Handling the Positive-Definite Constraint in the Bayesian Learning Rule

Wu Lin [1] Mark Schmidt [1 2] Mohammad Emtiyaz Khan [3]

Abstract

The Bayesian learning rule is a natural-gradient variational inference method, which not only contains many existing learning algorithms as special cases but also enables the design of new algorithms. Unfortunately, when variational parameters lie in an open constraint set, the rule may not satisfy the constraint and requires line-searches which could slow down the algorithm. In this work, we address this issue for positive-definite constraints by proposing an improved rule that naturally handles the constraints. Our modification is obtained by using Riemannian gradient methods, and is valid when the approximation attains a *block-coordinate natural parameterization* (e.g., Gaussian distributions and their mixtures). Our method outperforms existing methods without any significant increase in computation. Our work makes it easier to apply the rule in the presence of positive-definite constraints in parameter spaces.

1. Introduction

The Bayesian learning rule, a recently proposed method, enables derivation of learning algorithms from Bayesian principles (Khan & Rue, 2020). It is a natural-gradient variational inference method (Khan & Lin, 2017) where, by carefully choosing a posterior approximation, we can derive many algorithms in fields such as probabilistic graphical models, continuous optimization, and deep learning. Khan & Lin (2017) derive approximate inference methods, such as stochastic variational inference and variational message passing; Khan et al. (2018) derive connections to deep-learning algorithms; and Khan & Rue (2020) derive many classical algorithms such as least-squares, gradient descent, Newton's method, and the forward-backward algorithm. We can also design new algorithms using this rule such as uncertainty estimation in deep learning (Osawa et al., 2019) and the ensemble of Newton methods (Lin et al., 2019a).

[1]University of British Columbia, Vancouver, Canada. [2]CIFAR AI Chair, Alberta Machine Intelligence Institute, Canada. [3]RIKEN Center for Advanced Intelligence Project, Tokyo, Japan. Correspondence to: Wu Lin <wlin2018@cs.ubc.ca>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

An issue with the rule is that when parameters of an posterior approximation lie in an open constraint set, the update may not always satisfy the constraints. For Gaussian approximations, the posterior covariance needs to be positive definite but the rule may violate this; see Appendix D.1 in Khan et al. (2018) for detail. A straightforward solution is to use a backtracking line-search to keep the updates within the constraint set (Khan & Lin, 2017), but this can lead to slow convergence. In some cases, we can find an approximate update which always satisfies the constraints, e.g., for Gaussian approximations (Khan et al., 2018). However, in general, it is difficult to come up with such approximations that are both fast and reasonably accurate. Our goal in this paper is to modify the Bayesian learning rule so that it can naturally handle such constraints.

We propose an improved Bayesian learning rule to handle the positive-definite constraints. This is obtained by using a generalization of natural-gradient methods called Riemannian-gradient methods. We show that, for many useful approximations with a specific block-diagonal structure on the Fisher information matrix, the constraints are satisfied after an additional term is added to the rule. Such a structure is possible when the parameters of the approximation are partitioned in what we call the *block-coordinate natural (BCN) parameterizations*. Fortunately, for many approximations with such parameterizations, the improved rule requires almost the same computation as the original rule. An example is shown in Figure 1 where our improved rule fixes an implementation issue with an algorithm proposed by Osawa et al. (2019) for deep learning. We present examples where the improved rule converges faster than the original rule and many existing baseline methods.

2. Bayesian Learning Rule

Given a dataset $\mathcal{D}$, it is common to estimate unknown variables $\mathbf{z}$ of a statistical model by minimizing[1] $\bar{\ell}(\mathbf{z}) \equiv \ell(\mathcal{D}, \mathbf{z}) + R(\mathbf{z})$ where $\ell(\mathcal{D}, \mathbf{z})$ is a loss function and $R(\mathbf{z})$

[1]We assume $\nabla_z \bar{\ell}(\mathbf{z})$ and $\nabla_z^2 \bar{\ell}(\mathbf{z})$ exist almost surely whenever they are needed. ∇ denotes the standard derivative in this paper.

```
Variational Online Gauss-Newton (VOGN) Algorithm
1: $\mathbf{z} \leftarrow \boldsymbol{\mu} + (N\hat{\mathbf{s}})^{-1/2} \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: Randomly sample a minibatch $\mathcal{M}$ of size $M$
3: Compute and store individual gradients $\mathbf{g}_i, \forall i \in \mathcal{M}$
4: $\mathbf{g}_\mu \leftarrow \frac{\lambda}{N}\boldsymbol{\mu} + \frac{1}{M}\sum_{i=1}^M \mathbf{g}_i$
5: $\mathbf{m} \leftarrow r_1\,\mathbf{m} + (1-r_1)\,\mathbf{g}_\mu, \quad \bar{\mathbf{m}} \leftarrow \mathbf{m}/(1-r_1^k)$
6: $\mathbf{g}_s \leftarrow \frac{\lambda}{N} - \hat{\mathbf{s}} + \frac{1}{M}\sum_{i=1}^M (\mathbf{g}_i \odot \mathbf{g}_i)$
7: $\hat{\mathbf{s}} \leftarrow \hat{\mathbf{s}} + (1-r_2)\,\mathbf{g}_s$
8: $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - t\;\bar{\mathbf{m}}/\bar{\mathbf{s}}$, where $\bar{\mathbf{s}} \leftarrow \hat{\mathbf{s}}/(1-r_2^k)$
```

```
Our Adam-like Optimizer
1: $\mathbf{z} \leftarrow \boldsymbol{\mu} + (N\hat{\mathbf{s}})^{-1/2} \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: Randomly sample a minibatch $\mathcal{M}$ of size $M$
3: Compute a mini-batch gradient $\bar{\mathbf{g}} \leftarrow \frac{1}{M}\sum_{i=1}^M \mathbf{g}_i$
4: $\mathbf{g}_\mu \leftarrow \frac{\lambda}{N}\boldsymbol{\mu} + \bar{\mathbf{g}}$
5: $\mathbf{m} \leftarrow r_1\,\mathbf{m} + (1-r_1)\,\mathbf{g}_\mu, \quad \bar{\mathbf{m}} \leftarrow \mathbf{m}/(1-r_1^k)$
6: $\mathbf{g}_s \leftarrow \frac{\lambda}{N} - \hat{\mathbf{s}} + [(N\hat{\mathbf{s}}) \odot (\mathbf{z} - \boldsymbol{\mu})] \odot \bar{\mathbf{g}}$
7: $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - t\;\bar{\mathbf{m}}/\bar{\mathbf{s}}$, where $\bar{\mathbf{s}} \leftarrow \hat{\mathbf{s}}/(1-r_2^k)$
8: $\hat{\mathbf{s}} \leftarrow \hat{\mathbf{s}} + (1-r_2)\,\mathbf{g}_s + \frac{1}{2}(1-r_2)^2\mathbf{g}_s \odot \hat{\mathbf{s}}^{-1} \odot \mathbf{g}_s$
```

Figure 1. Our improved Bayesian learning rule solves an implementation issue with an existing algorithm known as VOGN (Khan et al., 2018) (shown in the left). VOGN is an Adam-like optimizer which gives state-of-the-art results on large deep learning problems (Osawa et al., 2019). However, it requires us to store individual gradients in a minibatch which makes the algorithm slow (shown with blue in line 3 and 6). This is necessary for the scaling vector $\hat{\mathbf{s}}$ to obtain a good estimate of uncertainty. Our work in this paper fixes this issue using the improved Bayesian learning rule. Our Adam-like optimizer (shown in the right) only requires average over the minibatch (see line 3). Line 6 is simply changed to use the re-parametrization trick with the averaged gradient. The additional terms added to the Bayesian learning rule is shown in red in line 8. These changes do not increase the computation cost significantly while fixing the implementation issue of VOGN. Due to our modification, the scaling vector $\hat{\mathbf{s}}$ always remains positive. A small difference is that the mean $\boldsymbol{\mu}$ is updated before in our optimizer (see line 7 and 8), while in VOGN it is the opposite. The difference shows that NGD depends on parameterization.

is a regularizer. Many estimation strategies can be used, giving rise to various learning algorithms. E.g., maximum-likelihood approaches use gradient-based methods such as gradient descent and Newton's method, while Bayesian approaches use inference algorithms such as message passing.

