Course Materials for GEN-AI

Northeastern University

These materials have been prepared and sourced for the course **GEN-AI** at Northeastern University. Every effort has been made to provide proper citations and credit for all referenced works.

If you believe any material has been inadequately cited or requires correction, please contact me at:

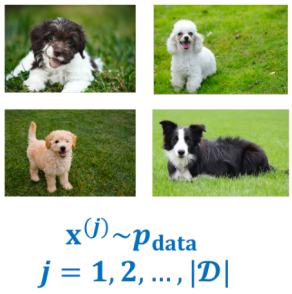
Instructor: Ramin Mohammadi
r.mohammadi@northeastern.edu

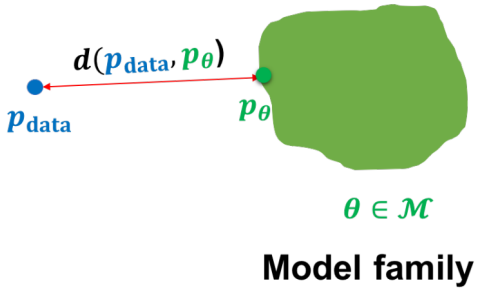
Thank you for your understanding and collaboration.

Content sourced from multiple references - see Reference section for details

Score-based Models

1 Introduction





- **Autoregressive Models:**

$$p_G(x) = \prod_{i=1}^{n} p_G(x_i \mid x_{<i})$$

- **Variational Autoencoders (VAE):**

$$p_G(x) = \int p_G(x, z)\, dz$$

- **Normalizing Flow Models:**

$$p_X(x; \theta) = p_Z\left(f_\theta^{-1}(x)\right) \left| \det \left(\frac{\partial f_\theta^{-1}(x)}{\partial x} \right) \right|$$

- **Energy-Based Models (EBM):**

$$p_G(x) = \frac{\exp(f_\theta(x))}{Z(\theta)}$$

  where $E_\theta(x)$ represents the energy function, and $Z(\theta)$ is the partition function that normalizes the probability distribution.

All the above families are trained by minimizing KL divergence $D_{KL}(p_{\text{data}} \,\|\, p_G)$ or equivalently maximizing likelihoods (or approximations).

Additionally, we have seen that we can learn distributions using likelihood-free models such as GANs and methods that approximate certain divergences or distances:

- **Generative Adversarial Networks (GANs):**

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))]$$

- **Two-Sample Tests:** These approaches can approximately optimize f-divergences and Wasserstein distances, which can be useful for learning generative models in a likelihood-free manner.

Very flexible model architectures. But likelihood is intractable, training is unstable, hard to evaluate, and has mode collapse issues.

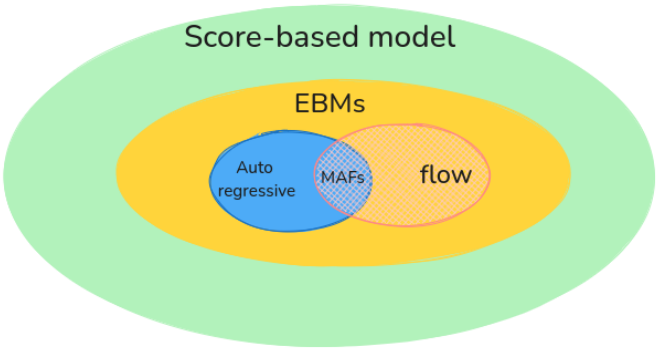
Sampling converges slowly in high-dimensional spaces and is thus very expensive. Yet, we need sampling for **each training iteration** in contrastive divergence.

