Score-Based Generative Models

Volodymyr Kuleshov

Cornell Tech

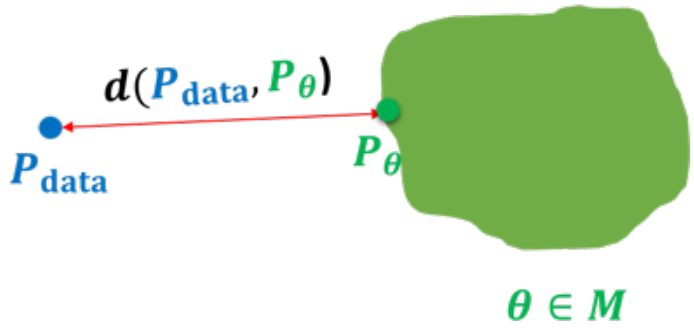
Lecture 12

Announcements

- Assignment 3 has been released
- 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 and Latent Variable Models:
   - Can perform generation, representation learning, outlier detection, etc.
   - Max-likelihood may not give best samples, imposes model constraints
2. Generative Adversarial Networks:
   - Can optimize arbitrary divergences; produces good samples
   - Training is difficult and unstable; no clear stopping criterion
3. 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

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

Pros:

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

What if we could train models without using Z?

Lecture Outline

1. Score Functions
   - Motivation
   - Definitions
   - Score estimation
2. Score Matching
   - Fisher Divergences
   - Score Matching
   - Sliced Score Estimation
3. Sample Generation
   - Langevin Dynamics
   - Manifold Hypothesis
   - Gaussian Smoothing
4. Connections to Diffusion Models

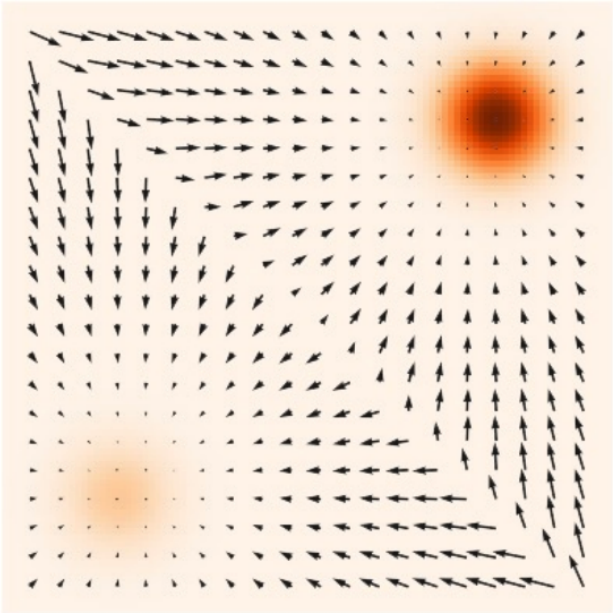
Figures and contents adapted from Stefano Ermon and Lily Weng

The (Stein) Score Function: Definition

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

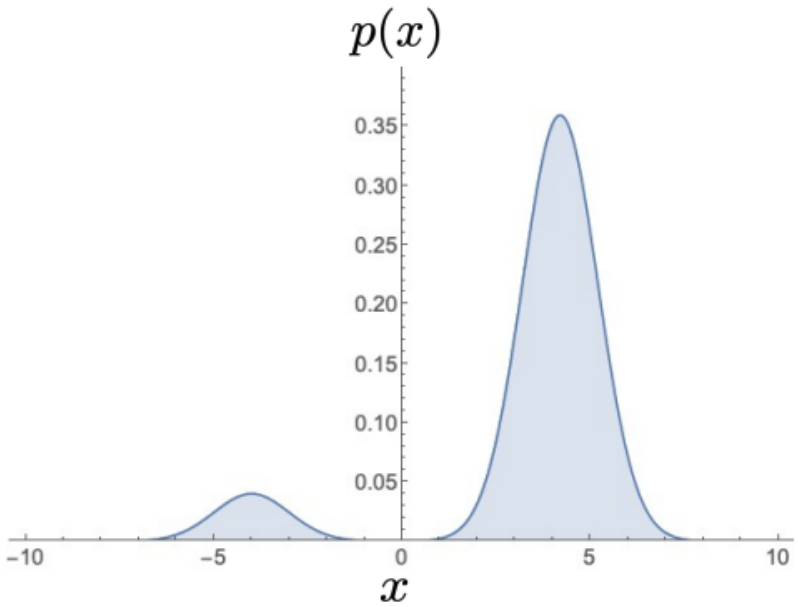
$$\nabla_{\mathbf{x}} \log p(\mathbf{x})$$

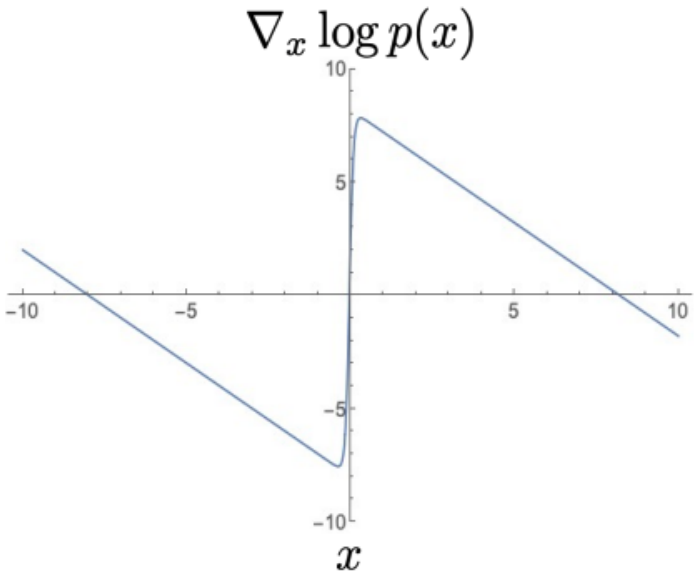
- This is the gradient of the density with respect to the input.
- Note that this is **different** from the gradient of the density with respect to the parameters (which is that we normally use in gradient descent).
- The score function points in the direction of increasing model probability.



