Natural Gradient Variational Bayes Without Fisher Matrix Analytic Calculation and Its Inversion

A. Godichon-Baggioni, D. Nguyen & M.-N. Tran

JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION
2024, VOL. 00, NO. 0, 1–12: Theory and Methods
https://doi.org/10.1080/01621459.2024.2392904

Natural Gradient Variational Bayes Without Fisher Matrix Analytic Calculation and Its Inversion

A. Godichon-Baggioni[a], D. Nguyen[b], and M.-N. Tran[c]

[a]Laboratoire de Probabilités, Statistique et Modélisation, Sorbonne-Université, Paris, France; [b]Department of Mathematics, Marist College, Poughkeepsie, NY; [c]Business Analytics Discipline, The University of Sydney Business School, Sydney, NSW, Australia

ABSTRACT

This article introduces a method for efficiently approximating the inverse of the Fisher information matrix, a crucial step in achieving effective variational Bayes inference. A notable aspect of our approach is the avoidance of analytically computing the Fisher information matrix and its explicit inversion. Instead, we introduce an iterative procedure for generating a sequence of matrices that converge to the inverse of Fisher information. The natural gradient variational Bayes algorithm without analytic expression of the Fisher matrix and its inversion is provably convergent and achieves a convergence rate of order $\mathcal{O}(\log s/s)$, with s the number of iterations. We also obtain a central limit theorem for the iterates. Implementation of our method does not require storage of large matrices, and achieves a linear complexity in the number of variational parameters. Our algorithm exhibits versatility, making it applicable across a diverse array of variational Bayes domains, including Gaussian approximation and normalizing flow Variational Bayes. We offer a range of numerical examples to demonstrate the efficiency and reliability of the proposed variational Bayes method. Supplementary materials for this article are available online, including a standardized description of the materials available for reproducing the work.

ARTICLE HISTORY
Received December 2023
Accepted June 2024

KEYWORDS
Bayesian computation; Bayesian neural network; Normalizing flow; Stochastic gradient descent

1. Introduction

The growing complexity of models used in modern statistics and machine learning has spurred the demand for more efficient Bayesian estimation techniques. Among the array of Bayesian tools available, Variational Bayes (Waterhouse, MacKay, and Robinson 1995; Jordan et al. 1999) has gained prominence as a remarkably versatile alternative to traditional Monte Carlo methods for tackling statistical inference in intricate models. Variational Bayes (VB) operates by approximating the posterior probability distribution using a member selected from a family of tractable distributions, characterized by variational parameters. The optimal member is determined through minimization of the Kullback-Leibler divergence, which quantifies the disparity between the chosen candidate and the posterior distribution. VB is a fast alternative to Markov chain Monte Carlo (MCMC) methods, and has found diverse applications, encompassing variational autoencoders (Kingma and Welling 2013), text analysis (Hoffman et al. 2013), Bayesian synthetic likelihood (Ong et al. 2018), deep neural networks (Graves 2011; Tran et al. 2020), to name a few. For recent advances in the field of VB and Bayesian approximation in general, please refer to the excellent survey papers of Blei, Kucukelbir, and McAuliffe (2017) and Martin, Frazier, and Robert (2023).

VB turns the Baysesian inference problem into an optimization problem, and a large class of VB methods use stochastic gradient descent (SGD) as their backbone. In the recent decades, a great deal of effort has been devoted to developing and improving optimization algorithms for big and high dimensional data. As a result, various first order stochastic optimization algorithms have been developed in response to these new demands; notable examples include AdaGrad of Duchi, Hazan, and Singer (2011), Adam of Kingma and Ba (2014), Adadelta of Zeiler (2012), and their variance reduction variations (Johnson and Zhang 2013; Defazio, Bach, and Lacoste-Julien 2014; Nguyen et al. 2017). For a detailed discussion on stochastic optimization, please refer to the excellent books of Kushner and Yin (2003), Goodfellow, Bengio, and Courville (2016), and Murphy (2012).

Gradient descent methods in VB rely on the gradient of the objective lower bound function, whose definition depends upon the metric on the variational parameter space. Optimization in conventional VB methods uses the Euclidean gradient defined using the usual Euclidean metric. It turns out that the natural gradient, the term coined by Amari (1998), represents a more adequate direction of ascent in the VB context as it takes into account the information geometry of the variational family (Martens 2020; Khan and Lin 2017). The natural gradient is defined using the Fisher-Rao metric, which resembles the Kullback-Leibler divergence between probability distributions parameterized by the variational parameters. More precisely, the natural gradient is the steepest ascent direction of the objective function on the variational parameter space equipped with the Fisher-Rao metric. Martens (2020) sheds light on the concept

CONTACT M.-N. Tran minh-ngoc.tran@sydney.edu.au Business Analytics Discipline, The University of Sydney Business School, NSW 2006, Australia.
Supplementary materials for this article are available online. Please go to www.tandfonline.com/r/JASA.

that natural gradient descent can be viewed as a second-order optimization method where the Fisher information assumes the role of the Hessian matrix. Because of this, natural gradients take into account the curvature information (through the Fisher-Rao metric) of the variational parameter space; therefore the number of iteration steps required to find a local optimum is often found significantly reduced (Tran, Nott, and Kohn 2017). According to Tan (2021), stochastic optimization guided by natural gradients has proven more resilient, capable of circumventing or escaping plateaus, ultimately resulting in faster convergence; see also (Rattray, Saad, and Amari 1998; Hoffman et al. 2013; Khan and Lin 2017; Wilkinson, Särkkä, and Solin 2023).

The natural gradient is calculated by pre-multiplying the Euclidean gradient of the lower bound function with the inverse Fisher information matrix, a process that is notably intricate. Computing the Fisher matrix, not to mention its inverse, is challenging. In the realm of Gaussian variational approximation, where the posterior is approximated by a Gaussian distribution, the natural gradient can be calculated efficiently. Tran et al. (2020) consider a factor structure for the covariance matrix, and derive a closed-form approximation for the natural gradient. Tan (2021) employs a Cholesky factor structure for the covariance matrix and the precision matrix, and derives an analytic natural gradient; see also Khan and Lin (2017) and Magris, Shabani, and Iosifidis (2022). On the other hand, for broader cases where the variational distribution is based on neural networks, Martens and Grosse (2015) approximate the Fisher matrix with a block diagonal matrix. It is important to highlight that existing techniques for computing the natural gradient are primarily restricted to certain contexts (like the Gaussian variational approximations mentioned above) or are heavily dependent on simplified approximations (such as employing a block-diagonal matrix). These constraints restrict the broader application of natural gradients. For a large class of VB methods, for example, when the variational distribution is a mixture (Giordani et al. 2013), a copula (Gunawan, Kohn, and Nott 2023) or a normalizing flow (Rezende and Mohamed 2015), it is challenging to use the natural gradient as the Fisher matrix is not available.

This article makes several important contributions that significantly improve the natural gradient VB method. First, we present an approach for efficiently approximating the *inverse* of Fisher information matrix. We emphasize that there are two main difficulties in calculating the natural gradient: (i) analytical calculation of the Fisher matrix that often involves intractable expectations, and (ii) computing its matrix inversion. A notable aspect of our approach is the avoidance of these two difficulties altogether. Instead, we introduce an iterative procedure for generating a sequence of positive definite matrices that converge to the inverse of Fisher information. Pre-multiplying the Euclidean gradient with these matrices provides estimates of the natural gradient. Our method of approximating the natural gradient is general, easy to implement, asymptotically exact and applies to any variational distribution including Gaussian distributions, mixtures and normalizing flow based distributions. It is important to note that, for high-dimensional applications, implementation of our method does not require storage of large matrices because the estimate of inverse Fisher matrix can be written using outer products. Second, we propose a VB method that streamlines the natural gradient estimation without matrix inversion within the VB training iteration. This leads to an efficient natural gradient VB algorithm, referred to as inversion-free variational Bayes (IFVB). We also present a weighted averaged estimate version of IFVB, called AIFVB, that converges faster than IFVB. Both IFVB and AIFVB are provably convergent, with AIFVB being shown asymptotically efficient and achieving a central limit theorem. Third, to substantiate the effectiveness and robustness of our proposed method, we offer a range of numerical examples to demonstrate its efficiency and reliability.

The rest of the article is organized as follows: Section 2 provides a brief overview of the variational Bayesian inference problem. Section 3 presents natural gradient and discusses its advantages as well as its computational difficulty. We introduce inversion free natural variational Bayes in Section 4. Section 5 is concerned with convergence analysis. Numerical examples are provided in Section 6. Section 7 concludes the article. The supplementary material contains the proofs of the main theorems and an Appendix with further technical details.

Notation. We denote by $\|x\| = (x_1^2 + \cdots + x_d^2)^{1/2}$ the ℓ^2-norm of the vector $x = (x_1, \ldots, x_d)^\top \in \mathbb{R}^d$. For a function f on $\mathbb{R}^d$, $\nabla_x f = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_d})^\top$ denotes the gradient vector, and $\nabla_x^2 f = (\frac{\partial^2 f}{\partial x_i \partial x_j})_{i,j=1,\ldots,d}$ is the Hessian. $\|A\|_{op} = \max_{\|x\| \le 1} \|Ax\|$ denotes the operator norm of a matrix A; $\lambda_{\min}(A), \lambda_{\max}(A)$ denote the minimum eigenvalue and maximum eigenvalue of matrix A, respectively. $\mathbb{I}_d$ denotes a $d \times d$ identity matrix. $\mathbb{E}_f(g(X)) = \int g(x)f(x)dx$ with $X \sim f$. We write $a = \mathcal{O}(b)$ to denote $a \le Cb$ for some constant $C > 0$, and $f(x) = o(g(x))$ means $|f(x)| \le \epsilon |g(x)|$ for all $\epsilon > 0$. We use $\mathcal{N}(\mu, \Sigma)$ to denote a Gaussian random variable, or a Gaussian distribution, with mean μ and covariance Σ.

