Score - Based Generative Modelling

Manos Plitsis

What is Generative Modelling?

Given a dataset $\{x_i \in \mathbb{R}^D\}_{i=1}^N$, model the *data distribution* $p_{data}(x)$.

Once we have $p(x)$ we can generate new data points by sampling from it.

Energy-Based Modelling

To model an arbitrarily flexible distribution $p(x)$, we can model it as:

$$p_\theta(x) = \frac{e^{-f_\theta(x)}}{Z_\theta}$$

e.g. with maximum likelihood.

$f_\theta(x)$ is called the energy function
$Z_\theta = \int e^{-f_\theta(x)}$ is a normalizing constant

Energy-Based Modelling

To model an arbitrarily flexible distribution $p(x)$, we can model it as:

$$p_\theta(x) = \frac{e^{-f_\theta(x)}}{Z_\theta}$$

e.g. with maximum likelihood.

But Z is not tractable!

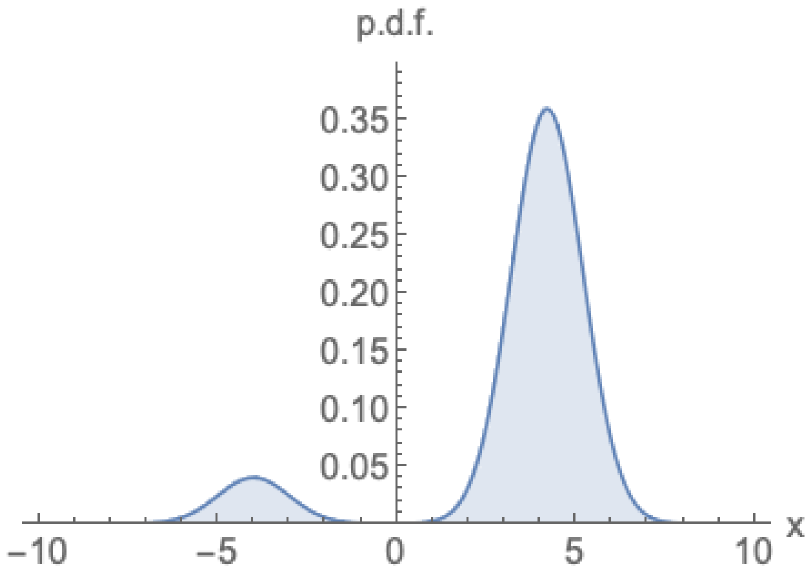
What if we could get rid of Z?

$$
\begin{aligned}
\nabla_x log p_\theta(x) &= \nabla_x log(\frac{e^{-f_\theta(x)}}{Z_\theta}) \\
&= \nabla_x log(\frac{1}{Z_\theta}) + \nabla_x log(e^{-f_\theta(x)}) \\
&= -\nabla_x f_\theta(x) \\
&\approx s_\theta(x)
\end{aligned}
$$

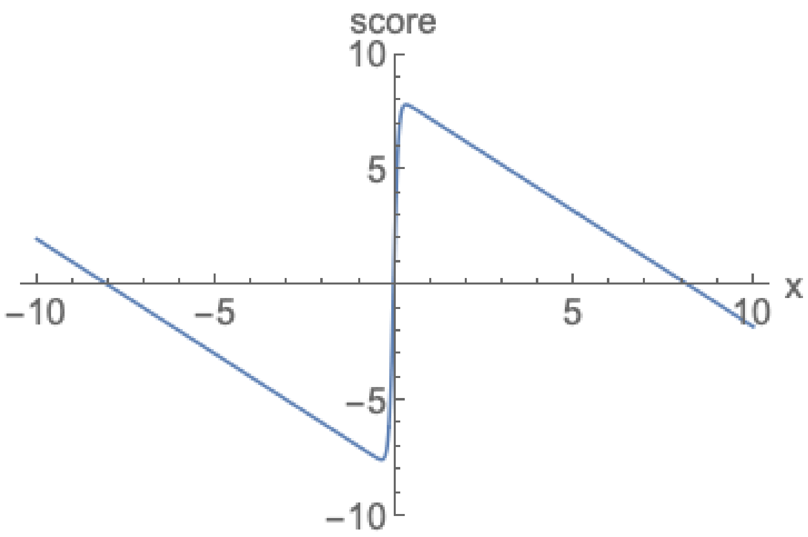
Score Networks

Def. The *(Stein) score* of a probability density $p(x)$ is $\nabla_x logp(x)$

A *score network* $s_\theta(x) : \mathbb{R}^D \to \mathbb{R}^D$ is a neural network trained to approximate the score of $p_{data}(x)$



