Training stochastic differential equations as ordinary differential equations

Liam Hodgkinson, Chris van der Heide, Fred Roosta, Michael W. Mahoney

Berkeley UNIVERSITY OF CALIFORNIA

ICSI INTERNATIONAL COMPUTER SCIENCE INSTITUTE

THE UNIVERSITY OF QUEENSLAND AUSTRALIA CREATE CHANGE

Initial Objective

Extend ***continuous normalizing flows*** (neural ODEs as generative models) to ***stochastic differential equations***

Neural Ordinary Differential Equations

An infinitesimal, infinite-depth limit of the ResNet architecture:

$$z_{k+1} = z_k + \alpha f(z_k, \theta), \qquad t = 0, 1, \ldots, K-1.$$

results in an ***ordinary differential equation***:

$$\frac{\mathrm{d}}{\mathrm{d}t} Z(t) = f(Z(t), t, \theta).$$

Ordinary differential equations formed from single-(or multi-)layer neural networks are called ***neural ordinary differential equations*** (Chen et al., 2018).

Excellent for bridging ML into scientific applications

ML and Population Modelling

A Basic Example of a Large-scale ODE

$$\frac{\mathrm{d}}{\mathrm{d}t}N_i(t) = \alpha_i N_i(t) + f_i\left(\sum_{j=1}^{n}\beta_{ij}N_j(t)\right), \quad i = 1, \ldots, n,$$

- Continuous-time recurrent neural networks

  Zhang, H., Wang, Z., & Liu, D. (2014). A comprehensive review of stability analysis of continuous-time recurrent neural networks.
- Metapopulation models

  Hanski, I., & Gyllenberg, M. (1997). Uniting two general patterns in the distribution of species.
- Network models

  Pastor-Satorras, R., & Vespignani, A. (2001). Epidemic dynamics and endemic states in complex networks.

The Adjoint Method

For a scalar loss $L(Z(T))$, the ***adjoint*** $a(t) = \frac{\partial L(Z(T))}{\partial Z(t)}$ satisfies

$$a'(t) = -\nabla_z f(Z(t), t, \theta) a(t), \qquad a(T) = \frac{\mathrm{d}L}{\mathrm{d}z}(Z(T)),$$

and furthermore,

$$\nabla_\theta L(Z(T)) = \int_0^T \nabla_\theta f(Z(t), t, \theta) a(t) \mathrm{d}t.$$

Summary: Gradients can be computed by solving another ODE backwards in time.

- Less memory required than backprop
- Independent of integrator – enables adaptive solvers
- ...but often requires regularization (Finlay et al., 2020)

Chen et al., 2018

Neural Ordinary Differential Equations

Ricky T. Q. Chen*, **Yulia Rubanova***, **Jesse Bettencourt***, **David Duvenaud**
University of Toronto, Vector Institute
{rtqichen, rubanova, jessebett, duvenaud}@cs.toronto.edu

Abstract

We introduce a new family of deep neural network models. Instead of specifying a discrete sequence of hidden layers, we parameterize the derivative of the hidden state using a neural network. The output of the network is computed using a black-box differential equation solver. These continuous-depth models have constant memory cost, adapt their evaluation strategy to each input, and can explicitly trade numerical precision for speed. We demonstrate these properties in continuous-depth residual networks and continuous-time latent variable models. We also construct continuous normalizing flows, a generative model that can train by maximum likelihood, without partitioning or ordering the data dimensions. For training, we show how to scalably backpropagate through any ODE solver, without access to its internal operations. This allows end-to-end training of ODEs within larger models.

Continuous Normalizing Flows

Under the assumption that $Z(0) \sim p_0(\theta)$, becomes a ***generative model*** – a continuous normalizing flow.

It can be trained by ***maximum likelihood estimation*** or ***variational inference*** using

Theorem (Chen et al., 2018)

If $Z(t)$ satisfies $\frac{\mathrm{d}}{\mathrm{d}t}Z(t) = f(Z(t), t, \theta)$, the probability density p_t of $Z(t)$ satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t} \log p_t(Z(t)) = -\nabla_z \cdot f(Z(t), t, \theta) \tag{1}$$

To compute $\log p_T(x)$***:***

1. solve the ODE backwards from $Z(T) = x$ to find $Z(0)$.
2. solve (1) forwards from $\log p_0(Z(0))$.

