Lecture 2

CME 296: Diffusion & Large Vision Models

Afshine Amidi & Shervine Amidi

Stanford University

ICME

Recap of last episode...

Suppose we are given a **sample** of observations from p_{data}

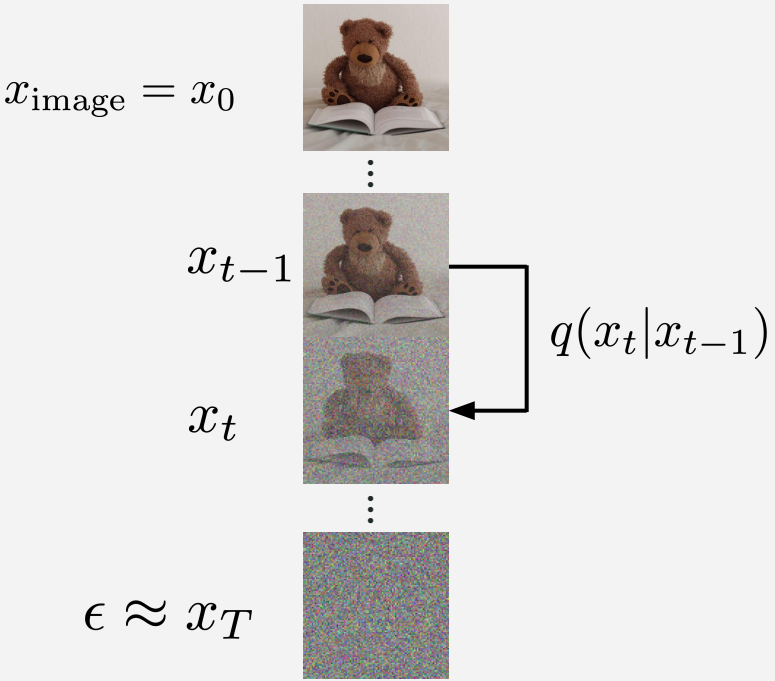
p_{data} 

Generated sample

Objective. Generate new images from a distribution of images p_{data}

Recap of last episode...

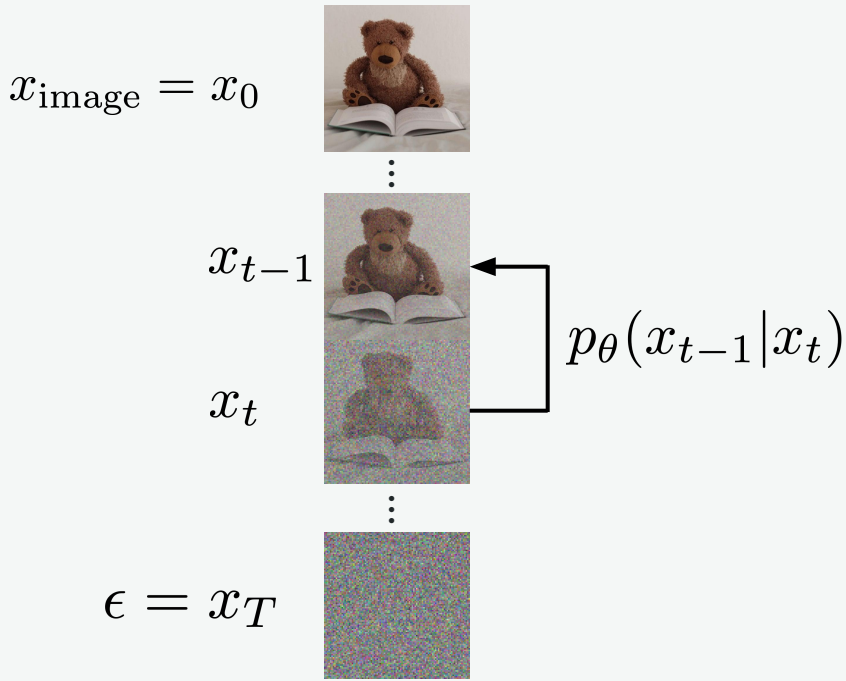
1. Add **noise** (**forward process**)



Recap of last episode...

1. Add **noise** (**forward process**)

2. Learn to **denoise** it (**reverse process**)



Recap of last episode...

Strategy.

1. Derive **lower bound** for maximum (log-)likelihood estimation ← **ELBO!**

Recap of last episode...

Strategy.

1. Derive **lower bound** for maximum (log-)likelihood estimation ← **ELBO!**
2. **Expand terms** of lower bound ← **Surfaced KL divergence**

Recap of last episode...

Strategy.

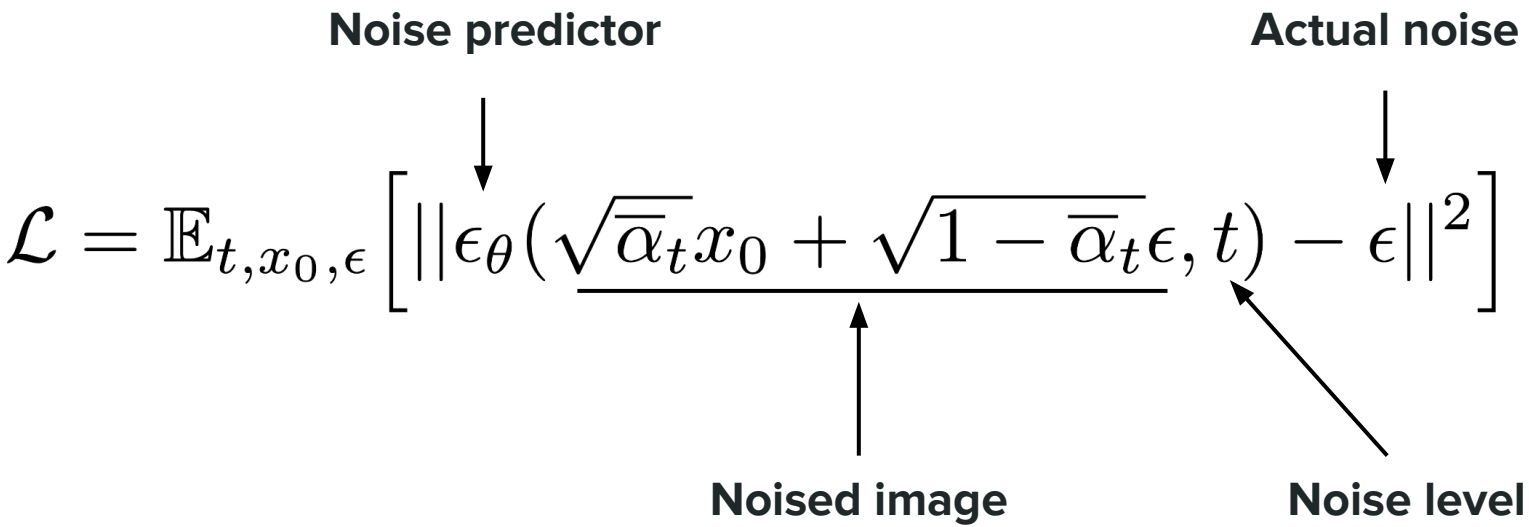
1. Derive **lower bound** for maximum (log-)likelihood estimation ← **ELBO!**
2. **Expand terms** of lower bound ← **Surfaced KL divergence**
3. Show lower bound is **tractable** ← **Bayes' rule + Gaussian properties**

Recap of last episode...

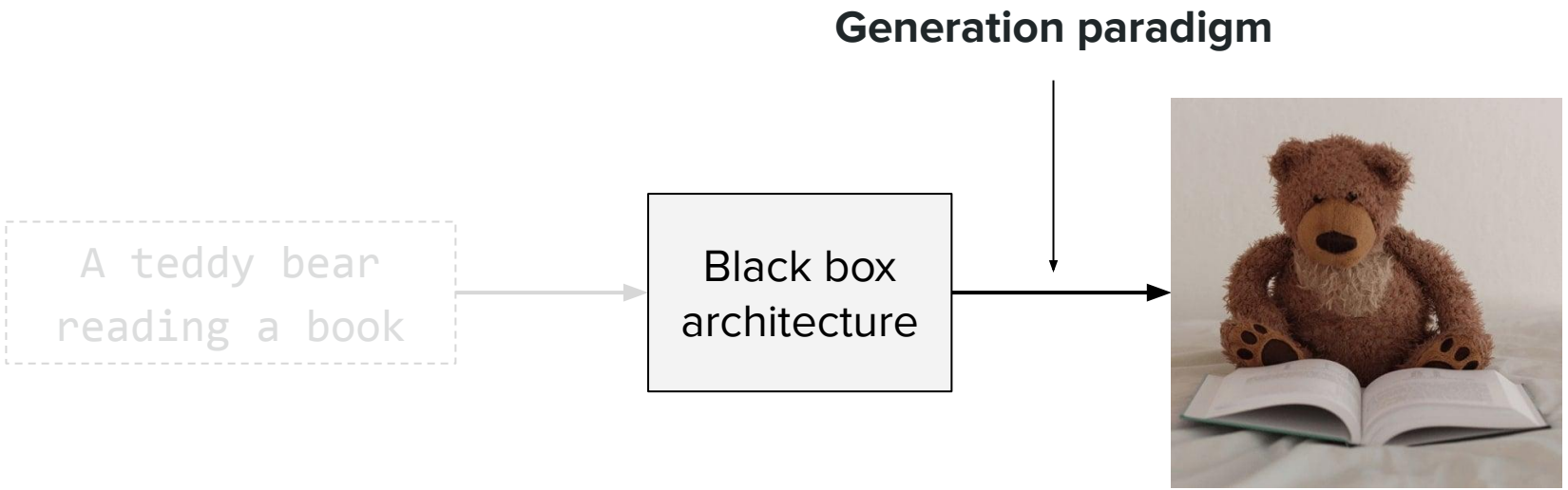
Strategy.

1. Derive **lower bound** for maximum (log-)likelihood estimation ← **ELBO!**
2. **Expand terms** of lower bound ← **Surfaced KL divergence**
3. Show lower bound is **tractable** ← **Bayes' rule + Gaussian properties**
4. Deduce **final loss function** ← **Incredibly simple noise prediction!**

Recap of last episode...



