Pursuits and Challenges Towards Simulation-free Training of Neural Sampler

Jiajun He & Yuanqi Du

@MILA Sampling Reading Group

Collaborators and Supervisors:

Francisco Vargas, Dinghuai Zhang, Shreyas Padhy, RuiKang OuYang, Carla Gomes, José Miguel Hernández-Lobato

Sampling

Unnormalized density function:

$$p_{\text{target}}(x) = \frac{\tilde{p}(x)}{Z}, \qquad Z = \int \tilde{p}(x)\mathrm{d}x$$

Obtain sample $x \sim p_{\text{target}}$.

Sampling

Unnormalized density function:

$$p_{\text{target}}(x) = \frac{\tilde{p}(x)}{Z}, \qquad Z = \int \tilde{p}(x)\mathrm{d}x$$

Obtain sample $x \sim p_{\text{target}}$.

👉 Bayesian inference: $p_{\text{target}} \propto \text{likelihood} \times \text{prior}$

Sampling

Unnormalized density function:

$$p_{\text{target}}(x) = \frac{\tilde{p}(x)}{Z}, \qquad Z = \int \tilde{p}(x)\mathrm{d}x$$

Obtain sample $x \sim p_{\text{target}}$.

- 👉 Bayesian inference: $p_{\text{target}} \propto \text{likelihood} \times \text{prior}$
- 👉 Boltzmann distribution (molecules, etc): $p_{\text{target}} \propto \exp(-\beta U)$

Sampling

Unnormalized density function:

$$p_{\text{target}}(x) = \frac{\tilde{p}(x)}{Z}, \qquad Z = \int \tilde{p}(x)\mathrm{d}x$$

Obtain sample $x \sim p_{\text{target}}$.

👉 Bayesian inference: $p_{\text{target}} \propto \text{likelihood} \times \text{prior}$

👉 Boltzmann distribution (molecules, etc): $p_{\text{target}} \propto \exp(-\beta U)$

👉 Rare event: $p_{\text{target}}(x) \propto \mathbf{1}_B(x)p(x)$

Sampling – classical approach

Markov chain Monte Carlo (MCMC)

Sampling – classical approach

Markov chain Monte Carlo (MCMC)

For example, unadjusted Langevin dynamics:

$$\mathrm{d}X_t = \nabla \log \tilde{p}(X_t)\mathrm{d}t + \sqrt{2}\mathrm{d}W_t$$

Sampling – classical approach

Markov chain Monte Carlo (MCMC)

For example, unadjusted Langevin dynamics:

$$\mathrm{d}X_t = \nabla \log \tilde{p}(X_t)\mathrm{d}t + \sqrt{2}\mathrm{d}W_t$$

😔 dependent samples; auto-correlation reduces efficiency sample size

Sampling – classical approach

Markov chain Monte Carlo (MCMC)

For example, unadjusted Langevin dynamics:

$$\mathrm{d}X_t = \nabla \log \tilde{p}(X_t)\mathrm{d}t + \sqrt{2}\mathrm{d}W_t$$

😔 dependent samples; auto-correlation reduces efficiency sample size

😔 ergodicity; only guarantee convergence with infinite steps

Neural samplers

Train a neural network to amortize the sampling process

Neural samplers

Train a neural network to amortize the sampling process

🙂 independent samples!

🙂 can mix in finite time

Neural samplers

Train a neural network to amortize the sampling process

🥹 independent samples!

🥹 can mix in finite time

Neural samplers are in fact generative models:

$$p_{\text{prior}} \longrightarrow p_{\text{target}}$$

generative model
NF, Diffusion, etc.

Diffusion Neural samplers

Train a diffusion (like) model

$$\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t,$$

Diffusion Neural samplers

Train a diffusion (like) model

$$\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t,$$

transporting samples from p_{prior} to p_{target}:

$$X_0 \sim p_{\text{prior}} \text{ , and want } X_T \sim p_{\text{target}}$$

Diffusion Neural samplers

1. Time-reversal sampler

Diffusion Neural samplers

1. Time-reversal sampler

$$\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}} \, ,$$

Want: $X_T \sim p_{\mathrm{target}}$

Diffusion Neural samplers

1. Time-reversal sampler

$$\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}} ,$$

Set a target process: $\mathrm{d}Y_t = g(Y_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, Y_0 \sim p_{\mathrm{target}},$

Diffusion Neural samplers

1. Time-reversal sampler

$$\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}} \,,$$

Set a target process: $\mathrm{d}Y_t = \underbrace{g(Y_t, t)}_{\text{a simple function, e.g., 0}}\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, Y_0 \sim p_{\mathrm{target}},$

Diffusion Neural samplers

1. Time-reversal sampler

$$\boxed{\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}}\,,}$$

align

Set a target process: $\boxed{\mathrm{d}Y_t = \underbrace{g(Y_t, t)}_{\text{a simple function, e.g., } 0}\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, Y_0 \sim p_{\text{target}},}$

Diffusion Neural samplers

1. Time-reversal sampler

$$\boxed{\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}},}$$

align

$X_t \sim Y_{T-t}$

Set a target process: $\boxed{\mathrm{d}Y_t = \underbrace{g(Y_t, t)}_{\text{a simple function, e.g., } 0}\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, Y_0 \sim p_{\text{target}},}$

Diffusion Neural samplers

1. Time-reversal sampler

$$\boxed{\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}} ,}$$

align

$X_t \sim Y_{T-t}$

$X_T \sim p_{\text{target}}$

Set a target process: $\boxed{\mathrm{d}Y_t = \underbrace{g(Y_t, t)}_{\text{a simple function, e.g., 0}}\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, Y_0 \sim p_{\text{target}},}$

Diffusion Neural samplers

1. Time-reversal sampler

Want a sample process (prior to target),

$$\boxed{\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}},}$$

align

$X_t \sim Y_{T-t}$

$X_T \sim p_{\text{target}}$

Set a target process: $\boxed{\mathrm{d}Y_t = \underbrace{g(Y_t, t)}_{\text{a simple function, e.g., 0}}\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, Y_t \sim p_{\text{target}},}$

Diffusion Neural samplers

1. Time-reversal sampler

Want a sample process (prior to target),

$$\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}},$$

To be the **time-reversal,**

align

$X_t \sim Y_{T-t}$

$X_T \sim p_{\text{target}}$

Set a target process: $\mathrm{d}Y_t = \underbrace{g(Y_t, t)}_{\text{a simple function, e.g., 0}}\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, Y_t \sim p_{\text{target}},$

Diffusion Neural samplers

1. Time-reversal sampler

Want a sample process (prior to target),

$$\boxed{\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}},}$$

To be the **time-reversal,**

align

$X_t \sim Y_{T-t}$

$X_T \sim p_{\text{target}}$

of a simple process (target to prior)

Set a target process: $\boxed{\mathrm{d}Y_t = \underbrace{g(Y_t, t)}_{\text{a simple function, e.g., 0}}\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, Y_t \sim p_{\text{target}},}$

Diffusion Neural samplers

1. Time-reversal sampler

This includes

(1) DDS (denoising diffusion sampler)

(2) PIS (path integral sampler)

(3) DIS (diffusion time-reversal sampler)

(4) GFlowNet (generative flow network)

(5) iDEM (iterated denoising energy matching)

(6) RDMC (reversal diffusion monte carlo)

(7) PINN (physics-informed neural networks) sampler

…

Diffusion Neural samplers

Any other ways?

Diffusion Neural samplers

2. Escorted transport sampler

Diffusion Neural samplers

2. Escorted transport sampler

$$\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}} \,,$$

Want: $X_T \sim p_{\mathrm{target}}$

Diffusion Neural samplers

2. Escorted transport sampler

$$\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}} \, ,$$

define a sequence of interpolants $\pi_t \colon \pi_0 = p_{\mathrm{prior}}, \pi_T = p_{\mathrm{target}}$

Diffusion Neural samplers

2. Escorted transport sampler

$$\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}} ,$$

define a sequence of interpolants π_t: $\pi_0 = p_{\text{prior}}, \pi_T = p_{\text{target}}$

If marginal of $X_t \sim \pi_t$ $\Rightarrow$ $X_T \sim p_{\text{target}}$

Diffusion Neural samplers

2. Escorted transport sampler

Want a sample process (prior to target),

$$\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}} ,$$

define a sequence of interpolants $\pi_t \colon \pi_0 = p_{\text{prior}}, \pi_T = p_{\text{target}}$

If marginal of $X_t \sim \pi_t$ $\Longrightarrow$ $X_T \sim \ p_{\text{target}}$

Diffusion Neural samplers

2. Escorted transport sampler

Want a sample process (prior to target),

$$\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}},$$

whose **marginal densities aligns with**

define a sequence of interpolants π_t: $\pi_0 = p_{\mathrm{prior}}, \pi_T = p_{\mathrm{target}}$

If marginal of $X_t \sim \pi_t$ $\Rightarrow$ $X_T \sim p_{\mathrm{target}}$

Diffusion Neural samplers

2. Escorted transport sampler

Want a sample process (prior to target),

$$\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}} \,,$$

whose **marginal densities aligns with**

define a sequence of interpolants $\pi_t: \pi_0 = p_{\mathrm{prior}}, \pi_T = p_{\mathrm{target}}$

pre-defined interpolants between prior and target

If marginal of $X_t \sim \pi_t$ $\Rightarrow$ $X_T \sim p_{\mathrm{target}}$

Diffusion Neural samplers

2. Escorted transport sampler

This includes

(1) CMCD (Controlled Monte Carlo Diffusions)

(2) NETS (non-equilibrium transport sampler)

(3) PINN (physics-informed neural networks) sampler

(4) LFIS (Liouville Flow Importance Sampler)

…

Diffusion Neural samplers

Any other ways?