2. Variational Bayes

This section gives a brief overview of the VB method. Let y be the data and $p(y|\theta)$ the likelihood function, with θ the set of model parameters. Let $p(\theta)$ be the prior. Bayesian inference requires computing expectations with respect to the posterior distribution with density

$$p(\theta|y) = \frac{p(\theta)p(y|\theta)}{p(y)},$$

where $p(y) = \int p(\theta)p(y|\theta)d\theta$ is often called the marginal likelihood. It is often difficult to compute such expectations, partly because the density $p(\theta|y)$ itself is intractable as the normalizing constant $p(y)$ is unknown. For simple models, Bayesian inference can be performed using Markov chain Monte Carlo (MCMC), which estimates expectations with respect to $p(\theta|y)$ by sampling from it. For models where θ is high dimensional or has a complicated structure, MCMC methods in their current development are either not applicable or very time consuming. In the latter case, VB is an attractive alternative to MCMC. VB approximates the posterior $p(\theta|y)$ by a probability distribution with density $q_\lambda(\theta)$, $\lambda \in \mathcal{M}$—the variational parameter space, belonging to some tractable family of distributions such as Gaussian. The best λ is found by minimizing the Kullback-Leibler

(KL) divergence of $p(\theta|y)$ from $q_\lambda(\theta)$

$$\lambda^* = \arg\min_{\lambda\in\mathcal{M}} \left\{ \text{KL}(q_\lambda \| p(\cdot|y)) = \int q_\lambda(\theta) \log \frac{q_\lambda(\theta)}{p(\theta|y)} d\theta \right\}. \tag{2.1}$$

One can easily check that

$$\text{KL}(q_\lambda \| p(\cdot|y)) = -\int q_\lambda(\theta) \log \frac{p(\theta)p(y|\theta)}{q_\lambda(\theta)} d\theta + \log p(y).$$

Thus minimizing KL is equivalent to maximizing the lower bound which is also called ELBO on $\log p(y)$

$$\text{LB}(\lambda) = \int q_\lambda(\theta) \log \frac{p(\theta)p(y|\theta)}{q_\lambda(\theta)} d\theta = \mathbb{E}_{q_\lambda}\left[\log \frac{p(\theta)p(y|\theta)}{q_\lambda(\theta)}\right]$$
$$= \mathbb{E}_{q_\lambda}\left[h_\lambda(\theta)\right], \tag{2.2}$$

where $h_\lambda(\theta) := \log p(\theta) + \log p(y|\theta) - \log q_\lambda(\theta)$. Using the fact that $\mathbb{E}_{q_\lambda}[\nabla_\lambda \log q_\lambda(\theta)] = 0$, it can be seen that

$$\nabla_\lambda \text{LB}(\lambda) = \mathbb{E}_{q_\lambda}\left[\nabla_\lambda \log q_\lambda(\theta) \times h_\lambda(\theta)\right]. \tag{2.3}$$

One then can obtain an unbiased estimate of $\nabla_\lambda \text{LB}(\lambda)$ by sampling from q_λ,

$$\widehat{\nabla_\lambda \text{LB}}(\lambda) = \frac{1}{B}\sum_{s=1}^{B} \nabla_\lambda \log q_\lambda(\theta_s) \times h_\lambda(\theta_s),$$
$$\theta_s \sim q_\lambda(\theta), \quad s = 1, \ldots, B. \tag{2.4}$$

Alternative to (2.3), the Euclidean gradient $\nabla_\lambda \text{LB}(\lambda)$ can be computed using the so-called reparameterization-trick method (Kingma and Welling 2013; Titsias and Lázaro-Gredilla 2014). The method for estimating the natural gradient proposed in this article is applicable in both cases.

Stochastic gradient ascent (SGA) techniques are often employed to solve the maximization problem in (2.1). More specifically, one can iteratively update λ as follows

$$\lambda^{(k+1)} = \lambda^{(k)} + \tau_{k+1} \widehat{\nabla_\lambda \text{LB}}(\lambda^{(k)}), \tag{2.5}$$

with the stepsize τ_k satisfying $\sum_{k=1}^\infty \tau_k = \infty$ and $\sum_{k=1}^\infty \tau_k^2 < \infty$. The convergence of the update in (2.5) has been studied in the literature (see, e.g., Robbins and Monro 1951; Spall 2005).

SGA approximates the exact gradient at each iteration by an estimate using a mini-batch of the full sample (in big data settings) or by sampling from q_λ (as in (2.4)). This reduces computational cost, and facilitates on-the-fly (online) learning as new samples arrive. Note that, in practice, the data-dependent term $h_\lambda(\theta)$ is often estimated by using a mini-batch of the data. It is documented extensively in the literature (see, e.g., Bercu, Godichon, and Portier 2020; Kirkby et al. 2022; Chau et al. 2024) that plain SGA as in (2.5) can lead to unsatisfactory estimates, as it is highly sensitive to the choice of hyper-parameters such as the step size or mini-batch size. In addition, SGA is known to have slow convergence when the Hessian of the cost function is ill-conditioned (Bottou, Curtis, and Nocedal 2018), and even in the best case SGA converges no faster than sublinearly (Pelletier 1998; Agarwal et al. 2009; Saad 2009). Significant effort in enhancing the plain SGA focuses on deriving adaptive learning step sizes; notable methods include Adam (Kingma and Ba 2014), AdaGrad (Duchi, Hazan, and Singer 2011) and Adadelta (Zeiler 2012). An alternative approach involves employing the natural gradient, which we will discuss in the following section.

3. Natural Gradient

Let $\mathcal{Q} = \{q_\lambda(\theta) : \lambda \in \mathcal{M} \subset \mathbb{R}^D\}$ be the set of VB approximating probability distributions parameterized by λ. We denote by d the dimension of the model parameter θ, and by D the dimension of the variational parameter λ. Gradient-based search for the optimal λ relies on the concept of gradient whose definition depends upon the metric on $\mathcal{M}$. It turns out that the regular Euclidean metric may not be appropriate for measuring the distance between two densities indexed by different variational parameters. For instance, by adapting examples given in Salimbeni, Eleftheriadis, and Hensman (2018), the pair of two Gaussians $\mathcal{N}(0, 0.1)$ and $\mathcal{N}(0, 1.1)$ look significantly different from each other, compared to the pair $\mathcal{N}(0, 1000)$ and $\mathcal{N}(0, 1001)$. Both pairs have the same Euclidean distance, while their KL divergences highlight a significant difference: the first pair exhibits a KL divergence of 1.6, whereas the second pair has a KL divergence of 2.5×10^{-7}.

Now consider two variational parameters λ, $\lambda + \delta\lambda$ and the KL divergence $\text{KL}(q_\lambda || q_{\lambda+\delta\lambda})$. From Tran, Nguyen, and Nguyen (2021), Tan (2021), it can be seen that

$$\text{KL}(q_\lambda || q_{\lambda+\delta\lambda}) \approx \frac{1}{2}(\delta\lambda)^\top I_F(\lambda)\delta\lambda,$$

where,

$$I_F(\lambda) := -\mathbb{E}_{q_\lambda}\left[\nabla_\lambda^2 \log q_\lambda(\theta)\right]$$
$$= \mathbb{E}_{q_\lambda}\left[\nabla_\lambda \log q_\lambda(\theta)(\nabla_\lambda \log q_\lambda(\theta))^\top\right] \tag{3.1}$$

is the Fisher information matrix of q_λ. This shows that the local KL divergence around the point $q_\lambda \in \mathcal{Q}$ is characterized by the Fisher matrix $I_F(\lambda)$. Therefore, a suitable metric between λ and $\lambda + \delta\lambda$ is the Fisher-Rao metric $(\delta\lambda)^\top I_F(\lambda)\delta\lambda$. As a result, assuming the objective function LB is smooth enough and for $l > 0$, if one considers the following optimization problem,

$$\arg \max_{\delta\lambda:(\delta\lambda)^\top I_F(\lambda)\delta\lambda = l} \left\{\nabla_\lambda \text{LB}(\lambda)^\top \delta\lambda\right\}, \tag{3.2}$$

then through the method of Lagrangian multipliers, the steepest ascent is

$$\delta\lambda = \nabla_\lambda^{\text{nat}} \text{LB}(\lambda) := I_F^{-1}(\lambda)\nabla_\lambda \text{LB}(\lambda). \tag{3.3}$$

Amari (1998) termed this the natural gradient and popularized it in machine learning.

Using the natural gradient, the update in (2.5) becomes

$$\lambda^{(k+1)} = \lambda^{(k)} + \tau_{k+1} I_F^{-1}(\lambda^{(k)})\nabla_\lambda \text{LB}(\lambda^{(k)}). \tag{3.4}$$

In the statistics literature, the steepest ascent in the form (3.4) has been used for a long time and is often known as Fisher's scoring in the context of maximum likelihood estimation (see, e.g., Longford 1987). The efficiency of the natural gradient over the Euclidean gradient has been well documented (Sato 2001; Hoffman et al. 2013; Tran, Nott, and Kohn 2017; Martens 2020; Tan 2021). The natural gradient is invariant under parameterization (Martens 2020), meaning it remains unchanged across different coordinate systems and is an intrinsic geometric object. This property makes it particularly suitable for use when the variational parameter space $\mathcal{M}$ is a Riemannian manifold (Tran, Nguyen, and Nguyen 2021) as it is coordinate-free and leverages the underlying geometry of the space.

In the special case of Gaussian approximations with a full covariance matrix or a Cholesky-factor covariance matrix, it is possible to obtain the inverse Fisher matrix in closed form (Tan 2021; Magris, Shabani, and Iosifidis 2022). Beyond these limited cases, however, it is challenging to accurately compute the natural gradient. The natural gradient method requires the *analytic* computation of the Fisher information matrix and its *inversion*. Even if an analytical expression of the Fisher matrix is obtained, computing its inverse has a complexity of $O(D^\kappa)$, with $2 < \kappa \leq 3$ depending on various algorithms. It is therefore either analytically infeasible or prohibitively computationally expensive to use natural gradient in many modern statistical applications; current practice resorts to heuristic workarounds that can affect the results of Bayesian inference (Martens 2020; Lopatnikova and Tran 2023).

4. Inversion Free Natural Gradient Variational Bayes

This section first presents the approach for approximating the inverse of Fisher matrix. We then present the inversion free natural gradient Variational Bayes method, referred to as IFVB, and its weighted averaged version AIFVB. The IFVB and AIFVB methods are stochastic natural gradient descent algorithms that avoid computing the Fisher matrix and its inversion altogether. As explained later, these methods also enable us to deal with situations where the estimate of Fisher matrix has eigenvalues with significantly different orders of magnitude (i.e., poor conditioning). In this section and Section 5, we will denote $\mathcal{L}(\lambda) = -\mathrm{LB}(\lambda)$, which can be viewed as the loss function, and the problem of maximizing the lower bound becomes the minimization of $\mathcal{L}$.