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

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

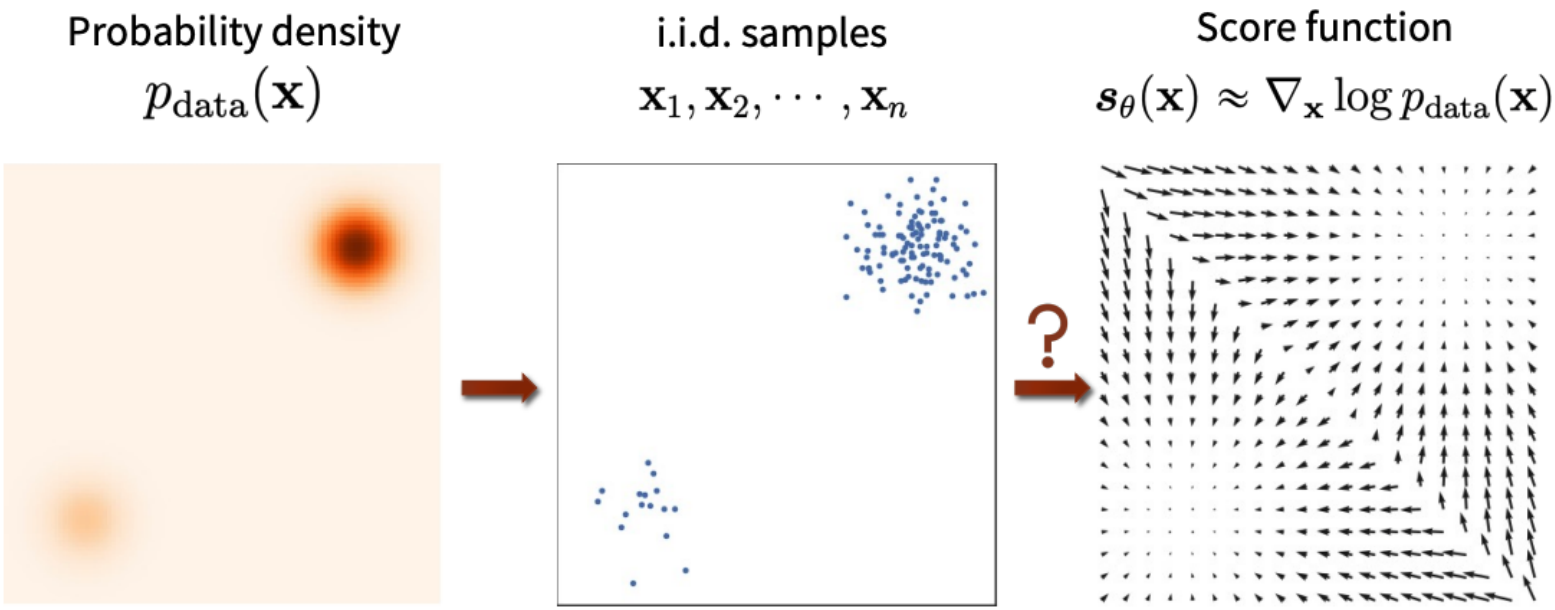
- Computing a score function of the model does not involve Z:

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

- **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 $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) \approx s_\theta(\mathbf{x})$

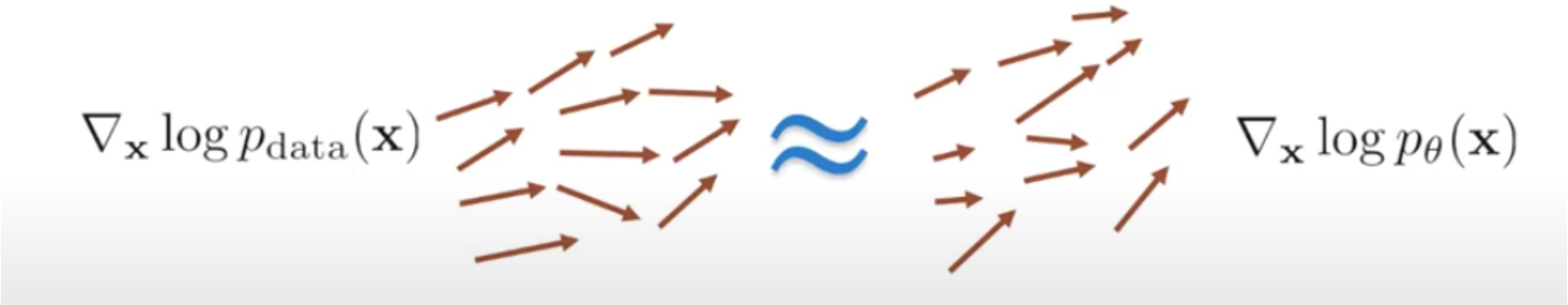
Lecture Outline

1. Score Functions
   - Motivation
   - Definitions
   - Score estimation
2. Score Matching
   - Fisher Divergences
   - Score Matching
   - Sliced Score Estimation
3. Sample Generation
   - Langevin Dynamics
   - Manifold Hypothesis
   - Gaussian Smoothing
4. Connections to Diffusion Models

Figures and contents adapted from Stefano Ermon and Lily Weng

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

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

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 second 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:

