Deep Generative Models: Score Matching with Diffusion Models

Fall Semester 2026

René Vidal
Director of the Center for Innovation in Data Engineering and Science (IDEAS)
Rachleff University Professor, University of Pennsylvania
Amazon Scholar & Chief Scientist at NORCE

Penn
University of Pennsylvania

Diffusion Models

- Derivation of Diffusion Models + Image Editing Applications (Last Lectures)
  - Markov Hierarchical Variational Auto Encoders (MHVAE)
  - Diffusion Models are VAEs with Linear Gaussian Autoregressive latent space
  - ELBO for Diffusion Models is a particular case of ELBO for VAEs with extra structure
  - Implementation Details
  - Latent Diffusion Models (Stable Diffusion) + Controllable generation

- **A Different Viewpoint of Diffusion Models (Today's Lecture)**
  - Denoising Score Matching

Connection with DDPM and Score-based Models



Stochastic Processes and their Applications 12 (1982) 313–326
North-Holland Publishing Company

REVERSE-TIME DIFFUSION EQUATION MODELS*

Brian D.O. ANDERSON**
Department of Electrical Engineering, The University of Newcastle, N.S.W. 2308, Australia

Received 13 November 1979
Revised 12 May 1980

Reverse-time stochastic diffusion equation models are defined and it is shown how most processes defined via a forward-time or conventional diffusion equation model have an associated reverse-time model.

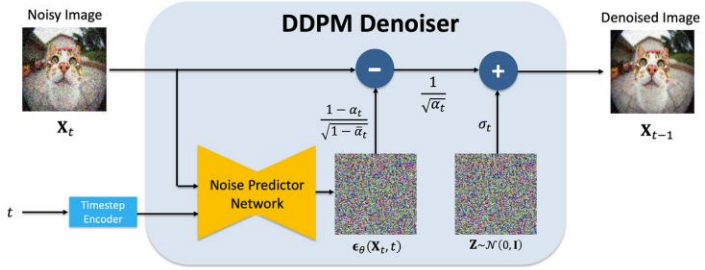
Stochastic equations	diffusion equations
reverse-time equations	Kolmogorov equations
Fokker–Planck equations	

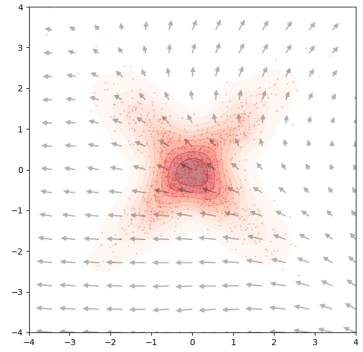
SDE Perspective (1982)

DDPM (2020)

Score-based models (2020)

$$\mathbf{X}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{X}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{X}_t, t)\right) + \sigma_t \mathbf{Z}$$