4.1. Recursive Estimation of $I_F^{-1}(\lambda)$

For each $s = 1, 2 \ldots$, let

$$H_s = H_0 + \sum_{j=1}^{s} \nabla_\lambda \log q_\lambda(\theta_j)(\nabla_\lambda \log q_\lambda(\theta_j))^\top, \quad \theta_j \sim q_\lambda, \quad (4.1)$$

where H_0 is some positive definite matrix, for example $H_0 = \epsilon \mathbb{I}_D$ with $\epsilon > 0$ and $\mathbb{I}_D$ the identity matrix of size D. Theorem 4.1 says that $\mathbf{H}_s := H_s/s$ is a consistent estimate of I_F defined in (3.1) as $s \to \infty$, and provides a recursive procedure for obtaining H_s^{-1}. This recursive procedure computes H_{s+1}^{-1} from H_s^{-1} without resorting to the usual (expensive and error prone) matrix inversion. Additionally, the symmetry and positivity of H_s, and hence of H_s^{-1}, is preserved, which is an important property.

Theorem 4.1. Let $\phi_s = \nabla_\lambda \log q_\lambda(\theta_s)$. We have that

$$H_{s+1}^{-1} = H_s^{-1} - \left(1 + \phi_{s+1}^\top H_s^{-1} \phi_{s+1}\right)^{-1} H_s^{-1} \phi_{s+1} \phi_{s+1}^\top H_s^{-1}, \quad s = 0, 1, \ldots \quad (4.2)$$

In particular, the positivity and symmetry of H_s is preserved for all s. Furthermore,

$$\mathbf{H}_s = \frac{1}{s} H_s \xrightarrow[s \to +\infty]{a.s} I_F(\lambda) \quad \text{and} \quad \mathbf{H}_s^{-1} \xrightarrow[s \to +\infty]{a.s} I_F^{-1}(\lambda).$$

The proof of Theorem 4.1 can be found in the Appendix, supplementary materials. The expression (4.2) suggests that H_s^{-1} is a sum of outer products—a property that can be exploited to avoid storage of large matrices; see Remark 4.2.

4.2. Inversion Free Natural Gradient Variational Bayes

Theorem 4.1 suggests that one can approximate the inverse Fisher matrix I_F^{-1} by $\mathbf{H}_s^{-1} = sH_s^{-1}$ for some large s, where H_s^{-1} is calculated recursively as in (4.2). This method does not require an analytic calculation of I_F and its inversion; also, the estimate $\mathbf{H}_s^{-1}$ is guaranteed to be symmetric and positive definite. However, a direct application of Theorem 4.1 for computing the natural gradient can be inefficient for two reasons.

First, in order to ensure the consistency of estimates $\lambda^{(k)}$ from (3.4), where the inverse Fisher matrix is replaced by its estimate $\mathbf{H}_s^{-1}$, one must control the eigenvalues of the estimate $\mathbf{H}_s^{-1}$. See Bercu, Godichon, and Portier (2020) and Boyer and Godichon-Baggioni (2023) for related discussion in the context of stochastic Newton's method. With this aim, we follow Boyer and Godichon-Baggioni (2023) and modify (4.1) as follows

$$A_s(\lambda) = H_0 + \sum_{j=1}^{s} \nabla_\lambda \log q_\lambda(\theta_j)(\nabla_\lambda \log q_\lambda(\theta_j))^\top + c_\beta \sum_{j=1}^{s} j^{-\beta} Z_j Z_j^\top, \quad s = 1, 2, \ldots \quad (4.3)$$

where $\theta_j \sim q_\lambda(\cdot)$, $Z_1, \ldots, Z_s \sim \mathcal{N}(0, \mathbb{I}_D)$ are independent standard Gaussian vectors of dimension D, $c_\beta \geq 0$ and $\beta \in (0, \alpha - 1/2)$ for some $\alpha \in (1/2, 1)$. Theorem A.1 in the Appendix, supplementary materials shows that A_s/s converges almost surely to $I_F(\lambda)$, and that A_{s+1}^{-1} can be computed recursively as follows

$$\begin{cases} A_{s+\frac{1}{2}}^{-1} = A_s^{-1} - \left(1 + \phi_{s+1}^\top A_s^{-1} \phi_{s+1}\right)^{-1} A_s^{-1} \phi_{s+1} \phi_{s+1}^\top A_s^{-1} \\ A_{s+1}^{-1} = A_{s+\frac{1}{2}}^{-1} - c_\beta (s+1)^{-\beta} \left(1 + c_\beta (s+1)^{-\beta} Z_{s+1}^\top A_{s+\frac{1}{2}}^{-1} Z_{s+1}\right)^{-1} A_{s+\frac{1}{2}}^{-1} Z_{s+1} Z_{s+1}^\top A_{s+\frac{1}{2}}^{-1} \end{cases} \quad (4.4)$$

with $\phi_s = \nabla_\lambda \log q_\lambda(\theta_s)$. As being shown later in the proof of Theorem 5.1, taking $c_\beta > 0$ ensures the smallest eigenvalue of the Fisher matrix estimate not going to zero faster than $O(s^{-\beta})$ almost surely, hence enabling strongly consistent estimates.

Second, a direct use of (4.3) would require a separate recursive procedure (4.4) to calculate the inverse Fisher estimate $sA_s^{-1}(\lambda^{(k)})$ in each update $\lambda^{(k)}$, $k = 1, 2, \ldots$, in (3.4). This might make the VB training procedure in (3.4) computationally expensive. Instead, we propose a “streamlined” version that updates the Fisher matrix estimate along with the iterates $\lambda^{(k)}$. To this end, we define

$$\tilde{H}_s = H_0 + \sum_{k=0}^{s-1} \nabla_\lambda \log q_{\lambda^{(k)}}(\theta_{k+1})\big(\nabla_\lambda \log q_{\lambda^{(k)}}(\theta_{k+1})\big)^\top + c_\beta \sum_{k=1}^{s} k^{-\beta} Z_k Z_k^\top, \quad s = 0, 1, \ldots, \tag{4.5}$$

where $\theta_{k+1} \sim q_{\lambda^{(k)}}(\cdot)$. Note that, unlike (4.3), $\tilde{H}_s$ in (4.5) depends on the iterates $\lambda^{(k)}$, $k < s$. Similar to (4.4), $\tilde{H}_{s+1}^{-1}$ can be computed recursively as follows

$$\begin{cases} \tilde{H}_{s+\frac{1}{2}}^{-1} = \tilde{H}_s^{-1} - \left(1 + \phi_{s+1}^\top \tilde{H}_s^{-1}\phi_{s+1}\right)^{-1} \tilde{H}_s^{-1}\phi_{s+1}\phi_{s+1}^\top \tilde{H}_s^{-1} \\ \tilde{H}_{s+1}^{-1} = \tilde{H}_{s+\frac{1}{2}}^{-1} - c_\beta (s+1)^{-\beta}\left(1 + c_\beta (s+1)^{-\beta} Z_{s+1}^\top \tilde{H}_{s+\frac{1}{2}}^{-1} Z_{s+1}\right)^{-1} \tilde{H}_{s+\frac{1}{2}}^{-1} Z_{s+1} Z_{s+1}^\top \tilde{H}_{s+\frac{1}{2}}^{-1}, \end{cases} \tag{4.6}$$

where $\phi_{s+1} = \nabla_\lambda \log q_{\lambda^{(s)}}(\theta_{s+1})$.

The Euclidean gradient of the loss function can be estimated as

$$\widehat{\nabla_\lambda \mathcal{L}}(\lambda) = -\frac{1}{B}\sum_{i=1}^{B} \nabla_\lambda \log q_\lambda(\theta_i) \times h_\lambda(\theta_i), \tag{4.7}$$

where the θ_i's are B iid samples from q_λ.

Putting these together, we propose the Inverse Free Natural Gradient VB algorithm, referred to as IFVB and outlined in Algorithm 1. The performance of this algorithm is found not sensitive to ϵ and c_β; both are set to 1 in our examples below unless stated otherwise. We found that the algorithm works well for β in the range (0.1,0.5).

Algorithm 1: Inversion-Free Natural Gradient Variational Bayes (IFVB)

```
Require: Choose an initial value $\lambda^{(0)}$, $\epsilon > 0$ and $c_\beta \ge 0$.
1: $H_0 = \epsilon \mathbb{I}_D$
2: for $s = 0, 1, \ldots,$ do
3:     Compute $\widehat{\nabla_\lambda \mathcal{L}}(\lambda^{(s)})$.
4:     Sample $\theta_{s+1} \sim q_{\lambda^{(s)}}(\theta)$, $Z_{s+1} \sim \mathcal{N}(0, \mathbb{I}_D)$
       and let $\phi_{s+1} = \nabla_\lambda \log q_{\lambda^{(s)}}(\theta_{s+1})$. Calculate $\tilde{H}_{s+1}^{-1}$ as in
       (4.6), and $\tilde{\mathbf{H}}_{s+1}^{-1} = (s+1)\tilde{H}_{s+1}^{-1}$.
5:     Update $\lambda^{(s+1)} = \lambda^{(s)} - \tau_{s+1}\tilde{\mathbf{H}}_{s+1}^{-1}\widehat{\nabla_\lambda \mathcal{L}}(\lambda^{(s)})$.
6: end for
```

Remark 4.1. Some remarks are in order. Line #4 in Algorithm 1 updates $\tilde{H}_{s+1}^{-1}$ from $\tilde{H}_s^{-1}$ using only one sample from $q_{\lambda^{(s)}}$. Depending on applications, however, it might be beneficial to use S samples ($S > 1$) to update $\tilde{H}_{s+1}^{-1}$. One then needs to run (4.6) for S times, and compute $\tilde{\mathbf{H}}_{s+1}^{-1} = S(s+1)\tilde{H}_{s+1}^{-1}$. The factor S is not practically important as gradient clipping, that keeps the norm of the natural gradient $\tilde{\mathbf{H}}_{s+1}^{-1}\widehat{\nabla_\lambda \mathcal{L}}(\lambda^{(s)})$ below some certain value, is often used in the SGD literature (see, e.g., Goodfellow, Bengio, and Courville 2016).

