Riemannian Laplace Approximation with the Fisher Metric

Hanlin Yu
Department of Computer Science
University of Helsinki

Marcelo Hartmann
Department of Computer Science
University of Helsinki

Bernardo Williams
Department of Computer Science
University of Helsinki

Mark Girolami
Department of Engineering
University of Cambridge
and The Alan Turing Institute

Arto Klami
Department of Computer Science
University of Helsinki

Abstract

Laplace's method approximates a target density with a Gaussian distribution at its mode. It is computationally efficient and asymptotically exact for Bayesian inference due to the Bernstein-von Mises theorem, but for complex targets and finite-data posteriors it is often too crude an approximation. A recent generalization of the Laplace Approximation transforms the Gaussian approximation according to a chosen Riemannian geometry providing a richer approximation family, while still retaining computational efficiency. However, as shown here, its properties depend heavily on the chosen metric, indeed the metric adopted in previous work results in approximations that are overly narrow as well as being biased even at the limit of infinite data. We correct this shortcoming by developing the approximation family further, deriving two alternative variants that are exact at the limit of infinite data, extending the theoretical analysis of the method, and demonstrating practical improvements in a range of experiments.

1 INTRODUCTION

Functional distributional approximations offer a computationally attractive alternative to Markov Chain
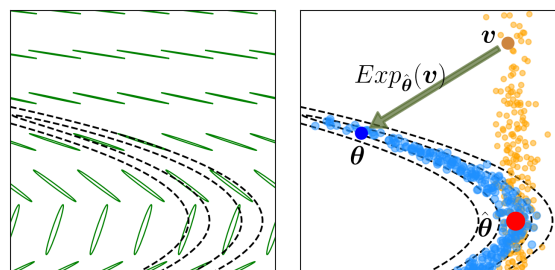

Figure 1: Left: The Fisher metric (green) captures the local curvature of the target density (black). Right: Samples from a Gaussian distribution (orange) are deterministically mapped using the Fisher metric to provide a flexible approximation (blue).

Monte Carlo (MCMC) methods for statistical (posterior in the Bayesian setting) inference of potentially large models, building on variational inference (VI) (Blei et al., 2017), expectation propagation (EP) (Minka, 2001), or Laplace approximation (LA) (Tierney and Kadane, 1986; Rue et al., 2009). Typically the methods need to make a compromise between computational efficiency and accuracy, and often the flexibility is achieved by non-linear transformations modelled as neural networks that need to be separately trained for a given model, as in amortized VI (Margossian and Blei, 2023) or normalizing flows (Papamakarios et al., 2021). We focus on perhaps the oldest approximation strategy, the Laplace Approximation (Tierney and Kadane, 1986), and study how to increase the flexibility of the approximation without resorting to trainable transformations.

Laplace Approximation fits a Gaussian distribution at the mode of the target density and is used in machine learning from non-conjugate Gaussian processes (Rasmussen and Williams, 2006) to deep learn-

ing (Daxberger et al., 2021). It is computationally efficient, only requiring the maximum a posteriori (MAP) estimate and the Hessian of the log-posterior that determine the mean and covariance of the approximation. Despite its simplicity, it is asymptotically exact at the limit of infinite data, due to the Bernstein-von Mises theorem stating that the posterior converges to a Gaussian when the probabilistic model has identifiable parameterisation (van der Vaart, 1998). Obviously, as has been shown many times in the literature, the disadvantage is that the approximation family is very limited for finite-data posteriors.

We seek to create an approximation that retains the advantages but increases the flexibility, and do so by relying on the tools of Riemannian geometry (Carmo, 1992). We change the metric of the parameter space in such a manner that after this modification the approximation more accurately characterises the target. Concepts of Riemannian geometry have been broadly considered in the approximate inference literature, with Riemannian extensions of MCMC (Girolami and Calderhead, 2011), stochastic-gradient MCMC (Patterson and Teh, 2013; Yu et al., 2023) and VI (Frank et al., 2021). Recently, Bergamin et al. (2023) introduced a Riemannian generalization of the Laplace Approximation. The core idea (illustrated conceptually in Figure 1, using the new metric from this paper) is to transform samples from a Gaussian distribution using numerical integrators to follow geodesic paths induced by a chosen geometry, which can be carried out in parallel. Some of the concepts required were already studied by Hauberg (2018), where Laplace approximation on a Riemannian manifold was considered.

We build on Bergamin et al. (2023), expanding both theoretical and practical understanding of the approximation. They used the Riemannian metric Hartmann et al. (2022) proposed for Riemann Manifold MCMC, which is computationally attractive due to the efficient *exponential map* for transforming the samples. However, as shown here, the approximation is not asymptotically exact and is also biased in practical cases. We resolve this theoretical and practical limitation in two alternative ways. We first show that the method of Bergamin et al. (2023) can be fixed by incorporating a *logarithmic map* to correct the bias, resulting in an asymptotically exact but somewhat computationally heavy and unstable approximation. A more practical solution is to replace the metric itself. We explain how a metric based on the Fisher Information Matrix (FIM) gives an approximation that is exact for targets that are diffeomorphisms of a Gaussian, requires fewer function evaluations in numerical integration, and is superior in how it performs in a range of tasks.

2 PRELIMINARIES

2.1 Laplace Approximation

Let $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ denote the D-dimensional vector in the parameter space $\boldsymbol{\Theta}$. Denote the log-posterior distribution given observed data set $\boldsymbol{y} = \{y_n\}_{n=1}^{N}$ as $\ell_{\boldsymbol{y}}(\boldsymbol{\theta}) := \log \pi(\boldsymbol{\theta} \,|\, \boldsymbol{y})$. Laplace's method expands a second-order Taylor series of $\ell_{\boldsymbol{y}}(\boldsymbol{\theta})$ at the mode (MAP) where the first-order term disappears. This is equivalent to using a Gaussian approximation

$$\pi(\boldsymbol{\theta} \,|\, \boldsymbol{y}) \approx \mathcal{N}\left(\boldsymbol{\theta} \,|\, \hat{\boldsymbol{\theta}}, -\nabla^2 \ell_{\boldsymbol{y}}(\hat{\boldsymbol{\theta}})^{-1}\right),$$

where $\hat{\boldsymbol{\theta}}$ is the MAP estimate and $\nabla^2 \ell_{\boldsymbol{y}}(\hat{\boldsymbol{\theta}})$ is the Hessian matrix of the log-posterior evaluated at $\hat{\boldsymbol{\theta}}$. We refer to this as *ELA*, for Euclidean LA.

2.2 Riemannian geometry

Key concepts. A *manifold* is a space more general than an Euclidean space but that locally acts like a Euclidean space. A *differentiable manifold* is a manifold that can be represented with a system of coordinates such that it will allow us to extend notions of derivatives. A *Riemannian manifold* is a differentiable manifold endowed with a metric g and this metric will, intrinsically, generalise the local notions of angles, distances and derivatives on the manifold.

The metric g is formally defined as a function $g_{\boldsymbol{\theta}}: T_{\boldsymbol{\theta}}\boldsymbol{\Theta} \times T_{\boldsymbol{\theta}}\boldsymbol{\Theta} \rightarrow \mathbb{R}$ where $T_{\boldsymbol{\theta}}\boldsymbol{\Theta}$ denotes the *tangent space* at $\boldsymbol{\theta}$, given by $T_{\boldsymbol{\theta}}\boldsymbol{\Theta} = \{\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{c}(t)|_{t=0} = \boldsymbol{v}$, and $\boldsymbol{c}(0) = \boldsymbol{\theta}\}$. The $\boldsymbol{v}$s are known as the *tangent vectors*. The function g is like the dot-product in Euclidean spaces, but takes a more general form as a mapping $(\boldsymbol{v}, \boldsymbol{u}) \mapsto \langle \boldsymbol{v}, \boldsymbol{G}(\boldsymbol{\theta})\boldsymbol{u} \rangle$ where $\boldsymbol{G}(\boldsymbol{\theta})$ is a symmetric positive-definite matrix called *metric tensor* that collects the coefficients of the metric as $g\left(\frac{\partial}{\partial \theta_i}, \frac{\partial}{\partial \theta_j}\right) = G(\boldsymbol{\theta})_{ij}$. We are free to choose the metric g to encode desired properties of the manifold.

Gradients and Hessians in Riemannian geometry. In a given basis, the *Riemannian gradient* of $\ell_{\boldsymbol{y}}(\boldsymbol{\theta})$ is given by $\operatorname{grad} \ell_{\boldsymbol{y}}(\boldsymbol{\theta}) = \boldsymbol{G}(\boldsymbol{\theta})^{-1}\nabla \ell_{\boldsymbol{y}}(\boldsymbol{\theta})$, and the *Riemannian Hessian* is $\operatorname{Hess} \ell_{\boldsymbol{y}}(\boldsymbol{\theta})[\boldsymbol{v}] = \nabla_{\boldsymbol{v}}\operatorname{grad} \ell_{\boldsymbol{y}}(\boldsymbol{\theta}) = \boldsymbol{G}^{-1}\left(\nabla^2 \ell_{\boldsymbol{y}}(\boldsymbol{\theta}) - \sum_k \boldsymbol{\Gamma}(\boldsymbol{\theta})^k \nabla \ell_{\boldsymbol{y}}(\boldsymbol{\theta})_k\right)\boldsymbol{v}$. Here $\boldsymbol{\Gamma}(\boldsymbol{\theta})^k = \{\boldsymbol{\Gamma}(\boldsymbol{\theta})^k\}_{i,j}$ are the *Christoffel symbols* of the second kind in matrix forms. They can also be expressed using the metric if the connection is chosen to be the Levi-Civita connection (Carmo, 1992) as

$$\Gamma_{ij}^k = \frac{1}{2}G^{kl}\left(\partial_i G_{jl} + \partial_j G_{il} - \partial_l G_{ij}\right),$$

where $\partial_i = \frac{\partial}{\partial \theta_i}$. Following differential geometric convention, we use the Einstein summation. Note that in

this convention G^{kl} refers to elements of the *inverse* of the matrix $\boldsymbol{G}(\boldsymbol{\theta})$.

Geodesic paths. Another important notion for the development of this work is the *exponential map* and its inverse known as the *logarithmic map*. Formally, the exponential-map is defined as a function $\mathrm{Exp}_{\boldsymbol{\theta}} : T_{\boldsymbol{\theta}}\boldsymbol{\Theta} \to \boldsymbol{\Theta}$ that takes a vector on the tangent space of $\boldsymbol{\Theta}$ and maps it back onto $\boldsymbol{\Theta}$ (Boumal, 2023).

Intuitively, the exponential map is a function that returns the final position after following the shortest path at $\boldsymbol{\theta}$ in the direction of $\boldsymbol{v}$ for unit time. These paths are called geodesics and they solve the geodesic equation given by (Carmo, 1992)

$$\frac{\mathrm{d}^2\theta^k}{\mathrm{d}t^2} + \frac{\mathrm{d}\theta^i}{\mathrm{d}t}\frac{\mathrm{d}\theta^j}{\mathrm{d}t}\Gamma_{ij}^k = 0, \ k = 1, 2, \ldots, D, \quad (1)$$

where $\frac{\mathrm{d}\theta^i}{\mathrm{d}t} = v^i$. The inverse $\mathrm{Log}_{\boldsymbol{\theta}} : \boldsymbol{\Theta} \to T_{\boldsymbol{\theta}}\boldsymbol{\Theta}$ of the above is defined as $\mathrm{Log}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{\theta}}) = \arg\min_{\boldsymbol{v}\in T_{\boldsymbol{\theta}}\boldsymbol{\Theta}} g_{\boldsymbol{\theta}}(\boldsymbol{v}, \boldsymbol{v})$ subject to $\mathrm{Exp}_{\boldsymbol{\theta}}(\boldsymbol{v}) = \tilde{\boldsymbol{\theta}}$ (see Boumal, 2023). It gives the $\boldsymbol{v}$ for which the exponential map starting from $\boldsymbol{\theta}$ maps to $\tilde{\boldsymbol{\theta}}$.

3 RIEMANNIAN LA

Here we outline a general recipe for Riemannian Laplace Approximation (RLA). The formulation matches the one recently proposed by Bergamin et al. (2023), though here presented in a more general form where e.g. the choice of the metric is still left open. We will later show how the metric influences both the theoretical and practical properties of the method.

The Bernstein-von Mises Theorem states that, under certain regularity conditions, the posterior at the limit of infinite data becomes Gaussian with inverse of the FIM as the covariance (van der Vaart, 1998). This makes ELA exact at the limit. We want a Riemannian generalization of ELA to retain this property but to adapt for the local curvature of the target for non-Gaussian targets and finite-data posteriors.

3.1 Principle and algorithm

The Riemannian extension of LA is based on a Taylor series expansion on the tangent space $T_{\boldsymbol{\theta}}\boldsymbol{\Theta}$ at a given point $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. Bergamin et al. (2023) used here the particular tangent space $T_{\hat{\boldsymbol{\theta}}}\boldsymbol{\Theta}$, where $\hat{\boldsymbol{\theta}}$ is the MAP estimate. When the parameter space of the model $\boldsymbol{\Theta}$ is endowed with a Riemannian geometry, the Taylor approximation of the function $\boldsymbol{\Theta} \ni \boldsymbol{\theta} \mapsto \ell_{\boldsymbol{y}}(\boldsymbol{\theta})$ at $\hat{\boldsymbol{\theta}}$ assumes a more general formulation (see Boumal, 2023;

Algorithm 1 RLA-B, producing N samples $\boldsymbol{\theta}^{[n]}$.

Obtain MAP estimate $\hat{\boldsymbol{\theta}}$
Set $\boldsymbol{\Sigma} = (-\nabla^2\ell_{\boldsymbol{y}}(\hat{\boldsymbol{\theta}}))^{-1}$
for $n \leftarrow 1, \ldots, N$ **do**
    Obtain velocity $\boldsymbol{v}^{[n]} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$
    $\boldsymbol{\theta}^{[n]} = \mathrm{Exp}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{v}^{[n]})$
end for

Bergamin et al., 2023)

$$\ell_{\boldsymbol{y}}\big(\mathrm{Exp}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{v})\big) \approx \ell_{\boldsymbol{y}}(\hat{\boldsymbol{\theta}}) + g_{\hat{\boldsymbol{\theta}}}\big(\mathrm{grad}\,\ell_{\boldsymbol{y}}(\hat{\boldsymbol{\theta}}), \boldsymbol{v}\big) \quad (2)$$
$$+ \frac{1}{2}g_{\hat{\boldsymbol{\theta}}}\big(\mathrm{Hess}\,\ell_{\boldsymbol{y}}(\hat{\boldsymbol{\theta}})[\boldsymbol{v}], \boldsymbol{v}\big).$$

Generally we cannot evaluate the approximation analytically, but can obtain samples from it following Algorithm 1 that comprises of three distinct steps, where the last two are repeated for each sample (and can be computed in parallel):

1. Find $\hat{\boldsymbol{\theta}}$ where the approximation is placed;

2. Sample initial velocity $\boldsymbol{v} \in T_{\hat{\boldsymbol{\theta}}}\boldsymbol{\Theta}$ from a Gaussian distribution with suitably chosen covariance $\boldsymbol{\Sigma}$;

3. Solve exponential map at $\hat{\boldsymbol{\theta}}$ with velocity $\boldsymbol{v}$ to obtain the sample $\boldsymbol{\theta}$.

