{\theta}) = \frac{N}{|B_t|} \sum_{i \in B_t} G_{\boldsymbol{\phi}}(\mathbf{x}_i, \mathbf{h}_i, \boldsymbol{\theta}) + \log p_0(\boldsymbol{\theta}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$. Then, the MC estimate of the variational lower bound is

$$\mathcal{L}(\boldsymbol{\phi}; \mathcal{D}) \approx \frac{1}{L} \sum_{l=1}^{L} F_{\boldsymbol{\phi}} \left(\{\mathbf{x}_i, g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^l, \mathbf{x}_i)\}_{i \in B_t}, f_{\boldsymbol{\phi}}(\boldsymbol{\zeta}^l) \right),$$
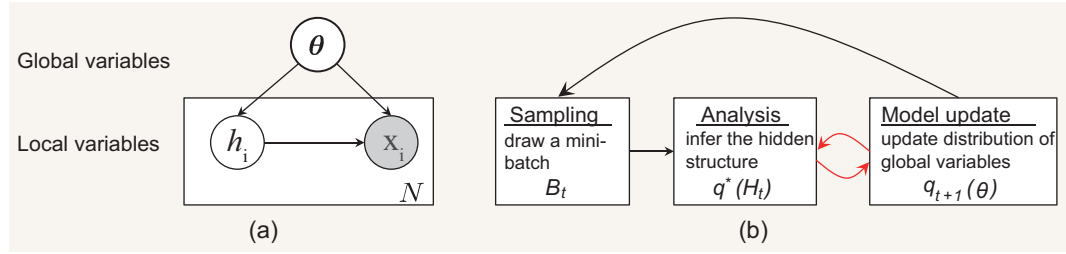
$$(24)$$

Figure 4. (a) The general structure of Bayesian LVM, where $\boldsymbol{h}_i$ denotes the local latent variables for each data i; (b) the process of stochastic variational inference, where the red arrows denote that in practice we may need multiple iterations between 'analysis' and 'model update' to have fast convergence.

where $\boldsymbol{\epsilon}^l \sim p(\boldsymbol{\epsilon})$ and $\boldsymbol{\zeta}^l \sim p(\boldsymbol{\zeta})$. This stochastic estimate can be maximized via gradient ascent methods.

It has been analyzed that CP and NCP possess complimentary strengths [111] in the sense that NCP is likely to work when CP does not and conversely. An accompany paper [112] to AEVB analyzes the conditions for gradient-based samplers (e.g. HMC) whether NCP can be effective or ineffective in reducing posterior dependencies; and it suggests to use the interleaving strategy between centered and NCP as previously studied in [113]. AEVB has been extended to learn deep generative models [16] using the similar reparameterization trick on continuous latent variables. However AEVB cannot be directly applied to deal with discrete variables. In contrast the work [114] presents a sophisticated method to reduce the variance of the naive MC estimate for deep autoregressive models; thus it is applicable to both continuous and discrete latent variables.

Stochastic MC Methods

The existing stochastic MC methods can be generally grouped into three categories namely, stochastic gradient-based methods, the methods using approximate MH test with randomly sampled mini-batches, and data augmentation.

Stochastic Gradient. The idea of using gradient information to improve the mixing rates has been systematically studied in various MC methods, including Langevin dynamics and Hamiltanian dynamics [41]. For example the Langevin dynamics is an MCMC method that produces samples from the posterior by means of gradient updates plus Gaussian noise, resulting in a proposal distribution $p(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t)$ by the following equation:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \frac{\epsilon_t}{2}(\nabla_{\boldsymbol{\theta}} \log p_0(\boldsymbol{\theta})$$
$$+ \nabla_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta})) + \zeta_t, \quad (25)$$

where $\zeta_t \sim \mathcal{N}(0, \epsilon_t I)$ is an isotropic Gaussian noise and $\log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_i \log p(\mathbf{x}_i|\boldsymbol{\theta})$ is the log-likelihood of the full data set. The mean of the proposal distribution is in the direction of increasing log posterior due to the gradient, while the Gaussian noise will prevent the samples from collapsing to a single maximum. A Metropolis-Hastings correction step is required to correct for discretization error [115].

The stochastic ideas have been successfully explored in these methods to develop efficient stochastic MC methods including stochastic gradient Langevin dynamics (SGLD) [116] and stochastic gradient Hamiltonian dynamics (SGHD) [117]. For example SGLD replaces the calculation of the gradient over the full data set with a stochastic approximation based on a subset of data. Let B_t be the subset of data points uniformly sampled from the full data set at iteration t. Then the gradient is approximated as:

$$\nabla_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta}) \approx \frac{N}{|B_t|} \sum_{i \in B_t} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}_i|\boldsymbol{\theta}).$$

$$(26)$$

Note that SGLD does not use an MH correction step, as calculating the acceptance probability requires use of the full data set. Convergence to the posterior is still guaranteed if the step sizes are annealed to zero at a certain rate, as rigorously justified in [118,119].

To further improve the mixing rates, the stochastic gradient Fisher scoring method [120] was developed which represents an extension of the Fisher scoring method based on stochastic gradients [121] by incorporating randomness in a subsampling process. Similarly exploring the manifold structure leads to the development of stochastic gradient MCMC methods on Riemann manifolds, such as stochastic gradient Riemannian Langevin dynamics [122] on the probability simplex space and stochastic gradient geodesic Monte Carlo [123] on manifolds with

known geodesic flow which can be framed under a general framework [124].

Approximate MH Test. Another category of stochastic MC methods rely on approximate MH test using randomly sampled subset of data points since an exact calculation of the MH test in Eq. (10) scales linearly to the data size which is prohibitive for large-scale data sets. For example, the work [125] presents an approximate MH rule via sequential hypothesis testing which allows us to accept or reject samples with high confidence using only a fraction of the data required for the exact MH rule. The systematic bias and its tradeoff with variance were theoretically analyzed. Specifically, it is based on the observation that the MH test rule in Eq. (10) can be equivalently written as follows.

(i) Draw $\gamma \sim \text{Uniform}[0, 1]$ and compute:

$$\mu_0 = \frac{1}{N} \log \left[\gamma \frac{p_0(\boldsymbol{\Theta}_t) q(\boldsymbol{\Theta}'|\boldsymbol{\Theta}_t)}{p_0(\boldsymbol{\Theta}') q(\boldsymbol{\Theta}_t|\boldsymbol{\Theta}')} \right]$$

$$\mu = \frac{1}{N} \sum_{i=1}^{N} \ell_i, \text{ where } \ell_i = \log \frac{p(\mathbf{x}_i|\boldsymbol{\Theta}')}{p(\mathbf{x}_i|\boldsymbol{\Theta}_t)};$$

(ii) If $\mu > \mu_0$ set $\boldsymbol{\Theta}_{t+1} \leftarrow \boldsymbol{\Theta}'$; otherwise $\boldsymbol{\Theta}_{t+1} \leftarrow \boldsymbol{\Theta}_t$.

Note that μ_0 is independent of the data set, thus can be easily calculated. This reformulation of the MH test makes it very easy to frame it as a statistical hypothesis test, that is, given μ_0 and a set of samples $\{\ell_{t_1}, \ldots, \ell_{t_n}\}$ drawn without replacement from the population $\{\ell_1, \ldots, \ell_N\}$, can we decide whether the population mean μ is greater than or less than the threshold μ_0? Such a test can be done by increasing the cardinality of the subset until a prescribed confidence level is reached. The MH test with approximate confidence intervals can be combined with the above stochastic gradient methods (e.g. SGLD) to correct their bias. The similar sequential testing ideas can be applied to Gibbs sampling, as discussed in [125].

Under the similar setting of approximate MH test with subsets of data the work [126] derives a new stopping rule based on some concentration bounds (e.g. the empirical Bernstein bound [127]) which leads to an adaptive sampling strategy with theoretical guarantees on the total variational norm between the approximate MH kernel and the target distribution of MH applied to the full data set.

Data Augmentation. The work [128] presents a Firefly Monte Carlo (FlyMC) method which is guaranteed to converge to the true target posterior. FlyMC relies on a novel data augmentation formulation [45]. Specifically let z_i be a binary variable indicating whether data i is active or not and $B_i(\boldsymbol{\Theta})$ be

a strictly positive lower bound of the ith likelihood: $0 < B_i(\boldsymbol{\Theta}) < L_i(\boldsymbol{\Theta}) \triangleq p(\mathbf{x}_i|\boldsymbol{\Theta})$. Then, the target posterior $p(\boldsymbol{\Theta}|\mathcal{D})$ is the marginal of the complete posterior with the augmented variables $\mathbf{Z} = \{z_i\}_{i=1}^{N}$:

$$p(\boldsymbol{\Theta}, \mathbf{Z}|\mathcal{D}) \propto p_0(\boldsymbol{\Theta}) \prod_{i=1}^{N} p(\mathbf{x}_i|\boldsymbol{\Theta}) p(z_i|\mathbf{x}_i, \boldsymbol{\Theta}), \tag{27}$$

where $p(z_i|\mathbf{x}_i, \boldsymbol{\Theta}) = (1 - \gamma_i)^{z_i} \gamma_i^{(1-z_i)}$ and $\gamma_i = B_i(\boldsymbol{\Theta})/L_i(\boldsymbol{\Theta})$. Then, we can construct a Markov chain for the complete posterior by alternating between updates of $\boldsymbol{\Theta}$ conditioned on $\mathbf{Z}$, which can be done with any conventional MCMC algorithm, and updates of $\mathbf{Z}$ conditioned on $\boldsymbol{\Theta}$, which can also been efficiently done as we only need to re-calculate the likelihoods of the data points with active z variables, thus effectively using a random subset of data points in each iteration of the MC methods.