Khan & Rue (2020) showed that many learning algorithms can be obtained from Bayesian principles. The key idea is to use the following Bayesian formulation where, instead of minimizing over $\mathbf{z}$, we minimize over a distribution $q(\mathbf{z})$:

$$\min_{q(z)\in\mathcal{Q}} \mathbb{E}_{q(z)}[\ell(\mathcal{D},\mathbf{z})] + \mathbb{D}_{KL}[q(\mathbf{z}) \,\|\, p(\mathbf{z})] \equiv \mathcal{L}(q). \qquad (1)$$

Here, $q(\mathbf{z})$ is an approximation of the posterior of $\mathbf{z}$ given $\mathcal{D}$, $\mathcal{Q}$ is the set of approximation distributions, $p(\mathbf{z}) \propto \exp(-R(\mathbf{z}))$ is the prior, and $\mathbb{D}_{KL}$ denotes the Kullback-Leibler divergence. To obtain existing learning algorithms from the above formulation, we need to carefully choose the approximation family $\mathcal{Q}$. Khan & Rue (2020) consider the following *minimal* exponential family (EF) distribution:

$$q(\mathbf{z}|\boldsymbol{\lambda}) := h(\mathbf{z})\exp\left[\langle \boldsymbol{\phi}(\mathbf{z}), \boldsymbol{\lambda}\rangle - A(\boldsymbol{\lambda})\right]$$

where $\boldsymbol{\phi}(\mathbf{z})$ is a vector containing sufficient statistics, $h(\mathbf{z})$ is the base measure, $\boldsymbol{\lambda} \in \Omega$ is the natural parameter, Ω is the set of valid natural-parameters so that the log-partition function $A(\boldsymbol{\lambda})$ is finite, and $\langle\cdot,\cdot\rangle$ denotes an inner product.

Khan & Rue (2020) present the Bayesian learning rule to optimize (1), which is a natural-gradient descent (NGD) update originally proposed by Khan & Lin (2017) for variational inference. The update takes the following form:

$$\text{NGD}: \quad \boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} - t\hat{\boldsymbol{g}}, \text{ with } \hat{\boldsymbol{g}} := \mathbf{F}^{-1}\partial_\lambda \mathcal{L}(\boldsymbol{\lambda}) \qquad (2)$$

where $t > 0$ is a scalar step-size and $\hat{\boldsymbol{g}}$ is the natural gradient defined using the Fisher information matrix (FIM) $\mathbf{F} := -\mathbb{E}_q[\partial^2_\lambda \log q(\mathbf{z}|\boldsymbol{\lambda})]$ of q and $\mathcal{L}(\boldsymbol{\lambda})$ which is equal to $\mathcal{L}(q)$ but defined in terms of $\boldsymbol{\lambda}$. Khan & Rue (2020) proposed further simplifications, e.g., for approximations with base measure $h(\mathbf{z}) \equiv 1$, we can write (2) as

$$\boldsymbol{\lambda} \leftarrow (1-t)\boldsymbol{\lambda} - t\partial_m \mathbb{E}_q\left[\bar{\ell}(\mathbf{z})\right] \qquad (3)$$

where $\mathbf{m} := \mathbb{E}_{q(z)}[\boldsymbol{\phi}(\mathbf{z})]$ denotes the expectation parameter.

Existing learning algorithms can be derived as special cases by choosing an approximate form for $q(\mathbf{z})$. For example, when $q(\mathbf{z}) := \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \mathbf{S}^{-1})$ is a multivariate Gaussian approximation with the mean $\boldsymbol{\mu}$ and the precision matrix $\mathbf{S}$, the learning rule (3) can be expressed as follows:

$$\mathbf{S} \leftarrow (1-t)\mathbf{S} + t\mathbb{E}_q\left[\nabla_z^2 \bar{\ell}(\mathbf{z})\right] \qquad (4)$$

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - t\mathbf{S}^{-1}\mathbb{E}_q\left[\nabla_z \bar{\ell}(\mathbf{z})\right] \qquad (5)$$

This algorithm uses the Hessian to update $\mathbf{S}$ which is then used to scale the update for $\boldsymbol{\mu}$, in a similar fashion as Newton's method. The main difference here is that the gradient and Hessian are obtained at samples from $q(\mathbf{z})$ instead of the current iterate $\boldsymbol{\mu}$. Khan & Rue (2020) approximate the expectation at $\boldsymbol{\mu}$ to obtain an online Newton method. This algorithm is closely related to deep-learning optimizers, such as, RMSprop and Adam (Khan et al., 2018; Zhang et al., 2018). A simplified version of this algorithm obtains state-of-the-art results on large deep-learning problems for uncertainty estimation as shown by Osawa et al. (2019).

Many other examples are discussed in Khan & Rue (2020), including algorithms such as stochastic gradient descent. The relationship to message passing algorithms and stochastic variational inference is shown in Khan & Lin (2017). In summary, the Bayesian learning rule is a generic learning rule that can be used not only to derive existing algorithms, but also to improve them and design new ones.

2.1. Positive-Definite Constraints

An issue with updates (2) and (3) is that the constraint $\boldsymbol{\lambda} \in \Omega$ is not taken into account, where Ω is the set of valid parameters. The update is valid when Ω is unconstrained (e.g., a Euclidean space), but otherwise it may violate the constraint. An example is the multivariate Gaussian of dimension d where the precision matrix $\mathbf{S} \in \mathbb{S}_{++}^{d\times d}$ is required to be real and positive-definite, while the mean $\boldsymbol{\mu} \in \mathbb{R}^d$ is unconstrained. In such cases, the update may violate the constraint. E.g., in the update (4), $\mathbf{S}$ can be indefinite, when the loss $\bar{\ell}(\mathbf{z})$ is nonconvex. A similar issue appears when flexible approximations are used such as Gaussian mixtures.

Another example is a gamma distribution: $q(z|\alpha,\beta) \propto z^{\alpha-1}e^{-z\beta}$ where both $\alpha, \beta > 0$. We denote the positivity constraint using $\mathbb{S}_{++}^1$. The rule takes the following form:

$$\alpha \leftarrow (1-t)\alpha - t\hat{g}_\alpha, \qquad \beta \leftarrow (1-t)\beta - t\hat{g}_\beta \tag{6}$$

where $\hat{g}_\alpha$ and $\hat{g}_\beta$ are gradient of $\mathbb{E}_{q(z)}\left[\bar{\ell}(z) - \log z\right]$ with respect to the expectation parameters $m_\alpha = \mathbb{E}_{q(z)}[\log z]$ and $m_\beta = \mathbb{E}_{q(z)}[-z]$ respectively; see a detailed derivation in Appedix E.3 in Khan & Lin (2017). Here again the learning rule does not ensure that α and β are always positive.

In general, a backtracking line search proposed in Khan & Lin (2017) can be used so that the iterates stay within the constraint set. However, this could be slow in practice. Khan et al. (2018) discuss this issue for the Gaussian case; see Appendix D.1 in their paper. They found that using line-search is computationally expensive and non-trivial to implement for deep-learning problems. They address this issue by approximating the Hessian in (4) with a positive-definite matrix. This ensures that $\mathbf{S}$ is always positive-definite. However, such approximations are difficult to come up with for general cases. E.g., for the gamma case, there is no such straight-forward approximation in update (6) to ensure positivity of α and β. It is also possible to use an unconstrained transformation (e.g., a Cholesky factor). This approach uses automatic-differentiation (Auto-Diff), which can be much slower than explicit gradient forms (see the discussion in Section 4). Handling constraints within the Bayesian learning rule is an open issue which limits its applications.

In this paper, we focus on positive-definite constraints and show that, in many cases, such constraints can be naturally handled by adding an additional term to the Bayesian learning rule. We show that, for this to happen, the approximation needs to follow a specific parameterization. We will now describe the modification in the next section, and later give its derivation using Riemannian gradient methods.

3. Improved Bayesian Learning Rule

We will give a new rule to handle the positive-definite constraints. Our idea is to partition the parameter into blocks so that each constraint is isolated in an individual block.

Assumption 1 [Mutually-Exclusive Constraints] : *We assume parameter $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}^{[1]}, \dots, \boldsymbol{\lambda}^{[m]}\}$ can be partitioned into m blocks with mutually-exclusive constraints $\Omega = \Omega_1 \times \cdots \times \Omega_m$, where square bracket $[i]$ denotes the i-th block and each block $\boldsymbol{\lambda}^{[i]}$ is either unconstrained or positive-definite.*

For example, consider multivariate Gaussian approximations with the two blocks: one block containing the mean $\boldsymbol{\mu}$ and another containing the full precision $\mathbf{S}$. This satisfies the above assumption because the first block is unconstrained and the second block is positive definite. In d-dimensional diagonal Gaussian cases, we consider $2d$ blocks: one block containing the mean μ_i and one block containing the precision s_i for each dimension i, where each s_i is positive. Other examples such as gammas and inverse Gaussians can be partitioned to two blocks, where each block is positive.

