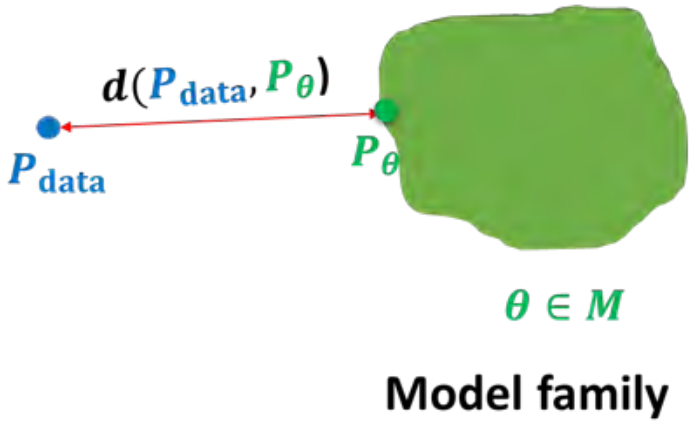
Today's lecture

$x_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



Goal: Training without sampling

- Score Matching
- Noise Contrastive Estimation
- Adversarial training

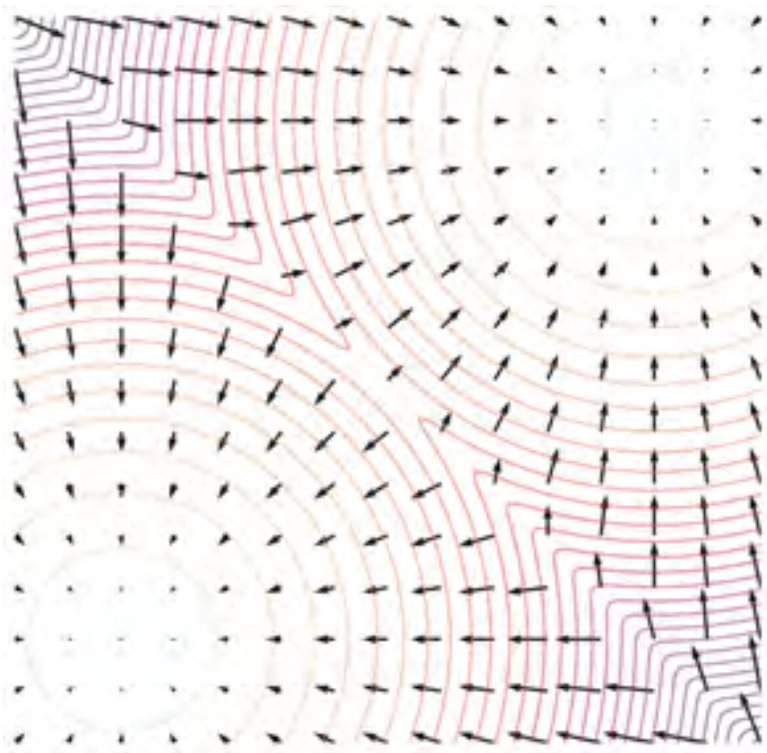
Score function

Energy-based model: $p_\theta(\mathbf{x}) = \frac{\exp\{f_\theta(\mathbf{x})\}}{Z(\theta)}$, $\log p_\theta(\mathbf{x}) = f_\theta(\mathbf{x}) - \log Z(\theta)$

(Stein) Score function:

$$s_\theta(\mathbf{x}) := \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} f_\theta(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}} \log Z(\theta)}_{=0} = \nabla_{\mathbf{x}} f_\theta(\mathbf{x})$$

- Gaussian distribution
  $p_\theta(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
  $\longrightarrow s_\theta(x) = -\frac{x-\mu}{\sigma^2}$
- Gamma distribution
  $p_\theta(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$
  $\longrightarrow s_\theta(x) = \frac{\alpha-1}{x} - \beta$



$p_\theta(\mathbf{x})$ vs. $s_\theta(\mathbf{x})$

Score matching

Observation

$s_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$ is independent of the partition function $Z(\theta)$.

Fisher divergence between $p(\mathbf{x})$ and $q(\mathbf{x})$:

$$D_F(p, q) := \frac{1}{2} E_{\mathbf{x} \sim p}[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \nabla_{\mathbf{x}} \log q(\mathbf{x})\|_2^2]$$

Score matching: minimizing the Fisher divergence between $p_{\text{data}}(\mathbf{x})$ and the EBM $p_\theta(\mathbf{x}) \propto \exp\{f_\theta(\mathbf{x})\}$

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}}[\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - s_\theta(\mathbf{x})\|_2^2]$$

$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}}[\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \nabla_{\mathbf{x}} f_\theta(\mathbf{x})\|_2^2]$$

Score matching

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}}[\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})\|_2^2]$$

How to deal with $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ given only samples? Integration by parts!

$$\frac{1}{2} E_{x \sim p_{\text{data}}}[(\nabla_x \log p_{\text{data}}(x) - \nabla_x \log p_\theta(x))^2] \quad \text{(Univariate case)}$$

$$= \frac{1}{2} \int p_{\text{data}}(x)[(\nabla_x \log p_{\text{data}}(x) - \nabla_x \log p_\theta(x))^2] \mathrm{d}x$$

$$= \frac{1}{2} \int p_{\text{data}}(x)(\nabla_x \log p_{\text{data}}(x))^2 \mathrm{d}x + \frac{1}{2} \int p_{\text{data}}(x)(\nabla_x \log p_\theta(x))^2 \mathrm{d}x$$

$$- \int p_{\text{data}}(x) \nabla_x \log p_{\text{data}}(x) \nabla_x \log p_\theta(x) \mathrm{d}x$$

Recall Integration by parts: $\int f'g = fg - \int g'f$.

$$- \int p_{\text{data}}(x) \nabla_x \log p_{\text{data}}(x) \nabla_x \log p_\theta(x) \mathrm{d}x$$

$$= - \int p_{\text{data}}(x) \frac{1}{p_{\text{data}}(x)} \nabla_x p_{\text{data}}(x) \nabla_x \log p_\theta(x) \mathrm{d}x$$

$$= \underbrace{-p_{\text{data}}(x) \nabla_x \log p_\theta(x)|_{x=-\infty}^{\infty}}_{=0} + \int p_{\text{data}}(x) \nabla_x^2 \log p_\theta(x) \mathrm{d}x$$

$$= \int p_{\text{data}}(x) \nabla_x^2 \log p_\theta(x) \mathrm{d}x$$

Note: we need to assume p_{data} decays sufficiently rapidly, $p_{\text{data}}(x) \to 0$ when $x \to \pm\infty$.

Score matching

Univariate score matching

$$\frac{1}{2} E_{x \sim p_{\text{data}}}[(\nabla_x \log p_{\text{data}}(x) - \nabla_x \log p_\theta(x))^2]$$

$$= \frac{1}{2} \int p_{\text{data}}(x)(\nabla_x \log p_{\text{data}}(x))^2 \mathrm{d}x + \frac{1}{2} \int p_{\text{data}}(x)(\nabla_x \log p_\theta(x))^2 \mathrm{d}x - \int p_{\text{data}}(x) \nabla_x \log p_{\text{data}}(x) \nabla_x \log p_\theta(x) \mathrm{d}x$$

$$= \underbrace{\frac{1}{2} \int p_{\text{data}}(x)(\nabla_x \log p_{\text{data}}(x))^2 \mathrm{d}x}_{\text{const. wrt } \theta} + \frac{1}{2} \int p_{\text{data}}(x)(\nabla_x \log p_\theta(x))^2 \mathrm{d}x + \int p_{\text{data}}(x) \nabla_x^2 \log p_\theta(x) \mathrm{d}x$$

$$= E_{x \sim p_{\text{data}}}[\tfrac{1}{2}(\nabla_x \log p_\theta(x))^2 + \nabla_x^2 \log p_\theta(x)] + \text{const.}$$

Multivariate score matching (integration by parts, i.e. Gauss theorem)

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}}[\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})\|_2^2]$$

$$= E_{\mathbf{x} \sim p_{\text{data}}}\left[\frac{1}{2}\|\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})\|_2^2 + \text{tr}\Big(\underbrace{\nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x})}_{\text{Hessian of } \log p_\theta(\mathbf{x})}\Big)\right] + \text{const.}$$

Score matching

1. Sample a mini-batch of datapoints $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$.
2. Estimate the score matching loss with the empirical mean

$$\frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{2}\|\nabla_{\mathbf{x}}\log p_\theta(\mathbf{x}_i)\|_2^2 + \text{tr}(\nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x}_i))\right]$$

$$=\frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{2}\|\nabla_{\mathbf{x}} f_\theta(\mathbf{x}_i)\|_2^2 + \text{trace}(\nabla_{\mathbf{x}}^2 f_\theta(\mathbf{x}_i))\right]$$

3. Stochastic gradient descent.
4. No need to sample from the EBM!

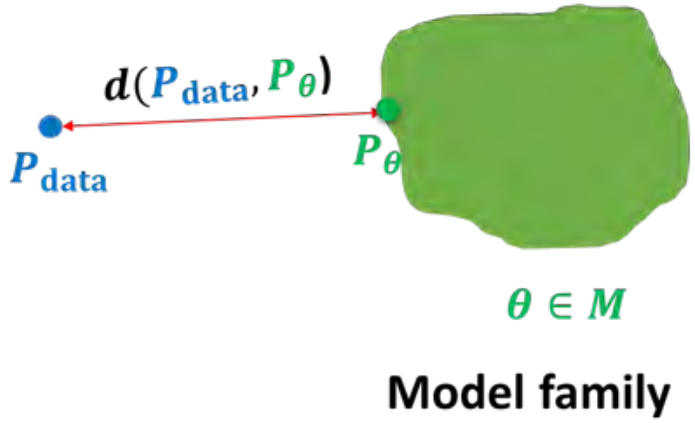
Caveat

Computing the trace of Hessian $\text{tr}(\nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x}))$ is in general very expensive for large models.

Denoising score matching (Vincent 2010) and sliced score matching (Song et al. 2019). More on this in the next lecture!

Recap.

$\mathbf{x}_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



Distances used for training energy-based models.

- KL divergence = maximum likelihood.

$$\nabla_\theta f_\theta(\mathbf{x}_{data}) - f_\theta(\mathbf{x}_{sample}) \quad \text{(contrastive divergence)}$$

- Fisher divergence = score matching.

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\| \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \nabla_{\mathbf{x}} f_\theta(\mathbf{x}) \|_2^2]$$

Noise contrastive estimation

Learning an energy-based model by contrasting it with a noise distribution.

- Data distribution: $p_{\text{data}}(\mathbf{x})$.
- Noise distribution: $p_n(\mathbf{x})$. Should be analytically tractable and easy to sample from.
- Training a discriminator $D_\theta(\mathbf{x}) \in [0, 1]$ to distinguish between data samples and noise samples.

$$\max_\theta E_{\mathbf{x}\sim p_{\text{data}}}[\log D_\theta(\mathbf{x})] + E_{\mathbf{x}\sim p_n}[\log(1 - D_\theta(\mathbf{x}))]$$

- What is the Optimal discriminator $D_{\theta^*}(\mathbf{x})$?

$$D_{\theta^*}(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_n(\mathbf{x})}$$

Noise contrastive estimation

What if the discriminator is parameterized by

$$D_\theta(\mathbf{x}) = \frac{p_\theta(\mathbf{x})}{p_\theta(\mathbf{x}) + p_n(\mathbf{x})}$$

- The optimal discriminator $D_{\theta^*}(\mathbf{x})$ satisfies

$$D_{\theta^*}(\mathbf{x}) = \frac{p_{\theta^*}(\mathbf{x})}{p_{\theta^*}(\mathbf{x}) + p_n(\mathbf{x})} = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_n(\mathbf{x})}$$

- By training the discriminator, we are implicitly learning $p_{\theta^*}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$. Particularly suitable for cases where $p_\theta(\mathbf{x})$ is defined up to a normalization constant (EBMs)
- Equivalently,

$$p_{\theta^*}(\mathbf{x}) = \frac{p_n(\mathbf{x}) D_{\theta^*}(\mathbf{x})}{1 - D_{\theta^*}(\mathbf{x})} = p_{\text{data}}(\mathbf{x})$$

  Classifier is used to correct density estimates from p_n. Can be used to improve a base generative model (*Boosted Generative Models*, Grover et al., 2018)

Noise contrastive estimation for training EBMs

Energy-based model:

$$p_\theta(\mathbf{x}) = \frac{e^{f_\theta(\mathbf{x})}}{Z(\theta)}$$

The constraint $Z(\theta) = \int e^{f_\theta(\mathbf{x})} \mathrm{d}\mathbf{x}$ is hard to satisfy.
Solution: Modeling $Z(\theta)$ with an additional trainable parameter Z that is not explicitly constrained to satisfy $Z = \int e^{f_\theta(\mathbf{x})} \mathrm{d}\mathbf{x}$.

$$p_{\theta,Z}(\mathbf{x}) = \frac{e^{f_\theta(\mathbf{x})}}{Z}$$

With noise contrastive estimation, the optimal parameters θ^*, Z^* are

$$p_{\theta^*,Z^*}(\mathbf{x}) = \frac{e^{f_{\theta^*}(\mathbf{x})}}{Z^*} = p_{\text{data}}(\mathbf{x})$$

The optimal parameter Z^* is the correct partition function, because

$$\int \frac{e^{f_{\theta^*}(\mathbf{x})}}{Z^*} \mathrm{d}\mathbf{x} = \int p_{\text{data}}(\mathbf{x}) \mathrm{d}\mathbf{x} = 1 \implies Z^* = \int e^{f_{\theta^*}(\mathbf{x})} \mathrm{d}\mathbf{x}$$

Noise contrastive estimation for training EBMs

The discriminator $D_{\theta,Z}(\mathbf{x})$ for probabilistic model $p_{\theta,Z}(\mathbf{x})$ is

$$D_{\theta,Z}(\mathbf{x}) = \frac{\frac{e^{f_\theta(\mathbf{x})}}{Z}}{\frac{e^{f_\theta(\mathbf{x})}}{Z} + p_n(\mathbf{x})} = \frac{e^{f_\theta(\mathbf{x})}}{e^{f_\theta(\mathbf{x})} + p_n(\mathbf{x})Z}$$

Noise contrastive estimation training

$$\max_{\theta,Z} E_{\mathbf{x}\sim p_{\text{data}}}[\log D_{\theta,Z}(\mathbf{x})] + E_{\mathbf{x}\sim p_n}[\log(1 - D_{\theta,Z}(\mathbf{x}))]$$

Equivalently,

$$\max_{\theta,Z} E_{\mathbf{x}\sim p_{\text{data}}}[f_\theta(\mathbf{x}) - \log(e^{f_\theta(\mathbf{x})} + Zp_n(\mathbf{x}))] + E_{\mathbf{x}\sim p_n}[\log(Zp_n(\mathbf{x})) - \log(e^{f_\theta(\mathbf{x})} + Zp_n(\mathbf{x}))]$$

Log-sum-exp trick for numerical stability:

$$\log(e^{f_\theta(\mathbf{x})} + Zp_n(\mathbf{x})) = \log(e^{f_\theta(\mathbf{x})} + e^{\log Z + \log p_n(\mathbf{x})}) = \text{logsumexp}(f_\theta(\mathbf{x}), \log Z + \log p_n(\mathbf{x}))$$

Noise contrastive estimation for training EBMs

1. Sample a mini-batch of datapoints $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n \sim p_{\text{data}}(\mathbf{x})$.
2. Sample a mini-batch of noise samples $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n \sim p_n(\mathbf{y})$.
3. Estimate the NCE loss.

$$\frac{1}{n}\sum_{i=1}^{n}[f_\theta(\mathbf{x}_i) - \text{logsumexp}(f_\theta(\mathbf{x}_i), \log Z + \log p_n(\mathbf{x}_i)) + \log Z + p_n(\mathbf{y}_i) - \text{logsumexp}(f_\theta(\mathbf{y}_i), \log Z + \log p_n(\mathbf{y}_i))]$$

4. Stochastic gradient ascent with respect to θ and Z.
5. No need to sample from the EBM!

Comparing NCE and GAN

Similarities:

- Both involve training a discriminator to perform binary classification with a cross-entropy loss.
- Both are likelihood-free (recall likelihood not tractable in EBM).

Differences:

- GAN requires adversarial training or minimax optimization for training, while NCE does not.
- NCE requires the likelihood of the noise distribution for training, while GAN only requires efficient sampling from the prior.
- NCE trains an energy-based model, while GAN trains a deterministic sample generator.

Flow contrastive estimation (Gao et al. 2020)

Observations:

- We need to both evaluate the probability of $p_n(\mathbf{x})$, and sample from it efficiently.
- We hope to make the classification task as hard as possible, i.e., $p_n(\mathbf{x})$ should be close to $p_{\text{data}}(\mathbf{x})$ (but not exactly the same).

Flow contrastive estimation:

- Parameterize the noise distribution with a normalizing flow model $p_{n,\phi}(\mathbf{x})$.
- Parameterize the discriminator $D_{\theta,Z,\phi}(\mathbf{x})$ as

$$D_{\theta,Z,\phi}(\mathbf{x}) = \frac{\frac{e^{f_\theta(\mathbf{x})}}{Z}}{\frac{e^{f_\theta(\mathbf{x})}}{Z} + p_{n,\phi}(\mathbf{x})} = \frac{e^{f_\theta(\mathbf{x})}}{e^{f_\theta(\mathbf{x})} + p_{n,\phi}(\mathbf{x})Z}$$

- Train the flow model to minimize $D_{JS}(p_{\text{data}}, p_{n,\phi})$:

$$\min_{\phi} \max_{\theta,Z} E_{\mathbf{x}\sim p_{\text{data}}}[\log D_{\theta,Z,\phi}(\mathbf{x})] + E_{\mathbf{x}\sim p_{n,\phi}}[\log(1 - D_{\theta,Z,\phi}(\mathbf{x}))]$$

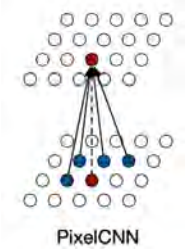
How to represent probability distributions?

