THE UNIVERSITY OF SYDNEY

The University of Sydney Business School
The University of Sydney

BUSINESS ANALYTICS WORKING PAPER SERIES

Random Effects Models with Deep Neural Network Basis Functions: Methodology and Computation

Minh-Ngoc Tran, Nghia Nguyen, David J. Nott and Robert Kohn

Abstract

Deep neural networks (DNNs) are a powerful tool for functional approximation. We describe flexible versions of generalized linear and generalized linear mixed models incorporating basis functions formed by a deep neural network. The consideration of neural networks with random effects seems little used in the literature, perhaps because of the computational challenges of incorporating subject specific parameters into already complex models. Efficient computational methods for Bayesian inference are developed based on Gaussian variational approximation methods. A parsimonious but flexible factor parametrization of the covariance matrix is used in the Gaussian variational approximation. We implement natural gradient methods for the optimization, exploiting the factor structure of the variational covariance matrix to perform fast matrix vector multiplications in iterative conjugate gradient linear solvers in natural gradient computations. The method can be implemented in high dimensions, and the use of the natural gradient allows faster and more stable convergence of the variational algorithm. In the case of random effects, we compute unbiased estimates of the gradient of the lower bound in the model with the random effects integrated out by making use of Fisher's identity. The proposed methods are illustrated in several examples for DNN random effects models and high-dimensional logistic regression with sparse signal shrinkage priors.

Keywords. Factor models; Reparametrization gradient; Stochastic optimization; Variational approximation.

February 2018

BA Working Paper No: BAWP-2018-01
http://sydney.edu.au/business/business_analytics/research/working_papers

Random effects models with deep neural network basis functions: methodology and computation

Minh-Ngoc Tran[*], Nghia Nguyen, David J. Nott and Robert Kohn

Abstract

Deep neural networks (DNNs) are a powerful tool for functional approximation. We describe flexible versions of generalized linear and generalized linear mixed models incorporating basis functions formed by a deep neural network. The consideration of neural networks with random effects seems little used in the literature, perhaps because of the computational challenges of incorporating subject specific parameters into already complex models. Efficient computational methods for Bayesian inference are developed based on Gaussian variational approximation methods. A parsimonious but flexible factor parametrization of the covariance matirx is used in the Gaussian variational approximation. We implement natural gradient methods for the optimization, exploiting the factor structure of the variational covariance matrix to perform fast matrix vector multiplications in iterative conjugate gradient linear solvers in natural gradient computations. The method can be implemented in high dimensions, and the use of the natural gradient allows faster and more stable convergence of the variational algorithm. In the case of random effects, we compute unbiased estimates of the gradient of the lower bound in the model with the random effects integrated out by making use of Fisher's identity. The proposed methods are illustrated in several examples for DNN random effects models and high-dimensional logistic regression with sparse signal shrinkage priors.

Keywords. Factor models; Reparametrization gradient; Stochastic optimization; Variational approximation.

1 Introduction

Deep neural network modeling provides a powerful technique for approximating multivariate functions, and has become increasingly popular recently. DNNs have been applied successfully in fields such as image processing, computer vision and language

[*]Corresponding author: minh-ngoc.tran@sydney.edu.au

recognition. See Schmidhuber (2015) for a historical survey of DNNs and Goodfellow et al. (2016) for a more comprehensive recent discussion. Polson and Sokolov (2017) provide a Bayesian perspective on DNN methodology and explain its connection to statistical techniques such as principal component analysis and reduced rank regression.

This paper considers generalized linear models (GLMs) and generalized linear mixed model (GLMMs) using DNNs as a way to efficiently transform a vector of p raw covariates $X = (X_1, ..., X_p)^\top$ into a new vector of m predictors Z in the model. We refer to these DNN-based versions of GLM and GLMM as flexible GLM (fGLM) and flexible GLMM (fGLMM), respectively. A conventional GLM uses a link function that links the conditional mean of the response variable Y to a linear combination of the predictors $Z = \phi(X) = (\phi_1(X), ..., \phi_m(X))^\top$, with each $\phi_j(X)$ a function of X. We refer to the original raw input variables X_j as covariates, and refer to the transformations $\phi_j(X)$ as predictors. Usually in conventional GLMs the $\phi_j(X)$ are chosen *a priori* in some way, but here we are concerned with learning an appropriate Z from data, and in the machine learning literature the predictors Z are commonly referred to as learned features. If a deep multi-layer perceptron is used for transforming the covariates, then Z has the form

$$Z = f_L\Big(W_L, f_{L-1}\big(W_{L-1}, \cdots f_1(W_1, X)\cdots\big)\Big). \qquad (1)$$

which is often graphically represented by a network as in Figure 1. Each vector-valued

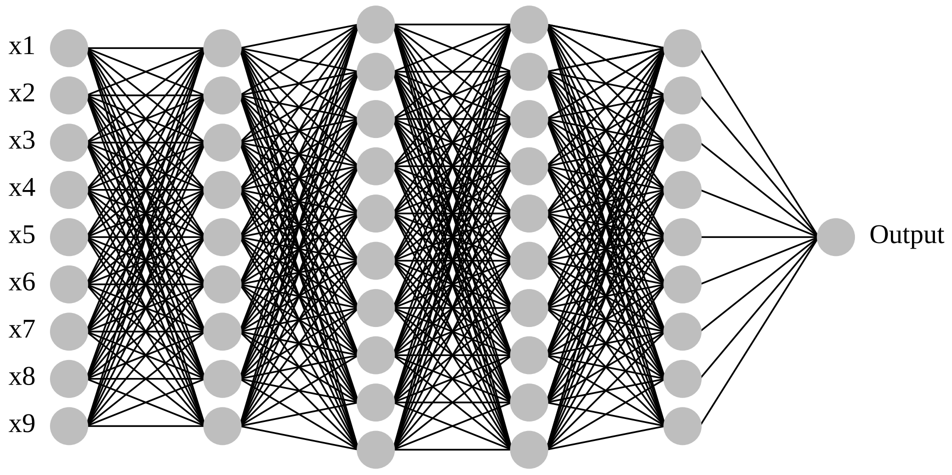

Figure 1: Graphical representation of a layered composition function with $L = 4$ hidden layers. The input layer represents 9 raw covariates X. The last hidden layer (hidden layer 4) represents the predictors Z.

function $Z_j = f_j(W_j, Z_{j-1})$, $j = 1, 2, ...$ is called a hidden layer, L is the number of hidden layers in the network, $w=(W_1,...,W_L)$ is the set of weights, and we define $Z_0=X$. The function $f_j(W_j,Z_{j-1})$ is assumed to be of the form $h_j(W_jZ_{j-1})$ where W_j is a matrix of weights for layer j and $h_j(\cdot)$ is a scalar function, called the activation function. Applying h_j to a vector should be understood component-wise. For a discussion of alternative kinds of architectures for deep neural networks see for example Goodfellow et al. (2016). The architecture (1) provides a powerful way to transform the raw covariate data X into summary statistics Z that have some desirable properties. In the fGLM, we link the conditional mean of the response $\xi=\mathbb{E}(Y|X)$ to a linear combination of Z

$$g(\xi)=\beta_0+Z^\top\widetilde{\beta}.$$

and the model parameters consist of w, β and other possible parameters such as any dispersion parameters. Section 2.2 defines fGLMM models by adding random effects to fGLM models. Such models are designed to model within-subject dependence. The use of neural network basis functions in the context of mixed effects models seems not considered much in the literature. Lai et al. (2006) is the earliest work that we know of that deals with a neural network basis and random effects, in the context of modelling pharmacokinetic data. They consider neural networks with only one hidden layer, however.

Estimating complex and high-dimensional models like fGLM and fGLMM is challenging. This article develops Bayesian inference based on variational approximation, which provides an approach to approximate Bayesian inference that is useful for many modern applications involving complex models and large datasets. When the approximation of the posterior distribution is Gaussian, parsimonious but flexible methods for parametrizing the covariance matrix are needed if the approach is to be useful for problems with a high-dimensional parameter. Here we consider factor parametrizations (Bartholomew et al., 2011) which are often effective for describing dependence in high dimensional settings. We discuss efficient methods for performing the variational optimization in this context using the natural gradient (Amari, 1998) by leveraging the factor structure and using iterative conjugate gradient methods for solving large linear systems. In the "isotropic" case with a one-factor decomposition and constant variance for the idiosyncratic noise components, we show that the natural gradient can be computed analytically and efficiently without iterative conjugate gradient methods which leads to a particularly simple Gaussian variational method for fitting high-dimensional models that can often be adequate for predictive inference.

Ong et al. (2017a) recently considered Gaussian variational approximation with a factor covariance structure using stochastic gradient approaches to the optimization. Their approach uses the so-called reparametrization trick (Kingma and Welling, 2013; Rezende et al., 2014) and its modification by Roeder et al. (2017) for estimating gradients of the variational objective. However, for certain models where the variational objective is very challenging to optimize, first-order optimization methods such as those considered in Ong et al. (2017a) may be very slow to converge. Ong et al. (2017b) earlier considered natural gradient methods for Gaussian variational approximations using a factor covariance structure. However, their work was in the context of likelihood-free inference methods with a parameter of dimension at most a few hundreds, and the approach they develop does not scale to larger problems. The current work shows how the natural gradient Gaussian variational approximation with factor covariance can be implemented even in very high dimensions.

Much of the recent literature relevant to our training method occurs in the field of deep learning. Martens (2010) considers second-order optimization methods in deep learning and describes Hessian-free optimization methods, adapting a long history of related methods in the numerical analysis literature to that context. A detailed discussion of the connections between natural gradient methods and second order optimization methods is given in Martens (2014). Pascanu and Bengio (2014) considers the use of the natural gradient in deep learning problems and its connections with Hessian-free optimization, but like Martens (2010) their work is not specifically concerned with variational objectives. Fan et al. (2015) consider how to implement Hessian-free methods for the case of a variational objective function and Gaussian approximating family. Similarly to our development here, they consider using conjugate gradient linear solvers as an efficient solution to the difficult matrix calculations that occur in a naive formulation of second-order methods. By a reparametrization approach they are able to obtain estimates of Hessian vector products. They do not consider factor parametrizations of the covariance structure, however. Here we leverage the factor covariance structure to calculate the matrix vector products we need directly, without the need to store large matrices. Recently Regier et al. (2017) consider a second order trust region method for black box variational inference. They show that while their approach may be more expensive per iteration than common first-order methods for variational Gaussian approximation such as those implemented in Titsias and Lázaro-Gredilla (2014) and Kucukelbir et al. (2017), it reduces total computation time and provably converges to a stationary point. It may be useful to consider how to exploit a factor parametrization of the covariance structure in the framework they develop, but this is not considered here.

We illustrate the DNN-based flexible models and the training method in a range of experimental studies and applications, with a focus on models involving complex hierarchically structured priors and datasets of only moderate size where quantification of uncertainty is important. Many successful applications of DNN are in image processing and speech recognition, where the datasets are large and have some special domain-application characteristics such as association between the local pixels in an image. Skepticism has sometimes been expressed about whether DNN methods are useful for applications involving limited data where uncertainty quantificaiton is the focus. The main conclusion in our examples is that the DNN-based regression models fGLM and fGLMM perform very well in terms of prediction accuracy, provided appropriate attention is paid to regularization methods, prior distributions and computational algorithms.