This constructs a wrapped Gaussian (De Bortoli et al., 2022) on the manifold. The primary design choice is the metric g that encodes the intrinsic geometry of the problem, but as will be explained later there is freedom also in the choices for the first two steps. Once g is given, the geodesic equation (1) can be formulated as an initial value problem of an ordinary system of differential equations (ODE) and solved relatively efficiently with numerical integration.

Euclidean Laplace Approximation. Consider the special case of g being the Euclidean inner product. Then $\boldsymbol{G}(\boldsymbol{\theta}) = \boldsymbol{I}_D$ and the exponential map becomes $\mathrm{Exp}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{v}) = \hat{\boldsymbol{\theta}} + \boldsymbol{v}$, a unit step in the direction of the velocity $\boldsymbol{v}$. If velocities are sampled using $\boldsymbol{\Sigma} = -(\nabla^2\ell_{\boldsymbol{y}}(\hat{\boldsymbol{\theta}}))^{-1}$ we get the classical ELA.

Method of Bergamin et al. (2023). In Equation (2), Bergamin et al. (2023) sets the metric tensor as $\boldsymbol{G}(\boldsymbol{\theta}) = \boldsymbol{I}_D + \nabla\ell_{\boldsymbol{y}}(\boldsymbol{\theta})\nabla\ell_{\boldsymbol{y}}(\boldsymbol{\theta})^{\top}$, which is a special case of the metric Hartmann et al. (2022) used for geometric MCMC and coined as the *Monge metric*. This metric is defined solely based on gradient information, and is a way to introduce curvature information using the first order derivative of the target density function. It enables fast computation of Christoffel symbols for the exponential map. We denote this specific
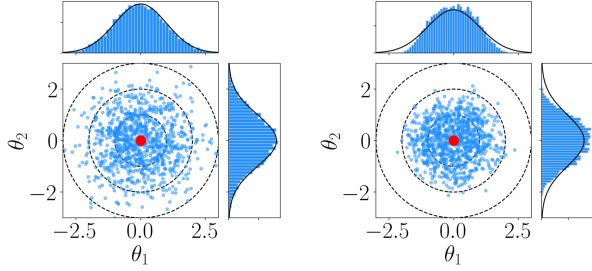
Figure 2: Euclidean LA (left) is exact for a Gaussian target, but RLA-B (right) is biased. Lines are the true contours and marginals, with samples and histograms characterizing the approximation.
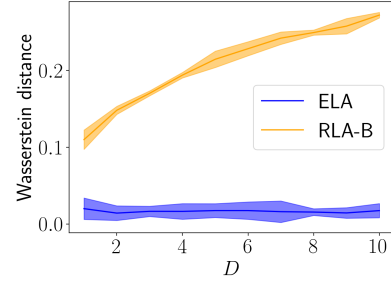

Figure 3: Wasserstein distances from approximate samples to true samples for isotropic Gaussians of varying D, computed for the first dimension. The lines are means and means ± 2.0 times standard deviations (stds) computed over 5 runs.

instance of RLA as *RLA-B* after the initial letter of the first author, to distinguish the specific variant from the general family. Even though RLA-B was shown to work well in a range of tasks and scales up for large problems, we will later observe that it does not always improve over ELA.

More importantly, the method is not exact even for Gaussian targets and hence not for infinite-data posteriors, but instead underestimates the uncertainty. This is seen by analysis of the exponential map. The solutions of the geodesic equation keep the norm $\|\boldsymbol{v}(t)\|^2_{\boldsymbol{G}(\boldsymbol{\theta}(t))}$ constant (Lee, John M., 2018), which in this metric expands to $\|\boldsymbol{v}(t)\|^2 + \langle \boldsymbol{v}(t), \nabla \ell_{\boldsymbol{y}}(\boldsymbol{\theta}(t)) \rangle^2$. At the MAP estimate we have $\nabla \ell_{\boldsymbol{y}}(\hat{\boldsymbol{\theta}}) = 0$ and hence $\boldsymbol{G}(\hat{\boldsymbol{\theta}}) = \boldsymbol{I}_D$. Denote the norm of the initial velocity by $\|\boldsymbol{v}(0)\|$. For every other t we have $\|\boldsymbol{v}(t)\| < \|\boldsymbol{v}(0)\|$ because $\langle \boldsymbol{v}(t), \nabla \ell_{\boldsymbol{y}}(\boldsymbol{\theta}(t)) \rangle^2 > 0$ except in the rare cases when the gradient is null or exactly orthogonal to velocity. Since the exponential map integrates for unit time, the distance of each sample to $\hat{\boldsymbol{\theta}}$ (in the Euclidean sense) is upper-bounded by and almost always smaller than for ELA, for which the distance is $\|\boldsymbol{v}(0)\|$. Figure 2 shows that RLA-B does not recover even a two dimensional isotropic Gaussian. Moreover, as shown in Figure 3, the bias of RLA-B becomes larger as D grows from 1 to 10 for isotropic Gaussians; see Section 11 in the Supplement for further remarks.

4 IMPROVING RIEMANNIAN LA

Next we describe two alternative asymptotically valid RLA methods, and discuss the choice of the starting point and covariance of velocities.

4.1 RLA-B with logarithmic map

We can make RLA-B exact for Gaussian targets by employing a logarithmic map to solve for the initial velocity $\boldsymbol{v}$ such that the result of the exponential map

retains the correct Euclidean distance to $\hat{\boldsymbol{\theta}}$.

Consider two manifolds $\mathcal{P}$ and $\mathcal{M}$. $\mathcal{P} = (\boldsymbol{\Theta}, g_{\mathcal{P}})$ has the metric tensor

$$\boldsymbol{G}_{\mathcal{P}}(\boldsymbol{\theta}) = I + \nabla \log \mathcal{N}(\boldsymbol{\theta} \,|\, \hat{\boldsymbol{\theta}}, \boldsymbol{\Sigma}) \nabla \log \mathcal{N}(\boldsymbol{\theta} \,|\, \hat{\boldsymbol{\theta}}, \boldsymbol{\Sigma})^{\top}$$

where $\boldsymbol{\Sigma} = (-\nabla^2 \ell_{\boldsymbol{y}}(\hat{\boldsymbol{\theta}}))^{-1}$, and $\mathcal{M} = (\boldsymbol{\Theta}, g_{\mathcal{M}})$ the metric tensor

$$\boldsymbol{G}_{\mathcal{M}}(\boldsymbol{\theta}) = I + \nabla \ell_{\boldsymbol{y}}(\boldsymbol{\theta}) \nabla \ell_{\boldsymbol{y}}(\boldsymbol{\theta})^{\top}.$$

$T_{\hat{\boldsymbol{\theta}}} \mathcal{P}$ and $T_{\hat{\boldsymbol{\theta}}} \mathcal{M}$ coincide at $\hat{\boldsymbol{\theta}}$ as the metric tensors are identical, and hence we have a diffeomorphism

$$\mathrm{Exp}_{\mathcal{M}, \hat{\boldsymbol{\theta}}} \circ \mathrm{Log}_{\mathcal{P}, \hat{\boldsymbol{\theta}}} : \mathcal{P} \to \mathcal{M}\,.$$

Suppose we sample from a Gaussian distribution as in ELA and consider the samples as points on manifold $\mathcal{P}$. If $\mathcal{P}$ is equal to $\mathcal{M}$ the transformation given by $\mathrm{Exp}_{\mathcal{M}, \hat{\boldsymbol{\theta}}} \circ \mathrm{Log}_{\mathcal{P}, \hat{\boldsymbol{\theta}}}$ results in exact samples from the target associated with $\mathcal{M}$. For non-Gaussian targets the method is not exact, but we observed it can still help reducing the underestimation tendency of RLA-B.

In Algorithm 2 the exponential map is computed as before on manifold $\mathcal{M}$, and the logarithmic map corresponds to a boundary value problem of the geodesic ODE on manifold $\mathcal{P}$. It is solved numerically and benefits again from efficient computation of Christoffel symbols. The logarithmic map is in general more costly than the exponential map (Arvanitidis et al., 2016), but we observed it can help reducing the number of function evaluations during the exponential map and the overall cost is not necessarily higher. We refer to this variant as RLA-BLog.

4.2 Fisher Information Matrix as metric

The choices of Bergamin et al. (2023) were motivated by computational arguments and relied on extrinsic

Algorithm 2 RLA-BLog, producing N samples $\boldsymbol{\theta}^{[n]}$. The manifold $\mathcal{P}$ uses the metric induced by Gaussian $\mathcal{N}(\tilde{\boldsymbol{\theta}}|\hat{\boldsymbol{\theta}}, \boldsymbol{\Sigma})$ and the manifold $\mathcal{M}$ the metric induced by the target $\pi(\boldsymbol{\theta}\,|\,\boldsymbol{y})$.

Obtain MAP estimate $\hat{\boldsymbol{\theta}}$
$\boldsymbol{\Sigma} = (-\nabla^2 \ell_{\boldsymbol{y}}(\hat{\boldsymbol{\theta}}))^{-1}$
for $n \leftarrow 1 \dots N$ **do**
    Obtain base sample $\bar{\boldsymbol{\theta}}^{[n]} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$
    $\boldsymbol{v}^{[n]} = \mathrm{Log}_{\mathcal{P},\hat{\boldsymbol{\theta}}}(\bar{\boldsymbol{\theta}}^{[n]})$
    $\boldsymbol{\theta}^{[n]} = \mathrm{Exp}_{\mathcal{M},\hat{\boldsymbol{\theta}}}(\boldsymbol{v}^{[n]})$
end for

---

notions in differential geometry, whereas we turn the attention to intrinsic notions and the natural Riemannian metric tensor for probabilistic machine learning.

4.3 Fisher metric

From a statistical point of view FIM is the lower bound for the variance of unbiased estimators. At a first glance, it may not tell us much about possible underlying geometries on $\boldsymbol{\Theta}$. However, FIM transforms like a 2-covariant tensor for regular probabilistic models as the expected score function is null and it is symmetric and positive-definite by definition (see Lehmann, 2003; Schervish, 2012). Therefore the FIM can act as the coefficients $\boldsymbol{G}(\boldsymbol{\theta})$ for the metric g and the pair $(\boldsymbol{\Theta}, g)$ defines an abstract Riemannian manifold.

We propose using a metric tensor

$$\boldsymbol{G}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{Y}\,|\,\boldsymbol{\theta}} \left[-\nabla^2 \log \pi(\boldsymbol{Y}\,|\,\boldsymbol{\theta})\right] - \nabla^2 \log \pi(\boldsymbol{\theta}), \quad (3)$$

which is the FIM plus the negative Hessian of the log-prior, accounting for both the likelihood and the prior. This formulation has been used by Girolami and Calderhead (2011) and Lan et al. (2015) as the metric for Riemann Manifold MCMC algorithms and by Hartmann and Vanhatalo (2019) in natural gradient descent for Gaussian processes with non log-concave likelihoods. For simplicity, we call it the *Fisher metric* and the resulting approximation RLA-F.

The same form can also be used as $\boldsymbol{\Sigma}$ when sampling the velocities $\boldsymbol{v}$, instead of the negative Hessian as in ELA and RLA-B. The negative Hessian is not guaranteed to be positive definite (e.g. for a neural network), which causes numerical problems, whereas (3) is when the negative Hessian of the log prior is positive definite, e.g. when the prior is Gaussian. In addition, we will later show that the Fisher metric can coincide with the negative Hessian for a specific class of targets (Theorem 3).

4.3.1 Hausdorff MAP

The classical MAP estimate is not invariant under reparameterizations in Euclidean geometry (see Jermyn, 2005). From the Riemannian perspective, a natural invariant alternative is the maximum value of the posterior density under the Hausdorff measure. When $\boldsymbol{\Theta}$ is endowed with g, the probability density function on the manifold is given by

$$\pi^{\boldsymbol{G}}(\boldsymbol{\theta}) = \frac{\pi(\boldsymbol{\theta})}{\sqrt{\det \boldsymbol{G}(\boldsymbol{\theta})}};$$

see Theorem 3.2.5 by Federer (1969).

If sampling the velocities $\boldsymbol{v}$ using the Fisher metric, we can justify the usage of the Hausdorff MAP as a reparameterization (from the differential geometric viewpoint), where the Fisher is locally identity around the MAP. This makes the gradients zero and consequently the second-order Taylor series still corresponds to a Gaussian; see Section 9 in the Supplement for further discussions. We will later show that both Hausdorff and Euclidean MAP can provide good approximations.

4.3.2 Theoretical basis

This variant is asymptotically exact for Gaussian targets, including posterior distributions of identifiable models at the limit of large data, as determined by the following theorems (see Section 8 in the Supplement for proofs). Note that Knollmüller and Enßlin (2020) also showed conclusions similar to Theorem 1 in the context of VI.

Theorem 1. *For Gaussian (or uniform) prior and Gaussian likelihood with fixed covariance, the Fisher metric is constant.*

Theorem 2. *Under the conditions of the Bernstein-von Mises theorem, the Fisher metric converges to a fixed matrix at the limit of infinite data.*

For constant metric (or for one that converges to a constant one), the geodesics become straight lines in the Euclidean sense. Consequently the approximation becomes identical to ELA and hence exact for the target that is (or converges to) a Gaussian.

