Continuous-Discrete Differentiable Particle Filtering

Yunpeng Li

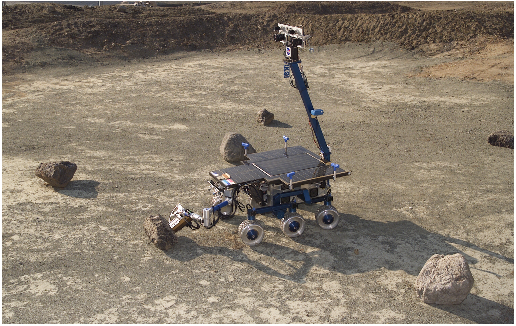
Major contributor: Xiongjie Chen

School of Computer Science and Electronic Engineering
University of Surrey, United Kingdom

Bellairs Workshop 2024

Motivating examples: Mars rover[1]

- Continuous-time sequential state estimation.
- Missing observations, irregularly distributed time grid.



K9 Mars rover
Image source: NASA

[1] Ng et al., "Continuous Time Particle Filtering", IJCAI, 2005.

Motivating examples: dental disease detection

- Continuous-time sequential state estimation.
- Missing observations, irregularly distributed time grid.

January 2019

February 2023

Some Initial Explorations of Differentiable Particle Filters

Yunpeng Li

Collaborators: Xiongjie Chen, Hao Wen, Ge
Conghui Hu

Department of Computer Scienc
University of Surrey, UK

Bellairs Workshop 2021

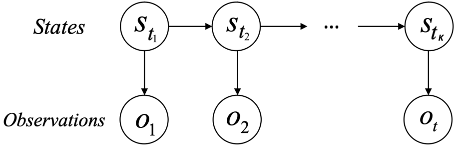
Future directions

- Online learning.
- Better dynamic models – e.g. mixture models?
- Better measurement models.
- Better proposal distributions – e.g. APF with normalizing flow, physics-inspired NNs?
- Better optimisation objectives – existing VI-based, self-supervised methods?
- Differentiable resampling schemes[13].
- Distributed learning and inference.
- Continuous-time filtering.

[13] Corenflos et al., "Differentiable Particle Filtering via Entropy-Regularized Optimal Transport", ICML, 2021.

44 / 44

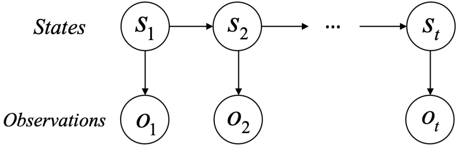
Continuous-discrete state-space models



- Dynamic model:
  $\mathrm{d}s_t = f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$
- Measurement model:
  $o_{t_k} = H_{t_k}(s_{t_k}, v_{t_k}, \theta)$

s_{t_k}	the hidden state at time t_k
θ	model parameters
$f_\theta, \sigma_\theta, H_t$	deterministic functions
o_{t_k}	the observation at time t_k
$\mathrm{d}B_t$	Brownian motion
v_{t_k}	measurement noise

Discrete state-space models

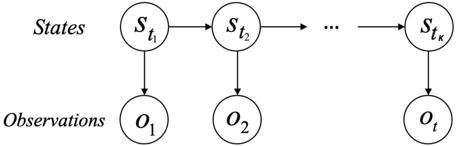


- Dynamic model:
  $s_t = K_t(s_{t-1}, u_t, \theta)$
- Measurement model:
  $o_t = H_t(s_t, v_t, \theta)$

s_t	the hidden state at time t
θ	model parameters
f_θ, K_t, H_t	deterministic functions
o_t	the observation at time t
u_t, v_t	noise terms

Examples of continuous-discrete state-space models

Continuous-discrete stochastic volatility model

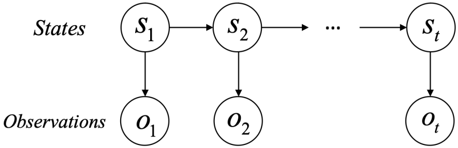


- Dynamic model:
  $\mathrm{d}s_t = f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$
- Measurement model:
  $o_{t_k} = H_{t_k}(s_{t_k}, v_{t_k}, \theta)$

- $\mathrm{d}s_t = (\eta - 1)(s_t - \mu)\mathrm{d}t + \beta \mathrm{d}B_t$
- $o_{t_k} | s_{t_k} \sim \mathcal{N}(0, \gamma^2 \exp(s_t))$

Examples of continuous-discrete state-space models

Discrete stochastic volatility model: evenly spaced time instances.

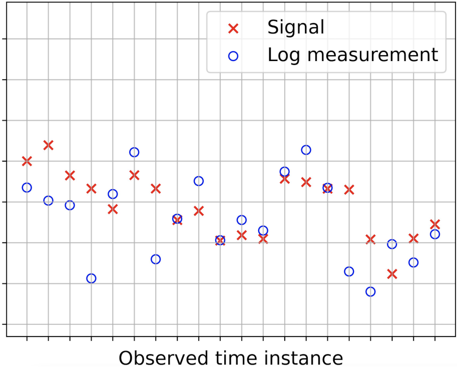


- Dynamic model: $s_t = K_t(s_{t-1}, u_t, \theta)$
- Measurement model: $o_t = H_t(s_t, v_t, \theta)$

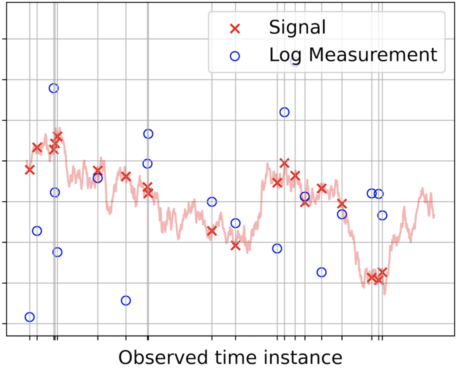
- $s_t = \eta(s_{t-1} - \mu) + \mu + u_t$, $u_t \sim \mathcal{N}(0, \beta^2)$
- $o_t | s_t \sim \mathcal{N}(0, \gamma^2 \text{exp}(s_t))$

Comparison of discrete and continuous-discrete state-space models

Stochastic volatility model



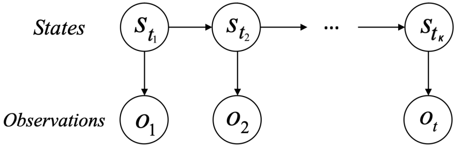
Discrete



Continuous-discrete

Examples of continuous-discrete state-space models

Noisy pendulum model

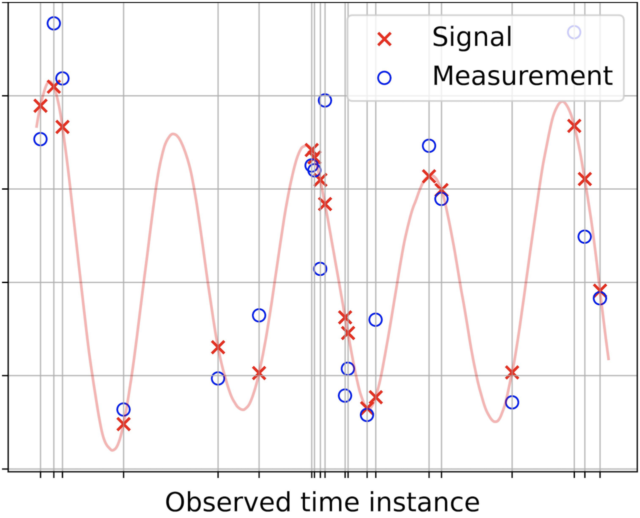


- Dynamic model:
  $\mathrm{d}s_t = f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$
- Measurement model:
  $o_{t_k} = H_{t_k}(s_{t_k}, v_{t_k}, \theta)$

