Scalable Gradients for Stochastic Differential Equations

Xuechen Li*
Google Research

Ting-Kam Leonard Wong
University of Toronto

Ricky T. Q. Chen
University of Toronto
Vector Institute

David Duvenaud
University of Toronto
Vector Institute

Published at AISTAT 2020

Material form books/paper from Simo Sarkka/Aron Solin, slides from xuecheng Li

Content

- Recap of SDE
- Recap of Neural ODE and adjoint method
- Stochastic adjoint method
- Latent-SDE and Bayesian learning(VI)
- Other

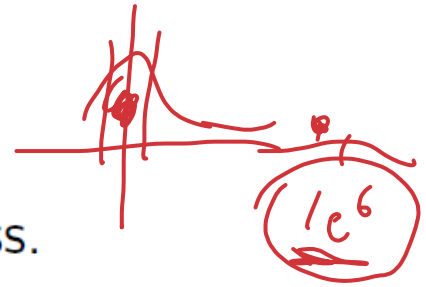
Stochastic Differential Equations

$$\frac{dz_t}{dt} = b(z_t, t) + \sigma(z_t, t)\, w_t$$

- *Stochastic differential equations* (SDEs) generalize ODEs to incorporate a noise component driven by a Brownian motion

$$Z_T = Z_0 + \int_0^T \underbrace{b(Z_t, t)}_{\text{drift}}\, \mathrm{d}t + \overbrace{\int_0^T \underbrace{\sigma(Z_t, t)}_{\text{diffusion}}\, \mathrm{d}W_t}^{\text{Itô integral}}.$$

dW_t: White noise (a random sample from zero-mean Gauss)



- $b : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and $\sigma : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^{d \times m}$ are the drift and diffusion coefficients, respectively.
- W_t is a (multi-dimensional) Brownian motion/Wiener process.
- SDEs are widely used in mathematical finance, population genetics, stochastic optimal control, etc.

Lebesgue–Stieltjes Integral vs Itô Integral

Lebesgue–Stieltjes

$$\int_a^b f(t)\,\mathrm{d}g(t) = \lim_{|\Pi|\to 0} \sum f(t_i^*)(g(t_{i+1}) - g(t_i))$$

- $t_i^* \in [t_i, t_{i+1})$
- $g(t)$ of bounded variation
- Obeys usual integration by parts rule
- Convergence is in usual metric on $\mathbb{R}$

Itô

$$\int_a^b f(t)\,\mathrm{d}W_t = \lim_{|\Pi|\to 0} \sum f(t_i^*)(W_{t_{i+1}} - W_{t_i})$$

- $t_i^* = t_i$
- W_t not of bounded variation
- Doesn't obey usual integration by parts rule
- Convergence is in L^2 and probability

What kind of solutions do SDEs have?

- Path of solution: Draw random path $\mathbf{w}(t)$ (or $\beta(t)$) and solve the equation using it as the input.
  - Monte Carlo simulation of SDE solutions.
  - Used in particle filtering and smoothing methods.
- Distribution of solution: Given many random $\mathbf{w}(t)$'s, what is the distribution of the state $p(\mathbf{x}(t))$?
  - Solution is given by the Fokker-Planck-Kolmogorov PDE.
  - Used in grid based and basis function methods (FEM, BEM).
- Moments: What are the mean and covariance of $\mathbf{x}(t)$?
  - Ordinary differential equations for the mean and covariance.
  - Used in non-linear Kalman (Gaussian) filters and smoothers.

What does a solution of SDE look like? (cont.)

Example 3.3 (Solution of Ornstein–Uhlenbeck equation). *The complete solution to the scalar SDE*

$$\mathrm{d}x = -\lambda\, x\ \mathrm{d}t + \mathrm{d}\beta, \qquad x(0) = x_0, \tag{3.32}$$

where $\lambda > 0$ *is a given constant and* $\beta(t)$ *is a Brownian motion is*

$$x(t) = \exp(-\lambda\, t)\, x_0 + \int_0^t \exp(-\lambda\,(t-\tau))\ \mathrm{d}\beta(\tau). \tag{3.33}$$

The solution, called the Ornstein–Uhlenbeck process, is illustrated in Figure 3.2.