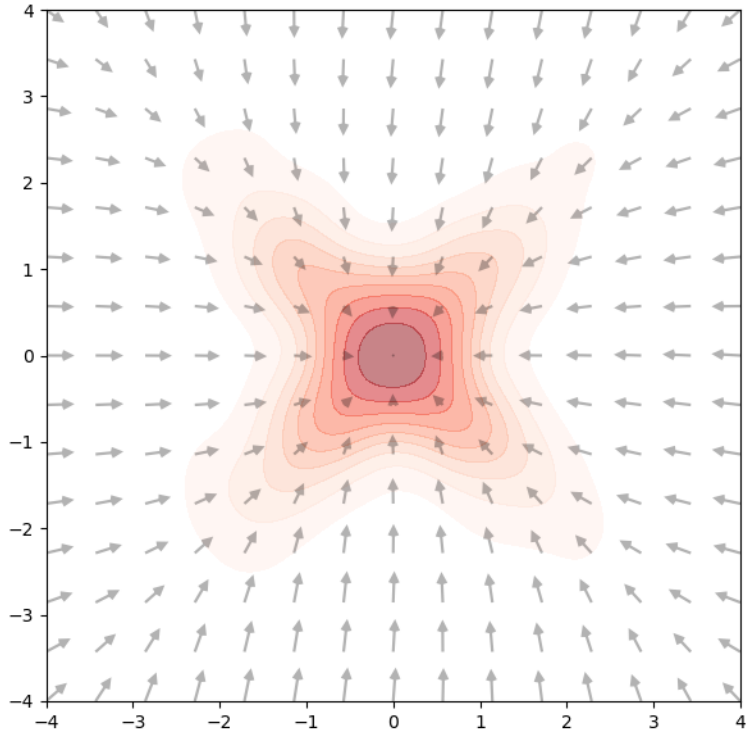
Generative Modeling via Score Functions

- **Goal**: Draw samples from $p(\boldsymbol{x})$
  - Rich history of sampling methods (MCMC, particle filter, etc.) – we won't cover these
- **So far**: Learning $p(\boldsymbol{x})$ by assuming a latent variable model
  - Recall PPCA, GMM, VAE, HMM, LDS, DDPM
  - Sampling is easy, since one can sample latent variables from $p(\mathbf{z})$ and data from $p(\boldsymbol{x}|\boldsymbol{z})$
- **Today**: Instead of learning $p(\boldsymbol{x})$ directly, we learn $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$
  - This quantity is known as the ***score function***
- **Challenges**
  - **Sampling:** Even if we had access to the score function, how do we sample from $p(\boldsymbol{x})$?
  - **Score Estimation:** How can we estimate score function from data?
  - **Diffusion:** What does this have to do with diffusion models?

Sampling: Intuition for Score Function

- The score function $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$ is a vector field (fn. from $\mathbb{R}^d$ to $\mathbb{R}^d$) that, at any point in data space, points to the direction of steepest increase in log-likelihood

The arrows indicate the score function vector field – they point towards the mode



Data distribution on $\mathbb{R}^2$
Darker colors indicate higher probability density

Sampling: Naïve Idea

$$\frac{d\boldsymbol{x}_t}{dt} = \lim_{\eta \to 0} \frac{\boldsymbol{x}_{t+\eta} - \boldsymbol{x}_t}{\eta}$$

- Simple sampling strategy: start at any point in data space and take some steps in the direction of the score function (gradient ascent)

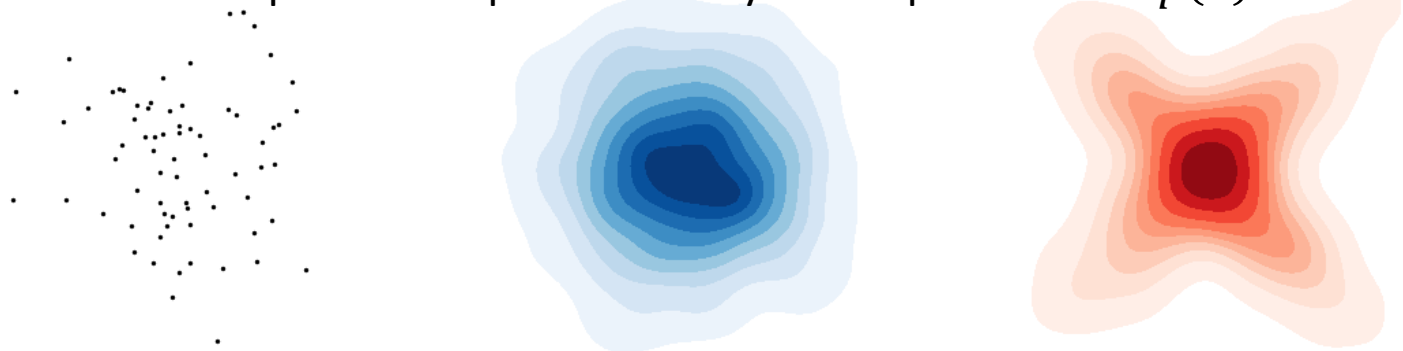
$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \eta \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{x}_t}$$

- If $\eta \to 0$, this process is described by the Ordinary Differential Equation (ODE)

$$d\boldsymbol{x}_t = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{x}_t} dt$$

- However, the trajectories of this ODE converge to samples from only the modes of the distribution: i.e., the ODE does not explore lower probability regions