The next section describes the two new classes of flexible statistical models fGLM and fGLMM. Section 3 describes the natural gradient Gaussian variational approximation method, Section 4 highlights advantages of our Bayesian treatment. Section 5 presents experimental studies and applications. Section 6 concludes. The Appendix gives details of the VB natural gradient computation and more details of an example.

2 Flexible regression models with DNN

2.1 Flexible generalized linear models

Consider a dataset $D=\{(y_i,x_i),i=1,...,n\}$ with y_i the response and $x_i=(x_{i1},...,x_{ip})^\top$ the vector of p covariates. We also use y and x to denote a generic response and a covariate vector, respectively. Consider a neural net with the input vector x and a scalar output as represented in Figure 1. Denote by $z_j=\phi_j(x,w)$, $j=1,...,m$, the units in the last hidden layer, where w is the weights up to the last hidden layer, and $\beta=(\beta_0,\beta_1,...,\beta_m)^\top$ are the weights that connect the z_j to the output

$$\mathfrak{N}(x,w,\beta)=\beta_0+\beta_1 z_1+...+\beta_m z_m=\beta_0+\widetilde{\beta}^\top z$$

with $\widetilde{\beta}=(\beta_1,...,\beta_m)^\top$ and $z=(z_1,...,z_m)^\top$.

We assume that the conditional density $p(y|x)$ has an exponential family form

$$p(y|x) = \exp\left(\frac{y\varpi - b(\varpi)}{\phi} + c(\phi, y)\right) \tag{2}$$

with the canonical parameter ϖ and the dispersion parameter ϕ. Let $g(\cdot)$ be the link function that links the conditional mean $\xi=\xi(x)=\mathbb{E}(y|x)$ to a function of the covariates x. In the conventional GLM, $g(\xi)$ is assumed to be a linear combination of x. In order to achieve flexibility and capture possible non-linear effects that x has on ξ, we propose the more flexible model

$$g(\xi) = \beta_0 + \widetilde{\beta}^\top z = \mathfrak{N}(x, w, \beta). \tag{3}$$

In our later examples we use the canonical link function where $\varpi=g(\xi)$, which is the log link for Poisson responses and the logit link for binomial responses. The predictors (learned features) z_j efficiently capture the important non-linear effects of the original raw covariates x on $g(\xi)$.

The model (3) is flexible, but can be hard to interpret. It may be useful in this respect to introduce additional structure. We might partition the covariates as $x=(x^{(1)^T},x^{(2)^T})^T$ so that $x^{(1)}$ and $x^{(2)}$ have non-linear and linear effects respectively on $g(\xi)$, so that

$$g(\xi) \;=\; \mathfrak{N}(x^{(1)}, w, \beta^{(1)}) + {\beta^{(2)}}^\top x^{(2)} \tag{4}$$

where $\beta^{(1)}$ parametrizes a nonlinear effect w.r.t. the covariates $x^{(1)}$ and $\beta^{(2)}$ parametrizes the linear effects w.r.t. the covariates $x^{(2)}$. Write $\beta=(\beta^{(1)},\beta^{(2)})$. We refer to the general regression model with the exponential family form response distribution (2) and the mean model (3) or (4) as flexible GLM (fGLM).

The vector of model parameters θ consists of w, β and possibly dispersion parameters ϕ if ϕ is unknown. The density $p(y|x)$ in (2) is now a function of θ: $p(y|x)=p(y|x,\theta)$. Given a dataset D, the likelihood function is

$$L(\theta) = \prod_{i=1}^{n} p(y_i|x_i,\theta), \tag{5}$$

and likelihood-based inference methods, including Bayesian methods, are easily applied.

2.2 Flexible generalized linear mixed models

Consider a panel dataset $D=\{(y_{it},x_{it}),t=1,...,T_i,i=1,...,n\}$ with y_{it} the response and x_{it} the vector of covariates of subject i at time t. Generalized linear mixed models (GLMM) use random effects to account for within-subject dependence. Let $z_{it,j}=\phi_j(x_{it},w)$, $j=1,...,m$, be the units in the last hidden layer of a neural net, $z_{it}=(z_{it,1},...,z_{it,m})^\top$. Similarly to GLMM, to account for within-subject dependence, we propose to link the conditional mean of y_{it}, given x_{it}, to the predictors $z_{it,j}$ as follows

$$\begin{aligned} g(\xi_{it}) &= \beta_0+\alpha_{i0}+(\beta_1+\alpha_{i1})z_{it,1}+...+(\beta_m+\alpha_{im})z_{it,m} \\ &= \mathfrak{N}(x_{it},w,\beta+\alpha_i) \end{aligned} \tag{6}$$

where $\alpha_i=(\alpha_{i0},...,\alpha_{im})^\top$ are random effects that reflect the characteristics of subject i. The variation between the subjects is captured in the distribution of α_i. Our paper assumes $\alpha_i\sim\mathcal{N}(0,\Gamma)$ but more flexible distributional specifications for the random effects can also be considered.

Similarly to the previous section, a more interpretable model can be developed if some additional structure is assumed. Partition the covariates x_{it} as $(x_{it}^{(1)^T},x_{it}^{(2)^T})^T$ where $x_{it}^{(1)}$ and $x_{it}^{(2)}$ have non-linear and linear effects respectively on $g(\xi_{it})$:

$$g(\xi_{it}) = \mathfrak{N}(x_{it}^{(1)},w,\beta^{(1)})+(\alpha_i+\beta^{(2)})^\top x_{it}^{(2)} \tag{7}$$

where $\beta^{(1)}$ parametrizes fixed non-linear effects for $x_{it}^{(1)}$, and $\beta^{(2)}$ and α_i are fixed and random linear effects w.r.t. the covariates $x_{it}^{(2)}$. Write $\beta=(\beta^{(1)},\beta^{(2)})$. The model parameters θ include w, β, Γ and any dispersion parameters ϕ if unknown. We refer to the panel data model with the distribution (2) and the link (6) or (7) as flexible GLMM (fGLMM).

The likelihood for the fGLMM is

$$L(\theta) = \prod_{i=1}^{n} L_i(\theta) \tag{8}$$

with the likelihood contribution

$$\begin{aligned} L_i(\theta) = p(y_i|x_i,\theta) &= \int p(y_i|x_i, w, \beta, \phi, \alpha_i) p(\alpha_i|\Gamma) d\alpha_i \\ &= \int \prod_{t=1}^{T_i} p(y_{it}|z_{it}, \beta, \phi, \alpha_i) p(\alpha_i|\Gamma) d\alpha_i. \end{aligned} \tag{9}$$

Section 3 describes a VB algorithm for fitting the fGLMM.

The likelihood for the fGLMM described in (8) and (9) is intractable, because the integral in (9) cannot be computed analytically, except for the case where the conditional distribution of y_{it} given z_{it} is normal. However, we can estimate each likelihood contribution $L_i(\theta)$ unbiasedly using importance sampling, and this allows estimation methods for intractable likelihoods, such as the block pseudo-marginal MCMC of Tran et al. (2016) or the VB method of Tran et al. (2017), to be used. We consider here an alternative VB method based on the reparametrization trick in Section 3.1, which utilizes the information of the gradient of the log-likelihood computed by the back-propagation algorithm. By Fisher's identity (Gunawan et al., 2017), the gradient of the log-likelihood contribution $\nabla_\theta \ell_i(\theta)$, $\ell_i(\theta) = \log L_i(\theta)$ with $L_i(\theta)$ in (9), is

$$\nabla_\theta \ell_i(\theta) = \int \nabla_\theta \left(\log \prod_{t=i}^{T_i} p(y_{it}|z_{it}, \beta, \phi, \alpha_i) p(\alpha_i|\Gamma) \right) p(\alpha_i|\theta, y_i, x_i) d\alpha_i, \tag{10}$$

where $p(\alpha_i|\theta,y_i,x_i)$ is the conditional distribution of the random effects α_i given data (y_i,x_i) and θ. The gradient inside the integral (10) can be computed by back-propagation, and then the integral can be estimated easily by importance sampling.

3 Gaussian variational approximation with factor covariance structure

This section describes the natural gradient Gaussian variational approximation method. Let D be the data and $\theta \in \Theta$ the vector of unknown parameters. Bayesian inference about θ is based on the posterior distribution with density function

$$\pi(\theta) = p(\theta|D) = \frac{p(\theta)L(\theta)}{p(D)}$$

with $p(\theta)$ the prior, $L(\theta)=p(D|\theta)$ the likelihood function and $p(D)=\int p(\theta)L(\theta)d\theta$ the marginal likelihood. In all but a few simple cases the posterior $\pi(\theta)$ is unknown, partly because $p(D)$ is unknown, which makes it challenging to carry out Bayesian inference.

In this work, we are interested in variational approximation methods, which are widely used as a scalable and computationally effective method for Bayesian computation (Bishop, 2006; Blei et al., 2017). We will approximate the posterior $\pi(\theta)$ by a distribution with density $q_\lambda(\theta)$ within some family of standard distributions. For example, if $q_\lambda(\theta)=\mathcal{N}(\theta;\mu,\Sigma)$, the density of a multivariate normal distribution with mean vector μ and covariance matrix Σ, then $\lambda=(\mu,\Sigma)$. The best λ is chosen by minimizing the Kullback-Leibler divergence between $q_\lambda(\theta)$ and $\pi(\theta)$

$$\begin{aligned}\mathrm{KL}(\lambda) &= \int q_\lambda(\theta)\log\frac{q_\lambda(\theta)}{\pi(\theta)}d\theta = \int q_\lambda(\theta)\log\frac{q_\lambda(\theta)}{p(\theta)L(\theta)}d\theta + \log p(D) \\ &= -\mathrm{LB}(\lambda) + \log p(D), \end{aligned} \tag{11}$$

where

$$\mathrm{LB}(\lambda)=\int q_\lambda(\theta)\log\frac{p(\theta)L(\theta)}{q_\lambda(\theta)}d\theta$$

is a lower bound on $\log p(D)$. Minimizing $\mathrm{KL}(\lambda)$ is therefore equivalent to maximizing the lower bound $\mathrm{LB}(\lambda)$. If we can obtain an unbiased estimator $\widehat{\nabla_\lambda\mathrm{LB}(\lambda)}$ of the gradient of the lower bound, then we can use stochastic optimization to maximize $\mathrm{LB}(\lambda)$, as in Algorithm 1 below.

Algorithm 1.
- *Initialize $\lambda^{(0)}$ and stop the following iteration if the stopping criterion is met.*
- *For $t=0,1,...,$ compute $\lambda^{(t+1)}=\lambda^{(t)}+a_t\widehat{\nabla_\lambda\mathrm{LB}}(\lambda^{(t)})$.*

The learning rate sequence $\{a_t\}$ in Algorithm 1 should satisfy the Robbins-Monro conditions, $a_t>0$, $\sum_t a_t=\infty$ and $\sum_t a_t^2<\infty$ (Robbins and Monro, 1951). Choice of a_t is discussed later on in some detail.

3.1 Reparametrization trick

As is typical of stochastic optimization algorithms, the performance of Algorithm 1 depends greatly on the variance of the noisy gradient. Techniques for variance reduction are needed. We will use the so-called reparametrization trick (Kingma and Welling, 2013; Rezende et al., 2014) in this paper, and its modification by Roeder et al. (2017), who generalized ideas considered in Han et al. (2016) and Tan and Nott (2017).