p.d.f.
0.35
0.30
0.25
0.20
0.15
0.10
0.05
−10
−5
0
5
10
x



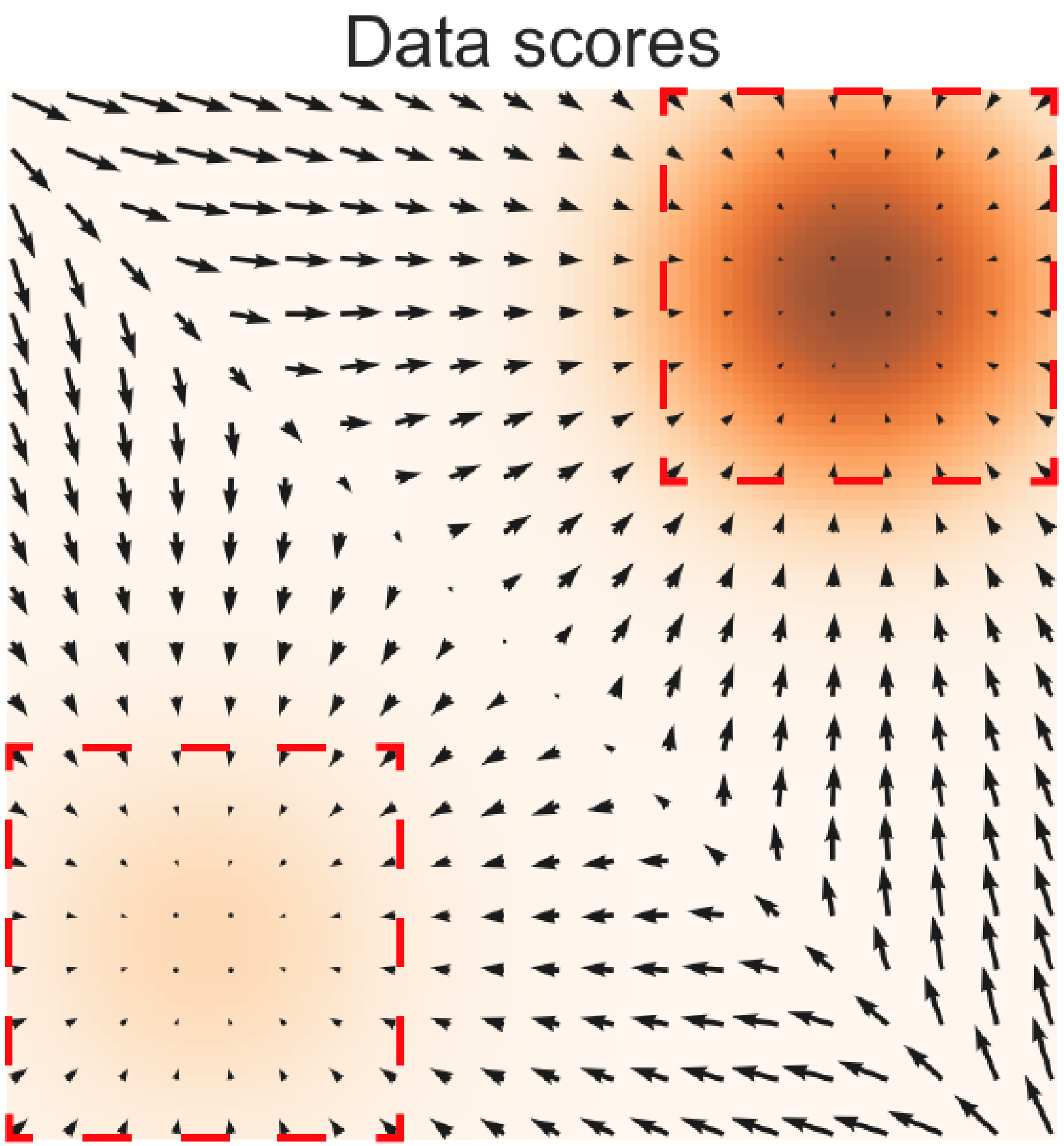
score
10
5
−5
−10
−10
−5
5
10
x

Score Networks

Def. The *(Stein) score* of a probability density $p(x)$ is $s(x) = \nabla_x \log p(x)$

A *score network* $s_\theta(x) : \mathbb{R}^D \to \mathbb{R}^D$ is a neural network trained to approximate the score of $p_{data}(x)$

Q1: How do we learn $s_\theta(x)$?
Q2: How do we use it to generate new samples?



Score Matching

Q1. How do we learn $s_\theta(x)$?

A. We optimize the score matching objective:

$$L(\theta) \triangleq \frac{1}{2}\mathbb{E}_{p_{data}}[\|s_\theta(x) - \nabla_x \log p_{data}(x)\|_2^2]$$

Score Matching

Q1: How do we learn $s_\theta(x)$?

A: We optimize the score matching objective:

$$L(\theta) \triangleq \frac{1}{2}\mathbb{E}_{p_{data}}[\|s_\theta(x) - \nabla_x \log p_{data}(x)\|_2^2]$$

But we don't have access to $\nabla_x \log p_{data}(x)$!

L can be shown equivalent (up to a constant) to:

$$\mathbb{E}_{p_{data}(x)}[tr(\nabla_x s_\theta(x)) + \frac{1}{2}\|s_\theta(x)\|_2^2]$$

We can compute this from data, but calculating $tr(\nabla_x s_\theta(x))$ is too costly.

Score Matching

Denoising Score Matching

We perturb the data with a noise distribution $q_\sigma(\tilde{x}|x)$, then estimate the score of $q_\sigma \triangleq \int q_\sigma(\tilde{x}|x)p_{data}(x)dx$ with objective:

$$\frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{x}|x)p_{\text{data}}(x)}[||s_\theta(\tilde{x}) - \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x)||_2^2]$$

E.g. for Gaussian $\nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x) = -\frac{\tilde{x}-x}{\sigma^2}$

Then $s_{\theta*}(x) = \nabla_x \log q_\sigma(x)$ almost surely.

But: $s_{\theta*}(x) = \nabla_x \log q_\sigma(x) \approx \nabla_x \log p_{data}(x)$ only when σ is small s.t. $q_\sigma(x) \approx p_{data}(x)$ (But not too small and the variance explodes)

Langevin Dynamics

Q2: How do we use $s_\theta(x)$ to generate new samples?

A: We recursively compute:

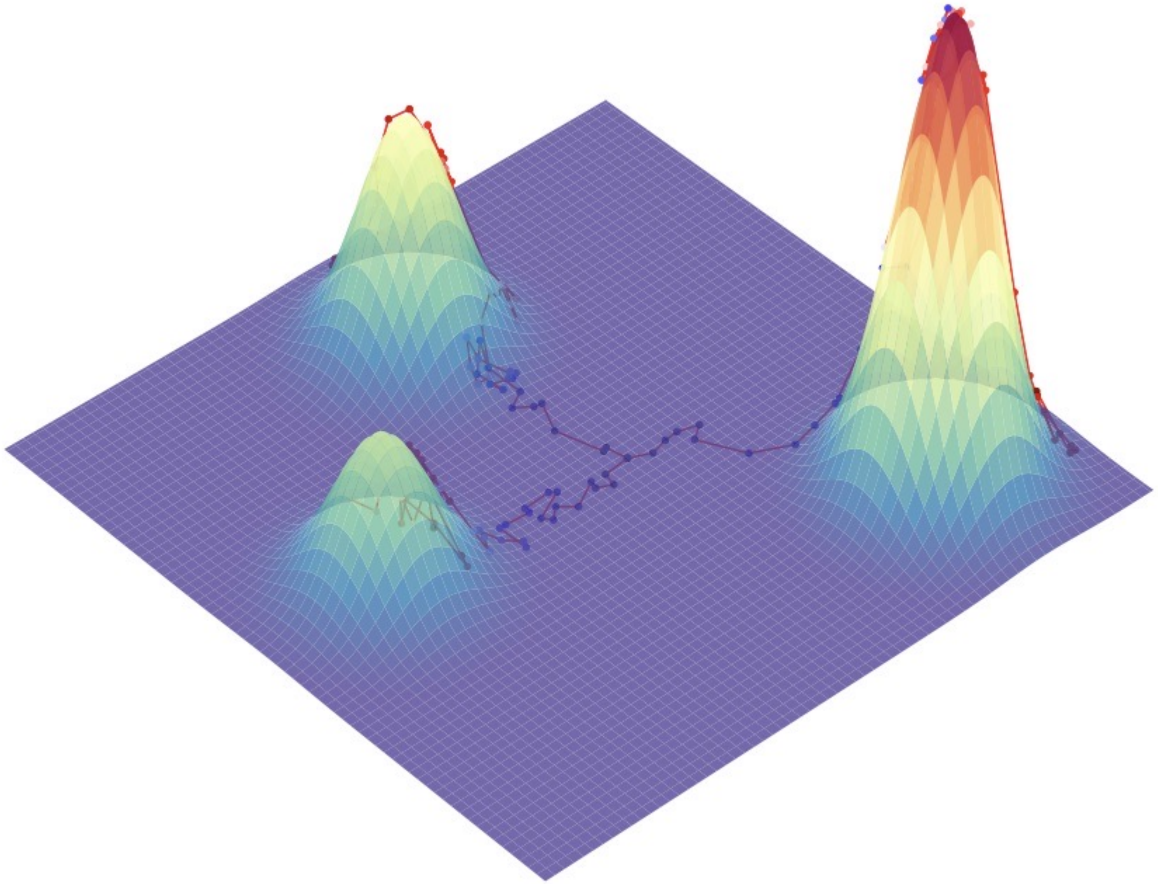
$$\tilde{x}_t = \tilde{x}_{t-1} + \frac{\epsilon}{2} \nabla_x \log p(\tilde{x}_{t-1}) + \sqrt{\epsilon} z_t$$