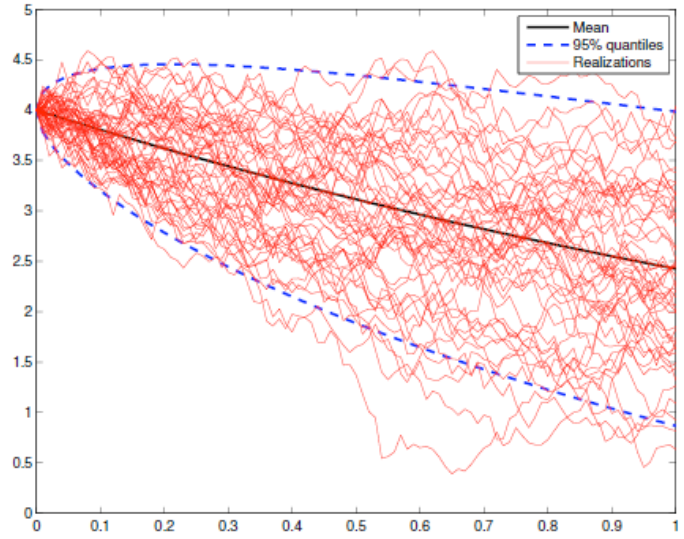
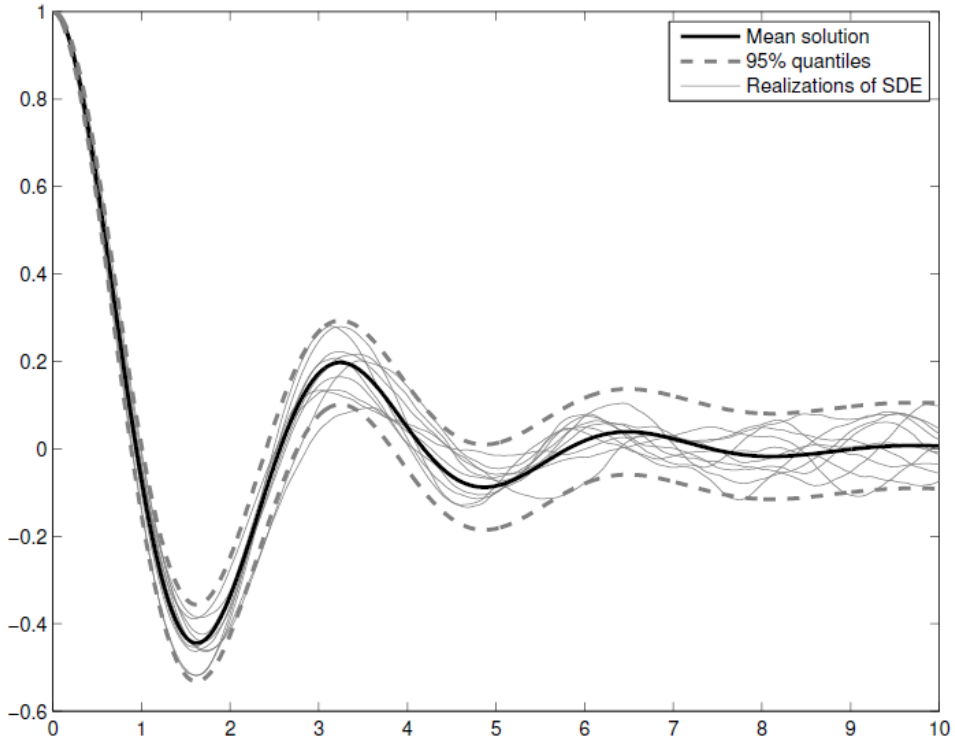


Figure 3.2: Realizations, mean, and 95% quantiles of Ornstein–Uhlenbeck process.

What does a solution of SDE look like? (cont.)



Paths of stochastic spring model

$$\frac{d^2x(t)}{dt^2} + \gamma \frac{dx(t)}{dt} + \nu^2 x(t) = w(t).$$

Example 2.8 (Stochastic Spring model). *If in the spring model of Equation* (1.4), *we replace the input force with a white noise with spectral density* q, *we get the following LTI SDE:*

$$\underbrace{\begin{pmatrix} \frac{dx_1(t)}{dt} \\ \frac{dx_2(t)}{dt} \end{pmatrix}}_{d\mathbf{x}(t)/dt} = \underbrace{\begin{pmatrix} 0 & 1 \\ -\nu^2 & -\gamma \end{pmatrix}}_{\mathbf{F}} \underbrace{\begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}}_{\mathbf{x}} + \underbrace{\begin{pmatrix} 0 \\ 1 \end{pmatrix}}_{\mathbf{L}} w(t). \tag{2.39}$$

The equations for the mean and covariance are thus given as

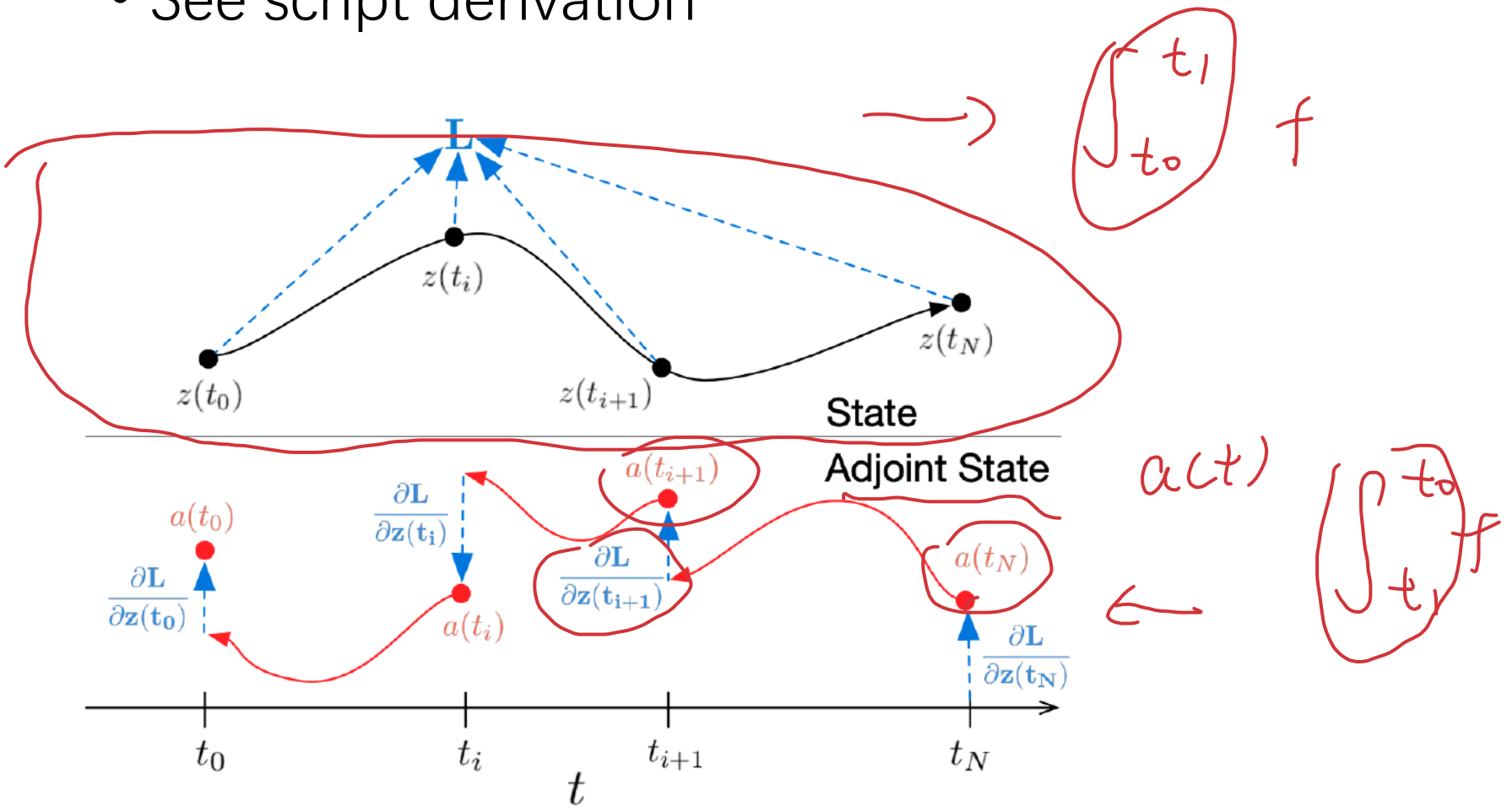
$$\begin{pmatrix} \frac{dm_1}{dt} \\ \frac{dm_2}{dt} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\nu^2 & -\gamma \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \end{pmatrix}$$

