Simulation methods for generative models

Monte Carlo methods, Score-based approaches, Posterior sampling and Simulation-based inference

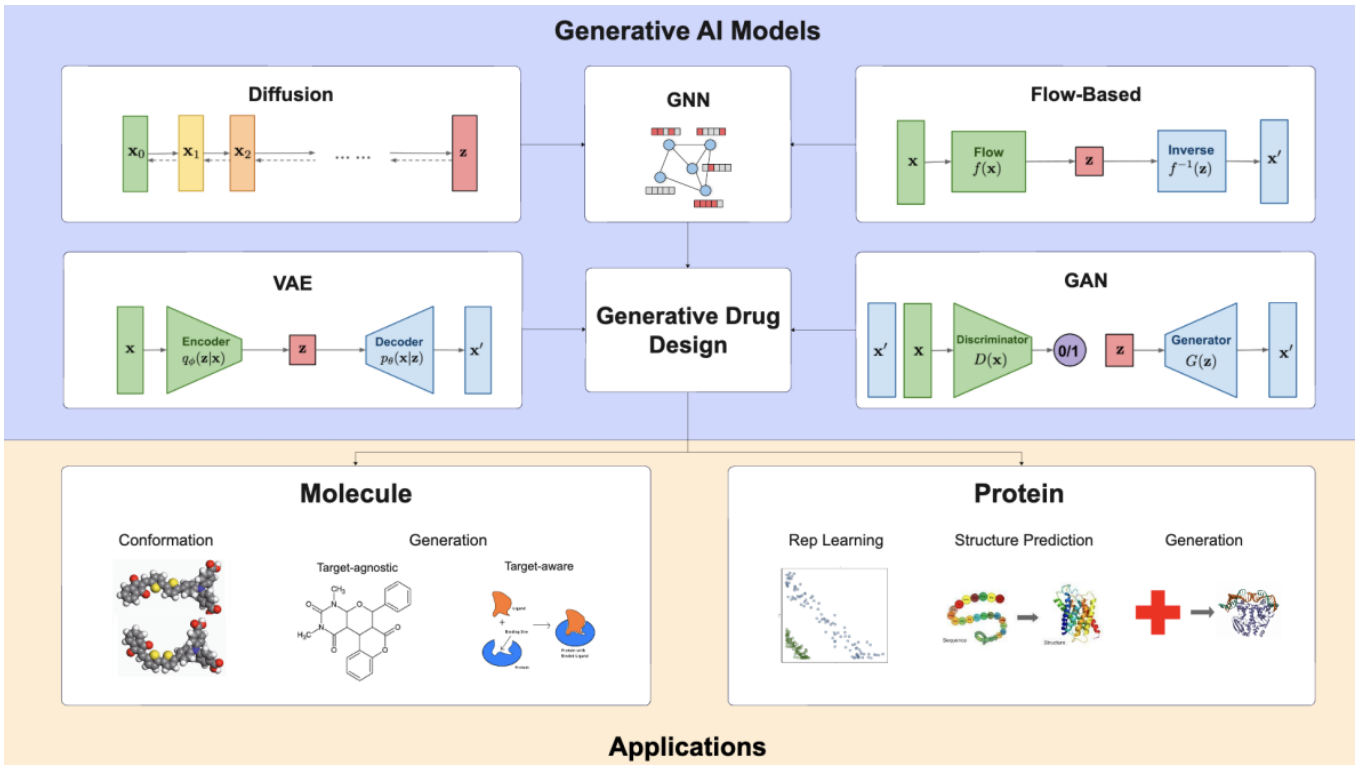
Sylvain Le Corff, Sorbonne Université, LPSM

Notebooks:
Some MCMC algoritms
Denoising diffusion in a toy setting

Generative modeling

Assumption consider a dataset $(X_i)_{1 \leq i \leq N}$ in $\mathbb{R}^{d_x}$ with unknown distribution π_{data}.

What estimate π_{data} and sample new instances from π_{data}.

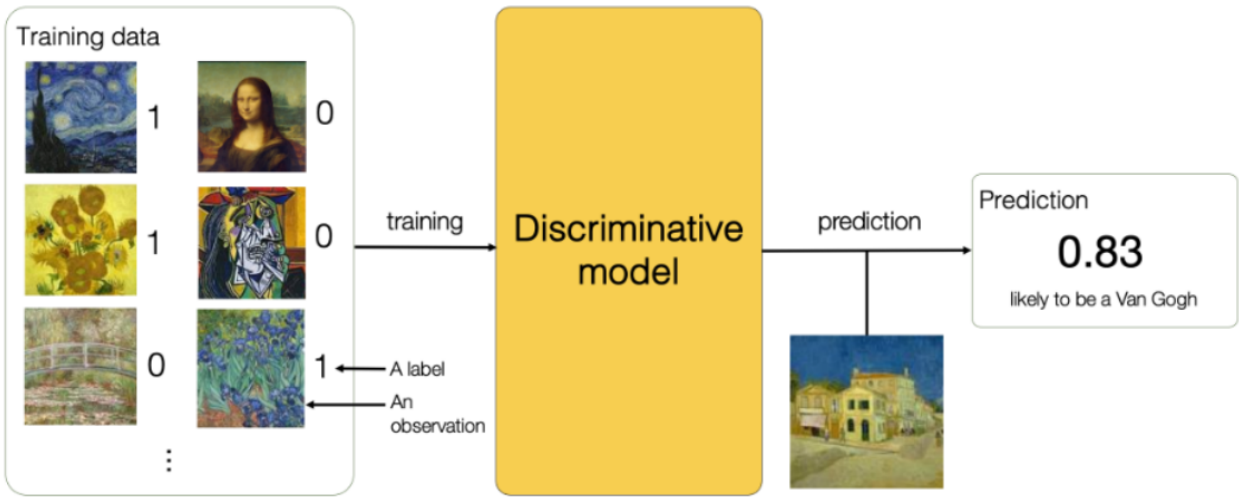


Generative modeling

Assumption consider an **input variable** $X \in \mathbb{R}^{d_x}$ and a **target observation** $Y \in \mathbb{R}^{d_y}$ or $Y \in \{1, \ldots, M\}$.

What a **generative model** is a statistical model for the joint distribution of (X, Y), a **discriminative model** is a statistical model for the conditional distribution of Y given X.

Generative learning consider a **parametric family** p_θ, $\theta \in \Theta$ and training data $\mathcal{D}$ to estimate the unknown parameter θ. [1]



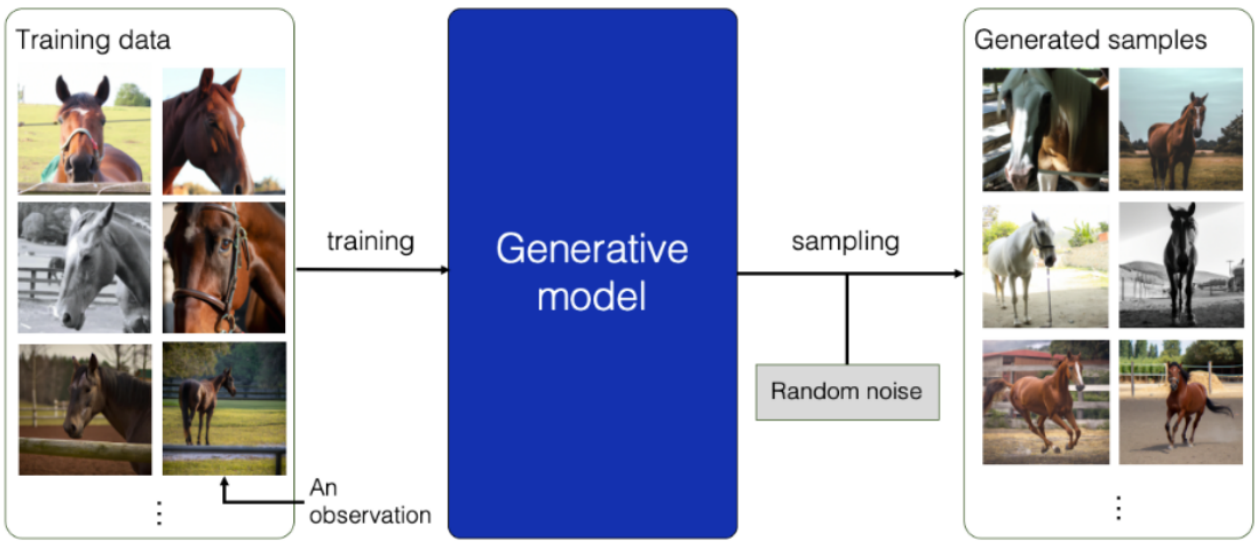
[1] Illustrations from https://vitalflux.com/generative-vs-discriminative-models-examples/

Generative modeling

Assumption consider an **input variable** $X \in \mathbb{R}^{d_x}$ and a **target observation** $Y \in \mathbb{R}^{d_y}$ or $Y \in \{1, \dots, M\}$.

What a **generative model** is a statistical model for the joint distribution of (X, Y), a **discriminative model** is a statistical model for the conditional distribution of Y given X.

Generative learning consider a **parametric family** p_θ, $\theta \in \Theta$ and training data $\mathcal{D}$ to estimate the unknown parameter θ. [2]



[2] Illustrations from https://vitalflux.com/generative-vs-discriminative-models-examples/

Generative modeling

(1) Estimate π_{data} with a **parametric** probability distribution p_θ.

1. Choose a suitable parametric form for p_θ.
   - $\rightsquigarrow$ In classical solutions, p_θ is parameterized using a Neural Network.
   - $\rightsquigarrow$ In most recent approaches, the score $\nabla \log \pi_{\text{data}}$ is direclty parameterized.
2. Data $(X^1, \dots, X^N) \sim \pi_{\text{data}}$.
   - $\rightsquigarrow$ How to train and use the score to generate new samples ?
   - $\rightsquigarrow$ Score-based generative models (Yang et al., 2024)

Controlled generation

(2) Perform controlled generation using p_{θ_*}.

$\rightsquigarrow$ Target distribution: weight p_{θ_*} with a function $x \mapsto g(x)$

$$\phi(\mathrm{d}x) = \frac{g(x)\mathsf{p}_{\theta_*}(\mathrm{d}x)}{\int g(z)\mathsf{p}_{\theta_*}(\mathrm{d}z)},$$

$\rightsquigarrow$ Posterior sampling: $g(x) = p(y|x)$.

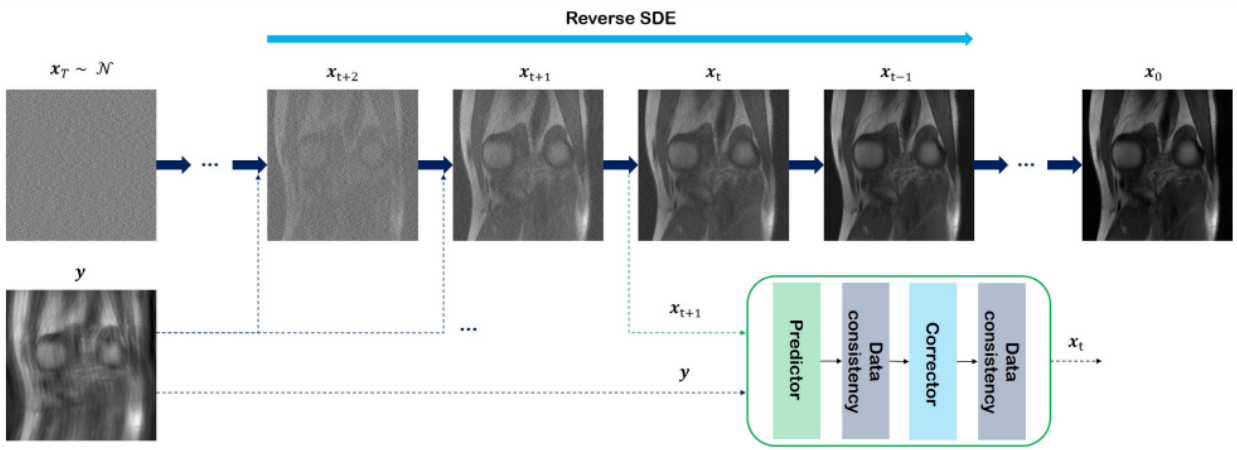
$\rightsquigarrow$ Inverse problem with $Y = f(X) + \varepsilon$ with $X \sim \pi_{\mathrm{data}}$

Controlled generation

(2) Perform controlled generation using p_{θ_*}.

$\rightsquigarrow$ Inverse problem with $Y = f(X) + \varepsilon$ with $X \sim \pi_{\text{data}}$

$\rightsquigarrow$ Use generative models to sample from the conditional law of X given Y[3].



[3] Illustration from (Karezouni et al., 2023)

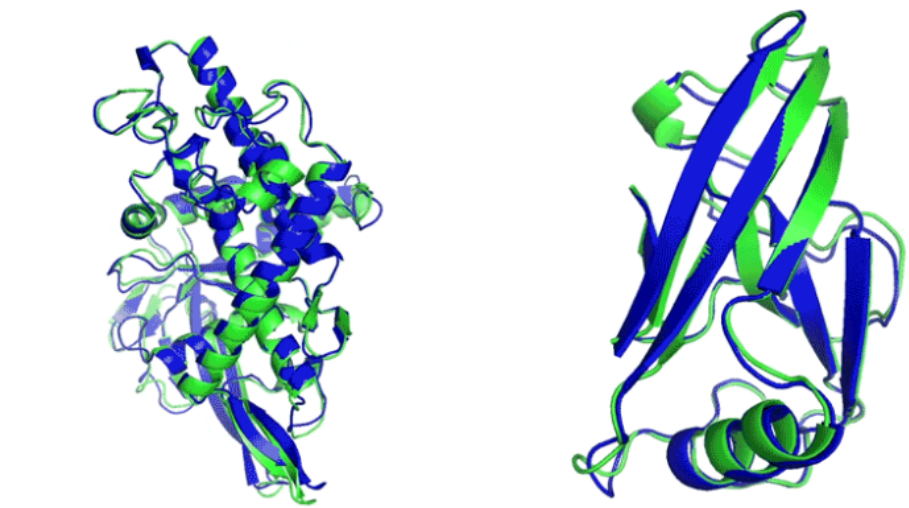
Generative modeling - Applications

Assumption access to samples from some **unknown** distribution π_{data}.

What sampling (approximately) synthetic instances from π_{data}

Predict a **protein's 3D structure from its amino acid sequence**.

$\rightsquigarrow$ Database of 200M proteins[4][5](Deepmind & European Molecular Biology Laboratory), trained with the 200k known conformations.



T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)

T1049 / 6y4f
93.3 GDT
(adhesin tip)

[4] https://alphafold.ebi.ac.uk/

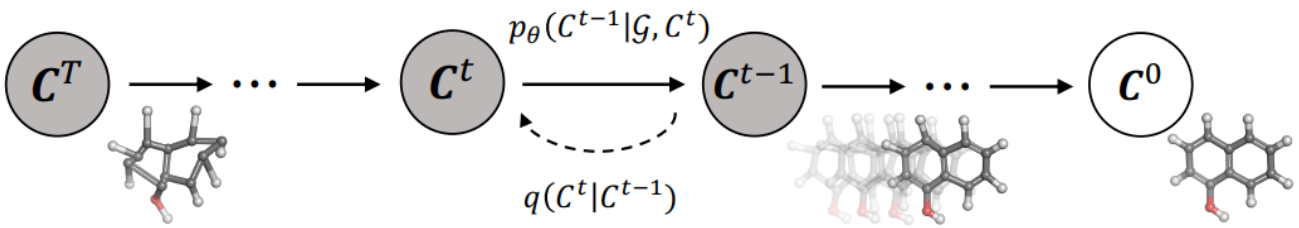
[5] https://github.com/google-deepmind/alphafold

Generative modeling - Applications

Assumption access to samples from some **unknown** distribution π_{data}.

What sampling (approximately) synthetic instances from π_{data}

Probabilistic model for generating molecular conformations
(GeoDiff, Xu et al., 2022)

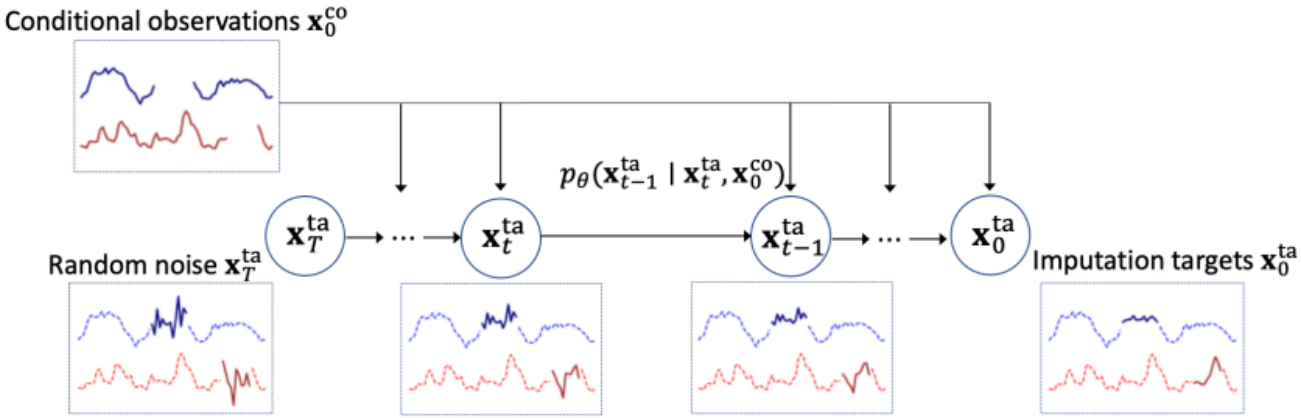


$\rightsquigarrow$ GEOM (37 million annotated molecular conformations annotated by energy): generates new structures + chemical toolkit to calculate conformation energy.[6]

[6] https://github.com/MinkaiXu/GeoDiff

Generative modeling - Applications

Probabilistic time series imputation (CSDI, Tashiro et al., 2021)



Healthcare dataset in PhysioNet Challenge 2012 (4000 clinical time series with 35 variables for 48 hours from intensive care unit).

Synthetic Data Generation for Privacy and Security (TabDDPM, Kotelnikov et al., 2023), etc.

Generative modeling - Applications

- ⇝ Sample **high-fidelity and diverse tunes**, (Jukebox, Dhariwal et al., 2020).
- ⇝ **Non-Intrusive Load Monitoring**, (DiffNILM, Sun et al., 2023).
- ⇝ Sample **Image super-resolution**, (Gao et al., 2023).
- ⇝ **Preliminary medical diagnostic + biomedical denoising**[7].

[7] https://www.academie-medecine.fr/wp-content/uploads/2024/03/Rapport-Systemes-dIA-generative-en-sante.pdf

Generative modeling - Tractable likelihood ?

Tractable likelihood allows for straightforward comparison between models, and straightforward optimization.

Flow-based generative models (Rezende & Mohamed, 2015, Normalizing Flows) or **Autoregressive models** (Van den Oord et al., 2016, PixelCNN)

The set of models with a tractable likelihood is constrained.

- $\rightsquigarrow$ The distribution is factorized as a product of conditional distributions
- $\rightsquigarrow$ The distribution is modeled as an invertible transformation of a base distribution

Generative modeling - Energy based models

Energy-based models specify only the unnormalized negative log-probability (Energy function).

Easy to leverage advances in architectures originally developed for classification or regression, and flexible to special-purpose architectures.

- $\rightsquigarrow$ **Image generation** (Du et al., 2019)
- $\rightsquigarrow$ **Natural language processing** (Deng et al., 2020)
- $\rightsquigarrow$ **Reinforcement learning** (Haarnoja et al., 2018)

Generative modeling - Energy based models

The target random variable take values in $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$ and the target distribution is written[8]:

$$x \mapsto \pi_\theta(x) \propto \exp\left(-\mathrm{E}_\theta(x)\right) = \frac{\exp\left(-\mathrm{E}_\theta(x)\right)}{\int \exp\left(-\mathrm{E}_\theta(u)\right) \mathrm{d}u},$$

where θ is an unknown parameter and E_θ is the energy function.

Normalizing constant/partition function:

$$Z_\theta = \int \exp\left(-\mathrm{E}_\theta(u)\right) \mathrm{d}u.$$

Gradient-based maximum likelihood requires to compute

$$x \mapsto \nabla_\theta \log \pi_\theta(x) = -\nabla_\theta \mathrm{E}_\theta(x) - \nabla_\theta \log Z_\theta.$$

[8] (Song & Kingma, 2021)

Generative modeling - Energy based models

Target $x \mapsto \pi_\theta(x) \propto \exp\left(-\mathrm{E}_\theta(x)\right)$

Gradient-based maximum likelihood requires to compute

$$\nabla_\theta \log Z_\theta = Z_\theta^{-1} \int \nabla_\theta \exp\left(-\mathrm{E}_\theta(u)\right) \mathrm{d}u$$

$$= \int \left\{-\nabla_\theta \mathrm{E}_\theta(u)\right\} Z_\theta^{-1} \exp\left(-\mathrm{E}_\theta(u)\right) \mathrm{d}u = \int \left\{-\nabla_\theta \mathrm{E}_\theta(u)\right\} \pi_\theta(u) \mathrm{d}u.$$

Therefore

$$\nabla_\theta \log Z_\theta = \mathbb{E}_{\pi_\theta}\left[-\nabla_\theta \mathrm{E}_\theta(X)\right]$$

where $\mathbb{E}_\mu[f(X)]$ denotes the expectation of $f(X)$ when $X \sim \mu$.

$\rightsquigarrow$ Possible to train an EBM by providing a Monte Carlo estimate of $\nabla_\theta \log Z_\theta$ which **requires to obtain samples from** π_θ.

Generative modeling - latent data

In some situations, observations are partial and do not contain some variables of interest.

Given a generative process for the data, we might be interested in reconstructing the distribution of the missing variables given the data.

We introduce a family of joint probability distributions $(z, x) \mapsto p_\theta(z, x)$, $\theta \in \Theta$ on $(\mathsf{Z} \times \mathsf{X}, \mathcal{Z} \times \mathcal{X})$ where Z **is a latent variable and** X **is the observation**.

In this setting, for all θ, x, z,

$$p_\theta(z, x) = p_\theta(z) p_\theta(x|z)$$

and π_{data} is estimated by the marginal

$$p_\theta(x) = \int p_\theta(z, x) \mathrm{d}z = \int p_\theta(z) p_\theta(x|z) \mathrm{d}z$$

Generative modeling - latent data

Major drawback: $p_\theta(x)$ is not available explicitly, nor the conditional distribution of the latent data given the observation:

$$p_\theta(z|x) = \frac{p_\theta(z)p_\theta(x|z)}{p_\theta(x)} = \frac{p_\theta(z)p_\theta(x|z)}{\int p_\theta(z)p_\theta(x|z)\mathrm{d}z}$$

$\rightsquigarrow$ **Challenging to train the model** !

$\rightsquigarrow$ **Challenging to sample latent data given the observations** !

Generative modeling - latent data - DLM[9]

(Kingma et al., 2013): deep latent variable model for multivariate Benoulli data

$X \in \{0,1\}^D$ and conditionally on a variable $Z \in \mathbb{R}^d$, $(X_1, \ldots, X_D)$ are independent with Bernoulli distribution with parameters $\mathbf{p}_\theta(X) = (p_{1,\theta}(X), \ldots, p_{D,\theta}(X))$, where $\mathbf{p}_\theta(X)$ is the output of a Multi-layer Perceptron with input X and parameters θ (weights and biases).

In this example, the input variable has a prior distribution $Z \sim \mathcal{N}(0, I_d)$ and, for any value of θ, the conditional distribution of Z given X is not available explicitly.

[9] Deep Latent Models

Generative modeling - latent data - HMM[10]

An observation is a sequence $X = X_{1:n}$ and the latent data is a sequence $Z = Z_{1:n}$.

Bivariate Markov chain $(Z_k)_{k\in\mathbb{N}} = (Z_k, X_k)_{k\in\mathbb{N}}$ where the transition is defined as follows.

$$\text{Conditionally on } Z_{0:k-1},\ X_k \sim Q_\theta(X_{k-1}, \cdot),$$
$$\text{Conditionally on } (X_k, Z_{0:k-1}),\ Y_k \sim G_\theta(X_k, \cdot),$$

with Q_θ a Markov kernel on $\mathsf{X} \times \mathcal{B}(\mathsf{X})$ and G_θ a Markov kernel on $\mathsf{X} \times \mathcal{B}(\mathsf{Y})$.

We know how to compute $p_\theta(Z_{1:n})$ and $p_\theta(X_{1:n}|Z_{1:n})$ but $p_\theta(X_{1:n})$ and $p_\theta(Z_{1:n}|X_{1:n})$ are intractable.

[10] Hidden Markov Models

Topics covered in this tutorial

How to sample from π_θ when the distribution is known up to a multiplicative constant - **MCMC**

How to sample from $\pi_\theta(z|x)$ when using latent data - **VAE**.

How to sample from π_θ when only $\nabla \log \pi_\theta$ is estimated - **Score-based diffusion models**.

Applications of all approaches

Sampling with MCMC algorithms

Bayesian setting

In a Bayesian setting, a parameter Z is embedded with a prior distribution p_θ and the observations are given by a probabilistic model:

$$X \sim \ell_\theta(\cdot|Z)\,.$$

The inference is then based on the posterior distribution:

$$\boxed{\pi_\theta(z|X) = \frac{p_\theta(z)\ell_\theta(X|z)}{\int p_\theta(u)\ell_\theta(X|u)\mathrm{d}u}\,.}$$

In most cases the normalizing constant is not tractable:

$$\pi_\theta(Z|X) \propto p_\theta(Z)\ell_\theta(X|Z)\,.$$

MCMC: rationale

Let X_1 be any starting point.

- For a given target distribution π, choose a π-reversible transition kernel with density k:

$$\pi(x)k(x,x') = \pi(x')k(x',x) \quad \text{[Reversibility]}$$

- Sample a Markov chain $X_1, \ldots, X_n$ with kernel k and compute

$$\hat{\pi}_n(f) = \frac{1}{n}\sum_{i=1}^{n} f(X_i)$$

to approximate $\pi(f) = \int f(x)\pi(\mathrm{d}x)$.

$\Rightarrow$ Doest it converge ? What is the rate of convergence ?

MCMC: rationale

Under regularity assumptions, if π is a stationary distribution:

- **Ergodic theorem**, under which condition can we establish, for $f \in \mathrm{L}^1(\pi)$,

$$\hat{\pi}(f) = \frac{1}{n} \sum_{i=1}^{n} f(X_i) \underset{a.s.}{\longrightarrow} \int f(x)\pi(x)\mathrm{d}x \,.$$

- **Central limit theorem**, under which condition can we establish, for $f \in \mathrm{L}^1(\pi)$,

$$\frac{\sqrt{n}}{\sigma_{\pi,q,f}} \left[\frac{1}{n} \sum_{i=1}^{n} f(X_i) - \int f(x)\pi(x)\mathrm{d}x \right] \xrightarrow{\mathcal{D}} \mathcal{N}(0,1) \,.$$

Key tool : the Accept-Reject algorithm

Assume we know that $\pi(x) \leq Mr(x)$ and that we know how to sample from r.

1. Sample $X \sim r$ and $U \sim U([0,1])$.
2. If
$$U \leq \frac{\pi(X)}{Mr(X)},$$
accept X.
3. Else go to 1.

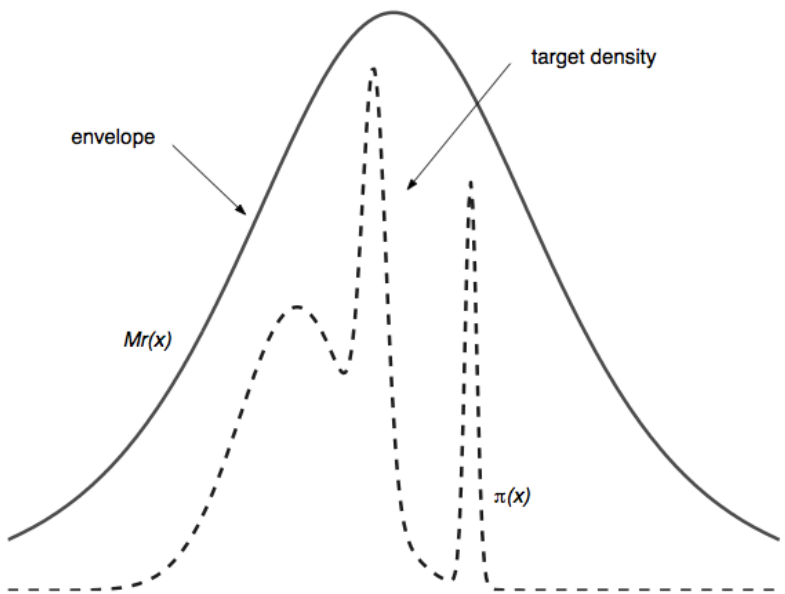


Illustration of the Accept-Reject method (Cappé, Moulines, Ryden 2005).

The Metropolis-Hastings algorithm

- Objective target density π.
- Instrumental transition density $q(x, y)$.

Given X_k,

1. Generate $Y_{k+1} \sim q(\cdot, X_k)$.
2. Set

$$X_{k+1} = \begin{cases} Y_{k+1} & \text{with probability } \alpha(X_k, Y_{k+1}) \,, \\ X_k & \text{with probability } 1 - \alpha(X_k, Y_{k+1}) \,. \end{cases}$$

where

$$\boxed{\alpha(x, y) = 1 \wedge \frac{\pi(y)}{\pi(x)} \frac{q(y, x)}{q(x, y)}} \,.$$

$\Rightarrow$ No restriction on π and q, with this choice of α the algorithm produces a Markov chain with stationary distribution π.

The Metropolis-Hastings algorithm

```
def HM_monte_carlo(n_samples, log_prob, initial_state, step_size = 0.1):
    """
    Inputs
    ----------
    n_samples: number of samples to return
    log_prob: opposite of the loglikelihood to sample from
    initial_state: initial sample
    step_size: standard deviation of the proposed moves

    Outputs
    -------
    samples: samples from the MCMC algorithm
    accepted: array of 0 and 1 to display which proposed moves have been accepted
    """
```

The Metropolis-Hastings algorithm

```python
def HM_monte_carlo(n_samples, log_prob, initial_state, step_size = 0.1):

    initial_state = np.array(initial_state)

    samples  = [initial_state]
    accepted = []

    size = (n_samples,) + initial_state.shape[:1]

    # random variable to sample proposed moves
    epsilon = st.norm(0, 1).rvs(size)

    for noise in tqdm(epsilon):

        q_new = samples[-1] + step_size*noise

        # acceptance rate
        old_log_p = log_prob(samples[-1])
        new_log_p = log_prob(q_new)

        if np.log(np.random.rand()) < old_log_p - new_log_p:
            samples.append(q_new)
            accepted.append(True)
        else:
            samples.append(np.copy(samples[-1]))
            accepted.append(False)

    return (np.array(samples[1:]),np.array(accepted),)
```

Independent case

In this case $q(x,y) = g(y)$.

1. Generate $Y_{k+1} \sim g(\cdot)$.
2. Set

$$X_{k+1} = \begin{cases} Y_{k+1} & \text{with probability } \alpha(X_k, Y_{k+1})\,, \\ X_k & \text{with probability } 1 - \alpha(X_k, Y_{k+1})\,. \end{cases}$$

where

$$\boxed{\alpha(x,y) = 1 \wedge \frac{\pi(y)}{\pi(x)} \frac{g(x)}{g(y)}}\,.$$

Alternative to importance sampling and Accept-Reject algorithms.