Diffusion Neural samplers

3. Annealed variance reduction sampler

Diffusion Neural samplers

3. Annealed variance reduction sampler

$$\mathrm{d}X_t = f_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}} \, ,$$

Want: $X_T \sim p_{\mathrm{target}}$

Diffusion Neural samplers

3. Annealed variance reduction sampler

$$\mathrm{d}X_t = \underbrace{f_\theta(X_t, t)}_{\text{What if we do not train it?}}\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}}\,,$$

Want: $X_T \sim p_{\mathrm{target}}$

Diffusion Neural samplers

3. Annealed variance reduction sampler

$$\mathrm{d}X_t = \overbrace{f(X_t, t)}^{\text{What if we do not train it?}}\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}}\,,$$

$$X_T \not\sim p_{\text{target}}$$

Diffusion Neural samplers

3. Annealed variance reduction sampler

$$\mathrm{d}X_t = f(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}},$$

$$\mathrm{d}X_t = g(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t^t, X_T \sim p_{\mathrm{target}},$$

Diffusion Neural samplers

3. Annealed variance reduction sampler

$$\mathrm{d}X_t = f(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}}, \quad \Rightarrow \quad \overrightarrow{\mathbf{Q}}(X)$$

$$\mathrm{d}X_t = g(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t^t, X_T \sim p_{\mathrm{target}}, \quad \Rightarrow \quad \overleftarrow{\mathbf{P}}(X)$$

Diffusion Neural samplers

3. Annealed variance reduction sampler

$$\mathrm{d}X_t = f(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}}, \Rightarrow \overrightarrow{\mathbf{Q}}(X)$$

$$\mathrm{d}X_t = g(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t^t, X_T \sim p_{\mathrm{target}}, \Rightarrow \overleftarrow{\mathbf{P}}(X)$$

Importance weight: $\dfrac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)}$

Diffusion Neural samplers

3. Annealed variance reduction sampler

$$\mathrm{d}X_t = f(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}}\,, \Rightarrow \overrightarrow{\mathbf{Q}}(X)$$

$$\mathrm{d}X_t = g_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t^t, X_T \sim p_{\mathrm{target}}, \Rightarrow \overleftarrow{\mathbf{P}_\theta}(X)$$

Importance weight: $\dfrac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}_\theta}(X)}$

Diffusion Neural samplers

3. Annealed variance reduction sampler

Align

$$\mathrm{d}X_t = f(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}}, \Rightarrow \overrightarrow{\mathbf{Q}}(X)$$

$$\mathrm{d}X_t = g_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t^t, X_T \sim p_{\text{target}}, \Rightarrow \overleftarrow{\mathbf{P}_\theta}(X)$$

Importance weight: $\dfrac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}_\theta}(X)}$

Diffusion Neural samplers

3. Annealed variance reduction sampler

Align

$$\mathrm{d}X_t = f(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}}, \Rightarrow \overrightarrow{\mathbf{Q}}(X)$$

$$\mathrm{d}X_t = g_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t^t, X_T \sim p_{\mathrm{target}}, \Rightarrow \overleftarrow{\mathbf{P}_\theta}(X)$$

Small variance

Importance weight: $\dfrac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}_\theta}(X)}$

Diffusion Neural samplers

3. Annealed variance reduction sampler

Predefine a sample process (prior to target),

Align

$$\mathrm{d}X_t = f(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}}, \Rightarrow \overrightarrow{\mathbf{Q}}(X)$$

$$\mathrm{d}X_t = g_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t^t, X_T \sim p_{\mathrm{target}}, \Rightarrow \overleftarrow{\mathbf{P}_\theta}(X)$$

Small variance

Importance weight: $\dfrac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}_\theta}(X)}$

Diffusion Neural samplers

3. Annealed variance reduction sampler

Predefine a sample process (prior to target),

$$\mathrm{d}X_t = f(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}}, \Rightarrow \overrightarrow{\mathbf{Q}}(X)$$

Align

define or train a backward process (target to prior),

$$\mathrm{d}X_t = g_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t^t, X_T \sim p_{\text{target}}, \Rightarrow \overleftarrow{\mathbf{P}_\theta}(X)$$

Small variance

Importance weight: $\dfrac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}_\theta}(X)}$

Diffusion Neural samplers

3. Annealed variance reduction sampler

Predefine a sample process (prior to target),

$$\mathrm{d}X_t = f(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}}, \Rightarrow \overrightarrow{\mathbf{Q}}(X)$$

Align

define or train a backward process (target to prior),

$$\mathrm{d}X_t = g_\theta(X_t, t)\mathrm{d}t + \sigma\sqrt{2}\mathrm{d}W_t^t, X_T \sim p_{\text{target}}, \Rightarrow \overleftarrow{\mathbf{P}_\theta}(X)$$

perform importance sampling

Small variance

Importance weight: $\dfrac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}_\theta}(X)}$

Diffusion Neural samplers

3. Annealed variance reduction sampler

This includes

(1) AIS (Annealed Importance Sampling)

(2) MCD (Monte Carlo Diffusion)

(3) LDVI (Langevin Diffusion Variational Inference)

…

Diffusion Neural samplers

1. Time-reversal sampler

2. Escorted transport sampler

3. Annealed variance reduction sampler

Diffusion Neural samplers

1. Time-reversal sampler

2. Escorted transport sampler

3. Annealed variance reduction sampler

These are design choices for the sampling processes

Diffusion Neural samplers

1. Time-reversal sampler

2. Escorted transport sampler

3. Annealed variance reduction sampler

These are design choices for the sampling processes
But **how to train them?**

Diffusion Neural samplers

1. Time-reversal sampler
2. Escorted transport sampler
3. Annealed variance reduction sampler

These are design choices for the sampling processes
But **how to train them?**

Diffusion Neural samplers

a. path-measure alignment

Diffusion Neural samplers

a. path-measure alignment

For any desired process $\overrightarrow{\mathbf{Q}}(X)$ or $\overrightarrow{\mathbf{Q}}_{\boldsymbol{\theta}}(X)$

Diffusion Neural samplers

a. path-measure alignment

For any desired process $\overrightarrow{\mathbf{Q}}(X)$ or $\overrightarrow{\mathbf{Q}}_{\boldsymbol{\theta}}(X)$

we can write down its desired time-reversal $\overleftarrow{\mathbf{P}}(X)$ or $\overleftarrow{\mathbf{P}_{\boldsymbol{\theta}}}(X)$,

Diffusion Neural samplers

a. path-measure alignment

For any desired process $\overrightarrow{\mathbf{Q}}(X)$ or $\overrightarrow{\mathbf{Q}}_{\boldsymbol{\theta}}(X)$

align

we can write down its desired time-reversal $\overleftarrow{\mathbf{P}}(X)$ or $\overleftarrow{\mathbf{P}_{\boldsymbol{\theta}}}(X)$,

Diffusion Neural samplers

a. path-measure alignment

$$D_{\mathrm{KL}}[\overrightarrow{\mathbf{Q}}||\overleftarrow{\mathbf{P}}] = \mathrm{E}_{\overrightarrow{\mathbf{Q}}}\left[\log \frac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)}\right]$$

$$D_{\mathrm{LV}}[\overrightarrow{\mathbf{Q}}||\overleftarrow{\mathbf{P}}] = \mathrm{Var}_{\overrightarrow{\boldsymbol{\pi}}}\left[\log \frac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)}\right]$$

$$D_{\mathrm{TB}}[\overrightarrow{\mathbf{Q}}||\overleftarrow{\mathbf{P}}] = \mathrm{E}_{\overrightarrow{\boldsymbol{\pi}}}\left[\left(\log \frac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)} - k\right)^2\right]$$

Other choices exist, including sub-TB, DB, etc…

Diffusion Neural samplers

b. marginal alignment

Diffusion Neural samplers

b. marginal alignment

For any desired process $\overrightarrow{\mathbf{Q}}(X)$ or $\overrightarrow{\mathbf{Q}}_{\boldsymbol{\theta}}(X)$

Diffusion Neural samplers

b. marginal alignment

For any desired process $\overrightarrow{\mathbf{Q}}(X)$ or $\overrightarrow{\mathbf{Q}}_{\boldsymbol{\theta}}(X)$

we can write down its desired marginal $q_t(X_t)$,

Diffusion Neural samplers

b. marginal alignment

For any desired process $\overrightarrow{\mathbf{Q}}(X)$ or $\overrightarrow{\mathbf{Q}}_{\boldsymbol{\theta}}(X)$

we can write down its desired marginal $q_t(X_t)$,

align

Diffusion Neural samplers

b. marginal alignment

Score matching with Score estimator

PINN

Action matching

…

Diffusion Neural samplers

	Time-reversal sampler	**Escorted transport sampler**	**Annealed Variance Reduction Sampler**
Path measure alignment	DDS, DIS, PIS, GFN	CMCD, SLCD	MCD
Marginal alignment	iDEM, RDMC, PINN-sampler	NETS, PINN-sampler, LFIS	

Diffusion Neural samplers - Desiderata

Let’s look at the loss again, for example:

Diffusion Neural samplers - Desiderata

Let's look at the loss again, for example:

$$D_{\mathrm{KL}}[\overrightarrow{\mathbf{Q}}||\overleftarrow{\mathbf{P}}] = \mathrm{E}_{\overrightarrow{\mathbf{Q}}}\left[\log \frac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)}\right]$$

$$D_{\mathrm{LV}}[\overrightarrow{\mathbf{Q}}||\overleftarrow{\mathbf{P}}] = \mathrm{Var}_{\overrightarrow{\boldsymbol{\pi}}}\left[\log \frac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)}\right]$$

$$D_{\mathrm{TB}}[\overrightarrow{\mathbf{Q}}||\overleftarrow{\mathbf{P}}] = \mathrm{E}_{\overrightarrow{\boldsymbol{\pi}}}\left[\left(\log \frac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)} - k\right)^2\right]$$

Diffusion Neural samplers - Desiderata

Let's look at the loss again, for example:

$$D_{\mathrm{KL}}[\overrightarrow{\mathbf{Q}}||\overleftarrow{\mathbf{P}}] = \mathbb{E}_{\overrightarrow{\mathbf{Q}}}\left[\log \frac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)}\right]$$

$$D_{\mathrm{LV}}[\overrightarrow{\mathbf{Q}}||\overleftarrow{\mathbf{P}}] = \mathrm{Var}_{\overrightarrow{\boldsymbol{\pi}}}\left[\log \frac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)}\right]$$

$$D_{\mathrm{TB}}[\overrightarrow{\mathbf{Q}}||\overleftarrow{\mathbf{P}}] = \mathbb{E}_{\overrightarrow{\boldsymbol{\pi}}}\left[\left(\log \frac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)} - k\right)^2\right]$$

Diffusion Neural samplers - Desiderata

Let’s look at the loss again, for example:

$$D_{\mathrm{KL}}[\overrightarrow{\mathbf{Q}}||\overleftarrow{\mathbf{P}}] = \mathbb{E}_{\overrightarrow{\mathbf{Q}}}\left[\log \frac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)}\right]$$

$$D_{\mathrm{LV}}[\overrightarrow{\mathbf{Q}}||\overleftarrow{\mathbf{P}}] = \mathrm{Var}_{\overrightarrow{\boldsymbol{\pi}}}\left[\log \frac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)}\right]$$

$$D_{\mathrm{TB}}[\overrightarrow{\mathbf{Q}}||\overleftarrow{\mathbf{P}}] = \mathbb{E}_{\overrightarrow{\boldsymbol{\pi}}}\left[\left(\log \frac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)} - k\right)^2\right]$$

😔 need to simulate the trajectory – expensive!

