On Estimation in Latent Variable Models

Guanhua Fang, Ping Li
Cognitive Computing Lab
Baidu Research
10900 NE 8th St Bellevue WA 98004 USA
{guanhuafang, liping11}@baidu.com

Abstract

Latent variable models have been playing a central role in statistics, econometrics, machine learning with applications to repeated observation study, panel data inference, user behavior analysis, etc. In many modern applications, the inference based on latent variable models involves one or several of the following features: the presence of complex latent structure, the observed and latent variables being continuous or discrete, constraints on parameters, and data size being large. Therefore, solving an estimation problem for general latent variable models is highly non-trivial. In this paper, we consider a gradient based method via using variance reduction technique to accelerate estimation procedure. Theoretically, we show the convergence results for the proposed method under general and mild model assumptions. The algorithm has better computational complexity compared with the classical gradient methods and maintains nice statistical properties. Various numerical results corroborate our theory.

1. Introduction

A latent variable model, as the name suggests, is a statistical model that contains latent/unobserved variables. Their roots trace back to Spearman's 1904 seminal work (Spearman, 1904) on factor analysis, which is arguably the first well-articulated latent variable model. In past years, latent variable models have been playing an important role in machine learning, statistics, econometrics, psychometrics, social sciences with applications to repeated observation study, panal data inference, user behavior analysis, etc (Aigner et al., 1984; Bishop, 1998; Bartholomew et al., 2011; Ahmed et al., 2012; Loehlin and Beaujean, 2016). Latent variables serve to reduce the dimensionality of data. Many observable variables can be aggregated in a model to represent an underlying concept, making it easier to do data analysis.

In many applications, the inference based on latent variable models involves one or several of the following features: 1) the presence of complex latent structure, 2) the observed and latent variables being either continuous or discrete, 3) constraints on parameters, 4) data size being large. Comparing with models without latent variables (e.g., linear regression and generalized linear regression), the estimation problem of latent variable models is typically more involved. In general, the estimation problem can have the following three perspectives: both latent variables and parameters are viewed as fixed quantities, both latent variables and parameters are viewed as random quantities, and latent variables are random while parameters are fixed.

The first perspective (i.e., fixed latent variables and parameters) leads to the joint maximum likelihood estimator. This estimator can usually be efficiently computed by alternating minimization-type algorithm (Birnbaum, 1968; Chen et al., 2020). However such estimator could be statistically inconsistent and may lead to biased statistical inference. The second perspective (i.e., random latent variables and parameters) leads to Markov Chain Monte Carlo (MCMC) methods (Metropolis et al., 1953). Metropolis-Hasting method (Hastings, 1970), Gibbs sampler methods (Gilks and Wild, 1992; Gilks et al., 1995) and other Bayesian methods (Gelman et al., 2013) are developed to solve the estimation problem. However, as data size increases, those methods can be computationally inefficient and need long time for Markov chain to be stable.

In this paper, we adopt the third perspective (i.e., fixed parameters and random latent variables) and develop a gradient-based computational method where we incorporate variance-reduced technique to accelerate the estimation procedure. Due to the existence of latent structure, we need to estimate the gradient via sampling the latent variables according to posterior distributions. When the analytical formula of posterior distribution is not obtainable, Q-function[1]

[1] Q-function is usually referred to as the objective in the M-step.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

in expectation-maximization (EM) method is hard to compute while our method allows latent variables being sampled from some approximate distributions (which can be easily constructed by using quasi Monte-Carlo methods or MCMC methods). On the theoretical side, we provide convergence analysis of the proposed method and show that it has better computational complexity. Especially, our results are established under very general statistical model assumptions including both independent and identically distributed (i.i.d.) cases and network (non-i.i.d.) cases. Moreover, we also consider the optimization settings with regularization terms. For both smooth and non-smooth cases, our estimator is shown to have nice statistical properties and achieves the efficiency.

The rest of paper is organized as follows. In Section 2, we give an introduction of latent variable models. In Section 3, we propose a variance-reduced method for parameter estimation in latent variable models. In Section 4, we provide theoretical analysis of proposed method under both smooth and non-smooth cases. We further extend our results into network setting in Section 5. Numerical results are presented in Section 6 and validate our theory. The concluding remarks are given in Section 7.

2. Preliminaries

2.1. Notation and Setting

We consider the estimation of a parametric latent variable model. Let Y be a random response representing the observed data and Z be a random variable representing the latent unobserved trait. We use y and z to denote their realizations, respectively. Let θ represent the generic vector of parameter which needs to be estimated. We assume $\theta \in \mathcal{B} \subset \mathbb{R}^p$ where p is the dimension of the parameter and $\mathcal{B}$ is the domain. We assume latent variable Z follows a prior $p(z)$ (the prior may depend on some nuisance parameters which we are not interested in). Given $Z = z$, we assume Y follows certain distribution function parameterized by θ with density $f_\theta(y|Z = z)$. We may write $f_\theta(y|Z = z) = f_\theta(y|z)$ for notational simplicity. Therefore, the joint density is $f_\theta(y, z) = f_\theta(y|z)p(z)$ and the marginal density is $f_\theta(y) = \int f_\theta(y, z)dz$. Throughout the paper, we use subscript i to indicate each individual sample.

In the sequel, we further adopt following notations. We use $\|x\|$ and $\|x\|_1$ to represent ℓ_2- and ℓ_1-norm of vector x. $B(\theta, \delta)$ is used to denote an ℓ_2 ball centered at θ with radius δ. We use $[n]$ to denote set $\{1, \ldots, n\}$ and $[n] \times [n]$ to denote set $\{(i, j) : i \in [n], j \in [n]\}$. We use θ^* to denote the true model parameter and $\bar{\theta}$ to denote the optimizer of (4). Moreover, $a = O(b)$ means there exists a constant K such that $a \leq Kb$; $a = \Omega(b)$ means there exists a sufficiently large constant K such that $a \geq Kb$. We use ∇f ($\nabla^2 f$) to represent the first (second) derivative of f with respect to θ.

2.2. Examples

The latent variable models can be mainly divided into two categories, factor analysis (latent variables are continuous) and latent class analysis (latent variables are discrete). In this section we provide several illustrative examples to let readers have better understandings on the structures of latent variable models.

Latent Factor Models. In latent factor analysis (LFA), the latent variable $Z := (\xi_1, \xi_2, \ldots, \xi_K)$ is assumed to follow a multivariate Gaussian distribution, e.g., $N(\mathbf{0}, \Sigma)$ with Σ being the covariance matrix. Response Y is also assumed to be multivariate, that is, $Y = (Y_1, \ldots, Y_J)$.

