Sampling meets machine learning

Hai-Dang Dau

4 Dec 2024

Table of Contents

Table of Contents

What is sampling

- $\pi(x)$: probability distribution in $\mathbb{R}^d$

$$\pi(x) = \frac{\gamma(x)}{Z}$$

  Z is an unknown normalizing constant

- **Sampling**: produce $X_1, \dots, X_N$ from $\pi(x)$

- Density estimation: estimate $\pi(x)$ from $X_1, \dots, X_N$

Examples of sampling

- **Bayesian inference**

$$\text{posterior}(x) = \frac{\text{prior}(x) \times \text{likelihood}(\text{data}|\text{x})}{\text{model evidence}}$$

- **Optimization**: Find

$$\arg\min_{x \in \mathbb{R}^d} f(x)$$

Consider

$$\pi_\lambda(x) = \frac{e^{-\lambda f(x)}}{Z_\lambda}$$

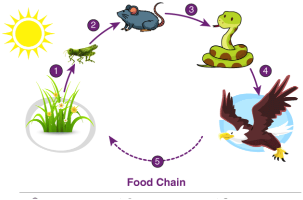
When $\lambda \to \infty$, the distribution concentrates on $\arg\min f(x)$. Simulated annealing.

An example of sampling: State-space models

$$\begin{array}{ccccccc} X_0 & \to & X_1 & \to & \cdots & \to & X_T \\ \downarrow & & \downarrow & & & & \downarrow \\ Y_0 & & Y_1 & & \cdots & & Y_T \end{array}$$

- $(X_0, \ldots, X_T)$ latent Markov chain
- $(Y_0, \ldots, Y_T)$ noisy observations, i.e. $Y_t = X_t + \varepsilon_t$
- Recover the hidden states: $p(x_{0:T} | y_{0:T})$
- Predict the future: $p(x_{T+1} | y_{0:T})$

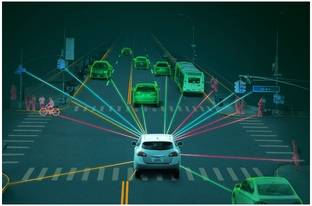
Examples of state-space models



Ecology

Finance

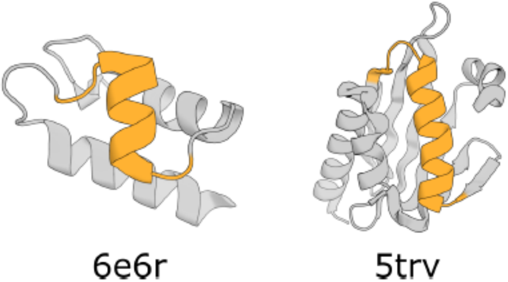
Neuroscience



Tracking and navigation

Modern inverse problems: Image recovery

Protein generation[1]



[1]Image taken from Trippe, B. L. et al. (2023). *Diffusion probabilistic modeling of protein backbones in 3D for the motif-scaffolding problem.* ICLR

Large language models[2]

$$\begin{array}{ccccccc} X_0 & \rightarrow & X_1 & \rightarrow & \cdots & \rightarrow & X_T \\ \downarrow & & \downarrow & & & & \downarrow \\ Y_0 & & Y_1 & & \cdots & & Y_T \end{array}$$

- $X_0, X_1, \ldots, X_T$ are sequential outputs of an LLM
- We want to generate contents satisfying a certain condition, e.g. non-harmful contents
- $Y_t = \mathbb{1}_{X_t \text{ harmful}}$
- Problem: Sample from $X_{0:T}$ given $Y_t = 0, \forall t$

[2] Zhao, S. et al (2024). *Probabilistic inference in language models via twisted sequential Monte Carlo*. ICML

Challenges for sampling

Common solution: MCMC

- Hard to initialize (burn-in)
- Hard to tune (covariance matrices, proposal magnitude, etc.)
- Sequential by nature

Table of Contents

Sequential Monte Carlo (SMC) Samplers[3]

- Core idea: $\pi(x)$ is difficult to sample from
- Consider a sequence of distributions $\pi_0, \ldots, \pi_T$
  - π_0 is an easy distribution e.g. Gaussian
  - $\pi_T \equiv \pi$, the target
- In state-space models:

$$\pi_t(x_t) = p(x_t | y_{0:t})$$

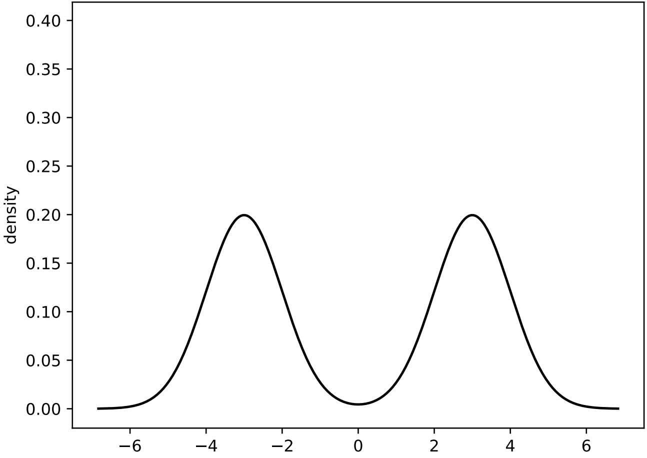
- More generally, geometric sequence:

$$\pi_t(x_t) = \frac{\pi_0(x_t)^{1-\lambda_t} \pi(x_t)^{\lambda_t}}{Z_t}$$

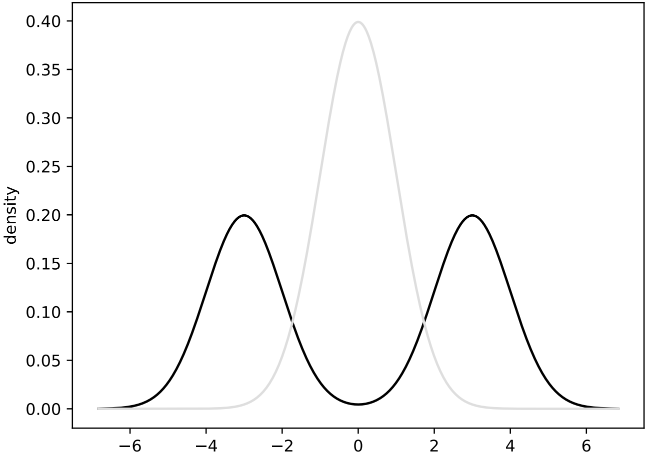
where $0 = \lambda_0 < \lambda_1 < \ldots < \lambda_T = 1$