Diffusion Neural samplers - Desiderata

Let's look at the loss again, for example:

$$D_{\mathrm{KL}}[\overrightarrow{\mathbf{Q}}||\overleftarrow{\mathbf{P}}] = \mathrm{E}_{\overrightarrow{\mathbf{Q}}}\left[\log \frac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)}\right]$$

$$D_{\mathrm{LV}}[\overrightarrow{\mathbf{Q}}||\overleftarrow{\mathbf{P}}] = \mathrm{Var}_{\overrightarrow{\boldsymbol{\pi}}}\left[\log \frac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)}\right]$$

$$D_{\mathrm{TB}}[\overrightarrow{\mathbf{Q}}||\overleftarrow{\mathbf{P}}] = \mathrm{E}_{\overrightarrow{\boldsymbol{\pi}}}\left[\left(\log \frac{\mathrm{d}\overrightarrow{\mathbf{Q}}(X)}{\mathrm{d}\overleftarrow{\mathbf{P}}(X)} - k\right)^2\right]$$

😔 need to simulate the trajectory – expensive!

Any ways for "simulation-free" training?

Simulation-free training of Diffusion Neural samplers

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

💡 using a time-dependent normalizing flow

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

💡 using a time-dependent normalizing flow

Define $F_\theta(\cdot, t)$ as an invertible function

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

💡 using a time-dependent normalizing flow

Define $F_\theta(\cdot, t)$ as an invertible function

The first way of sampling

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

💡 using a time-dependent normalizing flow

Define $F_\theta(\cdot, t)$ as an invertible function

The first way of sampling $X_t = F_\theta(Z, t), \quad Z \sim p_{\text{base}}$

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

💡 using a time-dependent normalizing flow

Define $F_\theta(\cdot, t)$ as an invertible function

The first way of sampling $$X_t = F_\theta(Z, t), \quad Z \sim p_{\text{base}}$$

The second way of sampling

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

💡 using a time-dependent normalizing flow

Define $F_\theta(\cdot, t)$ as an invertible function

The first way of sampling $X_t = F_\theta(Z, t), \quad Z \sim p_{\text{base}}$

The second way of sampling $X_0 = F_\theta(Z, 0), Z \sim p_{\text{base}}$

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

💡 using a time-dependent normalizing flow

Define $F_\theta(\cdot, t)$ as an invertible function

The first way of sampling $$X_t = F_\theta(Z, t), \quad Z \sim p_{\text{base}}$$

The second way of sampling $$X_0 = F_\theta(Z, 0), Z \sim p_{\text{base}}$$
$$\mathrm{d}X_t = \partial_t F_\theta(Z, t)\mathrm{d}t$$

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

💡 using a time-dependent normalizing flow

Define $F_\theta(\cdot, t)$ as an invertible function

The first way of sampling $\quad X_t = F_\theta(Z, t), \quad Z \sim p_{\mathrm{base}}$

The second way of sampling

$$X_0 = F_\theta(Z, 0), Z \sim p_{\mathrm{base}}$$
$$\mathrm{d}X_t = \partial_t F_\theta(\underbrace{Z,}_{Z = F_\theta^{-1}(X_t, t)} t)\mathrm{d}t$$

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

💡 using a time-dependent normalizing flow

Define $F_\theta(\cdot, t)$ as an invertible function

The first way of sampling $X_t = F_\theta(Z, t), \quad Z \sim p_{\mathrm{base}}$

The second way of sampling

$$X_0 = F_\theta(Z, 0), Z \sim p_{\mathrm{base}}$$

$$\mathrm{d}X_t = \partial_t F_\theta\big(F_\theta^{-1}(X_t, t), t\big)\mathrm{d}t$$

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

💡 using a time-dependent normalizing flow

Define $F_\theta(\cdot, t)$ as an invertible function

The first way of sampling $X_t = F_\theta(Z, t), \quad Z \sim p_{\text{base}}$

The second way of sampling

$$X_0 = F_\theta(Z, 0), Z \sim p_{\text{base}}$$
$$\mathrm{d}X_t = \underbrace{\partial_t F_\theta\left(F_\theta^{-1}(X_t, t), t\right)}_{\text{Standard form of ODE}}\mathrm{d}t$$

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

💡 using a time-dependent normalizing flow

Define $F_\theta(\cdot, t)$ as an invertible function

The first way of sampling $X_t = F_\theta(Z, t), \quad Z \sim p_{\text{base}}$

The second way of sampling
$$X_0 = F_\theta(Z, 0), Z \sim p_{\text{base}}$$
$$\mathrm{d}X_t = \partial_t F_\theta\big(F_\theta^{-1}(X_t, t), t\big)\mathrm{d}t$$

$$\mathrm{d}X_t = \partial_t F_\theta\big(F_\theta^{-1}(X_t, t), t\big)\mathrm{d}t \qquad + \sigma_t\sqrt{2}\mathrm{d}W_t$$

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

💡 using a time-dependent normalizing flow

Define $F_\theta(\cdot, t)$ as an invertible function

The first way of sampling $X_t = F_\theta(Z, t), \quad Z \sim p_{\text{base}}$

The second way of sampling $X_0 = F_\theta(Z, 0), Z \sim p_{\text{base}}$

$$\mathrm{d}X_t = \partial_t F_\theta\left(F_\theta^{-1}(X_t, t), t\right)\mathrm{d}t$$

$$\mathrm{d}X_t = \partial_t F_\theta\left(F_\theta^{-1}(X_t, t), t\right)\mathrm{d}t + \sigma_t^2 \nabla \log q_\theta(X_t, t)\mathrm{d}t + \sigma_t \sqrt{2}\mathrm{d}W_t$$

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

using a time-dependent normalizing flow

Define $F_\theta(\cdot, t)$ as an invertible function

The first way of sampling $X_t = F_\theta(Z, t), \quad Z \sim p_{\text{base}}$

The second way of sampling

$$X_0 = F_\theta(Z, 0), Z \sim p_{\text{base}}$$

$$\mathrm{d}X_t = \partial_t F_\theta\left(F_\theta^{-1}(X_t, t), t\right)\mathrm{d}t$$

Easily obtained by NF

$$\mathrm{d}X_t = \partial_t F_\theta\left(F_\theta^{-1}(X_t, t), t\right)\mathrm{d}t + \boxed{\sigma_t^2 \nabla \log q_\theta(X_t, t)}\,\mathrm{d}t + \sigma_t\sqrt{2}\,\mathrm{d}W_t$$

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

using a time-dependent normalizing flow

Define $F_\theta(\cdot, t)$ as an invertible function

The first way of sampling $X_t = F_\theta(Z, t), \quad Z \sim p_{\mathrm{base}}$

directly sample from time t

The second way of sampling

$$X_0 = F_\theta(Z, 0), Z \sim p_{\mathrm{base}}$$

$$\mathrm{d}X_t = \partial_t F_\theta\left(F_\theta^{-1}(X_t, t), t\right)\mathrm{d}t$$

$$\mathrm{d}X_t = \partial_t F_\theta\left(F_\theta^{-1}(X_t, t), t\right)\mathrm{d}t + \sigma_t^2 \nabla \log q_\theta(X_t, t)\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t$$

Simulation-free training of Diffusion Neural samplers

avoid simulating the trajectory (entirely) during training.

💡 using a time-dependent normalizing flow

Define $F_\theta(\cdot, t)$ as an invertible function

The first way of sampling $X_t = F_\theta(Z, t), \quad Z \sim p_{\text{base}}$

directly sample from time t

The second way of sampling

$$X_0 = F_\theta(Z, 0), Z \sim p_{\text{base}}$$

$$\mathrm{d}X_t = \partial_t F_\theta\left(F_\theta^{-1}(X_t, t), t\right)\mathrm{d}t$$

$$\mathrm{d}X_t = \partial_t F_\theta\left(F_\theta^{-1}(X_t, t), t\right)\mathrm{d}t + \sigma_t^2 \nabla \log q_\theta(X_t, t)\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t$$

Calculate the same loss as other diffusion samplers

Simulation-free training of Diffusion Neural samplers

$$\mathrm{d}X_t = \partial_t F_\theta\left(F_\theta^{-1}(X_t, t), t\right)\mathrm{d}t + \sigma_t^2 \nabla \log q_\theta(X_t, t)\mathrm{d}t + \sigma_t \sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}}$$

Simulation-free training of Diffusion Neural samplers

$$\mathrm{d}X_t = \partial_t F_\theta\left(F_\theta^{-1}(X_t, t), t\right)\mathrm{d}t + \sigma_t^2 \nabla \log q_\theta(X_t, t)\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}}$$

Align

$$\mathrm{d}X_t = \underbrace{g(X_t)}_{\text{a simple function, e.g., } 0}\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t^-, X_T \sim p_{\text{target}}$$

Simulation-free training of Diffusion Neural samplers

$$\mathrm{d}X_t = \partial_t F_\theta\big(F_\theta^{-1}(X_t, t), t\big)\mathrm{d}t + \sigma_t^2 \nabla \log q_\theta(X_t, t)\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}}$$

time-reversal

Align

$$\mathrm{d}X_t = \partial_t F_\theta\big(F_\theta^{-1}(X_t, t), t\big)\mathrm{d}t - \sigma_t^2 \nabla \log q_\theta(X_t, t)\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t^-, X_T \sim q_\theta(\cdot, T)$$

$$\mathrm{d}X_t = \underbrace{g(X_t)}_{\text{a simple function, e.g., 0}}\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t^-, X_T \sim p_{\text{target}}$$

