

# Statistical Models & Computing Methods

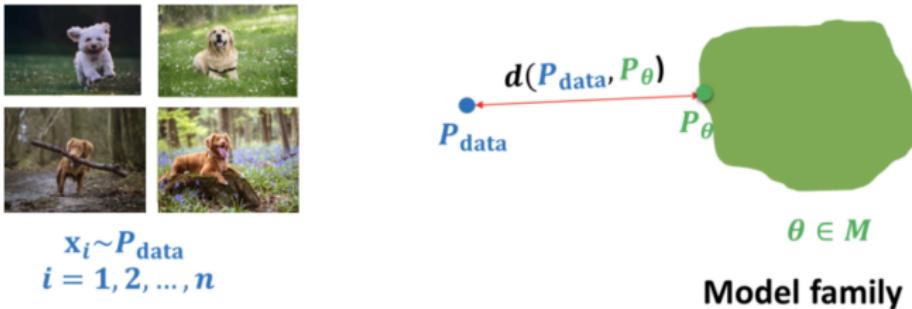
## Lecture 20: Energy-based and Score-based Generative Models



**Cheng Zhang**

School of Mathematical Sciences, Peking University

December 23, 2021



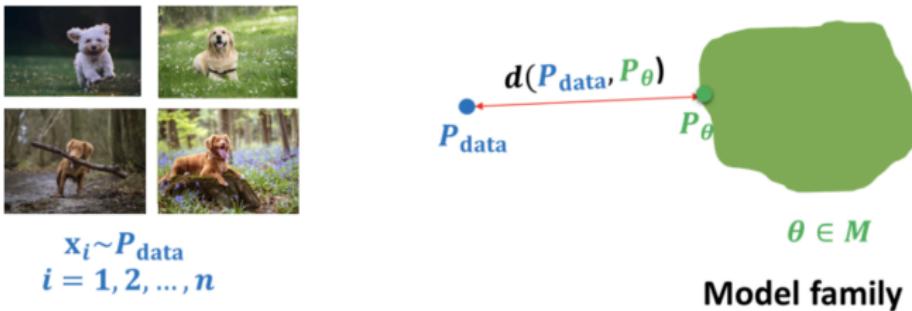
► Autoregressive Models:  $p_\theta(x) = \prod_{i=1}^n p_\theta(x_i | x_{<i})$

► Normalizing Flow Models:

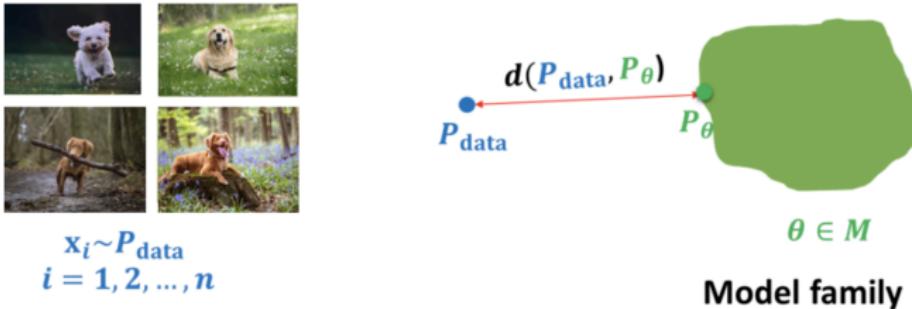
$$p_X(x; \theta) = p_Z(f_\theta^{-1}(x)) \left| \det \left( \frac{\partial f_\theta^{-1}(x)}{\partial x} \right) \right|$$

► Variational Autoencoders:  $p_\theta(x) = \int_z p_\theta(x, z) dz$

Cons: Model architectures are restricted.



- Generative Adversarial Networks (GANs).
  - $\min_{\theta} \max_{\phi} \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\phi}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\phi}(G_{\theta}(z)))$
  - Two sample tests. Can optimize  $f$ -divergences and the Wasserstein distance.
  - Very flexible model architectures. But likelihood is intractable, training is unstable, hard to evaluate, and has mode collapse issues.



Energy-based Models (EBMs).

- ▶ Very flexible model architectures.
- ▶ Stable training.
- ▶ Relatively high sample quality.
- ▶ Flexible composition.



Probability densities  $p(x)$  need to satisfy

- ▶ non-negative:  $p(x) \geq 0$ .
- ▶ sum-to-one:  $\sum_x p(x) = 1$  or  $\int p(x)dx = 1$  for continuous variables

Coming up with a non-negative function  $p_\theta(x)$  is not hard

- ▶  $p_\theta(x) = f_\theta(x)^2$
- ▶  $p_\theta(x) = \exp(f_\theta(x))$
- ▶  $p_\theta(x) = |f_\theta(x)|$

Sum to one is the key. Although many models allow analytical integration (e.g., autoregressive models, normalizing flows), what if the analytical integration is not available?



$$p_\theta(x) = \frac{\exp(f_\theta(x))}{Z(\theta)}, \quad Z(\theta) = \int \exp(f_\theta(x))dx$$

The normalizing constant  $Z(\theta)$  is also called the partition function. Why exponential (and not e.g.  $f_\theta(x)^2$ )?

- ▶ Want to capture very large variations in probability.  
log-probability is the natural scale we want to work with.  
Otherwise need highly non-smooth  $f_\theta$ .
- ▶ Exponential families. Many common distributions can be written in this form.
- ▶ These distributions arise under fairly general assumptions in statistical physics (maximum entropy, second law of thermodynamics).
  - ▶  $f_\theta(x)$  is called the energy, hence the name.
  - ▶ Intuitively, configurations  $x$  with low energy (high  $f_\theta(x)$ ) are more likely.

$$p_{\theta}(x) = \frac{\exp(f_{\theta}(x))}{Z(\theta)}, \quad Z(\theta) = \int \exp(f_{\theta}(x))dx$$

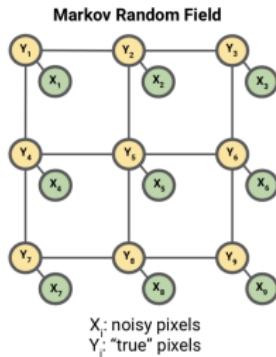
Pros:

- ▶ extreme flexibility. pretty much any function  $f_{\theta}(x)$  you want to use

Cons:

- ▶ Sampling from  $p_{\theta}(x)$  is hard
- ▶ Evaluating and optimizing likelihood  $p_{\theta}(x)$  is hard (learning is hard)
- ▶ No feature learning (but can add latent variables)

**Curse of dimensionality:** The fundamental issue is that computing  $Z(\theta)$  numerically (when no analytic solution is available) scales exponentially in the number of dimensions of  $x$ .