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

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

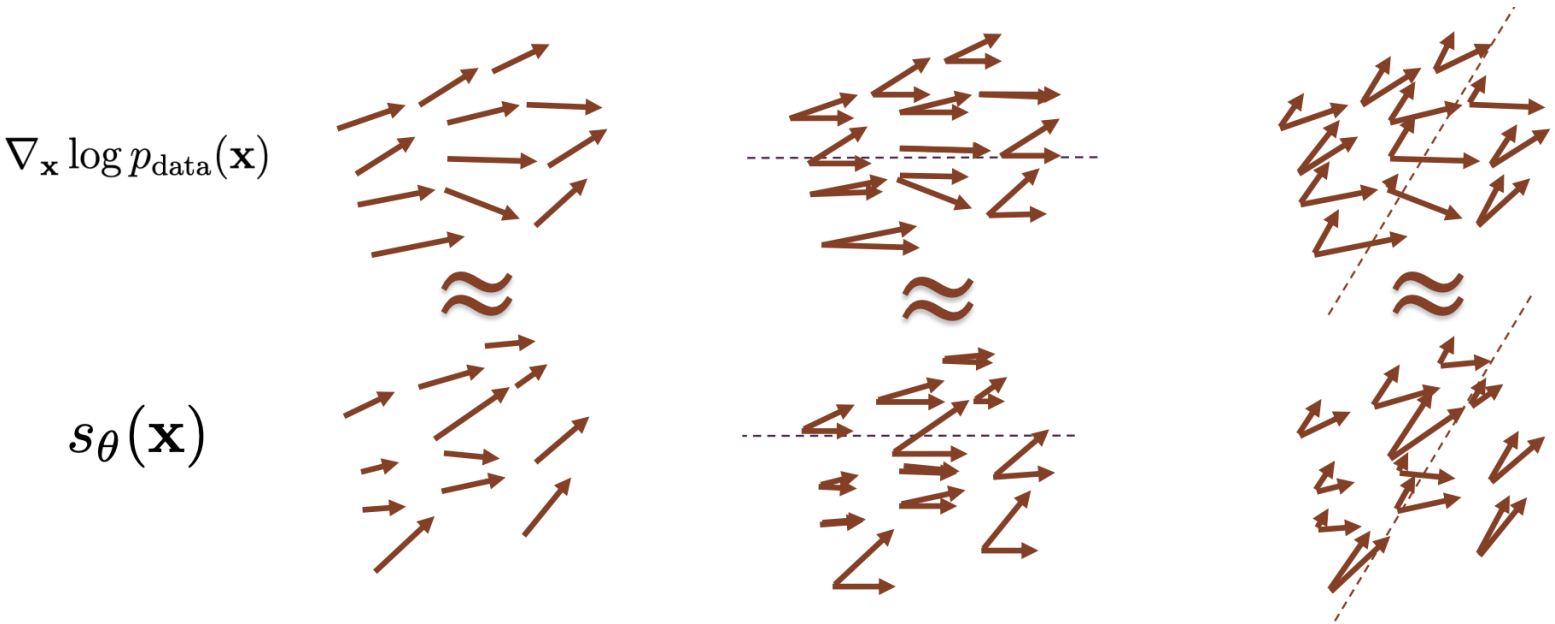
$$\nabla_{\mathbf{x}} s_\theta(\mathbf{x}) = \begin{pmatrix} \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_1} & \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_2} & \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_3} \\ \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_1} & \frac{\partial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

Sliced Score Matching: Intuition

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

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

- 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 optimized 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:

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

- 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.

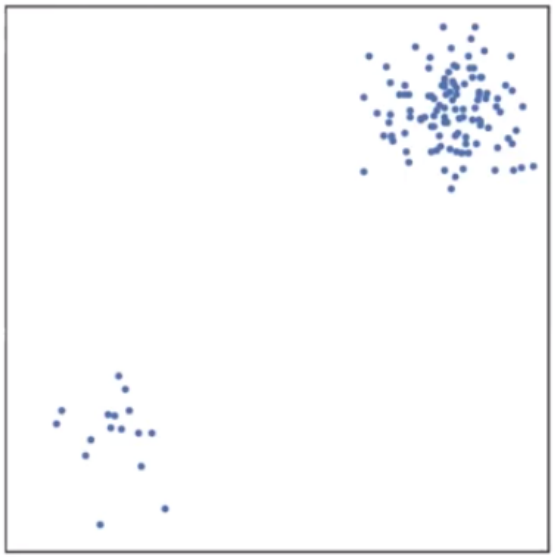
Lecture Outline

1. Score Functions
   - Motivation
   - Definitions
   - Score estimation
2. Score Matching
   - Fisher Divergences
   - Score Matching
   - Sliced Score Estimation
3. Sample Generation
   - Langevin Dynamics
   - Manifold Hypothesis
   - Gaussian Smoothing
4. Connections to Diffusion Models

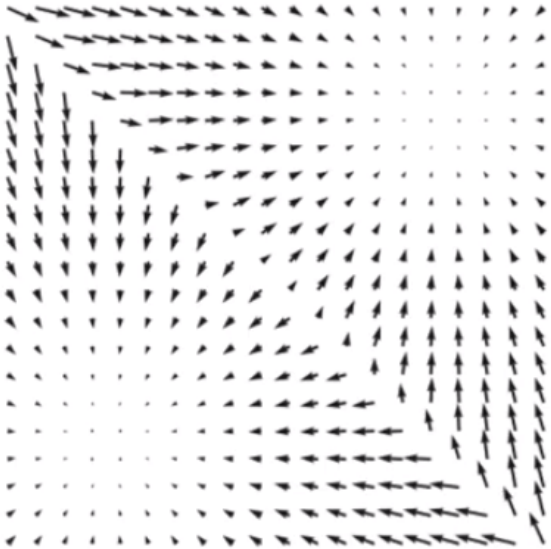
Figures and contents adapted from Stefano Ermon and Lily Weng

Sample Generation: Intuition

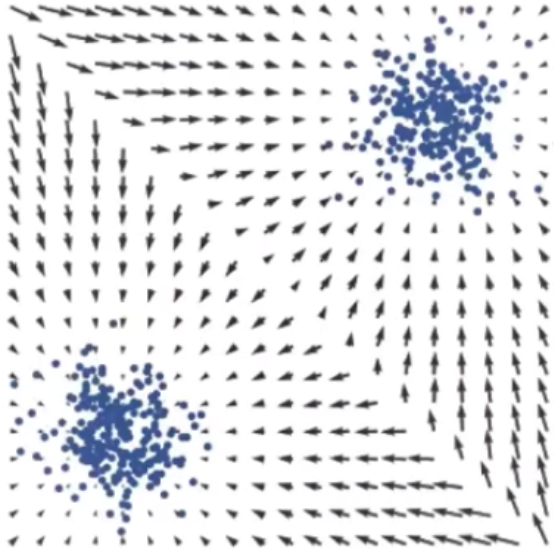
Next, we want to generate samples from a trained score function. We can do this by following the gradient of the data distribution