Streaming algorithms

We can see that both (21) and (26) need to know the data size N, which renders them unsuitable for learning with streaming data, where data comes in small batches without an explicit bound on the total number as times goes along, e.g. tracking an aircraft using radar measurements. This conflicts with the sequential nature of the Bayesian updating procedure. Specifically, let B_t be the small batch at time t. Given the posterior at time t, $p_t(\boldsymbol{\Theta}) := p(\boldsymbol{\Theta}|B_1, \ldots, B_t)$, the posterior distribution at time $t + 1$ is

$$\begin{aligned} p_{t+1}(\boldsymbol{\Theta}) &:= p(\boldsymbol{\Theta}|B_1, \ldots, B_{t+1}) \\ &= \frac{p_t(\boldsymbol{\Theta}) p(B_{t+1}|\boldsymbol{\Theta})}{p(B_1, \ldots, B_{t+1})}. \end{aligned} \tag{28}$$

In other words, the posterior at time t is actually playing the role of a prior for the data at time $t + 1$ for the Bayesian updating. Under the variational formulation of Bayes' rule, streaming RegBayes [129] can naturally be defined as solving:

$$\min_{q(\boldsymbol{\Theta}) \in \mathcal{P}} \text{KL}(q(\boldsymbol{\Theta}) \| p_t(\boldsymbol{\Theta})) + c \cdot \Omega(q(\boldsymbol{\Theta}); B_{t+1}) \tag{29}$$

whose streaming update rule can be derived via convex analysis under a quite general setting.

The sequential updating procedure is perfectly suitable for online learning with data streams, where a revisit to each data point is not allowed.

However, one challenge remains on evaluating the posteriors. If the prior is conjugate to the likelihood model (e.g. a linear Gaussian state-space model) or the state space is discrete (e.g. hidden Markov models [130,131]), then the sequential updating rule can be done analytically, for example, Kalman filters [132]. In contrast many complex Bayesian models (e.g. the models involving non-Gaussianity, non-linearity and high dimensionality) do not have closed-form expression of the posteriors. Therefore, it is computationally intractable to do the sequential update.

Streaming variational methods

Various effects have been made to develop streaming variational Bayesian methods [133]. Specifically let $\mathcal{A}$ be a variational algorithm that calculates the approximate posterior q: $q(\boldsymbol{\Theta}) = \mathcal{A}(p(\Theta); B)$. Then, setting $q_0(\boldsymbol{\Theta}) = p_0(\boldsymbol{\Theta})$, one way to recursively compute an approximation to the posterior is

$$p(\boldsymbol{\Theta}|B_1, \ldots, B_{t+1}) \approx q_{t+1}(\boldsymbol{\Theta})$$
$$= \mathcal{A}(q_t(\boldsymbol{\Theta}), B_{t+1}). \quad (30)$$

Under the exponential family assumption of q, the streaming update rule has some analytical form. The streaming RegBayes [129] provides a Bayesian generalization of online passive-aggressive (PA) learning [134] when the posterior regularization term is defined via the max-margin principle. The resulting online Bayesian passive-aggressive (BayesPA) learning adopts a similar streaming variational update to learn max-margin classifiers (e.g. SVMs) in the presence of latent structures (e.g. latent topic representations). Compared to the ordinary PA BayesPA is more flexible on modeling complex data. For example BayesPA can discover latent structures via a hierarchical Bayesian treatment as well as allowing for NPB inference to resolve the complexity of latent components (e.g. using a HDP topic model to resolve the unknown number of topics).

Streaming MC methods

Sequential Monte Carlo (SMC) methods [135–137] provide simulation-based methods to approximate the posteriors for online Bayesian inference. SMC methods rely on resampling and propagating samples over time with a large number of particles. A standard SMC method would require the full data to be stored for expensive particle rejuvenation to protect particles against degeneracy leading to an increased storage and processing bottleneck as more data are accrued. For simple conjugate models, such as linear Gaussian state-space models,

efficient updating equations can be derived using methods like Kalman filters. For a broader class of models, assumed density filtering (ADF) [138,139] was developed to extend the computational tractability. Basically, ADF approximates the posterior distribution with a simple conjugate family, leading to approximate online posterior tracking. Recent improvements on SMC methods include the conditional density filtering (C-DF) method [140] which extends Gibbs sampling to streaming data. C-DF sequentially draws samples from an approximate posterior distribution conditioned on surrogate conditional sufficient statistics, which are approximations to the conditional sufficient statistics using sequential samples or point estimates for parameters along with the data. C-DF requires only data at the current time and produces a provably good approximation to the target posterior.

Distributed algorithms

Recent progress has been made on both distributed variational and distributed MC methods.

Distributed variational methods

If the variational distribution is in some parametric family (e.g. the exponential family), the variational problem can be solved with generic optimization methods. Therefore, the broad literature on distributed optimization [141] provides rich tools for distributed variational inference. However the disadvantage of a generic solver is that it may fail to explore the structure of Bayesian inference.

First, many Bayesian models have a nature hierarchy, which encodes rich CI structures that can be explored for efficient algorithms, e.g. the distributed variational algorithm for LDA [142]. Second the inference procedure with Bayes' rule is intrinsically parallelizable. Suppose the data $\mathcal{D}$ is split into non-overlapping batches (often called shards), $B_1, \ldots, B_M$. Then, the Bayes posterior $p(\boldsymbol{\Theta}|\mathcal{D}) = \frac{p_0(\boldsymbol{\Theta}) \prod_{i=1}^{M} p(B_i|\boldsymbol{\Theta})}{p(\mathcal{D})}$ can be expressed as

$$p(\boldsymbol{\Theta}|\mathcal{D}) = \frac{1}{C} \prod_{i=1}^{M} \frac{p_0(\boldsymbol{\Theta})^{\frac{1}{M}} p(B_i|\boldsymbol{\Theta})}{p(B_i)}$$
$$= \frac{1}{C} \prod_{i=1}^{M} p(\boldsymbol{\Theta}|B_i), \quad (31)$$

where $C = \frac{p(\mathcal{D})}{\prod_{i=1}^{M} p(B_i)}$. Now, the question is how to calculate the local posteriors (or subset posteriors) $p(\boldsymbol{\Theta}|B_i)$ as well as the normalization factor. The work [133] explores this idea and presents a distributed variational Bayesian method which

approximates the local posterior with an algorithm $\mathcal{A}$ that is, $p(\mathbf{\Theta}|B_i) \approx \mathcal{A}\left(p_0(\mathbf{\Theta})^{1/M}, B_i\right)$. Under the exponential family assumption of the prior and the approximate local posteriors, the global posterior can be (approximately) calculated via density product. However, the parametric assumptions may not be reasonable, and the mean-field assumptions can get the marginal distributions right but not the joint distribution.

Distributed MC methods

For MC methods, if independent samples can be directly drawn from the posterior or some proposals (e.g. using importance sampling), it will be straightforward to parallelize, e.g. by running multiple independent samplers on separate machines and then aggregating the samples [143]. We consider the more challenging cases where directly sampling from the posterior is intractable and MCMC methods are among the natural choices. There are two groups of methods. One is to run multiple MCMC chains in parallel, and the other is to parallelize a single MCMC chain. The 'multiple-chain' parallelism is relatively straightforward if each single chain can be efficiently carried out and an appropriate combination strategy is adopted [143,144]. However, in Big data applications a single Markov chain itself is often prohibitively slow to converge, due to the massive data sizes or extremely high-dimensional sample spaces. Below, we focus on the methods that parallelize a single Markov chain, under three major categories.

Blocking. Methods in this category let each computing unit (e.g. a CPU processor or a graphics processing unit (GPU) core) to perform a part of the computation at each iteration. For example, they independently evaluate the likelihood for each shard across multiple units and combine the local likelihoods with the prior on a master unit to get estimates of the global posterior [145]. Another example is that each computing unit is responsible for updating a part of the state space [146]. These methods involve extensive communications and being problem specific.

In these methods several computing units collaborate to obtain a draw from the posterior. In order to effectively split the likelihood evaluation or the state space update over multiple computing units it is important to explore the CI structure of the model. Many hierarchical Bayesian models naturally have the CI structure (e.g. topic models), while some other models need some transformation to introduce CI structures that are appropriate for parallelization [147].