- $\mathrm{d}s_t(1) = s_t(2)\mathrm{d}t$
  $\mathrm{d}s_t(2) = -a^2 \sin(s_t(1))\mathrm{d}t + b^{\frac{1}{2}}\mathrm{d}B_t$
- $o_{t_k} | s_{t_k} \sim \mathcal{N}(s_t(1), \sigma^2)$

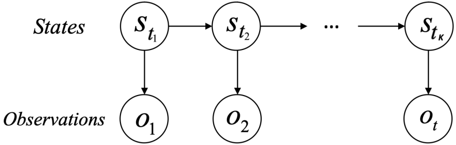
Examples of state-space models

Noisy pendulum model



Continuous-discrete filtering problem formulation

Continuous-discrete filtering



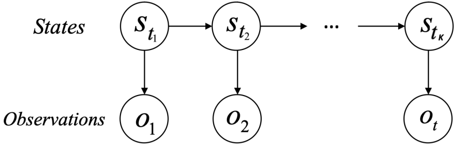
- Recursively estimate the posterior distribution of latent states $p(s_{t_{1:k}}|o_{t_{1:k}};\theta)$ in continuous-time.
- We can infer the posterior at arbitrary time instances.

[1] Ng et al., "Continuous Time Particle Filtering", IJCAI, 2005.

[2] Bucy et al., "Filtering for Stochastic Processes with Applications to Guidance", American Mathematical Society, 2005.

Continuous-discrete filtering problem formulation

Continuous-discrete filtering



- Recursively estimate the posterior distribution of latent states $p(s_{t_{1:k}}|o_{t_{1:k}};\theta)$ in continuous-time.
- We can infer the posterior at arbitrary time instances.

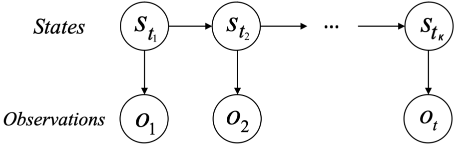
Linear Gaussian models: Kalman-Bucy filters[2].

[1] Ng et al., "Continuous Time Particle Filtering", IJCAI, 2005.

[2] Bucy et al., "Filtering for Stochastic Processes with Applications to Guidance", American Mathematical Society, 2005.

Continuous-discrete filtering problem formulation

Continuous-discrete filtering



- Recursively estimate the posterior distribution of latent states $p(s_{t_{1:k}} | o_{t_{1:k}}; \theta)$ in continuous-time.
- We can infer the posterior at arbitrary time instances.

Linear Gaussian models: Kalman-Bucy filters[2].

Non-linear non-Gaussian models: Continuous-discrete particle filters[1].

[1] Ng et al., "Continuous Time Particle Filtering", IJCAI, 2005.

[2] Bucy et al., "Filtering for Stochastic Processes with Applications to Guidance", American Mathematical Society, 2005.

Continuous-discrete particle filtering

Components

- System dynamics and proposal process are specified by stochastic differential equations (SDEs).
  - Dynamic process: $\mathrm{d}s_t = f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.
  - Proposal process: $\mathrm{d}s_t = g_\phi(s_t, o_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.

  f_θ, g_ϕ, and σ_θ are deterministic functions.

Continuous-discrete particle filtering

Components

- System dynamics and proposal process are specified by stochastic differential equations (SDEs).
  - Dynamic process: $\mathrm{d}s_t = f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.
  - Proposal process: $\mathrm{d}s_t = g_\phi(s_t, o_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.

  f_θ, g_ϕ, and σ_θ are deterministic functions.

- Measurement model: $p(o_{t_k} | s^i_{t_k}; \theta)$.

Continuous-discrete particle filtering

Components

- System dynamics and proposal process are specified by stochastic differential equations (SDEs).
  - Dynamic process: $\mathrm{d}s_t = f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.
  - Proposal process: $\mathrm{d}s_t = g_\phi(s_t, o_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.

  f_θ, g_ϕ, and σ_θ are deterministic functions.

- Measurement model: $p(o_{t_k}|s^i_{t_k};\theta)$.
- Update particles weights: $w^i_{t_k} = w^i_{t_{k-1}} \frac{p(o_{t_k}|s^i_{t_k};\theta)p(s^i_{t_k}|s^i_{t_{k-1}};\theta)}{q(s^i_{t_k}|s^i_{t_{k-1}},o_{t_k};\phi)}$.

Continuous-discrete particle filtering

Components

- System dynamics and proposal process are specified by stochastic differential equations (SDEs).
  - Dynamic process: $\mathrm{d}s_t = f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.
  - Proposal process: $\mathrm{d}s_t = g_\phi(s_t, o_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.

  f_θ, g_ϕ, and σ_θ are deterministic functions.

- Measurement model: $p(o_{t_k}|s^i_{t_k};\theta)$.
- Update particles weights: $w^i_{t_k} = w^i_{t_{k-1}} \frac{p(o_{t_k}|s^i_{t_k};\theta)p(s^i_{t_k}|s^i_{t_{k-1}};\theta)}{q(s^i_{t_k}|s^i_{t_{k-1}},o_{t_k};\phi)}$.
- Resampling.

Continuous-discrete particle filtering

Initialisation: Draw $\{s^i_{t_0}\}_{i=1}^{N_p}$ from $p(s_{t_0})$.

Set $\{\tilde{w}^i_{t_0} = \frac{1}{N_p}\}_{i=1}^{N_p}$.

for $k = 1$ to K:

for i= 1 to N_p:

$s^i_{t_k} = \int_{t_{k-1}}^{t_k} g_\phi(s_t, o_t, t)\mathrm{d}t + \int_{t_{k-1}}^{t_k} \sigma_\theta(s_t, t)\mathrm{d}B_t$ with $s_{t_{k-1}} = s^i_{t_{k-1}}$.

