The Variational Predictive Natural Gradient

Da Tang [1] **Rajesh Ranganath** [2]

Abstract

Variational inference transforms posterior inference into parametric optimization thereby enabling the use of latent variable models where otherwise impractical. However, variational inference can be finicky when different variational parameters control variables that are strongly correlated under the model. Traditional natural gradients based on the variational approximation fail to correct for correlations when the approximation is not the true posterior. To address this, we construct a new natural gradient called the Variational Predictive Natural Gradient (VPNG). Unlike traditional natural gradients for variational inference, this natural gradient accounts for the relationship between model parameters and variational parameters. We demonstrate the insight with a simple example as well as the empirical value on a classification task, a deep generative model of images, and probabilistic matrix factorization for recommendation.

1. Introduction

Variational inference (Jordan et al., 1999) transforms posterior inference in latent variable models into optimization. It posits a parametric approximating family and tries to find the distribution in this family that minimizes the Kullback-Leibler (KL) divergence to the posterior. Variational inference makes posterior computation practical where it would not be otherwise. It has powered many applications, including computational biology (Carbonetto et al., 2012; Stegle et al., 2010), language (Miao et al., 2016), compressive sensing (Shi et al., 2014), neuroscience (Manning et al., 2014; Harrison & Green, 2010), and medicine (Ranganath et al., 2016).

Variational inference requires choosing an approximating family. The variational family plus the model together define the variational objective. The variational objective can be optimized with stochastic gradients for a broad range of models (Kingma & Welling, 2014; Ranganath et al., 2014; Rezende et al., 2014). When the posterior has correlations, dimensions of the optimization problem become tied, i.e., there is curvature. One way to correct for curvature in optimization is to use natural gradients (Amari, 1998; Ollivier et al., 2011; Thomas et al., 2016) . Natural gradients for variational inference (Hoffman et al., 2013) adjust for the non-Euclidean nature of probability distributions. But they may not change the gradient direction when the variational approximation is far from the posterior.

To deal with curvature induced by dependent observation dimensions in the variational objective, we define a new type of natural gradient: the variational predictive natural gradient (VPNG). The VPNG rescales the gradient with the inverse of the expected Fisher information matrix of the reparameterized model likelihood. We relate this matrix to the negative Hessian of the expected log-likelihood part of the evidence lower bound (ELBO), thereby showing it captures the curvature of variational inference.

Our new natural gradient captures potential pathological curvature introduced by the log-likelihood traditional natural gradient cannot capture. Further, unlike traditional natural gradients for variational inference, the VPNG corrects for curvature in the objective between model parameters and variational parameters. In Section 3, we will design an illustrate example where the VPNG points almost directly to the optimum, while both the vanilla gradient and the natural gradient point in almost an orthogonal direction.

We show our approach outperforms vanilla gradient optimization and the traditional natural gradient optimization on several latent variable models, including Bayesian logistic regression on synthetic data, variational autoencoders (Kingma & Welling, 2014; Rezende et al., 2014) on images, and variational probabilistic matrix factorization (Mnih & Salakhutdinov, 2008; Gopalan et al., 2015; Liang et al., 2016) on movie recommendation data.

Related work. Variational inference has been transformed by the use of Monte Carlo gradient estimators (Kingma & Ba, 2014; Rezende et al., 2014; Mnih & Gregor, 2014; Ranganath et al., 2014; Titsias & Lázaro-Gredilla,

[1]Department of Computer Science, Columbia University, New York, New York, USA [2]The Courant Institute, New York University, New York, New York, USA. Correspondence to: Da Tang <datang@cs.columbia.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

2014). Though these approaches expand the applicability of variational inference, the underlying optimization problem can still be hard. Some recent work applied second-order optimization to solve this problem. For example, Fan et al. (2015) derived Hessian-free style optimization for variational inference. Another line of related work is on efficiently computing Fisher information and natural gradients for complex model likelihood such as the K-FAC approximation (Martens & Grosse, 2015; Grosse & Martens, 2016; Ba et al., 2016). Finally, the VPNG can be combined with methods for robustly setting step sizes, like using the VPNG curvature matrix to build the quadratic approximation in TrustVI (Regier et al., 2017).

2. Background

Latent variable models Latent variable models posit latent structure $\mathbf{z}$ to describe data $\mathbf{x}$ with parameters $\boldsymbol{\theta}$. The model is

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x} \mid \mathbf{z}; \boldsymbol{\theta}).$$

The model is split into a prior over the hidden structure $p(\mathbf{z})$ and likelihood that describes the probability of data.

Variational inference Variational inference (Jordan et al., 1999) approximates the posterior distribution $p(\mathbf{z} \mid \mathbf{x}; \boldsymbol{\theta})$ with a distribution $q(\mathbf{z} \mid \mathbf{x}; \boldsymbol{\lambda})$ over the latent variables indexed by parameter $\boldsymbol{\lambda}$. It works by maximizing the ELBO:

$$\mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\theta}) = \mathbb{E}_q \left[\log p(\mathbf{x} \mid \mathbf{z}; \boldsymbol{\theta})\right] - \mathrm{KL}(q(\mathbf{z} \mid \mathbf{x}; \boldsymbol{\lambda}) || p(\mathbf{z})) \tag{1}$$

Maximizing the ELBO minimizes the KL divergence to the posterior. The model parameters $\boldsymbol{\theta}$ and variational parameters $\boldsymbol{\lambda}$ can be optimized together. The family q is chosen to be amenable to stochastic optimization. One example is the mean-field family, where $q(\mathbf{z} \mid \mathbf{x})$ is factorized over all coordinates of $\mathbf{z}$ like in the variational autoencoder.

q-Fisher information The ELBO can be optimized with gradients. The effectiveness of gradient ascent methods relates to the geometry of the problem. When the loss landscape contains variables that control the objective in a coupled manner, like the means of two correlated latent variables, gradient ascent methods can be slow.

One way to adjust for this coupling or *curvature* is to use natural gradients (Amari, 1998). Natural gradients account for the non-Euclidean geometry of parameters of probability distributions by looking for optimal ascent directions in symmetric KL-divergence balls. The natural gradient relies on the Fisher information of q,

$$F_q = \mathbb{E}_q \left[\nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z}|\mathbf{x}; \boldsymbol{\lambda}) \cdot \nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z}|\mathbf{x}; \boldsymbol{\lambda})^\top\right]. \tag{2}$$

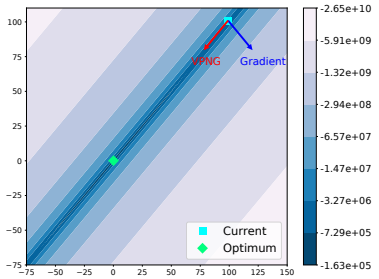

Figure 1. The VPNGs are more effective than vanilla gradients and traditional natural gradients (pointing into the same direction with the vanilla gradients for this example).

We call this matrix the q-Fisher information matrix. With this Fisher information matrix, the natural gradient is $\nabla^{\mathrm{NG}}_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}) = F_q^{-1} \cdot \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda})$.