[3]Del Moral et al (2006) *Sequential Monte Carlo samplers.* JRSS B

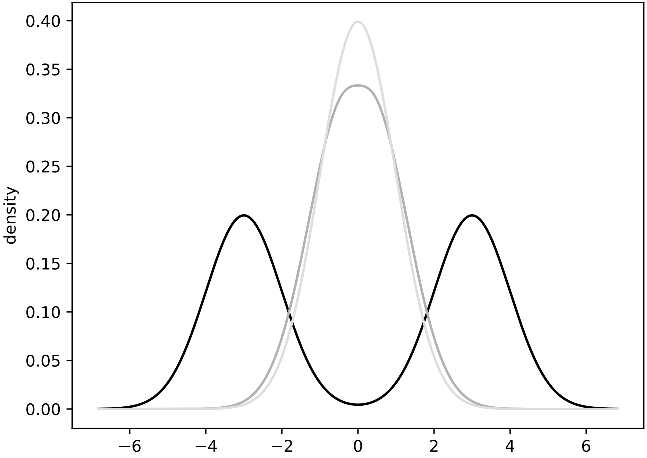
Sequence visualization



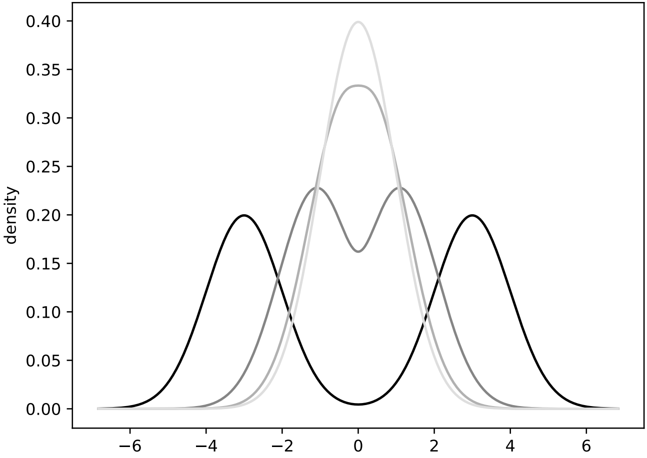
Sequence visualization



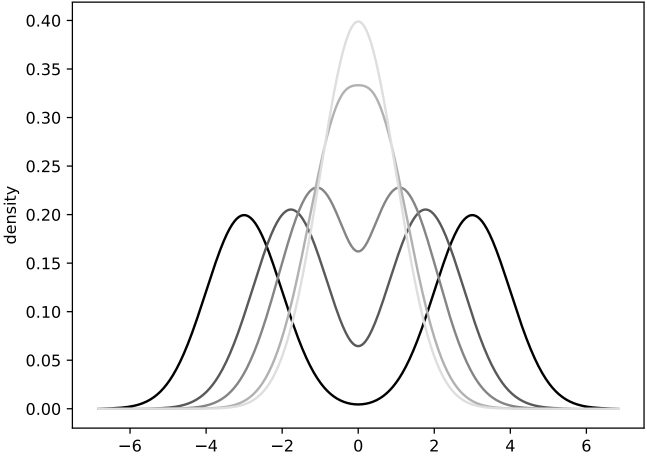
Sequence visualization



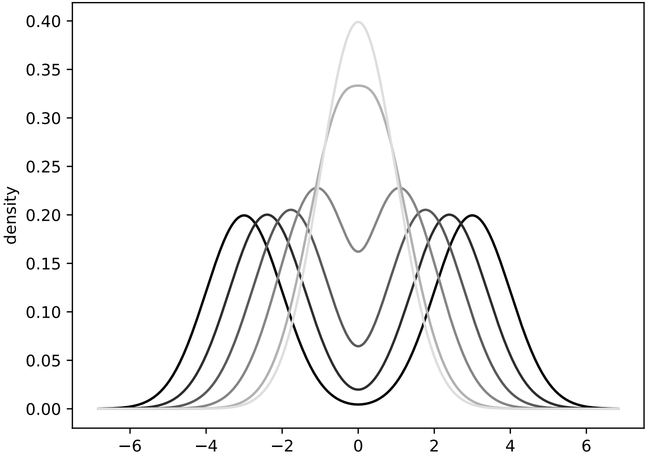
Sequence visualization



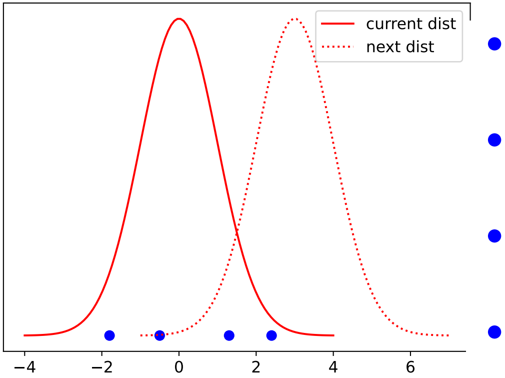
Sequence visualization



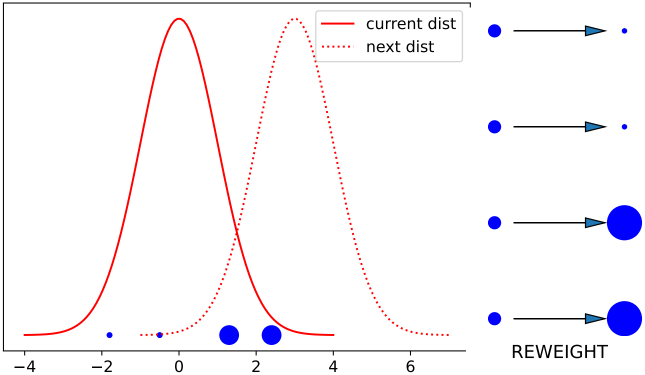
Sequence visualization



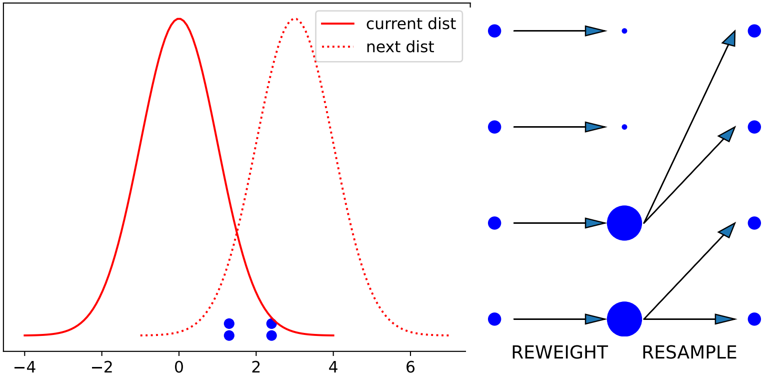
SMC explained



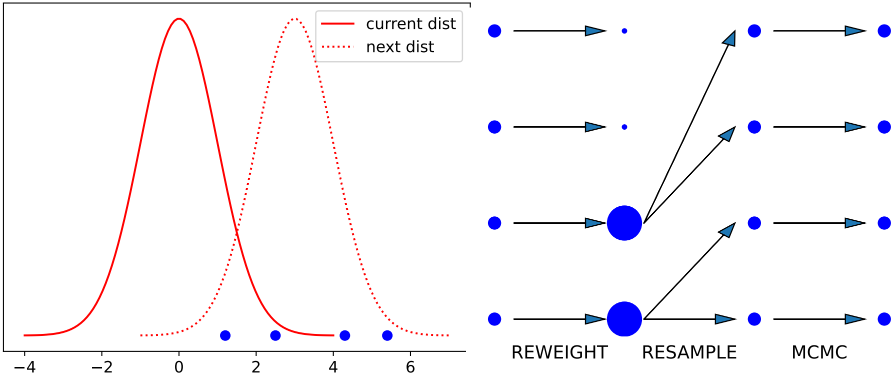
SMC explained



SMC explained



SMC explained



SMC versus MCMC

- No burn-in required (the particles are already at stationarity)
- Easier parameter tuning (using the current sample)
- Give estimate of model evidence (aka normalizing constant/partition function)
- **Massively** parallelizable (in particular with GPU)

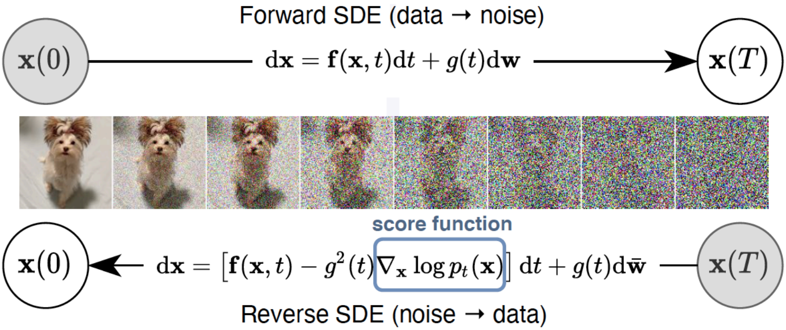
Table of Contents

What is generative modelling?

