# Riemannian Diffusion Models

Gabriela Moisescu-Pareja, Roland Riachi

November 11, 2022

# The Framework

- Let $(\mathcal{M}, g)$ be a Riemannian manifold
- The *generative process* $X_t \in \mathcal{M}$ is a stochastic process satisfying

$$\mathrm{d}X_t = V_0(X_t, t)\mathrm{d}t + V(X_t, t) \circ \mathrm{d}B_t, \qquad X_0 \sim p_0 \tag{1}$$

- $V_0$ and the columns of the diffusion matrix $V := [V_1, \ldots, V_w]$ are smooth vector fields on $\mathcal{M}$
- $B_t$ is a $w$-dimensional Brownian motion on $\mathcal{M}$

# SDEs in Euclidean Space (1/2)

## Brownian Motion

A *Brownian motion* is a stochastic process such that

- (Start at Zero) $B_0 = 0$ a.s.
- (Normally Distributed) $B_t \sim \mathcal{N}(0, t)$
- (Independent increments) $B_{t-r}$ is independent of $B_r$ for every $0 \leq r < t$.
- (Continuity) The sample paths are a.s. continuous

## SDEs

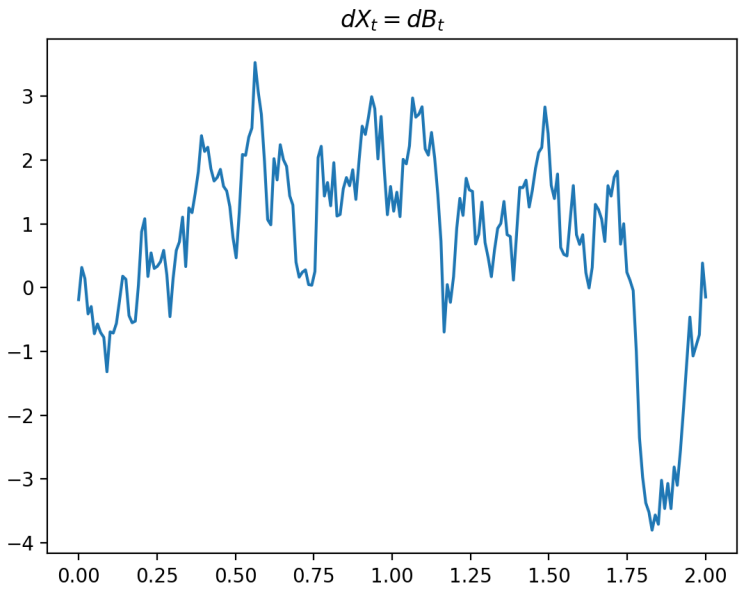
Let $\mu : \mathbb{R}^n \times (0, \infty) \to \mathbb{R}^n$ and $\sigma : \mathbb{R}^n \times (0, \infty) \to \mathbb{R}^{m \times n}$, we deal with SDEs of the form

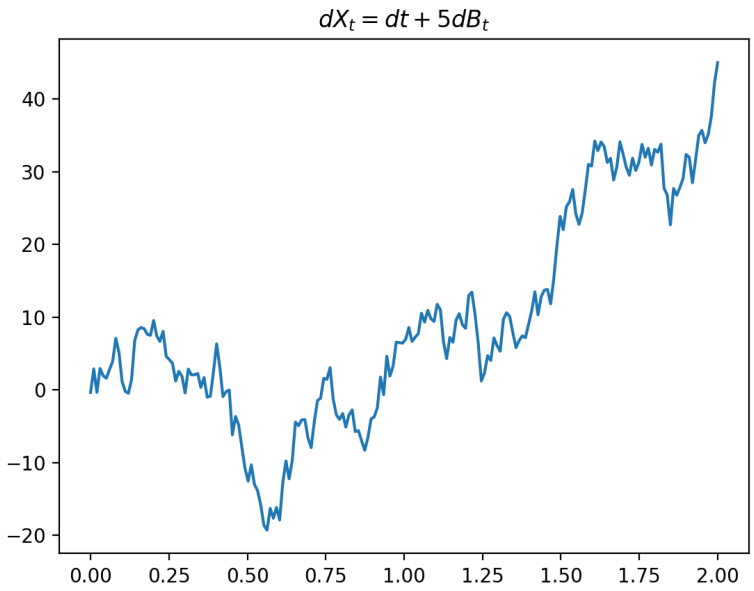
$$\mathrm{d}X_t = \mu(X_t, t)\mathrm{d}t + \sigma(X_t, t) \circ \mathrm{d}B_t$$