Divide-and-Conquer: Methods in this category avoid extensive communication among machines by
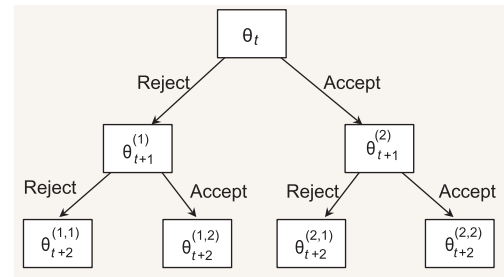

Figure 5. The possible outcomes in two iterations of a Metropolis-Hastings sampler.

running independent MCMC chains on each shard and aggregating samples drawn from local posteriors via a single communication. Aggregating the local samples is the key step with a lot of recent progress. For example, the consensus MC [148] directly combines local samples by a weighted average which is valid under an implicit Gaussian assumption while lacking of guarantees for non-Gaussian cases; [149] approximates each local posterior with either an explicit Gaussian or a Gaussian-kernel KDE so that combination follows an explicit density product; [150] builds upon the KDE idea one step further by representing the discrete KDE as a continuous Weierstrass transform; and [151] proposes to calculate the geometric median of local posteriors (or M-posterior) which is provably robust to the presence of outliers. The M-posterior is approximately solved by the Weiszfeld's algorithm [152] by embedding the local posteriors in a RKHS.

The potential drawback of these embarrassingly parallel MCMC sampling is that if the local posteriors differ significantly perhaps due to noise or non-random partitioning of the dataset across nodes the final combination stage can result in inaccurate global posterior. The recent work [153] presents a context aware distributed Bayesian posterior sampling method to improve inference quality. By allowing nodes to effectively and efficiently share information with each other each node will eventually draw samples from a more accurate approximate full posterior, and therefore no long needs any combination.

Pre-fetching: The idea of pre-fetching is to make use of parallel processing to calculate multiple likelihoods ahead of time, and only use the ones which are needed. Consider a generic random-walk metropolis-Hastings algorithm at time *t*. The subsequent steps can be represented by a binary tree, where at each iteration a single new proposal is drawn from a proposal distribution and stochastically accepted or rejected. So, at time $t + n$ the chain has 2^n possible future states, as illustrated in Fig. 5. The vanilla version of pre-fetching speculatively

evaluates all paths in this binary tree [154]. Since only one path of these will be taken with M cores, this approach achieves a speedup of $\log_2 M$ with respect to single core execution, ignoring communication overheads. More efficient pre-fetching approaches have been proposed in [155] and [156] by better guessing the probabilities of exploration of both the acceptance and the rejection branches at each node. The recent work [157] presents a delayed acceptance strategy for MH testing which can be used to improve the efficiency of pre-fetching.

As a special type of MCMC Gibbs sampling methods naturally follow a blocking scheme by iterating over some partition of the variables. The early asynchronous Gibbs sampler [40] is highly parallel by sampling all variables simultaneously on separate processors. However the extreme parallelism comes at a cost, e.g. the sampler may not converge to the correct stationary distribution in some cases [158]. The work [158] develops various variable partitioning strategies to achieve fast parallelization while maintaining the convergence to the target posterior and the work [159] analyzes the convergence and correctness of the asynchronous Gibbs sampler (a.k.a the Hogwild parallel Gibbs sampler) for sampling from Gaussian distributions. Many other parallel Gibbs sampling algorithms have been developed for specific models. For example various distributed Gibbs samplers [160–164] have been developed for the vanilla LDA, [58] develops a distributed Gibbs sampler via data augmentation to learn large-scale topic graphs with a logistic-normal topic model and parallel algorithms for DP mixtures have been developed by introducing auxiliary variables for additional CI structures [147] while with the potential risk of causing extremely imbalanced partitions [165].

Note that the stochastic methods and distributed computing are not exclusive. Combing both often leads to more efficient solutions. For example for optimization methods parallel SGD methods have been extensively studied [166,167]. In particular, [167] presents a parallel SGD algorithm without locks called Hogwild!, where multiple processors are allowed equal access to the shared memory and are able to update individual components of memory at will. Such a scheme is particularly suitable for sparse learning problems. For Bayesian methods, the distributed stochastic gradient Langevin dynamics method has been developed in [168] and further improved for topic models in [169].

TOOLS SOFTWARE AND SYSTEMS

Though stochastic algorithms are easy to implement distributed methods often need a careful design of the system architectures and programming libraries. For system architectures, we may have a shared memory computer with many cores, a cluster with many machines interconnected by network (either commodity or high speed), or accelerating hardware like GPUs and field-programmable gate arrays (FPGAs). We now review the distributed programming frameworks suitable for various system architectures and existing tools for Bayesian inference.

System primitives

Every architecture has its low-level libraries, in which the parallel computing units (e.g. threads, machines or GPU cores) are explicitly visible to the programmer.

Shared memory computer. A shared memory computer passes data from one CPU core to another by simply storing it into the main memory. Therefore, the communication latency is low. It is also easy to program and acquire. Meanwhile it is prevalent—it is the basic component of large distributed clusters and host of GPUs or other accelerating hardware. Due to these reasons, writing a multithread program is usually the first step towards large-scale learning. However, its drawbacks include limited memory/IO capacity and bandwidth, and restricted scalability, which can be addressed by distributed clusters.

Programmers work with threads in a shared memory setting. A threading library supports: (i) spawning a thread and wait it to complete; (ii) synchronization: method to prevent conflict access of resources, such as locks; (iii) atomic: operations, such as increment that can be executed in parallel safely. Besides threads and locks, there are alternative programming frameworks. For example, Scala uses 'actor', which responds to a message that it receives; Go uses 'channel', which is a multiprovider, multiconsumer queue. There are also libraries automating specific parallel pattern, e.g. OpenMP [170] supports parallel patterns like parallel for or reduction and synchronization patterns like barrier; TBB [171] has pipeline lightweight green threads and concurrent data structures. Choosing right programming models sometimes can simplify the implementation.

Accelerating hardware. GPUs are self-contained parallel computational devices that can be housed in desktop or laptop computers. A single GPU can provide floating operations per second performance as good as a small cluster. Yet compared to conventional multi-core processors, GPUs are cheap, easily accessible, easy to maintain, easy to code, and dedicated local devices with low power consumption. GPUs follow a single instruction multiple data

(SIMD) pattern, i.e. a single program will be executed on all cores given different data. This pattern is suitable for many ML applications. However, GPUs may be limited due to: (i) small memory capacity; (ii) restricted SIMD programming model; and (iii) high CPU–GPU or GPU–GPU communication latency.

Many Bayesian inference methods have been accelerated with GPUs. For example, [145] adopts GPUs to parallelize the likelihood evaluation in MCMC; [172] provides GPU parallelization for population-based MCMC methods [42] as well as SMC samplers [173]; and [174] uses GPU computing to develop fast Hamiltonian MC methods. For variational Bayesian methods [175] demonstrates an example of using GPUs to accelerate the collapsed variational Bayesian algorithm for LDA. More recently SaberLDA [176] implements a sparsity-aware sampling algorithm on GPU which scales sub-linearly with the number of topics. BID-Mach [177] is a distributed GPU framework for ML In particular BIDMach LDA with a single GPU is able to learn faster than the state-of-the-art CPU based LDA implementation [162] which use 100 CPUs.

Finally acceleration with other hardware (e.g. FP-GAs) has also been investigated [178].

Distributed cluster. For distributed clusters a low-level framework should allow users to do: (i) 'Communication': sending and receiving data from/to another machine or a group of machines; (ii) 'Synchronization': synchronize the processes; (iii) 'Fault handling': decide what to do if a process/machine breaks down. For example, MPI provides a set of primitives including 'send', 'receive', 'broadcast' and 'reduce' for communication. MPI also provides synchronization operations, such as 'barrier'. MPI handles fault by simply terminating all processes. MPI works on various network infrastructures, such as ethernet or Infiniband. Besides MPI, there are other frameworks that support communication, synchronization and fault handling, such as (i) message queues, where processes can put and get messages from globally shared message queues; (ii) remote procedural calls, where a process can invoke a procedure in another process, passing its own data to that remote procedure, and finally get execution results. MrBayes [179,180] provides a MPI-based parallel algorithm for Metropolis-coupled MCMC for Bayesian phylogenetic inference.

Programming with system primitive libraries are most flexible and lightweight. However for sophisticated applications, which may require asynchronous execution, need to modify the global parameters while running, or need many parallel execution blocks, it would be painful and error prone to write the parallel code using the low-level system primitives. Below, we review some high-level distributed computing frameworks, which automatically execute the user declared tasks on desired architectures. We refer the readers to [181] for more details on GPUs MapReduce and some other examples (e.g. parallel online learning).

MapReduce and Spark

MapReduce [182] is a distributed computing framework for key-value stores. It reads key-value stores from disk performs some transformations to these key-value stores in parallel, and writes the final results to disk. A typical MapReduce cycle involves the steps: (i) Spawn some workers on all machines; (ii) Workers read input key-value pairs in parallel from a distributed file system; (iii) 'Map': pass each key-value pair to a user defined function, which will generate some intermediate key-value pairs; (iv) According to the key, hash the intermediate key-value pairs to all machines, then merge key-value pairs that have the same key, result with (key, list of values) pairs; (5) 'Reduce': In parallel, pass each (key, list of values) pairs to a user defined function, which will generate some output key-value pairs; and (6) Write output key-value pairs to the file system.