- Probability density function (p.d.f.) or probability mass function (p.m.f.)

$$p(\mathbf{x})$$

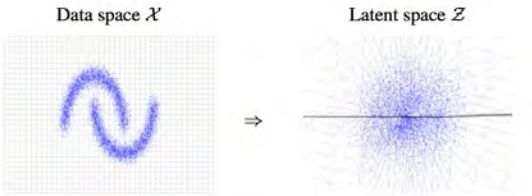
- Autogressive models

$$p_\theta(\mathbf{x}) = \prod_{i=1}^{d} p_\theta(\mathbf{x}_i \mid \mathbf{x}_{<i})$$



- Flow models

$$p_\theta(\mathbf{x}) = p(\mathbf{z})\left|\det(J_{f_\theta}(\mathbf{x}))\right|, \quad \mathbf{z} = f_\theta(\mathbf{x})$$



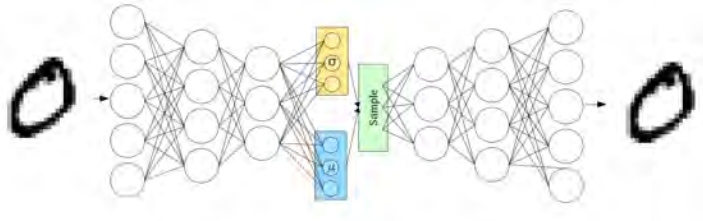
How to represent probability distributions?

- Probability density function (p.d.f.) or probability mass function (p.m.f.)

$$p(\mathbf{x})$$

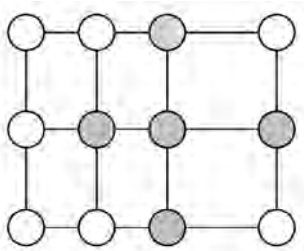
- Variational autoencoders

$$p_\theta(\mathbf{x}) = \int p(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z}) \, \mathrm{d}\mathbf{z}$$



- Energy-based models

$$p_\theta(\mathbf{x}) = \frac{e^{f_\theta(\mathbf{x})}}{Z(\theta)}$$



How to represent probability distributions?

- Probability density function (p.d.f.) or probability mass function (p.m.f.)

$$p(\mathbf{x})$$

- Pros
  - Maximum likelihood training
  - Principled model comparison via likelihoods

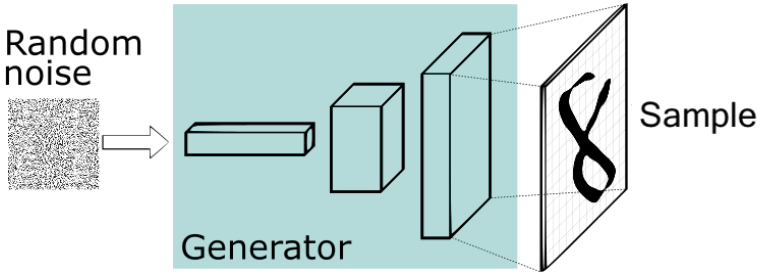
- Cons
  - Special architectures or surrogate losses to deal with intractable partition functions

How to represent probability distributions?

- Sampling process

- Generative adversarial networks (GANs)

$$\mathbf{z} \sim p(\mathbf{z})$$
$$\mathbf{x} = g_\theta(\mathbf{z})$$



GAN

How to represent probability distributions?

- Sampling process

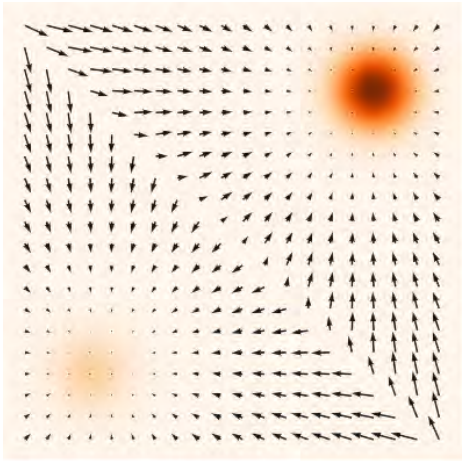
- Pros
  - Samples typically have better quality

- Cons
  - Require adversarial training. Training instability and mode collapse.
  - No principled way to compare different models
  - No principled termination criteria for training

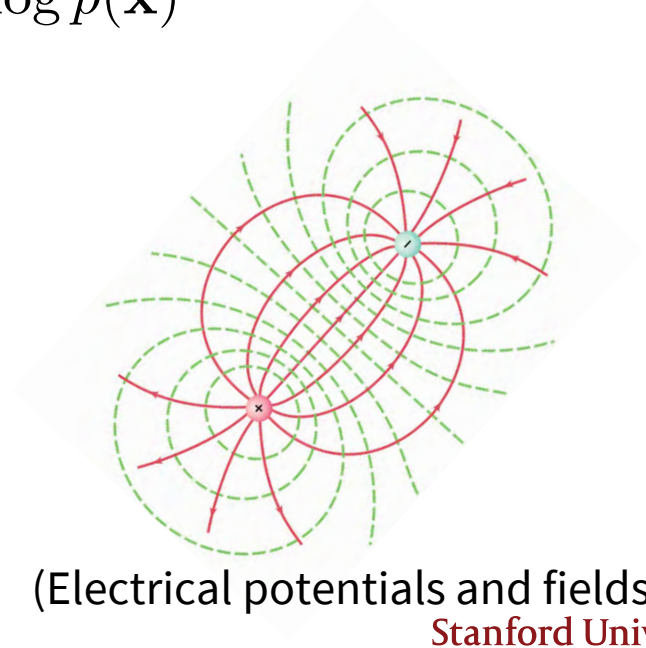
How to represent probability distributions?

- When the pdf is differentiable, we can compute the gradient of a probability density.

Score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$



(pdf and score)

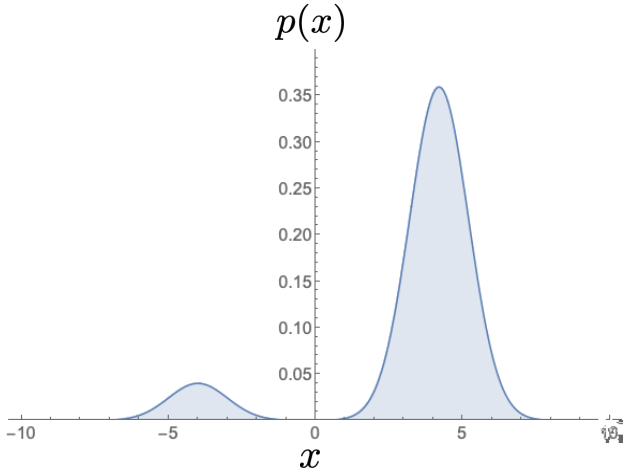


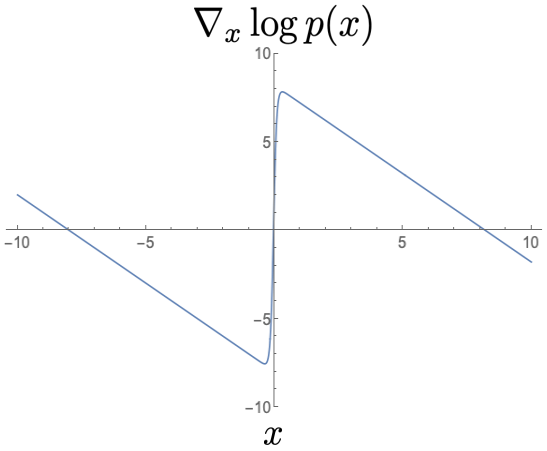
(Electrical potentials and fields)

How to represent probability distributions?

- When the pdf is differentiable, we can compute the gradient of a probability density.

Score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$



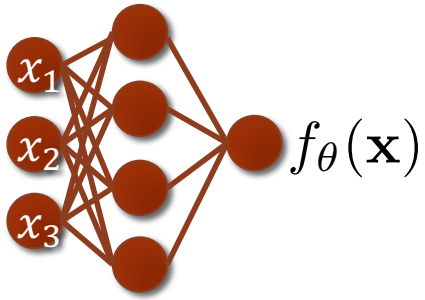


Recap on energy-based models

- Deep Energy-Based models (EBMs)

$$f_\theta(\mathbf{x}) \in \mathbb{R}$$

$$p_\theta(\mathbf{x}) = \frac{e^{f_\theta(\mathbf{x})}}{Z(\theta)}$$



- Maximum likelihood training: $\max_\theta f_\theta(\mathbf{x}_{\text{train}}) - \log Z(\theta)$
  - Contrastive divergence

$$\nabla_\theta f_\theta(\mathbf{x}_{\text{train}}) - \nabla_\theta \log Z(\theta) \approx \nabla_\theta f_\theta(\mathbf{x}_{\text{train}}) - \nabla_\theta f_\theta(\mathbf{x}_{\text{sample}})$$

$$\mathbf{x}_{\text{sample}} \sim p_\theta(\mathbf{x})$$

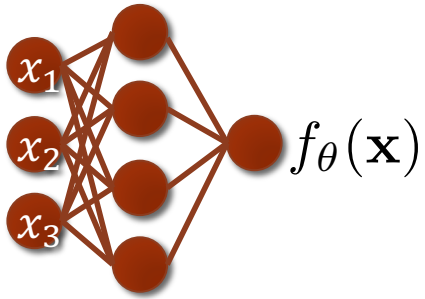
  - Requires iterative sampling during training

Recap on energy-based models

- Deep Energy-Based models (EBMs)

$$f_\theta(\mathbf{x}) \in \mathbb{R}$$

$$p_\theta(\mathbf{x}) = \frac{e^{f_\theta(\mathbf{x})}}{Z(\theta)}$$



- Minimizing Fisher divergence: $\min_\theta \frac{1}{2} E_{\mathbf{x}\sim p_{\text{data}}}[\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})\|_2^2]$
  - Score matching

$$\frac{1}{2} E_{\mathbf{x}\sim p_{\text{data}}}[\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})\|_2^2]$$
$$= \frac{1}{2} E_{\mathbf{x}\sim p_{\text{data}}}\left[\frac{1}{2}\|\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})\|_2^2 + \text{tr}(\nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x}))\right] + \text{const.}$$

Score matching for training EBMs

- Score function of EBMs

$$\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} f_\theta(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}} \log Z(\theta)}_{=0} = \nabla_{\mathbf{x}} f_\theta(\mathbf{x})$$

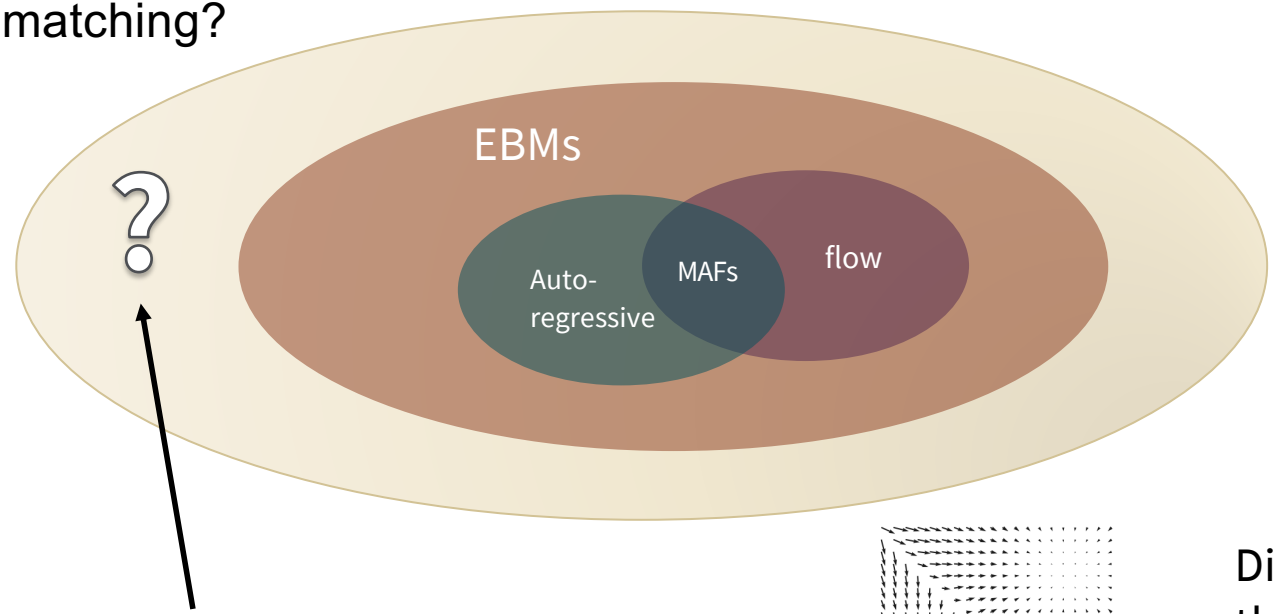
- Score matching for EBMs:

$$E_{\mathbf{x} \sim p_{\text{data}}} \left[\frac{1}{2} \left\| \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) \right\|_2^2 + \text{tr}(\nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x})) \right]$$
$$= E_{\mathbf{x} \sim p_{\text{data}}} \left[\frac{1}{2} \left\| \nabla_{\mathbf{x}} f_\theta(\mathbf{x}) \right\|_2^2 + \text{tr}(\nabla_{\mathbf{x}}^2 f_\theta(\mathbf{x})) \right]$$

- Is score matching limited to EBMs?
  - Autoregressive models
  - Normalizing flow models

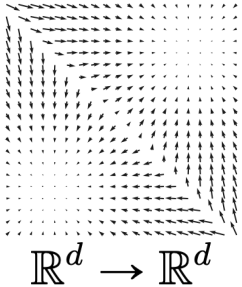
Score-based models

- What's the most general model that can be efficiently trained by score matching?



- Score-based model

$$\boldsymbol{s}_\theta(\mathbf{x})$$

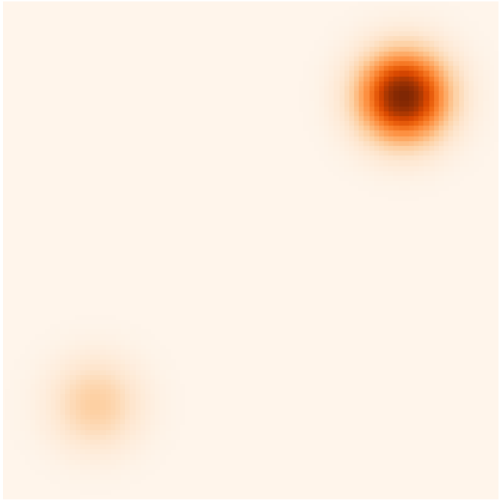


Directly model the vector field of gradients!

Score estimation by training score-based models

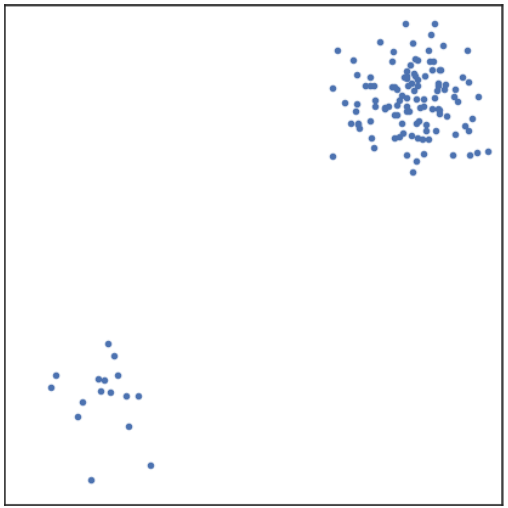
Probability density

$p_{\text{data}}(\mathbf{x})$



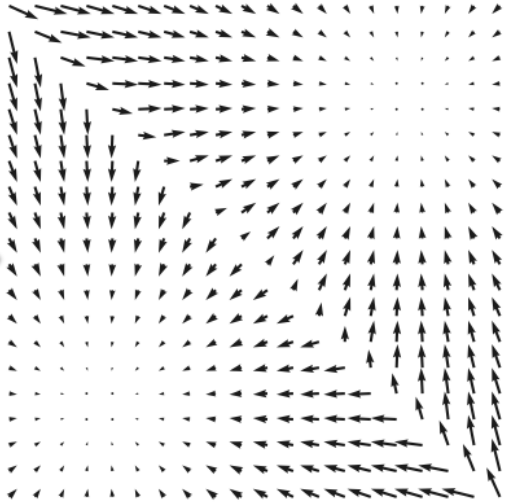
i.i.d. samples

$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$



Score function

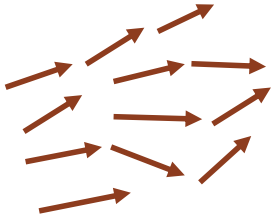
$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$



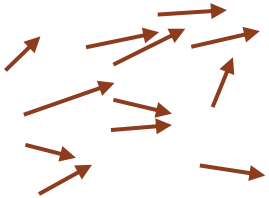
Score estimation by training score-based models

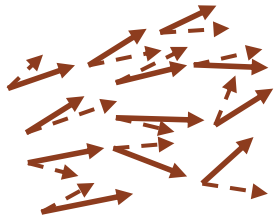
- **Given:** i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- **Task:** Estimating the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- **Score Model:** A learnable vector-valued function $\boldsymbol{s}_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$
- **Goal:** $\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- How to compare two vector fields of scores?

$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$



$s_\theta(\mathbf{x})$





Average Euclidean distance over the space

Score estimation by training score-based models

- **Objective:** Average Euclidean distance over the whole space.

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\| \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2]$$

(Fisher divergence)

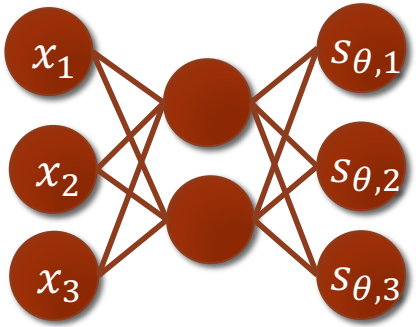
- **Score matching:**

$$E_{\mathbf{x} \sim p_{\text{data}}} \left[\frac{1}{2} \| \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2 + \text{tr}(\underbrace{\nabla_{\mathbf{x}} \boldsymbol{s}_\theta(\mathbf{x})}_{\text{Jacobian of } \boldsymbol{s}_\theta(\mathbf{x})}) \right]$$

- **Requirements:**
  - The score model must be efficient to evaluate.
  - Do we need the score model to be a proper score function (i.e., gradient of a scalar “energy” function)?

Score matching is not scalable

- Deep neural networks as more expressive score models

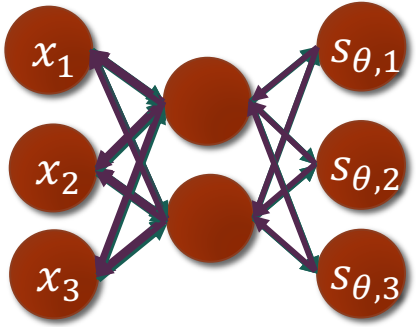


$$s_\theta(\mathbf{x})$$

Score Matching is not Scalable!