# SDEs on Euclidean Space

- Computationally, think of $X_t$ as a Markov chain:

$$X_{n\Delta t} = X_{(n-1)\Delta t} + \mu(X_{(n-1)\Delta t}, t)\Delta t + \sigma(X_{(n-1)\Delta t}, t)\mathcal{N}(0, \Delta t).$$



# SDEs on Riemannian Manifolds

- On a manifold $(\mathcal{M}, g)$, we write

$$\mathrm{d}X_t = V_0 \mathrm{d}t + \sum_k V_k(X_t) \circ \mathrm{d}B_t$$

if

$$\mathrm{d}f(X_t) = V_0(f)(X_t, t)\mathrm{d}t = \sum_k V_k(f)(X_t, t) \circ \mathrm{d}B_t$$

for all $f \in C^\infty(\mathcal{M})$.

- Computationally: do SDE in tangent space to get a vector and use exponential map to get the next point in the Markov chain.

```python
# noinspection PyMethodMayBeStatic,PyUnusedLocal
class CircularBrownianMotion(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.mani_name = "Circle"

    def f(self, y, t):
        return torch.zeros_like(y)

    def g(self, y, t):
        return torch.ones(list(y.shape))

    def sample(self, y0, t):
        """
        sample yt | y0
        """
        y = (y0 + torch.randn_like(y0) * t ** 0.5) % (2 * np.pi)
        return y


class CircularGenerative(torch.nn.Module):
    def __init__(self, sigma, a, T=0.2):
        # TODO: remove T, and integrate `sigma` & `a` into a single drift func
        super().__init__()
        self.T = T
        self.sigma = sigma
        self.a = a
        self.mani_name = "Circle"

    def mu(self, x, t):
        sa = self.sigma(x, self.T - t) * self.a(x, self.T - t)
        return sa

    def sample(self, x0, T, steps):
        """
        sample xt | x0
        """
        dt = T / steps
        x = x0
        for s in range(steps):
```

In one spatial dimension $x$, for an Itô process driven by the standard Wiener process $W_t$ and described by the stochastic differenti

$$dX_t = \mu(X_t, t)\,dt + \sigma(X_t, t)\,dW_t$$

with drift $\mu(X_t, t)$ and diffusion coefficient $D(X_t, t) = \sigma^2(X_t, t)/2$, the Fokker–Planck equation for the probability density $p(x,$

$$\frac{\partial}{\partial t}p(x,t) = -\frac{\partial}{\partial x}\left[\mu(x,t)p(x,t)\right] + \frac{\partial^2}{\partial x^2}\left[D(x,t)p(x,t)\right].$$

Consider the partial differential equation

$$\frac{\partial u}{\partial t}(x,t) + \mu(x,t)\frac{\partial u}{\partial x}(x,t) + \tfrac{1}{2}\sigma^2(x,t)\frac{\partial^2 u}{\partial x^2}(x,t) - V(x,t)u(x,t) + f(x,t) = 0,$$

defined for all $x \in \mathbb{R}$ and $t \in [0, T]$, subject to the terminal condition

$$u(x, T) = \psi(x),$$

where μ, σ, ψ, $V$, $f$ are known functions, $T$ is a parameter and $u : \mathbb{R} \times [0, T] \to \mathbb{R}$ is the unkn

$$u(x,t) = E^Q\left[\int_t^T e^{-\int_t^T V(X_\tau,\tau)\,d\tau} f(X_r, r)dr + e^{-\int_t^T V(X_\tau,\tau)\,d\tau}\psi(X_T)\,\middle|\,X_t = x\right]$$

