Score-Based Generative Models

Volodymyr Kuleshov

Cornell Tech

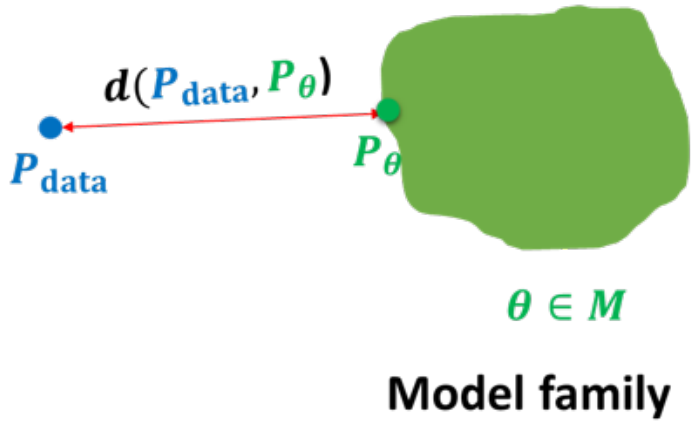
Lecture 12

Announcements

- Assignment 3 will be released today and is due in exactly two weeks.
- Please make sure to have signed up for a presentation slot **by the end of the week**.

Recap So Far

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



We have seen several generative model families so far:

1. Autoregressive, Latent Variable Models, and GANs:
   - Can perform generation, representation learning, outlier detection, etc.
   - Often require complex models (e.g., with tractable Jacobians)
   - GANs yield good samples with simple models but training is unstable
2. Energy-Based Models:
   - Make very little modeling assumptions
   - Maximum likelihood training with MCMC is slow

This lecture: can we train energy-based models without slow MCMC?

Energy-Based Models

Energy-based models specify distributions of the form

$$p_\theta(\mathbf{x}) = \frac{1}{\int \exp(f_\theta(\mathbf{x}))d\mathbf{x}} \exp(f_\theta(\mathbf{x})) = \frac{1}{Z(\theta)} \exp(f_\theta(\mathbf{x}))$$

where we used $Z(\theta) = \int \exp(f_\theta(\mathbf{x}))d\mathbf{x}$ to denote the partition function.

Pros of EBMs:

- extreme flexibility: can use pretty much any function $f_\theta(\mathbf{x})$ you want

Cons: Computing $Z(\theta)$ is generally intractable. As a result:

- Sampling from $p_\theta(\mathbf{x})$ is intractable in general
- Evaluating and optimizing likelihood $p_\theta(\mathbf{x})$ is hard (learning is hard)

Learning Energy-Based Models

Training EBMs using likelihood is intractable and requires approximations:

1. Variational inference: optimization-based approximations to likelihood
   - Have been used extensively to derive variational autoencoders.
   - Can also be applied to energy-based models (but it's less common)
2. MCMC-based methods: sampling-based approximations (last lecture)
   - Example: contrastive divergence for learning EBM model $p(\mathbf{x}; \theta_t)$.
   - Starting at some θ_0, for $t = 1, 2, ..., T$:
     1. Run MCMC chain to compute samples from $p(\mathbf{x}; \theta_t)$
     2. Use MCMC samples to compute gradient $\nabla \log p(\text{data}; \theta_t)$.
     3. Take a gradient step $\theta_{t+1} = \theta_t + \alpha \cdot \nabla \log p(\text{data}; \theta_t)$.
3. Alternatives to likelihood that don't require computing $Z(\theta)$ (today!)

Lecture Outline

1. Score Functions
   - Motivation
   - Definitions
   - Score Estimation
2. Score Matching
   - Fisher Divergences
   - Score Matching
   - Sliced and Denoised Score Estimation
3. Sample Generation
   - Langevin Dynamics
   - Manifold Hypothesis
   - Gaussian Smoothing

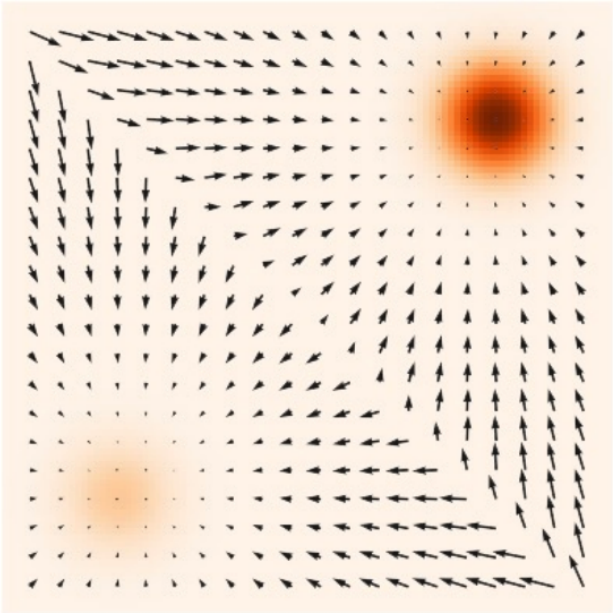
Figures and contents adapted from Yang Song and Stefano Ermon

The (Stein) Score Function: Definition

Given a probability density $p_\theta(\mathbf{x})$, its (Stein) score function is defined as

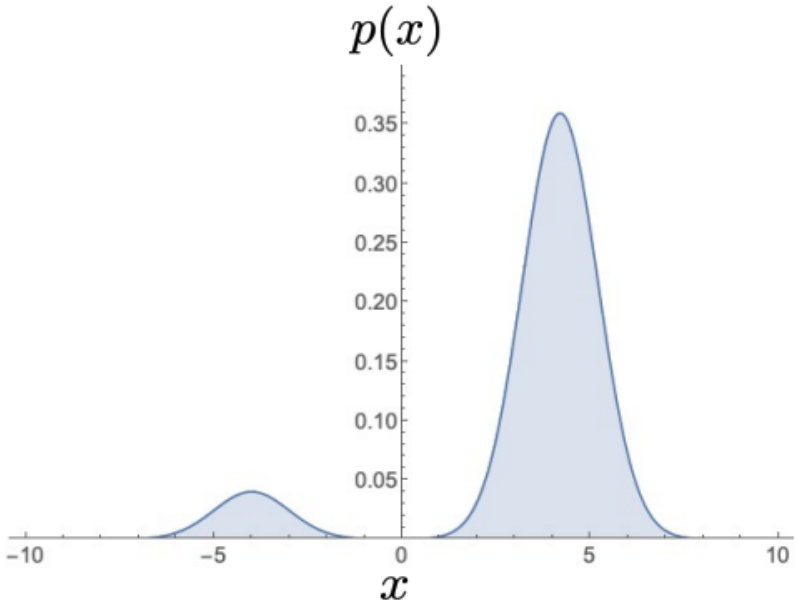
$$\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$$