Suppose that for $\theta\sim q_\lambda(\cdot)$, there exists a deterministic function $g(\lambda,\epsilon)$ such that $\theta=g(\lambda,\epsilon)\sim q_\lambda(\theta)$ where $\epsilon\sim p_\epsilon(\epsilon)$, which is independent of λ. For example, if $q_\lambda(\theta)=$

$\mathcal{N}(\theta;\mu,\Sigma)$ then $\theta=\mu+\Sigma^{1/2}\epsilon$ with $\epsilon\sim\mathcal{N}(0,I)$ and I the identity matrix. Writing $\text{LB}(\lambda)$ as an expectation with respect to $p(\epsilon)$ gives

$$\text{LB}(\lambda)=E_\epsilon\Big(h(g(\epsilon,\lambda))-\log q_\lambda(g(\epsilon,\lambda))\Big),$$

where $E_\epsilon(\cdot)$ denotes expectation with respect to $p_\epsilon(\epsilon)$, $h(\theta):=\log(p(\theta)L(\theta))$. Differentiating under the integral sign and simplifying as in Roeder et al. (2017) gives

$$\nabla_\lambda\text{LB}(\lambda)=E_\epsilon\Big(\nabla_\lambda g(\lambda,\epsilon)\nabla_\theta\{h(g(\epsilon,\lambda))-\log q_\lambda(g(\epsilon,\lambda))\}\Big). \tag{12}$$

The gradient (12) can be estimated unbiasedly using i.i.d samples $\epsilon_s\sim p_\epsilon(\cdot)$, $s=1,...,S$, as

$$\widehat{\nabla_\lambda\text{LB}}(\lambda)=\frac{1}{S}\sum_{s=1}^{S}\nabla_\lambda g(\lambda,\epsilon_s)\nabla_\theta\big\{h(g(\lambda,\epsilon_s))-\log q_\lambda(g(\lambda,\epsilon_s))\big\}. \tag{13}$$

The gradient estimator (13) has the advantage that if the variational family is rich enough to contain the exact posterior, so that $\exp(h(\theta))\propto q_\lambda(\theta)$ at the optimal λ, then the estimator (13) is exactly zero at this optimal value even for $s=1$ where we use just a single Monte Carlo sample from $p_\epsilon(\epsilon)$. Reparametrized gradient estimators are much more efficient than alternative approaches to estimating the lower bound gradient, partly because they take into account information from $\nabla_\theta h(\theta)$. For further discussion we refer the reader to Roeder et al. (2017).

3.2 Natural gradient

It is well-known that the ordinary gradient $\nabla_\lambda\text{LB}(\lambda)$ does not adequately capture the geometry of the approximating family $q_\lambda(\theta)$ (Amari, 1998). A small Euclidean distance between λ and λ' does not necessarily mean a small Kullback-Leibler divergence between $q_\lambda(\theta)$ and $q_{\lambda'}(\theta)$. Rao (1945) was the first to point out the importance of using the geometrical information on the manifold of a statistical model, and introduced the Riemannian metric on this manifold induced by the Fisher information matrix. Amari (1998) shows that the steepest direction for optimizing the objective function $\text{LB}(\lambda)$ on the manifold formed by the family $q_\lambda(\theta)$ is directed by the so-called natural gradient which is defined by pre-multiplying the ordinary gradient with the inverse of the Fisher information matrix

$$\nabla_\lambda\text{LB}(\lambda)^{\text{natural}}=I_F^{-1}(\lambda)\nabla_\lambda\text{LB}(\lambda), \tag{14}$$

with $I_F(\lambda)=\text{cov}_{q_\lambda}(\nabla_\lambda\log q_\lambda(\theta))$.

The use of the natural gradient in VB algorithms is considered, among others, by Sato (2001), Honkela et al. (2010), Hoffman et al. (2013), Salimans and Knowles (2013) and Tran et al. (2017). A simple demonstration of the importance of the natural gradient can be found in Tran et al. (2017). The use of the natural gradient in deep learning problems is considered in Pascanu and Bengio (2014), who show the connection between natural gradient descent and other second-order optimization methods such as Hessian-free optimization.

The main difficulty of using the natural gradient is the computation of $I_F(\lambda)$, and the solution of linear systems involving this matrix, which is required to compute (14). The problem is more severe in high dimensional models because this matrix often has a large size. Some approximation methods, such as the truncated Newton approach, may be needed (Pascanu and Bengio, 2014). We consider in the next section an efficient method for computing $I_F(\lambda)^{-1}\nabla_\lambda \text{LB}(\lambda)$ based on the use of iterative conjugate gradient methods for solving linear systems when the covariance matrix of the Gaussian variational approximation is parametrized by a factor model. We compute (14) by solving the linear system $I_F(\lambda)x=\nabla_\lambda \text{LB}(\lambda)$ for x using only matrix-vector products involving $I_F(\lambda)$ where the matrix vector products can be done efficiently both in terms of computation time and memory requirements by using the factor structure of the variational covariance matrix. In the special case of one factor with isotropic idiosyncratic noise the natural gradient in (14) can be computed analytically and efficiently.

3.3 Gaussian variational approximation with factor covariance

We now describe in detail the Gaussian variational approximation with factor covariance (VAFC) method of Ong et al. (2017a). The VAFC method considers the multivariate normal variational family $q_\lambda(\theta)=N(\mu,\Sigma)$ where Σ is parametrized as

$$\Sigma = BB^\top + D^2. \tag{15}$$

Here B is a $d\times f$ matrix (the factor loadings matrix) with d the dimension of θ and f the number of factors, $f \ll d$, and D is diagonal with diagonal entries $\Delta=(\Delta_1,\ldots,\Delta_d)^\top$. Δ is a vector of idiosyncratic noise standard deviations. Factor structures are well known to provide useful parsimonious representations of dependence in high-dimensional settings (Bartholomew et al., 2011). We assume B is lower triangular, i.e., $B_{ij}=0$ for $j>i$. Although imposing the constraint $B_{ii}>0$ makes the factor representation identifiable (Geweke and Zhou, 1996), we do not impose this constraint to simplify the optimization. The variational optimization simply locks onto one of the equivalent modes. An intuitive generative representation of the factor structure that is the basis of our application of the reparametrization trick is the following: if we consider $\theta\sim q_\lambda(\theta)=N(\mu,BB^\top+D^2)$, then we can represent θ as $\theta=\mu+B\epsilon_1+\Delta\circ\epsilon_2$ where $\epsilon=(\epsilon_1^\top,\epsilon_2^\top)^\top\sim N(0,I)$, ϵ_1 and ϵ_2 have dimensions f and d respectively, and $\circ$ denotes the Hadamard (element by element) product for two matrices of the same size. We can see from this representation that the latent variables ϵ_1 (the "factors", which are low-dimensional) explain all the correlation between the components, whereas component-specific idiosyncratic variance is being captured through ϵ_2.

The variational parameters are $\lambda = (\mu^\top, \text{vec}(B)^\top, \Delta^\top)^\top$ where we have written vec(B) for the vectorization of B obtained by stacking its columns from left to right. Ong et al. (2017a) show that the gradient of the lower bound takes the form

$$\nabla_\mu \text{LB}(\lambda) = E_\epsilon\Big(\nabla_\theta h(\mu + B\epsilon_1 + \Delta\circ\epsilon_2) + (BB^\top + D^2)^{-1}(B\epsilon_1 + \Delta\circ\epsilon_2)\Big), \qquad (16)$$

$$\nabla_B \text{LB}(\lambda) = E_\epsilon\Big(\nabla_\theta h(\mu + B\epsilon_1 + \Delta\circ\epsilon_2)\epsilon_1^\top + (BB^\top + D^2)^{-1}(B\epsilon_1 + \Delta\circ\epsilon_2)\epsilon_1^\top\Big), \qquad (17)$$

and

$$\nabla_\Delta \text{LB}(\lambda) = E_\epsilon\Big(\text{diag}\big(\nabla_\theta h(\mu + B\epsilon_1 + \Delta\circ\epsilon_2)\epsilon_2^\top + (BB^\top + D^2)^{-1}(B\epsilon_1 + \Delta\circ\epsilon_2)\epsilon_2^\top\big)\Big), \quad (18)$$

where we have written diag(A) for the diagonal elements of a square matrix A. Here, the inverse matrix $(BB^\top + D^2)^{-1}$ can be computed efficiently; see (19). We note that in the expression for the gradient of B above, we should set to zero the upper triangular components which correspond to elements of B which are fixed at zero. Unbiased estimation of gradients for stochastic gradient ascent can proceed based on these expressions by drawing one or more samples from $p_\epsilon(\epsilon)$ to estimate the expectations.

3.4 Efficient natural gradient VAFC method

We now describe how to efficiently implement natural gradient optimization for the VAFC method. Ong et al. (2017b) have also considered a natural gradient method for Gaussian variational approximation with factor covariance. However, this was in the context of likelihood-free inference methods where the dimension of θ is low compared to the models of interest here, and they simply used naive methods for solving the linear systems involving $I_F(\lambda)$ required to compute the natural gradient. This is impractical in high-dimensional problems, and here we demonstrate how to implement natural gradient VAFC when θ is high-dimensional using conjugate gradient methods (see, for example, Stoer (1983)).

Write $I_F(\lambda)$ in partitioned form as

$$I_F(\lambda) = \begin{bmatrix} I_{11} & I_{21}^\top & I_{31}^\top \\ I_{21} & I_{22} & I_{32}^\top \\ I_{31} & I_{32} & I_{33} \end{bmatrix},$$

where the blocks in the partition follow the partition of λ as $\lambda = (\mu^\top, \text{vec}(B)^\top, \Delta^\top)^\top$. Because the upper triangle of B is fixed at zero the corresponding rows and columns of $I_F(\lambda)$ should be omitted. Ong et al. (2017b) show that $I_{11} = \Sigma^{-1}$, $I_{21} = I_{31} = 0$, $I_{22} =$

$2(B^\top\Sigma^{-1}B\otimes\Sigma^{-1})$ (where $\otimes$ denotes the Kronecker product), $I_{33}=2(D\Sigma^{-1})\circ(\Sigma^{-1}D)$ and $I_{32}=2(B^\top\Sigma^{-1}D\otimes\Sigma^{-1})E_d^\top$ where E_d is the $d\times d^2$ matrix that picks out the diagonal elements of the $d\times d$ matrix A from its vectorization, so that $E_d\text{vec}(A)=\text{diag}(A)$. To use a conjugate gradient linear solver to compute $I_F(\lambda)^{-1}\nabla_\lambda\text{LB}(\lambda)$ we simply need to be able to compute matrix vector products of the form $I_F(\lambda)x$ for any vector x quickly without requiring to store the elements of $I_F(\lambda)$.

This can be done provided we can do matrix vector products for the matrices I_{11}, I_{22}, I_{33} and I_{32}. This is still difficult so we further approximate $I_F(\lambda)$ by setting $I_{32}=0$ and replacing I_{33} with $\tilde{I}_{33}=2(D\tilde{\Sigma}^{-1})\circ(\tilde{\Sigma}^{-1}D)$ where $\tilde{\Sigma}$ is the diagonal approximation to Σ obtained by setting its off-diagonal elements to zero. Using this approximation and $I_{32}=0$ we obtain a positive definite approximation $\tilde{I}_F(\lambda)$ to $I_F(\lambda)$ which we use instead of $I_F(\lambda)$ in the natural gradient.