Stochastic Differential Equations

Consider a (Itô) ***stochastic differential equation*** of the form

$$\mathrm{d}Z_t = \mu(Z_t, t, \theta)\mathrm{d}t + \sigma(Z_t, t, \theta)\mathrm{d}B_t,$$

where B_t is an m-dimensional Brownian motion and $\mu_t : \mathbb{R}^d \to \mathbb{R}^d$, $\sigma_t : \mathbb{R}^d \to \mathbb{R}^{d\times m}$ are the ***drift*** and ***diffusion*** coefficients, respectively.

Appears naturally as a stochastic analogue of ODEs:

- ***finance*** (fundamental theorems of asset pricing + martingale representation theorem)
- ***population modelling*** (diffusion approximation to master equation)
- ***fluid dynamics*** (Mori–Zwanzig equation near equilibrium)

Tzen & Raginsky, 2019

Neural Stochastic Differential Equations: Deep Latent Gaussian Models in the Diffusion Limit

Belinda Tzen* Maxim Raginsky†

Abstract

In deep latent Gaussian models, the latent variable is generated by a time-inhomogeneous Markov chain, where at each time step we pass the current state through a parametric nonlinear map, such as a feedforward neural net, and add a small independent Gaussian perturbation. This work considers the diffusion limit of such models, where the number of layers tends to infinity, while the step size and the noise variance tend to zero. The limiting latent object is an Itô diffusion process that solves a stochastic differential equation (SDE) whose drift and diffusion coefficient are implemented by neural nets. We develop a variational inference framework for these *neural SDEs* via stochastic automatic differentiation in Wiener space, where the variational approximations to the posterior are obtained by Girsanov (mean-shift) transformation of the standard Wiener process and the computation of gradients is based on the theory of stochastic flows. This permits the use of black-box SDE solvers and automatic differentiation for end-to-end

Peluchetti & Favaro, 2019

Infinitely deep neural networks as diffusion processes

Stefano Peluchetti
Cogent Labs
speluchetti@cogent.co.jp

Stefano Favaro
University of Torino and Collegio Carlo Alberto
stefano.favaro@unito.it

Abstract

When the parameters are independently and identically distributed (initialized) neural networks exhibit undesirable properties that emerge as the number of layers increases. These issues include a vanishing dependency on the input and a concentration on restrictive families of functions including constant functions. We consider parameter distributions that shrink as the number of layers increases in order to recover well-behaved stochastic processes in the limit of infinite total depth. Doing so we establish the connection between infinitely deep residual networks and solutions to stochastic differential equations, i.e. diffusion processes. We show that these limiting processes do not suffer from the aforementioned issues and investigate their properties.

1 Introduction

Modern neural network models featuring a large number of layers (depth) and features per layer

Neural Stochastic Differential Equations

Analogously to neural ODEs, ***neural SDEs*** can be constructed from deep Gaussian models:

$$z_{k+1} = z_k + \mathcal{N}(\alpha\mu(z_k, \theta), \alpha\sigma(z_k, \theta)), \quad k = 0, 1, \ldots, K-1,$$

converges to an SDE in the infinite-depth limit.

(Neural) SDEs are (neural) ODEs with injected noise.

Potential advantage: inherent and interpretable stochasticity – potentially more robust to noisy / adversarial data

Regularization: some suggest to take $\sigma \propto \text{diag}(\mu)$.

Neural Stochastic Differential Equations

How do we train neural SDEs?

- Solve Fokker–Planck equations (hugely expensive)
- Backpropagate through a discretization (huge variance)
- ***Kunita's theory of stochastic flows of diffeomorphisms***...

Liu et al., 2019

Neural SDE: Stabilizing Neural ODE Networks with Stochastic Noise

Xuanqing Liu
Department of Computer Science, UCLA
xqliu@cs.ucla.edu

Tesi Xiao
Department of Statistics, UC Davis
texiao@ucdavis.edu

Si Si
Google Research
sisidaisy@google.com

Qin Cao
Google Research
qincao@google.com

Sanjiv Kumar
Google Research
sanjivk@google.com

Cho-Jui Hsieh
Department of Computer Science, UCLA
chohsieh@cs.ucla.edu

Abstract

Neural Ordinary Differential Equation (Neural ODE) has been proposed as a continuous approximation to the ResNet architecture. Some commonly used regularization mechanisms in discrete neural networks (e.g. dropout, Gaussian noise) are missing in current Neural ODE networks. In this paper, we propose a new continuous neural network framework called Neural Stochastic Differential Equation (Neural SDE) network, which naturally incorporates various

Neural Stochastic Differential Equations

How do we train neural SDEs?