Remark 4.2. For applications such as deep learning where the size D of variational parameter λ is large, it is important to note that implementation of Algorithm 1 does not require storage of the matrices $\tilde{H}_{s+1}^{-1}$. Let us take $c_\beta = 0$ to simplify the exposition; the extension to the case $c_\beta > 0$ is straightforward. From (4.6), $\tilde{H}_{s+1}^{-1}$ takes the form

$$\tilde{H}_{s+1}^{-1} = \frac{1}{\epsilon}\mathbb{I}_D - \sum_{k=1}^{s+1} \psi_k \psi_k^\top,$$
$$\psi_k = \left(1 + \phi_{k+1}^\top \tilde{H}_k^{-1}\phi_{k+1}\right)^{-1/2} \tilde{H}_k^{-1}\phi_{k+1}, \tag{4.8}$$

which is presented by outer products. As the result, all the required matrix-vector multiplications can be performed efficiently. That is,

$$\tilde{H}_{s+1}^{-1}\widehat{\nabla_\lambda \mathcal{L}}\left(\lambda^{(s)}\right) = \frac{1}{\epsilon}\widehat{\nabla_\lambda \mathcal{L}}\left(\lambda^{(s)}\right)^\top \widehat{\nabla_\lambda \mathcal{L}}\left(\lambda^{(s)}\right) - \sum_{k=1}^{s+1}\left(\psi_k^\top \widehat{\nabla_\lambda \mathcal{L}}\left(\lambda^{(s)}\right)\right)\psi_k$$
$$\tilde{H}_s^{-1}\phi_{s+1} = \frac{1}{\epsilon}\phi_{s+1}^\top \phi_{s+1} - \sum_{k=1}^{s}\left(\psi_k^\top \phi_{s+1}\right)\psi_k$$

which does not necessitate storage of the matrices $\tilde{H}_s^{-1}$. It is also natural to only keep the last K outer products, for some $K \ge 1$ ($K = 100$ in our examples below), in (4.8) to further reduce the computation

$$\tilde{H}_{s+1}^{-1} \approx \frac{1}{\epsilon}\mathbb{I}_D - \sum_{k=s-K+2}^{s+1} \psi_k \psi_k^\top. \tag{4.9}$$

The computational complexity of Algorithm 1 is therefore $O(s \times D)$, with s the number of iterations. This complexity is the same as that of the popular adaptive learning methods such as Adam and AdaGrad.

4.3. Weighted Averaged IFVB Algorithm

It is a common practice in the SGD literature to use an average of the iterates $\lambda^{(k)}$ to form the final estimate of the optimal λ^* (Goodfellow, Bengio, and Courville 2016; Boyer and Godichon-Baggioni 2023). The weighted averaged estimate is of the form

$$\overline{\lambda}^{(s+1)} = \frac{1}{\sum_{k=1}^{s+1} w_k} \sum_{k=1}^{s+1} w_k \lambda^{(k)} = \overline{\lambda}^{(s)} + \frac{w_{s+1}}{\sum_{k=1}^{s+1} w_k} \left(\lambda^{(s+1)} - \overline{\lambda}^{(s)}\right),$$

where the weights $w_k > 0$. Inspired by Boyer and Godichon-Baggioni (2023), we select $w_k = \left(\log(k)\right)^w$ with $w \geq 0$. This averaging scheme puts more weight on recent estimates $\lambda^{(k)}$ if $w > 0$. If one chooses $w = 0$, this leads to the uniform averaging technique, that is,

$$\overline{\lambda}^{(s)} = \frac{1}{s} \sum_{k=1}^{s} \lambda^{(k)}.$$

We use $w = 2$ in all the numerical examples in Section 6. We arrive at the Weighted Averaged IFVB Algorithm (AIFVB)

$$\lambda^{(s+1)} = \lambda^{(s)} - \tau_{s+1} \tilde{\mathbf{H}}_{s+1}^{-1} \widehat{\nabla_\lambda \mathcal{L}}\left(\lambda^{(s)}\right)$$
$$\overline{\lambda}^{(s+1)} = \overline{\lambda}^{(s)} + \frac{\left(\log(s+1)\right)^w}{\sum_{k=0}^{s}\left(\log(k+1)\right)^w}\left(\lambda^{(s+1)} - \overline{\lambda}^{(s)}\right).$$

For the AIFVB algorithm, we consider the following estimate of the Fisher matrix

$$\tilde{\mathbf{H}}_s = \frac{1}{s}\left(H_0 + \sum_{k=0}^{s-1} \nabla_\lambda \log q_{\overline{\lambda}^{(k)}}(\overline{\theta}_{k+1})(\nabla_\lambda \log q_{\overline{\lambda}^{(k)}}(\overline{\theta}_{k+1}))^\top + c_\beta \sum_{k=1}^{s} k^{-\beta} Z_k Z_k^T\right), \tag{4.10}$$

for $s = 1, 2, \ldots$, where $\overline{\theta}_{k+1} \sim q_{\overline{\lambda}^{(k)}}(\theta)$, the Z_k's are defined as before. Similar to (4.6), $\tilde{\mathbf{H}}_{s+1}^{-1}$ can be updated recursively.

Remark 4.3. In the estimate (4.10), we compute $\tilde{\mathbf{H}}_s$ using the averaged iterates $\overline{\lambda}^{(k)}$, as this can lead to a faster convergence. Nonetheless, one can also use $\lambda^{(k)}$ instead of $\overline{\lambda}^{(k)}$ in (4.10).

Putting these together, we have the Weighted Averaged IFVB Algorithm, outlined in Algorithm 2.

Algorithm 2: Weighted Averaged IFVB Algorithm (AIFVB)

```
Require: Choose an initial value $\lambda^{(0)} = \overline{\lambda}^{(0)}$, $\epsilon > 0$ and $c_\beta \geq 0$.
1: $H_0 = \epsilon \mathbb{I}_D$
2: for $s = 0, 1, \ldots,$ do
3:     Compute $\widehat{\nabla_\lambda \mathcal{L}}(\lambda^{(s)})$.
4:     Sample $\overline{\theta}_{s+1} \sim q_{\overline{\lambda}^{(s)}}(\theta)$, $Z_{s+1} \sim \mathcal{N}(0, \mathbb{I}_D)$
       and let $\phi_{s+1} = \nabla_\lambda \log q_{\overline{\lambda}^{(s)}}(\overline{\theta}_{s+1})$. Calculate $\tilde{H}_{s+1}^{-1}$ as in
       (4.6), and $\tilde{\mathbf{H}}_{s+1}^{-1} = (s+1)\tilde{H}_{s+1}^{-1}$.
5:     Update $\lambda^{(s+1)} = \lambda^{(s)} - \tau_{s+1}\tilde{\mathbf{H}}_{s+1}^{-1}\widehat{\nabla_\lambda \mathcal{L}}(\lambda^{(s)})$.
6:     Update $\overline{\lambda}^{(s+1)} = \overline{\lambda}^{(s)} + \frac{(\log(s+1))^w}{\sum_{k=0}^{s}(\log(k+1))^w}(\lambda^{(s+1)} - \overline{\lambda}^{(s)})$.
7: end for
```

5. Convergence Analysis

For the convergence analysis in this section, we consider a step size of the form $\tau_k = \frac{c_\alpha}{(c'_\alpha + k)^\alpha}$ with $c_\alpha > 0$, $c'_\alpha \geq 0$ and $\alpha \in (1/2, 1)$. Write

$$\nabla_\lambda \mathcal{L}(\lambda) = \mathbb{E}_{q_\lambda}[-\nabla_\lambda \log q_\lambda(\theta) \times h_\lambda(\theta)] =: \mathbb{E}_{q_\lambda}[\ell(\theta, \lambda)].$$

It can be shown that (see Appendix A.3, supplementary materials for a detailed calculation; also, Tang and Ranganath 2019; Tan 2021)

$$\nabla_\lambda^2 \mathcal{L}(\lambda) = I_F(\lambda) + \int \nabla_\lambda^2 q_\lambda(\theta)\left(\log q_\lambda(\theta) - \log p(\theta|y)\right) d\theta.$$

At the optimal $\lambda = \lambda^*$, $q_{\lambda^*}(\theta) \approx p(\theta|y)$, the second term above expects to be close to zero. We can therefore expect that, in a neighborhood of λ^*, the Fisher $I_F(\lambda)$ behaves like the Hessian $\nabla_\lambda^2 \mathcal{L}(\lambda)$. This motivates us to adapt the results from stochastic Newton algorithms (see, e.g., Bercu, Godichon, and Portier 2020; Boyer and Godichon-Baggioni 2023) for convergence analysis of IFVB and AIFVB proposed in this article.

We now provide the convergence analysis for the AIFVB algorithm; a minor modification provides convergence results for IFVB.

Theorem 5.1. Assume that $\mathcal{L}$ is twice differentiable and that there is L_0 such that for all λ, $\left\|\nabla_\lambda^2 \mathcal{L}(\lambda)\right\|_{op} \leq L_0$. Suppose that $\nabla_\lambda \mathcal{L}(\lambda^*) = 0$ and $c_\beta > 0$. Assume also that there are nonnegative constants C_0, C_1 such that for all λ

$$\mathbb{E}\left[\|\ell(\theta, \lambda)\|^2\right] \leq C_0 + C_1(\mathcal{L}(\lambda) - \mathcal{L}(\lambda^*))$$

and that there are positive constants C'_0, C'_1 such that for all λ,

$$\mathbb{E}\left[\left\|\nabla_\lambda \log q_\lambda(\theta)\right\|^4\right] \leq C'_0 + C'_1\left(\mathcal{L}(\lambda) - \mathcal{L}(\lambda^*)\right)^2. \tag{5.1}$$

Suppose that $C'_1 = 0$ or that $\mathcal{L}$ is convex. Then the estimates $\lambda^{(k)}$, $\overline{\lambda}^{(k)}$ $k = 0, 1, \ldots$, from Algorithm 2 satisfy:

(i) $\mathcal{L}\left(\lambda^{(k)}\right) - \mathcal{L}\left(\lambda^*\right)$ converges almost surely to a finite random variable.
(ii) $\min_{k=0}^{s}\left\|\nabla \mathcal{L}\left(\lambda^{(k)}\right)\right\|^2 = o\left(s^{-(1-\alpha)}\right)$ a.s.
(iii) $\min_{k=0}^{s}\left\|\nabla \mathcal{L}\left(\overline{\lambda}^{(k)}\right)\right\|^2 = o\left(s^{-(1-\alpha)}\right)$ a.s.

If the convexity of $\mathcal{L}$ is satisfied, conclusions (ii) and (iii) in Theorem 5.1 imply the almost sure convergence of $\lambda^{(k)}$ and $\overline{\lambda}^{(k)}$ to λ^*. The assumption in (5.1) might appear to look irrelevant, as the right-hand side term is model-dependent, that is depending on the prior and likelihood, while the left-hand side is not. One can simply replace (5.1) with an assumption on the uniform bound of the fourth order moment of $\nabla_\lambda \log q_\lambda$, which obviously implies (5.1). We use (5.1) to keep the result as general as possible.

We now give the consistency of the estimates of the Fisher information given in (4.10).

Corollary 5.1. Suppose that $\lambda^{(k)}$ converges almost surely to λ^*, and that the map $\lambda \longmapsto I_F(\lambda)$ is continuous at λ^*. Suppose also that (5.1) is satisfied. Then

$$\tilde{\mathbf{H}}_s \xrightarrow[s \to +\infty]{a.s} I_F\left(\lambda^*\right).$$

Note that we can obtain the convergence rate of $\tilde{\mathbf{H}}_s$ to $I_F(\lambda^*)$, or even the rate of convergence of $\tilde{\mathbf{H}}_s$ to the Hessian $\nabla^2_\lambda \mathcal{L}(\lambda^*)$. Please see Appendix A.4, supplementary materials for more details.