Today's lecture



Today's lecture: **Score matching**

Diffusion & Large Vision Models

Motivation

Score estimation

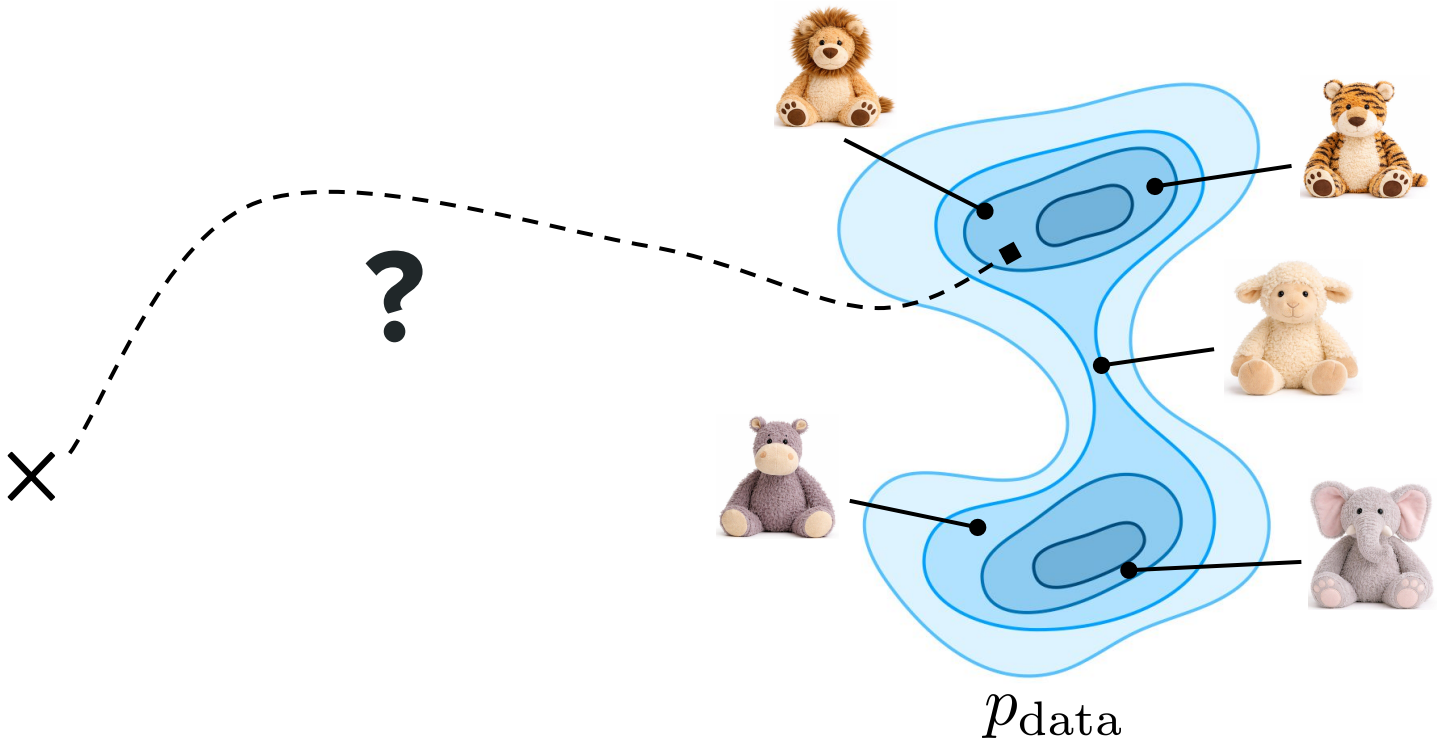
Differential formulation

Training

Inference

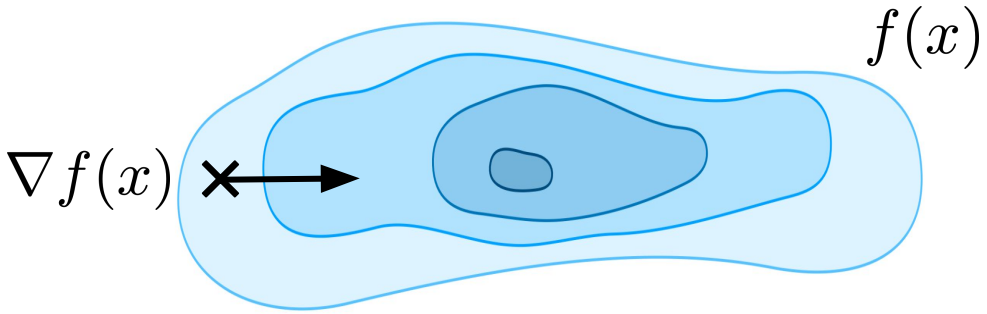
Probability flows

Intuition



Reminder of what a gradient is

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(x) \\ \frac{\partial f}{\partial x_2}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{pmatrix}$$



Intuition

$\times$ $\nabla_x p_{\text{data}}(x)$

Problems with gradient of probability

- Property of probabilities: $Z = \int p_{\text{data}}(x)dx = 1$

Therefore, $p_{\text{data}}(x) = \left(\frac{f(x)}{Z}\right)$

Intractable!

Problems with gradient of probability

- Property of probabilities: $Z = \int p_{\text{data}}(x)dx = 1$

Therefore, $\nabla_x p_{\text{data}}(x) = \nabla_x \left(\frac{f(x)}{Z} \right)$

Still intractable!

Problems with gradient of probability

- Property of probabilities: $Z = \int p_{\text{data}}(x)dx = 1$

Therefore, $\nabla_x p_{\text{data}}(x) = \nabla_x \left(\frac{f(x)}{Z} \right)$

Still intractable!

- Numerical stability issues in low-density regions

Problems with gradient of probability

- Property of probabilities: $Z = \int p_{\text{data}}(x)dx = 1$

Therefore, $\nabla_x p_{\text{data}}(x) = \nabla_x \left(\frac{f(x)}{Z} \right)$

Still intractable!

- Numerical stability issues in low-density regions

Proposal. Let's consider $\nabla_x \log(p_{\text{data}}(x))$ instead!

Reviewing gradient of log of p

- Not intractable anymore!

$$\nabla_x \log(p_{\text{data}}(x)) = \nabla_x \log(f(x)) - \nabla_x \log(Z) = \nabla_x \log(f(x))$$

Reviewing gradient of log of p

- Not intractable anymore!

$$\nabla_x \log(p_{\text{data}}(x)) = \nabla_x \log(f(x)) - \nabla_x \log(Z) = \nabla_x \log(f(x))$$

- Points in the same direction as gradient of p

$$\nabla \log(p) = \frac{\nabla p}{p}$$

Reviewing gradient of log of p

- Not intractable anymore!

$$\nabla_x \log(p_{\text{data}}(x)) = \nabla_x \log(f(x)) - \nabla_x \log(Z) = \nabla_x \log(f(x))$$

- Points in the same direction as gradient of p

$$\nabla \log(p) = \frac{\nabla p}{p}$$

- More numerically stable

Reviewing gradient of log of p

- Not intractable anymore!

$$\nabla_x \log(p_{\text{data}}(x)) = \nabla_x \log(f(x)) - \nabla_x \log(Z) = \nabla_x \log(f(x))$$

- Points in the same direction as gradient of p

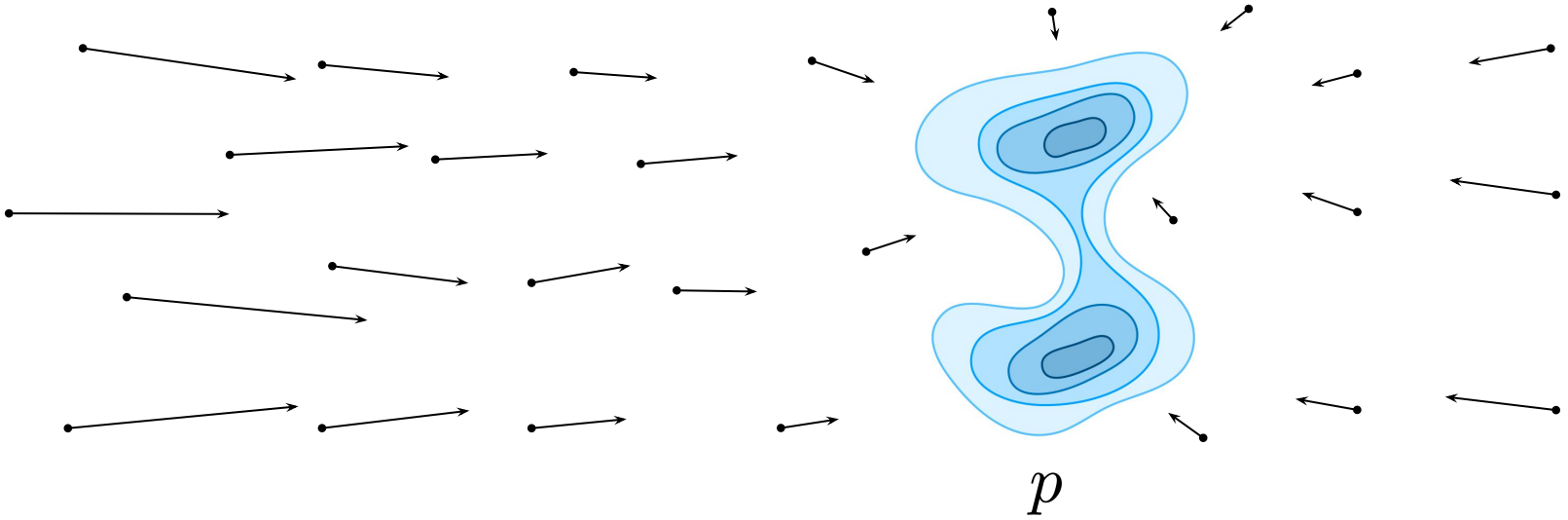
$$\nabla \log(p) = \frac{\nabla p}{p}$$

- More numerically stable

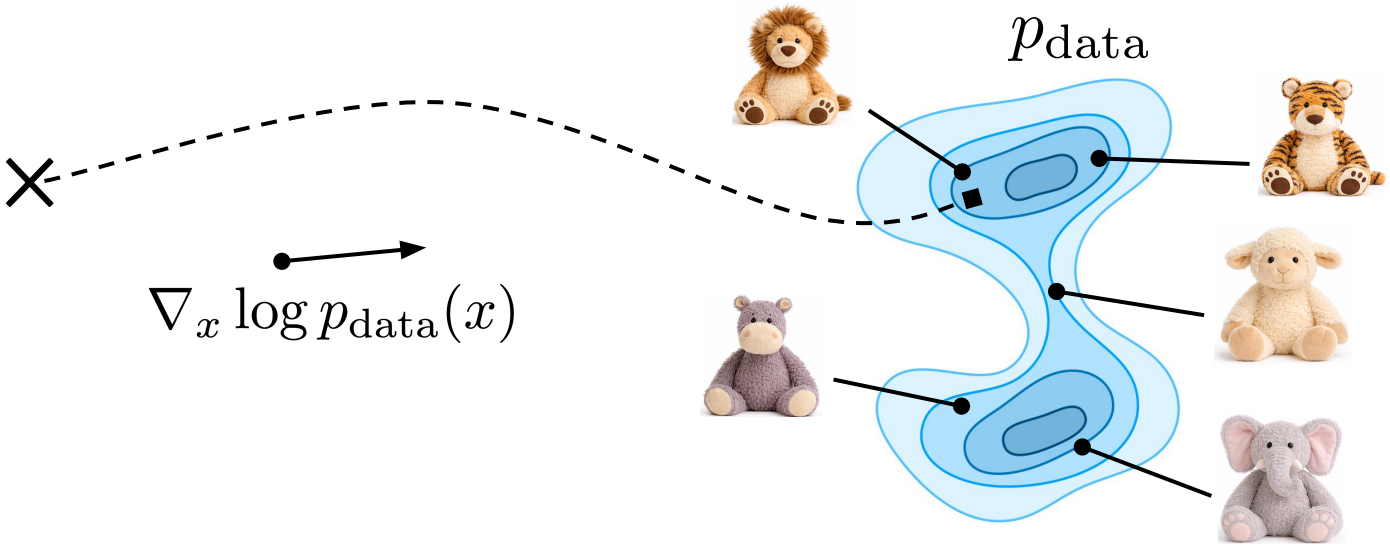
Score function $\boxed{s(x) = \nabla_x \log p(x)}$

Intuition behind score function

$$s(x) = \nabla_x \log p(x)$$



Using the score function in our problem



Langevin sampling $x_t = x_{t-1} + \frac{\alpha}{2}\nabla_x \log p_{\text{data}}(x_{t-1}) + \sqrt{\alpha}\,\epsilon_t$

Diffusion & Large Vision Models

Motivation

Score estimation

Differential formulation

Training

Inference

Probability flows

Score matching

SM = **S**core **M**atching

Goal. Estimate the score with $s_\theta(x)$

$$\mathcal{L}_{\text{SM}} = \mathbb{E}_x \left[\| s_\theta(x) - \nabla_x \log p_{\text{data}}(x) \|^2 \right]$$

Score matching

SM = **S**core **M**atching

Goal. Estimate the score with $s_\theta(x)$

$$\mathcal{L}_{\text{SM}} = \mathbb{E}_x \left[\| s_\theta(x) - \nabla_x \log p_{\text{data}}(x) \|^2 \right]$$

We don't have access to it

Attempts to estimate the score

- **I**mplicit **S**core **M**atching (ISM): integrate by parts

$$\mathcal{L}_{\text{ISM}} = \mathbb{E}_x \left[\frac{1}{2} ||s_\theta(x)||^2 + \nabla_x \cdot s_\theta(x) \right]$$

Estimation of Non-Normalized Statistical Models by Score Matching, Hyvärinen, 2005.

Attempts to estimate the score

- **I**mplicit **S**core **M**atching (ISM): integrate by parts

$$\mathcal{L}_{\text{ISM}} = \mathbb{E}_x \left[\frac{1}{2} ||s_\theta(x)||^2 + \nabla_x \cdot s_\theta(x) \right]$$

- **S**liced **S**core **M**atching (SSM): project on vectors

$$\mathcal{L}_{\text{SSM}} = \mathbb{E}_{x_0,t,\epsilon,v} \left[2\, v^\top \nabla_x s_\theta(x)\, v + \left| v^\top s_\theta(x) \right|^2 \right]$$

Sliced Score Matching: A Scalable Approach to Density and Score Estimation, Song et al., 2019.