Samples



Score Estimation



Langevin dynamics

Intuitively, we start with noise z and we follow the gradient of the data distribution until we generate accurate x

Langevin Dynamics

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

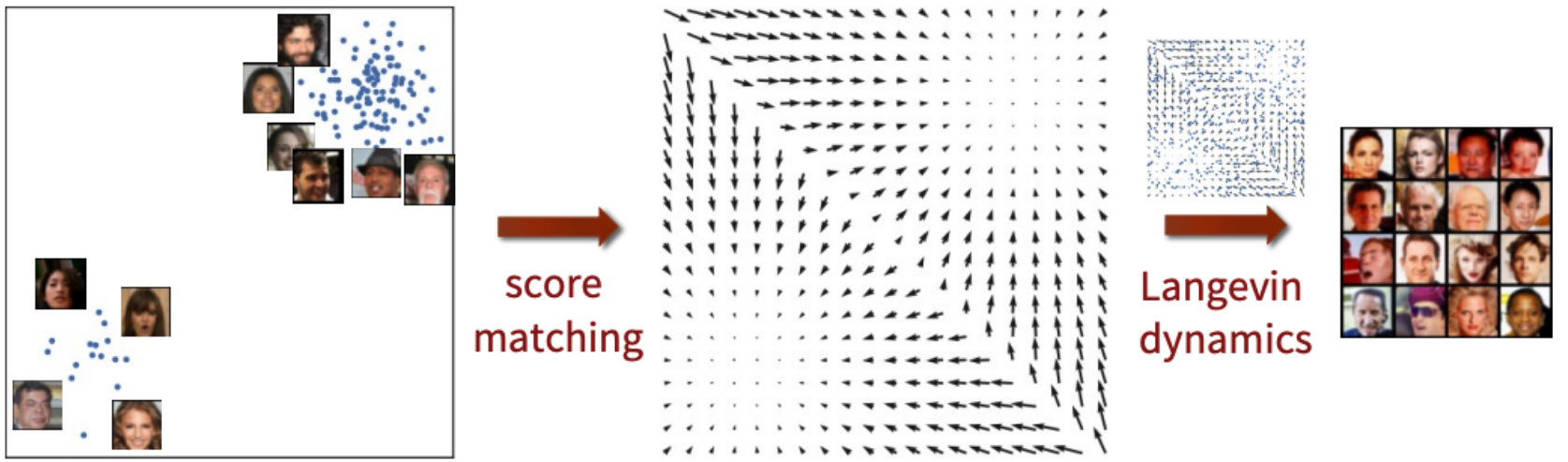
- Initialize $\mathbf{x}_0$ from a noise distribution
- For steps $t = 1, 2, \ldots, T$:
  - Sample a noise variable $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - $\mathbf{x}_t = \mathbf{x}_{t-1} + \frac{\alpha_t}{2} \nabla_x \log p_\theta(\mathbf{x}) + \sqrt{\alpha_t} \epsilon_t$

This process is guaranteed to produce samples from $p_\theta(\mathbf{x})$ when the step size $\alpha_t \to 0$ and $T \to \infty$.

We can use this process for sampling from a score function $s_\theta(\mathbf{x}) \approx \nabla_x \log p_\theta(x)$ by plugging s_θ into the above algorithm.

Score Function Modeling

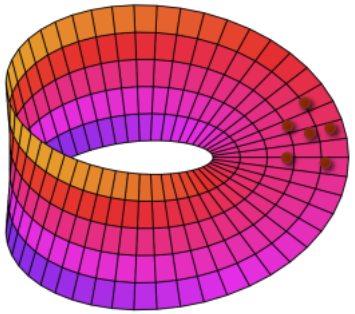
Thus, our high-level strategy is to learn $s_\theta(\mathbf{x}) \approx \nabla_x \log p_\theta(x)$ from a set of samples, and then generate new $\mathbf{x}$ via Langevin dynamics.



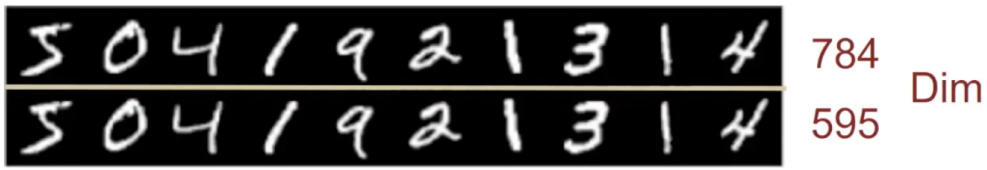
Manifold Hypothesis

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.