*where $T \leq \infty$ is a given constant and $B(t)$ is $n$-dimensional Brownian motion. Put*

$$M_t = \exp\left(-\int_0^t a(s,\omega)dB_s - \tfrac{1}{2}\int_0^t a^2(s,\omega)ds\right); \qquad t \leq T\,. \qquad (8.6.6)$$

# Continuous-Time ELBO (CT-ELBO) (1/2)

- Assuming $\mathrm{d}X_t = V_0(X_t, t)\mathrm{d}t + V(X_t, t) \circ \mathrm{d}B_t$ and $X_0 \sim p_0$ - all data comes from distribution $p(\cdot, T)$ for some $T > 0$.
- If we train model to learn the SDE, how do we talk about $p(\cdot, T)$?
- Answer: Probability Theorems

# CT-ELBO (1/2)

## CT-ELBO

The *inference process* $Y_t \in \mathcal{M}$ is a stochastic process satisfying

$$\mathrm{d}Y_s = (-V_0 + (V \cdot \nabla_g)V + Va)\mathrm{d}s + V \circ \mathrm{d}\hat{B}_s, \quad Y_0 \sim p(\cdot, T), \tag{2}$$

and which gives

$$\log p(x, T) \geq \mathbb{E}\left[\log p_0(Y_T) - \int_0^T \frac{1}{2}\|a(Y_s, s)\|_2^2 + \nabla_g \cdot (V_0 - \frac{1}{2}(V \cdot \nabla_g)V)\mathrm{d}s \middle| Y_0 = x\right].$$

# Computing the Riemannian divergence

- In the Euclidean setting, computing the CT-ELBO requires computing the divergence.
- In the Riemannian case, we must similarly compute the divergence of a vector field.

# Computing the Riemannian divergence

Possible suggestions to compute $\nabla_g$:

- Intrinsic coordinates: This method is not great because obtaining local coordinates might be difficult in some cases and the inverse scaling of $\sqrt{|G|}$ might result in numerical instability and high variance.
- Closest-point projection: Can be okay, but expensive for certain manifolds (e.g. hyperbolic) since you have to solve an optimization problem (involving the Euclidean norm). Imagine doing this optimization for each step of the Markov chain.

# Computing the Riemannian divergence

- QR Decomposition: As an alternative to the closest-point projection, search for an orthogonal basis for the tangent space. For low dimensions, this method is efficient.
- Projected Hutchinson: When QR is too expensive, approximate the trace in the expression of the Riemannian divergence identity (when the manifold is embedded in an ambient space) using an unbiased trace approximator. This involves projecting normally distributed vectors onto the tangent space.

# Fixed-inference Parameterization (1/n)

- The process by which you add noise is a design choice.
- Of course, this choice will have varying consequences.

# Fixed-inference Parameterization (2/n)

- A suggested design for the inference SDE is based on *Riemannian Langevin Dynamics.*
- This is the noise process which takes observed data to unstructured noise, and is defined by

$$dY = U_0 dt + V \circ d\hat{B}_s$$

  where the drift and diffusion vector fields are fixed to be $U_0 = \frac{1}{2}\nabla_g \log p_0$ and $V$ as the tangential projection matrix, respectively.
- This has several nice consequences:

# Fixed-inference Parameterization (3/n)

**Stable and efficient training.**

- No longer need to optimize the vector fields that generate $Y_s$ since we have explicit expressions instead of parameterized neural networks (which we'd need to optimize).
- Before, to sample $Y_s$ you had to do a forward pass of the NN. For every step in the MC, you'd have to run through the NN which is expensive (ideally want to only do a forward pass at test time). This is only a sample, but it's required for the ELBO!
- Instead, to sample you can pass the SDE through a numerical integrator to get a sample path since there is a closed-form expressions. Much cheaper.