There are two user defined functions, 'mapper' and 'reducer'. For ML, a key-value store is often data samples, mapper is often used for computing latent variables, likelihoods or gradients for each data sample, and reducer is often used to aggregate the information from each data sample, where the information can be used for estimating parameters or checking convergence. [183] discusses a number of ML algorithms on MapReduce including linear regression, naive Bayes, neural networks, PCA, SVM, etc. Mahout [184] is a ML package built upon Hadoop an open source implementation of MapReduce. Mahout provides collaborative filtering, classification, clustering, dimensionality reduction and topic modeling algorithms. [142] is a MapReduce based LDA. However a major drawback of MapReduce is that it needs to read the data from disk at 'every iteration'. The overhead of reading data becomes dominant for many iterative ML algorithms as well as interactive data analysis tools [185].

Spark [185] is another framework for distributed ML methods that involve iterative jobs. The core of Spark is resilient data sets (RDDs) which is essentially a dataset distributed across machines. RDD can be stored either in memory or disk: Spark decides it automatically and users can provide hints to Spark which to store in memory. This avoids reading the dataset at every iteration. Users can perform
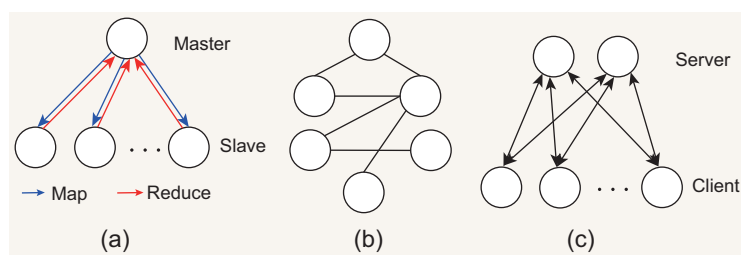
Figure 6. Various architectures: (a) MapReduce/Spark; (b) Pregel/GraphLab; (c) Parameter servers.

'parallel operations' to RDDs, which will transform a RDD to another. Available parallel operations are like 'foreach' and 'reduce'. We can use foreach to do the computation for each data, and use reduce to aggregate information from data. Because parallel operations are just a parallel version of the corresponding serial operations, a Spark program looks almost identical to its serial counterpart. Spark can outperform Hadoop for iterative ML jobs by 10x, and is able to interactively query a 39 GB dataset in 1 second [185].

Iterative graph computing

Both MapReduce and Spark have a star architecture as in Fig. 6a where only master-slave communication is permitted; they do not allow one key-value pair to interact with another, e.g. reading or modifying the value of another key-value pair. The interaction is necessary for applications like PageRank, Gibbs sampling, and variational Bayes optimized by coordinate descent, all of which require variables to get their own values based on other related variables. Hence there comes graph computing, where the computational task is defined by a sparse graph that specifies the data dependency, as shown in Fig. 6b.

Pregel [186] is a bulk synchronous parallel (BSP) graph computing engine. The computation model is a sparse graph with data on vertices and edges where each vertex receives all messages sent to it in the last iteration; updates data on the vertex based on the messages; and sends out messages along adjacent edges. For example, Gibbs sampling can be done easily by sending the vertex statistics to adjacent vertices and then the conditional probability can be computed. GPS [187] is an open source implementation of Pregel with new features (e.g. dynamic graph repartition).

GraphLab [188] is a more sophisticated graph computing engine that allows asynchronous execution and flexible scheduling. A GraphLab iteration picks up a vertex v in the task queue; and passes the vertex to a user defined function which may modify the data on the vertex its adjacent edges and vertices,

and finally may add its adjacent vertices to the task queue. Note that several nodes can be evaluated in parallel as long as they do not violate the consistency guarantee which ensures that GraphLab is equivalent with some serial algorithm. It has been used to parallelize a number of ML tasks, including matrix factorization, Gibbs sampling and Lasso [188]. [96] presents a distributed Gibbs sampler on GraphLab for an improved sLDA model using RegBayes. Several other graph computing engines have been developed. For example GraphX [189] is an extension of Spark for graph computing; and GraphChi [190] is a disk based version of GraphLab.

Parameter servers

All the above frameworks restrict the communication between workers. For example MapReduce and Spark don't allow communication between workers while Pregel and GraphLab only allow vertices to communicate with adjacent nodes. On the other side, many ML methods follow a pattern that: (1) Data are partitioned on many workers; (2) There are some shared global parameters (e.g. the model weights in a gradient descent method or the topic-word count matrix in the collapsed Gibbs sampler for LDA [191]); and (3) Workers fetch data and update (parts of) global parameters based on their local data (e.g. using the local gradients or local sufficient statistics). Though it is straightforward to implement on shared memory computers it is rather difficult in a distributed setting. The goal of parameter servers is to provide a distributed data structure for parameters.

A parameter server is a key-value store (like a hash map), accessible for all workers. It supports for get and set (or update) for each entry. In a distributed setting, both server and client consist of many nodes (see Fig. 6 (c)). Memcached [192] is an in memory key-value store that provides get and set for arbitrary data. However it doesn't have a mechanism to resolve conflicts raised by concurrent access e.g. concurrent writes for a single entry. Applications like [161] require to lock the global entry while updating which leads to suboptimal performance. Piccolo [193] addresses this by introducing user-defined accumulations which correctly address concurrent updates to the same key. Piccolo has a set of built-in user defined accumulations such as summation, multiplication, and min/max.

One important tradeoff made by parameter servers is that they sacrifice consistency for less latency—get may not return the most recent value, so that it can return immediately without waiting for most recent updates to reach the server. While this improves the performance significantly, it can

potentially slow down convergence due to outdated parameters. [194] proposed Stale Synchronous Parallel (SSP) where the staleness of parameters is 'bounded' and the fastest worker can be ahead of the slowest one by no more than τ iterations, where τ can be tuned to get a fast convergence as well as low waiting time. Petuum [195] is a SSP based parameter server. [196] proposed communication-reducing improvements including key caching message compression and message filtering, and it also supports elastically adding and removing both server and worker nodes.

Parameter servers have been deployed in learning very large-scale logistic regression [196] deep networks [197] LDA [162,198] and Lasso [195]. [196] learns a 2000-topic LDA with 5 billion documents and 5 million unique tokens on 6000 machines in 20 h. Yahoo! LDA [162] has a parameter server designed specifically for Bayesian LVM and it is the fastest available LDA software. There are a bunch of distributed topic modeling softwares based on Yahoo! LDA including [98] for MedLDA and [58] for correlated topic models.

Model parallel inference

MapReduce Spark and Parameter servers take the 'data-parallelism' approach where 'data' are partitioned across machines and computations are performed on each node given a copy of the globally shared 'model'. However, as the model size rapidly grows (i.e. the large M challenge), the models cannot fit in a single computer's memory. Model-parallelism addresses this challenge by partitioning the 'model' and storing a part of the model on each node. Then, partial updates (i.e. the updates of model parts) are carried out on each node. Benefits of model-parallelism include large model sizes, flexibility to focus workers on fastest-converging parameters, and more accurate convergence because no delayed update is involved.

STRADS [199] provides primitives for model-parallelism and it handles the distributed storage of model and data automatically. STRADS requires that a partial update could be computed using just the model part together with data. Users writes 'schedule' that assigns model sets to workers 'push' that computes the partial updates for model 'pop' that applies updates to model. An automatic 'sync' primitive will ensure that users always get the latest model. As a concrete example [200] demonstrates a model parallel LDA in which both data and model are partitioned by vocabulary. In each iteration, a worker only samples latent variables and updates the model related to the vocabulary part assigned to it. The model then rotates between

workers, until a full cycle is completed. Unlike data parallel LDA [160–162], the sampler always uses the latest models and no read-write lock is needed on models, thereby leading to faster convergence than data-parallel LDAs.

Note that model-parallelism is not a replacement but a complement of data-parallelism. For example, [201] showed a two layer LDA system where layer 1 is model-parallelism and layer 2 consists of several local model-parallelism clusters performing asynchronous updates on an globally distributed model.

CONCLUSIONS AND PERSPECTIVES

We present a survey of recent advances on Big Learning with Bayesian methods, including Bayesian non-parametrics, RegBayes, and scalable inference algorithms and the systems based-on stochastic sub-sampling or distributed computing. It is helpful to note that our review is not exhaustive. In fact, Big Learning has attracted intense interest with active research spanning diverse fields, including ML, databases, parallel and distributed systems, and programming languages.

As reviewed above, Big Learning with Bayesian methods has achieved substantial progress. However, considerable challenges still remain. We briefly discuss several directions that are of promise for future investigation. First, Bayesian methods have the advantage to incorporate prior knowledge for efficient learning, especially for the scenarios where a large number of training data is lacking, and characterize uncertainty. For instance, the recent work [202] demonstrates an example for the challenging task of one-shot learning which achieves human-level performance by encoding the domain knowledge as a hierarchical Bayesian model. In contrast, deep learning methods [203] stand at the other end of the spectrum—they are often learned in an end-to-end manner by feeding a large set of training data and they often do not represent the uncertainty in the structure or parameters of the neural networks. A natural and important question that remains under-addressed is how to conjoin the flexibility of deep learning and the learning efficiency of Bayesian methods for robust learning. Another related important question is how to effectively collect domain knowledge and incorporate it into the modeling and inference process. The work [92] has demonstrated an example that selectively incorporates the noisy knowledge collected from crowds for robust Bayesian inference but much more are left unexplored.