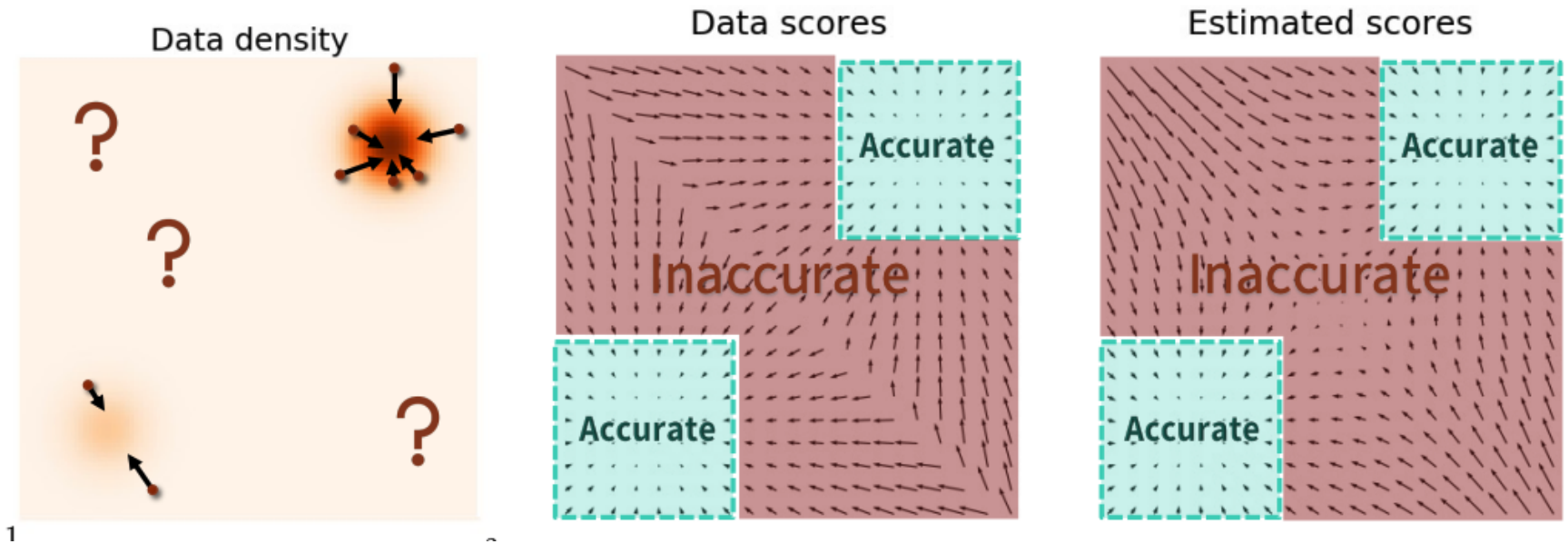
- This hypothesis can be verified empirically by examining the PCA reconstructions on typical datasets.



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

Learning and Sampling Scores in Low Density Regions

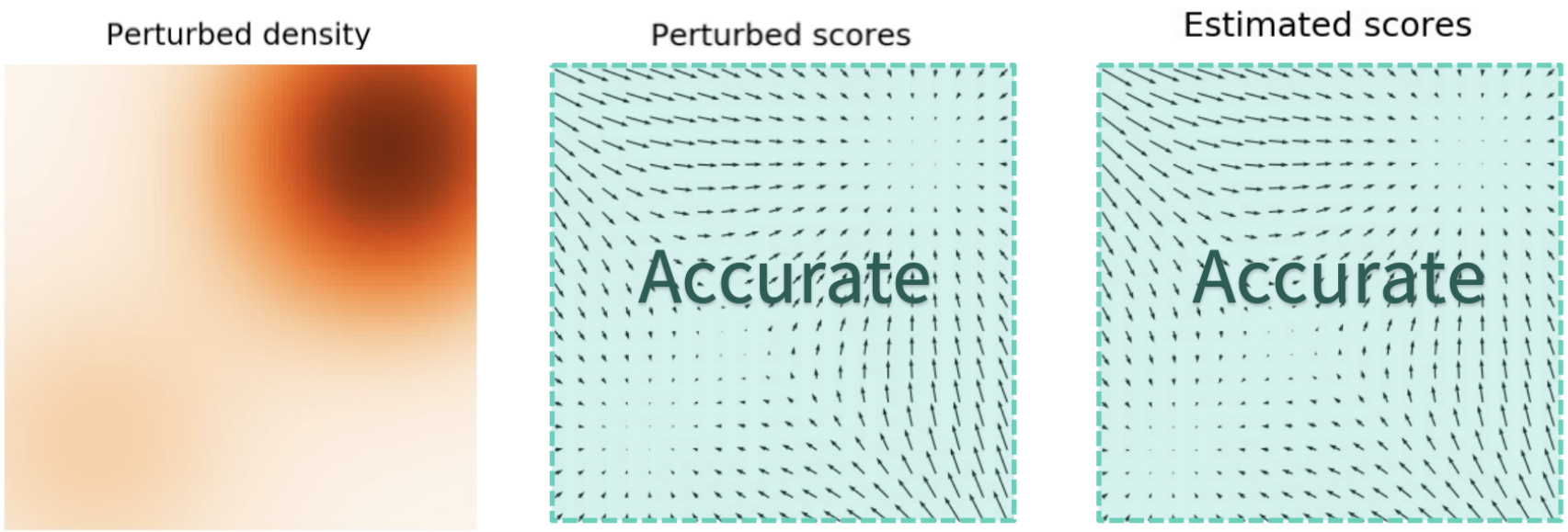
Thus, learning scores functions and generating samples from them is challenging in low density regions will be hard.



- 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

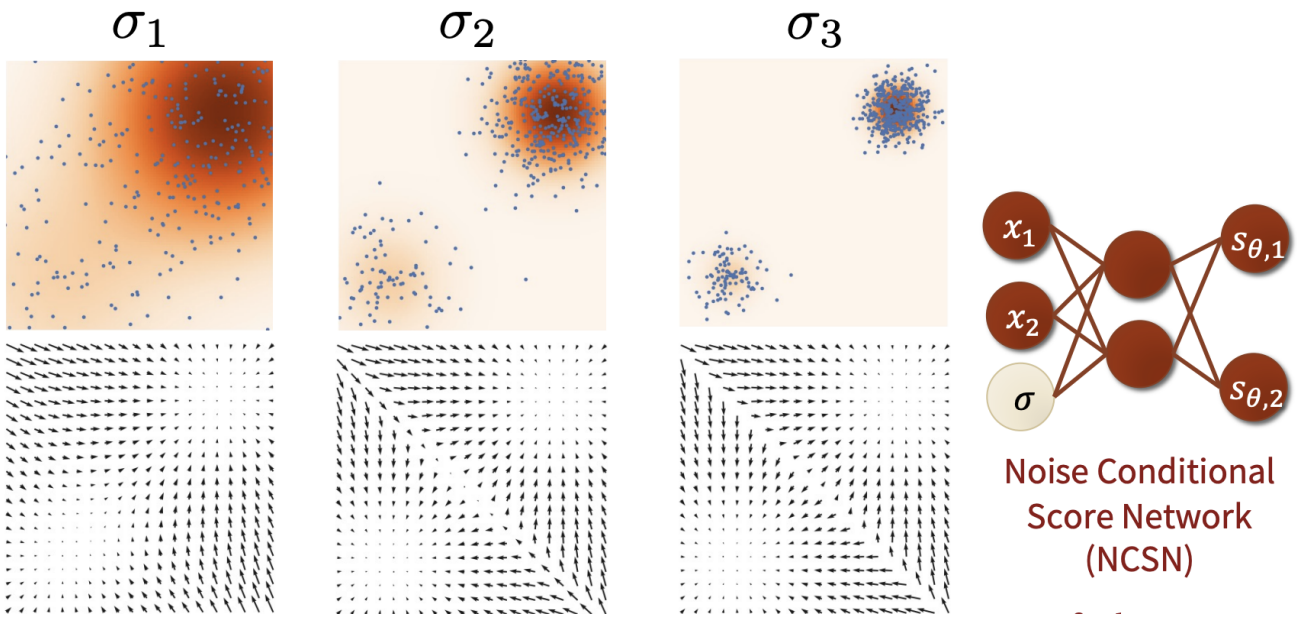
In order to trade off approximation accuracy and score function stability, we want to add noise at multiple scales