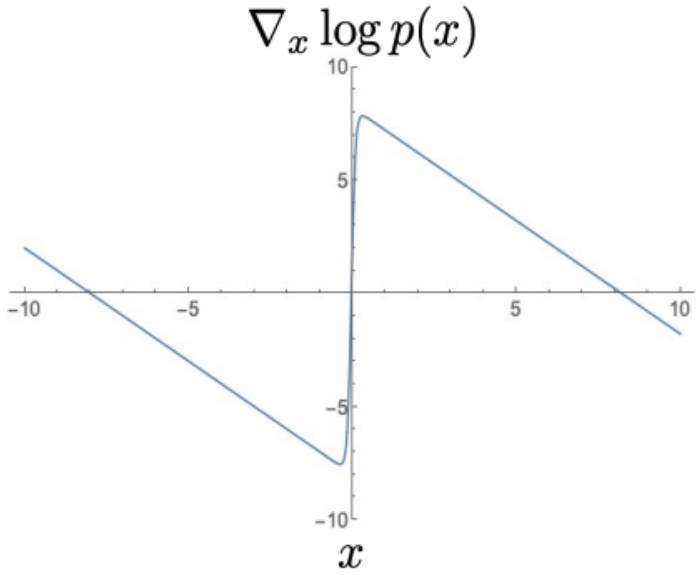
- This is the gradient of the density with respect to the input.
- Note that this is **different** from the gradient $\nabla_\theta \log p_\theta(\mathbf{x})$ of the log-density with respect to the parameters θ (which is what we use in gradient descent).
- The score function points in the direction of increasing model probability.



The (Stein) Score Function: 1D Example

On the left is a mixture of Gaussians. One the right is its score function in 1D.





Note that the score function is negative (points towards the left) near the left mode and is positive (points towards the right) near the right mode.

Score Functions Do Not Involve Normalizing Constants

Suppose we have an energy-based model of the form

$$p_\theta(\mathbf{x}) = \frac{1}{\int \exp(f_\theta(\mathbf{x}))d\mathbf{x}} \exp(f_\theta(\mathbf{x})) = \frac{1}{Z(\theta)} \exp(f_\theta(\mathbf{x}))$$

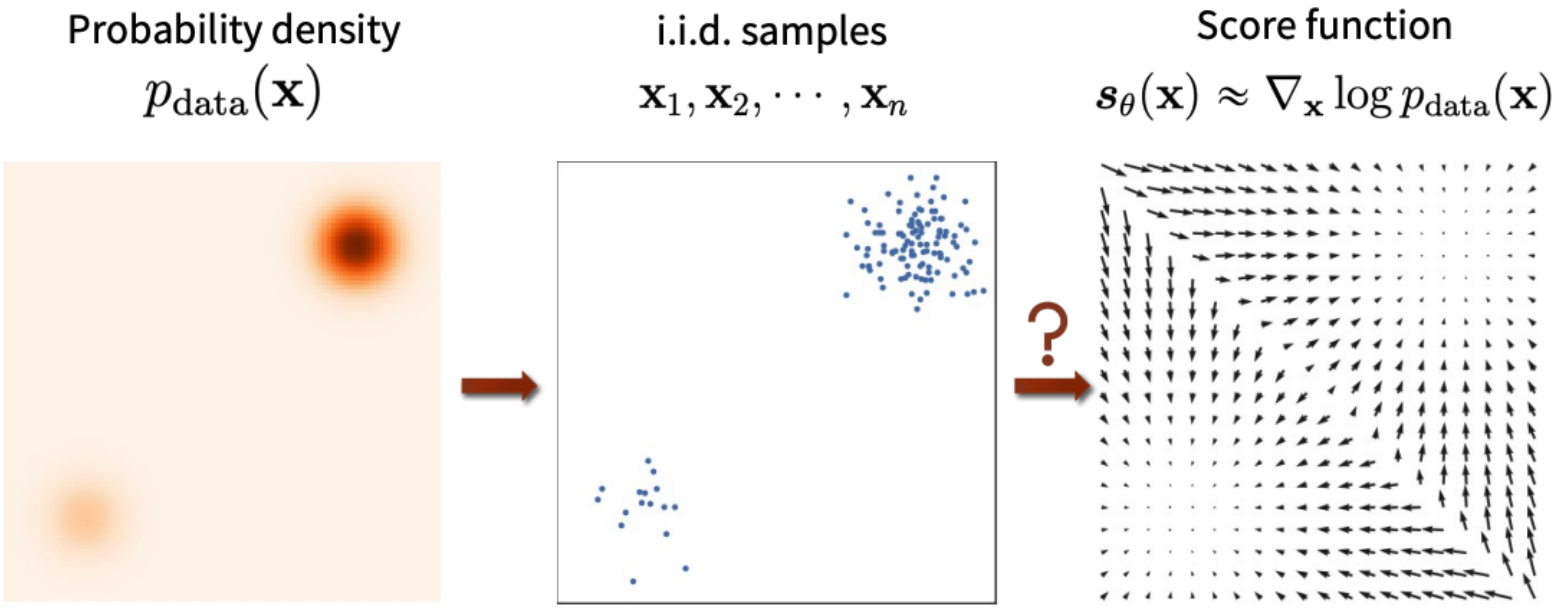
- Computing a score function of the model does not require $Z(\theta)$:

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

- **Idea**: Instead of learning $p_\theta(\mathbf{x})$, learn a model $\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$ of the score function of the data $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$!

Score Function Estimation: Definition

The idea of score function estimation is to learn a model $s_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$ of the score function from a dataset of sample points.