We can also make a stronger statement (proof in Supplement) for specific types of targets and priors:

Theorem 3. *With an invariant prior, e.g. Jeffreys prior, RLA-F with Hausdorff MAP is exact for probabilistic models whose target distributions are diffeomorphic with Gaussians, for which the negative Hessian at the Hausdorff MAP coincides with the Fisher metric.*

Denote a diffeomorphic transformation $\boldsymbol{\theta} = \boldsymbol{\phi}(\boldsymbol{\psi})$ for a Gaussian distribution $\mathcal{N}(\boldsymbol{\psi}\,|\,\boldsymbol{\mu}, \boldsymbol{S})$. Since Gaussians
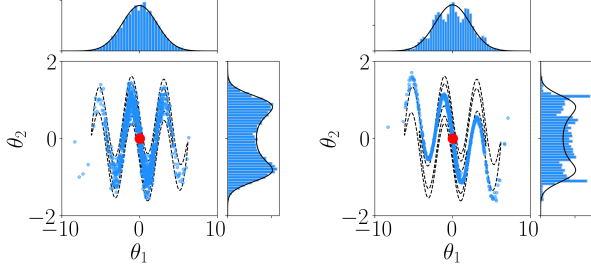
Figure 4: RLA-F (left) is exact (already for finite data) for the squiggle distribution with complex shape due to diffeomorphism with a Gaussian, whereas RLA-B (right) is too narrow everywhere and generally biased.

are symmetric, observing how Riemannian metrics transform, define a metric for the space of $\boldsymbol{\theta}$ as

$$G_{\boldsymbol{\Psi}} = \left(\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\psi}}\right)^{\top} G_{\boldsymbol{\Theta}} \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\psi}}.$$

With $(G_{\boldsymbol{\Psi}})_{i,j} = \mathbb{E}_{\boldsymbol{\psi}}\left(-\partial^2_{\mu_i,\mu_j} \log \mathcal{N}(\boldsymbol{\psi}\,|\,\boldsymbol{\mu}, \boldsymbol{S})\right) = \boldsymbol{S}^{-1}_{i,j}$ inspired by the Fisher metric, we have

$$G_{\boldsymbol{\Theta}} = \left(\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\psi}}\right)^{-\top} \boldsymbol{S}^{-1} \left(\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\psi}}\right)^{-1}.$$

Example. Consider the Squiggle distribution, used previously e.g. for evaluation of MCMC samplers (Hartmann et al., 2022),

$$\pi(\theta_1, \theta_2\,|\,\boldsymbol{\mu}, \boldsymbol{S}) = \mathcal{N}(\phi_a^{-1}(\theta_1, \theta_2)\,|\,\boldsymbol{\mu}, \boldsymbol{S}),$$
$$\phi_a^{-1}(\theta_1, \theta_2) = (\theta_1, \theta_2 + \sin(a\theta_1)).$$

This can be seen as a probability density as

$$\pi(\boldsymbol{\mu}\,|\,\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}\,|\,\phi_a^{-1}(\boldsymbol{\theta}), \boldsymbol{S}) \text{ and } \pi_J(\boldsymbol{\theta}) = \sqrt{\det G_{\boldsymbol{\Theta}}}$$

where π_J denotes the Jeffreys prior.

The Jacobian to form the Fisher metric is given by

$$\left(\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\psi}_a}\right)^{-1} = \begin{bmatrix} 1 & 0 \\ a\cos(a\theta_1) & 1 \end{bmatrix}$$

and for $\boldsymbol{\mu} = \boldsymbol{0}$ used here the Hausdorff MAP is also $\boldsymbol{0}$. As stated by Theorem 3, RLA-F is exact for this target due to the diffeomorphism detailed above. Figure 4 empirically validates this.

4.3.3 Computation of the Fisher metric

FIM for neural networks (NNs). The Fisher metric is known for many probabilistic models of basic form, i.e. the likelihood part (see Yang and Berger, 1996; George Casella, 2001). For instance, for exponential family the FIM always has closed-form expression (George Casella, 2001). Building on these, we

can easily form the Fisher metric for more complex models. Let ϕ be the parameters of a model $\pi(y|\phi)$ in its basic form in a P-dimensional parameter space Φ. Denote the FIM for Φ as $G_{\boldsymbol{\Phi}}(\cdot)$. Consider inputs $\{\boldsymbol{x}_n\}_{n=1}^N$ with associated observations $\boldsymbol{y} = \{y_n\}_{n=1}^N$ and a NN that maps $\mathcal{X} \ni \boldsymbol{x}_n \stackrel{f_{\boldsymbol{\theta}}}{\mapsto} \Phi$, where $\mathcal{X}$ is the space of inputs and $\boldsymbol{\theta}$ the vector of parameters. Let $\ell_{\boldsymbol{y}}(\boldsymbol{\theta}) = \sum_n^N \log \pi(y_n|f_{\boldsymbol{\theta}}(\boldsymbol{x}_n))$ denote the likelihood function of $\boldsymbol{\theta}$ and define $\phi_n = f_{\boldsymbol{\theta}}(\boldsymbol{x}_n) \in \Phi \;\forall n$. Using the chain rule for the Hessian matrix of $\ell_{\boldsymbol{y}}(\boldsymbol{\theta})$ w.r.t $\boldsymbol{\theta}$ the expression involving the score function vanishes in expectation. Therefore, the metric tensor on $\boldsymbol{\theta}$ reads

$$\mathbb{E}_{\boldsymbol{Y}}\big(-\nabla^2 \ell_{\boldsymbol{Y}}(\boldsymbol{\theta})_{i,j}\big) = -\sum_n^N \mathbb{E}_{Y_n}\left(\partial^2_{i,j} \log \pi\big(Y_n\,|\,\underbrace{f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)}_{\phi_n}\big)\right)$$
$$= \sum_n^N \boldsymbol{J}_n^{\top} G_{\boldsymbol{\Phi}}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_n))\boldsymbol{J}_n,$$

where $\boldsymbol{J}_n = [\nabla(\phi_n)_1 \cdots \nabla(\phi_n)_P]^{\top}$ is a $P \times D$ Jacobian matrix of the NN at the nth input. In other words, we can form the FIM for the whole network by transforming the FIM on its basic form with Jacobians that can be computed using standard automatic differentiation. Formally, this is a pullback metric from the parameter space $\boldsymbol{\Phi}$ to the NN parameter space $\boldsymbol{\Theta}$; see Section 14.2 in the Supplement for further details.

Computational cost. The core computational cost for RLA comes from obtaining the acceleration given the position and velocity, which is needed in each step of the ODE integrator. We have

$$\frac{\partial^2 \theta_k}{\partial t^2} = -\frac{1}{2}G^{kl}\left[\left(\partial_i G_{kl} + \partial_j G_{il} - \partial_l G_{ij}\right)v^i v^j\right],$$

which is a product of the inverse of the metric (G^{kl}) and a vector. For RLA-F, the computational cost is dominated by the inversion. The cost for drawing N samples hence becomes $O(NTD^3)$, where T is the number of evaluations during integration. RLA-B has lower cost due to avoiding direct inversion, but we will later see that with the Fisher metric we can often use considerably smaller T that balances the difference. Also note that already standard ELA has cost $O(D^3 + ND^2)$ due to inversion of Hessian and multivariate sampling. The sampling parallelizes over N for all methods.

5 EXPERIMENTS

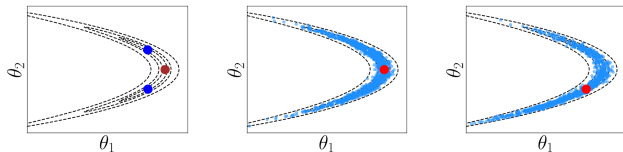Code for reproducing the experiments is available at https://github.com/ksnxr/RLAF.

Figure 5: Left: The banana distribution has two Euclidean MAPs (blue) but only one Hausdorff MAP (brown). RLA-F depends on the MAP choice, with Hausdorff (middle) being here superior to Euclidean (right).

5.1 Experimental setup

We evaluate three methods: RLA-F using the Fisher metric (Section 4.2), RLA-B as proposed by Bergamin et al. (2023), and RLA-BLog using the logarithmic map (Section 4.1). In addition, we show results for standard ELA and discuss the choice of the MAP estimate and covariance for sampling initial velocities when relevant. We repeat the experiments 5 times and report averaged results over the repetitions.

For evaluating the approximation accuracy, we generate $20{,}000$ samples using the NUTS sampler in Stan (Stan Dev Team, 2023) as the ground truth. For low-dimensional problems we directly compare the approximation with the posterior samples by computing the Wasserstein distance $\mathcal{W}$ as in Zhang et al. (2022). For the neural network experiments we compare the model predictions instead, due to non-identifiability of the posterior, measured using mean squared error (MSE) and negative log-likelihood (NLL).

Following Bergamin et al. (2023), we use a numerical ODE solver with adaptive step sizes. We report the actual number of integration steps T (and in some case running time) for the exponential map as an indicator of the complexity of the metric. The numbers are for producing one sample since the algorithm parallelizes trivially over the samples. For the NN experiment we use SciPy (Virtanen et al., 2020) for integration, and for other experiments Diffrax (Kidger, 2021). Additional details are provided in Supplement.

5.2 Banana distribution

We first compare the approximations in a 2D problem with difficult geometry, the banana distribution

$$\boldsymbol{\theta} \overset{\text{ind.}}{\sim} \mathcal{N}(0, \sigma_{\boldsymbol{\theta}}^2) \text{ and } Y_n | \boldsymbol{\theta} \sim \mathcal{N}(\theta_1 + \theta_2^2, \sigma_Y^2),$$

where we set $\sigma_{\boldsymbol{\theta}} = \sigma_Y = 2$, $N = 100$. The samples are obtained using $\theta_1 = 0.5$ and $\theta_2^2 = 0.75$. The distribution has closed-form FIM (Section 4.3.3 also applies).

The distribution has two modes symmetric across the

x-axis with the same log-posterior. However, the Hausdorff MAP is unique and at the x-axis. Figure 5 compares the resulting approximations with RLA-F for the different MAP estimates; with Euclidean MAP, we use Hessian precision for sampling the velocities, whereas with Hausdorff MAP we use the Fisher precision. Table 1 shows that all Riemannian methods clearly outperform classical ELA for this target, and for all Riemannian methods using the Hausdorff MAP is considerably better. Both of the newly proposed methods outperform RLA-B.

5.3 Bayesian logistic regression

Following Girolami and Calderhead (2011) and Lan et al. (2015), we applied Bayesian logistic regression on five datasets (details in Supplement) using the model

$$Y_n | \boldsymbol{\theta} \sim \text{Bernoulli}(\sigma(\boldsymbol{\theta}^\top \boldsymbol{x}_n)) \text{ and } \boldsymbol{\theta} \overset{\text{ind.}}{\sim} \mathcal{N}(0, \alpha),$$

where $\sigma : \mathcal{X} \to (0,1)$ is the sigmoid function and $\alpha = 100$. The parameter space of the Bernoulli model in its basic form is $\boldsymbol{\phi} \in (0,1) = \Phi$, whose FIM is known. Therefore the Fisher metric on the parameter space $\boldsymbol{\Theta}$ becomes $\boldsymbol{G} = \boldsymbol{X}^\top \boldsymbol{\Lambda} \boldsymbol{X} + \alpha^{-1} \boldsymbol{I}$, where $\boldsymbol{X} = [\boldsymbol{x}_1 \cdots \boldsymbol{x}_N]$ is the covariate matrix (or inputs) and $\boldsymbol{\Lambda}$ is a diagonal matrix with elements $\boldsymbol{\Lambda}_{nn} = \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}_n)(1 - \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}_n))$. We use Euclidean MAP, and here the negative Hessian coincides with the Fisher metric (Girolami and Calderhead, 2011) (so both choices for $\boldsymbol{\Sigma}$ are the same).

Table 2 reports approximation accuracies for two setups: With standardized (z-score) inputs to make the geometry of the problem easier, and with the raw inputs where the metric also needs to handle potentially large scale differences. For both cases, RLA-F is clearly the best on all data sets, and the approximations using the Monge metric are worse than ELA. The Fisher metric results in smoother integration surface, especially when not standardizing the inputs, with the Monge metric variants needing up to 1000 times more evaluations and consequently also more time despite lower complexity (see Supplement for times).

5.4 Neural network regression

Finally, we ran an experiment similar to the one by Bergamin et al. (2023), with data from Snelson and Ghahramani (2005). The task is a 1D regression problem using an NN of size 1-10-1 with tanh activation. The methods are implemented using Daxberger et al. (2021), with Fisher precision (the default option for full LA), $\hat{\boldsymbol{\theta}}$ found by standard MAP training, and prior precision and noise optimized posthoc. MSE and NLL are computed using 500 samples. Fisher metric and

	ELA	RLA-B		RLA-BLog		RLA-F	
MAP	$\mathcal{W}$	$\mathcal{W}$	T	$\mathcal{W}$	T	$\mathcal{W}$	T
Euclidean	[1.434, 0.01]	[0.811, 0.009]	70.4	[**0.788**, 0.013]	70.4	[0.791, 0.014]	**24.9**
Hausdorff	[1.386, 0.009]	[0.341, 0.006]	51.3	[0.208, 0.007]	60.0	[**0.143**, 0.009]	**32.7**

Table 1: Banana distribution results as [mean, std]. Bold font indicates the best method. $\mathcal{W}$ indicates Wasserstein distance to NUTS samples while T indicates the average number of function evaluations for one sample. For all evaluation metrics smaller is better.

		ELA	RLA-B		RLA-BLog		RLA-F	
	data	$\mathcal{W}$	$\mathcal{W}$	T	$\mathcal{W}$	T	$\mathcal{W}$	T
stand.	Ripl	[0.106, 0.004]	[0.236, 0.001]	29.7	[0.16, 0.065]	45.2	[**0.064**, 0.002]	**12.2**
	Pima	[0.149, 0.0]	[0.274, 0.0]	41.7	[0.21, 0.007]	78.5	[**0.147**, 0.0]	**12.2**
	Hear	[0.529, 0.001]	[0.649, 0.0]	40.2	[1.154, 0.189]	71.6	[**0.514**, 0.0]	**13.2**
	Aust	[0.441, 0.001]	[0.522, 0.001]	51.5	[0.745, 0.203]	79.1	[**0.417**, 0.001]	**18.0**
	Germ	[0.388, 0.001]	[0.431, 0.0]	58.4	[0.676, 0.1]	104.0	[**0.387**, 0.0]	**14.2**
raw	Ripl	[0.437, 0.017]	[0.489, 0.012]	30.2	[0.554, 0.253]	45.0	[**0.247**, 0.012]	**12.7**
	Pima	[0.21, 0.01]	[0.294, 0.004]	5632.9	[0.216, 0.014]	1690.4	[**0.112**, 0.008]	**15.4**
	Hear	[0.842, 0.026]	[0.96, 0.012]	*7868.2*	[0.898, 0.029]	*3161.1*	[**0.644**, 0.012]	**17.9**
	Aust	[0.454, 0.004]	[0.455, 0.01]	*18513.8*	[0.467, 0.005]	*12298.6*	[**0.378**, 0.005]	**18.0**
	Germ	[0.846, 0.002]	[0.92, 0.003]	3545.9	[0.898, 0.003]	2985.5	[**0.823**, 0.001]	**17.8**

Table 2: Logistic regression results as [mean, std]. Bold font indicates the best method. Italic font indicates the integrator reached maximum number of steps in at least one run. $\mathcal{W}$ indicates Wasserstein distance to NUTS samples while T indicates the average number of function evaluations for one sample. For all evaluation metrics smaller is better.

Christoffel symbols follow from Section 4.3.3 and are

$$G_{kl} = \frac{1}{\sigma^2} \sum_{n=1}^{N} \partial_k f_{\boldsymbol{\theta}}(\boldsymbol{x}_n) \partial_l f_{\boldsymbol{\theta}}(\boldsymbol{x}_n) + \alpha^{-1}\delta_l^k \text{ and }$$

$$v^i v^j \Gamma_{ij}^k = \frac{1}{\sigma^2} G^{kl} \sum_{n=1}^{N} \partial_l f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)\Big(v^i v^j \partial_{i,j}^2 f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)\Big),$$

as we use Gaussian prior with precision α on every $\boldsymbol{\theta}$ and Gaussian probabilistic model for Y; see Calin and Udrişte (2014) and Song et al. (2018) for derivations.