Natural gradients have been used to optimize the the ELBO (Hoffman et al., 2013). The natural gradient works because it approximates the Hessian of the ELBO at the optimum. The negative Hessian matrix of the ELBO is:

$$-\frac{\partial^2}{\partial \boldsymbol{\lambda}^2}\mathcal{L} = F_q + \int \frac{\partial^2}{\partial \boldsymbol{\lambda}^2} q \cdot (\log q(\mathbf{z} \mid \mathbf{x}; \boldsymbol{\lambda}) - \log p(\mathbf{z} \mid \mathbf{x}))d\mathbf{z}. \tag{3}$$

The last integral in the above equation is small when the variational distribution $q(\mathbf{z} \mid \mathbf{x}; \boldsymbol{\lambda})$ is close to the posterior distribution $p(\mathbf{z} \mid \mathbf{x})$. Hence, the q-Fisher information matrix can be viewed as a positive semi-definite version of the negative Hessian matrix of the ELBO. Thus natural gradients improve optimization efficiency, when the variational approximation is close to the posterior.

3. The Variational Predictive Natural Gradient

The q-Fisher information is insufficient. Consider the following example with bivariate Gaussian likelihood that has an unknown mean $\boldsymbol{\mu} = \begin{pmatrix}\mu_1 \\ \mu_2\end{pmatrix}$, a pathological known covariance $\Sigma = \begin{pmatrix}1 & 1-\varepsilon \\ 1-\varepsilon & 1\end{pmatrix}$ for some constant $0 \leq \varepsilon \ll 1$, and an isotropic Gaussian prior:

$$p(\mathbf{x}_{1:n}, \boldsymbol{\mu}) = p(\boldsymbol{\mu} \mid \mathbf{0}, I_2) \prod_{i=1}^{n} \mathcal{N}(\mathbf{x}_i \mid \boldsymbol{\mu}, \Sigma). \tag{4}$$

To do variational inference, we choose a mean-field approximation $q(\boldsymbol{\mu}; \boldsymbol{\lambda}) = \mathcal{N}(\mu_1 \mid \lambda_1, \sigma^2)\mathcal{N}(\mu_2 \mid \lambda_2, \sigma^2)$ with σ to be fixed. The posterior distribution for this problem is analytic: $p(\boldsymbol{\mu}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}', \Sigma')$ where $\Sigma' = (n\Sigma^{-1} + I_2)^{-1}$ and $\boldsymbol{\mu}' = (n \cdot I_2 + \Sigma)^{-1} \cdot \sum_{i=1}^{n} \mathbf{x}_i$. The optimal solution for the variational parameter $\boldsymbol{\lambda}$ should be $\boldsymbol{\mu}'$. The gradient

of the objective function $\mathcal{L}(\boldsymbol{\lambda})$ is

$$\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}) = -\boldsymbol{\lambda} + \Sigma^{-1} \cdot \left(-n\boldsymbol{\lambda} + \sum_{i=1}^{n} \mathbf{x}_i\right). \quad (5)$$

The precision matrix Σ^{-1} is pathological. It has an eigenvector $\mathbf{v}_1 = \frac{1}{\sqrt{2}}(1,1)^\top$ with eigenvalue $\frac{1}{2-\varepsilon}$, and an eigenvector $\mathbf{v}_2 = \frac{1}{\sqrt{2}}(1,-1)^\top$ with eigenvalue $\frac{1}{\varepsilon}$. As a result, vanilla gradients will almost always go along the direction of the eigenvector $\mathbf{v}_2$, as shown in Figure 1. Further, natural gradients fail to resolve this. The q-Fisher information matrix of this problem is diagonal, so it cannot help resolve the extreme curvature between the parameters λ_1 and λ_2.

Notice that this pathological curvature is not due to that mean-field approximation family on $q(\boldsymbol{\mu}; \boldsymbol{\lambda})$ does not contain the true posterior $p(\boldsymbol{\mu} \mid \mathbf{x})$. In fact, even if we optimize $q(\boldsymbol{\mu}; \boldsymbol{\lambda})$ over the family of all bivariate Gaussian distributions $\mathcal{N}(\boldsymbol{\mu} \mid \boldsymbol{\lambda}_{\boldsymbol{\mu}}, \boldsymbol{\lambda}_\Sigma)$, the partial gradient $\nabla_{\boldsymbol{\lambda}_{\boldsymbol{\mu}}}\mathcal{L}$ over the mean parameter vector $\boldsymbol{\lambda}_{\boldsymbol{\mu}}$ will still have the same curvature issue. The issue arises since the variational approximation does not approximate the posterior well at initialization . In general, if at some point the current q iterate cannot approximate the posterior well, then the corresponding q-Fisher information matrix may not be able to correct the curvature in the parameters.

3.1. Negative Hessian of the expected log-likelihood

The pathology of the ELBO for the model in Equation 4 comes from the ill-conditioned covariance matrix Σ. The covariance matrix of the posterior can correct for this pathology since its covariance matrix is $\Sigma' \approx \frac{1}{n}\Sigma$. The disconnect lies in that variational inference is only close to the posterior at its optimum, which implies that q-natural gradients only correct for the curvature well once the variational approximation is close to the posterior, i.e., the inference problem is almost solved.

The problem is that the q-Fisher information matrix measures how parameter perturbations alter the variational approximation, regardless of the current model parameters and the quality of the current variational approximation. We bring the model back into the picture by considering positive definite matrices that resemble the negative Hessian matrix of the expected log-likelihood part $\mathcal{L}^{\text{ll}} = \mathbb{E}_q\left[\log p(\mathbf{x} \mid \mathbf{z}; \boldsymbol{\theta})\right]$ of the ELBO, over both the variational parameter $\boldsymbol{\lambda}$ and the model parameter $\boldsymbol{\theta}$.

The expected log-likelihood contains where the model and variational approximation interact, so its Hessian contains the relevant curvature for optimize the ELBO. However, since we are maximizing the ELBO, the matrices need to not only resemble the negative Hessian, but should also be positive semidefinite. The negative Hessian of the expected log-likelihood is not guaranteed to be positive semi-definite.

Our goal is to construct a positive semidefinite matrix related to the negative Hessian that accelerate inference by considering the curvature both the variational parameter and the model parameter. In the sequel, we will show this new matrix is a type of Fisher information.

To compute gradients and Hessians, we need to compute derivatives over expectations controlled by the variational parameter $\boldsymbol{\lambda}$. In general, we can differentiate and use score function-style estimators from black box variational inference (Ranganath et al., 2014). For simplicity, consider the case where q is reparameterizable (Kingma & Welling, 2014; Rezende et al., 2014). Then draws for $\mathbf{z}$ from q can be written as deterministic transformations g of noise terms $\boldsymbol{\varepsilon}$ with parameter-free distributions s. This simplifies the computations:

$$\mathbf{z} = g(\mathbf{x}, \boldsymbol{\varepsilon}; \boldsymbol{\lambda}) \sim q(\mathbf{z} \mid \mathbf{x}; \boldsymbol{\lambda}) \iff \boldsymbol{\varepsilon} \sim s(\boldsymbol{\varepsilon}). \quad (6)$$

The reparameterization trick can be applied to many common distributions (i.e. reparameterize a Gaussian draw $\nu \sim \mathcal{N}(\mu, \sigma^2)$ as $\nu = \mu + \sigma\varepsilon$ where $\varepsilon \sim \mathcal{N}(0,1)$).

