Fast and Robust Shortest Paths on Manifolds Learned from Data — Supplementary Material —

Georgios Arvanitidis[†] **Søren Hauberg**[†] **Philipp Hennig**[‡] **Michael Schober**[⋆]

[†]Technical University of Denmark, Lyngby, Denmark
[‡]University of Tübingen, Tübingen, Germany
[‡]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[⋆]Bosch Center for Artificial Intelligence, Renningen, Germany

A Approximate Shortest Paths

The propose approximation to the shortest path is the posterior mean of a Gaussian process, and is parametrized by a set of second derivatives $\ddot{\mathbf{z}}_n$ on a discrete mesh $\Delta = \{t_0 = 0, t_1, \dots, t_{N-1} = 1\} \subset [0, 1]$ of evaluation knots t_n. Therefore, the shortest path is

$$\begin{aligned}
\boldsymbol{\mu}(t) &= \mathbf{m}(t) + \boldsymbol{\omega}^\intercal \mathrm{vec}\left(\begin{bmatrix} \mathbf{x} - \mathbf{m}(0) \\ \mathbf{y} - \mathbf{m}(1) \\ \ddot{\mathbf{z}}_\Delta - \ddot{\mathbf{m}}(\Delta) \end{bmatrix}^\intercal\right) \\
\mathbf{G} &= \mathbf{V} \otimes \left(\begin{bmatrix} k(\mathcal{B}, \mathcal{B}) & \frac{\partial^2}{\partial s^2} k(\mathcal{B}, \Delta) \\ \frac{\partial^2}{\partial t^2} k(\Delta, \mathcal{B}) & \frac{\partial^4}{\partial t^2 \partial s^2} k(\Delta, \Delta) \end{bmatrix} \right. \\
&\qquad \left. + \mathrm{diag}(0, 0, \boldsymbol{\Sigma}, \dots, \boldsymbol{\Sigma}) \right) \\
\boldsymbol{\omega}^\intercal &= \left(\mathbf{V} \otimes \begin{bmatrix} k(t, \mathcal{B}) & \frac{\partial^2}{\partial s^2} k(t, \Delta) \end{bmatrix}\right) \mathbf{G}^{-1}.
\end{aligned} \tag{1}$$

A fixed-point scheme to learn the parameters $\ddot{\mathbf{z}}_\Delta$ that satisfy the ODE of the geodesic curve is presented in Arvanitidis et al. (2019). Next we show how the components of the GP can be chosen.

Mean function

The most natural choice regarding the mean function of the prior is the straight line that connects the two boundary points $\mathbf{m}(t) = \mathbf{c}(0) + t \cdot (\mathbf{c}(1) - \mathbf{c}(0))$. This is the shortest path when the Riemannian manifold is flat. Also, when the curvature of the manifold is low, then the shortest path will be relatively close to the straight line. Likewise, when two points are very close on the manifold. Note that the mean function of the prior is the initial guess of the BVP solution.

For instance, if for the kernel we chose the SE, then implicitly the prior assumption is that the shortest paths are smooth curves varying on a length scale of λ along t. Also, the amplitude $\mathbf{V} = [(\mathbf{a} - \mathbf{b})^\intercal \mathbf{S}_\mathbf{x} (\mathbf{a} - \mathbf{b})] \cdot \mathbf{S}_\mathbf{x} \in \mathbb{R}^{D \times D}$, where $\mathbf{S}_\mathbf{x}$ is the sample covariance of the dataset $\mathbf{x}_{1:N}$ as in Hennig and Hauberg (2014).

Kernel

The kernel type implies the smoothness of the approximated curve. Since shortest paths are expected to be relatively smooth as two times differentiable parametric functions, a reasonable choice for the kernel is to be smooth, e.g. squared exponential (SE), Matern, etc.

Moreover, it is important to use *stationary* kernels, since they treat the two boundary points equally. For example, the non-stationary Wienner kernel is a common choice for IVPs. However, in a BVP such a kernel is a poor fit, because if the time interval is inverted, then the resulting curve will be different.

Mesh

The *Reproducing Kernel Hilbert Space (RKHS)* (Rasmussen and Williams, 2006) of the Gaussian process is spanned by the basis functions $\{k(t_n, t)\}_{n=0}^{N-1}$. The predictive posterior $\boldsymbol{\mu}(t)$ lies in the RKHS as a linear combination of the basis functions $k(t_n, t)$. Therefore, for our approximation to work, we need the true shortest path to be approximated sufficiently well by the RKHS. This, means that the $\boldsymbol{\mu}(t)$ has to be a smooth approximation to the true path.