Multiplications involving $\tilde{I}_{33}$ are simple since this matrix is diagonal, but we still need efficient methods to compute matrix vector products for I_{11} and I_{22}. Considering I_{11} first, we note that by using the Woodbury formula

$$I_{11}=\Sigma^{-1}=D^{-2}-D^{-2}B(I+B^\top D^{-1}B)^{-1}B^\top D^{-2}, \tag{19}$$

and then noting that D is diagonal, and $(I+B^\top D^{-2}B)$ is $f\times f$, $f\ll d$, we can calculate $I_{11}x=\Sigma^{-1}x$ without needing to store any $d\times d$ matrix or do any dense $d\times d$ matrix multiplications. Next, consider $I_{22}x$ for some vector x. We note that

$$I_{22}=2(B^\top\Sigma^{-1}B\otimes\Sigma^{-1})x=2\text{vec}(\Sigma^{-1}x^*B^\top\Sigma^{-1}B)$$

where x^* denotes the $d\times f$ matrix such that $x=\text{vec}(x^*)$ and where we have used the identity that for conformable matrices X,Y,Z, $\text{vec}(XYZ)=(Z^\top\otimes X)\text{vec}(Y)$. Then $\Sigma^{-1}x^*$ is easily computed efficiently by the Woodbury formula, and similarly for $B^\top\Sigma^{-1}B$.

3.4.1 Special case of isotropy factor decomposition

We now consider the special case where the covariance matrix Σ is parameterized as $\Sigma=BB^\top+c^2I_d$ with B a vector and c a constant. In this case, the natural gradient (14) can be computed in closed form and the computational complexity is $O(d)$. The detailed derivation can be found in the Appendix. This estimation method is computationally attractive, especially for cases where the dimension d is extremely large. Our experimental studies in Section 5 suggest that in some applications this method is able to produce a prediction accuracy comparable to the accuracy obtained by methods that use more flexible factor decomposition structures of Σ. A recent discussion of the phenomenon of richer variational families producing inferior performance in terms of predictive loss for neural networks models is given in Trippe and Turner (2018), with some theoretical insights into this phenomenon.

4 Practical recommendations in training DNN models

Our natural gradient Gaussian variational approximation method can be used as a general estimation method for any model. However, this section focuses on estimating DL-based models, and discusses some implementation recommendations that we found useful in practice.

4.1 Bayesian treatment

Selecting the tuning parameters in DNN models can be challenging and we now develop some Bayes and empirical Bayes methods for this task for the models we consider. In fGLM and fGLMM we use ridge-type regularization priors, one for the inner weights w with shrinkage hyperparameter γ_w, and one for the output weights β with shrinkage hyperparameter γ_β. Adaptive regularizations that use different shrinkage for different weights can also be used but we do not consider them here. Let $\widetilde{\beta}$ be β without the bias (also called intercept) β_0, and $\widetilde{w}$ be w without the biases. Let $\text{vec}(\widetilde{w})$ be the vectorized version of $\widetilde{w}$. We use the following priors

$$p(w) \propto \exp\Big(-\frac{\gamma_w}{2}\text{vec}(\widetilde{w})^\top \text{vec}(\widetilde{w})\Big),\;\; p(\beta) \propto \exp\Big(-\frac{\gamma_\beta}{2}\widetilde{\beta}^\top\widetilde{\beta}\Big).$$

To select the hyperparameters γ_w and γ_β we use an empirical Bayes approach within the iterative VB estimation method. Writing the Bayesian hierarchical model in the generic form

$$\begin{aligned} y|\psi,\theta &\sim p(y|\theta,\psi) \\ \theta|\psi &\sim p(\theta|\psi), \end{aligned}$$

with y the data, θ the model parameters and ψ the hyperparameters to be selected. The marginal likelihood for ψ is $p(y|\psi) = \int p(\theta|\psi)p(y|\theta,\psi)d\theta$, which can be maximized using an EM-type algorithm (Casella, 2001). Given an initial value $\psi^{(0)}$,

$$\psi^{(k+1)} = \text{argmax}_\psi\big\{\mathbb{E}_{\theta|y,\psi^{(k)}}\log p(y,\theta|\psi)\big\}.$$

Here $\mathbb{E}_{\theta|y,\psi^{(k)}}(\cdot)$ is the expectation with respect to the posterior distribution $p(\theta|y,\psi^{(k)})$. It can be shown that the updating rule for γ_w and γ_β is

$$\gamma_w^{(k+1)} = \frac{d_{\widetilde{w}}}{\mathbb{E}_{\theta|y,\psi^{(k)}}(\text{vec}(\widetilde{w})^\top\text{vec}(\widetilde{w}))},\quad \gamma_\beta^{(k+1)} = \frac{d_{\widetilde{\beta}}}{\mathbb{E}_{\theta|y,\psi^{(k)}}(\widetilde{\beta}^\top\widetilde{\beta})}, \tag{20}$$

where $d_{\widetilde{w}}$ and $d_{\widetilde{\beta}}$ denote the dimension of $\text{vec}(\widetilde{w})$ and $\widetilde{\beta}$, respectively. In our VB framework, the expectation $\mathbb{E}_{\theta|y,\psi^{(k)}}(\cdot)$ can be naturally approximated by the expectation w.r.t. the current VB approximation $q_{\lambda^{(k)}}(\theta)$, then the updates in (20) can be computed in closed form.

4.2 Other practical recommendations

We often use a fixed learning rate

$$a_t = \epsilon_0 \frac{\tau}{t} \tag{21}$$

where ϵ_0 is a small value, e.g. 0.01 and τ is some threshold from which the learning rate is reduced, e.g. $\tau = 200$. It might also be useful to use some adaptive learning rate methods, and later we consider one method adapted from Ranganath et al. (2013) which is described in Ong et al. (2018). This learning rate is suitable for use with the natural gradient.

As a method of accelerating the stochastic gradient optimization we also consider use of the momentum method (Polyak, 1964). The update rule is

$$\begin{aligned} \overline{\nabla_\lambda \text{LB}} &= \alpha \overline{\nabla_\lambda \text{LB}} + (1-\alpha) \widehat{\nabla_\lambda \text{KL}}(\lambda^{(t)})^{\text{natural}}, \\ \lambda^{(t+1)} &= \lambda^{(t)} + a_t \overline{\nabla_\lambda \text{LB}}, \end{aligned}$$

where $\alpha \in [0,1]$ is the momentum weight. The use of the moving average gradient $\overline{\nabla_\lambda \text{LB}}$ helps remove some of the noise inherent in the estimated gradients of the lower bound. See Goodfellow et al. (2016), Chapter 8, for a detailed discussion on the usefulness of the momentum method.

It is common in modern deep learning applications of neural network methods to implement early stopping in the optimization to avoid overfitting, even if regularization priors are used. We often observe that the training loss decreases steadily over the VB updates, while the validation loss starts increasing at some point. Therefore we also implement early stopping of the variational inference algorithm if the validation loss on a test set has not decreased after a certain number of iterations, and the set of model parameters with respect to the lowest validation loss is stored.

For initialization of variational parameters $\lambda^{(0)} = (\mu^{(0)}, B^{(0)}, \Delta^{(0)})$, we follow Glorot and Bengio (2010) and initialize each weight in $\mu^{(0)}$ by $\mathcal{U}(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}})$ where the weight connects a layer with m units to a layer with n units. The elements in $B^{(0)}$ are initialized by $\mathcal{N}(0, 0.001^2)$ and the elements in $\Delta^{(0)}$ are initialized by 0.001. It is advisable to first standardize the input data so that each column has a zero sample mean and a standard deviation of one. Finally, we use the rectified activation function

$$h(x) = \begin{cases} x, & \text{if } x > 0, \\ 0, & \text{else} \end{cases}$$

in all examples, if not otherwise stated.

5 Experimental studies and applications

To illustrate the prediction accuracy of fGLM and fGLMM, and the efficiency of our natural gradient training algorithms we consider a range of experimental studies and applications. All the examples are implemented in Matlab and run on a desktop computer with i5 3.3 Ghz Intel Quad Core.

5.1 Experimental studies

5.1.1 Efficiency of the natural gradient

We use a German credit dataset with 1000 observations, of which 750 are used as the training data and the rest as the test data. This dataset is obtained from the UCI Machine Learning Repository, `http://archive.ics.uci.edu/ml`. The data consist of a binary response variable, credit status, which is good credit (1) or bad credit (0), together with 30 covariate variables such as education, credit amount, employment status, etc. We consider a simple logistic regression model for predicting the credit status, based on the covariates. We use an improper flat prior for the coefficients θ, i.e. $p(\theta) \propto 1$.

We first study the performance of the natural gradient method compared to the ordinary gradient method. Figure 2 shows the convergence of the lower bound of the ordinary gradient method and the natural gradient method using a single factor and isotropic idiosyncratic noise, i.e. $\Sigma = BB^\top + c^2 I$ where B is a column vector as described in Section 3.4.1. For the ordinary gradient, we use the adaptive learning rate method ADADELTA of Zeiler (2012). For the natural gradient, we use both the fixed learning rate in (21) and the adaptive learning rate of Ong et al. (2018) (which is based on a similar method in Ranganath et al. (2013)). The figure shows that using both the natural gradient and adaptive learning rate speeds convergence. Figure 3 shows similar results for a one factor model with non-isotropic idiosyncratic noise, i.e. $\Sigma = BB^\top + D^2$ for D diagonal. The results have a similar interpretation to before - the natural gradient speeds up convergence and the adaptive learning rate is helpful.

Figure 4 compares lower bounds for the one-factor parameterization with and without isotropic noise - the difference in the lower bounds evident in the figure shows the improvement in the posterior approximation brought by the more general variational family but Figure 5 shows that the predictive loss is similar.