With this trick, denote $\boldsymbol{\eta} = (\boldsymbol{\lambda}^\top, \boldsymbol{\theta}^\top)^\top$, the negative Hessian matrix of $\mathcal{L}^{\text{ll}}$ becomes:

$$-\frac{\partial^2 \mathcal{L}^{\text{ll}}}{\partial \boldsymbol{\eta}^2} = -\mathbb{E}_{\boldsymbol{\varepsilon}}\left[\frac{\partial^2}{\partial \boldsymbol{\eta}^2} \log p(\mathbf{x} \mid \mathbf{z} = g(\mathbf{x}, \boldsymbol{\varepsilon}; \boldsymbol{\lambda}); \boldsymbol{\theta})\right].$$

Let us first consider the case where the variational distribution q factorizes over data points: $q(\mathbf{z} \mid \mathbf{x}; \boldsymbol{\lambda}) = \prod_{i=1}^n q(\mathbf{z}_i \mid \mathbf{x}_i; \boldsymbol{\lambda})$. This factorization occurs in many popular models, such as in variational autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014). Denote Q as the empirical distribution of the observed data $\mathbf{x}_{1:n}$. Also denote $p(\mathbf{z}_i)$ and $p(\mathbf{x}_i \mid \mathbf{z}_i)$ as the prior and likelihood function for any single data point $\mathbf{x}_i$. Moreover, for any data point $\mathbf{x}_i$ and $\mathbf{x}_i'$, we define the function

$$u(\mathbf{x}_i, \mathbf{x}_i', \boldsymbol{\varepsilon}_i, \boldsymbol{\eta}) = \frac{\partial^2}{\partial \boldsymbol{\eta}^2} \log p(\mathbf{x}_i' \mid \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta}).$$

Since we can use $\mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda})$ to reparameterize $\mathbf{z}_i$, we can assume that the Jacobian matrix $\frac{\partial \mathbf{z}_i}{\partial \boldsymbol{\varepsilon}_i}$ is always invertible and hence by the *inverse function theorem* we can also write $\boldsymbol{\varepsilon}_i$ as a function of $\mathbf{z}_i$, $\mathbf{x}_i$ and $\boldsymbol{\lambda}$. Hence, we can also express the above equation as

$$\frac{\partial^2}{\partial \boldsymbol{\eta}^2} \log p(\mathbf{x}_i' \mid \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta}) = v(\mathbf{x}_i, \mathbf{x}_i', \mathbf{z}_i, \boldsymbol{\eta}).$$

With this notation, we can rewrite the above negative Hes-

sian matrix for $\mathcal{L}^{\text{ll}}$ as

$$\begin{aligned}
-\frac{\partial^2 \mathcal{L}^{\text{ll}}}{\partial \boldsymbol{\eta}^2} &= -\sum_{i=1}^{n} \mathbb{E}_{\boldsymbol{\varepsilon}_i}\left[\frac{\partial^2}{\partial \boldsymbol{\eta}^2} \log p(\mathbf{x}_i \,|\, \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta})\right] \\
&= -n\mathbb{E}_{Q(\mathbf{x}_i)}\left[\mathbb{E}_{\boldsymbol{\varepsilon}_i}\left[\frac{\partial^2}{\partial \boldsymbol{\eta}^2} \log p(\mathbf{x}_i \,|\, \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta})\right]\right] \\
&= -n\mathbb{E}_{Q(\mathbf{x}_i)}\left[\mathbb{E}_{\boldsymbol{\varepsilon}_i}\left[u(\mathbf{x}_i, \mathbf{x}_i, \boldsymbol{\varepsilon}_i, \boldsymbol{\eta})\right]\right] \\
&= -n\mathbb{E}_{Q(\mathbf{x}_i)}\left[\mathbb{E}_{q(\mathbf{z}_i \,|\, \mathbf{x}_i; \boldsymbol{\lambda})}\left[v(\mathbf{x}_i, \mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\eta})\right]\right]
\end{aligned} \tag{7}$$

Assessing the positive definiteness of Equation 7 is a challenge because of the expectation with respect to the variational approximation. To make the positive definiteness easier to wrangle, we make the assumption that

$$p(\mathbf{z}_i)p(\mathbf{x}_i \,|\, \mathbf{z}_i) \approx Q(\mathbf{x}_i)q(\mathbf{z}_i \,|\, \mathbf{x}_i). \tag{8}$$

When our model is learning a successful parameter vector $\boldsymbol{\eta}$, the distribution $p(\mathbf{z}_i, \mathbf{x}_i) = p(\mathbf{z}_i)p(\mathbf{x}_i \,|\, \mathbf{z}_i)$ should be close to the distribution $Q(\mathbf{z}_i, \mathbf{x}_i) = Q(\mathbf{x}_i)q(\mathbf{z}_i \,|\, \mathbf{x}_i)$ since the variational distribution q is trying to learn the posterior distribution $p(\mathbf{z}_i \,|\, \mathbf{x}_i)$ while $p(\mathbf{x}_i)$ is trying to learn the empirical data distribution Q. This is the only approximation we will use to derive the VPNG.

This substitution is similar to $q(\mathbf{z} \,|\, \mathbf{x}) \approx p(\mathbf{z} \,|\, \mathbf{x})$ made when analyzing the q-Fisher information matrix. They can be quite different when the $q(\mathbf{z} \,|\, \mathbf{x}; \boldsymbol{\lambda})$ approximating family may not be large enough to accurately approximate the posterior distribution $p(\mathbf{z} \,|\, \mathbf{x})$, and when the $p(\mathbf{x} \,|\, \mathbf{z}; \boldsymbol{\theta})$ model may not be able to accurately learn the data distribution Q. With Equation 8 in hand, we have

$$-\frac{\partial^2 \mathcal{L}^{\text{ll}}}{\partial \boldsymbol{\eta}^2} \approx -n\mathbb{E}_{p(\mathbf{z}_i)}\left[\mathbb{E}_{p(\mathbf{x}_i \,|\, \mathbf{z}_i; \boldsymbol{\theta})}\left[v(\mathbf{x}_i, \mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\eta})\right]\right]. \tag{9}$$

This matrix is computable via Monte Carlo, however in the next section we show that this matrix may not be positive semidefinite and provide a method to derive a matrix that is positive semidefinite.

3.2. Predictive Sampling for Positive Semi-definiteness

The inner expectation of Equation 9 is an expectation of $v(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_i, \tilde{\mathbf{z}}_i, \boldsymbol{\eta})$ with respect to the distribution $p(\tilde{\mathbf{x}}_i \,|\, \tilde{\mathbf{z}}_i; \boldsymbol{\theta})$ on $\tilde{\mathbf{x}}_i$. This matrix appears to be an average of Fisher information matrices, and thus positive semidefinite. However, v is not the Hessian of a distribution over $\mathbf{x}_i$ since $\mathbf{x}_i$ appears on both sides of conditioning bar. The failure of v to be the Hessian of a distribution for $\mathbf{x}_i$ means Equation 9 may not be positive definite. Next, we provide a concrete example where its not positive definite.