- Density estimation: given $X_1, \ldots, X_N$; estimate $\pi(x)$
- Generative modelling: given $X_1, \ldots, X_N$; produce $X_{N+1}, X_{N+2}, \ldots$
- Conceptually very similar
- Main application: use the modelled distribution as a **prior**

These problems need very good prior distributions!

Diffusion models[4]



[4]Figure taken from Song et al (2021) *Score-based Generative Models through Stochastic Differential Equations*, ICLR 2021

From diffusions to sampling

- Noising mechanism creates a bridge between a difficult distribution and the Gaussian distribution
- This bridge is different from the geometric sequence
- If score is learned perfectly, just run the reverse SDE to generate data. **No MCMC.**

Table of Contents

Problem one: MCMC

- SMC Samplers still rely extensively on Markov chain Monte Carlo (MCMC) kernels
- MCMC is known to have degraded performance in high-dimensional or multimodal settings

Our solutions:

- Use MCMC outputs more efficiently[5]
- Use diffusion-based generative models[6].

[5] Dau, H.-D. and Chopin, N. (2022). *Waste-free sequential Monte Carlo*. JRSS B

[6] Phillips, A., Dau, H.-D., Hutchinson, M. J., De Bortoli, V., Deligiannidis, G., and Doucet, A. (2024). *Particle denoising diffusion sampler*. ICML

Problem two: degeneracy

- SMC is a ‘genetic’ algorithm: each step includes ‘selection of the fittest’
- After some generations, all individuals at time t have the same ancestor at time 0
- Well-known phenomenon even outside of particle filter literature: Wright-Fisher model, Genetic drift, etc.

Illustration of degeneracy

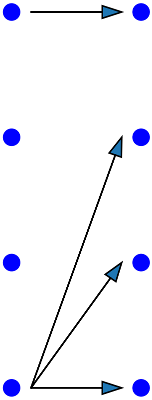


Illustration of degeneracy

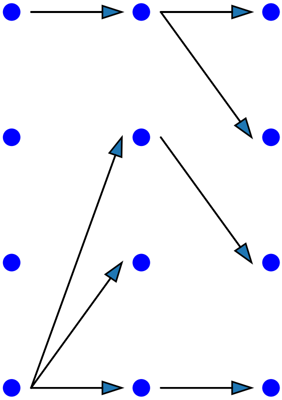


Illustration of degeneracy

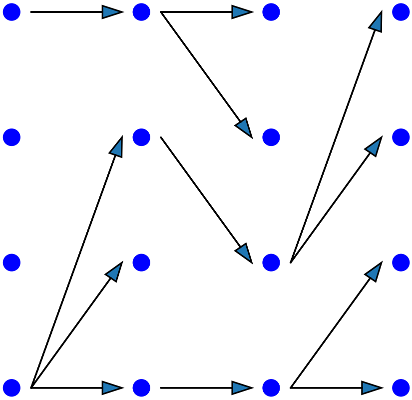


Illustration of degeneracy

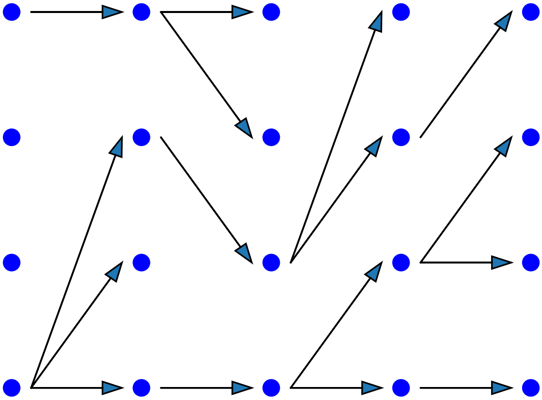


Illustration of degeneracy

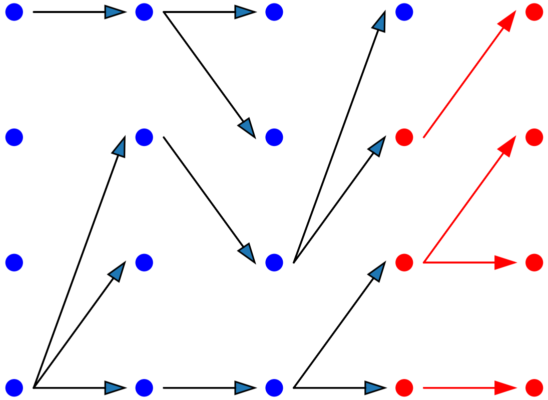


Illustration of degeneracy

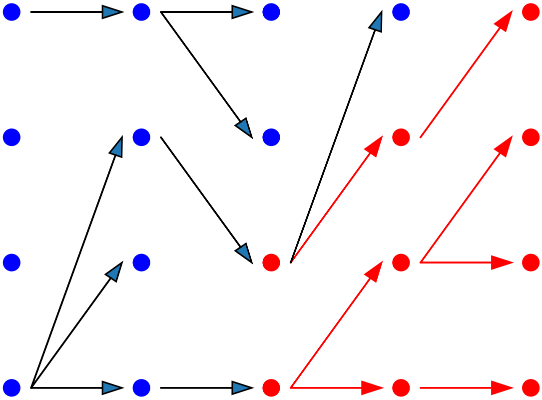


Illustration of degeneracy

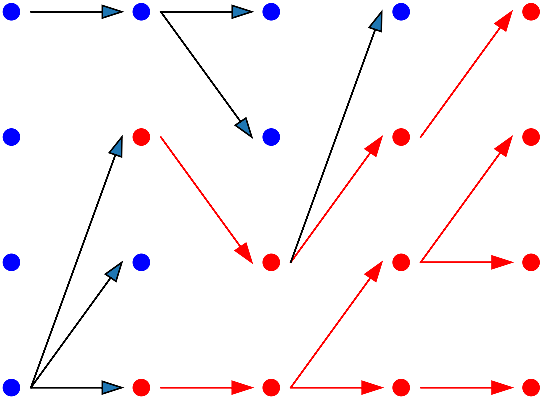
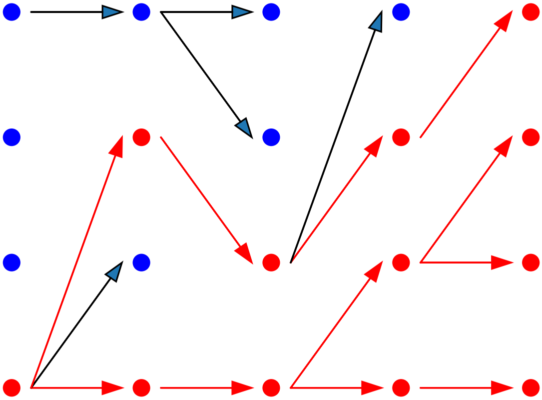