5.1.2 Prediction accuracy of DL models

We generate data from two models

$$\begin{aligned} (M_1) \qquad & y = 5 + 3(X_1 + 2X_2)^2 + 5X_3X_4 + 2X_5 + \epsilon \\ (M_2) \qquad & y = 5 + 10X_1 + \tfrac{10}{X_2^2+1} + 5X_3X_4 + 2X_4 + 5X_4^2 + 5X_5 + \epsilon \end{aligned}$$

where $\epsilon \sim \mathcal{N}(0,1)$. For each model, we generate a training dataset of $n_{\text{train}} = 100{,}000$ observations and a test dataset of $n_{\text{test}} = 100{,}000$ observations. For the first model the X_i are generated from the uniform distribution $\mathcal{U}(-1,1)$, for the second model $X = (X_1,...,X_5)^\top$ are generated from a multivariate normal distribution with mean

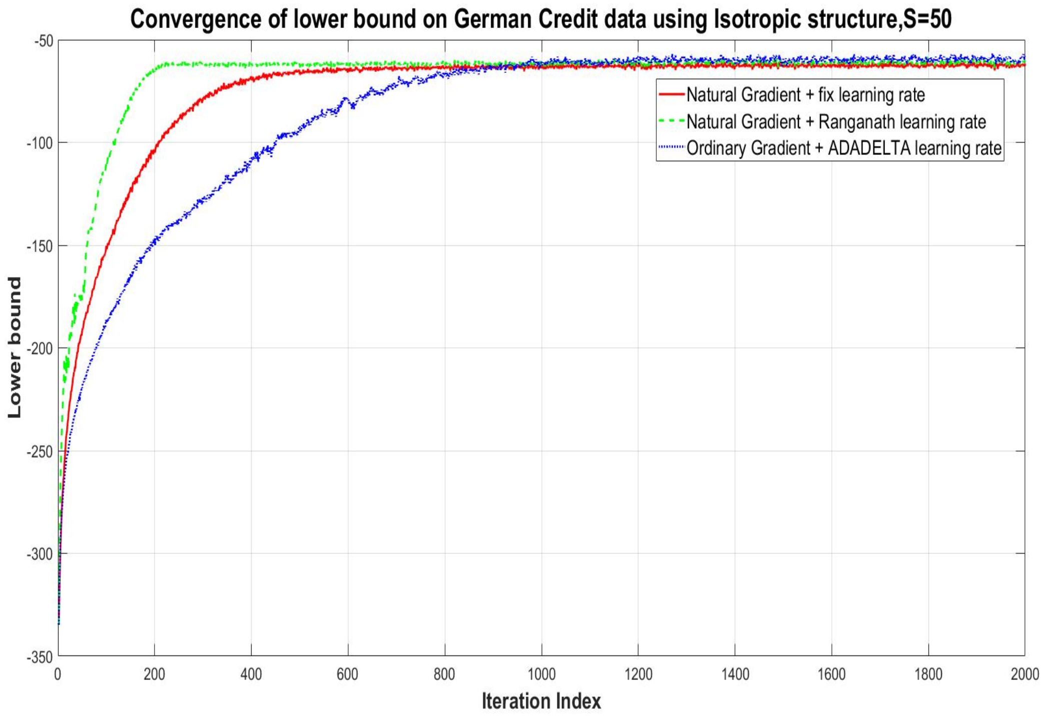

Figure 2: Lower bound versus iteration number for the ordinary gradient and natural gradient methods: the variational family uses a covariance matrix with one factor and isotropic idiosyncratic noise.

zero and covariance matrix $(0.5^{|i-j|})_{i,j}$. In both models the relationship between y and the X_i is non-linear, and it is more difficult to predict the response in the second model.

To measure the prediction accuracy, we use partial predictive score (PPS)

$$PPS = -\frac{1}{n_{\text{test}}}\sum_{(x_i,y_i)\in\text{test data}} \log p(y_i|x_i,\widehat{\theta}),$$

with $\widehat{\theta}$ a point estimate of the model parameters, and mean squared error (MSE)

$$MSE = \frac{1}{n_{\text{test}}}\sum_{(x_i,y_i)\in\text{test data}} (y_i-\widehat{y}_i)^2,$$

with $\widehat{y}_i$ a prediction of y_i. In our fGLM, for the dataset generated from M_1, we use a neural net with the structure (5,10,10,1), i.e. the input layer has 5 variables, two hidden layers each has 10 units and one scalar output. For the dataset generated from M_2, we use a neural net with the structure (5,10,10,10,10,1), i.e. there are four hidden layers each with 10 units. This selection was done after some experimental

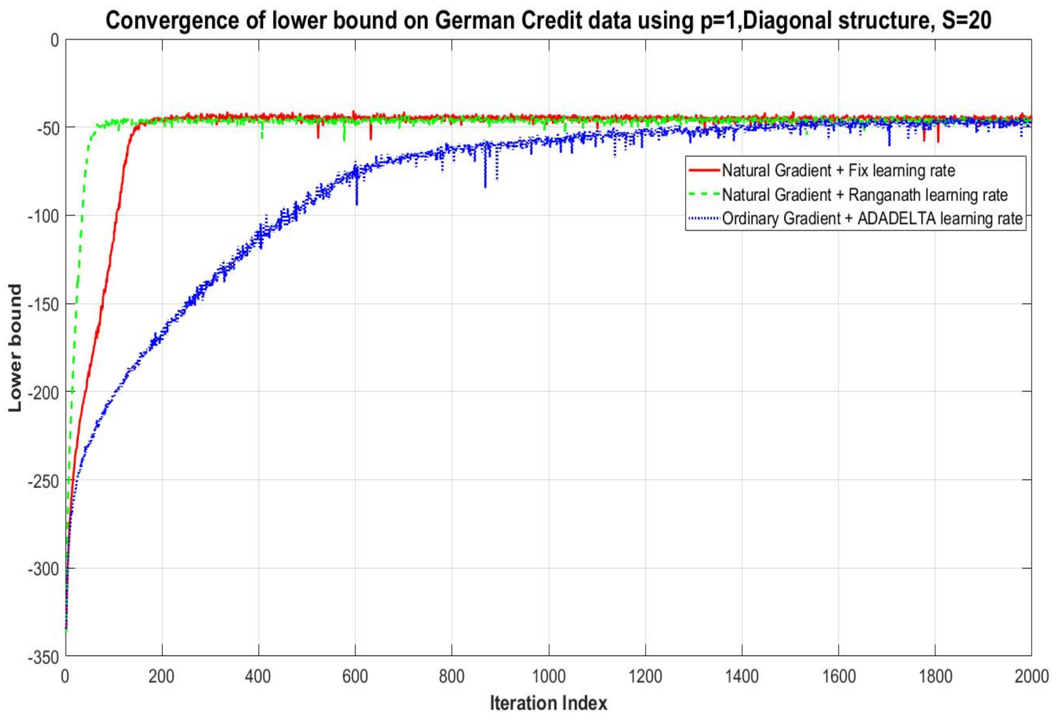

Figure 3: Lower bound versus iteration number for the ordinary gradient and natural gradient methods: the variational family uses a covariance matrix with one factor and non-isotropic idiosyncratic noise.

exploration. We use the one factor variational family with isotropic idiosyncratic noise and the natural gradient training method with fixed learning rate in (21). We compare the performance of fGLM to that of a linear model. Table 1 summarizes the results, which shows that fGLM outperforms the linear model in term of prediction accuracy. The RMSE of fGLM is close to 1, which is the best RMSE under the true model.

	M_1		M_2	
Method	PPS	MSE	PPS	MSE
GLM	2.26	5.84	2.98	11.62
fGLM	0.83	1.34	0.86	1.39

Table 1: Performance of fGLM v.s. GLM in term of the partial predictive score (PPS) and the mean squared error (MSE). Both are evaluated on the test dataset.

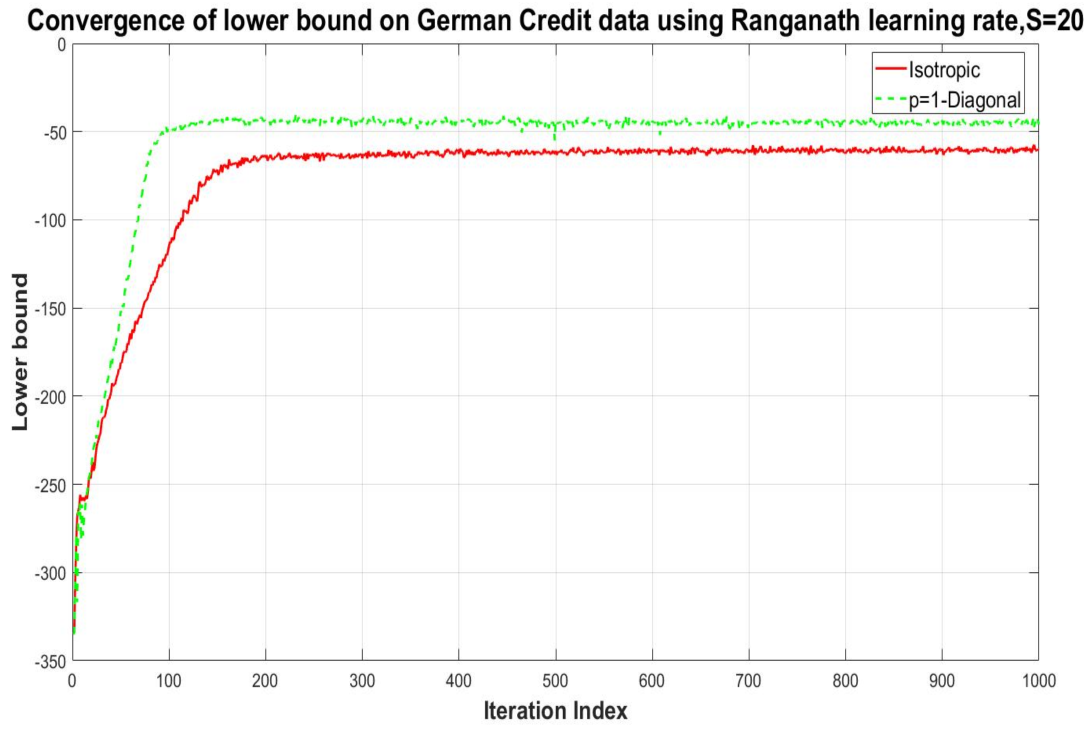

Figure 4: Lower bound versus iteration number for the natural gradient method: comparison of variational families using covariance matrices with one factor and with and without isotropic diosyncratic noise.

5.1.3 Binary panel data simulation

We study the fGLMM on a simulation binary panel dataset $D=\{(x_{it},y_{it});t=1,...,20;i=1,...,1000\}$ with x_{it} the vector of covariates and y_{it} the response of subject i at time t. The response y_{it} is generated from the following model:

$$\begin{aligned} a_{it} &= 2+3(x_{it,1}-2x_{it,2})^2-5\frac{x_{it,3}}{(1+x_{it,4})^2}-5x_{it,5}+b_i+\epsilon_{it}, \\ y_{it} &= \begin{cases} 1, & \text{if } a_{it}>0, \\ 0, & \text{otherwise,} \end{cases} \end{aligned}$$

where $b_i \sim \mathcal{N}(0,0.1)$ is a random effect intercept representing charactersistics of subject i and $\epsilon_{it} \sim \mathcal{N}(0,1)$ is random noise associated with reponses y_{it}. The $x_{it,j}, j=1,...,5$, are generated from a uniform distribution $\mathcal{U}(-1,1)$.

