FAST ALGORITHMS FOR BAYESIAN VARIABLE SELECTION

BY

XICHEN HUANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Statistics
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Doctoral Committee:

Professor Feng Liang, Chair, Director of Research
Professor Georgios Fellouris
Professor Annie Qu
Professor Xiaofeng Shao

Abstract

Variable selection of regression and classification models is an important but challenging problem. There are generally two approaches, one based on penalized likelihood, and the other based on Bayesian framework. We focus on the Bayesian framework in which a hierarchical prior is imposed on all unknown parameters including the unknown variable set. The Bayesian approach has many advantages, for example, we can access unknown obtain the posterior distribution of the sub-models. And more accurate prediction may be obtained by model averaging.

However, as the posterior distribution of the model parameters is usually not in closed form, posterior inference that relies on Markov Chain Monte Carlo (MCMC) has high computational cost especially in high-dimensional settings, which makes Bayesian approaches undesirable. In order to deal with datasets with large number of features, we aim to develop fast algorithms for Bayesian variable selection, which approximate the true posterior distribution, but yet still return the right inference (at least asymptotically).

In this thesis, we start with a variational algorithm for linear regression. Our algorithm is based on the work by Carbonetto and Stephens (2012), and with essential modifications including updating scheme and truncation of posterior inclusion probabilities. We have shown that our algorithm achieves both frequentist and Bayesian variable selection consistency.

Then we extend our variational algorithm to logistic regression by incorporating the Pólya-Gamma data-augmentation trick (Polson et al., 2013), which links our algorithm for linear regression with logistic regression. However, as the variational algorithm needs to update the variational distribution of all the latent Pólya-Gamma random variables of the same

size of the observations at every iteration, this algorithm is slow when there are huge amount of observations, or even be infeasible when the data is too large to be loaded into computer memory. We propose an online algorithm for the logistic regression, under the framework of online convex optimization. Our algorithm is fast, and achieves similar accuracy (log-loss) as the state-of-art algorithm (Follow-the-Regularized-Proximal algorithm).

Dedicated to my family,

for their love, encouragement and sacrifices.

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisor Professor Feng Liang for her guidance, advice and encouragement through my thesis work. Her patience, passion and great knowledge stimulate, facilitate and significantly improve the efficacy and efficiency of my Ph.D. research. Her constant encouragement and countless inspirations carry me through all the difficulties during my doctoral life. Besides research, her suggestion and knowledge benifits my career development a lot. I cannot imagine my doctoral research and life without her mentorship and support.

Secondly, I would like to thank my committee members Professor Georgios Fellouris, Professor Annie Qu and Professor Xiaofeng Shao for their valuable suggestions to my thesis research. My doctoral life cannot be strong and smooth without their generous help.

I would also share my thanks to the faculty and staff members in the Statistics Department for providing a wonderful environment for study and research. I am also grateful to the fellow students who spent years with me for being fantastic company, especially Xiwei Tang, Shun Yao, Xiaolu Zhu, Jin Wang, Jianjun Hu, Yunbo Ouyang, and Linrui Gan.

Finally, I want to thank my parents, my girl friend Weihong Huang, and my best friends for their constant love, encouragement, support, and being in my life.

Table of Contents

Chapter 1

Introduction

Consider a regression problem, where we model the expectation of a response variable Y by a linear combination of a set of p features $(X_1, \ldots, X_p)$ via a link function g:

$$g(\mathbb{E}(Y)) = X_1\beta_1 + \cdots X_p\beta_p.$$

In this document, we focus on linear regression and logistic regression models.

In the past three decades or so, there has been an intense development on the estimation of a sparse regression model. Here "sparse" means that only a small fraction of β_j's is believed to be non-zero. Identifying the set $S = \{j : \beta_j \neq 0, j = 1, \ldots, p\}$ is often referred to as the variable selection problem.

The current approaches to variable selection can be roughly divided into two categories. One category contains approaches based on penalized likelihood, including the classical variable selection procedures like AIC/BIC and the more recent ones like LASSO (Tibshirani, 1994) and SCAD (Fan and Li, 2001). As the name suggested, the penalized likelihood approach estimates the regression parameter by minimizing the log-likelihood plus some penalty function on $\boldsymbol{\beta}$. With a proper choice of the penalty function, the solution $\hat{\boldsymbol{\beta}}$ will have some of its components to be exactly zero, that is, parameter estimation and variable selection are carried out simultaneously. For an overview of recent developments of penalized likelihood approaches to variable selection in high dimensions, see See Fan and Lv (2010) and Bühlmann and van de Geer (2011).

We focus on the other category, the Bayesian methods, which starts with a hierarchical

prior on all the unknown parameters. For example, a widely used prior on $\boldsymbol{\beta}$ is the so-called spike-and-slab prior (Mitchell and Beauchamp, 1988):

$$\beta_j|\gamma_j, \sigma^2 \sim \gamma_j \, \mathrm{N}(0, v_1\sigma^2) + (1-\gamma_j)\delta_0, \quad \text{for } j = 1, \ldots, p, \tag{1.1}$$

where δ_0 is a point mass at 0, and $\gamma_j = 1$ if the j-th variable is included and 0 otherwise. The p-dimensional binary vector $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)^T$, which serves as a model index for all the 2^p sub-models, is then modeled by a product of *i.i.d.* Bernoulli distributions with parameter θ.

An advantage of the Bayesian approach is that, in addition to the posterior distribution on $\boldsymbol{\beta}$, we can also obtain the posterior distribution on all the sub-models. For example, we can discuss the probability of a sub-model $\boldsymbol{\gamma}$ or the inclusion probability of a particular feature, which can be of more interest than a point estimate of $\boldsymbol{\beta}$. Further, for prediction, it is well-known that model combination or aggregation has a better performance than a single model (Breiman, 2001). The Bayesian approach for variable selection gives rise to a natural averaging scheme: the prediction from various sub-models can be averaged with respect to their posterior probabilities (Raftery et al., 1997; Clyde and George, 2004).

Despite the aforementioned advantages, in practice, Bayesian variable selection is less preferable than those penalization algorithms. A major disadvantage of Bayesian variable selection is the computing cost. The posterior distribution usually does not have a closed-form expression, so posterior inference has to reply on Markov chain Monte Carlo (MCMC) methods, which could be time consuming especially when the number of predictors is large.

In order to do Bayesian variable selection for large-scale datasets, we seek fast algorithms that approximate the true posterior distributions of the parameters. We propose variational algorithms for linear and logistic regressions. We also show that our algorithm of linear regression model achieves variable selection consistency in high dimensions from both frequentist and Bayesian point of view.

Large-scale data sets, arising naturally in many modern statistical applications nowadays, pose a big challenge for computation. When both the number of observations (n) and that of variables (p) are large, the data set cannot be loaded into the computer memory. As such, our VB algorithms that need to keep the whole data in the memory cannot be used directly. We propose an online algorithm that can deal with the case when the data set is too large to be loaded into the computer memory.

The following chapters are organized as follows: In Chapter 2, we propose a variational algorithm for Bayesian variable selection under linear regression setup. Our work is based on an existing work, and given our modifications, we show that the new algorithm achieves variable selection consistency, and is efficient for large p. We then extend this algorithm to logistic regression in Chapter 3 by adopting a recently proposed data-augmentation method. In order to deal with the case when there are so many observations that the whole data set cannot be loaded into computer memory, we propose an online logistic regression algorithm in Chapter 4.

1.1 Notation

We define some symbols that will be used in the following chapters.

For sequences $\{a_n\}_{n=1}^{\infty}$ and $\{b_n\}_{n=1}^{\infty}$, we write

- $a_n = O(b_n)$, if $\exists\ c \in \mathbb{R}^+$ and $n_0 \in \mathbb{N}$, s.t. $|a_n/b_n| \leq c$, $\forall n \geq n_0$;
- $a_n = o(b_n)$, if $\lim_{n\to\infty} a_n/b_n = 0$;
- $a_n \asymp b_n$, if $\exists\ c_1, c_2 \in \mathbb{R}^+$ and $n_0 \in \mathbb{N}$, s.t. $c_1 \leq |a_n/b_n| \leq c_2$, $\forall n \geq n_0$;
- $a_n \sim b_n$ (asymptotically equivalent), if $\lim_{n\to\infty} a_n/b_n = 1$;
- $a_n \prec b_n$ if $a_n = o(b_n)$, and $a_n \preceq b_n$ if $a_n = O(b_n)$.

For a random variable sequence $\{X_n\}_{n=1}^{\infty}$ and a constant sequence $\{a_n\}_{n=1}^{\infty}$, we write

- $X_n = O_p(a_n)$ if $\forall \varepsilon > 0$, $\exists M > 0$ s.t. $\mathbb{P}\left(|\frac{X_n}{a_n}| > M\right) < \varepsilon$, $\forall n$;
- $X_n = o_p(a_n)$ when $\lim_{n\to\infty} \mathbb{P}\left(|\frac{X_n}{a_n}| \geq \varepsilon\right) = 0$, $\forall \varepsilon > 0$.

For $a, b \in \mathbb{R}$, we write $a \bigvee b$ to represent the larger number of a and b. We use $a \bigwedge b$ to represent the smaller one of a and b.

Chapter 2

Variational Algorithms for Linear Regression

2.1 Introduction

In this chapter, we propose a variational algorithm for Bayesian variable selection. It is a deterministic algorithm, seeking an approximation of the true posterior distribution over $(\boldsymbol{\beta}, \boldsymbol{\gamma})$, instead of running an MCMC chain. It converges very fast and can scale for large-sized data sets. Our work is motivated by the variational algorithm proposed by Carbonetto and Stephens (2012). The two algorithms have same prior specification and the same set of variational parameters Θ.

The two algorithms, however, update the variational parameters differently. In the algorithm by Carbonetto and Stephens (2012), the parameters associated with each feature are updated sequentially given the others; such a *component-wise updating* scheme is prone to error accumulation especially when p is large and predictors are correlated. In our algorithm, the updating is simultaneous for all features, which we refer to as the *batch-wise updating* scheme, therefore is more robust to errors and correlations among predictors.

In addition to the batch-wise updating scheme, our algorithm has two other modifications: stopping early and rescaling the posterior variances. Those modifications turn out to be crucial for us to show our algorithm achieves both frequentist consistency and Bayesian consistency even when p diverges at an exponential rate of the sample size n. To the best of our knowledge, no asymptotic results on variational algorithms for Bayesian variable selection are available in the literature.

2.2 Model and Prior

Represent the linear regression model in a matrix form:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{2.1}$$

where $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)^T$ is a vector that contains n *i.i.d.* random errors generated from a normal distribution $\mathrm{N}(0, \sigma^2)$, $\mathbf{y}$ is the response vector of length n, $\mathbf{X} = (x_{ij})$ is an $n \times p$ design matrix, and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$ is the coefficient vector of length p. Like in many other variable selection algorithms, we center and scale the data as follows:

$$\sum_i y_i = 0, \quad \sum_i x_{ij} = 0, \quad \sum_i x_{ij}^2 = \|\mathbf{X}_j\|_2^2 = n,$$

where $\mathbf{X}_j$ denotes the j-th column of $\mathbf{X}$.

The hierarchical prior is specified as follows:

$$\begin{aligned} \beta_j|\gamma_j &\sim \gamma_j \,\mathrm{N}(0, v_1\sigma^2) + (1-\gamma_j)\delta_0, \\ \gamma_j &\overset{i.i.d.}{\sim} \mathrm{Bern}(\theta), \\ \sigma^2 &\sim \mathrm{IG}\left(\frac{\nu}{2}, \frac{\nu\lambda}{2}\right), \\ \theta &\sim \mathrm{Beta}(a_0, b_0), \end{aligned} \tag{2.2}$$

where $j = 1, \ldots, p$, and ν, λ, a_0 and b_0 are hyper-parameters.

2.3 Algorithm 1: A Gibbs Sampler

As the posterior distribution of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ is not in closed form, one way to do posterior inference is to use MCMC. We follow a similar approach by Geweke et al. (1996), and propose a Gibbs sampler in Algorithm 1 for the parameters including $(\beta_1, \ldots, \beta_p)$, $(\gamma_1, \ldots, \gamma_p)$, θ, and

σ^2. The detailed derivation can be found in Appendix A.1.

However, this MCMC approach is slow for large-scale datasets, so the algorithm may not be feasible for large number of predictors. Therefore, we seek for faster algorithms.

Algorithm 1 Gibbs Sampler for Bayesian Variable Selection in Linear Regression

initialize $(\mu_1, \ldots, \mu_p)$, $(\sigma_1^2, \ldots, \sigma_p^2)$, $(\phi_1, \ldots, \phi_p)$, θ, and σ^2.
repeat
  for j in $1 : p$ **do**
    $BF_j \leftarrow \frac{\theta}{1-\theta} \frac{1}{\sqrt{1+v_1 \|\mathbf{X}_j\|_2^2}} \exp\left\{\frac{\mu_j^2}{2\sigma_j^2}\right\}$
    Sample $\gamma_j \sim \text{Bernoulli}\left(\frac{1}{1+\exp\{-BF_j\}}\right)$
    if $\gamma_j = 1$ **then**
      $\mu_j \leftarrow \frac{(\mathbf{y}-\mathbf{X}_{[,-j]}\boldsymbol{\beta}_{[-j]})^T \mathbf{X}_j}{\|\mathbf{X}_j\|_2^2 + \frac{1}{v_1}}$
      $\sigma_j^2 \leftarrow \frac{\sigma^2}{\|\mathbf{X}_j\|_2^2 + \frac{1}{v_1}}$
      Sample $\beta_j \sim \text{N}(\mu_j, \sigma_j^2)$
    else
      $\beta_j \leftarrow 0$
    end if
  end for
  Sample $\sigma^2 \sim IG\left(\frac{n+\sum_{j=1}^p \gamma_j + \nu}{2}, \frac{1}{2}\left((\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) + \frac{1}{v_1}\sum_j \beta_j^2 + \nu\lambda\right)\right)$
  Sample $\theta \sim \text{Beta}\left(\sum_{j=1}^p \gamma_j + a_0, p - \sum_{j=1}^p \gamma_j + b_0\right)$
until Converge
return A sequence of $(\mu_1, \ldots, \mu_p)$, $(\sigma_1^2, \ldots, \sigma_p^2)$, $(\phi_1, \ldots, \phi_p)$, θ, and σ^2

2.4 Variational Approximation

2.4.1 A Variational EM Algorithm

Variational methods have been widely used in different models, such as the Graphic models (Jordan et al., 1999). In the ordinary variational Bayesian approach (Bishop, 2006), an approximating distribution Q of all the latent variables and parameters, which takes a factorized form of $\prod_j Q_j$, is selected from a restricted family of distributions $\mathcal{Q}$, such that

the negative KL-divergence from the true posterior P to Q is maximized, i.e.,

$$\max_{Q\in\mathcal{Q}} \mathbb{E}^Q \log \frac{P}{Q} = \int Q \log \frac{P}{Q}\, \mathrm{d}Q.$$

Then one can solve each Q_j sequentially by fixing other Q's until convergence.

Our variational algorithm is based on a framework slightly different from the ordinary variational approach, but a hybrid of Expectation-Maximization (EM) and variational. Next we give a general description of the framework we use for posterior inference.

Let (Θ_1, Θ_2) denote the set of parameters of interest, and η denote the hyper-parameters. The goal is to obtain an approximation of the posterior distribution on (Θ_1, Θ_2). Define the following objective function

$$\Omega(q_1, q_2, \eta) = \mathbb{E}^{q_1,q_2}_{\Theta_1,\Theta_2} \log \frac{\pi(\Theta_1, \Theta_2, \eta|\text{Data})}{q_1(\Theta_1)q_2(\Theta_2)}, \tag{2.3}$$

where q_1 and q_2 are distributions on Θ_1 and Θ_2 respectively. Our goal is to find q_1, q_2, and a point estimate $\hat{\eta}$ to minimize the objective function. We will refer to the estimate $\hat{\eta}$ as the Maximum a posteriori (MAP) estimate: if we optimize (2.3) with respect to $q(\Theta_1, \Theta_2)$ and η, instead of restricting $q(\Theta_1, \Theta_2)$ to take a product form, then the corresponding $\hat{\eta}$ will be exactly the MAP estimate of η.

Applying the framework above on the Bayesian variable selection model, we estimate the MAP for σ^2 and θ, and approximate the posteriors for β_j's and γ_j's. We assume the following form of approximating posterior distribution of $(\boldsymbol{\beta}, \boldsymbol{\gamma})$:

$$\beta_j|\gamma_j \sim \gamma_j f_j(\beta_j) + (1-\gamma_j)\delta_0, \tag{2.4}$$

$$\gamma_j \sim \text{Bern}(\phi_j), \tag{2.5}$$

where $f_j(\beta_j)$ is a probability density. That is, the approximating posterior distributions of (β_j, γ_j)'s are marginally independent, each γ_j is from $\text{Bern}(\phi_j)$, and given γ_j, β_j is either

equal to zero or from a distribution with density f_j. Equivalently, we have

$$q(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \prod_{j=1}^{p} q_j(\beta_j, \gamma_j) = \prod_{j=1}^{p} [\phi_j f_j(\beta_j)]^{\gamma_j} [(1-\phi_j)\delta_0(\beta_j)]^{1-\gamma_j},$$

Given all the information above, we define the following objective function for this problem

$$\Omega(q_1, \ldots, q_p, \theta, \sigma^2) = \mathbb{E}^{q_1,\ldots,q_p} \log \frac{p(\mathbf{y}|\boldsymbol{\beta}, \sigma^2)\pi(\boldsymbol{\beta}|\gamma)\pi(\boldsymbol{\gamma}|\theta)\pi(\theta)\pi(\sigma^2)}{q(\boldsymbol{\beta}, \boldsymbol{\gamma})}.$$

2.4.2 Algorithm 2: Component-wise VB

The first algorithm is similar to the variational algorithm proposed by Carbonetto and Stephens (2012). In detail, we iteratively update the approximating distributions of $q_j(\beta_j, \gamma_j)$'s, and the MAP estimates $\hat{\theta}$ and $\hat{\sigma}^2$. As the algorithm loops over the p dimensions feature by feature, we refer to it as a "component-wise" VB algorithm, to highlight its difference with Algorithm 3, which we shall propose.

Updating Equations

Update $q_j(\beta_j, \gamma_j)$. For some $j \in \{1, \ldots, p\}$, by fixing other approximating distributions and point estimates, we maximize the objective function by changing q_j. Then it can be shown that $f_j(\beta_j)$ is the probability density of a Normal distribution $\mathrm{N}(\beta_j|\mu_j, \sigma_j^2)$ (albeit we do not assume f_j to be a normal at the beginning) with

$$\mu_j = \frac{\mathbf{X}_j^T \mathbb{E}_{[-j]} \left(\mathbf{y} - \sum_{l \neq j} \mathbf{X}_l \beta_l \right)}{n + \frac{1}{v_1}},$$
$$\sigma_j^2 = \frac{\hat{\sigma}^2}{n + \frac{1}{v_1}},$$

where $\mathbb{E}_{[-j]}$ denotes the expectations over all the β_l's with $l \neq j$ with respect to the variational distributions (Carbonetto and Stephens, 2012). By symmetry, we know that other $f_j(\beta_j)$'s are also Normal distributions. As such, we can write μ_j as

$$\mu_j = \frac{\left(\mathbf{y} - \mathbf{X}_{[-j]}\bar{\boldsymbol{\beta}}_{[-j]}\right)^T \mathbf{X}_j}{n + \frac{1}{v_1}}, \tag{2.6}$$

where $\mathbf{X}_{[-j]}$ denotes the design matrix without the j-th column, and $\bar{\boldsymbol{\beta}} = (\phi_1\mu_1, \ldots, \phi_p\mu_p)^T$ is the mean of $\boldsymbol{\beta}$ w.r.t. $q(\boldsymbol{\beta}, \boldsymbol{\gamma})$.

The log-odds of ϕ_j can be updated as

$$\text{Logit}(\phi_j) = \text{Logit}(\hat{\theta}) + \frac{1}{2}\log\frac{\sigma_j^2}{v_1\hat{\sigma}^2} + \frac{\mu_j^2}{2\sigma_j^2}.$$

Update $\hat{\theta}$. The point estimate of θ is updated by

$$\hat{\theta} = \frac{\sum_{j=1}^p \phi_j + a_0 - 1}{p + a_0 + b_0 - 2}. \tag{2.7}$$

Update $\hat{\sigma}^2$. The point estimate of σ^2 is updated by

$$\hat{\sigma}^2 = \frac{\|\mathbf{y} - \mathbf{X}\bar{\boldsymbol{\beta}}\|_2^2 + \sum_{j=1}^p[(n(1-\phi_j) + 1/v_1)\phi_j\mu_j^2 + (n + 1/v_1)\phi_j\sigma_j^2] + \nu\lambda}{n + \prod_{j=1}^p \phi_j + \nu + 2}. \tag{2.8}$$

The detailed derivation for this algorithm can be found in Appendix A, and the pseudo-code is shown in Algorithm 2.

The Drawback of Algorithm 2 in High Dimension To reveal the potential drawback of Algorithm 2 when being applied on a high-dimensional data set, we examine its asymptotic property.

Assume the response $\mathbf{y}$ is generated from the normal linear regression model (2.1) with $\boldsymbol{\beta}^* \in \mathbb{R}^p$ being the true regression coefficients. Consider a relatively easy setting where the

Algorithm 2 Component-wise VB

initialize $(\mu_1, \ldots, \mu_p)$, $(\sigma_1^2, \ldots, \sigma_p^2)$, $(\phi_1, \ldots, \phi_p)$, $\hat{\theta}$, and $\hat{\sigma}^2$.
repeat
  for j in $1:p$ **do**
    $\mu_j \leftarrow \frac{\left(\mathbf{y}-\sum_{l\neq j}\mathbf{X}_l\phi_l\mu_l\right)^T\mathbf{X}_j}{n+\frac{1}{v_1}}$
    $\sigma_j^2 \leftarrow \frac{\hat{\sigma}^2}{n+\frac{1}{v_1}}$
    $\phi_j \leftarrow \text{Logit}^{-1}\left\{\log\frac{\hat{\theta}}{1-\hat{\theta}} + \frac{1}{2}\log\frac{\sigma_j^2}{v_1\hat{\sigma}^2} + \frac{\mu_j^2}{2\sigma_j^2}\right\}$
  end for
  $\hat{\theta} \leftarrow \frac{\sum_{j=1}^p \phi_j + a_0 - 1}{p+a_0+b_0-2}$
  $\hat{\sigma}^2 \leftarrow \frac{\|\mathbf{y}-\mathbf{X}\bar{\boldsymbol{\beta}}\|_2^2 + \sum_{j=1}^p[(n(1-\phi_j)+1/v_1)\phi_j\mu_j^2 + (n+1/v_1)\phi_j\sigma_j^2] + \nu\lambda}{n+\prod_{j=1}^p\phi_j+\nu+2}$
until Converge
return $(\mu_1, \ldots, \mu_p)$, $(\sigma_1^2, \ldots, \sigma_p^2)$, $(\phi_1, \ldots, \phi_p)$, $\hat{\theta}$, and $\hat{\sigma}^2$

minimal eigenvalue of $\mathbf{X}^T\mathbf{X}$ is $O(n)$, i.e., the correlation among columns of $\mathbf{X}$ is small, and our starting values for μ_j's and ϕ_j's are very close to the truth: $\phi_j = 0.99$ if $\beta_j^* \neq 0$, $\phi_j = 0.01$ if $\beta_j^* = 0$, and

$$\mu_j - \beta_j^* = O_p\left(\frac{1}{\sqrt{n}}\right), \text{ for all } j = 1, \ldots, p.$$

Then, suppose we are updating the parameters associated with the j-th feature $(\mu_j, \sigma_j^2, \phi_j)$. After the update, will μ_j and ϕ_j still be close to the truth?

From Eq (2.6) we have

$$\begin{aligned}\mu_j =& \frac{\mathbf{X}_j^T(\mathbf{y} - \mathbf{X}_{[-j]}\bar{\boldsymbol{\beta}}_{[-j]})}{n + \frac{1}{v_1}} \\ =& \frac{nv_1}{nv_1+1}\Big[\beta_j^* + \frac{1}{n}\mathbf{X}_j^T\mathbf{X}_{[-j]}\big(\boldsymbol{\beta}_{[-j]}^* - \bar{\boldsymbol{\beta}}_{[-j]}\big) + \frac{1}{n}\mathbf{X}_j^T\boldsymbol{\epsilon}\Big].\end{aligned}$$

Suppose v_1 is chosen such that $v_1 n \to \infty$, a condition required for consistency as will be made clear in our analysis in Section 2.5. Then, we have

$$\mu_j = \beta_j^* + O_p\left(\frac{p}{\sqrt{n}}\right). \tag{2.9}$$

The result above shows the price we pay for Algorithm 2: even if we start with μ_j within a

$1/\sqrt{n}$ ball around the truth β_j^*, after the update, the new μ_j could be very far away from β_j^* when p is large, due to the accumulation of the errors from other dimensions via $\mathbf{X}_{[-j]}\bar{\boldsymbol{\beta}}_{[-j]}$.

Next we examine how ϕ_j is affected by the update. Suppose the j-th feature is an irrelevant feature, i.e., $\beta_j^* = 0$. The new log-odds of ϕ_j is computed as

$$\begin{aligned}\text{Logit}(\phi_j) =& \text{Logit}(\hat{\theta}) + \frac{1}{2}\log\frac{\sigma_j^2}{v_1\hat{\sigma}^2} + \frac{\mu_j^2}{2\sigma_j^2} \\ =& O(1) - \frac{1}{2}\log(v_1 n + 1) + \mu_j^2\frac{n + \frac{1}{v_1}}{2\hat{\sigma}^2},\end{aligned}$$

where $\text{Logit}(\hat{\theta}) = O(1)$ as long as we do not start with $\hat{\theta} = 0$ or 1. Since $\mu_j^2 = O_p\left(p^2/n\right)$ by (2.9), we have

$$2\,\text{Logit}(\phi_j) = -\log(v_1 n + 1) + O_p(p^2). \tag{2.10}$$

When p is very large, the right hand side of (2.10) may be positive. That is, the new ϕ_j could be bigger than 0.5, although we start with $\phi_j = 0.01$, a value that is very close to the truth.

Our analysis above is not rigorous, but it clearly reveals an issue with Algorithm 2: the noise can accumulate due to the feature by feature updating scheme. To address this issue, we propose another algorithm which updates $(\mu_1, \ldots, \mu_p)$ simultaneously for all p features.