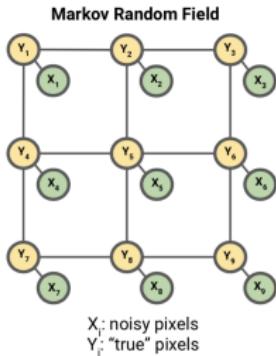
- ▶ There is a true image  $y \in \{0, 1\}^{3 \times 3}$ , and a corrupted image  $x \in \{0, 1\}^{3 \times 3}$ . We know  $x$ , and want to somehow recover  $y$ .
- ▶ We model the joint probability distribution  $p(y, x)$  as

$$p(y, x) \propto \exp \left( \sum_i \psi_i(x_i, y_i) + \sum_{i,j \in E} \psi_{i,j}(y_i, y_j) \right)$$



# Example: Ising Model

6/47



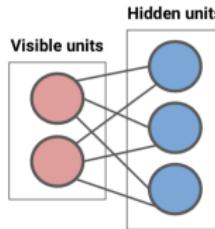
The energy is  $\sum_i \psi_i(x_i, y_i) + \sum_{i,j \in E} \psi_{i,j}(y_i, y_j)$

- ▶  $\psi_i(x_i, y_i)$ : the  $i$ -th corrupted pixel depends on the  $i$ -th original pixel
- ▶  $\psi_{i,j}(y_i, y_j)$ : neighboring pixels tend to have the same value

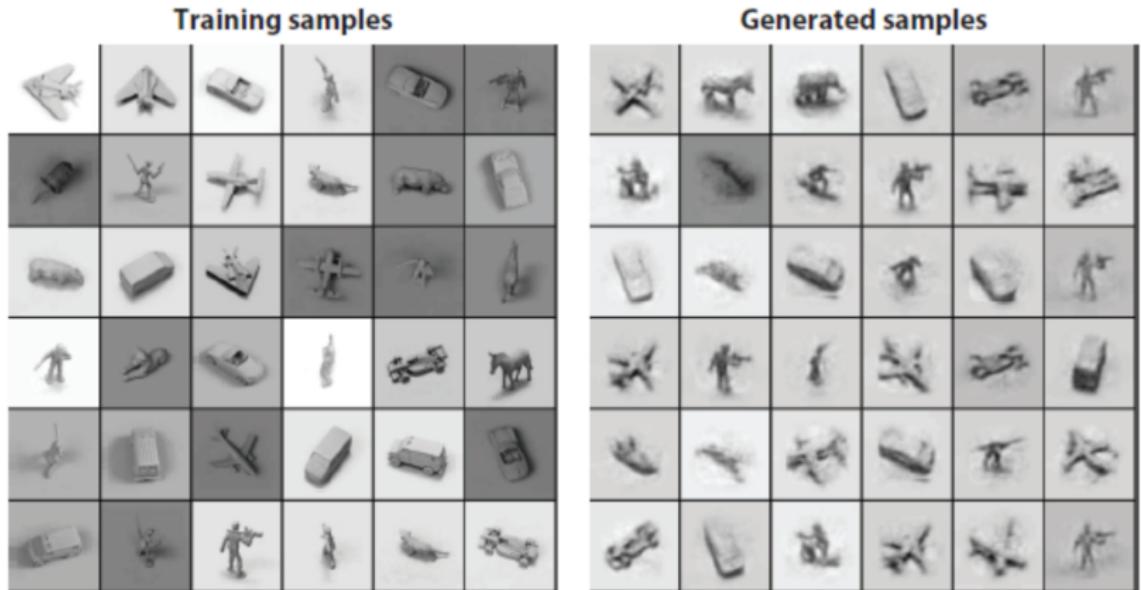
How did the original image  $y$  look like? Solution: maximize  $p(y|x)$ . Or equivalently, maximize  $p(y,x)$ .

- ▶ RBM: energy-based model with latent variables
- ▶ Two types of variables:
  - ▶  $x \in \{0, 1\}^n$  are visible variables (e.g., pixel values)
  - ▶  $z \in \{0, 1\}^m$  are latent ones
- ▶ The joint distribution is

$$p_{W,b,c}(x, z) \propto \exp(x^T W z + b^T x + c^T z)$$



- ▶ Restricted as there are no within-class connections.
- ▶ Can be stacked together to make deep RBMs (one of the first generative models).



Adapted from Salakhutdinov and Hinton, 2009.