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

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

Noise Conditional Score Networks

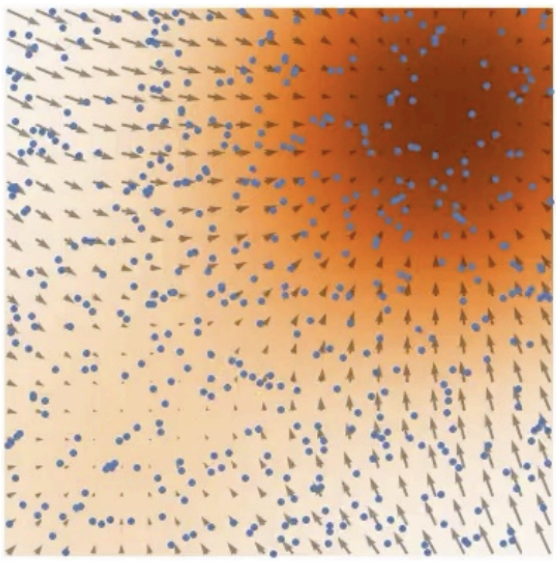
We implement this strategy by training a score function model $s_\theta(\mathbf{x}, \sigma)$ that is conditioned on the noise σ.



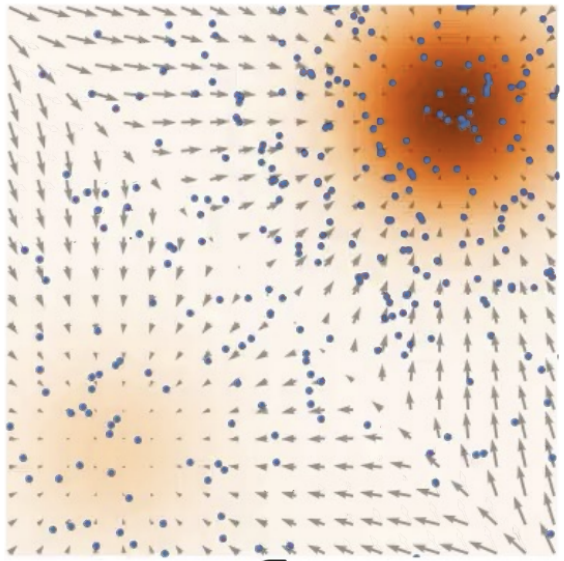
- The same model is trained for all σ.
- Parameters are shared across noise levels

Annealed Langevin Dynamics: Intuition

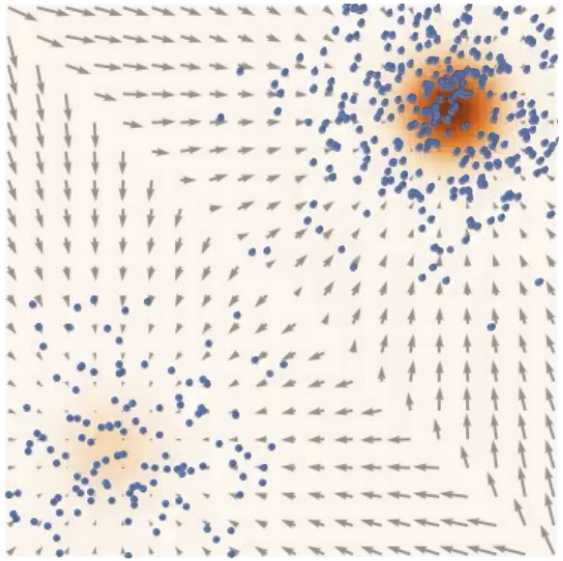
We can run Langevin Dynamics with decreasing noise levels. At each new noise level, we start where we left off with the previous noise level.