Following Bergamin et al. (2023), we run experiments with training samples covering a continuous area of inputs (*complete*) and with a gap between 1.5 and 3 (*gap*) to measure the in-between uncertainty quantification (Foong et al., 2019). Table 3 quantifies the results and shows that RLA-F is in general the best approximation and for the *complete* case matches also NUTS. As is well known, Euclidean LA does not work without linearization of the network (Daxberger et al., 2021). Figure 6 illustrates the two scenarios, where RLA-F is effectively indistinguishable from NUTS, whereas both RLA-B and RLA-BLog overestimate predictive variance and some of the posterior samples are clearly off.

6 DISCUSSION AND CONCLUSIONS

Main observation. Our key message is that the Riemannian Laplace Approximation proposed by Bergamin et al. (2023) is promising as practical and flexible approximation, but their specific variant is biased and the bias can be relatively large already in simple problems. The bias can be resolved by either slightly modifying the algorithm or by considering more suitable metrics, and the corrected approximation family has interesting theoretical properties. For instance, with the Fisher metric, it is exact for targets diffeomorphic with a Gaussian. This opens up new research directions for the study of metrics in inference tasks.

On Monge metric. One likely reason for the poor empirical performance of RLA-B is the (lack of) relative scaling of the identity and the outer product in the metric. Hartmann et al. (2022) and Yu et al. (2023) empirically showed that for MCMC the latter may need to be heavily down-weighted and even then it is not always better than Euclidean. RLA-B likely requires some form of scaling as well and downweighting the latter term would reduce the bias, but selecting the scaling remains an open problem. Hartmann et al.
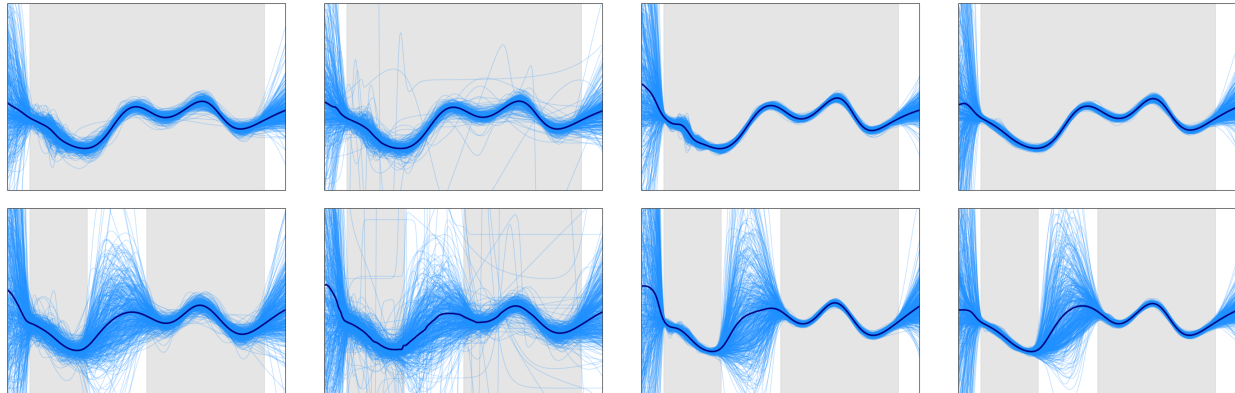
Figure 6: NN regression with complete (top) and gap (bottom) training data. Methods from left to right: RLA-B; RLA-BLog; RLA-F; NUTS. Gray shading denotes the part of x-axis with training data, dark line is the mean prediction, and blue lines are samples.

method	Complete				Gap			
	MSE	NLL	T	time	MSE	NLL	T	time
ELA	[2.235, 0.617]	[3.02, 0.064]	N/A	N/A	[1.365, 0.407]	[3.171, 0.075]	N/A	N/A
RLA-B	[0.2, 0.237]	[0.193, 0.003]	6677.8	34.1	[1.646, 2.722]	[1.031, 0.02]	6573.2	34.0
RLA-BLog	[0.107, 0.054]	[0.201, 0.004]	6674.7	37.7	[**0.27**, 0.068]	[1.122, 0.029]	4793.9	28.5
RLA-F	[**0.072**, 0.0]	[**0.121**, 0.002]	**242.9**	**0.8**	[0.564, 0.066]	[1.063, 0.017]	**727.7**	**2.4**
NUTS	[0.073, 0.0]	[0.126, 0.002]	N/A	N/A	[0.394, 0.02]	[**0.872**, 0.029]	N/A	N/A

Table 3: NN regression results as [mean, std]. Bold font indicates the best method. T indicates the average number of function evaluations for one sample while *time* indicates the average time for one sample. For all evaluation metrics smaller is better.

(2023) recently proposed a possible remedy that could be used as the basis for more accurate yet computationally efficient metric.

Recommendations. For small-to-medium problems RLA-F is clearly the best choice, based on consistent best performance and stable computation. For large-scale problems, for instance in the common use case of NNs (Daxberger et al., 2021), RLA-B remains currently the best choice, despite the bias.

We showed how to compute the FIM for arbitrary NNs, but only experimented on tiny ones to facilitate direct comparisons against NUTS. We also experimented with NN regression problems of varying numbers of parameters D and number of data points. Interestingly, RLA-F can be faster than RLA-B even for D larger than 1000; see Section 16.5.3 in the Supplement for details. We could naturally run RLA-F for still larger D in reasonable time, but very large models necessarily call for a more efficient metric. A promising direction for scalable and accurate method could build on scalable approximations for FIM (Martens and Grosse, 2015; George et al., 2018) with practical implementations as in Botev and Martens (2023) and George (2021), or on exact subsampled Fisher (Benz-

ing, 2022).

References

Arvanitidis, G., Hansen, L. K., and Hauberg, S. (2016). A Locally Adaptive Normal Distribution. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Atkinson, K., Han, W., and Stewart, D. (2009). *Numerical Solutions of Ordinary Differential Equations*. John Wiley & Sons.

Benzing, F. (2022). Gradient Descent on Neurons and

its Link to Approximate Second-order Optimization. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 1817–1853. PMLR.

Bergamin, F., Moreno-Muñoz, P., Hauberg, S., and Arvantidis, G. (2023). Riemannian Laplace approximations for Bayesian neural networks. In *Neural Information Processing Systems (NeurIPS)*.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877. Publisher: Taylor & Francis.

Botev, A. and Martens, J. (2023). KFAC-JAX.

Boumal, N. (2023). *An introduction to optimization on smooth manifolds*. Cambridge University Press.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

Brofos, J. A. and Lederman, R. R. (2021). On Numerical Considerations for Riemannian Manifold Hamiltonian Monte Carlo.

Calin, O. and Udrişte, C. (2014). *Geometric Modeling in Probability and Statistics*. Springer International Publishing, 1 edition.

Carmo, M. d. (1992). *Riemannian Geometry*. Birkhäuser, Boston.

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., and Hennig, P. (2021). Laplace Redux - Effortless Bayesian Deep Learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 20089–20103. Curran Associates, Inc.

De Bortoli, V., Mathieu, E., Hutchinson, M., Thornton, J., Teh, Y. W., and Doucet, A. (2022). Riemannian Score-Based Generative Modelling. In *Advances in Neural Information Processing Systems*, volume 35, pages 2406–2422. Curran Associates, Inc.

Dormand, J. R. and Prince, P. J. (1980). A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26.

Federer, H. (1969). *Geometric Measure Theory*. Springer-Verlag.

Feng, C., S., A., K., S., M., S., and Henery, R. (1992). Statlog Project. Published: UCI Machine Learning Repository.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T. H., Janati, H., Rakotomamonjy, A., Redko, I., Rolet, A., Schutz, A., Seguy, V., Sutherland, D. J., Tavenard, R., Tong, A., and Vayer, T. (2021). POT: Python Optimal Transport. *Journal of Machine Learning Research*, 22(78):1–8.

Foong, A. Y. K., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. (2019). 'In-Between' Uncertainty in Bayesian Neural Networks.

Frank, P., Leike, R., and Enßlin, T. A. (2021). Geometric Variational Inference. *Entropy*, 23(7).

George, T. (2021). NNGeometry: Easy and Fast Fisher Information Matrices and Neural Tangent Kernels in PyTorch.

George, T., Laurent, C., Bouthillier, X., Ballas, N., and Vincent, P. (2018). Fast Approximate Natural Gradient Descent in a Kronecker Factored Eigenbasis. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

George Casella, R. L. B. (2001). *Statistical Inference*. Duxbury Press, 2° edition.

Girolami, M. and Calderhead, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214.

Greff, K., Klein, A., Chovanec, M., Hutter, F., and Schmidhuber, J. (2017). The Sacred Infrastructure for Computational Research. In Huff, K., Lippa, D., Niederhut, D., and Pacer, M., editors, *Proceedings of the 16th Python in Science Conference*, pages 49 – 56.

Grosse, R. (2022). Chapter 3: Metrics.

Harris, C. R., Millman, K. J., Walt, S. J. v. d., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. v., Brett, M., Haldane, A., Río, J. F. d., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362. Publisher: Springer Science and Business Media LLC.

Hartmann, M., Girolami, M., and Klami, A. (2022). Lagrangian manifold Monte Carlo on Monge patches. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 4764–4781. PMLR.

Hartmann, M. and Vanhatalo, J. (2019). Laplace approximation and natural gradient for Gaussian

process regression with heteroscedastic student-t model. *Statistics and Computing*, 29(4):753–773.

Hartmann, M., Williams, B., Yu, H., Girolami, M., Barp, A., and Klami, A. (2023). Warped geometric information on the optimisation of Euclidean functions.

Hauberg, S. (2018). Directional Statistics with the Spherical Normal Distribution. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 704–711.

Jermyn, I. H. (2005). Invariant Bayesian estimation on manifolds. *The Annals of Statistics*, 33:583–605.

Kidger, P. (2021). *On Neural Differential Equations*. PhD Thesis, University of Oxford.

Kingma, D. P. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Knollmüller, J. and Enßlin, T. A. (2020). Metric Gaussian Variational Inference.

Kristiadi, A., Dangel, F., and Hennig, P. (2023). The Geometry of Neural Nets' Parameter Spaces Under Reparametrization.

Kunstner, F., Hennig, P., and Balles, L. (2019). Limitations of the empirical Fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Lan, S., Stathopoulos, V., Shahbaba, B., and Girolami, M. (2015). Markov Chain Monte Carlo From Lagrangian Dynamics. *Journal of Computational and Graphical Statistics*, 24(2):357–378.

Lee, John M. (2018). *Introduction to Riemannian Manifolds*. Graduate Texts in Mathematics. Springer International Publishing AG, Cham, Switzerland, 2nd edition.

Lehmann, G. C. (2003). *Theory of Point Estimation*. Springer texts in statistics. Springer, 2nd ed edition.

Magnusson, M., Bürkner, P., and Vehtari, A. (2022). posteriordb: a set of posteriors for Bayesian inference and probabilistic programming.

Margossian, C. C. and Blei, D. M. (2023). Amortized Variational Inference: When and Why?

Martens, J. (2020). New Insights and Perspectives on the Natural Gradient Method. *Journal of Machine Learning Research*, 21(146):1–76.

Martens, J. and Grosse, R. (2015). Optimizing Neural Networks with Kronecker-factored Approximate Curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2408–2417, Lille, France. PMLR.

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Roučka, v., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., and Scopatz, A. (2017). SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103.

Minka, T. P. (2001). Expectation propagation for approximate bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, page 362–369. Morgan Kaufmann Publishers Inc.

Neal, R. M. (2003). Slice sampling. *The Annals of Statistics*, 31(3):705 – 767. Publisher: Institute of Mathematical Statistics.

OpenAI (2023). ChatGPT.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 22(57):1–64.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Patterson, S. and Teh, Y. W. (2013). Stochastic Gradient Riemannian Langevin Dynamics on the Probability Simplex. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Petersen, K. B. and Pedersen, M. S. (2012). The Matrix Cookbook.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press.

Ripley, B. D. (1994). Neural networks and related methods for classification (with discussion). *Journal of the Royal Statistical Society series B*, pages 409–456.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.

Rue, H., Martino, S., and Chopin, N. (2009). Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations. *Journal Of The Royal Statistical Society*, 71:319–392.

Schervish, M. (2012). *Theory of Statistics.* Springer Series in Statistics. Springer New York.

Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., and Johannes, R. S. (1988). Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, Orlando.

Snelson, E. and Ghahramani, Z. (2005). Sparse Gaussian Processes using Pseudo-inputs. In *Advances in Neural Information Processing Systems*, volume 18. MIT Press.

Song, Y., Song, J., and Ermon, S. (2018). Accelerating Natural Gradient with Higher-Order Invariance. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4713–4722. PMLR.

Stan Dev Team (2023). CmdStanPy.

Stan Development Team (2023). Stan Modeling Language Users Guide and Reference Manual.

Tierney, L. and Kadane, J. B. (1986). Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, 81(393):82–86.

van der Vaart, A. (1998). *Asymptotic Statistics.* Cambridge University Press, Cambridge, UK.

Vershynin, R. (2018). *High-Dimensional Probability: An Introduction with Applications in Data Science.* Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, I., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Yang, R. and Berger, J. (1996). *A Catalog of Noninformative Priors.* Discussion papers. Institute of Statistics and Decision Sciences, Duke University.

Yu, H., Hartmann, M., Williams, B., and Klami, A. (2023). Scalable Stochastic Gradient Riemannian Langevin Dynamics in Non-Diagonal Metrics. *Transactions on Machine Learning Research*.

Zhang, L., Carpenter, B., Gelman, A., and Vehtari, A. (2022). Pathfinder: Parallel quasi-Newton variational inference. *Journal of Machine Learning Research*, 23(306):1–49.

Checklist

1. For all models and algorithms presented, check if you include:

    (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]

    (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]

    (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:

    (a) Statements of the full set of assumptions of all theoretical results. [Yes]

    (b) Complete proofs of all theoretical results. [Yes]

    (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:

    (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]

    (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]

    (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]

    (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

    (a) Citations of the creator If your work uses existing assets. [Yes]

    (b) The license information of the assets, if applicable. [Yes]

    (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]

    (d) Information about consent from data providers/curators. [Not Applicable]

    (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

    (a) The full text of instructions given to participants and screenshots. [Not Applicable]

    (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

    (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Riemannian Laplace Approximation with the Fisher Metric: Supplementary Materials

7 CONTENTS

This supplement complements the main article, by providing the proofs for the theorems, presenting details about the experiments as well as some additional empirical results, and expanding the theoretical basis. For clarity, we number the Sections, Equations *etc.* with a range that does not overlap with the main paper.