Simulation-free training of Diffusion Neural samplers

$$\mathrm{d}X_t = \partial_t F_\theta\left(F_\theta^{-1}(X_t, t), t\right)\mathrm{d}t + \sigma_t^2 \nabla \log q_\theta(X_t, t)\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}}$$

time-reversal

Align

$$\left(\begin{array}{l} \mathrm{d}X_t = \partial_t F_\theta\left(F_\theta^{-1}(X_t, t), t\right)\mathrm{d}t - \sigma_t^2 \nabla \log q_\theta(X_t, t)\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t^-, X_T \sim q_\theta(\cdot, T) \\ \\ \mathrm{d}X_t = \underbrace{g(X_t)}_{\text{a simple function, e.g., 0}}\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t^-, X_T \sim p_{\text{target}} \end{array}\right.$$

Simulation-free training of Diffusion Neural samplers

$$\mathrm{d}X_t = \partial_t F_\theta\big(F_\theta^{-1}(X_t, t), t\big)\mathrm{d}t + \sigma_t^2 \nabla \log q_\theta(X_t, t)\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\mathrm{prior}}$$

$$\Big\Downarrow \ \text{time-reversal}$$

Align

$$\left(\begin{aligned} &\mathrm{d}X_t = \partial_t F_\theta\big(F_\theta^{-1}(X_t, t), t\big)\mathrm{d}t - \sigma_t^2 \nabla \log q_\theta(X_t, t)\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t^-, X_T \sim q_\theta(\cdot, T) \\ &\mathrm{d}X_t = \underbrace{g(X_t)}_{\text{a simple function, e.g., 0}}\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t^-, X_T \sim p_{\mathrm{target}} \end{aligned}\right.$$

✅ **same direction – Girsanov Theorem applicable**

Simulation-free training of Diffusion Neural samplers

$$\mathrm{d}X_t = \partial_t F_\theta\big(F_\theta^{-1}(X_t,t),t\big)\mathrm{d}t + \sigma_t^2 \nabla \log q_\theta(X_t,t)\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t, X_0 \sim p_{\text{prior}}$$

time-reversal

Align

$$\mathrm{d}X_t = \partial_t F_\theta\big(F_\theta^{-1}(X_t,t),t\big)\mathrm{d}t - \sigma_t^2 \nabla \log q_\theta(X_t,t)\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t^-, X_T \sim q_\theta(\cdot,T)$$

$$\mathrm{d}X_t = \underbrace{g(X_t)}_{\text{a simple function, e.g., 0}}\mathrm{d}t + \sigma_t\sqrt{2}\mathrm{d}W_t^-, X_T \sim p_{\text{target}}$$

✅ **same direction – Girsanov Theorem applicable**

✅ **simulation-free evaluation – can always obtain sample by 1-step** $X_t = F_\theta(Z,t), Z \sim p_{\text{base}}$

Simulation-free training of Diffusion Neural samplers

😄 Great! How does it perform?

Simulation-free training of Diffusion Neural samplers

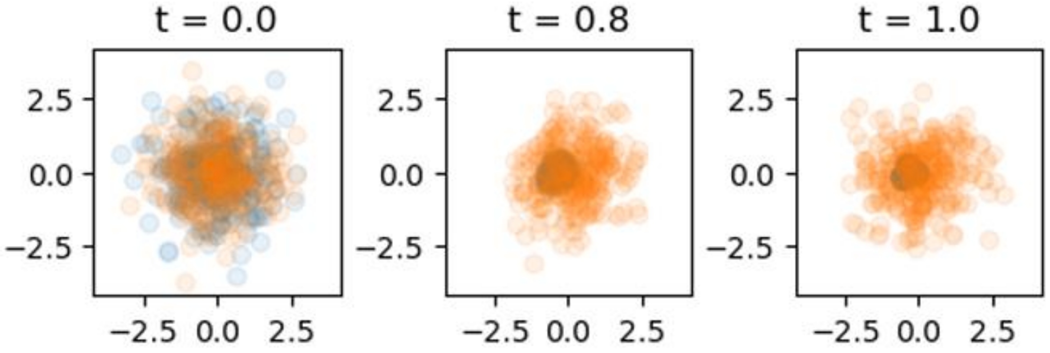
😄 Great! How does it perform?

😢 unfortunately…

Simulation-free training of Diffusion Neural samplers

😄 Great! How does it perform?

😢 unfortunately…

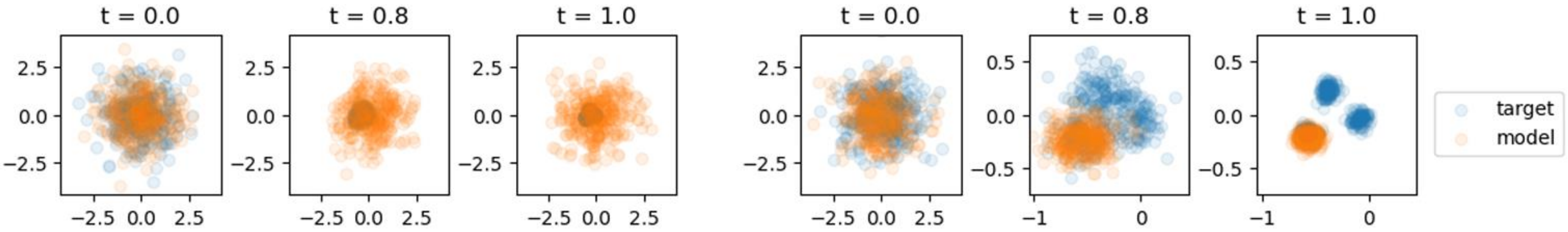


initialization

Simulation-free training of Diffusion Neural samplers

Great! How does it perform?

unfortunately…

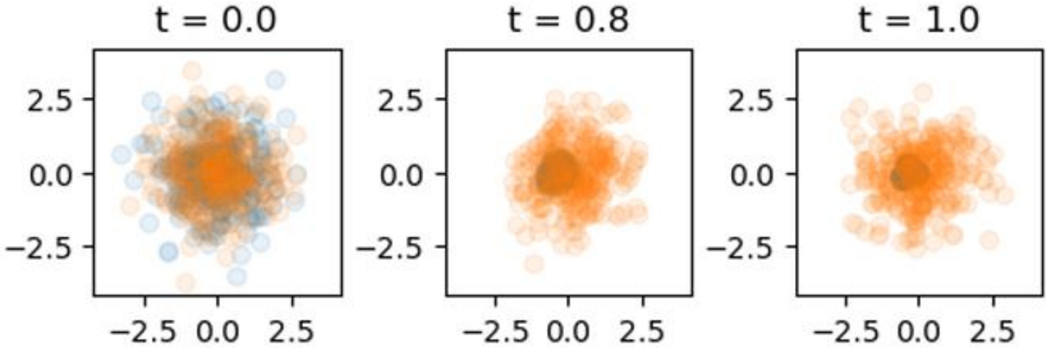


initialization

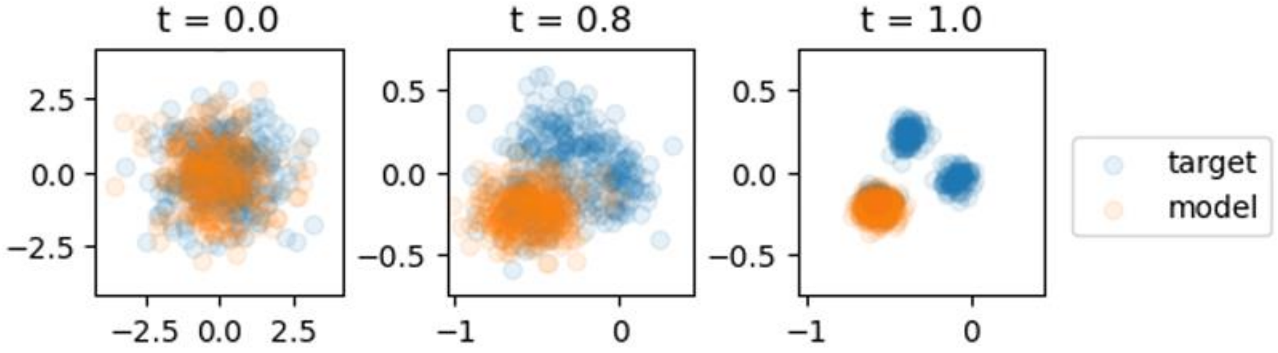
after training

Simulation-free training of Diffusion Neural samplers

Why?



initialization

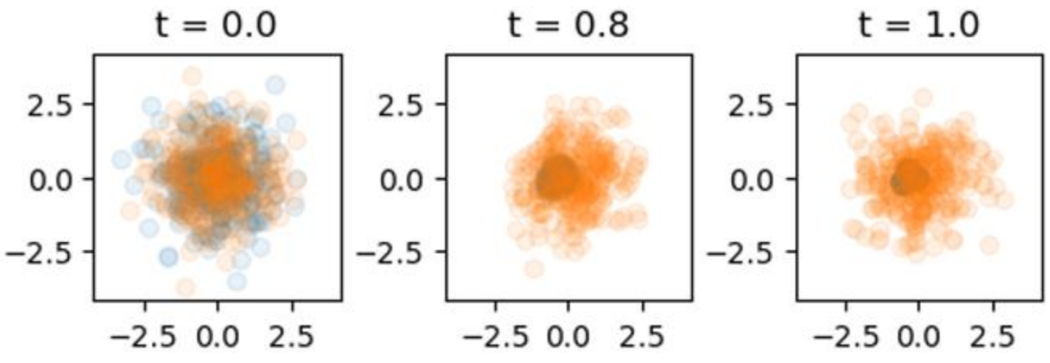


after training

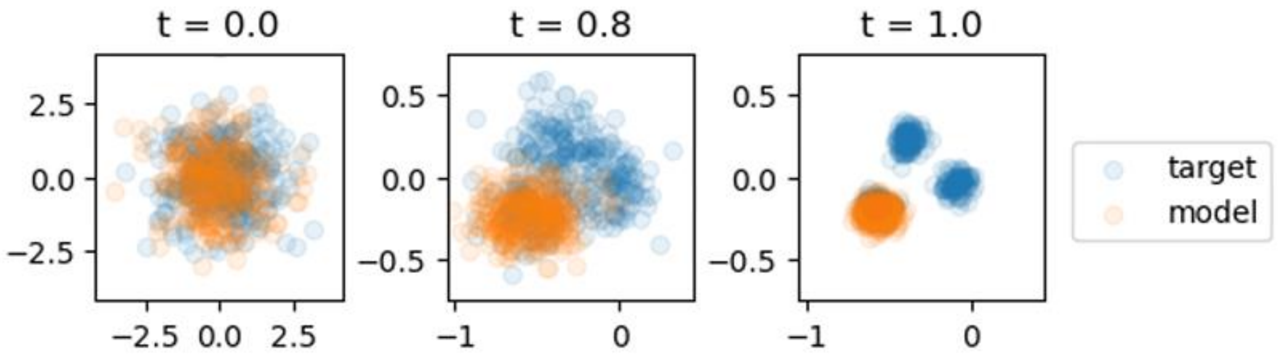
Simulation-free training of Diffusion Neural samplers

Why?