We fit use the following flexible logistic regression model with random effects to

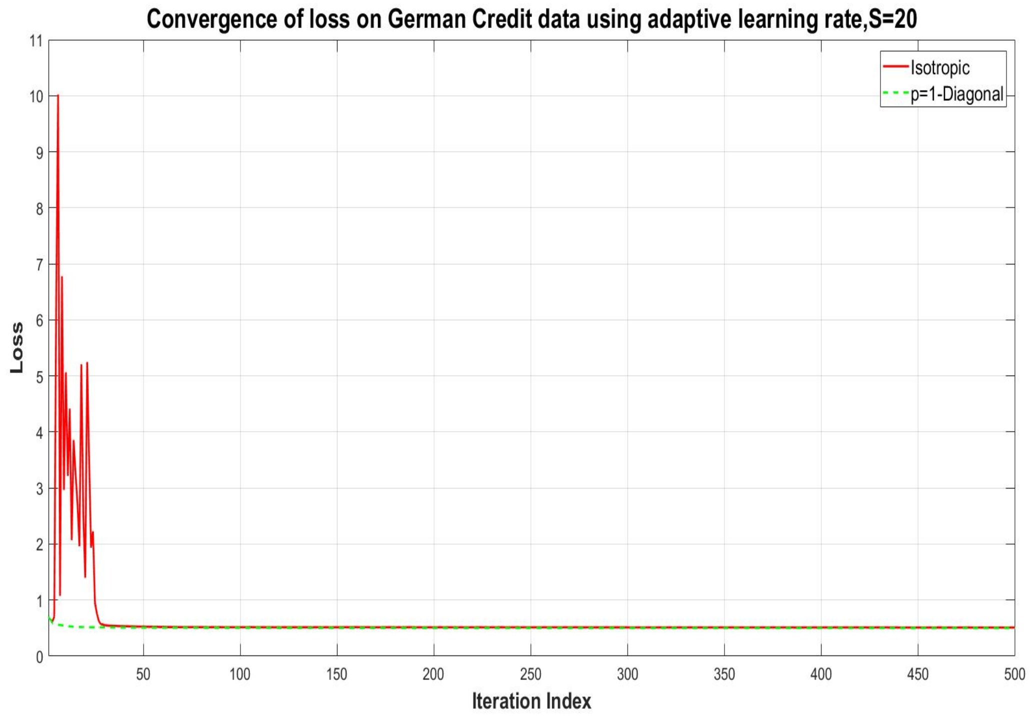

Figure 5: Prediction loss versus iteration number for the natural gradient method: comparison of variational families using covariance matrices with one factor and with and without isotropic diosyncratic noise.

this dataset

$$\begin{aligned} y_{it}|x_{it} &\sim \text{Binomial}(1,\mu_{it}) \\ \log\left(\frac{\mu_{it}}{1-\mu_{it}}\right) &= \mathfrak{N}(x_{it},w,\beta+\alpha_i), \end{aligned}$$

where $\mathfrak{N}(x_{it},w,\beta+\alpha_i)$ is the scalar output of a neural net with input x_{it}, inner weights w and output weights $\beta+\alpha_i$. We assume that the random effects $\alpha_i \sim \mathcal{N}(0,\Gamma)$ with $\Gamma = \text{diag}(\gamma_0,...,\gamma_m)$. The model parameters are $\theta = (w,\beta,\gamma_0,...,\gamma_m)$. We use Gamma priors on the γ_j, $\gamma_j \sim \text{Gamma}(a_0,b_0)$, and set the hyperparameters $a_0 = 1$ and $b_0 = 0.1$ in this example. The Appendix gives further details on training this model.

For each subject, we use the first 14 observations for training, next 3 observations for validation and the last 3 observations for testing. That is, we are interested in the within-subject prediction. The neural network has one hidden layer with 10 nodes, and we compare the fGLMM model with the logistic regression model with a random intercept.

We use two measures of predictive performance. The first is the partial predictive score (PPS), which is the average of the negative log-likelihood values. The second is the misclassification rate. Let $\bar{\theta}$ be the mean of the VB approximation $q_\lambda(\theta)$ after convergence. Let $\widehat{\mu}_{\alpha_i}$ in (31) be the mode of $p(\alpha_i|y_i,x_i,\bar{\theta})$, which can be used as the prediction of the random effects α_i of individual i. Let (y_{it_0},x_{it_0}) be a future data pair and

$$\widehat{\mu}_{it_0}=\frac{1}{1+\exp(-\mathfrak{N}(x_{it_0},w,\beta+\widehat{\mu}_{\alpha_i}))}.$$

We set the prediction of y_{it_0} as

$$\widehat{y}_{it_0}=1 \text{ if and only if } \mathfrak{N}(x_{it_0},w,\beta+\widehat{\mu}_{\alpha_i})\geq 0.$$

The classification error is

$$\text{MCR}=\sum I_{\widehat{y}_{it_0}\neq y_{it_0}}/\text{total number of future observations}$$

with the sum over the test data points (y_{it_0},x_{it_0}).

Table 2 summarizes the comparison of fGLMM and GLMM performance. The results clearly show that modelling covariate effects in a flexible way using the neural network basis functions is helpful here in terms of improving both PPS and MCR.

Model	PPS	MCR
GLMM	1.24	17.57%
fGLMM	0.13	5.27%

Table 2: Simulation binary panel dataset: Performance of fGLMM v.s. GLMM in term of the partial predictive score (PPS) and the misclassification rate (MCR). Both are evaluated on the test dataset.

5.2 Applications

5.2.1 High-dimensional logistic regression using the horseshoe prior

This application is concerned with high-dimensional logistic regression using a sparse signal shrinkage prior, the horseshoe prior (Carvalho et al., 2010). Here the variational optimization is challenging because of the strong dependence between local variance parameters and the corresponding coefficients. Using three real datasets we show that the natural gradient estimation method improves the performance of the approach described in Ong et al. (2017a).

Let $y_i\in\{0,1\}$ be a set of binary responses with corresponding covariates $x_i=(x_{i1},...,x_{ip})^\top$, $i=1,...,n$. We consider a logistic regression model

$$\log\frac{\mu_i}{1-\mu_i}=\beta_0+x_i^\top\beta,$$

where $\mu_i = \mathrm{P}(y_i = 1|x_i)$, β_0 is an intercept and $\beta = (\beta_1,...,\beta_p)^\top$ are coefficients. We consider the setting where p is large, possibly $p \gg n$, and use a shrinkage prior for β, the horseshoe prior (Carvalho et al., 2010). Specifically we assume $\beta_0 \sim N(0,10)$ and

$$\beta_j|\lambda_j \sim N(0,\lambda_j^2 g^2) \quad \lambda_j \sim C^+(0,1),$$

$j=1,...,p$, where $C^+(0,1)$ denotes the half-Cauchy distribution. The parameters λ_j, $j=1,...,p$ are local variance parameters providing shrinkage for individual coefficients, and the parameter g is a global shrinkage parameter which can adapt to the overall level of sparsity. For g a half-Cauchy prior is assumed, $g \sim C^+(0,1)$. The above prior settings are the same as those considered in Ong et al. (2017a).

The parameter θ consists of $\theta = (\beta_0,\beta^\top,\delta^\top,\gamma)^\top$, where $\delta = (\delta_1,...,\delta_p)^\top = (\log\lambda_1,...,\log\lambda_p)^\top$, and $\gamma = \log g$. We consider Gaussian variational approximation for the posterior of θ, using a factor covariance structure, for 3 gene expression datasets, the `Colon`, `Leukaemia` and `Breast` cancer datasets found at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`. These three datasets `Colon`, `Leukaemia` and `Breast` have training sample sizes of 42, 38 and 38 and test set sizes of 20, 34 and 4 respectively. The number of covariates is $p=2000$ for the `Colon` data, and $p=7120$ for the `Leukaemia` and `Breast datasets`. This means that for the `Leukaemia` and `Breast` datasets the dimension of θ is 14,242 so these are very high-dimensional examples in terms of the parameter. These data were also considered in Ong et al. (2017a) where slow convergence in the variational optimization was observed using their method; we show here that a natural gradient approach offers a significant improvement.

	VAFC of Ong et al. (2017a)			Natural gradient VAFC		
	Train Error	Test Error	CPU	Train Error	Test Error	CPU
Colon	0/42	0/20	4.92	0/42	0/20	0.17
Leukemia	0/38	6/34	61	0/38	1/34	0.56
Breast	0/38	1/4	61.6	0/38	0/4	0.56

Table 3: Performance of the ordinary gradient VAFC and natural gradient VAFC methods on three cancer datasets. Training and test errors rates are reported as the ratio of misclassified data points over the number of data points. Computational time CPU (per 100 iterations) is measured in second.

Table 3 compares the performance of VAFC of Ong et al. (2017a) and our natural gradient VAFC training methods. The table shows the predictive performance and computational time on three cancer datasets. We ran VAFC with $f=4$ factors and use only a single sample to estimate the gradient of lower bound ($S=1$) in two methods.

5.2.2 Flexible Poisson regression: bike sharing data

Bike-rental behavior is highly correlated to environmental and seasonal settings. Fanaee-T and Gama (2013) collected a dataset where the response data are hourly counts of rental bikes and the seven covariates are season (1: spring, 2: summer, 3: fall, 4: winter), holiday (1: if the day is holiday, 0: not), weekly day (1 if weekly day, 0 if weekend), weather, temperature, humidity and wind speed. The dataset is availalbe at the UCI Machine Learning Repository `http://archive.ics.uci.edu/ml`. To analyse this dataset, we use the following flexible Poisson regression model

$$\begin{aligned} y_i|x_i &\sim \text{Poisson}(\mu_i) \\ \log(\mu_i) &= \mathfrak{N}(x_i, w, \beta). \end{aligned}$$

The total 17379 observations are divided into a training dataset with 15000 observations, and a test dataset with 2379 observations.

We consider a neural network with 8 layers, each with 7 units. The model is fitted using the isotropy natural gradient variational inference approach. The fitted model provides a PPS on the test set of -1005.2. Compared to a conventional Poisson linear generalized model whose PPS is -820.4, which suggests that the more flexible model is warranted.

5.2.3 A continuous panel data set: Cornwell and Rupert data

This section analyzes a continuous panel data set originally analyzed in Cornwell and Rupert (1988). This is a balanced panel dataset with 595 individuals and 4165 observations, in which each individual was observed for 7 years. The dataset is available from the website of the textbook Baltagi (2013). The variables are listed in Table 4.

We are interested in predicting the wage (on the log scale) of each individual, given the covariates. Let y_{it} be a continuous variable indicating the log of wage of person i with the vector of covariates x_{it} in year t, $t=1,...,7$. We use the following flexible linear regression model with random effects

$$\begin{aligned} y_{it}|x_{it} &\sim \mathcal{N}(\mu_{it}, \sigma^2) \\ \mu_{it} &= \mathfrak{N}(x_{it}, w, \beta + \alpha_i), \end{aligned}$$

where $\mathfrak{N}(x_{it},w,\beta+\alpha_i)$, β and α_i have the same interpretation as in Section 5.1.3, and σ^2 is the variance of noise.

Since we are interested in within-subject prediction, for each individual, we use the first 4 observations as training data, the next 1 observation for validation data and the last 2 observations for test data. We use a neural network with 2 hidden layers 5 nodes each; this structure was selected after some experiments. We compare the performance of fGLMM to the linear regression model with a random effect, using PPS and MSE as evaluation metrics.

Variable	Meaning
EXP	Years of full-time work experience
WKS	Weeks worked
OCC	blue-collar occupation = 1;otherwise = 0
IND	manufacturing industry=1;otherwise = 0
SOUTH	south residence =1;otherwise = 0
SMSA	metropolitan residence=1,otherwise = 0
MS	married = 1; otherwise = 0
FEM	female = 1; otherwise = 0
UNION	Union contract wage = 1; otherwise = 0
ED	Years of education
BLK	black = 1; otherwise = 0
LWAGE	log of wage

Table 4: Cornwell and Rupert data: variables and their meaning

Table 5 summarizes the results, which again show that using the neural network as the basis functions to model covariate effects in a flexible way can significantly improve both PPS and MSE.