# Fixed-inference Parameterization (4/n)

**Simplified Riemannian CT-ELBO.**

- Since $V$ is the tangential operator, applying the differential operator $V \cdot \nabla_g$ to it is zero: $(V \cdot \nabla_g)V = 0$, and this yields a simplified CT-ELBO

$$\mathbb{E}[\log p_0(Y_T)] - \int_0^T \mathbb{E}_{Y_s}\left[\frac{1}{2}\|a\|_2^2 + \nabla_g \cdot (Va - U_0)\middle| Y_0 = x\right] ds.$$

  Here $V$ and $U_0$ are no longer neural networks.

# Fixed-inference Parameterization (4/n)

**Variance reduction.**

- It's possible to estimate the time integral of the (simplified) CT-ELBO via importance sampling.
- This avoids further complicating the parameterization of the model.

# Fixed-inference Parameterization (5/n)

**Connection to score-matching.**

- In Euclidean case, it's known that maximizing the variational lower bound of the fixed-inference diffusion model is equivalent to score matching.
- This is shown to be the case as well for Riemannian diffusion using the fixed-inference parameterization!

# Experiments

- Experiments on spherical manifolds modeling natural disaster data in Earth science, on tori (product of spherical manifolds) modeling protein and RNA data, on hyperbolic manifolds modeling synthetic data and on orthogonal groups.
- Notably, the methods used to compute the divergence (and taking the extrinsic view) allow modeling data on non-compact manifolds such as hyperbolic manifolds.
- Given the fixed inference framework, they simply use an MLP to parameterize $a$ in the simplified CT-ELBO.