In the linear bifactor model, response Y assumes the following form,

$$Y_j = a_{j0} + \mathbf{a}_j^T Z + \epsilon_j, \tag{1}$$

with ϵ_j's are i.i.d. $N(0, \sigma^2)$. In the item response cases, response Y takes discrete value and is usually binary.

$$\begin{aligned} P(Y_j = 1|Z) &= \Phi(a_{j0} + \mathbf{a}_j^T Z), \\ P(Y_j = 0|Z) &= 1 - P(Y_j = 1|Z), \end{aligned}$$

with $\Phi(\cdot)$ being certain link function. For example, $\Phi(x) = \exp\{x\}/(1 + \exp\{x\})$ when $\Phi(\cdot)$ is the logit link and $\Phi(x)$ is the cumulative function of standard normal distribution when $\Phi(\cdot)$ is the probit link. Here, a_{j0}'s are intercept parameters and $\mathbf{a}_j$'s are loading vectors. Suppose there are n individuals, then the likelihood function can be written as

$$L(\theta) = \prod_{i=1}^n L_i(\theta) = \prod_{i=1}^n \int \{\prod_{j=1}^J f_\theta(y_{ij}|z_i)\} p(z_i) dz_i$$

where $\theta = \{(a_{j0}, \mathbf{a}_j)\}_{j=1}^J$ and $p(z)$ is the density function of $N(0, \Sigma)$.

Latent Class Models. In statistics, a latent class model (LCM) relates a set of observed (usually discrete) to a set of latent variables. A class is characterized by a pattern of conditional probabilities that indicate the chance that the latent variables take on certain values. One of the most typical examples is the mixture Gaussian model that the response Y follows $\sum_{c=1}^C p_c f(y|\mu_c, \sigma_c^2)$ with p_c be the latent class probability and $f(y|\mu_c, \sigma_c^2)$ is the density function of normal distribution with mean μ_c and variance σ_c^2.

Furthermore, restricted latent class models are also widely used in social and behavioral sciences. For example, they are commonly used in education for cognitive diagnosis (von Davier and Lee, 2019). These models give the latent variable specific meanings and hence are easier to make diagnosis of the individuals. Here, we consider a setting where both response and latent variables are binary. Let $Z = (\alpha_1, \dots, \alpha_K) \in \{0,1\}^K$ with $\alpha_k \in \{0,1\}$ indicating the mastery of k-th latent trait/attribute. The response $Y = (Y_1, \dots, Y_J)$ with Y_j follows a Bernoulli distribution which satisfies

$$P(Y_j = 1|Z = z) = \frac{\exp\{\theta_{jz}\}}{1 + \exp\{\theta_{jz}\}}. \tag{2}$$

Furthermore, for each fixed j, $\theta_{j,Z}$'s satisfy partial order relationship. That is, $\theta_{jz_1} \le \theta_{jz_2}$ if $z_1 \preceq z_2$, representing that an examinee is more likely to answer the j-th item correctly if he is more capable (i.e., z_2 has more 1's). Under the restricted LCM setting with n individuals, the likelihood function can be written as

$$L(\theta) = \prod_{i=1}^n L_i(\theta) = \prod_{i=1}^n \sum_{z_i \in \{0,1\}^K} p_{z_i} \prod_{j=1}^J f_\theta(y|z_i)$$

with $\theta = (\theta_{jz};\ j = 1 \dots, J, z \in \{0,1\}^K)$.

Latent Network Model Latent network model is a generative model which is often used for characterizing the block structure in social networks. Assume there exist n nodes and an edge list $A \subset [n] \times [n]$. Each node i is assigned with a latent membership $z_i \in [K]$. For each pair of nodes $(i,j) \in A$, we can observe data $Y_{ij} \sim f_\theta(y|z_i, z_j)$ which are conditionally independent given z_i's.

For example, in stochastic block model (SBM), $A \equiv [n] \times [n]$ and z_i belongs to K different communities. In addition, $y_{ij} \sim$ Bernoulli$(\theta_{z_i z_j})$ indicating whether there exists an observed link between node i and j. Usually, $\theta_{z_i z_j}$ may take larger value if $z_i = z_j$ belong to the same community. Thus parameter $\theta = (\theta_{kl}, k, l \in [K])$ can be viewed as K by K symmetric edge probability matrix.

In online social platform, different users are observed to have interactions over a period of time. Then A can be viewed as a friendship list. Only friends can send messages to each other. For each pair (i,j), the observed data y_{ij} is a sequence of events that happen between these pair of users. A continuous time counting process is usually well suited to capture the event sequence y_{ij}. In the literature, Poisson process model or Hawkes process model are widely used to capture such event dynamics. By assuming the conditional independence between different node pairs, we can write the likelihood function as

$$L(\theta) = \sum_{\mathbf{z}} \prod_{i=1}^n p(z_i)\{\prod_{(i,j)\in A} f_\theta(y_{ij}|z_i, z_j)\}, \tag{3}$$

where $\mathbf{z} = (z_1, \dots, z_n)$, $p(z)$ is the prior probability of latent class z and the formula of $f_\theta(y_{ij}|z_i, z_j)$ depends on the model specification (e.g., Poisson or Hawkes Process).

In modern applications, the statistical model may admit certain structure (e.g., sparseness or monotonicity) some regularization terms can be imposed on the parameter θ. We use $R(\theta)$ to represent a general regularization term. Then the regularized maximal likelihood estimator is defined as

$$\bar{\theta} = \arg\min_\theta\{-\log L(\theta) + R(\theta)\}. \tag{4}$$

When $R(\theta) \equiv 0$, $\bar{\theta}$ becomes the maximal likelihood estimator (MLE). For simplicity, we may also write $F(\theta) := -\log L(\theta)$ and $P(\theta) := -\log L(\theta) + R(\theta)$ in the rest of paper. Then $\bar{\theta}$ is the minimizer of $P(\theta)$.

3. Algorithms

We consider a gradient-based method to estimate the parameters in the latent variable model setting when individuals are assumed to be mutually independent. To compute the gradient, we consider a variation-reduction technique to accelerate the optimization procedure. We first recall the Fisher's identity,

$$\nabla \log L_i(\theta) = \int \{\nabla \log f_\theta(y_i, z_i)\} p_\theta(z_i|y_i) dz_i,$$

which is a useful tool in maximum-likelihood parameter estimation problems. In other words, $\nabla H_i(\theta, z_i) := -\nabla \log f_\theta(y_i, z_i)$ with $z_i \sim p_\theta(z|y_i)$ will be an unbiased estimator of $-\nabla \log L_i(\theta)$. Thus, negative gradient $-\nabla \log L(\theta)$ can be approximated by

$$\nabla H(\theta) := \frac{1}{n}\sum_i \nabla H_i(\theta, z_i).$$

As data sample goes large, it could be time-consuming to compute the $\nabla H(\theta)$. A naive acceleration technique is using stochastic gradient method (SGD) by using $\nabla H_i(\theta)$ as the approximate gradient in each iteration. However, the variance of $\nabla H_i(\theta)$ does not vanish when we increase the sample size n. Here, we consider an alternative variance-reduced gradient to avoid this issue.