Assumption 2 [Block Coordinate Parameterization] : *A parameterization satisfied Assumption 1 is block coordinate (BC) if the FIM is block-diagonal according to the block structure of the parameterization.*

For Gaussians, using the mean and the covariance/precision as two blocks is a BC parameterization (see Appendix E), while the natural parameterization is not (Malagò & Pistone, 2015). For EFs, we could use the Crouzeix identity (Nielsen, 2019) to identify a BC parameterization.

Assumption 3 [Block Natural Parameterization for EF] : *For $q(\mathbf{z}|\boldsymbol{\lambda})$ and each block $\boldsymbol{\lambda}^{[i]}$, there exist function ϕ_i and h_i such that $q(\mathbf{z}|\boldsymbol{\lambda})$ can be re-expressed as a minimal EF distribution given that the rest of blocks $\boldsymbol{\lambda}^{[-i]}$ are known.*

$$q(\mathbf{z}|\boldsymbol{\lambda}) \equiv h_i\left(\mathbf{z}, \boldsymbol{\lambda}^{[-i]}\right) \exp\left[\left\langle \boldsymbol{\phi}_i\left(\mathbf{z}, \boldsymbol{\lambda}^{[-i]}\right), \boldsymbol{\lambda}^{[i]}\right\rangle - A(\boldsymbol{\lambda})\right]$$

Lin et al. (2019a) originally use Assumption 3 to define a multilinear EF. We illustrate this assumption on the Gaussian distribution which can be written as the following exponential form, where $A(\boldsymbol{\mu}, \mathbf{S}) = \frac{1}{2}\left[\boldsymbol{\mu}^T\mathbf{S}\boldsymbol{\mu} - \log|\mathbf{S}/(2\pi)|\right]$ is the log-partition function.

$$q(\mathbf{z}|\boldsymbol{\mu}, \mathbf{S}) = \exp\left(-\tfrac{1}{2}\mathbf{z}^T\mathbf{S}\mathbf{z} + \mathbf{z}^T\mathbf{S}\boldsymbol{\mu} - A(\boldsymbol{\mu}, \mathbf{S})\right)$$

Considering two blocks with $\boldsymbol{\mu}$ and $\mathbf{S}$ respectively, we can express this distribution as follows, where the first equation is for $\boldsymbol{\mu}$ while the second equation is for $\mathbf{S}$:

$$\begin{aligned} q(\mathbf{z}|\boldsymbol{\mu}, \mathbf{S}) &= \underbrace{\exp(-\tfrac{1}{2}\mathbf{z}^T\mathbf{S}\mathbf{z})}_{h_1(\mathbf{z},\mathbf{S})} \exp\left(\langle \underbrace{\mathbf{S}\mathbf{z}}_{\phi_1(\mathbf{z},\mathbf{S})}, \boldsymbol{\mu}\rangle - A(\boldsymbol{\mu}, \mathbf{S})\right) \\ &= \underbrace{1}_{h_2(\mathbf{z},\boldsymbol{\mu})} \exp\left(\langle \underbrace{-\tfrac{1}{2}\mathbf{z}\mathbf{z}^T + \boldsymbol{\mu}\mathbf{z}^T}_{\phi_2(\mathbf{z},\boldsymbol{\mu})}, \mathbf{S}\rangle - A(\boldsymbol{\mu}, \mathbf{S})\right) \end{aligned}$$

We define the *block-coordinate natural (BCN) parameterization* for an EF distribution as the parameterization which satisfies Assumptions from 1 to 3. Therefore, Gaussian distribution with $\boldsymbol{\mu}$ and $\mathbf{S}$ can be expressed in a BCN parameterization $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}^{[1]}, \boldsymbol{\lambda}^{[2]}\}$, where $\boldsymbol{\lambda}^{[1]} = \boldsymbol{\mu}$ and $\boldsymbol{\lambda}^{[2]} = \mathbf{S}$. Let λ^{a_i} denote the a-th entry of the i-th block parameter $\boldsymbol{\lambda}^{[i]}$, where a_i is a local index for block i. $\hat{g}^{c_i}$ is the c-th entry of natural gradient $\hat{\boldsymbol{g}}^{[i]}$ with respect to $\boldsymbol{\lambda}^{[i]}$.

We now present the rule (see Section 5 for a derivation). Under a BCN parameterization $\boldsymbol{\lambda}$, our rule for block i takes the following form with an extra term shown in red:

$$\lambda^{c_i} \leftarrow \lambda^{c_i} - t\hat{g}^{c_i} - \frac{t^2}{2}\sum_{a_i}\sum_{b_i}\Gamma^{c_i}{}_{a_i b_i}\hat{g}^{a_i}\hat{g}^{b_i}, \tag{7}$$

where each summation is to sum over all entries of the i-th block, $\Gamma^{c_i}{}_{a_i b_i} := \frac{1}{2}\partial_{m_{c_i}}\partial_{\lambda^{a_i}}\partial_{\lambda^{b_i}} A(\boldsymbol{\lambda})$, and m_{c_i} is the c-th entry of the BC expectation parameter $\mathbf{m}_{[i]} := \mathbb{E}_q\big[\boldsymbol{\phi}_i(\mathbf{z}, \boldsymbol{\lambda}^{[-i]})\big] = \partial_{\lambda^{[i]}} A(\boldsymbol{\lambda})$.

The modification involves computation of the third-order term of the log-partition function[2] $A(\boldsymbol{\lambda})$. In the following Section 3.1 and 3.2, we discuss two examples where this computation is simplified and can be carried out like the original rule with minimal computational increase.

Table 2 in Appendix C lists more examples satisfying Assumption 1-3, where our rule can be applied and simplified.

3.1. Example: Online Newton using Gaussian Approximation

The original rule for Gaussian approximations gives the update (4)-(5), where the natural parameterization of Gaussian is used. We consider the parameterization $\boldsymbol{\mu}$ and $\mathbf{S}$, in which the improved rule takes the form (a detailed simplification is in Appendix E) with an extra non-zero term shown in red:

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - t\mathbf{S}^{-1}\mathbb{E}_q\big[\nabla_z \bar{\ell}(\mathbf{z})\big] + \mathbf{0} \tag{8}$$

$$\mathbf{S} \leftarrow (1-t)\mathbf{S} + t\mathbb{E}_q\big[\nabla_z^2 \bar{\ell}(\mathbf{z})\big] + \frac{t^2}{2}\hat{\mathbf{G}}\mathbf{S}^{-1}\hat{\mathbf{G}}, \tag{9}$$

where $\hat{\mathbf{G}} := \mathbf{S} - \mathbb{E}_q\big[\nabla_z^2 \bar{\ell}(\mathbf{z})\big]$. The extra term ensures that the positive definite constraint is satisfied due to Theorem 1.

Theorem 1 *The updated* $\mathbf{S}$ *in* (9) *is positive definite if the initial* $\mathbf{S}$ *is positive-definite.*

The proof of Theorem 1 can be found in Appendix E.1.

Although (8)-(9) appear similar to (4)-(5), there is one difference – the old $\mathbf{S}$ is used as a preconditioner to update $\boldsymbol{\mu}$. Note that (8)-(9) becomes a natural-gradient descent (NGD) update if we ignore the additional term. Though natural gradient[3] is invariant to parameterization, NGD update depends on parameterization as shown by the difference. However, we expect the difference to make a small change in practice.

Like Khan & Rue (2020), an online Newton method can be obtained by approximating the expectations at $\boldsymbol{\mu}$, e.g., $\mathbb{E}_q\big[\nabla_z \bar{\ell}(\mathbf{z})\big] \approx \nabla_\mu \bar{\ell}(\boldsymbol{\mu})$ and $\mathbb{E}_q\big[\nabla_z^2 \bar{\ell}(\mathbf{z})\big] \approx \nabla_\mu^2 \bar{\ell}(\boldsymbol{\mu})$. In this case, the algorithm converges to a local minimal of the loss $\bar{\ell}(\mathbf{z})$. A key point is that, unlike Newton's method where the preconditioner may not be positive-definite for nonconvex functions, $\mathbf{S}$ is guaranteed to be positive definite.

When applied to factorized Gaussians, these updates give an improved version of the Variational Online Gauss-Newton (VOGN) algorithm in Osawa et al. (2019). It is shown in Figure 1 where the differences in our algorithm are shown in red. Our algorithm uses the reparameterization trick to avoid computing $\nabla_z^2 \bar{\ell}(\mathbf{z})$ in (9). A derivation is given in Appendix E.3. Our algorithm fixes an implementation issue in VOGN without comprising its performance and speed, where our update only stores a mini-batch gradient while VOGN has to store all individual gradients in a mini-batch.