$$\begin{pmatrix} \frac{dP_{11}}{dt} & \frac{dP_{12}}{dt} \\ \frac{dP_{21}}{dt} & \frac{dP_{22}}{dt} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\nu^2 & -\gamma \end{pmatrix} \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} + \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -\nu^2 & -\gamma \end{pmatrix}^{\mathsf{T}} + \begin{pmatrix} 0 & 0 \\ 0 & q \end{pmatrix} \tag{2.40}$$

When **F** and **L** are non-constant functions, always hard to get such a solution.

Recap Neural ODE and adjoint method

- See script derivation



Combining Adjoints with Reverse-mode Autodiff

- Chen et al. [1] combined adjoints with reverse-mode autodiff.
- Define loss that is a function of the end state $\mathcal{L} = \mathcal{L}(z(T))$.
- Define vector field $f(z, t, \theta)$, where parameter θ is fixed.
- θ is the control in the maximum principle framework.
- Define adjoint $a(t) = \partial\mathcal{L}/\partial z(t)$; it satisfies another ODE:

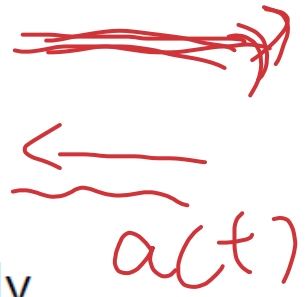
$$\frac{\partial \mathcal{L}}{\partial x(0)} = a(0) = a(T) - \int_T^0 \underbrace{a(t)\frac{\partial f(z, t, \theta)}{\partial z}}_{\text{vJp}} \mathrm{d}t. \quad (1)$$

- Term in blue is evaluated as *vector-Jacobian product* (vJp).
- vJps are almost as cheap as evaluating the function itself; much cheaper than evaluating the full Jacobian!

Nature extension:

- Use **NNs to parameterize a SDE ?**-easy and trivial
- Still use **original adjoint** to optimize? – non-trivial, very difficult

Generalizing Adjoints to SDEs is Difficult



- To machine learners, we want a method that scales gracefully in terms of both memory and time.
- For ODEs, the adjoint sensitivity method achieved this by
  - **i)** solving adjoint dynamics in reverse time (memory), and
  - **ii)** relying on vJps (time) in numerical solve.
- For SDEs, it is less clear what "solving (pathwise) in reverse time" means.
- The classical approach to SDEs is built upon Itô calculus, which
  - **a)** doesn't obey usual rules of integration (e.g. the usual integration by parts), and
  - **b)** rarely considers sample paths individually.

Challenge 1: Simulating SDEs Backward in Time

- One alternative way to think about SDEs is to reason about the pathwise behavior of solutions and consider the induced *stochastic flow*.
- Just like how ODE flows can be diffeomorphisms, so can SDE flows be, once given a realization of Brownian motion.
- When the SDE is in *Stratonovich form*, Kunita [4] showed a simple *time-reverse SDE* [1] has a solution that corresponds to the *inverse flow* of the original SDE.
- SDEs in Stratonovich form can be converted into Itô form, and vice versa.

Challenge 1: Simulating SDEs Ba
Time

Some advanced definitions in advanced random process and differential geometry, no need to go into such deep..