Independent case

The samples are not i.i.d. but, if there exists M such that $\pi(x) \leqslant Mg(x)$ then

$$\|K^n(x,\cdot) - \pi\|_{tv} \leqslant \left(1 - \frac{1}{M}\right)^n \quad \text{(Ergodicity)}$$

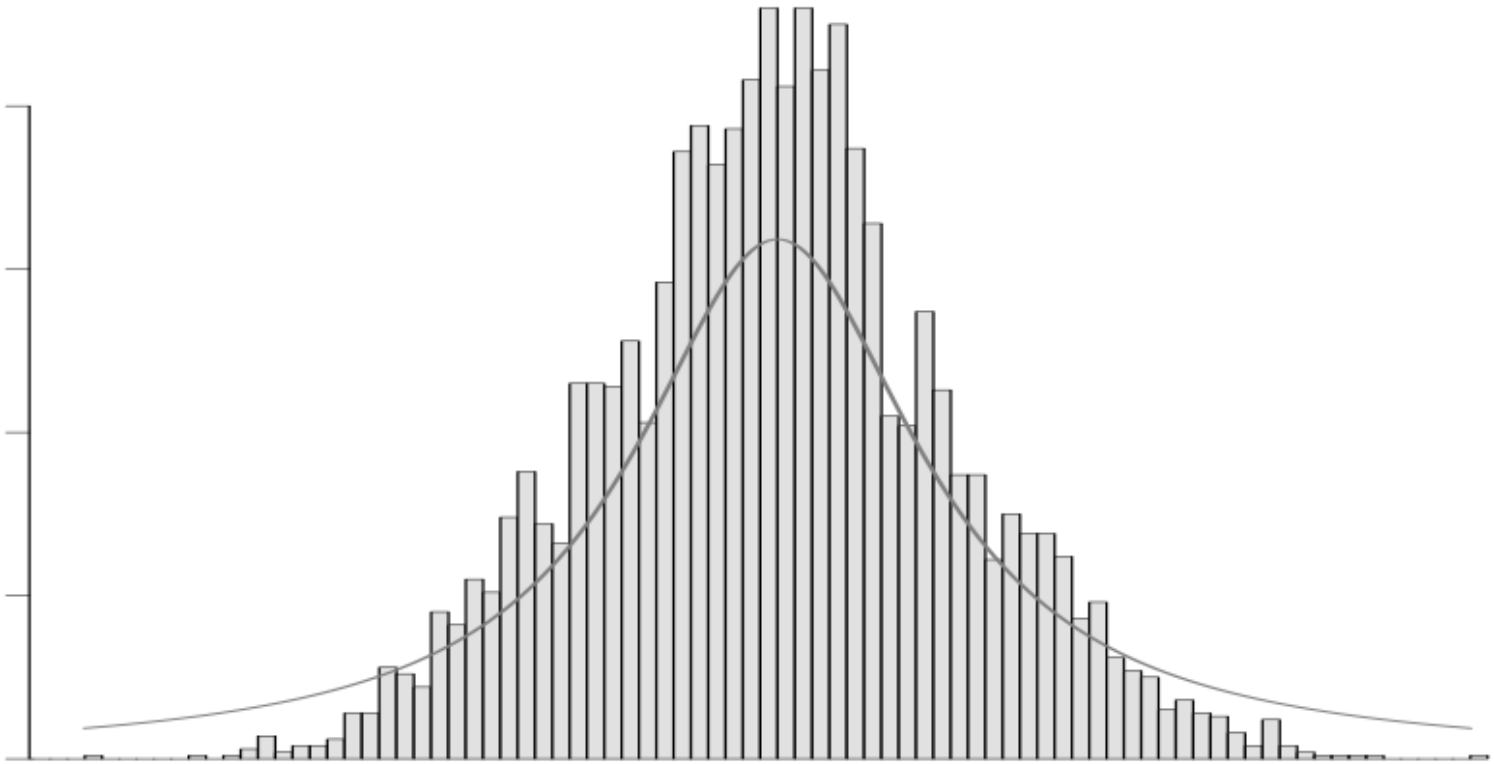
(Roberts, Tweedie 1996), (Mengersen, Tweedie 1996).

Expected acceptance probability is $1/M$, no need to know M.

If the majoration condition does not hold, no geometric ergodicity.

Cauchy vs Normal (I)

- Target distribution: $\pi(x) \propto (1+x^2)^{-1}$.
- Proposal distribution: $g(y) \sim \mathcal{N}(0,1)$.



Histogram of IMH with 5000 samples.

Random walk Metropolis-Hastings

The proposal mechanism is given by $Y_{k+1} = X_k + \varepsilon_{k+1}$, where ε_{k+1} is independent of X_{k+1}. The proposal distribution is of the form $q(x, y) = q(y - x)$ with q is symmetric.

1. Generate $Y_{k+1} \sim q(X_k, \cdot)$.
2. Set

$$X_{k+1} = \begin{cases} Y_{k+1} & \text{with probability } \alpha(X_k, Y_{k+1}) \,, \\ X_k & \text{with probability } 1 - \alpha(X_k, Y_{k+1}) \,. \end{cases}$$

where

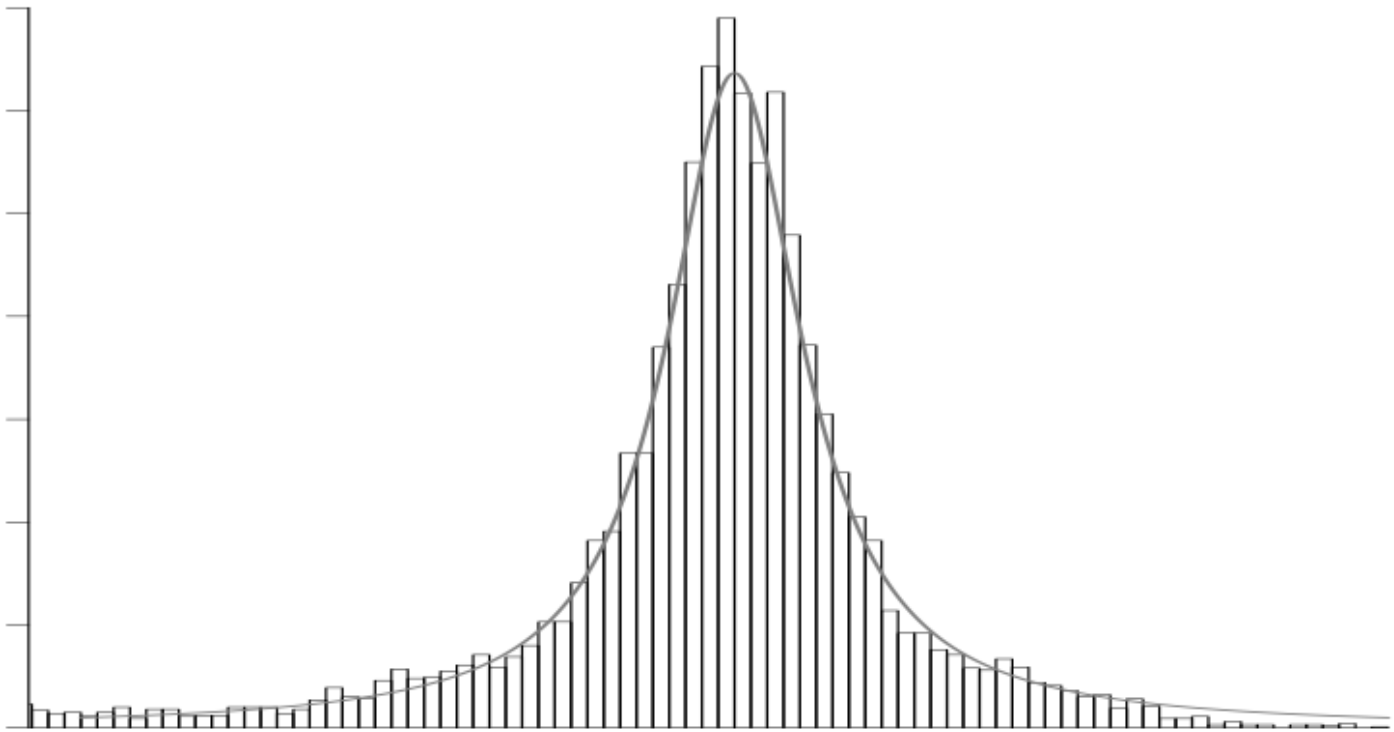
$$\boxed{\alpha(x, y) = 1 \wedge \frac{\pi(y)}{\pi(x)}} \,.$$

Using random walk moves prevents from being uniformly ergodic (Robert, Casella 2004).
But still, geometric ergodicity.

Cauchy vs Normal (II)

- Target distribution: $\pi(x) \propto (1+x^2)^{-1}$.
- Proposal distribution: $\mathcal{N}(0,1)$.

$$\alpha(x,y) = 1 \wedge \frac{1+x^2}{1+y^2}$$



Histogram of IMH with 10000 samples.

Improving Metropolis-Hastings using gradient information

Langevin equation associated with π:

$$\mathrm{d}X_t = (\Sigma/2)\nabla \log \pi(X_t)\mathrm{d}t + \Sigma^{1/2}\mathrm{d}W_t\,,$$

where W is a d-dimensional Brownian motion.

Under appropriate regularity assumptions, the generated dynamic is ergodic with unique invariant distribution π.

Solving this equation analytically would allow to sample exacty from π.
Not tractable in practice!

Another family of proposals is based on the Euler-Maruyama discretization of the equation...
Proposal mechanism of the form

$$Y_{k+1} = X_k + \frac{h\sigma^2}{2}\nabla \log \pi(X_k) + \sqrt{h}\sigma\varepsilon_{k+1}\,.$$

The MALA algorithm

```
def MALA_monte_carlo(n_samples, log_prob, initial_state, step_size = 0.1):
    """
    Inputs
    ----------
    n_samples: number of samples to return
    log_prob: opposite of the loglikelihood to sample from
    initial_state: initial sample
    step_size: standard deviation of the proposed moves

    Outputs
    -------
    samples: samples from the MCMC algorithm
    accepted: array of 0 and 1 to display which proposed moves have been accepted
```

The MALA algorithm

```
def MALA_monte_carlo(n_samples, log_prob, initial_state, step_size = 0.1):

    initial_state = np.array(initial_state)

    gradV = grad(log_prob)

    samples  = [initial_state]
    accepted = []

    size = (n_samples,) + initial_state.shape[:1]

    # random variable to sample proposed moves
    epsilon = st.norm(0, 1).rvs(size)
    step    = 0.5/(step_size**2)
    for noise in tqdm(epsilon):

        grad_new = gradV(samples[-1])
        mean_new = samples[-1] - step*grad_new
        q_new    = mean_new + step_size*noise

        grad_y   = gradV(q_new)
        mean_y   = q_new - step*grad_y

        # acceptance rate
        old_log_p = log_prob(samples[-1]) + step*np.dot(q_new-mean_new,q_new-mean_new)
        new_log_p = log_prob(q_new) + step*np.dot(samples[-1]-mean_y,samples[-1]-mean_y)

        if np.log(np.random.rand()) < old_log_p - new_log_p:
            samples.append(q_new)
            accepted.append(True)
        else:
            samples.append(np.copy(samples[-1]))
            accepted.append(False)

    return (np.array(samples[1:]),np.array(accepted),)
```

The MALA algorithm

If σ is chosen **either too small or too large**, the convergence can be arbitrarily slow:

- σ small
  Many moves are likely to be accepted.
  The chain visits the state-space very slowly.

- σ large
  Proposed moves often rejected.
  The algorithm may be stuck for a long time.

Challenge: scaling issues and high dimensionality

How to **choose the scaling (σ)** of the algorithm to optimize efficiency ?

Scaling problem mainly studied for:

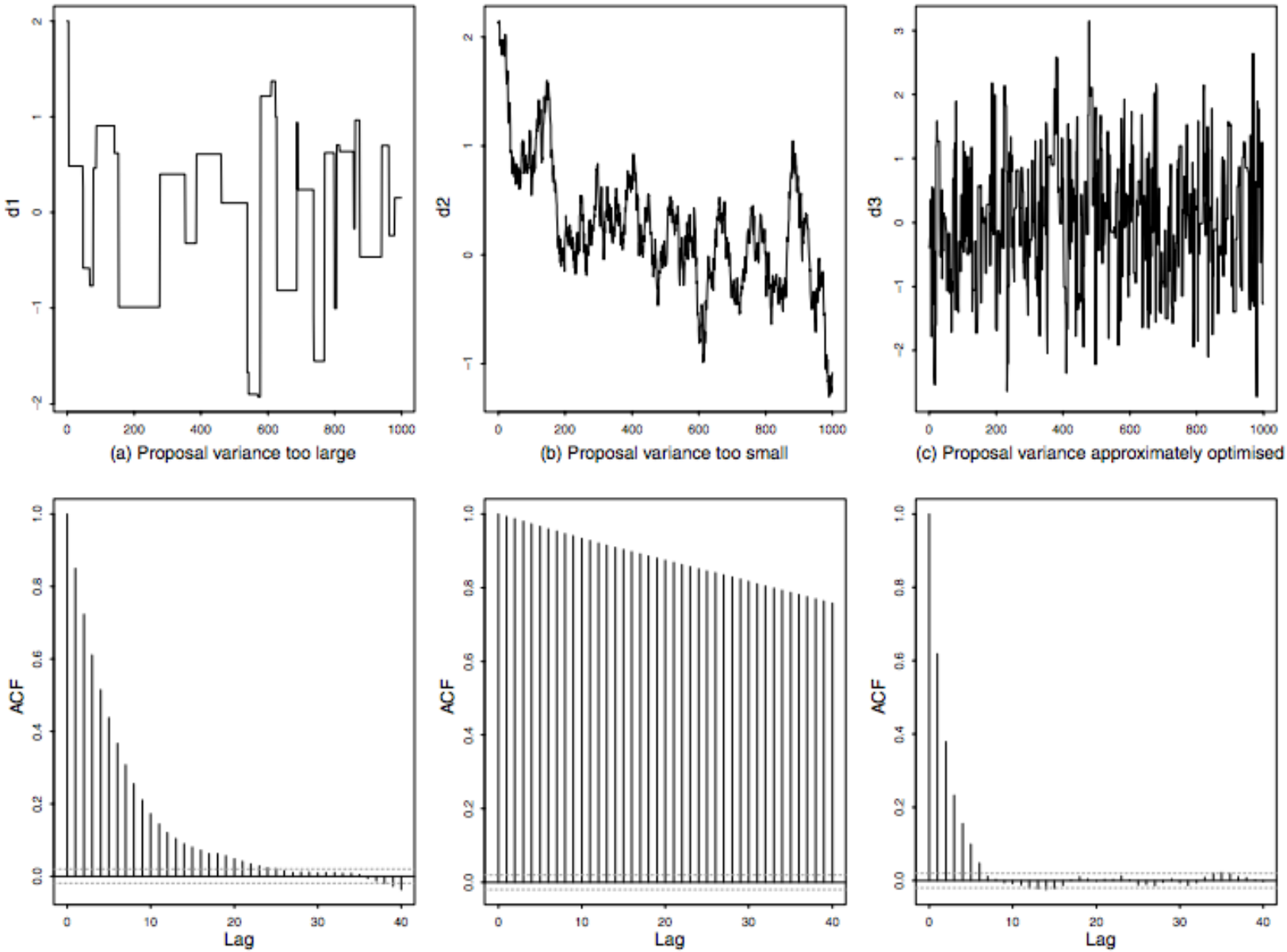
1. Random walk Metropolis-Hastings (RWM)
   - Proposal mechanism of the form $Y_{k+1} = X_k + \sigma\varepsilon_{k+1}$.
   - Acceptance rate:

$$\boxed{\alpha(x,y) = 1 \wedge \frac{\pi(y)}{\pi(x)}}.$$

2. Metropolis-Adjusted Langevin Algorithm (MALA)
   - Proposal mechanism of the form

$$Y_{k+1} = X_k + \frac{h\sigma^2}{2}\nabla \log \pi(X_k) + \sqrt{h}\sigma\varepsilon_{k+1}\,.$$

How to choose scaling



Optimal scaling

- The influence of scaling is better understood with high dimensional settings.
  $\Rightarrow$ Consider a state-space $\mathbb{R}^d$ when $d \to +\infty$.

- Each component of the Markov chain converges weakly to a diffusive limit.

- The choice of scaling can be obtained based on the behavior of this diffusive limit

1. Target distribution: $\pi_d(x) = \prod_{k=1}^{d} f(x_k)$.
2. RW proposal: $q_d(x) \sim \mathcal{N}(0, \sigma^2 I_d/d)$.

Optimal scaling

σ_d is of the form $\sigma_d^2 = \ell^2/d$.

Let $(Z_t)_{t\geq 0}$ be the linear interpolation of the Markov chain $\left(X_{k,1}^d\right)_{k\geq 0}$ after time rescaling. $(Z_t)_{t\geq 0}$ converge weakly to the diffusion process $(Z_t)_{t\geq 0}$:

$$\mathrm{d}Z_t = h(\ell)^{1/2}\mathrm{d}B_t + \frac{1}{2}h(\ell)\nabla \log \pi(Z_t)\mathrm{d}t$$

with $h(\ell) = 2\ell^2\Phi\left(-\sqrt{I}\ell/2\right)$.

By choosing the value $\ell_\star$ of ℓ which maximizes $h(\ell)$, the asymptotic acceptance rate is

$$\boxed{A(\ell_\star) \sim 0.234}.$$

For the MALA algorithm, by choosing $\sigma_d^2 = \ell/d^{1/3}$,

$$\boxed{A(\ell_\star) \sim 0.574}.$$

These MCMC algorithms, even optimally scaled, remain inefficient in high dimensional settings.

Other scaling results

- (Roberts et al., AAP, 2012)
  π^d is confined to the d-dimensional hypercube $(0,1)^d$ and twice continuously differentiable on the hypercube with bounded derivatives.

  Scaling $\sigma_d = \ell/d$.

  The asymptotic optimal acceptance rate obtained by maximizing the speed of the limiting diffusion is then equal to 0.1353.

- (Jourdain et al., AAP & Bernoulli, 2015)
  Initial distribution is not π.
  Same scaling using the weak formulation.

- (Durmus et al., JAP, 2017)
  Same scaling using the weak differentiability assumptions.

- Fast MALA (Durmus et al. 2017, AAP): second order discretization of the SDE: scaling of order $1/5$.

These MCMC algorithms, even optimally scaled, remain inefficient in high dimensional settings.

Extensions

- In practice the scale can be adapted on the fly to optimize the acceptance rate. [Adaptive algorithms].
- The MCMC algorithm can still have trouble capturing multimodality (trapped in local modes).
- [Parallel tempering] Design several target densities π^{T_k} with $T_1 \geq T_2 \cdots \geq T_p = 1$.
- Swaps between states of adjacent levels are proposed to allow an exchange of information.

And for dynamical data ?

Population dynamics in a predator-prey system

$$\mathrm{d}Z_t = \alpha_\theta(Z_t)\mathrm{d}t + \begin{pmatrix} Z_1(t) & 0 \\ 0 & Z_2(t) \end{pmatrix} \Gamma \mathrm{d}W_t$$

with

$$\alpha_\theta(Z_t) = \begin{pmatrix} Z_1(t)(a_{10} - a_{11}Z_1(t) - a_{12}Z_2(t)) \\ Z_2(t)(-a_{20} + a_{21}Z_1(t) - a_{22}Z_2(t)) \end{pmatrix}$$

The observation model is given by

$$X_{t_k} = \begin{pmatrix} c_1 Z_1(t_k)\mathrm{e}^{\varepsilon_k^{(1)}} \\ c_2 Z_2(t_k)\mathrm{e}^{\varepsilon_k^{(2)}} \end{pmatrix}$$

Objectives: estimate the unknown parameters and sample from $p_\theta(Z_{t_0:t_n}|X_{t_0:t_n})$

Lotka-Volterra

Objectives:

Estimate the unknown parameters and sample from $p_\theta(Z_{t_0:t_n}|X_{t_0:t_n})$

Problems:

$p_\theta(Z_{t_0:t_n}|X_{t_0:t_n}) \propto p_\theta(Z_{t_0:t_n})p_\theta(X_{t_0:t_n}|Z_{t_0:t_n})$ but $p_\theta(Z_{t_0:t_n})$ is unknown.

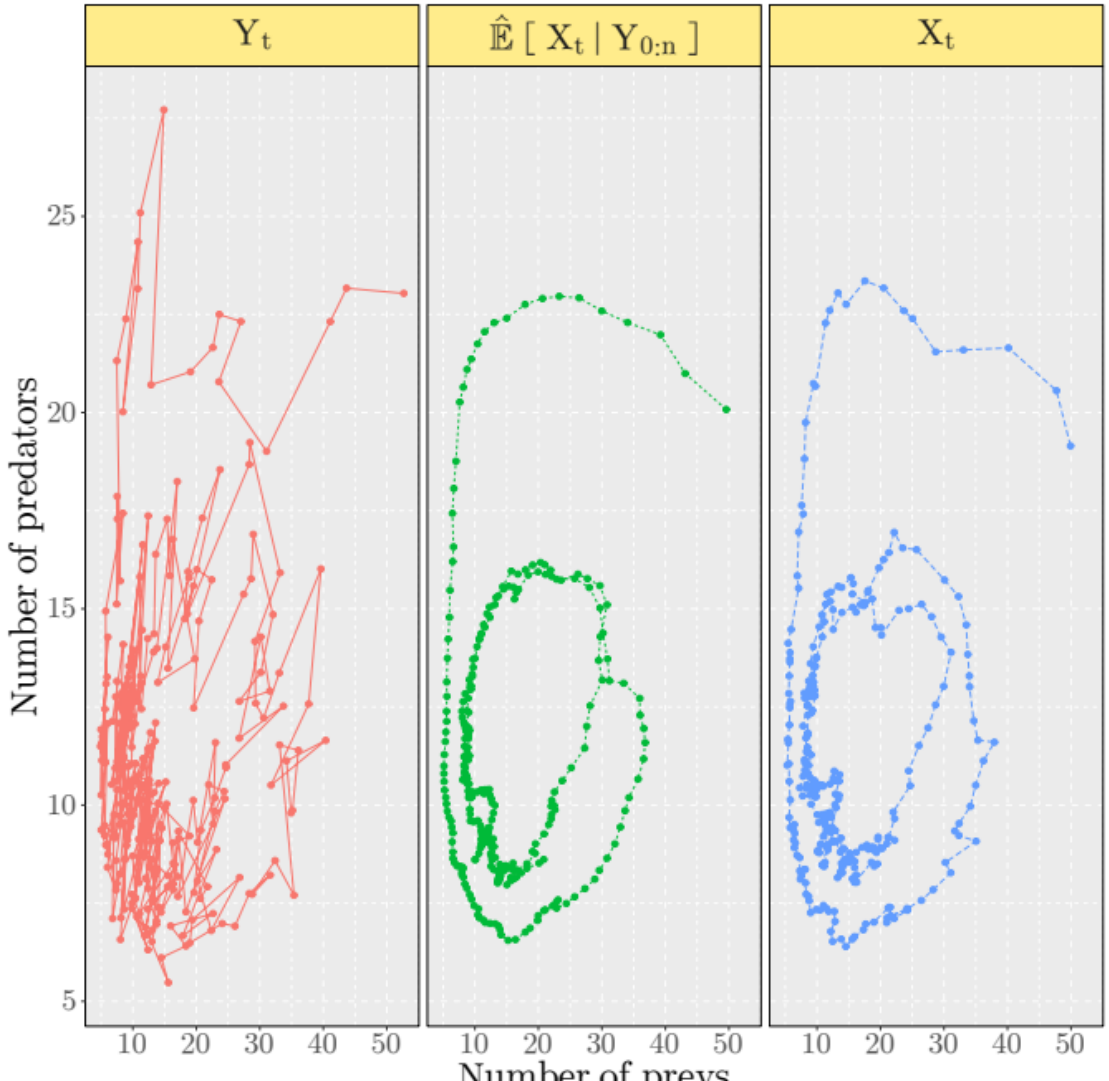
At iteration p, not easy to design a proposal distribution given $Z^{(p)}_{t_0:t_n}$.

Possible to use Sequential Importance Sampling to propose a new state: computationally intensive.

Synthetic data

300 time steps

The first 10000 samples are discarded, the mean over the next 10000 samples is displayed.



Bayesian setting

$\rightharpoonup$ Guess-and-check strategy of Random Walk Metropolis and the approximation of the Langevin equation are doomed to fail in high dimensional spaces.

$\rightharpoonup$ In high dimensional settings, the difficulty is to propose new states far from the initial point, and use the geometry of the target density to ensure a high acceptance probability.

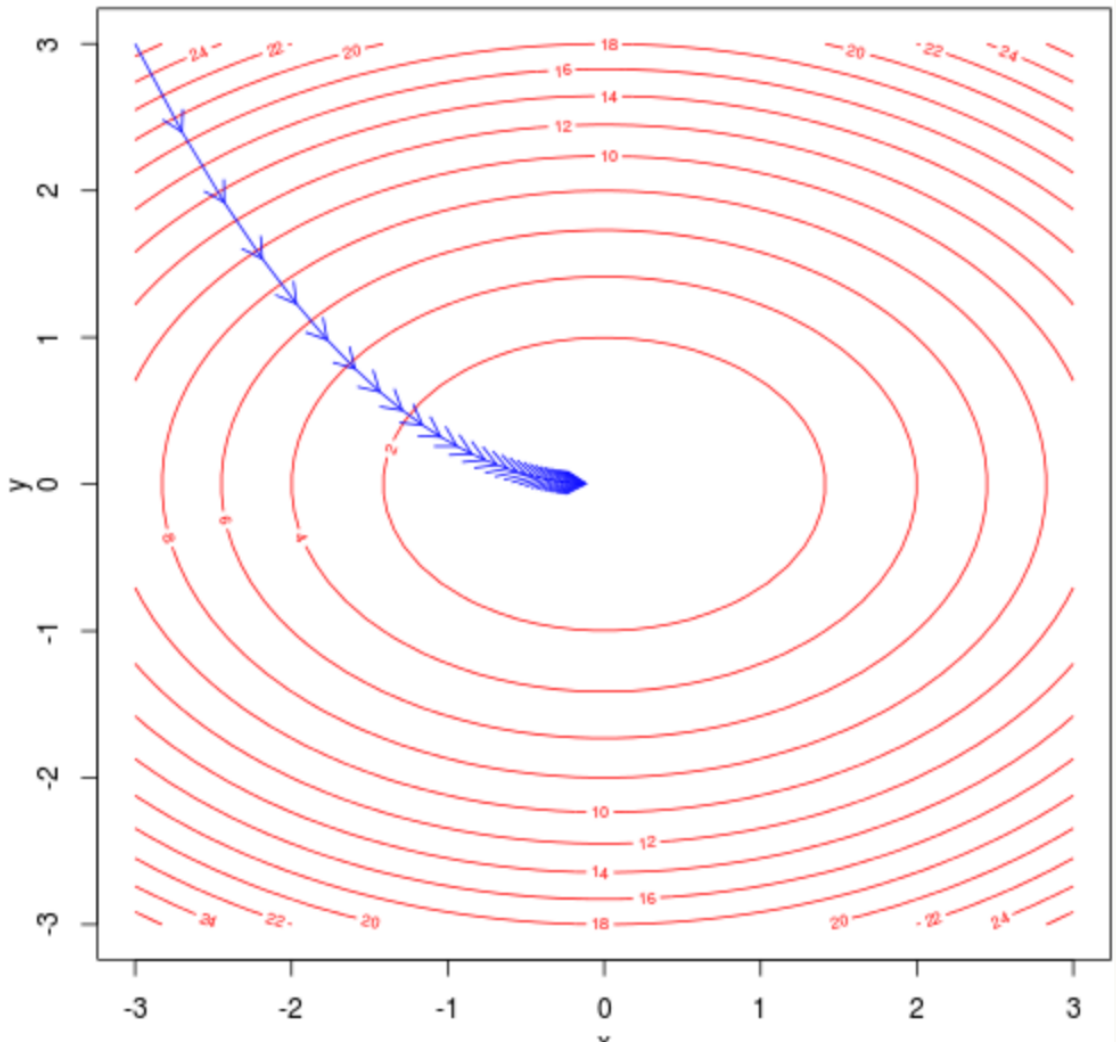
$\rightharpoonup$ A move based on the gradient pulls the state away from the level set towards the mode of the target density.

$\rightharpoonup$ Stability in the exploration may be ensured by adding **a momentum to counteract the "gravitational" (gradient) attraction**.
A well tuned momentum balances the corresponding dynamics and leads to conservative moves.

Heuristic: why gradient moves may not be sufficient ?

For a function $f : \mathbb{R}^d \to \mathbb{R}$, define the level sets:

$$\mathcal{C}_c = \{x \in \mathbb{R}^d, f(x) = c\}\,.$$



$\rightharpoonup$ The gradient is orthogonal to level sets.

Bayesian setting

$\rightharpoonup$ Define a target joint distribution:

$$\mu(q,p) \propto \exp\left(-H(q,p)\right) .$$

$\rightharpoonup$ In the case where $H(q,p) = U(q) + K(p)$,

$$\mu(q,p) \propto \exp\left(-U(q)\right) \exp\left(-K(p)\right) .$$

$\rightharpoonup$ In Bayesian analysis, the position q is understood as the parameter of interest θ and U is its log-posterior (unnormalized) distribution:

$$U(q) = -\log(\pi(q)\ell(q|Y)) ,$$

where π is the prior distribution of q and $\ell(\cdot|Y)$ the conditional distribution given the data.

If $(q_*, p_*) \sim \mu$, then q has the target distribution!

Equations of motion

$\rightharpoonup$ The system is described by a d-dimensional position q and a d-dimensional momentum p.

$\rightharpoonup$ The Hamiltonian function H describes the system dynamics: for all $1 \leqslant j \leqslant d$,

$$\frac{\mathrm{d}q_j}{\mathrm{d}t} = \frac{\partial H}{\partial p_j}\,,$$
$$\frac{\mathrm{d}p_j}{\mathrm{d}t} = -\frac{\partial H}{\partial q_j}\,.$$

This Hamiltonian dynamics generates a vector field oriented with the level set of the joint distribution !

$\rightharpoonup$ This system can be written using matrix products: if $z = (q, p)$ is joint state,

$$\frac{\mathrm{d}z}{\mathrm{d}t} = \begin{pmatrix} 0_d & I_d \\ -I_d & 0_d \end{pmatrix} \nabla H(z)\,.$$

$\rightharpoonup$ In most cases, the Hamiltonian is of the form $H(q,p) = U(q) + K(p)$ where U is the potential energy and $K(p) = p^T M^{-1} p/2$ is the kinetic energy with M a symmetric positive-definite matrix.

Properties of the Hamiltonian dynamics

Reversibility

For all $s \geqslant 0$, the mapping $T_s : (q(t), p(t)) \mapsto (q(t+s), p(t+s))$, is one-to-one: there exists a unique inverse transform.

Stationarity

Using the chain rule,

$$\frac{\mathrm{d}H(q,p)}{\mathrm{d}t} = \sum_{j=1}^{d} \left(\frac{\mathrm{d}q_j}{\mathrm{d}t} \frac{\partial H}{\partial q_j} + \frac{\mathrm{d}p_j}{\mathrm{d}t} \frac{\partial H}{\partial p_j} \right) = 0 \,.$$

Volume preservation (Liouville's theorem)

The vector field defined by Hamiltonian dynamics is divergence free so that the dynamics is volume preserving.

Hamiltonian within MCMC implementation

$\rightarrow$ Define a target joint distribution:

$$\mu(q,p) \propto \exp\left(-H(q,p)\right) .$$

where $H(q,p) = U(q) + K(p)$ and $U(q) = -\log \pi(q|Y)$.

Solving the Hamiltonian dynamics defines a new proposal moves for a MCMC approach.

Which time horizon to solve the system (**efficient exploration of a level set**)?

How to perform the integration numerically ?

How to choose the kinetic energy K ?