Score of a Gaussian distribution (example in 1D)

For $x \sim \mathcal{N}(\mu, \sigma^2)$

- Probability distribution function is

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- Therefore, **score** is:

$$s(x) = -\frac{x-\mu}{\sigma^2}$$

New attempt: add noise to data

Idea. Add noise to data and use analytical expression of the score

$$\tilde{x} = x + \sigma\epsilon \qquad \text{with} \qquad \epsilon \sim \mathcal{N}(0, I)$$

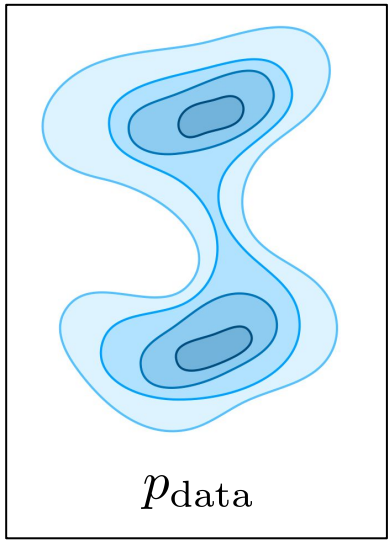
Therefore:

$$q_\sigma(\tilde{x}|x) = \mathcal{N}(x, \sigma^2 I) \longrightarrow \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x) = -\frac{\tilde{x} - x}{\sigma^2}$$

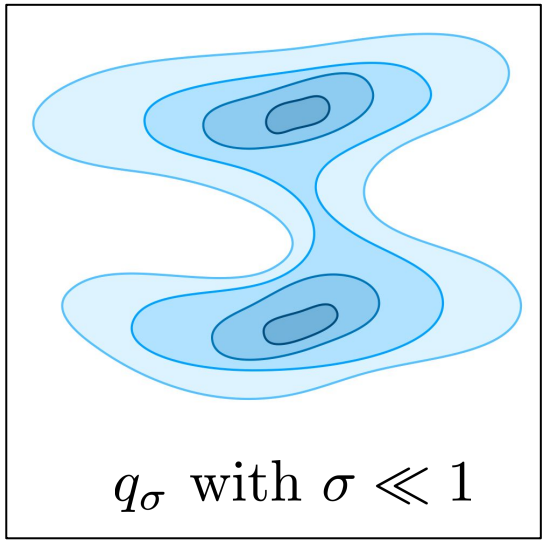
Derivation: Leverage expression of score for a Gaussian distribution

Add noise to distribution

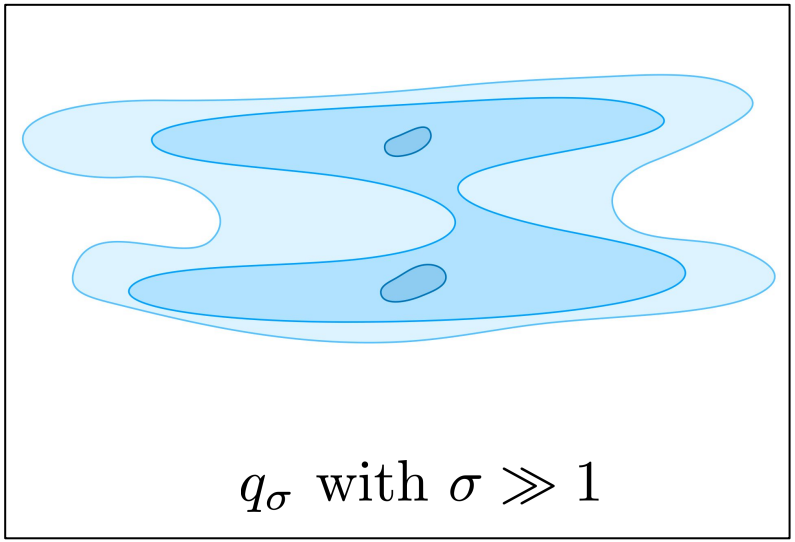
Let's define: $$q_\sigma(\tilde{x}) = \int q_\sigma(\tilde{x}|x)p_{\text{data}}(x)dx$$



p_{data}



q_σ with $\sigma \ll 1$



q_σ with $\sigma \gg 1$

Estimating the score of the noised distribution

Definition of score matching of q_σ

$$\mathcal{L}_{\mathrm{SM}}(q_\sigma) = \mathbb{E}_{\tilde{x}}\left[\left\|s_\theta(\tilde{x}) - \nabla_{\tilde{x}} \log q_\sigma(\tilde{x})\right\|^2\right]$$

We can show that this is also equal to:

$$\mathcal{L}_{\mathrm{SM}}(q_\sigma) = \mathbb{E}_{x,\tilde{x}}\left[\left\|s_\theta(\tilde{x}) - \underbrace{\nabla_{\tilde{x}} \log q_\sigma(\tilde{x} \mid x)}_{-\frac{\tilde{x}-x}{\sigma^2}}\right\|^2\right]$$

tractable!

Derivation: Expand squared norm and use definition of marginal

Full derivation in A Connection Between Score Matching and Denoising Autoencoders, Vincent, 2010.

Denoising score matching

DSM = **D**enoising **S**core **M**atching

We now have a **tractable** loss!

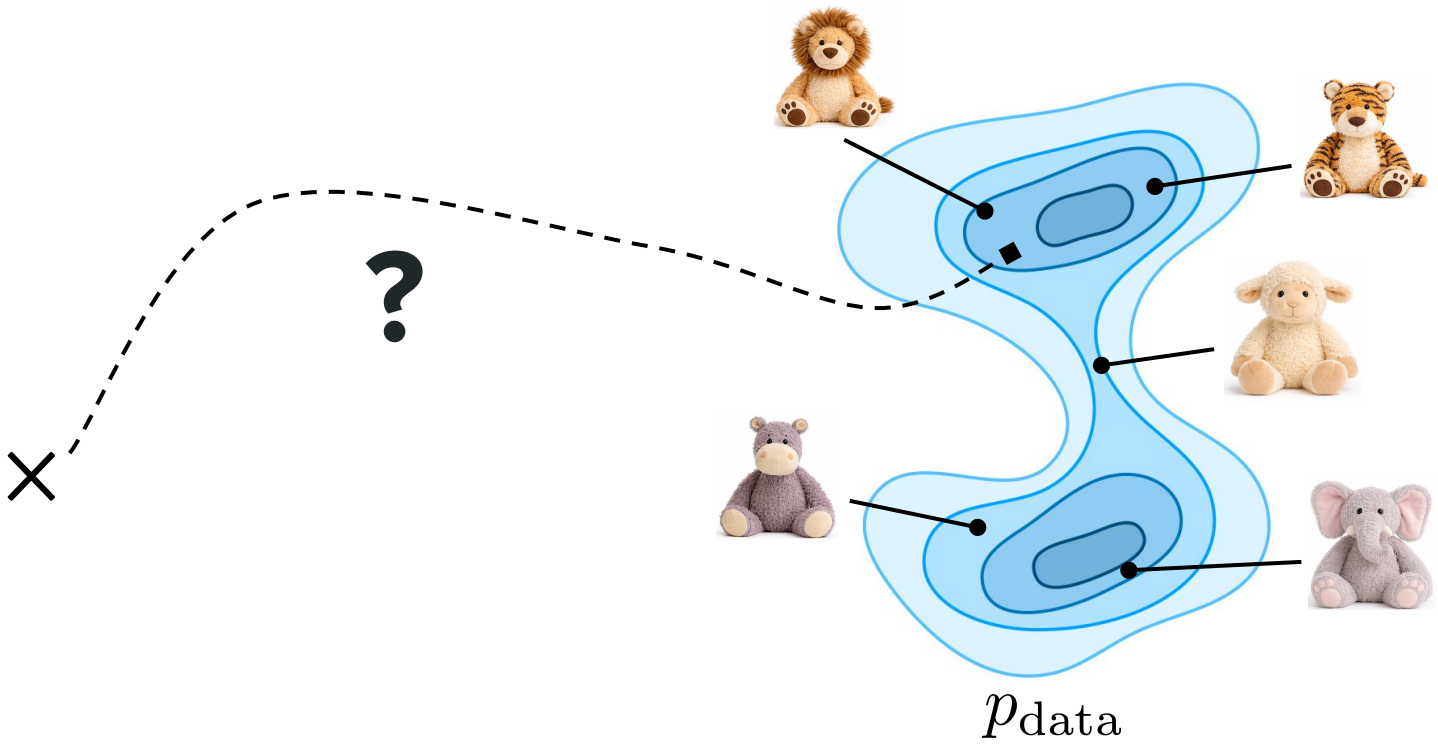
$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{x,\tilde{x}}\left[\left\| s_\theta(\tilde{x}) - \underbrace{\nabla_{\tilde{x}} \log q_\sigma(\tilde{x} \mid x)}_{-\frac{\tilde{x}-x}{\sigma^2}} \right\|^2\right]$$

However, $\mathcal{L}_{\text{DSM}} = \mathcal{L}_{\text{SM}}(q_\sigma)$

This is **not** exactly p_{data}

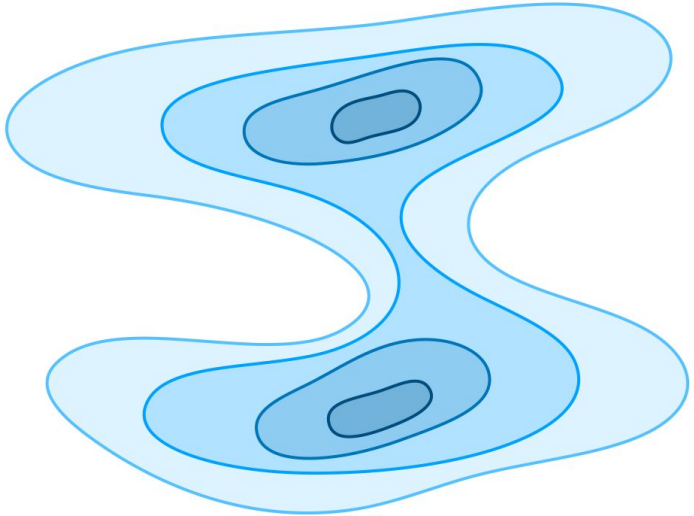
A Connection Between Score Matching and Denoising Autoencoders, Vincent, 2010.