Objective? 🫤 same as DDS



initialization



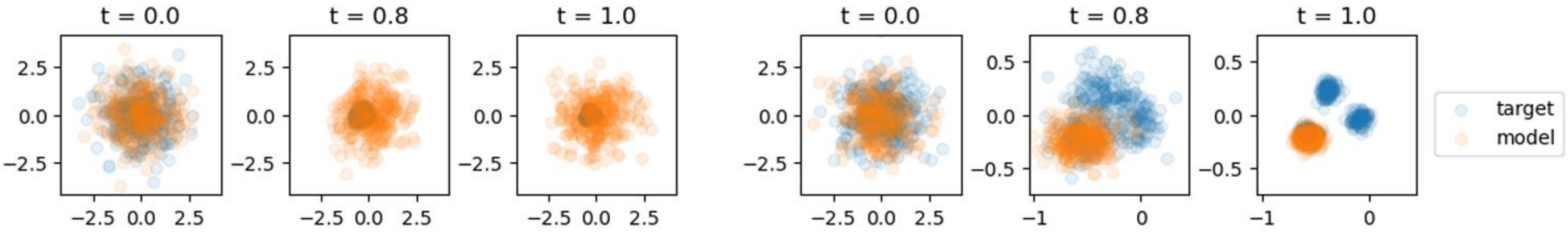
after training

Simulation-free training of Diffusion Neural samplers

Why?

Objective? 🫤 same as DDS

Capacity? 🫤 target is so simple



initialization

after training

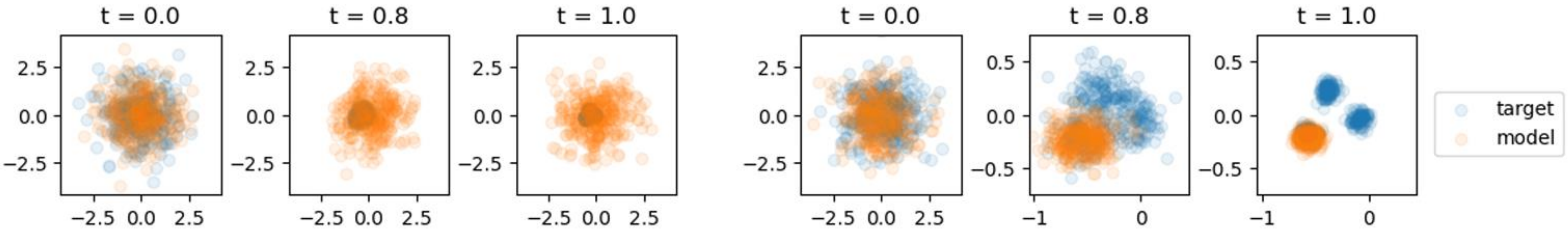
Simulation-free training of Diffusion Neural samplers

Why?

Objective? 🫤 same as DDS

Capacity? 🫤 target is so simple

Network parameterization? 🤔 might be the reason



initialization

after training

Langevin Preconditioning

Langevin Preconditioning

a. DDS/PIS

warm-up initialization

$$f_\theta(\cdot, t) = \boxed{\mathrm{NN}_{1,\theta}(\cdot, t)} + \mathrm{NN}_{2,\theta}(t) \circ \nabla \log p_{\mathrm{target}}(\cdot)$$

$$\approx 0$$

Langevin Preconditioning

a. DDS/PIS

warm-up initialization

$$f_\theta(\cdot, t) = \underbrace{\boxed{\mathrm{NN}_{1,\theta}(\cdot, t)}}_{\approx 0} + \mathrm{NN}_{2,\theta}(t) \circ \boxed{\nabla \log p_{\text{target}}(\cdot)}$$

Langevin gradient

b. CMCD/NETS

$$\pi_t(\cdot) = p_{\text{prior}}^{1-\beta}(\cdot) p_{\text{target}}^{\beta}(\cdot)$$

Langevin gradient

$$dX_t = \left(f_\theta(X_t, t) + \boxed{\sigma_t^2 \nabla \log \pi_t(X_t)) dt} + \sqrt{2}\, \sigma_t dW_t \qquad \overrightarrow{\mathbf{Q}_{\boldsymbol{\theta}}}(X)$$

$$dX_t = \left(f_\theta(X_t, t) - \sigma_t^2 \nabla \log \pi_t(X_t)\right) dt + \sqrt{2}\, \sigma_t dW_t^- \qquad \overleftarrow{\mathbf{P}_{\theta}}(X)$$

Langevin Preconditioning

a. DDS/PIS

warm-up initialization

$$f_\theta(\cdot,t) = \underbrace{\boxed{\mathrm{NN}_{1,\theta}(\cdot,t)}}_{\approx 0} + \mathrm{NN}_{2,\theta}(t) \circ \underbrace{\boxed{\nabla \log p_{\text{target}}(\cdot)}}_{\text{Langevin gradient}}$$

What if we remove this Langevin?

b. CMCD/NETS

$$\pi_t(\cdot) = p_{\text{prior}}^{1-\beta}(\cdot)p_{\text{target}}^{\beta}(\cdot)$$ Langevin gradient

$$dX_t = (f_\theta(X_t,t) + \boxed{\sigma_t^2 \nabla \log \pi_t(X_t))dt} + \sqrt{2}\,\sigma_t dW_t \qquad \overrightarrow{\mathbf{Q}_\theta}(X)$$

$$dX_t = (f_\theta(X_t,t) - \sigma_t^2 \nabla \log \pi_t(X_t))dt + \sqrt{2}\,\sigma_t dW_t^- \qquad \overleftarrow{\mathbf{P}_\theta}(X)$$

How to remove Langevin?

a. DDS

$$f_\theta(\cdot, t) = \mathrm{NN}_{1,\theta}(\cdot, t) + \cancel{\mathrm{NN}_{2,\theta}(t) \circ \nabla \log p_{\text{target}}(\cdot)}$$

b. CMCD's Optimality condition (Nelson's relation)

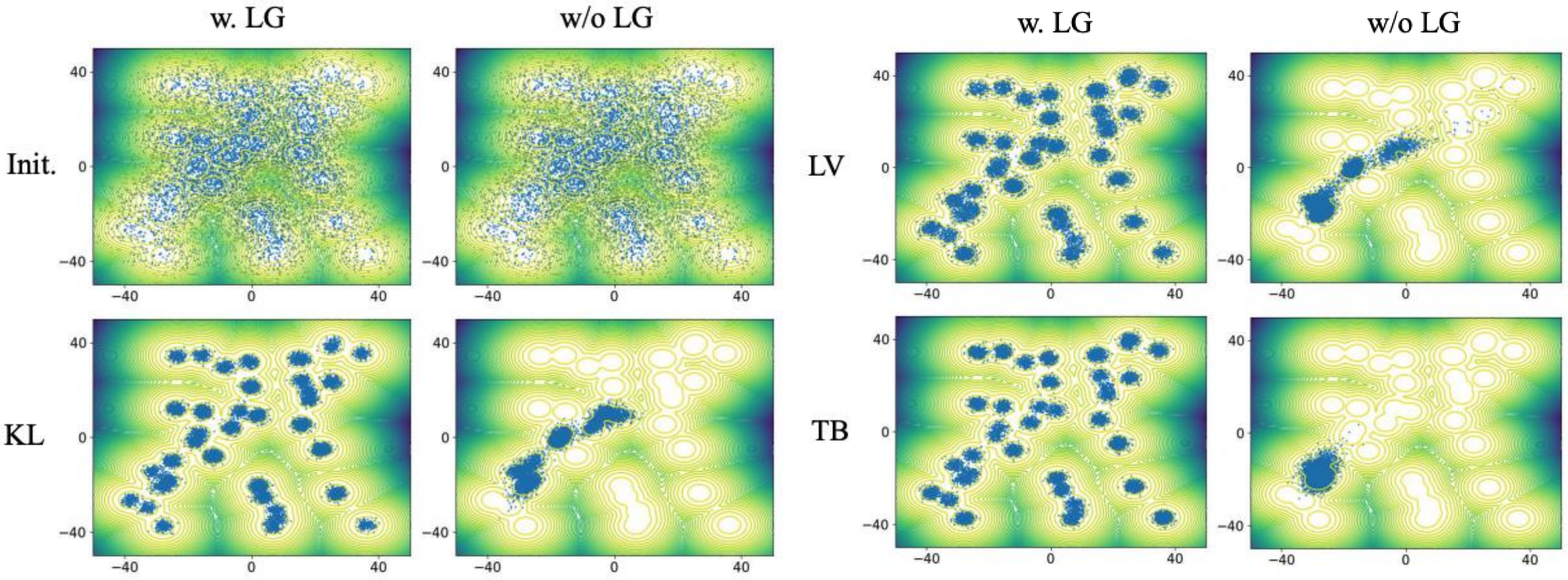
$$\boldsymbol{D}\left(\overrightarrow{\mathbf{Q}_\theta^{p_{\text{prior}},\, f_\theta + \sigma_t^2 \nabla \log \pi_t}},\ \overleftarrow{\mathbf{P}_\theta^{p_{\text{target}},\, f_\theta - \sigma_t^2 \nabla \log \pi_t}}\right)$$

But it is not necessary!

$$\boldsymbol{D}\left(\overrightarrow{\mathbf{Q}_\theta^{p_{\text{prior}},\, f_\theta}},\ \overleftarrow{\mathbf{P}_\theta^{p_{\text{target}},\, f_\theta - 2\sigma_t^2 \nabla \log \pi_t}}\right)$$