Euler discretization

$\rightharpoonup$ Starting at state (q_t, p_t), provides an approximate value for $(q_{t+\delta}, p_{t+\delta})$.
In the case where $H(q,p) = U(q) + K(p)$, Euler discretization is:

$$p_{t+\delta} = p_t - \delta \nabla U(q_t)\,,$$
$$q_{t+\delta} = q_t + \delta \nabla K(p_t)\,.$$

And, if it is assumed that $K(p) = p^T M^{-1} p/2$,

$$p_{t+\delta} = p_t - \delta \nabla U(q_t)\,,$$
$$q_{t+\delta} = q_t + \delta M^{-1} p_t\,.$$

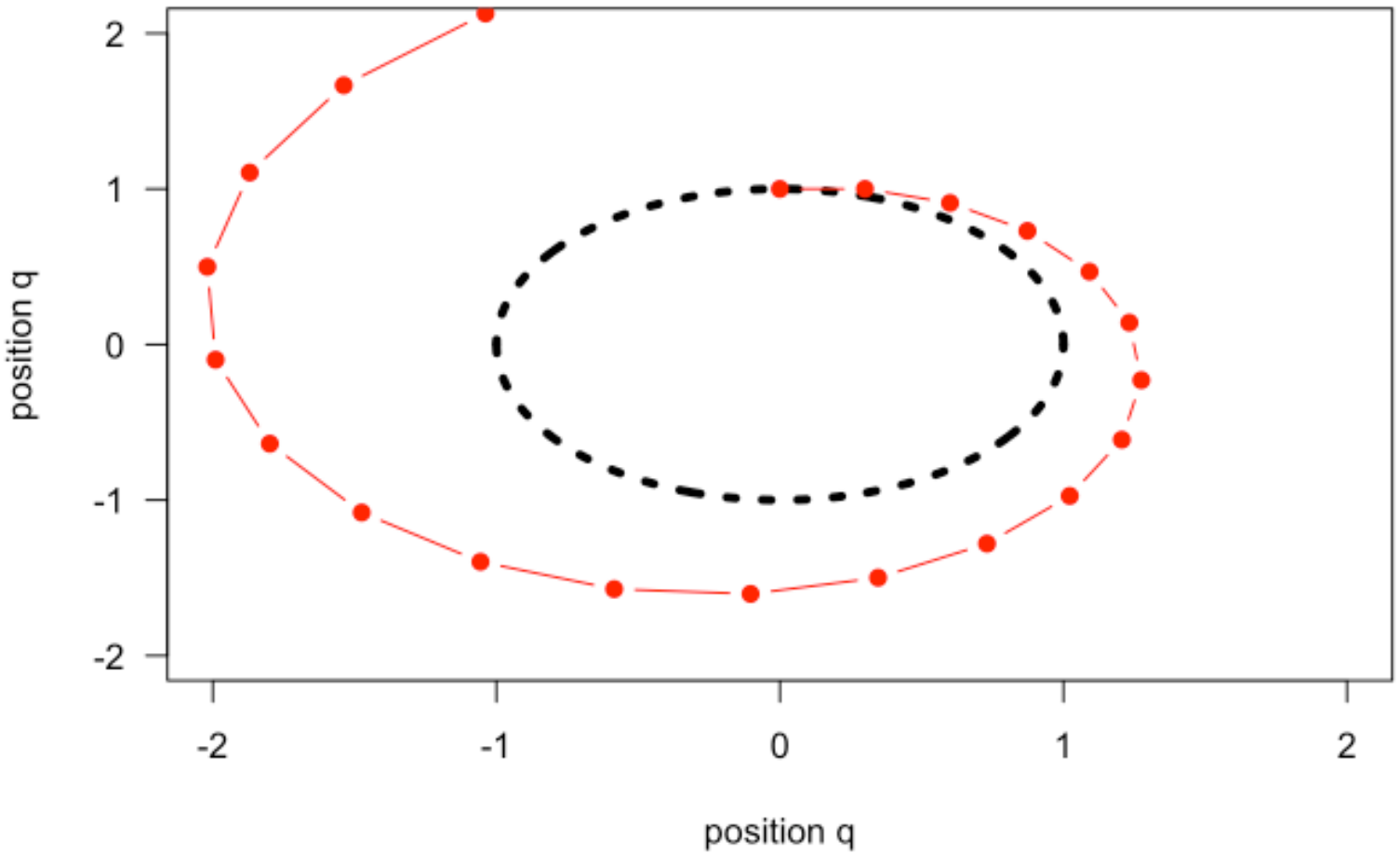
Euler discretization

```
def euler_integrator(q, p, gradientV, T, step):
    """
    Inputs
    ----------
    q: initial position
    p: initial momentum
    gradientV: gradient of the velocity
    T: time horizon
    step: step size to discretize the ODE


    q, p       = np.copy(q), np.copy(p)
    pos, moms = [np.copy(q)], [np.copy(p)]

    vq = gradientV(q)
    nb_steps = int(T / step)

    for it in range(nb_steps):
        p = p - step * vq
        q = q + step * p
        pos.append(np.copy(q))
        vq = gradientV(q)
        moms.append(np.copy(p))

    return q, -p, np.array(pos), np.array(moms)
```

Euler discretization

$\rightharpoonup H(q,p) = q^2/2 + p^2/2.$

$\rightharpoonup$ Initial state $(q,p) = (0,1)$, $L = 20$ steps.



Leapfrog integrator

$\rightharpoonup$ Starting at state (q_t, p_t), provides an approximate value for $(q_{t+\delta}, p_{t+\delta})$.
In the case where $H(q,p) = U(q) + K(p)$, the leapfrog integrator is:

$$
\begin{aligned}
p_{t+\delta/2} &= p_t - \delta \nabla U(q_t)/2\,, \\
q_{t+\delta} &= q_t + \delta \nabla K(p_{t+\delta/2})\,, \\
p_{t+\delta} &= p_{t+\delta/2} - \delta \nabla U(q_{t+\delta})/2\,.
\end{aligned}
$$

And, if it is assumed that $K(p) = p^T M^{-1} p/2$,

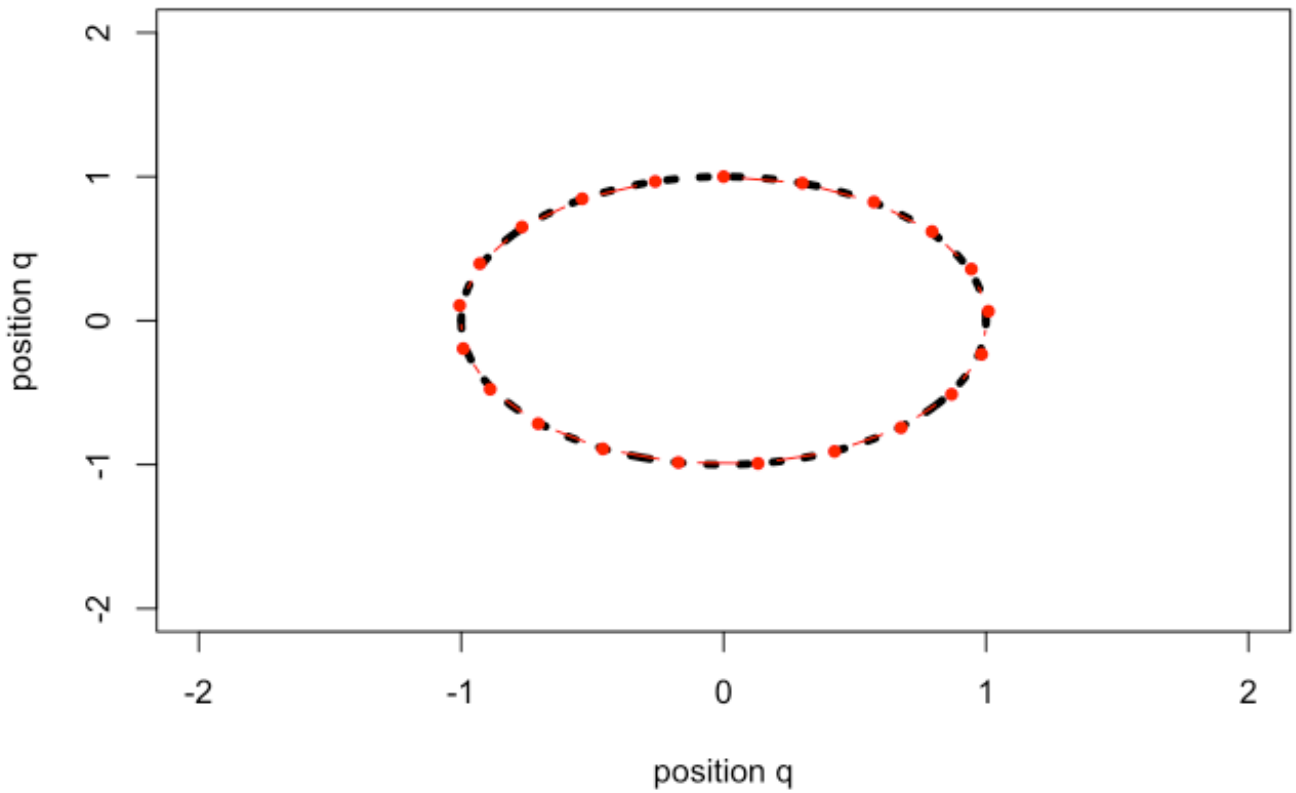
$$
\begin{aligned}
p_{t+\delta/2} &= p_t - \delta \nabla U(q_t)/2\,, \\
q_{t+\delta} &= q_t + \delta M^{-1} p_{t+\delta/2}\,, \\
p_{t+\delta} &= p_{t+\delta/2} - \delta \nabla U(q_{t+\delta})/2\,.
\end{aligned}
$$

Leapfrog discretization

```
def leapfrog_integrator(q, p, gradientV, T, step):
    """
    Inputs
    ----------
    q: initial position
    p: initial momentum
    gradientV: gradient of the velocity
    T: time horizon
    step: step size to discretize the ODE


    q, p       = np.copy(q), np.copy(p)
    pos, moms = [np.copy(q)], [np.copy(p)]

    vq = gradientV(q)
    nb_steps = int(T / step)

    for it in range(nb_steps):
        p = p - 0.5*step * vq
        q = q + step * p
        pos.append(np.copy(q))
        vq = gradientV(q)
        p = p - 0.5*step * vq
        moms.append(np.copy(p))

    return q, -p, np.array(pos), np.array(moms)
```

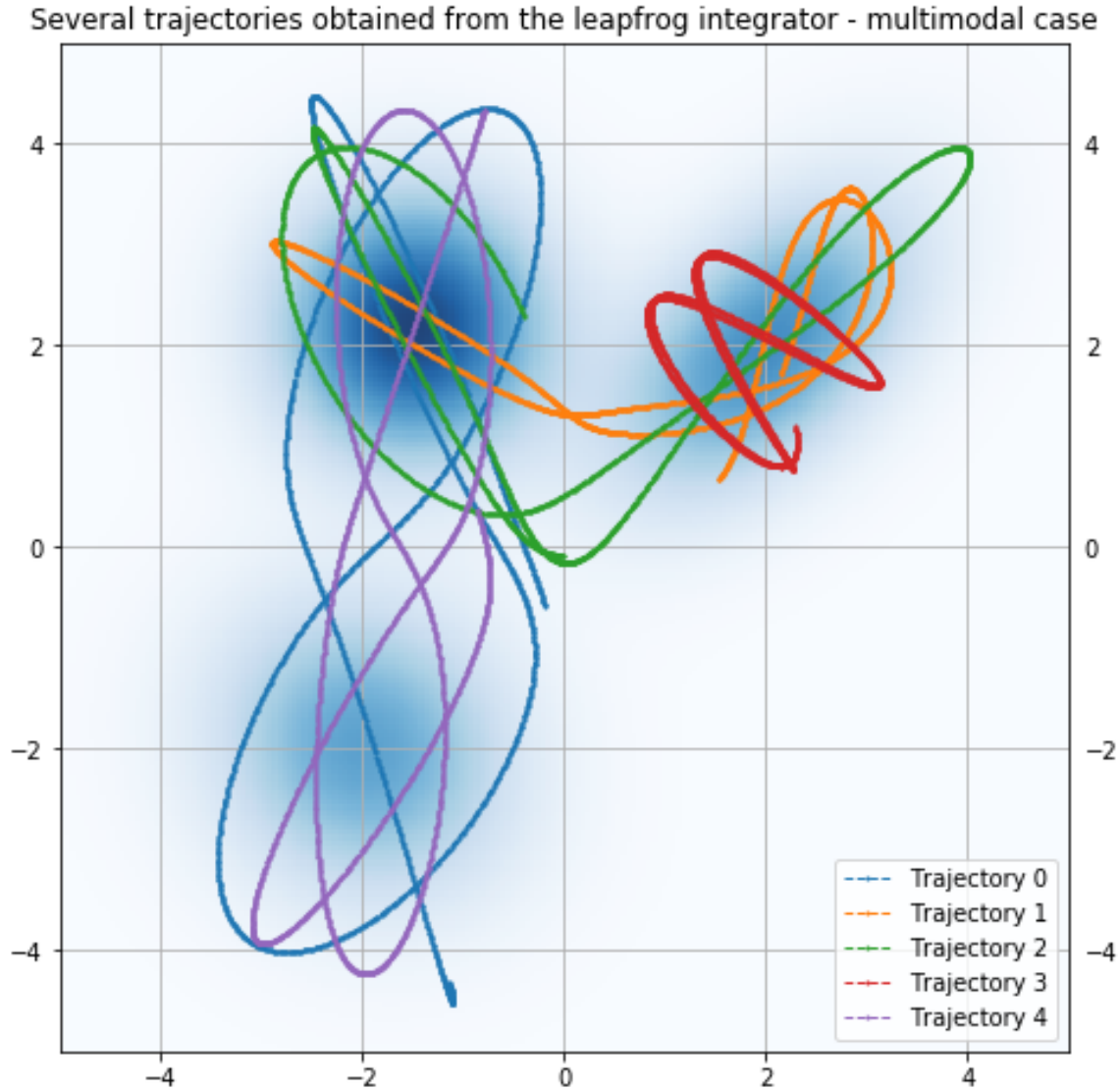
Leapfrog discretization

- $H(q,p) = q^2/2 + p^2/2$.
- Initial state $(q,p) = (0,1)$, $L = 20$ steps.



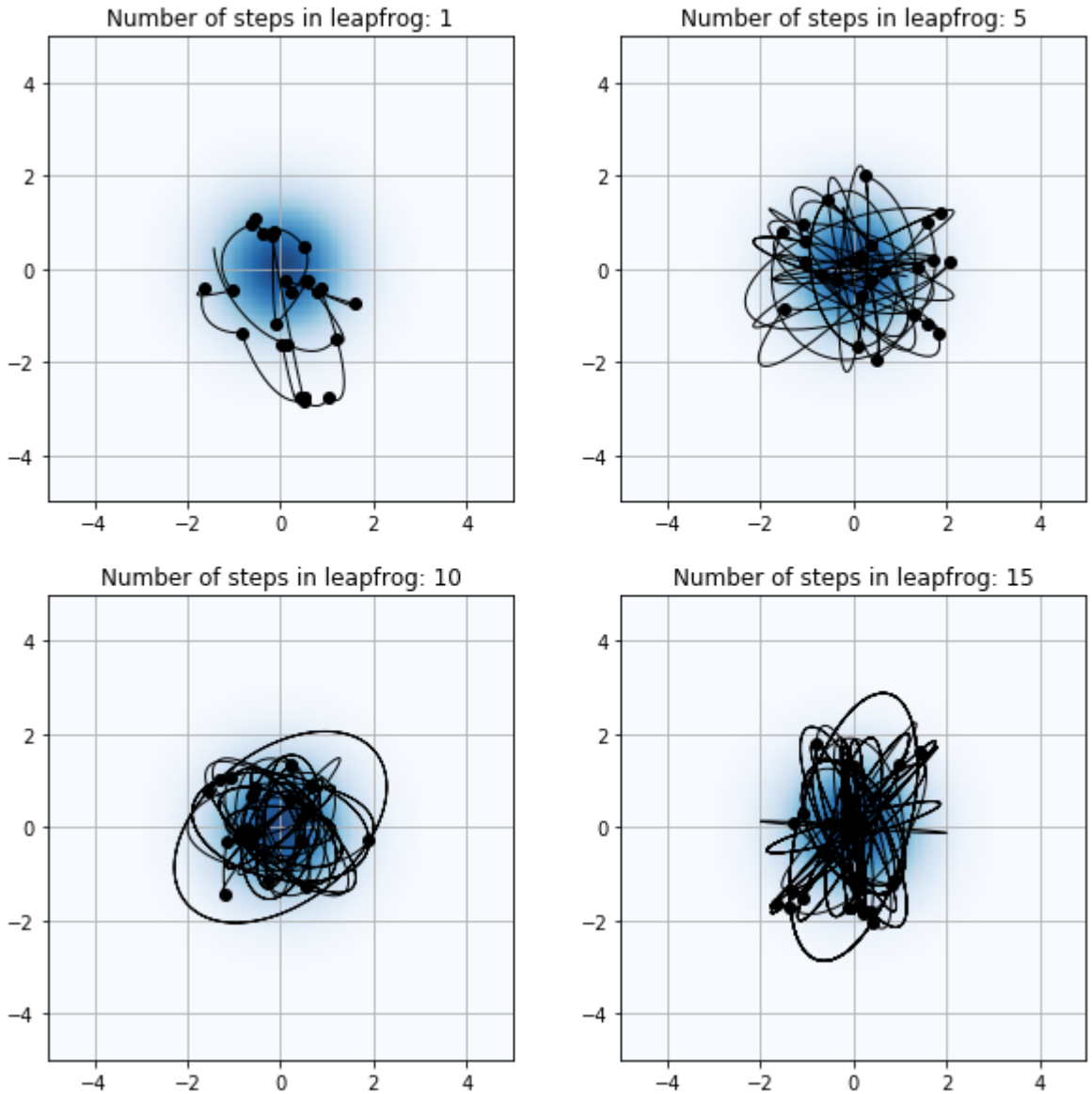
Leapfrog discretization - multimodal setting

$\rightharpoonup U(q)$ is the logdensity of a mixture of 3 Gaussian distributions.



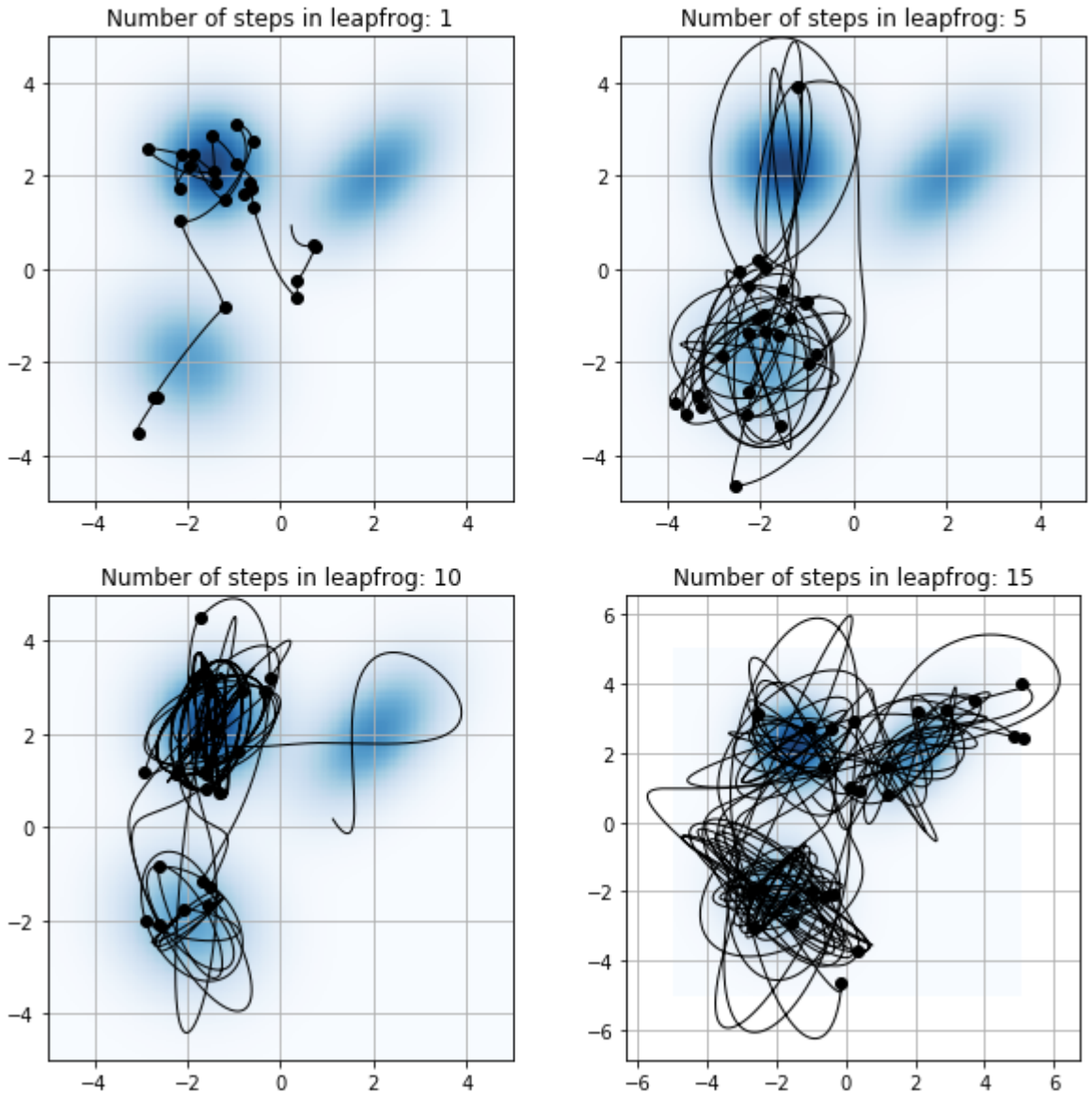
HMC - monomodal setting

$\rightharpoonup U(q)$ is the logdensity of a Gaussian distributions.

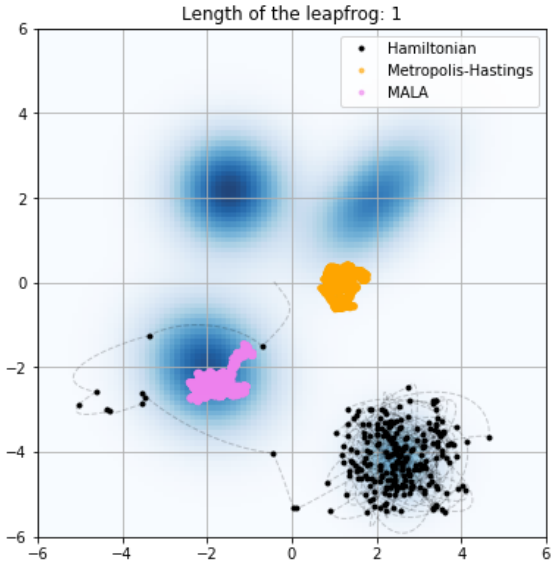


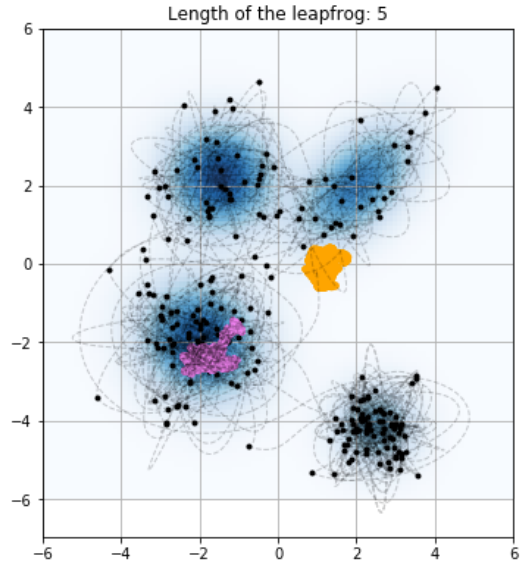
HMC - multimodal setting

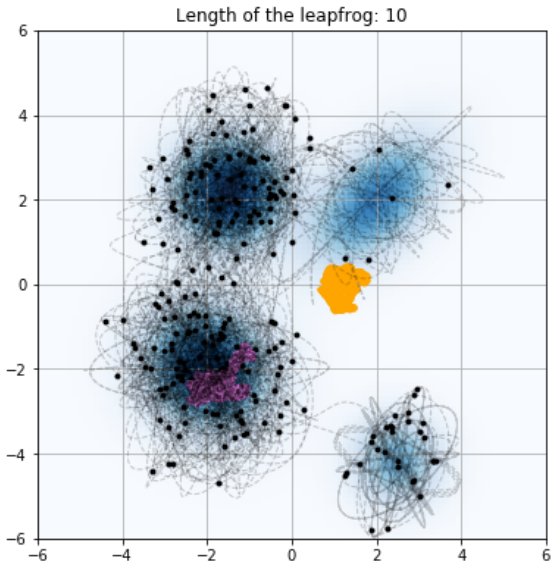
$\rightharpoonup U(q)$ is the logdensity of a mixture of 3 Gaussian distributions.

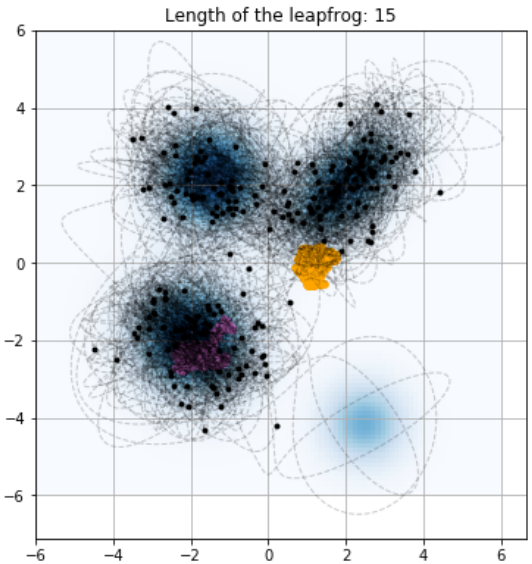


HMC - comparison with other MCMC









Variational Autoencoders

What we did yesterday

What estimate a distribution π_{data} and sample new instances from π_{data}.

Generative model introduce a parametric estimator π_θ and sample new instances from π_θ.

MCMC sample a Markov chain invariant w.r.t. π_θ even when π_θ is known up to a multiplicative constant

- Random Walk Metropolis Hastings: local perturbation of the state at each iteration.
- Metropolis Adjusted Langevin: local perturbation using $\nabla_x \log \pi_\theta(x)$ of the state at each iteration.
- Hamiltoninan Monte Carlo: extended state space to move approximately on level sets.

Generative modeling - latent data

In some situations, observations are partial and do not contain some variables of interest.

Given a generative process for the data, we might be interested in reconstructing the distribution of the missing variables given the data.

We introduce a family of joint probability distributions $(z, x) \mapsto p_\theta(z, x)$, $\theta \in \Theta$ on $(\mathsf{Z} \times \mathsf{X}, \mathcal{Z} \times \mathcal{X})$ where Z **is a latent variable and** X **is the observation**.

In this setting, for all θ, x, z,

$$p_\theta(z, x) = p_\theta(z) p_\theta(x|z)$$

and π_{data} is estimated by the marginal

$$p_\theta(x) = \int p_\theta(z, x) \mathrm{d}z = \int p_\theta(z) p_\theta(x|z) \mathrm{d}z$$

Generative modeling - latent data

Major drawback: $p_\theta(x)$ is not available explicitly, nor the conditional distribution of the latent data given the observation:

$$p_\theta(z|x) = \frac{p_\theta(z)p_\theta(x|z)}{p_\theta(x)} = \frac{p_\theta(z)p_\theta(x|z)}{\int p_\theta(z)p_\theta(x|z)\mathrm{d}z}$$

$\rightsquigarrow$ **Challenging to train the model** !

$\rightsquigarrow$ **Challenging to sample latent data given the observations** !

VAE

Variational Auto-Encoders (VAE) are very popular approaches to introduce approximations of a target conditional distribution in the context of latent data models.

Consider a family of joint probability distributions $(z, x) \mapsto p_\theta(z, x)$, $\theta \in \Theta$, on $(\mathsf{Z} \times \mathsf{X}, \mathcal{Z} \times \mathcal{X})$ where Z is a latent variable and X is the observation.

For all θ, x, z,

$$p_\theta(z, x) = p_\theta(z) p_\theta(x|z)$$

and the conditional distribution $p_\theta(z|x)$ is not availbale explicitly.

How to estimate θ and sample approximately from $p_\theta(z|x)$?

VAE- learn *disentangled* representations

- Common assumption in unsupervised representation learning: **low-dimensional latent variables generate observed data**.
- Knowledge of **true latent variables** useful in many tasks: classification, transfer learning, causal inference etc.
- Problem: **models used usually unidentifiable** (e.g. β-VAE), thus we cannot recover *true* data generating features.
- **General identifiable** framework for principled disentanglement. Deep leargning architectures for **structured VAE**. **Some theoretical guarantees** for VI for state spaces.

ELBO

A variational approach can be introduced. Considering a family $(z, x) \mapsto q_\varphi(z|x)$, $\varphi \in \Phi$ where Φ is a parameter space.

Then, we can write, for all φ, θ, x, z,

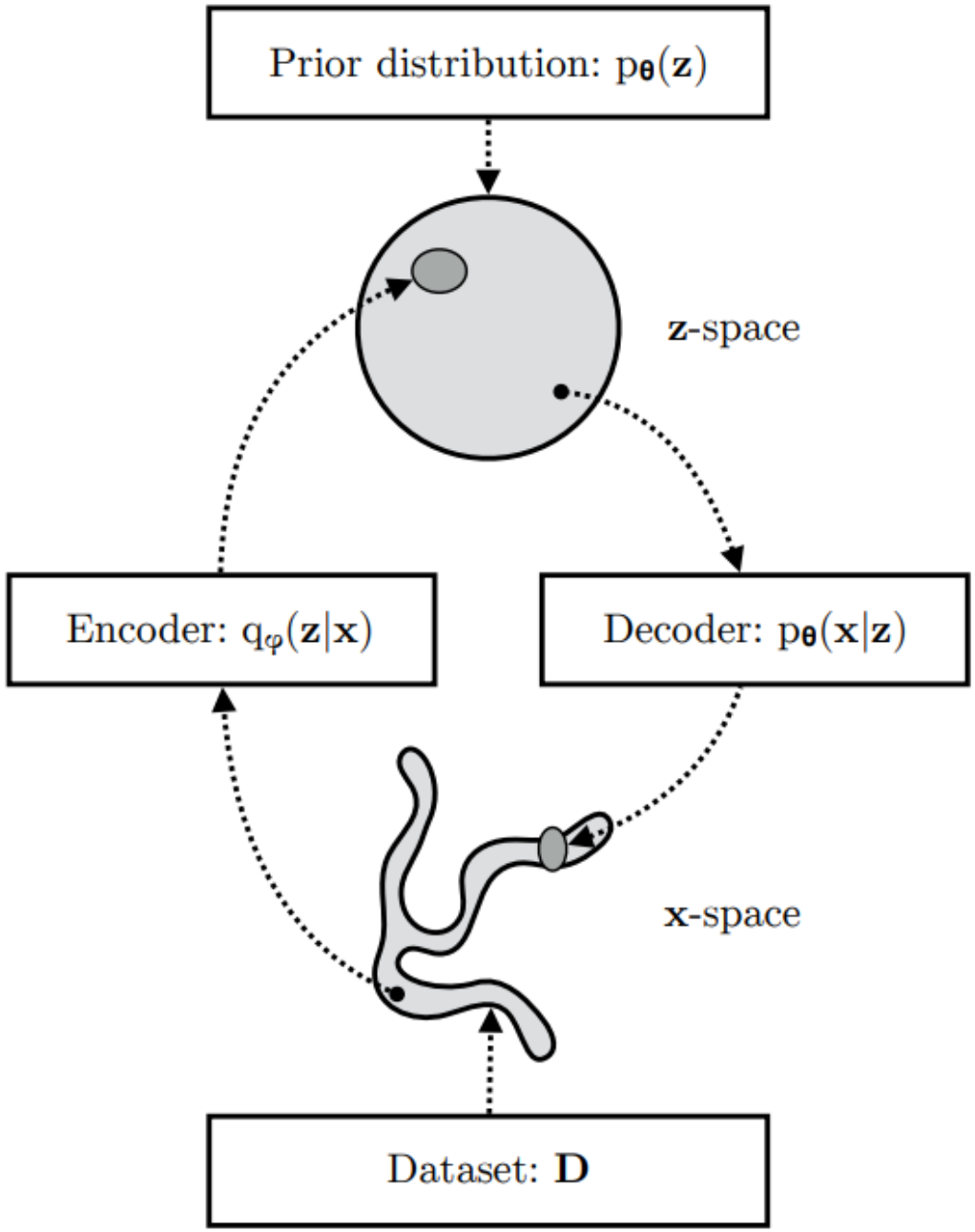
$$\begin{aligned}
\log p_\theta(x) &= \int \log p_\theta(x) q_\varphi(z|x) \mathrm{d}z \\
&= \mathbb{E}_{q_\varphi(\cdot|x)} \left[\log p_\theta(x) \right] \\
&= \mathbb{E}_{q_\varphi(\cdot|x)} \left[\log \frac{p_\theta(Z, x)}{p_\theta(Z|x)} \right] \\
&= \mathbb{E}_{q_\varphi(\cdot|x)} \left[\log \frac{q_\varphi(Z|x)}{p_\theta(Z|x)} \right] + \mathbb{E}_{q_\varphi(\cdot|x)} \left[\log \frac{p_\theta(Z, x)}{q_\varphi(Z|x)} \right].
\end{aligned}$$

The first term of the right-hand-side is the Kullback-Leibler divergence between $q_\varphi(\cdot|x)$ and $p_\theta(\cdot|x)$, so that $\log p_\theta(x) \geq \mathcal{L}(\theta, \varphi, x)$, where

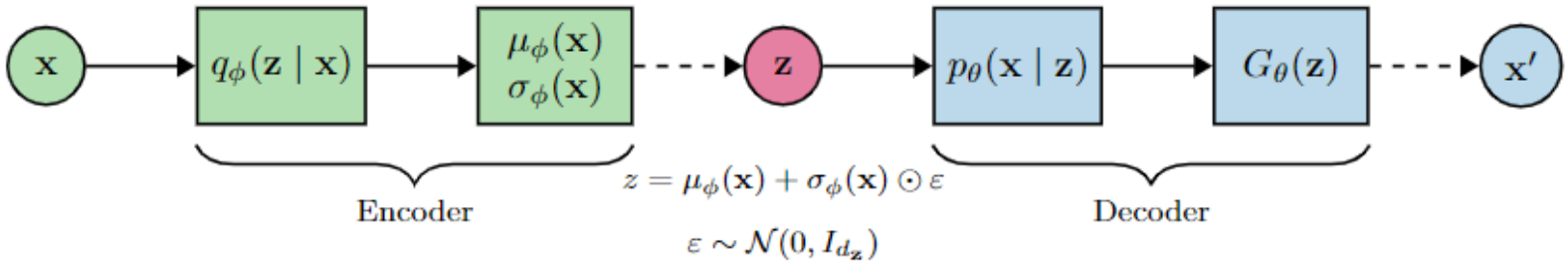
$$\mathcal{L}(\theta, \varphi, x) = \mathbb{E}_{q_\varphi(\cdot|x)} \left[\log \frac{p_\theta(Z, x)}{q_\varphi(Z|x)} \right]$$

is the ELBO in this setting.