2.4.3 Algorithm 3: Batch-wise VB

Recall the variational parameters we need to update are $\{\mu_j, \phi_j, \sigma_j^2\}_{j=1}^p$. At iteration t, instead of updating the triplet $\{\mu_j, \phi_j, \sigma_j^2\}$ sequentially for each j as in Algorithm 2, we consider the following batch-wise update: update $\{\sigma_j^2\}_{j=1}^p$, then update $\{\mu_j\}_{j=1}^p$, and finally update $\{\phi_j\}_{j=1}^p$. Given $\{\mu_j, \phi_j, \sigma_j^2\}_{j=1}^p$, we can update $(\hat{\theta}, \hat{\sigma}^2)$ using Eq (2.7) and Eq (2.8).

Updating Equations

Update $\{\sigma_j^2\}_{j=1}^p$. We update σ_j^2's by maximizing

$$\begin{aligned}
&\mathbb{E}^{q_1,\ldots,q_p}\left\{\sum_{i=1}^n\left[-\frac{(y_i-\mathbf{x}_i^T\boldsymbol{\beta})^2}{2\sigma^2}\right]+\sum_{j=1}^p\gamma_j\left[-\frac{\beta_j^2}{2v_1\sigma^2}-\log[f_j(\beta_j)]\right]\right\}\\
\propto\ &\sum_{j=1}^n-\phi_j\frac{n}{2\hat{\sigma}^2}\sigma_j^2-\phi_j\frac{1}{2v_1\hat{\sigma}^2}\sigma_j^2+\frac{1}{2}\phi_j\log(\sigma_j^2),
\end{aligned}$$

and the updating equation for σ_j^2 is

$$\sigma_j^2=\frac{\hat{\sigma}^2}{n+\frac{1}{v_1}},\quad j=1,\ldots,p. \tag{2.11}$$

Note that σ_j^2's take the same form for all j. The term n in the denominator is due to the fact that each column of $\mathbf{X}$ has been pre-processed such that $\|\mathbf{X}_j\|^2=n$. Later in Section 2.5, in light of the asymptotic analysis, we will suggest to replaced n by $a_n\asymp n^a$ as in Eq (2.21).

Update $\{\mu_j\}_{j=1}^p$. We update μ_j's by maximizing

$$\mathbb{E}^{q_1,\ldots,q_p}\left\{\sum_{i=1}^n\left[-\frac{(y_i-\mathbf{x}_i^T\boldsymbol{\beta})^2}{2\sigma^2}\right]+\sum_{j=1}^p\gamma_j\left[-\frac{\beta_j^2}{2v_1\sigma^2}-\log[f_j(\beta_j)]\right]\right\},$$

and the updating equation for $\boldsymbol{\mu}$ is

$$\boldsymbol{\mu}=\left(\boldsymbol{\Phi}\mathbf{X}^T\mathbf{X}\boldsymbol{\Phi}+\boldsymbol{\Delta}+\frac{1}{v_1}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}\mathbf{X}^T\mathbf{y}, \tag{2.12}$$

where $\boldsymbol{\Delta}=\mathrm{diag}\{\mathbf{X}^T\mathbf{X}\}\boldsymbol{\Phi}(\mathbf{I}-\boldsymbol{\Phi})=n\boldsymbol{\Phi}(\mathbf{I}-\boldsymbol{\Phi})$.

Update $\{\phi_j\}_{j=1}^p$. The objective function at this step involves a quadratic form of $\boldsymbol{\phi}=(\phi_1,\ldots,\phi_p)^T$. To simplify the computation, we apply a linear approximation to replace the quadratic term. The detailed derivation can be found in Appendix A, and the final

updating equation for ϕ_j is given by

$$\begin{aligned}\text{Logit}(\phi_j) &= \text{Logit}(\hat{\theta}) + \frac{1}{2}\log\frac{\sigma_j^2}{v_1\hat{\sigma}^2} + \frac{\mu_j^2}{2\hat{\sigma}^2}\left(n + \frac{1}{v_1}\right) \\ &= \text{Logit}(\hat{\theta}) + \frac{1}{2}\log\frac{\sigma_j^2}{v_1\hat{\sigma}^2} + \frac{\mu_j^2}{2\sigma_j^2}. \end{aligned} \tag{2.13}$$

We stop updating any ϕ_j's once they are close to 0 or 1. That is, for $t > 1$,

$$\phi_j^{(t)} = \begin{cases} \text{Logit}^{-1}\left(\log\frac{\hat{\theta}}{1-\hat{\theta}} + \frac{1}{2}\log\frac{\sigma_j^2}{v_1\hat{\sigma}^2} + \frac{\mu_j^2}{2\sigma_j^2}\right), & \text{if } \min\left(\phi_j^{(t-1)}, 1-\phi_j^{(t-1)}\right) > c; \\ \phi_j^{(t-1)}, & \text{if } \min\left(\phi_j^{(t-1)}, 1-\phi_j^{(t-1)}\right) \le c, \end{cases} \tag{2.14}$$

where $0 < c < 1$ is a small constant. This stop-early updating scheme can dramatically reduce the computation cost for our algorithm, as explained in Section 2.4.3.

Stopping Criterion. After we loop over all the parameters mentioned above, we need to decide when to stop. A natural choice is to stop when the change of the objective function is less than some threshold. Since our primary focus is variable selection, we use the maximum entropy criterion: we compute the entropy for each Bern(ϕ_j), and stop if the maximum of the change of the entropy is less than a pre-specified threshold.

Computational Complexity

The main computational cost lies in reverting a $p \times p$ matrix in Eq (2.12). The direct computation involves $O(p^3 \bigwedge n^3)$ operations which is time-consuming when both p and n are large. Next we describe the computation trick used in our implementation, which can dramatically reduce the computation cost.

At iteration t, Eq (2.12) can be rewritten as

$$\begin{aligned}\boldsymbol{\mu} &= \left(\mathbf{X}^T\mathbf{X}\boldsymbol{\Phi}^{(t)} + n(\mathbf{I} - \boldsymbol{\Phi}^{(t)}) + \frac{1}{v_1}\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y} \\ &= \left(A_{(t-1)} + \left(A_{(t)} - A_{(t-1)}\right)\right)^{-1}\mathbf{X}^T\mathbf{y},\end{aligned}$$

Algorithm 3 Batch-wise VB

Initialize $(\mu_1, \ldots, \mu_p)$, $(\sigma_1^2, \ldots, \sigma_p^2)$, $\hat{\theta}$, and $\hat{\sigma}^2$.
Initialize $(\phi_1, \ldots, \phi_p) = \mathbf{1}_{p\times 1}$, and truncation parameter c.
repeat
  $\boldsymbol{\mu} \leftarrow (\boldsymbol{\Phi}\mathbf{X}^T\mathbf{X}\boldsymbol{\Phi} + \boldsymbol{\Delta} + \frac{1}{v_1}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}\mathbf{X}^T\mathbf{y}$
  $\sigma_j^2 \leftarrow \frac{\hat{\sigma}^2}{a_n+1/v_1}$ or $\frac{\hat{\sigma}^2}{n+1/v_1}$
  for j in $1:p$ **do**
    if $\min(\phi_j, 1-\phi_j) > c$ **then**
      $\phi_j \leftarrow \text{Logit}^{-1}\left\{\text{Logit}(\hat{\theta}) + \frac{1}{2}\log\frac{\sigma_j^2}{v_1\hat{\sigma}^2} + \frac{\mu_j^2}{2\sigma_j^2}\right\}$
    end if
  end for
  $\hat{\theta} \leftarrow \frac{\sum_{j=1}^p \phi_j + a_0 - 1}{p+a_0+b_0-2}$
  $\hat{\sigma}^2 \leftarrow \frac{\|\mathbf{y}-\mathbf{X}\boldsymbol{\Phi}\boldsymbol{\mu}\|^2 + \sum_{j=1}^p [(\mathbf{X}_j^T\mathbf{X}_j(1-\phi_j)+1/v_1)\phi_j\mu_j^2 + (\mathbf{X}_j^T\mathbf{X}_j+1/v_1)\phi_j\sigma_j^2] + \frac{1}{v_1}\sum_{j=1}^p \phi_j(\mu_j^2+\sigma_j^2) + \nu\lambda}{n + \prod_{j=1}^p \phi_j + \nu + 2}$
until Converge
return $(\mu_1, \ldots, \mu_p)$, $(\sigma_1^2, \ldots, \sigma_p^2)$, $(\phi_1, \ldots, \phi_p)$, $\hat{\theta}$, and $\hat{\sigma}^2$

where $A_{(t)} = \left(\mathbf{X}^T\mathbf{X}\boldsymbol{\Phi}^{(t)} + n(\mathbf{I} - \boldsymbol{\Phi}^{(t)}) + \frac{1}{v_1}\mathbf{I}\right)$ and

$$A_{(t)} - A_{(t-1)} = \left(\mathbf{X}^T\mathbf{X} - n\mathbf{I}_p\right)\left(\boldsymbol{\Phi}^{(t)} - \boldsymbol{\Phi}^{(t-1)}\right).$$

At iteration $t > 1$, we would have $A_{(t-1)}^{-1}$ in hand. If the rank of $A_{(t)} - A_{(t-1)}$ is lower than p or n, then the problem can be reformulated as inverting a matrix of a lower rank.

Write $\mathbf{X}^T\mathbf{X} - n\mathbf{I}_p = B$ and $D^{(t)} = \boldsymbol{\Phi}^{(t)} - \boldsymbol{\Phi}^{(t-1)}$. Then $A_t - A_{(t-1)} = BD^{(t)}$. Based our experience, after the first several iterations, many ϕ_j's are close to 1 or 0, i.e., they will not be updated any more according to Eq (2.14). So many diagonal elements of $D^{(t)}$ are zero. Without loss of generality, assume only the first q elements of $D^{(t)}$ are not zero. Then we have $BD^{(t)} = UCV$, where $U = B_{[,1:q]}$ contains only the first q columns of B, $C = \text{diag}\{\phi_j^{(t)} - \phi_j^{(t-1)}\}_{j=1}^q$, and $V = (\mathbf{I}_q, \mathbf{0}_{q\times(p-q)})$. Applying the Woodbury formula, we have

$$\boldsymbol{\mu} = \left(A_{(t-1)}^{-1} - A_{(t-1)}^{-1}U\left(C^{-1} + VA_{(t-1)}^{-1}U\right)_{q\times q}^{-1}VA_{(t-1)}^{-1}\right)\mathbf{X}^T\mathbf{y}.$$

So we only need to invert a $q \times q$ matrix where q is much smaller than p and n.

2.5 Asymptotic Analysis

Assume the response $\mathbf{y}$ is generated from the normal linear regression model (2.1) with $\boldsymbol{\beta}^* \in \mathbb{R}^p$ being the true regression coefficients. Let $\boldsymbol{\gamma}^*$ denote the true model index, i.e., $\gamma_j^* = 1$ if $\beta_j^* \neq 0$ and $\gamma_j^* = 0$ if $\beta_j^* = 0$. Also define the true set of relevant variables as

$$S^* = \{j : \beta_j^* \neq 0\} = \{j : \gamma_j^* = 1\}.$$

In our analysis, the dimension $p = p_n$ is allowed to diverge with n, and therefore $\boldsymbol{\beta}^* = \boldsymbol{\beta}_n^*$, $\boldsymbol{\gamma}^* = \boldsymbol{\gamma}_n^*$ and $S^* = S_n^*$ may also vary with n.

We will show that Algorithm 3 achieves both the frequentist consistency and the Bayesian consistency. Recall that Algorithm 3 returns an estimate of the relevant variable set via

$$\hat{S}_n = \{j : \phi_j > 0.5\}.$$

The frequentist consistency refers to the convergence (in probability) of $\hat{S}_n$ to S^*. The cut-off value 0.5 can be changed to any other fixed value in $(0, 1)$, which will not affect the consistency result as shown in our analysis.

In addition to a point estimate of the true variable set, our algorithm also returns a probability distribution over all 2^p variable sets (or sub-models), namely,

$$q(\boldsymbol{\gamma}) = \prod_{j=1}^{p} \big(\phi_j\big)^{\gamma_j} \big(1 - \phi_j\big)^{1-\gamma_j}.$$

The aforementioned frequentist consistency corresponds to $q(\boldsymbol{\gamma}^*)$ is the largest, i.e., the truth model receives the largest posterior probability, while the Bayesian consistency requires $q(\boldsymbol{\gamma}^*)$ converges to 1 in probability, which is stronger than the frequentist consistency.

In addition to the ordinary regularity conditions, for our algorithm to achieve consistency, we need to let v_1, the prior variance on the non-zero β_j's as in Eq (2.2), to grow to infinity

at a certain rate of n. A similar condition also arises in the asymptotic study on Bayesian variable selection by Narisetty and He (2014) although their prior specification is different from ours. To help the readers to understand this condition, we first give the asymptotic analysis on a simple orthogonal design and then describe the general result.

2.5.1 The Orthogonal Design

Consider a simple case in which the design matrix is orthogonal, i.e., $\mathbf{X}^T\mathbf{X} = n$. To simplify our discussion, we also assume that σ^2 is known, θ is set to be $1/2$, and the minimal non-zero coefficient is bigger than some constant (i.e., the non-zero coefficients will not diminish to zero). These conditions will be relaxed in our result for the general case.

Suppose we run our algorithm for one step. From the updating equations of Algorithm 3, we have

$$2\,\mathrm{Logit}(\phi_j) = -\log(v_1 n + 1) + \frac{n\hat{\beta}_j^2}{\sigma^2}\frac{n}{n + \frac{1}{v_1}}, \tag{2.15}$$

where $\hat{\beta}_j = \frac{1}{n}\mathbf{X}_j^T\mathbf{y} \sim \mathrm{N}\left(\beta_j^*, \frac{\sigma^2}{n}\right)$ is the OLS estimator of the j-th coefficient.

When p is fixed, as in the classical asymptotic setting, it is easy to show that our algorithm has the desired asymptotic behavior as long as

$$v_1 n \to \infty, \quad \log(v_1 n) = o(n). \tag{2.16}$$

This is because: when $\beta_j^* = 0$, since $n\hat{\beta}_j^2 = O_p(1)$, the leading term in (2.15) is the first term that goes to $-\infty$, therefore ϕ_j goes to 0; when $\beta_j^* \neq 0$, the leading term in (2.15) is the second term that goes to ∞, therefore ϕ_j goes to 1.

When $p = p_n$ increases with n, the coefficients $(\beta_j^*)_{j=1}^p$ and the true variable set S_n^* may vary with n. As such, it is no longer meaningful to discuss the limit of Eq (2.15). Instead, we need to examine the limiting behavior of $\max_{j\notin S_n^*}\mathrm{Logit}(\phi_j)$ and $\min_{j\in S_n^*}\mathrm{Logit}(\phi_j)$.

First we show that the frequentist variable selection consistency could be achieved with

$p = O(\sqrt{v_1 n})$, in addition to condition (2.16). Let C be an arbitrary positive number. It suffices to show that

$$P\left(\max_{j\notin S_n^*} \text{Logit}(\phi_j) > -\frac{C}{2}\right) + P\left(\min_{j\in S_n^*} \text{Logit}(\phi_j) < \frac{C}{2}\right) \to 0. \tag{2.17}$$

By the Bonferroni correction and the tail probability inequality of the normal distribution, we have

$$\begin{aligned} & P\left(\max_{j\notin S_n^*} \text{Logit}(\phi_j) > -\frac{C}{2}\right) \\ \leq & \sum_{j\notin S_n^*} P\left(\frac{1}{\sigma^2} n\hat{\beta}_j^2 \frac{n}{n+\frac{1}{v_1}} > \log(v_1 n+1) - C\right) \\ \leq & \frac{c}{\sqrt{\log(v_1 n) - C}} \exp\left\{-\frac{\log(v_1 n) - C}{2} + \log p\right\} \to 0, \end{aligned} \tag{2.18}$$

as long as $p = O(\sqrt{v_1 n})$ and $v_1 n \to \infty$, where c is some constant. Similarly, we have

$$\begin{aligned} & P\left(\min_{j\in S_n^*} \text{Logit}(\phi_j) < \frac{C}{2}\right) \leq P\left(\frac{1}{2}\min_{j\in S_n^*} \frac{1}{\sigma^2} n\hat{\beta}_j^2 < \log(v_1 n+1) + C\right) \\ \leq & P\left(\frac{1}{\sigma}\sqrt{n}\min_{j\in S_n^*} |\beta_j^*| - \frac{1}{\sigma}\sqrt{n}\max_{j\in S_n^*} |\hat{\beta}_j - \beta_j^*| < \sqrt{2\log(v_1 n+1) + 2C}\right) \\ \leq & |S_n^*| P\left(\frac{1}{\sigma}\sqrt{n}|\hat{\beta}_j - \beta_j^*| > r\sqrt{n}\right), \end{aligned}$$

which also goes to zero by the tail probability of the normal distribution, where r is some constant.

Secondly, we show that the Bayesian consistency could be achieved with $p = o(\sqrt{v_1 n})$. We can show that, if $p = o(\sqrt{v_1 n})$, Eq (2.17) still holds with a varying constant $C_n = s\log(v_1 n)$ where $s \in (0, 1)$. Then with probability going to 1, we have

$$\max_{j\in S_n^*}(1-\phi_j) \bigvee \max_{j\notin S_n^*} \phi_j < \frac{1}{\exp\{\frac{1}{2}C_n\}} < \frac{1}{(v_1 n)^{s/2}}.$$

Using the inequality $\prod_j (1-p_j) \geq 1 - \sum_j p_j$, we have

$$\begin{aligned} q_(\boldsymbol{\gamma}^*) = \Big[\prod_{j:\gamma_j^*=1} \phi_j \Big] \Big[\prod_{j:\gamma_j^*=0} (1-\phi_j) \Big] \geq 1 - \sum_{j \in S_n^*} (1-\phi_j) + \sum_{j \notin S_n^*} \phi_j \\ \geq 1 - \frac{p}{(v_1 n)^{s/2}} \to 1. \end{aligned}$$

Our analysis for the orthogonal case indicates that the choice of v_1 affects how fast we could allow p to diverge with n. For example, if v_1 is a constant, then our algorithm can achieve the frequentist consistency for $p = O(\sqrt{n})$ and Bayesian consistency for $p = o(\sqrt{n})$. In order to achieve consistency for larger p, we need to let $v_1 \to \infty$ with n.

2.5.2 The General Case

Without loss of generality, assume $S_n^* = \{1, ..., q_n\}$ and the true coefficient $\boldsymbol{\beta}^* = (\boldsymbol{\beta}_1^{*T}, 0^T)^T$, i.e, the first q_n features are the relevant ones. Write the design matrix as $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)$ accordingly, where $\mathbf{X}_1$ is the $n \times q_n$ matrix corresponding to the signal features and $\mathbf{X}_2$ is the $n \times (p - q_n)$ matrix corresponding to the noise features.

In our analysis, we assume the following conditions hold.

(C1) Condition on model identification: Denote H_1 and H_2 as the projection matrices of $\mathbf{X}_1$ and $\mathbf{X}_2$ onto their column spaces respectively, then assume the rank of H_1 is q_n and the spectral norm of $H_1 H_2$ is upper bounded by 1, i.e., $\|H_1 H_2\|_2 < 1$.

(C2) Condition on the design matrix: Let λ_{n1} denote the minimal non-zero eigenvalue of matrix $\mathbf{X}^t\mathbf{X}$ and it satisfies

$$\lambda_{n1}^{-1} = O(n^{-\eta_1}), \quad 0 < \eta_1 \leq 1.$$

(C3) Condition on the sparsity of $\boldsymbol{\beta}^*$: The L_2 norm of the true regression coefficient $\boldsymbol{\beta}^*$

satisfies the following sparsity condition

$$\|\boldsymbol{\beta}^*\|_2^2 = O(n^{\eta_2}), \quad 0 \leq \eta_2 < \eta_1.$$

(C4) The beta-min condition: The minimal non-zero coefficient satisfies

$$\liminf_n \frac{\min_{j \in S_n^*} \sqrt{n}|\beta_j^*|}{n^{\eta_3/2}} \geq M, \tag{2.19}$$

where $\eta_3 \in (1 - \eta_1, 1]$ and $M > 0$ are two constants not depending on n.

(C5) Condition on the initial values: the initial value for all the inclusion probability $\phi_j^{(0)}$'s should be set to be 1, i.e., we start our algorithm with *all the variables in*. The initial values for the error variance $\hat{\sigma}^2$ and the Bernoulli parameter $\hat{\theta}$ could be any constants satisfying $0 < \hat{\sigma}^2 < \infty$ and $0 < \hat{\theta} < 1$. For the proof, we set

$$\phi_j^{(0)} = 1, \quad \hat{\sigma}^{2(0)} = 1, \quad \hat{\theta}^{(0)} = \frac{1}{2}. \tag{2.20}$$

In general, it is not realistic to derive consistent variable selection procedures and parameter estimation when the design matrix $\mathbf{X}$ is not of full rank (Shao and Deng, 2012). This is because the true coefficient $\boldsymbol{\beta}^*$ could be any vector from the set $\mathcal{B} = \{\boldsymbol{\beta} : \mathbf{X}\boldsymbol{\beta} = \mathbf{X}\boldsymbol{\beta}^*\}$ due to the collinearity among the columns of $\mathbf{X}$. Condition (C1) ensures that the true sparse coefficient $\boldsymbol{\beta}^*$ is identifiable. Let $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^T, \boldsymbol{\beta}_2^T)^T$ be any vector from $\mathcal{B}$. Then we have $\mathbf{X}_1\boldsymbol{\beta}_1^* = \mathbf{X}_1\boldsymbol{\beta}_1 + \mathbf{X}_2\boldsymbol{\beta}_2$. Meanwhile (C1) implies that $\mathbf{X}_1\boldsymbol{\beta}_1^* = \mathbf{X}_1\boldsymbol{\beta}_1$. So $\|\boldsymbol{\beta}\|_2 \geq \|\boldsymbol{\beta}^*\|_2$ with equality holds true only if $\boldsymbol{\beta} = \boldsymbol{\beta}^*$. That is, the true coefficient vector $\boldsymbol{\beta}^*$ is the one from $\mathcal{B}$ with the smallest L_2 norm, which is unique.

In our algorithm, we approximate the posterior distribution of β_j by a mixture of a point mass at zero and a normal distribution. The updating equation (2.11) implies that the posterior variance σ_j^2 for non-zero β_j's is of order $1/n$, a reasonable result in the classical

asymptotic setting where p is fixed and the minimal eigenvalue of $\mathbf{X}^T\mathbf{X}$ is of $O(n)$. However, in the diverging p case, as indicated by (C2), the minimal non-zero eigenvalue of $\mathbf{X}^T\mathbf{X}$ could be of order $O(n^{\eta_1})$. Therefore the posterior variance σ_j^2 defined in Eq (2.11) would be too small. In other words, we are too optimistic about the uncertainty of β_j. This is a common issue with variational algorithms, as pointed out in Bishop (2006) and shown in Figure 2.1: the variational distribution tends to have a smaller support than the true target distribution. To fix this problem, we need to correct the posterior variance at the right order as follows:

$$\sigma_j^2 = \frac{\hat{\sigma}^2}{a_n + \frac{1}{v_1}}, \tag{2.21}$$

where $a_n \asymp n^a$ with $1 - \eta_3 < a < \eta_1$.

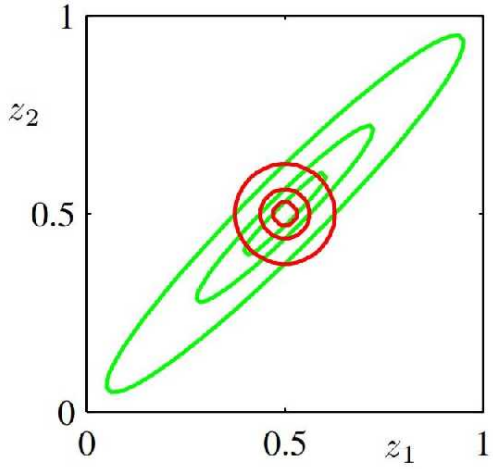


Figure 2.1: Page 468 from Bishop (2006): The green contours correspondens to the 1, 2, and 3 standard deviations for a correlated two-dimensional Gaussian distribution $p(z_1, z_2)$, and the red contours represent the corresponding levels for an approximating distribution $q_1(z_1)q_2(z_2)$ where q_1 and q_2 are obtained by minimization of the Kullback-Leibler divergence $KL(q\|p)$.

Below we summarize conditions on various rate parameters appearing in the our assumptions:

$$0 \leq \eta_2 < \eta_1 \leq 1, \tag{2.22}$$

$$0 \leq 1 - \eta_3 < a < \eta_1 \leq 1, \tag{2.23}$$

where (2.22) also appears in other work on variable selection, such as Shao and Deng (2012), and (2.23) indicates that the magnitude of the posterior variance (of order of $1/n^a$) should be between the minimal signal (of order $1/n^{1-\eta_3}$) and the maximal noise level (of order $1/n^{\eta_1}$). In the classical asymptotic setting, we have $\eta_1 = 1, \eta_2 = 0$ and $1 - \eta_3 = 0$.

With the modified σ_j^2 and some proper choice of v_1, we can show that our algorithm achieves the desired asymptotic property. We first present a lemma that shows that when the sample size is large enough, after one step of Algorithm 3, there is a gap between $\max_{j \notin S_n^*} \text{Logit}(\phi_j)$ and $\min_{j \in S_n^*} \text{Logit}(\phi_j)$ which is large enough for us to separate the relevant variables from the irrelevant ones. Using this lemma, we can then prove the frequentist consistency and the Bayesian consistency. We present our asymptotic results below and include the proofs in Appendix B.

Lemma 2.5.1. *Assume conditions (C1-C5). Suppose v_1 is chosen such that*

$$\left(n^{-a} \bigvee n^{-(2\eta_1 - a - \eta_2)/2}\right) \prec v_1 \prec e^{n^{(a+\eta_3-1)}}$$

and $p = p_n$ satisfies $\log p_n = o(n^{\eta_1 - a})$, then for any constant $C > 0$, after one step of Algorithm 3, we have

$$P\left(\max_{j \notin S_n^*} \text{Logit}(\phi_j) > -\frac{C}{2}\right) \longrightarrow 0, \tag{2.24}$$

$$P\left(\min_{j \in S_n^*} \text{Logit}(\phi_j) < \frac{C}{2}\right) \longrightarrow 0. \tag{2.25}$$

Theorem 2.5.2. *(Variable Selection Consistency) Assume all conditions in Lemma 2.5.1, then we have*

$$P(\hat{S}_n = S_n^*) \longrightarrow 1.$$

Theorem 2.5.3. *(Bayesian Consistency) Assume conditions (C1-C5). Suppose v_1 is chosen*

such that

$$\frac{p^2}{a_n} \prec v_1 \prec e^{n^{(a+\eta_3-1)}}$$

and $p = p_n$ *satisfies* $\log(p_n) = o(n^{\frac{a+\eta_3-1}{2}} \bigwedge n^{\eta_1-a})$*, then we have*

$$q(\boldsymbol{\gamma}^*) = \Big[\prod_{j:\gamma_j^*=1} \phi_j\Big]\Big[\prod_{j:\gamma_j^*=0} (1-\phi_j)\Big] \xrightarrow{P} 1.$$

In Lemma 2.5.1, p is in exponential order of n, which even does not need v_1 to go to infinity. This seems a contradiction with the result from the orthogonal case. But the difference is due to the different updating equation of σ_j^2's. In the orthogonal design, $a = \eta_1 = 1$ from Eq (2.11), so we need v_1 to go to infinity.