- Solve Fokker–Planck equations (hugely expensive)
- Backpropagate through a discretization (huge variance)
- Kunita's theory of stochastic flows of diffeomorphisms (***expensive***)
- ***Stochastic analogue of adjoint method***...

Li et al., 2020

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

Abstract

The adjoint sensitivity method scalably computes gradients of solutions to ordinary differential equations. We generalize this method to stochastic differential equations, allowing time-efficient and constant-memory computation of gradients with high-order adaptive solvers. Specifically, we derive a stochastic differential equation whose solution is the gradient, a memory-efficient algorithm for caching noise, and conditions under which numerical solutions converge. In addition, we combine

allele frequencies in a gene pool [14], or prices ir
market [76]. Previous attempts on fitting SDEs mos
relied on methods with poor scaling properties. T
pathwise approach [21, 85], a form of forward-mo
automatic differentiation, scales poorly in time w
the number of parameters and states in the model.
the other hand, simply differentiating through the
erations of an SDE solver [18] scales poorly in memo

In this work, we generalize the adjoint method
stochastic dynamics defined by SDEs. We give a si
ple and practical algorithm for fitting SDEs with te
of thousands of parameters, while allowing the use
high-order adaptive time-stepping SDE solvers. We

Proposed Training Procedure

Approximate the pathwise Itô differential equation

$$\mathrm{d}Z_t(\omega) = \mu(Z_t(\omega), t, \theta)\mathrm{d}t + \sigma(Z_t(\omega), t, \theta)\mathrm{d}B_t(\omega),$$

via the ordinary differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\tilde{Z}_t(\omega) = F_\omega(\tilde{Z}_t(\omega), t, \theta)$$

with corresponding function

$$F_\omega(z, t, \theta) = \underbrace{\tilde{\mu}_t(z, \theta)}_{\text{Stratonovich drift}} + \sigma_t(z, \theta) \underbrace{\frac{\mathrm{d}\tilde{B}_t(\omega)}{\mathrm{d}t}}_{\text{approximation}},$$

and train as a (random) ODE

Wong–Zakai Approximations

Let $\tilde{B}_t \approx B_t$ be a ***pathwise approximation*** of Brownian motion.

Examples include (letting $\omega_k \sim \mathcal{N}(0,1)$):

- ***Piecewise linear***: Linear interpolation between points of Brownian motion generated exactly:

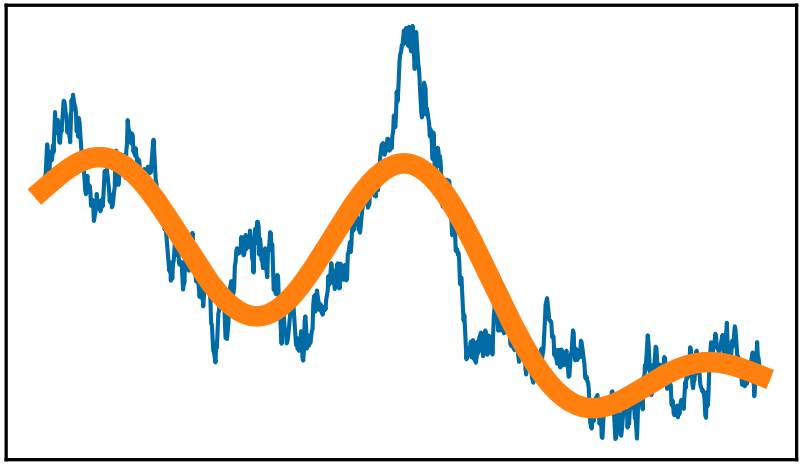
$$\tilde{B}^{(n)}_{t_{k+1}} = \tilde{B}^{(n)}_{t_k} + \omega_k \sqrt{\Delta t_k}, \qquad k = 0, 1, \ldots, n-1.$$

- ***Karhunen-Loève approximation***: An eigenfunction expansion – equivalent to Fourier expansion of white noise

$$\tilde{B}^{(n)}_t = \frac{\omega_0 t}{\sqrt{T}} + \sum_{k=1}^{n-1} \omega_k \frac{\sqrt{2T}\sin(k\pi t/T)}{k\pi}, \qquad n = 1, 2, \ldots$$