Different Samples | Empirical Density of Samples | $p(\boldsymbol{x})$



Sampling: Insights from Physics

- French physicist Paul Langevin proved exploration can be done by adding noise

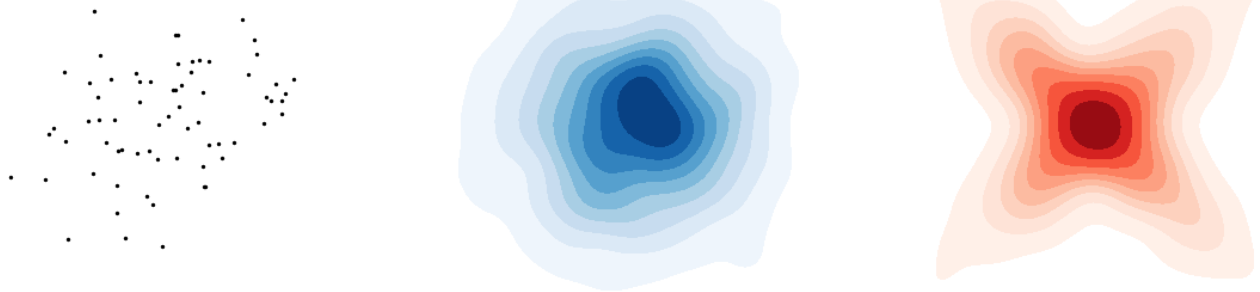
$$d\boldsymbol{x}_t = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{x}_t} dt + \sqrt{2} d\boldsymbol{w}_t$$

- This gives a Stochastic Differential Equation (SDE), where the Brownian motion $d\boldsymbol{w}_t$ is Gaussian noise with infinitesimally small variance:

$$d\boldsymbol{w}_t = \sqrt{dt}\, \boldsymbol{z}_t, \qquad \boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{z}_t; \mathbf{0}, \boldsymbol{I})$$

- Langevin proved that, as $t \to \infty$, the solutions to the SDE visit each $\boldsymbol{x}$ with probability $p(\boldsymbol{x})$. That is, this process draws samples from $p(\boldsymbol{x})$!

Different Samples | Empirical Density | $p(\boldsymbol{x})$



Langevin Dynamics Convergence Theorem

- **Theorem:** Consider the Langevin dynamics

$$d\boldsymbol{x}_t = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{x}_t} dt + \sqrt{2} d\boldsymbol{w}_t$$

where $\boldsymbol{w}_t$ is Brownian motion

$$d\boldsymbol{w}_t = \sqrt{dt}\, \boldsymbol{z}_t, \qquad \boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{z}_t; 0, \boldsymbol{I})$$

The process $(\boldsymbol{x}_t)_{t\geq 0}$ defines a **Markov diffusion** that admits $p(\boldsymbol{x})$ as a **unique invariant distribution**. Moreover, if p_t denotes the distribution of $\boldsymbol{x}_t$, then

$$p_t \underset{t\to\infty}{\to} p,$$

i.e., the distribution of $\boldsymbol{x}_t$ converges to p as $t \to \infty$.

Sampling as a Reverse Process

- Langevin dynamics resemble a reverse process: start with samples from a simple distribution such as $\mathcal{N}(0, \boldsymbol{I})$ and solve Langevin equation to sample from $p(\boldsymbol{x})$

$$d\boldsymbol{x}_t = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{x}_t} dt + \sqrt{2} d\boldsymbol{w}_t$$

- However, we don't know the distribution $p(\boldsymbol{x})$, hence we don't know the score function $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$. All we have are samples from $p(\boldsymbol{x})$.

- How can we estimate the score function $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$ from samples $p(\boldsymbol{x})$? Can we train a deep network that predicts the score by minimizing a reconstruction loss?

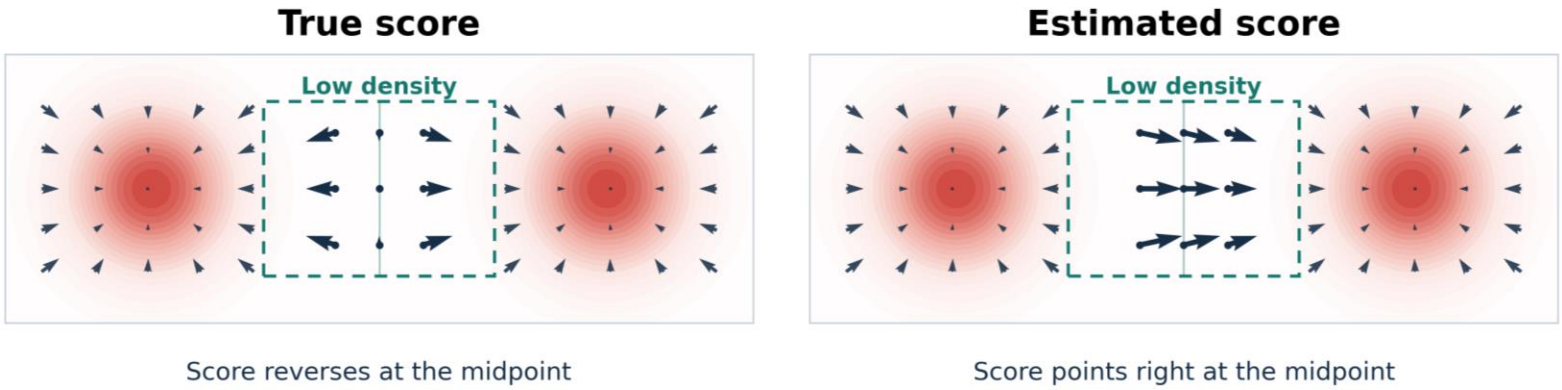
$$\min_\theta \frac{1}{2} \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})} \left[\left|\left| s_\theta(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) \right|\right|_2^2 \right]$$