A schematic view



An example with Gaussian distributions

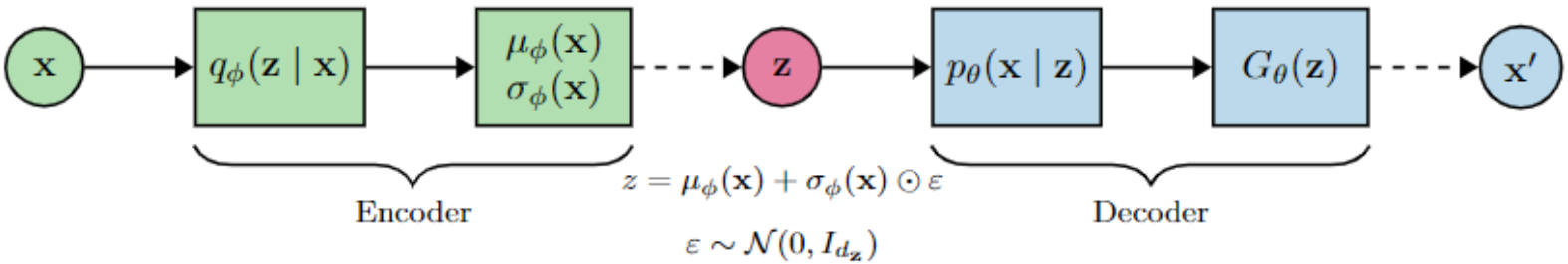


How to define q_φ? $q_\varphi(\cdot|x)$ can be defined as a Gaussian with mean $\mu_\varphi(x)$ and variance $\sigma^2_\varphi(x) I_{d_z}$.

$$
\begin{aligned}
z_0(x) &= x \\
z_k(x) &= \psi_k\left(W_k z_{k-1}(x) + b_{k-1}\right), \qquad 1 \leq k \leq L-1 \\
z_L(x) &= \psi_L\left(W_L z_{L-1}(x) + b_{L-1}\right) = \begin{pmatrix} \mu_\varphi(x) \\ \sigma^2_\varphi(x) \end{pmatrix}.
\end{aligned}
$$

Parameter to estimate $\varphi = (W_k, b_k)_{1 \leq k \leq L}$, functions $(\psi_k)_{1 \leq k \leq L}$ and dimensions of $(z_k(x))_{1 \leq k \leq L}$ to be chosen.

An example with Gaussian distributions



How to define p_θ? $p_\theta(\cdot|z)$ can be defined as a Gaussian with mean $\tilde{\mu}_\theta(z)$ and variance $\tilde{\sigma}^2_\theta(z) I_{d_x}$.

$$\tilde{z}_0(z) = z$$
$$\tilde{z}_k(z) = \tilde{\psi}_k\left(\tilde{W}_k \tilde{z}_{k-1}(z) + \tilde{b}_{k-1}\right), \qquad 1 \leq k \leq L-1$$
$$\tilde{z}_L(z) = \tilde{\psi}_L\left(\tilde{W}_L \tilde{z}_{L-1}(z) + \tilde{b}_{L-1}\right) = \begin{pmatrix} \tilde{\mu}_\theta(z) \\ \tilde{\sigma}^2_\theta(z) \end{pmatrix}.$$

Parameter to estimate $\theta = (\tilde{W}_k, \tilde{b}_k)_{1 \leq k \leq L}$, functions $(\tilde{\psi}_k)_{1 \leq k \leq L}$ and dimensions of $(\tilde{z}_k(z))_{1 \leq k \leq L}$ to be chosen.

Stochastic optimization

Assuming that we have a dataset with i.i.d. data $\{x_i\}_{1\leq i\leq n}$, VAE propose to solve the optimization problem:

$$(\widehat{\varphi}_{*,n}, \widehat{\theta}_{*,n}) \in \operatorname{argmax}_{\varphi\in\Phi,\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\theta, \varphi, x_i).$$

This is approximately solved using Stochastic Gradient ascent.

- Initialize $\widehat{\varphi}_0$, $\widehat{\theta}_0$.
- Draw $(X_{k_1}, \ldots, X_{k_M})$ randomly in the dataset.
-

$$\begin{pmatrix} \widehat{\varphi}_{k+1} \\ \widehat{\theta}_{k+1} \end{pmatrix} = \begin{pmatrix} \widehat{\varphi}_{k} \\ \widehat{\theta}_{k} \end{pmatrix} + \gamma_{k+1} \nabla_{\widehat{\varphi}_k, \widehat{\theta}_k} \left(\frac{1}{M}\sum_{i=1}^{M} \log \frac{p_\theta(Z_i, X_{k_i})}{q_\varphi(Z_i|X_{k_i})} \right)$$

  with $Z_i \sim q_{\widehat{\varphi}_k}(\cdot|X_{k_i})$.

Adaptive step-sizes + ascent direction with (ADAGRAD, Duchi et al., 2011) or (ADAM, Kingma & Ba, 2015)

After training

After the training phase VAE may be used for several objectives.

Loglikelihood estimation: $\log p_\theta(x) = \log \mathbb{E}_{q_\varphi(\cdot|x)}[p_\theta(Z,x)/q_\varphi(Z|x)]$, estimated by

$$\log\left(\frac{1}{M}\sum_{i=1}^{M} p_\theta(Z_i,x)/q_\varphi(Z_i|x)\right)$$

with $(Z_i)_{1\leq i\leq M}$ i.i.d. with distribution $q_\varphi(\cdot|x)$

Reconstruction: for any x, we can encode x by sampling $z \sim q_\varphi(\cdot|x)$ and decode z by sampling $\widehat{x} \sim p_\theta(\cdot|z)$.

Sampling: we can sample new data by $x \sim p_\theta(\cdot|z)$ with $z \sim p_\theta$.

Some theoretical properties

Assumptions on the variational family (Lipschitz in φ and z) and on the training loss (Lipschitz in all parameters)

(Tang & Yang, 2021) obtained an upper bound for the total variation distance between the **target distribution and the distribution generated from a latent space model**.

With probability at least $1 - c\exp(-\kappa(\log n)^{1/\alpha})$,

$$d_{tv}\left(\pi_{\text{data}}, \int \left(\frac{1}{n}\sum_{i=1}^{n} q_{\hat{\varphi}_n}(z|x_i)\right) p_{\hat{\theta}_n}(\cdot|z)\right)^2 \leq c\min_{\theta,\varphi}\mathbb{E}_{\pi_{\text{data}}}\left[m(\theta,\varphi,X)\right] + k\frac{d_*}{n}\log(dn)$$

with

$$\begin{aligned} m(\theta,\varphi,x) &= \log \pi_{\text{data}}(x) + \text{KL}\left(q_\varphi(\cdot|x))\|p_\theta(z)p_\theta(x|z)\right) \\ &= \log \pi_{\text{data}}(x) - \mathcal{L}(\theta,\varphi,x)\,. \end{aligned}$$

Some theoretical properties

With probability at least $1 - c\exp(-\kappa(\log n)^{1/\alpha})$,

$$d_{tv}\left(\pi_{\text{data}}, \int \left(\frac{1}{n}\sum_{i=1}^{n} q_{\hat{\varphi}_n}(z|x_i)\right) p_{\hat{\theta}_n}(\cdot|z)\right)^2 \leq c\min_{\theta,\varphi}\mathbb{E}_{\pi_{\text{data}}}\left[m(\theta,\varphi,X)\right] + k\frac{d_*}{n}\log(d_* n)$$

The estimation error (second term) scales as $O(1/n)$ up to a logarithmic term, which matches the rate of parametric density estimation.

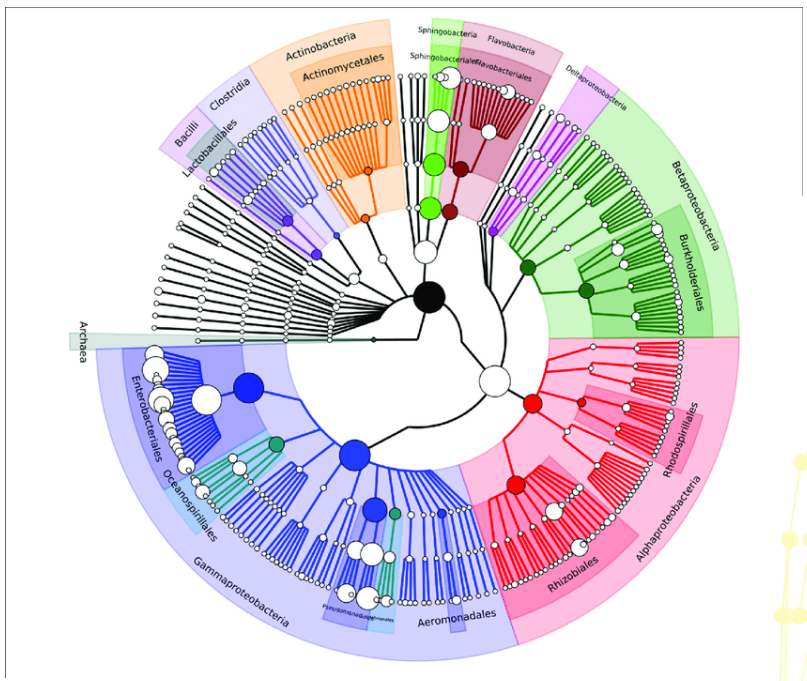
If the model is well-specified, i.e. $\pi_{\text{data}} = \int p_{\theta_*}(z)p_{\theta_*}(\cdot|z)\mathrm{d}z$ and $q_{\varphi(z|x)} = p_{\theta_*}(z|x)$,

$$\min_{\theta,\varphi}\mathbb{E}_{\pi_{\text{data}}}\left[m(\theta,\varphi,X)\right] = 0$$

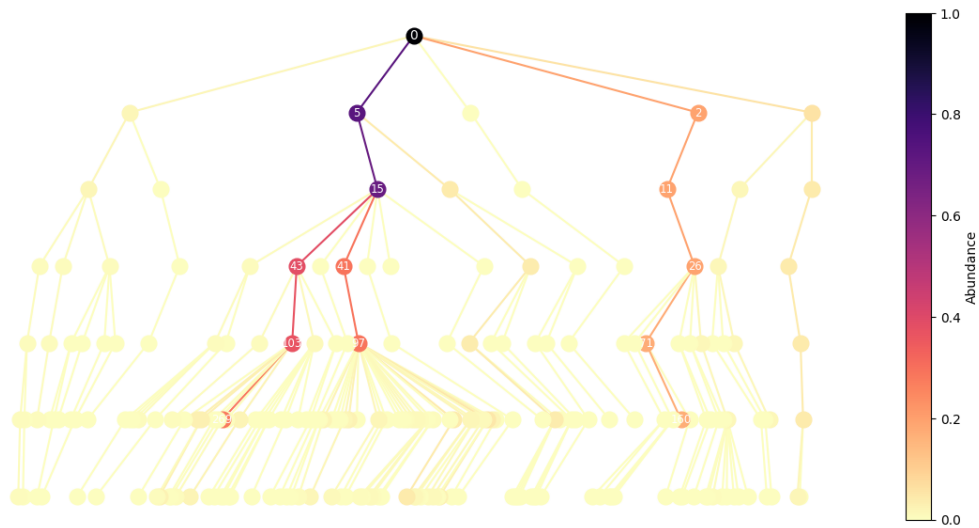
Context: example of the gut microbiome

Microbiota:

- Bacteria, viruses, and parasites living in the gastrointestinal tract.
- Genetic clustering in a tree graph called a taxonomy on which we annotate abundance values.



(a) Taxonomy of the microbiome of Arions species



(b) Taxa-abundance sample

Context: example of the gut microbiome

Motivations:

- Biomarker for several diseases: prediction of Crohn's disease complications through count data.
- Taxonomy not exploited yet despite the genetic correlations it yields.

Medical interpretability:

- The bacteria are forming an interaction network yielding biological functions of interest.

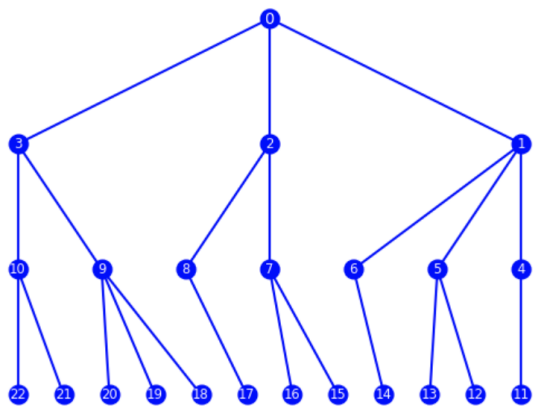
Figure: Functional groups among an interaction network

Context

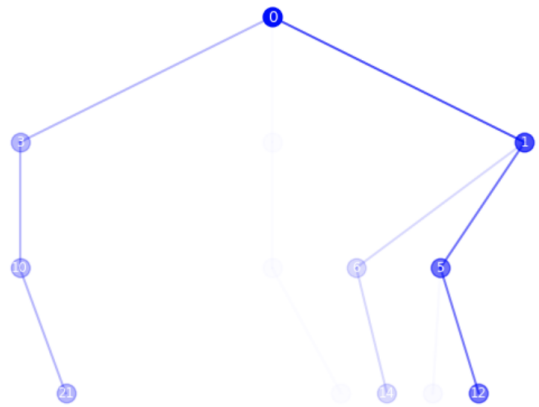
Challenges:

- How to model hierarchical count data?
- Highly variable and sparse data, high dimensionality, low sample size.
- **Hierarchical compositionality**: the children abundance $\mathcal{C}\left(X_k^\ell\right)$ of a node should sum to the value of their parent X_k^ℓ

$$\sum_{j \in \mathcal{C}_k^\ell} X_j^{\ell+1} = X_k^\ell \,.$$



(a) Tree graph



(b) Hierarchical count data

Application to gut microbiota

Using the tree graph while ensuring flexibility?

Top-Down PLN-Tree:

- Latent Markov dynamic (top-down):

$$Z^1 \sim \mathcal{N}\left(\mu_1, \Sigma_1\right) ,$$
$$Z^{\ell+1} \sim \mathcal{N}\left(\mu_{\theta_{\ell+1}}(Z^\ell), \Sigma_{\theta_{\ell+1}}(Z^\ell)\right) .$$

- Constrained observed dynamic:

$$X^1 \sim \mathcal{P}\left(\mathrm{e}^{Z^1}\right) ,$$
$$\mathcal{C}\left(X_k^\ell\right) \sim \mathcal{P}\left(\mathrm{e}^{\mathcal{C}\left(Z_k^\ell\right)} \middle| \sum_{j \in \mathcal{C}_k^\ell} X_j^{\ell+1} = X_k^\ell\right) .$$

PLN-Tree: model inference

Issue: Learning θ through maximum likelihood is intractable.

Approach: Variational inference introduces a learnable proxy to the intractable quantity $p_\theta(\mathbf{Z}|\bar{X})$ denoted by $q_\varphi(\mathbf{Z}|\bar{X})$.

Backward approximation[11]: $p_\theta(\mathbf{Z}|\bar{X})$ is a backward Markov Chain, then

$$q_{\varphi,1:L}(\mathbf{Z}|\bar{X}) = q_{\varphi,L}(\mathbf{Z}^L|\bar{X}^{1:L}) \prod_{\ell=1}^{L-1} q_{\varphi,\ell|\ell+1}(\mathbf{Z}^\ell|\mathbf{Z}^{\ell+1}, \bar{X}^{1:\ell}) \,.$$

Objective: ELBO is not explicit but approximated using Monte Carlo sampling.

[11] Amortized backward variant

PLN-Tree: model inference

Dataset: 650 microbiota from, patients with 7 diseases.

How to evaluate the performances?

Approach: learn the models, sample from them, and assess resemblance with the original data given several criteria.
Perfomance criteria (non-taxonomic dependent):

- Alpha diversity distribution: compare ecosystems by estimating species variety through richness and evenness (Shannon, Simpson, ...).
- Samples distribution: empirical Wasserstein.

Benchmarked models:

- PLN-Tree: backward and mean-field
- PLN per layer: yields non-valid hierarchical count data, just as a reference.
- PLN (fill): sample last layer, fill the rest using the hierarchical compositionality constraint.

Synthetic data

For individual i, if $p_{s,i}$ is the proportion of species i,

$$\text{Shannon}_{\text{i},\ell} = -\sum_{s=1}^{S_\ell} p_{s,i} \log p_{s,i} \quad \text{InvSimpson}_{\text{i},\ell} = \frac{1}{\sum_{s=1}^{S_\ell} p_{s,i}^2}.$$

After parameter inference, we sample $M = 25$ times $n = 2000$ microbiota to be compared with the dataset.

Alpha diversity	**PLN-Tree**	**PLN-Tree (MF)**	**PLN (fill)**	**PLN**
		Wasserstein Distance ($\times 10^2$)		
Shannon $\ell = 1$	**1.57** (0.50)	11.23 (0.73)	14.53 (2.37)	2.61 (0.85)
Shannon $\ell = 2$	**3.67** (1.33)	5.14 (1.20)	31.55 (3.42)	25.85 (2.57)
Shannon $\ell = 3$	**5.82** (1.51)	7.86 (1.47)	34.36 (3.68)	34.36 (3.68)
Inverse Simpson $\ell = 1$	**0.62** (0.21)	2.69 (0.27)	4.91 (0.93)	0.95 (0.34)
Inverse Simpson $\ell = 2$	**0.71** (0.24)	1.40 (0.31)	7.25 (0.98)	5.92 (0.69)
Inverse Simpson $\ell = 3$	**0.85** (0.24)	1.55 (0.34)	7.08 (0.96)	7.08 (0.96)

Public dataset

(Pasolli et al., 2016)

Label	Nb of training samples	Nb of test samples	Total
IBD (Crohn)	20	5	25
Colorectal Cancer	38	10	48
Leaness	71	18	89
Liver Cirrhosis	94	24	118
IBD (UC)	118	30	148
Obesity	131	33	164
Type 2 Diabetes	178	45	223
Total	650	165	815

Public dataset

	PLN-Tree	PLN-Tree (MF)	PLN (fill)	PLN
		Wasserstein distance ($\times 10^2$)		
$\ell = 1$	5.89 (0.29)	**4.67** (0.25)	15.18 (0.70)	8.52 (0.72)
$\ell = 2$	8.83 (0.28)	**7.55** (0.14)	20.47 (0.70)	15.80 (0.58)
$\ell = 3$	9.27 (0.27)	**7.76** (0.12)	20.72 (0.71)	18.68 (0.66)
$\ell = 4$	17.00 (0.22)	**15.59** (0.13)	29.41 (0.78)	29.41 (0.78)

Public dataset

Alpha diversity	PLN-Tree	PLN-Tree (MF)	PLN (fill)	PLN
		Wasserstein distance ($\times 10^2$)		
Shannon $\ell = 1$	**1.73** (0.44)	3.00 (0.44)	16.48 (1.37)	4.22 (1.44)
Shannon $\ell = 2$	**2.22** (0.73)	5.70 (0.97)	22.94 (2.29)	8.28 (1.89)
Shannon $\ell = 3$	**2.29** (0.63)	6.58 (1.02)	23.76 (2.29)	9.13 (2.24)
Shannon $\ell = 4$	**2.08** (0.62)	20.39 (1.08)	54.55 (2.98)	54.55 (2.98)
Inverse Simpson $\ell = 1$	0.84 (0.14)	**0.71** (0.12)	7.18 (0.56)	1.75 (0.59)
Inverse Simpson $\ell = 2$	0.92 (0.24)	**0.73** (0.19)	7.54 (0.79)	2.64 (0.44)
Inverse Simpson $\ell = 3$	0.91 (0.23)	**0.72** (0.19)	7.53 (0.79)	2.93 (0.50)
Inverse Simpson $\ell = 4$	**0.53** (0.13)	2.41 (0.21)	12.83 (0.89)	12.83 (0.89)

Public dataset - classification

Use a **VAE as preprocessing techniques to solve other tasks**

In the PLN-Tree framework, the first layer's latent variable models the total count, while the next variables account for how the counts progressively distribute over the layers in the observed space.

Consider a feature-engineered latent feature called Latent Tree Counts (LTC)

$$\begin{aligned} & \mathbf{V}^1 = \exp(\mathbf{Z}^1)\,, \\ \forall \ell < L, k \leq K_\ell\,, \quad & \check{\mathbf{V}}_k^\ell = \sigma\left(\check{\mathbf{Z}}_k^\ell\right) \times \mathbf{V}_k^\ell\,. \end{aligned}$$

Use the latent features as input of any classification procedure

Public dataset - classification

	Raw data	log-LTC (PLN-Tree)	log-LTC (MF)	PLN
Logistic Regression				
Balanced Accuracy	0.632 (0.042)	0.734 (0.039)	**0.742** (0.034)	0.716 (0.037)
Precision	0.701 (0.032)	0.778 (0.030)	**0.785** (0.026)	0.765 (0.028)
Recall	0.645 (0.039)	0.747 (0.033)	**0.750** (0.028)	0.727 (0.032)
F1 score	0.661 (0.036)	0.756 (0.031)	**0.759** (0.026)	0.738 (0.030)
ROC AUC	0.677 (0.045)	0.790 (0.036)	0.785 (0.037)	**0.791** (0.037)
ROC Precision-Recall	0.438 (0.061)	0.559 (0.060)	0.559 (0.057)	**0.601** (0.069)
Linear SVM				
Balanced Accuracy	0.586 (0.042)	0.733 (0.040)	**0.738** (0.034)	0.714 (0.033)
Precision	0.673 (0.035)	0.777 (0.030)	**0.781** (0.026)	0.763 (0.025)
Recall	0.584 (0.061)	**0.746** (0.035)	0.743 (0.031)	0.72 (0.031)
F1 score	0.598 (0.062)	**0.755** (0.033)	0.753 (0.029)	0.732 (0.029)
ROC AUC	0.545 (0.127)	**0.793** (0.033)	0.787 (0.035)	0.791 (0.036)
ROC Precision-Recall	0.336 (0.085)	0.562 (0.061)	0.551 (0.059)	**0.597** (0.067)
Neural Network				
Balanced Accuracy	0.704 (0.036)	**0.737** (0.030)	0.704 (0.041)	0.727 (0.034)
Precision	0.773 (0.026)	**0.799** (0.024)	0.778 (0.029)	0.795 (0.026)
Recall	0.777 (0.028)	**0.804** (0.023)	0.778 (0.035)	0.802 (0.025)
F1 score	0.772 (0.027)	**0.800** (0.023)	0.772 (0.032)	0.795 (0.026)
ROC AUC	0.782 (0.036)	0.827 (0.026)	0.792 (0.041)	**0.854** (0.024)
ROC Precision-Recall	0.62 (0.062)	0.658 (0.056)	0.634 (0.065)	**0.692** (0.051)

Reconstruction guarantees

State space models

$$\underbrace{\phi^{\theta}_{0:t}}_{Z_{0:t} \text{ given } X_{0:t}} h = \mathbb{E}_{\theta}\left[h(Z_{0:t})|X_{0:t}\right]$$

- $Z_{0:t}$ is a Markov chain with transition density m_θ.
- Conditionally on $Z_{0:t}$, the observations are independent with emission densities $g_\theta(Z_t, \cdot)$.

Additive state functionals

$$h_{0:t} : z_{0:t} \mapsto \sum_{s=1}^{t} \tilde{h}_s(z_{s-1}, z_s)$$

$\rightsquigarrow \phi^{\theta}_{0:t} h_{0:t}$ *crucial in both inference and parameter learning.*

Variational family

In practice the model is often estimated by **maximizing the ELBO**:

$$\mathcal{L}(\theta, \varphi) = \mathbb{E}_{q_{\varphi,0:t}} \left[\log \frac{p_\theta(\mathbf{z}_{1:t}, \mathbf{x}_{1:t})}{q_{\varphi,0:t}(\mathbf{z}_{1:t}|\mathbf{x}_{1:t})} \right]$$

where $q_{\varphi,0:t}(\mathbf{z}_{1:t}|\mathbf{x}_{1:t})$ is the **variational distribution**.

Traditional assumption on the variational family

$$q_{\varphi,0:t}(\mathbf{z}_{1:t}|\mathbf{x}_{1:t}) = \prod_{s=1}^{t} q_{\varphi,s}(\mathbf{z}_s|\mathbf{x}_{1:t}) \,.$$

$\rightharpoonup$ No theoretical results and does not fit classical posterior distributions (for instance in HMMs).

New framework: backward decomposition

$$q_{\varphi,0:t}(\mathbf{z}_{1:t}|\mathbf{x}_{1:t}) = q_{\varphi,t}(\mathbf{z}_t|\mathbf{x}_{1:t}) \prod_{s=2}^{t} q_{\varphi,s-1|s}(\mathbf{z}_{s-1}|\mathbf{z}_s, \mathbf{x}_{1:t}) \,.$$

$\rightharpoonup$ Some theoretical guarantees and well designed for online learning.

Goal - bias control

Theoretically validate backward variational smoothing as a valid approximation.

- Variational inference **is not consistent**.
- Bias depends on implementation / optimization.

$\rightsquigarrow$ Ensure that the bias is controlled w.r.t time.

Quantities of interest: $\phi_{0:t}^{\theta} h_{0:t} = \mathbb{E}_{\theta}\left[h_{0:t}(Z_{0:t}) | X_{0:t}\right]$

$h_{0:t}$ additive state functional.

$$|q_{\varphi,0:t} h_{0:t} - \phi_{0:t}^{\theta} h_{0:t}| \leq ?$$

$\rightsquigarrow$ Marginal smoothing as a byproduct.

An insightful result

Assumptions

- $\left\| q_{\varphi,t} - \phi_t^\theta \right\|_{\mathrm{tv}} \leq \varepsilon.$
- $\left\| q_{\varphi,s-1|s}(z_s, \cdot) - b_{s-1|s}^\theta(z_s, \cdot) \right\|_{\mathrm{tv}} \leq \varepsilon$ for all $s < t$, $x_s \in \mathsf{X}$.

Additive bound

$$\left| q_{\varphi,0:t} h_{0:t} - \phi_{0:t}^\theta h_{0:t} \right| \leq ct\varepsilon$$

Questions

Quantitative bounds without strong mixing ?

Does minimizing the ELBO ensure that the true and variational kernels are close ?

To obtain excess risk bound

Assumptions

- $\mathrm{KL}(q_{\varphi,t}, \phi_t^\theta) \leq \varepsilon$.
- $\mathrm{KL}(q_{\varphi,s-1|s}(z_s, \cdot), b^\theta_{s-1|s}(z_s, \cdot)) \leq \varepsilon$ for all $s < t$.
- Additional moment and Lipschitz assumptions.

There exist constants c_0, c_1, c_2, D such that with probability at least $1 - c_0 \exp(-c_1 \{\log n\}^{1 \wedge \alpha_*})$, for any $\gamma > 0$,

$$\mathrm{KL}\left(\pi_{\mathrm{data}} \middle\| P^X_{\widehat{\theta}_{n,T}}\right) \leq (1+\gamma)(T+1)\epsilon + c_2(1+\gamma^{-1}) \frac{D d_* T^3}{n} \log(d_* n)(\log n)^{1/\alpha_*} .$$

To obtain excess risk bound

There exist constants c_0, c_1, c_2, D such that with probability at least $1 - c_0 \exp(-c_1 \{d_* \log n\}^{1 \wedge \alpha_*})$, for any $\gamma > 0$,

$$\mathrm{KL}\left(\pi_{\mathrm{data}} \middle\| P^X_{\widehat{\theta}_{n,T}}\right) \leq (1+\gamma)(T+1)\epsilon + c_2(1+\gamma^{-1}) \frac{D d_* T^3}{n} \log(d_* n)(\log n)^{1/\alpha_*} .$$

Questions

Improving the dependency with respect to T ?

Specific results (constants) for specific deep architectures ?

Estimating π_{data} with diffusion models

What we already know

In some situations, observations are partial and do not contain some variables of interest.

Given a generative process for the data, we might be interested in reconstructing the distribution of the missing variables given the data.

We introduce a family of joint probability distributions $(z, x) \mapsto p_\theta(z, x)$, $\theta \in \Theta$ on $(\mathsf{Z} \times \mathsf{X}, \mathcal{Z} \times \mathcal{X})$ where Z **is a latent variable and** X **is the observation**.

In this setting, for all θ, x, z,

$$p_\theta(z, x) = p_\theta(z) p_\theta(x|z)$$

and π_{data} is estimated by the marginal

$$p_\theta(x) = \int p_\theta(z, x) \mathrm{d}z = \int p_\theta(z) p_\theta(x|z) \mathrm{d}z$$

What we already know

MCMC sample a Markov chain invariant w.r.t. $p_\theta(z|x)$ even if $p_\theta(z|x) \propto p_\theta(z)p_\theta(x|z)$ is known up to a multiplicative constant.

- Random Walk Metropolis Hastings: local perturbation of the state at each iteration.
- Metropolis Adjusted Langevin: local perturbation using $\nabla_z \log p_\theta(z|x)$ of the state at each iteration.
- Hamiltoninan Monte Carlo: extended state space to move approximately on level sets.

VAE to approximate $p_\theta(z|x)$ with variational family $q_\varphi(z|x)$.

- Easy optimization and large variety of choices for $q_\varphi(z|x)$.
- Latent space: useful to encode data, solve classification tasks.
- Challenge: design problem-tailored $q_\varphi(z|x)$.

Score-Based Generative Models (SGMs)

What: generate synthetic instances of a target distribution π_{data}

How: instead of providing a model for π_{data}, we model the score $\nabla_x \log \pi_{\text{data}}$

Key idea: perturb data with a sequence of intensifying Gaussian noise and jointly estimate the score functions for all noisy data distributions (Song & Ermon, 2019)

Why using the score function ?