3.2. Example: Gamma Approximation

Let's consider gamma cases. We use a BCN parameterization $\boldsymbol{\lambda} = \{\lambda^{[1]}, \lambda^{[2]}\}$ (see Appendix F for detail), where $\lambda^{[1]} = \alpha$ and $\lambda^{[2]} = \frac{\beta}{\alpha}$. The constraint is $\Omega = \mathbb{S}^1_{++} \times \mathbb{S}^1_{++}$. Since each block contains a scalar, we use global indexes as $\lambda^{(i)} = \lambda^{[i]}$ and $\hat{g}^{(i)} = \hat{g}^{[i]}$. Moreover, we use $\Gamma^i{}_{ii}$ to denote $\Gamma^{c_i}{}_{a_i b_i}$. Let $\mathrm{Ga}(\cdot)$ be the gamma function. Under this parameterization, a gamma distribution is expressed as:

$$q(z|\boldsymbol{\lambda}) = \frac{1}{z}\exp\Big(\lambda^{(1)}\log z - z\lambda^{(1)}\lambda^{(2)} - A(\boldsymbol{\lambda})\Big),$$

where $A(\boldsymbol{\lambda}) = \log \mathrm{Ga}(\lambda^{(1)}) - \lambda^{(1)}\left(\log\lambda^{(1)} + \log\lambda^{(2)}\right)$.

Let $\psi(\cdot)$ be the digamma function. We can compute the third derivatives (see Appendix F for a derivation) as:

$$\Gamma^1{}_{11} = \frac{\frac{1}{\lambda^{(1)}\times\lambda^{(1)}} + \partial^2_{\lambda^{(1)}}\psi(\lambda^{(1)})}{2\left(-\frac{1}{\lambda^{(1)}} + \partial_{\lambda^{(1)}}\psi(\lambda^{(1)})\right)}, \quad \Gamma^2{}_{22} = -\frac{1}{\lambda^{(2)}}$$

The proposed rule in this case is

$$\lambda^{(i)} \leftarrow \lambda^{(i)} - t\hat{g}^{(i)} - \frac{t^2}{2}\left(\Gamma^i{}_{ii}\right)\hat{g}^{(i)} \times \hat{g}^{(i)}, \quad i = 1, 2 \tag{10}$$

where each $\hat{g}^{(i)}$ is a natural gradient computed via the implicit re-parameterization trick as shown in Appendix F.2.

Theorem 2 *The updated* $\lambda^{(i)}$ *in* (10) *is positive if the initial* $\lambda^{(i)}$ *is positive.*

The proof of Theorem 2 can be found in Appendix F.1.

[2] We assume $A(\boldsymbol{\lambda})$ is (jointly) C^3-smooth. Note that $A(\boldsymbol{\lambda})$ is block-wisely C^3-smooth as shown in Johansen (1979). Approximations considered in this paper satisfy this assumption.

[3] It is a representation of an abstract (parameterization-free) tangent vector in a Riemannian manifold under a parameterization.

3.3. Extension to EF Mixtures

Our learning rule can be extended to mixture approximations, such as finite mixture of Gaussians (MOG) (shown in Appendix J) and skew Gaussian approximations (given in Appendix K) using the joint FIM – the FIM of the joint distribution of a mixture – as suggested by Lin et al. (2019a). By extending the definition of the BCN parameterization to the joint distribution, our rule can be easily applied to mixture cases (see Appendix I for detail). For example, our update for MOG approximation can be found at (27) in Appendix J, where our rule handles the positive-definite constraints in MOG. Our update can be viewed as an improved version of the ensemble of Newton methods proposed by Lin et al. (2019a). We also discuss why it is non-trivial to extend VOGN to MOG cases in Appendix J.1.

4. Related Works

In d-dim Gaussian cases, we can use unconstrained transformations (e.g., a Cholesky factor). However, the natural-gradient computation becomes complicated. Eq (2) gives $O(d^6)$ for direct computation. Salimbeni et al. (2018) propose an indirect approach via two additional vector-Jacobian products (VJPs), which could give $O(d^4)$. For some parameterizations, their approach gives an implicit $O(d^3)$ update, where Auto-Diff is needed to track non-zero terms in the additional Jacobians and to simplify the VJPs. Contrarily, our method gives a simple and explicit $O(d^3)$ update and builds a direct connection to Newton's method. In practice, our iterative update is 20 times faster than theirs in CPU time from $d = 10^1$ to $d = 10^4$ if both use Auto-Diff. Our approach is also easily extended to EFs and mixture cases.

Our work is closely related to the method of Tran et al. (2019). They propose a method based on a retraction map in Gaussian cases, which is a special case of ours (see Appendix E.4). However, their retraction map does not directly generalize to other distributions, while ours does. They do not provide a justification or derivation of the map. We fix this gap by deriving the map from first principles, justifying its use, and obtaining an Adam-like update by choosing a proper parametrization for Gaussian cases (see Appendix E). Moreover, in Tran et al. (2019), the retraction map and Riemannian gradients used in neural network cases are not derived from the same Riemannian metric. In our work, a retraction map induced by the proposed rule and Riemannian gradients are naturally derived from the same metric.

Song et al. (2018) give a similar update in non-Bayesian contexts, but the update does not always satisfy the constraints for univariate Gaussians (see Appendix A). Their update is neither simple nor efficient for multivariate cases such as multivariate Gaussians and MOGs (see Section 5.3).

Hosseini & Sra (2015) use a similar approach to ours but for parameter estimation of Gaussian mixtures. They propose a transformation for each Gaussian component so that the mean and the covariance together can be re-parameterized as an augmented positive-definite matrix with an extra constraint.[4] They show that a local minimum of the negative log-likelihood of a mixture automatically satisfies the extra constraint. Thus, they can employ Riemannian-gradient descent with a retraction map to update the augmented positive-definite matrix, where the extra constraint can be safely ignored. It is unclear if this approach generalizes to variational inference settings since MOG approximations require the extra constraint is satisfied at each iteration to generate samples, and thus the constraint cannot be omitted. Moreover, it is unclear whether Riemannian gradients and the retraction are derived from the same metric for the mixture since they are only designed for positive-definite matrix manifolds instead of MOGs. In this work, we derive them from the joint Fisher metric for MOGs in a principled way.

5. Derivation of the Improved Rule

5.1. Gradient Descent

We first review gradient descent in Euclidean spaces and generalize it to Riemannian manifolds, where we derive our rule. Recall that we want to minimize (1) in terms of $\boldsymbol{\lambda}$ as:

$$\min_{\lambda\in\Omega} \mathbb{E}_{q(z|\lambda)}[\ell(\mathcal{D},\mathbf{z})] + \mathbb{D}_{KL}[q(\mathbf{z}|\boldsymbol{\lambda}) \,\|\, p(\mathbf{z})] \equiv \mathcal{L}(\boldsymbol{\lambda}).$$

If $\Omega = \mathbb{R}^d$ is a Euclidean space,[5] we can solve the minimization problem using gradient descent (GD) as:

$$\text{GD}: \quad \boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} - t\mathbf{g}$$

where $\mathbf{g} = \partial_\lambda \mathcal{L}(\boldsymbol{\lambda})$ denotes a Euclidean gradient and $t > 0$ is a scalar step-size. We can view the update as a line (the shortest curve) $\mathbf{L}(t)$ in the Euclidean space $\mathbb{R}^d$ as t varies. Given a starting point $\boldsymbol{\lambda}$ and a Euclidean direction $-\mathbf{g}$, the line is a differentiable map $\mathbf{L}(t)$ so that the following ordinary differential equation[6] (ODE) is satisfied.

$$\dot{\mathbf{L}}(0) = -\mathbf{g}\ ;\ \ \mathbf{L}(0) = \boldsymbol{\lambda};\ \ \ddot{\mathbf{L}}(t) = \mathbf{0} \qquad (11)$$

where $\dot{\mathbf{L}}(x) := \frac{d\mathbf{L}(t)}{dt}\big|_{t=x}$, $\ddot{\mathbf{L}}(x) := \frac{d^2\mathbf{L}(t)}{dt^2}\big|_{t=x}$. The solution of the ODE is $\mathbf{L}(t) = \boldsymbol{\lambda} - t\mathbf{g}$, which is the GD update.

5.2. Exact Riemannian Gradient Descent (RGD)

Unfortunately, Ω usually is not a Euclidean space but a Riemannian manifold with a metric. A metric is used to characterize distances in a manifold. A useful metric for statistical manifolds is the FIM (Fisher, 1922; Rao, 1945).