Update weights: $w^i_{t_k} = w^i_{t_{k-1}} \frac{p(o_{t_k}|s^i_{t_k};\theta)p(s^i_{t_k}|s^i_{t_{k-1}};\theta)}{q(s^i_{t_k}|s^i_{t_{k-1}},o_{t_k};\phi)}$.

end for

for i= 1 to N_p:

Normalise weights: $\tilde{w}^i_{t_k} = w^i_{t_k} / \sum_{j=1}^{N_p} w^j_{t_k}$.

end for

if ESS $<$ threshold:

Resample $\{s^i_{t_k}, \tilde{w}^i_{t_k}\}_{i=1}^{N_p}$ to obtain $\{s^i_{t_k}, \frac{1}{N_p}\}_{i=1}^{N_p}$.

end if

end for

Data-adaptive continuous-discrete particle filtering

Construct continuous-discrete particle filters (DPFs) with machine learning tools:

- Build system dynamics and proposal processes with neural stochastic differential equations (neural SDEs) [3, 4, 5].
- Build measurement models with neural networks[6].

[3] Li et al., "Scalable gradients for stochastic differential equations", AISTATS, 2020.

[4] Deng et al., "Continuous Latent Process Flows", NeurIPS, 2021.

[5] Deng et al., "Continuous-time Particle Filtering for Latent Stochastic Differential Equations", arXiv, 2209.00173, 2022.

[6] Chen and Li, "Conditional Measurement Density Estimation in Sequential Monte Carlo via Normalizing Flow", EUSIPCO, 2022.

Neural ordinary differential equations

Neural ordinary differential equations (Neural ODEs)[7]:

$$\frac{\mathrm{d}s_t}{\mathrm{d}t} = f_\theta(s_t, t)\,,\; s_0 = s(0)\,,$$

model the dynamic function $f_\theta(s_t, t)$ with neural networks.

[7] Chen et al., "Neural odinary differential equations", NeurIPS, 2018.

Neural ordinary differential equations

Neural ordinary differential equations (Neural ODEs)[7]:

$$\frac{\mathrm{d}s_t}{\mathrm{d}t} = f_\theta(s_t, t)\,,\; s_0 = s(0)\,,$$

model the dynamic function $f_\theta(s_t, t)$ with neural networks.

- How to backpropagate gradients through ODE solvers?
  - Adjoint sensitivity method.
  - Can be trained in the same way as normal neural networks.
- Applications:
  - Irregularly-sampled time series modelling.
  - Continuous normalising flows.

[7] Chen et al., "Neural odinary differential equations", NeurIPS, 2018.

Neural stochastic differential equations

Neural stochastic differential equations (Neural SDEs)[8]:

$$\mathrm{d}s_t = f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t \,,$$

a stochastic variant of neural ODEs.

- Li et al. extended the adjoint sensitivity method developed for neural ODEs to neural SDEs[3] - we can backpropagate through SDE solvers.
- A natural choice when dealing with stochastic dynamic systems.

[3] Li et al., "Scalable gradients for stochastic differential equations", AISTATS, 2020.

[8] Tzen and Raginsky, "Neural stochastic differential equations: Deep latent Gaussian models in the diffusion limit", arXiv, 1905.09883, 2019.

Build system dynamics and proposals with neural SDEs

- Dynamic model.
  - Construct with neural SDEs.

$$
\begin{aligned}
\mathrm{d}s_{t_k} &= f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t\,, \\
s_{t_k} &= \int_{t_{k-1}}^{t_k} f_\theta(s_t, t)\mathrm{d}t + \int_{t_{k-1}}^{t_k} \sigma_\theta(s_t, t)\mathrm{d}B_t\,.
\end{aligned}
$$

Build system dynamics and proposals with neural SDEs

- Dynamic model.
  - Construct with neural SDEs.

$$
\begin{aligned}
\mathrm{d}s_{t_k} &= f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t \,, \\
s_{t_k} &= \int_{t_{k-1}}^{t_k} f_\theta(s_t, t)\mathrm{d}t + \int_{t_{k-1}}^{t_k} \sigma_\theta(s_t, t)\mathrm{d}B_t \,.
\end{aligned}
$$

- Proposal process.
  - Construct with neural SDEs, include observations as neural network inputs.

$$
\begin{aligned}
\mathrm{d}s_{t_k} &= g_\phi(s_t, {\color{red}o_t}, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t \,, \\
s_{t_k} &= \int_{t_{k-1}}^{t_k} g_\phi(s_t, {\color{red}o_t}, t)\mathrm{d}t + \int_{t_{k-1}}^{t_k} \sigma_\theta(s_t, t)\mathrm{d}B_t \,.
\end{aligned}
$$

Existing methods

- Latent SDEs[3], continuous latent process flow (CLPF)[4].
  - Generate variational posterior distributions.
  - Model observations as a continuous stochastic process.
- Continuous-time particle filters (CTPF)[5].
  - Non-differentiable resampling.

[3] Li et al., "Scalable gradients for stochastic differential equations", AISTATS, 2020.

[4] Deng et al., "Continuous Latent Process Flows", NeurIPS, 2021.

[5] Deng et al., "Continuous-time Particle Filtering for Latent Stochastic Differential Equations", arxiv, 2209.00173, 2022.

Challenges

Updating particle weights can be difficult:

1. Intractable transition densities.
2. How to design flexible measurement models?

Continuous-discrete DPF: weights update

Recall that when updating particle weights:

$$w^i_{t_k} = w^i_{t_{k-1}} \frac{p(s^i_{t_k} | s^i_{tk-1}) p(o_{t_k} | s^i_{t_k})}{q(s^i_{t_k} | s^i_{t_{k-1}}, o_{t_k})}$$

Continuous-discrete DPF: weights update

Recall that when updating particle weights:

$$w_{t_k}^i = w_{t_{k-1}}^i \frac{p(s_{t_k}^i | s_{tk-1}^i) p(o_{t_k} | s_{t_k}^i)}{q(s_{t_k}^i | s_{t_{k-1}}^i, o_{t_k})}$$

Transition density ratio $\dfrac{p(s_{t_k}^i | s_{t_{k-1}}^i)}{q(s_{t_k}^i | s_{t_{k-1}}^i, o_{t_k})}$.

- Neither $p(s_{t_k}^i | s_{t_{k-1}}^i)$ nor $q(s_{t_k}^i | s_{t_{k-1}}^i, o_{t_k})$ are tractable in continuous-discrete state-space models, because $p(s_{t_k}^i | s_{t_{k-1}}^i)$ and $q(s_{t_k}^i | s_{t_{k-1}}^i, o_{t_k})$ are implicitly defined by neural SDEs.

Continuous-discrete DPF: weights update

How to update particle weights?

$$w^i_{t_k} = w^i_{t_{k-1}} \frac{p(s^i_{t_k}|s^i_{tk-1})p(o_{t_k}|s^i_{t_k})}{q(s^i_{t_k}|s^i_{t_{k-1}}, o_{t_k})}$$

- Restrict to bootstrap filtering approaches, so that $q(s^i_{t_k}|s^i_{t_{k-1}}, o_{t_k})$ and $p(s^i_{t_k}|s^i_{t_{k-1}})$ are cancelled.