2.6 Simulation Studies

In this section, we conduct several simulation studies to compare the two VB algorithms: Algorithm 3 with Algorithm 2, and some other commonly used methods, like LASSO and SCAD.

In both VB algorithms, the choice of v_1 is chosen by cross-validation, and a sparse estimate of $\boldsymbol{\beta}$ is given by

$$\hat{\beta}_j = \mu_j \text{ if } \phi_j \geq 0.5; \quad 0 \text{ if } \phi_j < 0.5. \tag{2.26}$$

The value of a_n in the new variance update formula (2.21) is set to be λ_{n1}, the minimal non-zero eigenvalue of the sample covariance matrix. In our asymptotic analysis we set a_n to be of a smaller order of λ_{n1}; the main purpose of this choice is to ensure that the dimension p_n can grow exponentially fast, i.e., $\log p_n = o(n^{\eta_1-a})$. In practice we have found that setting a_n to be λ_{n1} work well, along with the adaptive choice if v_1 via cross-validation.

2.6.1 Example 1: Benchmark Data

This is a popular benchmark data set, initially designed by Tibshirani (1994) and latter used by Fan and Li (2001) to compare different variable selection methods. The true coefficient is $\boldsymbol{\beta}^* = (3, 1.5, 0, 0, 2, 0, 0, 0)^T \in \mathbb{R}^8$, and the covariance between the i-th and the j-th variable is $0.5^{|i-j|}$. Denote the sample size by n, and the standard deviation of the error term by σ, we consider three scenarios: (1) $n = 40$ and $\sigma = 3$; (2) $n = 40$ and $\sigma = 1$; and (3) $n = 60$ and $\sigma = 1$. We repeat the simulation for 100 times and compare the results of Algorithm 2 and Algorithm 3 with the results of LASSO, SCAD, and the Oracle model from Fan and Li (2001).

To evaluation the estimation accuracy, we compute the model error (ME) by

$$\text{ME} = (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*)^T \Sigma (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*)/\sigma^2 \tag{2.27}$$

where Σ is the covariance matrix of the eight covariates. We set the ME from the ordinary least square (OLS) model as the benchmark, and compute the relative model errors: dividing ME of the other models by that of the OLS. To obtain a robust criterion, we take the median of the relative model errors, namely the median of the relative model error (MRME). The results are shown in Table 2.1. When n gets larger or σ gets smaller, the two VB algorithms and SCAD become much better in terms of MRME, while Lasso does not gain obvious improvement. Overall, the two VB algorithms have lower MRME than SCAD. When the sample size gets larger and the noise level gets smaller, the MRMEs of the two VB algorithms are approaching to that of the Oracle. Also, Algorithm 3 is consistently better than Algorithm 2, especially when the sample size is small and the noise level is high.

To evaluation the selection accuracy, we count the number of zero coefficients among the signal and the noise variables. The results are also reported in Table 2.1. The "Correct"/"Incorrect" column records the average number of zero-coefficients returned by the method among noise/signal variables. A good method should have a number close to 5

for the "Correct" column and 0 for the "Incorrect" column. First, we notice that there is a trade-off between "Correct" and "Incorrect," or namely the trade-off between sensitivity and specificity of identifying noise variables. When the sample size is small ($n = 40$) and the noise level is high ($\sigma = 3$), Lasso identifies almost all the true predictors at the cost of including around 1.5 fake ones, while the variational Bayesian methods, on the other hand, correctly identify most noise variables at the cost of missing some signal variables. Hence, under such a low sample size and high variance setup, it is hard to tell which method is significantly better than the others. But when the sample size gets larger and/or the noise gets smaller, the two VB algorithms outperform all the other methods except the Oracle.

		Avg. No. of 0 Coefficients	
Method	MRME(%)	Correct	Incorrect
$n = 40, \sigma = 3$			
SCAD	72.90	4.20	0.21
Lasso	63.19	3.53	0.07
Alg 2	64.23	4.36	0.32
Alg 3	60.27	4.40	0.30
Oracle	33.31	5	0
$n = 40, \sigma = 1$			
SCAD	54.81	4.29	0
Lasso	63.19	3.51	0
Alg 2	39.34	4.85	0
Alg 3	37.37	4.92	0.12
Oracle	33.31	5	0
$n = 60, \sigma = 1$			
SCAD	47.54	4.37	0
Lasso	65.22	3.56	0
Alg 2	35.69	4.92	0
Alg 3	34.74	4.91	0
Oracle	29.82	5	0

Table 2.1: MRME and Average Number of 0 Coefficients. "Correct" represents how many noise variables are correctly identified, "Incorrect" means the number of true predictors being erroneously set to zero. The results for LASSO, SCAD, and Oracle are from Fan and Li (2001).

2.6.2 Example 2: Highly Correlated Noisy Data

This example is from Wang et al. (2011), in which the design matrix contains some highly correlated predictors with different signs. We use this example to demonstrate the advantage of the proposed batch-wise update used by Algorithm 3 over the component-wise update used by Algorithm 2: errors tend to accumulate with the component-wise update, especially when predictors are highly correlated.

The regression model has $p = 40$ highly correlated covariates and the true coefficient vector is $\boldsymbol{\beta}^0 = (3, 3, -2, 3, 3, -2, 0, \ldots, 0)^T$ with the last 34 elements being zero. The covariates are generated from a multivariate normal distribution: the variance of each variable is 1; the pairwise correlation of the first three variables is 0.9, that of the next three variables is 0.9, and all the other pairwise correlations are 0. The error terms are i.i.d. $N(0, 6^2)$. The sample size n is either 50 or 100. For each n, we repeat the experiment 100 times and compute ME that is defined in (2.27).

In Table 2.2, we report the average and the standard error of ME over 100 simulations. From the table, we can see that Algorithm 3 is better than Algorithm 2 for both $n = 50$ and $n = 100$. and it outperforms Lasso when the sample size grows.

	OLS	Lasso	Algorithm 2	Algorithm 3
$n = 50$	4913	233	322	305
	(323)	(11)	(17)	(18)
$n = 100$	706	144	139	109
	(25)	(6)	(7)	(7)

Table 2.2: $1000\times$ Average ME of Different Methods. The numbers in the parentheses are the corresponding $1000\times$ standard errors. The results for LASSO are from Wang et al. (2011).

In Table 2.3, we list the minimum, the median, and the maximum of the selection frequencies in pairs of parentheses of the 6 important variables (IV) and the 34 unimportant variables (UV). Although Algorithm 2 has the highest value for the maximum selection frequency of IV's, its median selection frequency of IV's is always the lowest. When the sample

size increases, its median selection frequency of IV's even drops. Overall, Algorithm 3 is better than Algorithm 2.

	Lasso	Algorithm 2	Algorithm 3
$n = 50$			
IV	(11, 70, 77)	(24, 41 ,82)	(24,61,71)
UV	(12, 17, 25)	(5, 11,17)	(7,11,19)
$n = 100$			
IV	(8, 84, 88)	(12, 33, 99)	(18, 68, 73)
UV	(12, 22, 31)	(4, 7, 10)	(0, 0, 2)

Table 2.3: Variable Selection Frequencies (%). IV: important variables; UV: unimportant variables. The three numbers in parentheses are the min, median, and max of selection frequencies among all important or unimportant variables, respectively. The results for LASSO are from Wang et al. (2011).

In Table 2.4 and 2.5, we compare the estimated coefficients, as well as their signs, from Lasso, Algorithm 2, and Algorithm 3. For $n = 50$, the three methods can hardly detect any negative signs of β_3 or β_6: the high correlations between β_2 and β_3, and between β_5 and β_6 makes it difficult to identify the opposite signs of the neighboring highly-correlated predictors. When $n = 100$, Algorithm 3 performs the best. Firstly, it can identify the negative signs of β_3 and β_6 in some simulations, while Lasso and Algorithm 2 can barely identify any negative signs of β_3 and β_6. Secondly, the magnitude of the estimates from Algorithm 3 is closer to the true coefficients, and the average coefficient estimates of β_3 and β_6 are negative, which is correct. Estimates from Lasso seem to deviate the most from the truth, which can also be confirmed by Table 2.2.

2.6.3 Example 3: Large p, Small n

We first consider a large-p-small-n example from Ročková and George (2014), in which only the first three of $p = 1000$ predictors are the true ones with non-zero coefficients to be 3, 2, and 1 respectively, and the sample size $n = 100$. The covariance between the i-th and j-th variables is $0.6^{|i-j|}$, and the error terms are generated from $i.i.d.$ $\mathrm{N}(0, 3)$.

	β_1	β_2	β_3	β_4	β_5	β_6
True Coef	3	3	-2	3	3	-2
Lasso						
Ave. of est.	1.41	1.27	0.12	1.36	1.36	0.09
	(0.12)	(0.14)	(0.04)	(0.12)	(0.13)	(0.06)
No. of pos. sgn.	73	69	13	74	68	13
No. of neg. sgn.	0	0	1	0	0	2
Alg 2						
Ave. of est.	2.08	0.75	0.23	2.19	0.61	0.23
	(0.16)	(0.11)	(0.05)	(0.16)	(0.09)	(0.04)
No. of pos. sgn.	80	43	24	82	40	26
No. of neg. sgn.	0	0	0	0	0	0
Alg 3						
Ave. of est.	1.63	1.17	0.30	1.36	1.44	0.28
	(0.16)	(0.13)	(0.09)	(0.14)	(0.14)	(0.07)
No. of pos. sgn.	71	57	31	65	66	24
No. of neg. sgn.	0	0	1	0	0	0

Table 2.4: Coefficient and Coefficient Sign Estimation, $n = 50$. The numbers in parenthesis are the standard errors. The results for LASSO are from Wang et al. (2011).

	β_1	β_2	β_3	β_4	β_5	β_6
True Coef	3	3	-2	3	3	-2
Lasso						
Ave. of est.	1.67	1.50	0.06	1.85	1.38	0.04
	(0.11)	(0.11)	(0.02)	(0.13)	(0.13)	(0.03)
No. of pos. sgn.	91	85	8	86	78	9
No. of neg. sgn.	0	0	0	0	0	2
Alg 2						
Ave. of est.	2.97	0.56	0.08	2.93	0.56	0.10
	(0.12)	(0.08)	(0.02)	(0.12)	(0.09)	(0.03)
No. of pos. sgn.	99	34	12	96	32	12
No. of neg. sgn.	0	0	0	0	0	0
Alg 3						
Ave. of est.	2.09	1.92	-0.11	2.33	1.85	-0.20
	(0.16)	(0.16)	(0.05)	(0.18)	(0.18)	(0.09)
No. of pos. sgn.	73	66	5	72	65	7
No. of neg. sgn.	0	1	13	0	3	16

Table 2.5: Coefficient and Coefficient Sign Estimation, $n = 100$. The numbers in parenthesis are the standard errors. The results for LASSO are from Wang et al. (2011).

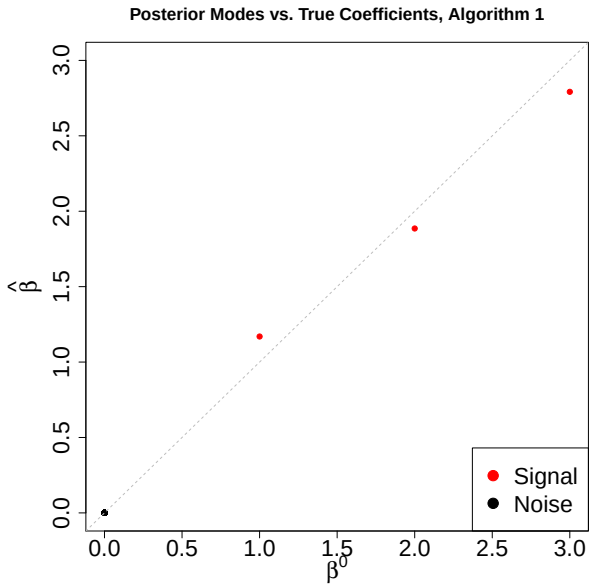
We fit the model using Algorithm 2 and Algorithm 3, with fixed $v_1 = 1$. As shown in Figure 2.2a and 2.2b, both algorithms make no mistake in variable selection and the estimates are very close to the true coefficients.

Next we make this example a little more challenging by adding more non-zero coefficients of different magnitudes. The true coefficient vector is now set to be $\boldsymbol{\beta}^0 = (\boldsymbol{\beta}_1^{0T}, 0, \ldots, 0)^T$, where the last 980 elements are all zero and the 20 non-zero coefficients in $\boldsymbol{\beta}_1^0$ contains randomly distributed ten 1's, seven 2's, and three 3's. With the 0.6 pairwise correlation, some weak signals may be overshadowed by nearby strong signals, which makes variable selection a challenging task. We repeat the experiment 100 times, and compare results from Algorithm 2, Algorithm 3 and Lasso. For the tuning parameter in Lasso, we use the default `cv.glmnet` function from the `R` package `glmnet` (Friedman et al., 2010), and report the results for both `lambda.min` that is the λ with the smallest CV error, and `lambda.1se` that is the largest λ whose CV error is within one standard error of the smallest CV error.

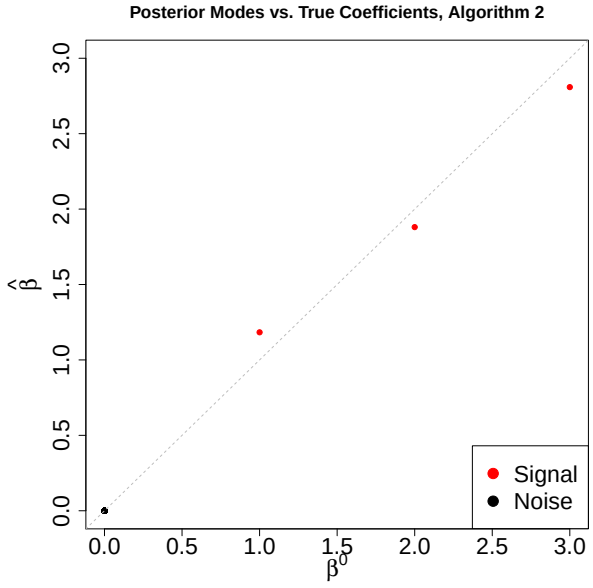
The results are summarized in Table 2.6. Regarding the accuracy for identifying important variables, Algorithm 2 is the worst: on average it only identifies 8.97 true predictors, fewer than a half. This is supported with our large sample analysis: with high correlation, the errors tend to accumulate, which makes some true predictors hard to be identified. Regarding the accuracy for identifying unimportant variables, `Lasso.min` is the worst: on average it selects 34.14 noise variables, more than doubled that of `Lasso.1se`, the second worst. Overall, Algorithm 3 performs the best.

	Model	Algorithm 2	Algorithm 3	Lasso.min	Lasso.1se
# of IV	Mean	8.97	15.61	19.92	19.77
	S.E.	(0.13)	(0.19)	(0.03)	(0.05)
# of UV	Mean	0.12	0.5	34.14	11.23
	S.E.	(0.04)	(0.08)	(1.50)	(0.81)

Table 2.6: Variable Selection Frequencies (%). IV: important variables; UV: unimportant variables. The numbers in parenthesis are the standard errors.



(a) True Coefficient vs. Sparse Estimates: Algorithm 2



(b) True Coefficient vs. Sparse Estimates: Algorithm 3

Figure 2.2: $\hat{\boldsymbol{\beta}}$ *vs.* $\boldsymbol{\beta}^0$ for the large-p-small-n example from Ročková and George (2014).

2.7 Real Data: Boston Housing Data

2.7.1 Introduction

The original data is from the `R` library `mlbench`, which has 506 observations on 19 variables. We apply some suggested transformations on the data according to Johnson et al. (1992), then remove three variables `medv`, `town`, and `tract`, and use `cmedv` as the response variable. We call this data set "Boston Housing 1" (BH1).

Then we create a larger data set called "Boston Housing 2" (BH2). First, we add all the 119 quadratic terms (including all pairwise interaction terms) of the predictors, and 500 noise features. We generate the noise features in 50 batches. For each batch, we randomly select 10 variables from the set of 134 "true" variables, which gives us a 506×10 data matrix; for each entry of the matrix, we add a small Gaussian error, and then randomly shuffle the 506 rows. So each noise feature looks like some true variable marginally, and in addition correlations among the true variables are preserved in the noise features. The final data set has 634 features, which is larger than the sample size.

We have shown that Algorithm 3 is better than Algorithm 2 in the simulation study. Now let's compare Algorithm 3 with Lasso, Ridge, and the full model (i.e., the model using all predictors). For Algorithm 3, we consider the following two prediction methods:

(i) Use the sparse estimate $\hat{\boldsymbol{\beta}}$ that is defined at (2.26) and the prediction is given by $\hat{\mathbf{y}} = \mathbf{X}^{new}\hat{\boldsymbol{\beta}}$. We abbreviate this approach by "S" that stands for "Sparsity Prediction."

(ii) Refit a linear regression model using the predictors from $\hat{S} = \{j : \phi_j > 0.5\}$ and denote the estimated coefficients from the OLS by $\hat{\boldsymbol{\beta}}^{ols}$. Then the prediction is given by $\hat{\mathbf{y}} = \mathbf{X}^{new}_{[,\hat{S}]}\hat{\boldsymbol{\beta}}^{ols}$, where $\mathbf{X}^{new}_{[,\hat{S}]}$ denotes a subset of the data matrix $\mathbf{X}^{new}$ with only columns from $\hat{S}$. We abbreviate this approach by "TS" that stands for "Two-Stage Sparse Prediction".

We run the following simulation for 50 times. In each iteration, we randomly subset

75% of the dataset as the training data (380 observations) and predict on the remaining validation data (126 observations). Then we record the selected model size and compute the mean squared prediction error (MSPE). As all the methods but the OLS have a tuning parameter, we select the tuning parameter using cross-validation. For Algorithm 3, we use a 5-fold cross validation for the two prediction approachs; for Lasso and ridge regression, we use the default `cv.glmnet` function from the `R` package `glmnet` (Friedman et al., 2010), and report results for both `lambda.min` and `lambda.1se`. The results are summarized in Table 2.7.

2.7.2 Boston Housing 1

Most methods perform similarly based on the prediction error. Surprisingly the full model performs the best. This is because the potential gain of variable selection for such a traditional small-p-large-n example is negligible. On the other hand, the potential bias introduced by variable selection or shrinkage procedures, when relevant variables are mistakenly excluded or over-shrunk, can be large. This also explains why Lasso.1se/ridge.1se performs worse than Lasso.min/ridge.min, since the former tends to pick a smaller model than the latter, i.e., has a higher chance of missing relevant variables.

2.7.3 Boston Housing 2

The ridge regression is the worst of all methods: both ridge.1se and ridge.min have relatively large effective dimensions, but high prediction errors. For prediction accuracy, Lasso.min, Alg 3(S) and Alg 3(TS) are better than the other methods; regarding sparsity, the models selected by Lasso.1se, Alg 3(S) and Alg 3(TS) are much smaller than the others. The best model is Alg 3(TS): it has the best prediction accuracy with the most sparse model.

	Model	Full	Ridge.min	Ridge.1se	Lasso.min	Lasso.1se	Alg 3(S)	Alg 3(TS)
BH 1	Mean	0.043	0.044	0.049	0.044	0.048	0.045	0.044
	SE	(0.008)	(0.009)	(0.010)	(0.008)	(0.008)	(0.009)	(0.008)
	Size	15	12.34	9.21	13.84	7.2	10.8	11.36
BH 2	Mean		0.065	0.071	0.043	0.047	0.046	0.042
	SE		(0.011)	(0.012)	(0.008)	(0.008)	(0.012)	(0.007)
	Size		52.21	40.21	38.7	8.68	9.18	7.74

Table 2.7: Boston Housing Data: Average and S.E. of MSPE, and Model Size

2.8 Conclusions

The Bayesian approach to variable selection is appealing since it outputs not a single model but a probability distribution over all possible models. Hence model uncertainty can be naturally incorporated into estimation, prediction, and many other statistical inferences. However, most Bayesian variable selection methods are implemented through MCMC, which is time consuming when the model dimension is large. In this chapter, we seek an approximation of the posterior distribution via a variational optimization. Our resulting algorithm converges very fast and can scale up with large data sets. We also showed that the approximation returned by our algorithm has the desired asymptotic behavior, which achieves both the frequentist consistency and Bayesian consistency asymptotically.

Chapter 3

A Variational Algorithm for Logistic Regression

3.1 Introduction

Statistical classification has many applications, such as handwriting recognition, disease classification, and document classification. Well-known classification methods include logistic regression, Fisher discriminant analysis, k-nearest-neighbor classifier, support vector machines (SVM), classification trees, random forest, and so on.

We focus on the logistic regression (Cox, 1958), which is a widely used machine learning algorithm that can be applied to a variety of classification problems. There are several advantages of logistic regression. First of all, it is easy to be statistically interpreted. Other commonly used models, for example SVM, tree and random forest models, cannot provide such easy statistical interpretation. Secondly, the logistic regression is naturally a probabilistic framework. The output of the model is the probability of a class, rather than just a class label. This is useful when the probabilities of outcomes are needed.

However, many recent applications involve large-scale data sets that have high dimension of predictors. One key to deal with the problems arising from large number of predictors is to obtain a sparse estimate for logistic regression model. Similar to Chapter 2, we follow the Bayesian variable selection approach, and seek to variational EM algorithm for fast computation.

However, there is a key difference between linear regression and logistic regression: The likelihood function of the logistic regression is not a density of normal in the coefficients. Therefore the spike-and-slab distribution and the likelihood of logistic regression model are

no longer "conjugate" when we use the variational approach. One solution is to use local approximation (Jaakkola and Jordan, 2000) which lower bounds the likelihood by Taylor expansion so that the approximation is a normal density in the coefficients. Carbonetto and Stephens (2012) used this approach and proposed a variational algorithm for the Bayesian logistic regression model with spike-and-slab prior.

We follow an alternative approach: the Pólya-Gamma data-augmentation (Polson et al., 2013). According to Polson et al. (2013), the Pólya-Gamma data-augmentation is exact, and is easier to be implemented than the other data-augmentation algorithms (Holmes et al., 2006; Frühwirth-Schnatter and Frühwirth, 2010) which involve multiple layers. The Pólya-Gamma data-augmentation has already been used for logistic regression in the EM framework. (Scott and Sun, 2013) proposed an EM algorithm for the ordinary logistic regression, and also with ridge and lasso regularization. Using the Pólya-Gamma data-augmentation, we propose a VB algorithm for the Bayesian logistic model with the spike-and-slab prior, which is fast and can scale to relatively large scale data sets.

3.2 Model Setup

Consider a logistic regression. The number of success Y is modeled by Binomial(m, s), where m is the total number of binary random trials, s is the probability of success of each random trial. We model s by a set of p predictors $X_1, \ldots, X_p$ via the logit link:

$$\text{Logit}(s) = \alpha + X_1\beta_1 + \cdots X_p\beta_p,$$

where $\text{Logit}(s) = \log(s/(1-s))$. In the usual setup of logistic regression, $m = 1$.

Suppose that there are n observations. Denote $\mathbf{y} = (y_1, \ldots, y_n)^T$, $\mathbf{m} = (m_1, \ldots, m_n)^T$ and $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^T$. Define the coefficient vector as $\boldsymbol{\beta} =$

$(\beta_1, \ldots, \beta_p)^T$. Then likelihood is

$$p(\mathbf{y}|\mathbf{X}, \mathbf{m}, \alpha, \boldsymbol{\beta}) \propto \prod_{i=1}^{n} \frac{\exp(\alpha + \mathbf{x}_i^T \boldsymbol{\beta})^{y_i}}{(1 + \exp(\alpha + \mathbf{x}_i^T \boldsymbol{\beta}))^{m_i}} \tag{3.1}$$

We put a spike-and-slab prior on each β_j according to Eq (1.1), and assume that $\gamma_j \overset{i.i.d.}{\sim}$ Bernoulli(θ). To complete the model setup, we put prior on α and θ:

$$\alpha \sim \text{Unif}(-\infty, +\infty), \quad \theta \sim \text{Beta}(a_0, b_0).$$

We apply the PG data-augmentation so that the complete-data likelihood is a density of normal in $\boldsymbol{\beta}$:

$$p(\mathbf{y}, \mathbf{w}|\mathbf{X}, \mathbf{m}, \alpha, \boldsymbol{\beta}) = \prod_{i=1}^{n} p(y_i|w_i, \mathbf{x}_i, m_i, \alpha, \boldsymbol{\beta}) g(w_i|m_i, 0),$$

where $p(y_i|w_i, \mathbf{x}_i, m_i, \alpha, \boldsymbol{\beta})$ is proportional to

$$\exp\left\{ \left(y_i - \frac{m_i}{2} \right) (\alpha + \mathbf{x}_i^T \boldsymbol{\beta}) - \frac{w_i}{2} (\alpha + \mathbf{x}_i^T \boldsymbol{\beta})^2 \right\}, \tag{3.2}$$

and $g(w|b, c)$ is the density function of Pólya-Gamma, $PG(b, c)$. Given that $W \sim g(w|b, c)$,

$$\mathbb{E}(W) = \frac{b}{2c} \tanh\left(\frac{c}{2} \right). \tag{3.3}$$

The details of PG data-augmentation and the property of PG distribution is in Appendix C.

3.3 A Variational Algorithm

3.3.1 Objective Function

In the Bayesian logistic regression model, we approximate the posterior of $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, and $\mathbf{w}$, and find the point estimates for α and θ. The objective function is

$$\Omega(q, \alpha, \theta) = \mathbb{E}^q_{\boldsymbol{\gamma},\boldsymbol{\beta},w} \log \frac{\pi(\boldsymbol{\beta}, \boldsymbol{\gamma}, \alpha, \theta, \mathbf{w}|\mathbf{y})}{q(\boldsymbol{\beta}, \boldsymbol{\gamma}, \mathbf{w})}. \tag{3.4}$$

Let's consider the following factorization of q:

$$q(\boldsymbol{\beta}, \boldsymbol{\gamma}, \mathbf{w}) = q_1(\boldsymbol{\beta}, \boldsymbol{\gamma}) q_2(\mathbf{w}) = \prod_{j=1}^{p} q_{1,j}(\beta_j, \gamma_j) \prod_{i=1}^{n} q_{2,i}(w_i),$$

where $q_{1,j}(\beta_j, \gamma_j) = [\phi_j f_j(\beta_j)]^{\gamma_j} [(1-\phi_j)\delta_0(\beta_j)]^{1-\gamma_j}$, and $f_j(\beta_j)$ is a density of β_j. A similar setup can be found in Bayesian linear regression (Carbonetto and Stephens, 2012; Huang et al., 2016).