Section 8 provides the proofs of the theorems stated in the main paper. Section 9 justifies the usage of Hausdorff MAP along with Fisher precision. Section 10 extends the theoretical discussion of the approximation in general, by providing a theorem characterising the coverage of RLA. Section 12 provides background on numerically solving ODEs and demonstrates how to formulate solving the exponential and logarithmic maps as ODE problems. Section 13 demonstrates how to obtain the Christoffel symbols based on a differentiable expression of the Riemannian metric using JAX (Bradbury et al., 2018), and Section 14 provides additional details concerning the computation of the Fisher metric and the resulting Christoffel symbols for certain models. Section 15 provides all sorts of details for the empirical experiments reported in the main article. Finally, Section 16 provides additional experimental results, for instances visual illustrations that were omitted from the main paper due to space constraints as well as demonstration of the methods for one more target distribution.

Code for reproducing the experiments is available at `https://github.com/ksnxr/RLAF`.

8 PROOFS

8.1 Theorem 1

Theorem 1. *For Gaussian (or uniform) prior and Gaussian likelihood with fixed covariance, the Fisher metric is constant.*

Proof. Consider the prior distribution on the parameters of interest. We consider two cases, the first where the prior is Gaussian with a fixed covariance and the second where the prior is uniform. For the first case, we have $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{S})$ for some $\boldsymbol{\mu}$ and $\boldsymbol{S}$. Therefore,

$$\log \pi(\boldsymbol{\theta}) = -\frac{k}{2}\log(2\pi) - \frac{1}{2}\log\det(\boldsymbol{S}) - \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu})^\top \boldsymbol{S}^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}),$$

and

$$\frac{\partial^2 \log \pi(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^2} = -\boldsymbol{S}^{-1}.$$

The negative Hessian is therefore constant. For the second case, where the prior is uniform, the negative Hessian is 0 everywhere, thus is also constant.

Consider the Gaussian distribution $\boldsymbol{Y} \mid \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\Sigma})$ for some fixed $\boldsymbol{\Sigma}$. We have

$$\frac{\partial^2 \log \pi(\boldsymbol{y} \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}^2} = -\boldsymbol{\Sigma}^{-1},$$

which does not depend on $\boldsymbol{y}$, thus the FIM is constant.

Therefore, the resulting Fisher metric is constant. □

8.2 Theorem 2

Theorem 2. *Under the conditions of Bernstein-von Mises theorem, the Fisher metric converges to a fixed matrix at the limit of infinite data.*

Proof. At the limit of infinite data, up to the regularity conditions characterised by the Bernstein-von Mises theorem (van der Vaart, 1998), the likelihood dominates the prior and the posterior converges in distribution to a Gaussian given by $\mathcal{N}(\boldsymbol{\theta}_{\mathrm{MLE}}, (NI(\boldsymbol{\theta}_{\mathrm{MLE}}))^{-1})$, where $\boldsymbol{\theta}_{\mathrm{MLE}}$ denotes the maximum likelihood estimate. In other words, with the same conditions, the prior's contribution converges to 0, and the likelihood converges to a Gaussian function, up to a scaling, given by $\mathcal{N}(\boldsymbol{\theta}_{\mathrm{MLE}} \,|\, \boldsymbol{\theta}, (NI(\boldsymbol{\theta}_{\mathrm{MLE}}))^{-1})$.

Since the FIM of the Gaussian model is given by $NI(\boldsymbol{\theta}_{\mathrm{MLE}})$, which is constant for all $\boldsymbol{\theta}$, the FIM of the limiting model is also constant, and the resulting convergence limit of the metric of the current model is also constant. □

8.3 Theorem 3

Theorem 3. *With an invariant prior, e.g. Jeffreys prior, RLA-F with Hausdorff MAP is exact for probabilistic models whose target distributions are diffeomorphic with Gaussians, for which the negative Hessian at the Hausdorff MAP coincides with the Fisher metric.*

Proof. Consider first the simple case with no transformations, such that the probabilistic model reduces to the following form

$$\pi(\boldsymbol{\mu} \,|\, \boldsymbol{\psi}) = \mathcal{N}(\boldsymbol{\mu} \,|\, \boldsymbol{\psi}, \boldsymbol{S}),$$
$$\pi(\boldsymbol{\psi}) = \mathrm{Jeffreys}(\boldsymbol{\psi}).$$

The FIM for $\boldsymbol{\psi}$ is constant, given by $\boldsymbol{S}^{-1}$. Therefore, The prior corresponds to a uniform distribution in the Euclidean sense, and the posterior is given by

$$\pi(\boldsymbol{\psi} \,|\, \boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\psi} \,|\, \boldsymbol{\mu}, \boldsymbol{S}).$$

It is clear that RLA-F is exact for this posterior.

Consider now a diffeomorphic transformation $\boldsymbol{\phi} : \boldsymbol{\psi} \to \boldsymbol{\theta}$. It is well known that the Fisher Information Matrix automatically transforms according to the correct transformation rule of a Riemannian metric (Martens, 2020; Kristiadi et al., 2023). The distribution of the transformed parameters is given by $\pi(\boldsymbol{\theta} \,|\, \boldsymbol{\mu}) = \pi(\boldsymbol{\phi}^{-1}(\boldsymbol{\theta}) \,|\, \boldsymbol{\mu}) \left| \det \frac{\partial \boldsymbol{\phi}^{-1}}{\partial \boldsymbol{\theta}} \right|$. It coincides with the posterior of the probabilistic model

$$\pi(\boldsymbol{\mu} \,|\, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu} \,|\, \boldsymbol{\phi}^{-1}(\boldsymbol{\theta}), \boldsymbol{S}),$$
$$\pi_J(\boldsymbol{\theta}) = \sqrt{\det \boldsymbol{G}_{\boldsymbol{\Theta}}(\boldsymbol{\theta})},$$

since Jeffreys prior accounts for the change of variable of the transformation,

$$\pi_J(\boldsymbol{\theta}) = \sqrt{\det \boldsymbol{G}_{\boldsymbol{\Theta}}(\boldsymbol{\theta})} = \left| \det \frac{\partial \boldsymbol{\phi}^{-1}}{\partial \boldsymbol{\theta}} \right| \sqrt{\det \boldsymbol{G}_{\boldsymbol{\Psi}}(\boldsymbol{\phi}^{-1}(\boldsymbol{\theta}))}.$$

The parameters that give the maximum value under the Hausdorff measure $\pi(\boldsymbol{\theta} \,|\, \boldsymbol{\mu})(\det \boldsymbol{G}_{\boldsymbol{\Theta}}(\boldsymbol{\theta}))^{-\frac{1}{2}} \propto \pi(\boldsymbol{\mu} \,|\, \boldsymbol{\phi}^{-1}(\boldsymbol{\theta}))$ are always given by the MLE estimate $\hat{\boldsymbol{\theta}} = \boldsymbol{\phi}(\boldsymbol{\mu})$.

The argument for exactness is as follows. Consider the two manifolds $\mathcal{M} : (\boldsymbol{\Psi}, \boldsymbol{G}_{\boldsymbol{\Psi}})$ and $\mathcal{N} : (\boldsymbol{\Theta}, \boldsymbol{G}_{\boldsymbol{\Theta}})$. Since the corresponding metrics satisfy the transformation rule

$$\boldsymbol{G}_{\boldsymbol{\Theta}} = \left(\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{\theta}} \right)^{\top} \boldsymbol{G}_{\boldsymbol{\Psi}} \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{\theta}}, \tag{4}$$

there exists an isomorphism between the tangent spaces of $\mathcal{M}$ and $\mathcal{N}$. As a result, the exponential map on $\mathcal{M}$ and $\mathcal{N}$ and the tangent vectors at the center of the distribution transform correctly. Since RLA-F is exact for the distribution corresponding to $\mathcal{M}$, it is also exact for the distribution corresponding to $\mathcal{N}$.

Moreover, the negative Hessian at the Hausdorff MAP coincides with the Fisher metric. This can be seen from the transformation rule of Hessian matrix under a transformation given by $\boldsymbol{\theta} = \boldsymbol{\phi}(\boldsymbol{\psi})$

$$\nabla_{\boldsymbol{\theta}}^2 f = \left(\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{\theta}}\right)^{\top} \nabla_{\boldsymbol{\psi}}^2 f \left(\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{\theta}}\right) + [\nabla_{\boldsymbol{\theta}}^2 \phi_1^{-1}(\boldsymbol{\theta}), \ldots, \nabla_{\boldsymbol{\theta}}^2 \phi_n^{-1}(\boldsymbol{\theta})] \left(\nabla_{\boldsymbol{\psi}} f \otimes \boldsymbol{I}_n\right),$$

because for the MAP estimate, $\nabla_{\boldsymbol{\psi}} f = 0$. $\qquad\square$

9   USING HAUSDORFF MAP AND FISHER PRECISION

In Section 4.3.1, we briefly discussed the choices of using Hausdorff MAP together with Fisher precision. Here we provide additional remarks on this.

We use notations inspired by Kristiadi et al. (2023). Consider a reparametrization $\boldsymbol{\phi} : \boldsymbol{\psi} \to \boldsymbol{\theta}$. Denote its Jacobian as $\boldsymbol{J}$, with

$$J_{ij} = \frac{\partial \theta_i}{\partial \psi_j}.$$

As noted by Kristiadi et al. (2023), the transformation rule of vector components is

$$\boldsymbol{v}_{\boldsymbol{\Theta}} = \boldsymbol{J}\,\boldsymbol{v}_{\boldsymbol{\Psi}},$$

and the transformation rule of covector components is

$$\boldsymbol{w}_{\boldsymbol{\Theta}} = \boldsymbol{J}^{-\top}\,\boldsymbol{w}_{\boldsymbol{\Psi}}.$$

Recall that the FIM transforms as a Riemannian metric under one-to-one mappings (Martens, 2020; Kristiadi et al., 2023). Therefore, with an invariant prior or ignoring the effect of the prior, the resulting metric transforms automatically.

Typical MAP estimates are not invariant under reparametrization. However, we can define an invariant MAP estimate by taking into account the Riemannian structure (Kristiadi et al., 2023). Specifically, instead of calculating the maximum of the PDF under Lebesgue measure, we calculate it under the Hausdorff measure. Denote the PDF under Lebesgue measure as $\pi_{\boldsymbol{\Theta}}(\boldsymbol{\theta})$, and the PDF under Hausdorff measure as $\pi_{\boldsymbol{\Theta}}^{\boldsymbol{G}}(\boldsymbol{\theta})$. Then, using the Riemannian volume form (Lee, John M., 2018), we have

$$\pi_{\boldsymbol{\Theta}}^{\boldsymbol{G}}(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\Theta}}(\boldsymbol{\theta})}{\sqrt{\det \boldsymbol{G}_{\boldsymbol{\Theta}}(\boldsymbol{\theta})}}.$$

Theorem 4. *The PDF under Hausdorff measure is invariant across reparametrization.*

Proof.

$$\begin{aligned}
\pi_{\boldsymbol{\Theta}}^{\boldsymbol{G}}(\boldsymbol{\theta}) &= \frac{\pi_{\boldsymbol{\Psi}}(\boldsymbol{\phi}^{-1}(\boldsymbol{\theta}))\sqrt{\det \boldsymbol{J}^{-1}(\boldsymbol{\theta})}}{\sqrt{\det\left(\boldsymbol{J}^{-T}(\boldsymbol{\theta})\,\boldsymbol{G}_{\boldsymbol{\Psi}}(\boldsymbol{\phi}^{-1}(\boldsymbol{\theta}))\,\boldsymbol{J}^{-1}(\boldsymbol{\theta})\right)}} \\
&= \frac{\pi_{\boldsymbol{\Psi}}(\boldsymbol{\phi}^{-1}(\boldsymbol{\theta}))}{\sqrt{\det \boldsymbol{G}_{\boldsymbol{\Psi}}(\boldsymbol{\phi}^{-1}(\boldsymbol{\theta}))}} = \frac{\pi_{\boldsymbol{\Theta}}(\boldsymbol{\theta})}{\sqrt{\det \boldsymbol{G}_{\boldsymbol{\Theta}}(\boldsymbol{\theta})}}.
\end{aligned}$$

$\qquad\square$

This naturally leads to the following corollary

Corollary 1. *The Hausdorff MAP is invariant across reparameterization.*

The following theorem shows that we can ignore the first order gradients in the Taylor series expansions as in Equation (2) of the main paper

Theorem 5. *Under normal coordinates, the Hausdorff MAP is a critical point under Lebesgue measure.*

Proof. It is a standard result in Riemannian geometry that for a Riemannian manifold, at each tangent space there exists a normal coordinate (Lee, John M., 2018), such that the components of the metric at the point form an identity matrix, and all first partial derivatives of the metric vanishes at the point (see Proposition 5.24 from Lee, John M., 2018). Therefore, consider such a transformation to the normal coordinate $\phi : \psi \to \theta$. $\hat{\theta} = \phi(\hat{\psi})$ is the Hausdorff MAP due to Theorem 1. Recall the definition of Hausdorff MAP; it is clear that $\frac{\partial \pi^{G}(\hat{\theta})}{\partial \theta_i} = 0$.

Moreover, since the first partial derivatives of the metric vanishes under a normal coordinate, recall $\partial(\det(\boldsymbol{X})) = \det(\boldsymbol{X}) \operatorname{Tr}\left(\boldsymbol{X}^{-1} \partial \boldsymbol{X}\right)$ (Petersen and Pedersen, 2012), we have $\frac{\partial \sqrt{\det \boldsymbol{G}(\hat{\theta})}}{\partial \theta_i} = 0$. Therefore,

$$\frac{\partial \pi(\hat{\theta})}{\partial \theta_i} = \frac{\partial \pi^{G}(\hat{\theta})\sqrt{\det \boldsymbol{G}(\hat{\theta})}}{\partial \theta_i} = \frac{\partial \pi^{G}(\hat{\theta})}{\partial \theta_i}\sqrt{\det \boldsymbol{G}(\hat{\theta})} + \pi^{G}(\hat{\theta})\frac{\partial \sqrt{\det \boldsymbol{G}(\hat{\theta})}}{\partial \theta_i} = 0.$$

$\square$

Under an invariant prior or ignoring the effect of prior, the Fisher metric transforms as a Riemannian metric, and RLA-F with Hausdorff MAP and Fisher precision can be interpreted as forming the approximate Gaussian with Fisher metric in place of the negative Hessian at $\hat{\theta}$ under normal coordinates and transforming the approximation back to the current approximation.

The following theorem justifies using the Fisher metric as the precision of the Gaussian approximation from a differential geometry viewpoint.

Theorem 6. *When the precision of a Gaussian distribution is a Riemannian metric, the samples from the Gaussian follow the transformation rule of the tangent vectors.*

Proof. We analyze the transformation property of samples from the resulting Gaussian distribution.