Continuous-discrete DPF: weights update

How to update particle weights?

$$w^i_{t_k} = w^i_{t_{k-1}} \frac{p(s^i_{t_k}|s^i_{tk-1})p(o_{t_k}|s^i_{t_k})}{q(s^i_{t_k}|s^i_{t_{k-1}}, o_{t_k})}$$

- Restrict to bootstrap filtering approaches, so that $q(s^i_{t_k}|s^i_{t_{k-1}}, o_{t_k})$ and $p(s^i_{t_k}|s^i_{t_{k-1}})$ are cancelled.
  - No, this will lead to very low sampling efficiency and high variance estimators.

Continuous-discrete DPF: weights update

How to update particle weights?

$$w^i_{t_k} = w^i_{t_{k-1}} \frac{p(s^i_{t_k}|s^i_{tk-1})p(o_{t_k}|s^i_{t_k})}{q(s^i_{t_k}|s^i_{t_{k-1}}, o_{t_k})}$$

- Restrict to bootstrap filtering approaches, so that $q(s^i_{t_k}|s^i_{t_{k-1}}, o_{t_k})$ and $p(s^i_{t_k}|s^i_{t_{k-1}})$ are cancelled.
  - No, this will lead to very low sampling efficiency and high variance estimators.
- Random weight approaches.
  - Update particle weights with unbiased estimators of the ratio

$$\frac{p(s^i_{t_k}|s^i_{t_{k-1}})}{q(s^i_{t_k}|s^i_{t_{k-1}}, o_{t_k})}.$$

Continuous-discrete DPF: weights update

- Importance sampling: $\mathbb{E}_p[f(x)] = \mathbb{E}_q\left[\frac{p(x)}{q(x)}f(x)\right]$.
  How to estimate $\mathbb{E}_p[f(x)]$?
  - Sample from q.
  - Estimate $\mathbb{E}_p[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N}\frac{p(x_i)}{q(x_i)}f(x_i)$.

- Random weight importance sampling[9].
  - Unbiased estimator of the ratio: $\mathbb{E}\Big[\mathcal{Q}\Big] = \frac{p(x)}{q(x)}$.
  - $\mathbb{E}_p[f(x)] = \mathbb{E}_q\left[\frac{p(x)}{q(x)}f(x)\right] = \mathbb{E}_q\Big[\mathbb{E}\big[\mathcal{Q}\big]f(x)\Big]$.

  How to estimate $\mathbb{E}_p[f(x)]$?
  - Sample from q.
  - Draw samples of $\mathcal{Q}$.
  - Estimate $\mathbb{E}_p[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N}\frac{1}{M}\sum_{j=1}^{M}\mathcal{Q}_j f(x_i)$.

[9] Chopin and Papaspiliopoulos, "An Introduction to Sequential Monte Carlo", Springer, 2020.

Continuous-discrete DPF: weights update

- Importance sampling: $\mathbb{E}_p[f(x)] = \mathbb{E}_q\left[\frac{p(x)}{q(x)} f(x)\right]$.

  How to estimate $\mathbb{E}_p[f(x)]$?
  - Sample from q.
  - Estimate $\mathbb{E}_p[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} \frac{p(x_i)}{q(x_i)} f(x_i)$.

- Random weight importance sampling[9].
  - Unbiased estimator of the ratio: $\mathbb{E}\left[\mathcal{Q}\right] = \frac{p(x)}{q(x)}$.
  - $\mathbb{E}_p[f(x)] = \mathbb{E}_q\left[\frac{p(x)}{q(x)} f(x)\right] = \mathbb{E}_q\left[\mathbb{E}\left[\mathcal{Q}\right] f(x)\right]$.

  How to estimate $\mathbb{E}_p[f(x)]$?
  - Sample from q.
  - Draw samples of $\mathcal{Q}$.
  - Estimate $\mathbb{E}_p[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} \frac{1}{M}\sum_{j=1}^{M} \mathcal{Q}_j f(x_i)$.

[9] Chopin and Papaspiliopoulos, "An Introduction to Sequential Monte Carlo", Springer, 2020.

Continuous-discrete DPF: weights update

Random weight particle filtering[10,11].

Initialisation: Draw $\{s_{t_0}^i\}_{i=1}^{N_p}$ from $p(s_{t_0})$.
Set $\{\tilde{w}_{t_0}^i = \frac{1}{N_p}\}_{i=1}^{N_p}$.

for $k = 1$ to K:
  for i= 1 to N_p:
    Draw $s_{t_k}^i$ from $q(s_{t_k}^i|s_{t_{k-1}}^i, o_{t_k}; \phi)$.
    Draw samples $\{\mathcal{Q}_{t_k}^j\}_{j=1}^M$, with $\mathbb{E}[\mathcal{Q}] = \frac{p(o_{t_k}|s_{t_k}^i;\theta)p(s_{t_k}^i|s_{t_{k-1}}^i;\theta)}{q(s_{t_k}^i|s_{t_{k-1}}^i,o_{t_k};\phi)}$.
    Update weights: $w_{t_k}^i = w_{t_{k-1}}^i \frac{1}{M} \sum_{j=1}^M \mathcal{Q}_j$.
  end for
  for i= 1 to N_p:
    Normalise weights: $\tilde{w}_{t_k}^i = w_{t_k}^i / \sum_{j=1}^{N_p} w_{t_k}^j$.
  end for
  if ESS $<$ threshold:
    Resample $\{s_{t_k}^i, \tilde{w}_{t_k}^i\}_{i=1}^{N_p}$ to obtain $\{s_{t_k}^i, \frac{1}{N_p}\}_{i=1}^{N_p}$.
  end if
end for

[10] Fearnhead et al., "Random-weight particle filtering of continuous time processes", JRSSB, 2010.

[11] Fearnhead et al., "Particle filters for partially observed diffusions", JRSSB, 2008.

Continuous-discrete DPFs: weights update

- Dynamic process: $\mathrm{d}s_t = f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.
- Proposal process: $\mathrm{d}s_t = g_\phi(s_t, o_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.

An unbiased estimator of $\frac{p(s^i_{t_{k+1}}|s^i_{t_k};\theta)}{q(s^i_{t_{k+1}}|s^i_{t_k},o_{t_{1:k}};\phi)}$ derived from the Girsanov theorem:

$$
\begin{aligned}
&Z(t_k, t_{k+1}; \omega^i_{k+1}) \\
&= \exp\Bigg(\int_{t_k}^{t_{k+1}} \Big[f_\theta(s_t, t) - g_\phi(s_t, o_{t_k}, t)\Big]^\top \big[\sigma_\theta^{-1}(s_t, t)\big]^\top \mathrm{d}B^i_{t_{k+1}-t_k} \\
&\quad -\frac{1}{2}\int_{t_k}^{t_{k+1}} \Big[f_\theta(s_t, t) - g_\phi(s_t, o_{t_k}, t)\Big]^\top \Big[\sigma_\theta(s_t, t)\sigma_\theta^\top(s_t, t)\Big]^{-1} \\
&\quad \Big[f_\theta(s_t, t) - g_\phi(s_t, h_{t_k}, t)\Big]\mathrm{d}t \Bigg), \\
&= \exp\Bigg(\int_{t_k}^{t_{k+1}} F_{\theta,\phi}(s_t, o_{t_k}, t)\mathrm{d}t + \int_{t_k}^{t_{k+1}} G_{\theta,\phi}(s_t, o_{t_k}, t)\mathrm{d}B^i_{t_{k+1}-t_k} \Bigg).
\end{aligned}
$$

Continuous-discrete DPFs: weights update

How to compute the Ito integral $Z(t_k, t_{k+1}; \omega^i_{k+1})$?