3.3.2 Updating Equations

We update the variational distributions and the MAP estimates via the coordinate ascent.

We denote $\bar{\Theta}$ to be the expectation of random variable Θ w.r.t. its variational distribution: $\bar{\Theta} \triangleq \mathbb{E}^{q(\Theta)}(\Theta)$. The updating equations are shown as follows. The pseudo-code is in Algorithm 4. The detailed derivation is in Appendix A.

Update $q_{1j}(\beta_j, \gamma_j)$. We maximize the objective function w.r.t. $q_{1,j}$, while fixing all the other approximating distributions and point estimates. Then we can show that the best

$f_j(\beta_j)$ is proportional to $\exp\{a\beta_j - \frac{b}{2}\beta_j^2\}$ with

$$\begin{aligned} a =& \left(\mathbf{y} - \frac{1}{2}\mathbf{m} - \overline{\mathbf{W}}(\hat{\alpha}\mathbb{1}_n + \mathbf{X}_{[-j]}\bar{\boldsymbol{\beta}}_{[-j]})\right)^T \mathbf{X}_j, \\ b =& \mathbf{X}_j^T\overline{\mathbf{W}}\mathbf{X}_j + \frac{1}{v_1}, \end{aligned}$$

where $\overline{\mathbf{W}} = \text{diag}\{\bar{w}_1, \ldots, \bar{w}_n\}$. Since, the variational distribution of w_i is Pólya-Gamma, which we will show in the next step, $\bar{w}_i$ can be calculated via (12). This implies that $f_j(\beta_j) = \text{N}(\beta_j|\mu_j, \sigma_j^2)$, with

$$\begin{aligned} \mu_j =& \frac{a}{b} = \frac{\left(\mathbf{y} - \frac{1}{2}\mathbf{m} - \overline{\mathbf{W}}(\hat{\alpha}\mathbb{1}_n + \mathbf{X}_{[-j]}\bar{\boldsymbol{\beta}}_{[-j]})\right)^T \mathbf{X}_j}{\mathbf{X}_j^T\overline{\mathbf{W}}\mathbf{X}_j + \frac{1}{v_1}} \\ \sigma_j^2 =& \frac{1}{b} = \frac{1}{\mathbf{X}_j^T\overline{\mathbf{W}}\mathbf{X}_j + \frac{1}{v_1}}. \end{aligned} \tag{3.5}$$

We plug μ_j and σ_j^2 back to the objective function, and the maximizer of ϕ_j satisfies

$$\text{Logit}(\phi_j) = \text{Logit}(\hat{\theta}) + \frac{1}{2}\log\frac{\sigma_j^2}{v_1} + \frac{\mu_j^2}{2\sigma_j^2}. \tag{3.6}$$

By symmetry, we know that for all $j \in \{1, \ldots, p\}$, $f_j(\beta_j) = \text{N}(\beta_j|\mu_j, \sigma_j^2)$. Hence, $\bar{\boldsymbol{\beta}}_{[-j]}$ in (3.5) can be easily calculated.

Update $q_2(\mathbf{w})$. Fixing q_{1j}'s and the point estimates, we can show that

$$\log q_2(\mathbf{w}) \propto \mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}}\left\{\log[\pi(\hat{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \hat{\theta}, \mathbf{w}|\mathbf{y})]\right\} \propto \sum_{i=1}^{n} \log g\left(w_i|m_i, z_i\right),$$

where

$$z_i^2 = \mathbb{E}(\hat{\alpha} + \mathbf{x}_i^T\boldsymbol{\beta})^2 = \hat{\alpha}^2 + 2\hat{\alpha}\mathbf{x}_i^T\bar{\boldsymbol{\beta}} + \mathbf{x}_i^T(\overline{\boldsymbol{\beta}\boldsymbol{\beta}^T})\mathbf{x}_i, \tag{3.7}$$

and

$$\begin{aligned}\overline{\boldsymbol{\beta}\boldsymbol{\beta}^T} =& \mathbb{E}(\boldsymbol{\beta})\mathbb{E}(\boldsymbol{\beta})^T + \mathrm{Cov}(\boldsymbol{\beta}) \\ =& \boldsymbol{\Phi}\boldsymbol{\mu}\boldsymbol{\mu}^T\boldsymbol{\Phi} + \mathrm{diag}\{\mu_j^2\phi_j(1-\phi_j) + \sigma_j^2\phi_j\}_{j=1}^p.\end{aligned}$$

Hence, the variational distribution of w_i is $PG\,(m_i, z_i)$.

Update $\hat{\alpha}$. Next we update the point estimates for α:

$$\hat{\alpha} = \arg\max_{\alpha} \mathbb{E}^{q_1,q_2}_{\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{w}} \left[\log \frac{\pi(\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}, \hat{\theta}, \mathbf{w}|\mathbf{y})}{q_1(\boldsymbol{\beta}, \boldsymbol{\gamma}) q_2(\mathbf{w})}\right] = \frac{\left(\mathbf{y} - \frac{1}{2}\mathbf{m} - \overline{\mathbf{W}}\mathbf{X}\boldsymbol{\beta}\right)^T \mathbb{1}_n}{\mathbb{1}_n^T \overline{\mathbf{W}} \mathbb{1}^T}$$

Update $\hat{\theta}$. Finally, we consider the point estimate of θ given all the approximating distributions and $\hat{\alpha}$:

$$\hat{\theta} = \arg\max_{\theta} \mathbb{E}^{q_1,q_2}_{\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{w}} \left[\log \frac{\pi(\hat{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \theta, \mathbf{w}|\mathbf{y})}{q_1(\boldsymbol{\beta}, \boldsymbol{\gamma}) q_2(\mathbf{w})}\right] = \frac{\sum_{j=1}^p \phi_j + a_0 - 1}{p + a_0 + b_0 - 2}.$$

3.3.3 Convergence Criterion

A typical convergence criterion is by monitoring the value of the objective function. However, calculating the objective function is time-consuming during each iteration. Also, as we focus on the variable selection, the posterior inclusion probabilities (ϕ_j) are more important than the posterior of the latent variables. Hence, we use the maximum change of entropy criterion instead. That is, we compute the entropy of γ_j's, and monitor their change from iteration to iteration. If the maximal difference is lower than a pre-specified threshold, we stop the updating.

Algorithm 4 VB Algorithm for Logistic Regression

input $\mathbf{X}, \mathbf{m}, \mathbf{y}, v_1, a_0, b_0$
initialize $(\mu_1, \ldots, \mu_p)$, $(\sigma_1^2, \ldots, \sigma_p^2)$, $(\phi_1, \ldots, \phi_p)$, $\hat{\alpha}$, and $\hat{\theta}$.
repeat
  for i in $1 : n$ **do**
    $z_i \leftarrow \sqrt{\hat{\alpha}^2 + 2\hat{\alpha}\mathbf{x}_i^T\bar{\boldsymbol{\beta}} + \mathbf{x}_i^T(\overline{\boldsymbol{\beta}\boldsymbol{\beta}^T})\mathbf{x}_i}$
    $\bar{w}_i \leftarrow \frac{m_i}{2z_i}\tanh\left(\frac{z_i}{2}\right)$
  end for
  for j in $1 : p$ **do**
    $\mu_j \leftarrow \frac{\left(\mathbf{y} - \frac{m_i}{2} - \overline{\mathbf{W}}(\hat{\alpha}\mathbb{1}_n + \mathbf{X}_{[-j]}\bar{\boldsymbol{\beta}}_{[-j]})\right)^T\mathbf{X}_j}{\mathbf{X}_j^T\overline{\mathbf{W}}\mathbf{X}_j + \frac{1}{v_1}}$
    $\sigma_j^2 \leftarrow \frac{1}{\mathbf{X}_j^T\overline{\mathbf{W}}\mathbf{X}_j + \frac{1}{v_1}}$
    $\phi_j \leftarrow \text{Logit}^{-1}\left\{\text{Logit}(\hat{\theta}) + \frac{1}{2}\log\frac{\sigma_j^2}{v_1} + \frac{\mu_j^2}{2\sigma_j^2}\right\}$
  end for
  $\hat{\alpha} \leftarrow \frac{\left(\mathbf{y} - \frac{m_i}{2} - \overline{\mathbf{W}}\mathbf{X}\boldsymbol{\beta}\right)^T\mathbb{1}_n}{\mathbb{1}_n^T\overline{\mathbf{W}}\mathbb{1}^T}$
  $\hat{\theta} \leftarrow \frac{\sum_{j=1}^p \phi_j + a_0 - 1}{p + a_0 + b_0 - 2}$
until Converge
return $(\mu_1, \ldots, \mu_p)$, $(\sigma_1^2, \ldots, \sigma_p^2)$, $(\phi_1, \ldots, \phi_p)$, $\hat{\alpha}$, and $\hat{\theta}$

3.3.4 Prediction Approaches

In terms of prediction, we need the estimated coefficients $\hat{\boldsymbol{\beta}}$. There are three approaches of estimation of β_j:

- Sparse: $\hat{\beta}_j = \mu_j$ if $\phi_j > 0.5$; otherwise, $\hat{\beta}_j = 0$.
- Dense: $\hat{\beta}_j = \phi_j\mu_j$, which is the posterior mean of β_j.
- Refit: Refit a logistic regression model for $j \in \hat{S} = \{j : \phi_j > 0.5\}$ if $|\hat{S}| < n$, and set $\hat{\beta}_j = 0$ for $j \notin \hat{S}$.

Using either one of the above approaches, we can predict $P(Y_{\text{new}} = 1)$ given the new predictors $\mathbf{x}_{\text{new}}$ via:

$$\text{Logit}(\hat{P}(Y_{\text{new}} = 1)) = \hat{\alpha} + \mathbf{x}_{\text{new}}^T\hat{\boldsymbol{\beta}}.$$

In some applications, the predicted class label, $\hat{y}_{\text{new}}$, is needed. Then we can set some pre-specified cut-off value, C, and set

$$\hat{y}_{\text{new}} = \begin{cases} 1, & \text{if } \hat{P}(Y_{\text{new}} = 1) > C \\ 0, & \text{if } \hat{P}(Y_{\text{new}} = 1) < C \end{cases} \tag{3.8}$$

In the following numerical studies, we set $C = 0.5$ for simplicity.

3.3.5 Tuning v_1

The v_1 plays the role of a tuning parameter in our model. Although $v_1 = 1$ usually works well, we may achieve higher prediction accuracy by using a K-fold cross-validation (CV) to select the best v_1 that returns the lowest misclassification error, $\frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - y_i|$, where $\hat{y}_i$ is obtained from Eq (3.8). If there is a tie in the CV error, we will use the largest corresponding v_1 to obtain the most sparse model.

In practice, we suggest to set $\log_{10}(v_1) \in [-2, 2]$, so that the true predictors and the noise ones can be separated by their posterior inclusion probabilities for some v_1 in that range. A path plot of $\log_{10}(v_1)$ versus ϕ is shown in Figure 3.1 using the data set from the first simulation study. From the plot, there is a gap between the inclusion probabilities of the true predictors and the noises when v_1 is greater than 10^{-2}. The ϕ of the noise variables will decrease with the increase of v_1, and the gap will become wider. In order not to kill the weak signal (the third variable in this case), we also need to restrict v_1 below an upper bound. If we select the cutoff point to be 0.5, the range of v_1 that can distinguish the signal and noise is around 10^{-1} and 10^{1}. So the choice of $\log_{10}(v_1) \in [-2, 2]$ would be wide enough to cover most cases.

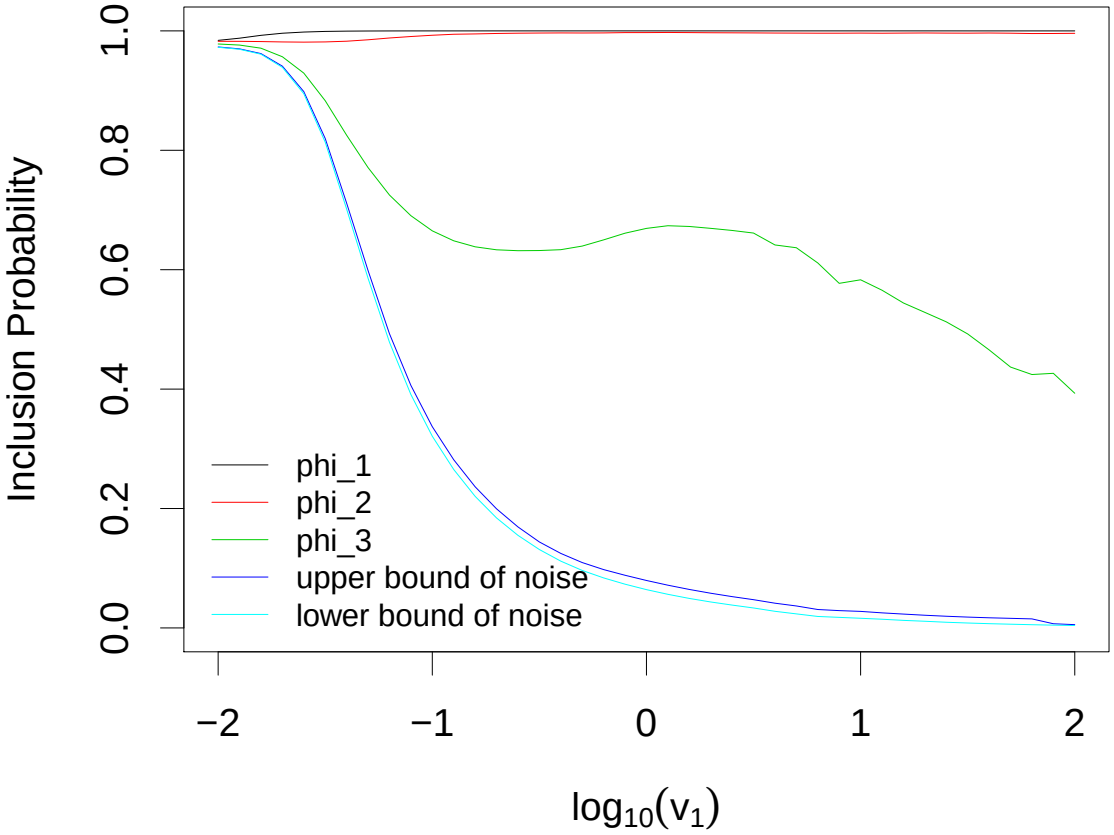


Figure 3.1: Path Plot: v_1 vs. ϕ

3.4 Connection with Local Approximation

Local approximation is another way used in variational method for logistic model. A variational lower bound is used to approximate the likelihood (Jaakkola and Jordan, 2000)

$$p(y_i|\mathbf{x}_i, m_i, \boldsymbol{\beta}) = C_i \cdot \frac{\left(e^{\mathbf{x}_i^T\boldsymbol{\beta}}\right)^{y_i}}{\left(1+e^{\mathbf{x}_i^T\boldsymbol{\beta}}\right)^{m_i}} \geq C_i \cdot \exp\left\{\kappa_i\psi_i - \frac{\xi_i}{2} + \log g(\xi_i) + \frac{1}{2}\lambda(\xi_i)(\psi_i^2 - \xi_i^2)\right\}$$

for all ξ_i, where $C_i = \binom{m_i}{y_i}$ is a constant w.r.t. $\boldsymbol{\beta}$, $\kappa_i = y_i - \frac{m_i}{2}$, $\psi_i = \alpha + \mathbf{x}_i^T\boldsymbol{\beta}$, $g(x) = -\log\left(e^{x/2} + e^{-x/2}\right)$, and $\lambda(\xi_i) = \frac{1}{2\xi_i}\tanh\left(\frac{\xi_i}{2}\right)$. Then we have

$$p(\mathbf{y}|\mathbf{x}, \mathbf{m}, \boldsymbol{\beta}) \geq h(\boldsymbol{\beta}, \xi) \triangleq \prod_{i=1}^{n} C_i \cdot \exp\left\{\kappa_i\psi_i - \frac{\xi_i}{2} + \log g(\xi_i) + \frac{1}{2}\lambda(\xi_i)(\psi_i^2 - \xi_i^2)\right\}$$

Adopt the same prior and class of variational distribution for $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ in the previous section, and replace the likelihood by $h(\boldsymbol{\beta}, \xi)$, then the new objective function becomes,

$$\widetilde{\Omega}(q, \alpha, \theta) = \mathbb{E}^q_{\boldsymbol{\beta},\boldsymbol{\gamma}}\left[\log \frac{h(\boldsymbol{\beta}, \xi)\prod_{j=1}^{p}\pi(\beta_j|\gamma_j)\pi(\gamma_j|\theta)\pi(\theta)}{\prod_{j=1}^{p} q_j(\boldsymbol{\beta}, \boldsymbol{\gamma})}\right].$$

To maximize the objective function, we iteratively solve for q_j's, $\hat{\alpha}$, and $\hat{\theta}$ via coordinate ascent. When updating $q_j(\beta_j, \gamma_j)$, it can be shown that $f_j(\beta_j) = \mathrm{N}(\beta_j|\mu_j, \sigma_j^2)$, where

$$\begin{aligned}\mu_j =& \frac{\left(\mathbf{y} - \frac{1}{2}\mathbf{m} - \hat{\Lambda}(\hat{\alpha}\mathbb{1}_n + \mathbf{X}_{[-j]}\bar{\boldsymbol{\beta}}_{[-j]})\right)^T\mathbf{X}_j}{\mathbf{X}_j^T\hat{\Lambda}\mathbf{X}_j + \frac{1}{v_1}}\\ \sigma_j^2 =& \frac{1}{\mathbf{X}_j^T\hat{\Lambda}\mathbf{X}_j + \frac{1}{v_1}}\end{aligned}$$

where $\hat{\Lambda} = \mathrm{diag}\{\lambda(\hat{\xi_1}), \ldots, \lambda(\hat{\xi_n})\}$. The updating equations for $q(\beta_j, \gamma_j)$'s are equivalent to those of Algorithm 4 if we let $\lambda(\hat{\xi_i}) = \bar{w}_i$. The updating equation of $\hat{\alpha}$ has the same change.

Given $q(\beta_j, \gamma_j)$'s, $\hat{\alpha}$ and $\hat{\theta}$, we maximize the objective function w.r.t. ξ_i, and have

$$\hat{\xi_i}^2 = \mathbb{E}[(\hat{\alpha} + \mathbf{x}_i^T \boldsymbol{\beta})^2] = \hat{\alpha}^2 + 2\hat{\alpha}\mathbf{x}_i^T \bar{\boldsymbol{\beta}} + \mathbf{x}_i^T (\overline{\boldsymbol{\beta}\boldsymbol{\beta}^T})\mathbf{x}_i,$$

where $\bar{\boldsymbol{\beta}} = \boldsymbol{\Phi}\boldsymbol{\mu}$, and $\mathrm{Cov}(\boldsymbol{\beta}) = \mathrm{diag}\{\phi_j(1-\phi_j)\mu_j^2 + \phi_j\sigma_j^2\}_{j=1}^p$. Hence maximizer of ξ_i^2 plays the role as z_i^2 in Algorithm 4.

To conclude, if we use local approximation in VB, the result is equivalent as introducing a latent variable w_i from Pólya-Gamma prior, with the local parameter ξ_i^2 playing the role of the square of the second PG parameter z_i^2.

3.5 Simulation Studies

3.5.1 Example 1: Large p and Correlated Features

Let's focus on a simulation data with $n \approx p$ and there is correlation among features. The sample size is set to be $n = 100$, and feature dimension $p = 99$. The true model is $y_i^* = \mathbb{1}\{\mathbf{x}_i^T \boldsymbol{\beta}^* + \epsilon_i > 0\}$, where ϵ_i's are *i.i.d.* $\mathrm{N}(0, 1)$, and the true coefficient $\boldsymbol{\beta}^* = \left((\boldsymbol{\beta}_0^*)^T, \mathbf{0}_{96}^T\right)^T$ with $\boldsymbol{\beta}_0^* = (3, 2, 1)^T$. The features are generated from a multivariate Normal distribution with mean 0, and the i-th and the j-th features are correlated, with $\mathrm{Cov}(X_i, X_j) = 0.6^{|i-j|}$. We repeat this procedure for 100 times.

Table 3.1 showed the descriptive statistics of the classification error, false positive (FP) rate and false negative (FN) rate. From the table, all the methods have similar error rates. In terms of the model sparsity, LASSO tends to include more variables, and the false positive rate is higher. To obtain a sparse model, our algorithm approach is better.

3.5.2 Example 2: Digits Data

We use this example to compare the prediction accuracy of our algorithm with Lasso. The data set is based on the scanned handwritten ZIP codes on envelopes from the U.S. postal

Table 3.1: Example 1: Error Rate and Variable Selection. FP: average False Positive rate, or how many noise variables are included. FN: average False Negative rate, or how many true predictors are dropped.

		BVS			Lasso	
		Sparse	Dense	Refit	lambda.min	lambda.1se
Error Rate	Median	0.0800	0.0900	0.0800	0.1000	0.0900
	Mean	0.0897	0.0920	0.0920	0.0993	0.0866
	S.E.	0.0036	0.0037	0.0047	0.0032	0.0030
Variable Selection	FP	0.14	–	1.93	14.46	5.03
	FN	0.49	–	0.39	0.07	0.12

mail (LeCun et al., 1989). We obtain the data from the online supplement material of Friedman et al. (2001). Each observation contains 256 features from the 16×16 gray-scale images. There are totally 1200 training samples and 332 test samples.

Let's consider a binary classification problem: "3" *vs.* "8". The experiment is as follows. We combine the original training and test sets, and randomly split them into 75% training and 25% test sets. We use cross-validation to select v_1 and train the model on the training set, and predict on the test set. We repeat this procedure for 100 times and record the error rate. In Table 3.2, we show the average and median classification error rate. We find that the error rate of `sparse` procedure of BVS is the lowest.

Table 3.2: Example 2: Digits. Test Error Rate and Model Size

	BVS			Lasso	
	Sparse	Dense	Refit	lambda.min	lambda.1se
Median	0.0261	0.0261	0.0391	0.0287	0.0313
Mean	0.0253	0.0267	0.0407	0.0282	0.0309
S.E.	0.0008	0.0008	0.0010	0.0008	0.0008

3.6 Real Data: Internet Advertisements Data

Internet users are exposed to tons of advertisements (ads) when surfing the Internet. Some users prefer not to see the pictures of ads. It may be too hard for the users to find the

useful information, when there are many ads on the webpage. Sometimes when the Internet connection speed is low, it takes too much time loading the ads. What's more, some ads contain malicious links which are unsafe for the users. Due to any of the above reasons, there is a need to develop some algorithms that identify the ads, so that they can be removed or not loaded.

We use the dataset from the UCI repository (Lichman, 2013), which is first used by Kushmerick (1999). The task is to predict whether an image on websites is an advertisement ("ad") or not ("nonad"). There are 3279 observations, including 2821 nonads, 458 ads. The features that were encoded directly from the raw HTML include 3 continuous attributes, and 1555 binary attributes. The continuous features are the geometric information of the image: height, width and the ratio of width to height. About 28% of the continuous features are missing, and we create three corresponding binary indicators for them. Most discrete features are indicators of the existence of words or phrases in the HTML file.

We follow the same approach described in Kushmerick (1999) to form a learning curve of our algorithm. That is, we provide 10%, 20%, ..., 90% of data to train the model, validate the model on the test set, and record the prediction accuracy rate. We repeat this process for 10 times. Figure 3.2 shows the average prediction accuracy of different partition percentages, along with 95% confidence intervals. With only 10% of data, we achieved almost 95% of accuracy, which is higher than the one reported in Figure 4(a) of Kushmerick (1999). With more and more data, the prediction accuracy reaches a limit of approximately 97.1%, which is the same as Kushmerick (1999). However, to reach such high accuracy, we only need around 50% of the data for training which already produces 96.9%. On the other hand, almost 80% of the data is needed to reach the same accuracy in Kushmerick (1999).

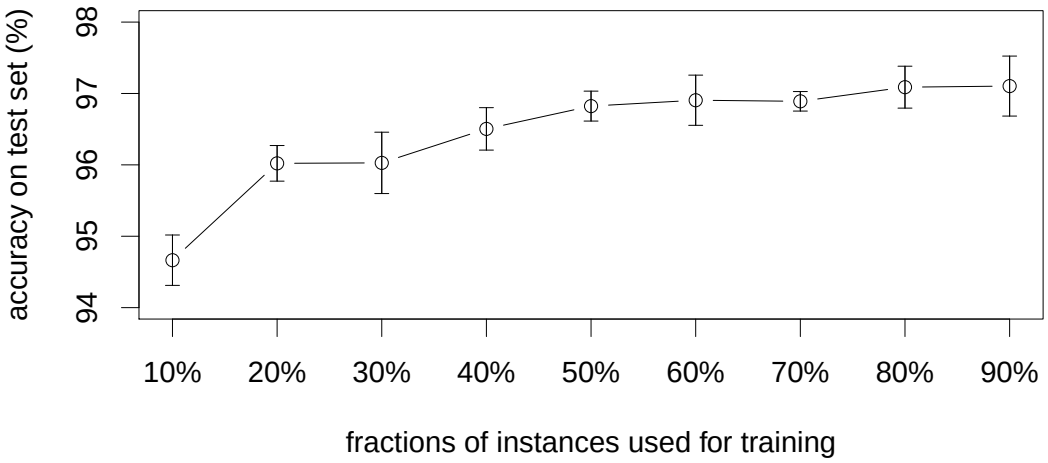


Figure 3.2: Learning Curves: average prediction accuracy with 95% confidence interval of 10 repititions

3.7 Conclusion

We propose a VB algorithm for Bayesian logistic regression model with the spike-and-slab prior. This algorithm is appealing as it can output posterior probability distribution over any the sub-models or a particular predictor, rather than a single MAP estimator of Lasso. Numerical results show that our model is comparable to LASSO in terms of the classification error rate, and returns more sparse models in some experiments.

Chapter 4

An Online Logistic Regression

4.1 Introduction

In Chapter 3, we propose a variational algorithm for Bayesian variable selection of logistic regression (Algorithm 4), which is fast and can scale to large-scaled data sets when p is large. However, in more and more real data applications, datasets may be too large to be loaded into the computer memory, which prevents us from using Algorithm 4. In such cases, our VB algorithm is no longer practical, as it needs to update the variational distributions of the latent variables which are related with the corresponding observations whose computational complexity is $O(n)$ in each iteration. It is pointed out by Hoffman et al. (2013) that variational Bayesian algorithms which requires a full pass through the data at each iteration are inefficient when the sample size is large.