- Compute $\|s_\theta(\mathbf{x})\|_2^2$ and $\mathrm{tr}(\nabla_{\mathbf{x}} s_\theta(\mathbf{x}))$

$$\frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_1}$$
$$\frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_2}$$
$$\frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_3}$$



$$s_{\theta,1}(\mathbf{x})$$
$$s_{\theta,2}(\mathbf{x})$$
$$s_{\theta,3}(\mathbf{x})$$

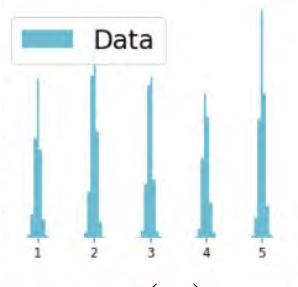
$O(d)$ Backprops!

$$\nabla_{\mathbf{x}} s_\theta(\mathbf{x}) = \begin{pmatrix} \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_1} & \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_2} & \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_3} \\ \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_1} & \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_2} & \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_3} \\ \frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_1} & \frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_2} & \frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_3} \end{pmatrix}$$

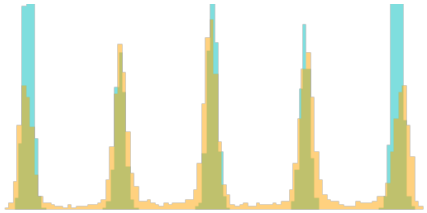
Denoising Score Matching (Vincent, 2011)

- Consider the perturbed distribution

$$q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = \mathcal{N}(\mathbf{x}; \sigma^2 I) \quad q_\sigma(\tilde{\mathbf{x}}) = \int p(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \mathrm{d}\mathbf{x}$$



$p(\mathbf{x})$



$p(\mathbf{x})$

q_σ

- Score estimation for $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})$ is easier
- If the noise level is small, this is a good approximation $q_\sigma(\tilde{\mathbf{x}}) \approx p(\tilde{\mathbf{x}})$

Denoising score matching

$\mathbf{x}$

$p_{\text{data}}(\mathbf{x})$

$q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$

$\tilde{\mathbf{x}}$

$q_\sigma(\tilde{\mathbf{x}})$

- Denoising score matching (Vincent 2011): matching the score of a noise-perturbed distribution

$$\frac{1}{2} E_{\tilde{\mathbf{x}} \sim q_\sigma}[\|\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) - \boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2]$$

$$= \frac{1}{2} \int q_\sigma(\tilde{\mathbf{x}}) \, \|\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) - \boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2 \, \mathrm{d}\tilde{\mathbf{x}}$$

$$= \frac{1}{2} \int q_\sigma(\tilde{\mathbf{x}}) \, \|\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})\|_2^2 \, \mathrm{d}\tilde{\mathbf{x}} + \frac{1}{2} \int q_\sigma(\tilde{\mathbf{x}}) \, \|\boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2 \, \mathrm{d}\tilde{\mathbf{x}}$$

$$- \int q_\sigma(\tilde{\mathbf{x}}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}}$$

$$= \text{const.} + \frac{1}{2} E_{\tilde{\mathbf{x}} \sim q_\sigma}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2] - \int q_\sigma(\tilde{\mathbf{x}}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}}$$

Denoising score matching

$\mathbf{x}$

$p_{\text{data}}(\mathbf{x})$

$q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$

$\tilde{\mathbf{x}}$

$q_\sigma(\tilde{\mathbf{x}})$

- Denoising score matching

$$
\begin{aligned}
&-\int q_\sigma(\tilde{\mathbf{x}}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}} \\
&= -\int q_\sigma(\tilde{\mathbf{x}}) \frac{1}{q_\sigma(\tilde{\mathbf{x}})} \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}} \\
&= -\int \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}} \\
&= -\int \nabla_{\tilde{\mathbf{x}}} \left(\int p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \, \mathrm{d}\mathbf{x} \right)^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}} \\
&= -\int \left(\int p_{\text{data}}(\mathbf{x}) \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \, \mathrm{d}\mathbf{x} \right)^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}} \\
&= -\int \left(\int p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \, \mathrm{d}\mathbf{x} \right)^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}} \\
&= -\iint p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\mathbf{x} \, \mathrm{d}\tilde{\mathbf{x}} \\
&= -E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})} [\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}})]
\end{aligned}
$$

Denoising score matching

$\mathbf{x}$

$p_{\text{data}}(\mathbf{x})$

$q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$

$q_\sigma(\tilde{\mathbf{x}})$

$\tilde{\mathbf{x}}$

- Denoising score matching

$$\frac{1}{2} E_{\tilde{\mathbf{x}} \sim q_\sigma}[\|\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) - \boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2]$$

$$= \text{const.} + \frac{1}{2} E_{\tilde{\mathbf{x}} \sim q_\sigma}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2] - \int q_\sigma(\tilde{\mathbf{x}}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}}$$

$$= \text{const.} + \frac{1}{2} E_{\tilde{\mathbf{x}} \sim q_\sigma}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2] - E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})}[\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}})]$$

$$= \text{const.} + \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2]$$

$$- \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})}[\|\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2]$$

$$= \text{const.} + \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2] + \text{const.}$$

$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2] + \text{const.}$$

Denoising score matching

- Estimate the score of a noise-perturbed distribution

$$\frac{1}{2} E_{\tilde{\mathbf{x}} \sim p_{\text{data}}} [\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) \|_2^2]$$
$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} | \mathbf{x})} [\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \|_2^2] + \text{const.}$$

- $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$ is easy to compute
  - $q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}} \mid \mathbf{x}, \sigma^2 \boldsymbol{I})$
  - $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = -\frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2}$
- **Pros:** efficient to optimize even for very high dimensional data, and useful for optimal denoising.
- **Con:** cannot estimate the score of clean data (noise-free)

Denoising score matching

- Sample a minibatch of datapoints $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- Sample a minibatch of perturbed datapoints $\{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \cdots, \tilde{\mathbf{x}}_n\} \sim q_\sigma(\tilde{\mathbf{x}})$

$$\tilde{\mathbf{x}}_i \sim q_\sigma(\tilde{\mathbf{x}}_i \mid \mathbf{x}_i)$$

- Estimate the denoising score matching loss with empirical means

$$\frac{1}{2n}\sum_{i=1}^{n}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}_i) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}_i \mid \mathbf{x}_i)\|_2^2]$$

- If Gaussian perturbation

$$\frac{1}{2n}\sum_{i=1}^{n}\left[\left\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}_i) + \frac{\tilde{\mathbf{x}}_i - \mathbf{x}_i}{\sigma^2}\right\|_2^2\right]$$

- Stochastic gradient descent
- Need to choose a very small σ!

Denoising Score Matching (Vincent, 2011)

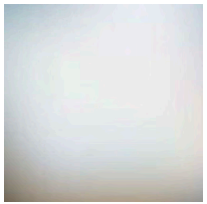
- Consider the perturbed distribution

$$q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = \mathcal{N}(\mathbf{x}; \sigma^2 I) \quad q_\sigma(\tilde{\mathbf{x}}) = \int p(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \mathrm{d}\mathbf{x}$$

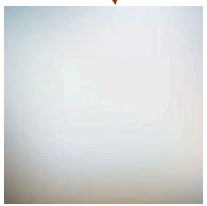
- Score estimation for $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})$ is easier

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

denoising $= \frac{1}{2}\mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}[\| \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}}) - s_\theta(\tilde{\mathbf{x}})\|_2^2] + \text{const.}$



x



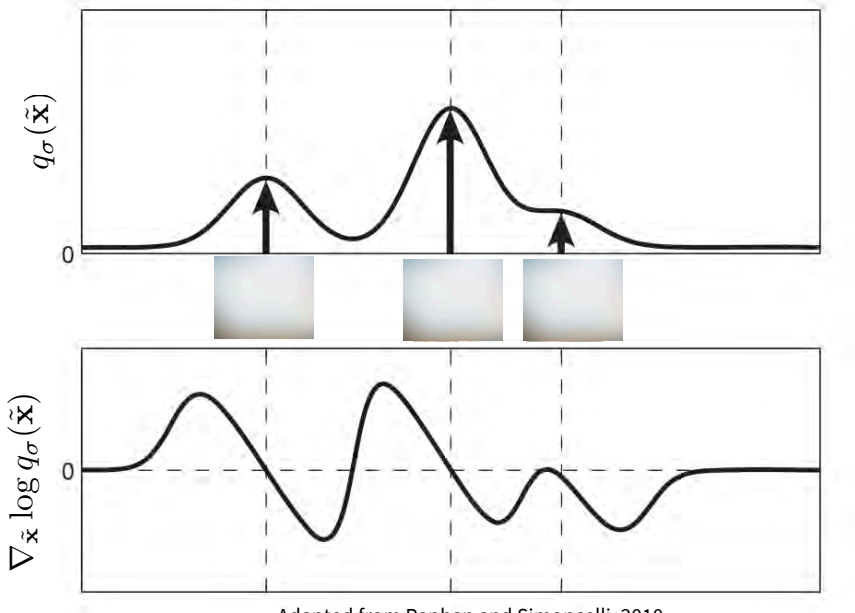
$\tilde{\mathbf{x}}$ = x + noise

$s_\theta(\tilde{\mathbf{x}})$ tries to estimate the noise that was added to produce $\tilde{\mathbf{x}}$

Tweedie's formula

- Score estimation for $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})$ is equivalent to denoising

$$\frac{1}{2}\mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\left[\left\| \frac{1}{\sigma^2}(\mathbf{x}-\tilde{\mathbf{x}}) - s_\theta(\tilde{\mathbf{x}})\right\|_2^2\right] + \text{const.}$$



Adapted from Raphan and Simoncelli, 2010

Tweedie's formula

Optimal denoising strategy is to follow the gradient (score):

$$\tilde{\mathbf{x}} + \sigma^2 \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})$$

Tweedie's formula and denoising score matching

- Denoising score matching is suitable for optimal denoising
- Given $p(\mathbf{x}), \quad q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}} \mid \mathbf{x}, \sigma^2 \boldsymbol{I})$, we can define the posterior $p(\mathbf{x} \mid \tilde{\mathbf{x}})$ with Bayes' rule

$$p(\mathbf{x} \mid \tilde{\mathbf{x}}) \propto p(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$$

- Recall that

$$q_\sigma(\tilde{\mathbf{x}}) = \int p(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \, \mathrm{d}\mathbf{x}$$

- Tweedie's formula:

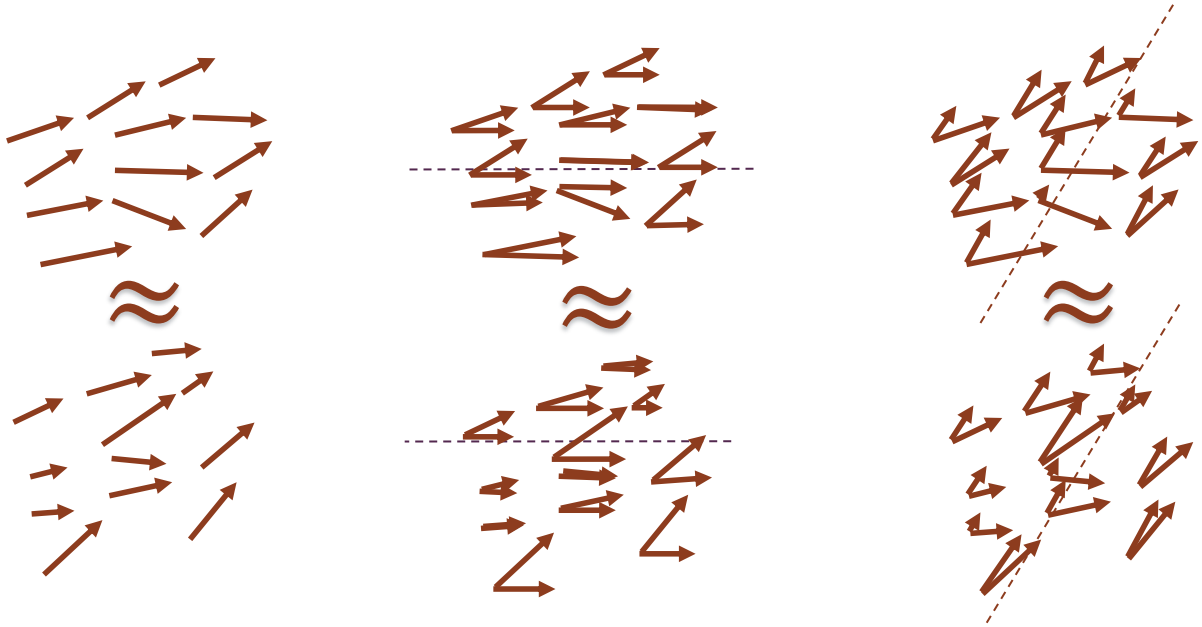
$$\begin{aligned} E_{\mathbf{x} \sim p(\mathbf{x} \mid \tilde{\mathbf{x}})}[\mathbf{x}] &= \tilde{\mathbf{x}} + \sigma^2 \nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}) \\ &\approx \tilde{\mathbf{x}} + \sigma^2 \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \end{aligned}$$

Sliced score matching

- One dimensional problems should be easier.
- Consider projections onto random directions.

$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$

$s_\theta(\mathbf{x})$



Song*, Garg*, Shi, Ermon. "Sliced Score Matching: A Scalable Approach to Density and Score Estimation." UAI 2019.

Sliced score matching

- **Objective:** Sliced Fisher Divergence

$$\frac{1}{2} E_{\mathbf{v} \sim p_{\mathbf{v}}} E_{\mathbf{x} \sim p_{\text{data}}} [(\mathbf{v}^\intercal \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \mathbf{v}^\intercal \boldsymbol{s}_\theta(\mathbf{x}))^2]$$

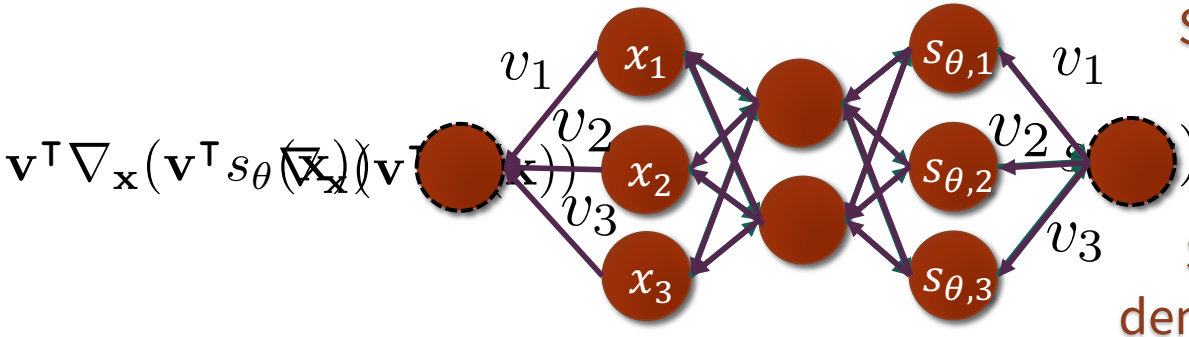
- **Integration by parts**

$$E_{\mathbf{v} \sim p_{\mathbf{v}}} E_{\mathbf{x} \sim p_{\text{data}}} \left[\mathbf{v}^\intercal \nabla_{\mathbf{x}} \boldsymbol{s}_\theta(\mathbf{x}) \mathbf{v} + \frac{1}{2} (\mathbf{v}^\intercal \boldsymbol{s}_\theta(\mathbf{x}))^2 \right]$$

$$\begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix} \begin{pmatrix} \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_1} & \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_2} & \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_3} \\ \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_1} & \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_2} & \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_3} \\ \frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_1} & \frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_2} & \frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_3} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

Computing Jacobian-vector products is scalable

$$\mathbf{v}^\intercal \nabla_{\mathbf{x}} s_\theta(\mathbf{x}) \mathbf{v} = \mathbf{v}^\intercal \nabla_{\mathbf{x}} (\mathbf{v}^\intercal s_\theta(\mathbf{x}))$$



One Backprop!

Sliced Score Matching is scalable

Slightly slower than denoising score matching

Sliced score matching

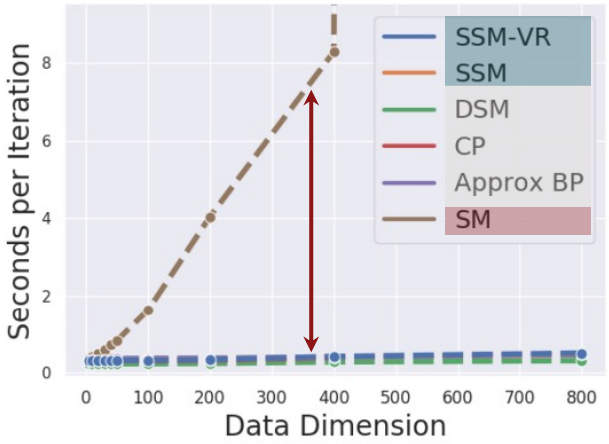
- Sample a minibatch of datapoints $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- Sample a minibatch of projection directions $\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n\} \sim p_{\mathbf{v}}$
- Estimate the sliced score matching loss with empirical means

$$\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{v}_i^\intercal \nabla_{\mathbf{x}} \boldsymbol{s}_\theta(\mathbf{x}_i)\mathbf{v}_i + \frac{1}{2}(\mathbf{v}_i^\intercal \boldsymbol{s}_\theta(\mathbf{x}_i))^2\right]$$

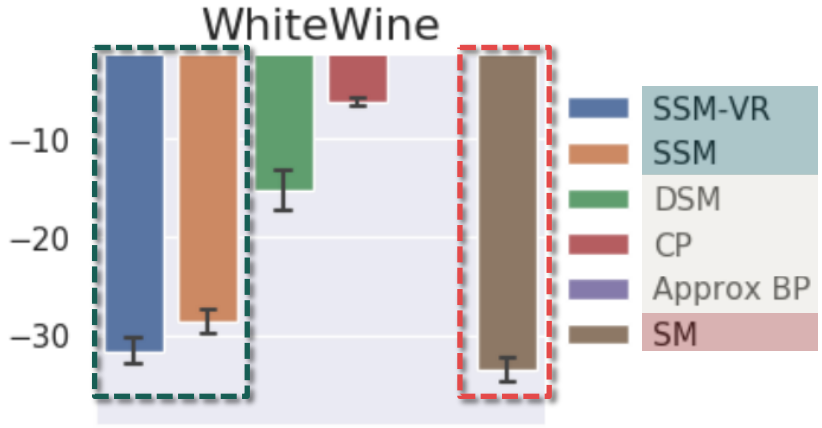
- The projection distribution is typically Gaussian or Rademacher
- Stochastic gradient descent
- Can use more projections per datapoint to boost performance

Experimental results for sliced score matching

Sliced score matching methods | Other Baselines | Score Matching



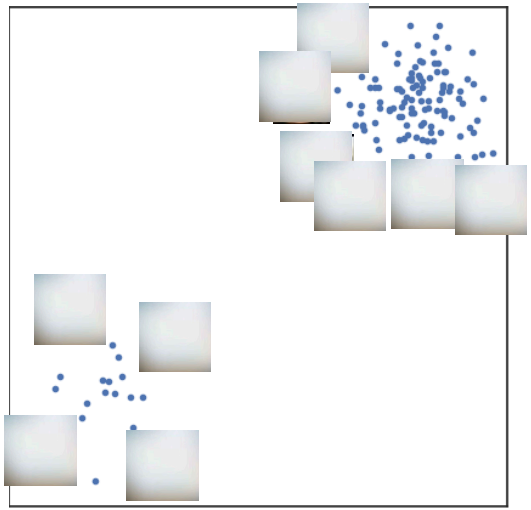
Efficiency



Performance on density estimation

Song*, Garg*, Shi, Ermon. "Sliced Score Matching: A Scalable Approach to Density and Score Estimation." UAI 2019.