The main procedure is described as follows. We first randomly choose an initial point θ^0 and set the snapshot parameter $\tilde{\theta}^0 = \theta^0$. In iteration s, we first sample a batch set of data B^s with size $n_1 < n$. For each data $i \in B^s$, we sample its corresponding latent variable z according to posterior distribution $p_\theta(z_i|y_i)$ (or any approximate distribution). We then compute a snapshot of stochastic gradient, that is,

$$\tilde{\nabla} f^{s+1} = \frac{1}{n_1}\sum_{i \in B^s} \nabla H_i(\tilde{\theta}^s, z_i).$$

For each iteration s, we further sequentially update the parameter by m times. For each $t \in [m]$, we denote the current parameter value as θ_t^{s+1}. We randomly pick a data sample i_t^{s+1} from $\{1, \dots, n\}$ and sample the corresponding latent variable $z_{i_t^{s+1}}$ according to posterior $p_{\theta_t^{s+1}}(z|y_i)$. We then define the variance-reduced gradient as

$$v_t^{s+1} = \nabla H_{i_t}(\theta_t^{s+1}, z_{i_t}) - \nabla H_{i_t}(\tilde{\theta}^s, z_{i_t}) + \tilde{\nabla} f^{s+1}.$$

Scalar γ represents the step size/learning rate. See Algorithm 1 for the complete procedure.

Algorithm 1 Latent Stochastic Gradient Algorithm

Input: Observations: $(y_i, i \in [n])$.
Output: Estimated parameter $\hat{\theta}$.
Set initial parameter θ^0 and let $\theta_0^0 = \tilde{\theta}^0 = \theta^0$.
for s = 0 to S - 1 **do**
  $\theta_0^{s+1} = \theta_m^s$
  Sample a subset $B^s \subset \{1, \ldots, n\}$ with size n_1
  Sample z_i according to posterior/approximate distribution for $i \in B^s$.
  Compute $\tilde{\nabla} f^{s+1} = \frac{1}{n_1} \sum_{i \in B^s} \nabla H_i(\tilde{\theta}^s, z_i)$.
  for t = 0 to $m - 1$ **do**
    Uniformly randomly pick i_t^{s+1} from $\{1, \ldots, n\}$.
    Sample $z_{i_t^{s+1}}$ according to posterior distribution $p_{\theta_t^{s+1}}(z|y_i)$ or approximate distribution.
    Compute $v_t^{s+1} = \nabla H_{i_t}(\theta_t^{s+1}, z_{i_t^{s+1}}) - \nabla H_{i_t}(\tilde{\theta}^s, z_{i_t^{s+1}}) + \tilde{\nabla} f^{s+1}$.
    Smooth case: update

$$\theta_{t+1}^{s+1} = \theta_t^{s+1} - \gamma v_t^{s+1}.$$

    Non-smooth case: update

$$\theta_{t+1}^{s+1} = \text{proximal}_{\gamma R}(\theta_t^{s+1} - \gamma v_t^{s+1}).$$

  end for
  Set $\tilde{\theta}^{s+1} = \theta_m^{s+1}$.
end for

Different from the classical stochastic variance-reduced algorithm (SVRG), some important features of Algorithm 1 are discussed as follows.

1. Algorithm 1 is constructed under the assumption that data are independent and identically distributed and follow a parameterized statistical model, while the usual SVRG does not require any model assumption. This is due to the existence of latent structure and the construction of snapshot gradient that $\tilde{f}^{s+1}$ depends on batch set B^s (instead of $[n]$).

2. In variance-reduction step, index i_t^{s+1} is sampled from $\{1, \ldots, n\}$ instead of B^s (The latter one also works). This is helpful since it allows the algorithm to explore the whole data structure faster.

3. Note that $z_{i_t^{s+1}}$ is plugged into both $\nabla H_{i_t}(\theta_t^{s+1}, z)$ and $\nabla H_{i_t}(\tilde{\theta}^s, z)$ in the variance-reduced gradient. This is also crucial since that $\nabla H_{i_t}(\theta_t^{s+1}, z_{i_t^{s+1}}) - \nabla H_{i_t}(\tilde{\theta}^s, z_{i_t^{s+1}})$ has smaller variance compared with that of $\nabla H_{i_t}(\theta_t^{s+1}, z_{i_t^{s+1}}) - \nabla H_{i_t}(\tilde{\theta}^s, \tilde{z}_{i_t}^s)$ where $\tilde{z}_{i_t}^s$ is the latent variable used in the previous iteration.

4. In fact, we do not require to compute the exact posterior distribution $p_{\theta_t^{s+1}}(z|y_i)$. It is sufficient to find an approximate distribution $\tilde{p}_t^{s+1}(z)$ such that $\|\tilde{p}_t^{s+1}(z) - p_{\theta_t^{s+1}}(z|y_i)\|_{TV} = O(\frac{1}{\sqrt{n_1}})$. This largely reduces the computational burden when the explicit form of $p_{\theta_t^{s+1}}(z|y_i)$ is not easily obtained. Under such case, we can either use quasi-Monte Carlo (Caflisch et al., 1998; Owen and Glynn, 2016) or Markov Chain Monte Carlo method (Andrieu et al., 2003; Liu, 2008; Robert and Casella, 2013) with ergodicity property to construct the approximate distribution.

Connections to stochastic variance-reduced method

Variance reduction technique (Owen and Zhou, 2000) is proved to be useful in integration approximation. In optimization, this technique is also widely adopted. Johnson and Zhang, 2013 proposed stochastic variance reduction gradient methods (SVRG). It shows empirically better than SGD methods and batch methods. Later on, a group of researchers (Reddi et al., 2016; Allen-Zhu and Hazan, 2016) established new theory for SVRG and showed that the varianced-reduced method converges faster than SGD and full gradient method. In recent years, stochastic variance-reduced gradient Hamiltonian Monte Carlo method and Langevin dynamic method (Dubey et al., 2016; Zou et al., 2018; Xie et al., 2021; Zhao et al., 2021) are proposed to solve the optimization method from a Bayesian view. Stochastic Variance-Reduced Expectation and Maximization methods are (Zhu et al., 2017; Karimi et al., 2019; Karimi and Li, 2021) also developed for solving an optimization problem under exponential family. Our methods can be viewed as the extended version of SVRG methods in the setting with the existence of latent structure. It is designed for solving general latent variable models that both observed responses or latent variables could be either discrete or continuous. Additionally, we do not require computing the exact posterior distributions. Hence it is more widely applicable.

Connections to stochastic approximation

The proposed method is also closely related to the stochastic approximation approach which was first proposed in Robbins and Monro (1951) and Kiefer and Wolfowitz (1952), and its variants given in Gu and Kong (1998); Cai (2010) that are specially designed for latent variable model estimation. Both methods (Gu and Kong, 1998; Cai, 2010) approximate the original Robbins-Monro method by using MCMC sampling to generate an approximate stochastic gradient in each iteration, when an unbiased stochastic gradient is difficult to obtain. However these methods do not handle with non-smooth objective functions. In addition, the computational cost is high if we compute the posterior distribution based on entire dataset and the convergence rate becomes slow.