Output of a deep network

Challenges with Learning Score Function

$$\min_\theta \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[\left|\left|s_\theta(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})\right|\right|_2^2\right]$$

- **Challenge 1**: Inaccurate score estimation in low data density regions



  - **Solution**: Perturb data with various levels of Gaussian noise, which ”fills” in low density regions of the space. We will define a family of perturbed distributions $(p_t(\boldsymbol{x}_t))_{t\geq 0}$

- **Challenge 2**: How do we optimize this because we don’t have ground truth score data to train on?
  - **Solution**: Clever manipulation of score function that we will see later

Perturbing Data: Forward Process

- We can arrive at the forward process mathematically by simply inverting the Langevin equation

- Instead of starting at $\mathcal{N}(0, \boldsymbol{I})$ and drawing samples from $p(\boldsymbol{x})$, what if we just start from $p(\boldsymbol{x})$, and follow score of standard Gaussian, and end at $\mathcal{N}(0, \boldsymbol{I})$?

$$d\boldsymbol{x}_t = \nabla_{\boldsymbol{x}} \log \mathcal{N}(\boldsymbol{x}; 0, \boldsymbol{I})\Big|_{\boldsymbol{x}=\boldsymbol{x}_t} dt + \sqrt{2} d\boldsymbol{w}_t$$
$$= -\boldsymbol{x}_t \, dt + \sqrt{2} d\boldsymbol{w}_t$$

- Recall that $d\boldsymbol{w}_t = \sqrt{dt}\, \boldsymbol{z}_t$, $\boldsymbol{z}_t \sim \mathcal{N}(0, \boldsymbol{I})$. After discretizing we get:

$$\boldsymbol{x}_{t+dt} = (1 - dt)\boldsymbol{x}_t + \sqrt{2dt}\boldsymbol{z}_t$$

- Looks like the forward process we know!

Sampling from perturbed distributions $p_t(\boldsymbol{x}_t)$

$$d\boldsymbol{x}_t = \nabla_{\boldsymbol{x}} \log \mathcal{N}(\boldsymbol{x}; 0, \boldsymbol{I})\Big|_{\boldsymbol{x}=\boldsymbol{x}_t} dt + \sqrt{2} d\boldsymbol{w}_t$$
$$= -\boldsymbol{x}_t \, dt + \sqrt{2} d\boldsymbol{w}_t$$