Illustration of degeneracy



Problem statement

- Degeneracy greatly compromises the accuracy of $p(x_t|y_{0:T})$ for any fixed t, when T becomes large
- We[7] analyse existing solutions and propose new ones

[7] Dau & Chopin (2023). *On backward smoothing algorithms.* Annals of Statistics

Table of Contents

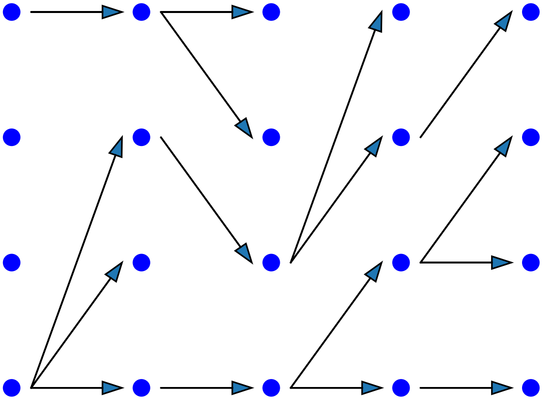
Table of Contents

Mathematical statement[8]

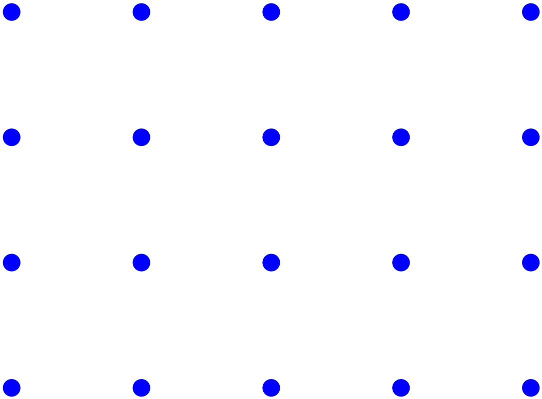
$$\mathbb{E}\left[\left(\int \varphi(x_t)\hat{p}(\mathrm{d}x_t|y_{0:T}) - \int \varphi(x_t)p(\mathrm{d}x_t|y_{0:T})\right)^2\right] = \mathcal{O}(T/N).$$

[8] Olsson & Westerborn (2017). *Efficient particle-based online smoothing in general hidden Markov models: the PaRIS algorithm*. Bernoulli

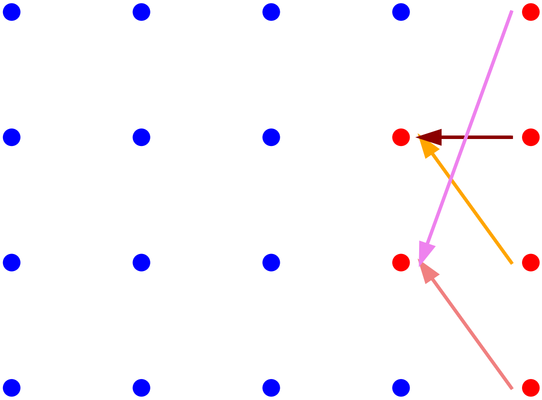
Backward sampling illustration



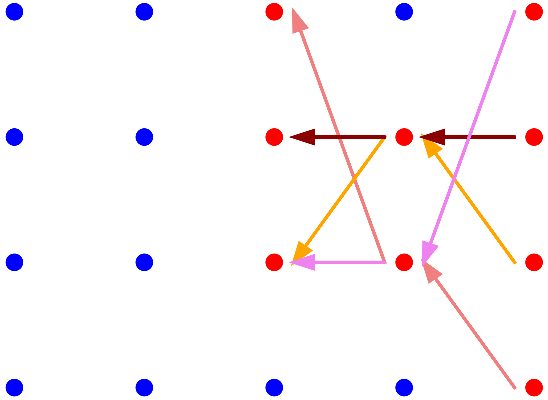
Backward sampling illustration



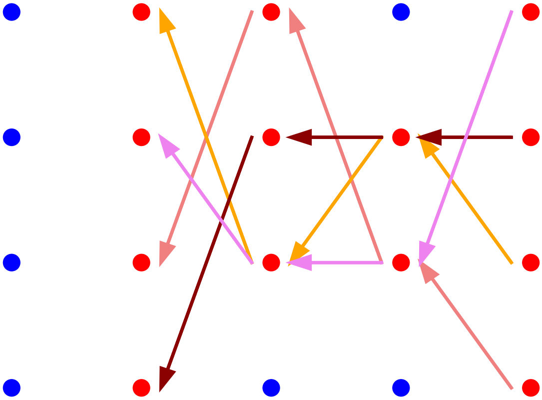
Backward sampling illustration



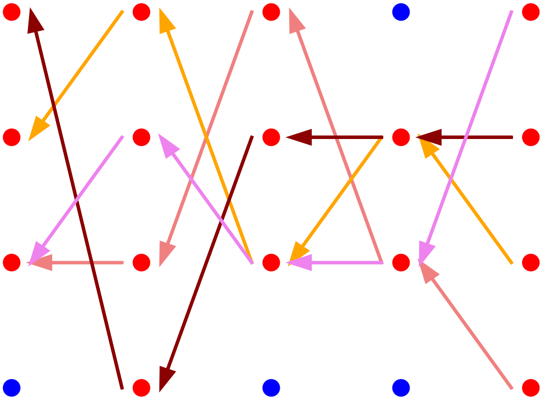
Backward sampling illustration



Backward sampling illustration



Backward sampling illustration



Backward sampling[9]

- Natural solution: find alternative ancestors for each particle, instead of merely follow information given by the forward pass
- Simulating the ancestor particle given the child particle is a discrete sampling problem in the space $\{1, \ldots, N\}$
- Reconstructing the ancestor for one particle takes $\mathcal{O}(N)$ computational time. Total cost $\mathcal{O}(N^2)$.

[9] Douc et al. (2010) *Sequential Monte Carlo smoothing for general state space hidden Markov models*. AoAP.

Cost reduction via discrete MCMC

- How to reduce this cost?
- Idea: we do not have to solve the backward sampling problem exactly. We can use MCMC instead, starting the MCMC chains from the forward ancestors
- How many MCMC iterations are needed? How to calibrate MCMC?
- Suprise answer: **One** MCMC step is enough to avoid degeneracy.

Formal statement

Theorem (Dau & Chopin (2023), AoS.)

Under appropriate hypotheses, backward sampling using MCMC has constant error rate:

$$\mathbb{E}\left[\left(\int \varphi(x_t)\hat{p}(\mathrm{d}x_t|y_{0:T}) - \int \varphi(x_t)p(\mathrm{d}x_t|y_{0:T})\right)^2\right] = \mathcal{O}(1/N)$$

as $T \to \infty$.

This enables the efficient inference for very large datasets.

