Deep Generative Models: Score Matching with Diffusion Models

Fall Semester 2024

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

Diffusion Models

- **Derivation of Diffusion Models + Image Editing Applications (Last Lecture)**
  - 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

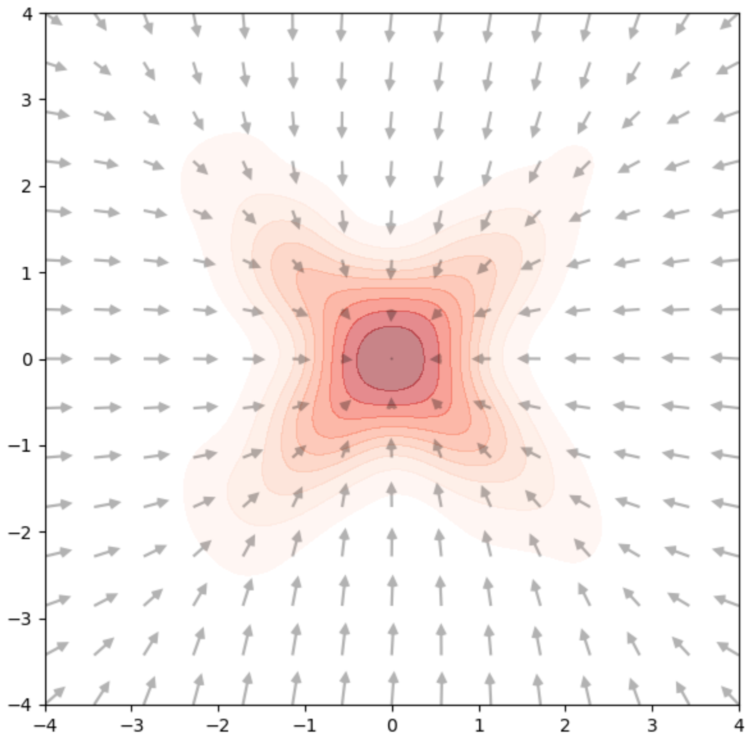
Generative Modelling via Score Functions

- Goal: Draw samples from $p(x)$
    - Rich history of sampling methods (MCMC, particle filter, etc.) – we won’t cover these
- So far: Learning $p(x)$ by assuming latent variable model
    - Recall PPCA, GMM, VAE, HMM, LDS, DDPM
    - Sampling is easy, since one can sample latent variables z and then x from $p(x|z)$
- Today: Instead of learning $p(x)$ directly, we can turn to learn $\nabla_x \log p(x)$
    - This quantity is known as the *score function*
- Challenges
    - **Sampling:** Even if we had access to score function, how do we draw samples from $p(x)$?
    - **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 $\nabla_x \log p(x)$ is a vector field that points to the direction of steepest increase in log-likelihood at any given point in data space

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

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

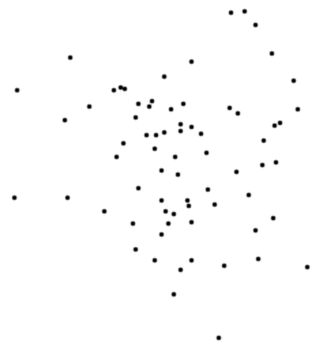
$$x_{t+1} = x_t + \eta \nabla_x \log p(x) \Big|_{x=x_t}$$

When $\eta \to 0$, this process is described as ODE

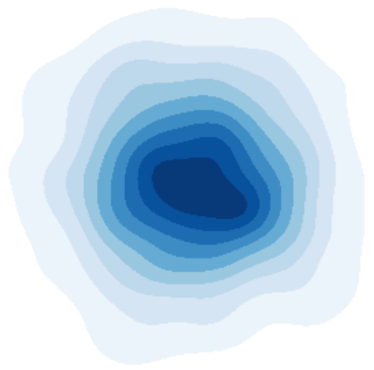
$$dx = \nabla_x \log p(x) dt$$

- But this is greedy! And it will only give samples from the modes of the distribution i.e. it does not explore

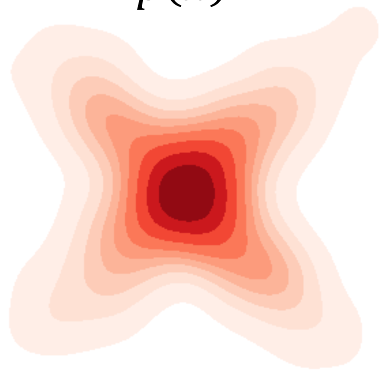
Different Samples



Empirical Density of Samples



$p(x)$



Sampling: Insights from Physics

- Turns out this process was well-studied in molecular dynamics by French physicist Paul Langevin
  - He found that exploration can be done by adding some random noise to the samples.