- One alternative way to think about SDEs i
  the pathwise behavior of solutions and con
  stochastic flow.
- Just like how ODE flows can be diffeomorphisms
  flows be, once given a realization of Brownian m
- When the SDE is in *Stratonovich form*, Kunita [
  simple *time-reverse SDE* [1] has a solution that co
  the *inverse flow* of the original SDE.
- SDEs in Stratonovich form can be converted into Itô form, and vice versa.

solution. Let $\Phi_{s,t}(z) := Z_t^{s,z}$ be the solution at time t when the process is started at z at time s. Given a realization of the Wiener process, this defines a collection of continuous maps $\mathcal{S} = \{\Phi_{s,t}\}_{s \le t; s,t \in \mathbb{T}}$ from $\mathbb{R}^d$ to itself.

Theorem 2.1 ([41, Theorem 3.7.1]). *(a) With probability* 1, *the collection* $\mathcal{S} = \{\Phi_{s,t}\}_{s \le t; s,t \in \mathbb{T}}$ *satisfies the flow property*

$$\Phi_{s,t}(z) = \Phi_{u,t}(\Phi_{s,u}(z)), \quad s \le u \le t, \ z \in \mathbb{R}^d.$$

The backward flow $\breve{\Psi}_{s,t} := \Phi_{s,t}^{-1}$ *satisfies the backward SDE:*

$$\breve{\Psi}_{s,t}(z) = z - \int_s^t b(\breve{\Psi}_{u,t}(z), u)\, \mathrm{d}u - \int_s^t \sigma(\breve{\Psi}_{u,t}(z), u) \circ \mathrm{d}\breve{W}_u, \quad (3)$$

for all $z \in \mathbb{R}^d$ *and* $s, t \in \mathbb{T}$ *such that* $s \le t$.

Stratonovich Integral vs Itô Integral

Stratonovich

$$\int_a^b f(t) \circ \mathrm{d}W_t = \lim_{|\Pi| \to 0} \sum \frac{f(t_i) + f(t_{i+1})}{2} (W_{t_{i+1}} - W_{t_i})$$

- Average of two ends
- W_t not of bounded variation
- **Obeys usual integration by parts rule**
- Convergence is in L^2 and probability

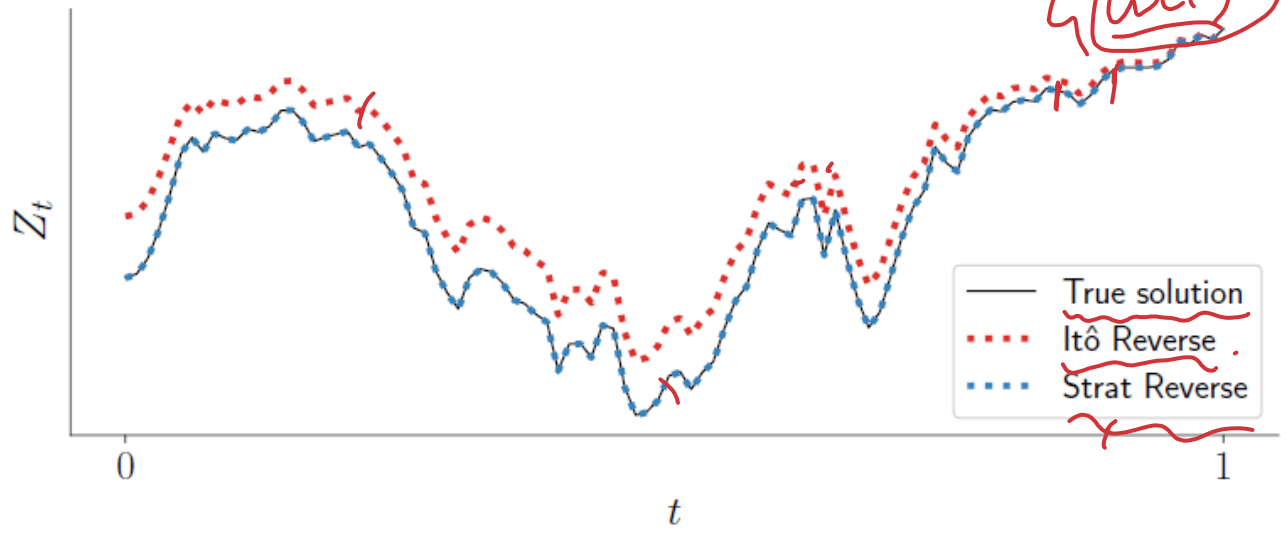
Itô

$$\int_a^b f(t)\, \mathrm{d}W_t = \lim_{|\Pi| \to 0} \sum f(t_i^*)(W_{t_{i+1}} - W_{t_i})$$

- $t_i^* = t_i$
- W_t not of bounded variation
- Doesn't obey usual integration by parts rule
- Convergence is in L^2 and probability

Challenge 1: Simulating Backward in Time

Recall, $\mathcal{O}(1)$ memory sol'n implies a need to reconstruct states.



$$\breve{\Psi}_{s,t}(z) = z - \int_s^t b(\breve{\Psi}_{r,t}(z), r)\,\mathrm{d}r - \sum_{i=1}^{m} \int_s^t \sigma_i(\breve{\Psi}_{r,t}(z), r) \circ \underbrace{\breve{\mathrm{d}}W_r^{(i)}}_{\text{Time-rev. BM}}$$

Stratonovich integral

Extending the Formulation to Adjoints

- With the time-reverse formulation, the task of extending adjoints to SDEs becomes that of rewriting the derivation for ODEs, but with Stratonovich integrals and the time-reverse Wiener process.
- We can still define the adjoint to be the derivative of the loss random variable w.r.t. the state, i.e. $A_t = \partial \mathcal{L} / \partial Z_t$, and show it is the sol'n to a time-reverse SDE:

$$\breve{A}_{s,t}(z) = \nabla \mathcal{L}(z) + \int_s^t \underbrace{\breve{A}_{r,t}(z) \nabla b(\breve{\Psi}_{r,t}(z), r)}_{\text{vJp}} \, \mathrm{d}r$$

$$+ \sum_{i=1}^{m} \int_s^t \underbrace{\breve{A}_{r,t}(z) \nabla \sigma_i(\breve{\Psi}_{r,t}(z), r)}_{\text{vJp}} \circ \breve{\mathrm{d}} W_r^{(i)}.$$

- Derivative of the expected loss is equal to the expectation of derivative $A_{0,T}$, assuming the existence of a dominating R.V.