- We are given a dataset $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$
- Our task is to estimate $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- Our model is a parameterized function $s_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$
- Our objective is that $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) \approx s_\theta(\mathbf{x})$

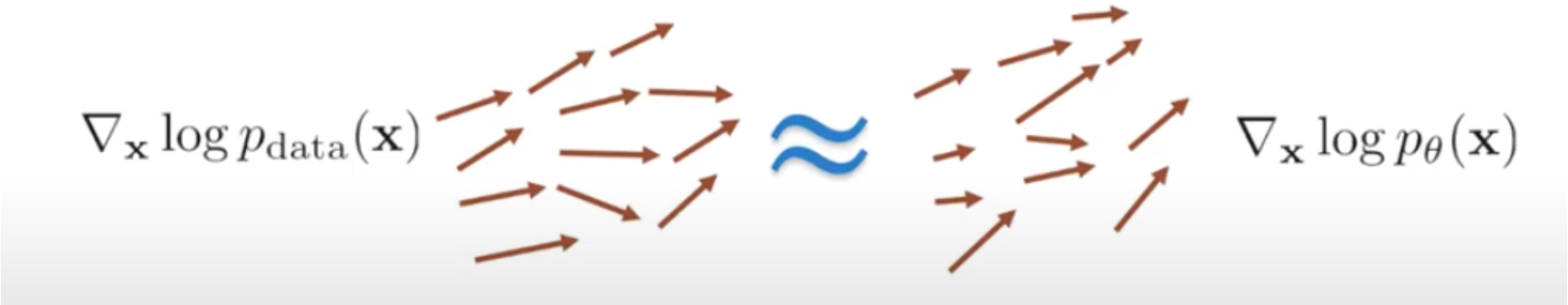
Lecture Outline

1. Score Functions
   - Motivation
   - Definitions
   - Score Estimation
2. Score Matching
   - Fisher Divergences
   - Score Matching
   - Sliced and Denoised Score Estimation
3. Sample Generation
   - Langevin Dynamics
   - Manifold Hypothesis
   - Gaussian Smoothing

Figures and contents adapted from Yang Song and Stefano Ermon

Score Matching via Fisher Divergences

How do we fit score functions?
Data and model scores are vector fields; we want them to be aligned:



- Formally, we use the Fisher divergence:

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

- It's the average Euclidean distance between two vectors over all $\mathbf{x}$
- The Fisher divergence is zero if and only if $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) \approx s_{\theta}(\mathbf{x})$

A Practical Objective for Score Matching

The Fisher divergence doesn't give us an objective we can easily optimize.

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

- We don't know $\log p_{\text{data}}(\mathbf{x})$, and much less its gradient.
- However, via a change of variables trick, we can obtain an equivalent objective (Hyvarinen, 2005)

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\frac{1}{2}||s_\theta(\mathbf{x})||_2^2 + \text{tr}\left(\nabla_{\mathbf{x}} s_\theta(\mathbf{x})\right)\right]$$

- This is the *score matching* objective. It can be further approximated via Monte-Carlo given a dataset $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$:

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

Integration by Parts: 1D Proof

We apply integration by parts as follows:

$$\frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[||\nabla_{\mathbf{x}}\log p(\mathbf{x}) - s_\theta(\mathbf{x})||_2^2\right]$$
$$= \underbrace{\frac{1}{2}\int_{-\infty}^{\infty} p(x)(\nabla_{\mathbf{x}}\log p(x))^2 dx}_{\text{const}} + \frac{1}{2}\int_{-\infty}^{\infty} p(x)s_\theta(x)^2 dx - \int_{-\infty}^{\infty} p(x)\nabla_{\mathbf{x}}\log p(x)s_\theta(x)^2 dx$$

We drop the first term and apply integration parts to the third term:

$$\int_{-\infty}^{\infty} p(x)\nabla_{\mathbf{x}}\log p(x)s_\theta(x)^2 dx = \int_{-\infty}^{\infty} p(x)\frac{\nabla_{\mathbf{x}}p(x)}{p(x)}s_\theta(x)^2 dx = \int_{-\infty}^{\infty} \nabla_{\mathbf{x}}p(x)s_\theta(x)^2 dx$$
$$= \underbrace{p(x)s_\theta(x)|_{-\infty}^{\infty}}_{=0} - \int_{-\infty}^{\infty} p(x)\nabla_x s_\theta(x)dx$$

The first term is zero because we assume x has bounded support. Plugging the above into the first expression yields the desired result.

Score Matching Doesn't Scale

We may now try to fit a score matching model:

- We parameterize $s_\theta(\mathbf{x})$ using a neural net and optimize the objective using gradient descent:

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

- However, this doesn't scale: gradient descent optimization requires us to compute nested backprop on each diagonal element of $\nabla_{\mathbf{x}} s_\theta(\mathbf{x})$:

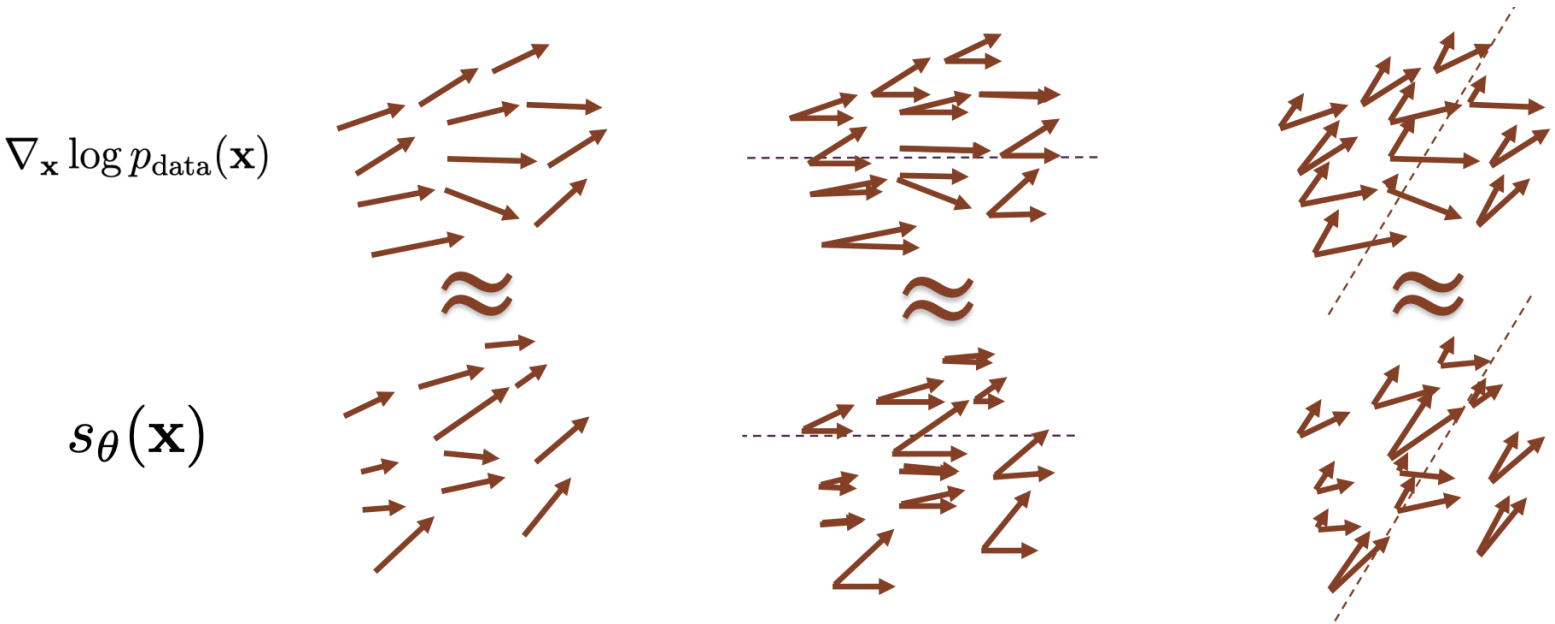
$$\nabla_{\mathbf{x}} s_\theta(\mathbf{x}) = \begin{pmatrix} \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_1} & \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_2} & \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_3} \\ \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_1} & \frac{\partial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

Sliced Score Matching: Intuition

We need to find more clever approximation to the score matching objective.