Non Positive Semi-definiteness of Second-Order Derivative. Consider a model with data points $x_1, \ldots, x_n \in \mathbb{R}$ and local latent variables $z_1, \ldots, z_n \in \mathbb{R}$. The prior is $p(\mathbf{z}) = \prod_{i=1}^{n} \mathcal{N}(z_i \,|\, 0, 1^2)$, the model distribution is $p(\mathbf{x} \,|\, \mathbf{z}; \theta) = \prod_{i=1}^{n} \mathcal{N}(x_i \,|\, \theta z_i, 1^2)$ and the variational distribution is $q(\mathbf{z} \,|\, \mathbf{x}; \lambda) = \prod_{i=1}^{n} \mathcal{N}(z_i \,|\, \lambda x_i, \sigma^2)$ with $\lambda, \theta \in \mathbb{R}$ and the hyperparameter $\sigma > 0$. Then we can reparameterize each $\tilde{z}_i = \lambda \tilde{x}_i + \tilde{\varepsilon}_i$ with $\tilde{\varepsilon}_i \sim \mathcal{N}(0, \sigma^2)$ drawn in an i.i.d. way. Under this model, Equation 9 equals

$$n \begin{pmatrix} \theta^2(\theta^2+1) & \theta^2 - 1 \\ \theta^2 - 1 & 1 \end{pmatrix}^{1},$$

which is not positive semi-definite when $|\theta| < \frac{1}{\sqrt{3}}$.

Predictive Sampling for Positive Semidefiniteness. The failure of the Hessian in Equation 9 to be positive definite stems from v not being the Hessian of a probability distribution. To remedy this, we sample the $\mathbf{x}_i$ on both side of the conditioning bar independently. That is replace

$$\mathbb{E}_{p(\mathbf{x}_i \,|\, \mathbf{z}_i; \boldsymbol{\theta})}\left[v(\mathbf{x}_i, \mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\eta})\right] \tag{10}$$

with

$$\mathbb{E}_{p(\mathbf{x}_i \,|\, \mathbf{z}_i; \boldsymbol{\theta})}\left[\mathbb{E}_{p(\mathbf{x}'_i \,|\, \mathbf{z}_i; \boldsymbol{\theta})}\left[v(\mathbf{x}_i, \mathbf{x}'_i, \mathbf{z}_i, \boldsymbol{\eta})\right]\right], \tag{11}$$

where $\mathbf{x}'_i$ is a newly drawn data point from the same distribution $p(\cdot \,|\, \mathbf{z}_i; \boldsymbol{\theta})$. This step is required. Rescaling the gradient with the inverse of the first equation does not guarantee convergence. This step will allows construction of a positive definite matrix that captures the essence of the negative Hessian. With this transformation, we get

$$\begin{aligned}
&-n\mathbb{E}_{p(\mathbf{z}_i)}\left[\mathbb{E}_{p(\mathbf{x}_i \,|\, \mathbf{z}_i; \boldsymbol{\theta})}\left[\mathbb{E}_{p(\mathbf{x}'_i \,|\, \mathbf{z}_i; \boldsymbol{\theta})}\left[v(\mathbf{x}_i, \mathbf{x}'_i, \mathbf{z}_i, \boldsymbol{\eta})\right]\right]\right] \\
\approx &-n\mathbb{E}_{Q(\mathbf{x}_i)}\left[\mathbb{E}_{q(\mathbf{z}_i \,|\, \mathbf{x}_i; \boldsymbol{\lambda})}\left[\mathbb{E}_{p(\mathbf{x}'_i \,|\, \mathbf{z}_i; \boldsymbol{\theta})}\left[v(\mathbf{x}_i, \mathbf{x}'_i, \mathbf{z}_i, \boldsymbol{\eta})\right]\right]\right] \\
= &n\mathbb{E}_{Q(\mathbf{x}_i)}\left[\mathbb{E}_{\boldsymbol{\varepsilon}_i}\left[\mathbb{E}_{p(\mathbf{x}'_i \,|\, \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta})}\left[-u(\mathbf{x}_i, \mathbf{x}'_i, \boldsymbol{\varepsilon}_i, \boldsymbol{\eta})\right]\right]\right].
\end{aligned} \tag{12}$$

The approximation step follows from the earlier assumption that the joint of p and q are close (see Equation 8).

The inner expectation of the above equation is the negative Hessian matrix of the logarithm of the density of the distribution $p(\mathbf{x}'_i \,|\, \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta})$ with respect to the parameter $\boldsymbol{\eta}$, given the latent variable $\boldsymbol{\varepsilon}_i$ and the data point $\mathbf{x}_i$. Therefore, this inner expectation equals the Fisher information matrix of this distribution, which is always positive semi-definite. The matrix in Equation 12 meets our desiderata: it maintains structure from the negative Hessian of the expected log-likelihood, is guaranteed to be positive semidefinite for any model and variational approximation to that optimization converges, and is computable via Monte Carlo samples. To see that it is computable,

[1] This matrix is normally related to both the variational parameter $\boldsymbol{\lambda}$ and the model parameter $\boldsymbol{\theta}$. Here this matrix is independent with $\boldsymbol{\lambda}$ since in this model $\frac{\partial \mathbf{z}}{\partial \boldsymbol{\lambda}}$ can be represented without $\boldsymbol{\lambda}$. The variational parameter will appear in this matrix if we set $z_i \sim \mathcal{N}(\lambda^2 x_i, \sigma^2)$ in this model.

the matrix in Equation 12 equals

$$
\begin{aligned}
&n\mathbb{E}_{Q(\mathbf{x}_i)}\left[\mathbb{E}_{\boldsymbol{\varepsilon}_i}\left[\mathbb{E}_{p(\mathbf{x}_i' \mid \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta})}\left[-u(\mathbf{x}_i, \mathbf{x}_i', \boldsymbol{\varepsilon}_i, \boldsymbol{\eta})\right]\right]\right] \\
=&n\mathbb{E}_{Q(\mathbf{x}_i)}[\mathbb{E}_{\boldsymbol{\varepsilon}_i}[\mathbb{E}_{p(\mathbf{x}_i' \mid \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta})}[\\
&\qquad \nabla_{\boldsymbol{\eta}} \log p(\mathbf{x}_i' \mid \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta})) \\
&\qquad \cdot \nabla_{\boldsymbol{\eta}} \log p(\mathbf{x}_i' \mid \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta}))^\top]]].
\end{aligned} \tag{13}
$$

This equation can be computed by sampling a data point from the observed data, sampling a noise term, and resampling a new data point from the model likelihood.