Algorithm 1 ODE Adjoint Sensitivity

Input: Parameters θ, start time t_0, stop time t_1, final state z_{t_1}, loss gradient $\partial \mathcal{L}/z_{t_1}$, dynamics $f(z, t, \theta)$.

def $\overline{f}([z_t, a_t, \cdot],\ t,\ \theta)$: ▷ Augmented dynamics
  $v = f(z_t, -t, \theta)$
  return $[-v,\ a_t \partial v/\partial z,\ a_t \partial v/\partial \theta]$

$$\begin{bmatrix} z_{t_0} \\ \partial \mathcal{L}/\partial z_{t_0} \\ \partial \mathcal{L}/\partial \theta \end{bmatrix} = \texttt{odeint}\left(\begin{bmatrix} z_{t_1} \\ \partial \mathcal{L}/\partial z_{t_1} \\ \mathbf{0}_p \end{bmatrix}, \overline{f}, -t_1, -t_0 \right)$$

return $\partial \mathcal{L}/\partial z_{t_0}, \partial \mathcal{L}/\partial \theta$

Algorithm 2 SDE Adjoint Sensitivity (Ours)

Input: Parameters θ, start time t_0, stop time t_1, final state z_{t_1}, loss gradient $\partial \mathcal{L}/z_{t_1}$, drift $f(z, t, \theta)$, diffusion $\sigma(z, t, \theta)$, Wiener process sample $w(t)$.

def $\overline{f}([z_t, a_t, \cdot],\ t,\ \theta)$: ▷ Augmented drift
  $v = f(z_t, -t, \theta)$
  return $[-v,\ a_t \partial v/\partial z,\ a_t \partial v/\partial \theta]$

def $\overline{\sigma}([z_t, a_t, \cdot],\ t,\ \theta)$: ▷ Augmented diffusion
  $v = \sigma(z_t, -t, \theta)$
  return $[-v, a_t \partial v/\partial z, a_t \partial v/\partial \theta]$

def $\overline{w}(t)$: ▷ Replicated noise
  return $[-w(-t), -w(-t), -w(-t)]$

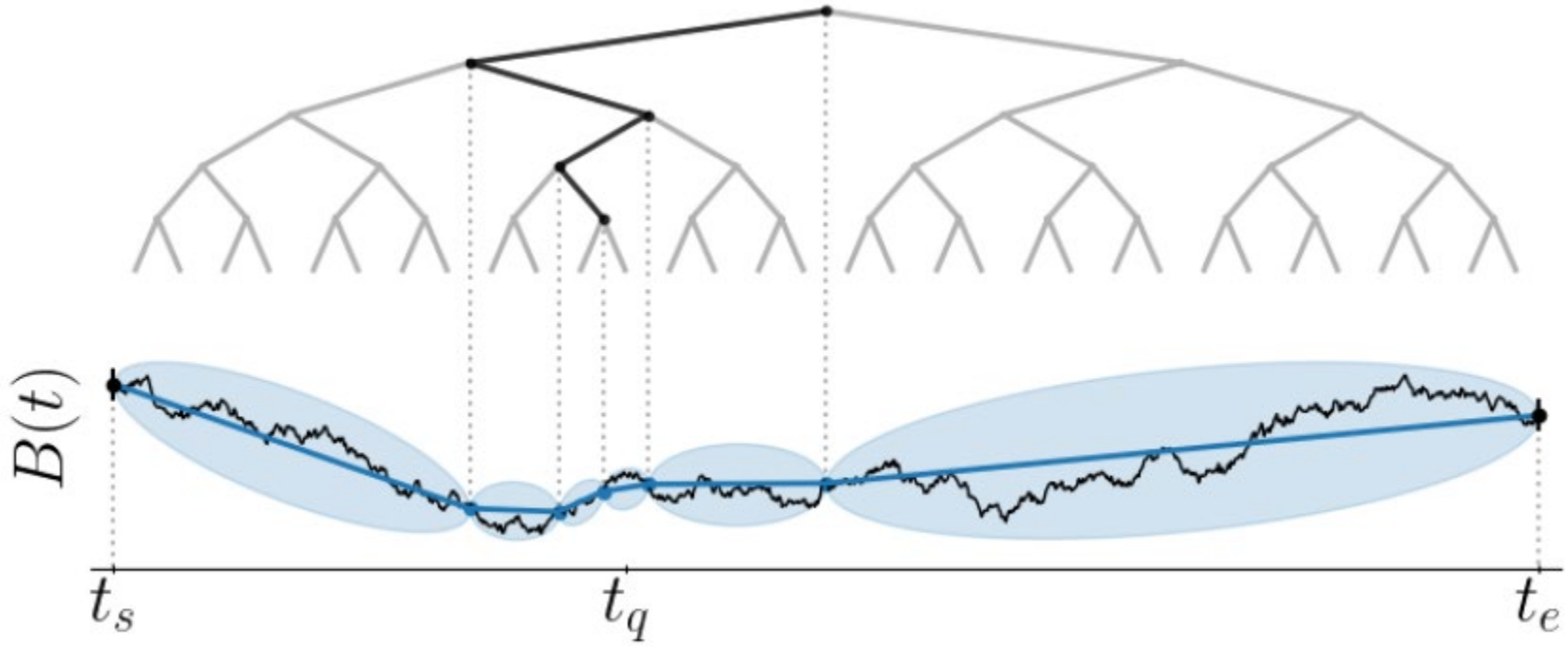
$$\begin{bmatrix} z_{t_0} \\ \partial \mathcal{L}/\partial z_{t_0} \\ \partial \mathcal{L}/\partial \theta \end{bmatrix} = \texttt{sdeint}\left(\begin{bmatrix} z_{t_1} \\ \partial \mathcal{L}/\partial z_{t_1} \\ \mathbf{0}_p \end{bmatrix}, \overline{f}, \overline{\sigma}, \overline{w}, -t_1, -t_0 \right)$$