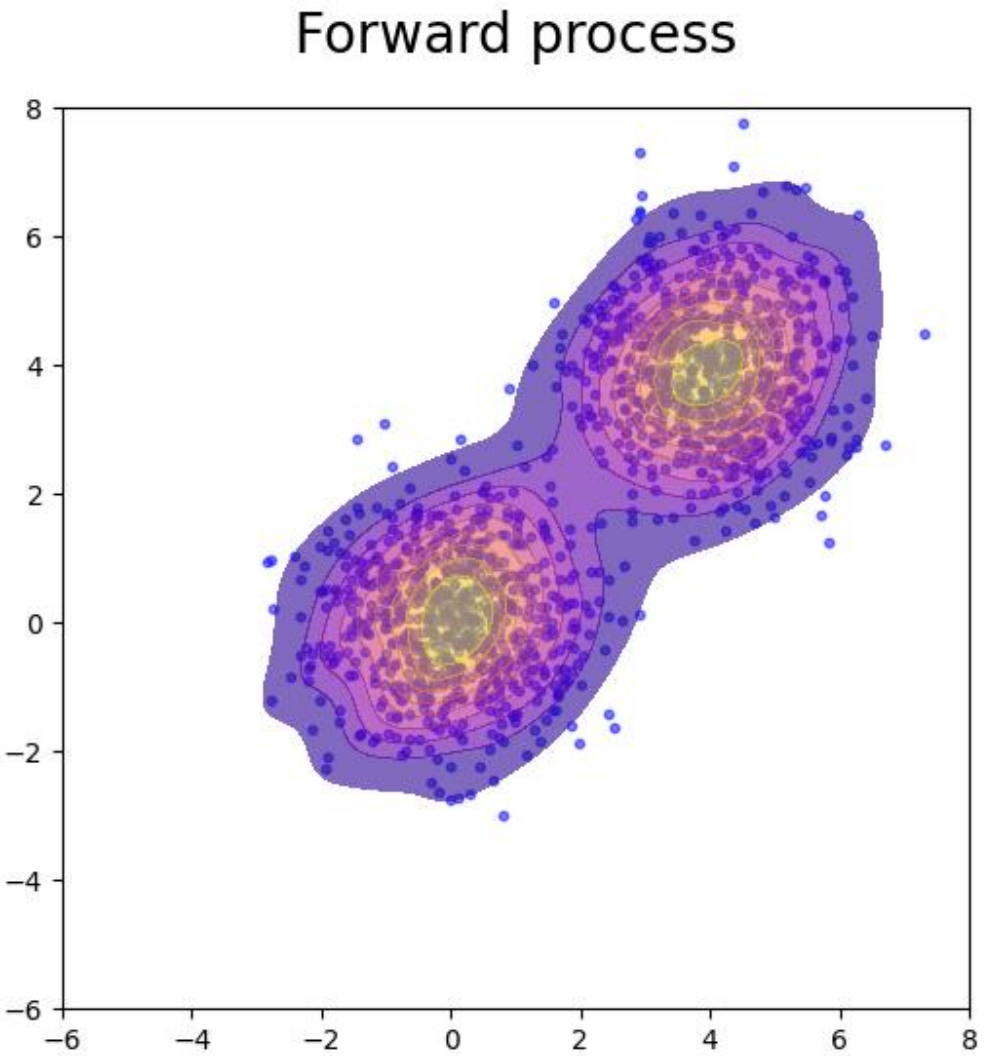
- Since the SDE is linear, $\boldsymbol{x}_t | \boldsymbol{x}_0$ is Gaussian

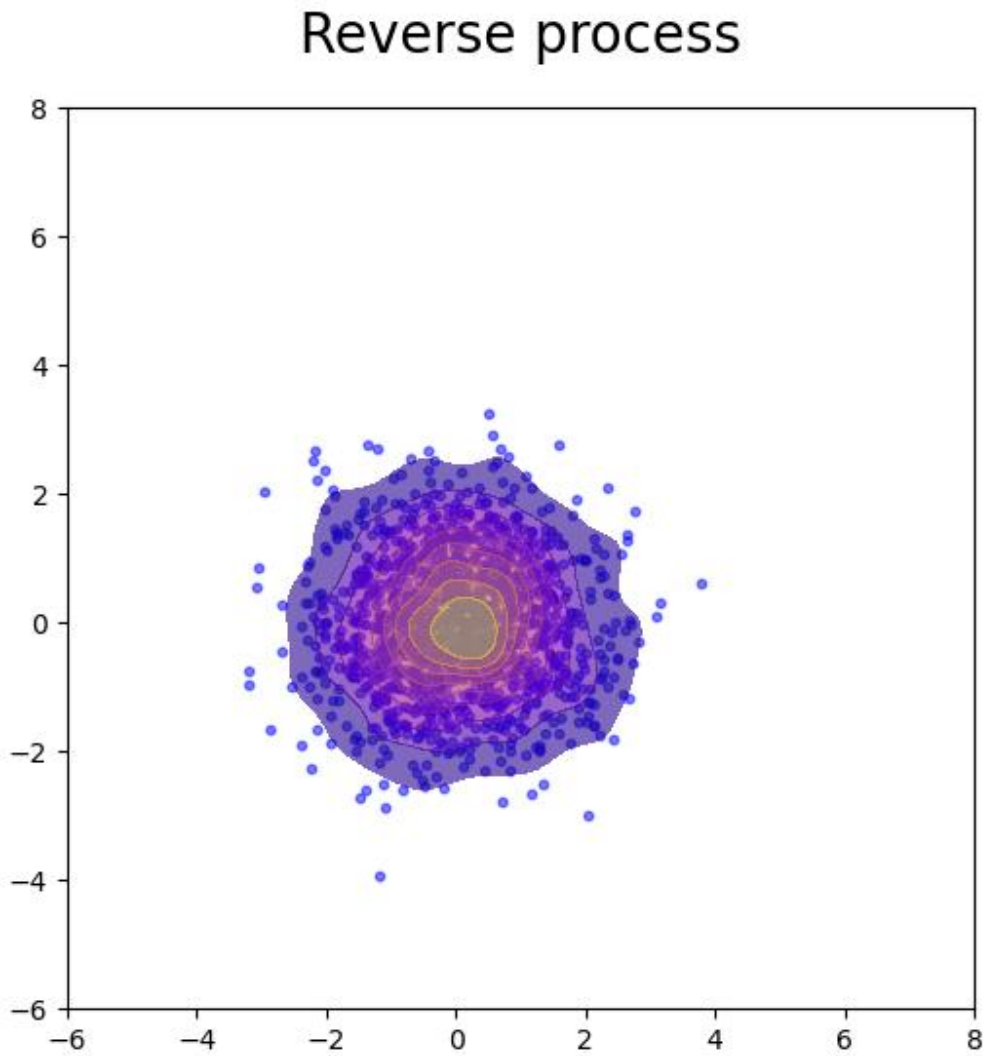
$$\boldsymbol{x}_t = e^{-t}\boldsymbol{x}_0 + \sqrt{1 - e^{-2t}}\boldsymbol{z}_t, \qquad \boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{z}_t; 0, \boldsymbol{I})$$