Connections to perturbed gradient algorithm

Our method is also related to perturbed gradient algorithms since we do not require to compute the posterior distribution exactly. The perturbed proximal gradient algorithm (Atchadé et al., 2017; Zhang and Chen, 2020) solves a similar optimization problem when the gradient is intractable and is approximated by Monte Carlo methods. Their method combines stochastic approximation, proximal gradient decent and Polyak-Ruppert averaging (Polyak, 1990; Ruppert, 1988). The theoretical analysis of Atchade et al.'s work focuses on convex optimization, while we consider a more general setting of non-convex optimization that includes a wide range of latent variable model estimation problems as special cases.

Comparison with Classical Methods

The gradients computed by different methods are summarized as follows.

- Proposed method:
$$\nabla H_{i_t}(\theta_t^{s+1}, z_{i_t^{s+1}}) - \nabla H_{i_t}(\tilde{\theta}^s, z_{i_t^{s+1}}) + \tilde{\nabla} f^{s+1}.$$
- Stochastic gradient descent: $\nabla H_{i_t}(\theta_t^{s+1}, z_{i_t^{s+1}})$.
- Stochastic batch method: $\tilde{\nabla} f^{s+1}$.

When θ_t^s becomes stable (i.e., $\|\theta_t^{s+1} - \theta_t^s\| = o(1)$), the gradient of SGD may not be necessarily close to zero due to the randomness of i_t^{s+1}. While the proposed method and batch method is guaranteed to be $o(1)$ under the mild statistical assumptions. On the other hand, the batch method may fail for convergence when the step size γ is set to be large value in practice. While the proposed method can still maintain a relatively good performance. For methods of other types, it is not straightforward to compare here. For example, expectation-maximization (EM) requires to compute the exact posterior distribution. Variational inference-based methods (Attias, 1999) gives a biased estimator if an unfavorable variational family is adopted.

4. Theoretical Analysis

In this section, we provide theoretical analysis for the proposed algorithm. Before introducing the main results, we first present several assumptions.

A1 Variables (y_i, z_i)'s are independently identically distributed with density $f_\theta(y, z) = f_\theta(y|z)p(z)$.

A2 The parameter θ lies in a bounded compact set $\mathcal{B} \subset \mathbb{R}^p$.

A3 Density $f_\theta(y|z)$ is a smooth function [2] of θ for all z.

A4 $\mathbb{E}_{y\sim f_{\theta^*}}[\log f_\theta(y)]$ is a strictly concave function of θ over set $\mathcal{B}$.

4.1. Smooth case

We consider the situation that there is no regularization term, that is $R(\theta) \equiv 0$. We define the stopping time $T(\epsilon) := \arg\min_s \min_t \mathbb{E}\|\nabla F(\theta_t^s)\|^2 \leq \epsilon$, where $\mathbb{E}$ is the conditional expectation given data and $\{B^s\}$. (Note that lines 9 - 10 in Algorithm 1 contain random index i_t and latent variable $z_{i_t^{s+1}}$. We take expectation with respect to these random variables.) In other words, $T(\epsilon)$ indicates the first time that $\mathbb{E}\|\nabla F(\theta_t^s)\|^2$ is below certain threshold ϵ.

Theorem 1 *Under Assumptions $A1 - A4$, then*

$$T(\epsilon) \leq C_1 \frac{F(\theta^0) - F(\bar{\theta})}{\epsilon m \gamma}$$

holds for any $\epsilon \geq \Omega(\max\{\frac{1}{n_1}, (m\gamma)^2\})$, with probability going to 1 as $n \to \infty$. Here, C_1 is a universal constant.

Theorem 1 gives the theoretical guarantee for the convergence of Algorithm 1. Quantity $\mathbb{E}\|\nabla F(\theta_t^s)\|^2$ will finally drop below the threshold ϵ when ϵ is at least of order $\frac{1}{n_1}$ and $(m\gamma)^2$. One thing should be noticed that the estimator does not necessarily converge to the global optimal solution. It may converge to any stationary point instead. The first term $\frac{1}{n_1}$ comes from the sampling noises from batch B^s and the second term $(m\gamma)^2$ appears since there exist gaps by computing the posterior distributions under different θ_t^s's in each iteration. In each iteration, we need to compute n_1 gradients for constructing the snapshots and compute m times for variance-reduced gradient. Therefore, the total computational complexity is $O((n_1 + m)T(\epsilon))$. By the special choice of n_1, m and γ, we have the following corollary.

Corollary 1 *We let $n_1 = n^{\frac{2}{3}\alpha}$, $m = n^{\frac{2}{3}\alpha}$, $\gamma = n^{-\alpha}$, then the total computational complexity is $O(n^\alpha/\epsilon)$ for any $\epsilon = \Omega(n^{-2\alpha/3})$.*

From corollary 1, we know that the total computational cost decreases as α decreases while the error term ϵ becomes larger. Hence, we should avoid choosing too small α to prevent large bias.

Refinement of Estimator

Note that the proposed algorithm has better computational complexity, however it could lead to larger estimation error when α is small. In this section, we make a refinement to

[2]Here smooth function means that the function can be differentiated for arbitrary number of times. This assumption is satisfied by most statistical model.

the estimator returned by Algorithm 1 such that the refined estimator is root-n consistent. Specifically, we consider to use the second order information to make the correction for $\hat{\theta}$. Recall Louis's Identity (Louis, 1982),

$$\begin{aligned}\frac{\partial^2 f(\theta)}{\partial\theta\partial\theta^T} \quad = \quad & \mathbb{E}[\frac{\partial^2 \log f_\theta(y,z)}{\partial\theta\partial\theta^T} + \frac{\partial \log f_\theta(y,z)}{\partial\theta} \\ & (\frac{\partial \log f_\theta(y,z)}{\partial\theta})^T|y,\theta] \\ & -\mathbb{E}[\frac{\partial \log f_\theta(y,z)}{\partial\theta}|y,\theta]\mathbb{E}[\frac{\partial \log f_\theta(y,z)}{\partial\theta}|y,\theta]^T.\end{aligned}$$

It implies that we can compute the Hessian matrix via using posterior approximation. Therefore,

$$\begin{aligned}\nabla^2 H(\theta) := -\frac{1}{n}\sum_i [\frac{\partial^2 \log f_\theta(y_i,z_i)}{\partial\theta\partial\theta^T} + \frac{\partial \log f_\theta(y_i,z_i)}{\partial\theta} \\ (\frac{\partial \log f_\theta(y_i,z_i)}{\partial\theta})^T] - \\ [\frac{1}{n}\sum_i \frac{\partial \log f_\theta(y_i,z_i)}{\partial\theta}][\frac{1}{n}\sum_i \frac{\partial \log f_\theta(y_i,z_i)}{\partial\theta}]^T\end{aligned}$$

is a Monte Carlo approximation of $-\frac{\partial^2 f(\theta)}{\partial\theta\partial\theta^T}$.