- ▶ Learning by maximizing the likelihood function

$$\max_{\theta} \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\theta}(x) = \max_{\theta} (\mathbb{E}_{x \sim p_{\text{data}}} f_{\theta}(x) - \log Z(\theta))$$

- ▶ Gradient of log-likelihood:

$$\begin{aligned}\mathbb{E}_{x \sim p_{\text{data}}} \nabla_{\theta} f_{\theta}(x) - \nabla_{\theta} \log Z(\theta) &= \mathbb{E}_{x \sim p_{\text{data}}} \nabla_{\theta} f_{\theta}(x) - \frac{\nabla_{\theta} Z(\theta)}{Z(\theta)} \\ &= \mathbb{E}_{x \sim p_{\text{data}}} \nabla_{\theta} f_{\theta}(x) - \int \frac{\exp(f_{\theta}(x))}{Z(\theta)} \nabla_{\theta} f_{\theta}(x) dx \\ &= \mathbb{E}_{x \sim p_{\text{data}}} \nabla_{\theta} f_{\theta}(x) - \int p_{\theta}(x) \nabla_{\theta} f_{\theta}(x) dx \\ &= \mathbb{E}_{x \sim p_{\text{data}}} \nabla_{\theta} f_{\theta}(x) - \mathbb{E}_{x \sim p_{\theta}(x)} \nabla_{\theta} f_{\theta}(x)\end{aligned}$$

- ▶ **Contrastive Divergence**: sample  $x_{\text{sample}} \sim p_{\theta}$ , take gradient step on  $\nabla_{\theta} f_{\theta}(x_{\text{train}}) - \nabla_{\theta} f_{\theta}(x_{\text{sample}})$ .

$$p_\theta(x) = \frac{\exp(f_\theta(x))}{Z(\theta)}, \quad Z(\theta) = \int \exp(f_\theta(x))dx$$

- ▶ No direct way to sample like in autoregressive or flow models.
- ▶ Can use gradient-based MCMC methods, e.g., SGLD

$$x^{t+1} = x^t + \epsilon \nabla_x \log p_\theta(x^t) + \sqrt{2\epsilon} \eta^t, \quad \eta^t \sim \mathcal{N}(0, I)$$

- ▶ Note that for energy-based models

$$s_\theta(x) = \nabla_x \log p_\theta(x) = \nabla_x f_\theta(x) - \nabla_x \log Z(\theta) = \nabla_x f_\theta(x)$$

The score function does not depend on  $Z(\theta)$ !



Langevin sampling



Face samples

Adapted from Nijkamp et al. 2019

$$x^{t+1} = x^t + \epsilon \nabla_x \log p_\theta(x^t) + \sqrt{2\epsilon} \eta^t, \quad \eta^t \sim \mathcal{N}(0, I)$$

- ▶ MCMC sampling converges slowly in high dimensional spaces, and repetitive sampling for each training iteration would be expensive.
- ▶ Can we train without sampling?
- ▶ Note that to generate samples from an EBM, we only need the **score function**  $\nabla_x \log p_\theta(x)$ .
- ▶ Can we properly train the score function without sampling?



- ▶ A key observation: two distributions are identical iff their scores are the same

$$p(x) = q(x) \Leftrightarrow \nabla_x \log \tilde{p}(x) = \nabla_x \log \tilde{q}(x)$$

where  $\tilde{p}, \tilde{q}$  are the unnormalized densities of  $p, q$ .

- ▶ Match the scores of the data distribution and EBMs by minimizing

$$\begin{aligned} & \frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \|\nabla_x \log p_{\text{data}}(x) - s_\theta(x)\|^2 \\ &= \frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \|\nabla_x \log p_{\text{data}}(x) - \nabla_x f_\theta(x)\|^2 \end{aligned}$$

This is also known as **Fisher divergence**.



- ▶ Using integration by parts, we have

$$\begin{aligned} & \frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \|\nabla_x \log p_{\text{data}}(x) - \nabla_x f_\theta(x)\|^2 \\ &= \mathbb{E}_{x \sim p_{\text{data}}} \left( \text{Tr}(\nabla_x^2 f_\theta(x)) + \frac{1}{2} \|\nabla_x f_\theta(x)\|^2 \right) + \text{Const} \end{aligned}$$

- ▶ Sample a mini-batch of datapoints

$$\{x_1, x_2, \dots, x_n\} \sim p_{\text{data}}(x)$$

- ▶ Estimate the score matching loss with the empirical mean

$$\frac{1}{n} \sum_{i=1}^n \left( \frac{1}{2} \|\nabla_x f_\theta(x_i)\|^2 + \text{Tr}(\nabla_x^2 f_\theta(x_i)) \right)$$



- ▶ Minimize the score matching loss via stochastic gradient descent.
- ▶ No need to sample from the EBM!
- ▶ Note that computing the trace of Hessian  $\text{Tr}(\nabla_x^2 f_\theta(x))$  is in general very expensive for large models.
- ▶ Scalable score matching methods: denoising score matching (Vincent 2010) and sliced score matching (Song et al. 2019).



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

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

$$\max_{\theta} \mathbb{E}_{x \sim p_{\text{data}}} \log D_\theta(x) + \mathbb{E}_{x \sim p_n} \log(1 - D_\theta(x))$$

- ▶ Optimal discriminator

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



- If the discriminator is parameterized as

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

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

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

or equivalently,

$$p_\theta^*(x) = p_{\text{data}}(x)$$



- ▶ EBMs require  $Z(\theta)$  to be a normalizing constant, which is hard to satisfy.
- ▶ We can model  $Z(\theta)$  with an additional trainable parameter  $Z$  that relaxes the normalizing constraint

$$p_{\theta,Z}(x) = \frac{\exp(f_\theta(x))}{Z}$$

- ▶ With noise contrastive estimation (NCE), the optimal parameters  $\theta^*, Z^*$  satisfy

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

- ▶ The optimal parameter  $Z^*$  hence automatically satisfies the constraint as  $p_{\theta^*,Z^*}$  is a valid density function.

- ▶ Now we parameterize the discriminator  $D_{\theta,Z}(x)$  as

$$D_{\theta,Z}(x) = \frac{\exp(f_\theta(x))/Z}{\exp(f_\theta(x))/Z + p_n(x)} = \frac{\exp(f_\theta(x))}{\exp(f_\theta(x)) + p_n(x)Z}$$

- ▶ The training objective in NCE becomes

$$\begin{aligned} & \max_{\theta,Z} \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta,Z}(x) + \mathbb{E}_{x \sim p_n} \log(1 - D_{\theta,Z}(x)) \\ &= \max_{\theta,Z} \mathbb{E}_{x \sim p_{\text{data}}} (f_\theta(x) - \log(\exp(f_\theta(x)) + Zp_n(x))) \\ & \quad + \mathbb{E}_{x \sim p_n} (\log(Zp_n(x)) - \log(\exp(f_\theta(x)) + Zp_n(x))) \end{aligned}$$

- ▶ One can use the log-sum-exp trick for numerical stability.