To deal with large-scaled datasets, Hoffman et al. (2013) proposed a stochastic variational inference (SVI) that processes one or a mini-batch of observations in an iteration which converges faster and has higher predictive ability than the ordinary batch VB algorithm. However, as pointed by Broderick et al. (2013), the objective function of SVI is based on the full data set involving n (fixed) data points. That is, n has to be specified in advance and used in each iteration. However, in real applications such as streaming data case, n is usually unknown in advance. Although an arbitrary large n can be set in advance, the estimate of SVI is still sensitive to the choice of n (Broderick et al., 2013).

In order to deal with either large-sized data sets or streaming data, we propose an online algorithm for Bayesian logistic regression, using the idea of the online learning: it processes

the observation once at a time, making predictions while updating the model. Hence, our online algorithm is fast, because it only needs to make $O(1)$ operations in each iteration, which is much smaller than the batch algorithm $O(n)$. The scheme of our proposed online learning algorithm is different from other online VB algorithms, such as Sato (2001); Hoffman et al. (2013), which need to know sample size, at least conceptually, beforehand.

4.2 Online Learning Framework

Online learning assumes that a sequence of rounds of game are carried out. At round t, the learner is given a predictor $\mathbf{x}_t \in \mathcal{X}$, and is asked to provide an answer $p_t \in \mathcal{D}$. After that, the true label, $y_t \in \mathcal{Y}$ is revealed by the adversary, and the learner suffers a loss, $l(p_t, y_t)$. In some cases, $\mathcal{D} \supseteq \mathcal{Y}$. For example, in binary prediction, $\mathcal{Y} = \{0, 1\}$, and $\mathcal{D} = [0, 1]$. Figure 4.1 shows the setup of online learning (OL).

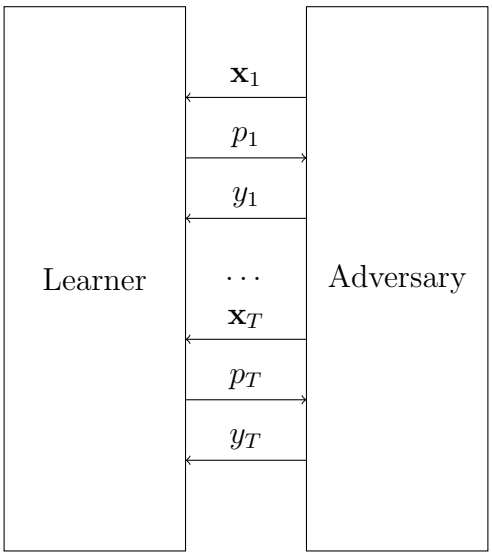


Figure 4.1: Online Learning Framework

4.3 Online Convex Optimization

Lots of efficient OL algorithms are developed in the context of online convex optimization (OCO) (Gordon, 1999; Zinkevich, 2003). The setup of OCO is very similar to OL. At round t, the learner predicts a vector $\mathbf{w}_t \in S$, where S is a convex set, receives a convex loss function $f_t : S \to \mathbb{R}$, and knows the loss to suffer by predicting $\mathbf{w}_t$, i.e., $f_t(\mathbf{w}_t)$. Then the learner may or may not update the prediction model given the new loss.

The ultimate goal of a learning problem is to minimize the cumulative loss using the best $\mathbf{w}^*$, i.e.,

$$\sum_{t=1}^{T} f_t(\mathbf{w}^*) = \min_{\mathbf{w} \in S} \sum_{t=1}^{T} f_t(\mathbf{w}). \tag{4.1}$$

However, in the online learning framework, the learner needs to make predictions at every round, so the actual loss occurred is

$$\sum_{t=1}^{T} f_t(\mathbf{w}_t). \tag{4.2}$$

Naturally, by taking the difference of (4.2) and (4.1), which measures how much the learner will suffer by not following the best $\mathbf{w}^*$, we know the performance of the learner. This leads to the so-called "regret".

Definition 1. *Define the regret of an OCO algorithm compared with a fixed prediction* $\mathbf{w} \in S$ *as*

$$\text{Regret}_T(\mathbf{w}) = \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \sum_{t=1}^{T} f_t(\mathbf{w}),$$

and the regret relative to S *as*

$$\text{Regret}_T(S) = \max_{\mathbf{w} \in S} \text{Regret}_T(\mathbf{w}).$$

Ideally, we want the regret to be sub-linear in T, meaning that $\text{Regret}_T(U)/T \to 0$ as $T \to \infty$.

One of the most widely-used OCO algorithms is the Follow-The-Regularized-Leader (FTRL) algorithm:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}\in S} \sum_{i=1}^{t} f_i(\mathbf{w}) + \psi(\mathbf{w}), \tag{4.3}$$

where $\psi : S \to \mathbb{R}$ is a regularization function. In Gordon (1999), this algorithm is called "MAP" algorithm in order to relate it with the Bayesian MAP (maximum a posteriori) estimate: $\psi(\mathbf{w}) = f_0(\mathbf{w})$ is the prior information.

When the loss function is linear (online linear optimization (OLO)), i.e., $f_t(\mathbf{w}) = \langle \mathbf{g}_t, \mathbf{w} \rangle$, and the regularizer is quadratic $\psi(\mathbf{w}) = \frac{1}{2\eta}\|\mathbf{w}\|_2^2$, FTRL works well in the sense that the updating of $\mathbf{w}_t$ is efficient and the regret is sub-linear in T if η is properly chosen. The updating equation of $\mathbf{w}$ only depends on the sum of $\mathbf{g}$'s and η, i.e.,

$$\mathbf{w}_{t+1} = -\eta \sum_{i=1}^{t} \mathbf{g}_i = \mathbf{w}_t - \eta \mathbf{g}_t.$$

Under some mild conditions, the upper bound of the regret is upper bounded by $O(\sqrt{T})$ (Zinkevich, 2003), which is sub-linear in T.

In more general problems, however, FTRL is computationally impractical, because it needs to minimize over all the loss functions plus a regularizer seen so far. If more general and complicating loss functions can be linearized, then the computation will be efficient. The linearization idea leads to the Online Sub-gradient Descent (OGD) algorithm.

Let's define the sub-gradient first.

Definition 2. *For a convex loss function f, define its sub-gradient ∂f at $\mathbf{w}$:*

$$\partial f(\mathbf{w}) = \{\mathbf{z} : f(\mathbf{u}) \geq f(\mathbf{w}) + \langle \mathbf{z}, \mathbf{u} - \mathbf{w} \rangle, \forall \mathbf{u} \in S\}.$$

If the function is differentiable at $\mathbf{w}$, then the sub-gradient contains only the gradient.

By convexity, we know that $\exists \mathbf{z} \in S$ s.t.

$$\begin{aligned} & f_t(\mathbf{w}) \geq f_t(\mathbf{w}_t) + \langle \mathbf{z}, \mathbf{w} - \mathbf{w}_t \rangle \\ \Longrightarrow & f_t(\mathbf{w}_t) - f_t(\mathbf{w}) \leq \langle \mathbf{z}, \mathbf{w}_t \rangle - \langle \mathbf{z}, \mathbf{w} \rangle \\ \Longrightarrow & \text{Regret}_T(\mathbf{w}) \leq \sum_{t=1}^{T} \left[\langle \mathbf{z}, \mathbf{w}_t \rangle - \langle \mathbf{z}, \mathbf{w} \rangle \right]. \end{aligned}$$

As such, for any convex loss function, if we replace it by the linearized loss function, the regret will still be bounded under some mild conditions, but the updating scheme will be much more efficient. This leads to the OGD algorithm:

$$\mathbf{w}_{t+1} = \arg\min_{w \in S} \sum_{i=1}^{t} \langle \mathbf{g}_i, \mathbf{w} \rangle + \frac{1}{2\eta} \|\mathbf{w}\|_2^2 = \mathbf{w}_t - \eta \mathbf{g}_t,$$

where $\mathbf{g}_i \in \partial f_i(\mathbf{w}_i)$. The updating scheme is efficient at constant costs per iteration.

4.4 Online Variational Algorithm

Suppose at iteration t, suppose that we know α, θ, ϕ_j for $j = 1, \ldots, p$, and $\bar{w}_i$ for $i = 1, \ldots, t$. Then we re-formulate the VB objective function of as

$$\begin{aligned} \Omega_t(q, \alpha, \theta, \mathbf{w}) = & \sum_{i=1}^{t} \mathbb{E} \log p(y_i | w_i, \alpha, \boldsymbol{\beta}) + \mathbb{E} \log \frac{\pi(\boldsymbol{\beta}, \boldsymbol{\gamma}, \alpha, \theta)}{q_2(\boldsymbol{\beta}, \boldsymbol{\gamma})} \\ & + \mathbb{E} \log \frac{p(\mathbf{w})}{q_1(\mathbf{w})} \\ = & - \sum_{i=1}^{t} \underbrace{f_i(\boldsymbol{\Theta})}_{\text{Loss function}} - \underbrace{\psi(\boldsymbol{\Theta})}_{\text{Regularizer}} + \text{Const}(\boldsymbol{\Theta}), \end{aligned}$$

where $\boldsymbol{\Theta} = (\mu_j, \sigma_j^2)_{j=1}^p$, and

$$
\begin{aligned}
f_i(\boldsymbol{\Theta}) =& -\left(y_i - \frac{1}{2}\right) \mathbb{E}\left(\alpha + \mathbf{x}_i^T \boldsymbol{\beta}\right) + \frac{1}{2}\bar{w}_i \mathbb{E}\left(\alpha + \mathbf{x}_i^T \boldsymbol{\beta}\right)^2, \\
\psi(\boldsymbol{\Theta}) =& \sum_{j=1}^{p} \phi_j \left[-\frac{1}{2}\log\frac{\sigma_j^2}{v_1} + \frac{\mu_j^2 + \sigma_j^2}{2v_1}\right],
\end{aligned}
$$

in which

$$
\begin{aligned}
\mathbb{E}\left(\alpha + \mathbf{x}_i^T \boldsymbol{\beta}\right) =& \alpha + \sum_{j=1}^{p} x_{ij}\mu_j\phi_j, \\
\mathbb{E}\left(\alpha + \mathbf{x}_i^T \boldsymbol{\beta}\right)^2 =& \left(\alpha + \sum_{j=1}^{p} x_{ij}\mu_j\phi_j\right)^2 + \sum_{j=1}^{p} x_{ij}^2 \left[\phi_j(1-\phi_j)\mu_j^2 + \phi_j\sigma_j^2\right].
\end{aligned}
$$

It is easy to verify that f_i and ψ are convex functions. So the original VB optimization given α, θ, ϕ_j's and $\bar{w}_i's$ is equivalent to an OCO problem w.r.t. μ_j and σ_j^2's.

Applying OGD, we have

$$
\begin{aligned}
\mu_j^{(t+1)} =& \mu_j^{(t)} - v_1 g_{t,\mu_j} \\
\frac{1}{\left(\sigma_j^2\right)^{(t+1)}} =& \frac{1}{v_1} + \sum_{i=1}^{t} \bar{w}_i x_{ij}^2 = \frac{1}{\left(\sigma_j^2\right)^{(t)}} + 2g_{t,\sigma_j^2},
\end{aligned}
$$

where

$$
\begin{aligned}
g_{t,\mu_j} =& -\left\{\left(y_t - \frac{1}{2}\right) - \bar{w}_t\left(\alpha + \sum_{k\neq j} x_{tk}\mu_k\phi_k + x_{tj}\mu_j\right)\right\} x_{tj}, \\
g_{t,\sigma_j^2} =& \frac{1}{2}\bar{w}_t x_{tj}^2,
\end{aligned}
$$

if we set $(\sigma_j^2)^{(0)} = \frac{1}{v_1}$.

Given μ_j's and σ_j^2's, at each iteration, we update ϕ_j's, $\hat{\alpha}$, and $\hat{\theta}$ using the updating

equations of Algorithm 4. That is

$$\begin{aligned}
\phi_j &\leftarrow \text{Logit}^{-1}\left\{\text{Logit}(\hat{\theta}) + \frac{1}{2}\log\frac{\sigma_j^2}{v_1} + \frac{\mu_j^2}{2\sigma_j^2}\right\} \\
\hat{\alpha} &\leftarrow \frac{\left(\mathbf{y} - \frac{1}{2} - \overline{\mathbf{W}}\mathbf{X}\bar{\boldsymbol{\beta}}\right)^T \mathbb{1}_n}{\mathbb{1}_n^T \overline{\mathbf{W}} \mathbb{1}^T} \\
\hat{\theta} &\leftarrow \frac{\sum_{j=1}^p \phi_j + a_0 - 1}{p + a_0 + b_0 - 2}
\end{aligned}$$

From Algorithm 4, $q_{2,i}(w_i) = g(w_i|m_i, z_i)$ for all $i \in \{1, \ldots, t\}$, where z_i can be updated via Eq (3.7), in which the expectation is taken w.r.t. the same current variational distribution, $q^{(t)}(\boldsymbol{\beta}, \boldsymbol{\gamma})$. However, when the data comes in streaming order, we need to re-compute all the variational distributions of w_i's at every iteration, because $q(\boldsymbol{\beta}, \boldsymbol{\gamma})$ is different throughout iterations. This causes the computing complexity to be higher and higher as time goes by, which contradicts with the idea of online learning. We therefore seek approximation: At time t, we only update the last one, i.e., $\bar{w}_t$ via

$$\begin{aligned}
z_t &\leftarrow \sqrt{\hat{\alpha}^2 + 2\hat{\alpha}\mathbf{x}_i^T\bar{\boldsymbol{\beta}} + \mathbf{x}_t^T(\overline{\boldsymbol{\beta}\boldsymbol{\beta}^T})\mathbf{x}_t} \\
\bar{w}_t &\leftarrow \frac{1}{2z_t}\tanh\left(\frac{z_t}{2}\right)
\end{aligned}$$

4.4.1 Per-Coordinate Learning Rules

A global learning rate v_1 may not be ideal, especially when the data is highly sparse (Streeter and McMahan, 2010; McMahan et al., 2013). In each iteration, only a small proportion of features are non-zero. Therefore, even if most features are not updated, their learning rate will decrease with the global one. As a result, coordinate-wise learning rate will help in the sparse setup: The learning rate will decrease only if the corresponding variable gets updated.

We use a similar per-coordinate learning rate as McMahan et al. (2013):

$$v_{t,j} = \frac{v_1}{1 + \sqrt{\sum_{i=1}^{t} g_i^2(\mu_j)}}.$$

where λ is a smoothing parameter for adaptive learning rate. We find that $v_1 = 0.1$ and $\lambda = 1$ works well in a variety of simulations and real data applications.

A summary of this online algorithm is in Algorithm 5.

4.5 Numerical Results

4.5.1 Online Algorithm *versus* Batch Algorithms

We compare Algorithm 4, Algorithm 5, and LASSO. The simulation setup is borrowed from Friedman et al. (2010). Briefly, the data generation is as follows:

1. Set sample size $n = 10000$, and feature dimension $p = 100$.

2. Randomly sample $\mathbf{X}$ from a multivariate Normal distribution, with mean 0, and the covariance $\mathrm{Cov}(\mathbf{X}_j, \mathbf{X}_{j'}) = \rho$ if $j \neq j'$, and 1 otherwise, and $\rho \in \{0, 0.1, 0.2, 0.5\}$.

3. Generate $Z_i = \sum_{j=1}^{n} X_{ij}\beta_j + k \cdot Z_i$, where $\beta_j = (-1)^j \exp(-2(j-1)/20)$, $Z \sim \mathrm{N}(0, 1)$, and k is chosen so that the signal-to-noise ratio is 3.

4. Generate the label $Y_i = 1$ with probability $\mathrm{sigmoid}(Z_i)$, and $Y_i = 0$ with probability $\mathrm{sigmoid}(-Z_i)$.

We train the three models on the training set, and test their prediction performance on a separate test set with 1000 observations which is generated from the same procedure. For Algorithm 4, we set $v_1 = 1$ for simplicity; and for LASSO, we use the default option of `cv.glmnet`.

Algorithm 5 Online VB Algorithm

Input Hyper-parameters v_1, a_0, b_0, λ.
initialize $(\mu_1, \ldots, \mu_p)$, $(\sigma_1^2, \ldots, \sigma_p^2)$, $(\phi_1, \ldots, \phi_p)$, $\hat{\alpha}$, and $\hat{\theta}$.
repeat
  Input $\mathbf{x}_t, \mathbf{y}_t, \mathbf{m}_t$
  Compute
  $z_t \leftarrow \sqrt{\hat{\alpha}^2 + 2\hat{\alpha}\mathbf{x}_i^T\bar{\boldsymbol{\beta}} + \mathbf{x}_t^T(\overline{\boldsymbol{\beta}\boldsymbol{\beta}^T})\mathbf{x}_t}$
  $\bar{w}_t \leftarrow \frac{1}{2z_t}\tanh\left(\frac{z_t}{2}\right)$

  Compute Gradients
  for j in $1:p$ **do**
    $g_{t,\mu_j} \leftarrow -\left\{\left(y_t - \frac{1}{2}\right) - \bar{w}_t\left(\alpha + \sum_{k\neq j} x_{tk}\mu_k\phi_k + x_{tj}\mu_j\right)\right\}x_{tj}$
    $g_{t,\sigma_j^2} \leftarrow \frac{1}{2}\bar{w}_t x_{tj}^2$
  end for

  Update
  for j in $1:p$ **do**
    if $x_{tj} \neq 0$ **then**
      $\mu_j^{(t+1)} \leftarrow \mu_j^{(t)} - v_{t,j}g_{t,\mu_j}$
      $\left(\sigma_j^{-2}\right)^{(t+1)} \leftarrow \left(\sigma_j^{-2}\right)^{(t)} + 2g_{t,\sigma_j^2}$
      $\phi_j^{(t+1)} \leftarrow \text{Logit}^{-1}\left\{\text{Logit}(\hat{\theta}^{(t)}) + \frac{1}{2}\log\frac{(\sigma_j^2)^{(t+1)}}{v_1} + \frac{(\mu_j^2)^{(t+1)}}{2(\sigma_j^2)^{(t+1)}}\right\}$
      $v_{t+1,j} \leftarrow \frac{v_1}{1+\sqrt{\left(\frac{v_1}{v_{t,j}}-1\right)^2 + g_t^2(\mu_j)}}$.
    else
      $\mu_j^{(t+1)} \leftarrow \mu_j^{(t)}$
      $\left(\sigma_j^{-2}\right)^{(t+1)} \leftarrow \left(\sigma_j^{-2}\right)^{(t)}$
      $\phi_j^{(t+1)} \leftarrow \phi_j^{(t)}$
      $v_{t+1,j} \leftarrow v_{t,j}$
    end if
  end for
  $\hat{\alpha} \leftarrow \frac{\left(\mathbf{y} - \frac{m_i}{2} - \overline{\mathbf{W}}\mathbf{X}\boldsymbol{\beta}\right)^T\mathbb{1}_n}{\mathbb{1}_n^T\overline{\mathbf{W}}\mathbb{1}^T}$
  $\hat{\theta} \leftarrow \frac{\sum_{j=1}^p \phi_j + a_0 - 1}{p + a_0 + b_0 - 2}$
until End of data stream.
return $(\mu_1, \ldots, \mu_p)$, $(\sigma_1^2, \ldots, \sigma_p^2)$, $(\phi_1, \ldots, \phi_p)$, $\hat{\alpha}$, and $\hat{\theta}$

We repeat this procedure for 100 times, and the prediction error rate is show in Table 4.1. The prediction error rate of the online algorithm is a little higher than the other two batch algorithms. This is what we have expected, as there is a trade-off between accuracy and efficiency. However, the online algorithm still gives reasonable prediction accuracy. Furthermore, as the online algorithm only loops over the training data set once, it is much more efficient than the batch algorithm, especially when the correlation is large. If the purpose is to efficiently learn the Bayesian logistic regression and meanwhile not to sacrifice too much accuracy, Algorithm 5 is more practical and feasible than Algorithm 4.

Table 4.1: Average Mis-Classification Error and Standard Error in Parenthesis.

ρ	Algorithm 4	Algorithm 5	LASSO
0	0.2405	0.2649	0.2414
	(0.0107)	(0.0075)	(0.0039)
0.1	0.2486	0.2659	0.2479
	(0.0081)	(0.0123)	(0.0028)
0.2	0.2562	0.2746	0.2556
	(0.0094)	(0.0152)	(0.0035)
0.5	0.2741	0.3010	0.2755
	(0.0163)	(0.0238)	(0.0053)

4.5.2 Real Data: Click-Through Rates Prediction

Background

In this section, we show the prediction performance of Algorithm 5 on a real data application. The problem is from a Kaggle competition supported by Criteo Labs (Kaggle, 2014). The goal is to predict the probability that a given user will click on an advertisement, given the page he is visiting.

The training data set contains a week of data, including $45,840,617$ observations. About 25.622% of data are labeled as 1, and the rest are 0. There are 13 integer predictors ($I1 \sim I13$) which are mostly counts, and 26 categorical predictors ($C1 \sim C26$). The competition host hashed all the categorical predictors onto 32 bits for anonymization purposes, so that

no extra information can be obtained except the data itself. For instance, the value of $C1$ of the first observation is "68fd1e64". The numbers of categories of the predictors are shown in Table 4.2, from which we can see that there are huge amounts of categories.

The test set contains $6,042,135$ observations and the same set of predictors, and most of them contain new categories.

Table 4.2: Number of Categories

Predictor	I1	I2	I3	I4	I5	I6	I7	I8
# of Categories	649	9364	14746	490	476707	11618	4142	1373
Predictor	I9	I10	I11	I12	I13	C1	C2	C3
# of Categories	7275	13	169	407	1376	1460	583	10131227
Predictor	C4	C5	C6	C7	C8	C9	C10	C11
# of Categories	2202608	305	24	12517	633	3	93145	5683
Predictor	C12	C13	C14	C15	C16	C17	C18	C19
# of Categories	8351593	3194	27	14992	5461306	10	5652	2173
Predictor	C20	C21	C22	C23	C24	C25	C26	
# of Categories	4	7046547	18	15	286181	105	142572	

The training data set is too large to be loaded into the memory on most personal computers. So batch algorithms like Algorithm 4 and `glmnet` cannot be used in this application. Instead, we compare Algorithm 5 with the FTRL-Proximal algorithm by McMahan et al. (2013) for the online logistic regression.

Feature Coding

For simplicity, we treat all the predictors as categorical. In practice, we do not find that there is much difference by treating the continuous variables as discrete or not. And as long as we feed all the models with the same set of features, this would be a fair comparison.

As p is large, the feature vector is sparse. In order to save the memory space, we apply the commonly used hash trick and the one-hot encoding. First of all, we pre-specified a large number $D = 2^k$ for $k \in \{14, 16, 18, 20\}$, which serves as the upper bound of the number of features that we used in the model. Secondly, we concatenate the name and the value of a

variable category into a string, and hase it into an integer between 0 and $D-1$. Finally, a list of hashed integers is sent to the algorithms for model training and/or prediction. That is to say, instead of using a long sparse vector, we feed a vector of length $13+26=39$ which records the position of none-zero elements to the computer.

Results from Kaggle

Table 4.3: Log-Loss from Kaggle, $D = 2^k$

	k			
	14	16	18	20
Algorithm 5	0.48045	0.47178	0.46720	0.46536
FTRL-Proximal	0.48065	0.47172	0.46692	0.46479

We submit the prediction on the test set to Kaggle. Table 4.3 shows the log-loss of Algorithm 5 and the FTRL-Proximal algorithm. The log-loss of the two algorithms are quite similar.

In this example, we do not rank the variables by their estimated posterior inclusion probabilities, because they are anonymous anyway. However, in other real applications where we know the meaning of the variables, our proposed algorithm is more preferable than the FTRL-Proximal algorithm in the sense that Algorithm 5 not only predicts as accuracy as the the FTRL-Proximal algorithm, but it can return the importance of the predictors based on the ϕ_j's.

4.6 Conclusion

The online VB algorithm is more practical than the batch VB algorithm when the datasets are too large to be loaded into the memory. Our online algorithm is established in the context of online learning that does not need to specify the sample size or given the whole training data beforehand. Its prediction accuracy is very close to the famous FTRL-Proximal

algorithm. However, it may be more preferable as it can also return the posterior inclusion probabilities of the predictors.

References

Bishop, C. M. (2006), *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Breiman, L. (2001), "Random forests," *Machine Learning*, 45, 5–32.

Broderick, T., Boyd, N., Wibisono, A., Wilson, A. C., and Jordan, M. I. (2013), "Streaming variational Bayes," in *Advances in Neural Information Processing Systems*, pp. 1727–1735.

Bühlmann, P. and van de Geer, S. (2011), *Statistics for High-Dimensional Data: Methods, Theory and Applications*, Springer Publishing Company, Incorporated, 1st ed.

Carbonetto, P. and Stephens, M. (2012), "Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies," *Bayesian Analysis*, 7, 73–108.

Clyde, M. and George, E. I. (2004), "Model uncertainty," *Statistical science*, 81–94.

Cox, D. R. (1958), "The regression analysis of binary sequences," *Journal of the Royal Statistical Society. Series B (Methodological)*, 20, 215–242.

Fan, J. and Li, R. (2001), "Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties," *Journal of the American Statistical Association*, 96, 1348–1360.

Fan, J. and Lv, J. (2010), "A selective overview of variable selection in high dimensional feature space," *Statistica Sinica*, 20, 101–148.

Friedman, J., Hastie, T., and Tibshirani, R. (2001), *The Elements of Statistical Learning*, vol. 1, Springer series in statistics Springer, Berlin.

— (2010), "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, 33, 1–22.

Frühwirth-Schnatter, S. and Frühwirth, R. (2010), "Data augmentation and MCMC for binary and multinomial logit models," in *Statistical modelling and regression structures*, Springer, pp. 111–132.

Geweke, J. et al. (1996), "Variable selection and model comparison in regression," *Bayesian statistics*, 5, 609–620.

Gordon, G. J. (1999), "Regret bounds for prediction problems," in *Proceedings of the twelfth annual conference on Computational learning theory*, ACM, pp. 29–40.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. W. (2013), "Stochastic variational inference." *Journal of Machine Learning Research*, 14, 1303–1347.

Holmes, C. C., Held, L., et al. (2006), "Bayesian auxiliary variable models for binary and multinomial regression," *Bayesian Analysis*, 1, 145–168.

Huang, X., Wang, J., and Liang, F. (2016), "A variational algorithm for Bayesian variable selection," *arXiv preprint arXiv:1602.07640*.