[4]The parameterization violates Assumption 1 since the positive-definite constraint of the augmented matrix and the extra constraint can not be partitioned into two blocks.

[5]In this paper, it always uses a Cartesian coordinate system.

[6]It is also known as an initial value problem.

Table 1. Table of Index Notation

$\boldsymbol{\lambda}^{[i]}$	i-th block parameter of parameterization $\boldsymbol{\lambda}$.
λ^{a_i}	a-th entry of block parameter $\boldsymbol{\lambda}^{[i]}$.
$\lambda^a,\ \lambda^{(a)}$	a-th entry of parameterization $\boldsymbol{\lambda}$.
g_a	a-th entry of Euclidean gradient $\mathbf{g}$.
$\hat{g}^a,\ \hat{g}^{(a)}$	a-th entry of Riemannian/natural gradient $\hat{\boldsymbol{g}}$.
F_{ab}	entry of $\mathbf{F}$ with global index (a,b).
F^{ab}	entry of $\mathbf{F}^{-1}$ with global index (a,b).
$\Gamma^c{}_{ab}$	entry with global index (c,a,b).
$F^{a_ib_i}$	entry with local index (a,b) in block i.
$\Gamma^{c_i}{}_{a_ib_i}$	entry with local index (c,a,b) in block i.

Now, we generalize gradient descent in a manifold. First, we introduce the index convention and the Einstein summation notation used in Riemannian geometry. The notation is summarized in Table 1. We denote a Euclidean gradient $\mathbf{g}$ using a subscript. A Riemannian gradient $\hat{\boldsymbol{g}}$ is denoted by a superscript. A metric[7] is used to characterize inner products and arc length in a manifold. Given a metric $\mathbf{F}$, let F_{ab} denote the element of $\mathbf{F}$ at position (a,b) and F^{ca} denote the element of $\mathbf{F}^{-1}$ at position (c,a). We use the Einstein notation to omit summation symbols such as $F^{ca}F_{ab} := \sum_a F^{ca}F_{ab}$. Therefore, we have $F^{ca}F_{ab} = I^c_b$, where I^c_b is the element of an identity matrix at position (c,b). A Riemannian gradient is defined as $\hat{g}^c = F^{ca}g_a$, where g_a is the a-th element of a Euclidean gradient $\mathbf{g}$. When $\mathbf{F}$ is the FIM, a Riemannian gradient becomes a natural gradient. If the metric $\mathbf{F}$ is positive-definite for all[8] $\boldsymbol{\lambda} \in \Omega$, an approximation family $q(\mathbf{z}|\boldsymbol{\lambda})$ induces a Riemannian manifold denoted by $(\Omega, \mathbf{F})$ where $\boldsymbol{\lambda}$ is a coordinate system.

Like GD, RGD can be derived from a geodesic,[9] which is a generalization of the "shortest" curve[10] to a manifold. Given a starting point $\boldsymbol{\lambda} \in \Omega$ and a Riemannian direction $-\hat{\boldsymbol{g}} = -\mathbf{F}^{-1}\mathbf{g}$, a geodesic is a differentiable map $\mathbf{L}(t)$ so that the following geodesic ODE[11] is satisfied.

$$\dot{L}^c(0) = -F^{ca}g_a \ ; \ \ L^c(0) = \lambda^c \tag{12}$$

$$\ddot{L}^c(t) = -\Gamma^c{}_{ab}(t)\dot{L}^a(t)\dot{L}^b(t) \tag{13}$$

where $L^c(t)$ is the c-th element of $\mathbf{L}(t)$, $\dot{L}^c(x) := \frac{dL^c(t)}{dt}\big|_{t=x}$, $\ddot{L}^c(x) := \frac{d^2L^c(t)}{dt^2}\big|_{t=x}$, $\Gamma^c{}_{ab}(t) := \Gamma^c{}_{ab}\big|_{\lambda=L(t)}$. $\Gamma^c{}_{ab}$ is the *Christoffel symbol of the 2nd kind* defined by

$$\Gamma^c{}_{ab} := F^{cd}\Gamma_{d,ab} \ ; \ \ \Gamma_{d,ab} := \tfrac{1}{2}\left[\partial_a F_{bd} + \partial_b F_{ad} - \partial_d F_{ab}\right]$$

where $\partial_a := \partial_{\lambda^a}$ is for notation simplicity and $\Gamma_{d,ab}$ is the *Christoffel symbol of the 1st kind*. $\ddot{\mathbf{L}}$ characterizes the curvature of a geodesic since a manifold is not flat in general. In Euclidean cases, the metric $\mathbf{F} = \mathbf{I}$ is a constant identity matrix and (13) vanishes since $\Gamma_{d,ab}$ and $\Gamma^c{}_{ab}$ are zeros, which implies Euclidean spaces are flat. Therefore, we recover the GD update in (11) since $\hat{\boldsymbol{g}} = \mathbf{I}^{-1}\mathbf{g} = \mathbf{g}$.

Given any parameterization with the FIM, we can compute $\Gamma_{d,ab}$ by using Eq (17) in Appendix D.1, which involves extra integrations. We will show that a BCN parameterization can get rid of the integrations (see Theorem 3).

5.3. Our Rule as an Inexact RGD Update

However, it is hard to exactly solve the geodesic ODE. Inexact RGD is derived by approximating the geodesic.[12] Recall that the original rule is a natural gradient descent (NGD) update. NGD can be derived by the first-order approximation of the geodesic $\mathbf{L}(t)$ at $t_0 = 0$ with the FIM.

$$\text{NGD}: \ \ \boldsymbol{\lambda} \leftarrow \mathbf{L}(t_0) + \dot{\mathbf{L}}(t_0)(t - t_0) = \boldsymbol{\lambda} - t\hat{\boldsymbol{g}}$$

Unfortunately, this approximation is only well-defined in a small neighborhood at t_0 with radius t. For a stochastic NGD update, the step-size t is very small, which often result in slow convergence. Our learning rule addresses this issue, which is indeed a new inexact RGD update. Moreover, our update can use a bigger step-size and often converges faster than NGD without introducing significant computational overhead in useful cases such as gamma, Gaussian, MOG.

Consider cases when $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}^{[1]}, \dots, \boldsymbol{\lambda}^{[m]}\}$ has m blocks. We can express a Riemannian gradient as $\hat{\boldsymbol{g}} = \{\hat{\boldsymbol{g}}^{[1]}, \dots, \hat{\boldsymbol{g}}^{[m]}\}$. We use the block summation notation to omit the summation signs in (7) as $\Gamma^{c_i}{}_{a_ib_i}\hat{g}^{a_i}\hat{g}^{b_i} := \sum_{a_i}\sum_{b_i}\Gamma^{c_i}{}_{a_ib_i}\hat{g}^{a_i}\hat{g}^{b_i}$. By the global index notation, we have $\Gamma^{c_i}{}_{a_ib_i}\hat{g}^{a_i}\hat{g}^{b_i} = \sum_{a\in[i]}\sum_{b\in[i]}\Gamma^{(c_i)}{}_{ab}\hat{g}^a\hat{g}^b$, where $[i]$ is the index set for block i, (c_i) is the corresponding global index of local index c_i, and a and b are global indexes.

We can extend the definition of a BC parameterization to any Riemannian metric $\mathbf{F}$. Given a metric $\mathbf{F}$, we have Lemma 1 for any block i (see Appendix B.1 for a proof) :

Lemma 1 *When $\boldsymbol{\lambda}$ is a BC parameterization of metric* $\mathbf{F}$*, we have $\hat{g}^{a_i} = F^{a_ib_i}g_{b_i}$ and $\Gamma^{c_i}{}_{a_ib_i} = F^{c_id_i}\Gamma_{d_i,a_ib_i}$.*

Given a manifold equipped with metric $\mathbf{F}$ and a BC parameterization $\boldsymbol{\lambda}$, consider the solution of the block-wise (geodesic) ODE[13] denoted by $\mathbf{R}^{[i]}(t)$ for block i:

$$\dot{R}^{\,c_i}(0) = -F^{c_ia_i}g_{a_i} \ ; \ \ R^{\,c_i}(0) = \lambda^{c_i} \tag{14}$$

$$\ddot{R}^{\,c_i}(t) = -\Gamma^{c_i}{}_{a_ib_i}(t)\dot{R}^{\,a_i}(t)\dot{R}^{\,b_i}(t) \tag{15}$$

[7]A metric is well-defined if it is positive definite everywhere.

[8]Such assumption is valid for minimal EF.

[9]The geodesic induces an exponential map used in exact RGD.

[10]Due to the Euler-Lagrange equation, a geodesic is a stationary curve. However, a geodesic may not be the shortest curve.