- ▶ Sample a mini-batch of datapoints  $x_1, x_2, \dots, x_n \sim p_{\text{data}}(x)$ .
- ▶ Sample a mini-batch of noise samples  $y_1, y_2, \dots, y_n \sim p_n(y)$ .
- ▶ Estimate the NCE loss

$$\frac{1}{n} \sum_{i=1}^n \left( f_\theta(x_i) - \log(\exp(f_\theta(x_i)) + Z p_n(x_i)) \right. \\ \left. + \log Z + \log p_n(y_i) - \log(\exp(f_\theta(y_i)) + Z p_n(y_i)) \right)$$

- ▶ Stochastic gradient ascent.
- ▶ No need to sample from the EBM!

## Similarities:

- ▶ Both involve training a discriminator to perform binary classification with a cross-entropy loss.
- ▶ Both are likelihood-free.

## 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.

Intuition:

- ▶ We need to both evaluate the probability of  $p_n(x)$ , and sample from it efficiently.
- ▶ We hope to make the classification task as hard as possible, i.e.,  $p_n(x)$  should be close to  $p_{\text{data}}(x)$ .

Flow contrastive estimation:

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

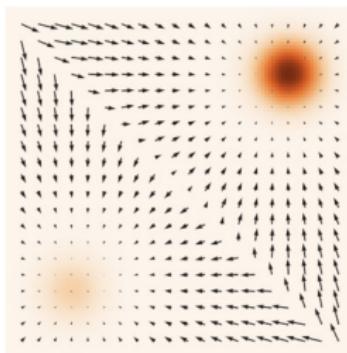
$$D_{\theta,Z,\phi}(x) = \frac{\exp(f_\theta(x))/Z}{\exp(f_\theta(x))/Z + p_{n,\phi}(x)} = \frac{\exp(f_\theta(x))}{\exp(f_\theta(x)) + Zp_{n,\phi}(x)}$$

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

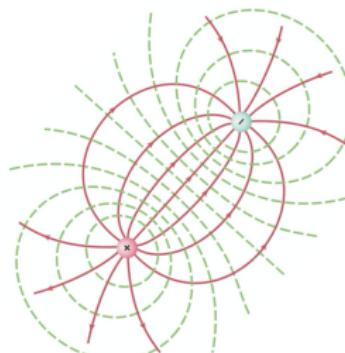
$$\min_{\phi} \max_{\theta, Z} \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta,Z,\phi}(x) + \mathbb{E}_{x \sim p_{n,\phi}} \log(1 - D_{\theta,Z,\phi}(x))$$

- When the pdf is differentiable, we can compute the gradient of a probability density, and use it to represent the distribution.

$$\text{Score function } \nabla_x \log p(x)$$



(pdf and score)



(Electrical potentials and fields)



- ▶ Given i.i.d. samples  $\{x_1, \dots, x_N\} \sim p(x)$
- ▶ We want to estimate the score  $\nabla_x \log p_{\text{data}}(x)$
- ▶ Score model: a learnable vector-valued function

$$s_\theta(x) : \mathbb{R}^D \rightarrow \mathbb{R}$$

- ▶ Goal:  $s_\theta(x) \approx \nabla_x \log p_{\text{data}}(x)$
- ▶ How to compare two vector fields of scores?



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

$$\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \|\nabla_x \log p_{\text{data}}(x) - s_\theta(x)\|^2$$

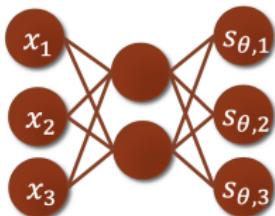
- ▶ **Score matching:**

$$\mathbb{E}_{x \sim p_{\text{data}}} \left( \frac{1}{2} \|s_\theta(x)\|^2 + \text{Tr}(\nabla_x s_\theta(x)) \right)$$

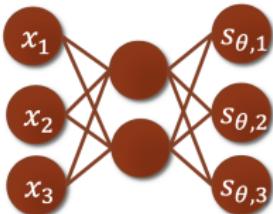
- ▶ Requirements:
  - ▶ The score model must be efficient to evaluated.
  - ▶ Do we need the score model to be a proper score function?



- We can use deep neural networks for more expressive score models



- We can use deep neural networks for more expressive score models



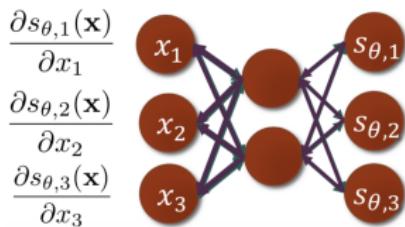
- However,  $\text{Tr}(\nabla_x s_\theta(x))$  can be a problem.

**$O(D)$  Backprops!**

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



- We can use deep neural networks for more expressive score models

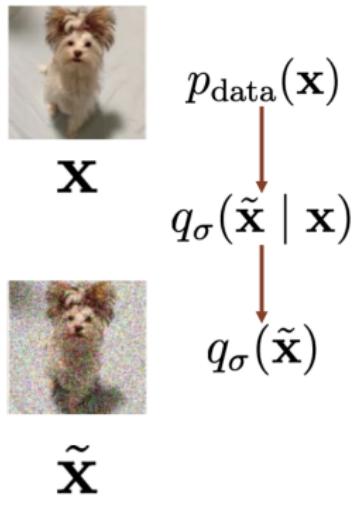


- However,  $\text{Tr}(\nabla_{\mathbf{x}} s_{\theta}(\mathbf{x}))$  can be a problem.