Jaakkola, T. S. and Jordan, M. I. (2000), "Bayesian parameter estimation via variational methods," *Statistics and Computing*, 10, 25–37.

Johnson, R. A., Wichern, D. W., et al. (1992), *Applied multivariate statistical analysis*, vol. 4, Prentice hall Englewood Cliffs, NJ.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999), "An introduction to variational methods for graphical models," *Machine Learning*, 37, 183–233.

Kaggle (2014), "Kaggle Display Advertising Challenge Dataset [Data file]," Available from http://labs.criteo.com/2014/02/kaggle-display-advertising-challenge-dataset/.

Kushmerick, N. (1999), "Learning to remove internet advertisements," in *Proceedings of the third annual conference on Autonomous Agents*, ACM, pp. 175–181.

LeCun, Y., Jackel, L., Boser, B., Denker, J., Graf, H., Guyon, I., Henderson, D., Howard, R., and Hubbard, W. (1989), "Handwritten digit recognition: Applications of neural network chips and automatic learning," *Communications Magazine, IEEE*, 27, 41–46.

Lichman, M. (2013), "UCI Machine Learning Repository," .

McMahan, H. B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., Nie, L., Phillips, T., Davydov, E., Golovin, D., et al. (2013), "Ad click prediction: a view from the trenches," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 1222–1230.

Mitchell, T. J. and Beauchamp, J. J. (1988), "Bayesian variable selection in linear regression." *Journal of the American Statistical Association*, 83, 1023–1032.

Narisetty, N. N. and He, X. (2014), "BAYESIAN VARIABLE SELECTION WITH SHRINKING AND DIFFUSING PRIORS," *The Annals of Statistics*, 42, 789–817.

Polson, N. G., Scott, J. G., and Windle, J. (2013), "Bayesian inference for logistic models using Pólya-Gamma latent variables," *Journal of the American Statistical Association*, 108, 1339–1349.

Raftery, A. E., Madigan, D., and Hoeting, J. A. (1997), "Bayesian model averaging for linear regression models," *Journal of the American Statistical Association*, 92, 179–191.

Ročková, V. and George, E. I. (2014), "EMVS: The EM approach to Bayesian variable selection," *Journal of the American Statistical Association*, 109, 828–846.

Sato, M.-A. (2001), "Online model selection based on the variational Bayes," *Neural Computation*, 13, 1649–1681.

Scott, J. G. and Sun, L. (2013), "Expectation-maximization for logistic regression," *arXiv preprint arXiv:1306.0040*.

Shao, J. and Deng, X. (2012), "Estimation in high-dimensional linear models with deterministic design matrices," *The Annals of Statistics*, 40, 812–831.

Streeter, M. and McMahan, H. B. (2010), "Less regret via online conditioning," *arXiv preprint arXiv:1002.4862*.

Tibshirani, R. (1994), "Regression Shrinkage and Selection Via the Lasso," *Journal of the Royal Statistical Society, Series B*, 58, 267–288.

Wang, S., Nan, B., Rosset, S., and Zhu, J. (2011), "Random lasso," *The Annals of Applied Statistics*, 5, 468–485.

Zinkevich, M. (2003), "Online Convex Programming and Generalized Infinitesimal Gradient Ascent," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, eds. Fawcett, T. and Mishra, N., pp. 928–936.

Appendices

Appendix A.

Appendix A.1 Gibbs sampler

We want to obtain samples of $(\beta_1, \ldots, \beta_p)$, $(\gamma_1, \ldots, \gamma_p)$, θ, and σ^2 from their joint distribution. using Gibbs sampling. That is to say, each parameter is sampled from its conditional distribution given all the other parameters and the data. However, as the marginal probability of β_j is a mixture of a normal distribution and a point mass, if we marginally sample them, the Markov chain is not ergodic. Hence, we use the strategy that we sample (β_j, γ_j) together given all the other parameters.

For $j = 1$, we sample (β_1, γ_1) given $\Theta = \{(\beta_1, \ldots, \beta_p), (\gamma_1, \ldots, \gamma_p), \theta, \sigma^2\}$. That conditional distribution is

$$\pi(\beta_1, \gamma_1|\Theta, \mathbf{y}) = \pi(\beta_1|\gamma_1, \Theta, \mathbf{y})\pi(\gamma_1|\Theta, \mathbf{y}).$$

By the Bayes' theorem, $\pi(\gamma_1|\Theta, \mathbf{y}) \propto p(\mathbf{y}|\gamma_1, \Theta)\pi(\gamma_1)$. Then we need to find $p(\mathbf{y}|\gamma_1, \Theta)$.

- Given $\beta_1|\gamma_1 = 1 \sim \mathrm{N}(0, v_1\sigma^2)$, and $\mathbf{y} = \mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]} + \mathbf{X}_1\beta_1 + \boldsymbol{\epsilon}_n$, we know that

$$\left.\begin{array}{l} \mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]}|\gamma_1, \Theta = \mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]} \\ \mathbf{X}_1\beta_1|\gamma_1, \Theta \sim \mathrm{N}(0, v_1\sigma^2\mathbf{X}_1\mathbf{X}_1^T) \\ \boldsymbol{\epsilon}_n|\gamma_1, \Theta \sim \mathrm{N}(0, \sigma^2\mathbf{I}_n) \end{array}\right\} \Longrightarrow \begin{array}{c} \mathbf{y}|\gamma_1, \Theta \sim \mathrm{N}\left(\mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]}, \sigma^2(v_1\mathbf{X}_1\mathbf{X}_1^T + \mathbf{I}_n)\right) \\ (\because \textit{Independence}) \end{array}$$

- Similarly, given $\beta_1|\gamma_1 = 0 \sim \delta_0$, we know that $\mathbf{y}|\gamma_1, \Theta \sim \mathrm{N}\left(\mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]}, \sigma^2\mathbf{I}_n\right)$

Next, we compute the Bayes factor (odds) of γ_1:

$$
\begin{aligned}
&BF(\gamma_1)\\
=&\frac{\pi(\gamma_1=1|\Theta,\mathbf{y})}{\pi(\gamma_1=0|\Theta,\mathbf{y})}\\
=&\frac{\pi(\gamma_1=1)p(\mathbf{y}|\gamma_1=1,\Theta)}{\pi(\gamma_1=0)p(\mathbf{y}|\gamma_1=0,\Theta)}\\
=&\frac{\theta(2\pi)^{-n/2}|\sigma^2(v_1\mathbf{X}_1\mathbf{X}_1^T+\mathbf{I}_n)|^{-1/2}}{(1-\theta)(2\pi\sigma^2)^{-n/2}}\\
&\cdot\frac{\exp\left\{-\frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]})^T(v_1\mathbf{X}_1\mathbf{X}_1^T+\mathbf{I}_n)^{-1}(\mathbf{y}-\mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]})\right\}}{\exp\left\{-\frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]})^T(\mathbf{y}-\mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]})\right\}}\\
=&\frac{\theta}{1-\theta}\frac{1}{\sqrt{1+v_1\mathbf{X}_1^T\mathbf{X}_1}}\exp\left\{\frac{1}{2\sigma^2\left(\frac{1}{v_1}+\mathbf{X}_1^T\mathbf{X}_1\right)}(\mathbf{y}-\mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]})^T\mathbf{X}_1\mathbf{X}_1^T(\mathbf{y}-\mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]})\right\}\\
=&\frac{\theta}{1-\theta}\frac{1}{\sqrt{1+v_1\|\mathbf{X}_1\|_2^2}}\exp\left\{\frac{\mu_1^2}{2\sigma_1^2}\right\},
\end{aligned}
$$

where $\mu_1=\frac{(\mathbf{y}-\mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]})^T\mathbf{X}_1}{\|\mathbf{X}_1\|_2^2+\frac{1}{v_1}}$, and $\sigma_1^2=\frac{\sigma^2}{\|\mathbf{X}_1\|_2^2+\frac{1}{v_1}}$. Then we sample γ_1 from a Bernoulli distribution, with posterior mean $\frac{1}{1+\exp\{-BF(\gamma_1)\}}$.

Next, we figure out $\pi(\beta_1|\gamma_1,\Theta,\mathbf{y})$. Given the current value of γ_1, we sample β_1 from the corresponding components. That is

$$
\begin{aligned}
&\pi(\beta_1|\gamma_1=1,\Theta,\mathbf{y})\\
\propto&p(\mathbf{y}|\beta_1,\gamma_1=1,\Theta)\pi(\beta_1|\gamma_1=1,\Theta)\\
\propto&\exp\left\{-\frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]}-\mathbf{X}_1\beta_1)^T(\mathbf{y}-\mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]}-\mathbf{X}_1\beta_1)\right\}\cdot\exp\left\{-\frac{\beta_1^2}{2v_1\sigma^2}\right\}\\
\propto&\exp\left\{\frac{1}{\sigma^2}(\mathbf{y}-\mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]})^T\mathbf{X}_1\beta_1-\frac{1}{2\sigma^2}\left[\|\mathbf{X}_1\|_2^2+\frac{1}{v_1}\right]\beta_1^2\right\}\\
\Longrightarrow&\beta_1|\gamma_1=1,\Theta,\mathbf{y}\sim\mathrm{N}\left(\frac{(\mathbf{y}-\mathbf{X}_{[,-1]}\boldsymbol{\beta}_{[-1]})^T\mathbf{X}_1}{\|\mathbf{X}_1\|_2^2+\frac{1}{v_1}},\frac{\sigma^2}{\|\mathbf{X}_1\|_2^2+\frac{1}{v_1}}\right),
\end{aligned}
$$

and

$$
\begin{aligned}
&\pi(\beta_1|\gamma_1 = 0, \Theta, \mathbf{y}) \\
\propto &p(\mathbf{y}|\beta_1, \gamma_1 = 0, \Theta)\pi(\beta_1|\gamma_1 = 0, \Theta)\delta_0(\beta_1) \\
\propto &\delta_0(\beta_1) \\
\Longrightarrow &\beta_1|\gamma_1 = 0, \Theta, \mathbf{y} = 0.
\end{aligned}
$$

Similarly, we can sample the other (β_j, γ_j)'s.

Next, we sample σ^2 while fixing all the other parameters

$$
\begin{aligned}
&\pi(\sigma^2|\boldsymbol{\beta}, \boldsymbol{\gamma}, \mathbf{y}, \theta) \\
\propto &p(\mathbf{y}|\boldsymbol{\beta}, \boldsymbol{\gamma}, \sigma^2, \theta)\pi(\boldsymbol{\beta}, \boldsymbol{\gamma}|\sigma^2, \theta)\pi(\sigma^2) \\
\propto &(\sigma^2)^{-\frac{n}{2}} \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\right\} \\
&\cdot \prod_{j=1}^{p}\left[\frac{1}{\sqrt{2\pi v_1 \sigma^2}} \exp\left(-\frac{\beta_j^2}{2v_1\sigma^2}\right)\right]^{\gamma_j} \\
&\cdot (\sigma^2)^{-\frac{\nu}{2}-1} \exp\left(-\frac{\nu\lambda}{2\sigma^2}\right) \\
= &(\sigma^2)^{-\frac{n+\sum_{j=1}^{p}\gamma_j+\nu}{2}-1} \exp\left\{-\frac{1}{2\sigma^2}\left[(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \frac{1}{v_1}\sum_j \beta_j^2 + \nu\lambda\right]\right\} \\
\Longrightarrow &\sigma^2|\boldsymbol{\beta}, \boldsymbol{\gamma}, \mathbf{y}, \theta \sim IG\left(\frac{n + \sum_{j=1}^{p}\gamma_j + \nu}{2}, \frac{1}{2}\left((\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \frac{1}{v_1}\sum_j \beta_j^2 + \nu\lambda\right)\right).
\end{aligned}
$$

Finally, we sample θ

$$
\begin{aligned}
&\pi(\theta|\boldsymbol{\beta}, \boldsymbol{\gamma}, \sigma^2, \mathbf{y}) \\
\propto &\pi(\boldsymbol{\beta}, \boldsymbol{\gamma}|\sigma^2, \theta)\pi(\theta) \\
\propto &\prod_{j=1}^{p} \theta^{\gamma_j}(1-\theta)^{1-\gamma_j} \cdot \theta^{a_0-1}(1-\theta)^{b_0-1} \\
= &\theta^{\sum_{j=1}^{p}\gamma_j + a_0 - 1}(1-\theta)^{p-\sum_{j=1}^{p}\gamma_j + b_0 - 1} \\
\Longrightarrow &\theta|\boldsymbol{\beta}, \boldsymbol{\gamma}, \sigma^2, \mathbf{y} \sim \text{Beta}\left(\sum_{j=1}^{p}\gamma_j + a_0, p - \sum_{j=1}^{p}\gamma_j + b_0\right).
\end{aligned}
$$

Appendix A.2 Updates for Algorithm 2

We provide the detailed derivation of the updating equations of Algorithm 2.

The full posterior is

$$
\begin{aligned}
& p(\boldsymbol{\beta}, \boldsymbol{\gamma}, \sigma^2, \theta | \mathbf{y}) \\
\propto\ & p(\mathbf{y}|\boldsymbol{\beta}, \sigma^2) p(\boldsymbol{\beta}|\boldsymbol{\gamma}) p(\boldsymbol{\gamma}|\theta) \pi(\theta) \pi(\sigma^2) \\
\propto\ & \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2}{2\sigma^2}\right) \\
& \cdot \prod_{j=1}^{p} \left[\frac{1}{\sqrt{2\pi v_1 \sigma^2}} \exp\left(-\frac{\beta_j^2}{2 v_1 \sigma^2}\right)\right]^{\gamma_j} [\delta_0(\beta_j)]^{1-\gamma_j} \\
& \cdot \prod_{j=1}^{p} \theta^{\gamma_j} (1-\theta)^{1-\gamma_j} \cdot \theta^{a_0 - 1} (1-\theta)^{b_0 - 1} \cdot (\sigma^2)^{-\frac{\nu}{2} - 1} \exp\left(-\frac{\nu\lambda}{2\sigma^2}\right)
\end{aligned}
\tag{4}
$$

The approximating posterior distribution of $(\boldsymbol{\beta}, \boldsymbol{\gamma})$ is

$$
q(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \prod_{j=1}^{p} q_j(\beta_j, \gamma_j) = \prod_{j=1}^{p} [\phi_j f_j(\beta_j)]^{\gamma_j} [(1-\phi_j)\delta_0(\beta_j)]^{1-\gamma_j} \tag{5}
$$

Combining Eq (4) and Eq (5), the objective function becomes

$$
\begin{aligned}
\Omega(q_1,\ldots,q_p,\theta,\sigma^2) =& \mathbb{E}^{q_1,\ldots,q_p} \log \frac{p(\mathbf{y}|\boldsymbol{\beta},\sigma^2)p(\boldsymbol{\beta}|\boldsymbol{\gamma})p(\boldsymbol{\gamma}|\theta)\pi(\theta)\pi(\sigma^2)}{\prod_{j=1}^{p}[\phi_j f_j(\beta_j)]^{\gamma_j}[(1-\phi_j)\delta_0(\beta_j)]^{1-\gamma_j}} \\
\propto& \mathbb{E}^{q_1,\ldots,q_p}\Bigg\{\sum_{i=1}^{n}\left[-\frac{1}{2}\log(2\pi)-\frac{1}{2}\log(\sigma^2)-\frac{(y_i-\mathbf{x}_i^T\boldsymbol{\beta})^2}{2\sigma^2}\right] \\
&+\sum_{j=1}^{p}\gamma_j\left[-\frac{1}{2}\log(2\pi)-\frac{1}{2}\log(v_1\sigma^2)-\frac{\beta_j^2}{2v_1\sigma^2}\right]+(1-\gamma_j)\log[\delta_0(\beta_j)] \\
&+\sum_{j=1}^{p}[\gamma_j\log(\theta)+(1-\gamma_j)\log(1-\theta)] \\
&+(a_0-1)\log(\theta)+(b_0-1)\log(1-\theta) \\
&+(-\frac{\nu}{2}-1)\log(\sigma^2)-\frac{\nu\lambda}{2\sigma^2} \\
&-\sum_{j=1}^{p}\gamma_j(\log(\phi_j)+\log[f_j(\beta_j)])+(1-\gamma_j)(\log(1-\phi_j)+\log[\delta_0(\beta_j)])\Bigg\}
\end{aligned}
$$

To update $q_j(\beta_j,\phi_j)$ for some $j\in\{1,\ldots,p\}$ given other approximating distributions, $\hat{\theta}$,

and $\hat{\sigma}^2$ fixed, we have

$$
\begin{aligned}
\Omega(q_j, \cdot) \propto & \mathbb{E}^{q_j} \Bigg\{ \sum_{i=1}^{n} \left[-\frac{(y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2}{2\hat{\sigma}^2} \right] \\
& + \gamma_j \left[-\frac{1}{2} \log(2\pi) - \frac{1}{2} \log(v_1 \sigma^2) - \frac{\beta_j^2}{2 v_1 \sigma^2} \right] + (1 - \gamma_j) \log[\delta_0(\beta_j)] \\
& + \gamma_j \log(\hat{\theta}) + (1 - \gamma_j) \log(1 - \hat{\theta}) \\
& - \gamma_j (\log(\phi_j) + \log[f_j(\beta_j)]) + (1 - \gamma_j)(\log(1 - \phi_j) + \log[\delta_0(\beta_j)]) \Bigg\} \\
= & \mathbb{E}^{q_j} \Bigg\{ -\frac{1}{2\hat{\sigma}^2} \sum_{i=1}^{n} \mathbb{E}^{q_{[-j]}} (y_i - \mathbf{x}_{i[-j]}^T \boldsymbol{\beta}_{[-j]} - x_{ij} \beta_j)^2 \\
& + \gamma_j \left[-\frac{1}{2} \log(2\pi) - \frac{1}{2} \log(v_1 \sigma^2) - \frac{\beta_j^2}{2 v_1 \sigma^2} + \log(\hat{\theta}) - \log(\phi_j) - \log[f_j(\beta_j)] \right] \\
& + (1 - \gamma_j)[\log(1 - \hat{\theta}) - \log(1 - \phi_j)] \Bigg\} \\
\propto & \mathbb{E}^{q_j} \Bigg\{ 2 \frac{1}{2\hat{\sigma}^2} \sum_{i=1}^{n} (y_i - \mathbf{x}_{i[-j]}^T \mathbb{E}^{q_{[-j]}} (\boldsymbol{\beta}_{[-j]})) x_{ij} \cdot \beta_j - \frac{1}{2\hat{\sigma}^2} \sum_{i=1}^{n} x_{ij}^2 \cdot \beta_j^2 \\
& + \gamma_j \left[-\frac{1}{2} \log(2\pi) - \frac{1}{2} \log(v_1 \sigma^2) - \frac{\beta_j^2}{2 v_1 \sigma^2} + \log(\hat{\theta}) - \log(\phi_j) - \log[f_j(\beta_j)] \right] \\
& + (1 - \gamma_j)[\log(1 - \hat{\theta}) - \log(1 - \phi_j)] \Bigg\} \\
= & \phi_j \mathbb{E}^{q_j}_{\beta_j | \gamma_j = 1} \Bigg\{ 2 \frac{1}{2\hat{\sigma}^2} \sum_{i=1}^{n} (y_i - \mathbf{x}_{i[-j]}^T \mathbb{E}^{q_{[-j]}} (\boldsymbol{\beta}_{[-j]})) x_{ij} \cdot \beta_j - \frac{1}{2\hat{\sigma}^2} \left[\sum_{i=1}^{n} x_{ij}^2 + \frac{1}{v_1} \right] \cdot \beta_j^2 \\
& - \log[f_j(\beta_j)] \Bigg\} \qquad (6) \\
& - \phi_j \left[\frac{1}{2} \log(2\pi) + \frac{1}{2} \log(v_1 \sigma^2) + \log(\phi_j) - \log(\hat{\theta}) \right] \\
& + (1 - \phi_j)[\log(1 - \hat{\theta}) - \log(1 - \phi_j)]
\end{aligned}
$$

The last step is from the fact:

$$
\begin{aligned}
q_j(\gamma_j) =& \phi_j,\ q_j(\beta_j|\gamma_j = 1) = f_j(\beta_j),\ q_j(\beta_j|\gamma_j = 0) = \delta_0(\beta_j) \\
\Longrightarrow \mathbb{E}^{q_j} g(\beta_j, \gamma_j) =& \phi_j \mathbb{E}^{f_j} g(\beta_j, \gamma_j = 1) + (1 - \phi_j) g(\beta_j = 0, \gamma_j)
\end{aligned}
$$

To maximize the objective function, we need to have

$$
\log[f_j(\beta_j)] \propto 2\frac{1}{2\hat{\sigma}^2} \sum_{i=1}^{n} (y_i - \mathbf{x}_{i[-j]}^T \mathbb{E}^{q_{[-j]}}(\boldsymbol{\beta}_{[-j]})) x_{ij} \cdot \beta_j - \frac{1}{2\hat{\sigma}^2} \left[\sum_{i=1}^{n} x_{ij}^2 + \frac{1}{v_1} \right] \cdot \beta_j^2,
$$

which implies that $f_j(\beta_j)$ is a probability density function of a Normal distribution $\mathrm{N}(\beta_j; \mu_j, \sigma_j^2)$, where

$$
\begin{aligned}
\mu_j =& \frac{(\mathbf{y} - \mathbf{X}[, -j]\mathbb{E}(\boldsymbol{\beta}_{[-j]}))^T \mathbf{X}_j}{\mathbf{X}_j^T \mathbf{X}_j + \frac{1}{v_1}} = \frac{(\mathbf{y} - \mathbf{X}[, -j]\mathbb{E}(\boldsymbol{\beta}_{[-j]}))^T \mathbf{X}_j}{n + \frac{1}{v_1}} \\
\sigma_j^2 =& \frac{\hat{\sigma}^2}{\mathbf{X}_j^T \mathbf{X}_j + \frac{1}{v_1}} = \frac{\hat{\sigma}^2}{n + \frac{1}{v_1}}
\end{aligned}
$$

By symmetry, we know that $\mu_j = \frac{(\mathbf{y} - \mathbf{X}[,-j]\boldsymbol{\Phi}_{[-j,-j]}\boldsymbol{\mu}_{[-j]}))^T \mathbf{X}_j}{n + \frac{1}{v_1}}$, where $\boldsymbol{\Phi} = \mathrm{diag}\{\phi_1, \ldots, \phi_p\}$. Then we plug in $f_j(\beta_j)$ to Eq (6), we have

$$
\begin{aligned}
\Omega(q_j, \cdot) \propto& \phi_j \mathbb{E}^{q_j}_{\beta_j|\gamma_j=1} \left\{ \frac{\mu_j \beta_j}{\sigma_2^2} - \frac{\beta_j^2}{2\sigma_j^2} + \frac{1}{2}\log(2\pi) + \frac{1}{2}\log(\sigma_j^2) + \frac{(\beta_j - \mu_j)^2}{2\sigma_j^2} \right\} \\
& - \phi_j \left[\frac{1}{2}\log(2\pi) + \frac{1}{2}\log(v_1\sigma^2) + \log(\phi_j) - \log(\hat{\theta}) \right] \\
& + (1 - \phi_j)[\log(1 - \hat{\theta}) - \log(1 - \phi_j)] \\
=& \phi_j \left[\frac{1}{2}\log\frac{\sigma_j^2}{v_1\sigma^2} + \frac{\mu_j^2}{2\sigma_j^2} - \log(\phi_j) + \log(\hat{\theta}) \right] + (1 - \phi_j)[\log(1 - \hat{\theta}) - \log(1 - \phi_j)] \\
\Longrightarrow \frac{\partial \Omega}{\partial \phi_j} =& \frac{1}{2}\log\frac{\sigma_j^2}{v_1\sigma^2} + \frac{\mu_j^2}{2\sigma_j^2} - \log(\phi_j) + \log(\hat{\theta}) - 1 - \log(1 - \hat{\theta}) + \log(1 - \phi_j) + 1 \stackrel{let}{=} 0 \\
\Longrightarrow \log\frac{\phi_j}{1 - \phi_j} =& \log\frac{\hat{\theta}}{1 - \hat{\theta}} + \frac{1}{2}\log\frac{\sigma_j^2}{v_1\sigma^2} + \frac{\mu_j^2}{2\sigma_j^2}.
\end{aligned}
$$

The point estimate of θ can be derived as follows

$$\begin{aligned}
\frac{\partial \Omega}{\partial \theta} =& \frac{\partial}{\partial \theta} \mathbb{E}^{q_1,\ldots,q_p} \log \left[\prod_{j=1}^{p} \theta^{\gamma_j} (1-\theta)^{1-\gamma_j} \cdot \theta^{a_0-1} (1-\theta)^{b_0-1} \right] \\
=& \frac{\partial}{\partial \theta} \left[\left(\sum_{j=1}^{p} \phi_j + a_0 - 1 \right) \log(\theta) + \left(p - \sum_{j=1}^{p} \phi_j + b_0 - 1 \right) \log(1-\theta) \right] \overset{let}{=} 0 \\
\Longrightarrow \hat{\theta} =& \frac{\sum_{j=1}^{p} \phi_j + a_0 - 1}{p + a_0 + b_0 - 2}.
\end{aligned}$$

Finally, we compute the point estimate of σ^2 as

$$\begin{aligned}
\frac{\partial \Omega}{\partial \sigma^2} =& \frac{\partial}{\partial \sigma^2} \mathbb{E}^{q_1,\ldots,q_p} \log \Bigg\{ \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2}{2\sigma^2} \right) \\
& \cdot \prod_{j=1}^{p} \left[\frac{1}{\sqrt{2\pi v_1 \sigma^2}} \exp\left(-\frac{\beta_j^2}{2 v_1 \sigma^2} \right) \right]^{\gamma_j} \\
& \cdot (\sigma^2)^{-\frac{\nu}{2}-1} \exp\left(-\frac{\nu\lambda}{2\sigma^2} \right) \Bigg\} \\
=& \frac{\partial}{\partial \sigma^2} \mathbb{E}^{q_1,\ldots,q_p} \Bigg\{ \sum_{i=1}^{n} -\frac{1}{2} \log \sigma^2 - \frac{(y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2}{2\sigma^2} + \sum_{j=1}^{p} \gamma_j \left[-\frac{1}{2} \log \sigma^2 - \frac{\beta_j^2}{2 v_1 \sigma^2} \right] \\
& - \left(\frac{\nu}{2} + 1 \right) \log \sigma^2 - \frac{\nu\lambda}{2\sigma^2} \Bigg\} \\
=& \frac{\partial}{\partial \sigma^2} \Bigg\{ -\frac{n}{2} \log \sigma^2 - \frac{\sum_{i=1}^{n} \mathbb{E}(y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2}{2\sigma^2} - \frac{\sum_{j=1}^{p} \phi_j}{2} \log \sigma^2 - \sum_{j=1}^{p} \phi_j \frac{\mu_j^2 + \sigma_j^2}{2 v_1 \sigma^2} \\
& - \left(\frac{\nu}{2} + 1 \right) \log \sigma^2 - \frac{\nu\lambda}{2\sigma^2} \Bigg\} \\
=& \frac{\partial}{\partial \sigma^2} \Bigg\{ -\frac{1}{2} \left[n + \sum_{j=1}^{p} \phi_j + \nu + 2 \right] \log \sigma^2 \\
& - \left[\sum_{i=1}^{n} \mathbb{E}(y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2 + \sum_{j=1}^{p} \phi_j \frac{\mu_j^2 + \sigma_j^2}{v_1} + \nu\lambda \right] \frac{1}{2\sigma^2} \Bigg\} \\
\overset{let}{=}& 0
\end{aligned}$$

Solving for the last equation, we have

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^n \mathbb{E}(y_i - \mathbf{x}_i^T\boldsymbol{\beta})^2 + \sum_{j=1}^p \phi_j \frac{\mu_j^2+\sigma_j^2}{v_1} + \nu\lambda}{n + \sum_{j=1}^p \phi_j + \nu + 2},$$

where $\mathbb{E}(y_i - \mathbf{x}_i^T\boldsymbol{\beta})^2 = [y_i - \mathbf{x}_i^T(\boldsymbol{\phi} \circ \boldsymbol{\mu})]^2 + \sum_{j=1}^p x_{ij}^2[\phi_j(1-\phi_j)\mu_j + \phi_j\sigma_j^2]$.