Our two-step refinement is specified as follows. Let θ^{r_1} be

$$\theta^{r_1} := \hat{\theta} - \frac{\nabla H(\hat{\theta})}{\nabla^2 H(\hat{\theta})},$$

where $\hat{\theta}$ is the estimator obtained from Algorithm 1 and is in the $n^{-\alpha/3}$-neighborhood of θ^*. We further define θ^{r_2} as

$$\theta^{r_2} := \theta^{r_1} - \frac{\nabla H(\theta^{r_1})}{\nabla^2 H(\theta^{r_1})}.$$

By such construction, we can show that θ^{r_2} is a root-n consistent and has asymptotic normal distribution when $3/4 < \alpha \leq 3/2$.

Theorem 2 *When $3/4 < \alpha \leq 3/2$, θ^{r_1} is $\sqrt{n}$-consistent estimator and*

$$\sqrt{n}(\theta^{r_2} - \theta^*) \to N(0, I^{-1}(\theta^*)V(\theta^*)I^{-1}(\theta^*)),$$

where $I(\theta^*)$ is the fisher information matrix and $V(\theta^*)$ is $\mathbb{E}\nabla H(\theta^*)\nabla H(\theta^*)^T$.

Here, variance $I^{-1}(\theta^*)V(\theta^*)I^{-1}(\theta^*)$ is larger than Cramer–Rao lower bound since we need sampling for latent variable z_i's.

4.2. Non-smooth Case

Next we consider situation when $R(\theta) \neq 0$. We define the stopping time $T(\epsilon) := \arg\min_s \min_t \mathbb{E}P(\theta_t^s) - P(\bar{\theta}) \leq \epsilon$, where $\mathbb{E}$ is still the conditional expectation given data and $\{B^s\}$. Quantity $T(\epsilon)$ is the first time that $\mathbb{E}P(\theta_t^s)$ drops below the $P(\bar{\theta})$ plus the threshold ϵ.

A5 Assume that $P(\theta)$ is strongly-convex in $\mathcal{B}_1 = \{\beta \in \mathcal{B} : \|\theta - \theta^*\| \leq \delta_0\}$, where $\delta_0 := \|\theta^* - \theta^0\|$.

Theorem 3 *Under Assumptions A1 - A5, then*

$$T(\epsilon) \leq C_2 \frac{P(\theta^0) - P(\bar{\theta})}{\epsilon m \gamma}$$

holds for any $\epsilon \geq \Omega(\max\{\frac{1}{\sqrt{n_1}}, m\gamma\})$, with probability going to 1 as $n \to \infty$. Here, C_1 is a universal constant.

Theorem 3 gives the convergence guarantee when the initial point lies in the region where the objective has strong convexity. By special choice of m, n_1 and γ, we have the following corollary.

Corollary 2 *Especially, we let $n_1 = n^{2/3\alpha}$, $m = n^{2/3\alpha}$, $\gamma = n^{-\alpha}$. Then the total computational complexity is $O(n^\alpha/\epsilon)$.*

Application for Sparse Learning

In particular, we take $R(\theta)$ as ℓ_1 norm $\|\theta\|_1$ to enforce the solution to be sparse. That is, we aim to solve

$$\hat{\theta} = \arg\min_\theta - \log L(\theta) + \tau\|\theta\|_1, \tag{5}$$

where τ is a tuning parameter controlling the penalty level. Then the proposed algorithm can recover the true support set of θ^*. We define $\mathcal{S}^* := \text{supp}(\theta^*)$ as the support of θ^* and define $\hat{\mathcal{S}} := \text{supp}(\hat{\theta})$ as the estimated support.

We further let S_1^* be the set of indices corresponding to position of θ^* where true value is non-zero and S_0^* be the set of indices corresponding to position of θ^* where true value is zero. For notational simplicity, we define $\theta_{(1)} = \theta[S_1^*]$ and $\theta_{(0)} = \theta[S_0^*]$. We then can write gradient and Hessian matrix in the block format, that is,

$$\nabla F(\theta) = \begin{pmatrix}\nabla_1 F(\theta) \\ \nabla_0 F(\theta)\end{pmatrix},$$

and

$$\nabla^2 F(\theta) = \begin{pmatrix}\nabla_{11}^2 F(\theta) & \nabla_{10}^2 F(\theta) \\ \nabla_{01}^2 F(\theta) & \nabla_{00}^2 F(\theta)\end{pmatrix},$$

where $\nabla_1 F(\theta)$ is the subvector of gradient corresponding to $\theta_{(1)}$ and $\nabla_{11}^2 F(\theta)$ is the block of Hessian matrix corresponding to $\theta_{(1)}$. $\nabla_1 F(\theta), \nabla_{10}^2 F(\theta), \nabla_{01}^2 F(\theta)$ and $\nabla_{00}^2 F(\theta)$ are defined in the same fashion.

We then introduce the following irrepresentable condition (Zhao and Yu, 2006).

A6 Assume there exists a positive constant η such that

$$|\nabla_{01} F(\theta^*)\nabla_{11}^2 F(\theta^*))^{-1}\text{sign}(\theta_{(1)}^*)| \preceq 1 - \eta.$$

Here the "$\preceq$" means that the inequality holds element-wisely.

Given such irrepresentable condition, we then have the following result.

Theorem 4 *Under Assumptions A1 - A6 with $\theta^{(0)}$ in the neighbor of θ^*, then $\hat{\mathcal{S}} = \mathcal{S}^*$ holds with probability tending to 1 as $n \to \infty$, if we set $\tau = n^{\kappa}$ with $-\frac{\alpha}{6} < \kappa < 0$.*

5. Analysis in Network Case

The analysis and algorithm in the previous section depend on the assumption that individuals are independent and identically distributed. However, in the latent network models, the individuals are no longer assumed to be mutually independent. In this section, we aim to solve the optimization problem under network setting.

5.1. Algorithm

Similar to Algorithm 1, we still consider a gradient-based method via variance-reduction technique. The main procedure is summarized as follows. We first randomly choose an initial point θ^0 and sample an initial latent vector $\mathbf{z}^0$. In iteration s, we first randomly sample a batch set of data B^s with size $n_1 < n$. For each data $i \in B^s$, we sample its corresponding latent variable z_i according to posterior distribution $p_\theta(z_i|y_i, \mathbf{z}_{-i})$ and update latent vector to get $\mathbf{z}^s$. We then compute a snapshot of stochastic gradient, that is,

$$\tilde{\nabla} H^{s+1} = \frac{1}{\sum_{i \in B^s} d_i} \sum_{i \in B^s} \nabla H_i(\tilde{\theta}^s, \mathbf{z}^s),$$

where $H_i(\theta, \mathbf{z}) := -\sum_{j:(i,j)\in A} \log f_\theta(y_{ij}|z_i, z_j)$. For each iteration s, we further sequentially update the parameter by m times. For each $t \in [m]$, we use similar procedure to compute the variance-reduced gradient as that in Algorithm 1. Again scalar γ represents the step size/learning rate. See Algorithm 2 for the complete procedure.