[11]The domain of $\mathbf{L}(t)$ is $\mathbb{R}$ for a complete manifold.

[12]A retraction map can be derived by approximating the geodesic. An exact RGD update is invariant under parameterization while inexact RGD updates including NGD often are not.

[13]$\mathbf{R}^{[i]}(t)$ is easier to solve compared to $\mathbf{L}(t)$. Note that (14) is the minimum requirement of a retraction map (Absil et al., 2009).

where $R^{c_i}(0)$, $\dot{R}^{c_i}(0)$, $\ddot{R}^{c_i}(t)$ respectively denote the c-th entry of $\mathbf{R}^{[i]}(0)$, $\dot{\mathbf{R}}^{[i]}(0)$, and $\ddot{\mathbf{R}}^{[i]}(t)$; $\Gamma^{c_i}{}_{a_i b_i}(t) := \Gamma^{c_i}{}_{a_i b_i}\big|^{\lambda^{[-i]}=R^{[-i]}(0)}_{\lambda^{[i]}=R^{[i]}(t)}$.

We use $\hat{q}(\mathbf{z}|\boldsymbol{\lambda}^{[i]})$ to denote $q(\mathbf{z}|\boldsymbol{\lambda})$ when $\boldsymbol{\lambda}^{[j]} = \mathbf{R}^{[j]}(0)$ is known for each block j except the i-th block. $F^{c_i a_i}$ is the entry of $(\mathbf{F}^{[i]})^{-1}$ at position (c,a), where $\mathbf{F}^{[i]}$ is the sub-block matrix of $\mathbf{F}$ for $\boldsymbol{\lambda}^{[i]}$. In fact, $\mathbf{F}^{[i]}$ is the induced metric for $\hat{q}(\mathbf{z}|\boldsymbol{\lambda}^{[i]})$, and $\mathbf{R}^{[i]}(t)$ is a geodesic for $\hat{q}(\mathbf{z}|\boldsymbol{\lambda}^{[i]})$ under BC parameterization $\boldsymbol{\lambda}$. Moreover, if $\boldsymbol{\lambda}$ is a BCN parameterization and $\mathbf{F}$ is the FIM, we have $F_{a_i b_i} = \partial_{\lambda^{a_i}}\partial_{\lambda^{b_i}} A(\boldsymbol{\lambda})$.

We define a curve $\mathbf{R}(t) := \{\mathbf{R}^{[1]}(t), \dots, \mathbf{R}^{[m]}(t)\}$. By Lemma 1, we can show that the first-order approximation of $\mathbf{R}(t)$ at $t_0 = 0$ induces NGD if $\mathbf{F}$ is the FIM and $\boldsymbol{\lambda}$ is a BC parameterization. Appendix B.2 shows this in detail.

We propose to use the second-order approximation[14] of $\mathbf{R}(t)$ at $t_0 = 0$ as below, where $\boldsymbol{\lambda}$ is a BC parameterization.

$$\begin{aligned}\text{Our}: \lambda^{c_i} &\leftarrow R^{c_i}(t_0) + \dot{R}^{c_i}(t_0)(t-t_0) + \tfrac{1}{2}\ddot{R}^{c_i}(t_0)(t-t_0)^2 \\ &= \lambda^{c_i} - t\hat{g}^{c_i} - \frac{t^2}{2}\Gamma^{c_i}{}_{a_i b_i}\hat{g}^{a_i}\hat{g}^{b_i} \qquad (16)\end{aligned}$$

where $\Gamma^{c_i}{}_{a_i b_i}$ is computed at $t_0 = 0$, $\hat{g}^{c_i} = F^{c_i a_i} g_{a_i}$, and c_i denotes the c-th element of the i-th block. Our rule works for both a BCN parameterization and a BC parameterization.

Song et al. (2018) suggest using the second-order approximation of $\mathbf{L}(t)$ at $t_0 = 0$, which has to compute the whole Christoffel symbol $\Gamma^{(c_i)}{}_{ab}$. However, their proposal does not guarantee the update stays in the constraint set even in univariate Gaussian cases (see Appendix A). Moreover, it is inefficient to compute the whole Christoffel symbol since all cross terms between any two blocks are needed in $\Gamma^{(c_i)}{}_{ab}\hat{g}^a\hat{g}^b$. When a parameterization has m blocks, $\Gamma^{c_i}{}_{a_i b_i}\hat{g}^{a_i}\hat{g}^{b_i} \neq \Gamma^{(c_i)}{}_{ab}\hat{g}^a\hat{g}^b$ since the hidden summations are taken over entries only in block i on the left while the summations are taken over entries in all m blocks on the right. This is the key difference between our method and their method. In our method, only the block-wise $\Gamma^{c_i}{}_{a_i b_i}$ is computed, which makes our method efficient in many cases such as multivariate Gaussians and MOGs. Moreover, a BCN parameterization can further simplify the computation of our rule due to Theorem 3 (see Appendix D for a proof).

Theorem 3 *Under a BCN parameterization of EF with the FIM, natural gradients and the Christoffel symbol for each block i can be simplified as*

$$\hat{g}^{a_i} = \partial_{m_{a_i}}\mathcal{L}(\boldsymbol{\lambda}) \;;\; \Gamma_{d_i, a_i b_i} = \tfrac{1}{2}\partial_{\lambda^{a_i}}\partial_{\lambda^{b_i}}\partial_{\lambda^{d_i}} A(\boldsymbol{\lambda})$$

where λ^{a_i} is the a-th entry of $\boldsymbol{\lambda}^{[i]}$; m_{a_i} is the a-th entry of the BC expectation parameter[15] $\mathbf{m}_{[i]} := \mathbb{E}_q\left[\boldsymbol{\phi}_i(\mathbf{z}, \boldsymbol{\lambda}^{[-i]})\right] = \partial_{\lambda^{[i]}} A(\boldsymbol{\lambda})$.

Since $A(\boldsymbol{\lambda})$ is C^3-smooth for block $\boldsymbol{\lambda}^{[i]}$ (Johansen, 1979), we have $\partial_{\lambda^{a_i}}\partial_{\lambda^{b_i}}\partial_{\lambda^{d_i}} A(\boldsymbol{\lambda}) = \partial_{\lambda^{d_i}}\partial_{\lambda^{a_i}}\partial_{\lambda^{b_i}} A(\boldsymbol{\lambda})$. Thus, by Theorem 3, we have $\Gamma^{c_i}{}_{a_i b_i} = \frac{1}{2}\partial_{m_{c_i}}\partial_{\lambda^{a_i}}\partial_{\lambda^{b_i}} A(\boldsymbol{\lambda})$.

A similar theorem for EF mixtures is in Appendix I.

To sum up, our rule is an instance of RGD with a retraction (see footnote 12,13) derived from a metric. The convergence of our rule could be obtained by existing analyses such as Bonnabel (2013) if the retraction satisfies certain properties.

6. Numerical Results

Our implementation: github.com/yorkerlin/iBayesLRule

6.1. Results on Synthetic Examples

To validate our rule, we visualize our approximations in 2-dimensional toy examples, where we use the re-parametrization trick suggested by Lin et al. (2019a;b) (see (20) in Appendix E for full Gaussian and (28) in Appendix J for mixture of Gaussians (MOG)) to compute gradients. Due to the trick, $\nabla_z^2 \bar{\ell}(\mathbf{z})$ is not needed. See Figure 5-6 in Appendix L for more visualization examples such as the banana distribution (Haario et al., 2001) and a BNN example taken from Au et al. (2020). We then compare our method to baseline methods in a higher dimensional example.

We first visualize Gaussian approximations with full covariance structures for the Bayesian Logistic regression example taken from Murphy (2013) ($N = 60, d = 2$). Figure 2(a) shows posterior approximations obtained from various methods. From the figure, our approximation matches the exact variational Gaussian approximation. For skew-Gaussian approximations (Lin et al., 2019a) and mean-field Gaussian approximations, see Figure 6 in Appendix L.

In the second example, we approximate the beta-binomial model (Salimans & Knowles, 2013) ($N = 20, d = 2$) by MOG. The exact posterior is skewed. From Figure 2(b), we see that the approximation matches the exact posterior better and better as we increase the number of mixtures.

In the third example, we approximate a correlated Laplace distribution $\exp(-\bar{\ell}(\mathbf{z})) = \text{Lap}(z_1|0,1)\text{Lap}(z_2|z_1,1)$ by using MOG, where $\text{Lap}(z_2|z_1,1) = \frac{1}{2}\exp(-|z_2 - z_1|)$. The distribution is non-smooth and $\nabla_z^2 \bar{\ell}(\mathbf{z})$ does not exist. However, we can use the re-parametrization trick since $\nabla_z \bar{\ell}(\mathbf{z})$ exists almost surely. From Figure 2(c), we see that our method gives smooth approximations of the target.