Consider the **Langevin SDE**

$$\mathrm{d}X_t = \nabla_x \log \pi_{\mathsf{data}}(X_t)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t$$

Markov semigroup associated with the Langevin diffusion $(X_t)_{t\geq 0}$ **is reversible w.r.t.** π_{data}

Euler–Maruyama discretization scheme associated with the Langevin diffusion

$$X_{k+1} = X_k + \gamma_{k+1}\nabla_x \log \pi_{\mathsf{data}}(X_k) + \sqrt{2\gamma_{k+1}}Z_{k+1}$$

If $(\gamma_k)_{k\geq 0}$ decreases to 0, **the marginal distribution of this chain converges to** π_{data}

Score-Based Generative Models (SGMs)

What: generate synthetic instances of a target distribution π_{data}

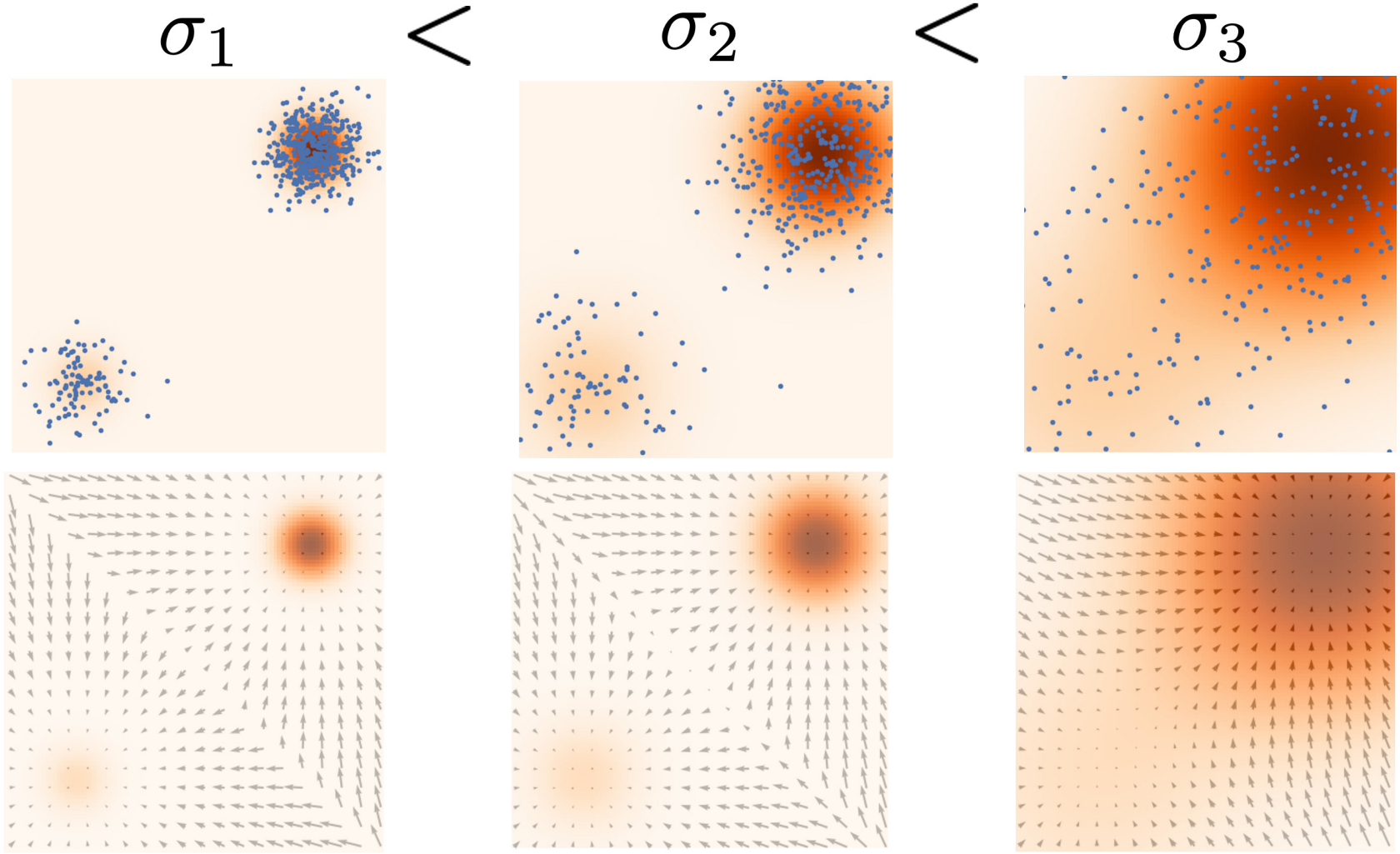
How: instead of providing a model for π_{data}, we model the score $\nabla_x \log \pi_{\text{data}}$

Let $\sigma_1 < \ldots < \sigma_T$ be a sequence of noise levels and write

$$X_t = X_0 + \sigma_t \varepsilon_t \,,$$

with $X_0 \sim \pi_{\text{data}}$ and $\varepsilon_t \sim \mathcal{N}(0, I_d)$.

The conditional distribution of X_t given X_0 $q_{t|0}(X_t|X_0)$ is known and Gaussian but the marginal density q_t of X_t is unknown.



Top: noised distributions
Bottom: noised scored

Score-Based Generative Models (SGMs)

$$X_t = X_0 + \sigma_t \varepsilon_t \,,$$

with $X_0 \sim \pi_{\text{data}}$ and $\varepsilon_t \sim \mathcal{N}(0, I_d)$.

Let $s_\theta(x,t)$ be a parametric approximation of $\nabla \log q_t(x)$.

The parameter θ can be estimated by minimizing a **score matching loss:**

$$\begin{aligned}
\mathcal{L}(\theta) &= \mathbb{E}\left[\lambda(t)\sigma_t^2 \left\|\nabla_x \log q_t(X_t) - s_\theta(X_t,t)\right\|^2\right] \\
&= \mathbb{E}\left[\lambda(t)\sigma_t^2 \left\|\nabla_x \log q_{t|0}(X_t|X_0) - s_\theta(X_t,t)\right\|^2\right] + C \\
&= \mathbb{E}\left[\lambda(t)\left\|-\frac{X_t - X_0}{\sigma_t} - \sigma_t s_\theta(X_t,t)\right\|^2\right] + C \\
&= \mathbb{E}\left[\lambda(t)\sigma_t^2 \left\|\varepsilon + \sigma_t s_\theta(X_t,t)\right\|^2\right] + C
\end{aligned}$$

Score-Based Generative Models (SGMs)

The parameter θ can be estimated by minimizing a **score matching loss:**

$$\mathcal{L}(\theta) = \mathbb{E}\left[\lambda(t)\sigma_t^2 \left\|\varepsilon + \sigma_t s_\theta(X_t, t)\right\|^2\right] + C$$

Then , **sampling can be performed with annealed Langevin dynamics**

- Start with $x_{T+1}^{(N)} \sim \mathcal{N}(0, I_d)$.
- For $t = T$ to $t = 1$
  Set $x_t^{(0)} = x_{t+1}^{(N)}$
  For $i = 1$ to $i = N - 1$
  $x_t^{(i+1)} = x_t^{(i)} + s_t s_\theta(x_t^i, t)/2 + s_t^{1/2}\varepsilon_t^i$

Score-Based Generative Models (SGMs)

What: generate synthetic instances of a target distribution π_{data}

Why: challenges in modeling the complexity of real data, preventing conventional parametric modeling or traditional maximum likelihood methods.

> *Creating noise from data is easy; creating data from noise is generative modeling.* (Song et al., *Score-Based Generative Modeling through Stochastic Differential Equations*)

Who: SGMs address this by

1. (forward phase) introducing progressively noise into the samples,
2. (backward phase) reversing the noising dynamics, with the help of a score function usually learned using deep neural networks.

Diffusion models: the forward process - DDPM (Ho et al., 2020)

Consider the *forward noising* process

$$X_k = \sqrt{1-\beta_k} X_{k-1} + \sqrt{\beta_k} Z_k\,, \quad \beta_k \in [0,1], \quad X_0 \sim \pi_{\text{data}}\,,$$

where $Z_k \sim \mathcal{N}(\mathbf{0}_{d_x}, \mathrm{I}_{d_x})$.

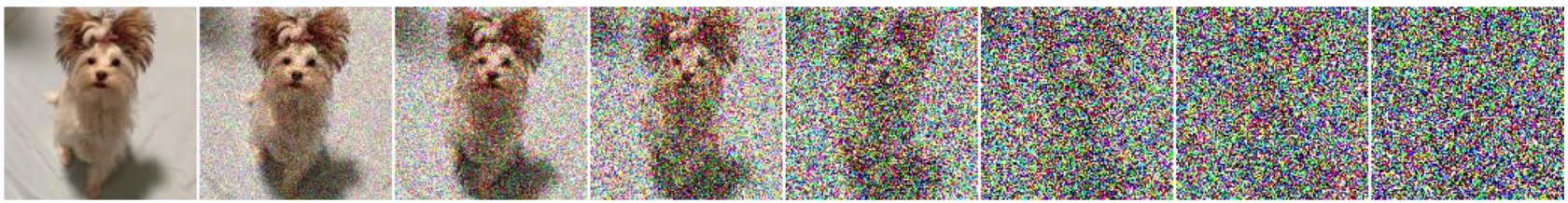


Figure: One sample $X_{0:n}$.

$X_k \sim \pi_k$ where [12] $\pi_k(\mathrm{d}x_k) := \int \pi_{\text{data}}(\mathrm{d}x_0)\mathcal{N}(\mathrm{d}x_k; \sqrt{\bar{\alpha}_k}x_0, (1-\bar{\alpha}_k)\mathrm{I}_{d_x})$.

$(X_k)_{k\geq 0}$ is a discrete-time OU process.

[12] $\bar{\alpha}_k := \prod_{j=1}^{k}(1-\beta_j)$.

Diffusion models: the backward process

Note that

$$\pi_{1:n|0}(x_{1:n}|x_0) = \pi_{n|0}(x_n|x_0) \prod_{k=2}^{n} \pi_{k-1|0,k}(x_{k-1}|x_0, x_k)$$

where $\pi_{n|0}(x_n|x_0) = \mathcal{N}(x_n; \bar{\alpha}_n^{1/2} x_0, (1 - \bar{\alpha}_n)\mathrm{I})$ and

$$\begin{aligned}\pi_{k-1|0,k}(x_{k-1}|x_0, x_k) &\propto \pi_{k-1|0}(x_{k-1}|x_0)\pi_{k|0}(x_k|x_0) \\ &= \mathcal{N}\left(x_{k-1}; \boldsymbol{\mu}_k(x_0, x_k), \sigma_k^2 \mathrm{I}_d\right) ,\end{aligned}$$

with

$$\boldsymbol{\mu}_k(x_0, x_k) = \bar{\alpha}_{k-1}^{1/2} x_0 + (1 - \bar{\alpha}_{k-1} - \sigma_k^2)^{1/2} (x_k - \bar{\alpha}_k^{1/2} x_0) / (1 - \bar{\alpha}_k)^{1/2}.$$

$\rightsquigarrow$ We know how to write the joint distribution of $X_{1:n}$ given X_0.

$\rightsquigarrow$ Use this decomposition to turn noise into samples from π_{data}.

$$\mathsf{p}_{0:n}^{\theta}(\mathrm{d}x_{0:n}) = \mathsf{p}_n(\mathrm{d}x_n) \prod_{k=0}^{n-1} p_k^{\theta}(\mathrm{d}x_k|x_{k+1}) .$$

Diffusion models: the backward process

$\rightsquigarrow$ Use this decomposition to turn noise into samples from π_{data}.

$$\mathsf{p}^{\theta}_{0:n}(\mathrm{d}x_{0:n}) = \mathsf{p}_n(\mathrm{d}x_n) \prod_{k=0}^{n-1} p^{\theta}_k(\mathrm{d}x_k|x_{k+1}) \,,$$

where p_n is a std Gaussian and

$$p^{\theta}_k(\mathrm{d}x_k|x_{k+1}) = \mathcal{N}(\mathrm{d}x_k; \mu^{\theta}_{k+1}(x_{k+1}), \beta_{k+1}\mathrm{I}_{d_x})$$

with $\mu^{\theta}_{k+1}(x_{k+1})$ obtained by replacing x_0 in $\boldsymbol{\mu}_{k+1}(x_0, x_{k+1})$ with a prediction

$$\hat{x}_{0|k,\theta}(x_{k+1}) := \bar{\alpha}_{k+1}^{-1/2} \left(x_{k+1} - (1-\bar{\alpha}_{k+1})^{1/2} \mathbf{e}^{\theta}(x_{k+1}, k+1) \right) .$$

Diffusion models: the backward process

We use

$$p_k^\theta(\mathrm{d}x_k|x_{k+1}) = \mathcal{N}(\mathrm{d}x_k; \mu_{k+1}^\theta(x_{k+1}), \beta_{k+1}\mathrm{I}_{d_x})$$

with $\mu_{k+1}^\theta(x_{k+1})$ obtained by replacing x_0 in $\boldsymbol{\mu}_{k+1}(x_0, x_{k+1})$ with a prediction

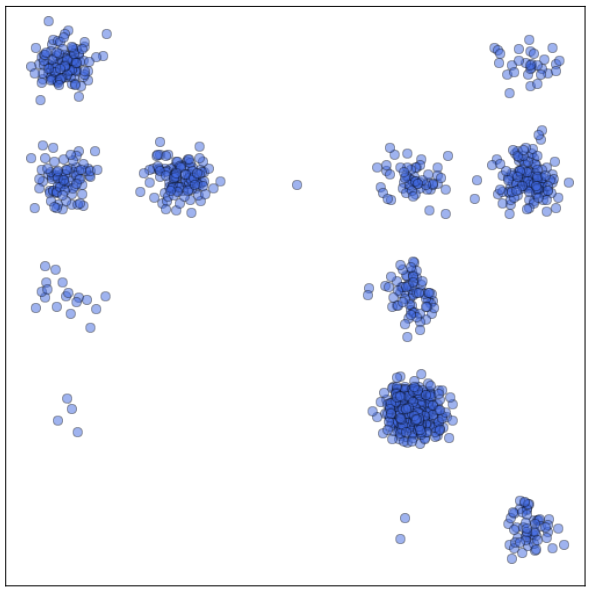
$$\hat{x}_{0|k,\theta}(x_{k+1}) := \bar{\alpha}_{k+1}^{-1/2}\left(x_{k+1} - (1-\bar{\alpha}_{k+1})^{1/2}\mathbf{e}^\theta(x_{k+1}, k+1)\right) .$$

$\mathbf{e}^{\theta_*}(X_t, t)$ might be seen as the predictor of the noise added to X_0 to obtain X_t (in the forward pass) and justifies the *prediction* terminology.

The parameter θ is obtained by **minimizing a variational loss between the forward and backward joint distributions**.

Diffusion models: an illustration

Data distribution π_{data}



Marginal p_{500}

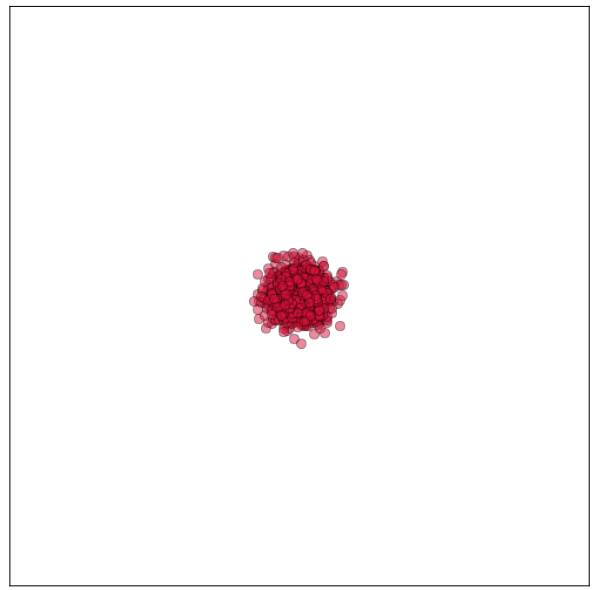
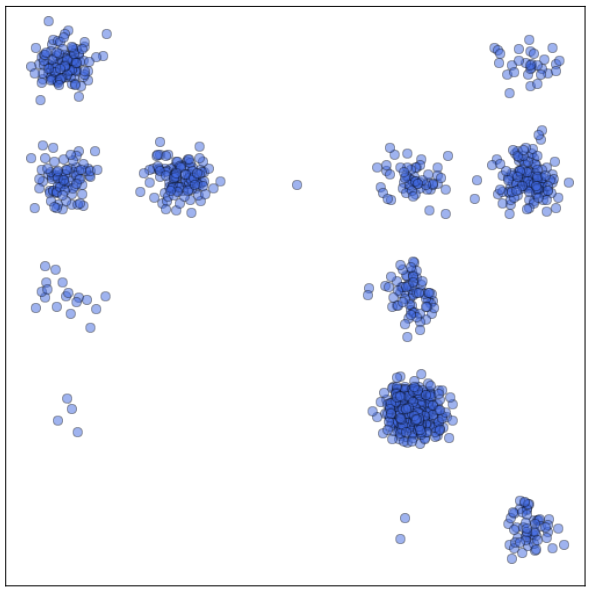


Figure: Samples from p_t for some time steps ranging from $n := 500$ to 1. π_0 is a Gaussian mixture.

Diffusion models: an illustration

Data distribution π_{data}



Marginal p_{150}

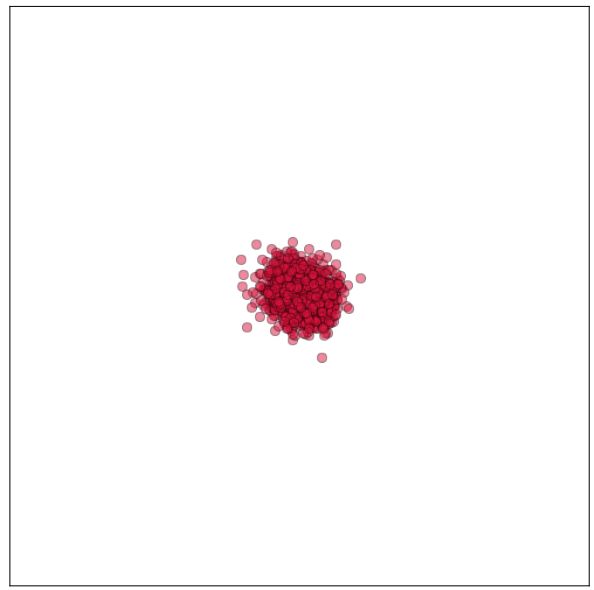
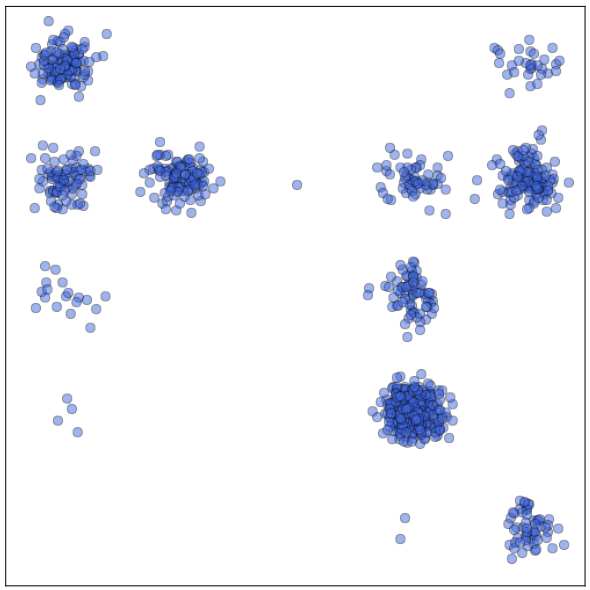


Figure: Samples from p_t for some time steps ranging from $n := 500$ to 1. π_0 is a Gaussian mixture.

Diffusion models: an illustration

Data distribution π_{data}



Marginal p_{100}

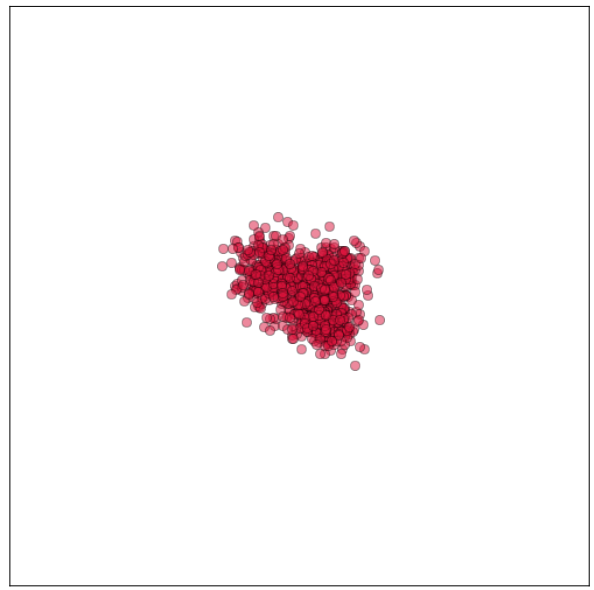
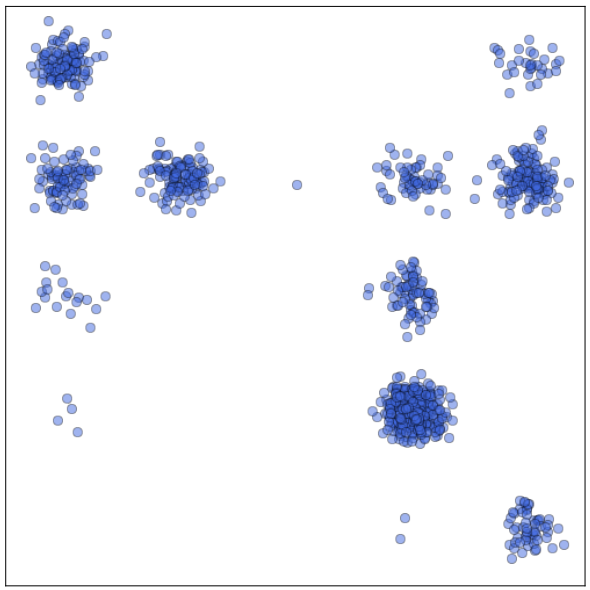


Figure: Samples from p_t for some time steps ranging from $n := 500$ to 1. π_0 is a Gaussian mixture.

Diffusion models: an illustration

Data distribution π_{data}



Marginal p_{80}

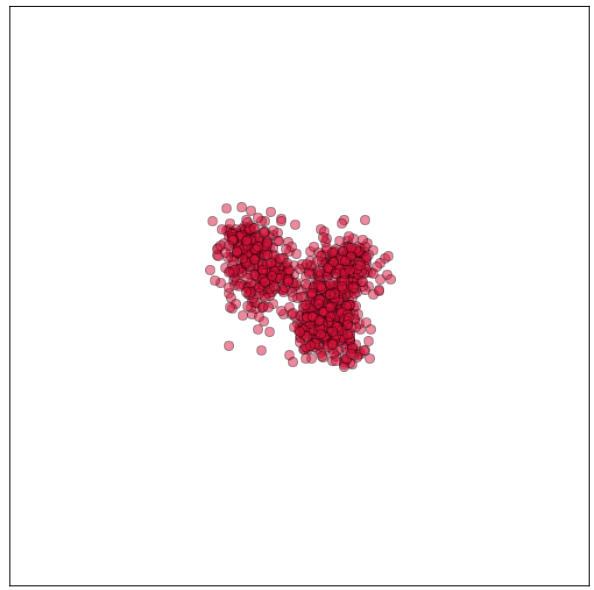
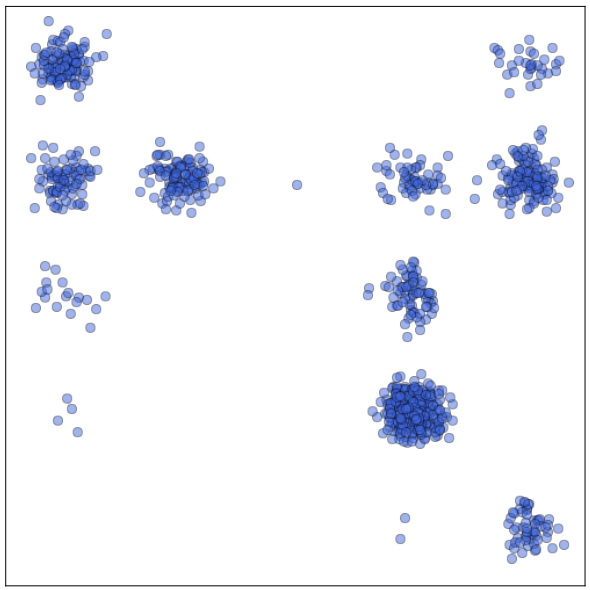


Figure: Samples from p_t for some time steps ranging from $n := 500$ to 1. π_0 is a Gaussian mixture.

Diffusion models: an illustration

Data distribution π_{data}



Marginal p_{70}

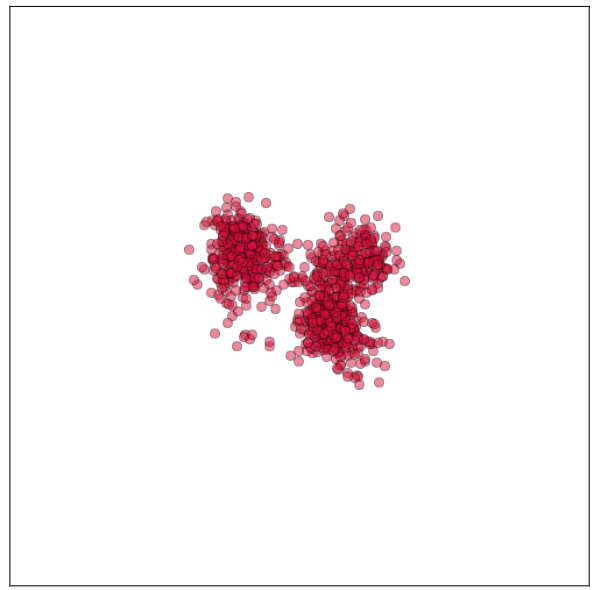
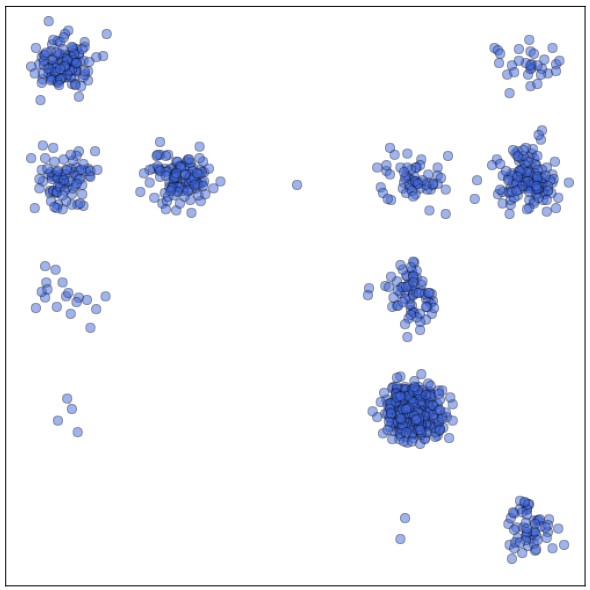


Figure: Samples from p_t for some time steps ranging from $n := 500$ to 1. π_0 is a Gaussian mixture.

Diffusion models: an illustration

Data distribution π_{data}



Marginal p_{50}

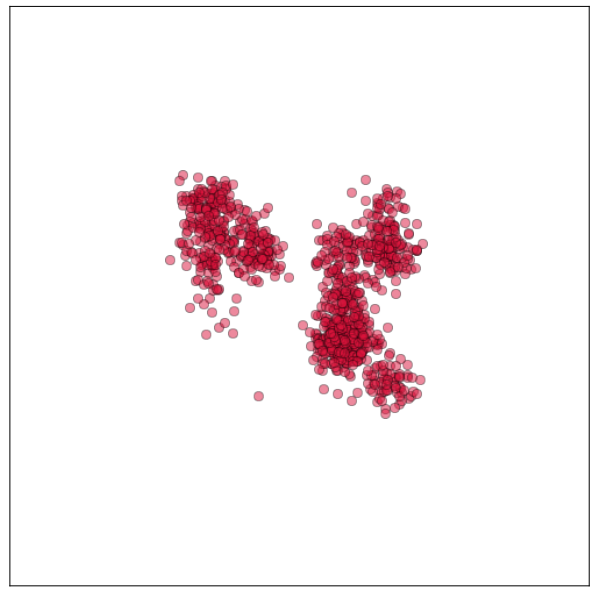
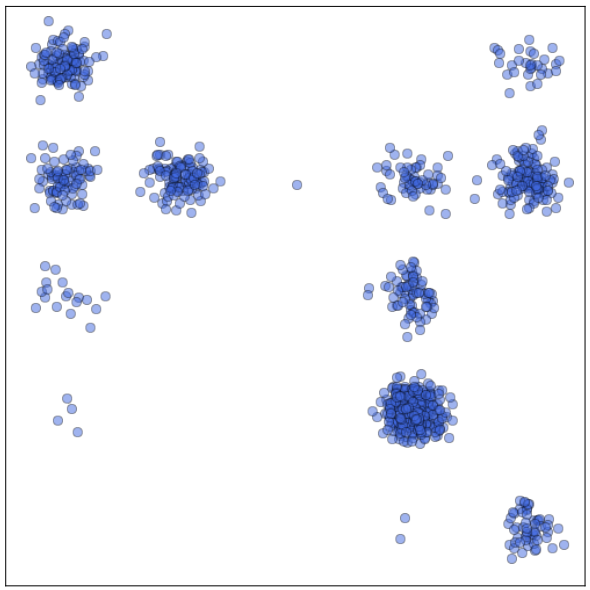


Figure: Samples from p_t for some time steps ranging from $n := 500$ to 1. π_0 is a Gaussian mixture.

Diffusion models: an illustration

Data distribution π_{data}



Marginal p_{40}

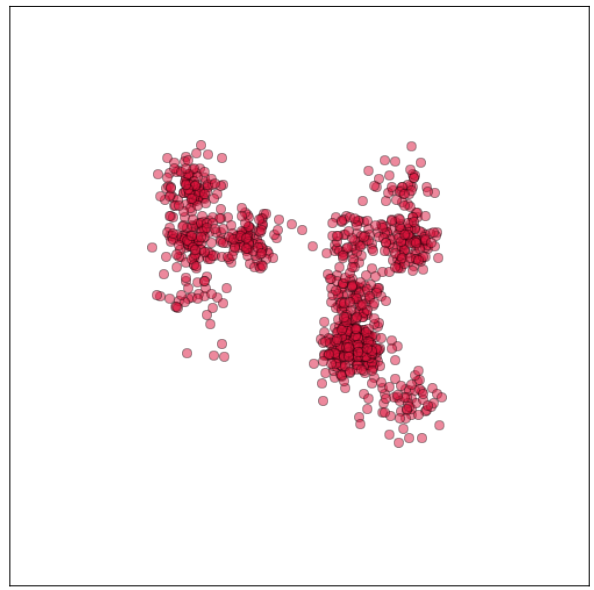
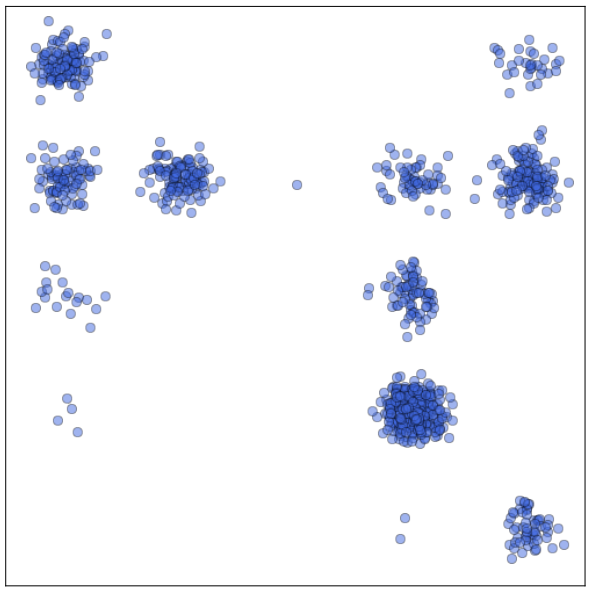


Figure: Samples from p_t for some time steps ranging from $n := 500$ to 1. π_0 is a Gaussian mixture.

Diffusion models: an illustration

Data distribution π_{data}



Marginal p_{20}

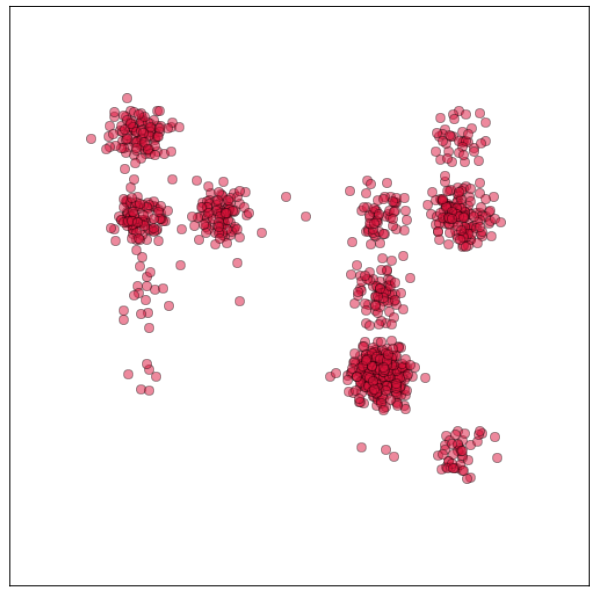
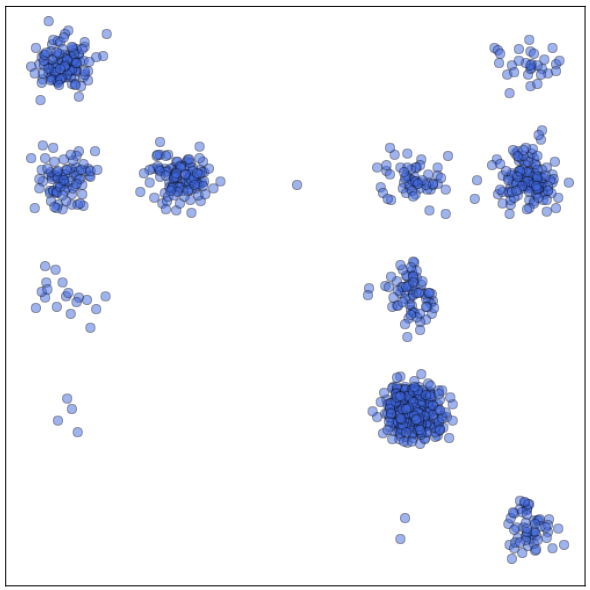


Figure: Samples from p_t for some time steps ranging from $n := 500$ to 1. π_0 is a Gaussian mixture.