3.3. The variational predictive natural gradient

The matrix in Equation 13 is the expectation over a type of Fisher information. First, define

$$
p(\mathbf{x}_i' \mid \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta})
$$

as the *reparameterized predictive model distribution*. The Fisher information of this matrix given $\mathbf{x}_i$ and $\boldsymbol{\varepsilon}_i$ is

$$
\begin{aligned}
F_{rep}(\mathbf{x}_i, \boldsymbol{\varepsilon}_i) =&\mathbb{E}_{p(\mathbf{x}_i' \mid \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta})}[\\
&\nabla_{\boldsymbol{\eta}} \log p(\mathbf{x}_i' \mid \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta}) \\
&\cdot \nabla_{\boldsymbol{\eta}} \log p(\mathbf{x}_i' \mid \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta})^\top].
\end{aligned}
$$

Averaging the Fisher information of the reparameterized predictive model distribution over observed data points and draws from the variational approximation and rescaling by the number of data points gives.

$$
\begin{aligned}
&n\mathbb{E}_{Q(\mathbf{x}_i)}\mathbb{E}_{\boldsymbol{\varepsilon}}[F_{rep}(\mathbf{x}_i, \boldsymbol{\varepsilon}_i)] \\
&= n\mathbb{E}_{Q(\mathbf{x}_i)}[\mathbb{E}_{\boldsymbol{\varepsilon}_i}[\mathbb{E}_{p(\mathbf{x}_i' \mid \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta})}[\\
&\qquad \nabla_{\boldsymbol{\eta}} \log p(\mathbf{x}_i' \mid \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta})) \\
&\qquad \cdot \nabla_{\boldsymbol{\eta}} \log p(\mathbf{x}_i' \mid \mathbf{z}_i = g(\mathbf{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\lambda}); \boldsymbol{\theta}))^\top]]] \\
&= \mathbb{E}_{\boldsymbol{\varepsilon}}[\mathbb{E}_{p(\mathbf{x}' \mid \mathbf{z} = g(\mathbf{x}, \boldsymbol{\varepsilon}; \boldsymbol{\lambda}); \boldsymbol{\theta})}[\nabla_{\boldsymbol{\eta}} \log p(\mathbf{x}' \mid \mathbf{z} = g(\mathbf{x}, \boldsymbol{\varepsilon}; \boldsymbol{\lambda}); \boldsymbol{\theta}) \\
&\qquad \cdot \nabla_{\boldsymbol{\eta}} \log p(\mathbf{x}' \mid \mathbf{z} = g(\mathbf{x}, \boldsymbol{\varepsilon}; \boldsymbol{\lambda}); \boldsymbol{\theta})^\top]] =: F_r.
\end{aligned}
$$

The positive semidefinite matrix related to the negative Hessian of the ELBO we derived in the previous section is exactly the expected Fisher information of the *reparameterized predictive model distribution* $p(\mathbf{x}' \mid \mathbf{z} = g(\mathbf{x}, \boldsymbol{\varepsilon}; \boldsymbol{\lambda}); \boldsymbol{\theta})$.

The expected density of reparameterized predictive model distribution can be viewed as the *variational predictive distribution* $r(\mathbf{x}' \mid \mathbf{x}; \boldsymbol{\lambda}, \boldsymbol{\theta})$ of new data

$$
\begin{aligned}
\mathbb{E}_{\boldsymbol{\varepsilon}}\left[p(\mathbf{x}' \mid \mathbf{z} = g(\mathbf{x}, \boldsymbol{\varepsilon}; \boldsymbol{\lambda}); \boldsymbol{\theta})\right] =&\mathbb{E}_{q(\mathbf{z} \mid \mathbf{x}; \boldsymbol{\lambda})}\left[p(\mathbf{x}' \mid \mathbf{z}; \boldsymbol{\theta})\right] \\
:=&r(\mathbf{x}' \mid \mathbf{x}; \boldsymbol{\lambda}, \boldsymbol{\theta}).
\end{aligned}
$$

This distribution is the predictive distribution with the posterior replaced by the variational approximation. Hence, we call the matrix in Section 3.3 as the *variational predictive Fisher information* matrix. This matrix can capture curvature. Though we derive it by assuming q factorizes, this matrix may still capture curvature for the general case.

To illustrate that variational predictive Fisher information matrix can capture curvature, consider the example in Equation 4, we can reparameterize latent variable $\boldsymbol{\mu}$ in the variational distribution $q(\boldsymbol{\mu}; \boldsymbol{\lambda}) = \mathcal{N}(\mu_1 \mid \lambda_1, \sigma^2)\mathcal{N}(\mu_2 \mid \lambda_2, \sigma^2)$ as $\boldsymbol{\mu} = \boldsymbol{\lambda} + \boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I_2)$. Then the reparameterized predicted distribution $p(\mathbf{x}' \mid \boldsymbol{\mu} = \boldsymbol{\lambda} + \boldsymbol{\varepsilon})$ equals $\prod_{i=1}^{n} \mathcal{N}(\mathbf{x}_i' \mid \boldsymbol{\lambda} + \boldsymbol{\varepsilon}, \Sigma)$, whose Fisher information matrix is just $n\Sigma^{-1}$. Hence the variational predictive Fisher information matrix for this model is $F_r = n\Sigma^{-1}$, which almost exactly matches with the pathological curvature structure in the gradient in Equation 5.

Therefore, our variational predictive Fisher information matrix contains the curvature we want to correct. Hence, we apply an update with the new natural gradient, the *variational predictive natural gradient* (VPNG):

$$
\nabla^{\text{VPNG}}_{\boldsymbol{\lambda}, \boldsymbol{\theta}} \mathcal{L} = F_r^{-1} \cdot \nabla_{\boldsymbol{\lambda}, \boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\theta}). \tag{14}
$$

With this new natural gradient, the algorithm can move towards the true mean rather than getting stuck on the line $\lambda_1 - \lambda_2 = 0$, as shown in Figure 1.

The variational predictive Fisher information matrix in Section 3.3 is a positive semi-definite matrix related to the negative Hessian of the expected log-likelihood part $\mathcal{L}^{\text{ll}}$ of the ELBO. It can capture the curvature of variational inference since the expected log-likelihood part of the ELBO usually plays a more important role in the whole objective and we can view the KL divergence part $\text{KL}(q(\mathbf{z} \mid \mathbf{x}) || p(\mathbf{z}))$ as a regularization for the q distribution. In practice, the KL divergence term gets scaled by a ratio $\beta \in (0, 1)$ to learn better representations (Bowman et al., 2016). With this scaling the curvature of the expected log-likelihood part $\mathcal{L}^{\text{ll}}$ is even more important.

3.4. Comparison with the traditional natural gradient

The traditional natural gradient points to the steepest ascent direction of the ELBO in the symmetric KL divergence space of the variational distribution q (Hoffman et al., 2013). The VPNG shares a similar type of geometric structure: it points to the steepest ascent direction of the ELBO in the "expected" (over the parameter-free distribution $s(\boldsymbol{\varepsilon})$ and data distribution $Q(\mathbf{x})$) symmetric KL divergence space of the reparameterized predictive distribution $p(\mathbf{x}' \mid \mathbf{z} = g(\mathbf{x}, \boldsymbol{\varepsilon}; \boldsymbol{\lambda}); \boldsymbol{\theta})$. Details are shown in appendix.