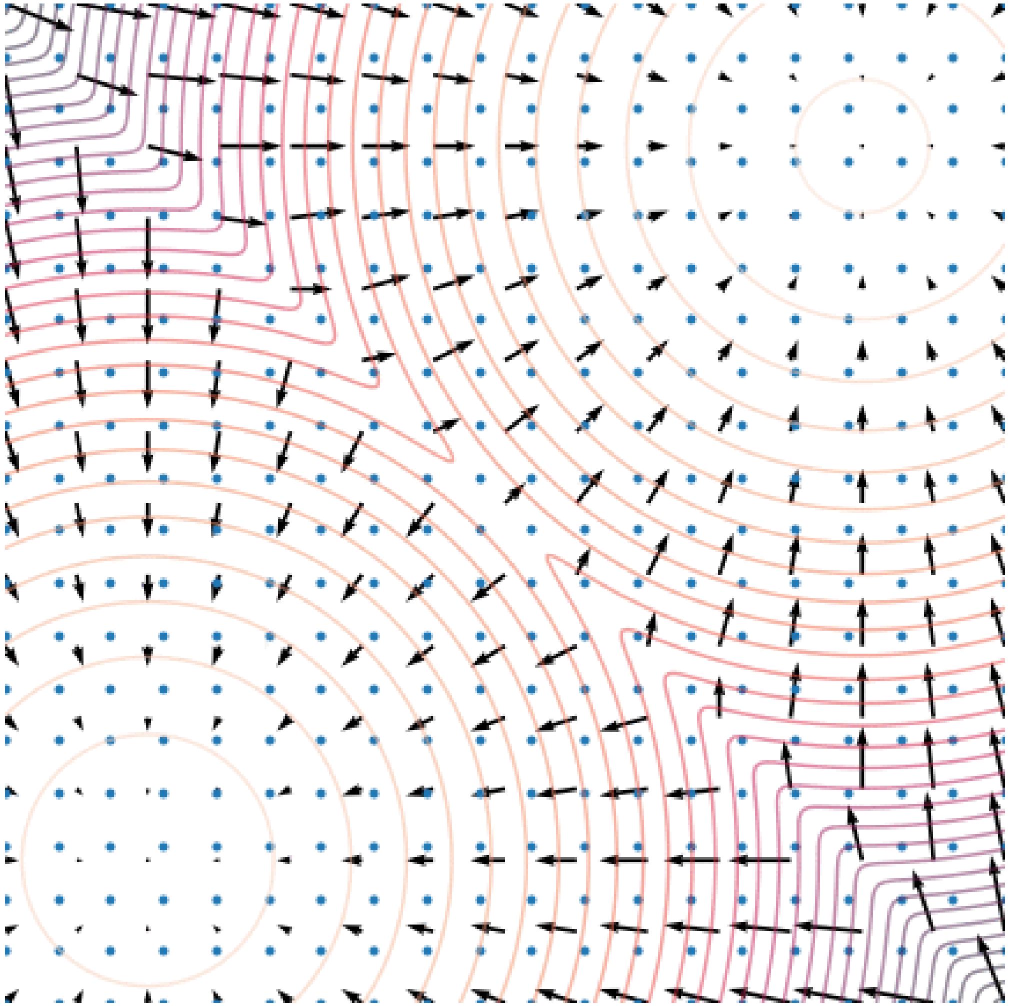
where

- $\epsilon > 0$ a fixed step size
- $\tilde{x}_0 \sim \pi(x)$ with π a prior distribution
- $z_t \sim \mathcal{N}(0, I)$

The distribution of $\tilde{x}_T$ approaches $p(x)$ when $\epsilon \to 0, T \to \infty$

Langevin Dynamics





Challenges of Score-Based Modelling

1. Data are concentrated on a low dimensional manifold
2. Scarcity of data leads to low density regions

1. The manifold hypothesis

Real world data tend to concentrate on **low dimensional manifolds** embedded in the ambient space.

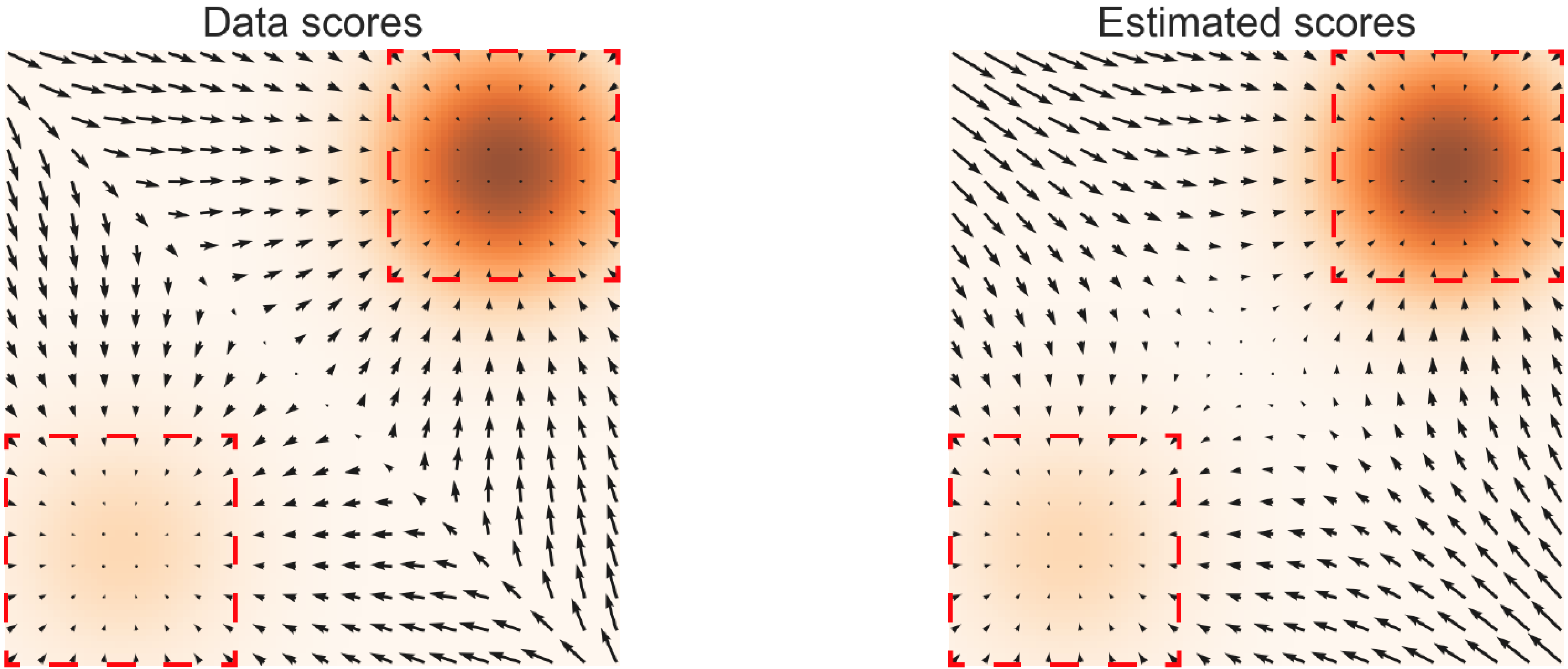
- Score is undefined outside the manifold.
- Score matching only works when the support of $p(x)$ is the ambient space [Hyvärinen 2005]

2. Scarcity of data

Inaccurate score estimation

In practice, $\mathbb{E}_{p_{data}}$ is estimated using i.i.d. samples.

For regions $\mathcal{R} \subset \mathbb{R}^D$ s.t $p_{data}(\mathcal{R}) \approx 0$ there usually isn't enough data to estimate the score.