return $\partial \mathcal{L}/\partial z_{t_0}, \partial \mathcal{L}/\partial \theta$

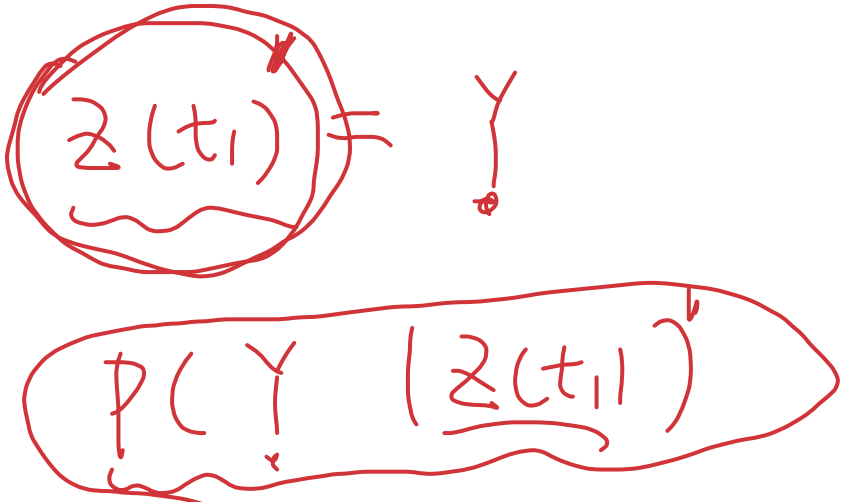
What omit here..

- Some numerical / converge proof..
- Some memory/time saving trick to reproducing noise (Brownian tree)

w(t)



Latent SDE



- Fitting SDEs with flexible drift and diffusion coefficients directly to data using maximum likelihood would likely cause a degenerate solution.
- It is reasonable to expect the diffusion to shrink to zero when flexible ODE models can already do a good job.
- One real benefit of using SDEs is when they are treated as latent variables: Place a prior, and infer the posterior given data.

Latent SDE

Different function

Same function (no parameter)

$$\mathrm{d}\tilde{Z}_t = h_\theta(\tilde{Z}_t, t)\,\mathrm{d}t + \sigma(\tilde{Z}_t, t)\,\mathrm{d}W_t, \quad \text{(prior)}$$

$$\mathrm{d}Z_t = h_\phi(Z_t, t)\,\mathrm{d}t + \sigma(Z_t, t)\,\mathrm{d}W_t, \quad \text{(approx. post.)}$$

where h_θ, h_ϕ, and σ are Lipschitz in both arguments, and both processes have the same starting value: $\tilde{Z}_0 = Z_0 = z_0 \in \mathbb{R}^d$.

If both processes share the same diffusion function σ, then the KL divergence between them is finite (under additional mild regularity conditions; see Appendix 9.7), and can be estimated by sampling paths from the approximate posterior process. Then, the

evidence lower bound (ELBO) can be written as:

$$\log p(x_1, x_2, \ldots, x_N|\theta) \geq \tag{10}$$
$$\mathbb{E}_{Z_t}\left[\sum_{i=1}^{N} \log p(x_{t_i}|z_{t_i}) - \int_0^T \frac{1}{2}|u(z_t,t)|^2 \, \mathrm{d}t\right],$$

where $u : \mathbb{R}^d \times [0,T] \to \mathbb{R}^m$ satisfies

$$\sigma(z,t)u(z,t) = h_\phi(z,t) - h_\theta(z,t),$$

and the expectation is taken over the approximate posterior process defined by (approx. post.). The likelihoods of observations $x_1, \ldots, x_N$ at times $t_1, \ldots, t_N$ depend only on latent states z_t at corresponding times.

Interpreting the Objective

$$\mathcal{L}_{\text{ELBO}} = \underbrace{-\mathbb{E}\left[\frac{1}{2}\int_0^T |u(q_t, t)|^2 \, \mathrm{d}t\right]}_{\text{KL divergence}} + \underbrace{\mathbb{E}\left[\sum_{i=1}^N \log p(y_{t_i}|q_{t_i})\right]}_{\text{negative reconstruction loss}}.$$

As a special case, when the diffusion is a diagonal matrix and non-zero, the loss reduces to

$$\mathcal{L}_{\text{ELBO}} = -\mathbb{E}\left[\frac{1}{2}\int_0^T |(f_p - f_q)/\sigma|^2 \, \mathrm{d}t\right] + \mathbb{E}\left[\sum_{i=1}^N \log p(y_{t_i}|q_{t_i})\right].$$

$f_p - f_q$ characterizes the difference; σ scales the penalty.