The q-Fisher information matrix tries to capture the curvature of the ELBO. However, it strongly relies on quality of the fidelity of the variational approximation to the posterior, $q(\mathbf{z} \mid \mathbf{x}) \approx p(\mathbf{z} \mid \mathbf{x})$. The new Fisher information matrix, F_r relies on a similar approximation $p(\mathbf{z})p(\mathbf{x} \mid \mathbf{z}) \approx Q(\mathbf{x})q(\mathbf{z} \mid \mathbf{x})$, these approximations are still quite different in many cases such as when the model does not approximate the true data distribution well (de-

Algorithm 1 Variational inference with VPNGs

Input: Data $\mathbf{x}_{1:n}$, Model $p(\mathbf{x}, \mathbf{z}, \boldsymbol{\beta})$.
Initialize the parameters $\boldsymbol{\lambda}$, $\boldsymbol{\theta}$, and μ.
repeat
  Draw samples $\hat{\boldsymbol{\beta}}$ and $\hat{\mathbf{z}}_i$ (Equation 15).
  Draw i.i.d samples $\hat{\mathbf{x}}_i^{\prime(k)}$ (Equation 15).
  Compute the Fisher information matrix $\hat{F}_r$ (Equation 16).
  Compute the natural gradient $\hat{\nabla}^{\text{VPNG}}_{\boldsymbol{\lambda},\boldsymbol{\theta}}\mathcal{L}$ (Equation 17).
  Update the parameters $\boldsymbol{\lambda}$, $\boldsymbol{\theta}$ with the gradient $\hat{\nabla}^{\text{VPNG}}_{\boldsymbol{\lambda},\boldsymbol{\theta}}\mathcal{L}$.
  (Optional) Adjust the dampening parameter μ.
until convergence

scribed in the paragraph after Equation 8). Moreover, F_r has the advantage that it considers the curvature from both the variational parameter $\boldsymbol{\lambda}$ and the model parameter $\boldsymbol{\theta}$ while the q-Fisher information matrix does not consider $\boldsymbol{\theta}$.

4. Variational Inference with Approximate Curvature

To build an algorithm with the VPNG, we need to compute the reparameterized predictive distribution and take an expectation with respect to its Fisher information. These steps will only be tractable for specific choices of models and variational approximations. We address how to compute it with Monte Carlo in a broader setting here.

We can generate samples for $\mathbf{x}'$ in the distribution $p(\mathbf{x}' \mid \mathbf{z}; \boldsymbol{\theta})$ for $\hat{\mathbf{z}}$ drawn from q. These samples can be used to estimate the integrals in the definition of F_r. They are generated through the following Monte Carlo sampling process. Using $k \in \{1, \ldots, M\}$ to index the Monte Carlo samples:

$$\hat{\mathbf{z}} \sim q(\mathbf{z} \mid \mathbf{x}; \boldsymbol{\lambda}), \;\; \hat{\mathbf{x}}_i^{\prime(k)} \sim p(\mathbf{x}' \mid \hat{\mathbf{z}}; \boldsymbol{\theta}). \tag{15}$$

Reparameterization makes it easy to approximate the needed gradients of $\log p(\mathbf{x}_i^{\prime(k)} \mid \hat{\mathbf{z}}; \boldsymbol{\theta})$ with respect to $\boldsymbol{\lambda}$:

$$\nabla_{\boldsymbol{\lambda}} \log p(\hat{\mathbf{x}}_i^{\prime(k)} \mid \hat{\mathbf{z}}; \boldsymbol{\theta}) \approx \nabla_{\boldsymbol{\lambda}} \hat{\mathbf{z}} \cdot \nabla_{\hat{\mathbf{z}}_i} \log p(\hat{\mathbf{x}}_i^{\prime(k)} \mid \hat{\mathbf{z}}; \boldsymbol{\theta})^\top.$$

Denote $\hat{b}_{i,k} = \nabla_{\boldsymbol{\lambda},\boldsymbol{\theta}} \log p(\hat{\mathbf{x}}_i^{\prime(k)} \mid \hat{\mathbf{z}}; \boldsymbol{\theta})$. Using samples from Equation 15, we can estimate the variational predictive Fisher information in Section 3.3 as

$$F_r \approx \hat{F}_r = \frac{1}{M} \sum_{k=1}^{M} \sum_{i=1}^{n} \hat{b}_{i,k} \hat{b}_{i,k}^\top. \tag{16}$$

This is an unbiased estimate of the variational predictive Fisher information matrix in Section 3.3.

The approximate variational predictive Fisher information matrix $\hat{F}_r$ might be non-invertible. Since $\text{rank}(\hat{F}_r) \leq Mn$, the matrix is non-invertible if $Mn < \dim(\boldsymbol{\lambda}) + \dim(\boldsymbol{\theta})$. We add a small dampening parameter μ to ensure invertibility. This parameter can be fixed or dynamically adjusted. With this dampening parameter, the approximate variational predictive natural gradient is

$$\hat{\nabla}^{\text{VPNG}}_{\boldsymbol{\lambda},\boldsymbol{\theta}}\mathcal{L} = (\hat{F}_r + \mu I)^{-1} \cdot \nabla_{\boldsymbol{\lambda},\boldsymbol{\theta}}\mathcal{L}. \tag{17}$$

Algorithm 1 summarizes VPNG updates. We set the dampening parameter μ to be a constant in our experiments. We show this algorithm works well in Section 5.

5. Experiments

We explore the empirical performance of variational inference using the VPNG updates in Algorithm 1[2]. We consider Bayesian Logistic regression on a synthetic dataset, the VAE on a real handwritten digit dataset, and variational matrix factorization on a real movie recommendation dataset. We test their performances using different metrics on both train and held-out data.

We compare VPNG with vanilla gradient optimization and traditional natural gradient optimization using RMSProp (Tieleman & Hinton, 2012) and Adam (Kingma & Ba, 2014) to set the learning rates in all three algorithms. For each algorithm in each task, we show the better result by applying these two learning rate adjustment techniques and select the best decay rate (if applicable) and step size. We use ten Monte Carlo samples to estimate the ELBO, its derivatives, and the variational predictive Fisher information matrix F_r.

5.1. Bayesian Logistic regression

We test Algorithm 1 with a Bayesian Logistic regression model on a synthetic dataset. We have the data $\mathbf{x}_{1:n}$ and the labels $y_{1:n}$ where $\mathbf{x}_i \in \mathbb{R}^4$ is a vector and $y_i \in \{0, 1\}$ is a binary label. Each pair of $(\mathbf{x}_i, y_i)$ is generated through the following process:

$$\begin{aligned}
a_i \sim &\text{Uniform}[-5, 5] \in \mathbb{R} \\
\varepsilon_i^k \sim &\text{Uniform}[-0.005, 0.005], \; k \in \{1, 2, 3, 4\}, \\
\mathbf{x}_i = &\left(a_i, \frac{a_i}{2}, \frac{a_i}{3}, \frac{a_i}{4}\right) + \boldsymbol{\varepsilon}_i \in \mathbb{R}^4, \\
y_i = &I\left[\langle (1, -2, -3, 4), \mathbf{x}_i \rangle \geq 0\right].
\end{aligned}$$

The generated data are all very close to the ground truth classification boundary $\langle (1, -2, -3, 4), \mathbf{x} \rangle = 0$. We use Logistic regression with parameter $\mathbf{w}$ to model this data. We place an isotropic Gaussian prior distribution $p_0(\mathbf{w}) = \mathcal{N}(\mathbf{w} \mid \mathbf{0}, \sigma_0^2 \cdot I_5)$ on the parameter $\mathbf{w}$ where the parameter $\sigma_0 = 100$. We apply mean-field variational inference to the parameter $\mathbf{w}$: $q(\mathbf{w}; \boldsymbol{\mu}, \boldsymbol{\sigma}) = \prod_{i=1}^{5} \mathcal{N}(w_i \mid \mu_i, \sigma_i^2)$.