Data scores
Estimated scores

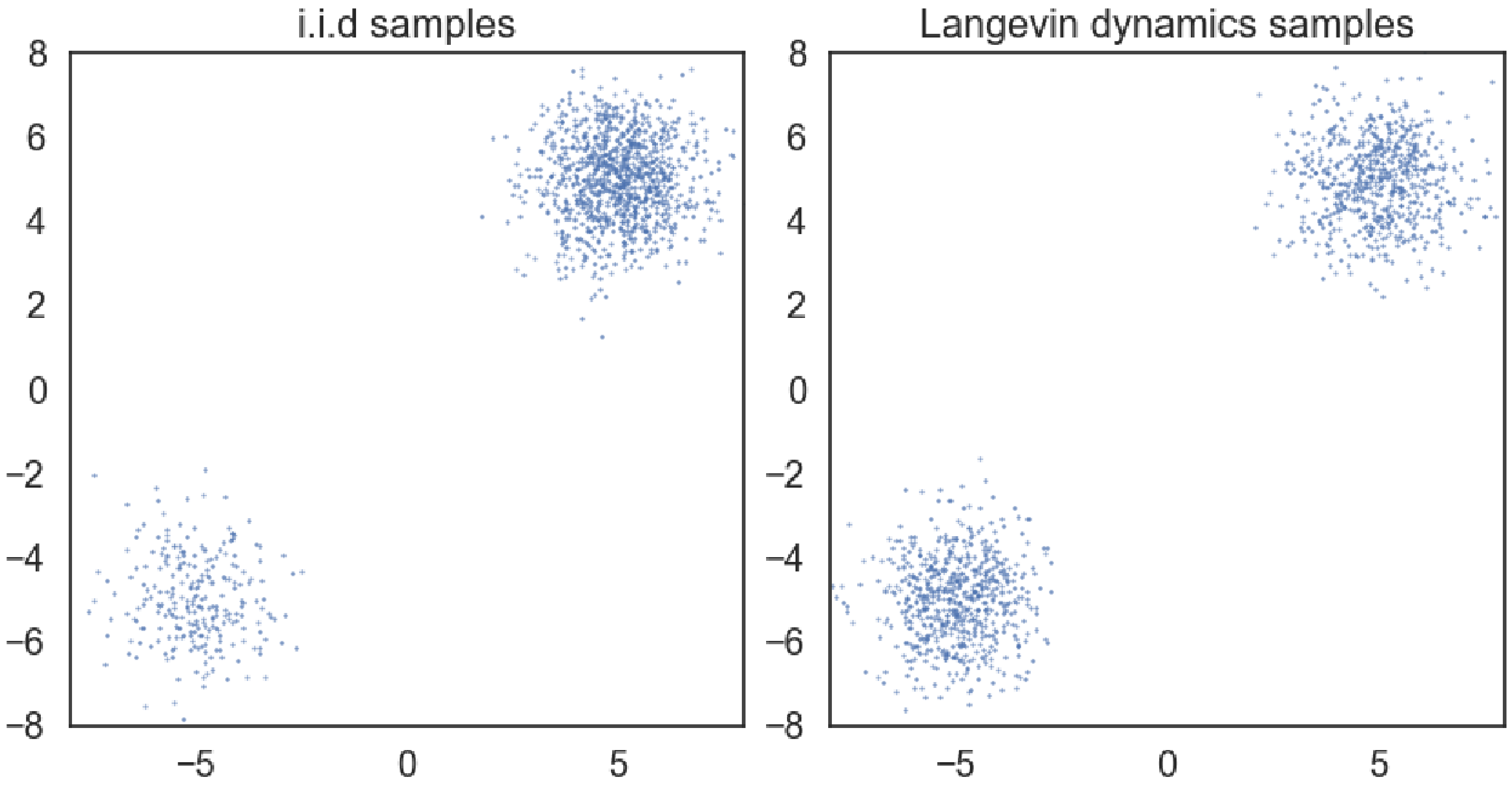
2. Scarcity of data

Slow mixing of Langevin Dynamics

Let $p_1(x), p_2(x)$ normalized distributions with disjoint support, and $p_{data}(x) = \pi p_1(x) + (1 - \pi)p_2(x)$, $\pi \in (0, 1)$

The score does not depend on π, so samples with LD will not depend on π.

In practice, also true when supports are *approximately disjoint*.



i.i.d samples
Langevin dynamics samples
8
6
4
2
0
−2
−4
−6
−8
−5
0
5

Noise Conditional Score Networks

with Annealed Langevin Dynamics

Idea:

- Perturb data with various noise levels $\sigma_1, \ldots, \sigma_L$ using distribution

$$q_\sigma(x) \triangleq \int p_{data}(t) \mathcal{N}(x|t, \sigma^2 I) dt$$

(this gives $q_\sigma(\tilde{x}|x) = \mathcal{N}(\tilde{x}|x, \sigma^2 I)$, the "noising" distribution)

- Train a single conditional score network $s_\theta(x, \sigma)$ for all noise levels σ_i to predict the noised data score $\nabla x \log q_{\sigma_i}(x)$.
- In LD, start with high noise level scores, and gradually decrease the noise level.

Noise Conditional Score Networks

- $\frac{\sigma_1}{\sigma_2} = \ldots = \frac{\sigma_{L-1}}{\sigma_L} > 1$
- σ_1 is large enough to avoid previous problems.
- σ_L is small enough to minimize effect on data.

Noise Conditional Score Networks

We defined $q_\sigma(\tilde{x}|x) = \mathcal{N}(\tilde{x}|x, \sigma^2 I)$,
so $\nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x) = -\frac{\tilde{x}-x}{\sigma^2}$.

For a given σ:

$$\ell(\theta; \sigma) \triangleq \frac{1}{2} \mathbb{E}_{p_{\text{data}}(x)} \mathbb{E}_{\tilde{x} \sim \mathcal{N}(x, \sigma^2 I)} \left[\left\| s_\theta(\tilde{x}, \sigma) + \frac{\tilde{x} - x}{\sigma^2} \right\|_2^2 \right]$$

So the final loss is:

$$\mathcal{L}(\theta; \{\sigma_i\}_{i=1}^L) \triangleq \frac{1}{L} \sum_{i=1}^{L} \lambda(\sigma_i) \ell(\theta; \sigma_i)$$

Noise Conditional Score Networks

How to choose $\lambda(\sigma)$?

- We want $\lambda(\sigma_i)\ell(\theta;\sigma_i)$ to have the same order of magnitude for all $\{\sigma_i\}_{i=1}^{L}$.
- With optimally trained networks, we observe $\|s_\theta(x,\sigma)\|_2 \propto \frac{1}{\sigma}$.
- This inspires the choice $\lambda(\sigma) = \sigma^2$, leading to $\lambda(\sigma)\ell(\theta;\sigma) = \sigma^2\ell(\theta;\sigma) = \frac{1}{2}\|\sigma s_\theta(x,\sigma) + \frac{\hat{x}-x}{\sigma}\|_2^2$.
- Given $\frac{\hat{x}-x}{\sigma} \sim \mathcal{N}(0,I)$ and $\|\sigma s_\theta(x,\sigma)\|_2 \propto 1$, the magnitude of $\lambda(\sigma)\ell(\theta;\sigma)$ does not depend on σ.