- Compare to

$$\boldsymbol{x}_t = \sqrt{\overline{\alpha_t}}\boldsymbol{x}_0 + \sqrt{1 - \overline{\alpha_t}}\,\overline{\boldsymbol{\epsilon}}_{\boldsymbol{t}}, \qquad \overline{\boldsymbol{\epsilon}}_{\boldsymbol{t}} \sim \mathcal{N}(\overline{\boldsymbol{\epsilon}}_{\boldsymbol{t}}; 0, \boldsymbol{I})$$

Back and forth from GMM to Normal distribution





Summary so Far

- We have shown that with the score function given, we can draw samples from the underlying data distribution using the Langevin equation
- We described the forward process to get a learning objective

$$\min_\theta \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}\{0,\dots,T\},\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t)}\left[\left|\left|s_\theta(\boldsymbol{x}_t,t)-\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t)\right|\right|_2^2\right]$$

- **Next**: How to learn time dependent score function without ground truth access to the score function?

Rewriting the Score Function (Tweedie's formula)

- From the forward process, we generated data

$$\boldsymbol{x}_t = \sqrt{\overline{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha}_t}\,\overline{\boldsymbol{\epsilon}}_t, \quad \overline{\boldsymbol{\epsilon}}_t \sim \mathcal{N}(\overline{\boldsymbol{\epsilon}}_t; 0, \boldsymbol{I})$$

- Let's cleverly rewrite the score function given this

$$\begin{aligned}
\nabla_{x_t} \log p_t(\boldsymbol{x}_t) &= \frac{\nabla_{x_t} p_t(\boldsymbol{x}_t)}{p_t(\boldsymbol{x}_t)} \\
&= \frac{1}{p_t(x_t)} \int \nabla_{x_t} p_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0) p_0(\boldsymbol{x}_0)\, d\boldsymbol{x}_0 \\
&= \frac{1}{p_t(x_t)} \int p_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0) \nabla_{x_t} \log p_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0) p_0(\boldsymbol{x}_0)\, d\boldsymbol{x}_0 \\
&= \int \nabla_{x_t} \log p_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0) p_{0|t}(\boldsymbol{x}_0|\boldsymbol{x}_t)\, d\boldsymbol{x}_0 \\
&= \int \frac{\sqrt{\overline{\alpha}_t}\boldsymbol{x}_0 - \boldsymbol{x}_t}{1-\overline{\alpha}_t} p_{0|t}(\boldsymbol{x}_0|\boldsymbol{x}_t)\, d\boldsymbol{x}_0 \\
&= \frac{\sqrt{\overline{\alpha}_t}\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t] - \boldsymbol{x}_t}{1-\overline{\alpha}_t}
\end{aligned}$$

$$\nabla_{x_t} \log p_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0) = \frac{\nabla_{x_t} p_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0)}{p_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0)}$$

$$p_{0|t}(\boldsymbol{x}_0|\boldsymbol{x}_t) = \frac{p_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0) p_0(\boldsymbol{x}_0)}{p_t(x_t)}$$

$$\nabla_x \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})$$

$$\nabla_x \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{\overline{\alpha}_t}\boldsymbol{x}_0, \sqrt{1-\overline{\alpha}_t} I\right) = -\frac{\boldsymbol{x}_t - \sqrt{\overline{\alpha}_t}\boldsymbol{x}_0}{1-\overline{\alpha}_t}$$