Model	PPS	MSE
GLMM	0.05	0.18
fGLMM	-0.87	0.05

Table 5: Cornwell and Rupert data: Performance of fGLMM v.s. GLMM in term of the partial predictive score (PPS) and the mean square error (MSE). Both are evaluated on the test dataset.

6 Discussions and conclusions

This paper is concerned with flexible versions of generalized linear and generalized linear mixed models where DNN methodology is used to automatically choose transformations of the raw covariates. The challenges of Bayesian computation have been addressed using variational approximation methods with a parsimonious factor covariance structure. In the case of random effects models we are able to estimate the gradient of the log likelihood unbiasedly with the random effects integrated out efficiently using Fisher's identity. We have demonstrated that a natural gradient approach to the variational optimization with this family of approximations is feasible even in high dimensions using iterative conjugate gradient methods for solving large positive definite linear systems in computation of the natural gradients. Using simulation and real datasets and several different models we show that the improvement that these methods can bring in terms of speed of convergence and computation time are substantial, and that the use of neural network basis functions with random effects is a class of models that deserve more consideration in the literature.

Appendix

Isotropic natural gradient VB estimation method

This section presents in detail the isotropic natural gradient VB estimation method for estimating a posterior distribution $\pi(\theta)\propto\exp(h(\theta))$, $h(\theta)=\log(p(\theta)L(\theta))$. With the VB approximation $q_\lambda(\theta)=\mathcal{N}(\theta;\mu,\Sigma)$, $\Sigma=bb^\top+c^2I_d$, the vector of the VB parameters is $\lambda=(\mu^\top,b^\top,c)^\top$. The lower bound

$$\text{LB}(\lambda)=\mathbb{E}_{\theta\sim q_\lambda}\Big(h(\theta)-\log q_\lambda(\theta)\Big)=\mathbb{E}_\epsilon\Big(h(g(\epsilon,\lambda))-\log q_\lambda(g(\epsilon,\lambda))\Big),$$

where $\theta=g(\lambda,\epsilon)=\mu+b\epsilon_1+c\epsilon_2$ with a scalar ϵ_1 and d-vector ϵ_2, $\epsilon=(\epsilon_1,\epsilon_2^\top)^\top\sim\mathcal{N}(0,I_{d+1})$. The ordinary gradient of the lower bound

$$\nabla_\lambda\text{LB}(\lambda)=\mathbb{E}_\epsilon\Big(\nabla_\lambda g(\lambda,\epsilon)\nabla_\theta\left\{h(\theta)-\log q_\lambda(\theta)\right\}|_{\theta=g(\epsilon,\lambda)}\Big),$$

which is estimated unbiasedly using i.i.d samples $\epsilon_s\sim\mathcal{N}(0,I_{d+1})$, $s=1,...,S$ as

$$\widehat{\nabla_\lambda\text{LB}}(\lambda)=\frac{1}{S}\sum_{s=1}^{S}\nabla_\lambda g(\lambda,\epsilon_s)\nabla_\theta\big\{h(\theta)-\log q_\lambda(\theta)\big\}|_{\theta=g(\epsilon_s,\lambda)}. \quad (22)$$

It's easy to see that

$$\nabla_\lambda g(\lambda,\epsilon)=\begin{pmatrix}I_d\\ \epsilon_1 I_d\\ \epsilon_2^\top\end{pmatrix},$$

while the gradient $\nabla_\theta h(\theta)$ depends on the model being estimated, and finally by (29)

$$\nabla_\theta\log q_\lambda(\theta)=-\Sigma^{-1}(\theta-\mu)=-\frac{1}{c^2}(\theta-\mu)+\frac{b^\top(\theta-\mu)}{c^2+b^\top b}b.$$

In order to implement the natural gradient VB algorithm we need the information matrix $I_F(\lambda)=\text{cov}_{q_\lambda}(\nabla_\lambda\log q_\lambda(\theta))$, where

$$\log q_\lambda(\theta)\propto-\frac{1}{2}\log|\Sigma|-\frac{1}{2}(\theta-\mu)^\top\Sigma^{-1}(\theta-\mu).$$

Let $U = U(x)$ be a matrix-valued function of a scalar x, and $g(U)$ be a real-valued function of U. By the chain rule (Petersen and Pedersen, 2012)

$$\nabla_x\big(g(U(x))\big) = \mathrm{tr}\Big[\nabla_U\big(g(U)\big)^\top \nabla_x\big(U(x)\big)\Big].$$

Note that $\nabla_U(a^\top U^{-1}b) = -U^{-1}ab^\top U^{-1}$ (Petersen and Pedersen, 2012), we have

$$\nabla_b\Big((\theta-\mu)^\top\Sigma^{-1}(\theta-\mu)\Big) = -2\Sigma^{-1}(\theta-\mu)(\theta-\mu)^\top\Sigma^{-1}b. \tag{23}$$

Because $\nabla_U(\log|U|) = U^{-1}$, we have $\nabla_x(\log|U|) = \mathrm{tr}(U^{-1}\nabla_x U)$. Then

$$\nabla_b\Big(\log|\Sigma|\Big) = 2\Sigma^{-1}b. \tag{24}$$

Similarly,

$$\nabla_c\Big(\log|\Sigma|\Big) = 2c\times\mathrm{tr}(\Sigma^{-1}) \tag{25}$$

and

$$\nabla_c\Big((\theta-\mu)^\top\Sigma^{-1}(\theta-\mu)\Big) = -2c(\theta-\mu)^\top\Sigma^{-2}(\theta-\mu). \tag{26}$$

So

$$\nabla_\lambda \log q_\lambda(\theta) = \begin{pmatrix}\nabla_\mu \log q_\lambda(\theta)\\ \nabla_b \log q_\lambda(\theta)\\ \nabla_c \log q_\lambda(\theta)\end{pmatrix} = \begin{pmatrix}\Sigma^{-1}(\theta-\mu)\\ -\Sigma^{-1}b+\Sigma^{-1}(\theta-\mu)(\theta-\mu)^\top\Sigma^{-1}b\\ -c\times\mathrm{tr}(\Sigma^{-1})+c(\theta-\mu)^\top\Sigma^{-2}(\theta-\mu)\end{pmatrix}. \tag{27}$$

Let $X = \Sigma^{-1}(\theta-\mu)\sim\mathcal{N}(0,\Sigma^{-1})$, and using the results on cubic and quadratic forms of a Gaussian random vector (Petersen and Pedersen, 2012), it can be shown that

$$I_F(\lambda) = \begin{pmatrix}\Sigma^{-1} & O_{d\times d} & O_{d\times 1}\\ O_{d\times d} & \Sigma^{-1}bb^\top\Sigma^{-1}+(b^\top\Sigma^{-1}b)\Sigma^{-1} & 2c\Sigma^{-2}b\\ O_{1\times d} & 2cb^\top\Sigma^{-2} & 2c^2\mathrm{tr}(\Sigma^{-2})\end{pmatrix}. \tag{28}$$

Now note that

$$\Sigma^{-1} = (bb^\top + c^2 I)^{-1} = \frac{1}{c^2}\left(I - \frac{1}{c^2+b^\top b}bb^\top\right) = \frac{1}{c^2}\left(I-\alpha bb^\top\right) \tag{29}$$

and

$$\Sigma^{-1}b = \alpha b,$$

with $\alpha = 1/(c^2+b^\top b)$. The Fisher information matrix in (28) can be written as

$$I_F(\lambda) = \begin{pmatrix}\Sigma^{-1} & O_{d\times d} & O_{d\times 1}\\ O_{d\times d} & A & u\\ O_{1\times d} & u^\top & \omega\end{pmatrix}$$

with

$$\begin{aligned}
A &= \Sigma^{-1}bb^T\Sigma^{-1} + (b^T\Sigma^{-1}b)\Sigma^{-1} = \frac{\alpha(b^\top b)}{c^2}I_d + \alpha^2\left(1-\frac{b^\top b}{c^2}\right)bb^\top \\
u &= 2c\Sigma^{-2}b = 2c\alpha^2 b \\
\omega &= 2c^2\mathrm{tr}(\Sigma^{-2}) = \frac{2}{c^2}\left[d-1+(\frac{c^2}{c^2+b^\top b})^2\right] = \frac{2}{c^2}\left[d-1+(c^2\alpha)^2\right].
\end{aligned}$$

Hence,

$$I_F^{-1}(\lambda) = \begin{pmatrix} \Sigma^{-1} & O_{d\times(d+1)} \\ O_{(d+1)\times d} & \begin{pmatrix} A & u \\ u^\top & \omega \end{pmatrix}^{-1} \end{pmatrix}$$

with

$$\begin{pmatrix} A & u \\ u^\top & \omega \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} + \frac{1}{\omega - u^\top A^{-1}u}A^{-1}uu^\top A^{-1} & -\frac{1}{\omega - u^\top A^{-1}u}A^{-1}u \\ -\frac{1}{\omega - u^\top A^{-1}u}u^\top A^{-1} & \frac{1}{\omega - u^\top A^{-1}u} \end{pmatrix}.$$

After some algebra, it can be seen that $I_F^{-1}(\lambda)$ is

$$I^{-1}(\lambda) = \begin{pmatrix} bb^\top + c^2 I_d & O_{d\times d} & O_{d\times 1} \\ O_{d\times d} & \kappa_3 bb^\top + \kappa_4 I_d & -\frac{\kappa_1}{\kappa_2}b \\ O_{1\times d} & -\frac{\kappa_1}{\kappa_2}b^\top & 1/\kappa_2 \end{pmatrix}. \tag{30}$$

with

$$\begin{aligned}
\kappa_1 &= \left[(1+\frac{c^2}{b^\top b}) - \frac{1}{2}(1+\frac{c^2}{b^\top b})^2\right]\frac{2c(b^\top b)}{(c^2+b^\top b)^2} + \frac{2c^3}{b^\top b(c^2+b^\top b)} \\
\kappa_2 &= \frac{2}{c^2}\left(d-1+(\frac{c^2}{c^2+b^\top b})^2\right) - \frac{2c\kappa_1}{(c^2+b^\top b)^2}b^\top b \\
\kappa_3 &= (1+\frac{c^2}{b^\top b}) - \frac{1}{2}(1+\frac{c^2}{b^\top b})^2 + \frac{\kappa_1}{\kappa_2} \\
\kappa_4 &= c^2\left(1+\frac{c^2}{b^\top b}\right).
\end{aligned}$$

It's important to note that it's unnecessary to store the matrix $I^{-1}(\lambda)$. All we need is the matrix-vector product $\nabla_\lambda \mathrm{KL}(\lambda)^{\mathrm{natural}} = I^{-1}(\lambda)\nabla_\lambda \mathrm{KL}(\lambda)$. Write $\nabla_\lambda \mathrm{KL}(\lambda) = (g_1^\top, g_2^\top, g_3)^\top$ with g_1 the vector formed by the first d elements of $\nabla_\lambda \mathrm{KL}(\lambda)$, g_2 the vector formed by the next d elements, and g_3 the last element in $\nabla_\lambda \mathrm{KL}(\lambda)$. The natural gradient is

$$\nabla_\lambda \mathrm{KL}(\lambda)^{\mathrm{natural}} = \begin{pmatrix} (g_1^\top b)b + c^2 g_1 \\ \left(\kappa_3(g_2^\top b) - g_3\frac{\kappa_1}{\kappa_2}\right)b + \kappa_4 g_2 \\ \frac{g_3}{\kappa_3} - (g_2^\top b)\frac{\kappa_1}{\kappa_2} \end{pmatrix}.$$

The complexity of computing the natural gradient is $O(d)$.