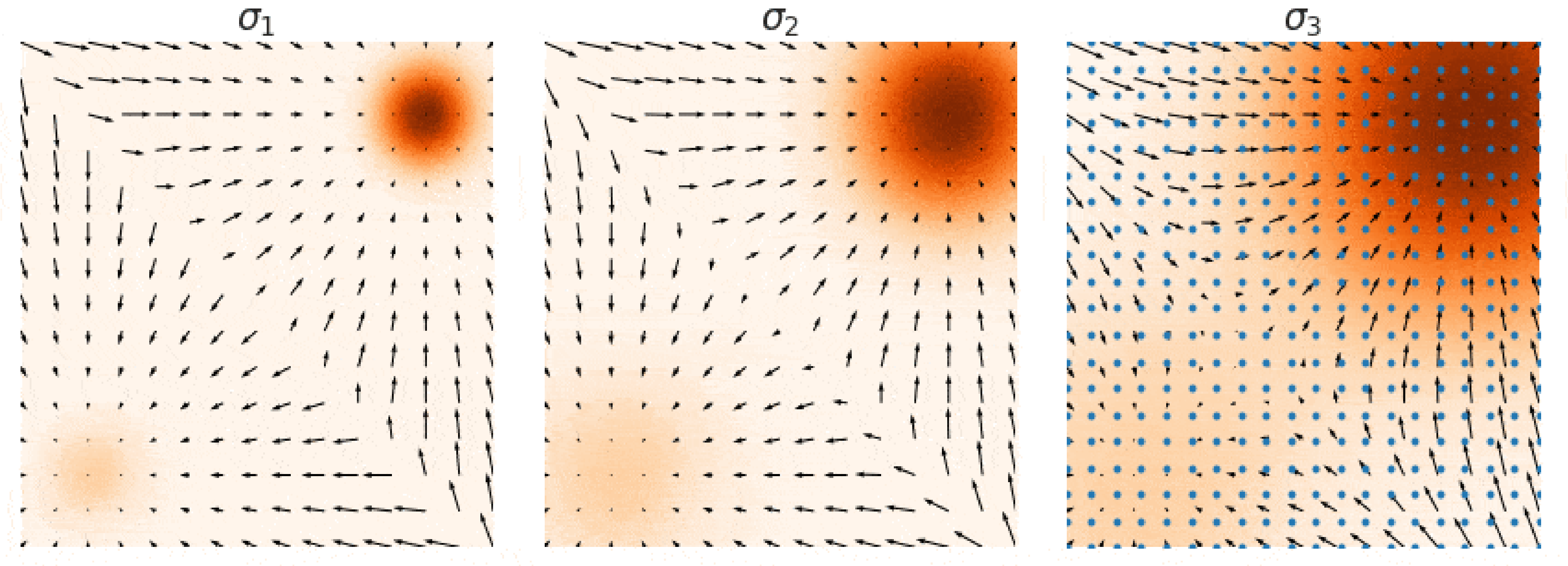
Annealed Langevin Dynamics

Algorithm 1: Annealed Langevin dynamics

Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T.$

Initialize $\tilde{x}_0$
for $i = 1$ **to** L **do**
    $\alpha_i \leftarrow \epsilon \sigma_i^2 / \sigma_L^2$ (α_i is the step size)
    for $t = 1$ **to** T **do**
        Draw $z_t \sim \mathcal{N}(0, I)$
        $\tilde{x}_t \leftarrow \tilde{x}_{t-1} + \frac{\alpha_i}{2} s_\theta(\tilde{x}_{t-1}, \sigma_i) + \sqrt{\alpha_i} z_t$
    end for
    $\tilde{x}_0 \leftarrow \tilde{x}_T$
end for
return $\tilde{x}_T$

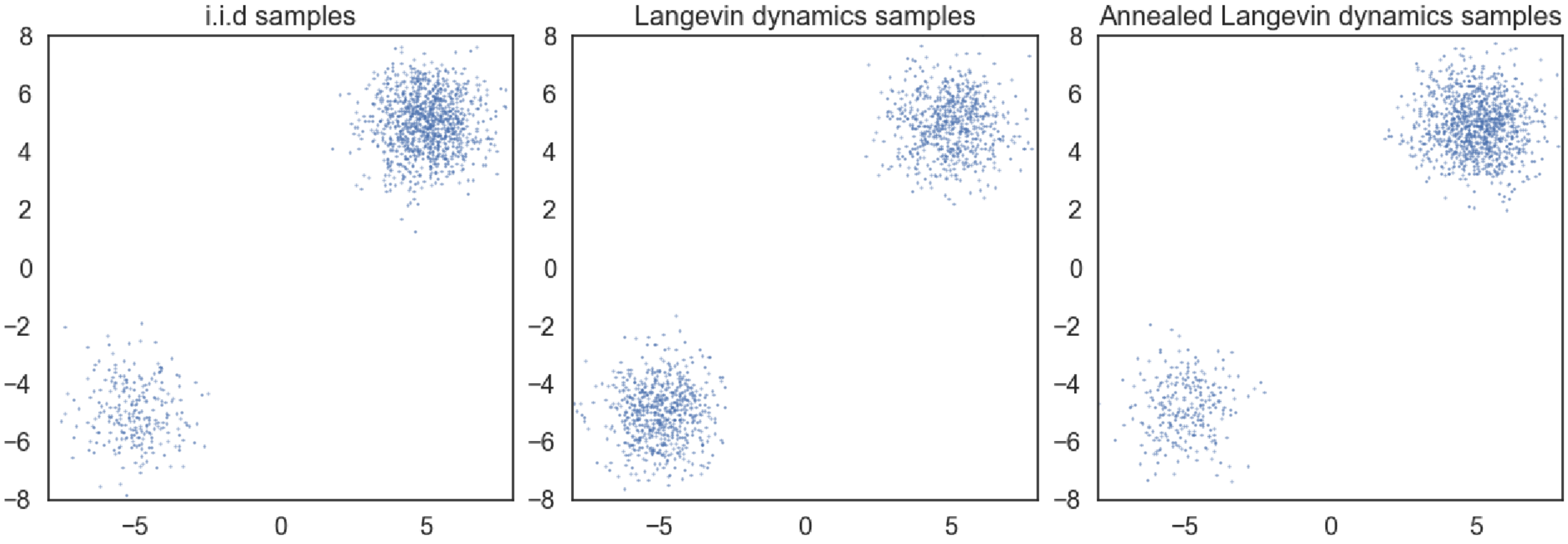
Annealed Langevin Dynamics



Annealed Langevin Dynamics

How to choose α_i?

- Many ways to tune α_i. We use $\alpha_i \propto \sigma_i^2$.
- Aim: Fix the magnitude of the "signal-to-noise" ratio $\frac{\alpha_i s_\theta(x,\sigma_i)}{2\sqrt{\alpha_i} z}$ in Langevin dynamics w.r.t. σ_i.
- $\mathbb{E}\left[\|\frac{\alpha_i s_\theta(x,\sigma_i)}{2\sqrt{\alpha_i} z}\|_2^2\right] \approx \mathbb{E}\left[\frac{\alpha_i \|s_\theta(x,\sigma_i)\|_2^2}{4}\right] \propto \frac{1}{4}\mathbb{E}\left[\|\sigma_i s_\theta(x,\sigma_i)\|_2^2\right]$.
- Empirically, when networks are optimally trained, $\|s_\theta(x,\sigma_i)\|_2 \propto \frac{1}{\sigma_i}$.
- Thus $\mathbb{E}\left[\|\sigma_i s_\theta(x,\sigma_i)\|_2^2\right] \propto 1$.
- Therefore, the choice of α_i does not depend on σ_i.


i.i.d samples
Langevin dynamics samples
Annealed Langevin dynamics samples
8
6
4
2
0
−2
−4
−6
−8
−5
0
5

Experimental Setup

- $L = 10$
- $\{\sigma_i\}_{i=1}^{L}$ is a geometric sequence with:
  - $\sigma_1 = \sigma_{max} = 1$
  - $\sigma_{10} = \sigma_{min} = 0.01$
- For LD sampling:
  - $T = 10$
  - $\epsilon = 2 \times 10^-5$
  - Initial samples are uniform noise.