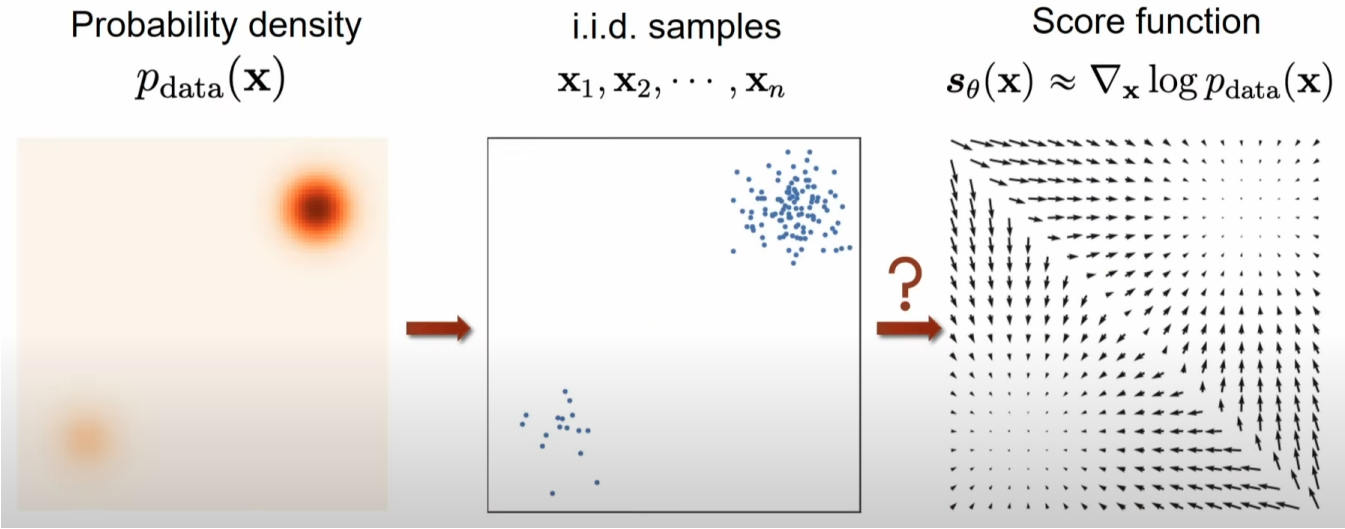
As we saw, sampling techniques tends to be expensive. Could we train without sampling? as an alternative to Contrastive Divergence algorithm? Yes, we could use methods like: **Score Matching, Noise Contrastive Estimation and Adversarial training** which do not depends on the constant function Z.

2 Score Based Models

Score-based models represent a versatile and powerful class of machine learning frameworks, characterized by their ability to be efficiently trained using score matching techniques. Unlike energy-based models, which explicitly parameterize the energy function, score-based models directly estimate the gradient field of the data distribution.

This approach offers greater flexibility and often simplifies the optimization process. By focusing on the score function of the gradient of the log probability density, these models excel in tasks such as generative modeling, sampling, and denoising.

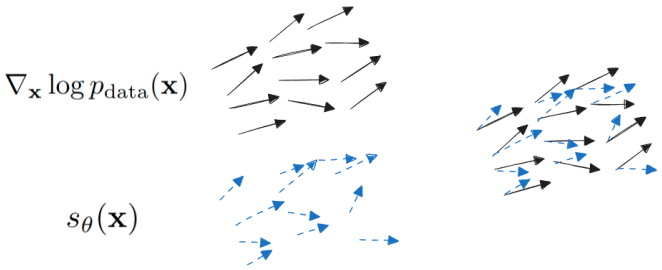




In practice, $p_{\text{data}}(\mathbf{x})$ is unknown, but we assume access to samples drawn from it. In score based models, the model is not a likelihood or energy and instead the model is going to be vector-valued function(s). This is due that by changing θ which is our NNs, we will get a different vector fields which will be used to describe our distribution.

2.1 Score Matching

- **Given:** i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- **Task:** Estimating the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ as p_{data} is unknown and we can not have the actual $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$.
- **Score Model:** A learnable vector-valued function $\mathbf{s}_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$ parametrized by NNs.
- **Goal:** To choose θ somehow that $\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- How to compare two vector fields of scores?



One idea is to compare the true gradients (black arrows) vs estimated gradients (blue arrows) via average euclidean distance.

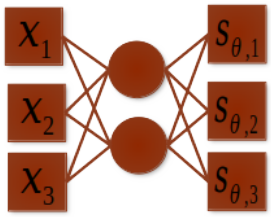
This is independent of the partition function $Z(\theta)$. The idea is that if p and q are similar for a given x then they should also have similar vector fields of gradients.

Score Matching:

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

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

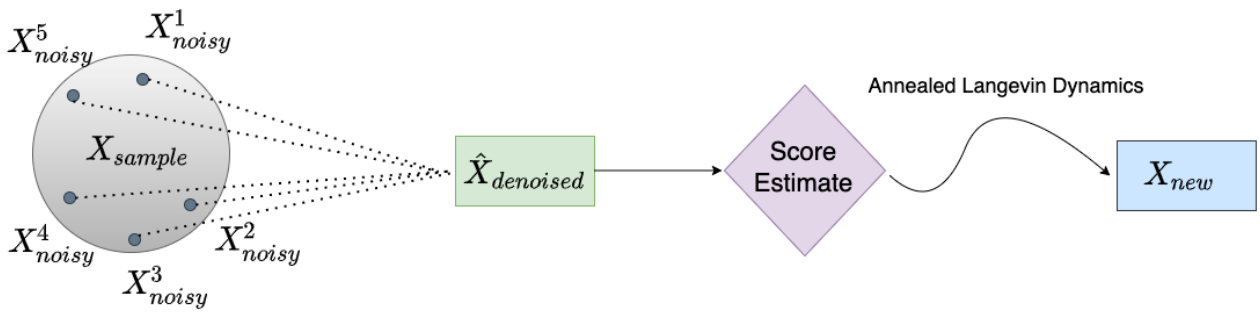
Caveat: Computing the trace of the Jacobian $\text{tr}(\nabla_{\mathbf{x}} S_\theta(\mathbf{x}))$ is, in general, very expensive for large models. Calculating this gradient will requires number of back-propagation steps which increase linearly wrt dimension.