Langevin equation $$dx = \nabla_x \log p(x) dt + \sqrt{2} dB_t$$

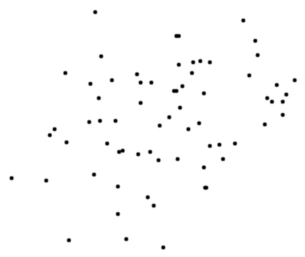
Discrete-time version $$x_{t+1} = x_t + \eta \nabla_x \log p(x) + \sqrt{2\eta}\, z$$ where $z \sim N(0, I)$

Source of noise: "Brownian motion" and is basically Gaussian noise with infinitesimally small variance

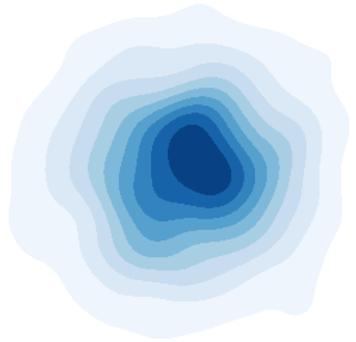
$$dB_t = \sqrt{dt}\, z\, , z \sim N(0, I)$$

  - He proved that if you run this for infinite time, this visits each x with probability $e^{\log p(x)} = p(x)$, so this draws samples from $p(x)$!

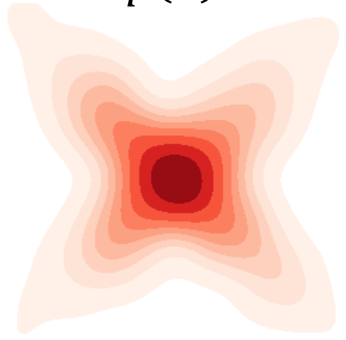
Different Samples



Empirical Density



$p(x)$



Sampling as a Reverse Process

- Already, this procedure resembles a "reverse process": we start with samples from some simple distribution such as $N(0, I)$ and then follow Langevin equation to get sample from $p(x)$
- We need to estimate the score function: neural networks to the rescue?

$$dx_t = \nabla_x \log p(x) \Big|_{x=x_t} dt + \sqrt{2} dB_t$$

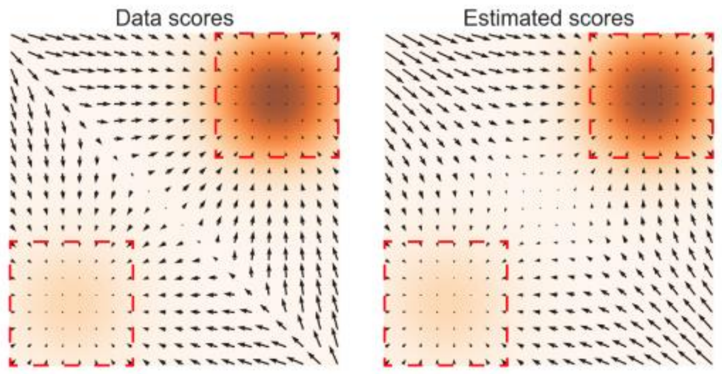
- How about we just start from a loss function that we can minimize to get to the true score?

$$\text{argmin}_\theta \frac{1}{2} E_{x \sim p(x)} \left[\left| \left| s_\theta(x) - \nabla_x \log p(x) \right| \right|_2^2 \right]$$

Challenges with Learning Score Function

$$\text{argmin}_\theta \frac{1}{2} E_{x\sim p(x)}\left[\left|\left|s_\theta(x) - \nabla_x \log p(x)\right|\right|_2^2\right]$$

- 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

- 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 $N(0, I)$ and drawing samples from $p(x)$, what if we just start from $p(x)$, follow score of standard Gaussian, and end at $N(0, I)$?

$$\begin{aligned} dx &= \nabla_x \log N(0, I) dt + \sqrt{2} dB_t \\ &= -x \, dt + \sqrt{2} dB_t \end{aligned}$$

  - Discretizing,

$$x_{t+dt} = (1 - dt) x_t + \sqrt{2dt} \, \mathrm{z}$$

  - Looks like the forward process we already know!