- Augment the latent state dimension with an concatenated dimension.

$$\mathrm{d}S_t = \begin{bmatrix} \mathrm{d}s_t \\ \mathrm{d}s'_t \end{bmatrix} = \begin{bmatrix} g_\phi(s_t, o_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t \\ F_{\theta,\phi}(s_t, o_{t_k}, t)\mathrm{d}t + G_{\theta,\phi}(s_t, o_{t_k}, t)\mathrm{d}B_t \end{bmatrix} \quad (1)$$

[12] Higham, "An algorithmic introduction to numerical simulation of stochastic differential equations", SIAM review, 2001.

Continuous-discrete DPFs: weights update

How to compute the Ito integral $Z(t_k, t_{k+1}; \omega^i_{k+1})$?

- Augment the latent state dimension with an concatenated dimension.

$$\mathrm{d}S_t = \begin{bmatrix} \mathrm{d}s_t \\ \mathrm{d}s'_t \end{bmatrix} = \begin{bmatrix} g_\phi(s_t, o_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t \\ F_{\theta,\phi}(s_t, o_{t_k}, t)\mathrm{d}t + G_{\theta,\phi}(s_t, o_{t_k}, t)\mathrm{d}B_t \end{bmatrix} \tag{1}$$

- Solve the concatenated SDE using SDE solvers, e.g. Euler-Maruyama and Runge-Kutta methods[12].
  - Simultaneously draw particles and compute $Z(t_k, t_{k+1}; \omega^i_{k+1})$ by solving the concatenated SDE.

[12] Higham, "An algorithmic introduction to numerical simulation of stochastic differential equations", SIAM review, 2001.

Continuous-discrete DPF

Initialisation: Draw $\{s^i_{t_0}\}_{i=1}^{N_p}$ from $p(s_{t_0})$.
Set $\{\tilde{w}^i_{t_0} = \frac{1}{N_p}\}_{i=1}^{N_p}$.

for $k = 1$ to K:
  for i= 1 to N_p:
    Draw $s^i_{t_k}$ and estimate $Z(t_{k-1}, t_k; \omega^i_k)$ by solving Equation (1).
    Update weights: $w^i_{t_k} = w^i_{t_{k-1}} p(o_{t_k} | s^i_{t_k}; \theta) Z(t_{k-1}, t_k; \omega^i_k)$.
  end for
  for i= 1 to N_p:
    Normalise weights: $\tilde{w}^i_{t_k} = w^i_{t_k} / \sum_{j=1}^{N_p} w^j_{t_k}$.
  end for
  if ESS $<$ threshold:
    Resample $\{s^i_{t_k}, \tilde{w}^i_{t_k}\}_{i=1}^{N_p}$ to obtain $\{s^i_{t_k}, \frac{1}{N_p}\}_{i=1}^{N_p}$.
  end if
end for

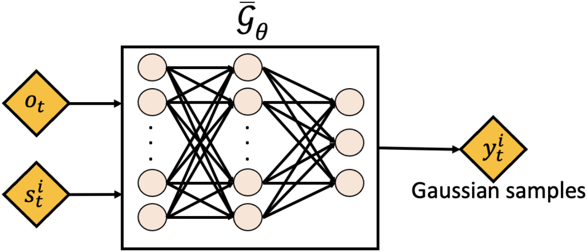
Continuous-discrete DPFs: weights update

Theorem

Given $s^i_{t_{k+1}}$, $s^i_{t_k}$, and $\omega^i_{k+1} \in \Omega_{k+1}$ an outcome that generates a standard Brownian motion $B^i_{t_{k+1}-t_k}$ driving $s^i_{t_k}$ to $s^i_{t_{k+1}}$ in the proposal process, $Z(t_k, t_{k+1}; \omega^i_{k+1})$ is an unbiased estimator of the transition density ratio $\frac{p(s^i_{t_{k+1}}|s^i_{t_k};\theta)}{q(s^i_{t_{k+1}}|s^i_{t_k},o_{t_{1:k}};\phi)}$.

Continuous-discrete DPFs: model design

Conditional Normalising Flow-based Measurement Model[6].

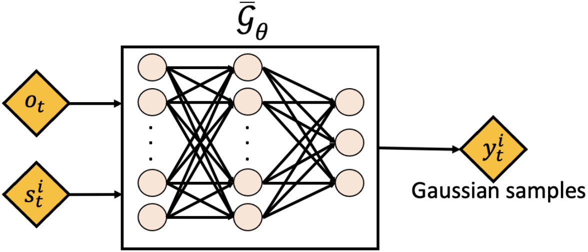


1. Map the observation o_t to a variable y_t^i that follows a Gaussian distribution $p_Y(\cdot)$ through the conditional normalising flow $\bar{\mathcal{G}}_\theta$.

[6] Chen et al., "Conditional Measurement Density Estimation in Sequential Monte Carlo via Normalising Flow", EUSIPCO, 2022.

Continuous-discrete DPFs: model design

Conditional Normalising Flow-based Measurement Model[6].



1. Map the observation o_t to a variable y_t^i that follows a Gaussian distribution $p_Y(\cdot)$ through the conditional normalising flow $\bar{\mathcal{G}}_\theta$.

2. Compute the likelihood of observations o_t given state s_t using the change of variable formula: $p(o_t|s_t^i;\theta) = p_Y(\bar{\mathcal{G}}_\theta(o_t, s_t^i)) \left| \det \frac{\partial \bar{\mathcal{G}}_\theta(o_t, s_t^i)}{\partial o_t} \right|$.

[6] Chen et al., "Conditional Measurement Density Estimation in Sequential Monte Carlo via Normalising Flow", EUSIPCO, 2022.

Continuous-discrete DPFs: model design

An overview

- Construct dynamic process and proposal process with neural SDEs.
  - Dynamic process: $\mathrm{d}s_t = f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.
  - Proposal process: $\mathrm{d}s_t = g_\phi(s_t, o_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.

f_θ, g_ϕ, and σ_θ are neural networks.

[13] Corenflos et al. "Differentiable Particle Filtering via Entropy-regularized Optimal Transport." ICML, 2021.

Continuous-discrete DPFs: model design

An overview

- Construct dynamic process and proposal process with neural SDEs.
  - Dynamic process: $\mathrm{d}s_t = f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.
  - Proposal process: $\mathrm{d}s_t = g_\phi(s_t, o_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.

  f_θ, g_ϕ, and σ_θ are neural networks.