Second, one of the lessons we learn from Big Learning is that the best predictive performance is

often obtained by building a highly flexible model (e.g. deep neural networks [203]). Although NPB techniques are powerful in theory to represent flexible models and automatically infer their complexity from an unbounded space there is still a large gap in practice, with very few real applications. Most of the evaluations are proof-of-concepts by being hindered on small-scale problems or those with relatively simple structures. For example, although some attempts have demonstrated that a cascade IBP can be applied to infer the structure of a sparse deep belief network [60] these results are preliminary and can only learn toy network structures. It needs further study on how to learn the structure of a sophisticated network with state-of-the-art performance. In order to fill up the practical gap of non-parametric models, we need to develop the algorithms that are accurate and scalable as well as the theory of defining flexible nonparametric processes that can properly consider the rich structures in various domains.

Third, a more powerful way of composing Bayesian models is offered by probabilistic programming (http://probabilistic-programming.org), which uses general-purpose computer programs to represent probabilistic models and automates the inference procedure by building a universal engine. Several probabilistic programming languages have been developed, including BUGS (http://www.mrc-bsu.cam.ac.uk/software/bugs/), Stan (http://mc-stan.org/), BLOG (https://bayesianlogic.github.io/), Church (https://projects.csail.mit.edu/church/wiki/Church) and Infer.Net (http://research.microsoft.com/en-us/um/cambridge/projects/infernet/). However, scalable inference is still a considerable challenge for these languages. The existing platforms for Bayesian inference do not well support the advanced deep models and the recent scalable algorithms in distributed/stochastic settings. They do not well support the accelerating hardware (e.g. GPUs and FPGAs) either. In fact, the existence of user-friendly platforms (e.g. Tensorflow [204] Theano [205] and Caffe [206]) has significantly boosted the applications of deep learning in industry. It will be very useful to fill up this gap for Bayesian methods which can allow for rapid prototyping and testing of different models therefore motivating wider adoption of Bayesian methods. Edward (http://edwardlib.org/) is a recent system that builds on Tensorflow for scalable Bayesian inference, but much work needs to be done.

Finally, the current ML methods in general still require considerable human expertise in devising appropriate features, priors, models and algorithms. Much work has to be done in order to make ML more widely used and eventually become a common part of our day to day tools in data sciences. Along this line, several promising projects have been started. Google prediction API is one of the earliest efforts that try to make ML accessible for beginners by providing easy-to-use service. Microsoft AzureML takes a similar approach by providing a visual interface to help design experiments. SystemML [207] provides an R-like declarative language to specify ML tasks based on MapReduce and MLBase [208] further improves it by providing learning-specific optimizer that transforms a declarative task into a sophisticated learning plan. Finally, Automated Statistician (AutoStat) [209] aims to automate the process of statistical modeling, by using Bayesian model selection strategies to automatically choose good models/features and to interpret the results in easy-to-understand ways, in terms of automatically generated reports. Though still at a very early stage, such efforts would have a tremendous impact on the fields that currently rely on expert statisticians, ML researchers and data scientists.

FUNDING

REFERENCES

1. Brumfiel G. High-energy physics: down the petabyte highway. *Nature* 2011; **469**: 282–3.
2. Doctorow C. Big data: Welcome to the petacentre. *Nature* 2008; **455**: 16–21.
3. Reichman OJ, Jones MB and Schildhauer MP. Challenges and Opportunities of Open Data in ecology. *Science* 2011; **331**: 703–5.
4. Fan J, Han F and Liu H. Challenges of Big Data analysis. *Nat Sci Rev* 2013; **1**: 293–314.
5. Mitchell T. *Machine Learning.* NY, USA: McGraw-Hill Education, 1997.
6. Weinberger K, Dasgupta A and Langford J *et al.* Feature hashing for large scale multitask learning. In: *International Conference on Machine Learning.* Montreal, Canada, 2009, 1113–20.
7. Tan M, Tsang I and Wang L. Towards ultrahigh dimensional feature selection for big data. *JMLR* 2014; **15**: 1371–429.
8. http://www.image-net.org/about-overview (23 April 2017, date last accessed).
9. http://lshtc.iit.demokritos.gr/ (23 April 2017, date last accessed).
10. Bengio S, Weston J and Grangier D. Label embedding trees for large multi-class tasks. In: *Advances in Neural Information Processing Systems* 2010, 163–71.

11. Deng J, Satheesh S and Berg A *et al.* Fast and balanced: efficient label tree learning for large scale object recognition. In: *Advances in Neural Information Processing Systems*. Granada, Spain, 2011, 567–75.

12. Hinton G, Deng L and Yu D *et al.* Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Process Mag* 2012; **29**: 82–97.

13. Vincent P, Larochelle H and Bengio Y *et al.* Extracting and composing robust features with denoising autoencoders. In: *International Conference on Machine Learning*. Helsinki, Finland, 2008, 1096–103.

14. Le and Quoc V. Building high-level features using large scale unsupervised learning. *IEEE International Conference on Speech and Signal Processing*, 2013, 8595–8.

15. Salakhutdinov R. Learning deep generative models. *Ph.D. Thesis*. University of Toronto, Department of Computer Science, 2009.

16. Rezende DJ, Mohamed S and Wierstra D. Stochastic backpropagation and approximate inference in deep generative models. In: *International Conference on Machine Learning*. Beijing, China, 2014, 1278–86.

17. Jordan MI. The era of Big Data. *ISBA Bulletin* 2011; **18**: 1–3.

18. Zhu J, Chen N and Xing EP. Bayesian inference with posterior regularization and applications to infinite latent SVMs. *JMLR* 2014; **15**: 1799–1847.

19. Hjort N, Holmes C and Muller P *et al. Bayesian Nonparametrics: Principles and Practice*. Cambridge, UK: Cambridge University Press, 2010.

20. Bialek W, Nemenman I and Tishby N. Predictability, complexity and learning. *Neural Comput* 2001; **13**: 2409–63.

21. Srivastava N, Hinton G and Krizhevsky A *et al.* Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 2014; **15**: 1929–58.

22. Welling M. Exploiting the statistics of learning and inference. In: *Neural Information Processing Systems workshop on 'Probabilistic Models for Big Data'*. Lake Tahoe, USA, 2013.

23. Gelman A, Carlin J and Stern H *et al. Bayesian Data Analysis*. 3rd edn. Chapman & Hall/CRC Texts in Statistical Science, Florida, USA: CRC Press, 2013.

24. Efron B. Bayes' Theorem in the 21st Century. *Science* 2013; **340**: 1177–8.

25. Ghosh JK and Ramamoorthi RV. *Bayesian Nonparametrics*. New York, NY: Springer, 2003.

26. Williams PM. Bayesian Conditionalisation and the Principle of Minimum Information. *Br J Philos Sci* 1980; **31**.

27. Bishop CM. *Pattern Recognition and Machine Learning*. New York: Springer, 2006, 152–64, 217–20, 359–418.

28. Kadane JB and Lazar NA. Methods and criteria for model selection. *JASA* 2004; **99**: 279–90.

29. Kass RE, Raftery AE and Bayes Factors. *JASA* 1995; **90**: 773: 95.

30. Grelaud A, Robert CP and Marin JM *et al.* Likelihood-free methods for model choice in Gibbs random fields. *Bayesian Anal* 2009; **4**: 317–36.

31. Turnera BM and Zandtb TV. A tutorial on approximate Bayesian computation. *J Math Psychol* 2012; **56**: 69–85.

32. Robert CP, Cornuet JM and Marin JM *et al.* Lack of confidence in approximate Bayesian computation model choice. *Proc Natl Acad Sci USA* 2011; **108**: 15112–7.

33. Wainwright M and Jordan M. Graphical models, exponential families, and variational inference. *Found Trends Mach Learn* 2008; **1**: 1–305.

34. Jordan M, Ghahramani Z and Jaakkola T *et al.* An Introduction to variational methods for graphical models. *MLJ* 1999; **37**: 183–233.

35. Beal MJ. Variational Algorithms for approximate Bayesian inference. *Ph.D. Thesis*. University of Cambridge. Gatsby Computational Neuroscience Unit, University College London, 2003.

36. Robert C and Casella G. *Monte Carlo Statistical Methods*. New York: Springer, 2005.

37. Andrieu C, Freitas ND and Doucet A *et al.* An introduction to MCMC for machine learning. *Mach Learn* 2003; **50**: 5–43.

38. Metropolis N, Rosenbluth A and Rosenbluth M *et al.* Equation of state calculations by fast computing machines. *J Chem Phys* 1953; **21**: 1087.

39. Hastings W. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 1970; **57**: 97–109.

40. Geman S and Geman D. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Trans PAMI* 1984; **6**: 721–41.

41. Neal RM. MCMC using Hamiltonian Dynamics. In: Brooks S, Gelman A and Jones G *et al.* (eds.). *Handbook of Markov Chain Monte Carlo*. Florida, USA: Chapman & Hall/CRC Press, 2010.