# Experiments: Sphere

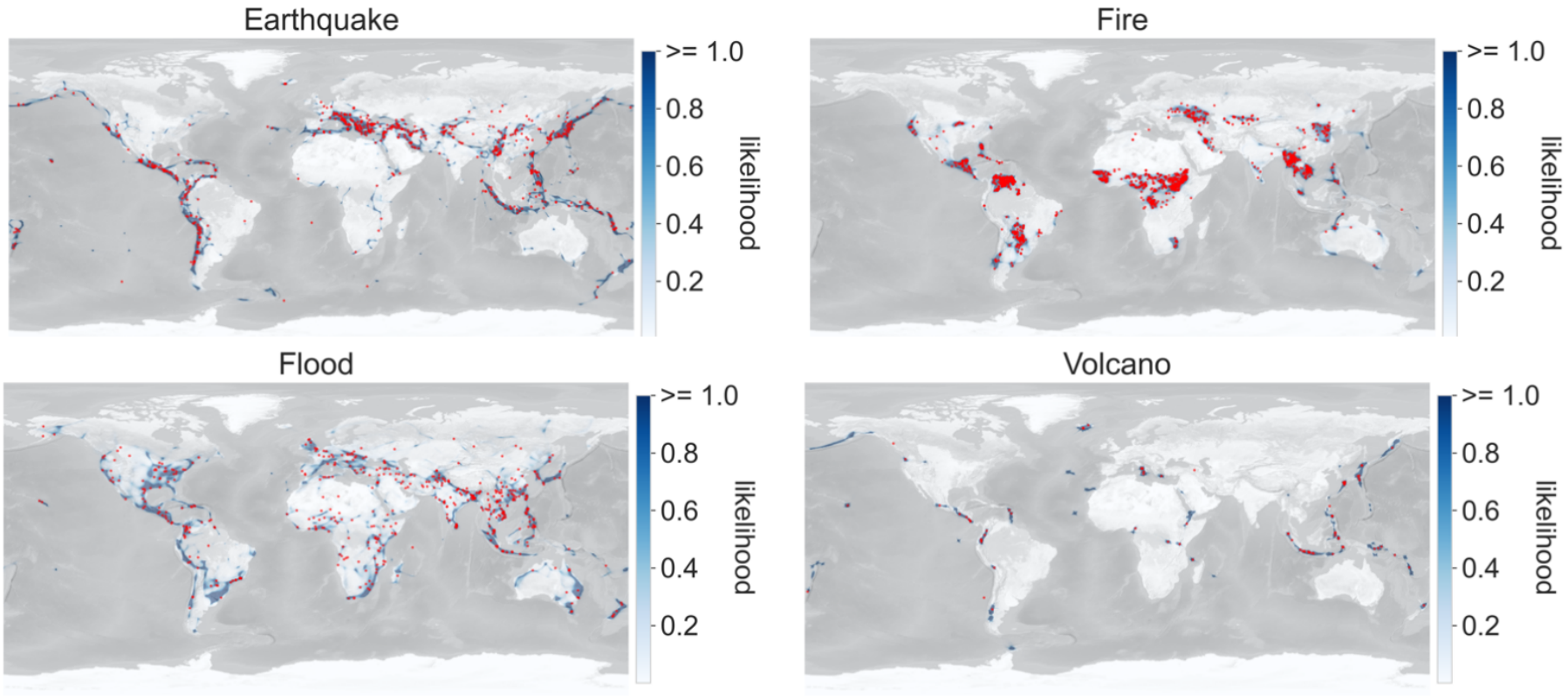



Figure 1: Density of models trained on earth datasets. Red dots are samples from the test set.

# Experiments: Sphere

| | Volcano | Earthquake | Flood | Fire |
|---|---|---|---|---|
| Mixture of Kent | $-0.80_{\pm 0.47}$ | $0.33_{\pm 0.05}$ | $0.73_{\pm 0.07}$ | $-1.18_{\pm 0.06}$ |
| Riemannian CNF (Mathieu & Nickel, 2020) | $-0.97_{\pm 0.15}$ | $0.19_{\pm 0.04}$ | $0.90_{\pm 0.03}$ | $-0.66_{\pm 0.05}$ |
| Moser Flow (Rozen et al., 2021) | $-2.02_{\pm 0.42}$ | $-0.09_{\pm 0.02}$ | $0.62_{\pm 0.04}$ | $-1.03_{\pm 0.03}$ |
| Stereographic Score-Based | $-4.18_{\pm 0.30}$ | $-0.04_{\pm 0.11}$ | $1.31_{\pm 0.16}$ | $0.28_{\pm 0.20}$ |
| Riemannian Score-Based (De Bortoli et al., 2022) | $-5.56_{\pm 0.26}$ | $-0.21_{\pm 0.03}$ | $0.52_{\pm 0.02}$ | $-1.24_{\pm 0.07}$ |
| RDM | $\mathbf{-6.61}_{\pm 0.97}$ | $\mathbf{-0.40}_{\pm 0.05}$ | $\mathbf{0.43}_{\pm 0.07}$ | $\mathbf{-1.38}_{\pm 0.05}$ |
| Dataset size | 827 | 6120 | 4875 | 12809 |

Table 1: NLL scores for each method on earth datasets. Bold shows best results (up to statistical significance). Means and standard deviations are calculated over 5 runs. Baselines taken from De Bortoli et al. (2022).

# Experiments: Variance reduction and efficiency.

- Figure 2 shows that for the spherical dataset, variance reduction is achieved with importance sampling, and training is sped up.
- Figure 3 shows for the tori dataset the results when estimating the simplified CT-ELBO. Tori allow direct sampling of Brownian motion without numerical integration, which is used as a reference. Numerical integration: dividing into few timesteps tends to underestimate loss, but for 100 steps, it resembles the model trained with direct sampling.