In our case, the mesh Δ specifies the locations, as well as the number of the basis functions. Consequently, by increasing the size of mesh, we essentially increase the RKHS such that to be able to approximate more complicated true shortest paths. However, knowing in prior the correct number and the placements of the knots is unrealistic. For that reason a reasonable solution is to use a uniform grid for the interval $[0, 1]$. Moreover, Δ can be seen as a common hyper-parameter for every choice of kernel.

Hyper-parameters

The hyper-parameters of each kernel are kept fixed, because learning the hyper-parameters in parallel with the artificial dataset $\ddot{\mathbf{z}}_\Delta$ may lead to degenerate solutions.

N	5	10	15	25	50	100
#1	2.52(± 0.4693)	2.51(± 0.3296)	2.51(± 0.1562)	2.49(± 0.1476)	2.47(± 0.0043)	2.47(± 0.0004)
#2	2.36(± 0.4541)	2.33(± 0.1800)	2.34(± 0.2426)	2.32(± 0.1162)	2.32(± 0.0011)	2.32(± 0.0004)
#3	2.20(± 0.5315)	2.19(± 0.1653)	2.18(± 0.0742)	2.17(± 0.0559)	2.17(± 0.0017)	2.17(± 0.0004)
#4	2.17(± 0.5496)	2.16(± 0.1972)	2.15(± 0.1028)	2.15(± 0.0417)	2.14(± 0.0020)	2.14(± 0.0003)

Table 1: Example for constant speed curves.

B Scaling of the algorithm with respect to mesh and dimensions

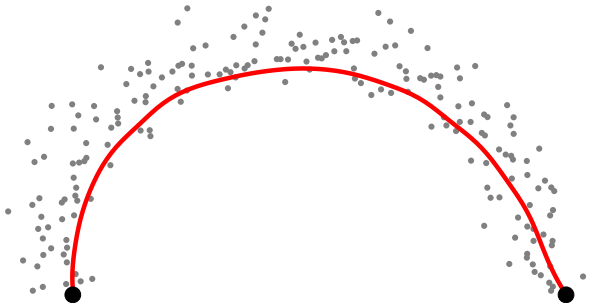

Figure 1: Example of a shortest path.

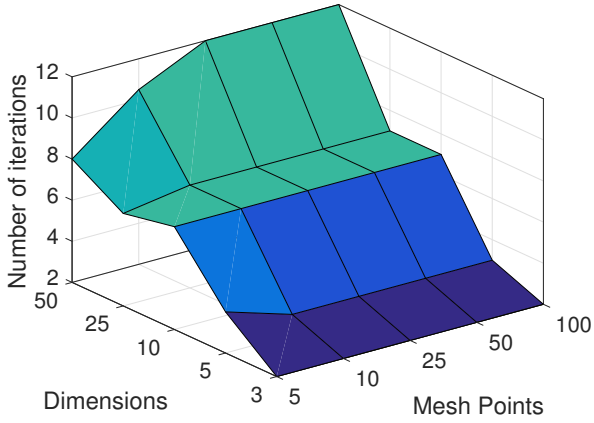

Figure 2: Scaling of the algorithm.

The curvature of the Riemannian manifold $\mathcal{M}$ i.e., the behavior of the learned metric, implies the complexity of the shortest paths. As regards the iterations that the algorithm needs in order to find the parameters which solve the ODE, these are related to the ability of the RKHS to approximate the true shortest path. In other words, when the true shortest path can be approximated easily by the RKHS, then the only few fixed point iterations are utilized.

For instance, in Fig. 1 we show a challenging shortest path for a non-parametric metric with $\sigma_{\mathcal{M}} = 0.1$, which means that the curvature is high. When $N = 10$ the RKHS is not large enough to approximate easily the true path, so 300 iterations are needed in order for the algorithm to converge. When we increase the $N = 50$ the true path can be smoothly approximated easier by the enlarged RKHS, so that only 80 fixed point iterations are needed. When we increase the $\sigma_{\mathcal{M}} = 0.15$ the curvature of $\mathcal{M}$ reduces, so now 85 and 32 iterations are needed, respectively.