Diffusion models: an illustration

Data distribution π_{data}



Marginal p_{15}

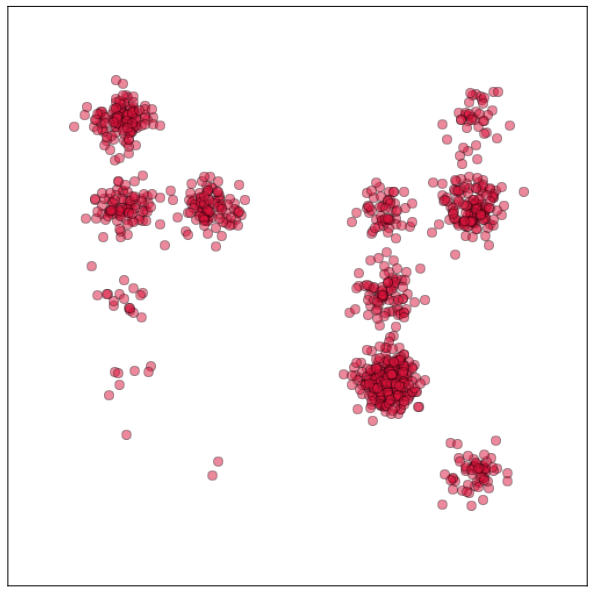
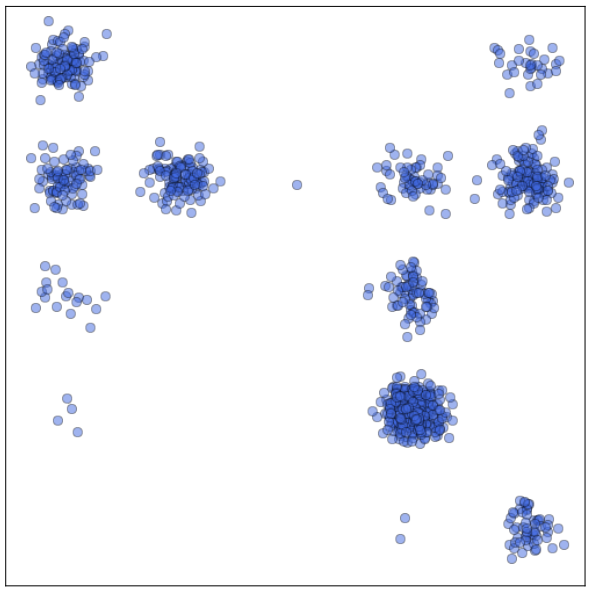


Figure: Samples from p_t for some time steps ranging from $n := 500$ to 1. π_0 is a Gaussian mixture.

Diffusion models: an illustration

Data distribution π_{data}



Marginal p_5

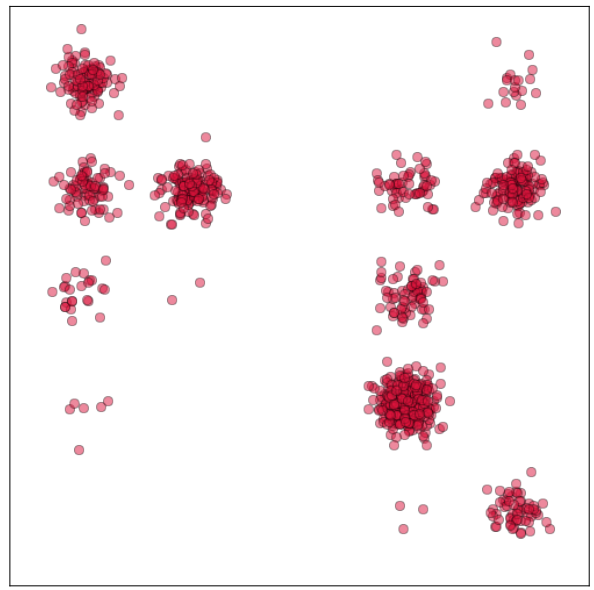
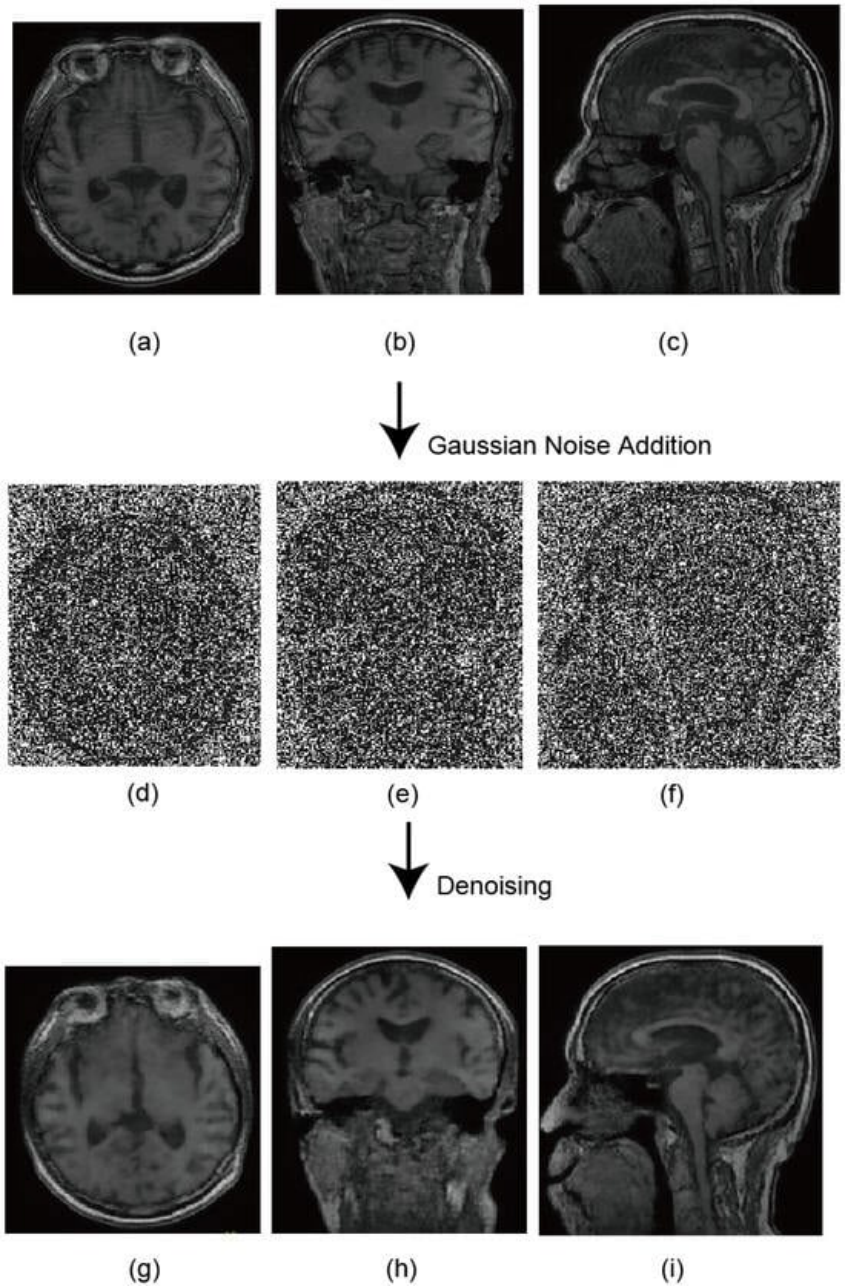


Figure: Samples from p_t for some time steps ranging from $n := 500$ to 1. π_0 is a Gaussian mixture.

(Shibata et al., 2024)



Why it works ? (Ronneberger et al., 2015)

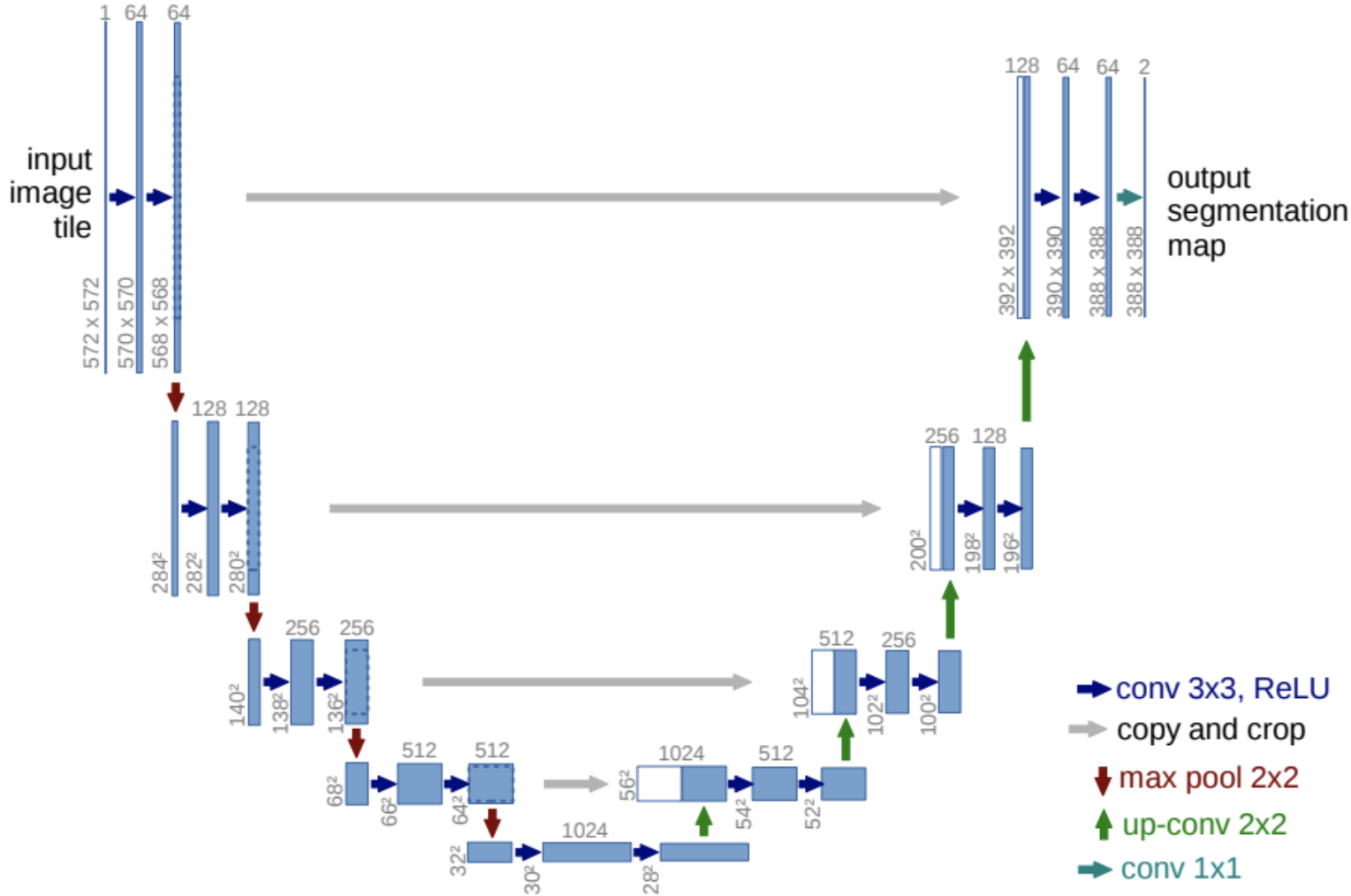
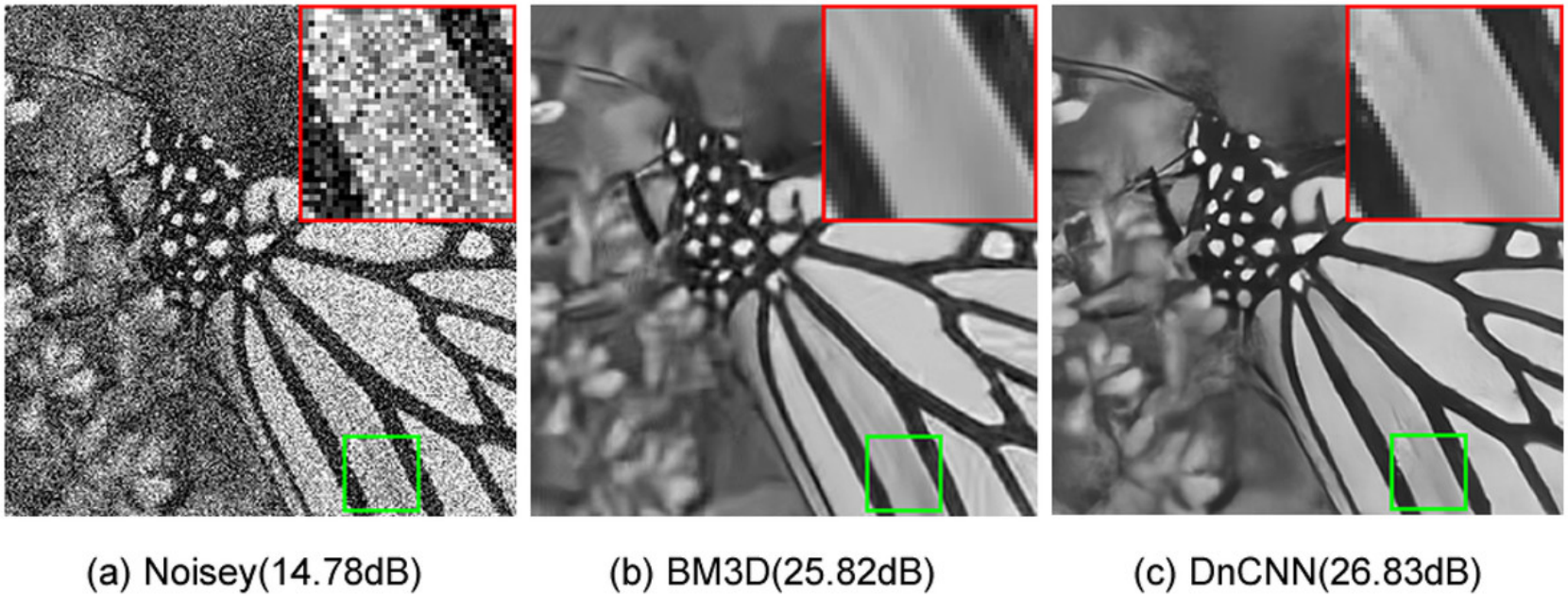


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

(Zhang et al., 2022)



(a) Noisey(14.78dB) (b) BM3D(25.82dB) (c) DnCNN(26.83dB)

(d) FFDNet(26.92dB) (e) RNAN(27.18dB) (f) RatUNet(27.22dB)

Guarantees on the approximation of π_{data}
Score-based training procedures

SGM - Forward phase

Data noised using the Ornstein–Uhlenbeck (OU) process:

$$\mathrm{d}\overrightarrow{X}_t = -\frac{\beta(t)}{2}\overrightarrow{X}_t \mathrm{d}t + \beta^{1/2}(t)\mathrm{d}B_t, \quad \overrightarrow{X}_0 \sim \pi_{\mathrm{data}}\,.$$

Fix $T > 0$, then,

$$\mathrm{KL}\left(\mathcal{L}\left(\overrightarrow{X}_T\right), \pi_\infty\right) \lesssim \exp\left(-\frac{1}{2}\int_0^T \beta(s)\mathrm{d}s\right)\mathrm{KL}\left(\pi_{\mathsf{data}}, \pi_\infty\right)$$

with

$$\mathrm{KL}\left(\mu, \nu\right) := \int \log\left(\frac{\mathrm{d}\mu(x)}{\mathrm{d}\nu(x)}\right)\mu(\mathrm{d}x)\,.$$

Fokker-Planck for $(p_t)_{0\leq t\leq T}$ + logarithmic Sobolev inequality + Gronwall's inequality

Time reversal

For $p_t := \mathcal{L}(\overrightarrow{X}_t)$, $(t,x) \mapsto \nabla \log p_t(x)$ is the **score function**. We consider the time reversal of the forward process, $i.e.$, the process satisfying

$$\mathrm{d}\overleftarrow{X}_t = \left(\frac{\bar{\beta}(t)}{2} \overleftarrow{X}_t + \bar{\beta}(t) \nabla \log p_{T-t} \left(\overleftarrow{X}_t \right) \right) \mathrm{d}t + \bar{\beta}^{1/2}(t) \mathrm{d}B_t, \quad \overleftarrow{X}_0 \sim p_T \,,$$

with $\bar{\beta}(t) := \beta(T-t)$. It satisfies

$$\left(\overrightarrow{X}_t \right)_{t \in [0,T]} = \left(\overleftarrow{X}_{T-t} \right)_{t \in [0,T]} \,.$$

In our setting,

$$\mathcal{L}\left(\overrightarrow{X}_T \right) = \mathcal{L}\left(\overleftarrow{X}_0 \right) \approx \pi_\infty \,, \qquad \mathcal{L}\left(\overleftarrow{X}_T \right) = \mathcal{L}\left(\overrightarrow{X}_0 \right) \approx \pi_{\mathsf{data}} \,.$$

Score matching in theory

The backward dynamics is

$$\mathrm{d}\overleftarrow{X}_t = \left(\frac{\bar{\beta}(t)}{2} \overleftarrow{X}_t + \bar{\beta}(t) \nabla \log p_{T-t} \left(\overleftarrow{X}_t \right) \right) \mathrm{d}t + \bar{\beta}^{1/2}(t) \mathrm{d}B_t, \quad \overleftarrow{X}_0 \sim p_T \, .$$

If the score is known, we can (in theory) **simulate the backward process and get data from noise**.

Let $s_\theta : [0, T] \times \mathbb{R}^d \mapsto \mathbb{R}^d$ be such that

$$\mathcal{L}_{\text{score}}(\theta) = \mathbb{E} \left[\left\| s_\theta \left(\tau, \overrightarrow{X}_\tau \right) - \nabla \log p_\tau \left(\overrightarrow{X}_\tau \right) \right\|^2 \right] ,$$

with $\tau \sim \mathcal{U}(0, T)$ independent of the forward process $(\overrightarrow{X}_t)_{t \geq 0}$.

Score matching in practice

As the linear part can be simulated exactly, we consider the **Exponential Integrator scheme**:

Let $0 =: t_0 \leq t_1 \leq \cdots \leq t_N := T$. Consider

$$\mathrm{d}\overleftarrow{X}^{\theta}_t = \bar{\beta}(t)\left(\frac{1}{2\sigma^2}\overleftarrow{X}^{\theta}_t + s_\theta\left(T - t_k, \overleftarrow{X}^{\theta}_{t_k}\right)\right)\mathrm{d}t + \bar{\beta}^{1/2}(t)\mathrm{d}B_t\,,$$

for $t \in [t_k, t_{k+1})$, with $\overleftarrow{X}^{\theta}_0 \sim \pi_\infty$.

We denote $\widehat{\pi}^{(\beta,\theta)}_N$ the marginal probability density of $\overleftarrow{X}^{\theta}_T$.

The loss function is built using the conditional score:

$$\mathcal{L}(\theta) = \mathbb{E}\left[\alpha_\tau \left\|\nabla \log p_{\tau|0}(X_\tau|X_0) - s_\theta(\tau, X_\tau)\right\|^2\right]\,.$$

Diffusion models: convergence

In early works: strong assumptions on the data distribution (polynomial growth of the score), (de Bortoli et al., 2021):

$$\left\| \pi_{\mathrm{data}} - \widehat{\pi}_N^{(\beta,\theta)} \right\|_{\mathrm{tv}} \leq c_0 M \exp(c_1 T) + c_2 \left(T^{-1} + T^{-1/2} \right) ,$$

where M **quantifies the quality of the score approximation**.

In most recent works, we only require π_{data} to have a finite relative Fisher information w.r.t the standard Gaussian distribution, (Conforti et al., 2023):

$$\mathrm{KL}(\pi_{\mathrm{data}}, \widehat{\pi}_N^{(\beta,\theta)}) \leq \exp(-c_0 T) \mathrm{KL}(\mathcal{N}(0, I_d), \pi_{\mathrm{data}}) + MT + c_1 h \, ,$$

Assuming that $\mathbb{E}_{\pi_{\mathrm{data}}}[\|\nabla \log(\mathrm{d}\pi_{\mathrm{data}}/\mathrm{d}\gamma_d)\|^2] < \infty$.

Diffusion models: convergence

In more recent works, assuming that π_{data} is smooth and strongly log-concave, (Lee et al., 2022), (Gao et al., 2023) : **explicit upper bounds in Wasserstein distance**.

Under similar assumptions, (Strasman et al., 2024) proposed a **unified framework for time-inhomogeneous SGMs, with joint theoretical analyses in KL and Wasserstein metrics**.

$$\mathrm{KL}\left(\pi_{\text{data}} \middle|\middle| \widehat{\pi}_N^{(\beta,\theta)}\right) \leq \mathcal{E}_1^{\mathrm{KL}}(\beta) + \mathcal{E}_2^{\mathrm{KL}}(\theta,\beta) + \mathcal{E}_3^{\mathrm{KL}}(\beta)\,,$$

$$\mathcal{E}_1^{\mathrm{KL}}(\beta) = \mathrm{KL}\left(\pi_{\text{data}} || \pi_\infty\right) \exp\left\{ -\frac{1}{\sigma^2}\int_0^T \beta(s)\mathrm{d}s \right\},$$

$$\mathcal{E}_2^{\mathrm{KL}}(\theta,\beta) = \sum_{k=0}^{N-1} \mathbb{E}\left[\left\| \nabla \log p_{\tau_k}\left(\overrightarrow{X}_{\tau_k}\right) - s_\theta\left(\tau_k, \overrightarrow{X}_{\tau_k}\right) \right\|^2 \right] \int_{\tau_{k+1}}^{\tau_k} \beta(t)\mathrm{d}t\,,$$

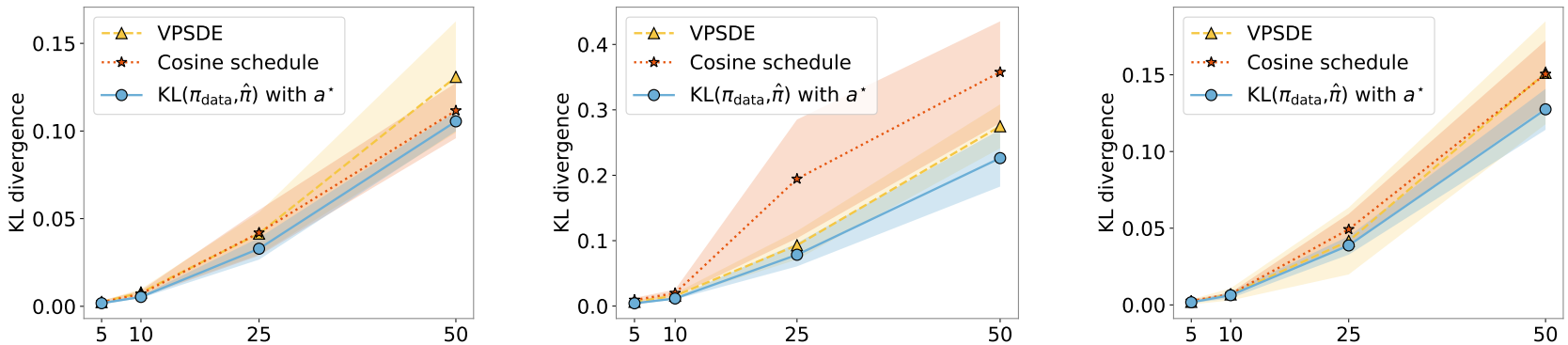
$$\mathcal{E}_3^{\mathrm{KL}}(\beta) = 2h\beta(T)\mathcal{I}(\pi_{\text{data}}|\pi_\infty)\,,$$

Gaussian case

Let the true distribution be Gaussian in dimension $d = 50$ with mean $\mathbf{1}_d$ and different choices of covariance structure.

1. (Isotropic) $\Sigma^{(\mathrm{iso})} = 0.5\mathrm{I}_d$.
2. (Heteroscedastic) $\Sigma^{(\mathrm{heterosc})} \in \mathbb{R}^{d \times d}$ is a diagonal matrix such that $\Sigma_{jj}^{(\mathrm{heterosc})} = 10$ for $1 \leq j \leq 5$, and $\Sigma_{jj}^{(\mathrm{heterosc})} = 0.1$ otherwise.
3. (Correlated) $\Sigma^{(\mathrm{corr})} \in \mathbb{R}^{d \times d}$ is a full matrix whose diagonal entries are equal to one and the off-diagonal terms are $\Sigma_{jj'}^{(\mathrm{corr})} = 1/\sqrt{|j - j'|}$ for $1 \leq j \neq j' \leq d$.

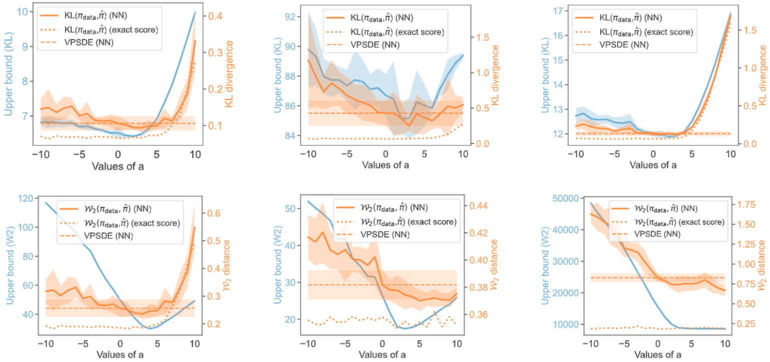
Comparison with existing literature



(a) Isotropic setting (b) Heteroscedastic setting (c) Correlated setting

Figure: Comparison of the empirical KL divergence between π_{data} and the generative distribution $\widehat{\pi}_N^{(\beta,\theta)}$ w.r.t. SGM for $\beta_{a^\star}$, the VPSDE model and the one with a cosine schedule, presented in Chen et al. (2023).

- Optimizing the noise schedule has an impact even with simple parametrization of the β scheduling.


KL(π_{data},$\hat{\pi}$) (NN)
KL(π_{data},$\hat{\pi}$) (exact score)
VPSDE (NN)
Upper bound (KL)
KL divergence
Values of a
$\mathcal{W}_2$(π_{data}, $\hat{\pi}$) (NN)
$\mathcal{W}_2$(π_{data},$\hat{\pi}$) (exact score)
VPSDE (NN)
Upper bound (W2)
$\mathcal{W}_2$ distance
Values of a

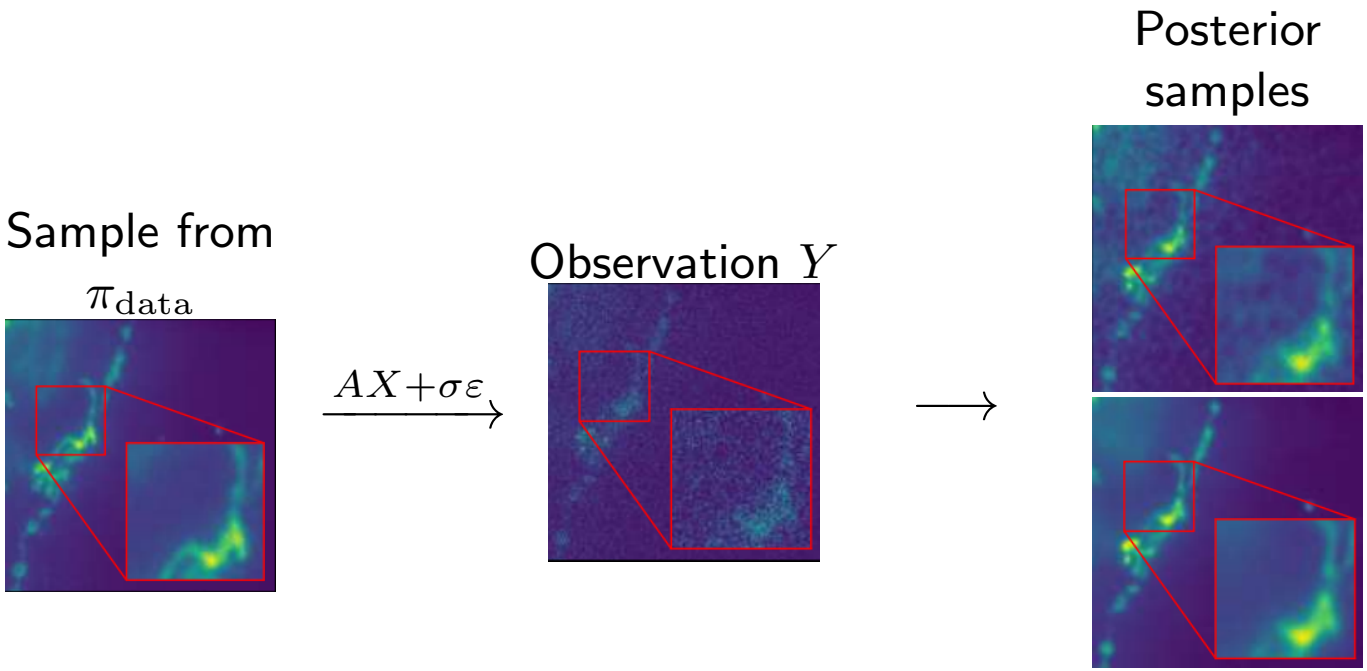
Bayesian inverse problem

Original problem

Bayesian inverse problem:

$$Y = f(X) + \sigma\varepsilon, \quad \text{where} \quad \varepsilon \sim \mathcal{N}(\mathbf{0}_{d_x}, \mathrm{I}_{d_x}), \quad X \sim \pi_{\text{data}}, \quad \sigma \geq 0\,.$$

Objective: Sample the distribution of X given Y.



Sample from the W2S dataset [Zhou et al., 2020].

Proposed framework

Objective: Sample the distribution of X given a realisation of Y.

We only have access to samples $X^1, \dots, X^M \sim \pi_{\text{data}}$.

$\rightsquigarrow$ **Step 1**: train a diffusion model $\mathsf{p}_0^{\theta^*}$ to estimate π_{data}.

$\rightsquigarrow$ **Step 2:** consider the approximate Bayesian inverse problem:

$$Y = AX + \sigma\varepsilon, \quad \text{where} \quad \varepsilon \sim \mathcal{N}(\mathbf{0}_{d_y}, \mathrm{I}_{d_y}), \quad X \sim \mathsf{p}_0^{\theta^*}, \quad \sigma \geq 0\,.$$

A specific application (**in-painting**):

$$Y = \overline{X} + \sigma\varepsilon, \quad \text{where} \quad \varepsilon \sim \mathcal{N}(\mathbf{0}_{d_y}, \mathrm{I}_{d_y}), \quad X \sim \mathsf{p}_0^{\theta^*}, \quad \sigma \geq 0\,,$$

where $\overline{X}$ are the top d_y coordinates of X.

$\rightsquigarrow$ **Step 3:** extend to settings where the conditional distribution of Y given X has probability density $x \mapsto g_0^Y(x)$.

Proposed framework

For all y, draw *approximate samples* from the posteriors

$$\phi_0^y(\mathrm{d}x) \propto g_0^y(x)\mathsf{p}_0^{\theta_*}(\mathrm{d}x)\,,$$

where $g_0^y(x) \propto \mathcal{N}(y, \overline{x}, \sigma^2 \mathrm{I}_{d_y})$.

$\rightarrow$ Classical setting where the target distribution is known up to a multiplicative constant (MCMC, Self-Normalized Importance Sampling, etc.).

$\rightarrow$ We do not know how to evaluate $\mathsf{p}_0^{\theta_*}$, only sample from it.