Reminder about the objective of interest



Limitations with vanilla DSM

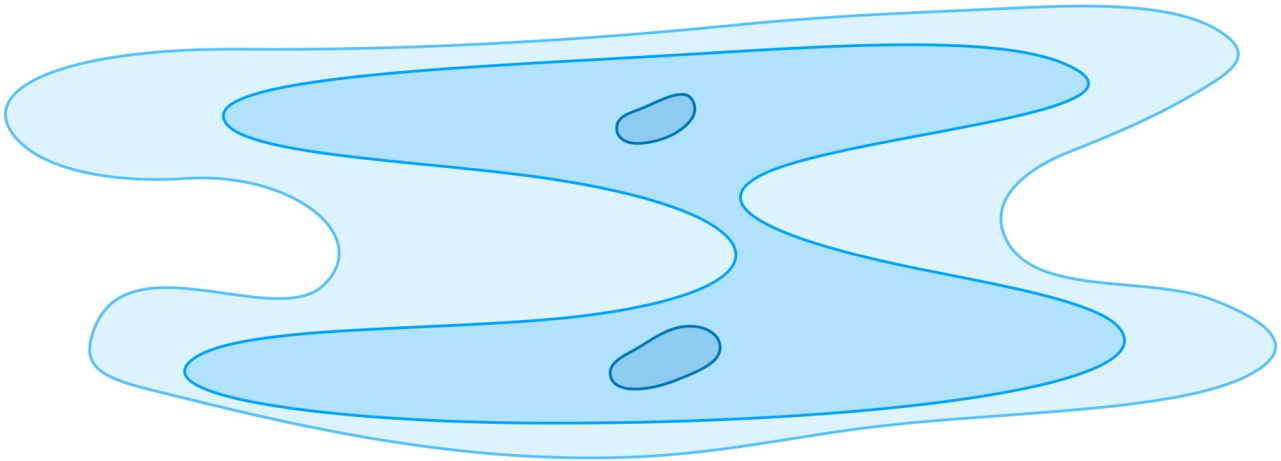
q_σ with $\sigma \ll 1$



q_σ **close** to p_{data} but **poor estimations** in low-density regions

Limitations with vanilla DSM

$$q_\sigma \text{ with } \sigma \gg 1$$



q_σ **far** from p_{data} but **good estimations** in low-density regions

Score matching with varying noise via NCSN

NCSN = **N**oise **C**onditional **S**core **N**etwork

Idea. Estimate score at different noise levels $\sigma_1 < \ldots < \sigma_T$

$$s_\theta(x) \longrightarrow s_\theta(x, \sigma_i)$$

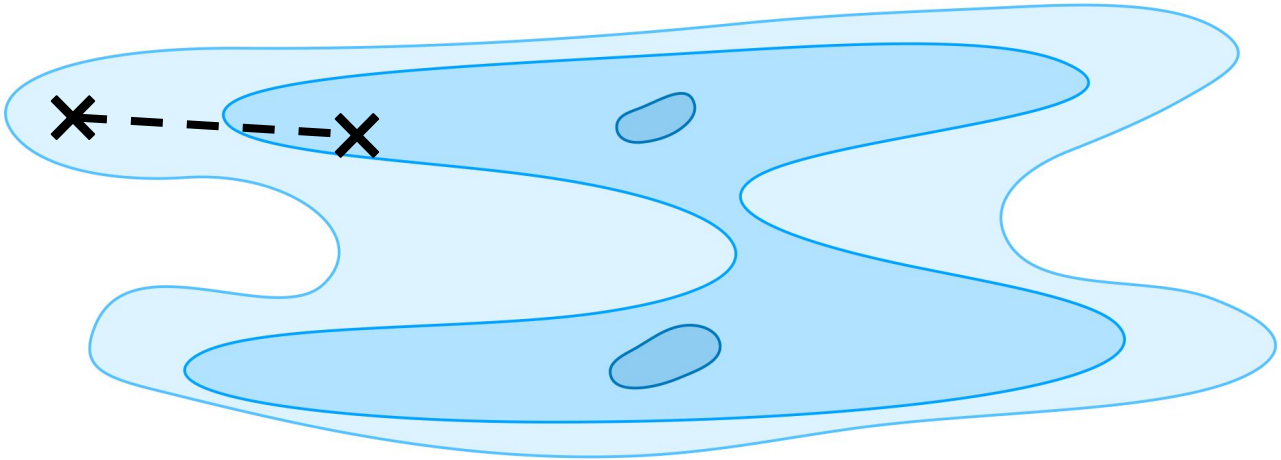
Loss.

$$\mathcal{L}_{\text{NCSN}} = \sum_{i=1}^{L} \lambda(i)\, \mathbb{E}_x \left[\| s_\theta(x, \sigma_i) - \nabla_x \log q_{\sigma_i}(x) \|_2^2 \right]$$

Generative Modeling by Estimating Gradients of the Data Distribution, Song et al., 2019.

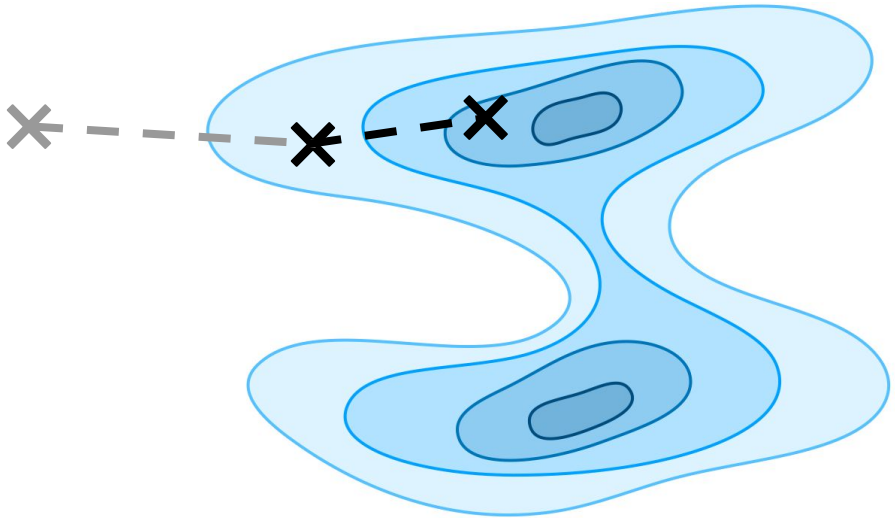
Intuition behind sampling strategy

$$q_{\sigma_T} \text{ with } \sigma_T \gg 1$$



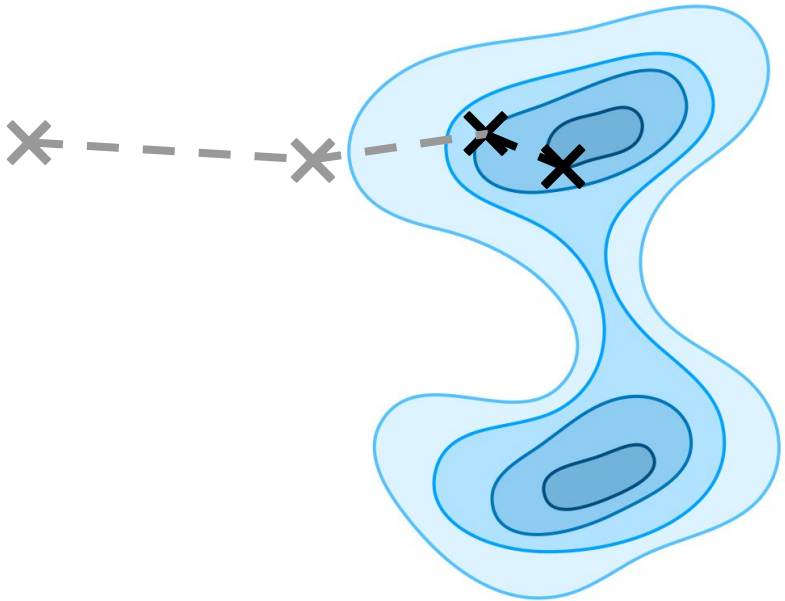
Intuition behind sampling strategy

$q_{\sigma_{T-1}}$ with $\sigma_T > \sigma_{T-1}$



Intuition behind sampling strategy

$$q_{\sigma_1} \quad \text{with} \quad \sigma_T > \sigma_{T-1} > \ldots > \sigma_1$$



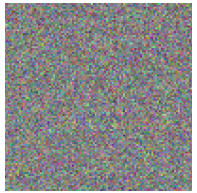
ALD = **A**nnealed **L**angevin **D**ynamics

Sampling with Annealed Langevin Dynamics

1. Sample:

noise

$$x_T \sim \mathcal{N}(0, \sigma_T^2 I)$$

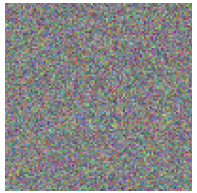


$$\sigma_T > \ldots > \sigma_1$$

2. For each σ_i

Sampling with Annealed Langevin Dynamics

1. Sample: **noise** $x_T \sim \mathcal{N}(0, \sigma_T^2 I)$

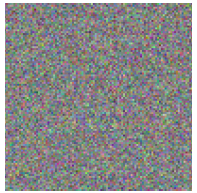


$\sigma_T > ... > \sigma_1$

2. For each σ_i , perform **iterative update** for K steps

Sampling with Annealed Langevin Dynamics

1. Sample: **noise** $x_T \sim \mathcal{N}(0, \sigma_T^2 I)$



$$\sigma_T > \ldots > \sigma_1$$

2. For each σ_i , perform **iterative update** for K steps

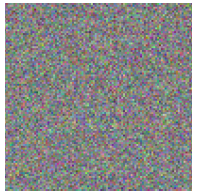
x

Sampling with Annealed Langevin Dynamics

1. Sample:

noise

$$x_T \sim \mathcal{N}(0, \sigma_T^2 I)$$



$$\sigma_T > \ldots > \sigma_1$$

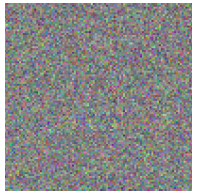
2. For each σ_i, perform **iterative update** for K steps

$$x + \quad s_\theta(x, \sigma_i)$$

Predicted score to move towards

Sampling with Annealed Langevin Dynamics

1. Sample: **noise** $x_T \sim \mathcal{N}(0, \sigma_T^2 I)$



$$\sigma_T > \ldots > \sigma_1$$

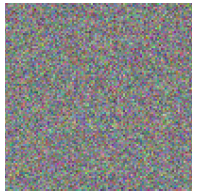
2. For each σ_i, perform **iterative update** for K steps

$$x + \frac{\alpha_i}{2} s_\theta(x, \sigma_i)$$

Predicted score to move towards

Sampling with Annealed Langevin Dynamics

1. Sample: **noise** $x_T \sim \mathcal{N}(0, \sigma_T^2 I)$



$$\sigma_T > \ldots > \sigma_1$$

2. For each σ_i, perform **iterative update** for K steps

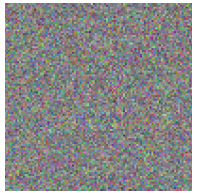
$$x + \frac{\alpha_i}{2} s_\theta(x, \sigma_i) + \epsilon$$

Predicted score to move towards

Sampled from a Normal distribution

Sampling with Annealed Langevin Dynamics

1. Sample: **noise** $x_T \sim \mathcal{N}(0, \sigma_T^2 I)$



$$\sigma_T > \ldots > \sigma_1$$

2. For each σ_i, perform **iterative update** for K steps

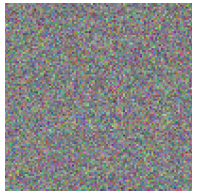
$$x + \frac{\alpha_i}{2} s_\theta(x, \sigma_i) + \sqrt{\alpha_i}\,\epsilon$$

Predicted score to move towards

Sampled from a Normal distribution

Sampling with Annealed Langevin Dynamics

1. Sample: **noise** $x_T \sim \mathcal{N}(0, \sigma_T^2 I)$



$$\sigma_T > \ldots > \sigma_1$$

2. For each σ_i, perform **iterative update** for K steps

$$x \leftarrow x + \frac{\alpha_i}{2} s_\theta(x, \sigma_i) + \sqrt{\alpha_i}\,\epsilon$$

Predicted score to move towards

Sampled from a Normal distribution

Sampling with Annealed Langevin Dynamics

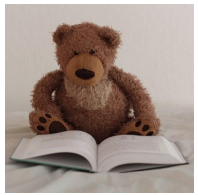
1. Sample: **noise** $x_T \sim \mathcal{N}(0, \sigma_T^2 I)$

$\sigma_T > ... > \sigma_1$

2. For each σ_i, perform **iterative update** for K steps

$$x \leftarrow x + \frac{\alpha_i}{2} s_\theta(x, \sigma_i) + \sqrt{\alpha_i}\,\epsilon$$

3. Obtain **final** image x_0



Parallel between noise (DDPM) and score (NCSN)

DDPM

$$x_t = \sqrt{\overline{\alpha}_t} x_0 + \sqrt{1 - \overline{\alpha}_t}\epsilon$$

$$\downarrow$$

$$q(x_t|x_0) = \mathcal{N}(\sqrt{\overline{\alpha}_t} x_0, (1 - \overline{\alpha}_t) I)$$

$$\downarrow$$

$$\nabla_{x_t} \log(q(x_t|x_0)) = -\frac{\epsilon}{\sqrt{1 - \overline{\alpha}_t}}$$

Parallel between noise (DDPM) and score (NCSN)