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

- Note that in low-dimensions, the problem is tractable.
- Therefore, we will try to optimize 1D projections of the score function



Sliced Fisher Divergence

Sliced score matching optimizes the sliced Fisher divergence:

$$\frac{1}{2}\mathbb{E}_{\mathbf{v}\sim q(\mathbf{v})}\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[(\mathbf{v}^\top \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) - \mathbf{v}^\top \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}))^2\right]$$

- Where $q(\mathbf{v})$ is a distribution over projection vectors $\mathbf{v}$ that we choose.
- As before, this is not practical because we don't know p_{data}. However, via a change of variables trick, we can obtain an equivalent objective:

$$\mathbb{E}_{\mathbf{v}\sim q(\mathbf{v})}\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\mathbf{v}^\top \nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x})\mathbf{v} + \frac{1}{2}(\mathbf{v}^\top \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}))^2\right]$$

- Each term in the above objective can be computed efficiently.
- $q(\mathbf{v})$ can be multivariate standard normal or Rademacher.
- Objective features consistency and asymptotic normality.

Denoising Score Matching

Alternatively, suppose that we have access to a noised version $q_\sigma(\tilde{\mathbf{x}})$ of the clean data $\mathbf{x} \sim p(\mathbf{x})$. For example, we can add Gaussian noise to $\mathbf{x}$:

$$\tilde{\mathbf{x}} = \mathbf{x} + \sigma \cdot \epsilon \text{ for } x \sim p(\mathbf{x}), \epsilon \sim \mathcal{N}(0, I)$$

Denoising score matching (Vincent, 2011) approximates the Fisher divergence between s_θ and $q_\sigma(\tilde{\mathbf{x}})$ as:

$$\frac{1}{2}\mathbb{E}_{\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}})}\left[||\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) - s_\theta(\tilde{\mathbf{x}})||_2^2\right] = \frac{1}{2}\mathbb{E}_{\mathbf{x} \sim p, \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\left[||\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) - s_\theta(\tilde{\mathbf{x}})||_2^2\right].$$

- $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$ is easy to compute e.g., $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = -\frac{\mathbf{x}-\tilde{\mathbf{x}}}{\sigma^2}$ in the above example with Gaussian noise
- This is faster than slicing, but can only learn noised distributions.

Algorithm for Learning Score-Based Generative Models

In summary, we have derived the following algorithm for learning a model $s_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$ of the score function:

- Given an initial set of parameters θ_0, for $t = 1, 2, ...$:
  1. Sample a batch of training points $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$
  2. Compute the gradient $\nabla_\theta J(\theta)$ of the objective, where $J(\theta)$ is either:

$$J_{\text{sliced}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left[\mathbf{v}_i^\top \nabla_{\mathbf{x}}^2 s_\theta(\mathbf{x}_i) \mathbf{v}_i + \frac{1}{2} (\mathbf{v}_i^\top \nabla_{\mathbf{x}} s_\theta(\mathbf{x}_i))^2 \right]$$

for some random projection vectors $\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n\}$ or

$$J_{\text{denoised}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left[||\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}_i) - s_\theta(\tilde{\mathbf{x}}_i)||_2^2 \right]$$

for some q_σ, or is an average of the two (Song et al., 2019).

  3. Take a gradient step $\theta_t = \theta_{t-1} - \alpha \cdot \nabla_\theta J(\theta_{t-1})$.

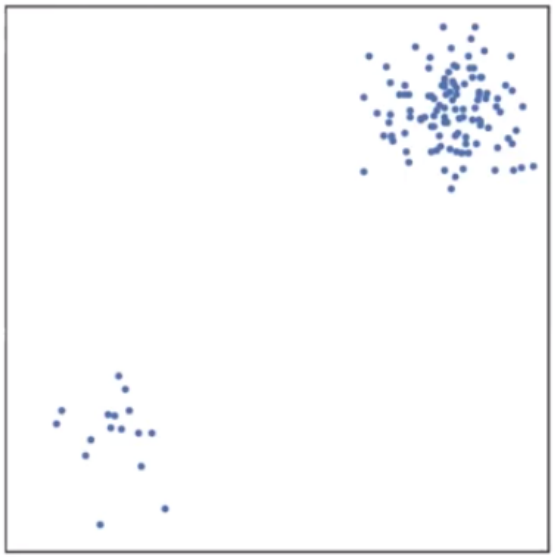
Lecture Outline

1. Score Functions
   - Motivation
   - Definitions
   - Score Estimation
2. Score Matching
   - Fisher Divergences
   - Score Matching
   - Sliced and Denoised Score Estimation
3. Sample Generation
   - Langevin Dynamics
   - Manifold Hypothesis
   - Gaussian Smoothing

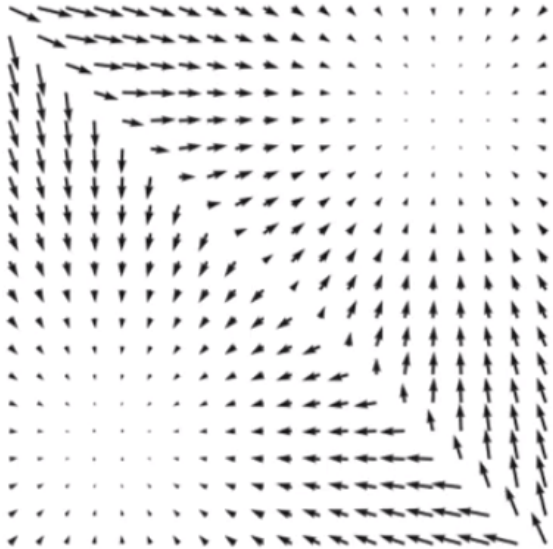
Figures and contents adapted from Yang Song and Stefano Ermon