Without the convexity assumption, the proof of Theorem 5.1 implies that the estimates $\lambda^{(k)}$ converge to a stationary point λ^* of the objective $\mathcal{L}$, that is $\nabla_\lambda \mathcal{L}(\lambda^*) = 0$. The result below gives a convergence rate of the estimates $\lambda^{(k)}$ to such λ^*.

Theorem 5.2. Suppose that $\lambda^{(k)}$ converges almost surely to λ^*. Assume that the functional $\mathcal{L}$ is differentiable with $\nabla_\lambda \mathcal{L}(\lambda^*) = 0$ and twice continuously differentiable in a neighborhood of λ^*. Suppose also that there are $\eta > \frac{1}{\alpha} - 1$ and positive constants $C_{\eta,0}, C_{\eta,1}$ such that for all λ,

$$\mathbb{E}\left[\|\ell(\theta,\lambda)\|^{2+2\eta}\right] \le C_{\eta,0} + C_{\eta,1}\left(\mathcal{L}(\lambda) - \mathcal{L}\left(\lambda^*\right)\right)^{1+\eta}$$

and that inequality (5.1) holds. Suppose also that $\nabla^2_\lambda \mathcal{L}\left(\lambda^*\right)$ and $I_F\left(\lambda^*\right)$ are positive. Then

$$\left\|\lambda^{(s)} - \lambda^*\right\|^2 = \mathcal{O}\left(\frac{\log s}{s^\alpha}\right) \quad a.s. \tag{5.2}$$

Observe that (5.2) is the usual rate of convergence for (adaptive) stochastic gradient type algorithms (Bercu, Godichon, and Portier 2020; Nguyen et al. 2021). The following theorem shows that the weighted averaged estimates achieve a better convergence rate and are asymptotically efficient.

Theorem 5.3. Suppose that the assumptions in Theorem 5.2 hold. Assume that the functional

$$\lambda \longmapsto \Sigma(\lambda) := \mathbb{E}_{q_\lambda}\left[\ell(\theta,\lambda)\ell(\theta,\lambda)^\top\right]$$

is continuous at λ^*, and there are a neighborhood V^* of λ^* and a constant L_δ such that for all $\lambda \in V^*$,

$$\left\|\nabla_\lambda \mathcal{L}(\lambda) - \nabla^2 \mathcal{L}(\lambda^*)(\lambda - \lambda^*)\right\| \le L_\delta \left\|\lambda - \lambda^*\right\|^2. \tag{5.3}$$

Then

$$\left\|\overline{\lambda}^{(s)} - \lambda^*\right\|^2 = \mathcal{O}\left(\frac{\log s}{s}\right) \quad a.s.$$

In addition,

$$\sqrt{Bs}\left(\overline{\lambda}^{(s)} - \lambda^*\right) \xrightarrow[n\to+\infty]{law} \mathcal{N}\left(0, \nabla^2_\lambda \mathcal{L}\left(\lambda^*\right)^{-1} \Sigma\left(\lambda^*\right) \nabla^2_\lambda \mathcal{L}\left(\lambda^*\right)^{-1}\right).$$

Remark 5.1. Comparing Theorems 5.2 and 5.3, it is intriguing to observe that the averaging technique clearly contributes to an improvement in the rate of convergence. Moreover, observe that Bs represent the total number of samples generated for the AIFVB algorithm (without taking into account the ones for estimating the inverse of the Fisher information). In addition, following Godichon-Baggioni and Werge (2023), that is taking $B = D$ and generating only one sample to estimate the inverse of the Fisher information, it leads to an algorithm with $O(sBD)$ operations, which is the same computational complexity as stochastic gradient type algorithms. To be more precise, taking $B = D$, and denoting by $N = sB$ the total number of simulated data used for estimating the gradients at each step of the algorithm, one has

$$\sqrt{N}\left(\overline{\lambda}^{N/D} - \lambda^*\right) \xrightarrow[N\to+\infty]{\text{law}} \mathcal{N}\left(0, \nabla^2_\lambda \mathcal{L}\left(\lambda^*\right)^{-1} \Sigma\left(\lambda^*\right) \nabla^2_\lambda \mathcal{L}\left(\lambda^*\right)^{-1}\right),$$

with $O(ND)$ operations, which is exactly the same rate and complexity as averaged stochastic gradient algorithms.

Remark 5.2. To the best of our knowledge, this article is the first to give a central limit theorem for the estimate of variational parameters in VB. This result can have some interesting implications. Given a variational family $\mathcal{Q}$ and data y, one can define the "final" variational approximation of the posterior distribution $p(\theta|y)$ as

$$q(\theta|y,\mathcal{Q}) = \int q_\lambda(\theta) p(\lambda|y) d\lambda \tag{5.4}$$

with $p(\lambda|y)$ approximated by $\mathcal{N}\left(\lambda^*, \frac{1}{Bs}\nabla^2_\lambda \mathcal{L}\left(\lambda^*\right)^{-1} \Sigma\left(\lambda^*\right) \nabla^2_\lambda \mathcal{L}\left(\lambda^*\right)^{-1}\right)$, who in turn can be further approximated by $\mathcal{N}\left(\lambda^*, \frac{1}{Bs} I_F\left(\lambda^*\right)^{-1} \Sigma\left(\lambda^*\right) I_F\left(\lambda^*\right)^{-1}\right)$. The posterior approximation in (5.4) not only takes into account the uncertainty in the SGD training procedure of λ, but also enlarges the variance in $q_{\lambda^*}(\theta)$. We recall that underestimating the posterior variance is a well perceived problem in the VB literature (Blei, Kucukelbir, and McAuliffe 2017).

6. Numerical Examples

This section provides a range of examples to demonstrate the applicability of the inversion free variational Bayes methods. In Sections 4 and 5, to be consistent with the optimization literature, we presented the VB problem in terms of optimizing the negative lower bound $\mathcal{L}(\lambda)$. In this section, to be consistent with the VB literature, we present the numerical examples in terms of maximizing the lower bound LB(λ). The five examples encompass a diverse array of domains ranging from Gaussian approximation to normalizing flow Variational Bayes. We provide four examples in this section; another example is provided in the supplementary material. The implementation code is available at *https://github.com/VBayesLab/Inversion-free-VB*.

Example 1 (Beta variational approximation). We consider an example in which the posterior distribution and the natural gradient are given in closed-form. This helps facilitate the comparison among the considered approximation approaches. Following Tran, Nott, and Kohn (2017), we generated $n = 200$ observations $y_1, y_2, \dots, y_{200} \sim$ Bernoulli$(1, \theta = 0.3)$, and let $\kappa = \sum_{i=1}^n y_i = 57$. The prior distribution of θ is chosen to be the uniform distribution on $(0,1)$. The posterior distribution is Beta$(\kappa+1, n-\kappa+1)$. Let $\lambda^* = (\kappa+1, n-\kappa+1)^\top$. The variational distribution $q_\lambda(\theta)$ is chosen to be Beta(α,β), which belongs to the exponential family with the natural parameter $\lambda = (\alpha,\beta)^\top$. We have

$$\begin{aligned}\log q_\lambda(\theta) = {}& \log\Gamma(\alpha+\beta) - \log\Gamma(\alpha) - \log\Gamma(\beta) \\ & +(\alpha-1)\log\theta + (\beta-1)\log(1-\theta).\end{aligned}$$

Hence,

$$\nabla_\lambda \log q_\lambda(\theta) = \big(\psi(\alpha+\beta) - \psi(\alpha) + \log(\theta), \psi(\alpha+\beta) - \psi(\beta) + \log(1-\theta)\big)^\top,$$

where ψ is the digamma function. In this case, the Fisher information matrix is available in a closed-form

$$I_F(\lambda) = \begin{pmatrix} \varphi_1(\alpha) - \varphi_1(\alpha+\beta) & -\varphi_1(\alpha+\beta) \\ -\varphi_1(\alpha+\beta) & \varphi_1(\beta) - \varphi_1(\alpha+\beta) \end{pmatrix},$$

where $\varphi_1(x)$ is the trigamma function. In addition, we have that

$$\nabla_\lambda \mathrm{LB}(\lambda) = \begin{pmatrix} (\kappa+1-\alpha)\big(\varphi_1(\alpha) - \varphi_1(\alpha+\beta)\big) - (n-\kappa+1-\beta)\varphi_1(\alpha+\beta) \\ (n-\kappa+1-\beta)\big(\varphi_1(\beta) - \varphi_1(\alpha+\beta)\big) - (\kappa+1-\alpha)\varphi_1(\alpha+\beta) \end{pmatrix}.$$

The Euclidean gradient ascent update is given by

$$\lambda^{(k+1)} = \lambda^{(k)} + \tau_k \nabla_\lambda \mathrm{LB}(\lambda^{(k)}),$$

and the natural gradient ascent update is given by

$$\lambda^{(k+1)} = \lambda^{(k)} + \tau_k I_F^{-1}(\lambda^{(k)}) \nabla_\lambda \mathrm{LB}(\lambda^{(k)}).$$

Figure 1 compares the true posterior density with those obtained by the Euclidean gradient ascent, the (exact) natural gradient VB algorithm (NGVB) and IFVB, using two initializations for λ: $\lambda^{(0)} = (5;45)^\top$ (Figure 1, left) and $\lambda^{(0)} = (25;25)^\top$ (Figure 1, right). We set $c_\beta = 0$ in this example. The step size for IFVB is set as $\tau_k = 10/(1+k)^\alpha$ with $\alpha = 0.6$ (see Section 5), while the step size for other methods is set to $1/(1+k)$. All the algorithms used the same stopping rule, which terminates the training if the difference in l_2-norm of two successive updates is less than a tolerance of 10^{-5}. The figure shows that the posterior densities obtained by NGVB and IFVB are superior to using the traditional Euclidean gradient. Furthermore, these algorithms are insensitive to the initial $\lambda^{(0)}$.

As we know the exact posterior distribution in this case, we now further compare the performance of NGVB, IFVB, and AIFVB using the initialization $\lambda^{(0)} = (5;45)^\top$. Figure 2 plots the error $\|\lambda^{(s)} - \lambda^*\|$ ($\|\overline{\lambda}^{(s)} - \lambda^*\|$ for AIFVB) as the function of the number of iterations. It is clear from Figure 2 that both the IFVB and AIFVB algorithms require more steps to achieve a (relatively) equivalent error level compared to the exact natural gradient ascent algorithm. This observation aligns with the fact that both algorithms rely on an approximation of the inverse Fisher information matrix. As time advances, the quality of this approximation improves. Furthermore, it is evident that the incorporation of an averaging technique significantly aids in minimizing the error associated with the approximation.