For completeness, we test how the method scales to higher dimensions as well. In this dataset, we fix a random point as the base point, and a subset of 100 points. We fix the $\sigma_{\mathcal{M}} = 0.25$ and we chose the dimensions $[3, 5, 10, 25, 50]$ and the mesh sizes $[5, 10, 25, 50, 100]$. Then, we map the 2-dimensional dataset into each dimension using an orthogonal map, we add noise $\mathcal{N}(0, 0.01)$ and we compute all the shortest paths between the subset and the base point for different mesh sizes. As we see in Fig. 2, the scaling is sublinear as regards the mesh size. Of course, as the dimension increases the problem becomes more complex, so more iterations are needed. Note that the $\mathcal{M}$ does not have high curvature, which means that the true shortest path can be approximated relatively easy by each RKHS.

C Constant Speed Curves

The exact definition of the *geodesic* is that, it is a locally minimizing curve with constant speed. This means that the geodesic might be not the global shortest path, but any segment on the geodesic curve is minimizing the length locally. However, it is important that the geodesic has constant speed. Also, by definition a curve that satisfies the ODE has constant speed.

Here we test how the mesh size N affects the speed of the resulting curve. In Table 1 we show the mean and the standard deviation of the speed for 4 curves in the data manifold of Fig. 1. The results show that when the mesh increases, the speed becomes more constant since the standard deviation decreases. Instead, for small N the curve satisfies the ODE only at the knots t_n, however, it does not have the exact dynamics of the true curve. In other words, with only N points we are not always capable to approximate exactly the true curve. This means that our solution is a smooth approximation of the true curve, but it is not able to have constant speed. As we increase the N the RKHS can approximate exactly the true curve, which satisfies the ODE for every t, and thus, it has constant speed.

D Robustness of the Solver

We conducted an experiment to test the robustness of our solver. In particular, we computed a challenging shortest path in the latent space of the deterministic generator $f(x, y) = [x, y, x^2 + y^2]$. In Fig. 3 we show the paths found by `bvp5c`, our method when initialized with the straight line and when it is initialized by the `bvp5c`'s solution. Obviously, the `bvp5c` converges to a suboptimal solution, while our method manages to find the true shortest path when initialized with the straight line. Interestingly, due to its robustness our solver manages to find a geodesic even by initializing it with the suboptimal solution of `bvp5c`. Of course, this is not the shortest path but it is a geodesic, because it has constant speed as it satisfies the ODE $\forall t$, and also, it is locally length minimizing.

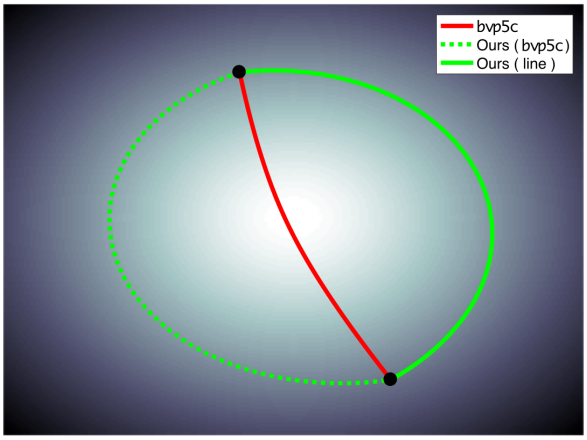

Figure 3: Example of robustness.

E Downstream Tasks

We also compared the performance of our solver on downstream tasks.

From the LAND experiment (Arvanitidis et al., 2019, Sec. 4.1) we clustered the data using the trained mixture models and a linear model, and we get the errors: 0% (ours), 15% (bvp5c), 21% (linear). We also numerically measure the KL divergence between the learned distributions and the generating distribution, and observe that the proposed solver improves the fit: 0.35 (ours), 0.65 (bvp5c), 0.53 (linear).

Additionally, we performed k-means clustering on a 2-dimensional latent space of a VAE trained on MNIST digits 0,1,2 and the resulting errors: 92($\pm$5)% (ours, 1.6($\pm$0.7) hours), 91($\pm$5)% (bvp5c, 8($\pm$4.5) hours), 83($\pm$4)% (linear). The proposed model is, thus, both faster and more accurate on downstream tasks.

References

G. Arvanitidis, S. Hauberg, P. Hennig, and M. Schober. Fast and robust shortest paths on manifolds learned from data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

P. Hennig and S. Hauberg. Probabilistic Solutions to Differential Equations and their Application to Riemannian Statistics. In *Proc. of the 17th int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, volume 33, 2014.

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT, 2006.