Several distinct features are described here. 1) We need to maintain the vector of latent membership, since the individuals are no longer independent of each other. The conditional posterior distribution of z_i's depends on the node(s) that i connects to. 2) In outer loop s, each node has equal probability to be included in batch set such that every node has chance to get its latent membership updated even if it has small number of degree. 3) In the inner loop, the node is sampled according to its degree. Therefore, a node with larger degree is more likely to impact the gradient.

5.2. Analysis

We establish the convergence property for Algorithm 2 in this section. We first introduce several assumptions.

Algorithm 2 Latent Network Stochastic Gradient Algorithm.

```
Input: Observations
Output: Estimated parameter
Initialization: Set initial parameter θ^0 and let θ_0^0 = θ̃^0 = θ^0. For i ∈ 1, ..., n, sample z_i^0 according to
    initial prior distribution. Denote z^0 = (z_1^0, ..., z_n^0).
for s = 0 to S - 1 do
    θ_0^{s+1} = θ_m^s
    Sample a subset B^s ⊂ {1, ..., n} with size n_1
    Sample new z_i according to approximate posterior distribution for i ∈ B^s and replace the old value to get z^s.
    Compute ∇̃H^{s+1} = 1/(Σ_{i∈B^s} d_i) Σ_{i∈B^s} ∇H_i(θ̃^s, z^s).
    for t = 0 to m − 1 do
        Randomly pick i_t^{s+1} from {1, ..., n} according to distribution p_i ∝ d_i.
        Sample latent variable z_{i_t^{s+1}} according to current posterior distribution to replace its old value and get z_t^{s+1}.
        Compute v_t^{s+1} = 1/d_{i_t^{s+1}} {∇H_{i_t^{s+1}}(θ_t^{s+1}, z_t^{s+1}) − ∇H_{i_t^{s+1}}(θ̃^s, z_t^{s+1})} + ∇̃H^{s+1}.
        Update θ_{t+1}^{s+1} = θ_t^{s+1} − γ v_t^{s+1}.
    end for
    Set θ̃^{s+1} = θ_m^{s+1}.
end for
```

B1 The parameter θ lies in a bounded compact set $\mathcal{B} \subset \mathbb{R}^p$.

B2 The density $f_\theta(y_{ij}|z_i, z_j)$ is a smooth function of θ for all z_i, z_j's.

B3 $\mathbb{E}_{y\sim f_{\theta^*}}[\log f_\theta(y)]$ is a strictly concave function of θ over set $\mathcal{B}$.

B4 Let $d_{min} := \min_{i\in[n]} d_i$ be the minimal degree. Assume $d_{min} = n^{\alpha_0}$ ($\alpha_0 > 0$).

We define $T(\epsilon) := \arg\min_s \min_t \mathbb{E}\|\nabla F(\theta_t^s)\|^2 \leq \epsilon$. We then have the following local convergence results.

Theorem 5 *Under Assumption B1 - B4, there exists δ such that,*

$$T(\epsilon) \leq C \frac{F(\theta^0) - F(\bar{\theta})}{\epsilon m \gamma}$$

holds for any $\epsilon = \Omega(\max\{\frac{1}{n_1 n^{\alpha_0}}, (m\gamma)^2\})$ and $\theta^0 \in B(\theta^*, \delta)$, with probability tending to 1 as $n \to \infty$.

Since the optimization problem over a graph is NP-hard problem, Theorem 5 only guarantees that the estimator will converge to the global optimal solution when the initial point is not far from true one. Given special choice of m, n_1 and γ, we have the following corollary.

Corollary 3 *Additionally, we assume* $d_{min} \asymp d_{max} = n^{\alpha_0}$*, where* $d_{max} := \max_{i \in [n]} d_i$*. Let* $m = n^{\alpha_1}$*,* $\gamma = n^{-\alpha}$*,* $n_1 = n^{2(\alpha - \alpha_1)}/d_0$ *and* $\alpha_1 = \frac{2}{3}(\alpha - \alpha_0)$*, then the total computational complexity is* $O(n^{\alpha + \alpha_0}/\epsilon)$ *for* $\alpha_0/2 < \alpha < 1$*.*

To end this section, we provide a brief discussion on the stability of gradient. It is known that the nodes with larger degrees will impact the gradient more. For networks with a few high degree nodes and more low degree nodes will this make the gradient calculation unstable. Let $\mathcal{N}_{small}$ be the set of nodes whose degree is $n^{o(1)}$. If we additionally assume that $(\sum_{i \in \mathcal{N}_{small}} d_i)/|A| \to 0$, then the gradient will not be affected by low degree nodes too much and thus becomes stable. It remains an open question that how unstable the gradient is when the graph becomes super sparse, i.e., $\lim\inf_n (\sum_{i \in \mathcal{N}_{small}} d_i)/|A| > 0$.

6. Numerical Experiments

DINA Model We consider the deterministic-input, noisy-and-gate (DINA; Rupp and Templin, 2008) model which is a special restricted latent class model. It is widely applied in psychometric and educational testing to make diagnosis of examinees. Suppose there are J items and let Y_j be the response to j-th item. Y_j takes value in $\{0, 1\}$; "1" means correct and "0" means incorrect. The model adopts the following formulation. For each examinee, he/she is associated with a latent vector $Z = (\alpha_1, \ldots, \alpha_K) \in \{0, 1\}^K$, where α_k is interpreted as k-th skill/attribute and K is the total number of attributes.

$$P(Y_j = 1|Z) = (1 - s_j)^{\xi(Z,Q)} g_j^{1-\xi(Z,Q)}, \tag{6}$$

where Q is a binary matrix specifying the relationship between item and attribute and $\xi(Z, Q) = \prod_k \mathbf{1}\{Z_k \geq Q_{jk}\}$. In other words, if the examinee has all attributes required by j-th item, he/she will have higher probability $(1 - s_j)$ to answer the j-th item correctly. Otherwise, he/she will only have probability (g_j) to answer correctly. Thus s_j and g_j can be interpreted as slipping and guessing parameters.

We generate the data based on DINA model with $n = 2000$ individuals and $J = 30$ items. We generate true s_j's and g_j's within $[0, 0.2]$. We compare the proposed method with "batch method" (gradient is computed based on the batch set instead of using VR-gradient) and "full batch method" (gradient is computed based on full data set). We set $m = n_1 = n^{2\alpha/3}$ and $\gamma = n^{-\alpha}$ ($\alpha = 1.2$). For batch and full batch methods, we scale the step sizes such that they roughly have the same magnitude per sample. The results are reported in Figure 1. We can see that the proposed method converges faster compared with other two methods. We can also see that the proposed method achieves faster convergence rate when α decreases, while the error becomes larger. This is consistent with our theoretical results.