42. Jasra A, Stephens DA and Holmes CC. On population-based simulation for static inference. *Stat Comput* 2007; **17**: 263–79.

43. Geyer CJ and Thompson EA. Annealing Markov Chain Monte Carlo with applications to ancestral inference. *JASA* 1995; **90**: 909–20.

44. Tanner M and Wong W. The calculation of posterior distributions by data augmentation. *JASA* 1987; **82**: 528–40.

45. Dyk DV and Meng X. The art of data augmentation. *JCGS* 2001; **10**: 1–50.

46. Neal R. Slice sampling. *Ann Statist* 2003; **31**: 705–67.

47. van Dyk D and Park T. Partially collapsed gibbs samplers: theory and methods. *JASA* 2008; **103**: 790–6.

48. Blei D, Ng A and Jordan MI. Latent Dirichlet allocation. *JMLR* 2003; **3**: 993–1022.

49. Li FF and Perona P. A Bayesian hierarchical model for learning natural scene categories. In: *Conference on Computer Vision and Pattern Recognition*. 2005.

50. Jeffreys H. An invariant form for the prior probability in estimation problems. In: *Proceedings of the Royal Society of London Series A, Math Phys Sci* 1945; **186**: 453–61.

51. Jaynes ET. Prior probabilities. *IEEE Trans Sys Sci Cybern* 1968; **4**: 227–41.

52. George EI and Foster DP. Calibration and empirical Bayes variable selection. *Biometrika* 2000; **87**: 731–47.

53. McAuliffe JD, Blei DM and Jordan MI. Nonparametric empirical Bayes for the Dirichlet process mixture model. *Statist Comput* 2006; **16**: 5–14.

54. Petrone S, Rousseau J and Scricciolo C. Bayes and empirical Bayes: Do they merge? *Biometrika* 2014; **101**: 1–18.

55. Donnet S, Rivoirard V and Rousseau J *et al.* On convergence rates of empirical Bayes Procedures. In: *47th Scientific Meeting of the Italian Statistical Society*. Cagliari, Italy, 2014.

56. Blei D and Lafferty J. Correlated topic models. In: *Advances in Neural Information Processing Systems*. Vancouver, Canada, 2006.

57. Aitchison J and Shen SM. Logistic-normal distributions: some properties and uses. *Biometrika* 1980; **67**: 261–72.

58. Chen J, Zhu J and Wang Z *et al.* Scalable Inference for Logistic-Normal Topic Models. In: *Advances in Neural Information Processing Systems*. Lake Tahoe, USA, 2013, 2445–53.

59. Bengio Y, Courville A and Vincent P. Representation learning: a review and new perspectives. *IEEE Trans PAMI* 2013; **35**: 1798–828.

60. Adams R, Wallach H and Ghahramani Z. Learning the structure of deep sparse graphical models. In: *Artificial Intelligence and Statistics Conference*. Sardinia, Italy, 2010, 1–8.

61. Ghahramani Z. Bayesian nonparametrics and the probabilistic approach to modelling. *Phil Trans Royal Soc* 2013; **371**: 20110553.

62. Gershmana S and Blei D. A tutorial on Bayesian nonparametric models. *J Math Psychol* 2012; **56**: 1–12.

63. Muller P and Quintana FA. Nonparametric Bayesian Data Analysis. *Stat Sci* 2004; **19**: 95–110.

64. Ferguson TS. A Bayesian analysis of some nonparametric problems. *Ann Stats* 1973; **1**: 209–30.

65. Sethuraman J. A Constructive definition of Dirichlet Priors. *Statistica Sinica* 1994; **4**: 639–50.

66. Blei DM and Jordan MI. Variational inference for Dirichlet process mixtures. *Bayesian Anal* 2006; **1**: 121–44.

67. Pitman J. Combinatorial Stochastic Processes. *Technical Report No. 621*. Department of Statistics, UC, Berkeley. 2002.

68. Antoniak CE. Mixture of Dirichlet Process with Applications to Bayesian Nonparametric Problems. *Ann Stats* 1974; **273**: 1152–74.

69. Neal R. Markov chain sampling methods for Dirichlet process mixture models. *JCGS* 2000; **9**: 249–65.

70. Walker SG. Sampling the Dirichlet mixture model with slices. *Commun Stat* 2007; **36**: 45–54.

71. Griffiths TL and Ghahramani Z. Infinite latent feature models and the Indian buffet process. In: *Advances in Neural Information Processing Systems*, 2005, **18**: 475–82.

72. Thibaux R and Jordan MI. Hierarchical beta processes and the Indian buffet process. In: *Artificial Intelligence and Statistics Conference*, 2007, 564–71.

73. Teh YW, Gorur D and Ghahramani Z. Stick-breaking construction for the Indian buffet process. In: *Artificial Intelligence and Statistics Conference*, 2007, 556–63.

74. Doshi-Velez F, Miller K and Gael JV *et al.* Variational inference for the Indian buffet process. In: *Artificial Intelligence and Statistics Conference*. Clearwater Beach, Florida, USA, 2009.

75. Hofmann T, Scholkopf B and Smola AJ. Kernel methods in machine learning. *Ann Statist* 2008; **36**: 1171–220.

76. Rasmussen CE and Williams CKI. *Gaussian Processes for Machine Learning*. Cambridge, Massachusetts, USA: The MIT Press, 2006.

77. Lawrence N. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *JMLR* 2005; **6**: 1783–816.

78. Teh YW, Jordan MI and Beal MJ *et al.* Hierarchical Dirichlet processes. *JASA* 2006; **101**: 1566–81.

79. Dallaire P, Giguere P and Chaib-draa B. Learning the structure of probabilistic graphical models with an extended cascading indian buffet process. In: *Association for the Advancement of Artificial Intelligence*. Quebec, Canada, 2014, 1774–80.

80. Beal MJ, Ghahramani Z and Rasmussen CE. The infinite hidden Markov model. In: *Advances in Neural Information Processing Systems*, 2002, 577–84.

81. Zhang A, Zhu J and Zhang B. Max-margin Infinite Hidden Markov Models. In: *International Conference on Machine Learning*. Beijing, China, 2014, 315–23.

82. MacEachern S. Dependent nonparametric processes. In: *ASA proceedings of the section on Bayesian statistical science*, 1999.

83. Duan J, Guindani M and Gelfand A. Generalized spatial Dirichlet process models. *Biometrika* 2007; **94**: 809–25.

84. Blei D and Frazier P. Distance dependent Chinese restaurant processes. In: *International Conference on Machine Learning*. Haifa, Israel, 2010, 2461–88.

85. Williamson S, Orbanz P and Ghahramani Z. Dependent Indian buffet processes. In: *Artificial Intelligence and Statistics Conference*. Sardinia, Italy, 2010, 924–31.

86. Gershman SJ, Frazier PI and Blei DM. Distance dependent infinite latent feature models. *IEEE transactions on pattern analysis and machine intelligence* 2015; **37**: 334–45.

87. Miller K, Griffiths T and Jordan M. Nonparametric latent feature models for link prediction. In: *Advances in Neural Information Processing Systems*, 2009, 1276–84.

88. Zhu J. Max-Margin Nonparametric latent feature models for link prediction. In: *International Conference on Machine Learning*. Edinburgh, Scotland, 2012, 719–26.

89. Gopalan P and Blei D. Efficient discovery of overlapping communities in massive networks. *Proc Natl Acad Sci USA* 2013; **110**: 14534–9.

90. Kim DI, Gopalan P and Blei DM *et al.* Efficient online inference for bayesian nonparametric relational models. In: *Advances in Neural Information Processing Systems*. Lake Tahoe, USA, 2013, 962–70.

91. Zhu J, Ahmed A and Xing EP. MedLDA: maximum margin supervised topic models. *JMLR* 2012; **13**: 2237–78.

92. Mei S, Zhu J and Zhu X. Robust RegBayes: selectively incorporating first-order logic domain knowledge into Bayesian models. In: *International Conference on Machine Learning*. Beijing, China, 2014, 253–61.

93. Koyejo O and Ghosh J. Constrained Bayesian inference for low rank multitask learning. In: *Conference on Uncertainty in Artificial Intelligence*. Washington, USA, 2013, 341–50.

94. Song Y, Zhu J and Ren Y. Kernel Bayesian Inference with posterior regularization. In: *Advances in Neural Information Processing Systems*. Barcelona, Spain, 2016, 4763–71.

95. Blei D and McAuliffe J. Supervised topic models. In: *Advances in Neural Information Processing Systems*, 2007, 121–8.

96. Zhu J, Zheng X and Zhang B. Improved Bayesian logistic supervised topic models with data augmentation. In: *Association for Computational Linguistics*. Sofia, Bulgaria, 2013.

97. Zhu J, Chen N and Perkins H *et al.* Gibbs Max-margin topic models with data augmentation. *JMLR* 2014; **15**: 1073–110.

98. Zhu J, Zheng X and Zhou L *et al.* Scalable inference in max-margin topic models. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. Chicago, USA: ACM, 2013, 964–72.