Additive functions

Corollary (Dau & Chopin (2023), AoS.)

For additive functions $\psi(x_{0:T})$

$$\mathbb{E}\left[\left(\int \psi(x_{0:T})\hat{p}(\mathrm{d}x_{0:T}|y_{0:T}) - \int \psi(x_{0:T})p(\mathrm{d}x_{0:T}|y_{0:T})\right)^2\right] = \mathcal{O}(T/N).$$

Without backward sampling, the error is $\mathcal{O}(T^2/N)$.

Coupling

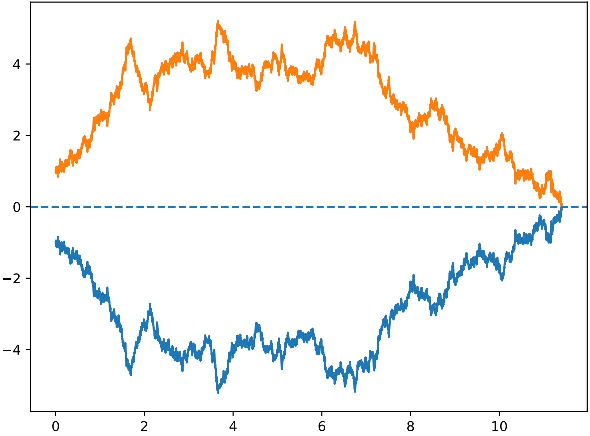
- What if a particle has more than one ancestor right from the *forward* pass? In that case, explicit backward sampling might not be necessary

- This can be realized using probabilistic *coupling*

- *Coupling* is especially helpful when backward sampling is not even possible

- By *coupling*, we mean joint distributions (X, Y) where $\mathrm{Law}(X) \neq \mathrm{Law}(Y)$ but $\mathbb{P}(X = Y) > 0$

Reflection coupling of two Brownian motions



Two **correlated** Brownian motions starting from 1 and -1. The noises are symmetric through the dashed reflection "plane"

Illustration of coupling

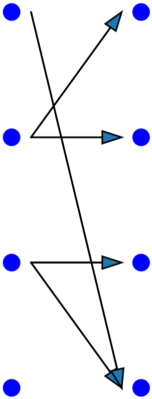


Illustration of coupling

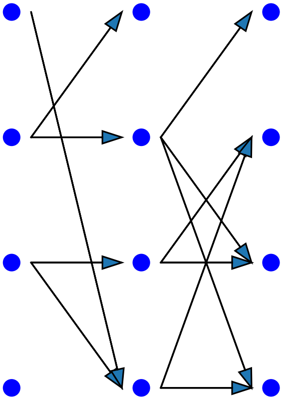


Illustration of coupling

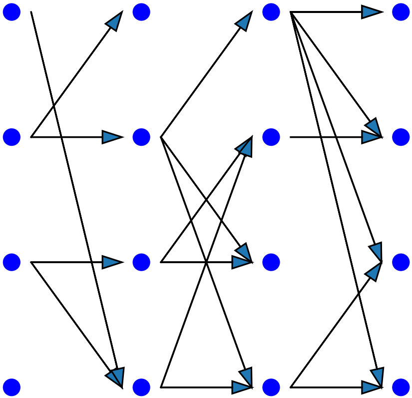


Illustration of coupling

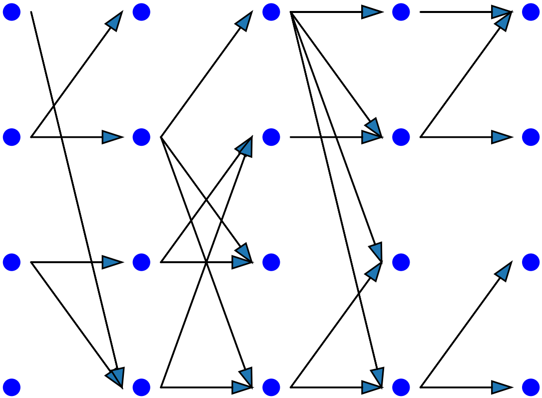


Illustration of coupling

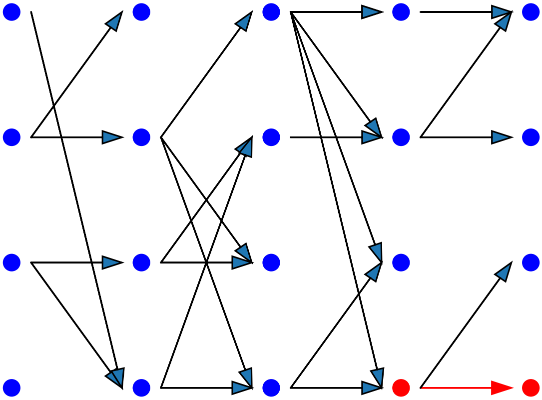


Illustration of coupling

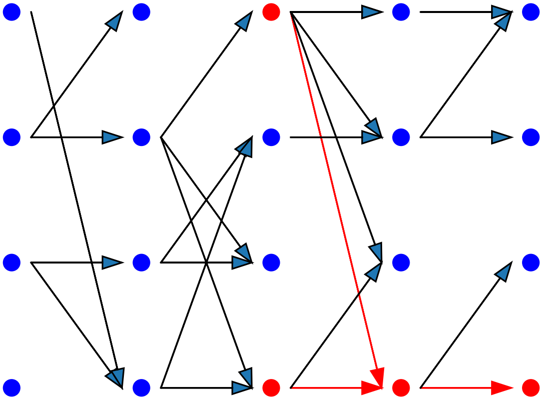


Illustration of coupling

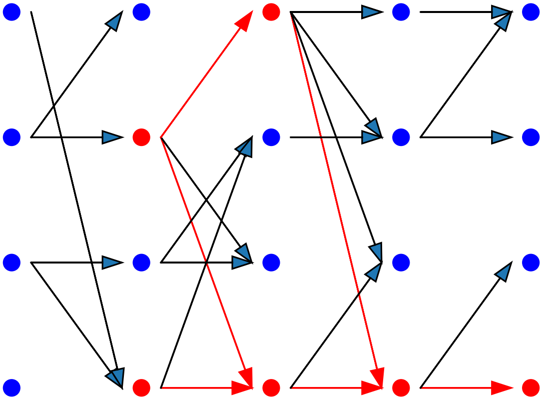
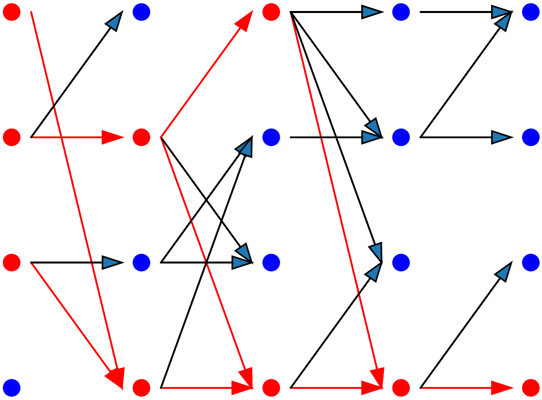