Denoising Score Matching

- Plugging into our objective before and simplifying, we have

$$\min_\theta \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}\{0,\dots,T\},\boldsymbol{x}_t\sim p(\boldsymbol{x}_t)}\left[\left\|\left\|s_\theta(\boldsymbol{x}_t,t)-\frac{\sqrt{\bar{\alpha}_t}\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t]-\boldsymbol{x}_t}{1-\bar{\alpha}_t}\right\|\right\|_2^2\right]$$

$$\mathbb{E}[||x||^2]=||\mathbb{E}[x]||^2+\mathrm{Var}(x)$$

$$\equiv \min_\theta \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}\{0,\dots,T\},\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t),\boldsymbol{x}_0\sim p_{0|t}(\boldsymbol{x}_0|\boldsymbol{x}_t)}\left[\left\|\left\|s_\theta(\boldsymbol{x}_t,t)-\frac{\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0-\boldsymbol{x}_t}{1-\bar{\alpha}_t}\right\|\right\|_2^2\right]$$

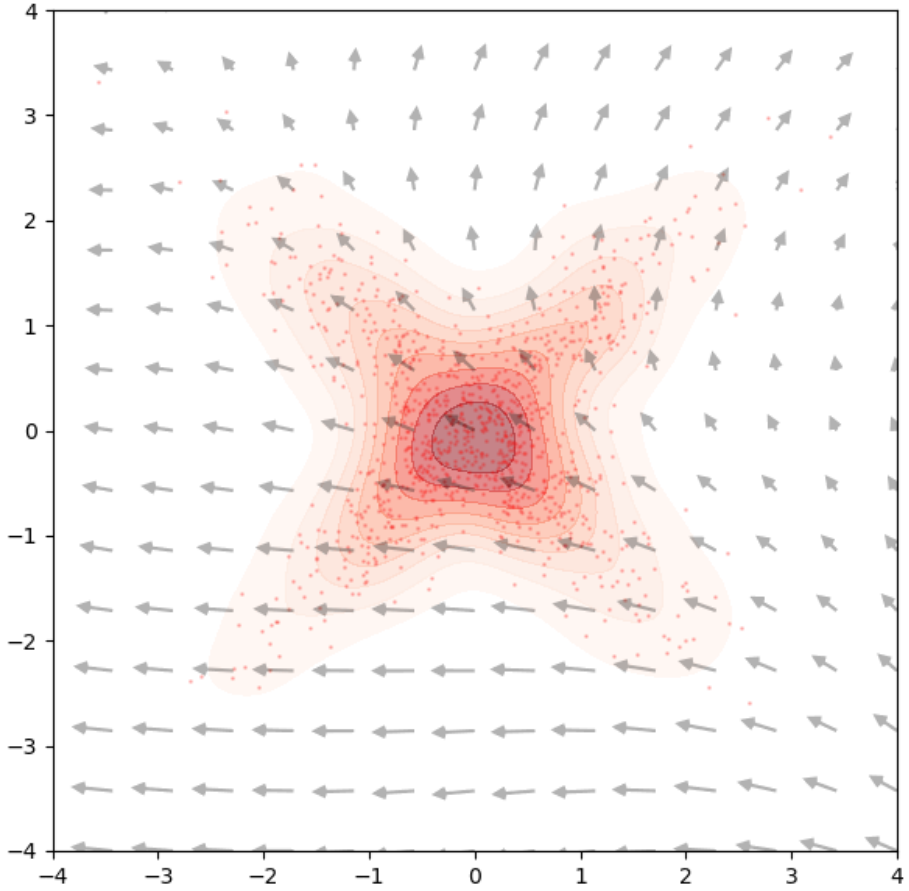
$$\bar{\boldsymbol{\epsilon}}_t=\frac{\boldsymbol{x}_t-\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0}{\sqrt{1-\bar{\alpha}_t}}$$