- Conditional normalising flow-based measurement model.
  - $p(o_t|s_t^i;\theta) = p_Y(\bar{\mathcal{G}}_\theta(o_t, s_t^i))\left|\det\frac{\partial\bar{\mathcal{G}}_\theta(o_t, s_t^i)}{\partial o_t}\right|$.

[13] Corenflos et al. "Differentiable Particle Filtering via Entropy-regularized Optimal Transport." ICML, 2021.

Continuous-discrete DPFs: model design

An overview

- Construct dynamic process and proposal process with neural SDEs.
  - Dynamic process: $\mathrm{d}s_t = f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.
  - Proposal process: $\mathrm{d}s_t = g_\phi(s_t, o_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.

  f_θ, g_ϕ, and σ_θ are neural networks.
- Conditional normalising flow-based measurement model.
  - $p(o_t|s_t^i;\theta) = p_Y(\bar{\mathcal{G}}_\theta(o_t, s_t^i))\left|\det\frac{\partial\bar{\mathcal{G}}_\theta(o_t, s_t^i)}{\partial o_t}\right|$.
- Update weights with the unbiased estimator $Z(t_k, t_{k+1}; \omega_{k+1}^i)$.
  - $w_{t_{k+1}}^i \propto w_{t_k}^i p(o_{t_{k+1}}|s_{t_{k+1}}^i;\theta) Z(t_k, t_{k+1}; \omega_{k+1}^i)$.

[13] Corenflos et al. "Differentiable Particle Filtering via Entropy-regularized Optimal Transport." ICML, 2021.

Continuous-discrete DPFs: model design

An overview

- Construct dynamic process and proposal process with neural SDEs.
  - Dynamic process: $\mathrm{d}s_t = f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.
  - Proposal process: $\mathrm{d}s_t = g_\phi(s_t, o_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t$.

  f_θ, g_ϕ, and σ_θ are neural networks.

- Conditional normalising flow-based measurement model.
  - $p(o_t|s_t^i;\theta) = p_Y(\bar{\mathcal{G}}_\theta(o_t, s_t^i))\left|\det\frac{\partial\bar{\mathcal{G}}_\theta(o_t, s_t^i)}{\partial o_t}\right|$.
- Update weights with the unbiased estimator $Z(t_k, t_{k+1}; \omega_{k+1}^i)$.
  - $w_{t_{k+1}}^i \propto w_{t_k}^i p(o_{t_{k+1}}|s_{t_{k+1}}^i;\theta) Z(t_k, t_{k+1}; \omega_{k+1}^i)$.
- Differentiable resampling.
  - Entropy-regularised optimal transport resampling[13].

[13] Corenflos et al. "Differentiable Particle Filtering via Entropy-regularized Optimal Transport." ICML, 2021.

Continuous-discrete DPF: training objective

End-to-End learning by minimising a given loss function:

1. Supervised loss;
   - The mean squared error (MSE):
     $$L_{MSE}(\theta,\phi)=\frac{1}{K}\sum_{k=1}^{K}(s^*_{t_k}-s_{t_k})^T(s^*_{t_k}-s_{t_k})\,,$$
     $s^*_{t_k}$: ground truth state, s_{t_k}: estimated states.
2. Unsupervised loss.
   - SMC evidence lower bound:
     $$L_{\mathsf{ELBO}}(\theta,\phi)=\mathbb{E}\Big[\log\ \hat{p}_{N_p}(o_{t_{1:K}};\theta)\Big]\,,$$
     $\hat{p}(o_{1:t};\theta)$: an estimate of the marginal likelihood computed with particle weights.

Continuous-discrete DPF: filtering

A set of observed time instances $\{t_1, \cdots, t_K\}$ with observations $\{o_{t_1}, \cdots, o_{t_K}\}$.

Initialisation: Draw $\{s^i_{t_0}\}_{i=1}^{N_p}$ from $p(s_{t_0})$.
Set $\{\tilde{w}^i_{t_0} = \frac{1}{N_p}\}_{i=1}^{N_p}$.

for $k = 1$ to K:
  for $i = 1$ to N_p:
    Draw $s^i_{t_k}$ and estimate $Z(t_{k-1}, t_k; \omega^i_k)$ by solving Equation (1).
    Update weights: $w^i_{t_k} = w^i_{t_{k-1}} p(o_{t_k} | s^i_{t_k}; \theta) Z(t_{k-1}, t_k; \omega^i_k)$.
  end for
  for $i = 1$ to N_p:
    Normalise weights: $\tilde{w}^i_{t_k} = w^i_{t_k} / \sum_{j=1}^{N_p} w^j_{t_k}$.
  end for
  if ESS $<$ threshold:
    Resample $\{s^i_{t_k}, \tilde{w}^i_{t_k}\}_{i=1}^{N_p}$ to obtain $\{s^i_{t_k}, \frac{1}{N_p}\}_{i=1}^{N_p}$.
  end if
end for

Continuous-discrete DPF: prediction

How to approximate $p(s_{t'}|o_{t_{1:k}};\theta)$ at an arbitrary time instance t'?

- Assume $t' = t_k + \Delta t$.

$$p(s_{t_k+\Delta t}|o_{t_{1:k}};\theta) = \int p(s_{t_{1:k}}, s_{t_k+\Delta t}|o_{t_{1:k}};\theta)\mathrm{d}s_{t_{1:k}}$$
$$= \int p(s_{t_{1:k}}|o_{t_{1:k}};\theta)p(s_{t_k+\Delta t}|s_{t_k};\theta)\mathrm{d}s_{t_{1:k}}\,.$$

- Draw samples $\{s^i_{t_k+\Delta t}\}_{i=1}^{N_p}$ from the dynamic process:

$$\mathrm{d}s_t = f_\theta(s_t, t)\mathrm{d}t + \sigma_\theta(s_t, t)\mathrm{d}B_t\,.$$

- Approximate $p(s_{t_k+\Delta t}|o_{t_{1:k}};\theta) \approx \frac{1}{N_p}\sum_{i=1}^{N_p} s^i_{t_k+\Delta t}$.

Experiments

Two sets of experiments:

- Sequential state estimation (supervised training).
  - Benes-Daum filtering problem[14].
  - Angular position prediction in a noisy physical pendulum model[15].
- Observation prediction (unsupervised training).
  - Geometric Brownian motion[4].
  - Stochastic Lorenz attractor[3].

[3] Li et al., "Scalable gradients for stochastic differential equations", AISTATS, 2020.

[4] Deng et al., "Continuous Latent Process Flows", NeurIPS, 2021.

[14] Daum "Exact finite-dimensional nonlinear filters", IEEE Trans. Automatic Control, 1986.

[15] Sarkka, "Recursive Bayesian inference on stochastic differential equations", PhD thesis, Helsinki University of Technology, 2006.