Sample Generation: Intuition

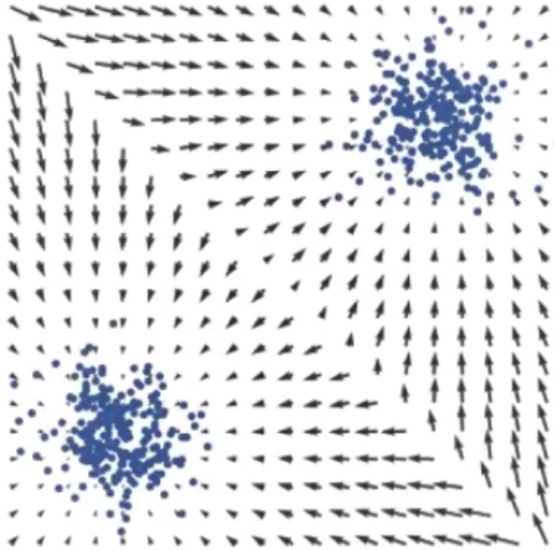
Next, we want to generate samples $\mathbf{x}$ from a score function $\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$. We will do this by following the gradient of the data distribution



Samples



Score Estimation



Langevin dynamics

Intuitively, we start with noise $\mathbf{z}$ and we follow the gradient of the data distribution until we generate accurate $\mathbf{x}$

Langevin Dynamics

Langevin dynamics is an MCMC sampling process that allows us to sample from $p_\theta(\mathbf{x})$ using only its score $\nabla_x \log p_\theta(\mathbf{x})$.

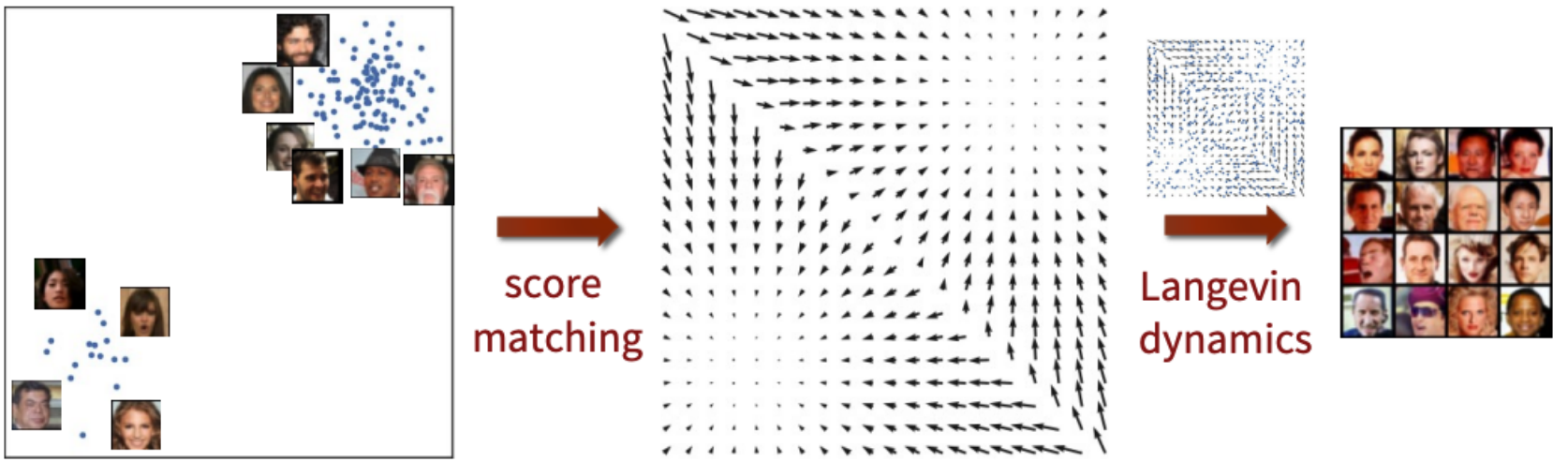
- Initialize $\mathbf{x}_0$ from a noise distribution
- For steps $t = 1, 2, \ldots, T$:
  - Sample a noise variable $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - $\mathbf{x}_t = \mathbf{x}_{t-1} + \frac{\alpha_t}{2} \nabla_x \log p_\theta(\mathbf{x}) + \sqrt{\alpha_t} \epsilon_t$

Note that:

- This process can be seen as gradient ascent on $\log p_\theta(\mathbf{x})$;
- It is guaranteed to produce samples from $p_\theta(\mathbf{x})$ when the step size $\alpha_t \to 0$ as $T \to \infty$ at the right annealing rate.
- We can use this process for sampling from a score function $s_\theta(\mathbf{x}) \approx \nabla_x \log p_\theta(\mathbf{x})$ by plugging s_θ into the above algorithm.

Score Function Modeling

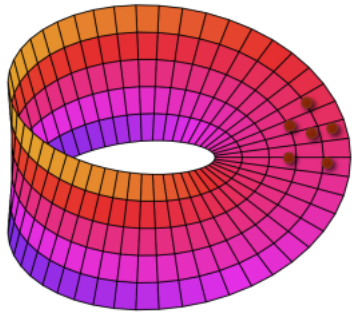
Thus, our high-level strategy is to learn $s_\theta(\mathbf{x}) \approx \nabla_x \log p_\theta(\mathbf{x})$ from a set of samples, and then generate new $\mathbf{x}$ via Langevin dynamics.



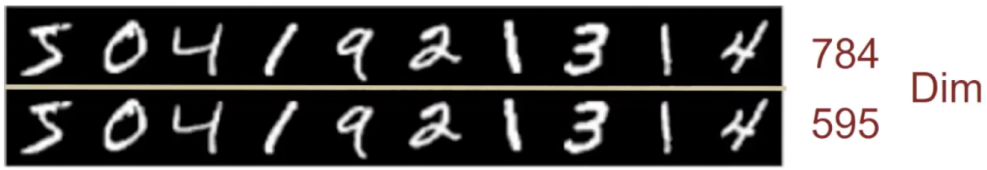
Manifold Hypothesis

The above sampling approach faces challenges arising from the manifold structure of the data.

- The manifold approach states that most of our data lives on a low dimensional manifold (subspace) in a high-dimensional space.