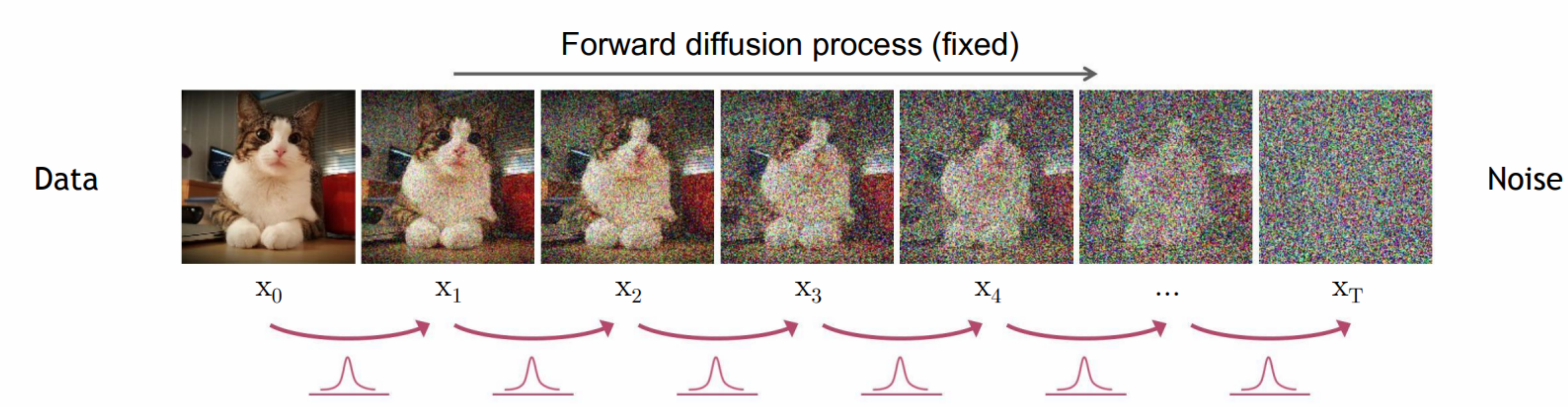
Results

Model	Inception	FID
CIFAR-10 Unconditional		
PixelCNN [59]	4.60	65.93
PixelIQN [42]	5.29	49.46
EBM [12]	6.02	40.58
WGAN-GP [18]	$7.86 \pm .07$	36.4
MoLM [45]	$7.90 \pm .10$	**18.9**
SNGAN [36]	$8.22 \pm .05$	21.7
ProgressiveGAN [25]	$8.80 \pm .05$	-
NCSN (Ours)	$\mathbf{8.87} \pm .12$	25.32
CIFAR-10 Conditional		
EBM [12]	8.30	37.9
SNGAN [36]	$8.60 \pm .08$	25.5
BigGAN [6]	**9.22**	**14.73**

NCSM vs DDPM

Reminder: DDPM

- $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t} x_{t-1}, (1-\alpha_t)I)$ (noising step)
- $p_\theta(x_T) = \mathcal{N}(x_T; 0, I)$.
- $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma(x_t, t)^2 I)$ (denoising step)



ELBO for DDPM

$$
\begin{aligned}
\log p(x) \geq\ & \mathbb{E}_{q(x_1|x_0)}\left[\log p_\theta(x_0|x_1)\right] && (L_0 :\ \text{Reconstruction term}) \\
& - D_{KL}(q(x_T|x_0)||p(x_T)) && (L_T : \text{Prior matchng term}) \\
& - \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)}\left[D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))\right] && (L_{t-1} : \text{Denoising matching term})
\end{aligned}
$$

To **maximize** the ELBO, we need to minimize the denoising matching term.

We can write $x_t = \sqrt{\bar{a}_t} x_0 + \sqrt{1 - \bar{a}_t} \epsilon_0 \sim \mathcal{N}(x_t; \sqrt{\bar{a}_t} x_t, (1 - \bar{a}_t) I)$

with $\bar{a}_t = \prod_{i=1}^{t} a_i$

Then:

$$
\begin{aligned}
q(x_{t-1}|x_t, x_0) &= \frac{q(x_t|x_{t-1}, x_0) q(x_{t-1}|x_0)}{q(x_t|x_0)} \qquad \text{(Bayes rule)} \\
&= \frac{\mathcal{N}(x_t; \sqrt{\bar{a}_t} x_t, (1 - \bar{a}_t) I) \ \ \mathcal{N}(x_{t-1}; \sqrt{\bar{a}_{t-1}} x_0, (1 - \bar{a}_{t-1}) I)}{\mathcal{N}(x_t; \sqrt{\bar{a}_t} x_0, (1 - \bar{a}_t) I)} \\
&= \ \ldots \\
&= \mathcal{N}(x_{t-1}; \underbrace{\frac{\sqrt{a_t}(1 - \bar{a}_{t-1}) x_t + \sqrt{\bar{a}_{t-1}}(1 - a_t) x_0}{1 - \bar{a}_t}}_{\mu_q(x_t, x_0)}, \underbrace{\frac{(1 - a_t)(1 - \bar{a}_{t-1})}{(1 - \bar{a}_t)} I)}_{\sum_q(t)}
\end{aligned}
$$

Learning μ_θ

$$\begin{aligned}
&\operatorname{argmin}_\theta D_{KL}(q(x_{t-1}|x_t, x_0) \mid\mid p_\theta(x_{t-1}|x_t)) \\
&= \operatorname{argmin}_\theta D_{KL}(\mathcal{N}(x_{t-1}; \mu_q(t), \Sigma_q(t)) \mid\mid \mathcal{N}(x_{t-1}; \mu_\theta(t), \Sigma_q(t))) && \text{(set denoising transition variance to be } \mathit{\Sigma}_q(t)) \\
&= \quad \ldots && \text{(KL Divergence Gaussians)} \\
&= \operatorname{argmin}_\theta \frac{1}{2\sigma_q^2(t)} \left[||\mu_\theta - \mu_q||_2^2 \right]
\end{aligned}$$

Learning ϵ_θ

We can choose the parameterization: $x_0 = \frac{x_t + \sqrt{1-\bar{a}_t}\epsilon_0}{\sqrt{\bar{a}_t}}$

- $\mu_q(x_t, x_0) = \frac{1}{\sqrt{a_t}} x_t - \frac{1-a_t}{\sqrt{1-\bar{a}_t}\sqrt{a_t}} \epsilon_0$
- $\mu_\theta(x_t, t) = \frac{1}{\sqrt{a_t}} x_t - \frac{1-a_t}{\sqrt{1-\bar{a}_t}\sqrt{a_t}} \epsilon_\theta(x_t, t)$

Reformulate the loss to:

$$\mathrm{argmin}_\theta = \underbrace{\frac{(1-a_t)^2}{2\sigma_q^2(t)(1-\bar{a}_t)a_t}}_{\lambda_t} \left[||e_0 - e_\theta(x_t, t)||_2^2 \right]$$

Learning $\nabla_x log p_\theta(x)$

Given a Gaussian variable $z \sim \mathcal{N}(z; \mu_z, \Sigma_z)$, Tweedie's Formula states:

$$\mathbb{E}[\mu_z|z] = z + \Sigma_z \nabla_z \log p(z)$$

From a known equation, we have:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_0, (1-\alpha_t)I)$$