99. Bottou L and Bousquet O. The Tradeoffs of large scale learning. In: *Advances in Neural Information Processing Systems*, 2008, 161–8.

100. Boyd S and Vandenberghe L. *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

101. Duchi J, Hazan E and Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR* 2011; **12**: 2121–59.

102. Bottou L. *Online Algorithms and Stochastic Approximations. Online Learning and Neural Networks*. Cambridge, UK: Cambridge University Press, 1998.

103. Amari S. Natural gradient works efficiently in learning. *Neural Comput* 1998; **10**: 251–76.

104. Hoffman MD, Blei D and Wang C *et al.* Stochastic variational inference. *JMLR* 2013; **14**: 1303–47.

105. Ranganath R, Wang C and Blei D *et al.* An adaptive learning rate for stochastic variational inference. In: *International Conference on Machine Learning*. Atlanta, USA, 2013, 298–306.

106. Snoek J, Larochelle H and Adams RP. Practical Bayesian optimization of machine learning algorithms. In: *Advances in Neural Information Processing Systems*. Lake Tahoe, USA, 2012, 2951–9.

107. Mandt S and Blei D. Smoothed gradients for stochastic variational inference. In: *Advances in Neural Information Processing Systems*. Montreal, Canada, 2014, 2438–46.

108. Marlin B, Khan E and Murphy K. Piecewise bounds for estimating Bernoulli-logistic latent Gaussian models. In: *International Conference on Machine Learning*. Washington, USA, 2011, 633–40.

109. Paisley J, Blei DM and Jordan MI. Variational Bayesian inference with stochastic search. In: *International Conference on Machine Learning*. Edinburgh, Scotland, 2012, 1367–74.

110. Kingma D and Welling M. Auto-encoding variational Bayes. In: *International Conference on Learning Representations*. Banff, Canada, 2014.

111. Papaspiliopoulos O, Roberts GO and Skold M. A general framework for the parametrization of hierarchical models. *Stat Sci* 2007; **22**: 59–73.

112. Kingma D and Welling M. Efficient gradient-based inference through Transformations between Bayes nets and neural nets. In: *International Conference on Machine Learning*. Beijing, China, 2014, 1782–90.

113. Yu Y and Meng XL. To center or not to center: That is not the question–an Ancillarity–Sufficiency Interweaving Strategy (ASIS) for boosting MCMC efficiency. *JCGS* 2011; **20**: 531–70.

114. Mnih A and Gregor K. Neural Variational Inference and Learning in Belief Networks. In: *International Conference on Machine Learning*. Beijing, China, 2014, 1791–9.

115. Roberts G and Strame O. Langevin Diffusions and Metropolis-Hastings algorithms. *Methodol Comput Appl Probab* 2002; **4**: 337–57.

116. Welling M and Teh YW. Bayesian learning via Stochastic gradient langevin dynamics. In: *International Conference on Machine Learning*, 2011.

117. Chen T, Fox EB and Guestrin C. Stochastic gradient Hamiltonian Monte Carlo. In: *International Conference on Machine Learning*. Beijing, China, 2014, 1683–91.

118. Pillai N and Smith A. Ergodicity of approximate MCMC Chains with applications to large data sets. arXiv:1405.0182. 2014.

119. Teh YW, Thiery AH and Vollmer SJ. Consistency and fluctuations for stochastic gradient Langevin dynamics. *J Mach Learn Res.* 2016; **17**: 1–33.

120. Ahn S, Korattikara A and Welling M. Bayesian posterior sampling via stochastic gradient fisher scoring. In: *International Conference on Machine Learning*. Edinburgh, Scotland, 2012, 1591–8.

121. Schraudolph N, Yu J and Gunter S. A Stochastic Quasi-Newton method for on-line convex optimization. In: *Artificial Intelligence and Statistics Conference*, 2007, 436–43.

122. Patterson S and Teh YW. Stochastic Gradient Riemannian Langevin Dynamics on the Probability Simplex. In: *Advances in Neural Information Processing Systems*. Lake Tahoe, USA, 2013, 3102–10.

123. Liu C, Zhu J and Song Y. Stochastic Gradient Geodesic MCMC Methods. In: *Advances in Neural Information Processing Systems*. Barcelona, Spain, 2016, 3009–17.

124. Ma Y, Chen T and Fox E. A complete recipe for stochastic gradient MCMC. In: *Advances in Neural Information Processing Systems*. Montreal, Canada, 2015, 2917–25.

125. Korattikara A, Chen Y and Welling M. Austerity in MCMC Land: Cutting the Metropolis-Hastings Budget. In: *International Conference on Machine Learning*. Beijing, China, 2014, 181–9.

126. Bardenet R, Doucet A and Holmes C. Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In: *International Conference on Machine Learning*. Beijing, China, 2014, 405–13.

127. Bardenet R and Maillard OA. Concentration inequalities for sampling without replacement. *Bernoulli.* 2015; **21**: 1361–85.

128. Maclaurin D and Adams RP. Firefly Monte Carlo: exact MCMC with Subsets of Data. In: *Conference on Uncertainty in Artificial Intelligence*. Quebec, Canada, 2014, 4289–95.

129. Shi T and Zhu J. Online Bayesian passive-aggressive learning. In: *International Conference on Machine Learning*. Beijing, China, 2014, 378–86.

130. Rabiner LR. A tutorial on hidden Markov Models and selected applications in speech recognition. *Proc of the IEEE* 1989; **77**: 257–86.

131. Scott SL. Bayesian Methods for Hidden Markov Models. *JASA* 2002; **97**: 337–51.

132. Kalman RE. A New Approach to Linear Filtering and Prediction Problems. *J Fluids Eng* 1960; **82**: 35–45.

133. Broderick T, Boyd N and Wibisono A *et al.* Streaming Variational Bayes. In: *Advances in Neural Information Processing Systems*, 2013.

134. Crammer K, Dekel O and Keshet J *et al.* Online Passive-Agressive Algorithms. *JMLR* 2006; 551–85.

135. Andrieu C, Doucet A and Holenstein R. Particle Markov chain Monte Carlo methods. *J R Stat Soc Ser B* 2010; **72**: 269–342.

136. Liu JS and Chen R. Sequential Monte Carlo Methods for Dynamic Systems. *JASA* 1998; **93**: 1032–44.

137. Arulampalam M, Maskell S and Gordon N *et al.* A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Trans Signal Process* 2002; **50**: 174–88.

138. Lauritzen SL. Propagation of probabilities, means and variances in mixed graphical association models. *JASA* 1992; **87**: 1098–108.

139. Opper, OM. A Bayesian approach to on-line learning. *On-Line Learning in Neural Networks.* Cambridge University Press, 1999.

140. Guhaniyogi R, Qamar S and Dunson D. Bayesian Conditional Density Filtering for Big Data. arXiv:1401.3632. 2014.

141. Boyd S, Parikh N and Chu E *et al.* Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Found Trends Mach Learn* 2011; **3**: 1–122.

142. Zhai K, Boyd-Graber J and Asadi N *et al.* Mr. LDA: a flexible large scale topic modeling package using variational inference in MapReduce. In: *International World Wide Web Conference*. Lyon, France, 2012, 879–88.

143. Wu XL, Sun C and Beissinger T *et al.* Parallel Markov chain Monte Carlo - bridging the gap to high-performance Bayesian computation in animal breeding and genetics. *Genet Sel Evol* 2012; **44**: 29–46.

144. Gelman A and Rubin D. Inference from iterative simulation using multiple simulations. *Stat Sci* 1992; **7**: 457–511.

145. Suchard MA, Wang Q and Chan C *et al.* Understanding GPU Programming for Statistical Computation: Studies in Massively Parallel Massive Mixtures. *JCGS* 2010; **19**: 419–38.

146. Wilkinson DJ. Parallel bayesian computation. *Statistics Textbooks and Monographs*, 2006; **184**: 477.

147. Williamson SA, Dubey A and Xing EP. Parallel Markov Chain Monte Carlo for Nonparametric Mixture Models. In: *International Conference on Machine Learning*. Atlanta, USA, 2013, 98–106.

148. Scott SL, Blocker AW and Bonassi FV *et al.* Bayes and big data: the consensus Monte Carlo algorithm. *Int J Manage Sci Eng Manage* 2016; **11**: 78–88.

149. Neiswanger W, Wang C and Xing EP. Asymptotically Exact, Embarrassingly Parallel MCMC. In: *Conference on Uncertainty in Artificial Intelligence*. Quebec, Canada, 2014, 623–32.

150. Wang X and Dunson DB. Parallelizing MCMC via Weierstrass Sampler. arXiv:1312.4605. 2013.

151. Minsker S, Srivastava S and Lin L *et al.* Scalable and Robust Bayesian Inference via the Median Posterior. In: *International Conference on Machine Learning*. Beijing, China, 2014, 1656–64.

152. Beck A and Sabach S. Weiszfelds method: old and new results. *J Opt Theory Appl* 2015; **164**, 1–40.

153. Xu M, Lakshminarayanan B and Teh YW *et al.* Distributed Bayesian Posterior Sampling via Moment Sharing. In: *Advances in Neural Information Processing Systems*, 2014, 3356–64.