DDPM

$$x_t = \sqrt{\overline{\alpha}_t} x_0 + \sqrt{1 - \overline{\alpha}_t}\epsilon$$

$$\downarrow$$

$$q(x_t|x_0) = \mathcal{N}(\sqrt{\overline{\alpha}_t} x_0, (1 - \overline{\alpha}_t) I)$$

$$\downarrow$$

Score from noisy to clean

Noise to remove from noisy sample

$$\nabla_{x_t} \log(q(x_t|x_0)) = -\frac{\epsilon}{\sqrt{1 - \overline{\alpha}_t}}$$

Diffusion & Large Vision Models

Motivation

Score estimation

Differential formulation

Training

Inference

Probability flows

Intuition

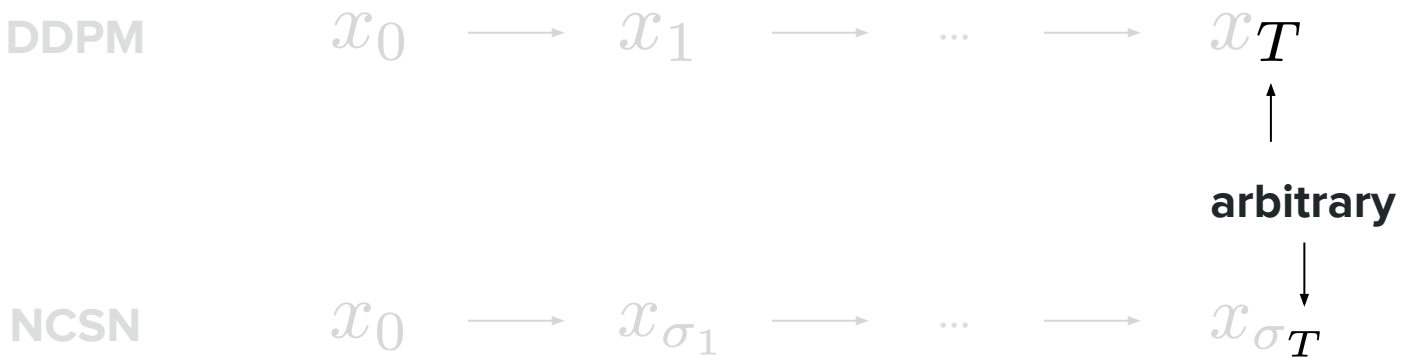
DDPM $x_0 \longrightarrow x_1 \longrightarrow \dots \longrightarrow x_T$

Intuition

DDPM $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

NCSN $x_0 \longrightarrow x_{\sigma_1} \longrightarrow \cdots \longrightarrow x_{\sigma_T}$

Intuition



Intuition

💡 What if we considered **continuous** evolutions instead?

Let's first define Wiener processes

Idea. **Continuous** equivalent of **adding increments** of **Gaussian noise**

Wiener process. Stochastic process with the following special properties:

- $W_0 = 0$
- $W_t - W_s \sim \mathcal{N}(0, (t-s)I)$
- Independent increments

DDPM continuous formulation

Discrete $x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \epsilon$

$\Delta t \longrightarrow 0$

Continuous $dx = -\frac{1}{2}\beta(t)\, x\, dt + \sqrt{\beta(t)}\, dW$

Derivation: Use Taylor expansion + reparametrize β for small incremental noise

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

NCSN continuous formulation

Discrete $x_t = x_{t-1} + \sqrt{\sigma_t^2 - \sigma_{t-1}^2}\epsilon$

$\Delta t \longrightarrow 0$

Continuous $dx = \sqrt{\frac{d[\sigma^2(t)]}{dt}}\, dW$

Derivation: Use definition of a Wiener process

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

Differential formulation

SDE = **S**tochastic **D**ifferential **E**quation

$$dx = \underbrace{f(x,t)}_{\textbf{Drift}}dt + \underbrace{g(t)}_{\textbf{Diffusion}}dW$$

deterministic

stochastic

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

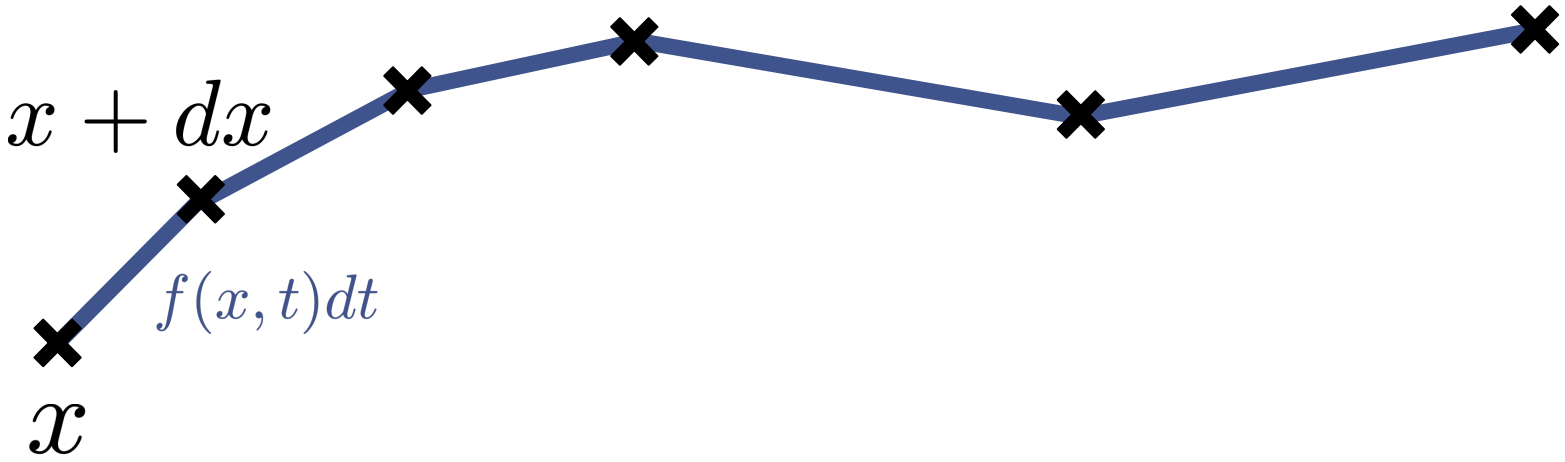
Intuition behind differential formulation

$$dx = f(x,t)dt + g(t)dW$$

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

Intuition behind differential formulation

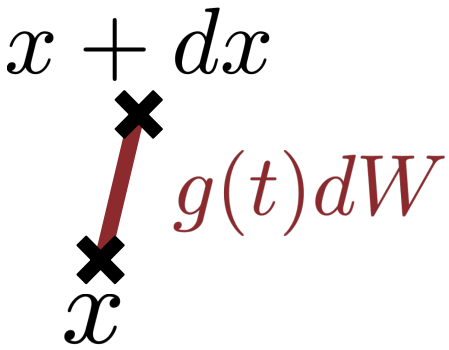
$$dx = f(x,t)dt$$



Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

Intuition behind differential formulation

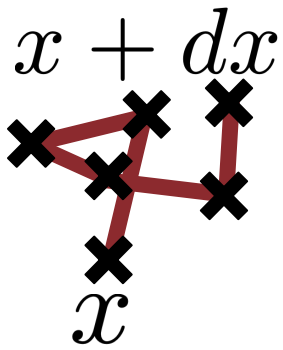
$$dx = g(t)\underbrace{dW}_{\sqrt{dt}\,\epsilon}$$



Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

Intuition behind differential formulation

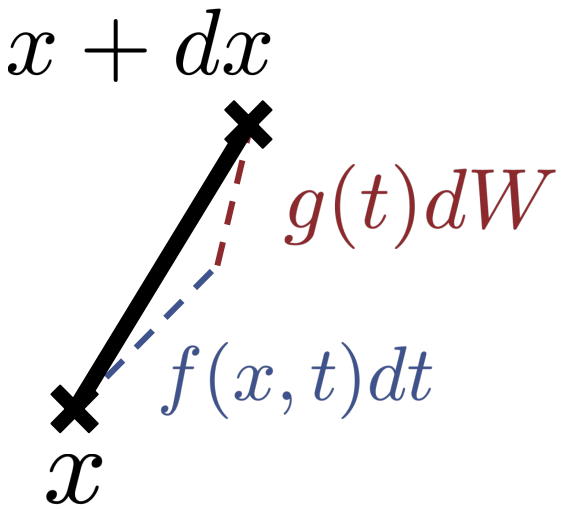
$$dx = g(t)\underbrace{dW}_{\sqrt{dt}\,\epsilon}$$



Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

Intuition behind differential formulation

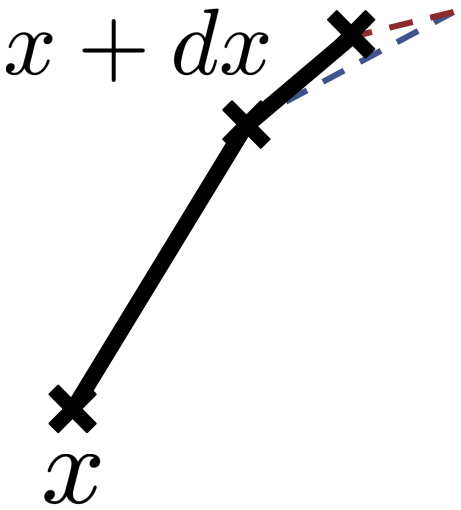
$$dx = f(x,t)dt + g(t)dW$$



Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

Intuition behind differential formulation

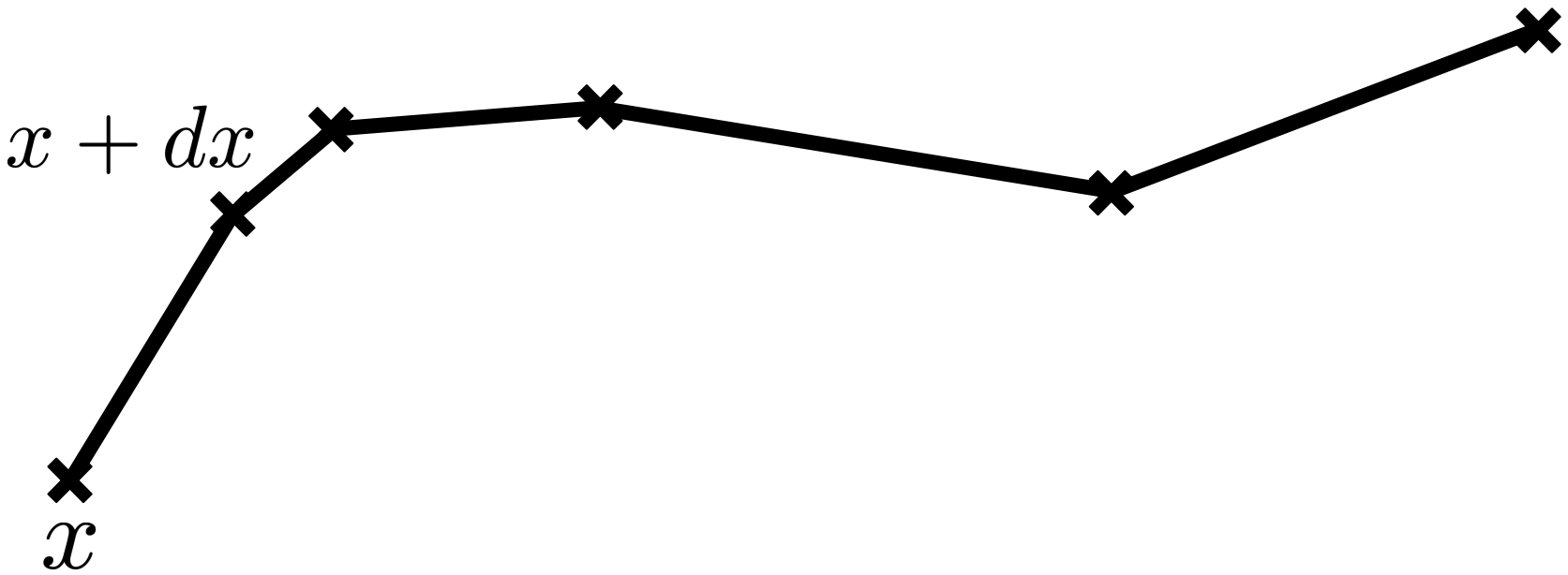
$$dx = f(x,t)dt + g(t)dW$$



Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

Intuition behind differential formulation

$$dx = f(x,t)dt + g(t)dW$$



Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

SDE variants

Variance Preserving

Example: DDPM

$$f(x,t) = -\frac{1}{2}\beta(t)x$$

$$g(t) = \sqrt{\beta(t)}$$

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

SDE variants

Variance **P**reserving

Example: DDPM

$$f(x,t) = -\frac{1}{2}\beta(t)x$$

$$g(t) = \sqrt{\beta(t)}$$

Variance **E**xploding

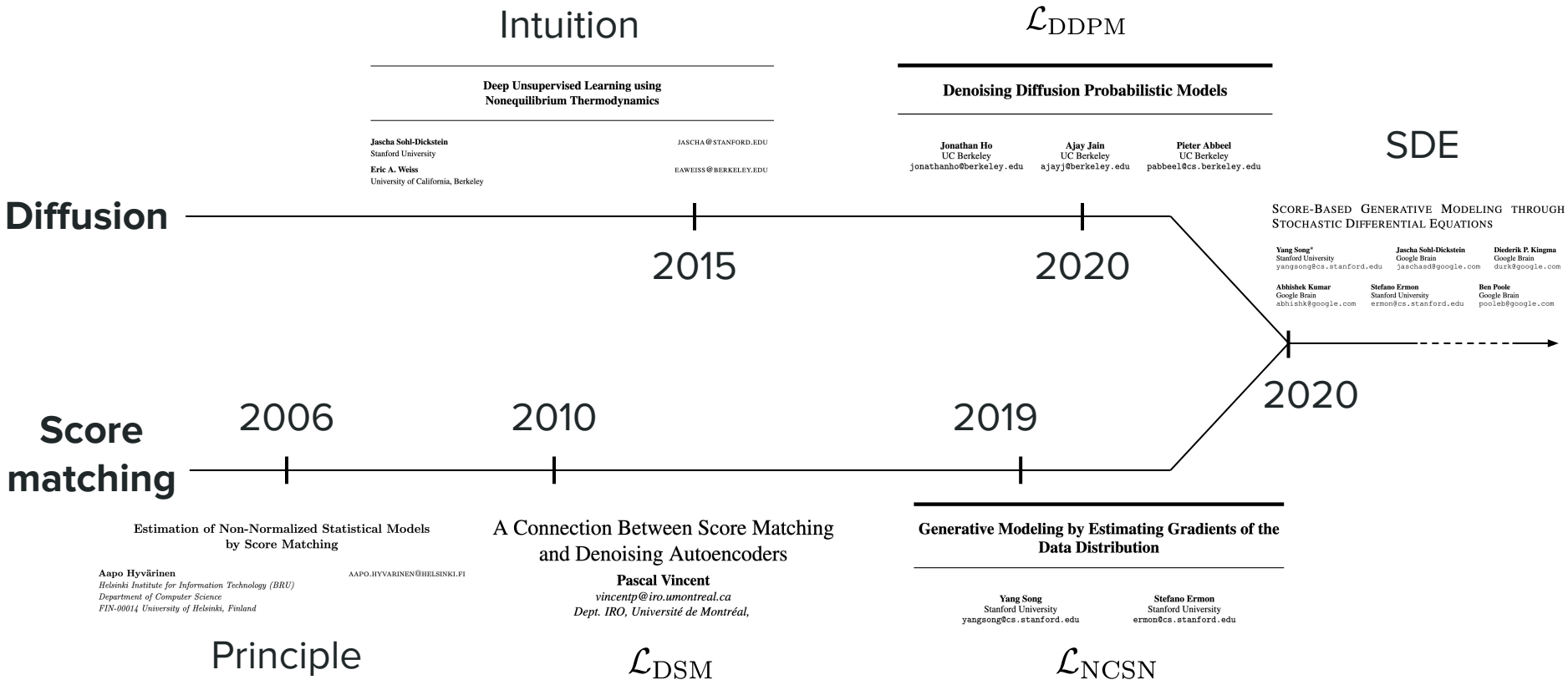
Example: NCSN

$$f(x,t) = 0$$

$$g(t) = \sqrt{\frac{d[\sigma^2(t)]}{dt}}$$

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

Timeline and unification of methods



Diffusion & Large Vision Models

Motivation

Score estimation

Differential formulation

Training

Inference

Probability flows

Training derivation

Objective. Estimate the score $s_\theta(x_t, t)$

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{t,x_0,x_t}\left[\lambda_t \| s_\theta(x_t, t) - \nabla_{x_t} \log p(x_t|x_0)\|^2\right]$$

Interesting discussion in Elucidating the Design Space of Diffusion-Based Generative Models, Karras et al., 2022.

Training derivation

Objective. Estimate the score $s_\theta(x_t, t)$

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{t,x_0,\boldsymbol{x_t}}\left[\lambda_t \| s_\theta(x_t, t) - \nabla_{\boldsymbol{x_t}} \log p(\boldsymbol{x_t}|x_0)\|^2\right]$$

Variance Preserving
(DDPM-style)

$$x_t \mid x_0 \sim \mathcal{N}\left(\sqrt{\overline{\alpha}_t} x_0, (1 - \overline{\alpha}_t) I\right)$$

Interesting discussion in Elucidating the Design Space of Diffusion-Based Generative Models, Karras et al., 2022.

Training derivation

Objective. Estimate the score $s_\theta(x_t, t)$

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{t, x_0, \boldsymbol{x_t}}\left[\lambda_t \| s_\theta(x_t, t) - \nabla_{x_t} \log p(\boldsymbol{x_t} | x_0) \|^2\right]$$

Variance Preserving
(DDPM-style)

$$x_t \mid x_0 \sim \mathcal{N}\left(\sqrt{\overline{\alpha}_t} x_0, (1 - \overline{\alpha}_t) I\right)$$

Variance Exploding
(NCSN-style)

$$x_t \mid x_0 \sim \mathcal{N}\left(x_0, \sigma_t^2 I\right)$$

Interesting discussion in Elucidating the Design Space of Diffusion-Based Generative Models, Karras et al., 2022.

Training derivation

Objective. Estimate the score $s_\theta(x_t, t)$

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{t, x_0, \boldsymbol{x_t}}\left[\lambda_t \| s_\theta(x_t, t) - \nabla_{\boldsymbol{x_t}} \log p(\boldsymbol{x_t}|x_0)\|^2\right]$$

$$x_t = \alpha_t x_0 + \sigma_t \epsilon$$

Variance Preserving
(DDPM-style)

$$x_t \mid x_0 \sim \mathcal{N}\left(\sqrt{\overline{\alpha}_t} x_0, (1 - \overline{\alpha}_t) I\right)$$

Variance Exploding
(NCSN-style)

$$x_t \mid x_0 \sim \mathcal{N}\left(x_0, \sigma_t^2 I\right)$$

Interesting discussion in Elucidating the Design Space of Diffusion-Based Generative Models, Karras et al., 2022.

Training derivation

Objective. Estimate the score $s_\theta(x_t, t)$

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{t,x_0,x_t}\left[\lambda_t \| s_\theta(x_t, t) - \underbrace{\nabla_{x_t} \log p(x_t|x_0)}_{-\frac{\epsilon}{\sigma_t}} \|^2\right]$$

Variance Preserving
(DDPM-style)

$$x_t \mid x_0 \sim \mathcal{N}\left(\sqrt{\overline{\alpha}_t} x_0, (1 - \overline{\alpha}_t) I\right)$$

Variance Exploding
(NCSN-style)

$$x_t \mid x_0 \sim \mathcal{N}\left(x_0, \sigma_t^2 I\right)$$

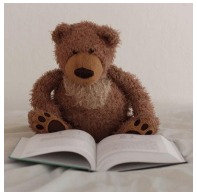
Interesting discussion in Elucidating the Design Space of Diffusion-Based Generative Models, Karras et al., 2022.

Training recipe

1. Sample:

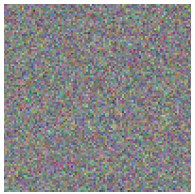
clean image

$x_0 \sim p_{\text{data}}$



noise

$\epsilon \sim \mathcal{N}(0, I)$



time step

$t \sim \mathcal{U}(0, T)$

noised image

$$x_t = \alpha_t x_0 + \sigma_t \epsilon$$

Training recipe

1. Sample:

clean image $x_0 \sim p_{\text{data}}$

noise $\epsilon \sim \mathcal{N}(0, I)$

time step $t \sim \mathcal{U}(0, T)$

noised image

$$x_t = \alpha_t x_0 + \sigma_t \epsilon$$

2. Use x_t and t to **predict** $-\frac{\epsilon}{\sigma_t}$ via $s_\theta(x_t, t)$

Training recipe

1. Sample:

clean image $x_0 \sim p_{\text{data}}$

noise $\epsilon \sim \mathcal{N}(0, I)$

time step $t \sim \mathcal{U}(0, T)$

noised image $x_t = \alpha_t x_0 + \sigma_t \epsilon$

2. Use x_t and t to **predict** $-\frac{\epsilon}{\sigma_t}$ via $s_\theta(x_t, t)$

3. Compute **loss** $\mathcal{L} = \lambda_t \left\| s_\theta(x_t, t) + \frac{\epsilon}{\sigma_t} \right\|^2$ and **backpropagate** through s_θ

Diffusion & Large Vision Models

Motivation

Score estimation

Differential formulation

Training

Inference

Probability flows

Introducing the reverse formula

Forward. Perturb data into noise.

$$dx = f(x,t)dt + g(t)dW$$

Introducing the reverse formula

Forward. Perturb data into noise.

$$dx = f(x,t)dt + g(t)dW$$

Reverse. Transform noise back into data.

$$dx = \Big[f(x,t) - g(t)^2 \nabla_x \log(p_t(x))\Big]dt + g(t)d\overline{W}$$

Derivation: Use the Fokker-Planck equation (beyond the scope of this class)

Reverse-time diffusion equation models, Anderson, 1982.

Intuition behind reverse time formula

Reverse. Transform noise back into data.

Reverse-time diffusion equation models, Anderson, 1982.

Intuition behind reverse time formula

Reverse. Transform noise back into data.

$$dx =$$

Reverse-time diffusion equation models, Anderson, 1982.

Intuition behind reverse time formula

Reverse. Transform noise back into data.

Forward drift coefficient

$$dx = f(x,t)\,dt$$

Reverse-time diffusion equation models, Anderson, 1982.

Intuition behind reverse time formula

Reverse. Transform noise back into data.

Forward drift coefficient

$$dx = f(x,t)\,dt + g(t)d\overline{W}$$

Forward diffusion coefficient

Wiener process but different than forward process one

Reverse-time diffusion equation models, Anderson, 1982.

Intuition behind reverse time formula

Reverse. Transform noise back into data.

Forward drift coefficient

Forward diffusion coefficient

$$dx = \left[f(x,t) - \underbrace{g(t)^2 \nabla_x \log(p_t(x))}_{\text{Correction to drift}} \right] dt + g(t) d\overline{W}$$

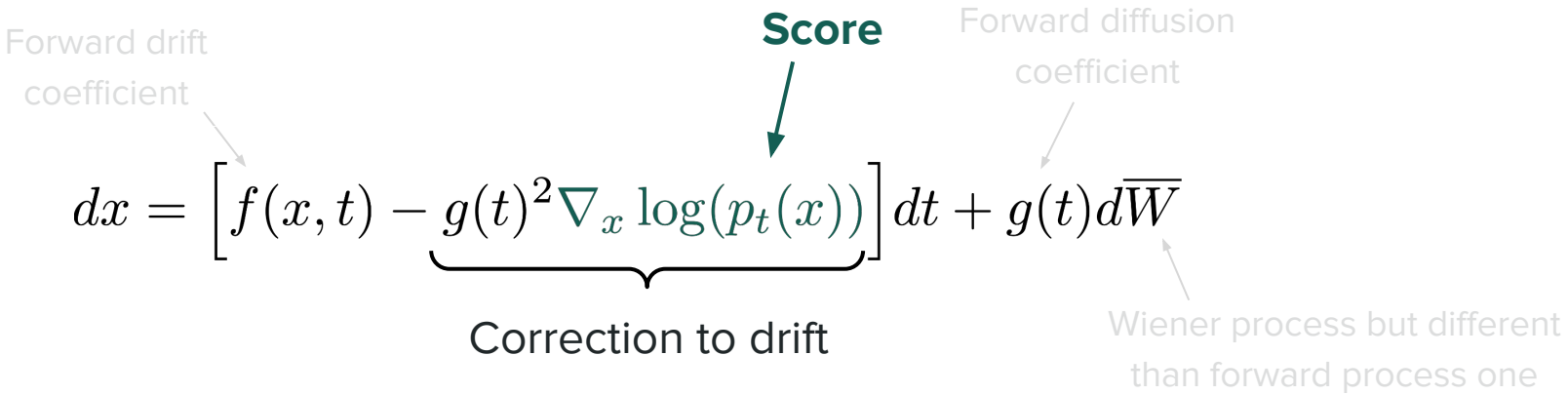
Wiener process but different than forward process one

Need to go even more towards regions of high density with the **score** to compensate for diffusion

Reverse-time diffusion equation models, Anderson, 1982.