Further details for the example in Section 5.1.3

The likelihood contribution w.r.t. panel (y_i,x_i) is

$$\begin{aligned} L_i(\theta) &= \int p(y_i|x_i,w,\beta,\alpha_i)p(\alpha_i|\Gamma)d\alpha_i \\ &= \int \exp\left(\sum_{t=1}^{T_i} y_{it}\mathfrak{N}(x_{it},w,\beta+\alpha_i) - \log\left(1+e^{\mathfrak{N}(x_{it},w,\beta+\alpha_i)}\right)\right)p(\alpha_i|\Gamma)d\alpha_i. \end{aligned}$$

By Fisher's identity (Gunawan et al., 2017)

$$\nabla_\theta \ell_i(\theta)=\int \nabla_\theta\left\{\log p(\alpha_i|\Gamma)+\sum_{t=1}^{T_i} y_{it}\mathfrak{N}(x_{it},w,\beta+\alpha_i)-\log\left(1+e^{\mathfrak{N}(x_{it},w,\beta+\alpha_i)}\right)\right\}p(\alpha_i|y_i,x_i,\theta)d\alpha_i.$$

We have

$$\begin{aligned} p(\alpha_i|y_i,x_i,\theta) &\propto p(\alpha_i|\Gamma)p(y_i|x_i,w,\beta,\alpha_i) \\ &\propto \exp\left(\sum_{t=1}^{T_i}\left[y_{it}z_{it}^\top(\beta+\alpha_i)-\log(1+e^{z_{it}^\top(\beta+\alpha_i)})\right]-\frac{1}{2}\alpha_i^\top\Gamma^{-1}\alpha_i\right) = \exp(f(\alpha_i)). \end{aligned}$$

$$\nabla_{\alpha_i} f(\alpha_i)=\boldsymbol{Z}_i^\top(y_i-p_i)-\Gamma^{-1}\alpha_i,\ p_i=p_i(\alpha_i)=(p_{i1},...,p_{iT_i})^\top$$
$$\nabla_{\alpha_i\alpha_i^\top} f(\alpha_i)=-\boldsymbol{Z}_i^\top \mathrm{diag}(p_i\circ(1-p_i))\boldsymbol{Z}_i-\Gamma^{-1}$$

Let $\widehat{\mu}_{\alpha_i}$ be the maximizer of $f(\alpha_i)$ which can be obtained by the Newton-Raphson method, and let

$$\widehat{\Sigma}_{\alpha_i} = \left(-\nabla_{\alpha_i\alpha_i^\top} f(\alpha_i)|_{\alpha_i=\widehat{\mu}_{\alpha_i}}\right)^{-1} = \left(\boldsymbol{Z}_i^\top \mathrm{diag}(p_i\odot(1-p_i))\boldsymbol{Z}_i+\Gamma^{-1}\right)^{-1},\ \ p_i=p_i(\widehat{\mu}_{\alpha_i}) \tag{31}$$

We note that for the Gaussian flexible linear mixed model in Section 5.2.3, $\widehat{\mu}_{\alpha_i}$ and $\widehat{\Sigma}_{\alpha_i}$ can be derived analytically.

The gradient $\nabla_\theta\ell_i(\theta)$ can be estimated as follows.

- Generate N samples $\alpha_i^{(j)}\sim\mathcal{N}(\widehat{\mu}_{\alpha_i},\widehat{\Sigma}_{\alpha_i})$, $j=1,...,N$.
- Compute the weights

$$W_i^{(j)}=\exp\left(f(\alpha_i^{(j)})+\frac{1}{2}(\alpha_i^{(j)}-\widehat{\mu}_{\alpha_i})^\top\widehat{\Sigma}_{\alpha_i}^{-1}(\alpha_i^{(j)}-\widehat{\mu}_{\alpha_i})\right)$$

  and $W_i^{(j)}=W_i^{(j)}/\sum_{k=1}^N W_i^{(k)}$.
- Compute the estimate

$$\widehat{\nabla_\theta\ell_i(\theta)}=\sum_{j=1}^N\nabla_\theta\left\{\log p(\alpha_i^{(j)}|\Gamma)+\sum_{t=1}^{T_i}\left[y_{it}z_{it}^\top(\beta+\alpha_i^{(j)})-\log(1+e^{z_{it}^\top(\beta+\alpha_i^{(j)})})\right]\right\}W_i^{(j)}.$$

Because the parameters $\gamma_j > 0$, a suitable transformation is needed before applying the Gaussian VB approximation. We use the transformation $\theta_{\gamma_j} = \log(\gamma_j)$, $j = 0,...,M$. Let $\widetilde{\theta} = (w,\beta,\theta_{\gamma_0},...,\theta_{\gamma_M})$, then

$$\theta = \theta(\widetilde{\theta}) = (w,\beta,\exp(\theta_{\gamma_0}),...,\exp(\theta_{\gamma_M})).$$

The posterior distribution of $\widetilde{\theta}$ is

$$p(\widetilde{\theta}|D) \propto \left|\frac{\partial\theta(\widetilde{\theta})}{\partial\widetilde{\theta}}\right| p(\theta(\widetilde{\theta}))p(\theta(\widetilde{\theta})|D) = \exp(\theta_{\gamma_0} + ... + \theta_{\gamma_M} + \theta_{\gamma_w} + \theta_{\gamma_\beta}))p(\theta(\widetilde{\theta}))p(\theta(\widetilde{\theta})|D)$$

We then approximate $p(\widetilde{\theta}|D)$ by $q_\lambda(\widetilde{\theta})$.

References

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276.

Baltagi (2013). *Econometric Analysis of Panel Data, 5th Edition.* Wiley.

Bartholomew, D. J., Knott, M., and Moustaki, I. (2011). *Latent variable models and factor analysis: A unified approach, 3rd edition.* John Wiley & Sons.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* New York: Springer.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Carvalho, C. M., Polson, N. G., and Scott, J. G. (2010). The horseshoe estimator for sparse signals. *Biometrika*, 97:465–480.

Casella, G. (2001). Empirical bayes gibbs sampling. *Biostatistics*, 2:485–500.

Cornwell, C. and Rupert, P. (1988). Efficient estimation with panel data: An empirical comparison of instrumental variable estimators. *Journal of Applied Econometrics*, 3(2):149–155.

Fan, K., Wang, Z., Beck, J., Kwok, J., and Heller, K. (2015). Fast second-order stochastic backpropagation for variational inference. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 1387–1395. Curran Associates, Inc.

Fanaee-T, H. and Gama, J. (2013). Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, pages 1–15.

Geweke, J. and Zhou, G. (1996). Measuring the pricing error of the arbitrage pricing theory. *Review of Financial Studies*, 9(2):557–587.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, PMLR*, volume 9, pages 249–256.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning.* MIT Press.

Gunawan, D., Tran, M.-N., and Kohn, R. (2017). Fast inference for intractable likelihood problems using variational Bayes. Technical report. arXiv:1705.06679.

Han, S., Liao, X., Dunson, D. B., and Carin, L. C. (2016). Variational Gaussian copula inference. In Gretton, A. and Robert, C. C., editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51, pages 829–838, Cadiz, Spain. JMLR Workshop and Conference Proceedings.

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

Honkela, A., Raiko, T., Kuusela, M., Tornio, M., and Karhunen, J. (2010). Approximate Riemannian conjugate gradient learning for fixed-form variational Bayes. *Journal of Machine Learning Research*, 11:3235–3268.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational Bayes. Technical report. arXiv:1312.6114v10.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., and Blei, D. M. (2017). Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18(14):1–45.

Lai, T. L., Shih, M.-C., and Wong, S. P. (2006). A new approach to modeling covariate effects and individualization in population pharmacokinetics-pharmacodynamics. *Journal of Pharmacokinetics and Pharmacodynamics*, 33(1):49–74.

Martens, J. (2010). Deep learning via Hessian-free optimization. In *27th International Conference on Machine Learning*, Haifa, Israel.

Martens, J. (2014). New insights and perspectives on the natural gradient method. Technical report. arXiv:1412.1193.

Ong, V. M.-H., Nott, D. J., and Smith, M. S. (2017a). Gaussian variational approximation with factor covariance structure. *Journal of Computational and Graphical Statistics*, To Appear.

Ong, V. M.-H., Nott, D. J., Tran, M.-N., Sisson, S., and Drovandi, C. (2018). Variational Bayes with synthetic likelihood. *Statistics and Computing*, To appear.

Ong, V. M.-H., Nott, D. J., Tran, M.-N., Sisson, S. A., and Drovandi, C. C. (2017b). Likelihood-free inference in high dimensions with synthetic likelihood. Technical report, Queensland University of Tehcnology, `https://eprints.qut.edu.au/112213/`.

Pascanu, R. and Bengio, Y. (2014). Revisiting natural gradient for deep networks. Technical report. arXiv:1301.3584v7.

Petersen, K. B. and Pedersen, M. S. (2012). The matrix cookbook. http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3274/pdf/imm3274.pdf.

Polson, N. G. and Sokolov, V. O. (2017). Deep learning: A Bayesian perspective. Technical report. arXiv:1706.00473v1.

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5).

Ranganath, R., Wang, C., Blei, D. M., and Xing, E. P. (2013). An adaptive learning rate for stochastic variational inference. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 298–306.

Rao, C. R. (1945). Information and accuracy attainable in the estimation of statistical parameters. *Bull. Calcutta. Math. Soc*, 37:81–91.

Regier, J., Jordan, M. I., and McAuliffe, J. (2017). Fast black-box variational inference through stochastic trust-region optimization. *arXiv preprint arXiv:1706.02375*.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1278–1286.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407.

Roeder, G., Wu, Y., and Duvenaud, D. (2017). Sticking the landing: Simple, lower-variance gradient estimators for variational inference. *arXiv preprint arXiv:1703.09194*.

Salimans, T. and Knowles, D. A. (2013). Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):741–908.

Sato, M. (2001). Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117.

Stoer, J. (1983). Solution of large linear systems of equations by conjugate gradient type methods. In Bachem, A., Korte, B., and Grötschel, M., editors, *Mathematical Programming The State of the Art: Bonn 1982*, pages 540–565. Springer Berlin Heidelberg, Berlin, Heidelberg.

Tan, L. S. L. and Nott, D. J. (2017). Gaussian variational approximation with sparse precision matrices. *Statisics and Computing*, To Appear.

Titsias, M. and Lázaro-Gredilla, M. (2014). Doubly stochastic variational Bayes for non-conjugate inference. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1971–1979.

Tran, M., Nott, D., and Kohn, R. (2017). Variational Bayes with intractable likelihood. *Journal of Computational and Graphical Statistics*. to appear.

Tran, M.-N., Kohn, R., Quiroz, M., and Villani, M. (2016). Block-wise pseudo-marginal Metropolis-Hastings. Technical report. arXiv:1603.02485.

Trippe, B. L. and Turner, R. E. (2018). Overpruning in variational Bayesian neural networks. arXiv:1801.06230v1.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. Technical report. arXiv:1212.5701.