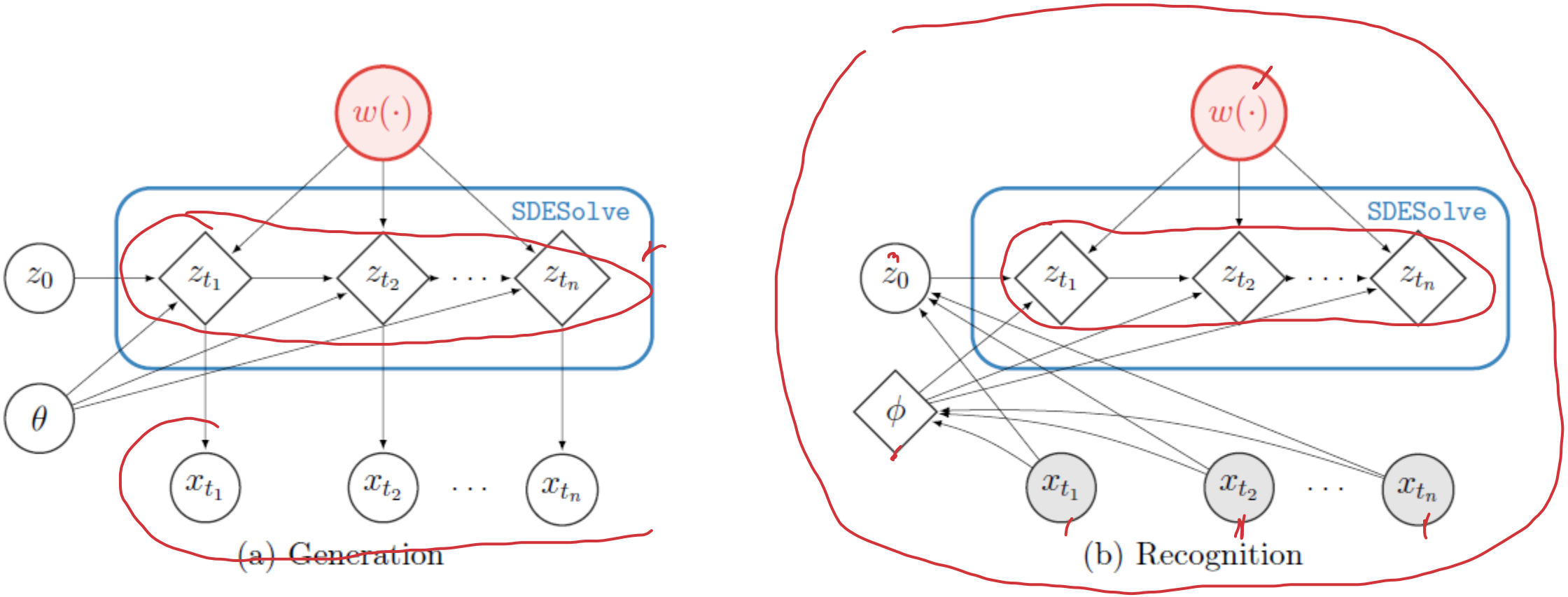


Figure 4: Graphical models for the generative process (decoder) and recognition network (encoder) of the latent stochastic differential equation model. This model can be viewed as a variational autoencoder with infinite-dimensional noise. Red circles represent entire function draws from Brownian motion. Given the initial state z_0 and a Brownian motion sample path $w(\cdot)$, the intermediate states $z_{t_1}, \ldots, z_{t_n}$ are deterministically approximated by a numerical SDE solver.