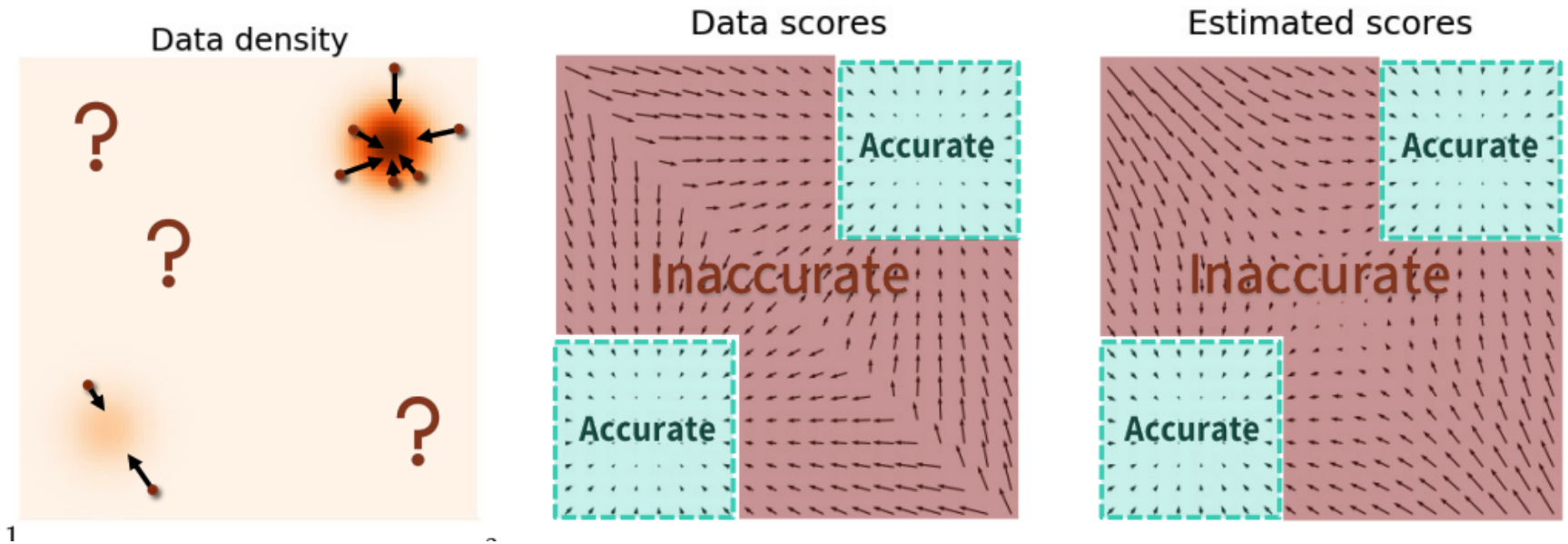
- This hypothesis can be verified empirically by examining the PCA reconstructions on typical datasets.



- This is a problem as $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ is not defined where $p_{\text{data}}(\mathbf{x}) = 0$

Learning and Sampling Scores in Low Density Regions

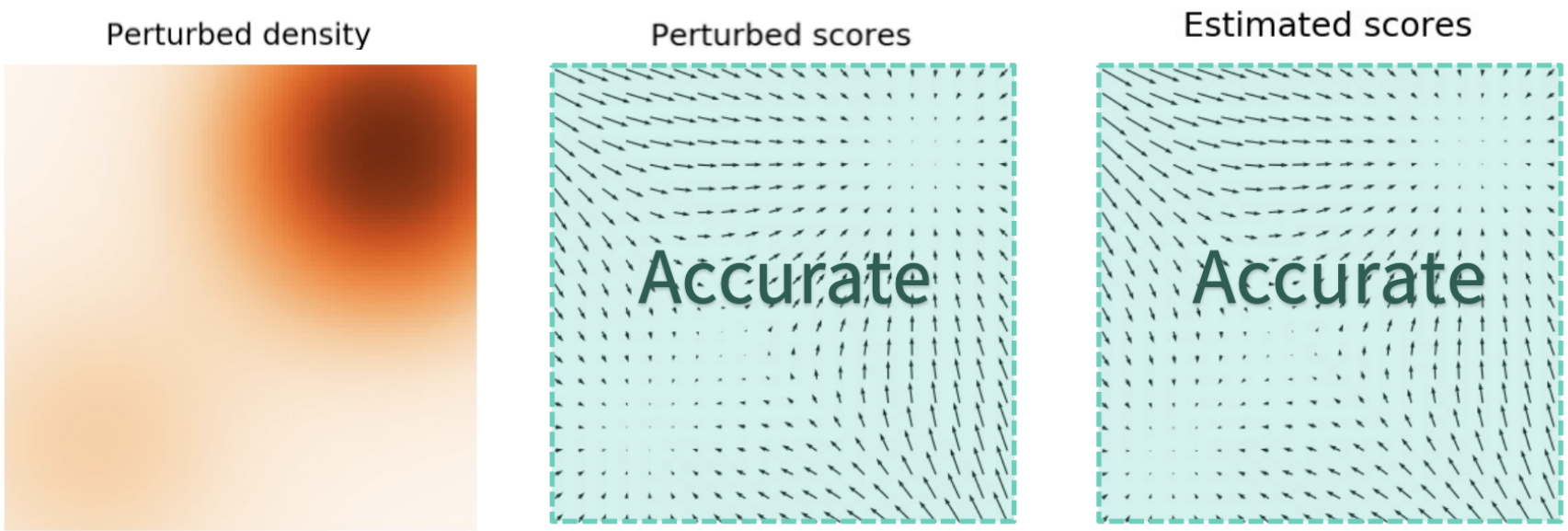
Thus, learning scores functions and generating samples from them is challenging in low density regions will be hard.



- We don't have enough samples to learn s_θ in low density regions
- Even if we knew $\nabla_x \log p_\theta$, it may be too weak to generate samples.

Learning and Sampling Scores in Low Density Regions

One solution to this problem is to add Gaussian noise to the data (i.e., smooth the data):



- The score function is now well-defined everywhere.
- However, it's defined on a different distribution.
- There is a tradeoff: smooth more and get better score vs. smooth less and better approximate the data distribution