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 (for all t)

$$\mathrm{argmin}_\theta \frac{1}{2} E_{x\sim p(x)}\left[\left|\left|s_\theta(x,t) - \nabla_{x_t} \log p(x_t)\right|\right|_2^2\right]$$

- Next up: How to actually learn this without ground truth access to the score function?

Rewriting the Score Function

- From the forward process, we generated data $x_t = \sqrt{\overline{\alpha_t}}x_0 + \sqrt{1 - \overline{\alpha_t}}\,\epsilon, \epsilon \sim \mathcal{N}(\epsilon; 0, I)$
- Let's cleverly rewrite the score function given this

$$
\begin{aligned}
\nabla_{x_t} \log p(x_t) &= \frac{\nabla_{x_t} p(x_t)}{p(x_t)} \\
&= \frac{1}{p(x_t)} \int \nabla_{\mathrm{x_t}} \mathrm{p}(\mathrm{x_t}|\mathrm{x_0})\mathrm{p}(\mathrm{x_0})\, dx_0 \\
&= \frac{1}{p(x_t)} \int p(x_t|x_0) \nabla_{\mathrm{x_t}} \log \mathrm{p}(\mathrm{x_t}|\mathrm{x_0})\mathrm{p}(\mathrm{x_0})\, dx_0 \\
&= \int \nabla_{\mathrm{x_t}} \log \mathrm{p}(\mathrm{x_t}|\mathrm{x_0})\mathrm{p}(\mathrm{x_0}|x_t)\, dx_0 \\
&= \int \frac{\sqrt{\bar{\alpha}_t} x_0 - x_t}{1 - \bar{\alpha}_t} \mathrm{p}(\mathrm{x_0}|x_t)\, dx_0 \\
&= \frac{\sqrt{\bar{\alpha}_t} E[x_0|x_t] - x_t}{1 - \bar{\alpha}_t}
\end{aligned}
$$

Denoising Score Matching

- Plugging into our objective before and simplifying, we have

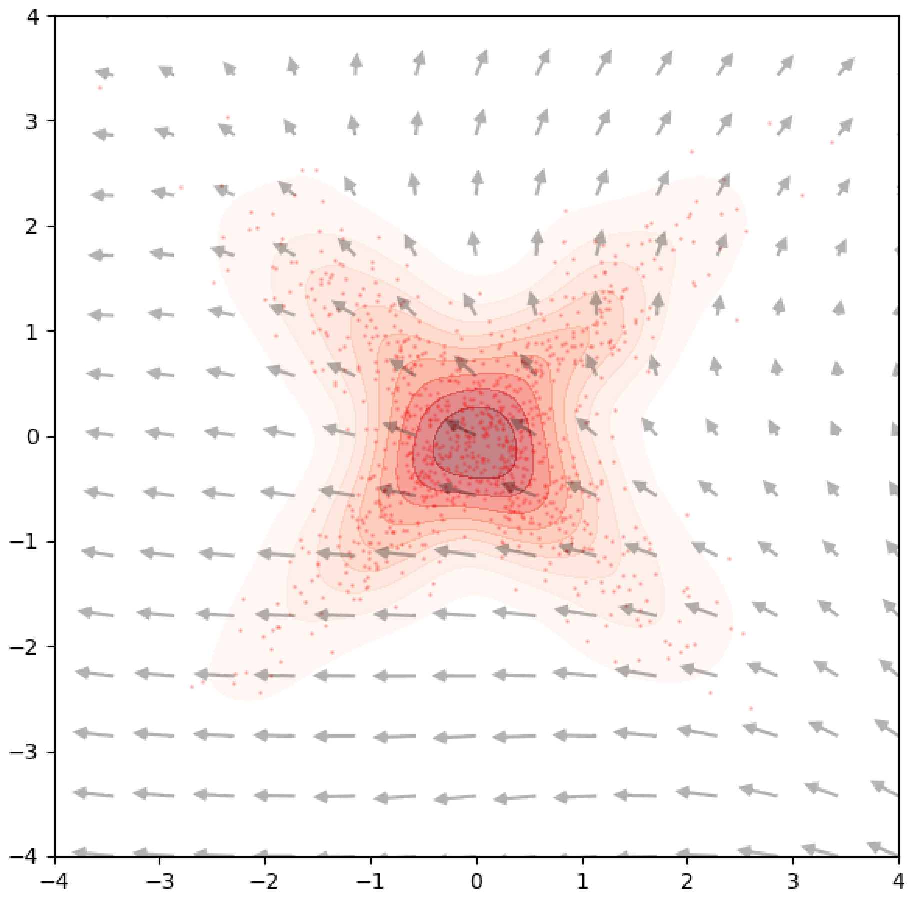
$$\text{argmin}_\theta \ \frac{1}{2} E_{x_t \sim p(x_t)} \left[\left\| s_\theta(x, t\,) - \frac{\sqrt{\bar{\alpha}_t} E[x_0 | x_t] \ - x_t}{1 \ - \bar{\alpha}_t} \right\|_2^2 \right]$$