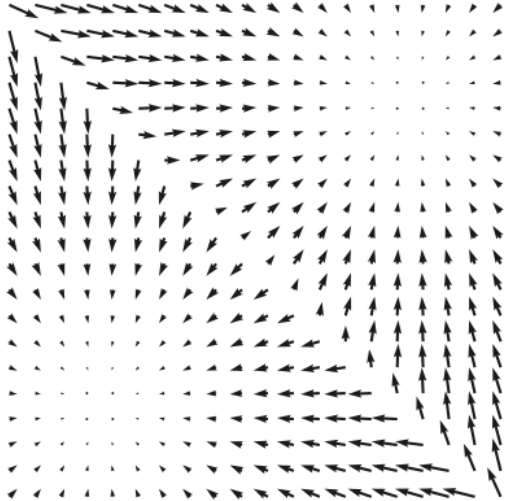
Score-based generative modeling



Data samples

$$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$$

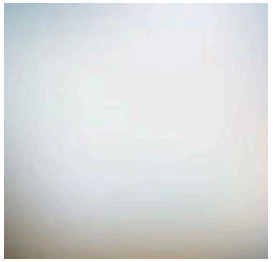
Score Matching



Scores

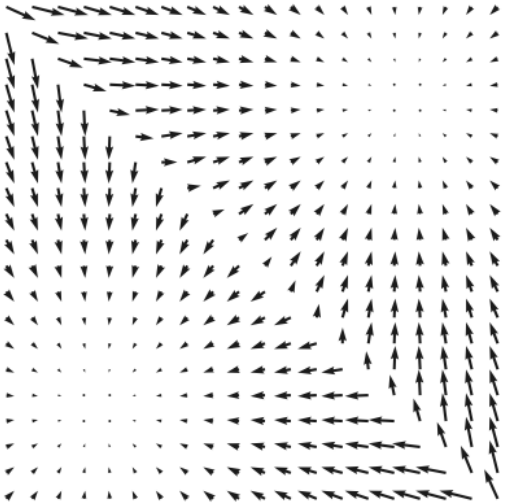
$$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

?



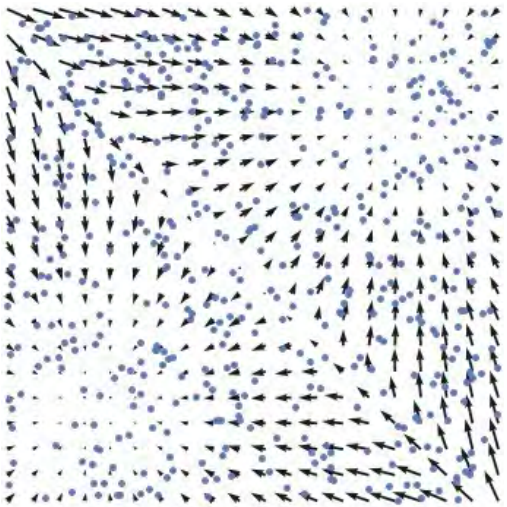
New samples

From scores to samples: Langevin MCMC



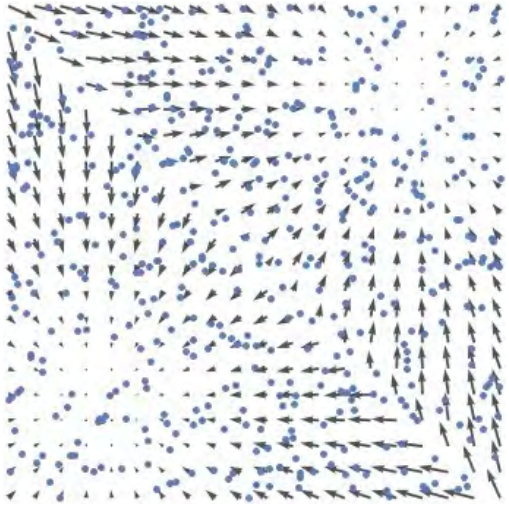
Scores

$\mathbf{s}_{\theta}(\mathbf{x})$



Follow the scores

$$\tilde{\mathbf{x}}_{t+1} \leftarrow \tilde{\mathbf{x}}_t + \frac{\epsilon}{2}\mathbf{s}_{\theta}(\tilde{\mathbf{x}}_t)$$



Follow noisy scores: Langevin MCMC

$$\mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\tilde{\mathbf{x}}_{t+1} \leftarrow \tilde{\mathbf{x}}_t + \frac{\epsilon}{2}\mathbf{s}_{\theta}(\tilde{\mathbf{x}}_t) + \sqrt{\epsilon}\,\mathbf{z}_t$$

Langevin dynamics sampling

- Sample from $p(\mathbf{x})$ using only the score $\nabla_{\mathbf{x}} \log p(\mathbf{x})$
- Initialize $\mathbf{x}^0 \sim \pi(\mathbf{x})$
- Repeat for $t \leftarrow 1, 2, \cdots, T$

$$\mathbf{z}^t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

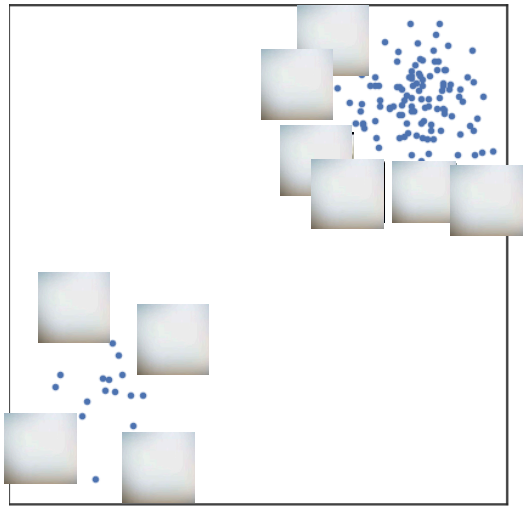
$$\mathbf{x}^t \leftarrow \mathbf{x}^{t-1} + \frac{\epsilon}{2} \nabla_{\mathbf{x}} \log p(\mathbf{x}^{t-1}) + \sqrt{\epsilon}\, \mathbf{z}^t$$

- If $\epsilon \to 0$ and $T \to \infty$, we are guaranteed to have $\mathbf{x}^T \sim p(\mathbf{x})$

- Langevin dynamics + score estimation

$$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$$

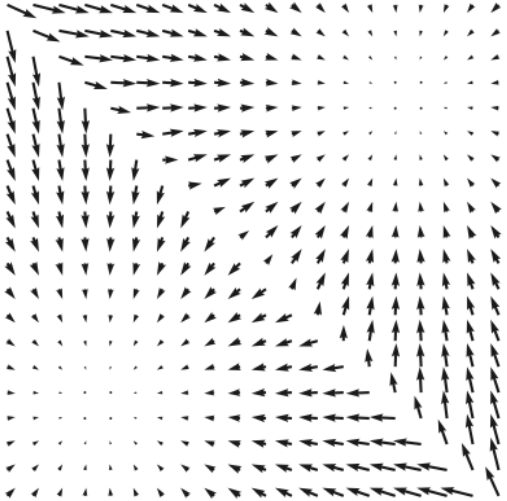
Score-based generative modeling



Data samples

$$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$$

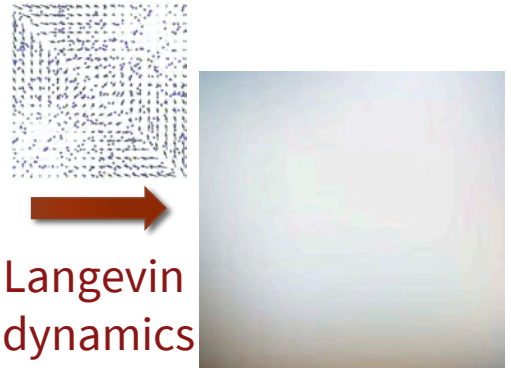
score matching



Scores

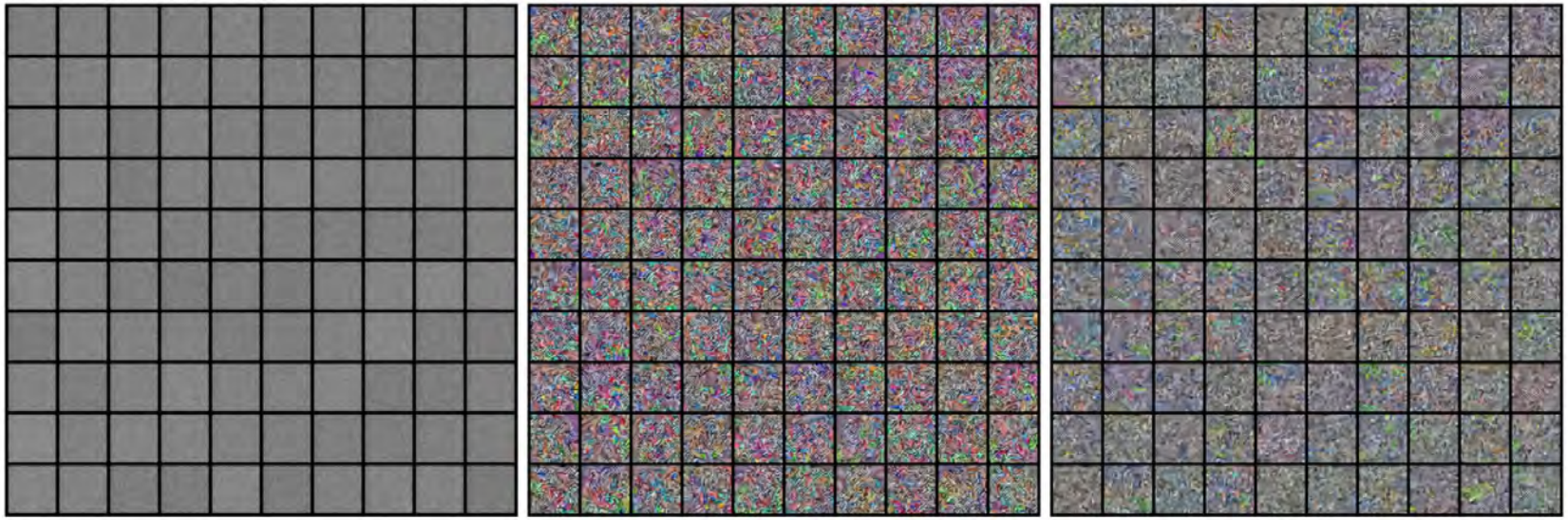
$$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

Langevin dynamics



New samples

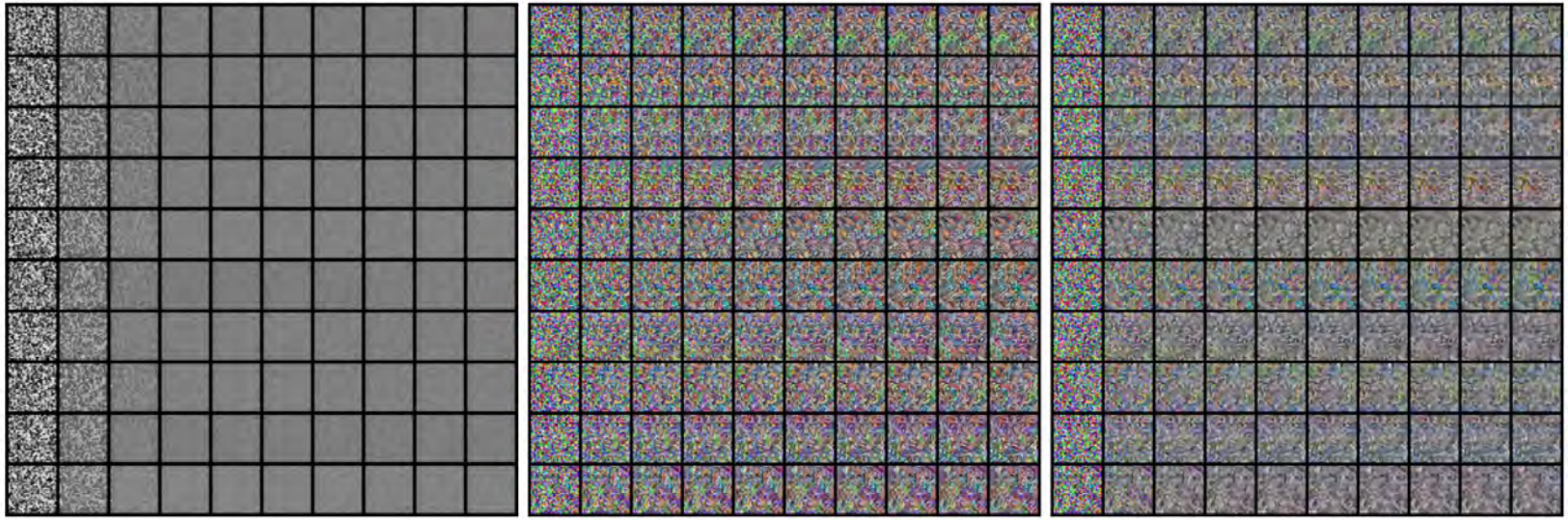
Score-based generative modeling: results



(a) MNIST (b) CelebA (c) CIFAR-10

Final samples

Score-based generative modeling: results

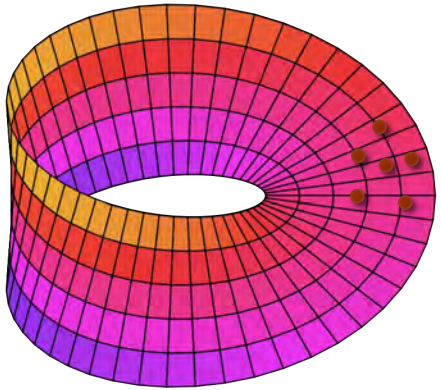


(a) MNIST (b) CelebA (c) CIFAR-10

Langevin sampling process

Pitfall 1: manifold hypothesis

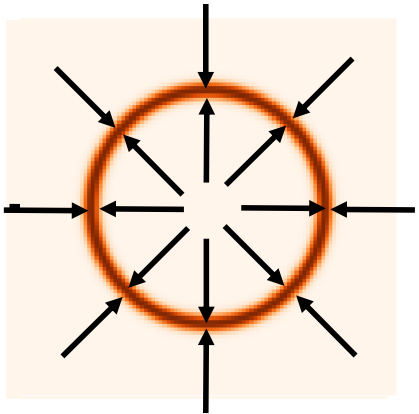
- Manifold hypothesis.



Data points

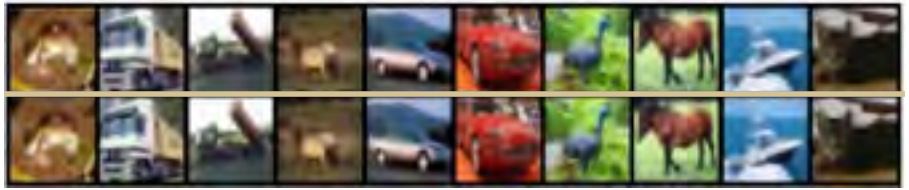
- Data score is undefined.

$$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

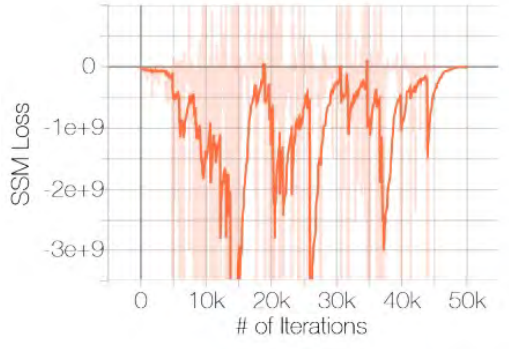


Pitfall 1: manifold hypothesis

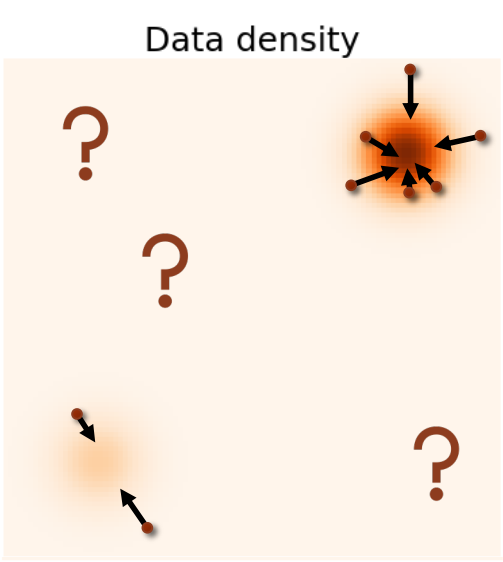
- Fitting the data with a low-dimensional linear manifold (PCA)

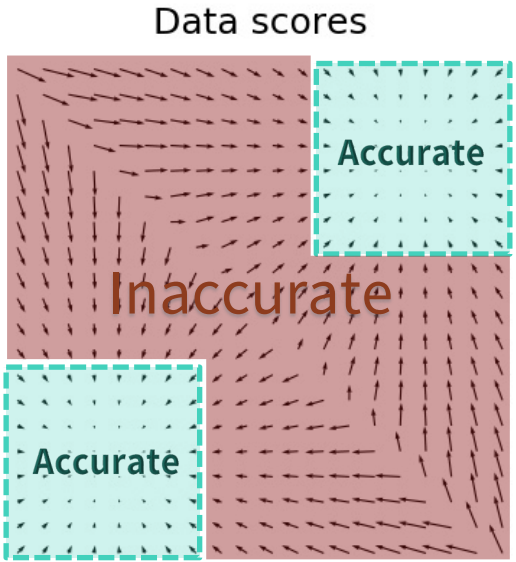


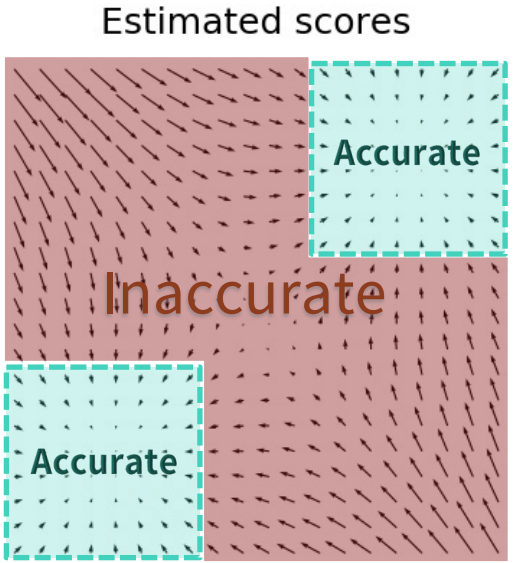
- Score estimation on CIFAR-10.