Full expansion of the objective function

$$\begin{aligned}
&E\left\{\sum_{i=1}^n \left(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\hat{\sigma}^2) - \frac{(y_i - \hat{\alpha} - \mathbf{x}_i^T\boldsymbol{\beta})^2}{2\hat{\sigma}^2}\right)\right. \\
&+\sum_{j=1}^p \left[\gamma_j\left(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(v_1\hat{\sigma}^2) - \frac{\beta_j^2}{2v_1\hat{\sigma}^2} + \log(\hat{\theta})\right)\right. \\
&\left.\left. + (1-\gamma_j)\left(\log\delta_0(\beta_j) + \log(1-\hat{\theta})\right)\right] \\
&+(a_0-1)\log(\hat{\theta}) + (b_0-1)\log(1-\hat{\theta}) - (\frac{\nu}{2}+1)\log(\hat{\sigma}^2) - \frac{\nu\lambda}{2\hat{\sigma}^2} \\
&-\sum_{j=1}^p \left[\gamma_j\left(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\sigma_j^2) - \frac{(\beta_j-\mu_j)^2}{2\sigma_j^2} + \log(\phi_j)\right)\right. \\
&\left.\left. + (1-\gamma_j)\big(\log\delta_0(\beta_j) + \log(1-\phi_j)\big)\right]\right\} \\
&= -\frac{n}{2}\log(\hat{\sigma}^2) - \frac{\sum_{i=1}^n E(y_i - \hat{\alpha} - \mathbf{x}_i^T\boldsymbol{\beta})^2}{2\hat{\sigma}^2} \\
&+\sum_{j=1}^p \left[\phi_j\left(\frac{1}{2}\log\frac{\sigma_j^2}{v_1\sigma^2} + \log\frac{\hat{\theta}}{\phi_j} + \frac{1}{2} - \frac{\mu_j^2+\sigma_j^2}{2v_1\hat{\sigma}^2}\right) + (1-\phi_j)\log\frac{1-\hat{\theta}}{1-\phi_j}\right] \\
&+(a_0-1)\log(\hat{\theta}) + (b_0-1)\log(1-\hat{\theta}) - (\frac{\nu}{2}+1)\log(\hat{\sigma}^2) - \frac{\nu\lambda}{2\hat{\sigma}^2} \\
&=\sum_{j=1}^p \left[\phi_j\left(\frac{1}{2}\log\frac{\sigma_j^2}{v_1\sigma^2} + \log\frac{\hat{\theta}}{\phi_j} + \frac{1}{2} - \frac{\mu_j^2+\sigma_j^2}{2v_1\hat{\sigma}^2}\right) + (1-\phi_j)\log\frac{1-\hat{\theta}}{1-\phi_j}\right] \\
&+(a_0-1)\log(\hat{\theta}) + (b_0-1)\log(1-\hat{\theta}) - \left(\frac{n}{2}+\frac{\nu}{2}+1\right)\log(\hat{\sigma}^2) \\
&-\frac{1}{2\hat{\sigma}^2}\left(\sum_{i=1}^n E(y_i - \hat{\alpha} - \mathbf{x}_i^T\boldsymbol{\beta})^2 + \nu\lambda\right).
\end{aligned}$$

Appendix A.3 Updates for Algorithm 3

We provide the derivation of the updating equation for ϕ_j's, which involves some linear approximations. The derivation of other variational distribution and MAP estimators is trivial.

We provide the detailed derivation of the updating equations of Algorithm 2.

i. We update $\{\sigma_j^2\}_{j=1}^p$ while fixing $\{\phi_j\}_{j=1}^p$, $\{\mu_j\}_{j=1}^p$, and other point estimates. Then the objective function becomes

$$\begin{aligned}
\Omega(\sigma_1^2,\ldots,\sigma_p^2) \propto & \mathbb{E}^{q_1,\ldots,q_p}\Bigg\{\sum_{i=1}^n\left[-\frac{(y_i-\mathbf{x}_i^T\boldsymbol{\beta})^2}{2\hat{\sigma}^2}\right] \\
& + \sum_{j=1}^p \gamma_j\left[-\frac{1}{2}\log(2\pi)-\frac{1}{2}\log(v_1\sigma^2)-\frac{\beta_j^2}{2v_1\sigma^2}\right]+(1-\gamma_j)\log[\delta_0(\beta_j)] \\
& + \sum_{j=1}^p \gamma_j\log(\hat{\theta})+(1-\gamma_j)\log(1-\hat{\theta}) \\
& - \sum_{j=1}^p \gamma_j(\log(\phi_j)+\log[f_j(\beta_j)])+(1-\gamma_j)(\log(1-\phi_j)+\log[\delta_0(\beta_j)])\Bigg\} \\
\propto & \mathbb{E}^{q_1,\ldots,q_p}\left\{\sum_{i=1}^n\left[-\frac{(y_i-\mathbf{x}_i^T\boldsymbol{\beta})^2}{2\sigma^2}\right]+\sum_{j=1}^p\gamma_j\left[-\frac{\beta_j^2}{2v_1\sigma^2}-\log[f_j(\beta_j)]\right]\right\} \\
\propto & \sum_{j=1}^p -\phi_j\frac{\|\mathbf{X}_j\|^2}{2\hat{\sigma}^2}\sigma_j^2-\phi_j\frac{1}{2v_1\hat{\sigma}^2}\sigma_j^2+\frac{1}{2}\phi_j\log(\sigma_j^2) \\
\Longrightarrow \sigma_j^2 = & \frac{\hat{\sigma}^2}{n+\frac{1}{v_1}}, \quad j=1,\ldots,p.
\end{aligned}$$

ii. Next, we update $\{\mu_j\}_{j=1}^p$ while fixing $\{\phi_j\}_{j=1}^p$, $\{\sigma_j^2\}_{j=1}^p$, and other point estimates. Then

the objective function becomes

$$
\begin{aligned}
\Omega(\boldsymbol{\mu}) \propto & \mathbb{E}^{q_1,\ldots,q_p}\left\{\sum_{i=1}^{n}\left[-\frac{(y_i-\mathbf{x}_i^T\boldsymbol{\beta})^2}{2\hat{\sigma}^2}\right]\right. \\
& +\sum_{j=1}^{p}\gamma_j\left[-\frac{1}{2}\log(2\pi)-\frac{1}{2}\log(v_1\sigma^2)-\frac{\beta_j^2}{2v_1\sigma^2}\right]+(1-\gamma_j)\log[\delta_0(\beta_j)] \\
& +\sum_{j=1}^{p}\gamma_j\log(\hat{\theta})+(1-\gamma_j)\log(1-\hat{\theta}) \\
& \left.-\sum_{j=1}^{p}\gamma_j(\log(\phi_j)+\log[f_j(\beta_j)])+(1-\gamma_j)(\log(1-\phi_j)+\log[\delta_0(\beta_j)])\right\} \\
\propto & \mathbb{E}^{q_1,\ldots,q_p}\left\{\sum_{i=1}^{n}\left[-\frac{(y_i-\mathbf{x}_i^T\boldsymbol{\beta})^2}{2\hat{\sigma}^2}\right]+\sum_{j=1}^{p}\gamma_j\left[-\frac{\beta_j^2}{2v_1\sigma^2}+\frac{(\beta_j-\mu_j)^2}{2\hat{\sigma}_j^2}]\right]\right\} \\
\propto & -\frac{1}{2\hat{\sigma}^2}\mathbb{E}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2-\frac{1}{2v_1\hat{\sigma}^2}\sum_{j=1}^{p}\mathbb{E}(\gamma_j\beta_j^2) \\
= & -\frac{1}{2\hat{\sigma}^2}\left[(\mathbf{y}-\mathbf{X}\boldsymbol{\Phi}\boldsymbol{\mu})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\Phi}\boldsymbol{\mu})+\mathbb{E}\|\mathbf{X}(\boldsymbol{\Phi}\boldsymbol{\mu}-\boldsymbol{\beta})\|_2^2\right]-\frac{1}{2v_1\hat{\sigma}^2}\phi_j(\mu_j^2+\sigma_j^2) \\
\propto & -\frac{1}{2\hat{\sigma}^2}\left[(\mathbf{y}-\mathbf{X}\boldsymbol{\Phi}\boldsymbol{\mu})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\Phi}\boldsymbol{\mu})+\operatorname{tr}(\mathbf{X}^T\mathbf{X}\operatorname{Cov}(\boldsymbol{\beta}))\right]-\frac{1}{2v_1\hat{\sigma}^2}\boldsymbol{\mu}^T\boldsymbol{\Phi}\boldsymbol{\mu} \\
\propto & -\frac{1}{2\hat{\sigma}^2}\left[(\mathbf{y}-\mathbf{X}\boldsymbol{\Phi}\boldsymbol{\mu})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\Phi}\boldsymbol{\mu})+\boldsymbol{\mu}^T\operatorname{diag}\{\mathbf{X}^T\mathbf{X}\}\boldsymbol{\Phi}(\mathbf{I}-\boldsymbol{\Phi})\boldsymbol{\mu}\right]-\frac{1}{2v_1\hat{\sigma}^2}\boldsymbol{\mu}^T\boldsymbol{\Phi}\boldsymbol{\mu} \\
\Longrightarrow & \frac{\partial\Omega}{\partial\boldsymbol{\mu}}\propto\boldsymbol{\Phi}\mathbf{X}^T(\mathbf{y}-\mathbf{X}\boldsymbol{\Phi}\boldsymbol{\mu})-\operatorname{diag}\{\mathbf{X}^T\mathbf{X}\}\boldsymbol{\Phi}(\mathbf{I}-\boldsymbol{\Phi})\boldsymbol{\mu}-\frac{1}{v_1}\boldsymbol{\Phi}\boldsymbol{\mu}\overset{let}{=}0 \\
\Longrightarrow & \boldsymbol{\mu}=\left(\boldsymbol{\Phi}\mathbf{X}^T\mathbf{X}\boldsymbol{\Phi}+\boldsymbol{\Delta}+\frac{1}{v_1}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}\mathbf{X}^T\mathbf{y},
\end{aligned}
$$

where $\boldsymbol{\Delta}=\operatorname{diag}\{\mathbf{X}^T\mathbf{X}\}\boldsymbol{\Phi}(\mathbf{I}-\boldsymbol{\Phi})$.

iii. We fix $\{\mu_j\}_{j=1}^p$, $\{\sigma_j^2\}_{j=1}^p$, $\hat{\theta}$, and $\hat{\sigma}^2$, and update $\{\phi_j\}_{j=1}^p$. The objective function is given

by

$$\begin{aligned}\Omega(\phi_1,\ldots,\phi_p) =& -\frac{1}{2\hat{\sigma}^2}\left[\mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\mathbb{E}(\boldsymbol{\beta}) + \mathbb{E}\left(\boldsymbol{\beta}^T(\mathbf{X}^T\mathbf{X})\boldsymbol{\beta}\right)\right] \\ &+ \sum_{j=1}^{p}\phi_j\left(\frac{1}{2}\log\frac{\sigma_j^2}{v_1\hat{\sigma}^2} + \frac{1}{2} - \frac{\mu_j^2+\sigma_j^2}{2v_1\hat{\sigma}^2} + \log(\hat{\theta}) - \log(\phi_j)\right) \\ &+ (1-\phi_j)[\log(1-\hat{\theta}) - \log(1-\phi_j)] + \text{Constant}.\end{aligned}$$

Denote $\mathbf{U} = \text{diag}\{\mu_1,\ldots,\mu_p\}$ and $\boldsymbol{\phi} = (\phi_1,\ldots,\phi_p)^T$, we have $\mathbb{E}(\boldsymbol{\beta}) = \mathbf{U}\boldsymbol{\phi}$ and

$$\begin{aligned}&\mathbb{E}\left(\boldsymbol{\beta}^T(\mathbf{X}^T\mathbf{X})\boldsymbol{\beta}\right) \\ =&\,\text{tr}\left((\mathbf{X}^T\mathbf{X})\,\text{Cov}(\boldsymbol{\beta})\right) + \mathbb{E}(\boldsymbol{\beta})^T(\mathbf{X}^T\mathbf{X})\mathbb{E}(\boldsymbol{\beta}) \\ =&\sum_{j=1}^{p} n(\mu_j^2+\sigma_j^2)\phi_j - n\boldsymbol{\phi}^T\mathbf{U}\mathbf{U}\boldsymbol{\phi} + \boldsymbol{\phi}^T\mathbf{U}(\mathbf{X}^T\mathbf{X})\mathbf{U}\boldsymbol{\phi}.\end{aligned}$$

Then the objective function becomes

$$\begin{aligned}\Omega(\phi_1,\ldots,\phi_p) =& \frac{1}{\hat{\sigma}^2}\mathbf{y}^T\mathbf{X}\mathbf{U}\boldsymbol{\phi} - \frac{n}{2\hat{\sigma}^2}\sum_{j=1}^{p}(\mu_j^2+\sigma_j^2)\phi_j \\ &- \frac{1}{2\hat{\sigma}^2}\boldsymbol{\phi}^T\mathbf{U}(\mathbf{X}^T\mathbf{X} - n\mathbf{I})\mathbf{U}\boldsymbol{\phi} \\ &+ \sum_{j=1}^{p}\phi_j\left(\frac{1}{2}\log\frac{\sigma_j^2}{v_1\hat{\sigma}^2} + \frac{1}{2} - \frac{\mu_j^2+\sigma_j^2}{2v_1\hat{\sigma}^2} + \log(\hat{\theta}) - \log(\phi_j)\right) \\ &+ (1-\phi_j)[\log(1-\hat{\theta}) - \log(1-\phi_j)] + \text{Constant}.\end{aligned}$$

Taking derivative w.r.t. $\boldsymbol{\phi}$, we have

$$\begin{aligned}\nabla\Omega =& \frac{1}{\hat{\sigma}^2}\mathbf{y}^T\mathbf{X}\mathbf{U} - \frac{n}{2\hat{\sigma}^2}\{(\mu_j^2+\sigma_j^2)\}_{j=1}^{p} \\ &- \frac{1}{\hat{\sigma}^2}\mathbf{U}(\mathbf{X}^T\mathbf{X} - n\mathbf{I})\mathbf{U}\boldsymbol{\phi} + Rest.\end{aligned}$$

Direct optimization involves numerical methods due to the none-linear system with

constraints. Thus, we seek approximation approach. Denote $\Delta = \mathbf{U}(\mathbf{X}^T\mathbf{X} - n\mathbf{I}_p\})\mathbf{U}$ and define $g(\boldsymbol{\phi}) = \boldsymbol{\phi}^T\Delta\boldsymbol{\phi}$. At the t-th iteration, using the Taylor expansion, we approximate the quadratic form $g(\boldsymbol{\phi}^{(t)})$ by

$$\begin{aligned} g(\boldsymbol{\phi}^{(t)}) \approx & g(\boldsymbol{\phi}^{(t-1)}) + \nabla(g(\boldsymbol{\phi}^{(t-1)}))^T(\boldsymbol{\phi}^{(t)} - \boldsymbol{\phi}^{(t-1)}) \\ = & (\boldsymbol{\phi}^{(t-1)})^T\Delta\boldsymbol{\phi}^{(t-1)} + 2(\boldsymbol{\phi}^{(t-1)})^T\Delta(\boldsymbol{\phi}^{(t)} - \boldsymbol{\phi}^{(t-1)}) \\ \propto & 2(\boldsymbol{\phi}^{(t-1)})^T\Delta\boldsymbol{\phi}^{(t)}. \end{aligned}$$

Hence, we have

$$\begin{aligned} \Omega \approx & \frac{1}{\hat{\sigma}^2}\mathbf{y}^T\mathbf{X}\mathbf{U}\boldsymbol{\phi} - \frac{n}{2\hat{\sigma}^2}\sum_{j=1}^{p}(\mu_j^2 + \sigma_j^2)\phi_j \\ & - \frac{1}{\hat{\sigma}^2}(\boldsymbol{\phi}^{(t-1)})^T\mathbf{U}(\mathbf{X}^T\mathbf{X} - n\mathbf{I})\mathbf{U}\boldsymbol{\phi} \\ & + \sum_{j=1}^{p}\phi_j\left(\frac{1}{2}\log\frac{\sigma_j^2}{v_1\hat{\sigma}^2} + \frac{1}{2} - \frac{\mu_j^2 + \sigma_j^2}{2v_1\hat{\sigma}^2} + \log(\hat{\theta}) - \log(\phi_j)\right) \\ & + (1 - \phi_j)[\log(1 - \hat{\theta}) - \log(1 - \phi_j)] + \text{Constant}. \end{aligned}$$

Taking partial derivative w.r.t. ϕ_j's respectively, we have

$$\begin{aligned} \frac{\partial\Omega}{\partial\phi_j} \approx & \frac{1}{\hat{\sigma}^2}\mathbf{y}^T\mathbf{X}_j\mu_j - \frac{n}{2\hat{\sigma}^2}(\mu_j^2 + \sigma_j^2) - \frac{1}{\hat{\sigma}^2}\sum_{k\neq j}\mu_j\mu_k\mathbf{X}_k^T\mathbf{X}_j\phi_k^{(t-1)} \\ & + \frac{1}{2}\log\frac{\sigma_j^2}{v_1\hat{\sigma}^2} + \frac{1}{2} - \frac{\mu_j^2 + \sigma_j^2}{2v_1\hat{\sigma}^2} + \log(\hat{\theta}) - \log(\phi_j) - 1 - \log(1 - \hat{\theta}) + \log(1 - \phi_j) + 1. \end{aligned}$$

Setting $\frac{\partial\Omega}{\partial\phi_j} = 0$, we have

$$\text{Logit}(\phi_j) = \text{Logit}(\hat{\theta}) + \frac{1}{2}\log\frac{\sigma_j^2}{v_1\hat{\sigma}^2} - \frac{\mu_j^2}{2\sigma_j^2} + \frac{1}{\hat{\sigma}^2}\mu_j\mathbf{X}_j^T\left(\mathbf{y} - \mathbf{X}_{[,-j]}\boldsymbol{\Phi}_{[-j,-j]}^{(t-1)}\boldsymbol{\mu}_{[-j]}\right), \quad (7)$$

where the last term is equal to $\frac{1}{\hat{\sigma}^2}\mu_j\mathbf{X}_j^T\mu_j(n + 1/v_1)$ according to Algorithm 2 . Therefore, we further approximate it by $\frac{1}{\hat{\sigma}^2}\mu_j^2(\|\mathbf{X}_j\|_2^2 + 1/v_1)$ to reduce computational complexity.

Then we have

$$\begin{aligned}\text{Logit}(\phi_j) &= \text{Logit}(\hat{\theta}) + \frac{1}{2}\log\frac{\sigma_j^2}{v_1\hat{\sigma}^2} + \frac{\mu_j^2}{2\hat{\sigma}^2}\left(n + \frac{1}{v_1}\right) \\ &= \text{Logit}(\hat{\theta}) + \frac{1}{2}\log\frac{\sigma_j^2}{v_1\hat{\sigma}^2} + \frac{\mu_j^2}{2\sigma_j^2}.\end{aligned}$$

Appendix B. Proofs for Variable Selection Consistency

Appendix B.1 Proof for Lemma 2.5.1

In Algorithm 3, we first update $\{\mu_{nj}\}_{j=1}^{p}$, given the initial value $\phi_j = 1$. The updating formula for $\{\mu_{nj}\}_{j=1}^{p}$ at the first iteration is

$$\begin{aligned}\boldsymbol{\mu}_n &= \left(\boldsymbol{\Phi}\mathbf{X}^T\mathbf{X}\boldsymbol{\Phi} + n\boldsymbol{\Phi}(\mathbf{I}_p - \boldsymbol{\Phi}) + \frac{1}{v_1}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}\mathbf{X}^T\mathbf{y}_n \\ &= \left[\mathbf{X}^T\mathbf{X} + \frac{1}{v_1}\mathbf{I}_p\right]^{-1}\mathbf{X}^T(\mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\epsilon}_n),\end{aligned}$$

where $\boldsymbol{\Phi} = \mathbf{I}_p$. After our updating $\boldsymbol{\mu}_n$, given the initial values $\hat{\theta} = 1/2$ and $\hat{\sigma}^2 = 1$, we update the logit of ϕ_j using

$$2\,\mathrm{Logit}(\phi_j) = -\log(a_n v_1 + 1) + \mu_{nj}^2\left(a_n + \frac{1}{v_1}\right).$$

To quantify the magnitude of $\mathrm{Logit}(\phi_j)$, the key is to quantify μ_{nj}. Decompose $\boldsymbol{\mu}_n$ into three parts: the true coefficient vector $\boldsymbol{\beta}^*$, the bias $\mathbf{b}_n$, and the error projection $\mathbf{w}_n$.

$$\begin{aligned}\boldsymbol{\mu}_n =& \boldsymbol{\beta}^* - \left(v_1\mathbf{X}^T\mathbf{X} + \mathbf{I}_p\right)^{-1}\boldsymbol{\beta}^* + \left(\mathbf{X}^T\mathbf{X} + \frac{1}{v_1}\mathbf{I}_p\right)^{-1}\mathbf{X}^T\boldsymbol{\epsilon}_n \\ =& \boldsymbol{\beta}^* - \mathbf{b}_n + \mathbf{w}_n.\end{aligned}$$

Next we prove the following results.

1. **Bound for $\mathbf{b}_n$.** Denote the singular value decomposition of $\mathbf{X}$ as PDQ^T, where r is the rank of $\mathbf{X}$, the dimension of P, D, and Q are $n \times r$, $r \times r$, and $p \times r$, respectively. Condition (C1) implies that the true coefficient vector $\boldsymbol{\beta}^*$ is the projection of the set

$\mathcal{B}$ onto the row space of $\mathbf{X}$, i.e., $\boldsymbol{\beta}^* = QQ^T\boldsymbol{\beta}^*$. Then the bias term can be written as

$$\begin{aligned}\mathbf{b}_n &= \left(v_1\mathbf{X}^T\mathbf{X} + \mathbf{I}_p\right)^{-1}\boldsymbol{\beta}^* \\ &= (v_1 QD^2Q^T + \mathbf{I}_p)^{-1}QQ^T\boldsymbol{\beta}^* \\ &= Q(v_1D^2 + \mathbf{I}_r)^{-1}Q^T\boldsymbol{\beta}^*.\end{aligned}$$

So we can bound the maximal of the bias term b_{nj} by

$$\begin{aligned}\max_j b_{nj}^2 &\leq \|\mathbf{b}_n\|_2^2 = \|Q(v_1D^2 + \mathbf{I}_r)^{-1}Q^T\boldsymbol{\beta}^*\|_2^2 \\ &\leq \left(\frac{1}{v_1\lambda_{n1} + 1}\right)^2 \|\boldsymbol{\beta}^*\|_2^2 = \frac{O(n^{\eta_2})}{(v_1\lambda_{n1})^2}. \qquad (8)\end{aligned}$$

2. **Bound for the variance of w_{nj}.** $\mathbf{w}_n$ is a Gaussian random variable with mean zero and covariance

$$\begin{aligned}\mathrm{Cov}(\mathbf{w}_n) =& \mathrm{Cov}((\mathbf{X}^T\mathbf{X} + \frac{1}{v_1}\mathbf{I}_p)^{-1}\mathbf{X}^T\boldsymbol{\epsilon}_n) \\ =& \sigma^2 Q\,\mathrm{diag}\left(\frac{d_j^2}{(d_j^2 + 1/v_1)^2}\right)Q^T\end{aligned}$$

where d_j's are the elements from the diagonal matrix D. So the variance of each w_{nj} is bounded by the largest eigenvalue of $\mathrm{Cov}(\mathbf{w}_n)$:

$$\mathrm{Var}(w_{nj}) \leq \frac{\sigma^2\lambda_{n1}}{(\lambda_{n1} + 1/v_1)^2} \leq \frac{\sigma^2}{\lambda_{n1}}. \qquad (9)$$

3. **Inequalities for μ_{nj}^2.** Using the inequality $(\sum_{i=1}^n a_i)^2 \leq n\sum_{i=1}^n a_i^2$, we have

$$\begin{aligned}\max_{j\notin S_n^*} \mu_{nj}^2 &\leq 2(\max_j b_{nj}^2 + \max_j w_{nj}^2), \\ \min_{j\in S_n^*} \mu_{nj}^2 &\geq \frac{1}{3}\min_{j\in S_n^*}(\beta_j^*)^2 - \max_j b_{nj}^2 - \max_j w_{nj}^2.\end{aligned}$$

First we show (2.24). Note

$$\max_{j\notin S_n^*} 2\,\mathrm{Logit}(\phi_j) \le -\log(a_n v_1+1) + 2\Big(a_n+\frac{1}{v_1}\Big)\Big(\max_j b_{nj}^2 + \max_j w_{nj}^2\Big).$$

We have

(a) $\log(v_1 a_n+1)\to\infty$ since $v_1 \succ n^{-a}$.

(b) By (8), $(a_n+\frac{1}{v_1})\max_j b_{nj}^2 = O(n^{\eta_2+a-2\eta_1}/v_1^2)\to 0$ since $v_1 \succ n^{-\frac{1}{2}(2\eta_1-a-\eta_2)}$.