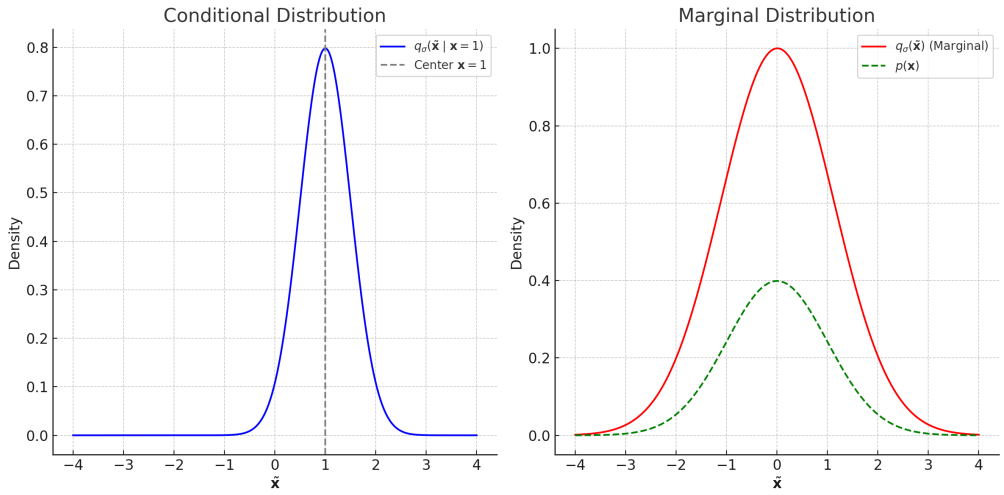
$$O(d)\text{Backprops!}$$

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

2.2 Denoising Score Matching



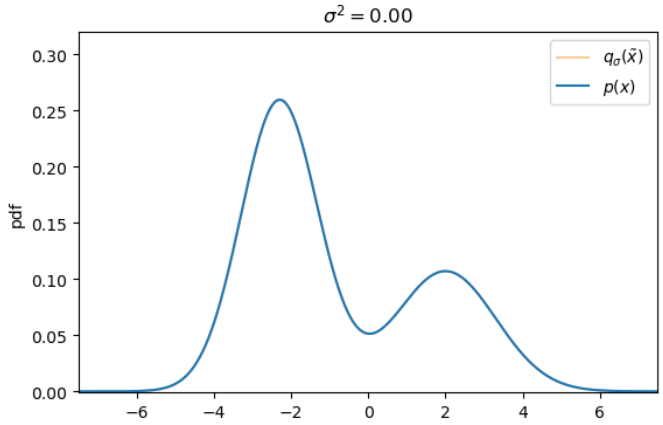
The idea is instead of estimating the gradient of the data we estimate the gradient of the data perturbed with noise.



$$\underbrace{q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = \mathcal{N}(\mathbf{x}; \sigma^2 \mathbf{I})}_{\text{conditional}} \qquad \underbrace{q_\sigma(\tilde{\mathbf{x}}) = \int p(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\, d\mathbf{x}}_{\text{marginal}}$$

It turns out that:

- Score estimation for $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})$ is easier.
- If the noise level is small, this is a good approximation.



https://johfischer.com/wp-content/uploads/2022/12/noisy-2.gif

Defining the explicit score-matching objective for the corrupted dataset $q_\sigma(\tilde{\mathbf{x}})$ yields:

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

$\mathbf{x}$ $\qquad$ $\tilde{\mathbf{x}}$

$$p_{\text{data}}(\mathbf{x}) \dashrightarrow q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \dashrightarrow q_\sigma(\tilde{\mathbf{x}})$$

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

$$= \frac{1}{2}\int q_\sigma(\tilde{\mathbf{x}})\| - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) + s_\theta(\tilde{\mathbf{x}})\|_2^2 \, d\tilde{\mathbf{x}}$$

$$= \frac{1}{2}\int q_\sigma(\tilde{\mathbf{x}})\|\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})\|_2^2 \, d\tilde{\mathbf{x}} + \frac{1}{2}\int q_\sigma(\tilde{\mathbf{x}})\|s_\theta(\tilde{\mathbf{x}})\|_2^2 \, d\tilde{\mathbf{x}}$$

$$- \int q_\sigma(\tilde{\mathbf{x}})\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})^T s_\theta(\tilde{\mathbf{x}}) \, d\tilde{\mathbf{x}}$$