$$\text{argmin}_\theta \ \frac{1}{2} E_{x_t \sim p(x_t), x_0 \sim p(x_0|x_t)} \left[\left\| s_\theta(x, t\,) - \frac{\sqrt{\bar{\alpha}_t} x_0 - x_t}{1 \ - \bar{\alpha}_t} \right\|_2^2 \right]$$

$$\text{argmin}_\theta \ \frac{1}{2(1 \ - \bar{\alpha}_t)} E_{x_t\,, x_0 \sim p(x_t, x_0)} \left[|| \epsilon_\theta(x, t\,) - \epsilon ||_2^2 \right]$$

- 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 x_t!

Learning Score Function Example



Why is this viewpoint useful?

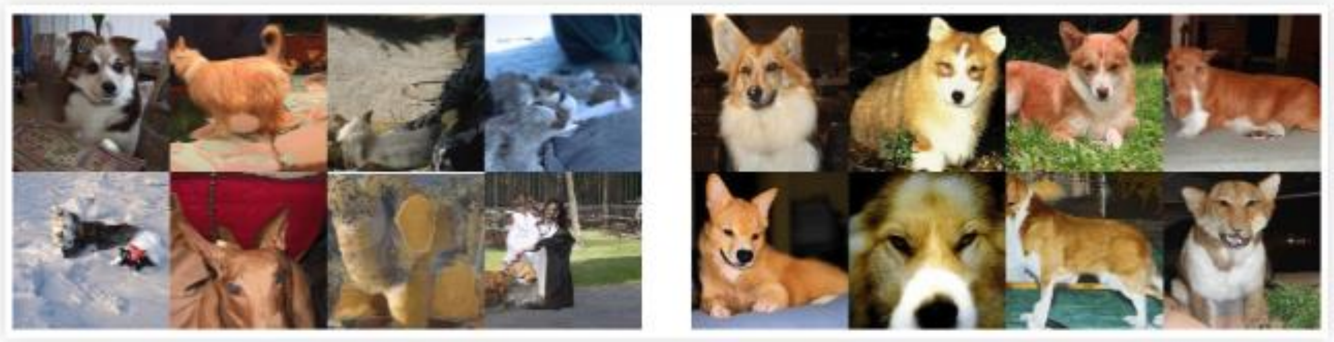
- It unifies diffusion model forward process as following Langevin equation to transform data distribution into Gaussian distribution
  - Different noise schedules just correspond to how fast this transition is
- Connects learning denoisers with estimating the score function, which we can then use to 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 solve $E[x_0|x_t]$ given the score function – one step denoising
  - Precisely what is used in DDIM to predict the x_0
- Conditional Sampling becomes straightforward (see next slide)

Conditional Diffusion Models - Guidance

- Conditional Sampling: Sample from $p(x \,|y)$ where y is some conditioning term (another image, text, etc.)
- The score function for this distribution is via Bayes rule

$$\nabla_{x_t} \log p(x_t|y) = \nabla_{x_t} \log p(y \,|x_t) + \nabla_{x_t} \log p(x_t)$$