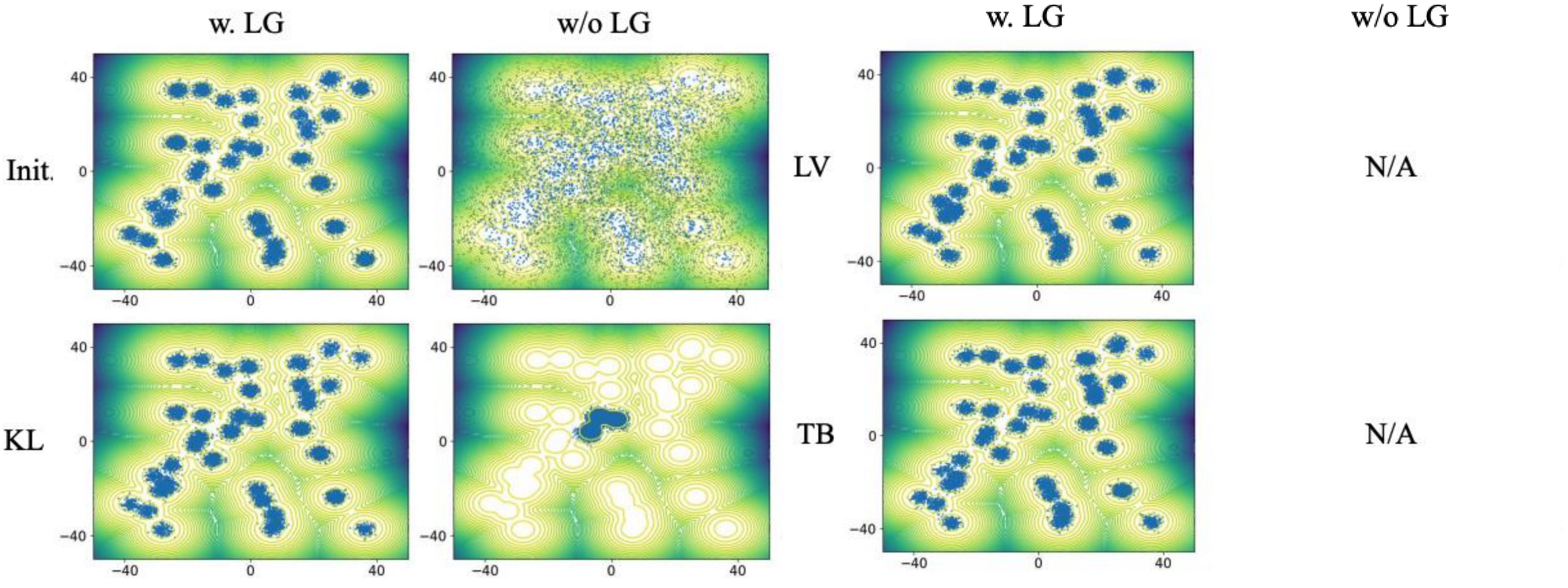
Empirical Results

a. Langevin precondition is necessary to prevent mode collapse



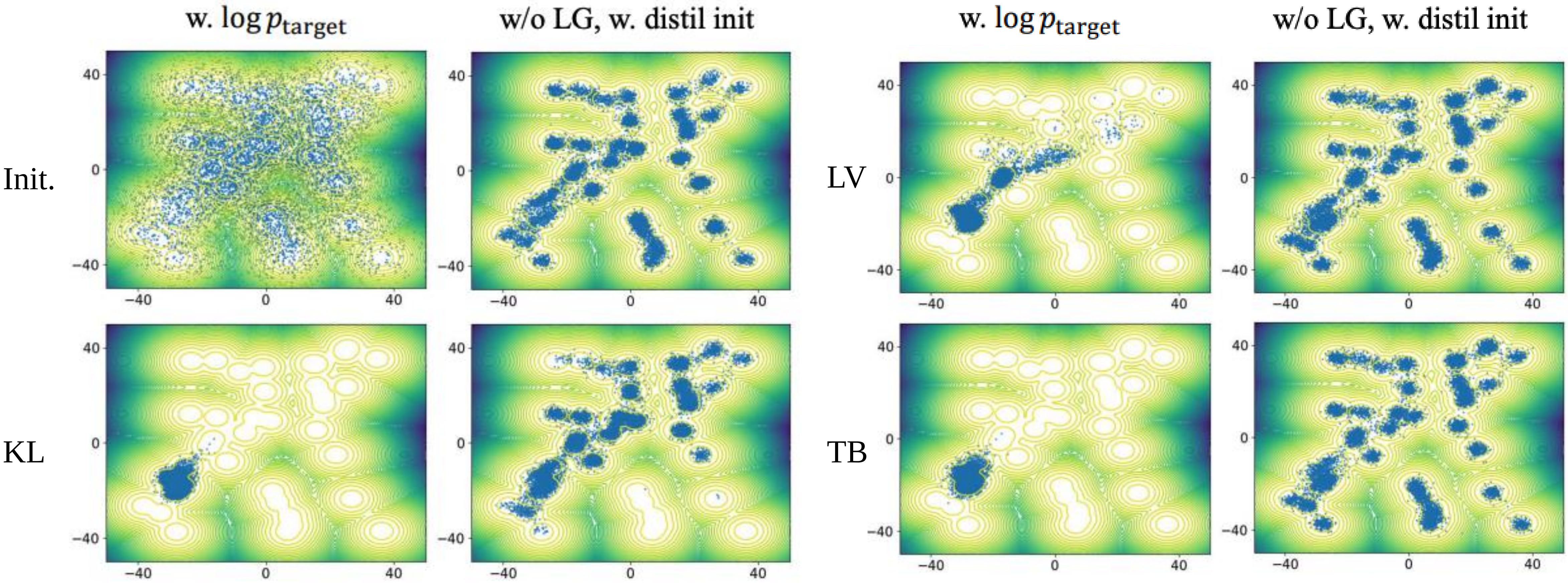
Empirical Results

a. Langevin precondition is necessary to prevent mode collapse



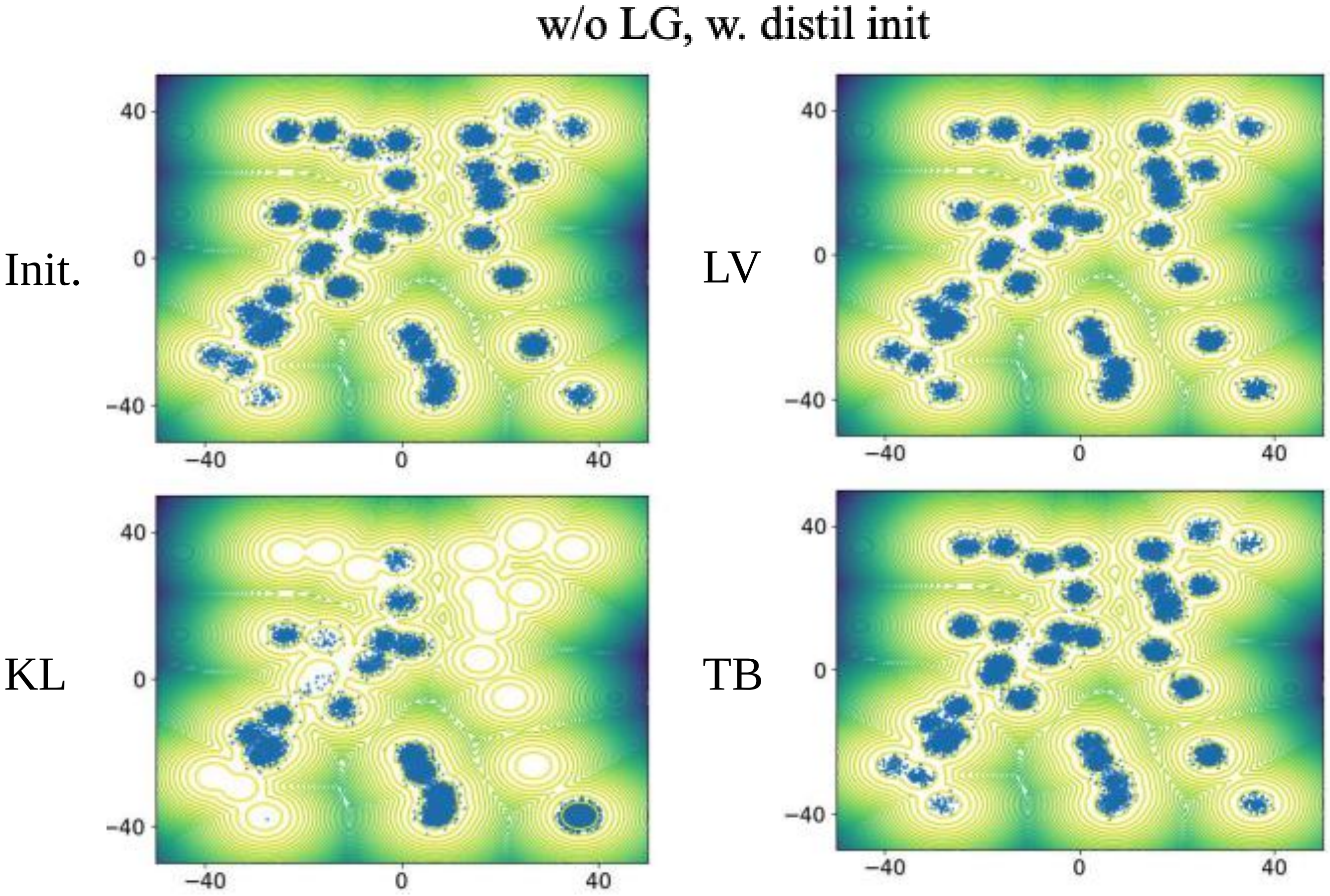
Empirical Results

b. Does other ways of incorporating the target information help?



Empirical Results

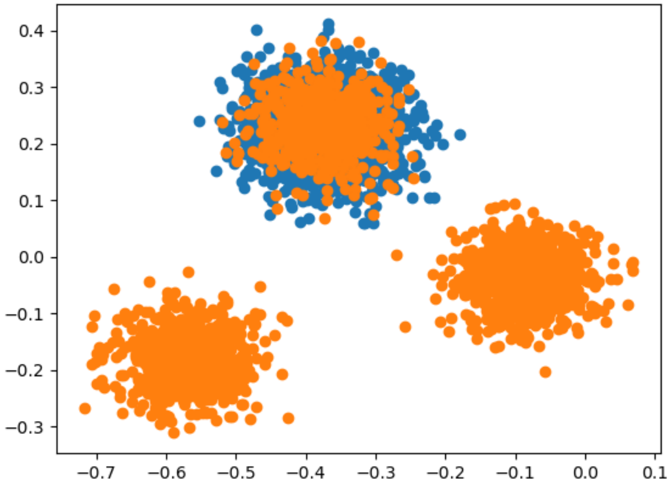
b. Do other ways of incorporating the target information help?