The transformation of the components of a Riemannian metric, which is a $(0, 2)$ tensor, follows

$$\boldsymbol{G}_{\boldsymbol{\Theta}} = \boldsymbol{J}^{-\top} \boldsymbol{G}_{\boldsymbol{\Psi}} \boldsymbol{J}^{-1}.$$

The transformation rule of the inverse of the Riemannian metric follows

$$\boldsymbol{G}_{\boldsymbol{\Theta}}^{-1} = \boldsymbol{J} \boldsymbol{G}_{\boldsymbol{\Psi}}^{-1} \boldsymbol{J}^{\top} = (\boldsymbol{J} \boldsymbol{L}_{\boldsymbol{\Psi}})(\boldsymbol{J} \boldsymbol{L}_{\boldsymbol{\Psi}})^{\top},$$

where $\boldsymbol{L}$ indicates the decomposition of the inverse of the metric. We therefore have

$$\boldsymbol{L}_{\boldsymbol{\Theta}} = \boldsymbol{J} \boldsymbol{L}_{\boldsymbol{\Psi}},$$

and the samples from the resulting Gaussian distribution transform in the same way as vector components. $\square$

10 COVERAGE OF RLA SAMPLES

An interesting insight is that, under mild assumptions, we can obtain samples over the entire $\mathbb{R}^{D}$ space for any metric and any starting point.

Theorem 7. *For target distributions of unconstrained parameters, if the induced Riemannian manifold $(\boldsymbol{\Theta}, g)$ is connected, there always exists one length-minimizing geodesic starting from the MAP and passing through an arbitrary point $\boldsymbol{\theta}$ on the induced manifold.*

Proof. Under unconstrained parametrization, for a point p on the connected Riemannian manifold $\boldsymbol{\Theta}$, the geodesics are defined on the entire tangent space. Therefore, using Lemma 6.18 from (Lee, John M., 2018), for any two points $\boldsymbol{p}, \boldsymbol{q} \in \boldsymbol{\Theta}$, there exists one length-minimizing geodesic. $\square$

11 BIAS OF RLA-B

Consider a D dimensional Gaussian distribution with mean $\mathbf{0}$ and covariance $\boldsymbol{I}_D$. Isotropic multivariate Gaussian distributions are known to be rotation invariant (Vershynin, 2018), and the distribution is the same from all directions at the origin. Moreover, the distances from the samples to the origin follow a χ distribution by definition. Also hinted by Vershynin (2018), it is clear that we can represent samples from an isotropic multivariate Gaussian distribution of dimension D such that the directions are sampled uniformly at random from a sphere, and the distances to the origin sampled from a χ distribution with degrees of freedom D. In this case, for RLA-B, an algorithm based on the Riemannian metric using the gradient information of the distribution, is also rotation invariant, and the geodesics travel in straight lines. As such, we can consider an arbitrary direction without losing generality.

Specifically, we consider the following case

$$
\boldsymbol{v} = \begin{bmatrix} v \\ \vdots \\ v \end{bmatrix}_D, \boldsymbol{x} = \begin{bmatrix} x \\ \vdots \\ x \end{bmatrix}_D, \nabla\ell = \begin{bmatrix} -x \\ \vdots \\ -x \end{bmatrix}_D, \boldsymbol{v}_0 = \begin{bmatrix} v_0 \\ \vdots \\ v_0 \end{bmatrix}_D,
$$

where a sample with initial velocity $\boldsymbol{v}_0$ reaches position $\boldsymbol{x}$, at which point the gradient is given by $\nabla\ell$ and the sample has velocity $\boldsymbol{v}$, with every occurrence of the same letter denoting the same value, and v, x and v_0 being non-negative. It is clear that all particles with $\boldsymbol{v}_0$ of the above form reach $\boldsymbol{x}(t)$ at some time. In the following analysis, we assume that all the samples reach the point $\boldsymbol{x}$ within time 1 for the value of v_0 considered for simplicity; this generally holds for sufficiently small x. Recall that $\|\boldsymbol{v}(t)\|^2 + \langle \boldsymbol{v}(t), \nabla\ell \rangle^2 = \|\boldsymbol{v}_0\|^2$ due to the property of the geodesic, we have

$$
Dv^2 + (Dvx)^2 = Dv_0^2,
$$
$$
v^2 + Dx^2v^2 = v_0^2,
$$
$$
v^2 = \frac{v_0^2}{1 + Dx^2}.
$$

As such, for fixed v_0 and x, with RLA-B, as D increases, v decreases. However, under the Euclidean metric, given v_0 and x, v is constant for all D. This demonstrates that, for a fixed v_0, the final distance traveled by the sample decreases as D increases, and, in a sense, the resulting algorithm leads to additional bias as D increases.

Additionally, we can consider the expected value of v at a given position x under v_0. We can define the distribution of v_0 such that it induces the correct distribution of distances to the origin, in which case $\sqrt{D}v_0$ follows a χ distribution with degrees of freedom D. We have

$$
\mathbb{E}[v] = \mathbb{E}\left[\sqrt{\frac{v_0^2}{1 + Dx^2}} \right] = \mathbb{E}\left[\frac{v_0}{\sqrt{1 + Dx^2}} \right].
$$

Consider a positive x. It is clear that $\frac{1}{\sqrt{1+Dx^2}}$ decreases as D increases. For smaller D, the expected velocity as calculated above may increase as D increases, since $\mathbb{E}[v_0]$ itself may increase. However, as D approaches infinity, $\mathbb{E}\left[\sqrt{D}v_0\right]$ approaches $\sqrt{D-1}$, $\mathbb{E}[v_0]$ approaches 1 and $\mathbb{E}[v]$ approaches 0.

The above theoretical analysis complements the empirical evidence as shown in Figure 3, which demonstrates that the bias, as measured by the Wasserstein distances from approximate samples to true samples computed for the first dimension, grows continuously as D increases from 1 to 10.

12 GEODESIC ODE

In this section, we provide some background on the numerical solutions of ODEs, while also demonstrating the formulations of exponential map and logarithmic map as ODE problems. Unless otherwise stated, the technical contents are based on Atkinson et al. (2009).

Generally, an ODE system can be written in the form of

$$
\frac{\mathrm{d}\,\boldsymbol{x}(t)}{\mathrm{d}t} = \boldsymbol{f}(t, \boldsymbol{x}(t)),
$$

where $\boldsymbol{x}$ is an unknown function dependent on time t, and $\boldsymbol{f}$ is a function describing the change in $\boldsymbol{x}$.

The geodesic equation induces the following ODE system

$$\frac{\mathrm{d}[\boldsymbol{\theta}(t), \boldsymbol{v}(t)]}{\mathrm{d}t} = [\boldsymbol{v}(t), \boldsymbol{a}(t)],$$

where $\boldsymbol{a}(t)$ is given by the geodesic equation, such that

$$a^k(t) = -v^i(t)v^j(t)\Gamma_{ij}^k(t).$$

It is a nonlinear second order ODE.

An initial value problem (IVP) solves for $\boldsymbol{x}(b)$ given the following conditions

$$\frac{\mathrm{d}\,\boldsymbol{x}(t)}{\mathrm{d}t} = \boldsymbol{f}(t, \boldsymbol{x}(t)), \quad t_0 \leq t \leq b,$$
$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0\,.$$

Probably the simplest solver is Euler's method, which divides the times into a discrete set of nodes $t_0 < t_1 < t_2 < \cdots < t_N \leq b$, and iteratively solves it as

$$\boldsymbol{x}(n+1) = \boldsymbol{x}(n) + h\,\boldsymbol{f}(t_n, \boldsymbol{x}_n).$$

However, this is a naive method, and often leads to large integration errors. In practice, one commonly used solver is Dormand-Prince's 5(4) method (Dormand and Prince, 1980); this is the default option in SciPy (Virtanen et al., 2020). It generally requires 6 function evaluations per step.

A general nonlinear two-point boundary value problem (BVP) can be formulated as

$$\boldsymbol{x}''(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{x}'(t)), \quad a < t < b,$$
$$\boldsymbol{A} \begin{bmatrix} \boldsymbol{x}(a) \\ \boldsymbol{x}'(a) \end{bmatrix} + \boldsymbol{B} \begin{bmatrix} \boldsymbol{x}(b) \\ \boldsymbol{x}'(b) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\gamma}_1 \\ \boldsymbol{\gamma}_2 \end{bmatrix},$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are square matrices, and the conditions on the second line are known as the boundary conditions.

The logarithmic map of the geodesic ODE can be formulated as a BVP, by observing that we can make the following choices

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I} & 0 \\ 0 & 0 \end{bmatrix}, \ \boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ \boldsymbol{I} & 0 \end{bmatrix}, \ \boldsymbol{\gamma}_1 = \boldsymbol{x}_a, \ \boldsymbol{\gamma}_2 = \boldsymbol{x}_b,$$

and can thus be solved by a BVP solver.

13 CHRISTOFFEL SYMBOLS BASED ON EXPRESSION OF THE RIEMANNIAN METRIC

With modern autodiff frameworks, it is possible to directly calculate the Christoffel symbols for a given $\boldsymbol{\theta}$, given that we have a differentiable expression of the Riemannian metric. As noted by Song et al. (2018), we only need the quantities in the form of

$$\Gamma_{ij}^k v^i v^j.$$

We provide a JAX (Bradbury et al., 2018) implementation to obtain the above quantities as below. [1]

```
import jax
import jax.numpy as jnp
import jax.scipy as jsp
```

---

[1] In all presented experiments *jax.numpy.linalg.solve* is used instead of *jax.scipy.linalg.solve*. However, the implementation in *jax.scipy* could make use of the positive definite structure of the matrix, theoretically leading to faster inversions, and we thus present this implementation.

```
def christoffel_fn (g, theta , v):
    # Adapted based on ChatGPT
    d_g = jax.jacfwd(g)(theta)

    # Compute the Christoffel symbols
    partial_1 = jnp.einsum("jli ,i ,j—>l", d_g, v, v)
    partial_2 = jnp.einsum("ilj ,i ,j—>l", d_g, v, v)
    partial_3 = jnp.einsum("ijl ,i ,j—>l", d_g, v, v)
    result = jsp.linalg.solve(g(theta), 0.5 * (partial_1 + partial_2 − partial_3),
    assume_a="pos")

    return result
```

In the above code, *theta* and *v* are the current position and velocity, and *g* is a differentiable function that returns the Riemannian metric given the position. It is possible to analytically derive the Christoffel symbols given the expression of the metric. We observe that using analytical expressions can lead to more efficient and numerically stable results. We show some analytical derivations in Section 14. However, we observed that for small scale problems, using the numerical results are often fast and accurate enough.

14 FISHER METRIC AND CHRISTOFFEL SYMBOL FOR SPECIFIC MODELS

14.1 Logistic regression

In Section 5.3, we considered Bayesian logistic regression. Here we present some derivations on the Christoffel symbols of Fisher metric for that problem.

For logistic regression, the partial derivatives of the Fisher metric are given by (Girolami and Calderhead, 2011)

$$\frac{\partial\, \boldsymbol{G}(\boldsymbol{\theta})}{\partial \theta_i} = \boldsymbol{X}^\top \boldsymbol{\Lambda}\, \boldsymbol{V}^i\, \boldsymbol{X},$$

where, using s_k to denote $s(\boldsymbol{X}_k\, \boldsymbol{\theta})$, we have $\Lambda_{kk} = s_k(1 - s_k)$ and $V^i_{kk} = (1 - 2s_k)X_{ni}$.

One interesting property is the following (Lan et al., 2015)

$$\frac{\partial(\boldsymbol{G}(\boldsymbol{\theta}))_{jl}}{\partial \theta_i} = \frac{\partial(\boldsymbol{G}(\boldsymbol{\theta}))_{il}}{\partial \theta_j} = \frac{\partial(\boldsymbol{G}(\boldsymbol{\theta}))_{ij}}{\partial \theta_l}.$$

In order to see that, note that

$$\frac{\partial(\boldsymbol{G}(\boldsymbol{\theta}))_{jl}}{\partial \theta_i} = (\boldsymbol{X}^\top \boldsymbol{\Lambda}\, \boldsymbol{V}^i\, \boldsymbol{X})_{jl} = \sum_k X_{kj}(\boldsymbol{\Lambda}\, \boldsymbol{V}^i)_{kk}X_{kl} = \sum_k X_{kj}s_k(1 - s_k)(1 - 2s_k)X_{ki}X_{kl}$$

$$= \frac{\partial(\boldsymbol{G}(\boldsymbol{\theta}))_{il}}{\partial \theta_j} = \frac{\partial(\boldsymbol{G}(\boldsymbol{\theta}))_{ij}}{\partial \theta_l}.$$

Therefore, the resulting Christoffel symbols have a relatively simple form, given by

$$\Gamma^m_{ij} = \frac{1}{2}G^{ml}\left(\partial_i G_{jl} + \partial_j G_{il} - \partial_l G_{ij}\right) = \frac{1}{2}G^{ml}\partial_i G_{jl}.$$

14.2 Neural Networks (NN)

Here we expand the discussions on FIM for NNs in Section 4.3.3 of the main paper.

Recall the probabilistic model $\pi(\boldsymbol{y}\,|\,\boldsymbol{\phi})$ in its basic form. Given an NN that maps $\mathcal{X} \overset{f_\theta}{\mapsto} \boldsymbol{\Phi}$ where $\boldsymbol{\Phi}$ is a P-dimensional parameter space with given Fisher metric, we can define a Fisher metric on the parameter space of

the NN as follows. Using the chain rule for the Hessian matrix of $\ell_{\boldsymbol{y}}(\boldsymbol{\theta})$ w.r.t $\boldsymbol{\theta}$, observe that the extra expression involving the score function will be zero in expectation (Schervish, 2012, Definition 2.78, p. 111). Therefore, the metric tensor w.r.t $\boldsymbol{\theta}$ reads

$$\mathbb{E}(-\nabla^2 \ell_{\boldsymbol{Y}}(\boldsymbol{\theta})_{i,j}) = \mathbb{E}\left(-\sum_n^N \partial_{i,j}^2 \log \pi(Y_n | \boldsymbol{\phi}_n)\right)$$

$$= \sum_n^N \sum_p^P \sum_q^P \mathbb{E}\left(-\partial_{q,p}^2 \log \pi(Y_n | \boldsymbol{\phi}_n)\right) \partial_i(\boldsymbol{\phi}_n)_p \partial_j(\boldsymbol{\phi}_n)_q.$$

$$= \sum_n^N \sum_p^P \sum_q^P \boldsymbol{G}_{q,p}(\boldsymbol{\phi}_n) \partial_i(\boldsymbol{\phi}_n)_p \partial_j(\boldsymbol{\phi}_n)_q$$

$$= \sum_n^N \sum_p^P \sum_q^P \boldsymbol{G}_{q,p}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)) \partial_i(f_{\boldsymbol{\theta}}(\boldsymbol{x}_n))_p \partial_j(f_{\boldsymbol{\theta}}(\boldsymbol{x}_n))_q,$$

where the expectation taken w.r.t $\boldsymbol{Y}$ is over the Hessian of $\ell_{\boldsymbol{Y}}(\boldsymbol{\theta})$ on $\boldsymbol{\phi}$ (in between 2nd and 3rd step) and that comes for granted once the Fisher metric is known in the basic form of the probabilistic model $\pi(\boldsymbol{y} | \boldsymbol{\phi})$. In the last passage we see that $\partial_i(f_{\boldsymbol{\theta}}(\boldsymbol{x}_n))_p$ is the Euclidean derivative with respect to $\boldsymbol{\theta}_i$ of the pth output of the neural network f evaluated at the nth input. Then using matrix notations we write,

$$\boldsymbol{G}(\boldsymbol{\theta}) = \sum_n^N \boldsymbol{J}_n^\top \boldsymbol{G}_{\boldsymbol{\Phi}}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)) \boldsymbol{J}_n,$$

where $\boldsymbol{J}_n = [\nabla(\boldsymbol{\phi}_n)_1 \cdots \nabla(\boldsymbol{\phi}_n)_P]^\top$ denotes the $P \times D$ Jacobian matrix of the NN at the nth input.