[2]Code is available at: https://github.com/datang1992/VPNG.

Table 1. Bayesian Logistic regression AUC

Method	Train AUC	Test AUC
Gradient	0.734 ± 0.017	0.718 ± 0.022
NG	0.744 ± 0.043	0.751 ± 0.047
VPNG	$\mathbf{0.972 \pm 0.011}$	$\mathbf{0.967 \pm 0.011}$

Mean-field variational families are popular primarily for their optimization efficiency. We aim to show that VPNG improves upon the speed of mean-field approaches. The data generative process and the initial prior parameter σ_0 makes the ELBO pathological. Specifically, the covariates are strongly correlated while all data points have small margins with respect to the ground truth boundary.

We generate 500 samples and select a fixed set which contains 80% of the whole data for training and use the rest for testing. We test Algorithm 1 and the baseline methods on this data. We do not need Monte Carlo samples of predicted data as the F_r can be computed efficiently given samples from the latent variables in this problem. To compare performances, we allow each algorithm to run 2000 iterations for 10 runs with various step sizes and compare the AUC scores for both the train and test procedure. The AUC scores are computed with the mean prediction.

The results are shown in Table 1. In the experiments, we calculate the train and test AUC scores for every 100 iterations and and report the average of the last 5 outputs for each method. Table 1 shows the train and test AUC scores for each method, over all 10 runs. Our method outperforms the baselines. We show a test AUC-iteration curve for this experiment in Appendix B. The vanilla gradient and traditional natural gradient do not perform well because of the curvature induced by the correlation in the covariates.

5.2. Variational autoencoder

We also study VPNGs for variational autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014) on binarized MNIST (LeCun et al., 1998). MNIST contains 70,000 images (60,000 for training and 10,000 for testing) of handwritten digits, each of size 28×28.

We use a 100-dimensional latent representation $\mathbf{z}_i$. Our variational distribution factorizes and we use a three-layer inference network to output the mean and variance of the variational distribution given a datapoint. The generative model transforms $\mathbf{z}$ using a three-layer neural network to output logits for each pixel. We use 200 hidden units for both the inference and generative networks.

To efficiently compute variational predictive Fisher information matrices, we view the entire VAE structure as a 6-layer neural network with a stochastic layer between the third and fourth layer. We then apply the tridiagonal block-wise Kronecker-factored curvature approximation (K-FAC), (Martens & Grosse, 2015). This enables us to compute Fisher information matrices faster in feed-forward neural networks. We further improve efficiency by constructing low-rank approximations of large matrices. Finally, we use exponential moving averages of quantities related to the K-FAC approximations. We show more details in appendix.

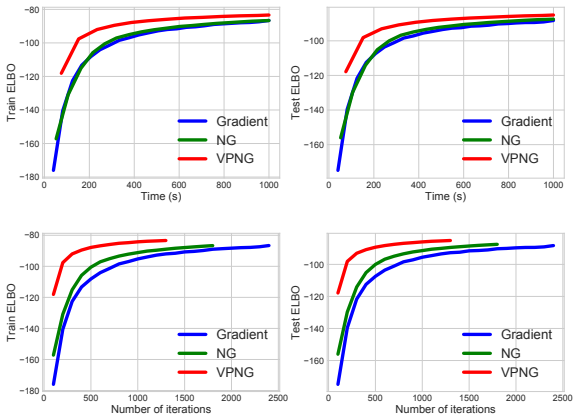

Figure 2. VAE Learning curves on binarized MNIST

We compare the VPNG method with the vanilla gradient and natural gradient optimizations. Since the traditional natural gradient does not deal with the model parameter $\boldsymbol{\theta}$, we use the vanilla gradient for $\boldsymbol{\theta}$ in this setting. We do not need to compare the performances of the VPNG with the traditional natural gradient by fixing the model parameter $\boldsymbol{\theta}$ and learning only the variational parameter $\boldsymbol{\lambda}$ for two reasons. First, this setting is not common for VAEs. Second, we need to have a fixed value for $\boldsymbol{\theta}$ and it is difficult to obtain an optimal value for it before running the algorithms.

We select a batch size of 600 since we print the ELBO values every 100 iterations. Hence, we evaluate the performances for each algorithm exactly once per epoch. The test ELBO values are computed over the whole test set and the train ELBO values are computed over a fixed set of 10,000 randomly-chosen (out of the whole 60,000) images. We allow each method to run for 1,000 seconds (we found similar results at longer runtimes) and select the best step sizes among several reasonable choices. Figure 2 shows the results. Though the VPNG method is the slowest per iteration, it outperforms the baseline optimizations on both the train and test sets, even on running time. We also compare these methods with the second-order optimization method, the Hessian-free Stochastic Gaussian Variational Inference (HFSGVI) (Fan et al., 2015). However, it was not fast enough due to the large amount of Hessian-vector product computations. The ELBO values with this method are still far below

-200 within 1,000 seconds, which is much slower than the methods shown in Figure 2.

The intuitive reason for the performance gain stems from the fact that the VAE parameters control pixels that are highly correlated across images. The VPNG corrects for this correlation.

5.3. Variational matrix factorization

Our third experiment is on MovieLens 20M (Harper & Konstan, 2016). This is a movie recommendation dataset that contains 20 million movie ratings from $n \approx 135$K users on $m_{\text{total}} \approx 27$K movies. Each rating $R^{\text{raw}}_{u,i}$ of the movie i by the user u is a value in the set $\{0.5, 1.0, 1.5, \ldots, 5.0\}$. We convert the ratings to integer values between 0 and 9 and select all movies with at least 5K ratings yielding $m \approx 1$K movies. We model the zeros as in implicit matrix factorization (Gopalan et al., 2015). We use Poisson matrix factorization to model this data. Assume there is a latent representation $\boldsymbol{\beta}_u \in \mathbb{R}^d$ for each user u and there is a latent representation $\boldsymbol{\theta}_i \in \mathbb{R}^d$ for each movie i. Here $d = 100$ is the latent variable dimensionality. Denote $\text{softplus}(t) = \log(1 + e^t)$. We model the likelihood as

$$p(R \mid \boldsymbol{\theta}, \boldsymbol{\beta}) = \prod_{u=1}^{n} \prod_{i=1}^{m} \text{Poisson}(R_{u,i} \mid \mu = \text{softplus}(\boldsymbol{\beta}_u^\top \boldsymbol{\theta}_i)).$$

We do variational inference on the user latent variable $\boldsymbol{\beta}$ and treat the movie variables $\boldsymbol{\theta}$ as model parameters. The prior on each user latent variable is a standard Normal. We set the variational distribution as $q(\boldsymbol{\beta} \mid R; \boldsymbol{\lambda}) = \prod_{u=1}^{n} q(\boldsymbol{\beta}_u \mid \mathbf{R}_u; \boldsymbol{\lambda})$, where $q(\boldsymbol{\beta}_u \mid \mathbf{R}_u; \boldsymbol{\lambda})$ uses an inference network that takes as input the row u of the rating matrix R. Similar to the VAE experiment, we use a 3-layer feed-forward neural network. We use 300 hidden units for this experiment.