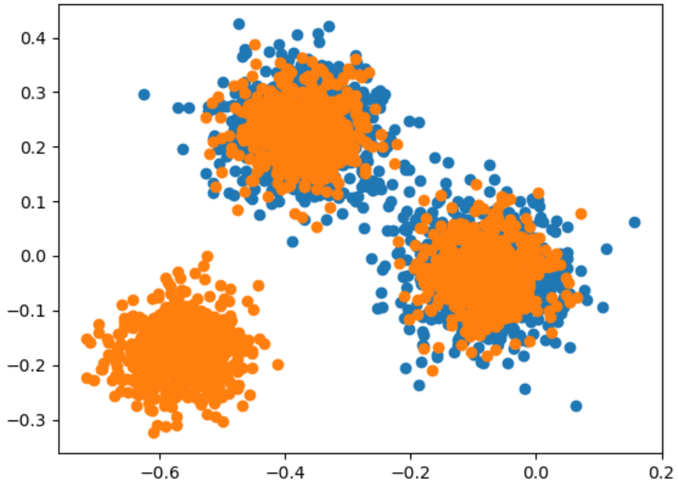
Empirical Results

DDS w/o Langevin for GMM-3:

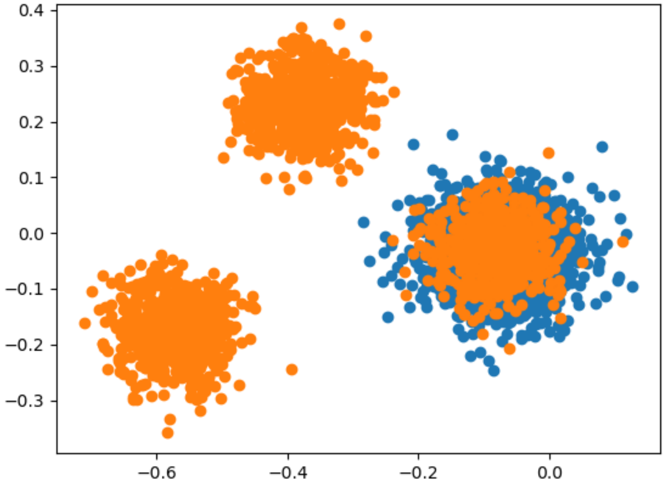
KL



LV



TB



Empirical Results

c. How about sample efficiency?

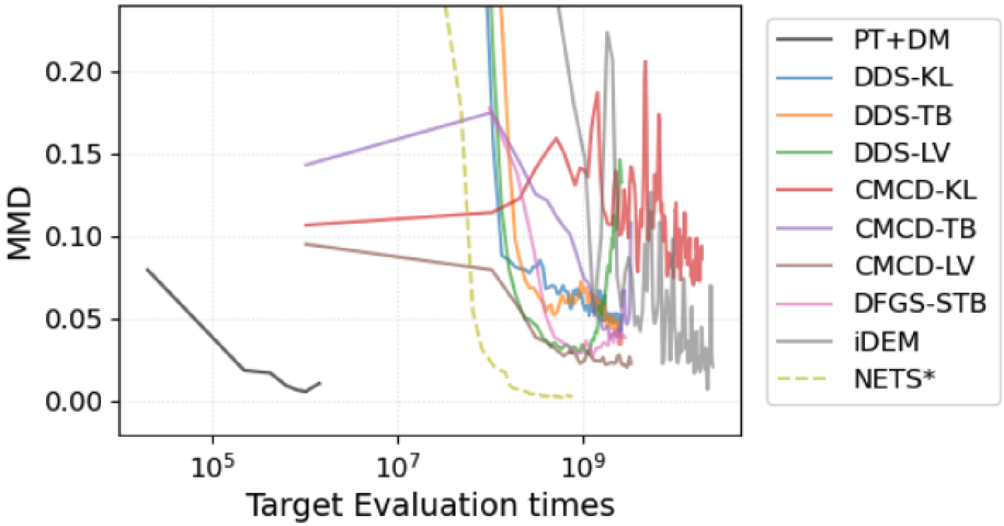
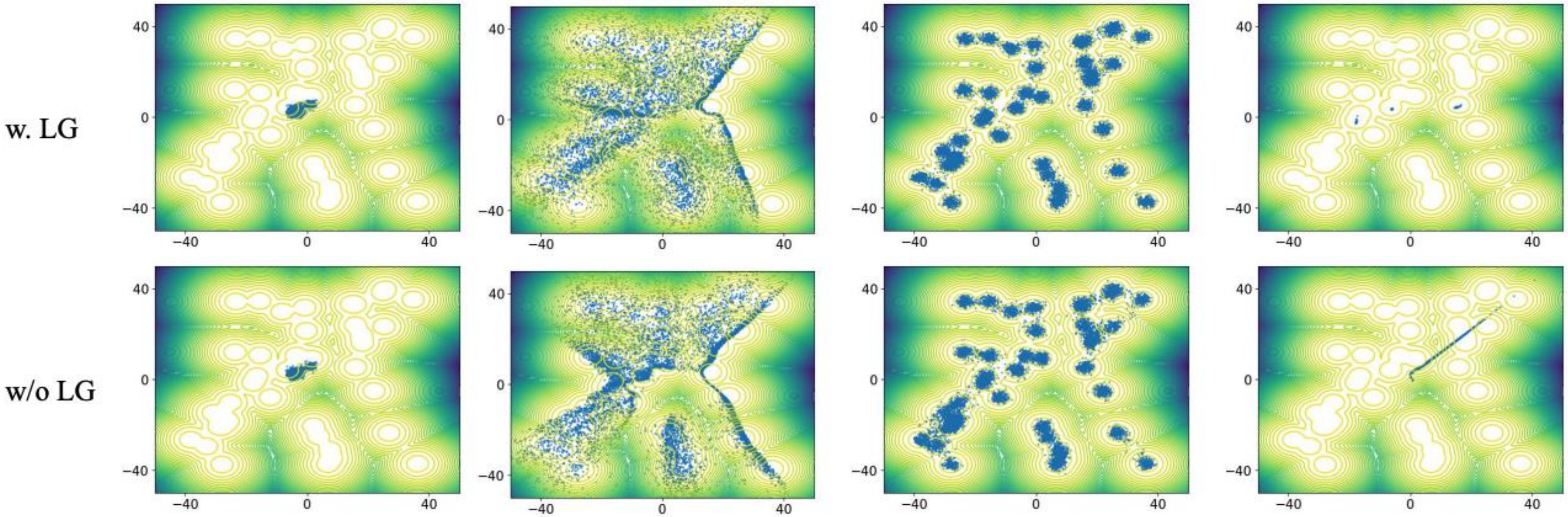


Figure 2: Sample quality vs target evaluation times for different approaches with different objectives on GMM-40 target. *NETS uses mode interpolation, which is distinct from that employed in others.

Empirical Results

d. PINN objective is different

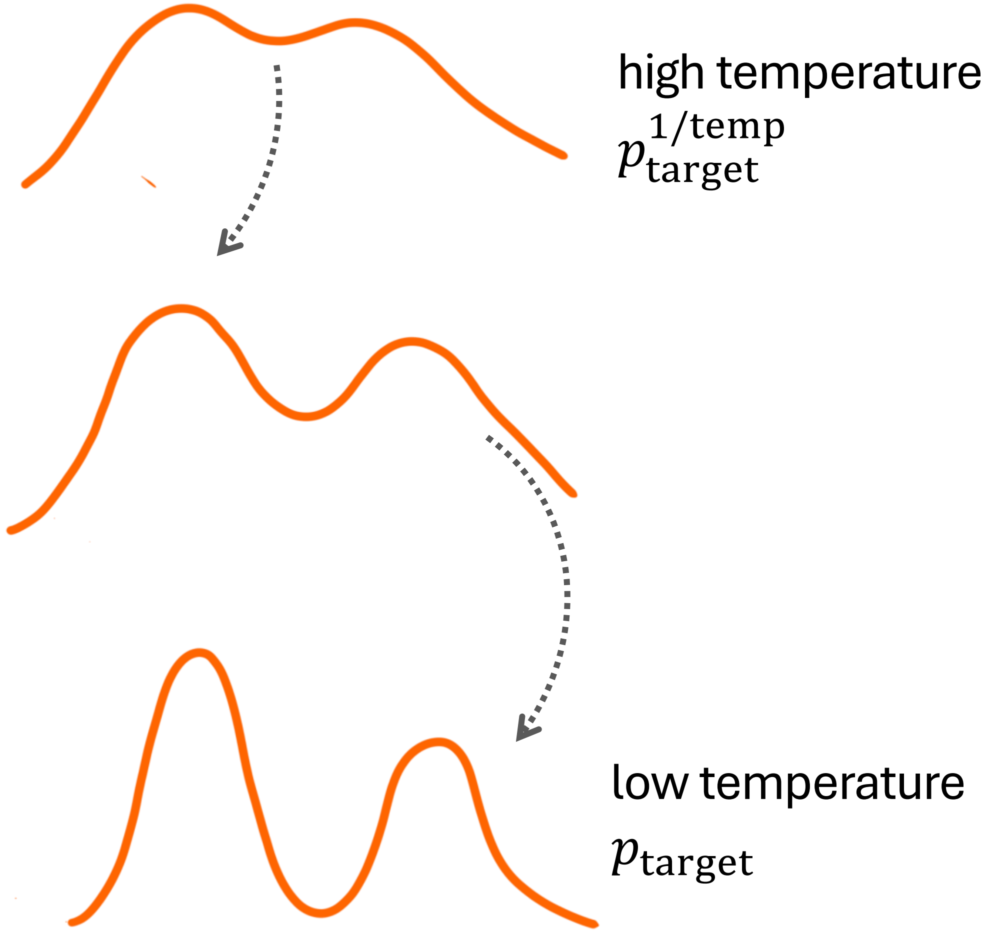
1. different interpolant
2. different prior size
3. "consistent" behavior



geom interpolate, prior $N(0, 2\boldsymbol{I})$ geom interpolate, prior $N(0, 30^2\boldsymbol{I})$ mode interpolate, prior $N(0, 2\boldsymbol{I})$ mode interpolate, prior $N(0, 30^2\boldsymbol{I})$