Now, it is clear from the above form that its structure resembles that of a pullback metric in Riemannian manifolds. To see this fix the inputs (or covariates) in the input space and look instead to the output of the NN as a map $\boldsymbol{\Theta} \overset{h_n}{\to} \boldsymbol{\Phi}$ with $h_n(\boldsymbol{\theta}) = f_{\boldsymbol{\theta}}(\boldsymbol{x}_n) \ \forall n$. Once we have endowed a metric $g_{\boldsymbol{\Phi}}$ on $\boldsymbol{\Phi}$ and defined a map for every n, the pullback metric on $\boldsymbol{\Theta}$ given all the inputs can be formally written as the sum of all input information related to $\boldsymbol{\theta}$. That is,

$$g_{\boldsymbol{\Theta}}(\boldsymbol{u}, \boldsymbol{v}) = \sum_n^N g_{\boldsymbol{\Phi}}\left(\mathrm{d}h_n(\boldsymbol{\theta})[\boldsymbol{u}], \mathrm{d}h_n(\boldsymbol{\theta})[\boldsymbol{v}]\right),$$

where $\mathrm{d}h_n(\boldsymbol{\theta})[\cdot] : T_{\boldsymbol{\theta}} \boldsymbol{\Theta} \to T_{\boldsymbol{\phi}_n} \boldsymbol{\Phi}$ is the differential at a point $\boldsymbol{\theta} \ \forall n$. Since $\boldsymbol{\phi}_n = f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)$ can be seen as a function of $\boldsymbol{\theta}$, the differential takes the form $\mathrm{d}h_n(\boldsymbol{\theta})[\boldsymbol{v}] = \boldsymbol{J}_n \boldsymbol{v}$. Plugging this into the expression above and using that the matrix of coefficients of the metric $g_{\boldsymbol{\Phi}}$ is $\boldsymbol{G}_{\boldsymbol{\phi}}$, we obtain the metric tensor above; similar observation can be found in e.g. Grosse (2022). Also note that because the function h_n may not be one-to-one for all n, $\boldsymbol{x}_n \in \mathcal{X}$, the metric tensor $\boldsymbol{G}(\boldsymbol{\theta})$ may not satisfy the definition of a pullback (see Carmo, 1992).

15 EXPERIMENTAL DETAILS

In this section, we describe the computation environment, some implementation details, the employed methods to find the MAP estimate, to obtain the NUTS samples, to integrate the ODEs and to calculate the Wasserstein distances, and describe the basic setups for several experiments.

15.1 Computation environment

Apart from some preliminary ones, all experiments are carried out on the computer cluster using CPUs with type *Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz*. When we benchmark the running times of the algorithms, we always use two cores for each task. The detailed configurations differ based on the nature of the jobs.

15.2 Implementation details

Sacred (Greff et al., 2017) is used to manage the experiments. Code for experiments other than neural network ones are implemented mainly using JAX (Bradbury et al., 2018), benefiting from fast computations of gradients

and other quantities and easy vectorizations. Code for neural network experiments are written mainly in PyTorch (Paszke et al., 2019), using *torch.func* to obtain the gradients and hessian vector products, etc, because we build on the existing library provided by Daxberger et al. (2021). We always use the default data types of the libraries (*float64* in NumPy (Harris et al., 2020) and *float32* in PyTorch and JAX) when initializing the variables. Note that in experiments other than neural network ones, the random numbers are generated by NumPy; In neural network experiments, they are generated by PyTorch.

Bergamin et al. (2023) formed the metric as identity plus the outer product of gradients of negative log-posterior, while Hartmann et al. (2022) formed the metric as identity plus a scaled version of the outer product of gradients of log-posterior. We note that, when the scaling of Monge metric is 1, these are strictly mathematically equivalent formulations. In experiments other than neural network ones, similar to Monge metric (Hartmann et al., 2022), we form the metric as identity plus the outer product of gradients of log-posterior; for neural network experiments, we form the metric similar to Bergamin et al. (2023).

We used SymPy (Meurer et al., 2017) to verify some derivations, and ChatGPT (OpenAI, 2023) as an assistance for writing code, scientific discussions etc.

15.3 Finding the MAP estimate

For experiments other than neural network experiments, unless there is an analytical solution of the MAP estimate, we find the MAP estimate by collecting 20 runs of BFGS optimization with random initialization using the implementation from Virtanen et al. (2020) which yield negative log-posterior values other than plus infinity or NaN, with maximum number of iterations $1e6$. We choose the run that yields the smallest negative log-posterior with *numpy.argmin*.

For neural network experiments, we find the MAP estimate using $lr = 1e - 2$, $weight\_decay = 1e - 5$ for 20000 epochs with the Adam (Kingma and Ba, 2015) optimizer. The prior precision and noise std are optimized posthoc following standard procedure from Daxberger et al. (2021). During the optimization process, we fix the random seeds, in order to have consistent prior precision and noise std for NUTS.

15.4 NUTS samples

We use CmdStanPy (Stan Dev Team, 2023) to generate the NUTS samples. Following the default options for reference posterior in posteriordb (Magnusson et al., 2022), we run the NUTS sampler using 10 chains, a thinning of 10, a warm-up of 10000 iterations and 20000 iterations per chain. We fix the random seeds for reproducibility, generating a total of 20000 samples.

For sampling from banana distribution, based on preliminary runs, we set *adapt_delta* to be 0.95. We otherwise keep the hyper parameters to be their default values.

15.5 ODE integration

For the exponential maps, other than neural network experiments, ODE integrations are carried out using Diffrax (Kidger, 2021). Inspired by the default settings of the function *solve_ivp* in SciPy (Virtanen et al., 2020), we use Dormand-Prince's 5(4) method (Dormand and Prince, 1980). Diffrax reports the number of steps *num_steps* used by the integrator. In order to make the results approximately comparable to the number of function evaluations *nfev* reported by SciPy, we multiply *num_steps* by 6, the number of function evaluations per step of Dormand-Prince's 5(4) method. We use adaptive step size with initial step size chosen by the algorithm, with $rtol = 1e - 3$ and $atol = 1e - 6$, matching the default option of SciPy. We set the maximum number of steps allowed to be 4096 (approximately equivalent to 24576 function evaluations), following the default configuration of Diffrax. For neural network experiments, following Bergamin et al. (2023), we use SciPy's *solve_ivp* and use the default hyper parameters.

For the logarithmic maps, we implement it using analytical gradients and hessian vector products in NumPy.

In all experiments apart from logistic regression (in which we use analytical gradients), we use the gradients and hessian vector products calculated by the autodiff framework.

We always use analytical expressions for the Fisher metric. For logistic regression and NN experiments, we use

analytical expressions of the Christoffel symbols, while in other experiments the Christoffel symbols are solved numerically based on the metric.

When reporting the numbers of function evaluations, we report the numbers of function evaluations required to solve the exponential maps. We observe that in certain scenarios the number of integration steps can exceed the upper limit. In which case, we treat the number of steps for those samples exceeding the limit as the limit 4096 while calculating T, and calculate the Wasserstein distance of the samples not exceeding the limit.

15.6 Calculating the Wasserstein distance

We use Python Optimal Transport (Flamary et al., 2021) to calculate the Wasserstein distance, which can be formulated as (Flamary et al., 2021)

$$\min_{\boldsymbol{\gamma}} \langle \boldsymbol{\gamma}, \boldsymbol{M} \rangle_F,$$
$$s.t. \quad \boldsymbol{\gamma} \mathbf{1} = \boldsymbol{a},$$
$$\boldsymbol{\gamma}^\top \mathbf{1} = \boldsymbol{b},$$
$$\boldsymbol{\gamma} \geq 0,$$

where $\boldsymbol{\gamma}$ is the transportation plan, $\boldsymbol{M}$ is the distance matrix induced by the Euclidean metric for the two sets of samples, and $\boldsymbol{a}$ and $\boldsymbol{b}$ are uniform numbers summing up to 1 with the lengths the same as the numbers of respective samples. The maximum number of iterations is set to be $1e10$.

Wasserstein distance is a proper distance metric, and it becomes zero when the two distributions are the same.

15.7 Squiggle

The experiment presented in Section 4.3.2 was carried out using the following hyper parameters on the squiggle distribution

$$a = 1.5, \ \boldsymbol{S} = \begin{bmatrix} 5.0 & 0.0 \\ 0.0 & 0.05 \end{bmatrix}.$$

15.8 Banana distribution

Results for sampling from banana distribution were presented in Section 5.2. The Fisher metric for banana distribution can be found in Brofos and Lederman (2021), and is given by

$$\boldsymbol{G}(\theta_1, \theta_2) = \begin{bmatrix} \frac{1}{\sigma_\theta^2} + \frac{N}{\sigma_y^2} & \frac{2N\theta_2}{\sigma_y^2} \\ \frac{2N\theta_2}{\sigma_y^2} & \frac{1}{\sigma_\theta^2} + \frac{4N\theta_2^2}{\sigma_y^2} \end{bmatrix}.$$

15.9 Bayesian logistic regression

We report the details on the 5 logistic regression datasets as used in Section 5.3, covering the dimensionality of parameter of interest D and the number of data points N.

Ripley (Ripley, 1994, 1996), abbreviated as *Ripl*: $D = 3$, $N = 250$.

Pima (Smith et al., 1988): $D = 8$, $N = 532$.

Heart (Feng et al., 1992), abbreviated as *Hear*: $D = 14$, $N = 270$.

Australian (Feng et al., 1992), abbreviated as *Aust*: $D = 15$, $N = 690$.

German (Feng et al., 1992), abbreviated as *Germ*: $D = 25$, $N = 1000$.

15.10 Neural networks

In Section 5.4, we performed experiments on NN regression tasks. Similar to Bergamin et al. (2023), we use the dataset from Snelson and Ghahramani (2005), which has 200 data points with labels, with the x coordinates distributed between 0 and 6.
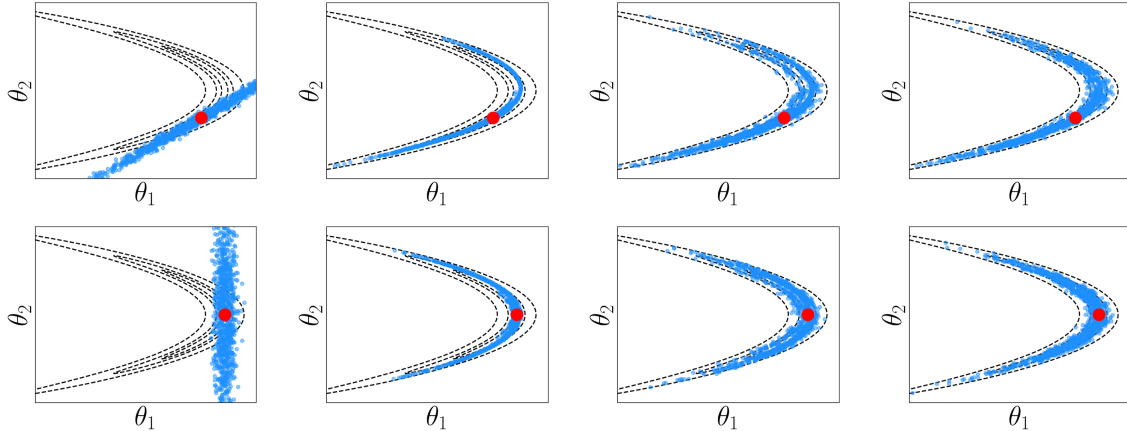
Figure 7: The first row shows approximations starting from one of the Euclidean MAPs, and the second row shows approximations starting from the Hausdorff MAP. Each row from left to right: ELA; RLA-B; RLA-BLog; RLA-F

For *complete* training set, a test set of size 50 is selected from the 200 data points with labels with random seed 1. As mentioned in the main paper, the *gap* training set is constructed by using data points with labels with x coordinates not between 1.5 and 3.0.

16 ADDITIONAL EXPERIMENTAL RESULTS

16.1 Samples from banana distribution

In Section 5.2, we consider the problem of sampling from banana distribution. We reported the numerical qualities of the results in Table 1, and the different MAP estimates and the resulting approximations using RLA-F in Figure 5. Here we provide additional plots that show how all other variants work for the same problem.

In the Banana distribution, there are two Euclidean MAPs, with approximately the same log-posterior and approximately symmetric on the two sides of axis.

The results of starting from one Euclidean MAP and the Hausdorff MAP are shown in Figure 7. We report the results of sampling from one particular Euclidean MAP; starting from the other Euclidean MAP would simply (approximately) mirror the approximation.

Visually, RLA-B, RLA-BLog and RLA-F all adapt to the local curvature. RLA-B is generally narrower, and is slightly in the wrong direction. RLA-BLog alleviates these issues similar to RLA-F, and for all methods the Hausdorff MAP is clearly better as already indicated by the numerical results in the main paper.

16.2 Squiggle

In Section 4.3.2, RLA-B was demonstrated to struggle sampling from a squiggle distribution. Moreover, in the particular runs shown in Figure 4 which generated 10000 samples respectively, RLA-B took on average 75 function evaluations, while RLA-F only took 32.