Challenge in low data density regions







$$\frac{1}{2}\mathbb{E}_{p_{\text{data}}(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})\|_2^2]$$

Langevin MCMC will have trouble exploring low density regions

Song and Ermon. "Generative Modeling by Estimating Gradients of the Data Distribution." NeurIPS 2019.

Pitfall 3: slow mixing of Langevin dynamics between data modes

- Suppose the data distribution has two modes with disjoint supports:

$$p_{\text{data}}(\mathbf{x}) = \pi p_1(\mathbf{x}) + (1-\pi)p_2(\mathbf{x})$$

$$\mathcal{A} \cap \mathcal{B} = \varnothing \qquad p_{\text{data}}(\mathbf{x}) = \begin{cases} \pi p_1(\mathbf{x}), & \mathbf{x} \in \mathcal{A} \\ (1-\pi)p_2(\mathbf{x}), & \mathbf{x} \in \mathcal{B} \end{cases}$$

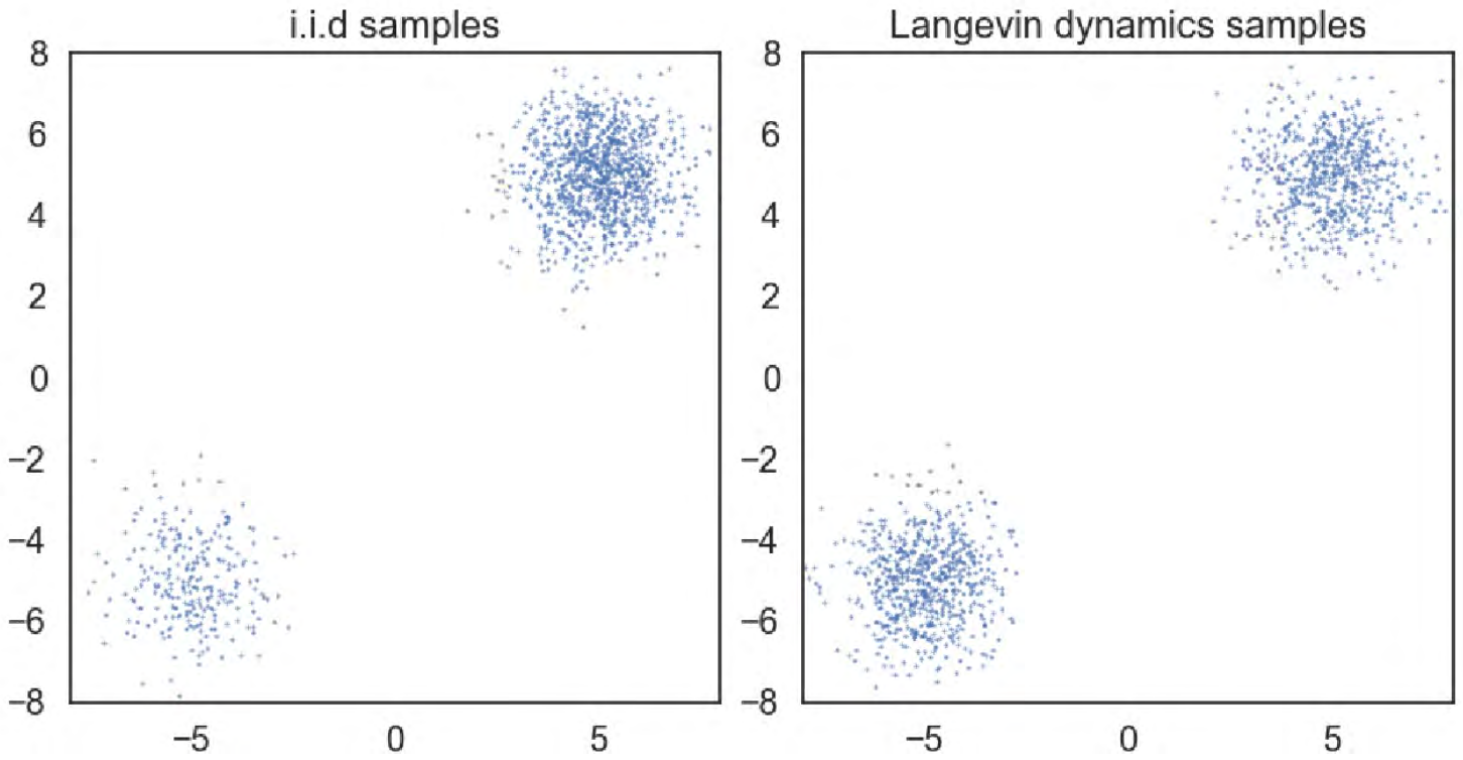
- Data score function:

$$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) = \begin{cases} \nabla_{\mathbf{x}}[\log \pi + \log p_1(\mathbf{x})], & \mathbf{x} \in \mathcal{A} \\ \nabla_{\mathbf{x}}[\log(1-\pi) + \log p_2(\mathbf{x})], & \mathbf{x} \in \mathcal{B} \end{cases}$$

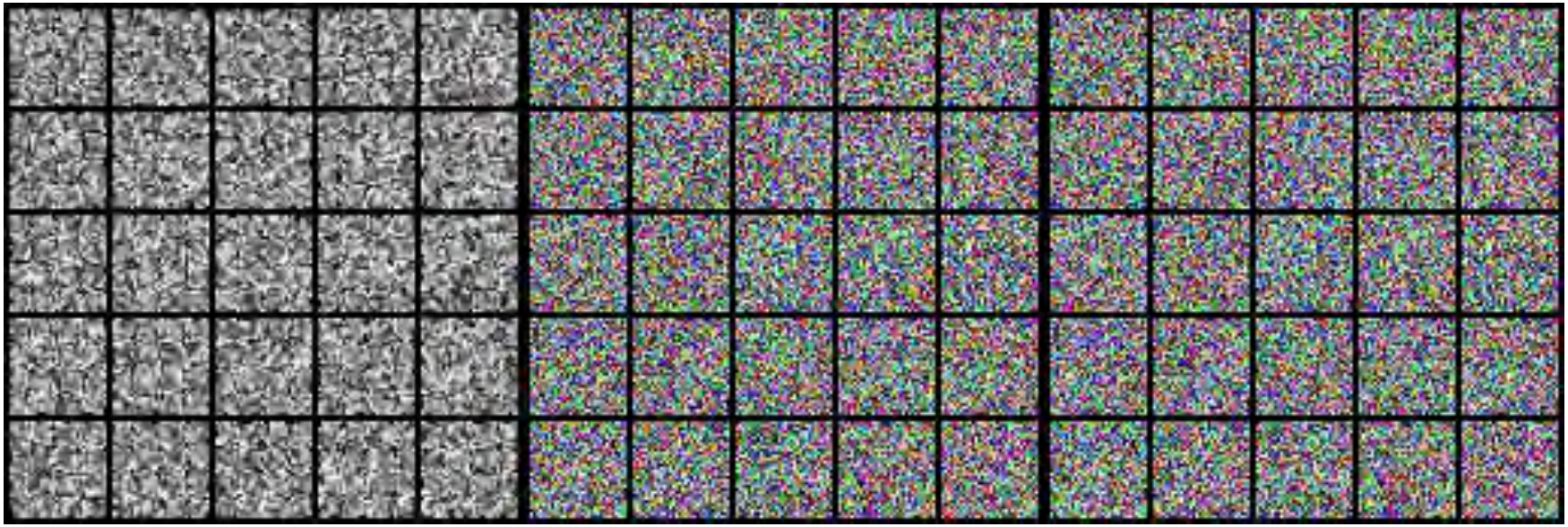
$$= \begin{cases} \nabla_{\mathbf{x}} \log p_1(\mathbf{x}), & \mathbf{x} \in \mathcal{A} \\ \nabla_{\mathbf{x}} \log p_2(\mathbf{x}), & \mathbf{x} \in \mathcal{B} \end{cases}$$

- The score function has no dependence on the mode weighting π at all!
- Langevin sampling will not reflect π

Pitfall 3: slow mixing of Langevin dynamics between data modes



After fixing these pitfalls (next lecture)

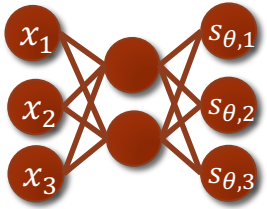


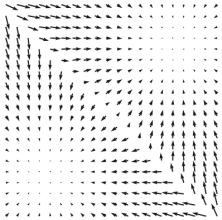
Song, Yang, and Stefano Ermon. "Generative Modeling by Estimating Gradients of the Data Distribution." NeurIPS 2019.

Score-based models

- A model that represents the score function

$$s_\theta(\mathbf{x})$$





- **Score estimation:** training the score-based model from datapoints
- Score matching

$$\frac{1}{2}E_{\mathbf{x}\sim p_{\text{data}}}[\|\nabla_{\mathbf{x}}\log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x})\|_2^2]$$
$$= \frac{1}{2}E_{\mathbf{x}\sim p_{\text{data}}}\left[\frac{1}{2}\|\boldsymbol{s}_\theta(\mathbf{x})\|_2^2 + \text{tr}(\nabla_{\mathbf{x}}\boldsymbol{s}_\theta(\mathbf{x}))\right] + \text{const.}$$

- Not scalable for deep score-based models and high dimensional data

Denoising score matching

$\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$

Data distribution

$q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$

Perturbation distribution/kernel

$\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}})$

Noise-perturbed data distribution

$$\frac{1}{2} E_{\tilde{\mathbf{x}} \sim p_{\text{data}}}[\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) \|_2^2]$$

$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})}[\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \|_2^2] + \text{const.}$$

$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[\left\| \boldsymbol{s}_\theta(\mathbf{x} + \sigma \mathbf{z}) + \frac{\mathbf{z}}{\sigma} \right\|_2^2 \right] + \text{const.}$$

- **Pros:**
  - Much more scalable than score matching
  - Reduces score estimation to a denoising task
- **Con:** estimates score of noise-perturbed data

$$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log q_\sigma(\mathbf{x}) \neq \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

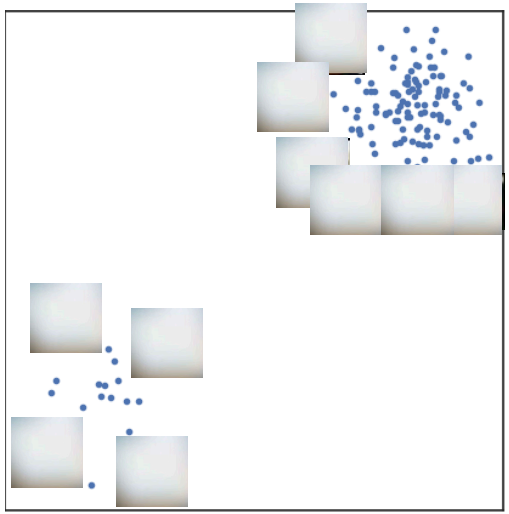
Sliced score matching

$$\frac{1}{2} E_{\mathbf{v}\sim p_{\mathbf{v}}} E_{\mathbf{x}\sim p_{\text{data}}} [(\mathbf{v}^\intercal \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \mathbf{v}^\intercal \boldsymbol{s}_\theta(\mathbf{x}))^2]$$

$$= E_{\mathbf{v}\sim p_{\mathbf{v}}, \mathbf{x}\sim p_{\text{data}}} \left[\mathbf{v}^\intercal \nabla_{\mathbf{x}} \boldsymbol{s}_\theta(\mathbf{x}) \mathbf{v} + \frac{1}{2} (\mathbf{v}^\intercal \boldsymbol{s}_\theta(\mathbf{x}))^2 \right] + \text{const.}$$

- Projection distribution $p_{\mathbf{v}}$ can be Gaussian or Rademacher
- **Pros:**
  - Much more scalable than score matching;
  - Estimates the true data score.
- **Con**: Slower than denoising score matching.

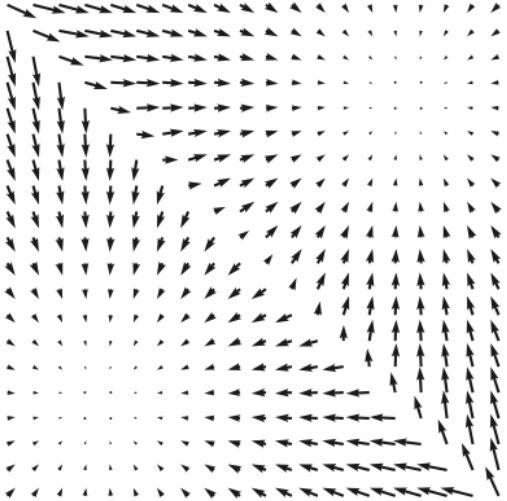
Score-based generative modeling



Data samples

$$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$$

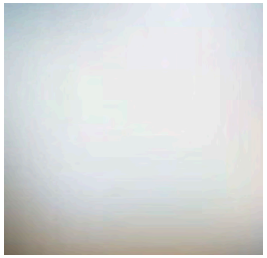
score matching



Scores

$$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

Langevin dynamics

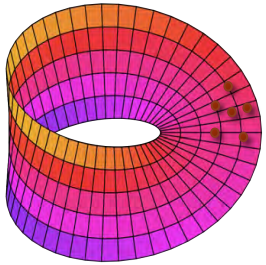


New samples

Pitfalls

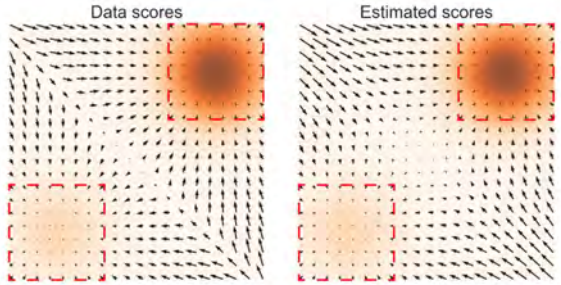
- Manifold hypothesis. Data score is undefined.

$$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

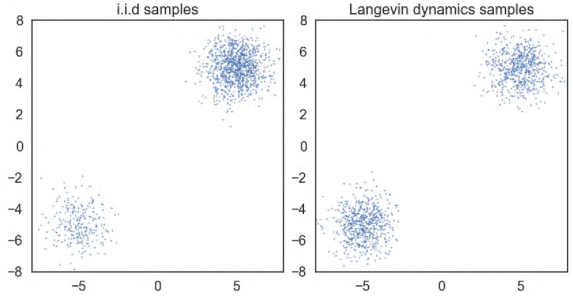


Data points

- Score matching fails in low data density regions

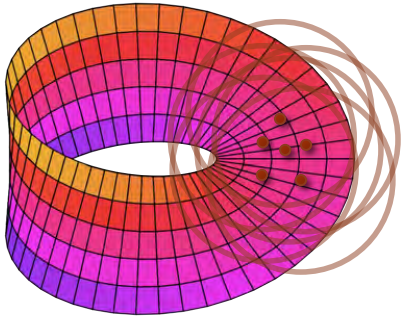


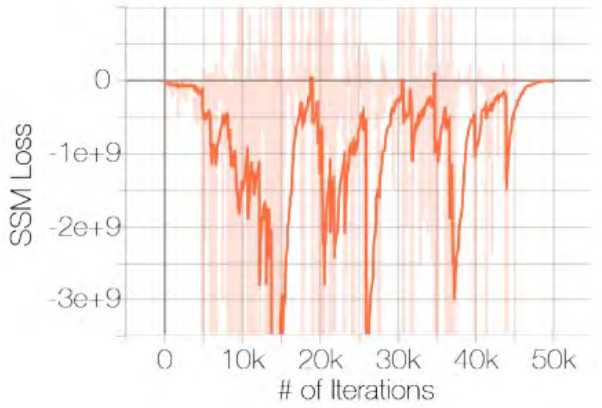
- Langevin dynamics converges very slowly



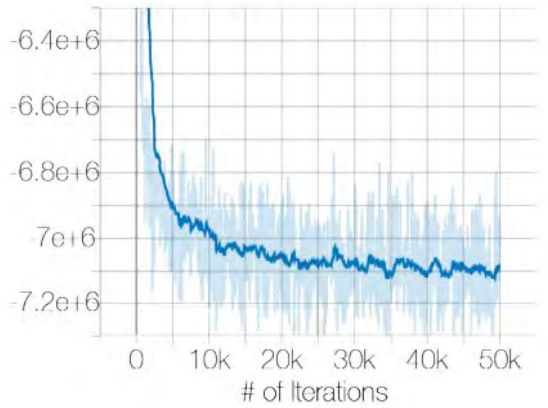
Gaussian perturbation

- The solution to all pitfalls: **Gaussian perturbation!**
- Manifold + noise
- Score matching on noisy data.