**$O(D)$  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) used a noise-perturbed data distribution

$$\begin{aligned} & \frac{1}{2} \mathbb{E}_{\tilde{x} \sim q_\sigma} \|\nabla_{\tilde{x}} \log q_\sigma(\tilde{x}) - s_\theta(\tilde{x})\|^2 \\ &= \frac{1}{2} \int q_\sigma(\tilde{x}) \|\nabla_{\tilde{x}} \log q_\sigma(\tilde{x}) - s_\theta(\tilde{x})\|^2 d\tilde{x} \\ &= \frac{1}{2} \int q_\sigma(\tilde{x}) \|s_\theta(\tilde{x})\|^2 d\tilde{x} \\ &\quad - \int q_\sigma(\tilde{x}) \nabla_{\tilde{x}} \log q_\sigma(\tilde{x})^T s_\theta(\tilde{x}) d\tilde{x} + \text{Const} \end{aligned}$$

- The second term can be rewritten as

$$\begin{aligned} - \int q_\sigma(\tilde{x}) \nabla_{\tilde{x}} \log q_\sigma(\tilde{x})^T s_\theta(\tilde{x}) d\tilde{x} &= - \int \nabla_{\tilde{x}} q_\sigma(\tilde{x})^T s_\theta(\tilde{x}) d\tilde{x} \\ &= - \int \nabla_{\tilde{x}} \left( \int p_{\text{data}}(x) q_\sigma(\tilde{x}|x) dx \right)^T s_\theta(\tilde{x}) d\tilde{x} \\ &= - \int \left( \int p_{\text{data}}(x) \nabla_{\tilde{x}} q_\sigma(\tilde{x}|x) dx \right)^T s_\theta(\tilde{x}) d\tilde{x} \\ &= - \int \int p_{\text{data}}(x) q_\sigma(\tilde{x}|x) \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x)^T s_\theta(\tilde{x}) dx d\tilde{x} \\ &= - \mathbb{E}_{x \sim p_{\text{data}}(x), \tilde{x} \sim q_\sigma(\tilde{x}|x)} \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x)^T s_\theta(\tilde{x}) \end{aligned}$$

- ▶ Plug it back we have

$$\begin{aligned} & \frac{1}{2} \mathbb{E}_{\tilde{x} \sim q_\sigma} \| \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}) - s_\theta(\tilde{x}) \|^2 \\ &= \frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}(x), \tilde{x} \sim q_\sigma(\tilde{x}|x)} \| s_\theta(\tilde{x}) - \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x) \|^2 + \text{Const} \end{aligned}$$

- ▶ The noise score  $\nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x)$  is easy to compute. For example, when use Gaussian noise  $q_\sigma(\tilde{x}|x) = \mathcal{N}(\tilde{x}|x, \sigma^2 I)$ , the score is

$$\nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x) = -\frac{\tilde{x} - x}{\sigma^2}$$

- ▶ Pros: efficient to optimize even for very high dimensional data, and useful for optimal denoising.
- ▶ Cons: cannot estimate the score of clean data (noise-free)

- ▶ Sample a minibatch of datapoints  $\{x_1, \dots, x_n\} \sim p_{\text{data}}(x)$ .
- ▶ Sample a minibatch of perturbed datapoints

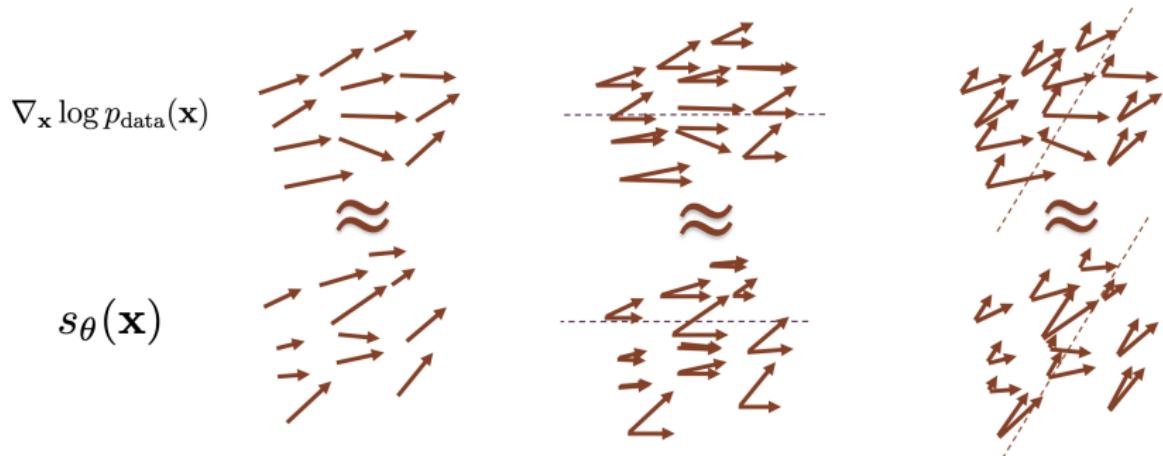
$$\tilde{x}_i \sim q_\sigma(\tilde{x}_i|x_i), \quad i = 1, 2, \dots, n$$

- ▶ Estimate the denoising score matching loss with empirical means

$$\frac{1}{2n} \sum_{i=1}^n \|s_\theta(\tilde{x}) - \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}_i|x)\|^2$$

- ▶ Stochastic gradient descent
- ▶ Need to choose a very small  $\sigma$ ! However, the loss variance would also increase drastically as  $\sigma \rightarrow 0$ !

- ▶ One dimensional problems should be easier.
- ▶ Consider projections onto random directions.
- ▶ Sliced score matching (Song et al 2019).



- ▶ Objective: Sliced Fisher Divergence

$$\frac{1}{2} \mathbb{E}_{v \sim p_v} \mathbb{E}_{x \sim p_{\text{data}}} (v^T \nabla_x \log p_{\text{data}}(x) - v^T s_\theta(x))^2$$

- ▶ Similarly, we can do integration by parts

$$\mathbb{E}_{v \sim p_v} \mathbb{E}_{x \sim p_{\text{data}}} \left( v^T \nabla_x s_\theta(x) v + \frac{1}{2} (v^T s_\theta(x))^2 \right)$$

- ▶ Computing Jacobian-vector products is scalable

$$v^T \nabla_x s_\theta(x) v = v^T \nabla_x (s_\theta(x)^T v)$$

This only requires one backpropagation!