(c) For $(a_n+1/v_1)\max_j w_{nj}^2$, the variance is upper bounded by $O(n^{a-\eta_1})$. Hence, for any constant c,

$$\begin{aligned} &P\left((a_n+1/v_1)\max_j w_{nj}^2 > c\right)\\ \le&\sum_{j=1}^{p} P\left((a_n+1/v_1)w_{nj}^2 > c\right)\\ \le&\frac{p}{\sqrt{2\pi n^{\eta_1-a}}}\exp\left\{-\frac{c}{2}n^{\eta_1-a}\right\}\to 0, \end{aligned}$$

since $\log p = o(n^{\eta_1-a})$.

Thus, $\max_{j\notin S_n^*} 2\,\mathrm{Logit}(\phi_j)\xrightarrow{P}-\infty$ as $n\to\infty$, and therefore (2.24) holds true.

Next we show (2.25). Note

$$\min_{j\in S_n^*} 2\,\mathrm{Logit}(\phi_{nj}) \ge -\log(a_n v_1+1) + \Big(a_n+\frac{1}{v_1}\Big)\Big(\frac{1}{3}\min_{j\in S_n^*}(\beta_j^*)^2 - \max_j b_{nj}^2 - \max_j w_{nj}^2\Big).$$

Since $\left(a_n+\frac{1}{v_1}\right)\frac{1}{3}\min_{j\in S_n^*}(\beta_j^*)^2 \succeq n^{a-(1-\eta_3)} \succ \log(a_n v_1+1)$ due to the condition that $v_1 \prec \exp(n^{a-(1-\eta_3)})$, it is the leading term. So $\min_{j\in S_n^*} 2\,\mathrm{Logit}(\phi_{nj})\xrightarrow{P}\infty$ as $n\to\infty$ and therefore (2.25) holds true.

Appendix B.2 Proof for Theorem 2.5.2

With Lemma 2.5.1, it is easy to show that when the sample size is large enough, our algorithm will stop with one update. For any threshold value c in (2.14), we can set C in Lemma 2.5.1 to be bigger than $2\log(\frac{1}{c}-1)$. Then with probability going to 1, after the first iteration, we will have $\max_{j\notin S_n^*}\phi_j < c$, and $\min_{j\in S_n^*}\phi_{nj} > 1-c$ and the algorithm will halt. Therefore, $\hat{S}_n = S_n^*$ with probability 1 when $n\to\infty$. Hence, the frequentist selection consistency follows.

Appendix B.3 Proof for Theorem 2.5.3

Let $C_n = s\log(v_1 a_n)$, where $s \in (0,1)$. Following the same argument used in the proof of Lemma 2.5.1, we can show that

$$P\left(2\max_{j\notin S_n^*}\text{Logit}(\phi_j) > -C_n\right) \to 0 \quad \text{and} \quad P\left(2\min_{j\in S_n^*}\text{Logit}(\phi_j) < C_n\right) \to 0,$$

if $\log p = o(n^{\eta_1 - a})$. Therefore, with probability going to 1, we have

$$\begin{aligned}\max_{j\notin S_n^*}\log\frac{\phi_j}{1-\phi_j} < -\frac{1}{2}C_n \Rightarrow \max_{j\notin S_n^*}\phi_j < \frac{1}{\exp\{\frac{1}{2}C_n\}},\\ \min_{j\in S_n^*}\log\frac{\phi_j}{1-\phi_j} > \frac{1}{2}C_n \Rightarrow \max_{j\in S_n^*}(1-\phi_j) < \frac{1}{\exp\{\frac{1}{2}C_n\}}.\end{aligned}$$

Denote the true model by $\boldsymbol{\gamma}^*$. Using the inequality $\prod_j(1-p_j) \geq 1 - \sum_j p_j$, we have

$$1 - q(\boldsymbol{\gamma}^*) \leq \sum_{j\in S_n^*}(1-\phi_j) + \sum_{j\notin S_n^*}\phi_j \leq p \times \left[\max_{j\in S_n^*}(1-\phi_j) \bigvee \max_{j\notin S_n^*}\phi_j\right] = \frac{p}{(v_1 a_n)^{s/2}}.$$

If $p \prec (v_1 a_n)^{s/2}$, we have

$$1 - q(\boldsymbol{\gamma}^*) \leq \frac{p}{e^{\frac{1}{2}C_n}} = \frac{p}{(v_1 a_n)^{s/2}} \xrightarrow{P} 0. \tag{10}$$

As $v_1 \prec \exp(n^{a-(1-\eta_3)})$, $p \prec (v_1 a_n)^{s/2} \prec \exp(\sqrt{n}^{a-(1-\eta_3)})n^{s/2}$. This implies that if $p = o(\exp(n^{\frac{a}{2}-\frac{1-\eta_3}{2}}))$, we can achieve Bayesian consistency by letting v_1 going to infinity in exponential order.

Appendix C. Derivation of the Updating Equations of Logistic Regression

Given the data and the prior, the posterior is

$$
\begin{aligned}
&\pi(\boldsymbol{\beta}, \boldsymbol{\gamma}, \alpha, \theta | \mathbf{y}) \\
\propto & p(y|\alpha, \beta)\pi(\alpha)\pi(\boldsymbol{\beta}|\boldsymbol{\gamma})\pi(\boldsymbol{\gamma}|\theta)\pi(\theta) \\
= & \prod_{i=1}^{n} \frac{(e^{\alpha+\mathbf{x}_i^T\boldsymbol{\beta}})^{y_i}}{(1+e^{\alpha+\mathbf{x}_i^T\boldsymbol{\beta}})^{m_i}} \pi(\alpha)\pi(\boldsymbol{\beta}|\boldsymbol{\gamma})\pi(\boldsymbol{\gamma}|\theta)\pi(\theta) \\
\propto & \prod_{i=1}^{n} \int_0^{\infty} \exp\left\{ \left(y_i - \frac{1}{2} m_i \right) (\alpha + \mathbf{x}_i^T\boldsymbol{\beta}) - \frac{1}{2} w_i (\alpha + \mathbf{x}_i^T\boldsymbol{\beta})^2 \right\} f(w_i|m_i, 0)\, \mathrm{d}w_i \\
& \times \pi(\alpha)\pi(\boldsymbol{\beta}|\boldsymbol{\gamma})\pi(\boldsymbol{\gamma}|\theta)\pi(\theta),
\end{aligned}
$$

The last step is from the Pólya-Gamma data-augmentation, i.e., we treat $w_i, \ldots, w_n$ as latent variables. Hence, the full posterior is

$$
\begin{aligned}
&\pi(\boldsymbol{\beta}, \boldsymbol{\gamma}, \alpha, \theta, \mathbf{w} | \mathbf{y}) \\
\propto & \prod_{i=1}^{n} \left[\exp\left\{ \left(y_i - \frac{1}{2} m_i \right) (\alpha + \mathbf{x}_i^T\boldsymbol{\beta}) - \frac{1}{2} w_i (\alpha + \mathbf{x}_i^T\boldsymbol{\beta})^2 \right\} f(w_i|m_i, 0) \right] \pi(\alpha)\pi(\boldsymbol{\beta}|\boldsymbol{\gamma})\pi(\boldsymbol{\gamma}|\theta)\pi(\theta) \\
\propto & \prod_{i=1}^{n} \left[\exp\left\{ \left(y_i - \frac{1}{2} \right) (\alpha + \mathbf{x}_i^T\boldsymbol{\beta}) - \frac{1}{2} w_i (\alpha + \mathbf{x}_i^T\boldsymbol{\beta})^2 \right\} f(w_i|m_i, 0) \right] \\
& \times \prod_{j=1}^{p} \left(\frac{1}{\sqrt{2\pi v_1}} \exp\left\{ -\frac{\beta_j^2}{2v_1} \right\} \right)^{\gamma_j} (\delta_0(\beta_j))^{1-\gamma_j} \\
& \times \prod_{j=1}^{p} \theta^{\gamma_j} (1-\theta)^{1-\gamma_j} \\
& \times \theta^{a_0-1} (1-\theta)^{b_0-1}.
\end{aligned}
$$

We maximize the following objective function:

$$\Omega(q_1, q_2, \alpha, \theta) = \mathbb{E}^{q_1,q_2}_{\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{w}} \left[\log \frac{\pi(\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}, \theta, \mathbf{w}|\mathbf{y})}{q_1(\boldsymbol{\beta}, \boldsymbol{\gamma}) q_2(\mathbf{w})} \right],$$

where $q_1(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \prod_{j=1}^{p} \{q_{1,j}(\beta_j, \gamma_j)\} = \prod_{j=1}^{p} \{[\phi_j f_j(\beta_j)]^{\gamma_j} [(1-\phi_j)\delta_0(\beta_j)]^{1-\gamma_j}\}$.

First of all, we find the approximating distribution $q_{1,l}(\beta_l, \gamma_l)$ of β_l and γ_l for some $l = 1, \ldots, p$. To achieve this, we look at the objective function when all the other approximating

distributions and point estimates are fixed:

$$
\begin{aligned}
&\Omega(q_1, q_2, \hat{\alpha}, \hat{\theta}) \\
=&\mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}} \mathbb{E}^{q_2}_{\mathbf{w}} \log \left\{ \frac{p(\mathbf{y}|\hat{\alpha}, \boldsymbol{\beta}, \mathbf{w}) \pi(\boldsymbol{\beta}|\boldsymbol{\gamma}) \pi(\boldsymbol{\gamma}|\hat{\theta})}{q_{1,l}(\beta_l, \gamma_l)} \right\} + Const \\
=&\mathbb{E}^{q_{1,l}}_{[\beta_l,\gamma_l]} \mathbb{E}^{q_{1,-l}}_{-[\beta_l,\gamma_l]} \left\{ \sum_{i=1}^{n} \left[\left(y_i - \frac{1}{2} m_i \right) (\hat{\alpha} + x_i^T \boldsymbol{\beta}) - w_i \frac{(\hat{\alpha} + x_i^T \boldsymbol{\beta})^2}{2} \right] \right. \\
&+ \sum_{j=1}^{p} \left[\gamma_j \left(-\frac{1}{2} \log(v_1) - \frac{\beta_j^2}{2v_1} + \log(\hat{\theta}) \right) + (1 - \gamma_j) \left(\log \delta_0(\beta_j) + \log(1 - \hat{\theta}) \right) \right] \\
&\left. - \left(\gamma_l \log[\phi_l f_l(\beta_l)] + (1 - \gamma_l) \log[(1 - \phi_l) \delta(\beta_l)] \right) \right\} + Const \\
=&\mathbb{E}^{q_{1,l}}_{[\beta_l,\gamma_l]} \left[\sum_{i=1}^{n} \left\{ \left[y_i - \frac{1}{2} - \hat{\alpha} \mathbb{E}^{q_2}_{\mathbf{w}}(w_i) - \mathbb{E}^{q_2}_{\mathbf{w}}(w_i) \sum_{k \neq j}^{p} \left(x_{ik} \mathbb{E}^{q_{1,k}}_{[\beta_k,\gamma_k]}(\beta_k) \right) \right] x_{il} \beta_l - \frac{1}{2} \mathbb{E}^{q_2}_{\mathbf{w}}(w_i) x_{il}^2 \beta_l^2 \right\} \right. \\
&+ \gamma_l \left(-\frac{1}{2} \log(v_1) - \frac{\beta_l^2}{2v_1} + \log(\hat{\theta}) \right) + (1 - \gamma_l) \log(1 - \hat{\theta}) \\
&\left. - \left(\gamma_l \log[\phi_l f_l(\beta_l)] + (1 - \gamma_l) \log(1 - \phi_l) \right) \right] + Const \\
=&\mathbb{E}^{q_{1,l}}_{[\beta_l,\gamma_l]} \left[\sum_{i=1}^{n} \left\{ \left[y_i - \frac{1}{2} m_i - \hat{\alpha} \mathbb{E}^{q_2}_{\mathbf{w}}(w_i) - \mathbb{E}^{q_2}_{\mathbf{w}}(w_i) \sum_{k \neq j}^{p} (x_{ik} \phi_k \mu_k) \right] x_{il} \beta_l - \frac{1}{2} \mathbb{E}^{q_2}_{\mathbf{w}}(w_i) x_{il}^2 \beta_l^2 \right\} \right. \\
&+ \gamma_l \left(-\frac{1}{2} \log(v_1) - \frac{\beta_l^2}{2v_1} + \log(\hat{\theta}) \right) + (1 - \gamma_l) \left(\frac{1}{2} \log(2\pi) + \log \frac{1 - \hat{\theta}}{1 - \phi_l} \right) \\
&\left. - \left(\gamma_l \log[\phi_l f_l(\beta_l)] \right) \right] + Const \\
=&\phi_l \mathbb{E}^{f_l}_{\beta_l|\gamma_l=1} \left[\log \frac{\exp\{a\beta_l - \frac{b}{2}\beta_l^2\}}{\phi_l f_l(\beta_l)} - \frac{1}{2} \log(v_1) + \log(\hat{\theta}) \right] \\
&+ (1 - \phi_l) \log \frac{1 - \hat{\theta}}{1 - \phi_l} + Const
\end{aligned}
$$

where

$$
\begin{aligned}
\mu_j =& \mathbb{E}_{\beta_j|\gamma_j=1}(\beta_j) \\
\phi_j =& \mathbb{E}^{q_{2j}}(\gamma_j = 1) \\
a =& \sum_{i=1}^{n} \left[y_i - \frac{1}{2} m_i - \hat{\alpha} \mathbb{E}_{\mathbf{w}}^{q_2}(w_i) - \mathbb{E}_{\mathbf{w}}^{q_2}(w_i) \sum_{k \neq j}^{p} (x_{ik} \phi_k \mu_k) \right] x_{il} \\
b =& \sum_{i=1}^{n} \mathbb{E}_{\mathbf{w}}^{q_2}(w_i) x_{il}^2 + \frac{1}{v_1}.
\end{aligned}
$$

Then to maximize the objective function, we need $f_l(\beta_l) \propto \exp\{a\beta_l - \frac{b}{2}\beta_l^2\}$, which implies that $f_l(\beta_l) = \mathrm{N}(\beta_l|\mu_l, \sigma_l^2)$, with

$$
\begin{aligned}
\mu_l =& \frac{a}{b} = \frac{\sum_{i=1}^{n} \left[y_i - \frac{1}{2} m_i - \hat{\alpha} \mathbb{E}_{\mathbf{w}}^{q_2}(w_i) - \frac{1}{2} \mathbb{E}_{\mathbf{w}}^{q_2}(w_i) \sum_{k \neq j}^{p} (x_{ik} \phi_k \mu_k) \right] x_{il}}{\sum_{i=1}^{n} \mathbb{E}_{\mathbf{w}}^{q_2}(w_i) x_{il}^2 + \frac{1}{v_1}} \\
\sigma_l^2 =& \frac{1}{b} = \frac{1}{\sum_{i=1}^{n} \mathbb{E}_{\mathbf{w}}^{q_2}(w_i) x_{il}^2 + \frac{1}{v_1}}.
\end{aligned}
$$

Plug $f_l(\beta_l) = \mathrm{N}(\beta_l|\mu_l, \sigma_l^2)$ in the objective function, we have:

$$
\begin{aligned}
\Omega(q_1, q_2, \hat{\alpha}, \hat{\theta}) =& Const - \phi_l \left[\log \left(\phi_l \sqrt{\frac{b}{2\pi}} \exp\left\{ -\frac{a^2}{2b} \right\} \right) - \frac{1}{2} \log(v_1) + \log(\hat{\theta}) \right] \\
& + (1 - \phi_l) \left[\frac{1}{2} \log(2\pi) - \log \frac{1 - \phi_l}{1 - \hat{\theta}} \right] \\
=& \phi_l \left(-\log \phi_l + \frac{1}{2} \log \sigma_l^2 + \frac{1}{2} \log(2\pi) + \frac{\mu_l^2}{2\sigma_l^2} - \frac{1}{2} \log(v_1) + \log(\hat{\theta}) \right) \\
& + (1 - \phi_l) \left[\frac{1}{2} \log(2\pi) - \log \frac{1 - \phi_l}{1 - \hat{\theta}} \right]
\end{aligned}
$$

Then by taking partial derivative w.r.t. ϕ_l, we get

$$\begin{aligned}\frac{\partial\Omega(q_1,q_2,\hat{\alpha},\hat{\theta})}{\partial\phi_l} =& -\log\phi_l+\frac{1}{2}\log\sigma_l^2+\frac{1}{2}\log(2\pi)+\frac{\mu_l^2}{2\sigma_l^2}-\frac{1}{2}\log(v_1)+\log(\hat{\theta})-1\\ &-\frac{1}{2}\log(2\pi)+\log(1-\phi_l)-\log(1-\hat{\theta})+1\\ \overset{set}{=}&0\\ \Longrightarrow\quad \log\frac{\phi_l}{1-\phi_l}=&\log\frac{\hat{\theta}_l}{1-\hat{\theta}_l}-\frac{1}{2}\log\frac{v_1}{\sigma_l^2}+\frac{\mu_l^2}{2\sigma_l^2}.\end{aligned}$$

Secondly, we will update $q_2(\mathbf{w})$ as follows:

$$\begin{aligned}\log q_2(\mathbf{w}) \propto&\mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}}\left\{\log[\pi(\hat{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\hat{\theta},\mathbf{w}|\mathbf{y})]\right\}\\ =&\mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}}\log\left\{\prod_{i=1}^{n}\left[\exp\left\{\left(y_i-\frac{m_i}{2}\right)(\hat{\alpha}+\mathbf{x}_i^T\boldsymbol{\beta})-\frac{w_i}{2}(\hat{\alpha}+\mathbf{x}_i^T\boldsymbol{\beta})^2\right\}g(w_i|m_i,0)\right]\right\}\\ &+Const\\ =&\sum_{i=1}^{n}\left[-\frac{\mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}}(\hat{\alpha}+\mathbf{x}_i^T\boldsymbol{\beta})^2}{2}w_i+\log g(w_i|1,0)\right]\\ &+Const\\ =&\sum_{i=1}^{n}\log\left[\exp\left\{-\frac{(\hat{\alpha}^2)+2\hat{\alpha}\mathbf{x}_i^T\mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}}(\boldsymbol{\beta})+\mathbf{x}_i^T\mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}}(\boldsymbol{\beta}\boldsymbol{\beta}^T)\mathbf{x}_i}{2}w_i\right\}g(w_i|m_i,0)\right]\\ &+Const\\ =&\sum_{i=1}^{n}\log g\left(w_i\middle|m_i,\sqrt{\hat{\alpha}^2+2\hat{\alpha}\mathbf{x}_i^T\mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}}(\boldsymbol{\beta})+\mathbf{x}_i^T\mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}}(\boldsymbol{\beta}\boldsymbol{\beta}^T)\mathbf{x}_i}\right)+Const\end{aligned}$$

Hence, the approximating distribution of w_i is $PG\left(m_i,\sqrt{\hat{\alpha}^2+2\hat{\alpha}\mathbf{x}_i^T\mathbb{E}^{q_1}_{\boldsymbol{\beta}}(\boldsymbol{\beta})+\mathbf{x}_i^T\mathbb{E}^{q_1}_{\boldsymbol{\beta}}(\boldsymbol{\beta}\boldsymbol{\beta}^T)\mathbf{x}_i}\right)$,

where

$$\begin{aligned}
\mathbb{E}^{q_1}_{\boldsymbol{\beta}}(\boldsymbol{\beta}) &= (\phi_1\mu_1, \ldots, \phi_n\mu_n)^T, \\
\mathbb{E}^{q_1}_{\boldsymbol{\beta}}(\boldsymbol{\beta}\boldsymbol{\beta}^T) &= \mathrm{Cov}^{q_1}_{\boldsymbol{\beta}}(\boldsymbol{\beta}) + \mathbb{E}^{q_1}_{\boldsymbol{\beta}}(\boldsymbol{\beta})\mathbb{E}^{q_1}_{\boldsymbol{\beta}}(\boldsymbol{\beta}^T) \\
&= \mathrm{diag}\{\mu_j^2\phi_j(1-\phi_j) + \phi_j\sigma_j^2 + v_0(1-\phi_j)\}_{j=1}^p + (\phi_1\mu_1, \ldots, \phi_p\mu_p)^T(\phi_1\mu_1, \ldots, \phi_p\mu_p).
\end{aligned}$$

Then, we compute $\hat{\alpha}$ given other approximating distributions and $\hat{\theta}$ being fixed. That is

$$\begin{aligned}
\hat{\alpha} &= \arg\max_{\alpha} \mathbb{E}^{q_1,q_2}_{\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{w}} \left[\log \frac{\pi(\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}, \hat{\theta}, \mathbf{w}|\mathbf{y})}{q_1(\boldsymbol{\beta}, \boldsymbol{\gamma})q_2(\mathbf{w})}\right] \\
&= \arg\max_{\alpha} \mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}}\mathbb{E}^{q_2}_{\mathbf{w}} \log\left[p(\mathbf{y}|\alpha, \boldsymbol{\beta}, \mathbf{w})\pi(\alpha)\right] \\
&= \arg\max_{\alpha} \mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}}\mathbb{E}^{q_2}_{\mathbf{w}} \sum_{i=1}^n \left[\left(y_i - \frac{1}{2}m_i\right)(\alpha + \mathbf{x}_i^T\boldsymbol{\beta}) - w_i\frac{(\alpha + x_i^T\boldsymbol{\beta})^2}{2}\right] \\
&= \arg\max_{\alpha} \sum_{i=1}^n \left[\left(y_i - \frac{1}{2}m_i\right)\alpha - \mathbb{E}^{q_1,q_2}_{\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{w}} w_i\frac{\alpha^2 + 2\alpha\mathbf{x}_i^T\boldsymbol{\beta}}{2}\right] \\
&= \arg\max_{\alpha} -\frac{1}{2}\sum_{i=1}^n \mathbb{E}^{q_2}_{\mathbf{w}}(w_i)\alpha^2 + \alpha\sum_{i=1}^n \left[\left(y_i - \frac{1}{2}m_i\right) - \mathbf{x}_i^T\mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}}(\boldsymbol{\beta})\right] \\
&= \frac{\sum_{i=1}^n \left[\left(y_i - \frac{1}{2}m_i\right) - \mathbf{x}_i^T\mathbb{E}^{q_1}_{\boldsymbol{\beta}}(\boldsymbol{\beta})\right]}{\sum_{i=1}^n \mathbb{E}^{q_3}_{\mathbf{w}}(w_i)} \\
&= \frac{\sum_{i=1}^n \left[\left(y_i - \frac{1}{2}m_i\right) - \mathbf{x}_i^T(\phi_1\mu_1, \ldots, \phi_p\mu_p)^T\right]}{\sum_{i=1}^n \mathbb{E}^{q_3}_{\mathbf{w}}(w_i)}.
\end{aligned}$$

Finally, we consider the point estimate of θ given all the approximating distributions and

$\hat{\alpha}$ being fixed:

$$
\begin{aligned}
\hat{\theta} &= \arg\max_{\theta} \mathbb{E}^{q_1,q_2}_{\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{w}} \left[\log \frac{\pi(\hat{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \theta, \mathbf{w}|\mathbf{y})}{q_1(\boldsymbol{\beta}, \boldsymbol{\gamma}) q_2(\mathbf{w})} \right] \\
&= \arg\max_{\theta} \mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}} \left[\pi(\boldsymbol{\gamma}|\theta) \pi(\theta) \right] \\
&= \arg\max_{\theta} \sum_{j=1}^{p} \left[\mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}}(\gamma_j) \log(\theta) + \mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}}(1-\gamma_j)(1-\log(\theta)) \right] + (a_0 - 1)\log(\theta) \\
&\quad + (b_0 - 1)\log(1-\theta) \\
&= \arg\max_{\theta} \left[\sum_{j=1}^{p} \mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}}(\gamma_j) + a_0 - 1 \right] \log(\theta) + \left[\sum_{j=1}^{p} \mathbb{E}^{q_1}_{\boldsymbol{\beta},\boldsymbol{\gamma}}(1-\gamma_j) + b_0 - 1 \right] \log(1-\theta) \\
&= \arg\max_{\theta} \left[\sum_{j=1}^{p} \phi_j + a_0 - 1 \right] \log(\theta) + \left[p - \sum_{j=1}^{p} \phi_j + b_0 - 1 \right] \log(1-\theta) \\
&= \frac{\sum_{j=1}^{p} \phi_j + a_0 - 1}{p + a_0 + b_0 - 2}.
\end{aligned}
$$

Appendix D. Pólya-Gamma Data-Augmentation

In this section, we describe the Pólya-Gamma distribution, some of its important properties, and how to implement data-augmentation trick for logistic regression by Polson et al. (2013).

A random variable X has a Pólya-Gamma distribution with $b > 0$ and $c \in \mathcal{R}$, if

$$X \stackrel{D}{=} \frac{1}{2\pi^2} \sum_{k=1}^{\infty} \frac{g_k}{(k-1/2)^2 + c^2/(4\pi^2)},$$

where g_k's are independent random variables that follow Gamma$(b, 1)$, and $\stackrel{D}{=}$ means that they are equal in distribution.

The followings are some properties of the Pólya-Gamma distribution.

- If $W \sim PG(b, c)$, then its probability density function has the following property:

$$g(w|b, c) \propto \exp\left\{-\frac{c^2}{2}w\right\} g(w|b, 0), \tag{11}$$

  where $g(w|b, c)$ is the density function of $PG(b, c)$.

- If $W \sim PG(b, c)$, its expectation is

$$\mathbb{E}(W) = \frac{b}{2c} \tanh\left(\frac{c}{2}\right). \tag{12}$$

The following identity is showed in Theorem 1 of Polson et al. (2013)

$$\frac{(e^{\psi})^a}{(1+e^{\psi})^b} = 2^{-b} e^{\kappa\psi} \int_0^{\infty} e^{-w\psi^2/2} g(w|b, 0)\, \mathrm{d}w, \tag{13}$$

for $a \in \mathcal{R}$, and $b > 0$, where $w \sim PG(b, 0)$ and $\kappa = a - \frac{b}{2}$. Equation (13) naturally gives rise to a data-augmentation for the logistic regression. In our model, by incorporating the

Pólya-Gamma latent variable to (3.1), we obtain the following complete-data likelihood

$$p(\mathbf{y}, \mathbf{w}|\alpha, \boldsymbol{\beta}) \propto \prod_{i=1}^{n} \exp\left\{\left(y_i - \frac{m_i}{2}\right)(\alpha + \mathbf{x}_i^T\boldsymbol{\beta}) - \frac{w_i}{2}(\alpha + \mathbf{x}_i^T\boldsymbol{\beta})^2\right\} g(w|m_i, 0),$$

This complete-data likelihood of $\boldsymbol{\beta}$ is a density of normal, and hence is conjugate with the priors.