We note that its properties can vary. Figure 8 shows the results of RLA-B and RLA-F on another version of squiggle, given by

$$a = 1.5, \ \boldsymbol{S} = \begin{bmatrix} 10.0 & 0.0 \\ 0.0 & 0.001 \end{bmatrix}.$$

The high density area is clearly narrower. While in this version of squiggle RLA-B seems to yield better results than the version as shown in the main paper, there still exists a non-negligible mismatch. Moreover, the resulting integration problem seems more difficult than using RLA-F: For this particular run, RLA-F only took 41 function evaluations on average compared to 300 for RLA-B; Moreover, there were at least one sample where RLA-B exceeded the maximum steps, while there were no such issues with RLA-F.
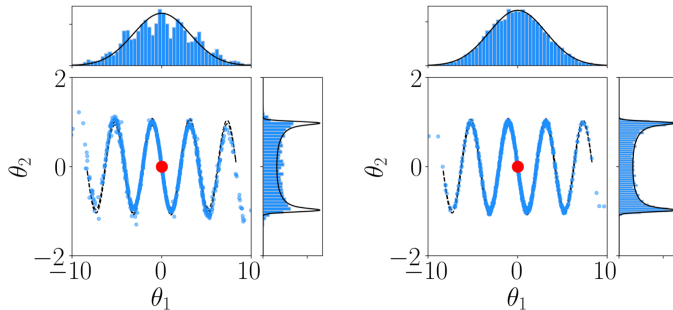
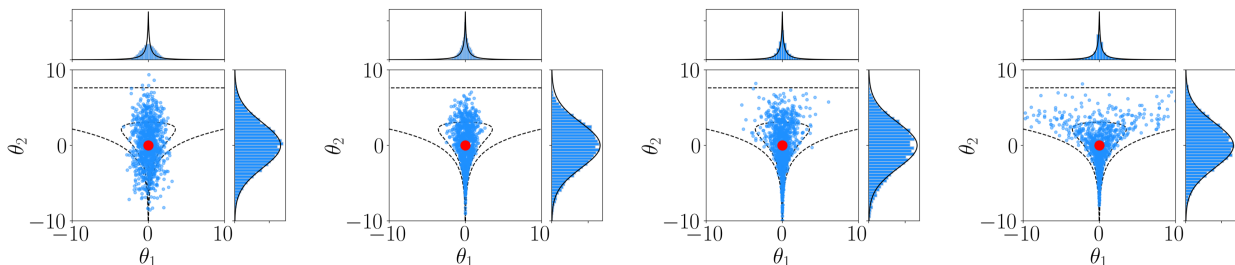Figure 8: Samples from RLA-B (left) and RLA-F (right) on another version of squiggle.

Figure 9: Results on funnel. From left to right: ELA; RLA-B; RLA-BLog; RLA-F.

16.3 Funnel

Here we demonstrate how the approximations work in one more case of a distribution of complex geometry that is diffeomorphic with a Gaussian. We consider the problem of sampling from the funnel distribution (Neal, 2003). It can be formulated as

$$p(\boldsymbol{\theta}) = \mathcal{N}(\theta_D|0, \sigma) \prod_{d=1}^{D-1} \mathcal{N}\left(\theta_d|0, \exp\left(\tfrac{1}{2}\theta_D\right)\right).$$

Consider the two dimensional funnel. It admits a natural reparameterization, and can be formulated as a transformed version of multivariate Gaussian (Stan Development Team, 2023), given by

$$\boldsymbol{\psi} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}),$$
$$\boldsymbol{\theta} = \boldsymbol{\phi}(\boldsymbol{\psi}) = \begin{bmatrix} \exp(\sigma\psi_2/2)\psi_1 \\ \sigma\psi_2 \end{bmatrix}.$$

As such, it is possible to obtain a Riemannian metric, following the general procedure in Section 4.3.2 of the main paper. We have

$$\boldsymbol{\phi}^{-1}(\mathbf{0}) = \mathbf{0},$$
$$\left(\frac{\partial\boldsymbol{\theta}}{\partial\boldsymbol{\psi}}\right)^{-1} = \begin{bmatrix} \exp\left(-\tfrac{1}{2}\theta_2\right) & -\tfrac{1}{2}\theta_1\exp\left(-\tfrac{1}{2}\theta_2\right) \\ 0 & \tfrac{1}{\sigma} \end{bmatrix},$$
$$\boldsymbol{G_\theta} = \left(\frac{\partial\boldsymbol{\theta}}{\partial\boldsymbol{\psi}}\right)^{-\top}\left(\frac{\partial\boldsymbol{\theta}}{\partial\boldsymbol{\psi}}\right)^{-1}.$$

The samples obtained using RLA-F on a two dimensional funnel with $\sigma = 3$ are shown in Figure 9. Interestingly, all variants capture the marginal distribution of θ_2 well even though many MCMC methods struggle to reach the narrow funnel. However, ELA and RLA-B fail to expand in the direction of θ_1 for larger values of θ_2. RLA-BLog alleviates the issue but is still not perfect, but RLA-F is exact.
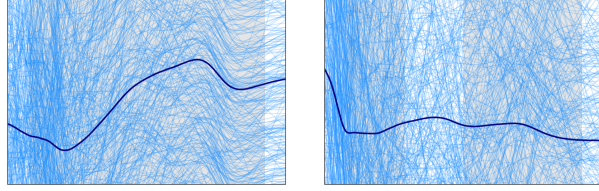
Figure 10: Neural network regression with ELA. Left: complete data; right: gap data.

16.4 Bayesian logistic regression

In Section 5.3, we considered Bayesian logistic regression. We observed that for RLA-B and RLA-BLog, there can be huge differences in terms of number of function evaluations between standardized and raw inputs. We hypothesize that this is in part caused by the large differences in the scales of the raw inputs, and hence the gradient term in Monge metric, which is of quadratic nature, has very different scales across dimensions. This induces strong curvature and leads to challenging numerical integration problems.

We benchmark the time to obtain a single sample based on a sample from Gaussian distribution in seconds of different algorithms and report them in Table 4. We use a slightly optimized version of function for logarithmic map implemented in NumPy and functions for exponential maps are implemented in JAX. We use SciPy for integrations, averaging the obtained times over 2500 samples. RLA-F consistently takes less time to obtain one sample, despite requiring direct matrix inversions.

16.5 Neural networks

In Section 5.4, we presented NN regression results. Here we present further visualizations. Figure 10 explicitly shows how ELA fails to generate meaningful samples for this problem.

16.5.1 Numerical stability

In Figure 6, we purposefully reported runs which resulted in relatively well-behaved predictive distributions for the methods under comparison. RLA-F is numerically stable but the other methods occasionally fail. This is demonstrated in Figures 11 and 12 that show (examples of) bad runs where both RLA-B and RLA-BLog have extremely bad and isolated samples that influence even the mean prediction significantly. For completeness, Figure 13 shows the only run among the 10 independent runs where RLA-F has any issues, in form of two minor outlier samples that do not have notable influence on the whole predictive distribution.

16.5.2 Standardized data

In the neural network regression experiment from the main paper, we directly run the neural network on the original dataset from Snelson and Ghahramani (2005). We later observed that using standardized data may lead to reduced running times for all methods while reducing possible numerical issues, and we report the results in Table 5 and Figure 14. Interestingly, while in the main experiment RLA-B was demonstrated to lead to predictions wider than predictions based on NUTS samples, with standardized data it leads to narrower

data	stand.			raw		
	RLA-B	RLA-BLog	RLA-F	RLA-B	RLA-BLog	RLA-F
Ripl	0.011	0.048	**0.006**	0.011	0.045	**0.006**
Pima	0.016	0.26	**0.006**	1.989	1.183	**0.006**
Hear	0.015	0.638	**0.007**	2.33	2.683	**0.006**
Aust	0.02	0.734	**0.01**	6.183	4.632	**0.008**
Germ	0.026	2.215	**0.014**	1.435	3.4	**0.015**

Table 4: Time per sample in seconds for logistic regression. Bold font indicates the fastest method. Smaller is better.
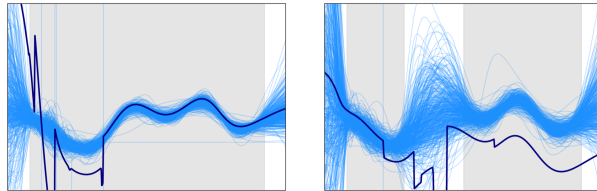
Figure 11: Bad runs with RLA-B on neural networks. The left shows the complete case, and the right shows the gap case.
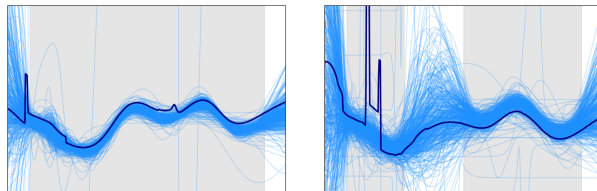

Figure 12: Bad runs with RLA-BLog on neural networks. The left shows the complete case, and the right shows the gap case.

predictions. RLA-BLog reduces the narrowness, while being noisier. RLA-F samples are similar to NUTS samples, while needing significantly fewer function evaluations and being faster in terms of running times.

16.5.3 Scalability experiment

We next consider a larger scale experiment similar to (Bergamin et al., 2023). We generate synthetic regression datasets with varying numbers of data points with the data generation mechanism based on Daxberger et al. (2021), and consider fully connected neural networks of size $[1, N, N, 1]$ with varying N. An example of the generated dataset and the average time to obtain one sample in seconds of RLA-B and RLA-F under different number of data points and different number of parameters are shown in Figure 15. Interestingly, RLA-F can be faster than RLA-B for D up to 1000. With large number of data points, RLA-F becomes slower due to needing per-sample gradients. However, this may be resolved by mini-batching.

16.6 Empirical Fisher

Some works have considered an alternative Riemannian metric known as the Empirical Fisher Information Matrix, given by

$$\sum_{n=1}^{N} \nabla \log \pi(y_n | \boldsymbol{\theta}) \nabla \log \pi(y_n | \boldsymbol{\theta})^\top.$$

There have been arguments that it may not lead to desirable behaviors (Kunstner et al., 2019). Lan et al. (2015) also considers another Empirical Fisher Information Matrix whose estimates are more accurate than the previous

method	Complete				Gap			
	MSE	NLL	T	time	MSE	NLL	T	time
ELA	[0.718, 0.231]	[2.538, 0.041]	N/A	N/A	[1.497, 0.221]	[2.659, 0.032]	N/A	N/A
RLA-B	[0.072, 0.0]	[**0.11**, 0.001]	2065.4	10.4	[0.149, 0.011]	[**0.599**, 0.022]	2109.5	10.6
RLA-BLog	[0.072, 0.0]	[0.121, 0.002]	1428.1	10.2	[0.107, 0.008]	[0.706, 0.028]	1415.3	10.0
RLA-F	[**0.071**, 0.0]	[0.115, 0.001]	**77.7**	**0.3**	[0.122, 0.003]	[0.805, 0.01]	**88.7**	**0.3**
NUTS	[0.073, 0.0]	[0.122, 0.001]	N/A	N/A	[**0.093**, 0.003]	[0.684, 0.017]	N/A	N/A

Table 5: NN regression results with standardized data as [mean, std]. T indicates the average number of function evaluations for one sample while *time* indicates the average time for one sample. Bold font indicates the best method. For all evaluation metrics smaller is better.
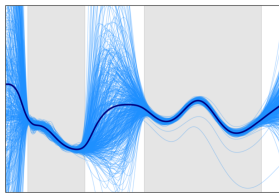
Figure 13: Bad run with RLA-F on neural networks, for the gap case.
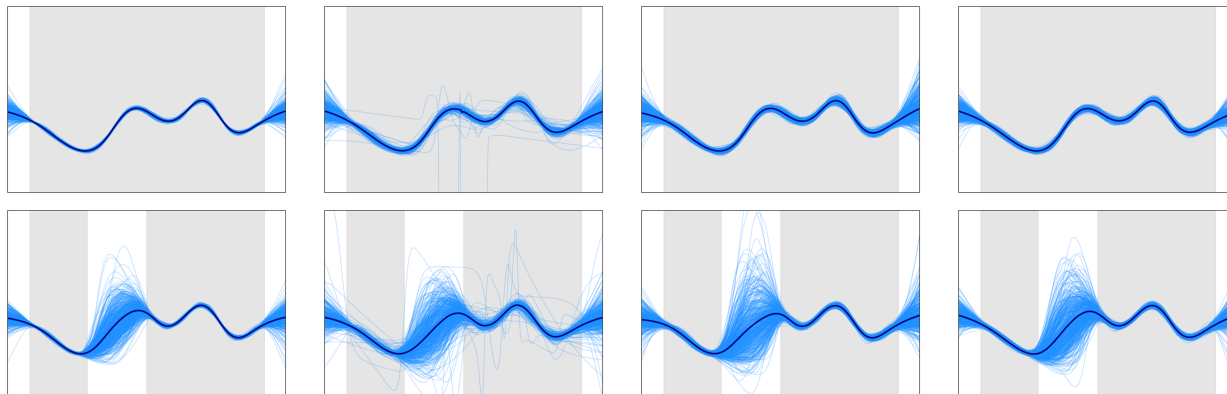

Figure 14: NN regression with complete (top) and gap (bottom) standardized training data. Methods from left to right: RLA-B; RLA-BLog; RLA-F; NUTS. Gray shading denotes the part of x-axis with training data, dark line is the mean prediction, and blue lines are samples.

ones. We tried replacing the FIM with the above empirical version in the Fisher metric, terming the resulting metric as Empirical Fisher metric, and performed a range of preliminary experiments to evaluate whether the Empirical Fisher could also be used inside the metric in RLA, always using numerical Christoffel symbols.

While it works reasonably well for sampling from banana distribution, yielding a Wasserstein distance of $[0.799, 0.008]$ with Euclidean MAP and $[0.141, 0.003]$ with Hausdorff MAP (reported as $[\text{mean}, \text{std}]$), it becomes much worse for Bayesian logistic regression, as shown in Table 6. Especially, for the experiment on Heart dataset without standardization, it is much worse than Euclidean. We therefore did not explore it further.

	data	$\mathcal{W}$	T
stand.	Ripl	[0.106, 0.004]	12.0
	Pima	[0.148, 0.0]	12.1
	Hear	[0.53, 0.0]	13.8
raw	Ripl	[0.426, 0.011]	12.1
	Pima	[0.199, 0.001]	12.5
	Hear	[0.942, 0.02]	17.0

Table 6: Logistic regression results with Empirical Fisher metric, reported as $[\text{mean}, \text{std}]$. $\mathcal{W}$ indicates Wasserstein distance to NUTS samples while T indicates the average number of function evaluations for one sample. For all evaluation metrics smaller is better.
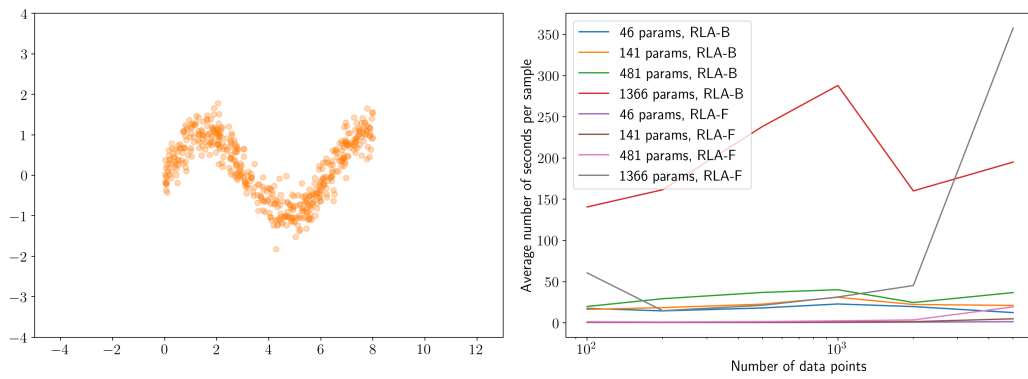
Figure 15: Scalability experiment. Left: An example dataset. Right: Running times of RLA-B and RLA-F under different settings.