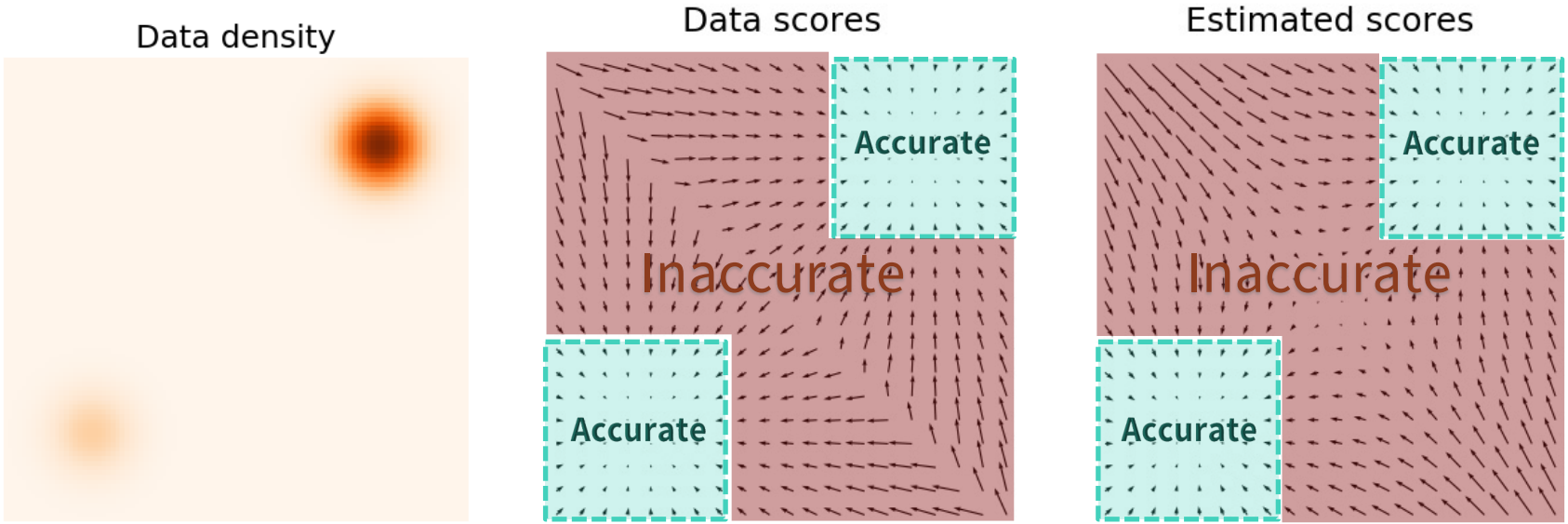
CIFAR-10



Noisy CIFAR-10

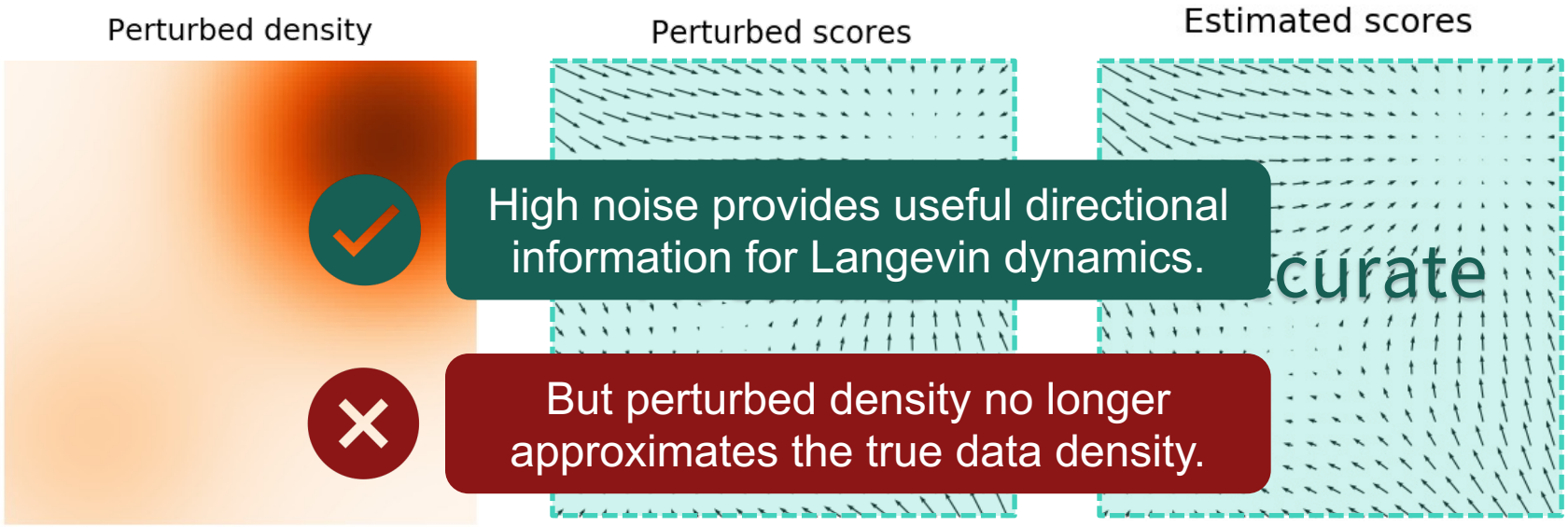
$\mathcal{N}(0; 0.0001)$

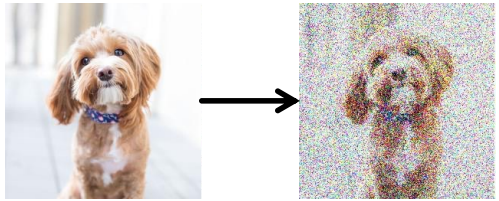
Challenge in low data density regions



Song and Ermon. “Generative Modeling by Estimating Gradients of the Data Distribution.” NeurIPS 2019.

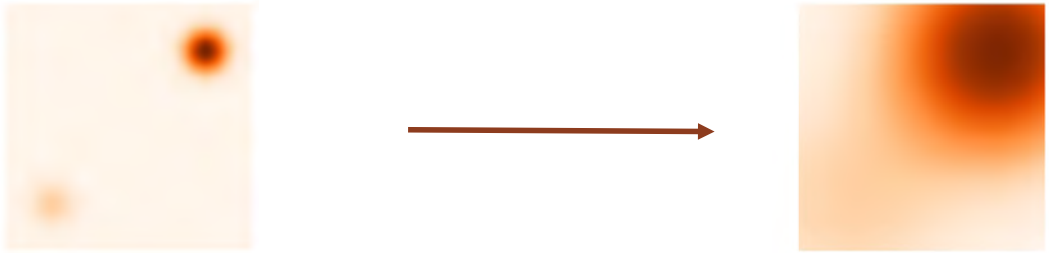
Improving score estimation by adding noise





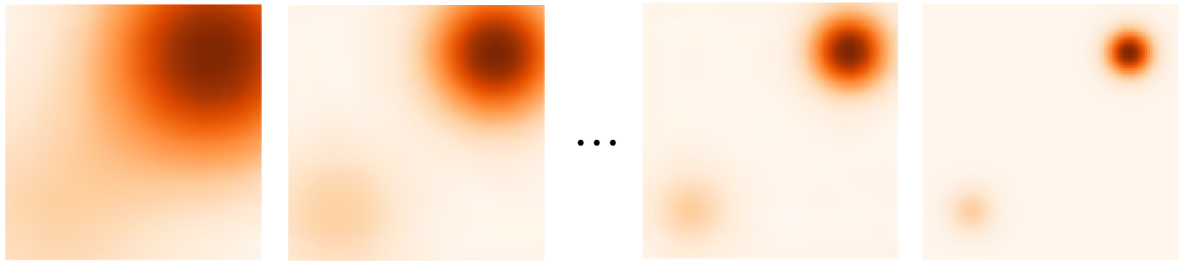
Multi-scale Noise Perturbation

- How much noise to add?

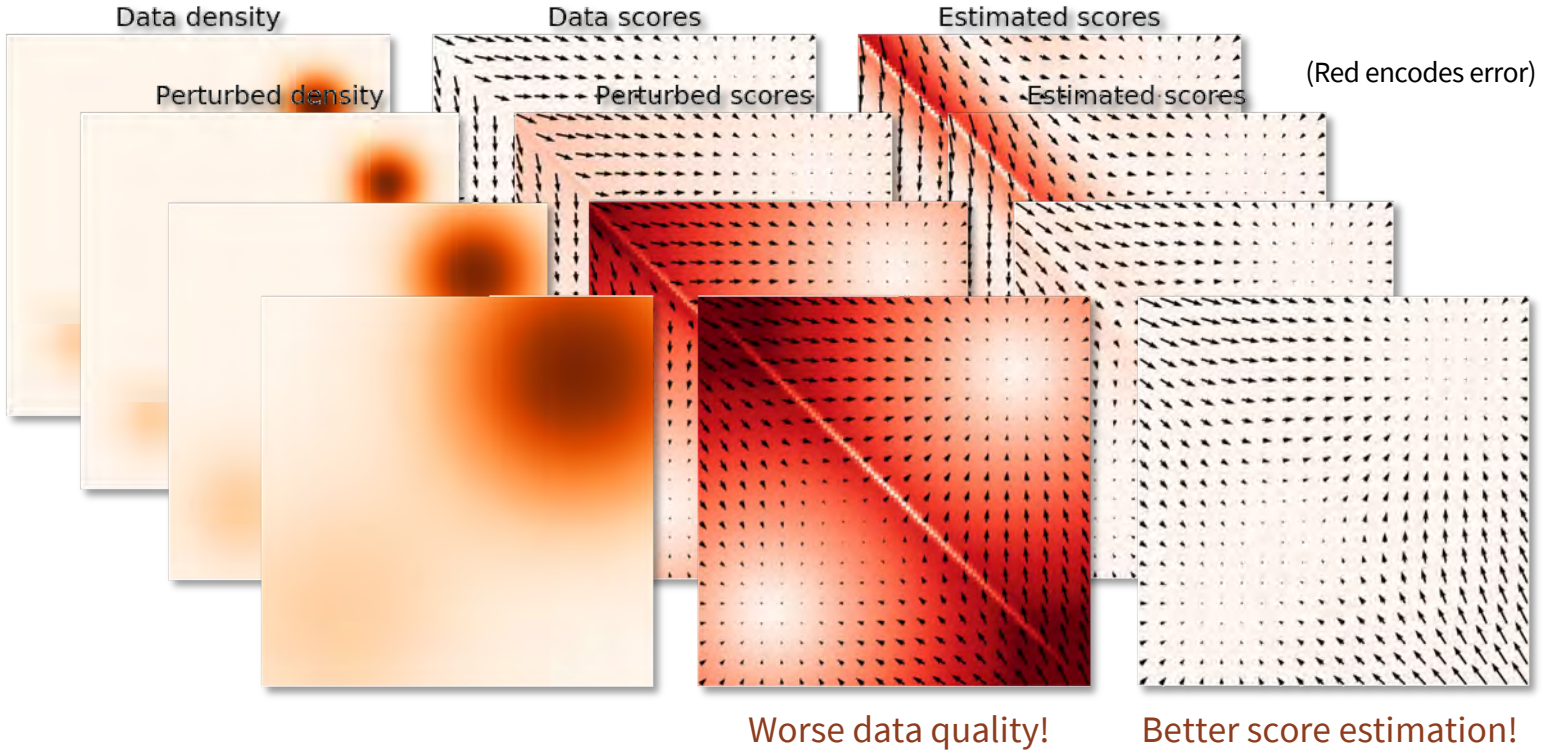


- Multi-scale noise perturbations.

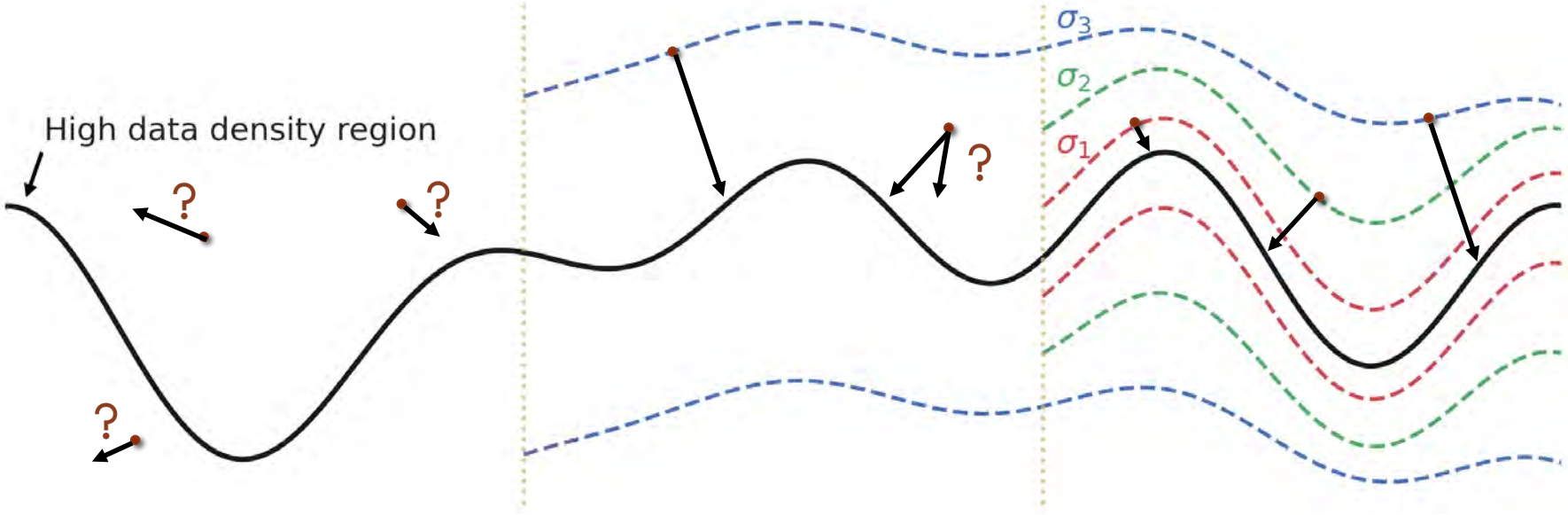
$$\sigma_1 > \sigma_2 > \cdots > \sigma_{L-1} > \sigma_L$$



Trading off Data Quality and Estimation Accuracy

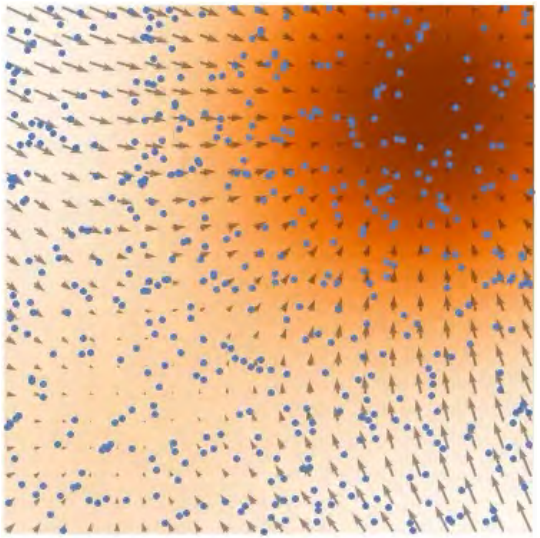


Using multiple noise scales

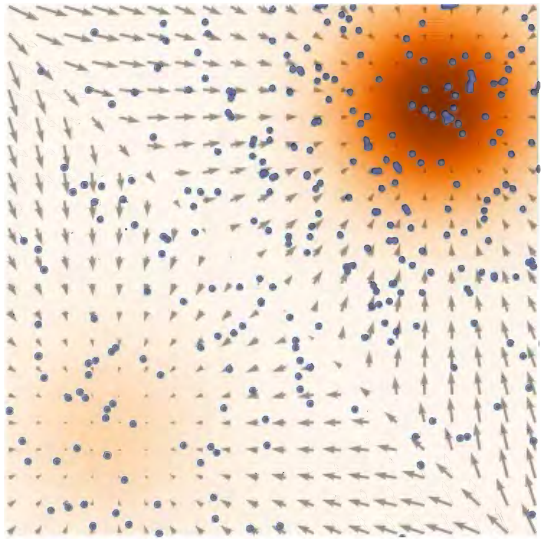


Annealed Langevin Dynamics: Joint Scores to Samples

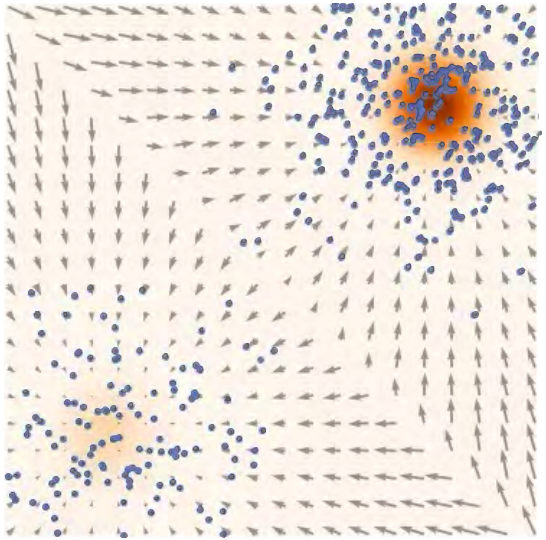
- Sample using $\sigma_1, \sigma_2, \cdots, \sigma_L$ sequentially with Langevin dynamics.
- Anneal down the noise level.
- Samples used as initialization for the next level.