Example 2 (Gaussian approximation). We consider a Gaussian approximation example which was considered in Tan (2021). Specifically, it is assumed that $q_\lambda(\theta) = \mathcal{N}(\theta|\mu,\Sigma)$ which belongs to the exponential family and can be written as

$$q_\lambda(\theta) = \exp\Big(s(\theta)^\top\lambda - a(\lambda)\Big),$$

where $s(\theta) = (\theta^\top, \mathrm{vech}(\theta\theta^\top)^\top)^\top$ is the sufficient statistics, $\lambda = (\mu^\top\Sigma^{-1}, -\frac{1}{2}\mathrm{vec}(\Sigma^{-1})^\top\Gamma)^\top$ is the natural parameter, with Γ

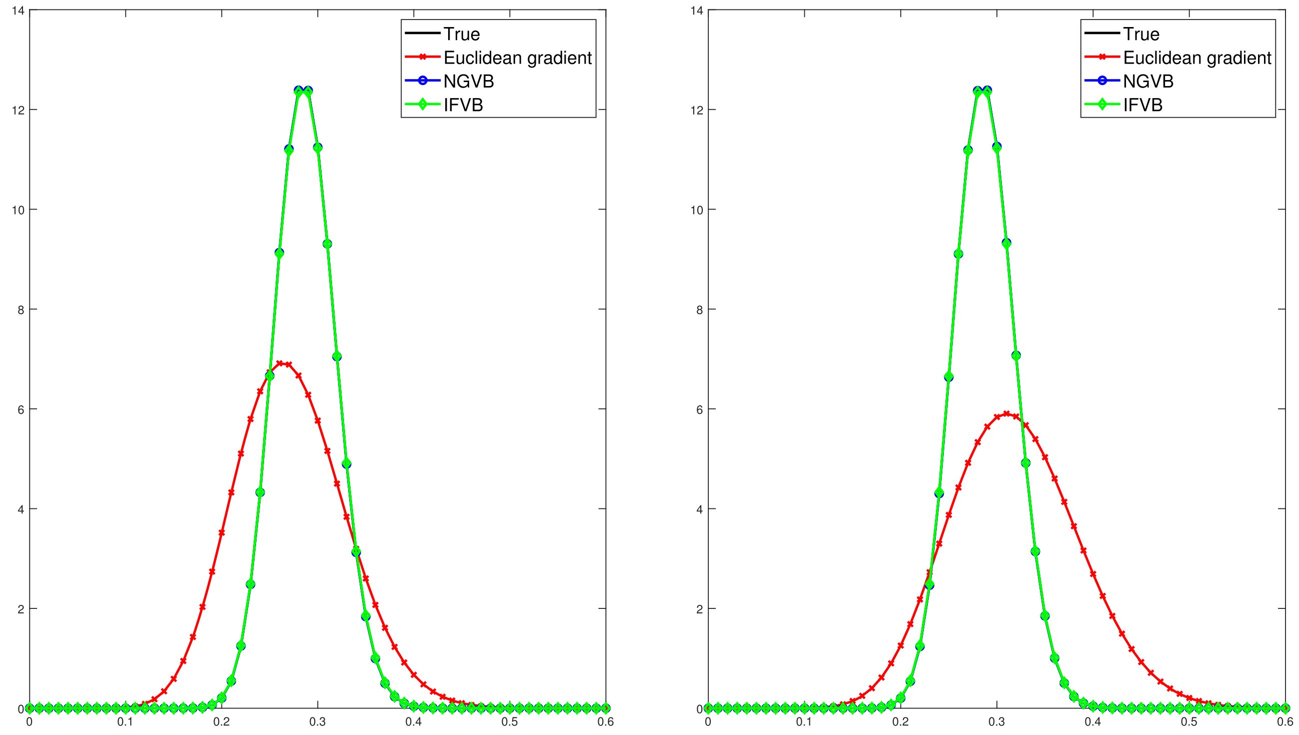

Figure 1. Comparing the true posterior density versus those obtained by Euclidean gradient ascent, NGVB and IFVB.

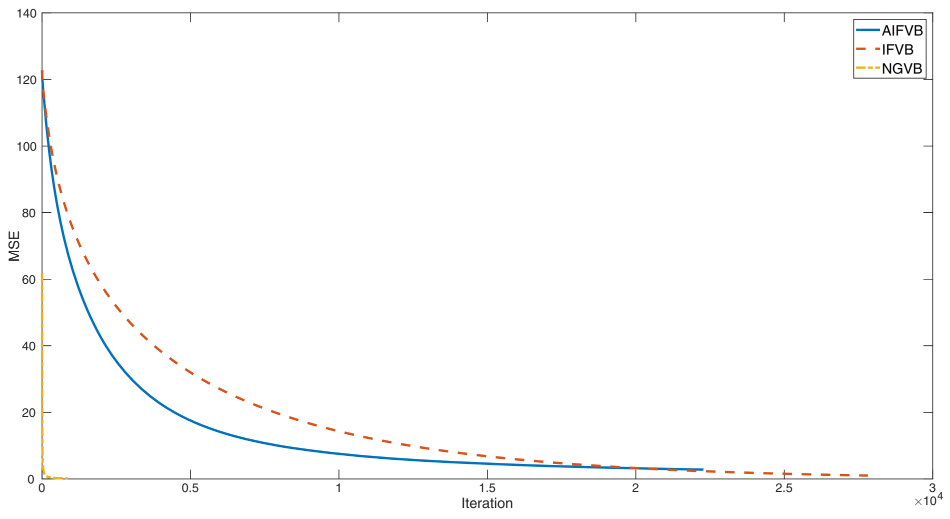

Figure 2. Error versus iterations.

the duplication matrix. It can be seen that $a(\lambda) = \frac{1}{2}\mu^\top\Sigma^{-1}\mu + \frac{1}{2}\log|\Sigma| + \frac{d}{2}\log(2\pi)$ is the log-partition function. This implies the natural gradient

$$\begin{aligned}\nabla^{nat}_\lambda \text{LB} &= I_F^{-1}(\lambda)\nabla_\lambda \text{LB}(\lambda)\\ &= I_F^{-1}(\lambda)I_F(\lambda)\nabla_m \text{LB}(m) = \nabla_m \text{LB}\\ &= \begin{pmatrix}\nabla_\mu \text{LB} - 2(\nabla_\Sigma \text{LB})\mu\\ \Gamma^\top \text{vec}(\nabla_\Sigma \text{LB})\end{pmatrix},\end{aligned}$$

where $\text{vec}(\nabla_\Sigma \text{LB}) = \nabla_{\text{vec}(\Sigma)}\text{LB}$. From here Tan (2021) derives the updates for λ as follows,

1. $\Sigma^{-1} \leftarrow \Sigma^{-1} - 2\tau_k \nabla_\Sigma \text{LB}$
2. $\mu \leftarrow \mu + \tau_k \Sigma \nabla_\mu \text{LB}$.

Obviously, the traditional (Euclidean) gradient ascent is given by

1. $\mu \leftarrow \mu + \tau_k \nabla_\mu \text{LB}$
2. $\Sigma \leftarrow \Sigma + \tau_k \nabla_\Sigma \text{LB}$.

Let us consider a concrete example: Consider the log-linear model for counts, $y_i \sim \text{Poisson}(\delta_i), \delta_i = \exp(x_i^\top\theta)$ where $x_i, \theta \in \mathbb{R}^d$ are the vector of covariates and regression coefficients, respectively. Assume the prior of θ is $p(\theta) \sim \mathcal{N}(0, \sigma_0^2\mathbb{I}_d)$. We use a Gaussian $q_\lambda(\theta) = \mathcal{N}(\theta|\mu, \Sigma)$ to approximate the true posterior distribution of θ. Let $y = y_{1:n}$ and $X = (x_1, x_2, \ldots, x_n)^\top$. For each $i = 1, \ldots, n$ let $w_i = \exp(x_i^\top\mu + \frac{1}{2}x_i^\top\Sigma x_i)$ and $W = \text{diag}(w_1, \ldots, w_n)$. By computing directly Tan (2021) showed that the closed form for the ELBO is,

$$\begin{aligned}\text{LB}(\lambda) &= y^\top X\mu - \sum_{i=1}^n (w_i + \log(y_i!)) - \frac{\mu^\top\mu + Tr(\Sigma)}{2\sigma_0^2}\\ &\quad + \frac{1}{2}\log|\Sigma| + \frac{d}{2}(1 - \log(\sigma_0^2)).\end{aligned}$$

From here it is immediate to see that $\nabla_\mu \text{LB}$, $\nabla_\Sigma \text{LB}$, and $\nabla_{\text{vec}(\Sigma)}\text{LB}$ are given by

$$\nabla_\mu \text{LB} = X^\top(y - w) - \frac{\mu}{\sigma_0^2},$$

$$\nabla_\Sigma \text{LB} = -\frac{1}{2}X^\top WX - \frac{1}{2\sigma_0^2}\mathbb{I} + \frac{1}{2}\Sigma^{-1}$$

$$\nabla_{\text{vec}(\Sigma)}\text{LB} = \frac{1}{2}\text{vec}(\Sigma^{-1} - \frac{1}{\sigma_0^2}\mathbb{I} - X^\top WX).$$

Also we have

$$\begin{aligned}\log q_\lambda(\theta) &= -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma|\\ &\quad - \frac{1}{2}(\theta - \mu)^\top\Sigma^{-1}(\theta - \mu).\end{aligned}$$

Hence

$$\begin{aligned}\nabla_\lambda q_\lambda(\theta) &= (\nabla_\mu q_\lambda(\theta), \nabla_{\text{vec}(\Sigma)} q_\lambda(\theta))\\ &= \Bigg((\theta - \mu)^\top\Sigma^{-1}, \text{vec}\Big(-\frac{1}{2}\Sigma^{-1} + \frac{1}{2}\Sigma^{-1}\\ &\qquad (\theta - \mu)(\theta - \mu)^\top\Sigma^{-1}\Big)^\top\Bigg)^\top \in \mathbb{R}^{d+d^2}.\end{aligned}$$

We choose $n = 200, d = 3, \sigma_0^2 = 100, \theta = (1, \ldots, 1), x_{ij} \sim \mathcal{N}(0, 1)$, and simulate $y_1, \ldots, y_{200}$ from those parameters. In this case $\lambda \in \mathbb{R}^{12}$ hence $I_F \in \mathbb{R}^{12\times 12}$. The initial guesses are $\mu^{(0)} = (0, \ldots, 0)^\top$ and $\Sigma^{(0)} = 10^{-2}\mathbb{I}_d$. Moreover, we choose the step size $1/(1000+k)^{0.75}$ and $c_\beta = 1$. As we know the analytical form of the ELBO in this example, in Figure 3 we plot the graphs of $\text{LB}(\lambda)$ (as a function of the number of iterations) obtained from the exact natural gradient ascent algorithm (NGVB) , inversion free gradient algorithm (IFVB) and weighted inversion free algorithm (AIFVB). It can be seen that the three algorithms have almost identical performance in this case. One striking note is that AIFVB even outperforms (obtaining a larger lower bound value) the exact natural gradient algorithm in this case. It again helps to confirm that the averaging technique is useful. We note that we are tempted to include the ELBO obtained from the Euclidean gradient ascent but its performance is too unstable to include. This phenomenon has been observed in Tan (2021) as the matrix Σ obtained from the traditional gradient ascent is usually not positive definite, which results in an unstable performance.