Intuition behind reverse time formula

Reverse. Transform noise back into data.



Need to go even more towards regions of high density with the **score** to compensate for diffusion

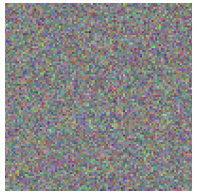
Reverse-time diffusion equation models, Anderson, 1982.

Inference recipe

1. Sample:

noise

$$x_T \sim \mathcal{N}(0, \sigma_T^2 I)$$



$$\sigma_T > \ldots > \sigma_1$$

2. Use Euler-Maruyama to go through reverse SDE:

$$x_{t_{i-1}} = x_{t_i} + \left[f(x_{t_i}, t_i) - g(t_i)^2 s_\theta(x_{t_i}, t_i) \right] \Delta t + g(t_i)\sqrt{\Delta t}\xi_i$$

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

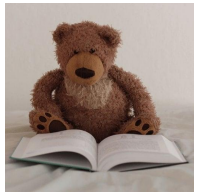
Inference recipe

1. Sample: noise $x_T \sim \mathcal{N}(0, \sigma_T^2 I)$

$$\sigma_T > \ldots > \sigma_1$$

2. Use Euler-Maruyama to go through reverse SDE:

$$x_{t_{i-1}} = x_{t_i} + \left[f(x_{t_i}, t_i) - g(t_i)^2 s_\theta(x_{t_i}, t_i) \right] \Delta t + g(t_i)\sqrt{\Delta t}\xi_i$$

3. Obtain **final** image x_0 

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

Diffusion & Large Vision Models

Motivation

Score estimation

Differential formulation

Training

Inference

Probability flows

Limitations of an SDE

$$dx = \left[f(x,t) - g(t)^2 \nabla_x \log(p_t(x)) \right] dt + g(t) d\overline{W}$$

Stochastic term means:

- **Slower solver**. Need 1000-2000 steps because we don't know in advance which region is "easy" vs. "hard".

Limitations of an SDE

$$dx = \left[f(x,t) - g(t)^2 \nabla_x \log(p_t(x)) \right] dt + g(t) d\overline{W}$$

Stochastic term means:

- **Slower solver**. Need 1000-2000 steps because we don't know in advance which region is "easy" vs. "hard".

- **More sources of error**. Both discretization error from finite step sizes and injected stochastic noise

Hypothetically: writing the SDE as an ODE

$$dx = [\text{something}]\,dt + \varnothing$$ ← No stochasticity!

No stochastic term would mean:

- **Faster solver**. Leverage mathematical properties to go fast vs. slow.

Hypothetically: writing the SDE as an ODE

$$dx = \boxed{\text{something}}\, dt + \varnothing$$ ← No stochasticity!

No stochastic term would mean:

- **Faster solver**. Leverage mathematical properties to go fast vs. slow.

- **Focused source of error**. Only finite step sizes contribute.

ODE derivation

Forward SDE. $dx = f(x,t)dt + g(t)dW$

Justification: Theorem (out of scope)

Fokker-Planck equation. $\frac{\partial p_t(x)}{\partial t} = -\nabla \cdot (f(x,t)p_t(x)) + \frac{1}{2}g(t)^2 \Delta p_t(x)$

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

ODE derivation

Forward SDE. $dx = f(x,t)dt + g(t)dW$

Fokker-Planck equation. $\frac{\partial p_t(x)}{\partial t} = -\nabla \cdot (f(x,t)p_t(x)) + \frac{1}{2}g(t)^2 \Delta p_t(x)$

Derivation: Algebraic operations

Continuity equation. $\frac{\partial p}{\partial t}(x) = -\nabla \cdot ([f(x,t) - \frac{1}{2}g(t)^2 \nabla_x \log p_t(x)]p_t(x))$

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

ODE derivation

Forward SDE. $dx = f(x,t)dt + g(t)dW$

Fokker-Planck equation. $\frac{\partial p_t(x)}{\partial t} = -\nabla \cdot (f(x,t)p_t(x)) + \frac{1}{2}g(t)^2 \Delta p_t(x)$

Continuity equation. $\frac{\partial p}{\partial t}(x) = -\nabla \cdot \left(\left[f(x,t) - \frac{1}{2}g(t)^2 \nabla_x \log p_t(x) \right] p_t(x) \right)$

$v(x,t)$

Derivation: Recognize velocity of probability flow

PF-ODE. $dx = \left[f(x,t) - \frac{1}{2}g(t)^2 \nabla \log(p_t(x)) \right] dt$

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

ODE derivation

Forward SDE. $dx = f(x,t)dt + g(t)dW$

Fokker-Planck equation. $\frac{\partial p_t(x)}{\partial t} = -\nabla \cdot (f(x,t)p_t(x)) + \frac{1}{2}g(t)^2 \Delta p_t(x)$

Continuity equation. $\frac{\partial p}{\partial t}(x) = -\nabla \cdot ([f(x,t) - \frac{1}{2}g(t)^2 \nabla_x \log p_t(x)]p_t(x))$

modeling assumptions

PF-ODE. $dx = \left[f(x,t) - \frac{1}{2}g(t)^2 \nabla \log(p_t(x))\right]dt$

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

ODE derivation

Forward SDE. $dx = f(x,t)dt + g(t)dW$

↓

Fokker-Planck equation. $\frac{\partial p_t(x)}{\partial t} = -\nabla \cdot (f(x,t)p_t(x)) + \frac{1}{2}g(t)^2 \Delta p_t(x)$

↓

Continuity equation. $\frac{\partial p}{\partial t}(x) = -\nabla \cdot ([f(x,t) - \frac{1}{2}g(t)^2 \nabla_x \log p_t(x)]p_t(x))$

↓

score approximated by our model

PF-ODE. $dx = \left[f(x,t) - \frac{1}{2}g(t)^2 \nabla \log(p_t(x))\right] dt$

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

PF-ODE meaning

PF-ODE = **P**robability **F**low-**O**rdinary **D**ifferential **E**quation

$$dx = \underbrace{\left[f(x,t) - \frac{1}{2}g(t)^2 \nabla \log(p_t(x))\right]}_{v(x,t)} dt \; + \; \varnothing$$

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

PF-ODE meaning

PF-ODE = **P**robability **F**low-**O**rdinary **D**ifferential **E**quation

$$dx = \left[f(x,t) - \frac{1}{2} g(t)^2 \nabla \log(p_t(x)) \right] dt$$

satisfies

$$\frac{\partial p}{\partial t}(x) = -\nabla \cdot (v(x,t) p_t(x))$$

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

PF-ODE meaning

PF-ODE = **P**robability **F**low-**O**rdinary **D**ifferential **E**quation

$$dx = \left[f(x,t) - \frac{1}{2} g(t)^2 \nabla \log(p_t(x)) \right] dt$$

satisfies

$$\frac{\partial p}{\partial t}(x) = -\nabla \cdot (v(x,t) p_t(x))$$

⚠️ **Same marginal densities ≠ Same trajectories** ⚠️

$p_t(x)$ $x(t)$

Score-Based Generative Modeling through Stochastic Differential Equations, Song et al., 2020.

Comparison between reverse SDE and PF-ODE

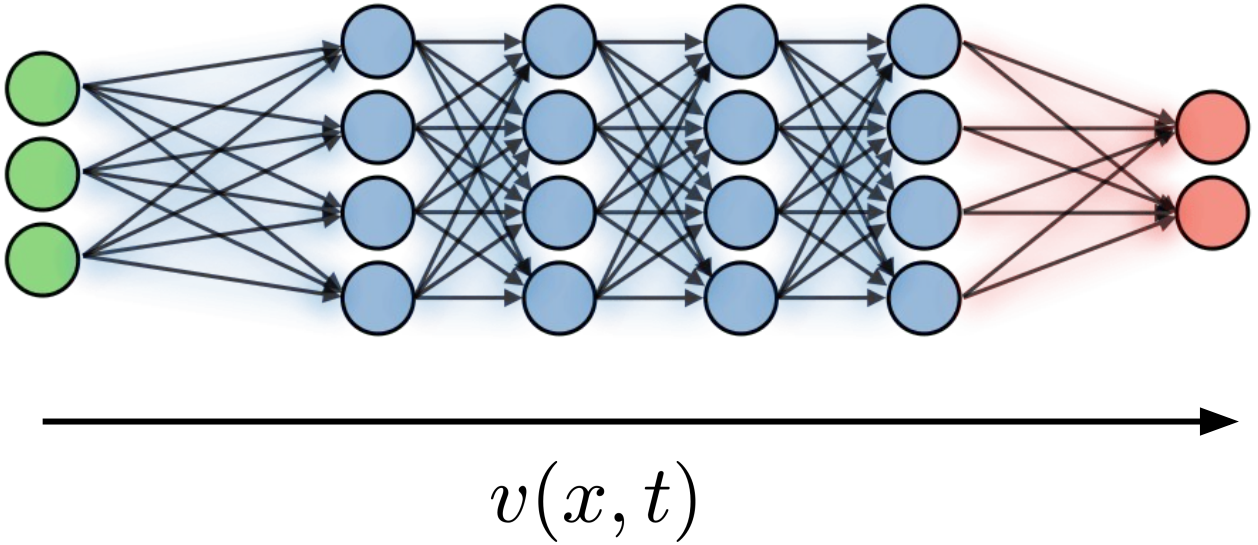
	Reverse SDE	**PF-ODE**
Equation	$dx = \left[f(x,t) - g(t)^2 \nabla_x \log(p_t(x))\right] dt + g(t) d\overline{W}$	$dx = \left[f(x,t) - \frac{1}{2} g(t)^2 \nabla \log(p_t(x))\right] dt$
Nature of process	Stochastic	Deterministic
Sampling diversity	Higher	Lower
Sampling quality	Higher	Lower
Sampling speed	Slower	Faster

Similarity with DDIM

	DDIM	PF-ODE
Deterministic counterpart of	DDPM	Reverse SDE
Can reused trained model?	Yes	
Sampling diversity	Lower	
Sampling quality	Lower	
Sampling speed	Faster	