σ_1

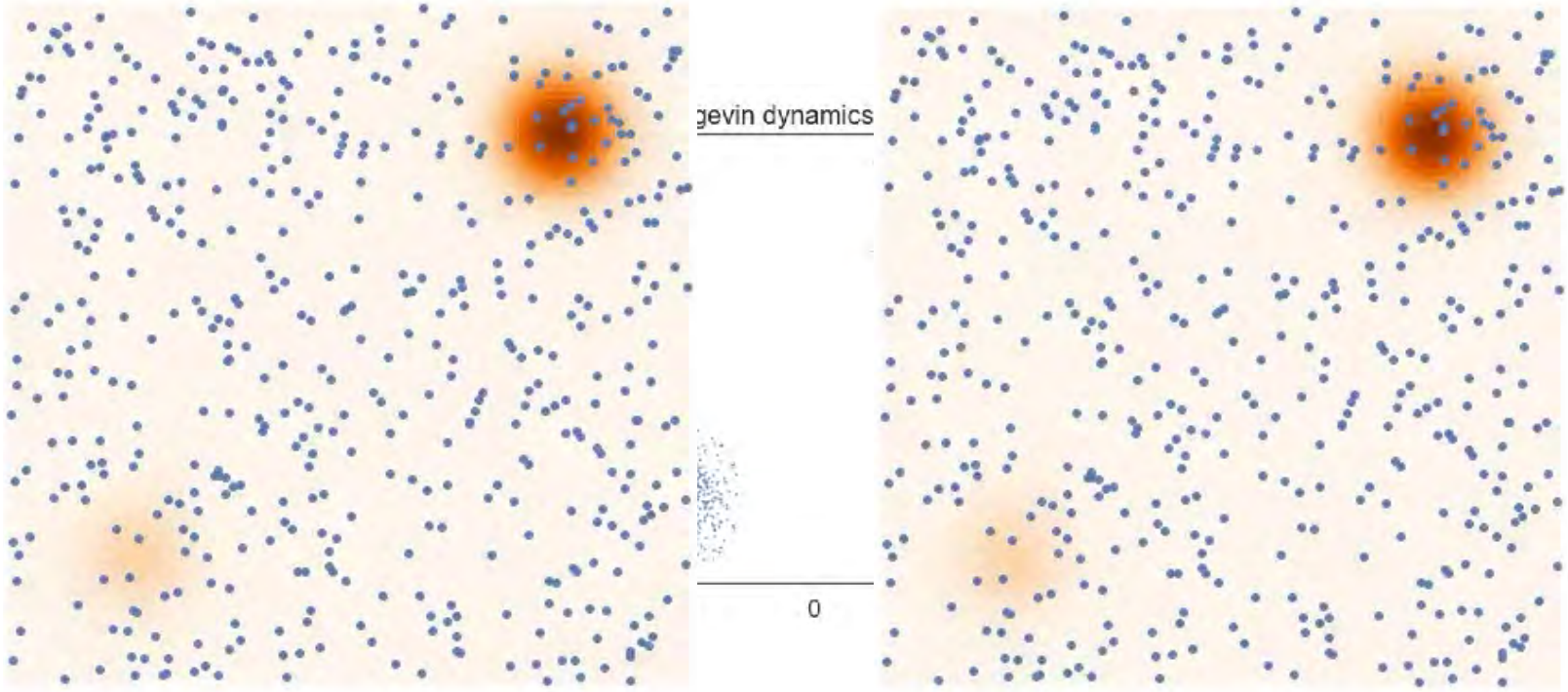


σ_2



σ_3

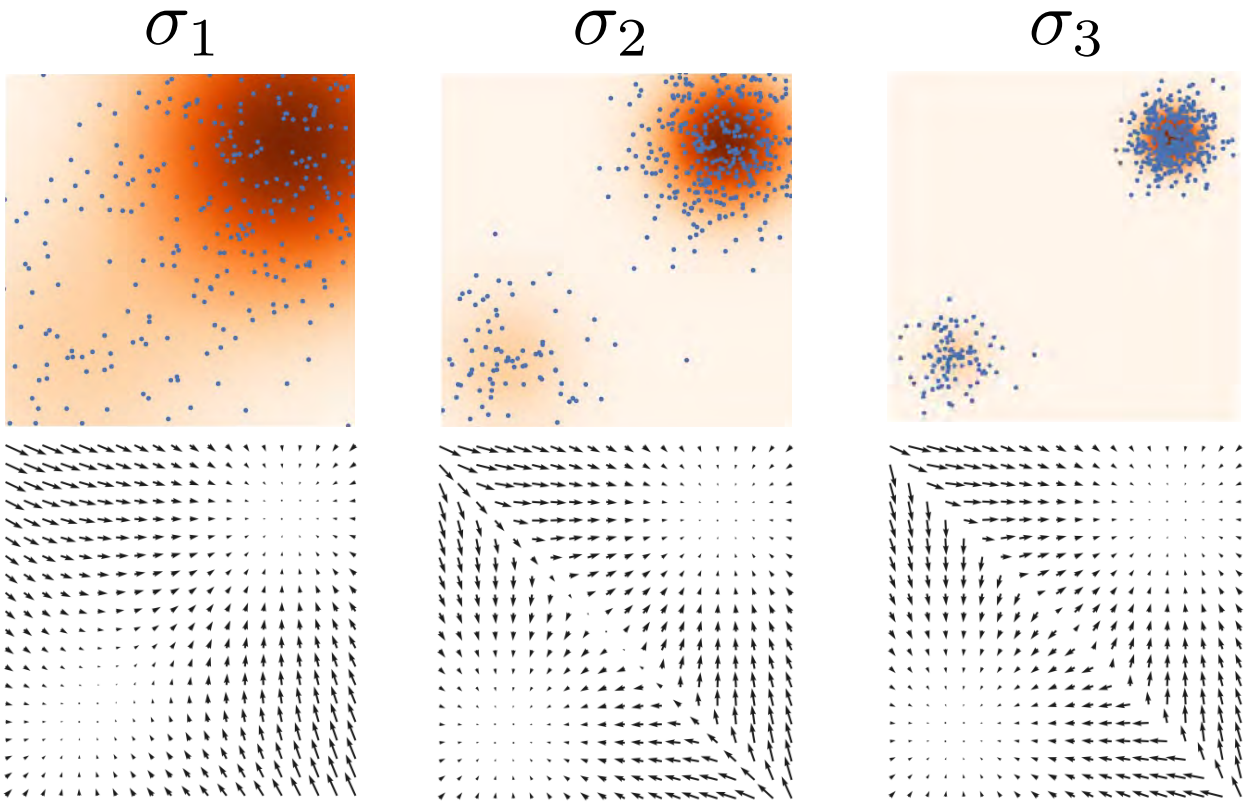
Comparison to the vanilla Langevin dynamics

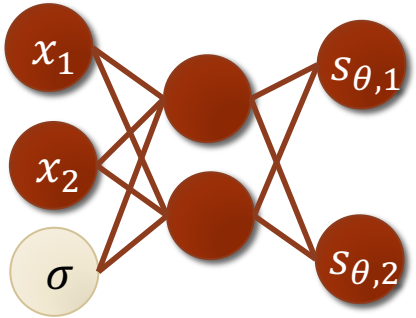


Langevin dynamics

Annealed Langevin dynamics

Joint Score Estimation via Noise Conditional Score Networks





Noise Conditional Score Network (NCSN)

Stanford University

Training noise conditional score networks

- Which score matching loss?
  - Sliced score matching?
  - Denoising score matching?
- Denoising score matching is naturally suitable, since the goal is to estimate the score of perturbed data distributions.
- Weighted combination of denoising score matching losses

$$\frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)E_{q_{\sigma_i}(\mathbf{x})}[\|\nabla_{\mathbf{x}}\log q_{\sigma_i}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x},\sigma_i)\|_2^2]$$

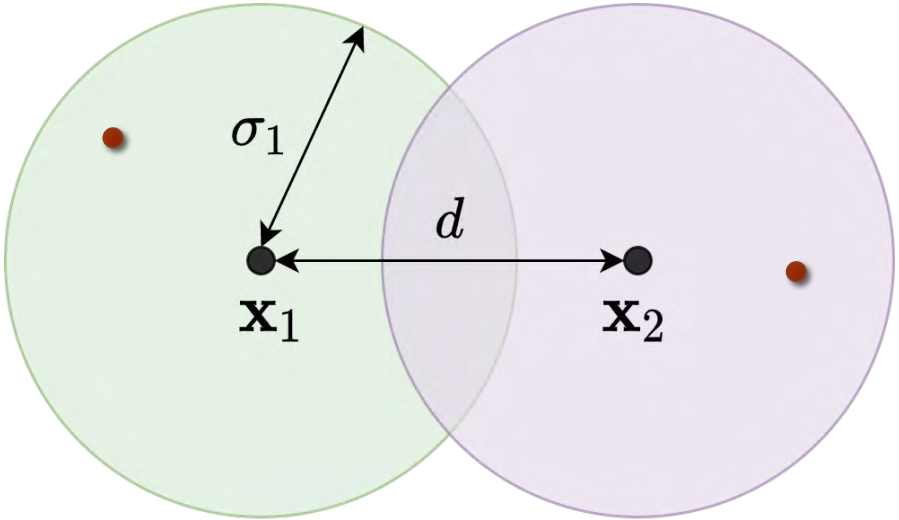
$$= \frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)E_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}[\|\nabla_{\tilde{\mathbf{x}}}\log q_{\sigma_i}(\tilde{\mathbf{x}}\mid\mathbf{x}) - \boldsymbol{s}_\theta(\tilde{\mathbf{x}},\sigma_i)\|_2^2] + \text{const.}$$

$$= \frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)E_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\left\|\boldsymbol{s}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i) + \frac{\mathbf{z}}{\sigma_i}\right\|_2^2\right] + \text{const.}$$

Choosing noise scales

- Maximum noise scale

$$\sigma_1 \approx \text{maximum pairwise distance between datapoints}$$



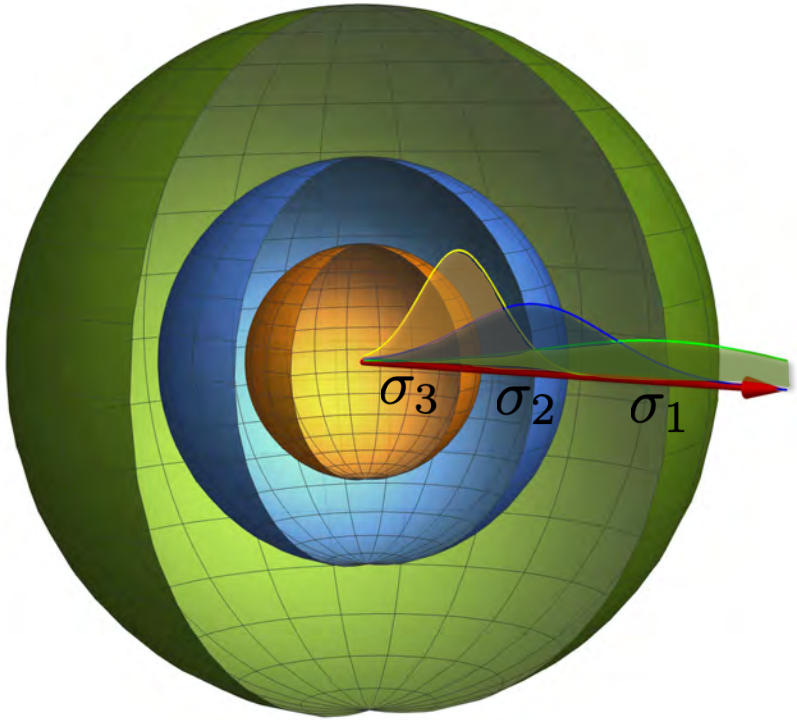
- Minimum noise scale: σ_L should be sufficiently small so that noise in final samples is negligible.

Choosing noise scales

- **Key intuition:** adjacent noise scales should have sufficient overlap to facilitate transitioning across noise scales in annealed Langevin dynamics.
- A geometric progression with sufficient length.

$$\sigma_1 > \sigma_2 > \sigma_3 > \cdots > \sigma_{L-1} > \sigma_L$$

$$\frac{\sigma_1}{\sigma_2} = \frac{\sigma_2}{\sigma_3} = \cdots = \frac{\sigma_{L-1}}{\sigma_L}$$



Choosing the weighting function

- Weighted combination of denoising score matching losses

$$\frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)E_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\left\|\boldsymbol{s}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\frac{\mathbf{z}}{\sigma_i}\right\|_2^2\right]$$

- How to choose the weighting function $\lambda : \mathbb{R}_{>0} \rightarrow \mathbb{R}_{>0}$?
- **Goal:** balancing different score matching losses in the sum → $\lambda(\sigma_i) = \sigma_i^2$

$$\begin{aligned}
&\frac{1}{L}\sum_{i=1}^{L}\sigma_i^2 E_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\left\|\boldsymbol{s}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\frac{\mathbf{z}}{\sigma_i}\right\|_2^2\right] \\
&= \frac{1}{L}\sum_{i=1}^{L} E_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\left\|\sigma_i\boldsymbol{s}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\mathbf{z}\right\|_2^2\right] \\
&= \frac{1}{L}\sum_{i=1}^{L} E_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\left\|\boldsymbol{\epsilon}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\mathbf{z}\right\|_2^2\right] \qquad \left[\ \boldsymbol{\epsilon}_\theta(\cdot,\sigma_i) := \sigma_i\boldsymbol{s}_\theta(\cdot,\sigma_i)\ \right]
\end{aligned}$$

Training noise conditional score networks

- Sample a mini-batch of datapoints $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}$
- Sample a mini-batch of noise scale indices

$$\{i_1, i_2, \cdots, i_n\} \sim \mathcal{U}\{1, 2, \cdots, L\}$$

- Sample a mini-batch of Gaussian noise $\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Estimate the weighted mixture of score matching losses

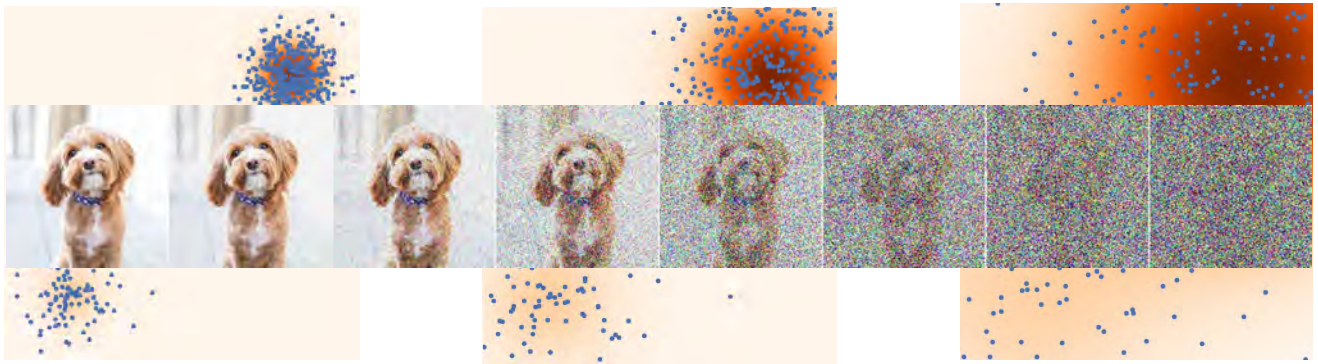
$$\frac{1}{n} \sum_{k=1}^{n} \left[\left\| \sigma_{i_k} \boldsymbol{s}_\theta(\mathbf{x}_k + \sigma_{i_k} \mathbf{z}_k, \sigma_{i_k}) + \mathbf{z}_k \right\|_2^2 \right]$$

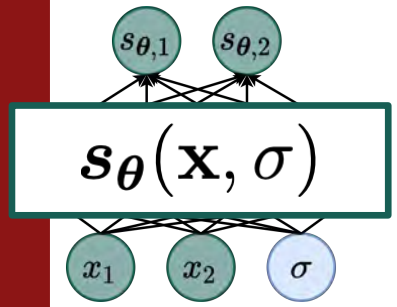
- Stochastic gradient descent
- As efficient as training one single non-conditional score-based model

Using multiple noise levels

$p_{\sigma_1}(\mathbf{x}) < p_{\sigma_2}(\mathbf{x}) < p_{\sigma_3}(\mathbf{x})$

Data





Noise Conditional Score Model

Positive weighting function

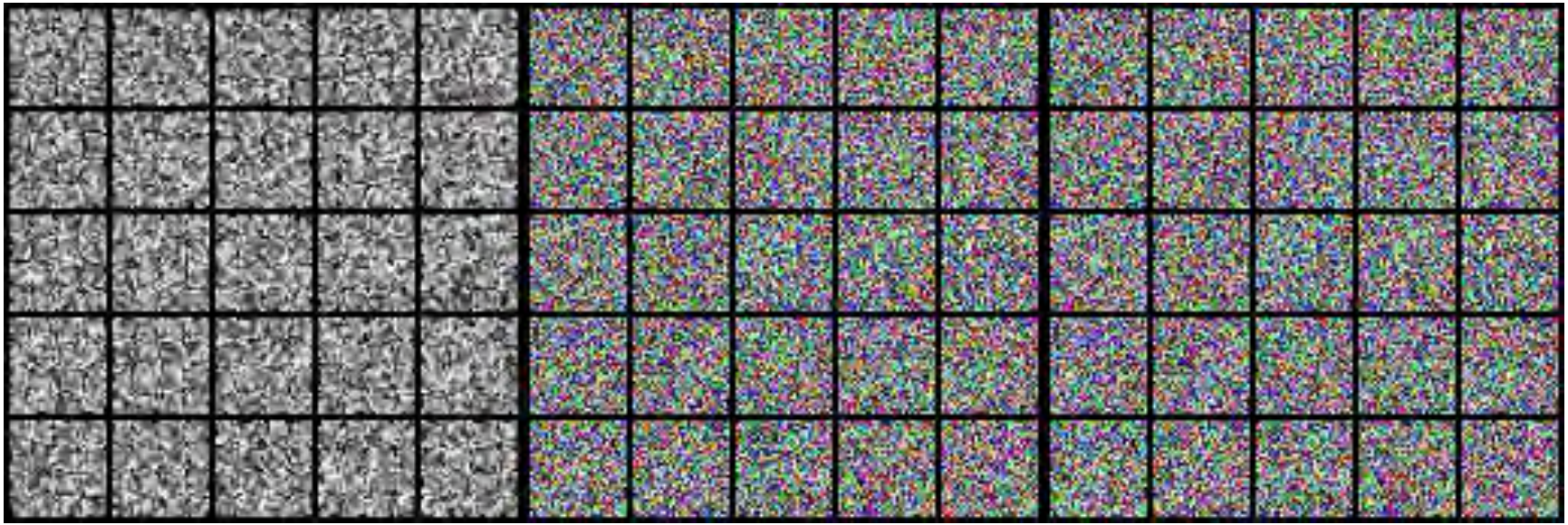
$$\frac{1}{N}\sum_{i=1}^{N} \lambda(\sigma_i)\, \mathbb{E}_{p_{\sigma_i}(\mathbf{x})}\left[\left\| \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x}) - s_\theta(\mathbf{x}, \sigma_i) \right\|_2^2\right]$$

Score matching loss

Annealed Langevin dynamics

$s_\theta(\mathbf{x}, \sigma_1)$ $s_\theta(\mathbf{x}, \sigma_2)$ $s_\theta(\mathbf{x}, \sigma_3)$