Example 3 (Normalizing flow Variational Bayes). The choice of flexible variational distributions q_λ is important in VB. Normalizing flows (Rezende and Mohamed 2015) are a class of techniques for designing flexible and expressive q_λ. We consider a

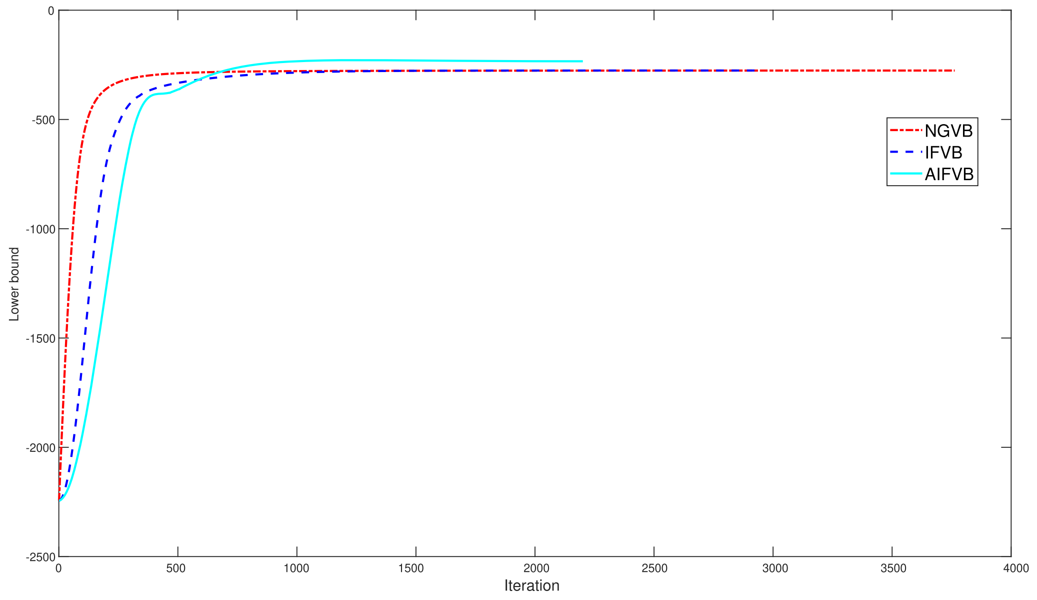

Figure 3. Gaussian Approximation: Comparing NGVB, IFVB, and AIFVB.

flexible VB framework where the VB approximation distribution $q_\lambda(\theta)$ is constructed based on a normalizing flow as follows

$$\epsilon \sim \mathcal{N}_d(0, I), \quad Z = \psi(W_1\epsilon + b_1), \quad \theta = W_2 Z + b_2 \qquad (6.1)$$

where $W_1, W_2 \in \text{Mat}(d, d)$ and b_1, b_2 are d-vectors, and $\psi(\cdot)$ is an activation function such as sigmoid. The transformation from ϵ to θ in (6.1) can be viewed as a neural network with one hidden layer Z; it might be desirable to consider deeper neural nets with more than one hidden layers, but we do not consider it here. To be able to compute the density $q_\lambda(\theta)$ resulting from the transformations in (6.1), these transformations should be invertibe and the determinants of the Jacobian matrices should be easy to compute. To this end, we impose the orthogonality constraint on W_1 and W_2: $W_1^\top W_1 = W_2^\top W_2 = \mathbb{I}_d$, that is the columns are orthonormal. Details on the derivation of the lower bound gradient and score function $\nabla \log q_\lambda(\theta)$ can be found in the Appendix, supplementary materials.

Numerical Results

We apply the manifold normalizing flow VB (NLVB) (6.1) to approximate the posterior distribution in a neural network classification problem, using the German Credit dataset. This dataset, available on the UCI Machine Learning Repository: *https://archive.ics.uci.edu/ml/index.php*, consists of observations on 1000 customers, each was already rated as being "good credit" (700 cases) or "bad credit" (300 cases). We create 10 predictors out of the available covariate variables including credit history, education, employment status, etc. The classification problem is based on a neural network with one hidden layer of 5 units. As W_1 and W_2 belong to the Stiefel manifold, for a comparison we use the VB on manifold algorithm of Tran, Nguyen, and Nguyen (2021) for updating these parameters. Figure 4 plots the lower bounds of the IFVB algorithm (solid red) together with the conventional VB algorithm (dash blue) using the Euclidean gradient. As shown, the IFVB algorithm converges quicker and achieves a higher lower bound. Note that we do not consider the AIFVB algorithm in this example, as the variational parameters

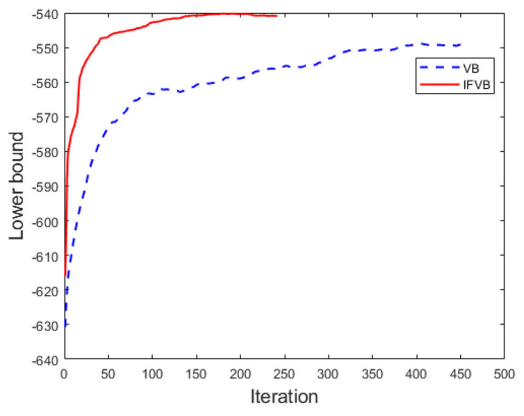

Figure 4. Normalizing flow VB: Lower bound values of Euclidean gradient VB (dash blue) versus IFVB (solid red) over the iterations.

belong to the Stiefel manifold, making the averaging technique challenging to use. It is interesting to extend the work to handle cases where the parameter space is a Riemannian manifold; however, we do not consider this in the present article.

Example 4 (Bayesian neural network). This example considers a Bayesian neural network for regression

$$y = \eta(x, \omega) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2), \qquad (6.2)$$

where y is a real-valued response variable, $\eta(x, \omega)$ denotes the output of a neural network with the input vector $x = (x_1, \dots, x_p)^\top \in \mathbb{R}^p$ and the vector of weights w. As neural networks are prone to overfitting, we follow Tran et al. (2020) and place a Bayesian adaptive group Lasso prior on the first-layer weights

$$w_{x_j}|\kappa_j \sim \mathcal{N}(0, \kappa_j \mathbb{I}_m),$$

$$\kappa_j|\gamma_j \sim \text{Gamma}\left(\frac{m+1}{2}, \frac{\gamma_j^2}{2}\right), \quad j = 1, \dots, p, \qquad (6.3)$$

with the $\gamma_j > 0$ the shrinkage parameters; no regularization prior is put on the rest of the network weights. Here w_{x_j} denotes

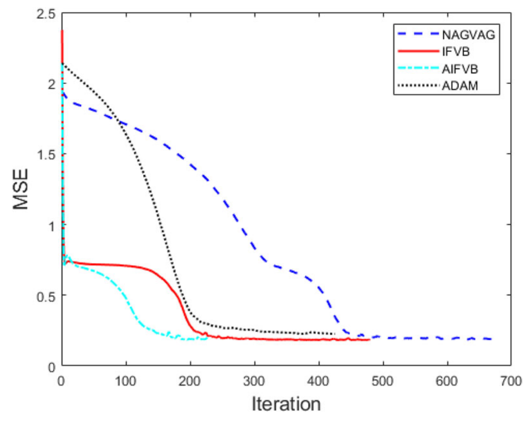

Figure 5. Bayesian neural network: Validation MSE values of ADAM (dotted black), AIFVB (dot greenish), IFVB (red solid) and NAGVAC (dash blue) over the iterations.

the vector of weights that connect the input x_j to the m units in the first hidden layer. An inverse-Gamma prior is used for σ^2. We use empirical Bayes for selecting the shrinkage parameters γ_j, and the posterior of κ_j and σ^2 is approximated by a fixed-form within mean-filed VB procedure. See Tran et al. (2020) for the details.

The main task is to approximate the posterior of the network weights ω. Let d be the dimension of ω. We choose to approximate this posterior by a Gaussian variational distribution of the form $q_\lambda(\omega) = \mathcal{N}(\mu, \Sigma)$ with covariance matrix Σ having a factor form $\Sigma = bb^\top + \text{diag}(c)^2$, where b and c are vectors in $\mathbb{R}^d$. The vector of the variational parameters is $\lambda = (\mu^\top, b^\top, c^\top)^\top$. This factor structure of the covariance matrix significantly reduces the size of variational parameters, making the Gaussian variational approximation method computationally efficient for Bayesian inference in large models such as Bayesian neural networks.

Tran et al. (2020) exploit the factor structure of Σ, and by setting certain sub-blocks of the Fisher information matrix $I_F(\lambda)$ to zero, to be able to derive a closed-form approximation of the inverse $I_F^{-1}(\lambda)$. Their VB method, termed the NAtural gradient Gaussian Variational Approximation with factor Covariance method (NAGVAC), is highly computationally efficient; however, the approximation of $I_F^{-1}(\lambda)$ might offset the VB approximation accuracy.

We now fit the Bayesian neural network model (6.2)–(6.3) to the Direct Marketing dataset (Jank 2011) that consists of 1000 observations, of which 800 were used for training, and the rest for validation. The response y is the amount (in \$1000) a customer spends on the company's products per year, and 11 covariates include gender, income, married status, etc. We use a neural network with two hidden layers, each with ten units. Figure 5 plots the mean squared error (MSE) values, computed on the validation set, of the VB training using ADAM (Kingma and Ba 2014), AIFVB, IFVB, and NAGVAC. With the same stopping rule, the four methods, ADAM, AIFVB, IFVB, and NAGVAC, stop after 425, 221, 480, and 700 iterations, respectively. This confirms the theoretical result in Theorem 5.3 that the averaging technique speeds up the convergence of AIFVB compared to IFVB. On the validation set, the smallest MSE values produced by ADAM, AIFVB, IFVB, and NAGVAC are 0.2273, 0.1750, 0.1749, and 0.1992, respectively. Both IFVB and AIFVB perform better than ADAM and NAGVAC, probably because the natural gradient approximation in NAGVAC, although being highly computationally efficient, might offset the approximation accuracy.