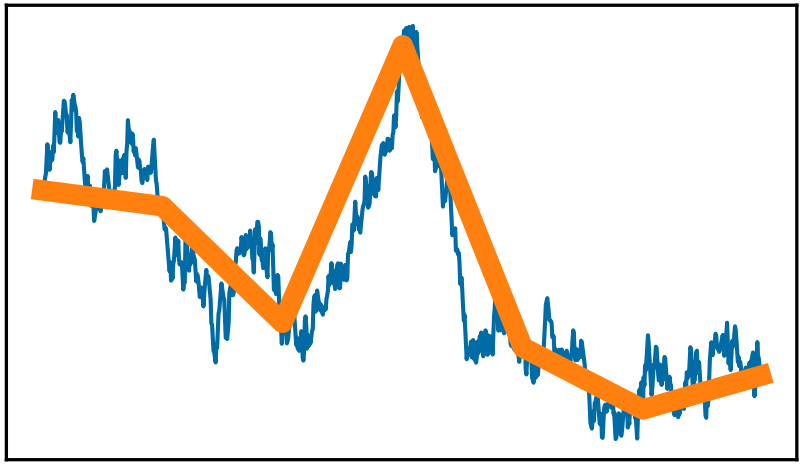
Comparison

Training
Karhunen-Loève ($n = 6$)



- Solve adaptively

Testing
Piecewise linear ($\Delta t = 1/6$)



- Use Euler method (cheap)

Stratonovich Correction

Because the chain rule does not hold in Itô calculus, *naive approximations do not work.*

Solution: Use a different calculus

An Itô SDE can be ***converted to a Stratonovich SDE***:

$$\mathrm{d}Z_t = \tilde{\mu}(Z_t, t, \theta)\mathrm{d}t + \sigma(Z_t, t, \theta) \circ \mathrm{d}B_t, \tag{2}$$

using ***Stratonovich drift correction*** ($z^* = z$):

$$\tilde{\mu}^i(z, t, \theta) = \mu^i(z, t, \theta) - \tfrac{1}{2}\nabla_z \cdot (\sigma_t(z, t, \theta)\sigma_t(z^*, t, \theta)^\top)_i.$$

Stratonovich SDEs satisfy the chain rule, and ***behave like ODEs*** (reversible, approximable).

Stratonovich Correction

Itô Geometric Brownian Motion

$$\mathrm{d}Z_t = \sigma Z_t \mathrm{d}B_t, \qquad Z_t = Z_0 \exp\left(-\frac{\sigma^2}{2}t + \sigma B_t\right)$$

vs.

Classical Exponential

$$\frac{\mathrm{d}\tilde{Z}_t}{\mathrm{d}t} = \sigma \tilde{Z}_t \frac{\mathrm{d}\tilde{B}_t}{\mathrm{d}t}, \qquad \tilde{Z}_t = Z_0 \exp(\sigma \tilde{B}_t).$$

Stratonovich Correction

Stratonovich Geometric Brownian Motion

$$\mathrm{d}Z_t = \sigma Z_t \circ \mathrm{d}B_t, \quad Z_t = Z_0 \exp\left(\sigma B_t\right)$$

vs.

Classical Exponential

$$\frac{\mathrm{d}\tilde{Z}_t}{\mathrm{d}t} = \sigma \tilde{Z}_t \frac{\mathrm{d}\tilde{B}_t}{\mathrm{d}t}, \qquad \tilde{Z}_t = Z_0 \exp(\sigma \tilde{B}_t).$$

Training a Random ODE

The resulting approximation is an ***ODE with latent variables*** ω_k. It can be trained ***using any ODE training procedure***.

Metatheorem

The following procedures are ***consistent*** as $\tilde{B}_t^{(n)} \to B_t$.

- ***Stochastic adjoint method:*** Adjoint method applied to the random ODE – equivalent to (Li et al., 2020).
- ***Density estimation:*** Monte Carlo estimation for density using CNF with random ODE.
- ***Stochastic continuous normalizing flow:*** Perform semi-implicit variational inference using CNF with random ODE.

Density Estimation (naive)

The SDE is ***approximated pathwise*** by the ***random ODE***

$$\frac{\mathrm{d}}{\mathrm{d}t}Z(t) = F(Z(t), t, \theta, \omega),$$

where ω are ***random variables*** used to generate the Brownian motion path.