Complexity metric

NFE = **N**umber of **F**unction **E**valuations

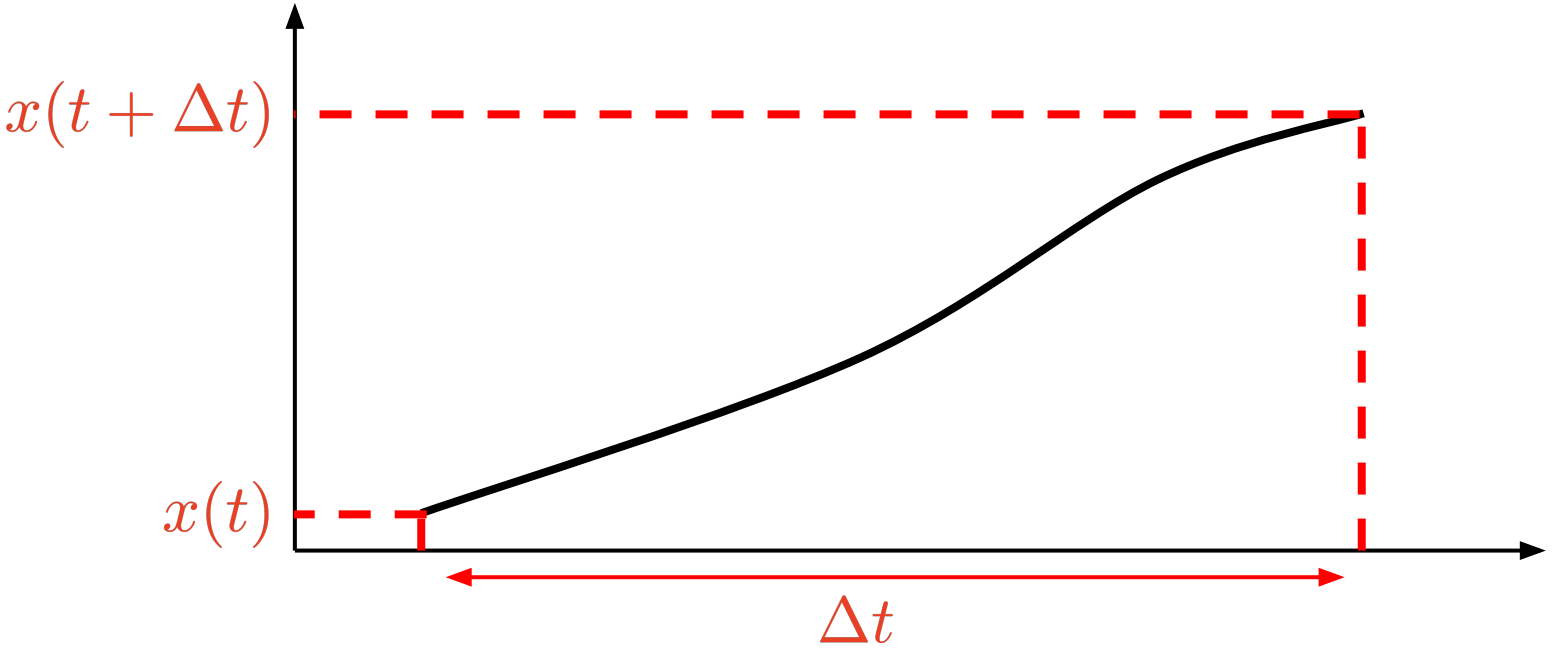


$$v(x,t)$$

Image from VIP cheatsheets for CS 230 - Deep Learnings Tips and Tricks, Amidi, 2019.

Solving for x

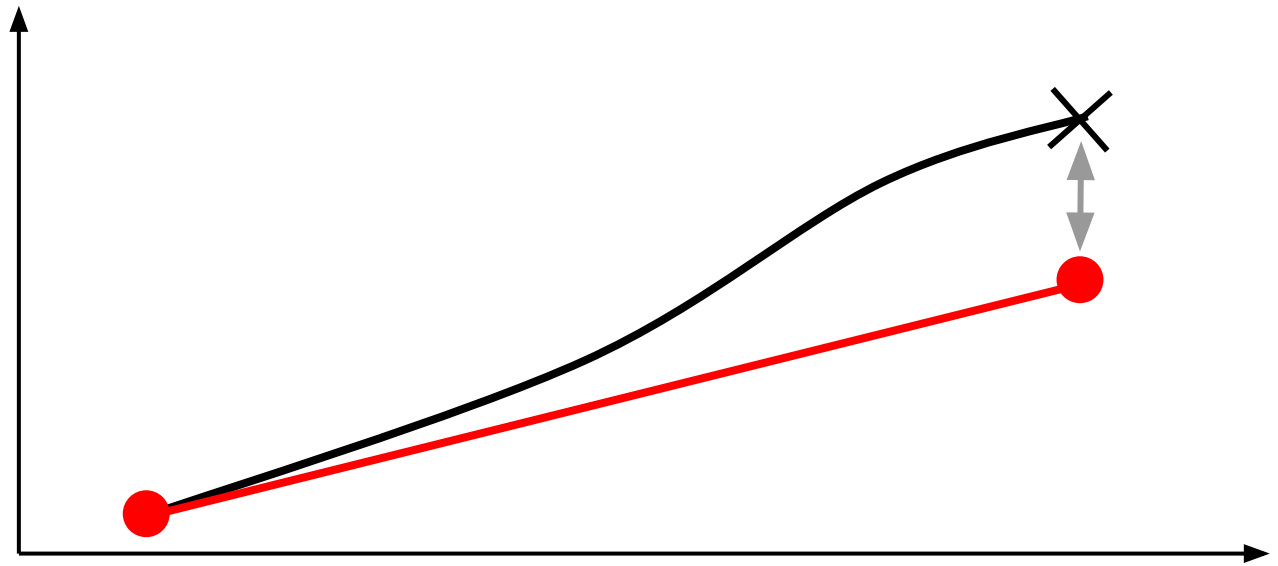
Problem statement. $dx = v(x,t)dt$ with $x(t_0) = x_0$



Simplest method

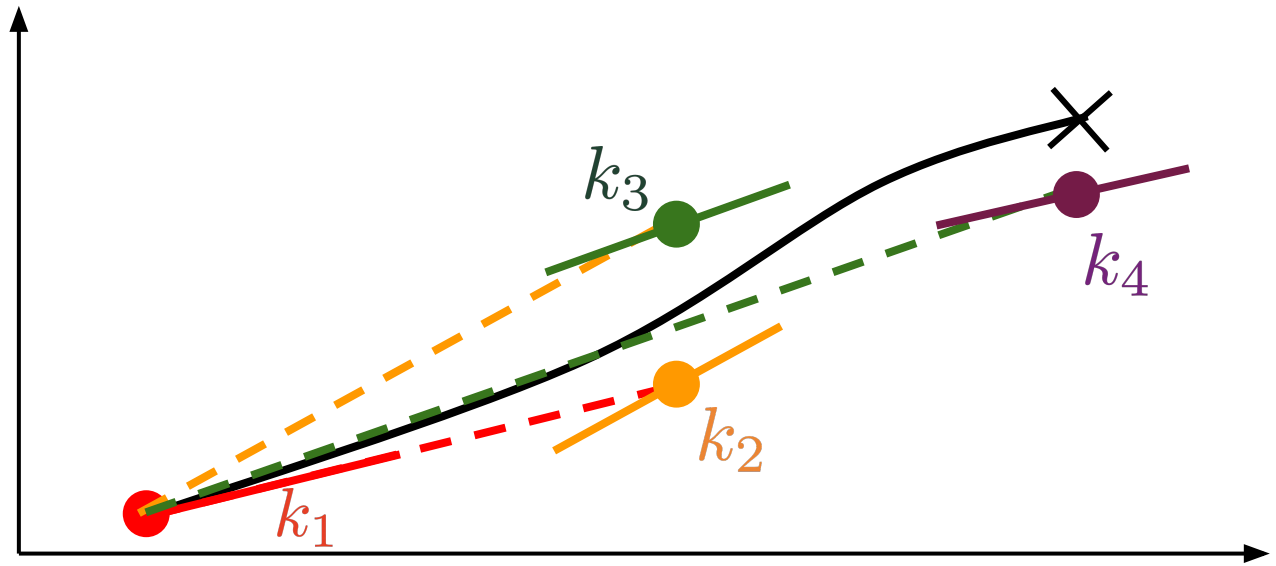
Euler method. 1 NFE, large error

$$x_{t_{i+1}} = x_{t_i} + v(x_{t_i}, t_i)\Delta t$$



More commonly used method

Runge-Kutta 4. 4 NFE, smaller error $x_{t_{i+1}} = x_{t_i} + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}\Delta t$



Properties of the ODE

Problem statement. $dx = v(x,t)dt$ with $x(t_0) = x_0$

Justification: Definition of v

$$f(x,t) - \frac{1}{2}g(t)^2 \nabla_x \log p_t(x)$$

DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps, Lu et al., 2022.

Properties of the ODE

Problem statement. $dx = v(x,t)dt$ with $x(t_0) = x_0$

$$f(x,t) - \frac{1}{2}g(t)^2 \nabla_x \log p_t(x)$$

Justification: Diffusion case

$$f(t)x - \frac{1}{2}g(t)^2 \nabla_x \log p_t(x)$$

DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps, Lu et al., 2022.

Properties of the ODE

Problem statement. $dx = v(x,t)dt$ with $x(t_0) = x_0$

$$f(x,t) - \frac{1}{2}g(t)^2 \nabla_x \log p_t(x)$$

$$f(t)x - \frac{1}{2}g(t)^2 \nabla_x \log p_t(x)$$

linear in x non-linear in x

DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps, Lu et al., 2022.

Is there a better way?

Traditional solvers. $dx = \left[f(t)x - \frac{1}{2} g(t)^2 \nabla_x \log p_t(x) \right] dt$

discretize

"Smarter" solver. $dx = \left[f(t)x \right] - \left[\frac{1}{2} g(t)^2 \nabla_x \log p_t(x) \right] dt$

solve exactly discretize

DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps, Lu et al., 2022.

Solver tailored to PF-ODE

DPM-Solver = **D**iffusion **P**robabilistic **M**odel-Solver

$$x_{t_{i-1}} = \frac{\alpha_{t_{i-1}}}{\alpha_{t_i}} x_{t_i} - \alpha_{t_{i-1}} \int_{\lambda_{t_i}}^{\lambda_{t_{i-1}}} e^{-\lambda} \hat{\epsilon}_\theta(\hat{x}_{t_\lambda}, t_\lambda) d\lambda$$

with $\lambda_t \triangleq \log \frac{\alpha_t}{\sigma_t}$ and $\hat{\epsilon}_\theta(x_t, t) = -\sigma_t \hat{s}_\theta(x_t, t)$

Derivation: Variation of constants and change of variables

DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps, Lu et al., 2022.

Solver tailored to PF-ODE

DPM-Solver = **D**iffusion **P**robabilistic **M**odel-Solver

$$x_{t_{i-1}} = \frac{\alpha_{t_{i-1}}}{\alpha_{t_i}} x_{t_i} - \alpha_{t_{i-1}} \int_{\lambda_{t_i}}^{\lambda_{t_{i-1}}} e^{-\lambda} \hat{\epsilon}_\theta(\hat{x}_{t_\lambda}, t_\lambda) d\lambda$$

what to do with this term?

with $\lambda_t \triangleq \log \frac{\alpha_t}{\sigma_t}$ and $\hat{\epsilon}_\theta(x_t, t) = -\sigma_t \hat{s}_\theta(x_t, t)$

DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps, Lu et al., 2022.

Variants of DPM-Solver

DPM-Solver-1. $\hat{\epsilon}_\theta(\hat{x}_{t_\lambda}, t_\lambda) \approx \hat{\epsilon}_\theta(\hat{x}_{t_i}, t_i)$ 1 NFE

DPM-Solver-2. $\hat{\epsilon}_\theta(\hat{x}_{t_\lambda}, t_\lambda) \approx \hat{\epsilon}_\theta(\hat{x}_{t_i}, t_i) + (\lambda - \lambda_{t_i}) \frac{\delta \hat{\epsilon}_\theta(\hat{x}_{t_i}, t_i)}{\delta \lambda}$ 2 NFE

⋮

DPM-Solver-k. Taylor expansion of order (k-1) k NFE

DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps, Lu et al., 2022.

DPM-Solver conclusion

Mindset.

- Forward SDE → Fokker-Planck eq. → Continuity eq. → **PF-ODE**
- Leveraging structure of PF-ODE → **DPM-Solver**

DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps, Lu et al., 2022.

DPM-Solver conclusion

Mindset.

- Forward SDE → Fokker-Planck eq. → Continuity eq. → **PF-ODE**
- Leveraging structure of PF-ODE → **DPM-Solver**

Performance.

- Beats “traditional” solvers given an NFE budget
- Difference most pronounced as low NFE regimes

DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps, Lu et al., 2022.

DPM-Solver conclusion

Mindset.

- Forward SDE → Fokker-Planck eq. → Continuity eq. → **PF-ODE**
- Leveraging structure of PF-ODE → **DPM-Solver**

Performance.

- Beats “traditional” solvers given an NFE budget
- Difference most pronounced as low NFE regimes

Use in practice.

- No retraining needed
- Reasonable results after only 10-20 NFEs

DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps, Lu et al., 2022.

Thank you for your attention!