Illustration of coupling



Coupling of two stochastic processes

- $\mathrm{d}X_t^{\mathrm{A}} = b(X_t^{\mathrm{A}})\mathrm{d}t + \sigma(X_t^{\mathrm{A}})\mathrm{d}W_t^{\mathrm{A}}$
- $\mathrm{d}X_t^{\mathrm{B}} = b(X_t^{\mathrm{B}})\mathrm{d}t + \sigma(X_t^{\mathrm{B}})\mathrm{d}W_t^{\mathrm{B}}$
- If two Brownian motions are correlated via

$$\mathrm{d}W_t^{\mathrm{B}} = [\mathrm{Id} - 2u(X^{\mathrm{A}}, X^{\mathrm{B}})u(X^{\mathrm{A}}, X^{\mathrm{B}})^\top]\mathrm{d}W_t^{\mathrm{A}}$$

where

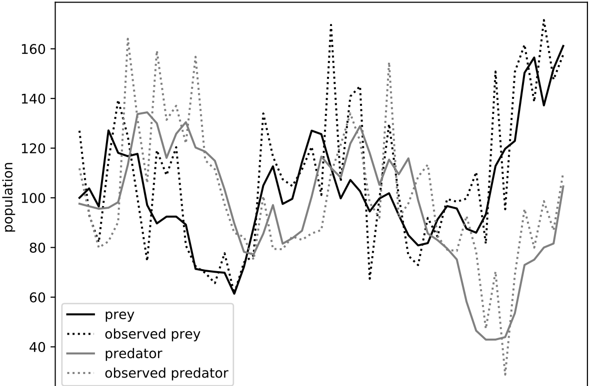
$$u(x, x') = \frac{\sigma(x')^{-1}(x - x')}{\|\sigma(x')^{-1}(x - x')\|_2}$$

then under regularity conditions[10], there exists $\tau > 0$ a.s. such that $X_\tau^{\mathrm{A}} = X_\tau^{\mathrm{B}}$

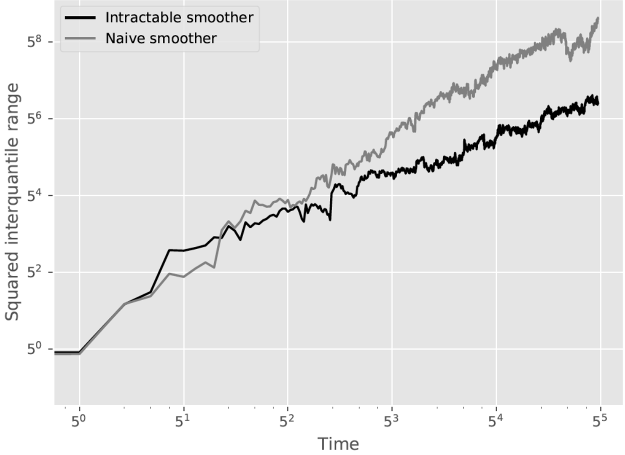
[10] Lindvall & Rogers (1986) Coupling of multidim diffusions by reflection. *AoP*

Lotka-Volterra stochastic differential equation

$$\begin{cases} \mathrm{d}X_t(0) = [\beta_0 X_t(0) - \frac{1}{2}\tau_0[X_t(0)]^2 \quad - \tau_1 X_t(0)X_t(1)]\mathrm{d}t + X_t(0)\mathrm{d}E_t(0) \\ \mathrm{d}X_t(1) = [\qquad -\beta_1 X_t(1) \qquad + \tau_1 X_t(0)X_t(1)]\mathrm{d}t + X_t(1)\mathrm{d}E_t(1) \end{cases}$$



Lotka-Volterra result



- Our method shines for large datasets ($\times 25$ accuracy)
- **It can handle SDE well**

Technical implementations

\+ =

- *Some* Python programs can be compiled via JAX to run on GPU
- Programs needed to be written in a *compilable* manner: all arrays must have fixed size
- In our algorithm, a particle has sometimes one, sometimes two ancestors (whether coupling of two stochastic processes is successful)
- Special attention is needed

High-dimensional dynamical systems

$$\begin{array}{ccccccc} X_0 & \to & X_1 & \to & \cdots & \to & X_T \\ \downarrow & & \downarrow & & & & \downarrow \\ Y_0 & & Y_1 & & \cdots & & Y_T \end{array}$$

- Backward sampling doesn't suffice
- The main obstacle is to get good proposal distributions in the first place
- We know $p(x_t|x_{t-1})$ by def., but

$$p(x_t|x_{t-1}, y_{0:T}) \neq p(x_t|x_{t-1}).$$

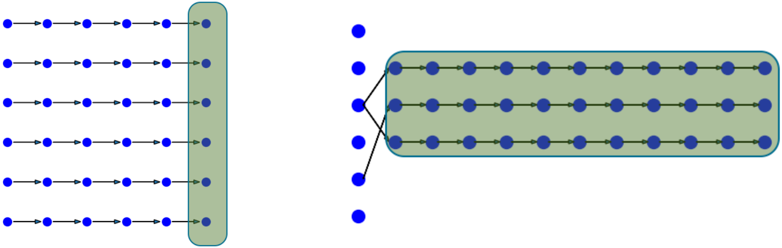
- Define

$$h_t(x_t) = \frac{p(x_t|x_{t-1}, y_{0:T})}{p(x_t|x_{t-1})}.$$

- There are several ways to learn this unknown *h-transform* using neural networks, none is fully satisfying

Table of Contents

Standard vs Waste-free[11] SMC



Does this change anything? A lot, both theoretically and practically!

[11] Dau, H.-D. and Chopin, N. (2022). *Waste-free sequential Monte Carlo.* JRSS B

Theoretical underpinnings

Theorem (Dau & Chopin (2022), JRSS B.)

The estimates produced by Waste-free SMC satisfy

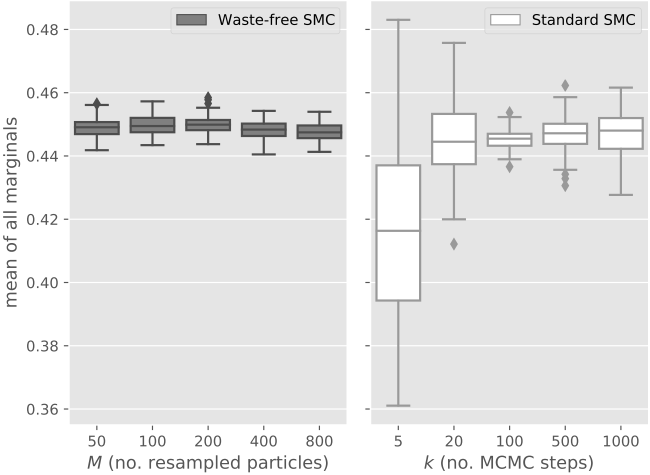
$$\mathbb{E}[\hat{Z}] = Z$$

$$\sqrt{N}(\log \hat{Z} - \log Z) \Rightarrow \mathcal{N}(0, \sum_{t=1}^{T} \sigma_t^2)$$

$$\sqrt{N}\left(\int \varphi(x)\hat{\pi}(\mathrm{d}x) - \int \varphi(x)\pi(\mathrm{d}x)\right) \Rightarrow \mathcal{N}(0, \sigma_\varphi^2)$$

Moreover, we provide estimators for the variances.

Stability of Waste-free SMC



Instability of Standard SMC

Proposition (Dau & Chopin (2022), JRSS B.)