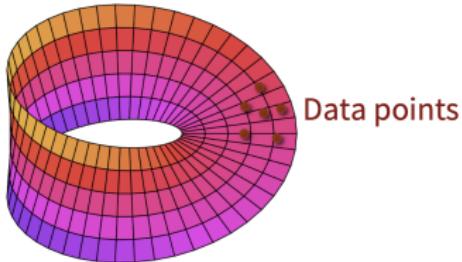


- ▶ Sample a minibatch of datapoints  $\{x_1, \dots, x_n\} \sim p_{\text{data}}(x)$
- ▶ Sample a minibatch of projection directions  $\{v_i \sim p_v\}_{i=1}^n$
- ▶ Estimate the sliced score matching loss with empirical means

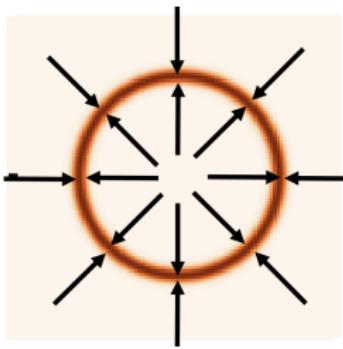
$$\frac{1}{n} \sum_{i=1}^n \left( v_i^T \nabla_x s_\theta(x_i) v_i + \frac{1}{2} (v_i^T s_\theta(x_i))^2 \right)$$

- ▶ The perturbation distribution is typically Gaussian or Rademacher. When  $\mathbb{E} vv^T = I$ , this is equivalent to the **Hutchinson's trick**.
- ▶ Can use  $\|s_\theta(x)\|^2$  instead of  $(v^T s_\theta(x))^2$  to reduce variance.
- ▶ Can use more projections per datapoint to boost performance.

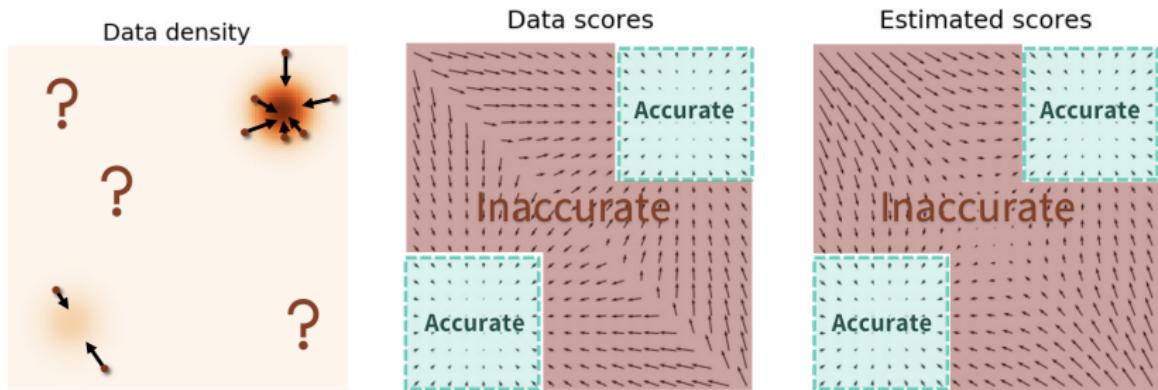
- ▶ Datapoints would lie on a lower dimensional manifold.



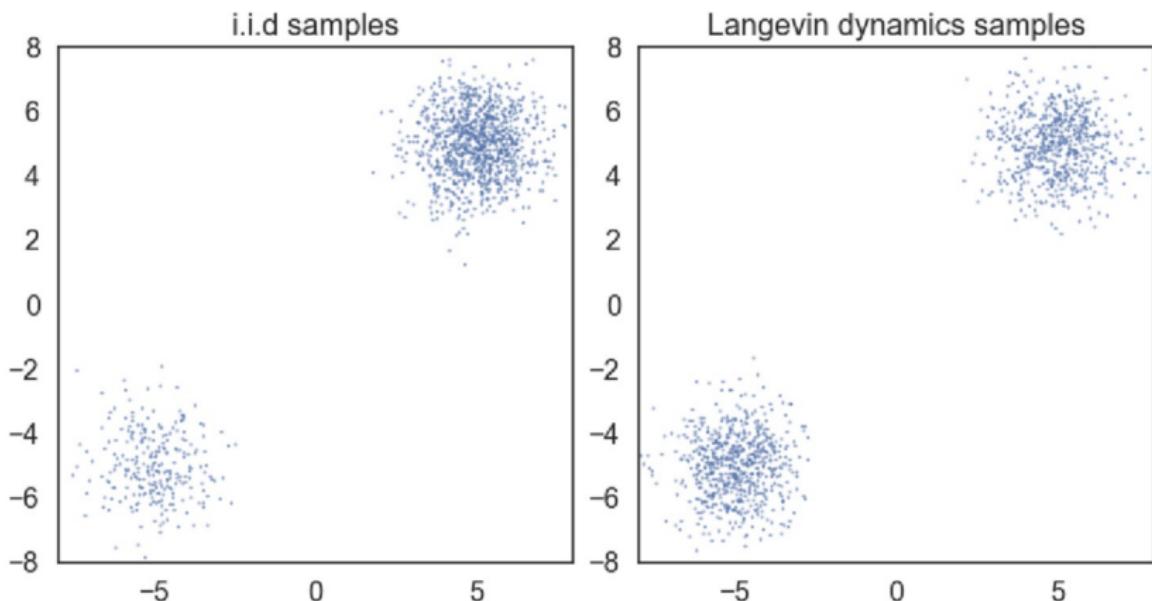
- ▶ Data score hence would be undefined.



$$\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \| \nabla_x \log p_{\text{data}}(x) - s_\theta(x) \|^2$$

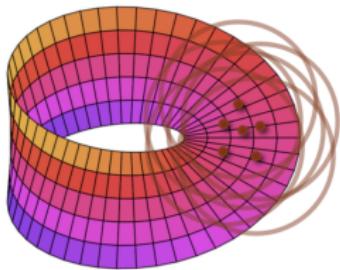


- ▶ Poor score estimation in low data density regions.
- ▶ Langevin MCMC will also have trouble exploring low density regions.