σ_1



σ_2



σ_3

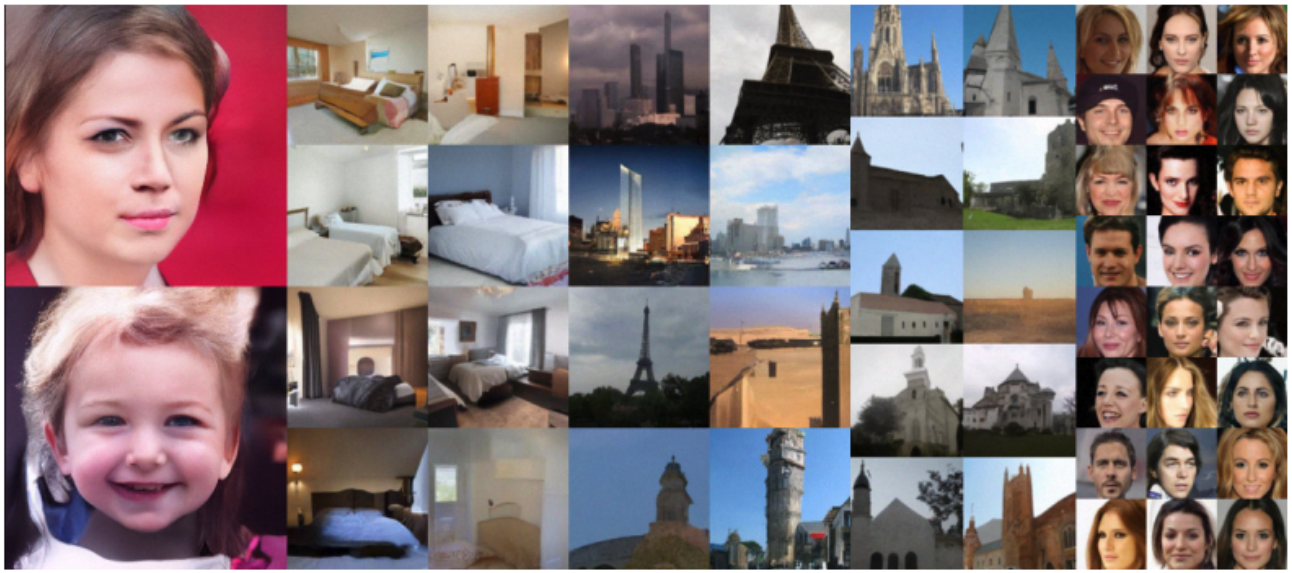
Annealed Langevin Dynamics

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

- Initialize $\mathbf{x}_0$ from a noise distribution
- For noise levels $\sigma_\ell \in \{\sigma_1, \sigma_2, ..., \sigma_L\}$:
  - Set step size to $\alpha_t = \beta_t \cdot \frac{\sigma_\ell}{\sigma_L}$
  - 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 version match) the quality of GANs:



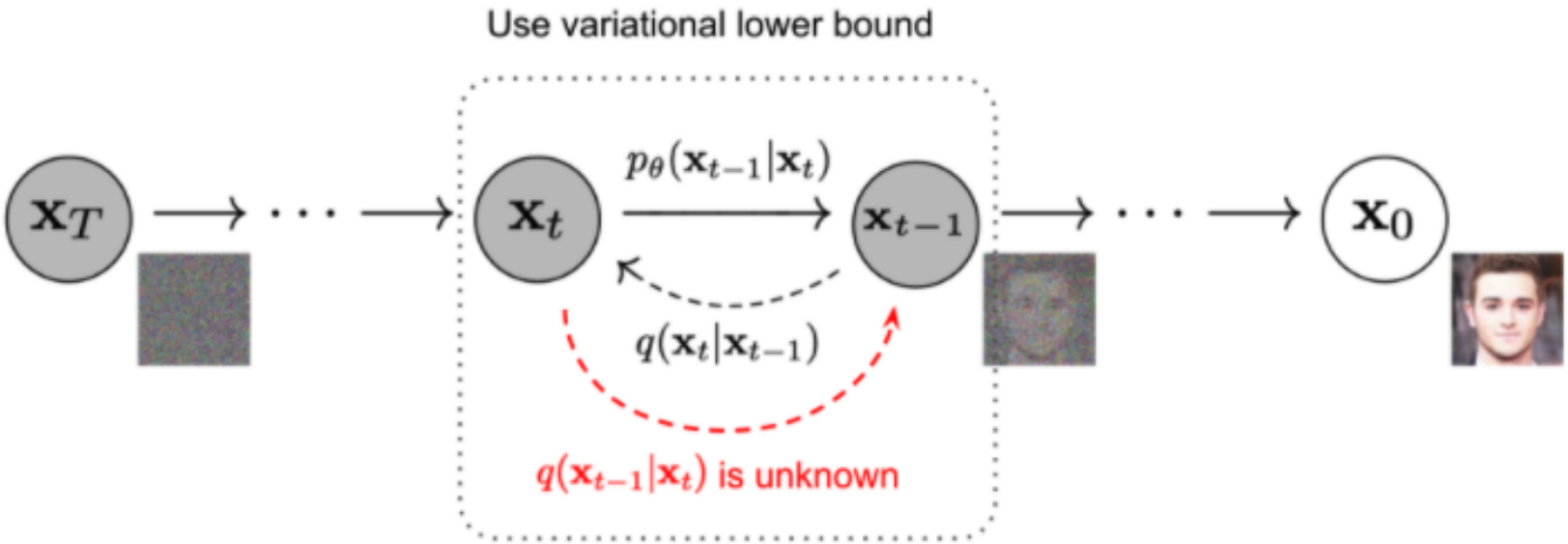
Lecture Outline

1. Score Functions
   - Motivation
   - Definitions
   - Score estimation
2. Score Matching
   - Fisher Divergences
   - Score Matching
   - Sliced Score Estimation
3. Sample Generation
   - Langevin Dynamics
   - Manifold Hypothesis
   - Gaussian Smoothing
4. Connections to Diffusion Models

Figures and contents adapted from Stefano Ermon and Lily Weng

Diffusion Models: Intuition

The intuition behind diffusion models is to define a process for gradually turning data to noise, and learning the inverse of this process.



- Noise process $q(x_t|x_{t-1})$ is defined by user.
- Forward process $q(x_{t-1}|x_t)$ is unknown.
- We approximate it by learning a model $p(x_{t-1}|x_t)$ of $q(x_{t-1}|x_t)$.

Forward Diffusion (Noising) Process

To define a diffusion model, we start by defining the noising or forward diffusion process.

- We start with data samples $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
- We run a Markov chain that gradually adds noise to the data, producing sequence of increasingly noisy samples $\mathbf{x}_1, ...\mathbf{x}_T$.
- At each step t, we sample x_t from the following Markov operator:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

  where $\beta_t \in (0, 1)$, $\beta_t \to 1$.
- At the end, $\mathbf{x}_T$ is a standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$

Forward Diffusion (Noising) Process

Next, we want to learn a model that undoes the noising process.