Under simplified conditions, for ***standard SMC****, there exists a k_0 such that*

- *If the number of MCMC steps $k < k_0$, the error of standard SMC explodes exponentially when $T \to \infty$*
- *If the number of MCMC step $k \geq k_0$, the error of standard SMC is stable*

Applications

Waste-free SMC is now implemented in the Python packages **particles** (76 forks, 416 stars) and **BlackJAX** (106 forks, 850 stars).

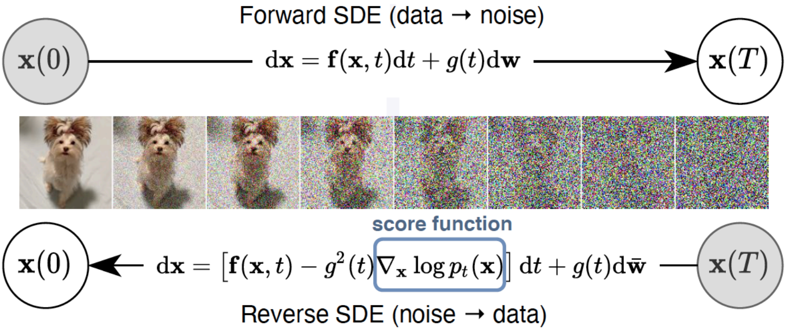
Applied to sample from stiffness field in human lung model[12] and to predict sudden cardiac deaths[13]

[12] Dinkel, M. et al. (2024). *Solving Bayesian inverse problems with expensive likelihoods using constrained Gaussian processes and active learning.* Inverse Problems.

[13] Youssfi, Y. and Chopin, N. (2024). *Scalable Bayesian bi-level variable selection in generalized linear models.* Foundations of Data Science.

Table of Contents

Illustration[14]



[14] Figure taken from Song et al (2021) *Score-based Generative Models through Stochastic Differential Equations*, ICLR 2021

Forward and backward processes[16]

$$\mathrm{d}X_t = -X_t \mathrm{d}t + \sqrt{2}\mathrm{d}W_t$$

$$Y_t = X_{T-t}$$

$$\mathrm{d}Y_t = \left[Y_t + 2\nabla \log \pi_{T-t}(Y_t)\right] \mathrm{d}t + \sqrt{2}\mathrm{d}B_t$$

- Backward process: it's not MCMC. It's much better than MCMC
- Mixing independent of dimension[15]

---

[15] De Bortoli et al (2021) *Diffusion Schrodinger bridge with applications to score-based generative modeling*. NeurIPS

[16] Haussmann & Pardoux (1986). *Time reversal of diffusions*. AoP.

Forward and backward processes[15]

$$\mathrm{d}X_t = -X_t \mathrm{d}t + \sqrt{2}\mathrm{d}W_t$$

$$Y_t = X_{T-t}$$

$$\mathrm{d}Y_t = [Y_t + 2\nabla \log \pi_{T-t}(Y_t)]\, \mathrm{d}t + \sqrt{2}\mathrm{d}B_t$$

- **Question 1.** How to learn the score $\nabla \log \pi_t$?
- **Question 2.** How to get consistent samples with approximate scores?

[15] Haussmann & Pardoux (1986). *Time reversal of diffusions.* AoP.

Diffusion-based SMC Samplers[16]

Proposal

$$q(x_t|x_{t+\varepsilon}) = \mathcal{N}(x_t|x_{t+\varepsilon} + 2\varepsilon x_{t+\varepsilon} + 2\varepsilon\nabla \log \hat{\pi}_{t+\varepsilon}(x_{t+\varepsilon}), 2\varepsilon)$$

Weight

$$\omega(x_t, x_{t+\varepsilon}) = \frac{\hat{\pi}_t(x_t)\pi(x_{t+\varepsilon}|x_t)}{\hat{\pi}_{t+\varepsilon}(x_{t+\varepsilon})q(x_t|x_{t+\varepsilon})}$$

[16] Phillips, Dau, Hutchinson, De Bortoli, Deligiannidis, and Doucet (2024). *Particle denoising diffusion sampler*. ICML

Learn the score

$$\begin{aligned}\nabla \log \pi_t(X_t) &= \mathbb{E}\left[\nabla \log \pi_t(X_t|X_0) \middle| X_t\right] \\ &= \mathbb{E}\left[-\frac{X_t - \sqrt{1-\lambda_t}X_0}{\lambda_t} \middle| X_t\right]\end{aligned}$$

- $\nabla \log \pi_t(x_t) \approx s_\theta(t, x_t)$ neural network parametrized by θ
- Minimize $\mathbb{E}_\pi \left[\left\| s_\theta(t, X_t) + \frac{X_t - \sqrt{1-\lambda_t}X_0}{\lambda_t} \right\|^2\right]$ by stochastic gradient descent

Learn the score (next)

- We don't have samples from π
- We first run the sampler with an initial approximation $\hat{\pi}_t$
- Use the output sample to learn a better $\hat{\pi}_t$
- Rinse and repeat

Theoretical guarantees[17]

Theorem

The produced estimate satisfies

$$\mathbb{E}\left[\left(\frac{\hat{Z}_{N,\varepsilon}}{Z} - 1\right)^2\right] \lesssim \frac{\sigma_\varepsilon^2}{N}$$

where

$$\limsup_{\varepsilon \to 0} \sigma_\varepsilon^2 \leq \int_0^T \mathbb{E}\left[\frac{\hat{\pi}_t(X_t)}{\pi_t(X_t)} \|\nabla \log \hat{\pi}_t(X_t) - \nabla \log \pi_t(X_t)\|^2\right] \mathrm{d}t$$

[17] Phillips, Dau, Hutchinson, De Bortoli, Deligiannidis, and Doucet (2024). *Particle denoising diffusion sampler*. ICML

Result interpretation

Consider the **path measures**

$$\pi(\mathrm{d}y_{[0:T]}): \quad \mathrm{d}Y_t = [Y_t + 2\nabla \log \pi_{T-t}(Y_t)]\,\mathrm{d}t + \sqrt{2}\mathrm{d}B_t$$
$$\hat{\pi}(\mathrm{d}y_{[0:T]}): \quad \mathrm{d}Y_t = [Y_t + 2\nabla \log \hat{\pi}_{T-t}(Y_t)]\,\mathrm{d}t + \sqrt{2}\mathrm{d}B_t$$

Then $\mathrm{KL}(\pi|\hat{\pi}) = \int_0^T \mathbb{E}_\pi \left[\|\nabla \log \pi_t - \nabla \log \hat{\pi}_t\|^2\right] \mathrm{d}t$

- The SMC error is $\approx \mathrm{KL}(\pi|\hat{\pi})$
- The error of direct importance sampling from $\hat{\pi}$ to π is $\chi^2(\pi|\hat{\pi}) \geq \exp\{\mathrm{KL}\} - 1$
- By breaking up naive importance sampling into multiple resampling/reweighting steps, SMC dramatically reduces the error
- First result of its kind for continuous SMC