By Tweedie's Formula, we get:

$$\mathbb{E}[\mu_{x_t}|x_t] = x_t + (1-\alpha_t)\nabla_{x_t} \log p(x_t)$$

The best estimate for the true mean $\mu_{x_t} = \sqrt{\alpha_t}x_0$, is:

$$\sqrt{\alpha_t}x_0 = x_t + (1-\alpha_t)\nabla_{x_t} \log p(x_t)$$

$$\Rightarrow x_0 = x_t + \frac{(1-\alpha_t)}{\sqrt{\alpha_t}}\nabla_{x_t} \log p(x_t)$$

Learning $\nabla_x log p_\theta(x)$

Remember,

$$\mu_q(x_t, x_0) = \frac{\sqrt{a_t}(1-\bar{a}_{t-1})x_t + \sqrt{\bar{a}_{t-1}}(1-a_t)x_0}{1-\bar{a}_t} = \frac{\sqrt{a_t}(1-\bar{a}_{t-1})x_t + \sqrt{\bar{a}_{t-1}}(1-a_t)(x_t + \frac{(1-\alpha_t)}{\sqrt{\alpha_t}}\nabla_{x_t}\log p(x_t))}{1-\bar{a}_t}$$

- $\mu_q(x_t, x_0) = \frac{1}{\sqrt{a_t}}x_t - \frac{1-a_t}{\sqrt{a_t}}\nabla_{x_t}\log p(x_t)$
- $\mu_\theta(x_t, t) = \frac{1}{\sqrt{a_t}}x_t - \frac{1-a_t}{\sqrt{a_t}}s_\theta(x_t, t)$

Reformulate the loss to:

$$\text{argmin}_\theta = \frac{(1-a_t)^2}{2\sigma_q^2(t)a_t}\left[||s_\theta(x_t, t) - \nabla_{x_t}\log p(x_t)||_2^2\right]$$

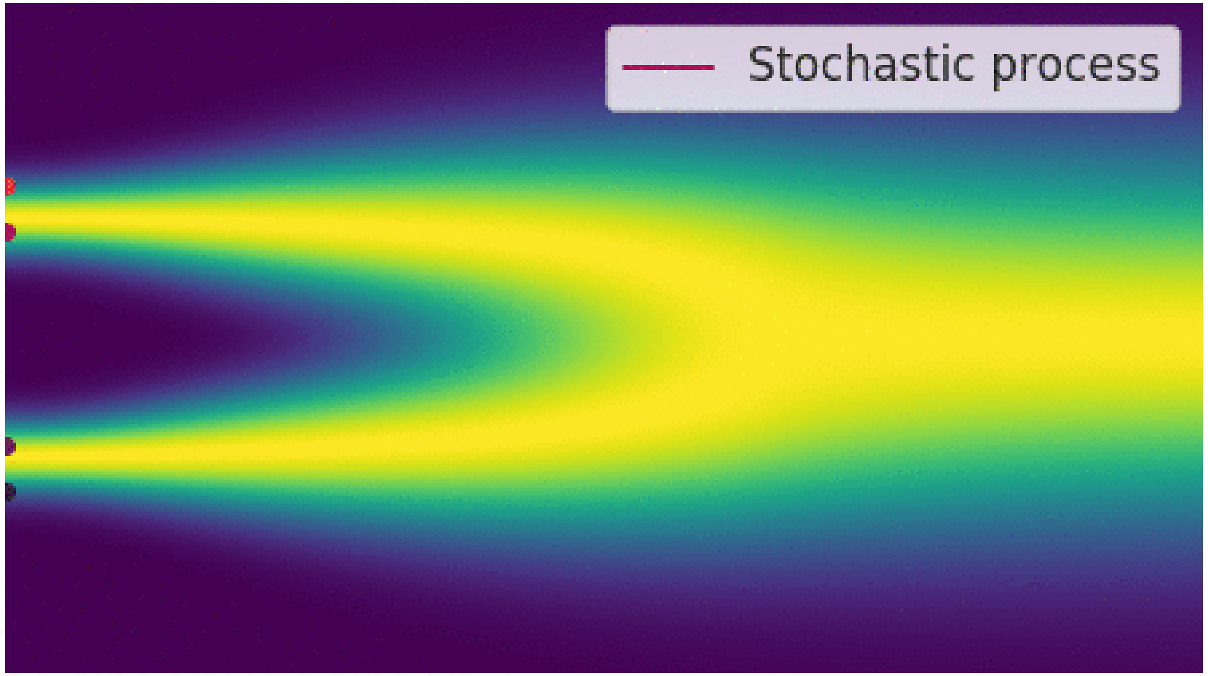
The score looks like ϵ_0 (?!)

$$x_0 = \frac{x_t + \sqrt{1-\bar{a}_t}\epsilon_0}{\sqrt{\bar{a}_t}} = x_t + \frac{(1-\alpha_t)}{\sqrt{\alpha_t}} \nabla_{x_t} \log p(x_t)$$

$$\Rightarrow \nabla_{x_t} \log p(x_t) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \epsilon_0$$

Perturbing images with SDEs

When the noise levels approach infinity, we essentially perturb the data with a **Stochastic Differential Equation (SDE)**



SDEs

In general, SDEs have the form

$$dx = f(x,t)dt + g(t)dw$$

where $f(.\,,t) : \mathbb{R}^D \to \mathbb{R}^D$ is called the *drift coefficient* and $g(t) \in \mathbb{R}$ is called the *diffusion coefficient*.

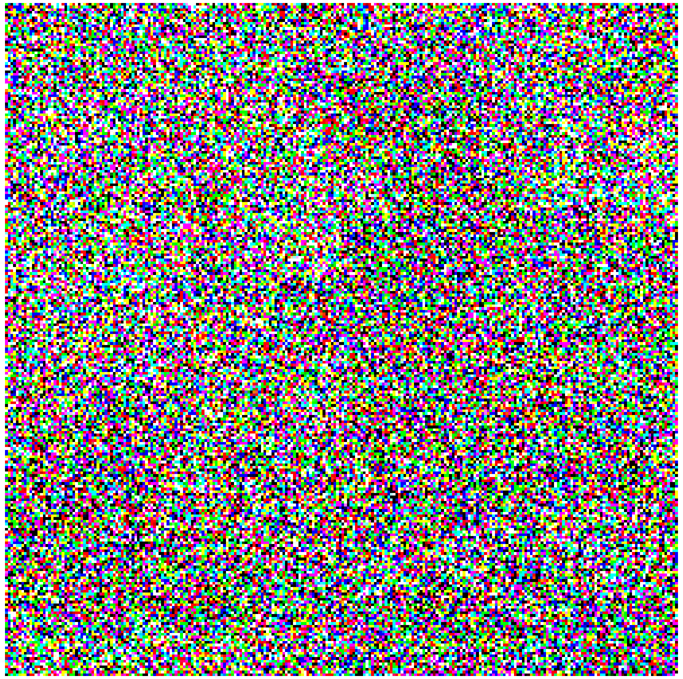
w denotes a standard Brownian motion, and dw can be considered an infinitesimal white noise.