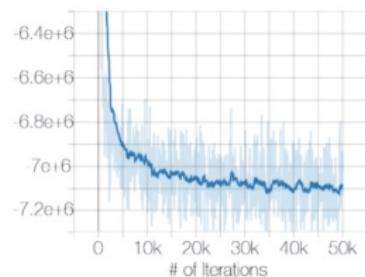
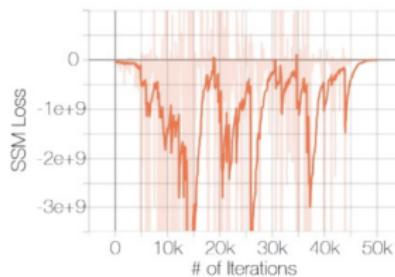


Adapted from Song et al 2019.

- ▶ The solution to all pitfalls: Gaussian perturbation!
- ▶ Inflate the flat manifold with noise.

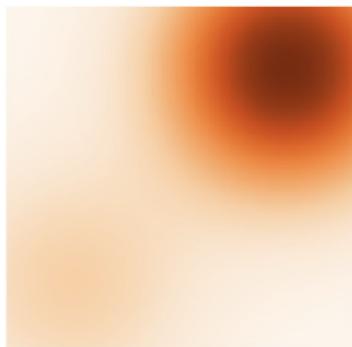


- ▶ Score matching on noise data

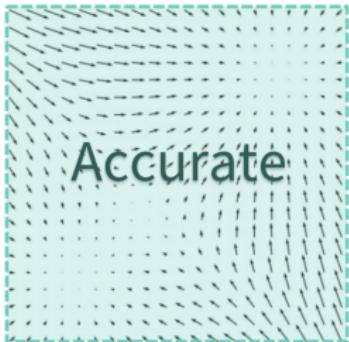


$$\frac{1}{2} \mathbb{E}_{\tilde{x} \sim q_\sigma} \| \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}) - s_\theta(\tilde{x}) \|^2$$

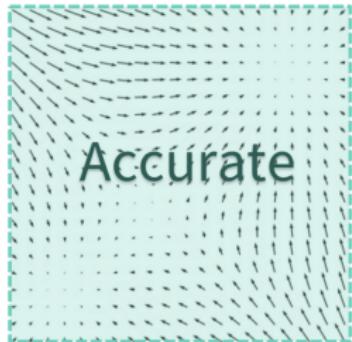
Perturbed density



Perturbed scores



Estimated scores



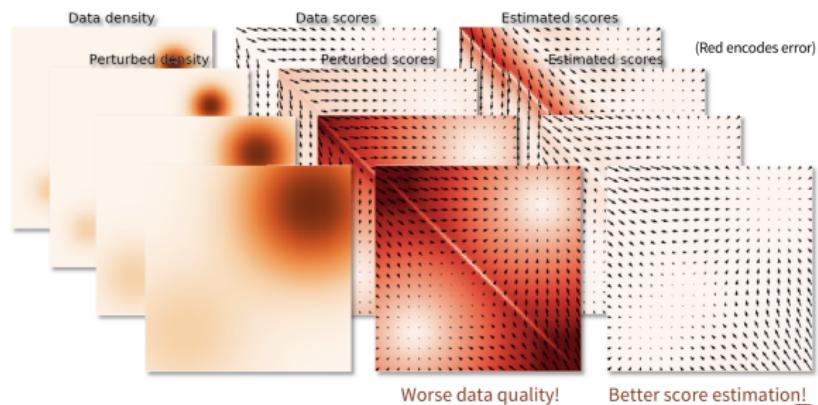
- ▶ Noisy score can provide useful directional information for Langevin MCMC.

- ▶ Multi-scale noise perturbations.

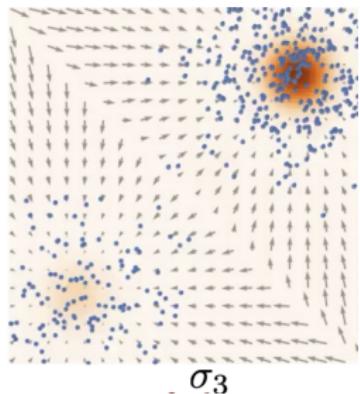
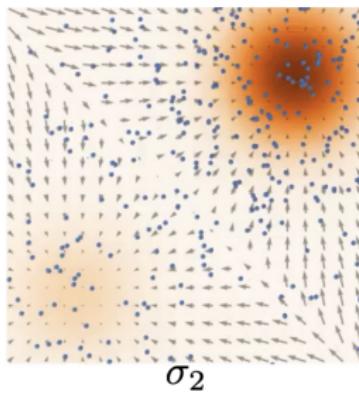
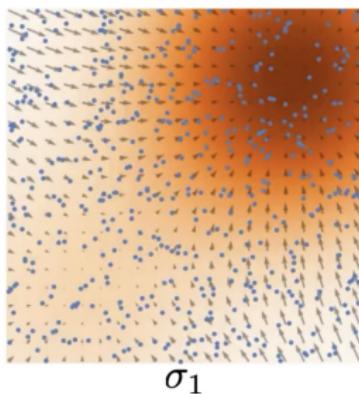
$$\sigma_1 > \sigma_2 > \dots > \sigma_{L-1} > \sigma_L$$