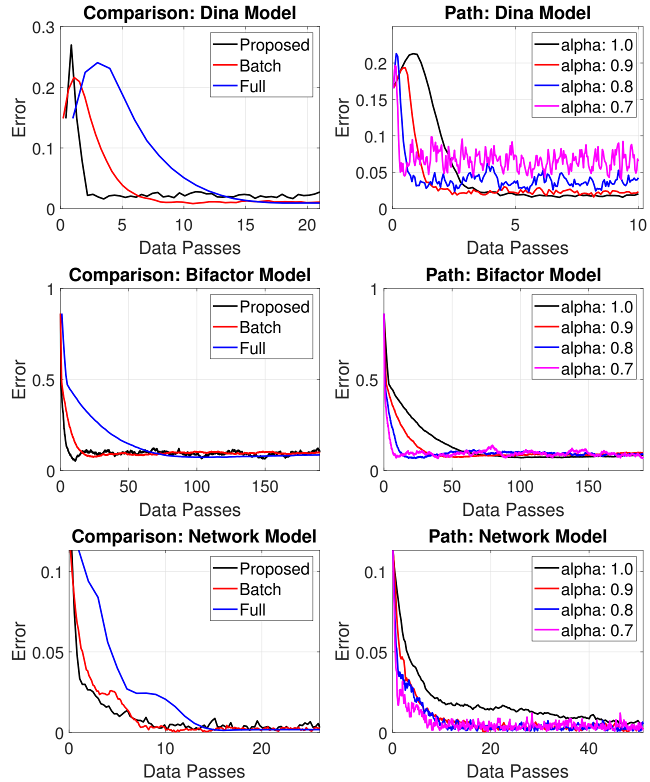

Figure 1. The simulation results for DINA, bifactor and latent network model. The plots on left column are the comparisons between three different methods. The plots on right column show the solution paths under different α's which take values in $\{0.7, 0.8, 0.9, 1.0\}$. Here "# Data Passes" is the cumulative number of samples divided by n.

Bifactor Model Bifactor model (Reise, 2012) is a latent factor model where the loading matrix admits a special structure. The first column of loading matrix is known as the main factor/dimension. Rest of columns are known as the sub-domain factor/dimension. Different from the DINA model, latent variable Z is continuous instead of discrete. Bifactor model postulates the following formulation

$$P(Y_j = 1|Z) = \frac{\exp\{a_{j0} + \mathbf{a}_j^T Z\}}{1 + \exp\{a_{j0} + \mathbf{a}_j^T Z\}}, \tag{7}$$

where $Z \in \mathbb{R}^{G+1}$ follows $N(\mathbf{0}, I_{(G+1)\times(G+1)})$. For each j, loading $\mathbf{a}_j$ has only one non-zero entries excluding the main dimension. Thus, the model parameter is identifiable and does not have rotational indeterminacy issue.

With knowing positions of non-zero entries of loading matrix, we generate the data from bifactor model with $n = 2000$ and $J = 15$. Main factor loading a_{j0}'s are sampled from $N(0, 2)$ and non-zero entries of testlet factor loading $\mathbf{a}_j$ are set to be 0.5. We use the similar strategy to choose m, n_1 and γ as that in DINA model setting with $\alpha = 0.9$. Note that it is prohibitively hard to compute the exact posterior distribution. We instead use MCMC to sample latent variables. From Figure 1, we observe that the proposed model again converges faster than other two methods.

Latent Network Model We consider a latent network model with $n = 400$ nodes and set the number of latent classes $K = 3$. The edge list A is constructed by randomly generating a subset of $[n] \times [n]$. For each pair $(i, j) \in A$, y_{ij} is a homogeneous Poisson process over time period $[0, T]$ ($T = 10$) with intensity parameter $\theta_{z_i z_j}$. Here $\theta_{kl} = \theta_{lk}$, and their true values are generated from the interval $[1, 5]$. We again compare three methods and set $m = n_1 = n^{2\alpha/3}$ and $\gamma = n^{-\alpha}$ ($\alpha = 0.9$). The result is shown in Figure 1. From solution path, we can see that the numerical results match our theoretical findings that smaller α can leader to faster convergence and a little bit larger bias.

Refinement and Support Recovery The performance of refined estimator under DINA model is shown in Figure 2. Here, the sample size varies from $n = 500$ to $n = 8000$. Based on the curve, we can see that the decay rate of estimation error is almost $n^{-1/2}$, which is consistent with our theory. For bifactor model, we further consider the situations without knowing the positions of non-zero entries. That is, we need to impose a regularization term to make the loading matrix sparse. A minimax concave penalty (MCP; Zhang, 2010) is considered.

$$R(\theta) = \begin{cases} \lambda|\theta| - \frac{\theta^2}{2a} & \text{if } |\theta| \le a\lambda \\ a\lambda^2/2 & \text{if } |\theta| > a\lambda. \end{cases}$$

By this choice, the formula of proximal operator can be simplified as

$$\text{proximal}_{\gamma R}(\theta) = sgn(\theta)\frac{a}{a-1}\max\{|\theta| - \lambda\gamma, 0\},$$

where $sgn(x)$ represents the sign of scalar x and tuning parameter a is larger than 1. The recovery results for support of loading matrix is shown in Figure 2. The proposed method can identify the non-zero positions well when we choose suitable penalty level.

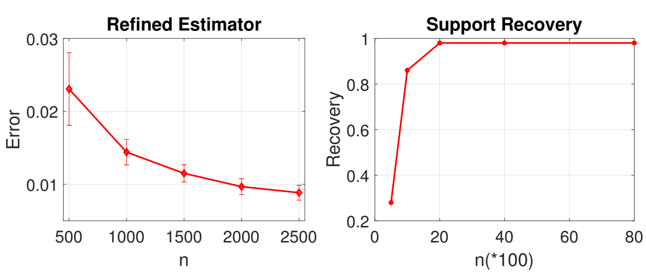

Figure 2. The left plot shows the estimation error of refined estimator under different sample sizes. The right plot shows the support recovery of loading matrix under different sample sizes. ($\lambda = \log n/(n^{1/2}\gamma), a = 3.$)

NESARC Data The data extracted from National Epidemiological Survey on Alcohol and Related Conditions (NESARC) concerning social phobia contains the responses of 728 respondents to 13 questions. We apply DINA model to fit this data set. We compare the proposed method with stochastic gradient method (SGD) and the result in shown in Figure 3. Here we set $m = n^{\frac{2}{3}\alpha}$ and $\gamma = n^{-\frac{2}{3}\alpha}$ with $\alpha = 1.4$ for both methods. We can see that the solution of SGD has a larger variance while the proposed method is more stable.

PISA Data The data was collected in the collaborative problem solving (CPS) test from 2015 Programme for International Student Assessment (PISA). Students were chosen from all the OECD countries and regions where the English version of exam was administrated. The dataset contains 8856 students in total. Each student has responses to 29 questions. We use bifactor model to fit the data set. The solution paths of proposed method and SGD are given in Figure 3. In particular, we set $m = n^{\frac{2}{3}\alpha}$ and $\gamma = n^{-0.3\alpha}$ with $\alpha = 1.0$ for both methods. We can see that SGD has a relatively larger bias compared with the proposed method.