Multi-Scale Noise

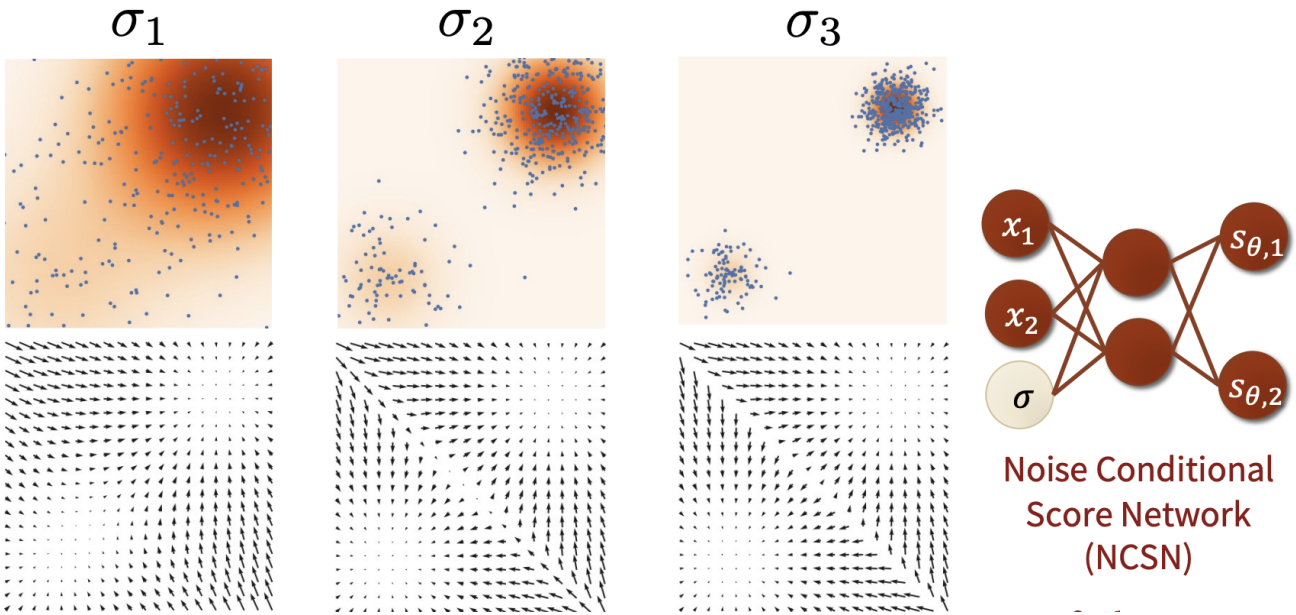
In order to trade off approximation accuracy and score function stability, we want to add noise at multiple scales

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

- We train using both small and large amounts of noise.
- At the initial stages of sampling, we use a larger noise.
- We gradually reduce noise over the course of sampling.

Noise Conditional Score Networks

We implement this strategy by training a score function model $s_\theta(\mathbf{x}, \sigma)$ that is conditioned on the noise σ.

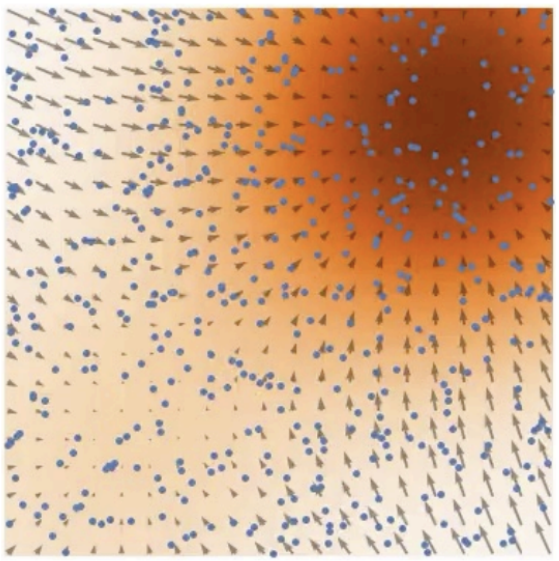


- The same model is trained for all σ. Parameters are shared across noise levels. For example, the denoised objective becomes:

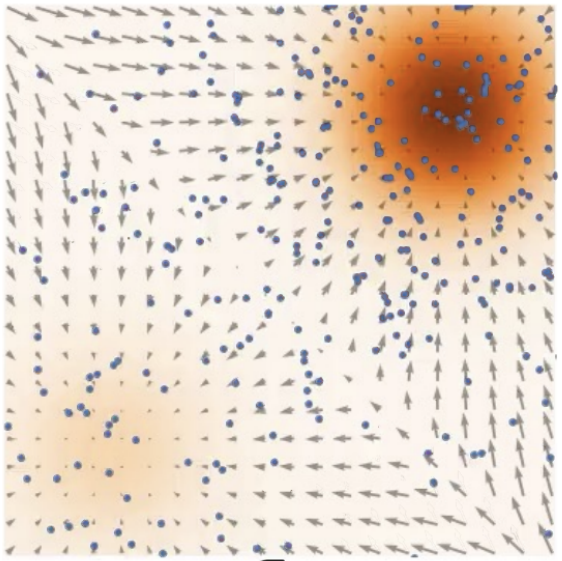
$$\mathbb{E}_{\sigma\sim\{\sigma_1,\sigma_2,\ldots,\sigma_L\}}\left[\frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p,\tilde{\mathbf{x}}\sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\left[||\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) - s_\theta(\tilde{\mathbf{x}},\sigma)||_2^2\right]\right].$$

Annealed Langevin Dynamics: Intuition

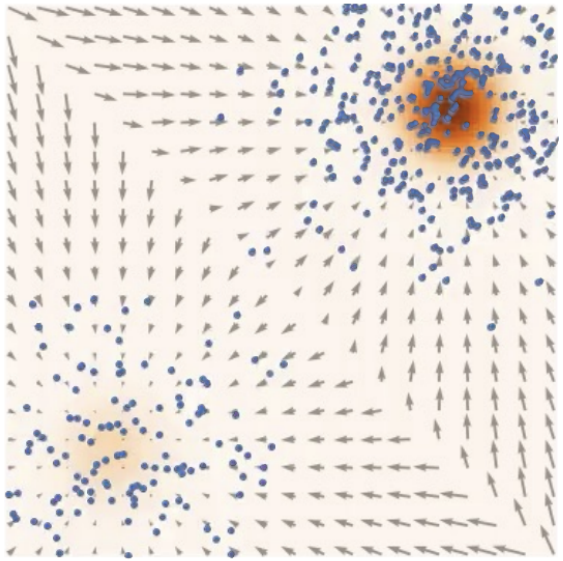
We can run Langevin Dynamics with decreasing noise levels. At each new noise level, we start where we left off with the previous noise level.