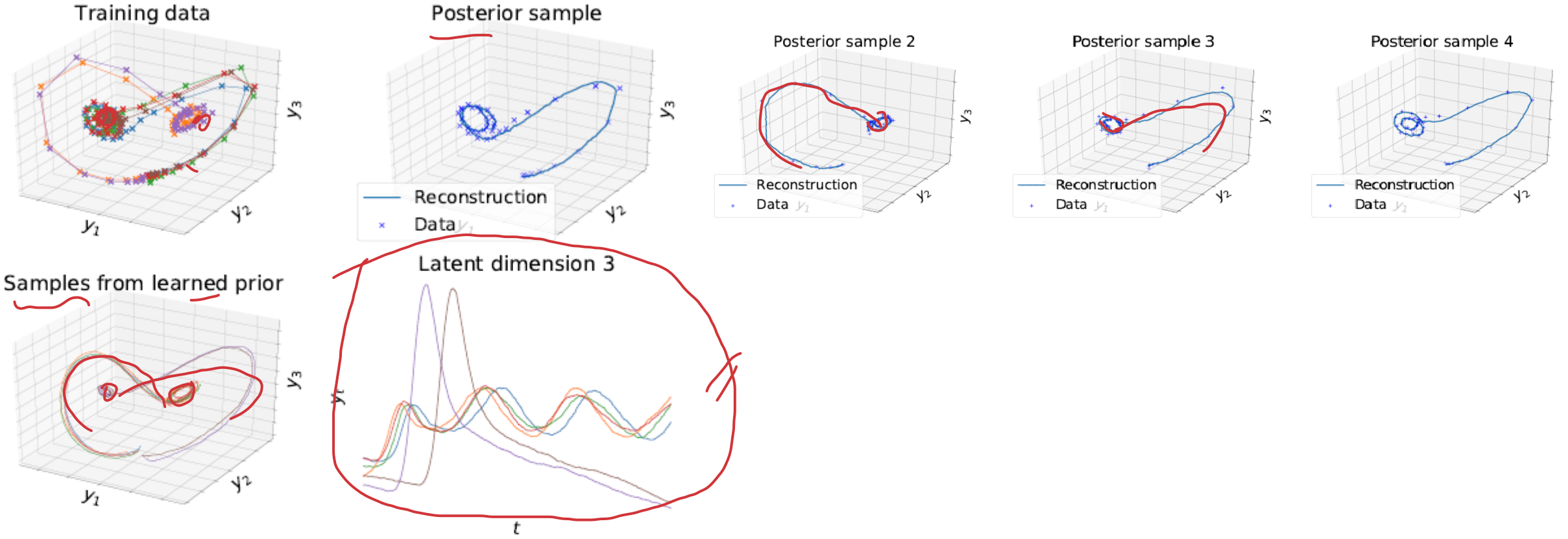


Figure 6: Learned posterior and prior dynamics on data from a stochastic Lorenz attractor. All samples from our model are continuous-time paths, and form a multi-modal, non-Gaussian distribution.

Applications: Fitting Multiple Trajectories

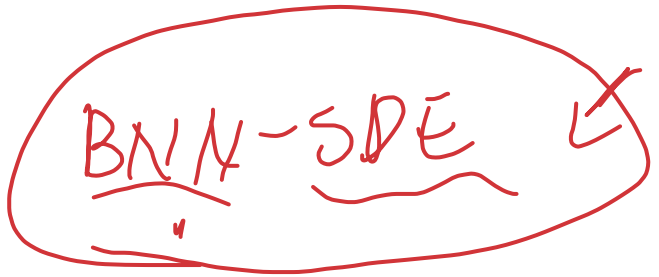
Table: Test MSE for a task on a Mocap dataset; 95% confidence interval averaged over 50 samples. †from [10].

Method	Test MSE
npODE [3]	22.96†
NeuralODE [1]	22.49 ± 0.88†
ODE^{2}VAE [10]	10.06 ± 1.4†
ODE^{2}VAE-KL [10]	8.09 ± 1.95†
Latent ODE [9]	5.98 ± 0.28
Latent SDE (this work)	$\mathbf{4.03 \pm 0.20}$

Summary

- We presented a generalization of the adjoint sensitivity method to SDEs.
- The procedure relies on vJps, and its memory scaling can interpolate between $\mathcal{O}(1)$ and $\mathcal{O}(\text{steps})$ (depending on how much compute time we are willing to trade in).
- We applied adjoints to train latent SDEs.
- Theoretical results also studied a new type of convergence for SDE schemes, i.e. local uniform convergence, for which we focused on the Euler-Maruyama scheme.

Next Steps



- Modeling:
  - Comprehensive empirical comparison: GPs vs SDEs.
  - Stationary and multi-timescale models.
  - Bayesian neural networks.
- Numerical analysis:
  - Convert the method to solve high-dimensional PDEs.
  - New theoretical interpretations using rough path theory.
- Other areas of science and engineering:
  - Better models of allele frequency in population genetics.
  - Fast and memory-frugal algorithms for greeks in finance.
  - Neural science applications?
- Software: A PyTorch package is on its way! Stay tuned!

Neural ODE:

$$\frac{dh(t)}{dt} = f(h, \theta, t) \overset{=}{\underset{NN}{}}$$

$h(t_0) \xrightarrow{f} h(t_1)$

$h(t_1)$: some observed y

$\Uparrow$: if NN given

$$h(t) = h(t_0) + \underbrace{\int_{t_0}^{t} f(h(t'), \theta(t'), t')\, dt'}_{\text{solved by numerical method.}}, \quad (0)$$

Given some loss function L., target value.

Goal: $\frac{\partial L}{\partial \theta}$.

Define: $a(t) = \frac{\partial L}{\partial h(t)}$ (adjoint state)

after some derivation, we get

$$\frac{da(t)}{dt} = -a(t) \frac{\partial f(h(t), t, \theta)}{\partial h(t)} \quad (1)$$

analytic form / auto-diff

a new ode system, solve by numerical.

to get $a(t)$ [bounty : $a(t_1) = \frac{\partial L}{\partial h(t_1)}$

we can get $a(t_0) = \frac{\partial L}{\partial h(t_0)}$ by

inverse inte :

$$a(t_0) = a(t_1) + \int_{t_1}^{t_0} \frac{da(t)}{dt} dt.$$

Similar, we can define adjoint state for θ, t :

$$\begin{cases} a_\theta(t) = \frac{\partial L}{\partial \theta} \\ a_t = \frac{\partial L}{\partial t} \end{cases} \quad (2)$$

with same derivation, get :

$$\begin{cases} \dfrac{a_\theta(t)}{dt} = -\alpha(t)\dfrac{\partial f}{\partial \theta} \\ \dfrac{a_t(t)}{dt} = -\alpha(t)\dfrac{\partial f}{\partial t}. \end{cases} \quad (3)$$

when we get $\alpha(t)$ from ①,

we can use the same numeral solver to

get $a_\theta(t)$, $a_t(t)$

The general step: (for each t)

a. solve ② to get $h(t)$

b. solve ① to get $\alpha(t)$

c. solve ③ to get $\underset{\frac{\partial L}{\partial \theta}}{a_\theta(t)}$, $\underset{\frac{\partial L}{\partial t}}{a_t(t)}$