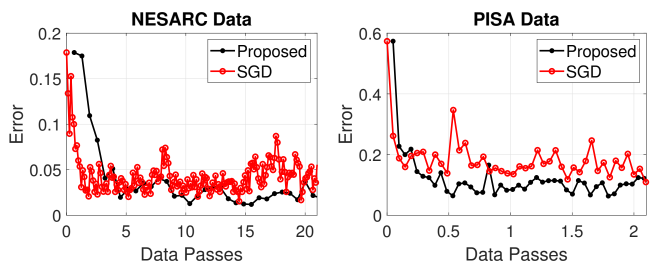

Figure 3. The right plot is for NESARC data and the left plot is for PISA data. The "Error" represents the difference between the current estimate and the optimal parameter (optimal parameter is computed via using full batch method).

7. Conclusion

In this paper, we consider an optimization problem for general latent variable models. Our proposed algorithm is a gradient-based method by adopting variation reduction technique. Our method does not require to compute the exact posterior distribution, which increases computational efficiency. The theoretical analysis is established and accommodates for both smooth and non-smooth settings. The theory also considers different types of statistical assumptions (i.e., data is independent and identically distributed or follows a network model). The numerical results match our theoretical findings. In future work, it is of interest to study model-specific algorithm. The structures of different latent variable models/simple neural networks may vary from one to another. In addition, one may also consider Adam (Kingma and Ba, 2015) or Nesterov's accelerated method (Nesterov, 1983) for computing the gradient. Then the variance-reduced step might be further improved to accelerate the algorithm.

References

Amr Ahmed, Mohamed Aly, Joseph Gonzalez, Shravan M. Narayanamurthy, and Alexander J. Smola. Scalable inference in latent variable models. In *Proceedings of the Fifth International Conference on Web Search and Web Data Mining (WSDM)*, pages 123–132, Seattle, WA, 2012.

Dennis J Aigner, Cheng Hsiao, Arie Kapteyn, and Tom Wansbeek. Latent variable models in econometrics. *Handbook of econometrics*, 2:1321–1393, 1984.

Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 699–707, New York City, NY, 2016.

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to MCMC for machine learning. *Machine learning*, 50(1):5–43, 2003.

Yves F Atchadé, Gersende Fort, and Eric Moulines. On perturbed proximal gradient algorithms. *The Journal of Machine Learning Research*, 18(1):310–342, 2017.

Hagai Attias. Inferring parameters and structure of latent variable models by variational bayes. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 21–30, Stockholm, Sweden, 1999.

David J Bartholomew, Martin Knott, and Irini Moustaki. *Latent variable models and factor analysis: A unified approach*, volume 904. John Wiley & Sons, 2011.

Zygmund William Birnbaum. On the importance of different components in a multicomponent system. Technical report, Washington Univ Seattle Lab of Statistical Research, 1968.

Christopher M. Bishop. Latent variable models. In *Learning in Graphical Models*, pages 371–403. 1998.

Russel E Caflisch et al. Monte carlo and quasi-monte carlo methods. *Acta numerica*, 1998:1–49, 1998.

Li Cai. High-dimensional exploratory item factor analysis by a metropolis–hastings robbins–monro algorithm. *Psychometrika*, 75(1):33–57, 2010.

Yunxiao Chen, Xiaoou Li, and Siliang Zhang. Structured latent factor analysis for large-scale data: Identifiability, estimability, and their implications. *Journal of the American Statistical Association*, 115(532):1756–1770, 2020.

Kumar Avinava Dubey, Sashank J. Reddi, Sinead A. Williamson, Barnabás Póczos, Alexander J. Smola, and Eric P. Xing. Variance reduction in stochastic gradient langevin dynamics. pages 1154–1162, 2016.

Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

Wally R Gilks, Nicky G Best, and KKC Tan. Adaptive rejection metropolis sampling within gibbs sampling. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 44(4):455–472, 1995.

Walter R Gilks and Pascal Wild. Adaptive rejection sampling for gibbs sampling. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 41(2):337–348, 1992.

Ming Gao Gu and Fan Hui Kong. A stochastic approximation algorithm with markov chain monte-carlo method for incomplete data estimation problems. *Proceedings of the National Academy of Sciences*, 95(13):7270–7274, 1998.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1): 97–109, 1970.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NIPS)*, pages 315–323, Lake Tahoe, NV, 2013.

Belhal Karimi and Ping Li. Two-timescale stochastic em algorithms. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2021.

Belhal Karimi, Hoi-To Wai, Eric Moulines, and Marc Lavielle. On the global convergence of (fast) incremental expectation maximization methods. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2833–2843, Vancouver, Canada, 2019.

Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, 2015.

Jun S Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

John C Loehlin and A Alexander Beaujean. *Latent Variable Models: An Introduction to Factor, Path, and Structural Equation Analysis*. Taylor & Francis, 2016.

Thomas A Louis. Finding the observed information matrix when using the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 44(2):226–233, 1982.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady an ussr*, volume 269, pages 543–547, 1983.

Art Owen and Yi Zhou. Safe and effective importance sampling. *Journal of the American Statistical Association*, 95(449):135–143, 2000.

Art B Owen and Peter W Glynn. *Monte Carlo and Quasi-Monte Carlo Methods*. Springer, 2016.

Boris T Polyak. New stochastic approximation type procedures. *Automat. i Telemekh*, 7(98-107):2, 1990.

Sashank J. Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alexander J. Smola. Stochastic variance reduction for nonconvex optimization. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 314–323, New York City, NY, 2016.

Steven P Reise. The rediscovery of bifactor measurement models. *Multivariate Behavioral Research*, 47(5):667–696, 2012.

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

Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

Andre A Rupp and Jonathan Templin. The effects of q-matrix misspecification on parameter estimates and classification accuracy in the dina model. *Educational and Psychological Measurement*, 68(1):78–96, 2008.

David Ruppert. Efficient estimators from a slowly convergent robbins-monro procedure. *Cornell University Technical Report*, 1988.

Charles Spearman. "general intelligence," objectively determined and measured. *American Journal of Psychology*, 15:201–292, 1904.

Matthias von Davier and Young-Sun Lee. Handbook of diagnostic classification models. *Cham: Springer International Publishing*, 2019.

Jianwen Xie, Zilong Zheng, and Ping Li. Learning energy-based model with variational auto-encoder as amortized sampler. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, pages 10441–10451, Virtual Event, 2021.

Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2): 894–942, 2010.

Siliang Zhang and Yunxiao Chen. Computation for latent variable model estimation: A unified stochastic proximal framework. *arXiv preprint arXiv:2008.07214*, 2020.

Peng Zhao and Bin Yu. On model selection consistency of lasso. *The Journal of Machine Learning Research*, 7: 2541–2563, 2006.

Yang Zhao, Jianwen Xie, and Ping Li. Learning energy-based generative models via coarse-to-fine expanding and sampling. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, 2021.

Rongda Zhu, Lingxiao Wang, Chengxiang Zhai, and Quanquan Gu. High-dimensional variance-reduced stochastic gradient expectation-maximization algorithm. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 4180–4188, Sydney, Australia, 2017.

Difan Zou, Pan Xu, and Quanquan Gu. Stochastic variance-reduced Hamilton Monte Carlo methods. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 6028–6037, Stockholm, Sweden, 2018.