Experiments: Benes-Daum filtering problem

Benes-Daum filtering problem:

$$\mathrm{d}s_t = \tanh(s_t)\mathrm{d}t + \mathrm{d}B_t \,,$$
$$o_t \sim \mathcal{N}(s_t, \sigma^2)\,.$$

- Estimate s_t at time t_k given observations $\{o_{t_0}, o_{t_1}, \cdots, o_{t_k}\}$.
- Time stamps are sampled from a homogeneous Poisson process with intensity $\lambda \in \{0.5\,, 2\}$ in the interval [0, 10].

Experiments: Benes-Daum filtering problem

Experimental results:

- Evaluated methods are compared by computing the RMSEs between their estimation of latent states and the ground-truth latent state.

Method	Poisson intensity	
	λ=0.5	λ=2
CD-DPF (ours)	**1.33**	**0.689**
Latent SDE[3]	1.48	0.846
CLPF[4]	1.51	0.879
CTPF[5]	1.46	0.724

Filtering RMSE.

[3] Li et al., "Scalable gradients for stochastic differential equations", AISTATS, 2020.

[4] Deng et al., "Continuous Latent Process Flows", NeurIPS, 2021

[5] Deng et al., "Continuous-time Particle Filtering for Latent Stochastic Differential Equations", arXiv, 2209.00173.

Experiments: angular position prediction

Noisy pendulum system:

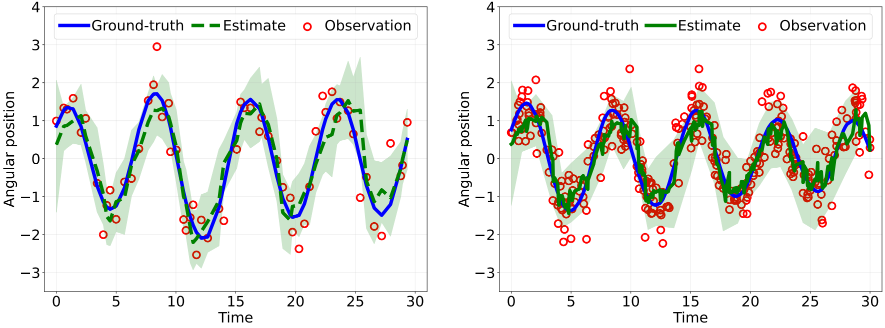
$$\mathrm{d}s_t(1) = s_t(2)\mathrm{d}t\,,$$
$$\mathrm{d}s_t(2) = -a^2\sin(s_t(1))\mathrm{d}t + b^{\frac{1}{2}}\mathrm{d}B_t\,,$$
$$o_t|s_t \sim \mathcal{N}(s_t(1), \sigma^2)\,.$$

- Predict the angular position $s_{t_{k+1}}(1)$ of the pendulum at the next time stamp t_{k+1} given observations $\{o_{t_0}, o_{t_1}, \cdots, o_{t_k}\}$.
- Time stamps are sampled from a homogeneous Poisson process with intensity $\lambda \in \{2.0\,, 10.0\}$ in the interval [0, 30].

Experiments: angular position prediction

Experimental results:

- Shaded areas specify the 95% quantiles of empirical distributions given by the CD-DPF.



[3] Li et al., "Scalable gradients for stochastic differential equations", AISTATS, 2020.

[4] Deng et al., "Continuous Latent Process Flows", NeurIPS, 2021

[5] Deng et al., "Continuous-time Particle Filtering for Latent Stochastic Differential Equations", arXiv, 2209.00173.

Experiments: angular position prediction

Experimental results:

Method	Poisson intensity	
	λ=2.0	λ=10.0
CD-DPF (ours)	**0.487**	**0.451**
Latent SDE[3]	0.749	0.612
CLPF[4]	0.622	0.587
CTPF [5]	0.501	0.489

Filtering RMSE.

[3] Li et al., "Scalable gradients for stochastic differential equations", AISTATS, 2020.

[4] Deng et al., "Continuous Latent Process Flows", NeurIPS, 2021

[5] Deng et al., "Continuous-time Particle Filtering for Latent Stochastic Differential Equations", arXiv, 2209.00173.

Experiments: observation prediction and interpolation

- Geometric Brownian motion

$$\mathrm{d}s_t = \mu s_t \mathrm{d}t + \sigma s_t \mathrm{d}B_t \,,$$
$$s_{t_0} = s_0 \,,$$

- Stochastic Lorenz Attractor

$$\mathrm{d}s_t(1) = \eta\Big(s_t(2) - s_t(1)\Big)\mathrm{d}t + \alpha(1)\mathrm{d}B_t, \quad s_{t_0}(1) = x_0(1)\,,$$
$$\mathrm{d}s_t(2) = \Big(s_t(1)(\rho - s_t(3)) - s_t(2)\Big)\mathrm{d}t + \alpha(2)\mathrm{d}B_t, \quad s_{t_0}(2) = x_0(2)\,,$$
$$\mathrm{d}s_t(3) = \Big(s_t(3)s_t(2) - \beta s_t(3)\Big)\mathrm{d}t + \alpha(3)\mathrm{d}B_t, \quad s_{t_0}(3) = x_0(3)\,.$$

Both experiments have Gaussian measurements:

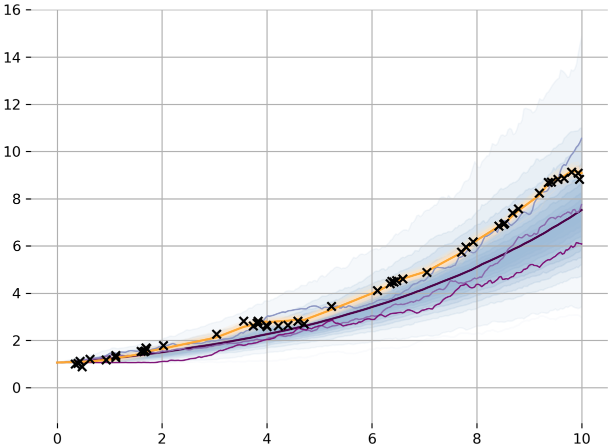
$$o_t \sim \mathcal{N}(s_t, \sigma^2)$$

.

Experiments: observation prediction and interpolation

Geometric Brownian motion experimental results:

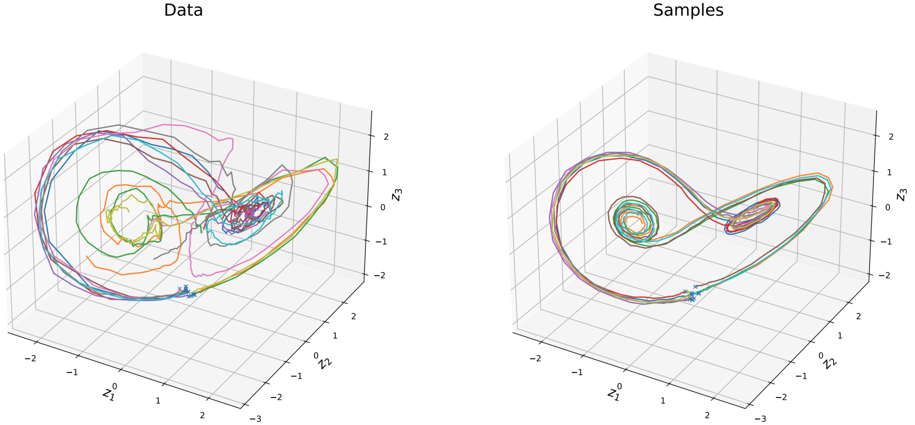
- Yellow curve: prediction. Black cross: observations. Purple and blue: training samples.