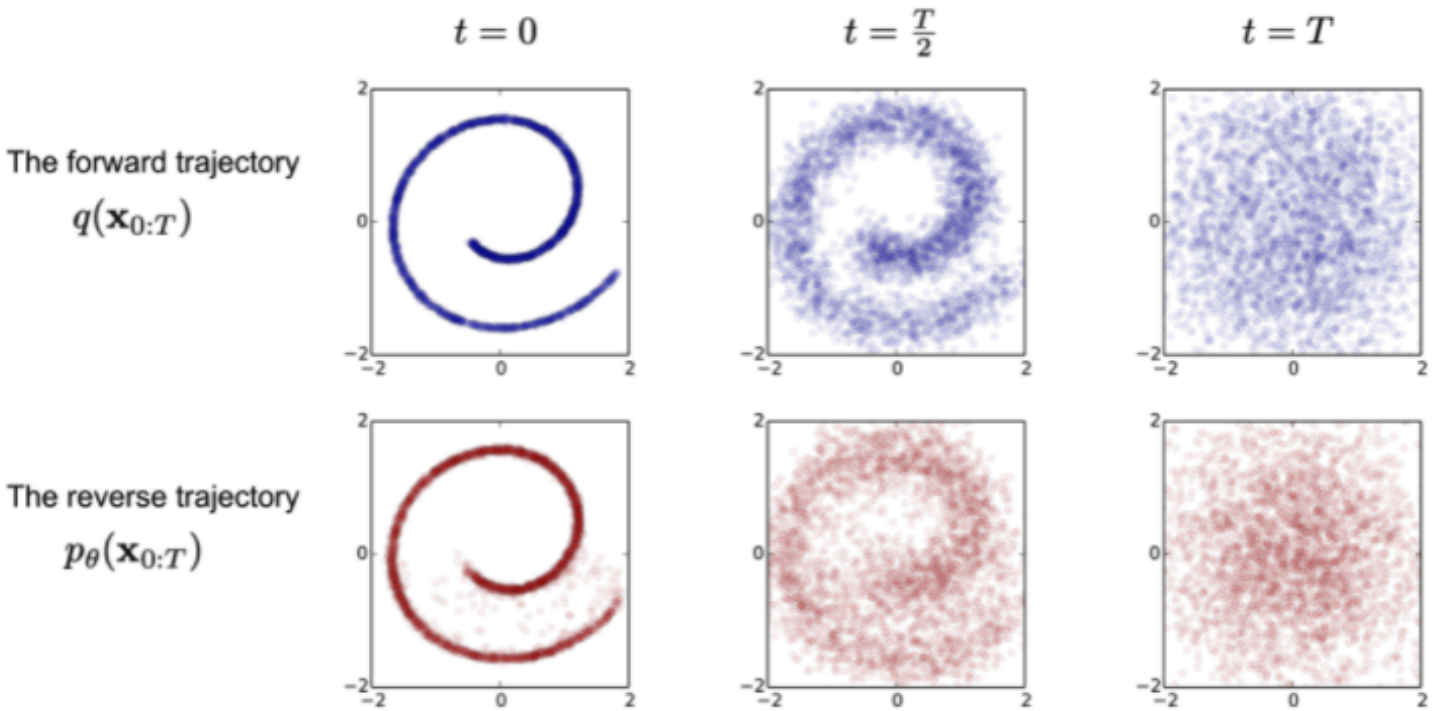
- The ideal denoising process is the inverse of the above Markov chain. At time t we sample from $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$
- We don't know this process. Therefore, we will try to learn it from running the noising process.
- We define out model as a distribution $p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ with the following structure:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

- To generate from this model, we sample noise $\mathbf{x}_T$, and sample from $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ until we get an image $\mathbf{x}_0$.

Forward and Backward Diffusion: Example

Here is the result of running forward (top) and backward (bottom) diffusion processes on a 2D spiral dataset:



Learning Diffusion Models via Max Likelihood

We may learn diffusion models by maximizing an evidence lower bound on the likelihood $\log p(\mathbf{x}_0)$ of a data point $\mathbf{x}_0$:

$$
\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{\mathrm{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)\|p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)}\right] \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0)\right] \\
&= \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \\
\text{Let } L_{\mathrm{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0)
\end{aligned}
$$

Transforming the ELBO Bound

We can use the conditional independence structure of p and q to transform the ELBO into a sum of simpler terms:

$$L_{\text{VLB}} = \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] = \mathbb{E}_q\Big[\log \frac{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T)\prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\Big]$$

$$= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big]$$

$$= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \Big(\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} \cdot \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}\Big) + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big]$$

$$= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big]$$

$$= \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\Big]$$

$$= \mathbb{E}_q[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^T \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}]$$

A More Convenient ELBO

This transformation yields the following ELBO:

$$
\begin{aligned}
L_{\text{VLB}} &= L_T + L_{T-1} + \cdots + L_0 \\
\text{where } L_T &= D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T)) \\
L_t &= D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \leq t \leq T-1 \\
L_0 &= -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)
\end{aligned}
$$

It's not hard to show that $q(\mathbf{x}_t|\mathbf{x}_0)$ has an analytic form:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$ and from that we can derive that

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I})$$

This gives us all the ingredients to learn these models by gradient descent.

Connection to Score-Based Models

Diffusion and score-based models are closely related:

- We may train score-based models using an alternative denoised score matching objective (Vincent et al., 2011)
- There also exists an alternative reparameterization of the diffusion model objective in which we try to learn the "noise" $z_t = x_{t+1} - x_t$ instead of the parameters of the Gaussian.
- These two objectives are almost identical to each other. See Ho et al., 2020 for details.

Score-Based Generative Models: Pros and Cons

Pros:

1. Make very few model assumptions
2. Can train energy-based models without manipulating normalizing constant
3. Can produce very high quality samples (matching GANs)

Cons:

1. Sampling is still quite slow (need to run MCMC chain)
2. Do not always provide likelihoods (though we get them for certain models)