σ_1



σ_2



σ_3

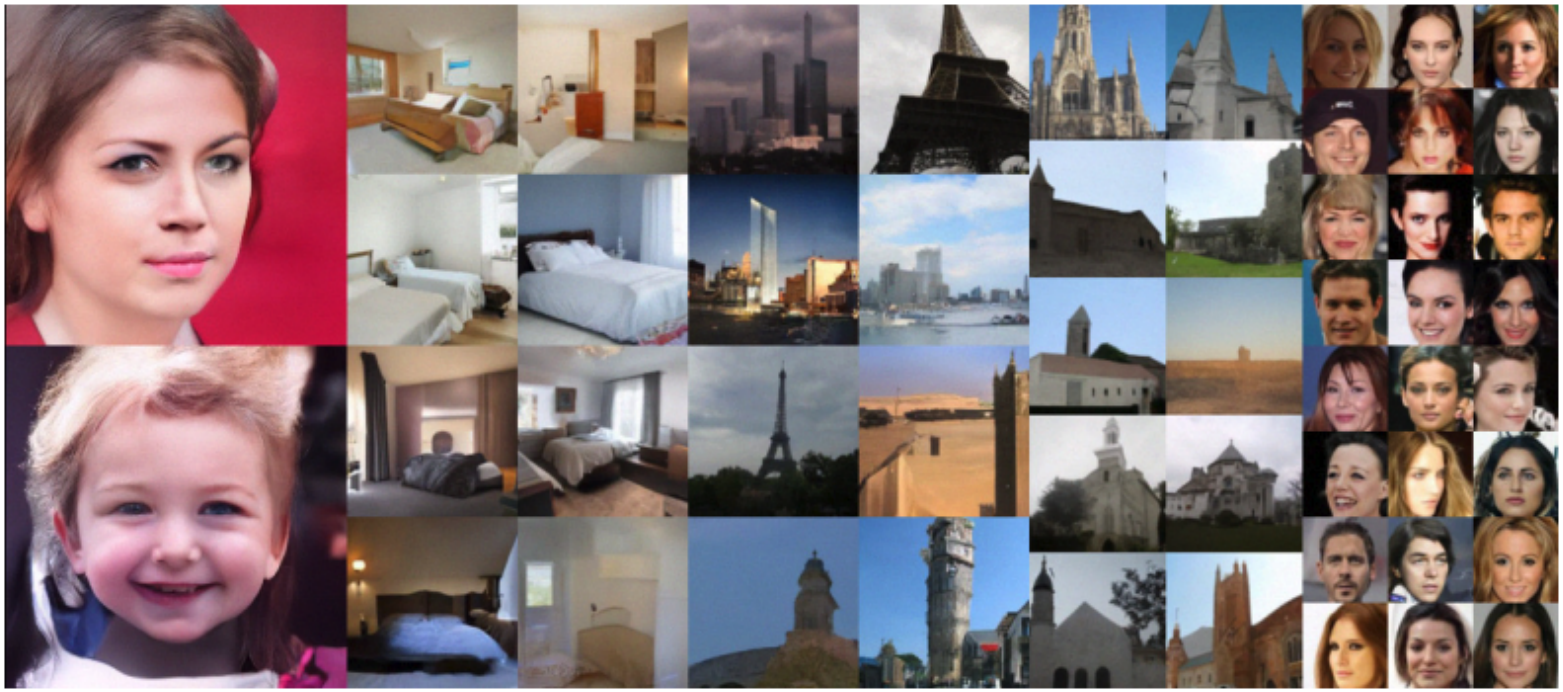
Annealed Langevin Dynamics

Annealed Langevin dynamics is an MCMC sampling process that allows us to sample from a noise-conditional score model $s_\theta(\mathbf{x}, \sigma)$:

- Initialize $\mathbf{x}_0$ from a noise distribution
- For noise levels $\sigma_\ell \in \{\sigma_1, \sigma_2, ..., \sigma_L\}$:
  - Set step size to $\alpha_t = \beta \cdot \frac{\sigma_\ell}{\sigma_L}$ for some $\beta > 0$
  - For steps $t = 1, 2, \ldots, T$:
    - Sample a noise variable $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    - $\mathbf{x}_t = \mathbf{x}_{t-1} + \frac{\alpha_t}{2} s_\theta(\mathbf{x}, \sigma_\ell) + \sqrt{\alpha_t}\epsilon_t$
  - Set $\mathbf{x}_0 = \mathbf{x}_T$: start next run at the end of last run.

Score-Based Generative Models: Samples

Score-based generative models can generate high-resolution samples that approach (and in recent versions match) the quality of GANs:

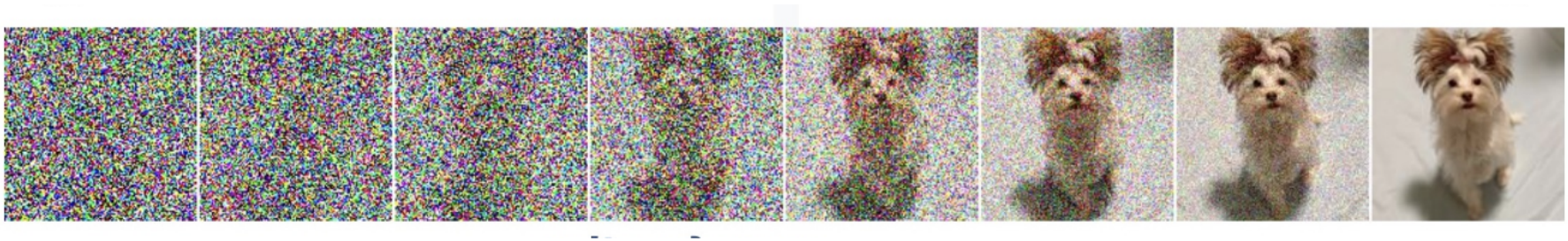


Score-Based Generative Models: Iterative Refinement

Score-based generative models learn to model distributions with increasing levels of Gaussian noise:

Perturbing an image with multiple scales of Gaussian noise.

When sampling using annealed Langevin dynamics, we undo this noise: the image gradually appears out of Gaussian noise.



We may refer to this type of gradual denoising as *iterative refinement*.

Conditional Sampling and Controllable Generation

Suppose we know a process $p(\mathbf{y}|\mathbf{x})$ mapping data $\mathbf{x}$ to auxiliary $\mathbf{y}$ (e.g., labels, noised images). We seek to model $p(\mathbf{x}|\mathbf{y})$ (a form of inverse).

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}).$$

Score-based models are naturally suited to this task:

- We can learn $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ unconditionally.
- The $\nabla_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x})$ is already known (e.g., it's a classifier)
- Thus, we can sample from $\nabla_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{y})$ using Langevin dynamics.

Score-Based Generative Models: Conditional Samples

Here is the result of generating class-conditional CIFAR-10 samples:

Other examples include: conditioning colored images on monochrome images, noisy MRI on clean MRI data, noisy CT on clean CT, etc.

Score-Based Generative Models: Pros and Cons

Pros:

1. Can train energy-based models without worrying about $Z(\theta)$;
2. Produce high quality samples (matching GANs) with stable training

Cons:

1. Sampling is much slower than in GANs (need to run Langevin MCMC for up to thousands of steps);
2. Do not perform density estimation or provide likelihoods;
3. Lack of latent variables for representation learning

Can we address some of the cons? Yes! See the next lecture on diffusion models.