[14]Our approximation allows us to use a bigger step-size than NGD. In many cases, the underlying parameterization constraints are satisfied regardless of the choice of the step-size and therefore, a line search for the constraint satisfaction is no longer required.

[15]Instead of using $\mathbf{m}^{[i]}$, we use $\mathbf{m}_{[i]}$ to emphasize that Euclidean gradient for $\mathbf{m}_{[i]}$ is equivalent to natural gradient for $\boldsymbol{\lambda}^{[i]}$.

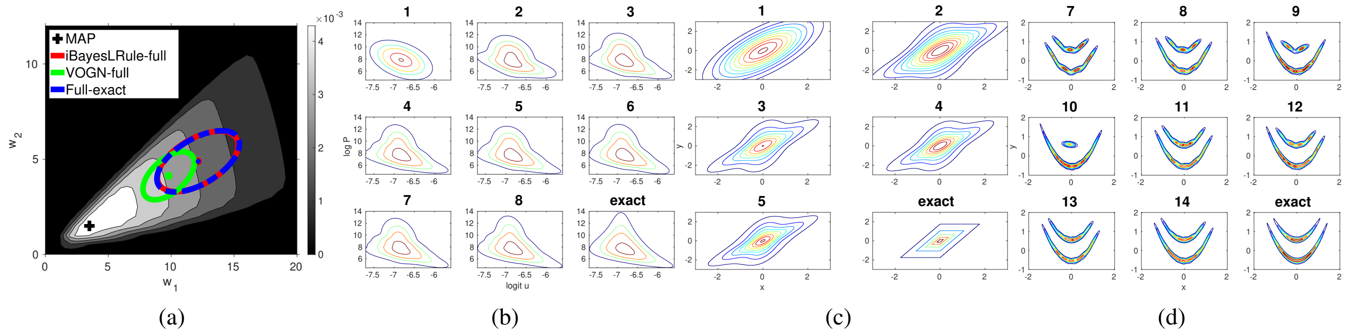


Figure 2. Visualization of posterior approximations on 2-D toy examples. Figure 2(a) shows the Gaussian approximation to fit a Bayesian logistic model, where our approximation matches the exact variational Gaussian approximation. Figure 2(b) shows MOG approximation fit to a beta-binomial model in a 2-D problem. The number indicates the number of mixture components. By increasing the number of components, we get better results. Figure 2(c) shows MOG approximation fit to a correlated 2-D Laplace distribution. The number indicates the number of mixtures. We get smooth approximations of the non-smooth distribution. Figure 2(d) shows MOG approximation fit to a double banana distribution. The number indicates the number of mixtures, where we only show the last 8 MOG approximations. The complete MOG approximations can be found in Appendix L. As the number of components increases, we get better results.

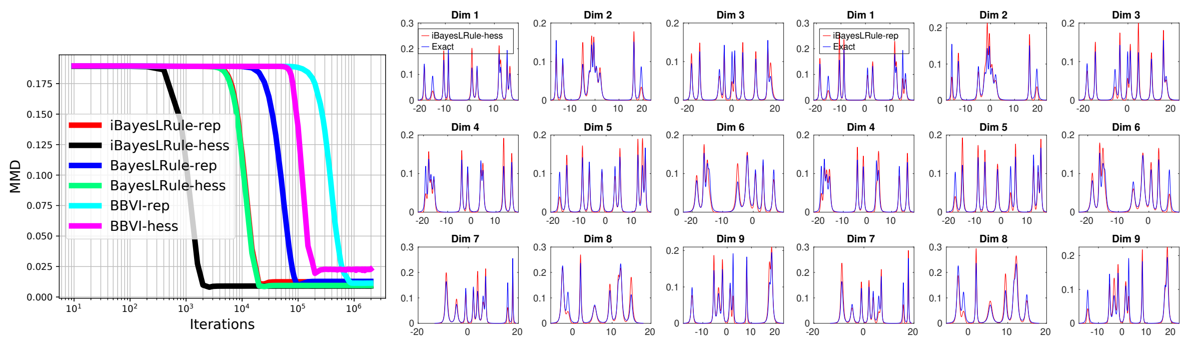
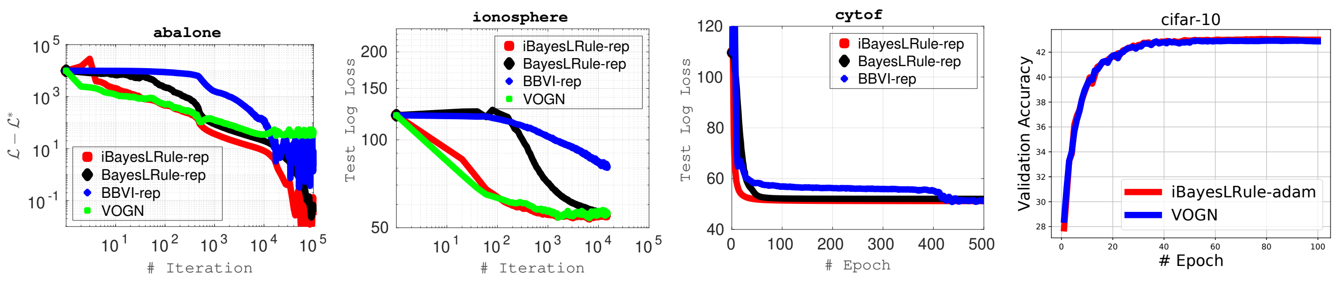


Figure 3. Comparison results on a 20-D mixture of Student's t distributions with 10 components by MOG approximations. The leftmost figure shows the performance of each method, where our method outperforms existing methods. The first 9 dimensions obtained by our method are shown in the figure where MOG approximation fits the marginals well. We also test a 300-D mixture problem in Appendix L.

In the fourth example, we approximate the double banana distribution constructed by Detommaso et al. (2018). The true distribution has two modes and is skewed. As shown in Figure 2(d), our MOG approximation approximates the target better and better when we increase the number of mixtures. For a complete plot using the approximation with various components, see Figure 5(b) in Appendix L.

Finally, we conduct a comparison study on approximations for a mixture of Student's t distributions $\exp(-\bar{\ell}(\mathbf{z})) = \frac{1}{C}\sum_{k=1}^{C}\mathcal{T}(\mathbf{z}|\mathbf{u}_k, \mathbf{V}_k, \alpha)$ with degrees of freedom $\alpha = 2$, where $\mathbf{z} \in \mathbb{R}^d$. We generate each entry of location vector $\mathbf{u}_k$ uniformly in an interval $(-s, s)$. Each shape matrix $\mathbf{V}_k$ is taken a form of $\mathbf{V}_k = \mathbf{A}_k^T\mathbf{A}_k + \mathbf{I}_d$, where each entry of the $d \times d$ matrix $\mathbf{A}_k$ is independently drawn from a Gaussian distribution with mean 0 and standard deviation $0.1d$. We approximate the posterior distribution by MOG with K components and use the importance sampling technique to compute gradients as suggested by Lin et al. (2019a) so that the number of Monte Carlo (MC) gradient evaluations is independent of the number of components K. We compare our method to existing methods, where the BayesLRule for MOG is proposed by Lin et al. (2019a).

We consider a case with $K = 25, C = 10, d = 20, s = 20$. For simplicity, we fix the mixing weight to be $\frac{1}{K}$ and only update each Gaussian component with the precision $\mathbf{S}_c$ and the mean $\boldsymbol{\mu}_c$ during training. We use 10 MC samples to compute gradients, where gradients are computed using either the re-parametrization trick (referred to as "-rep") as shown in (28) in Appendix J or the Hessian trick (referred to as "-hess") as shown in (29) in Appendix J . Note that BayesLRule with either the re-parametrization trick or the Hessian trick does not stay in the constraint set. We use the same initialization and tune the step size by grid search for each method . The leftmost plot of Figure 3 shows the performance. We clearly see that our methods converge fastest, when we use the maximum mean discrepancy (MMD) to measure the difference between an approximation and the ground-truth. The remaining plots of Figure 3 show the first 9 marginal distributions of the true distribution and our approximations with two kinds of gradient estimation, where



(a) (b) (c) (d)

Figure 4. Results on real-world datasets showing the performances of our method (iBayesLRule) highlighted in red compared to BBVI, BayesLRule with a line search, and VOGN. Figure 4(a) and 4(b) show the performances using Gaussian approximations with full covariance structure to fit a Bayesian linear regression and a Bayesian logistic regression, respectively, where our method converges faster than BayesLRule and BBVI and gives a more accurate approximation than VOGN. Figure 4(c) shows the performances using Gamma approximations to fit a Gamma factor model, where our method converges faster. Figure 4(d) shows the performances of methods in a Bayesian MLP network with diagonal Gaussian approximations, where our method performs comparably to VOGN.