- ▶ Trading off data quality and estimator accuracy



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



---

**Algorithm 1** Annealed Langevin dynamics.

---

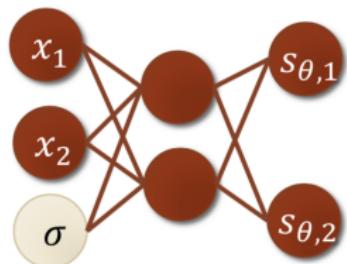
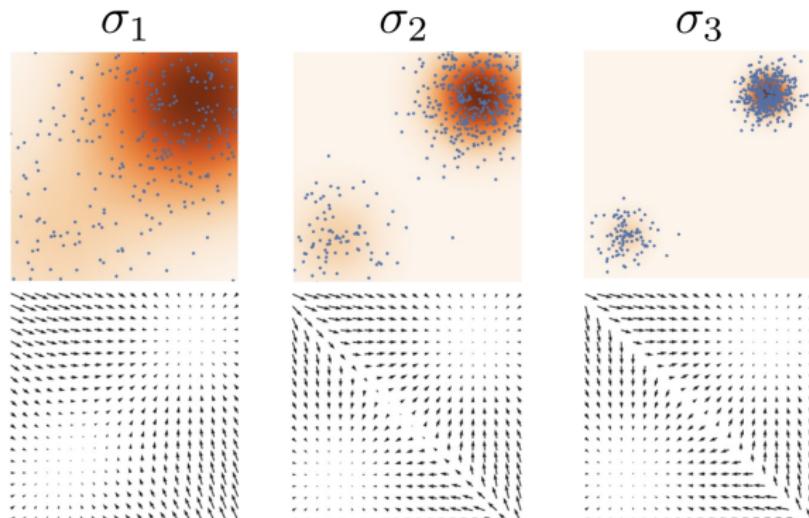
**Require:**  $\{\sigma_i\}_{i=1}^L, \epsilon, T$ .

- 1: Initialize  $\tilde{\mathbf{x}}_0$
- 2: **for**  $i \leftarrow 1$  to  $L$  **do**
- 3:      $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$        ▷  $\alpha_i$  is the step size.
- 4:     **for**  $t \leftarrow 1$  to  $T$  **do**
- 5:         Draw  $\mathbf{z}_t \sim \mathcal{N}(0, I)$
- 6:          $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_\theta(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i} \mathbf{z}_t$
- 7:     **end for**
- 8:      $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
- 9: **end for**
- return  $\tilde{\mathbf{x}}_T$

---



- ▶ Learning score functions jointly with noise conditional score networks!



Noise Conditional  
Score Network  
(NCSN)

- ▶ As the goal is to estimate the score of perturbed data distributions, we can use denoising score matching for training.
- ▶ Assign different weights to combine denoising score matching losses for different noise levels.

$$\begin{aligned} & \frac{1}{L} \sum_{i=1}^L \lambda(\sigma_i) \mathbb{E}_{\tilde{x} \sim q_{\sigma_i}(\tilde{x})} \| \nabla_{\tilde{x}} \log q_{\sigma_i}(\tilde{x}) - s_{\theta}(\tilde{x}, \sigma_i) \|^2 \\ = & \frac{1}{L} \sum_{i=1}^L \lambda(\sigma_i) \mathbb{E}_{x \sim p_{\text{data}}, \tilde{x} \sim q_{\sigma_i}(\tilde{x}|x)} \| \nabla_x \log q_{\sigma_i}(\tilde{x}|x) - s_{\theta}(\tilde{x}, \sigma_i) \|^2 + \text{Const} \\ = & \frac{1}{L} \sum_{i=1}^L \lambda(\sigma_i) \mathbb{E}_{x \sim p_{\text{data}}, z \sim \mathcal{N}(0, I)} \| s_{\theta}(x + \sigma_i z, \sigma_i) + \frac{z}{\sigma_i} \|^2 + \text{Const}. \end{aligned}$$



- Adjacent noise scales should have sufficient overlap to ease transitioning across noise scales in annealed Langevin dynamics.
- For example, a geometric progression

$$\frac{\sigma_i}{\sigma_{i+1}} = \alpha > 1, \quad i = 1, \dots, L-1$$

- What about the weighting function  $\lambda$ ?
- Use  $\lambda(\sigma) = \sigma^2$  to balance different score matching losses

$$\begin{aligned} & \frac{1}{L} \sum_{i=1}^L \sigma_i^2 \mathbb{E}_{x \sim p_{\text{data}}, z \sim \mathcal{N}(0, I)} \|s_\theta(x + \sigma_i z, \sigma_i) + \frac{z}{\sigma_i}\|^2 \\ &= \frac{1}{L} \sum_{i=1}^L \mathbb{E}_{x \sim p_{\text{data}}, z \sim \mathcal{N}(0, I)} \|\sigma_i s_\theta(x + \sigma_i z, \sigma_i) + z\|^2 \end{aligned}$$

- ▶ Sample a mini-batch of datapoints  $\{x_1, \dots, x_n\} \sim p_{\text{data}}$ .
- ▶ Sample a mini-batch of noise scale indices

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

- ▶ Sample a mini-batch of Gaussian noise

$$\{z_1, \dots, z_n\} \sim \mathcal{N}(0, I)$$

- ▶ Estimate the weighted mixture of score matching losses

$$\frac{1}{n} \sum_{k=1}^n \|\sigma_{i_k} s_\theta(x_k + \sigma_{i_k} z_k, \sigma_{i_k}) + z_k\|^2$$

- ▶ As efficient as training one single non-conditional score-based model.

- ▶ Salakhutdinov, R. and Hinton, G. Deep boltzmann machines. In Artificial intelligence and statistics, 2009.
- ▶ Aapo Hyvarinen. Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research, 6(Apr):695–709, 2005.
- ▶ Gutmann, M. and Hyvarinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In AISTATS 2010.
- ▶ Pascal Vincent. A connection between score matching and denoising autoencoders. Neural computation, 23(7):1661–1674, 2011.



- ▶ Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019.
- ▶ Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In Advances in Neural Information Processing Systems, pp. 11895–11907, 2019.