Experiments: observation prediction and interpolation

Stochastic Lorenz Attractor experimental results:

- We can discover the true system model from corrupted data.



Some unresolved problems

- There are a number of unbiased estimators of the weights.
  - Unclear which one is better in what cases.

Some unresolved problems

- There are a number of unbiased estimators of the weights.
  - Unclear which one is better in what cases.
- Exact simulation is required.
  - No longer unbiased estimates if approximation or discretisation is needed in simulation.
  - Exact simulation techniques only apply to very limited and simple SDEs.

Some unresolved problems

- There are a number of unbiased estimators of the weights.
  - Unclear which one is better in what cases.
- Exact simulation is required.
  - No longer unbiased estimates if approximation or discretisation is needed in simulation.
  - Exact simulation techniques only apply to very limited and simple SDEs.
- System identification problem in unsupervised training.
  - Different latent dynamics and measurements can possibly interpret observations equally well but some may produce poor filtering results.

Some unresolved problems

- There are a number of unbiased estimators of the weights.
  - Unclear which one is better in what cases.

- Exact simulation is required.
  - No longer unbiased estimates if approximation or discretisation is needed in simulation.
  - Exact simulation techniques only apply to very limited and simple SDEs.

- System identification problem in unsupervised training.
  - Different latent dynamics and measurements can possibly interpret observations equally well but some may produce poor filtering results.

Thank you!

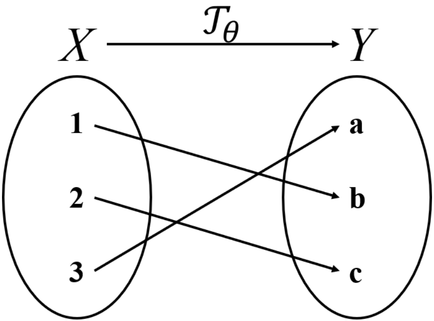
Appendix

Normalising Flows

Definition of normalising flows:

$$y = \mathcal{T}_\theta(x),$$

where $\mathcal{T}_\theta$ is required to be an invertible transformation.



Normalising Flows

Definition of normalising flows:

$$y = \mathcal{T}_\theta(x),$$

where $\mathcal{T}_\theta$ is required to be an invertible transformation.

Why invertible transformations?

- Invertibility allows density estimation (change of variable):

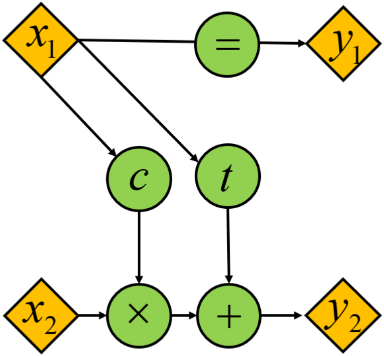
$$p(y) = p(x)\left|\mathsf{det}\frac{dy}{dx}\right|^{-1}$$

An Example of Normalising Flow: Coupling Layer[16]

Real-NVP

- Coupling layers.

$$x = [x_1,\ x_2] \qquad y = [y_1,\ y_2]$$



[15] Ding et al. "Density Estimation Using Real NVP", ICLR, 2017.

An Example of Normalising Flow: Coupling Layer[16]

Real-NVP

- Coupling layers.

The special structure of coupling layers leads to triangular Jacobian matrix:

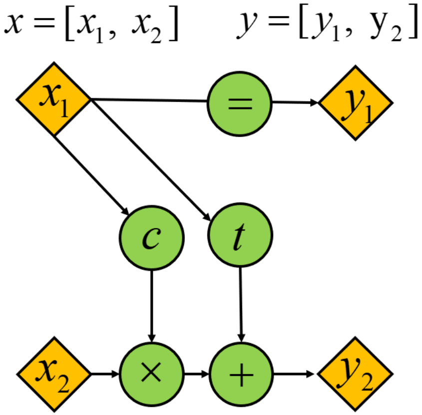
$$
\begin{aligned}
\underset{1:d}{y} &= \underset{1:d}{x} \\
\underset{d+1:D}{y} &= \underset{d+1:D}{x} \odot \exp(c(\underset{1:d}{x})) + t(\underset{1:d}{x})
\end{aligned}
$$

$$
\frac{\partial y}{\partial x} = \begin{bmatrix} \mathbb{I} & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & \mathsf{diag}(\exp[c(x_{1:d})]) \end{bmatrix}
$$

[15] Ding et al. "Density Estimation Using Real NVP", ICLR, 2017.

Conditional Coupling Layer

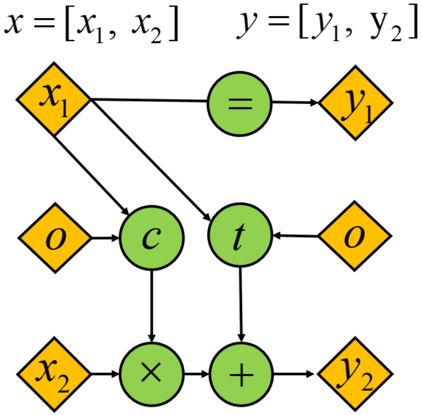
We use conditional coupling layer to construct conditional Real-NVP:



Standard coupling layer

Conditional Coupling Layer

We use conditional coupling layer to construct conditional Real-NVP:



Conditional coupling layer

Conditional Coupling Layer

- Conditional coupling layer:

$$\underset{1:d}{y} = \underset{1:d}{x}$$
$$\underset{d+1:D}{y} = \underset{d+1:D}{x} \odot \exp(c(\underset{1:d}{x}, o)) + t(\underset{1:d}{x}, o)$$

- Standard coupling layer:

$$\underset{1:d}{y} = \underset{1:d}{x}$$
$$\underset{d+1:D}{y} = \underset{d+1:D}{x} \odot \exp(c(\underset{1:d}{x})) + t(\underset{1:d}{x})$$

Conditional Coupling Layer

- Conditional coupling layer:

$$\underset{1:d}{y} = \underset{1:d}{x}$$
$$\underset{d+1:D}{y} = \underset{d+1:D}{x} \odot \exp(c(\underset{1:d}{x}, o)) + t(\underset{1:d}{x}, o)$$

Still invertible and lead to triangular Jacobian matrix:

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \mathbb{I} & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}} & \mathsf{diag}(\exp[c(x_{1:d}, o)]) \end{bmatrix}$$