$\rightarrow$ Upper bound the error between $\phi_0^y(\mathrm{d}x)$ and the true posterior ?

Illustration of the posterior distribution

Assume $d_x = 2$, $d_y = 1$. Inverse problem:

$$Y = X_1, \quad (X_1, X_2) \sim \text{GaussianMixture}\,.$$

Illustration of the posterior distribution

Assume $d_x = 2$, $d_y = 1$. Inverse problem:

$$Y = X_1, \quad (X_1, X_2) \sim \text{GaussianMixture}\,.$$

p_0 and p_0 and ϕ_0^y

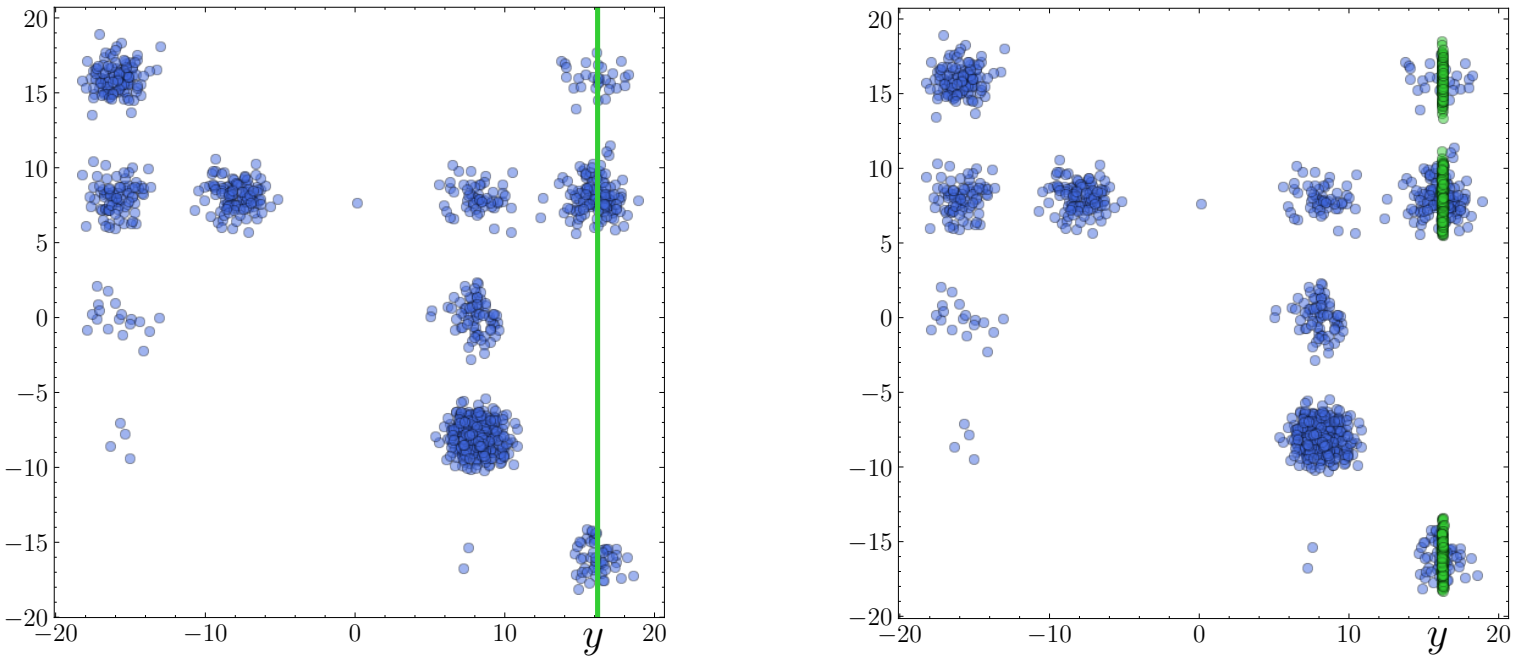


Figure: **Left plot**: prior in blue and green vertical line of points with first coordinate equal to y. **Right plot**: samples from ϕ_0^y are shown in green.

Posterior sampling problem

The target ϕ_0^y is approximated by our parametric posterior distribution:

$$\phi_0^y(x_0) = p_0^\theta(x_0|y) \propto p_0^\theta(x_0) g_0^y(x_0)\,,$$
$$\propto \int p_n(x_n) \left\{ \prod_{t=1}^{n-1} p_t^\theta(x_t|x_{t+1}) \right\} p_0^\theta(x_0|x_1) g_0^y(x_0) \mathrm{d}x_{1:n}\,.$$

The integral is not tractable!

Consider a sequence of positive potential functions $(g_k^y)_{1\leq k\leq n}$ and the sequence of probability distributions defined by

$$\phi_n^y(x_n) \propto g_n^y(x_n) p_n(x_n)$$

and for $1 \leq k \leq n-1$,

$$\phi_k^y(x_k) \propto \int \frac{g_k^y(x_k)}{g_{k+1}^y(x_{k+1})} p_k^\theta(x_k|x_{k+1}) \phi_{k+1}^y(x_{k+1}) \mathrm{d}x_{k+1}\,,$$

so that for all $k \geq 1$, $\phi_k^y(x_k) \propto g_k^y(x_k) p_k^\theta(x_k)$.

Posterior sampling proposal: recursion

Recall that $\phi_k^y(\mathrm{d}x_k) \propto g_k^y(x_k)\mathsf{p}_k(\mathrm{d}x_k)$. We cannot sample from it in all but trivial cases.

Using $\mathsf{p}_k(\mathrm{d}x_k) = \int p_k(\mathrm{d}x_k|x_{k+1})\mathsf{p}_{k+1}(\mathrm{d}x_{k+1})$,

$$\begin{aligned}\phi_k^y(\mathrm{d}x_k) &\propto \int \frac{g_k^y(x_k)}{g_{k+1}^y(x_{k+1})} p_k(\mathrm{d}x_k|x_{k+1})\phi_{k+1}^y(\mathrm{d}x_{k+1}) \\ &\propto \int \underbrace{\frac{\int g_k^y(z_k)p_k(\mathrm{d}z_k|x_{k+1})}{g_{k+1}^y(x_{k+1})}}_{:=\widetilde{\omega}_k(x_{k+1})} p_k^y(\mathrm{d}x_k|x_{k+1})\phi_{k+1}^y(\mathrm{d}x_{k+1})\,,\end{aligned}$$

where $p_k^y(\mathrm{d}x_k|x_{k+1}) = p_k(\mathrm{d}x_k|x_{k+1})g_k^y(x_k)$ is available in closed form in the linear and Gaussian setting.

Posterior sampling proposal

Construct a sequence of distributions $\{\phi_t^y\}_{t=1}^n$ that bridge between $\mathcal{N}(\mathbf{0}_{d_x}, \mathrm{I}_{d_x})$ and ϕ_0^y by leveraging the diffusion marginals $\{\mathsf{p}_t\}_{t=1}^n$.

Intuitive choice

- $\phi_n^y(\mathrm{d}x_n) \propto g_n^y(x_n)\mathsf{p}_n(\mathrm{d}x_n)$ with $g_n^y(x_n) = \mathcal{N}(\overline{x_n}; \bar{\alpha}_n^{1/2} y, (1-\bar{\alpha}_n)\mathrm{I})$.
- Proposal distribution.

$$
\begin{aligned}
p_k^y(\mathrm{d}x_k|x_{k+1}) &\propto p_k(\mathrm{d}x_k|x_{k+1}) g_k^y(x_k) \\
&\propto p_k(\mathrm{d}x_k|x_{k+1}) \mathcal{N}(\overline{x}_k; \bar{\alpha}_k^{1/2} y, (1-\bar{\alpha}_k)\mathrm{I}) \,.
\end{aligned}
$$

As $p_k^\theta(\mathrm{d}x_k|x_{k+1}) = \mathcal{N}(\mathrm{d}x_k; \mu_{k+1}^\theta(x_{k+1}), \beta_{k+1}\mathrm{I}_{d_x})$, the proposal distribution is available explicitly.

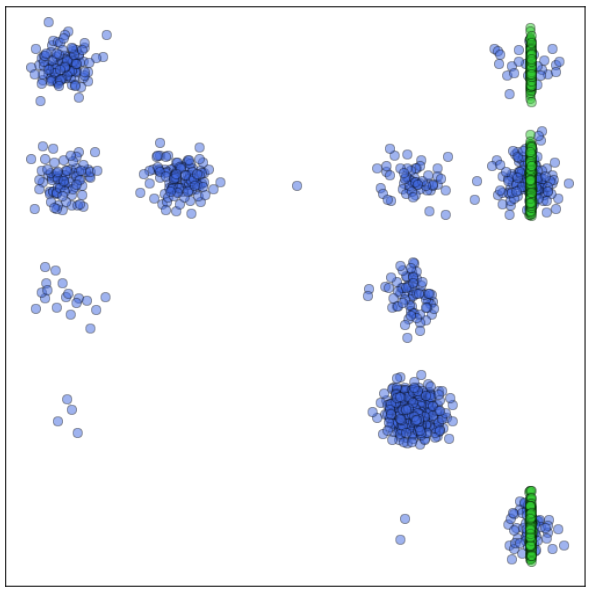
Posterior sampling proposal

$\rightsquigarrow$ $\{\phi_t^y\}_{t=1}^n$ is available in closed form for the Gaussian mixture example.

Posterior sampling proposal

$\rightsquigarrow \{\phi_t^y\}_{t=1}^n$ is available in closed form for the Gaussian mixture example.

π_0 and ϕ_0^y



Posterior ϕ_{450}^y

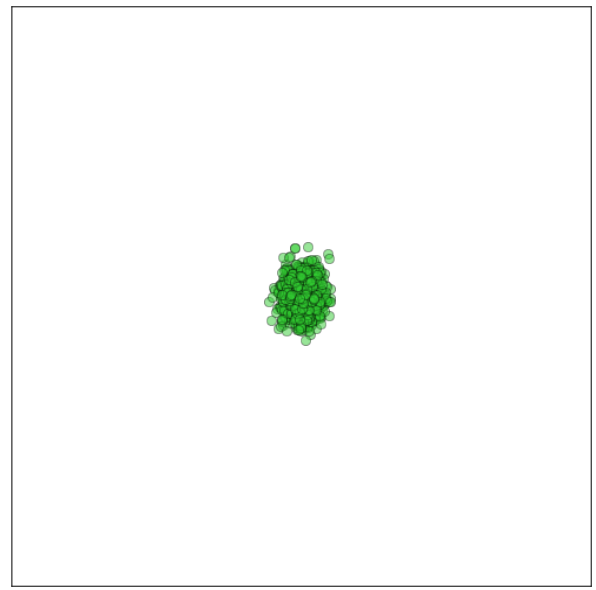
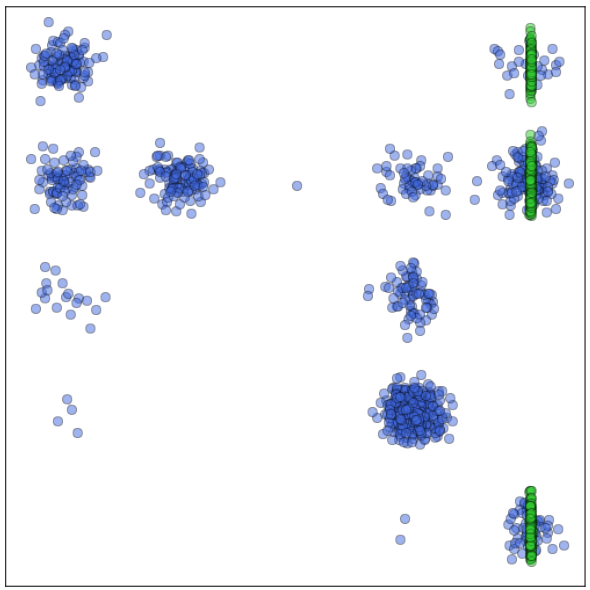


Figure: **Left plot:** samples from the prior π_0 and posterior ϕ_0^y. **Right plot:** samples from the posterior proposals ϕ_t^y for time steps ranging from $n := 500$ to 0.

Posterior sampling proposal

$\rightsquigarrow \{\phi_t^y\}_{t=1}^n$ is available in closed form for the Gaussian mixture example.

π_0 and ϕ_0^y



Posterior ϕ_{150}^y

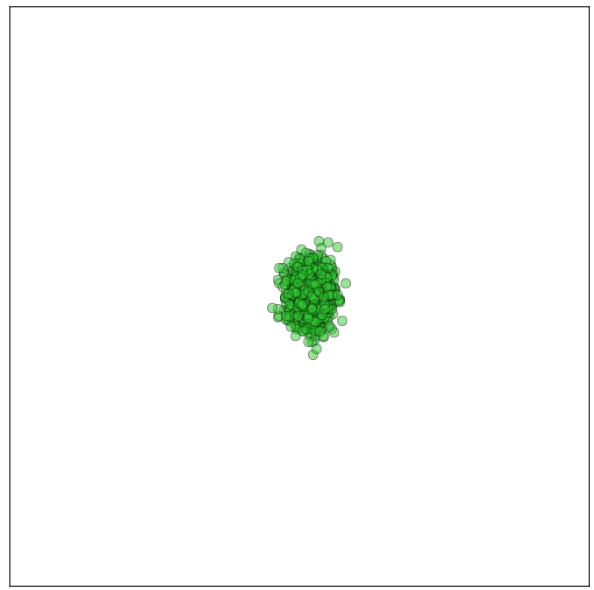


Figure: **Left plot:** samples from the prior π_0 and posterior ϕ_0^y. **Right plot:** samples from the posterior proposals ϕ_t^y for time steps ranging from $n := 500$ to 0.

Posterior sampling proposal

$\rightsquigarrow \{\phi_t^y\}_{t=1}^n$ is available in closed form for the Gaussian mixture example.

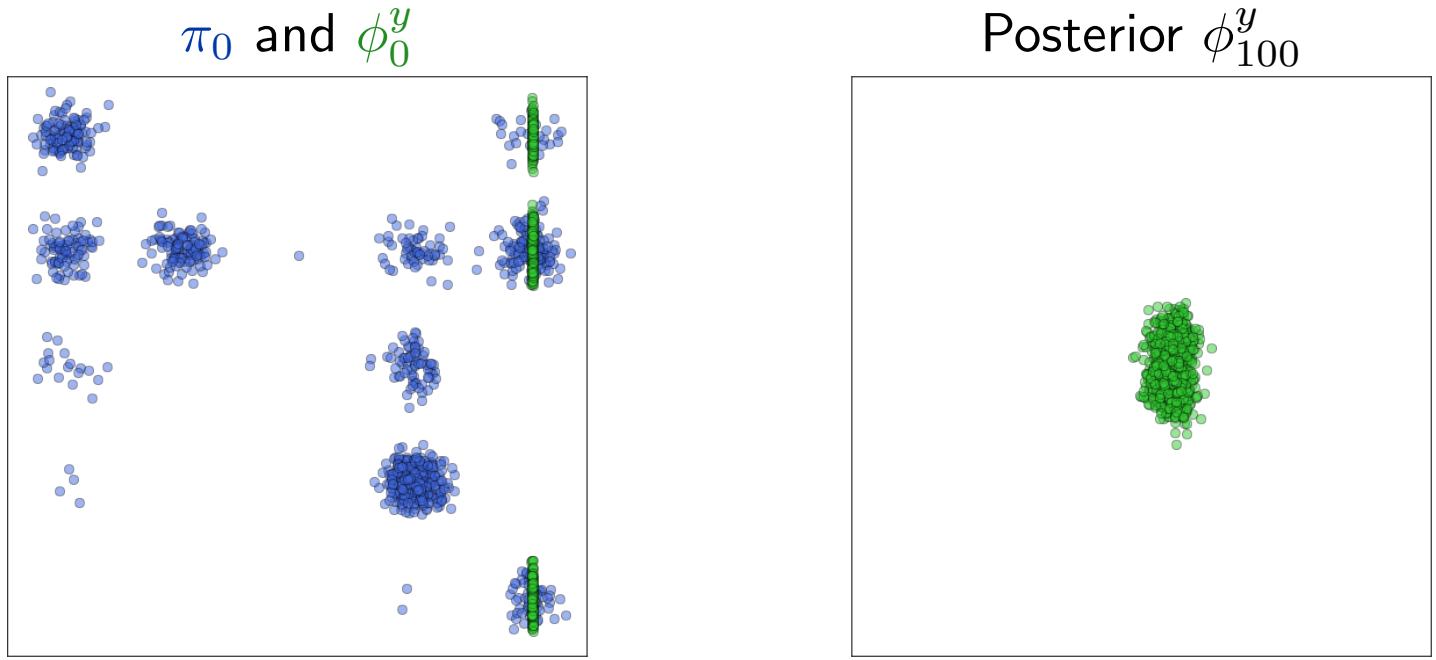
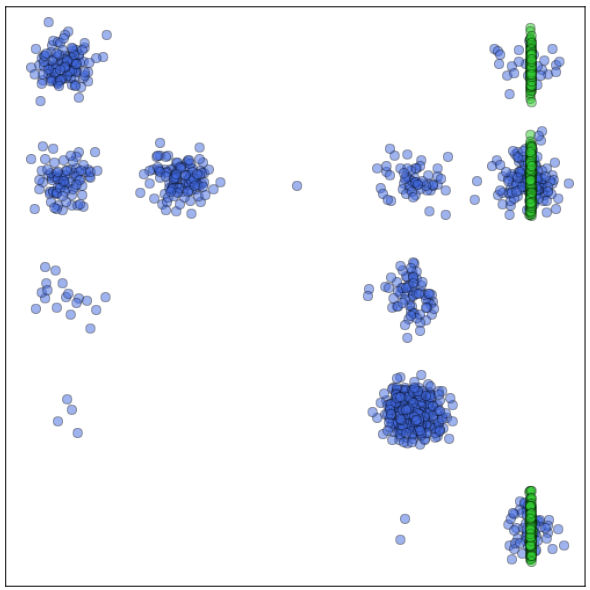


Figure: **Left plot:** samples from the prior π_0 and posterior ϕ_0^y. **Right plot:** samples from the posterior proposals ϕ_t^y for time steps ranging from $n := 500$ to 0.

Posterior sampling proposal

$\rightsquigarrow \{\phi_t^y\}_{t=1}^n$ is available in closed form for the Gaussian mixture example.

π_0 and ϕ_0^y



Posterior ϕ_{80}^y

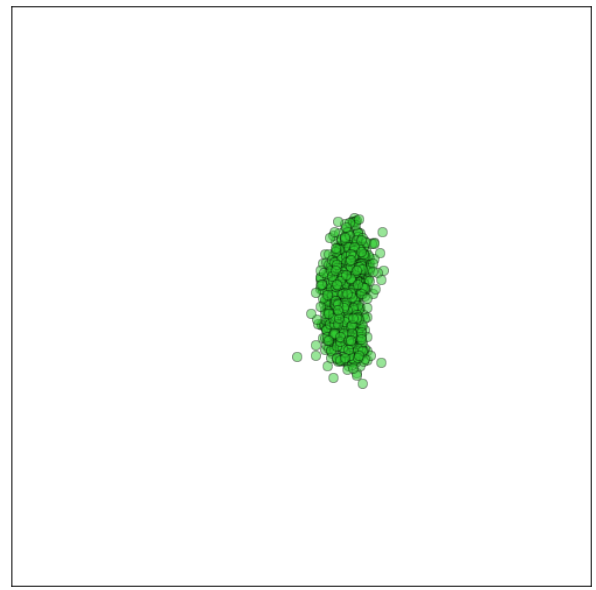
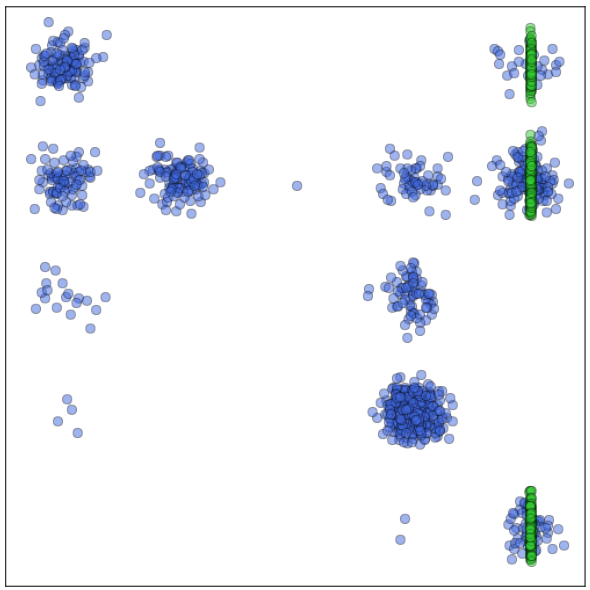


Figure: **Left plot:** samples from the prior π_0 and posterior ϕ_0^y. **Right plot:** samples from the posterior proposals ϕ_t^y for time steps ranging from $n := 500$ to 0.

Posterior sampling proposal

$\rightsquigarrow \{\phi_t^y\}_{t=1}^n$ is available in closed form for the Gaussian mixture example.

π_0 and ϕ_0^y



Posterior ϕ_{70}^y

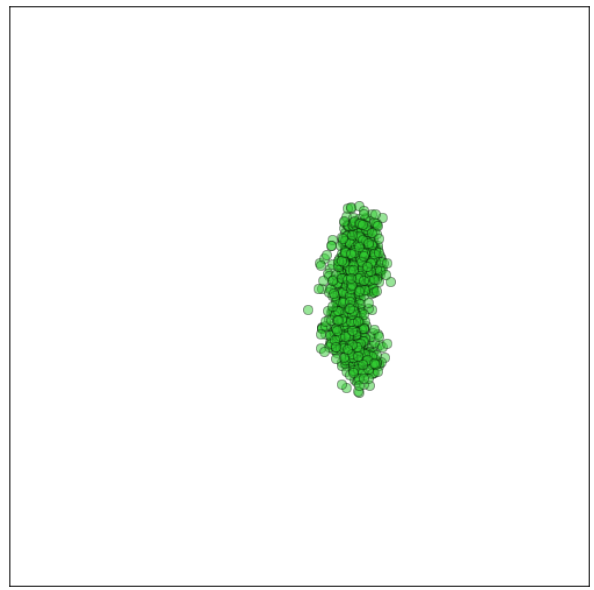
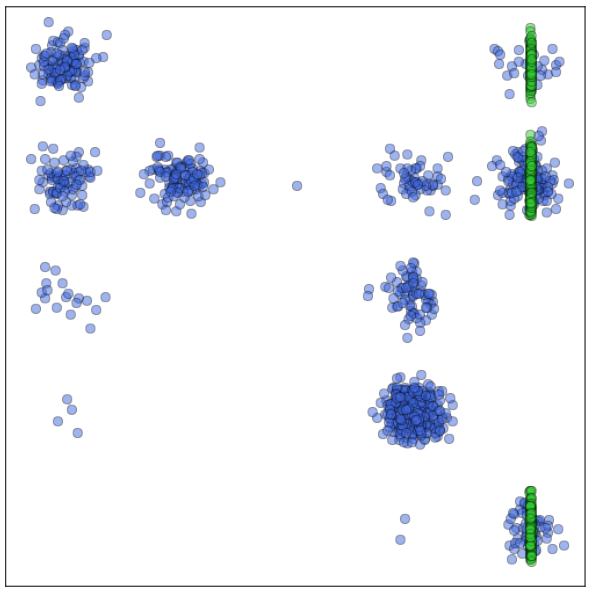


Figure: **Left plot:** samples from the prior π_0 and posterior ϕ_0^y. **Right plot:** samples from the posterior proposals ϕ_t^y for time steps ranging from $n := 500$ to 0.

Posterior sampling proposal

$\rightsquigarrow \{\phi_t^y\}_{t=1}^n$ is available in closed form for the Gaussian mixture example.

π_0 and ϕ_0^y



Posterior ϕ_{50}^y

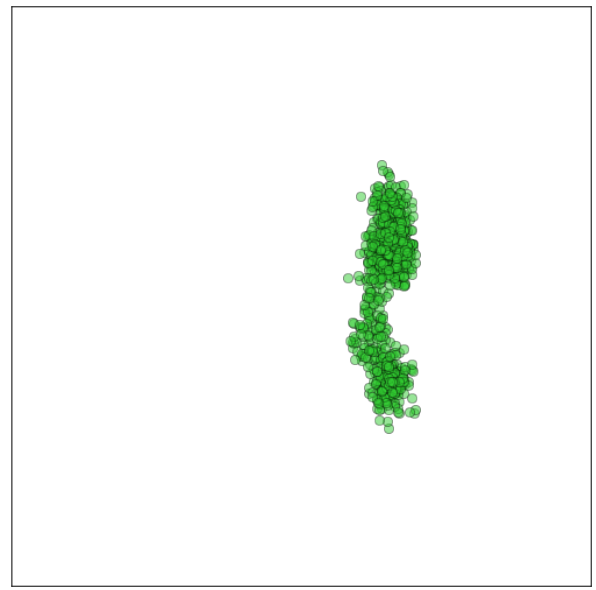
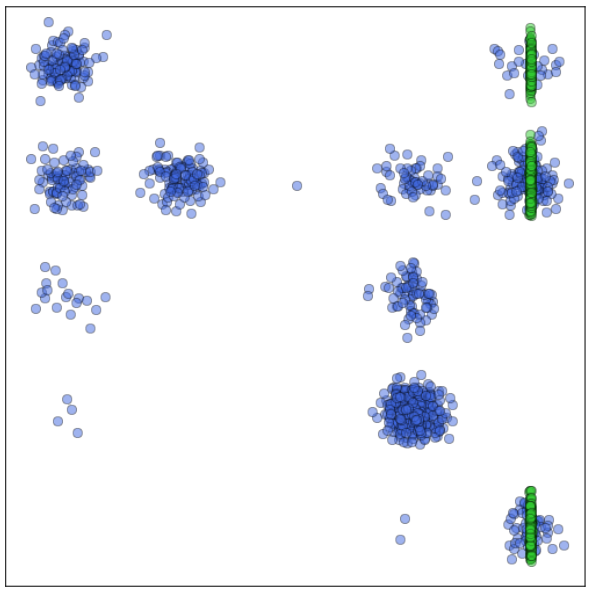


Figure: **Left plot:** samples from the prior π_0 and posterior ϕ_0^y. **Right plot:** samples from the posterior proposals ϕ_t^y for time steps ranging from $n := 500$ to 0.

Posterior sampling proposal

$\rightsquigarrow \{\phi_t^y\}_{t=1}^n$ is available in closed form for the Gaussian mixture example.

π_0 and ϕ_0^y



Posterior ϕ_{40}^y

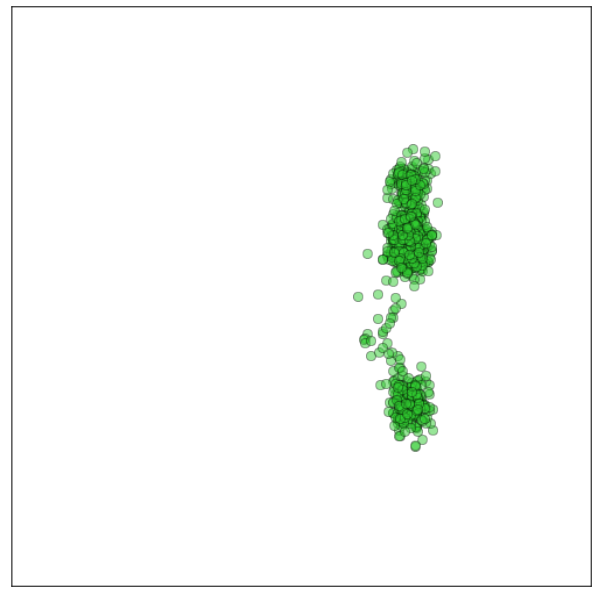
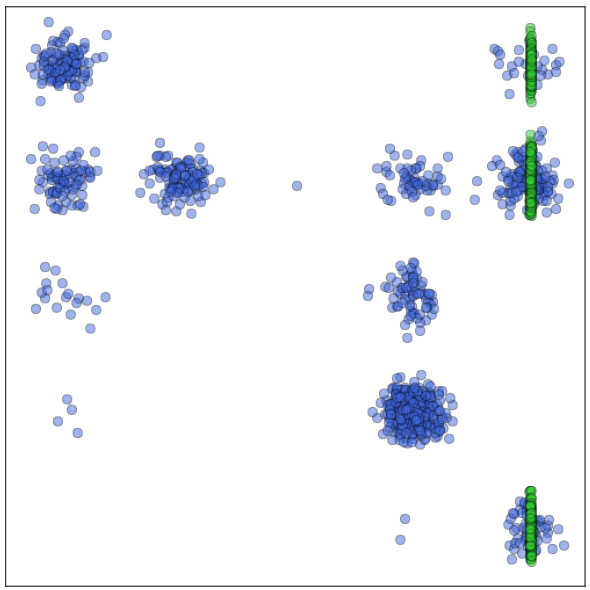


Figure: **Left plot:** samples from the prior π_0 and posterior ϕ_0^y. **Right plot:** samples from the posterior proposals ϕ_t^y for time steps ranging from $n := 500$ to 0.

Posterior sampling proposal

$\rightsquigarrow \{\phi_t^y\}_{t=1}^n$ is available in closed form for the Gaussian mixture example.

π_0 and ϕ_0^y



Posterior ϕ_{20}^y

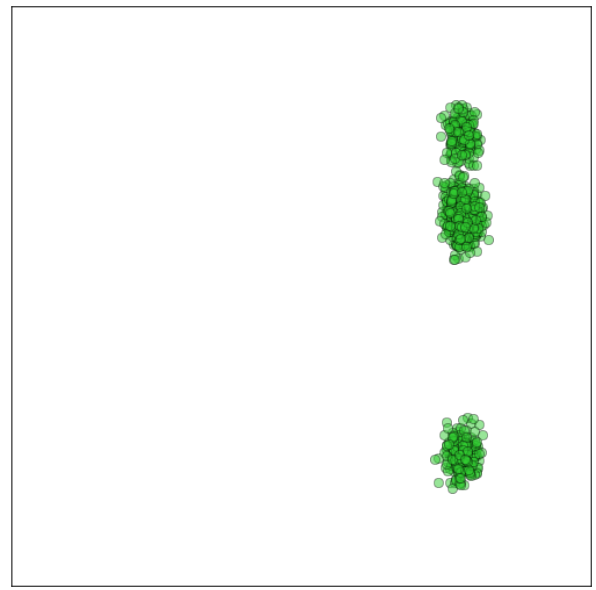
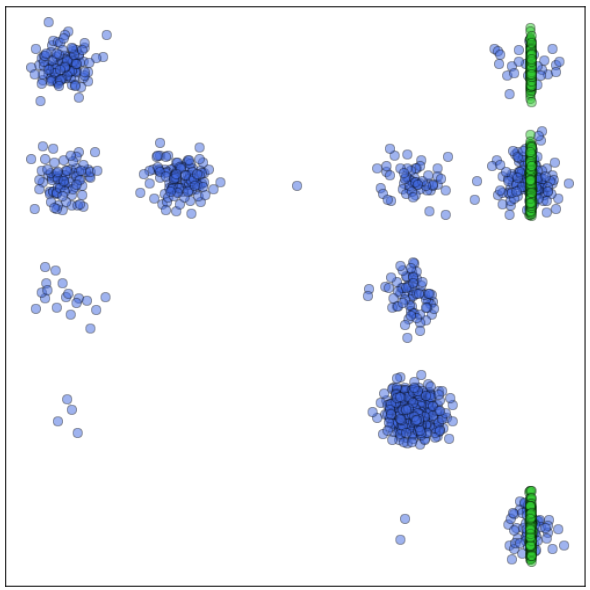


Figure: **Left plot:** samples from the prior π_0 and posterior ϕ_0^y. **Right plot:** samples from the posterior proposals ϕ_t^y for time steps ranging from $n := 500$ to 0.

Posterior sampling proposal

$\rightsquigarrow \{\phi_t^y\}_{t=1}^n$ is available in closed form for the Gaussian mixture example.