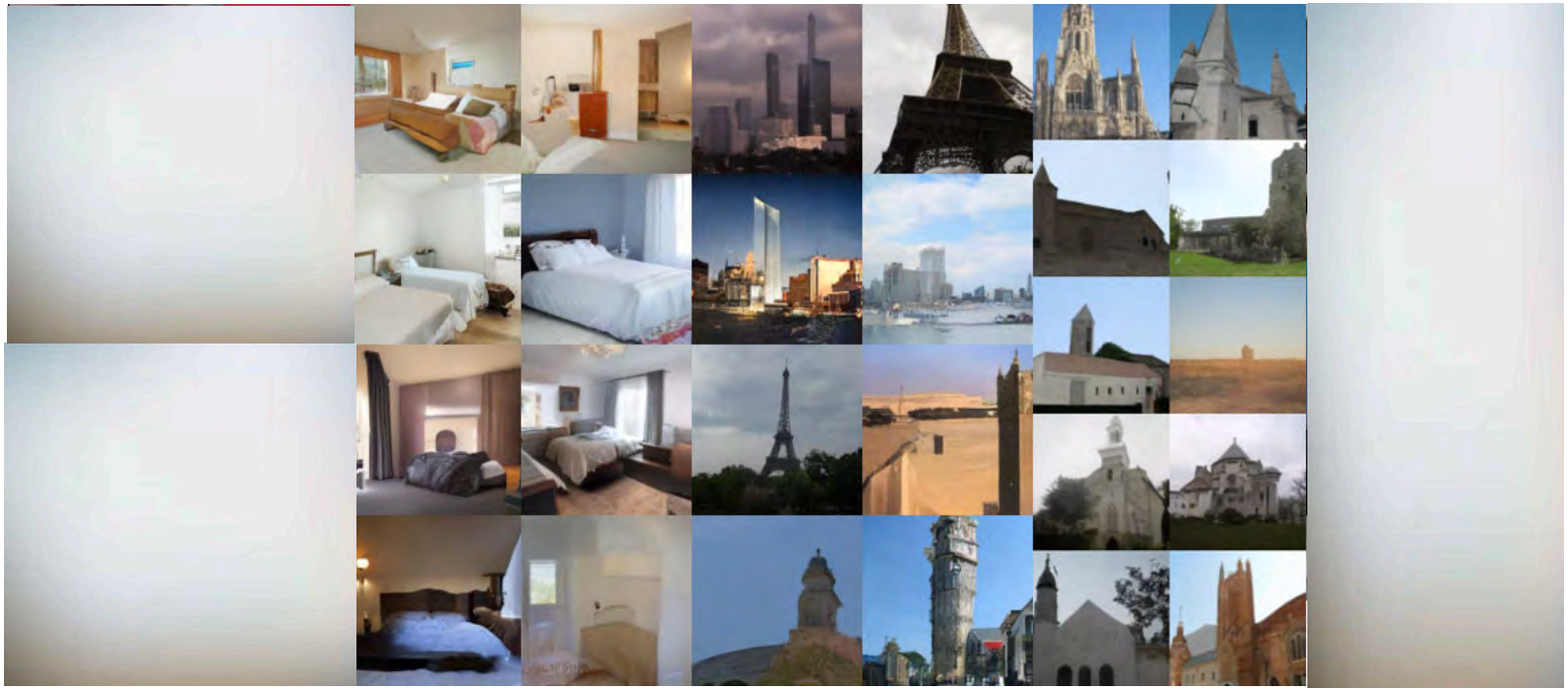
Experiments: Sampling



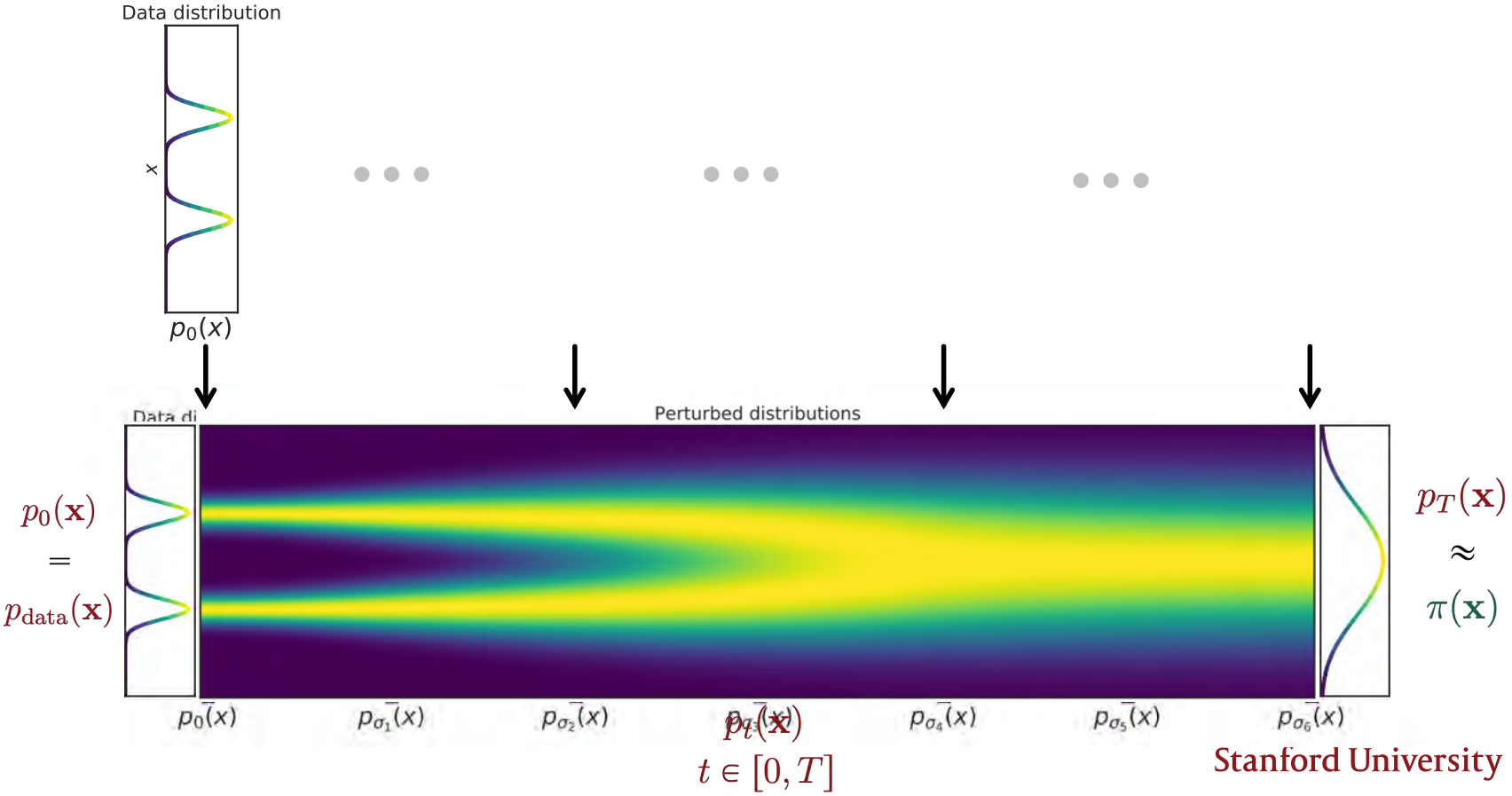
Experiments: sampling

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

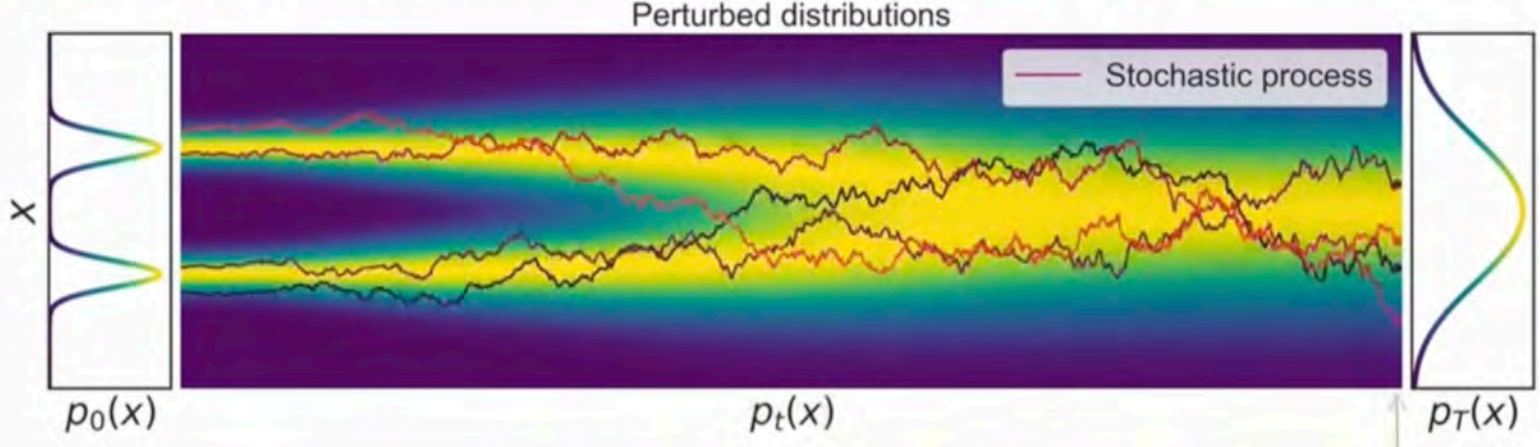
High Resolution Image Generation



Infinite noise levels



Perturbing data with stochastic processes



Stochastic process

$\{\mathbf{x}_t\}_{t\in[0,T]}$

Probability densities

$\{p_t(\mathbf{x})\}_{t\in[0,T]}$

Stochastic differential equation (SDE)

$$\mathrm{d}\mathbf{x}_t = \boldsymbol{f}(\mathbf{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}_t$$

Deterministic drift

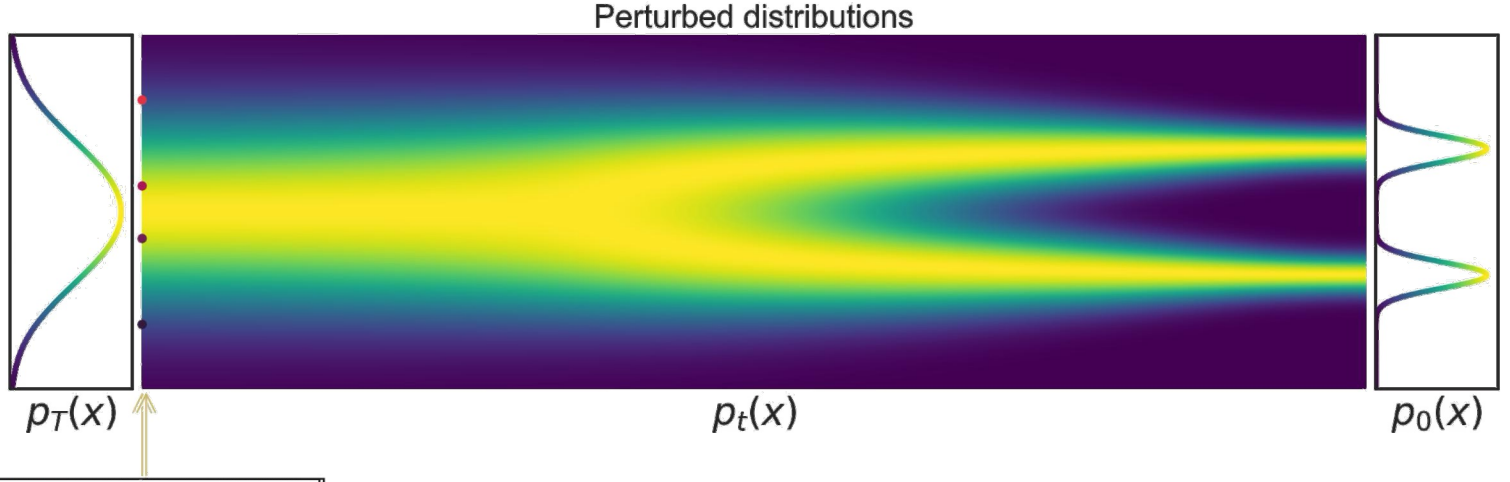
Infinitesimal noise

WLOG: Toy SDE

$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t$$

$p_T(\mathbf{x}) \approx \pi(\mathbf{x})$

Generation via reverse stochastic processes



$\pi(\mathbf{x}) \approx p_T(\mathbf{x})$

Forward SDE (t: 0→T)

$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t$$

Reverse SDE (t: T→0)

$$\mathrm{d}\mathbf{x}_t = -\sigma(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}\bar{\mathbf{w}}_t$$

Infinitesimal noise in the reverse time direction

Score function!

Score-based generative modeling via SDEs

- Time-dependent score-based model

$$\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

- Training:

$$\mathbb{E}_{t \in \mathcal{U}(0,T)}[\lambda(t) \mathbb{E}_{p_t(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2]]$$

- Reverse-time SDE

$$\mathrm{d}\mathbf{x} = -\sigma^2(t)\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\mathrm{d}t + \sigma(t)\mathrm{d}\bar{\mathbf{w}}$$

- Euler-Maruyama (analgous to Euler for ODEs)

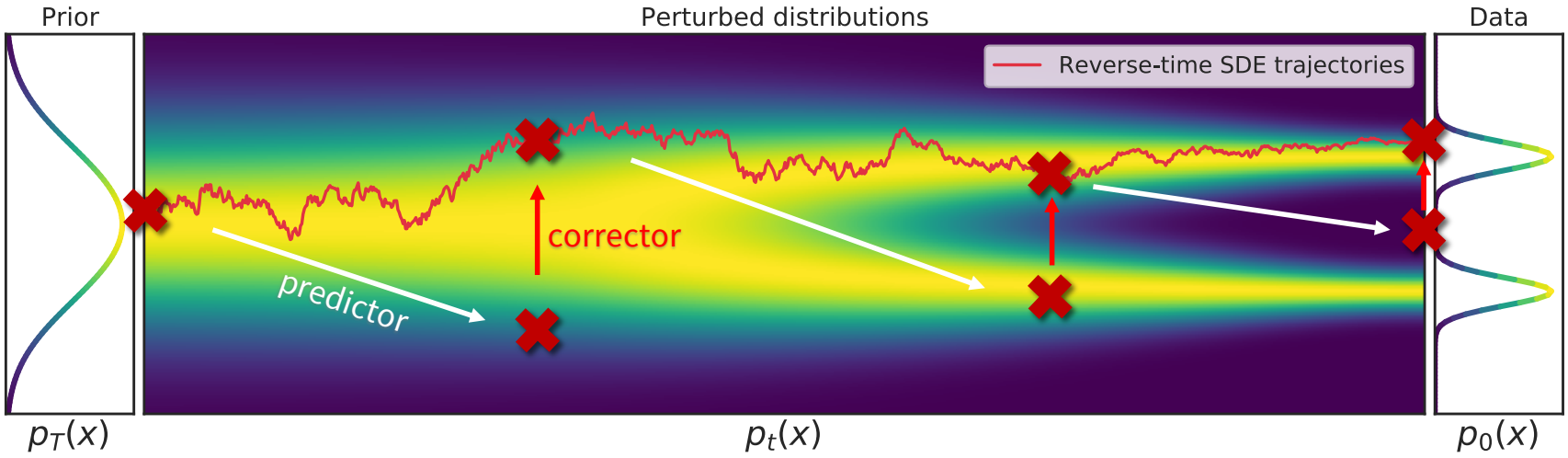
$$\mathbf{x} \leftarrow \mathbf{x} - \sigma(t)^2 \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\Delta t + \sigma(t)\ \mathbf{z} \quad (\mathbf{z} \sim \mathcal{N}(\mathbf{0}, |\Delta t|\ \mathbf{I}))$$

$$t \leftarrow t + \Delta t$$

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.

Predictor-Corrector sampling methods

- Predictor-Corrector sampling.
  - **Predictor:** Numerical SDE solver
  - **Corrector:** Score-based MCMC



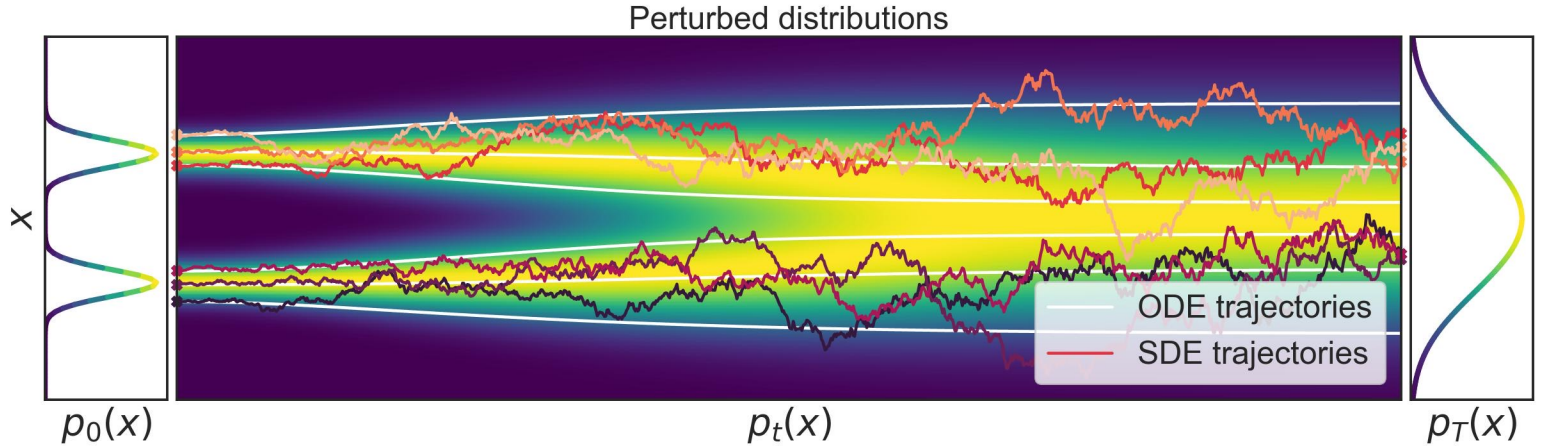
Results on predictor-corrector sampling

Model	FID↓	IS↑
Conditional		
BigGAN (Brock et al., 2018)	14.73	9.22
StyleGAN2-ADA (Karras et al., 2020a)	**2.42**	**10.14**
Unconditional		
StyleGAN2-ADA (Karras et al., 2020a)	2.92	9.83
NCSN (Song & Ermon, 2019)	25.32	$8.87 \pm .12$
NCSNv2 (Song & Ermon, 2020)	10.87	$8.40 \pm .07$
DDPM (Ho et al., 2020)	3.17	$9.46 \pm .11$
DDPM++	2.78	9.64
DDPM++ cont. (VP)	2.55	9.58
DDPM++ cont. (sub-VP)	2.61	9.56
DDPM++ cont. (deep, VP)	2.41	9.68
DDPM++ cont. (deep, sub-VP)	2.41	9.57
NCSN++	2.45	9.73
NCSN++ cont. (VE)	2.38	9.83
NCSN++ cont. (deep, VE)	**2.20**	**9.89**

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.

High-Fidelity Generation for 1024x1024 Images

Converting the SDE to an ODE



SDE **Ordinary differential equation (ODE)**

$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t \quad \Longleftrightarrow \quad \frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -\frac{1}{2}\sigma(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$$

Score function $\approx s_\theta(\mathbf{x}, t)$