Notice that the above likelihood is exactly a 1-layer feedforward neural network (without the bias term) that takes the latent representations drawn from the variational distribution $q(\boldsymbol{\beta} \mid R; \boldsymbol{\lambda})$ and outputs the rating matrix as a random matrix with a pointwise Poisson likelihood. Hence, we could view the model as a single-layer generative network and treat the latent variable $\boldsymbol{\theta}$ as its parameter. We have transformed variational matrix factorization to a task similar to the VAE. Hence, when we apply Algorithm 1 to this model, we can apply the same tricks used in the VAE experiments to accelerate the performances. We treat the whole model as a 4-layer feedforward neural network and again apply the tridiagonal block-wise K-FAC approximation (Martens & Grosse, 2015) and adopt low-rank approximations of large matrices (again, more details in appendix).

The results are shown in Figure 3. We randomly split the data matrix R into train and test sets where the train set contains 90% of the rows of R (it contains ratings from 90% of the users) and the test set contains the remaining rows. The test ELBO values are computed over the random sampled test set and the train ELBO values are computed over a fixed subset (with its size equal to the test set size) of the whole train set. Since this dataset is larger, we use a batch size of 3000. As can be seen in this figure, the VPNG updates outperform the baseline optimizations on both the train and test learning curves. The curves look slightly different among various train/test splits of the dataset but Algorithm 1 consistently outperforms the baseline methods. The difference stems from the correlations in the ratings of the movies. The traditional natural gradient performs the worst at the beginning since it is only guaranteed to perform well at the end (when $q(\mathbf{z} \mid \mathbf{x})$ is close to the posterior distribution $p(\mathbf{z} \mid \mathbf{x})$, Equation 3 explains this), but not necessarily at the beginning, due to it does not consider potential curvature information in the model distribution. Across both experiments, we find that VPNG dramatically improves estimation and inference at early iterations.

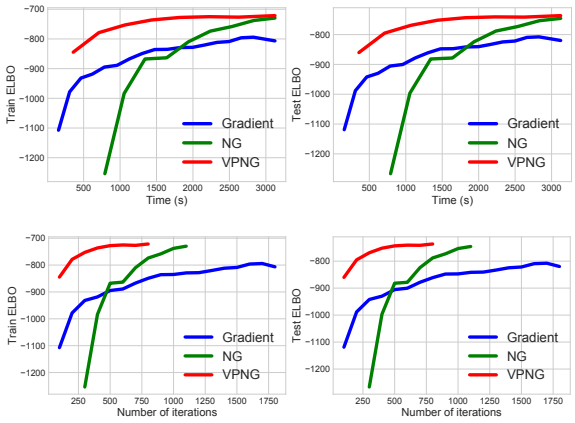

Figure 3. VMF Learning curves on MovieLens 20M

6. CONCLUSION

We introduced the variational predictive natural gradients. They adjust for parameter dependencies in variational inference induced by correlations in the observations. We show how to approximate the Fisher information without manual model-specific computations. We demonstrate the insight on a bivariate Gaussian model and the empirical value on a classification model on synthetic data, a deep generative model of images, and matrix factorization for movie recommendation. Future work includes extending to general Bayesian networks with multiple stochastic layers.

Acknowledgements

We want to thank Jaan Altosaar, Bharat Srikishan, Dawen Liang and Scott Linderman for their helpful comments and suggestions on this paper.

References

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

Ba, J., Grosse, R., and Martens, J. Distributed second-order optimization using kronecker-factored approximations. 2016.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A., Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 10–21, 2016.

Carbonetto, P., Stephens, M., et al. Scalable variational inference for bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian analysis*, 7(1):73–108, 2012.

Fan, K., Wang, Z., Beck, J., Kwok, J., and Heller, K. A. Fast second order stochastic backpropagation for variational inference. In *Advances in Neural Information Processing Systems*, pp. 1387–1395, 2015.

Gopalan, P., Hofman, J. M., and Blei, D. M. Scalable recommendation with hierarchical poisson factorization. In *UAI*, pp. 326–335, 2015.

Grosse, R. and Martens, J. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pp. 573–582, 2016.

Harper, F. M. and Konstan, J. A. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2016.

Harrison, L. M. and Green, G. G. A bayesian spatiotemporal model for very large data sets. *NeuroImage*, 50(3):1126–1141, 2010.

Hoffman, M., Blei, D., Wang, C., and Paisley, J. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Jordan, M., Ghahramani, Z., Jaakkola, T., and Saul, L. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

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

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Liang, D., Charlin, L., McInerney, J., and Blei, D. M. Modeling user exposure in recommendation. In *Proceedings of the 25th International Conference on World Wide Web*, pp. 951–961. International World Wide Web Conferences Steering Committee, 2016.

Manning, J. R., Ranganath, R., Norman, K. A., and Blei, D. M. Topographic factor analysis: a bayesian model for inferring brain networks from neural data. *PloS one*, 9 (5):e94914, 2014.

Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pp. 2408–2417, 2015.

Miao, Y., Yu, L., and Blunsom, P. Neural variational inference for text processing. In *International Conference on Machine Learning*, pp. 1727–1736, 2016.

Mnih, A. and Gregor, K. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*, 2014.

Mnih, A. and Salakhutdinov, R. R. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pp. 1257–1264, 2008.

Ollivier, Y., Arnold, L., Auger, A., and Hansen, N. Information-geometric optimization algorithms: A unifying picture via invariance principles. *arXiv preprint arXiv:1106.3708*, 2011.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *Artificial Intelligence and Statistics*, pp. 814–822, 2014.

Ranganath, R., Perotte, A., Elhadad, N., and Blei, D. Deep survival analysis. In *Machine Learning for Healthcare Conference*, pp. 101–114, 2016.

Regier, J., Jordan, M. I., and McAuliffe, J. Fast black-box variational inference through stochastic trust-region optimization. In *Advances in Neural Information Processing Systems*, pp. 2399–2408, 2017.

Rezende, D., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pp. 1278–1286, 2014.

Shi, T., Tang, D., Xu, L., and Moscibroda, T. Correlated compressive sensing for networked data. In *UAI*, pp. 722–731, 2014.

Stegle, O., Parts, L., Durbin, R., and Winn, J. A bayesian framework to account for complex non-genetic factors in gene expression levels greatly increases power in eqtl

studies. *PLoS computational biology*, 6(5):e1000770, 2010.

Thomas, P., Silva, B. C., Dann, C., and Brunskill, E. Energetic natural gradient descent. In *International Conference on Machine Learning*, pp. 2887–2895, 2016.

Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4 (2):26–31, 2012.

Titsias, M. and Lázaro-Gredilla, M. Doubly stochastic variational bayes for non-conjugate inference. In *International Conference on Machine Learning*, pp. 1971–1979, 2014.