The ***CNF framework*** is used to estimate $\log p_t(Z(t)|\omega)$. Then

$$p_t(Z(t)) = \mathbb{E}p_t(Z(t)|\omega) \approx \frac{1}{N}\sum_{i=1}^{N} \exp\left(\log p_t(Z(t)|\omega_i)\right),$$

where each ω_i is independent and identically distributed.

Implementation with TorchDyn

```
from torchdyn import NeuralDE
from torchdyn.models import CNF

# Neural ODE
nde = NeuralDE(integrand)

# Neural SDE
stoch_fn = SDE_RODE(drift, diffusion)
sde      = NeuralDE(stoch_fn)

# Continuous normalizing flow
cnf = NeuralDE(CNF(integrand))

# Stochastic continuous normalizing flow
stoch_fn = SDE_RODE(drift, diffusion)
scnf     = NeuralDE(CNF(stoch_fn))
```

A Toy – Training a Sampler

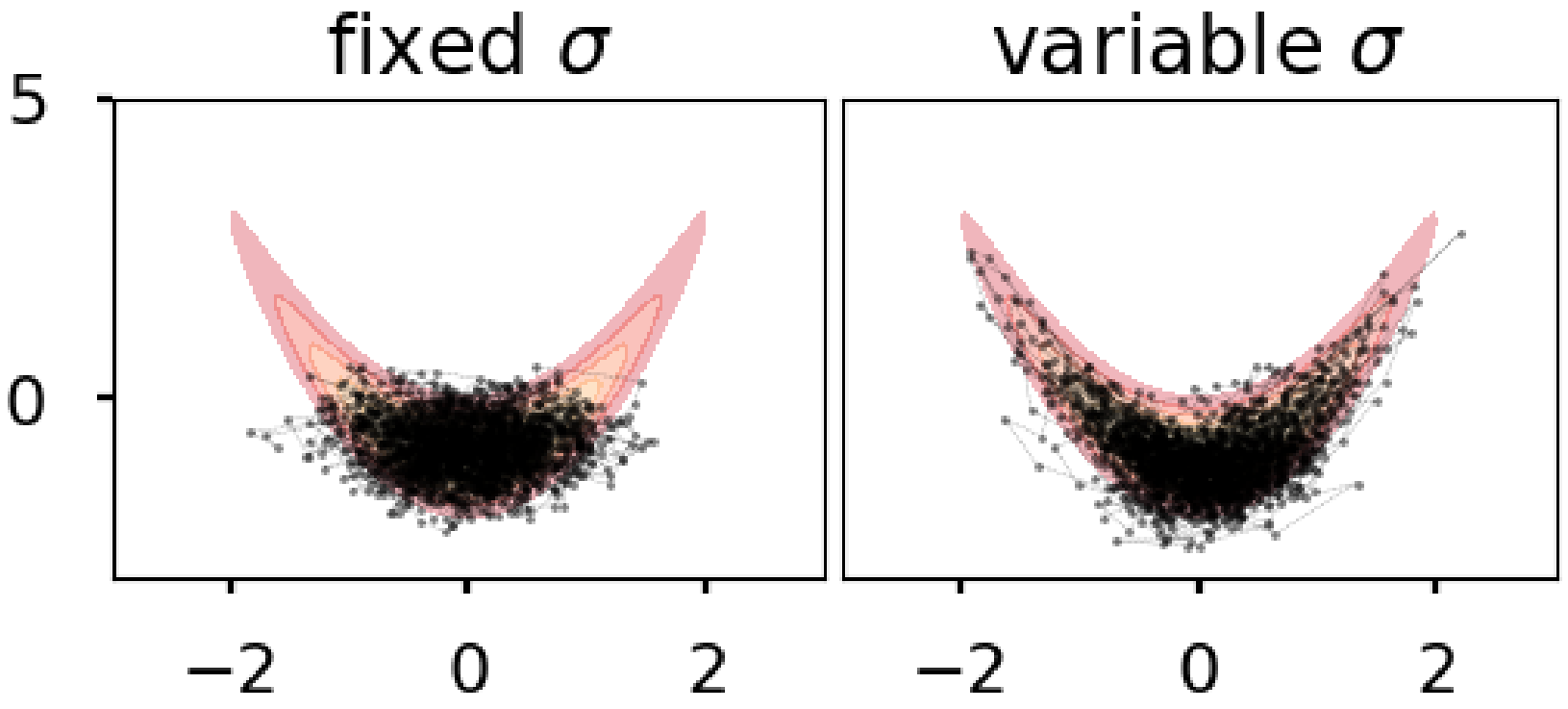


Figure: Sample paths from SDEs trained as SCNF to a target ***banana-shaped distribution***. The drift μ is parameterized as a 3-layer NN. (Left) constant diffusion $\sigma = I$; (right) diffusion σ parameterized as a 3-layer NN.