Alternative objectives

- Score marginalization identity
$\nabla \log \pi_t(X_t) = \mathbb{E}\left[\nabla \log \pi(X_t|X_0) | X_t\right]$
- Training objective $\mathbb{E}_{\pi_{0,t}(x_0,x_t)}\left\|s_\theta(t,X_t) - \nabla \log \pi_{t|0}(X_t|X_0)\right\|^2$
- Problem: $\mathrm{Var}(\nabla \log \pi(X_t|X_0)|X_t)$ can be large
- Solution: control variates. We know that

$$\mathbb{E}\left[\nabla_{x_0} \log \pi_{0|t}(X_0|X_t)\middle| X_t\right] = 0$$

and

$$\nabla_{x_0} \log \pi_{0|t}(x_0|x_t) = \nabla_{x_0} \log \pi_0(x_0) + \nabla_{x_0} \log \pi_{t|0}(x_t|x_0)$$

is tractable

Alternative objectives (continued)

$$\nabla \log \pi_t(X_t) = \mathbb{E}\left[\nabla \log \pi(X_t|X_0) + \alpha \nabla_{x_0} \log \pi_{0|t}(X_0|X_t) \middle| X_t \right]$$

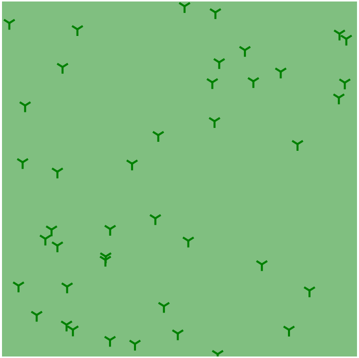
for any α. Choice of α.

- This equation gives a new objective that incorporates information from $\pi_0(x_0)$ that we know.

Numerical experiment: Log-Gaussian Cox Process

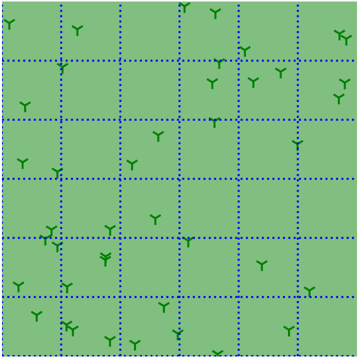
- State space: two dimensional lattice θ_{ij}
- Observations: $y_{ij}|\theta_{ij} \sim \text{Poisson}(e^{\theta_{ij}})$
- Prior: $\theta_{ij} \sim$ Gaussian Process
- High dimension: $M \times M$

Numerical experiment: Log-Gaussian Cox Process



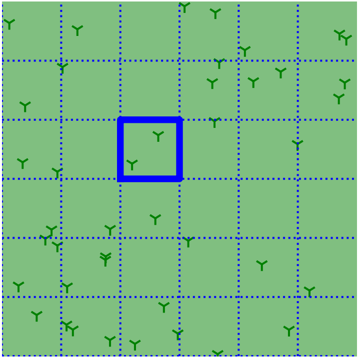
- State space: two dimensional lattice θ_{ij}
- Observations: $y_{ij}|\theta_{ij} \sim \text{Poisson}(e^{\theta_{ij}})$
- Prior: $\theta_{ij} \sim$ Gaussian Process
- High dimension: $M \times M$

Numerical experiment: Log-Gaussian Cox Process



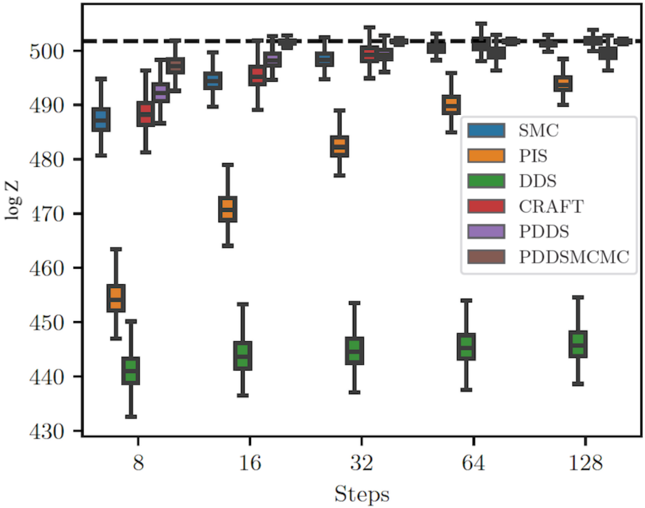
- State space: two dimensional lattice θ_{ij}
- Observations: $y_{ij}|\theta_{ij} \sim \text{Poisson}(e^{\theta_{ij}})$
- Prior: $\theta_{ij} \sim$ Gaussian Process
- High dimension: $M \times M$

Numerical experiment: Log-Gaussian Cox Process



- State space: two dimensional lattice θ_{ij}
- Observations: $y_{ij}|\theta_{ij} \sim \text{Poisson}(e^{\theta_{ij}})$
- Prior: $\theta_{ij} \sim$ Gaussian Process
- High dimension: $M \times M$

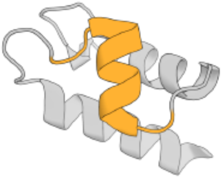
Result


500
490
480
470
460
450
440
430
log Z
8
16
32
64
128
Steps
SMC
PIS
DDS
CRAFT
PDDS
PDDSMCMC

Remarks

- More adapted training objectives for the sampling problem
- SMC is a parallel sampling paradigm. Another one is parallel tempering.
- How to integrate diffusion models into parallel tempering framework?

Discrete problems: Proteins



6e6r

5trv

Table of Contents

Summary

- We have defined the sampling problem and described some of its modern applications
- We have looked at how SMC is a natural framework to tackle sampling
- We have considered several improvements to vanilla SMC:
  - Using MCMC output more efficiently
  - Integrating diffusion-based generative models
  - Fighting degeneracy

Implementation

+ =

Special emphasis is put on:

- Parallelizable algorithms (in particular on GPU)
- Compilable implementations

Next steps

The sampling community shouldn't miss out on the opportunities offered by GPU parallelization and deep learning.

1. Build good and reusable **implementations** (e.g. BlackJAX).
2. Explain the gain via appropriate **theories**.
3. Build better **methodologies**.
4. Adapt algorithms to specific **applications**.

Next steps

The sampling community shouldn't miss out on the opportunities offered by GPU parallelization and deep learning.

1. Build good and reusable **implementations** (e.g. BlackJAX).
2. Explain the gain via appropriate **theories**.
   - Continuous-time SMC
   - Manifold hypothesis
3. Build better **methodologies**.
4. Adapt algorithms to specific **applications**.

Next steps

The sampling community shouldn't miss out on the opportunities offered by GPU parallelization and deep learning.

1. Build good and reusable **implementations** (e.g. BlackJAX).

2. Explain the gain via appropriate **theories**.

3. Build better **methodologies**.
   - Algorithm architectures (parallel tempering, simulated tempering, etc.)
   - Rethinking training objectives
   - Look beyond diffusion models (stochastic interpolants/flows, Koopman operators)

4. Adapt algorithms to specific **applications**.

Next steps

The sampling community shouldn't miss out on the opportunities offered by GPU parallelization and deep learning.

1. Build good and reusable **implementations** (e.g. BlackJAX).
2. Explain the gain via appropriate **theories**.
3. Build better **methodologies**.
4. Adapt algorithms to specific **applications**.
   - Proteins, large language models

Thank you for your attention

The sampling community shouldn't miss out on the opportunities offered by GPU parallelization and deep learning.

1. Build good and reusable **implementations** (e.g. BlackJAX).
2. Explain the gain via appropriate **theories**.
3. Build better **methodologies**.
4. Adapt algorithms to specific **applications**.