MOG closely matches the marginals. All 20 marginal distributions are in Figure 7 in Appendix L. We also consider a more difficult case with $K = 60, C = 20, d = 300, s = 25$. Figure 8-12 in Appendix L show all 300 marginal distributions obtained by our method.

6.2. Results on Real Data

Now, we show results on real-world datasets. We consider four models in our experiments. The first model is the Bayesian linear regression, where we can obtain the exact solution and the optimal negative ELBO denoted by $\mathcal{L}^*$. We present results for full Gaussian approximations on the "Abalone" dataset ($N = 4,177, d = 8$) with 3341 chosen for training. We train the model with mini-batch size 168. In Figure 4(a), we plot the difference of ELBO between the exact and an approximation. We compare our method (referred to as "iBayesLRule") to the black-box gradient method (referred to as "BBVI") using the Adam optimizer (Kingma & Ba, 2015) and the original Bayesian learning rule (referred to as "BayesLRule") with the re-parametrization trick (referred to as "-rep") and the VOGN method. BBVI requires us to use an unconstrained parametrization. BayesLRule with the re-parametrization trick does not stay in the constraint set so a line search has to be used in BayesLRule. We can see that our method converges faster than BayesLRule and BBVI and is more accurate than VOGN.

Next, we consider the Bayesian logistic regression and present results for full Gaussian approximations on the "Ionosphere" dataset ($N = 351, d = 34$) with 175 chosen for training. We train the model with mini-batch size 17. In Figure 4(b), we plot the test log-loss and compare our method to BBVI and BayesLRule with the reparametrization trick (referred to as "-rep"). We also consider the VOGN method proposed for Gaussian approximations. Note that BayesLRule using the re-parametrization trick does not stay in the constraint set and a line search is used. From the plot, we can see our method outperforms BayesLRule and performs comparably to VOGN.

Then, we consider the Gamma factor model (Knowles, 2015; Khan & Lin, 2017) using Gamma approximations on the "CyTOF" dataset ($N = 522,656, d = 40$) with 300,000 chosen for training, where gradients are computed using the implicit re-parametrization trick (Figurnov et al., 2018) (referred to as "-rep"). We train the model with mini-batch size 39 and tune the step size for all methods. In Figure 4(c), we plot the test log-loss and compare our to BayesLRule and BBVI. BayesLRule uses a line search since the updates using the re-parametrization trick do not satisfy the constraint. Our method outperforms BayesLRule and BBVI.

Finally, we consider a Bayesian MLP network with 2 hidden layers, where we use 1000 units for each layer. We train the network with diagonal Gaussian approximations on the "CIFAR-10" dataset ($N = 60,000, d = 3 \times 32 \times 32$) with 50,000 images for training and 10,000 images for validation. We train the model with mini-batch size 128 and compare our Adam-like update (referred to as "iBayesLRule-adam") to VOGN. We use the same initialization and hyperparameters in both methods. In Figure 4(d), we plot the validation accuracy. Our method performs similarly to VOGN.

7. Discussion

We present an improved learning rule to handle positive-definite parameterization constraints. We propose a BCN parameterization so that natural gradients and the extra terms are easy to compute. Under this parameterization, the Fisher matrix and the Christoffel symbols admit a closed-form via differentiation without introducing extra integrations.

Our main focus is on the derivation of simple and efficient updates that naturally handle positive-definite constraints. We give examples where our updates have low iteration cost. We hope to perform large-scale experiments in the future.

Acknowledgements

We would like to thank Hiroyuki Kasai (Waseda University) for useful discussions at ICML 2019 at the early stage of this project. WL is supported by a UBC International Doctoral Fellowship.

References

Absil, P.-A., Mahony, R., and Sepulchre, R. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

Au, K. X., Graham, M. M., and Thiery, A. H. Manifold lifting: scaling MCMC to the vanishing noise regime. *arXiv preprint arXiv:2003.03950*, 2020.

Batir, N. Some new inequalities for gamma and polygamma functions. *J. Inequal. Pure Appl. Math*, 6(4):1–9, 2005.

Bonnabel, S. Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control*, 58 (9):2217–2229, 2013.

Detommaso, G., Cui, T., Marzouk, Y., Spantini, A., and Scheichl, R. A Stein variational Newton method. In *Advances in Neural Information Processing Systems*, pp. 9169–9179, 2018.

Figurnov, M., Mohamed, S., and Mnih, A. Implicit reparameterization gradients. In *Advances in Neural Information Processing Systems*, pp. 441–452, 2018.

Fisher, R. A. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222(594-604):309–368, 1922.

Fletcher, P. T. and Joshi, S. Principal geodesic analysis on symmetric spaces: Statistics of diffusion tensors. In *Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis*, pp. 87–98. Springer, 2004.

Haario, H., Saksman, E., Tamminen, J., et al. An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242, 2001.

Hosseini, R. and Sra, S. Matrix manifold optimization for Gaussian mixtures. In *Advances in Neural Information Processing Systems*, pp. 910–918, 2015.

Johansen, S. Introduction to the theory of regular exponential famelies. 1979.

Khan, M. and Lin, W. Conjugate-computation variational inference: Converting variational inference in non-conjugate models to inferences in conjugate models. In *Artificial Intelligence and Statistics*, pp. 878–887, 2017.

Khan, M. E. and Rue, H. Learning-algorithms from Bayesian principles. 2020. https://emtiyaz.github.io/papers/learning_from_bayes.pdf.

Khan, M. E., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. Fast and scalable Bayesian deep learning by weight-perturbation in Adam. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 2611–2620, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Knowles, D. A. Stochastic gradient variational Bayes for gamma approximating distributions. *arXiv preprint arXiv:1509.01631*, 2015.

Koumandos, S. Monotonicity of some functions involving the gamma and psi functions. *Mathematics of Computation*, 77(264):2261–2275, 2008.

Lin, W., Khan, M. E., and Schmidt, M. Fast and simple natural-gradient variational inference with mixture of exponential-family approximations. In *International Conference on Machine Learning*, pp. 3992–4002, 2019a.

Lin, W., Khan, M. E., and Schmidt, M. Stein's Lemma for the Reparameterization Trick with Exponential-family Mixtures. *arXiv preprint arXiv:1910.13398*, 2019b.

Malagò, L. and Pistone, G. Information geometry of the Gaussian distribution in view of stochastic optimization. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, pp. 150–162, 2015.

Minh, H. Q. and Murino, V. Covariances in computer vision and machine learning. *Synthesis Lectures on Computer Vision*, 7(4):1–170, 2017.

Murphy, K. P. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass., 2013. ISBN 9780262018029 0262018020.

Nielsen, F. On geodesic triangles with right angles in a dually flat space. *arXiv preprint arXiv:1910.03935*, 2019.

Opper, M. and Archambeau, C. The variational Gaussian approximation revisited. *Neural computation*, 21(3):786–792, 2009.

Osawa, K., Swaroop, S., Khan, M. E. E., Jain, A., Eschenhagen, R., Turner, R. E., and Yokota, R. Practical deep learning with Bayesian principles. In *Advances in neural information processing systems*, pp. 4287–4299, 2019.

Pennec, X., Fillard, P., and Ayache, N. A Riemannian framework for tensor computing. *International Journal of computer vision*, 66(1):41–66, 2006.

Rao, C. R. Information and accuracy attainable in the estimation of statistical parameters. *Bulletin of the Calcutta Mathematical Society*, 37(3):81–91, 1945.

Salimans, T. and Knowles, D. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013.

Salimbeni, H., Eleftheriadis, S., and Hensman, J. Natural gradients in practice: Non-conjugate variational inference in gaussian process models. *arXiv preprint arXiv:1803.09151*, 2018.

Särkkä, S. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.

Song, Y., Song, J., and Ermon, S. Accelerating natural gradient with higher-order invariance. In *International Conference on Machine Learning*, pp. 4720–4729, 2018.

Tran, M.-N., Nguyen, D. H., and Nguyen, D. Variational Bayes on manifolds. *arXiv preprint arXiv:1908.03097v2*, 2019.

Tseng, P.-H. and Lee, T.-C. Numerical evaluation of exponential integral: Theis well function approximation. *Journal of Hydrology*, 205(1-2):38–51, 1998.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, pp. 5847–5856, 2018.