Losing Detail – Regularization

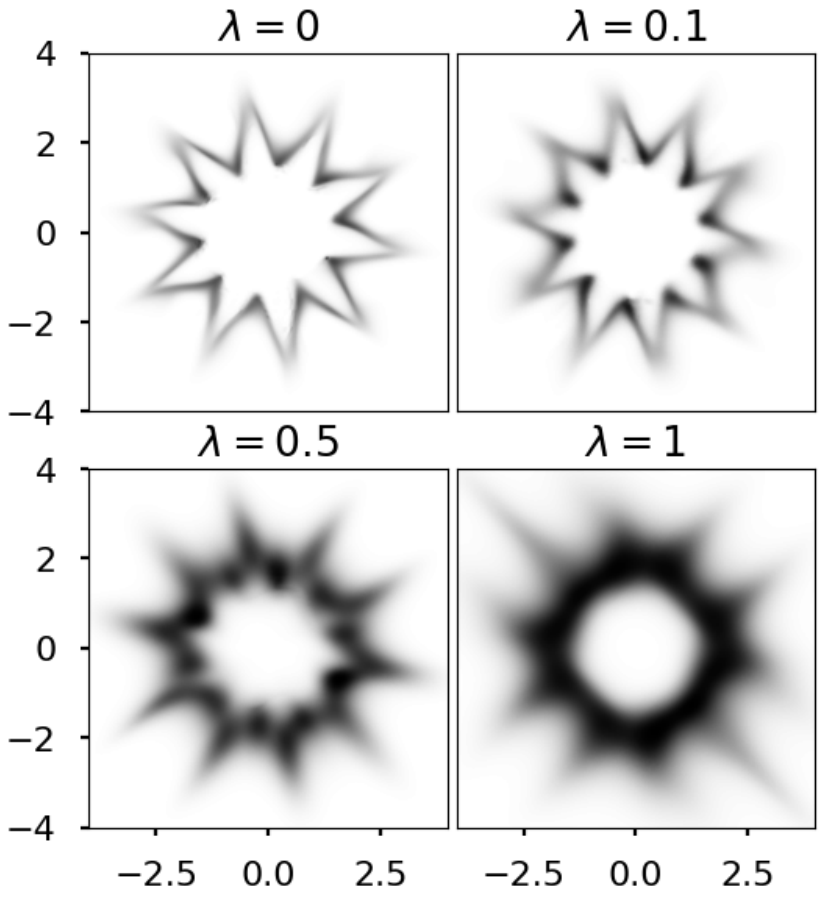


Figure: Density estimates from SCNF trained to a target ***star-shaped distribution***. Drift and diffusion are parameterized as 3-layer NN, with $\text{tr}(\sigma(x)) \equiv 2\lambda$, for varying λ.

Conclusions

- A ***simple, general*** procedure to train SDE-based models
- Extension of continuous normalizing flows to a ***stochastic*** setting.
- ***Future Work:*** consider ***scientific applications*** (fitting large metapopulation diffusion models).

Caveats and Current Problems:

- A stochastic differential equation may not be the best model!
- ***Overfitting:*** Without regularization or appropriate class of σ, training to $\sigma \equiv 0$ is *always* preferable in the overparameterized regime.
- Variance for SCNF is still large – need more sophisticated variance reduction techniques

Theoretical Justification

To justify this idea, we will need to appeal to some very technical probabilistic machinery: ***the theory of rough paths.***

...please bear with me...

Rough Path Theory

First introduced in (Lyons, 1998) to provide a supporting ***pathwise theory*** for SDEs.

Suppose that we would like to prescribe meaning to the infinitesimal limit of the sequence of iterates

$$Z_{t+h} = Z_t + f(Z_t)(X_{t+h} - X_t), \qquad \text{as } h \to 0^+.$$

(e.g. $X_t = B_t$ Brownian motion).

A strong limit may fail to exist if X_t is too ***rough***.