154. Brockwell AE. Parallel Markov chain Monte Carlo Simulation by Pre-Fetching. *JCGS* 2006; **15**: 246–61.

155. Angelino E, Kohler E and Waterland A *et al.* Accelerating MCMC via Parallel Predictive Prefetching. arXiv:1403.7265. 2014.

156. Strid I. Efficient parallelisation of Metropolis-Hastings algorithms using a prefetching approach. *Comput Stat Data Anal* 2010; **54**: 2814–35.

157. Banterle M, Grazian C and Robert CP. Accelerating Metropolis-Hastings algorithms: Delayed acceptance with prefetching. arXiv:1406.2660. 2014.

158. Gonzalez JE, Low Y and Gretton A *et al.* Parallel Gibbs Sampling: From Colored Fields to Thin Junction Trees. In: *Artificial Intelligence and Statistics Conference.* Florida, USA, 2011, 324–32.

159. Johnson MJ, Saunderson J and Willsky AS. Analyzing Hogwild Parallel Gaussian Gibbs Sampling. In: *Advances in Neural Information Processing Systems*, 2013.

160. Newman D, Asuncion A and Smyth P *et al.* Distributed Inference for latent Dirichlet allocation. In: *Advances in Neural Information Processing Systems*, 2007, 1081–8.

161. Smola A and Narayanamurthy S. An architecture for parallel topic models. In: *Proceedings of the VLDB Endowment*, 2010; **3**: 703–10.

162. Ahmed A, Aly M and Gonzalez J *et al.* Scalable inference in latent variable models. In: *International Conference on Web Search and Data Mining.* Washington, USA, 2012, 123–32.

163. Liu Z, Zhang Y and Chang EY *et al.* PLDA+: Parallel latent Dirichlet allocation with data placement and pipeline processing. *TIST* 2011; **2**: 26.

164. Chen J, Li K and Zhu J *et al.* WarpLDA: a cache efficient O (1) algorithm for latent dirichlet allocation. In: *Proceedings of the VLDB Endowment*, 2016; **9**: 744–55.

165. Gal Y and Ghahramani Z. Pitfalls in the use of Parallel Inference for the Dirichlet Process. In: *International Conference on Machine Learning.* Beijing, China, 2014, 208–16.

166. Zinkevich MA, Weimer M and Smola A *et al.* Parallelized StochasticGradient Descent. In: *Advances in Neural Information Processing Systems*, 2010, 2595–603.

167. Niu F, Recht B and Re C *et al.* Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In: *Advances in Neural Information Processing Systems*, 2011, 693–701.

168. Ahn S, Shahbaba B and Welling M. Distributed Stochastic Gradient MCMC. In: *International Conference on Machine Learning.* Beijing, China, 2014, 1044–52.

169. Yang Y, Chen J and Zhu J. Distributing the Stochastic Gradient Sampler for Large-Scale LDA. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* California, USA, 2016, 1975–84.

170. http://www.openmp.org (23 April 2017, date last accessed).

171. https://www.threadingbuildingblocks.org/ (23 April 2017, date last accessed).

172. Lee A, Yau C and Giles MB *et al.* On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *JCGS* 2010; **19**: 769–89.

173. Moral PD, Doucet A and Jasra A. Sequential Monte Carlo samplers. *J R Stat Soc Ser B* 2006; **68**: 411–36.

174. Beam AL, Ghosh SK and Doyle J. Fast hamiltonian monte carlo using gpu computing. *J Comput Graph Stat*, 2016; **25**: 536–48.

175. Yan F, Xu N and Qi A. Parallel Inference for Latent Dirichlet Allocation on Graphics Processing Units. In: *Advances in Neural Information Processing Systems*, 2009, 2134–42.

176. Li K, Chen J and Chen W *et al.* SaberLDA: Sparsity-Aware Learning of Topic Models on GPUs. *International Conference on Architectural Support for Programming Languages and Operating Systems* 2017.

177. Canny J and Zhao H. BIDMach: Large-scale Learning with Zero Memory Allocation. In: *Advances in Neural Information Processing Systems. Big Learning Workshop*, 2013.

178. Chau T, Targett J and Wijeyasinghe M *et al.* Accelerating sequential Monte Carlo method for real-time air traffic management. *SIGARCH Comp Arch News* 2013; **41**: 35–40.

179. Ronquist F and Huelsenbeck JP. MrBayes: Bayesian inference of phylogenetic trees. *Bioinformatics* 2003; **19**: 1572–4.

180. Altekar G, Dwarkadas S and Huelsenbeck JP *et al.* Parallel Metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. *Bioinformatics* 2004; **20**: 407–15.

181. Bekkerman R, Bilenko M and Langford J. *Scaling up machine learning: Parallel and distributed approaches.* Cambridge, UK: Cambridge University Press, 2011.

182. Dean J and Ghemawat S. MapReduce: simplified data processing on large clusters. *Communications of the ACM* 2008; **51**: 107–13.

183. Chu C, Kim SK and Lin YA *et al.* Map-reduce for machine learning on multicore. In: *Advances in Neural Information Processing Systems*, 2007, 281–8.

184. Apache Mahout: https://mahout.apache.org/.

185. Zaharia M, Chowdhury M and Franklin MJ *et al.* Spark: cluster computing with working sets. In: *Hot Topics in Cloud Computing*, 2010.

186. Malewicz G, Austern MH and Bik AJ *et al.* Pregel: a system for large-scale graph processing. In: *Special Interest Group on Management of Data*, 2010.

187. Salihoglu S and Widom J. GPS: A Graph Processing System. In: *Conference on Scientific and Statistical Database Management.* Maryland, USA, 2013.

188. Low Y, Gonzalez J and Kyrola A *et al.* Graphlab: A new framework for parallel machine learning. In: *Conference on Uncertainty in Artificial Intelligence.* Washington, USA, 2013, 340–9.

189. Xin RS, Gonzalez JE and Franklin MJ *et al.* Graphx: A resilient distributed graph system on spark. In: *Workshop on Graph Data Management Experiences and Systems*, 2013.

190. Kyrola A, Blelloch GE and Guestrin C. GraphChi: Large-Scale Graph Computation on Just a PC. In: *Operating Systems Design and Implementation.*

191. Griffiths T and Steyvers M. Finding scientific topics. *Proc Natl Acad Sci USA* 2004; 101.

192. http://memcached.org (23 April 2017, date last accessed).

193. Power R and Li J. Piccolo: Building Fast, Distributed Programs with Partitioned Tables. In: *Operating Systems Design and Implementation.* Vancouver, Canada, 2010, 1–14.

194. Ho Q, Cipar J and Cui H *et al.* More effective distributed ML via a stale synchronous parallel parameter server. In: *Advances in Neural Information Processing Systems*, 2013, 1223–31.

195. Dai W, Wei J and Zheng X *et al.* Petuum: A Framework for Iterative-Convergent Distributed ML. arXiv:1312.7651; 2013.

196. Li M, Andersen D and Park JW *et al.* Scaling Distributed Machine Learning with the Parameter Server. In: *Operating Systems Design and Implementation.* Broomfield, USA, 2014, 583–98.

197. Dean J, Corrado G and Monga R *et al.* Large scale distributed deep networks. In: *Advances in Neural Information Processing Systems*, 2012, 1223–31.

198. Li AQ, Ahmed A and Ravi S *et al.* Reducing the sampling complexity of topic models. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, USA, 2014, 891–900.

199. Lee S, Kim JK and Zheng X *et al.* Primitives for Dynamic Big Model Parallelism. arXiv:1406.4580. 2014.

200. Zheng X, Kim JK and Ho Q *et al.* Model-Parallel Inference for Big Topic Models. arXiv:1411.2305. 2014.

201. Wang Y, Zhao X and Sun Z *et al.* Towards Topic Modeling for Big Data. arXiv:1405.4402. 2014.

202. Lake B, Salakhutdinov R and Tenenbaum J. Human-level concept learning through probabilistic program induction. *Science* 2015; **350**: 1332–8.

203. LeCun Y, Bengio Y and Hinton G. Deep learning. *Nature* 2015; **521**: 436–44.

204. Abadi M, Agarwal A and Barham P *et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*; 2015. Software available from tensorflow.org. http://tensorflow.org/ (23 April 2017, date last accessed).

205. Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints. 2016 May;abs/1605.02688. http://arxiv.org/abs/1605.02688 (23 April 2017, date last accessed).

206. Jia Y, Shelhamer E and Donahue J *et al.* Caffe: Convolutional architecture for fast feature embedding. In: *Proceedings of the 22nd ACM international conference on Multimedia*. ACM. 2014, 675–8.

207. Ghoting A, Krishnamurthy R and Pednault E *et al.* SystemML: Declarative machine learning on MapReduce. In: *International Conference on Data Engineering*. Hannvor, Germany, 2011, 231–42.

208. Kraska T, Talwalkar A and Duchi J *et al.* MLbase: A Distributed Machine-learning System. In: *Conference on Innovative Data Systems Research*, 2013.

209. Lloyd J, Duvenaud D and Grosse R *et al.* Automatic Construction and Natural Language Description of Nonparametric Regression Models. In: *Association for the Advancement of Artificial Intelligence*. Quebec, Canada, 2014, 1242–50.