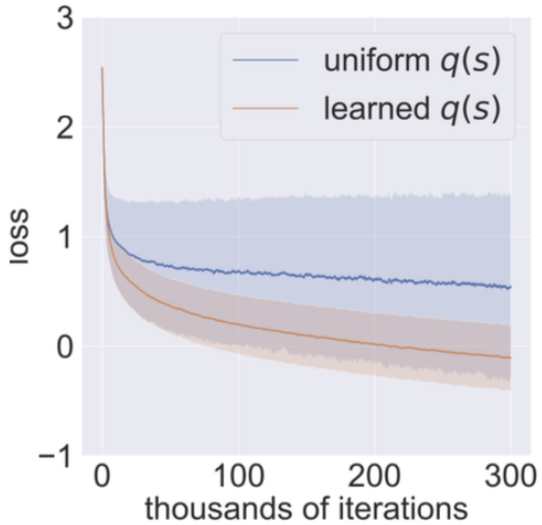
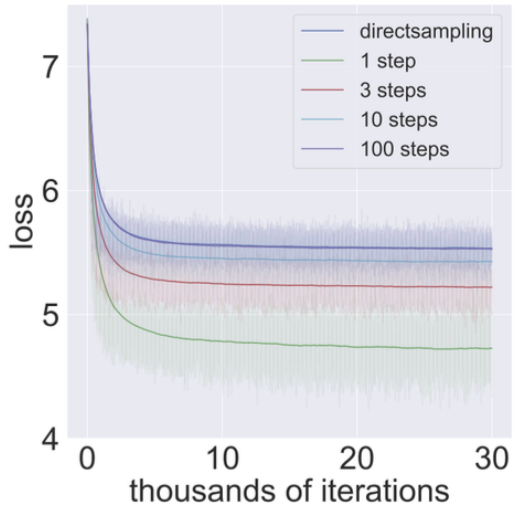



Figure 2: Variance reduction with importance sampling.



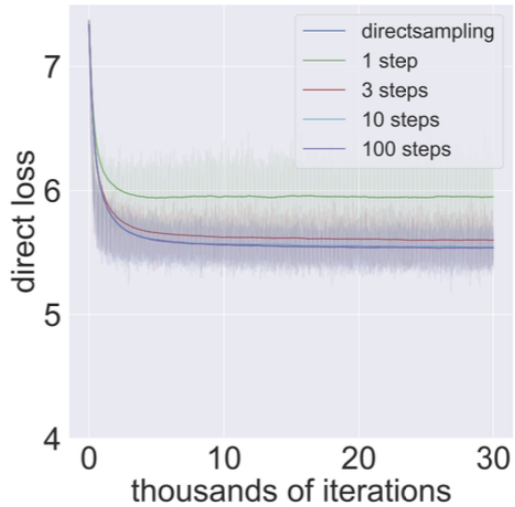




Figure 3: Direct sampling vs numerical integration of Brownian motion. Numbers in legends indicate the number of time steps.

# Experiments: Tori

| | General | Glycine | Proline | Pre-Pro | RNA |
|---|---|---|---|---|---|
| MoPS | $1.15_{\pm 0.002}$ | $2.08_{\pm 0.009}$ | $0.27_{\pm 0.008}$ | $1.34_{\pm 0.019}$ | $4.08_{\pm 0.368}$ |
| RDM | $\mathbf{1.04}_{\pm 0.012}$ | $\mathbf{1.97}_{\pm 0.012}$ | $\mathbf{0.12}_{\pm 0.011}$ | $\mathbf{1.24}_{\pm 0.004}$ | $\mathbf{-3.70}_{\pm 0.592}$ |
| Dataset size | 138208 | 13283 | 7634 | 6910 | 9478 |

Table 2: Negative test log-likelihood for each method on Tori datasets. Bold shows best results (up to statistical significance). Means and standard deviations are calculated over 5 runs.

# Conclusion

- Extended continuous-time diffusion models to arbitrary Riemannian manifolds.
- Provided a variational framework to train RDMs by optimizing a novel objective (Riemanninan CT-ELBO).
- Theoretically, through the construction of marginally equivalent SDEs, show connections between their proposed variational framework and Riemannian score-matching.
- Practically, provided methods to enable efficient and stable training: fixed-inference parameterization of the SDE in the ambient space, new techniques to compute the Riemannian divergence and an importance sampling technique wrt the time integral to reduce variance in the loss.
- Implemented RDMs that show SOTA performance on various datasets.