- The second term is already modelled by a unconditional diffusion model!
- For the first term, perhaps we can use a classifier?
    - This is known as classifier guidance, where you can scale 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

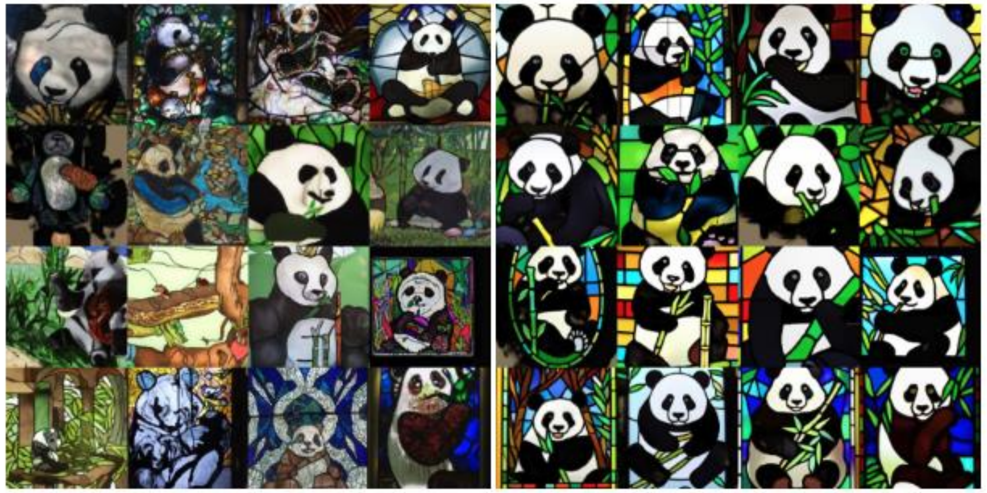
- Classifier Guidance required access to a classifier that can handle noisy inputs
- Classifier-free Guidance allows us a way to do guidance without training any auxiliary classifiers

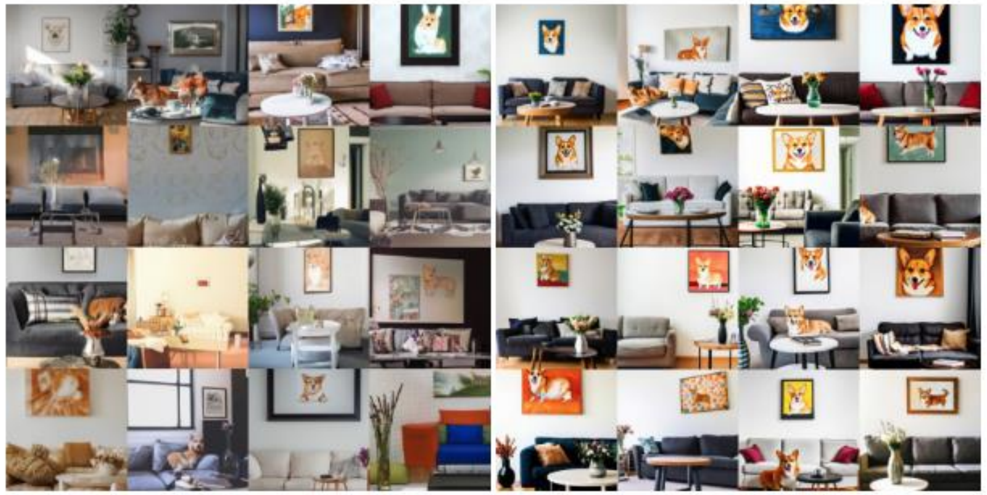
$$\begin{gathered}\nabla_{x_t} \log p_\gamma(x_t|y) = \gamma \nabla_{x_t} \log p(y\,|x_t) + \nabla_{x_t} \log p(x_t) \\ = \gamma(\nabla_{x_t} \log p(x_t|y) - \nabla_{x_t} \log p(x_t)) + \nabla_{x_t} \log p(x_t) \\ = (1\ - \gamma)\nabla_{x_t} \log p(x_t) + \gamma \nabla_{x_t} \log p(x_t|y)\end{gathered}$$

- $\gamma > 1$ 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

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