Rough Path Theory

The problem is that the discretization invokes the zeroth-order approximation

$$f(Z_{t+s}) \approx f(Z_t), \qquad \text{for } 0 \leq s \leq h.$$

By instead taking a ***first-order*** approximation:

$$\begin{aligned} f(Z_{t+s}) &\approx f(Z_t) + \nabla_z f(Z_t)(Z_{t+s} - Z_t) \\ &\approx f(Z_t) + \nabla_z f(Z_t) f(Z_t)(X_{t+s} - X_t), \end{aligned}$$

we arrive at the ***Davie scheme***:

$$Z_{t+h} = Z_t + f(Z_t)(X_{t+h} - X_t) + \nabla_z f(Z_t) f(Z_t) \mathbb{X}_{t,t+h},$$

where $\mathbb{X}_{s,t} \simeq \int_s^t X_r \mathrm{d}X_r^\top$.

Rough Path Theory

The "integral" $\mathbb{X}_{s,t} \simeq \int_s^t X_r \mathrm{d}X_r^\top$ cannot be determined from the path X itself, so instead we prescribe it.

Each choice $\mathbb{X}$ satisfying ***Chen's relations***

$$\mathbb{X}_{s,t} - \mathbb{X}_{s,u} - \mathbb{X}_{u,t} = (X_s - X_u)(X_t - X_u)^\top,$$

will reveal a ***different limit*** for

$$Z_{t+h} = Z_t + f(Z_t)(X_{t+h} - X_t) + \nabla_z f(Z_t) f(Z_t) \mathbb{X}_{t,t+h},$$

as $h \to 0^+$ (for sufficiently "nice" X, including Brownian motion!).

The pair $(X, \mathbb{X})$ is a ***rough path***, and Z solves the RDE $\mathrm{d}\boldsymbol{Z}_t = f(Z_t)\mathrm{d}\boldsymbol{X}_t$.

Connections to SDE

Pathwise solutions to SDEs may be interpreted as RDEs

- Itô Brownian motion: $\mathbb{B}_{s,t}^{\text{Itô}} = B_t(B_t - B_s)^\top - \frac{1}{2}(t-s)I$
- Stratonovich Brownian motion: $\mathbb{B}_{s,t}^{\text{Strat}} = B_t(B_t - B_s)^\top$

In particular, Stratonovich Brownian motion is a ***geometric rough path***. These have two key properties:

1. Any solution to an RDE involving a geometric rough path can be uniformly approximated by solutions to corresponding ODEs.
2. Antisymmetric time-reversal: $\mathrm{d}\boldsymbol{X}_{-t} = -\mathrm{d}\boldsymbol{X}_t$.

Main Result

Theorem (Setup)

Let $\boldsymbol{X} = (X, \mathbb{X})$ be an α-Hölder geometric rough path, and $\{X^{(n)}\}_{n=1}^{\infty}$ a sequence of increasingly accurate approximations of X. Let $\boldsymbol{Z}, Z^1, Z^2, \ldots$ be solutions to

$$\mathrm{d}\boldsymbol{Z}_t = f(Z_t, t, \theta)\mathrm{d}\boldsymbol{X}_t \qquad Z_0 \sim p_0$$

$$\frac{\mathrm{d}Z_t^{(n)}}{\mathrm{d}t} = f(Z_t^{(n)}, t, \theta)\frac{\mathrm{d}X_t^{(n)}}{\mathrm{d}t} \qquad Z_0^{(n)} = Z_0,$$

where $f \in \mathcal{C}_b^4(\mathbb{R}^d \times [0, T] \times \mathbb{R}^m)$ and p_0 is a density on $\mathbb{R}^d$ such that $\log p_0$ is continuous...

Main Result

Theorem (Conclusion)

Then Z_t has density p_t satisfying

1. For $x \in \mathbb{R}^d$, $\sup_{t\in[0,T]} |\log p_t^{(n)}(x) - \log p_t(x)| \to 0$ as $n \to \infty$.
2. The path $t \mapsto \log p_t(Z_t)$ is the unique solution to the rough differential equation

$$\mathrm{d}\log p_t(Z_t) = -\nabla_z \cdot (f(Z_t, t, \theta)\mathrm{d}\boldsymbol{X}_t).$$

3. For any smooth loss function $L : \mathbb{R}^{d+1} \to \mathbb{R}$ and $t \geq 0$, as $n \to \infty$,

$$\nabla_\theta L(Z_t^{(n)}, \log p_t^{(n)}(Z_t^{(n)})) \to \nabla_\theta L(Z_t, \log p_t(Z_t)).$$