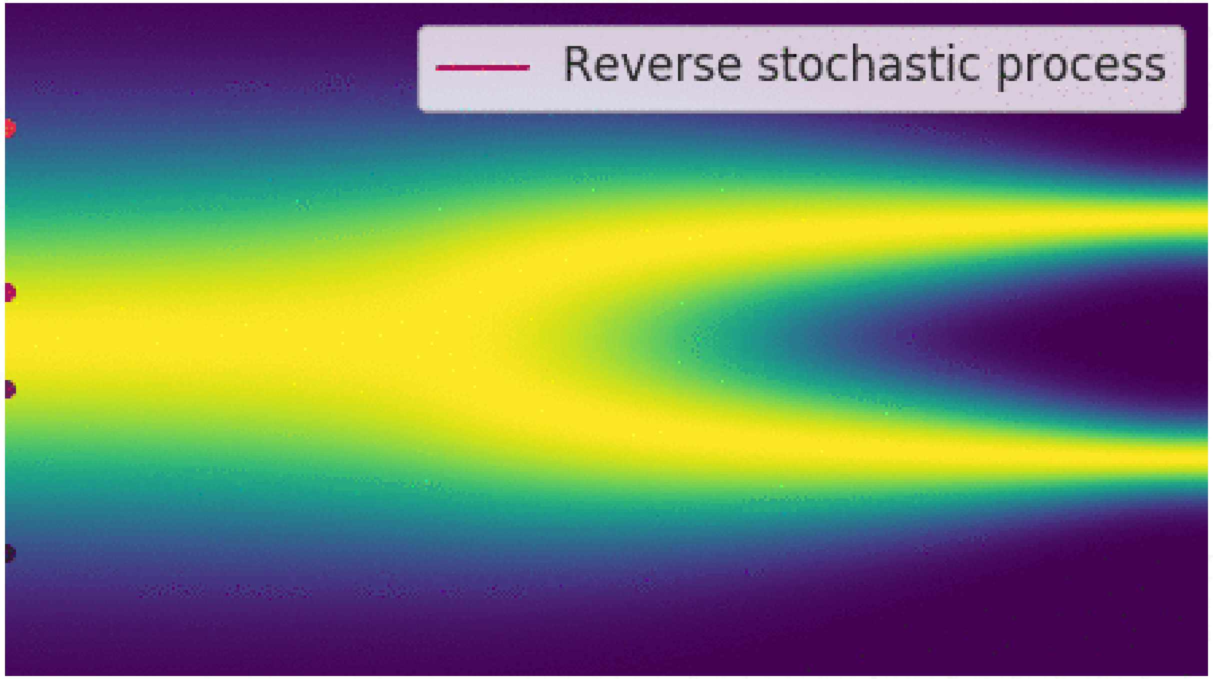
The solution is a continuous collection of random variables $\{x(t)\}_{t\in[0,1]}$.

Reversing the SDE

To sample from $x(T) \sim p_T$ and get new data from p_{data}, we can reverse the SDE (the reverse of a diffusion process is also a diffusion process [Anderson 1982]):

$$dx = \left[f(x,t) - g^2(t)\nabla_x \log p_t(x)\right]dt + g(t)d\tilde{w}.$$





Estimating the scores

To solve the reverse-time SDE we need:

- The terminal distribution $p(T) \approx \pi(x)$
- The score $\nabla_x \log p_t(x)$

To estimate the score we can use score matching techniques to train a *time dependent score model*, with objective:

$$\mathbb{E}_{t \in U(0,T)} \mathbb{E}_{p_t(x)} [\lambda(t) \|\nabla_x \log p_t(x) - s_\theta(x,t)\|_2^2]$$

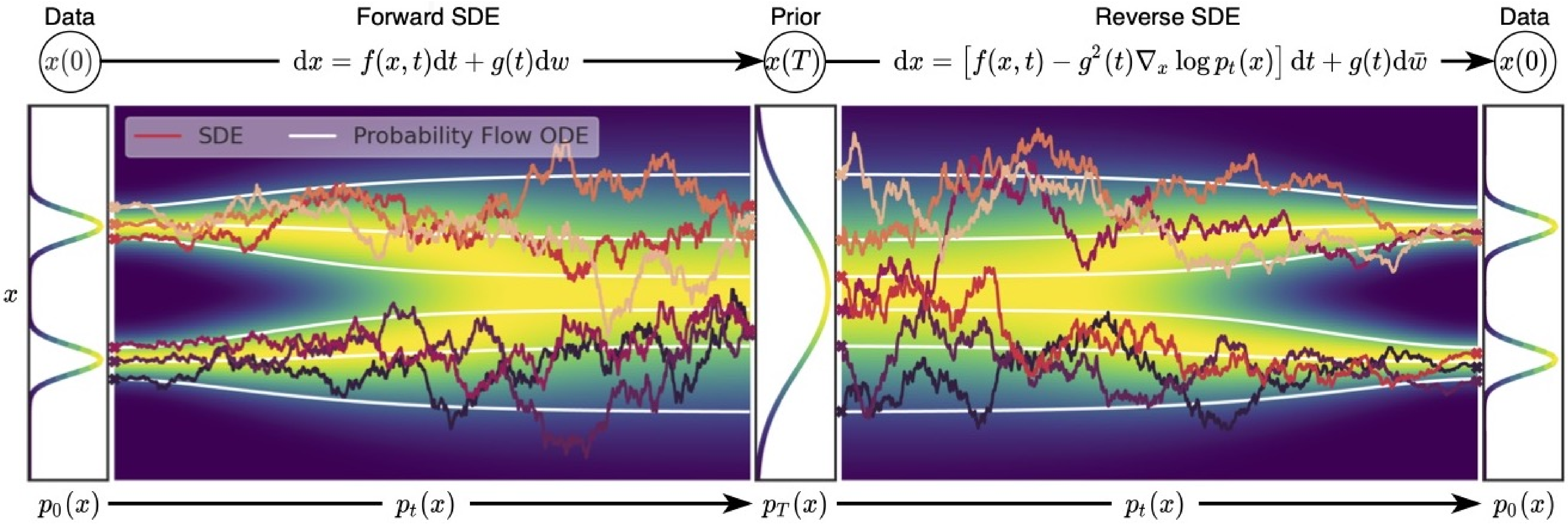
Then we can solve the reverse-time SDE with numerical SDE solvers (e.g. Euler - Maruyama)

Probability flow ODEs

For all diffusion processes, there exists a deterministicc process whose trajectories share the same marginal probabilities $\{p_t(x)\}_{t=0}^T$ as the SDE.

This process satisfies the *probability flow ODE*:

$$dx = \left[f(x, t) - \frac{1}{2} g^2(t) \nabla_x \log p_t(x) \right] dt.$$


Data
Forward SDE
Prior
Reverse SDE
Data
$x(0)$
$\mathrm{d}x = f(x,t)\mathrm{d}t + g(t)\mathrm{d}w$
$x(T)$
$\mathrm{d}x = \left[f(x,t) - g^2(t)\nabla_x \log p_t(x)\right]\mathrm{d}t + g(t)\mathrm{d}\bar{w}$
$x(0)$
SDE
Probability Flow ODE
x
$p_0(x)$
$p_t(x)$
$p_T(x)$
$p_t(x)$
$p_0(x)$

Neural ODE

When $\nabla_x \log p_t(x)$ is replaced by $s(x, t)$, it becomes a special case of *neural ODE*, specifically continuous normalizing flows.

So we get:

- Exact likelihood estimation.
- Encoding data points $x(0)$ to latent space $x(T)$.
  - Decoding by integrating corresponding ODE for reverse-time SDE.
  - We can manipulate the latent representation for editing by interpolation, temperature scaling.
- We get a uniquely identifiable encoding given sufficient data and model capacity.
- Efficient sampling by discretizing the ODE (link with DDIM?)

Thank you for your attention!