$$\equiv \min_\theta \mathbb{E}_{t\sim\mathcal{U}\{0,\dots,T\},\boldsymbol{x}_t,\boldsymbol{x}_0\sim p(\boldsymbol{x}_t,\boldsymbol{x}_0)}\left[\frac{1}{2(1-\bar{\alpha}_t)}\left|\left|\hat{\boldsymbol{\epsilon}}_\theta(\boldsymbol{x}_t,t)-\bar{\boldsymbol{\epsilon}}_t\right|\right|_2^2\right]$$

$$\hat{\boldsymbol{\epsilon}}_\theta(\boldsymbol{x}_t,t):=-\sqrt{1-\bar{\alpha}_t}s_\theta(\boldsymbol{x}_t,t)$$

- We have a way to estimate the score without ever needing ground truth score, which is remarkable
- Moreover, this is exactly our denoising objective from diffusion models!
  - Maximizing our ELBO is equivalent to learning the score function at $\boldsymbol{x}_t$!

Learning Score Function Example



Why is this viewpoint useful?

- It unifies diffusion models' forward process as following Langevin equation to transform data distribution into Gaussian distribution
  - Different noise schedules just correspond to how fast this transition is
- Connects learning denoisers with estimating the score function, which we can then use to sample from the distribution
- Allows us to naturally think of continuous-time diffusion models
- Our rewriting of the score function is called **Tweedie's formula**
  - Can use it to estimate $\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t]$ given the score function – one step denoising
  - Precisely what is used in DDIM to predict the $\boldsymbol{x}_0$
- Conditional Sampling becomes straightforward (see next slide)

Conditional Diffusion Models - Guidance

- Conditional Sampling: Sample from $p(\boldsymbol{x} \mid y)$ where y is another image, text, etc.
- The score function for this distribution can be obtained via Bayes' rule

$$\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|y) = \nabla_{\boldsymbol{x}_t} \log p(y \,|\boldsymbol{x}_t) + \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$$

- The second term is already modeled by an unconditional diffusion model!
- For the first term, we could use a classifier. This is known as classifier guidance, where we scale the effect of classifier by some factor γ in front of the first term



Samples from an unconditional diffusion model with classifier guidance, for guidance scales 1.0 (left) and 10.0 (right), taken from Dhariwal & Nichol (2021).

Classifier-free Guidance

- Classifier Guidance required access to a classifier that can handle noisy inputs
- Classifier-free Guidance allows us to do guidance without training any auxiliary classifiers

$$\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|y) = \gamma \nabla_{\boldsymbol{x}_t} \log p(y\,|\boldsymbol{x}_t) + \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$$
$$= \gamma(\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|y) - \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)) + \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$$
$$= (1\ -\gamma)\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t) + \gamma \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|y)$$

- $\gamma > 1$ is found to be the regime that works well
- Train a diffusion model with conditioning dropout: some percentage of the time (10-20%), just remove the conditioning information, so the model learns to be both a conditional and unconditional model

$$q_{\gamma,t}(x_t \mid y) \propto p_t(x_t)\, p_t(y \mid x_t)^{\gamma} \propto p_t(x_t \mid y)\, p_t(y \mid x_t)^{\gamma-1}$$

Classifier Free Guidance Results

Seems to work better than classifier guidance because the “classifier” is constructed from the generative model itself



Two sets of samples from OpenAI's GLIDE model, for the prompt *'A stained glass window of a panda eating bamboo.'*, taken from their paper. Guidance scale 1 (no guidance) on the left, guidance scale 3 on the right.



Two sets of samples from OpenAI's GLIDE model, for the prompt *'"A cozy living room with a painting of a corgi on the wall above a couch and a round coffee table in front of a couch and a vase of flowers on a coffee table.'*, taken from their paper. Guidance scale 1 (no guidance) on the left, guidance scale 3 on the right.

Conclusion

- We saw how diffusion models implicitly learn the score function and perform reverse process sampling using the Langevin equation
  - Connects to many core topics in stochastic differential equations and clean mathematical derivations
- Used the score function viewpoint to easily derive guidance (classifier and classifier-free) for conditional sampling from diffusion models
- Three great resources for more information
  - ICLR Blogpost – Building Diffusion Models Theory From Scratch (https://iclr-blogposts.github.io/2024/blog/diffusion-theory-from-scratch/)
  - Sander Dieleman Blog – Guidance: A Cheat Code for Diffusion Models (https://sander.ai/2022/05/26/guidance.html)
  - See references in the blog posts