Parallel Tempering/Replica Exchange

- 🥹 SOTA MCMC in MD simulation
- 🥹 Highly parallel

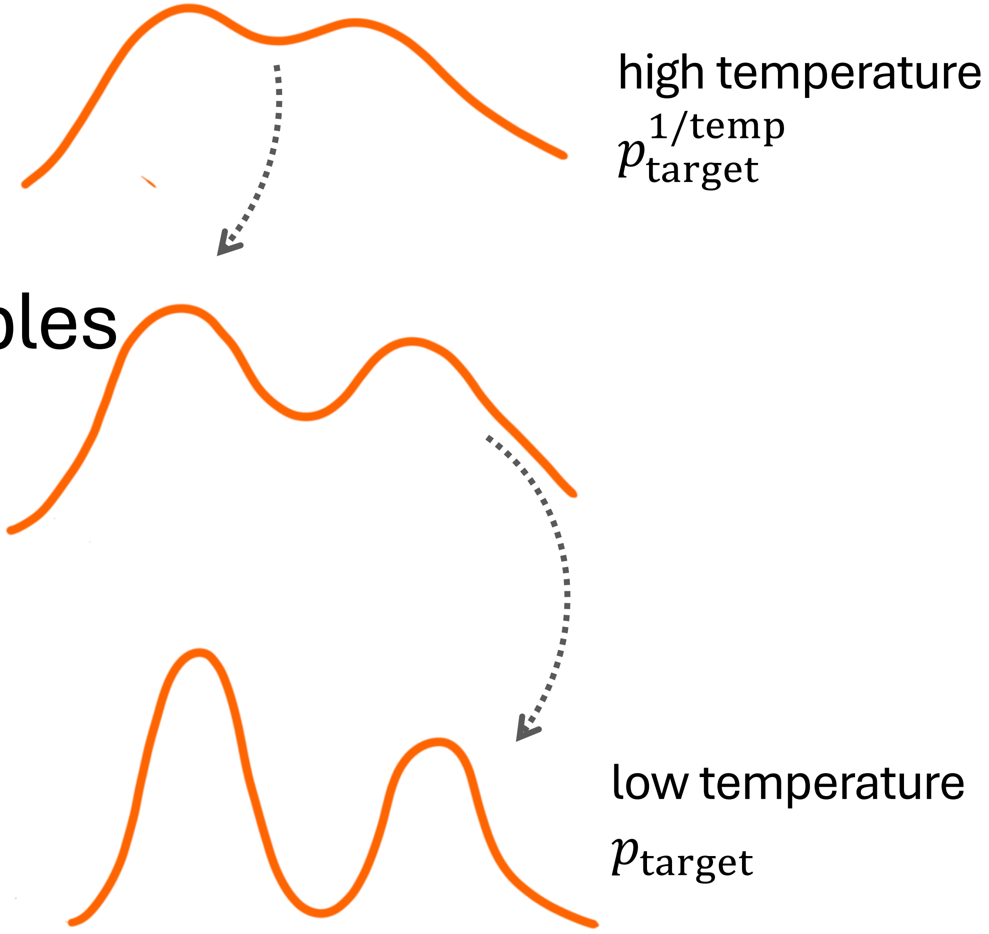


Parallel Tempering/Replica Exchange

Different use from neural samplers...

😔 Correlated samples

😔 Need more simulation for new samples



Parallel Tempering/Replica Exchange

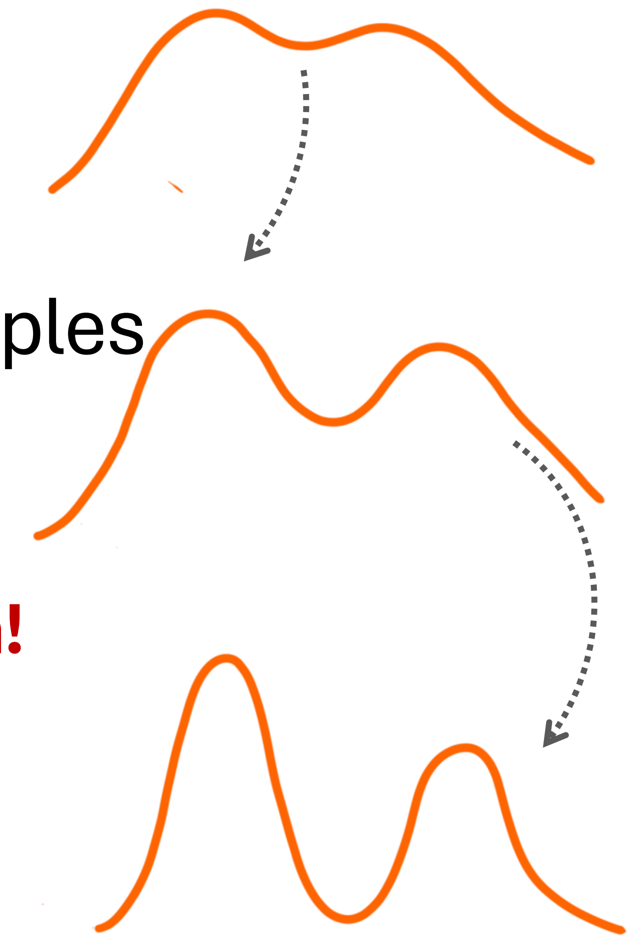
Different use from neural samplers...

😔 Correlated samples

😔 Need more simulation for new samples

Generative models can easily address them!

But is it worth it?



high temperature

$p_{\text{target}}^{1/\text{temp}}$

low temperature

p_{target}

Parallel Tempering/Replica Exchange

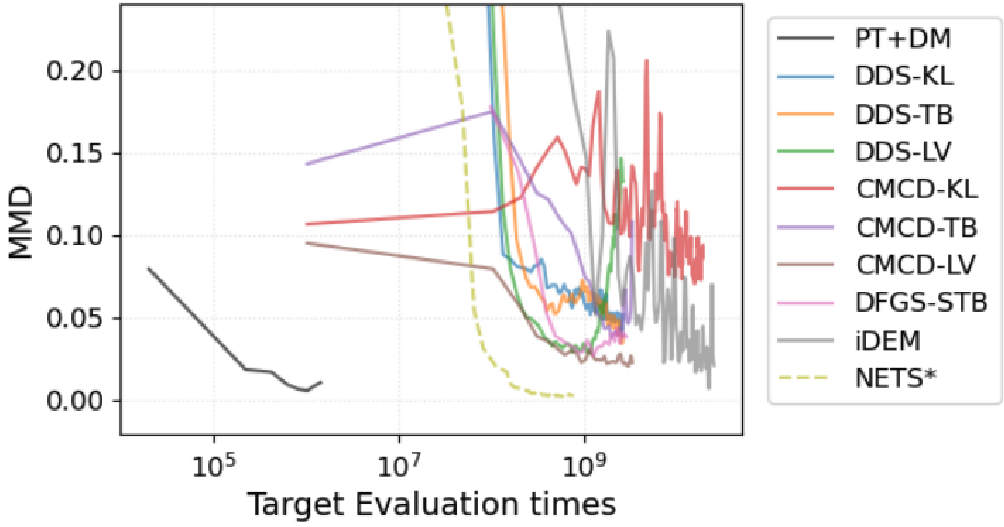
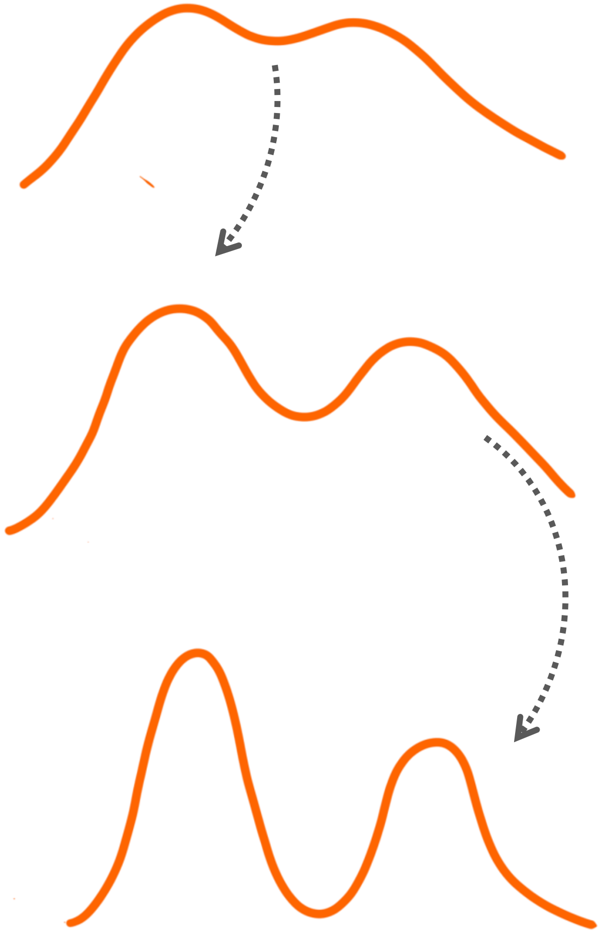


Figure 2: Sample quality vs target evaluation times for different approaches with different objectives on GMM-40 target. *NETS uses mode interpolation, which is distinct from that employed in others.



high temperature

$p_{\text{target}}^{1/\text{temp}}$

low temperature

p_{target}

Discussion & Takeaway

1. We **should not hide Langevin** gradient used

Discussion & Takeaway

1. We **should not hide Langevin** gradient used

2. If we need Langevin gradient anyway, we need to **talk about sample efficiency** (we should also be open to initialize using data)

Discussion & Takeaway

1. We **should not hide Langevin** gradient used

2. If we need Langevin gradient anyway, we need to **talk about sample efficiency** (we should also be open to initialize using data)

3. Improving **PT** is a promising direction (solve challenges with neural network ansatz)

Discussion & Takeaway

1. We **should not hide Langevin** gradient used

2. If we need Langevin gradient anyway, we need to **talk about sample efficiency** (we should also be open to initialize using data)

3. Improving **PT** is a promising direction (solve challenges with neural network ansatz)

4. **Better prior, interpolant, explorative objectives** still needed