7. Conclusion

The article introduced an efficient approach for approximating the inverse of Fisher information, a crucial component in variational Bayes used for approximating posterior distributions. An outstanding feature of our algorithm is its avoidance of calculating the Fisher matrix and its inversion. Instead, our approach generates a sequence of matrices converging to the inverse of Fisher information. Implementation of our method for natural gradient estimate does not require storage of large matrices. Our inversion free VB framework showcases versatility, enabling its application in a wide range of domains, including Gaussian approximation and normalizing flow Variational Bayes, and makes the natural gradient VB method applicable in cases that were impossible before. To demonstrate the efficiency and reliability of the method, we provided numerical examples as evidence of its effectiveness. We find it intriguing to consider expanding the scope of our approach to scenarios where the variational parameter space is a Riemannian manifold and to develop a rigorous theoretical framework for such cases. We plan to explore this avenue in our future research studies.

Supplementary Materials

The online supplementary materials contain the proofs of the main theorems and an Appendix with further technical details.

Acknowledgments

We extend our gratitude to two anonymous reviewers for their insightful comments and suggestions. One of these comments was instrumental in deriving Remark 4.2.

Disclosure Statement

The authors report there are no competing interests to declare.

References

Agarwal, A., Wainwright, M. J., Bartlett, P., and Ravikumar, P. (2009), "Information-Theoretic Lower Bounds on the Oracle Complexity of Convex Optimization," in *Advances in Neural Information Processing Systems* (Vol. 22), pp. 1–9. [3]

Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural Computation*, 10, 251–276. [1,3]

Bercu, B., Godichon, A., and Portier, B. (2020), "An Efficient Stochastic Newton Algorithm for Parameter Estimation in Logistic Regressions," *SIAM Journal on Control and Optimization*, 58, 348–367. [3,4,6,7]

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017), "Variational Inference: A Review for Statisticians," *Journal of the American statistical Association*, 112, 859–877. [1,7]

Bottou, L., Curtis, F. E., and Nocedal, J. (2018), "Optimization Methods for Large-Scale Machine Learning," *Siam Review*, 60, 223–311. [3]

Boyer, C., and Godichon-Baggioni, A. (2023), "On the Asymptotic Rate of Convergence of Stochastic Newton Algorithms and their Weighted

Averaged Versions," *Computational Optimization and Applications*, 84, 921–972. [4,5,6]

Chau, H. N., Lars Kirkby, J., Nguyen, D. H., Nguyen, D., Nguyen, N. N., and Nguyen, T. (2024), "On the Inversion-Free Newton's Method and its Applications," *International Statistical Review*, 92, 284–321. [3]

Defazio, A., Bach, F., and Lacoste-Julien, S. (2014), "Saga: A Fast Incremental Gradient Method with Support for Non-strongly Convex Composite Objectives," in *Advances in Neural Information Processing Systems*, pp. 1646–1654. [1]

Duchi, J., Hazan, E., and Singer, Y. (2011), "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, 12, 2121–2159. [1,3]

Giordani, P., Mun, X., Tran, M.-N., and Kohn, R. (2013), "Flexible Multivariate Density Estimation with Marginal Adaptation," *Journal of Computational and Graphical Statistics*, 22, 814–829. [2]

Godichon-Baggioni, A., and Werge, N. (2023), "On Adaptive Stochastic Optimization for Streaming Data: A Newton's Method with o (dn) Operations," arXiv preprint arXiv:2311.17753. [7]

Goodfellow, I., Bengio, Y., and Courville, A. (2016), *Deep Learning*, Cambridge, MA: MIT Press. [1,5]

Graves, A. (2011), "Practical Variational Inference for Neural Networks," in *Advances in Neural Information Processing Systems* (Vol. 24). [1]

Gunawan, D., Kohn, R., and Nott,D. (2023), "Flexible Variational Bayes Based on a Copula of a Mixture," *Journal of Computational and Graphical Statistics*, 33, 665–680. [2]

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013), "Stochastic Variational Inference," *Journal of Machine Learning Research*, 14, 1303–1347. [1,2,3]

Jank, W. (2011), *Business Analytics for Managers*, New York: Springer-Verlag. [11]

Johnson, R., and Zhang, T. (2013), "Accelerating Stochastic Gradient Descent Using Predictive Variance Reduction," in *Advances in Neural Information Processing Systems* (Vol. 26), pp. 315–323. [1]

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. [1]

Khan, M., and Lin, W. (2017), "Conjugate-Computation Variational Inference: Converting Variational Inference in Non-conjugate Models to Inferences in Conjugate Models," in *Artificial Intelligence and Statistics*, pp. 878–887, PMLR. [1,2]

Kingma, D. P., and Ba, J. (2014), "Adam: A Method for Stochastic Optimization," arXiv preprint arXiv:1412.6980. [1,3,11]

Kingma, D. P., and Welling, M. (2013), "Auto-Encoding Variational Bayes," arXiv preprint arXiv:1312.6114. [1,3]

Kirkby, L. J., Nguyen, D. H., Nguyen, D., and Nguyen, N. N. (2022), "Inversion-Free Subsampling Newton's Method for Large Sample Logistic Regression," *Statistical Papers*, 63, 943–963. [3]

Kushner, H., and Yin, G. G. (2003), *Stochastic Approximation and Recursive Algorithms and Applications* (Vol. 35), New York: Springer. [1]

Longford, N. T. (1987), "A Fast Scoring Algorithm for Maximum Likelihood Estimation in Unbalanced Mixed Models with Nested Random Effects," *Biometrika*, 74, 817–827. [3]

Lopatnikova, A., and Tran, M.-N. (2023), "Quantum Variational Bayes on Manifolds," in *2023 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2023)*, pp. 1–5. [4]

Magris, M., Shabani, M., and Iosifidis, A. (2022), "Exact Manifold Gaussian Variational Bayes," arXiv preprint arXiv:2210.14598. [2,4]

Martens, J. (2020), "New Insights and Perspectives on the Natural Gradient Method," *The Journal of Machine Learning Research*, 21, 5776–5851. [1,3,4]

Martens, J., and Grosse, R. (2015), "Optimizing Neural Networks with Kronecker-Factored Approximate Curvature," in *International Conference on Machine Learning*, pp. 2408–2417, PMLR. [2]

Martin, G. M., Frazier, D. T., and Robert, C. P. (2023), "Approximating Bayes in the 21st Century," *Statistical Science*, 1, 1–26. [1]

Murphy, K. P. (2012), *Machine Learning: A Probabilistic Perspective*, Cambridge, MA: MIT Press. [1]

Nguyen, L. M., Liu, J., Scheinberg, K., and Takáč, M. (2017), "Sarah: A Novel Method for Machine Learning Problems Using Stochastic Recursive Gradient," in *International Conference on Machine Learning*, pp. 2613–2621, PMLR. [1]

Nguyen, L. M., Tran-Dinh, Q., Phan, D. T., Nguyen, P. H., and Van Dijk, M. (2021), "A Unified Convergence Analysis for Shuffling-Type Gradient Methods," *The Journal of Machine Learning Research*, 22, 9397–9440. [7]

Ong, V. M. G., Nott, D. J., Tran, M.-N., Sisson, S. A., and Drovandi, C. C. (2018), "Variational Bayes with Synthetic Likelihood," *Statistics and Computing*, 28, 971–988. [1]

Pelletier, M. (1998), "On the Almost Sure Asymptotic Behaviour of Stochastic Algorithms," *Stochastic Processes and their Applications*, 78, 217–244. [3]

Rattray, M., Saad, D., and Amari, S.-i. (1998), "Natural Gradient Descent for On-line Learning," *Physical Review Letters*, 81, 5461. [2]

Rezende, D., and Mohamed, S. (2015), "Variational Inference with Normalizing Flows," in *International Conference on Machine Learning*, pp. 1530–1538, PMLR. [2,9]

Robbins, H., and Monro, S. (1951), "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, 22, 400–407. [3]

Saad, D. (2009), *On-Line Learning in Neural Networks*, Cambridge: Cambridge University Press. [3]

Salimbeni, H., Eleftheriadis, S., and Hensman, J. (2018), "Natural Gradients in Practice: Non-Conjugate Variational Inference in Gaussian Process Models," in *International Conference on Artificial Intelligence and Statistics*, pp. 689–697, PMLR. [3]

Sato, M.-A. (2001), "Online Model Selection based on the Variational Bayes," *Neural Computation*, 13, 1649–1681. [3]

Spall, J. C. (2005), *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, Hoboken, NJ: Wiley. [3]

Tan, L. S. L. (2021), "Analytic Natural Gradient Updates for Cholesky Factor in Gaussian Variational Approximation," arXiv preprint arXiv:2109.00375. [2,3,4,6,8,9]

Tang, D., and Ranganath, R. (2019), "The Variational Predictive Natural Gradient," in *International Conference on Machine Learning*, pp. 6145–6154, PMLR. [6]

Titsias, M., Lázaro-Gredilla, M. (2014), "Doubly Stochastic Variational Bayes for Non-conjugate Inference," in *International Conference on Machine Learning*, pp. 1971–1979, PMLR. [3]

Tran, M.-N., Nguyen, N., Nott, D., and Kohn, R. (2020), "Bayesian Deep Net GLM and GLMM," *Journal of Computational and Graphical Statistics*, 29, 97–113. DOI: 10.1080/10618600.2019.1637747. [1,2,10,11]

Tran, M.-N., Nott, D. J., and Kohn, R. (2017), "Variational Bayes with Intractable Likelihood," *Journal of Computational and Graphical Statistics*, 26, 873–882. [2,3,7]

Tran, M.-N., Nguyen, D. H., and Nguyen, D. (2021), "Variational Bayes on Manifolds," *Statistics and Computing*, 31, 1–17. DOI: 10.1007/s11222-021-10047-1. [3,4,10]

Waterhouse, S., MacKay, D., and Robinson, A. (1995), "Bayesian Methods for Mixtures of Experts," in *Advances in Neural Information Processing Systems* (Vol. 8). [1]

Wilkinson, W. J., Särkkä, S., and Solin, A. (2023), "Bayes–Newton Methods for Approximate Bayesian Inference with PSD Guarantees," *Journal of Machine Learning Research*, 24, 1–50. [2]

Zeiler, M D. (2012), "Adadelta: An Adaptive Learning Rate Method," arXiv preprint arXiv:1212.5701. [1,3]