π_0 and ϕ_0^y



Posterior ϕ_{15}^y

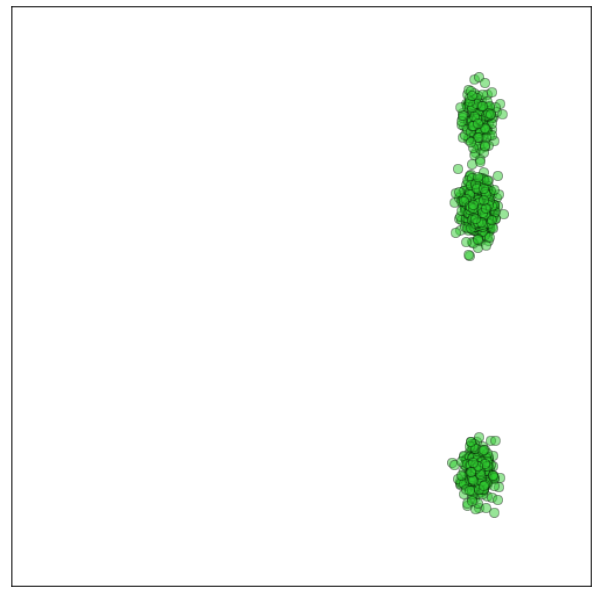


Figure: **Left plot:** samples from the prior π_0 and posterior ϕ_0^y. **Right plot:** samples from the posterior proposals ϕ_t^y for time steps ranging from $n := 500$ to 0.

Posterior sampling proposal

$\rightsquigarrow \{\phi_t^y\}_{t=1}^n$ is available in closed form for the Gaussian mixture example.

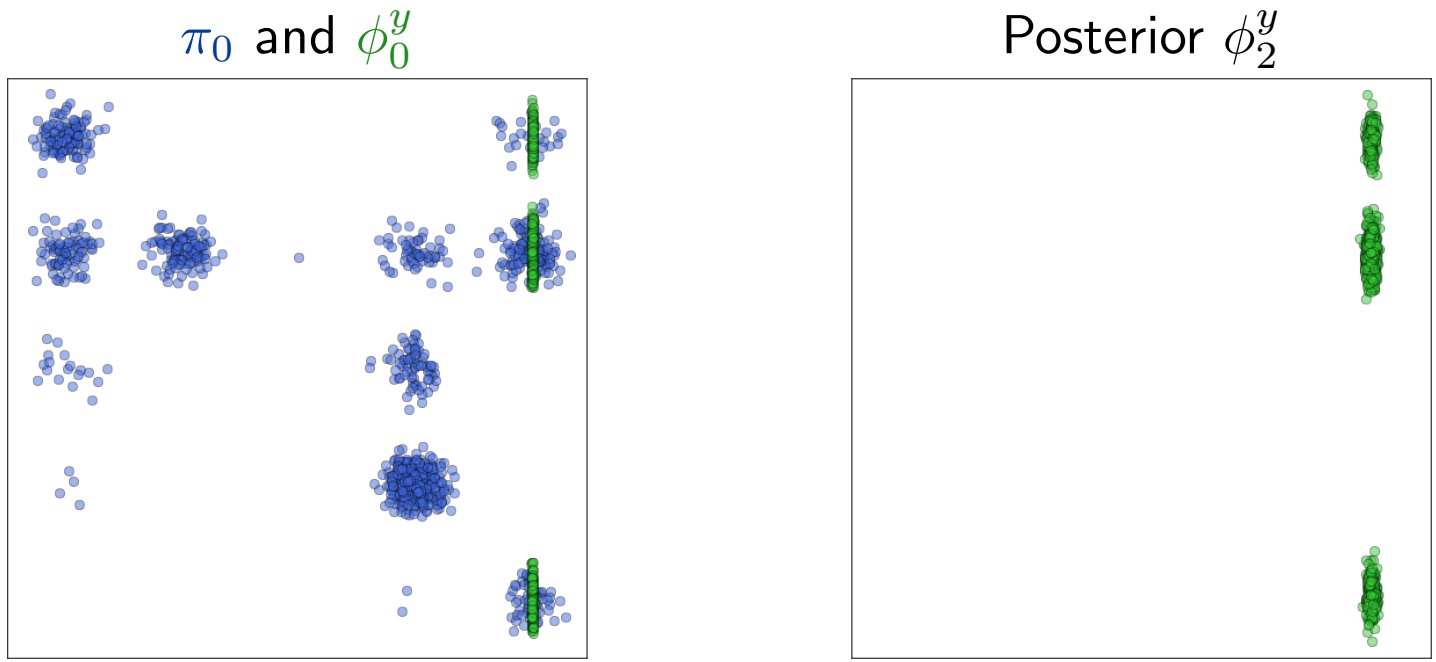


Figure: **Left plot:** samples from the prior π_0 and posterior ϕ_0^y. **Right plot:** samples from the posterior proposals ϕ_t^y for time steps ranging from $n := 500$ to 0.

Posterior sampling proposal

$\rightsquigarrow \{\phi_t^y\}_{t=1}^n$ is available in closed form for the Gaussian mixture example.

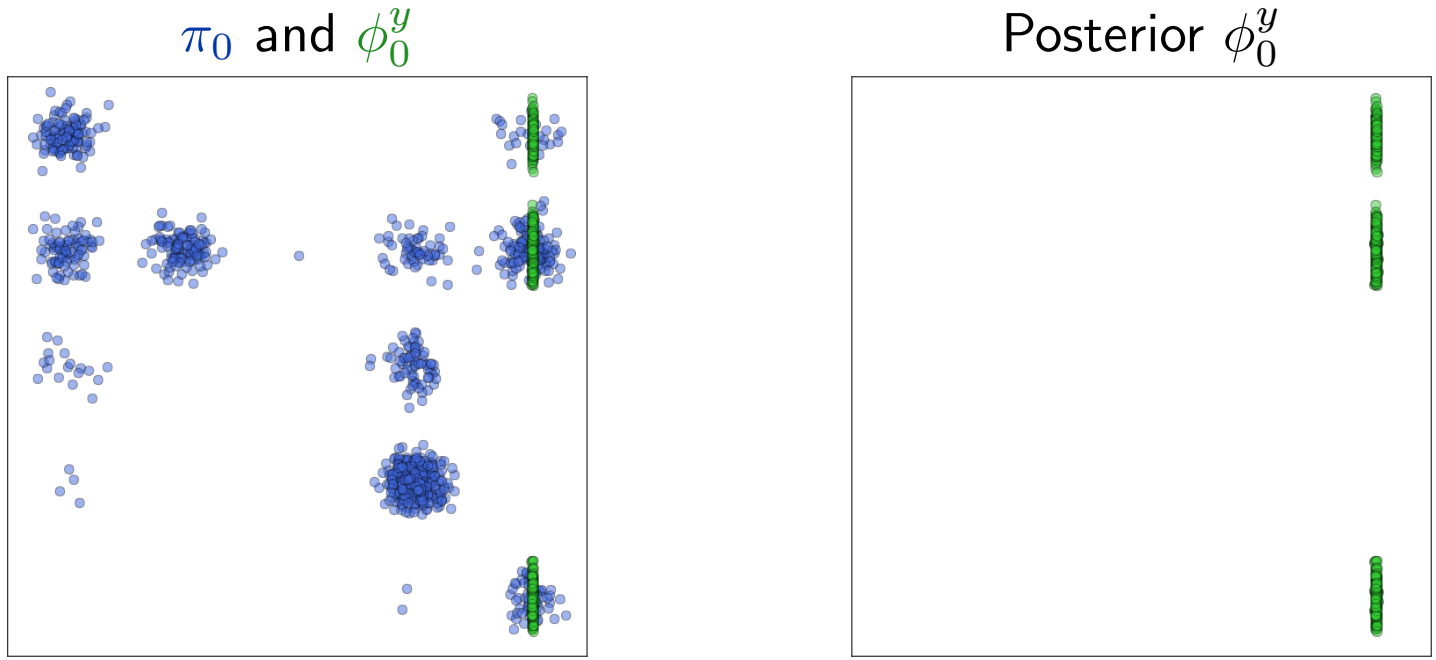


Figure: **Left plot:** samples from the prior π_0 and posterior ϕ_0^y. **Right plot:** samples from the posterior proposals ϕ_t^y for time steps ranging from $n := 500$ to 0.

Posterior sampling proposal: SMC approximation

Assume that $\phi_{t+1}^N = \frac{1}{N}\sum_{i=1}^N \delta_{\xi_{t+1}^i}$ is a particle approximation of ϕ_{t+1}^y.

$$\phi_t^y(\mathrm{d}x_t) = \int p_t^y(\mathrm{d}x_t|x_{t+1}) \frac{\widetilde{\omega}_t(x_{t+1})\phi_{t+1}^y(\mathrm{d}x_{t+1})}{\int \widetilde{\omega}_t(z_{t+1})\phi_{t+1}^y(\mathrm{d}z_{t+1})} ,$$

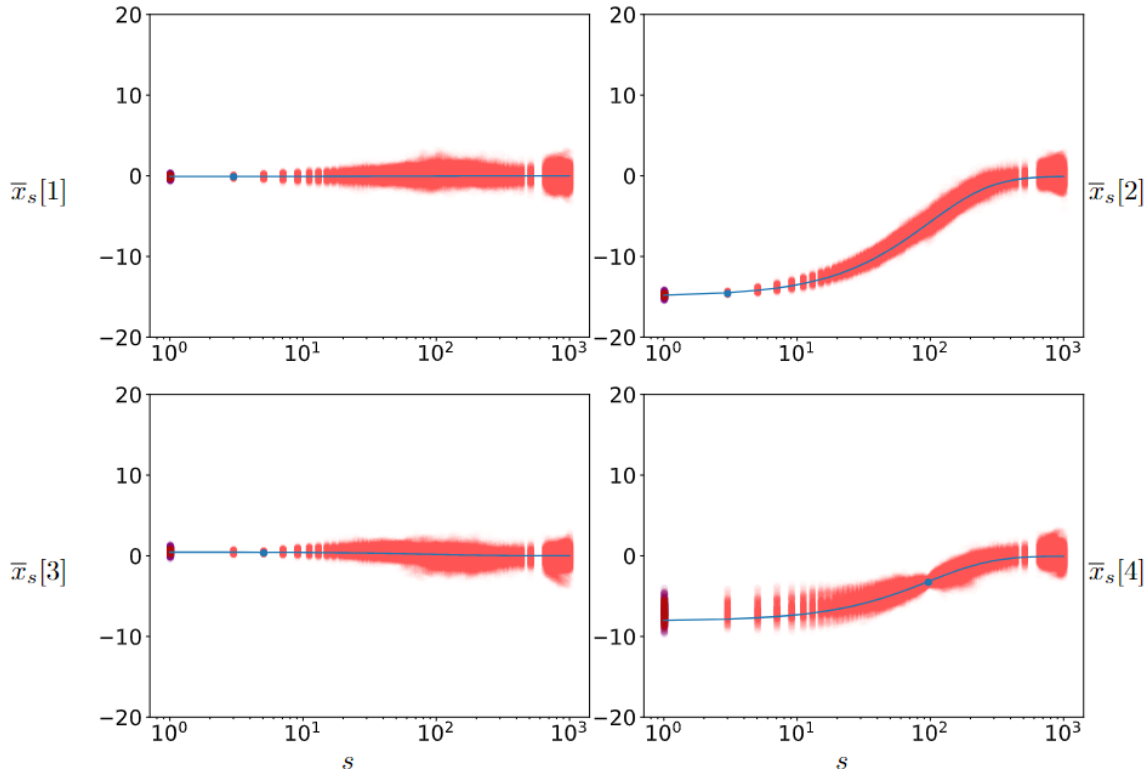
$\rightsquigarrow$ **Weight:**

$$\phi_t^y(\mathrm{d}x_t) \approx \sum_{i=1}^N \frac{\widetilde{\omega}_t(\xi_{t+1}^i)}{\sum_{j=1}^N \widetilde{\omega}_t(\xi_{t+1}^j)} p_t^y(\mathrm{d}x_t|\xi_{t+1}^i) .$$

$\rightsquigarrow$ **Resample:** Draw $A_{t+1}^{1:N} \overset{\text{iid}}{\sim} \mathrm{Categorical}(\{\omega_t^j\}_{j=1}^N)$ where $\omega_t^j \propto \widetilde{\omega}_t(\xi_{t+1}^i)$.

$\rightsquigarrow$ **Mutate:** Sample $\xi_t^i \sim p_t^y(\cdot|\xi_{t+1}^{A_{t+1}^i})$ for $i \in [1:N]$.

$$\phi_t^N(\mathrm{d}x_t) = \frac{1}{N}\sum_{i=1}^N \delta_{\xi_t^i}(\mathrm{d}x_t) .$$



Particle cloud of the 4 first observed coordinates in the case $(d_x, d_y) = (4, 800)$.

Red points: particle cloud, Purple points at the origin: true posterior samples.

The blue curve is the mean of the artificial taget at each time.

Posterior sampling proposal: SMC approximation

The final particle approximation is

$$\phi_0^N(\mathrm{d}x_0) = \frac{1}{N}\sum_{i=1}^{N} \delta_y(\mathrm{d}x_0)\delta_{\xi_0^i}(\mathrm{d}x_0)\,.$$

$\rightsquigarrow$ By **standard SMC convergence results** and under the assumption that $\sup_{x_{t+1}\in\mathbb{R}^{d_x}} \widetilde{\omega}_t(x_{t+1}) < \infty$ for all $t \in [0:n-1]$,

$$\left\|\phi_0^y - \phi_0^N\right\|_2^2 \lesssim \frac{1}{N}\,.$$

$\rightsquigarrow$ **Standard inequalities for SMC filters**. There exist constants $c_{1,n}, c_{2,n} \in (0,\infty)$ such that, for all $N \in \mathbb{N}$, $\varepsilon > 0$ and bounded function $h : \mathbb{R}^{d_x} \mapsto \mathbb{R}$,

$$\mathbb{P}\left[\left|\phi_0^N(h) - \phi_0^{\mathbf{y}}(h)\right| \geq \varepsilon\right] \leq c_{1,n}\exp(-c_{2,n}N\varepsilon^2/|h|_\infty^2)\,.$$

Posterior sampling proposal: SMC approximation

The final particle approximation is

$$\phi_0^N(\mathrm{d}x_0) = \frac{1}{N}\sum_{i=1}^{N} \delta_y(\mathrm{d}x_0)\delta_{\xi_0^i}(\mathrm{d}x_0)\,.$$

There exist C_n^y and D_n^y such that

$$\mathsf{KL}(\phi_0^y \,\|\, \mathbb{E}\big[\phi_0^N\big]) \leq \frac{C_n^y}{N} + \frac{D_n^y}{N^2}\,.$$

Results **non explicit in n without additional assumptions**.

Synthetic example

25 component Gaussian mixture

$\rightsquigarrow$ Gaussian random variables with mean $\boldsymbol{\mu}_{i,j} := (8i, 8j, \cdots, 8i, 8j) \in \mathbb{R}^{d_x}$ for $(i, j) \in \{-2, -1, 0, 1, 2\}^2$ and unit variance.

20 component Funnel mixture

$\rightsquigarrow$ A funnel distribution is defined by the following density

$$\mathcal{N}(x_1; 0, 1) \prod_{i=1}^{d} \mathcal{N}(x_i; 0, \exp(x_1/2)) \,.$$

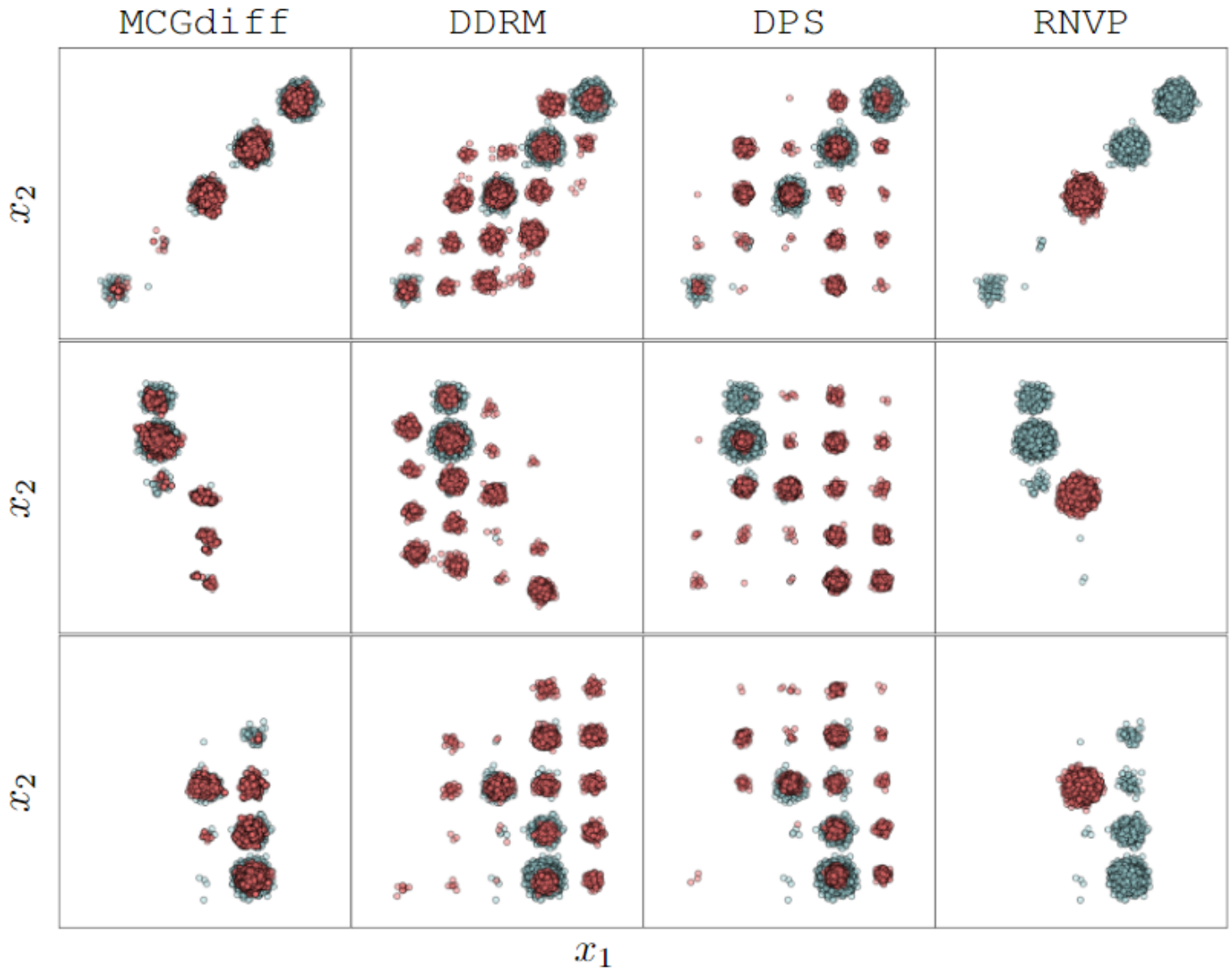
$\rightsquigarrow$ Sampling $(\mu_i, R_i)_{i=1}^{20}$ uniformly in $([-20, 20]^d \times \mathrm{SO}(R^d))^{\times 20}$. The mixture consists of 20 Funnel random variables translated by μ_i and rotated by R_i.

Observation model

For a pair of dimensions (d_x, d_y) the measurement model is $Y = AX + \sigma\varepsilon$ where A and σ are drawn randomly.

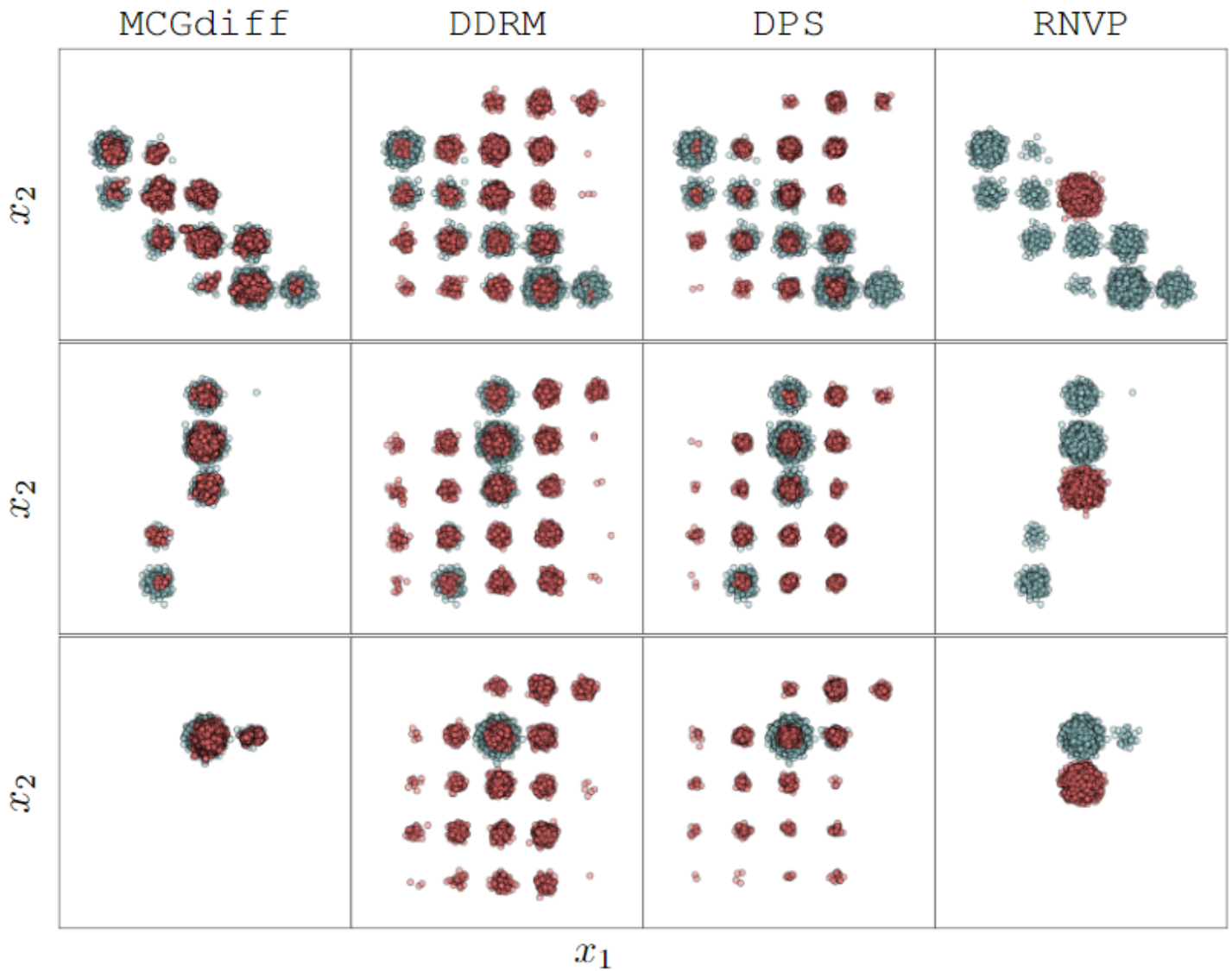
Synthetic example

First two dimensions for the GMM case with $d_x = 8$. The rows represent $d_y = 1, 2, 4$.



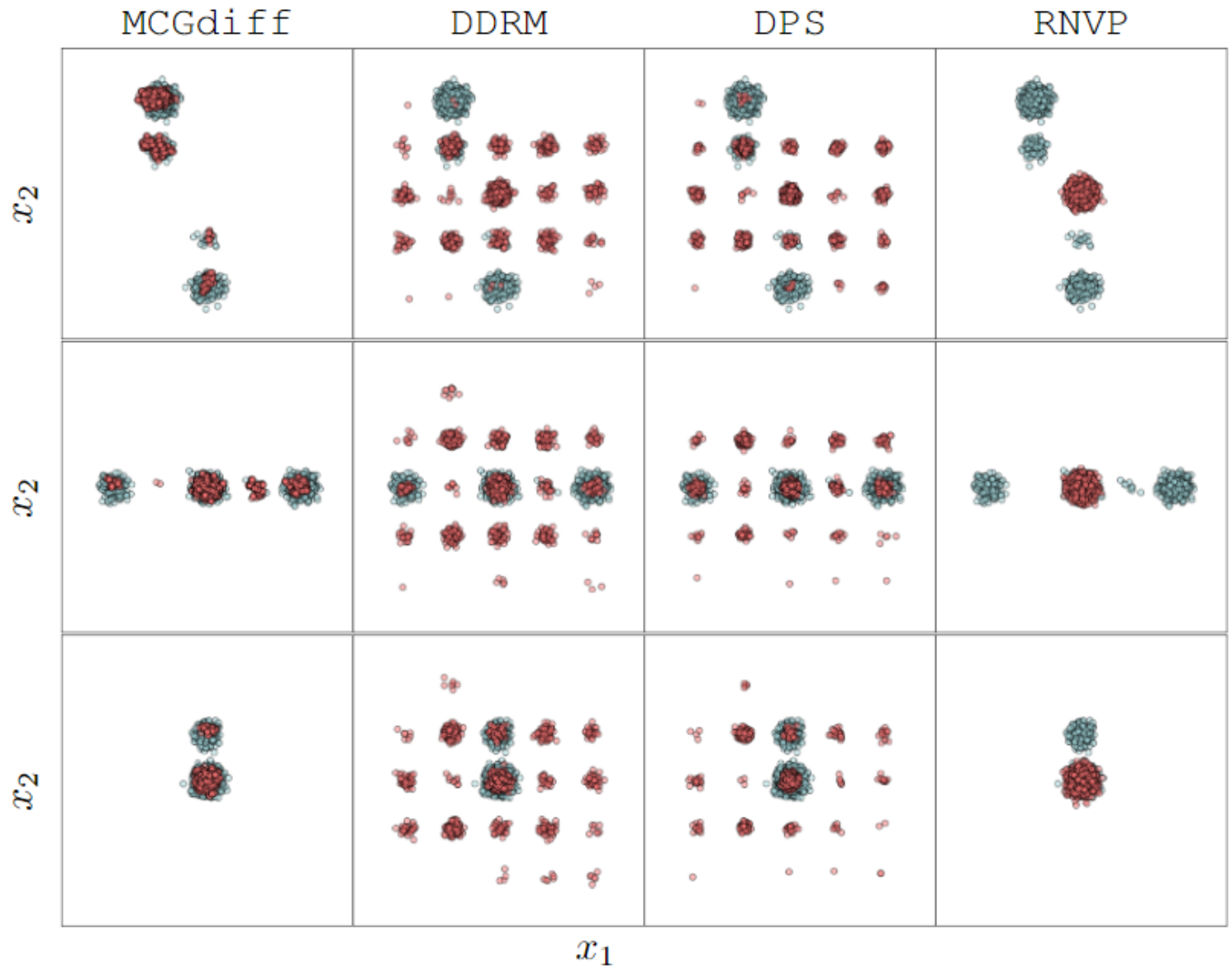
Synthetic example

First two dimensions for the GMM case with $d_x = 80$. The rows represent $d_y = 1, 2, 4$.



Synthetic example

First two dimensions for the GMM case with $d_x = 800$. The rows represent $d_y = 1, 2, 4$.



Synthetic example

(1) 25 component Gaussian mixture (2) 20 component Funnel mixture. $A \in \mathbb{R}^{d_y \times d}$ and σ are drawn randomly.

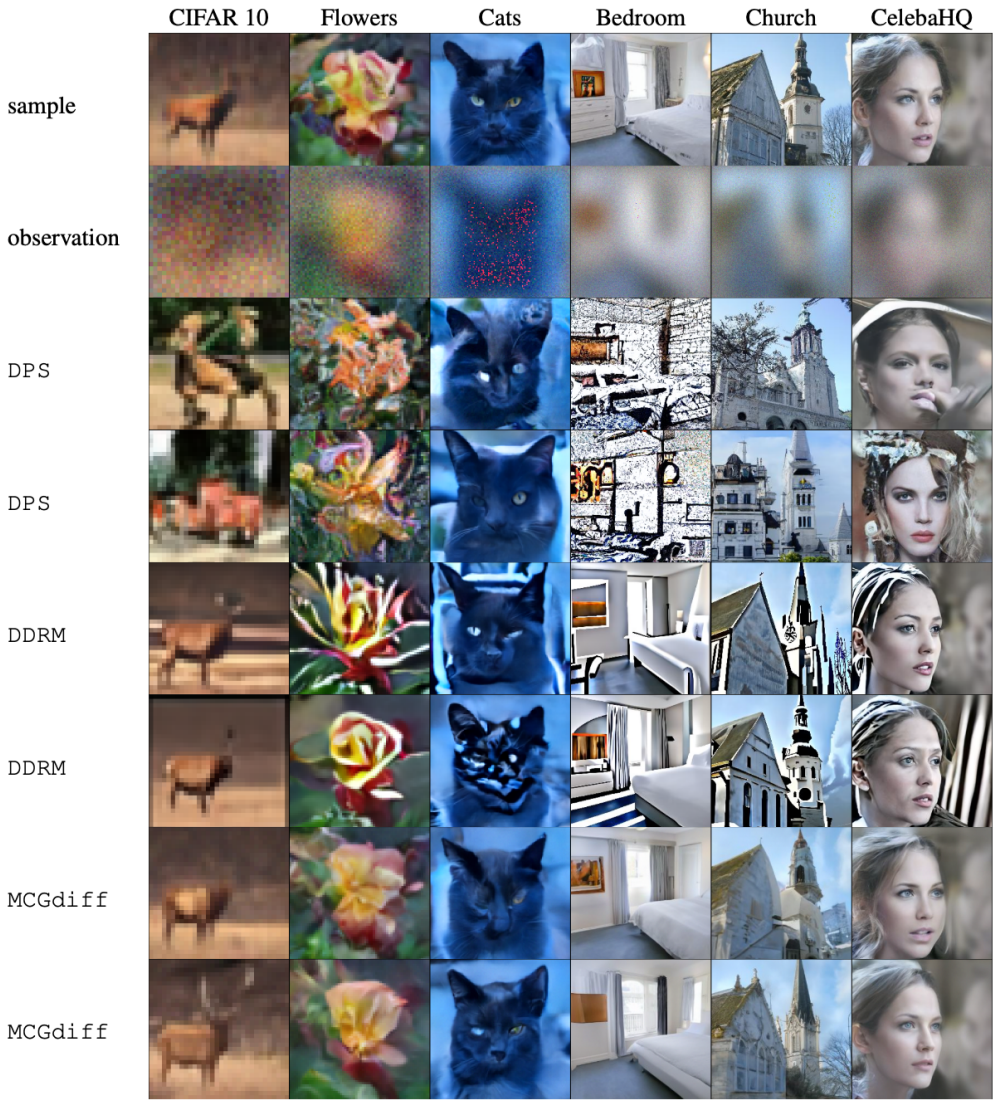
d	d_y	MCGdiff	DDRM	DPS	RNVP
80	1	**1.39 ± 0.45**	5.64 ± 1.10	4.98 ± 1.14	6.86 ± 0.88
80	2	**0.67 ± 0.24**	7.07 ± 1.35	5.10 ± 1.23	7.79 ± 1.50
80	4	**0.28 ± 0.14**	7.81 ± 1.48	4.28 ± 1.26	7.95 ± 1.61
800	1	**2.40 ± 1.00**	7.44 ± 1.15	6.49 ± 1.16	7.74 ± 1.34
800	2	**1.31 ± 0.60**	8.95 ± 1.12	6.88 ± 1.01	8.75 ± 1.02
800	4	**0.47 ± 0.19**	8.39 ± 1.48	5.51 ± 1.18	7.81 ± 1.63

d	d_y	MCGdiff	DDRM	DPS	RNVP
6	1	**1.95 ± 0.43**	4.20 ± 0.78	5.43 ± 1.05	6.16 ± 0.65
6	3	**0.73 ± 0.33**	2.20 ± 0.67	3.47 ± 0.78	4.70 ± 0.90
6	5	**0.41 ± 0.12**	0.91 ± 0.43	2.07 ± 0.63	3.52 ± 0.93
10	1	**2.45 ± 0.42**	3.82 ± 0.64	4.30 ± 0.91	6.04 ± 0.38
10	3	**1.07 ± 0.26**	4.94 ± 0.87	5.38 ± 0.84	5.91 ± 0.64
10	5	**0.71 ± 0.12**	2.32 ± 0.74	3.74 ± 0.77	5.11 ± 0.69

Figure: Sliced Wasserstein between samples of the target posterior and the empirical measure returned by each method. **Top**: Gaussian mixture. **Bottom**: Funnel mixture. We show the 95% CLT interval over 20 seeds.

Super-resolution example

Deblurring example



Many problems ahead

$\rightsquigarrow$ Not so many solutions proposed to adapt such methods to data from ecosystems abundancy, population dynamics, time series data.

$\rightsquigarrow$ Generic methods that can be used so sample form distribution or conditional distribution but choice of the proposal distribution (MCMC), variational family (VAE), score network (SBGM) is crucial.

$\rightsquigarrow$ Theoretical results under realistic assumptions.