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

Gradient Expansion

$$- \int q_\sigma(\tilde{\mathbf{x}})\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})^T s_\theta(\tilde{\mathbf{x}}) \, d\tilde{\mathbf{x}}$$

$$= -\int q_\sigma(\tilde{\mathbf{x}}) \frac{1}{q_\sigma(\tilde{\mathbf{x}})} \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^T s_\theta(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}}$$

$$= -\int \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^T s_\theta(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}}$$

$$= -\int \nabla_{\tilde{\mathbf{x}}} \left(\int p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\, d\mathbf{x} \right)^T s_\theta(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}}$$

$$= -\int \left(\int p_{\text{data}}(\mathbf{x}) \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\, d\mathbf{x} \right)^T s_\theta(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}}.$$

$$\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = \frac{\nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})}{q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})}.$$

Thus, we rewrite $\nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$ as:

$$\nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}).$$

$$= -\int \left(\int p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\, d\mathbf{x} \right)^T s_\theta(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}}.$$

$$= -\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})} \left[\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})^T s_\theta(\tilde{\mathbf{x}}) \right].$$

Final Denoising Score Matching Objective

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

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

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

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

While $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})$ is a complex distribution and we thus cannot evaluate it, $q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$ follows a normal distribution $\mathcal{N}(\mathbf{x}, \sigma^2 \mathbf{I})$ with mean $\mathbf{x}$ and variance σ^2, as:

$$\tilde{\mathbf{x}} = \mathbf{x} + \epsilon \iff \tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \sigma^2 \mathbf{I}),$$

with $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$

Gradient Derivation

This allows us to easily derive the gradient:

$$\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = \nabla_{\tilde{\mathbf{x}}} \log \mathcal{N}(\mathbf{x}, \sigma^2 \mathbf{I}).$$

Expanding this:

$$\nabla_{\tilde{\mathbf{x}}} \log \mathcal{N}(\mathbf{x}, \sigma^2 \mathbf{I}) = \nabla_{\tilde{\mathbf{x}}} \log \frac{1}{\sqrt{2\pi\,\sigma^2}}\, \exp\!\left(-\frac{\left(\tilde{x} - x\right)^2}{2\,\sigma^2}\right).$$

The gradient of the log exponential term simplifies to:

$$= -\frac{1}{\sigma^2}(\tilde{\mathbf{x}} - \mathbf{x}).$$

Thus (Tweedie formula):

$$\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = \frac{\mathbf{x} - \tilde{\mathbf{x}}}{\sigma^2}.$$

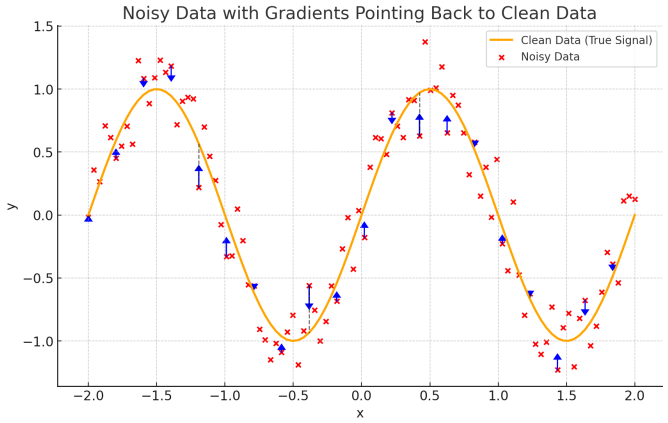
which means:

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

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

$$= \frac{1}{2n}\sum_{i=1}^{n}\left[\| \underbrace{s_\theta(\tilde{\mathbf{x}_\mathbf{i}})}_{\text{score of noisy x}} - \underbrace{\frac{\mathbf{x_i} - \tilde{\mathbf{x_i}}}{\sigma^2}}_{\text{Noise}}) \|_2^2\right]$$

- $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$: A clean data point $\mathbf{x}$ is sampled from the data distribution.
- $\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$: Noise is added to $\mathbf{x}$, producing a noisy version $\tilde{\mathbf{x}}$.
- $s_\theta(\tilde{\mathbf{x}})$: The score model takes the noisy data $\tilde{\mathbf{x}}$ as input and predicts the score.
- $\frac{\mathbf{x_i} - \tilde{\mathbf{x_i}}}{\sigma^2}$: The true image - noisy image which will return the noise.
- The loss function minimizes the squared error between score and the noise and our goal is to minimize this as much as possible which is why is called a denoising model.

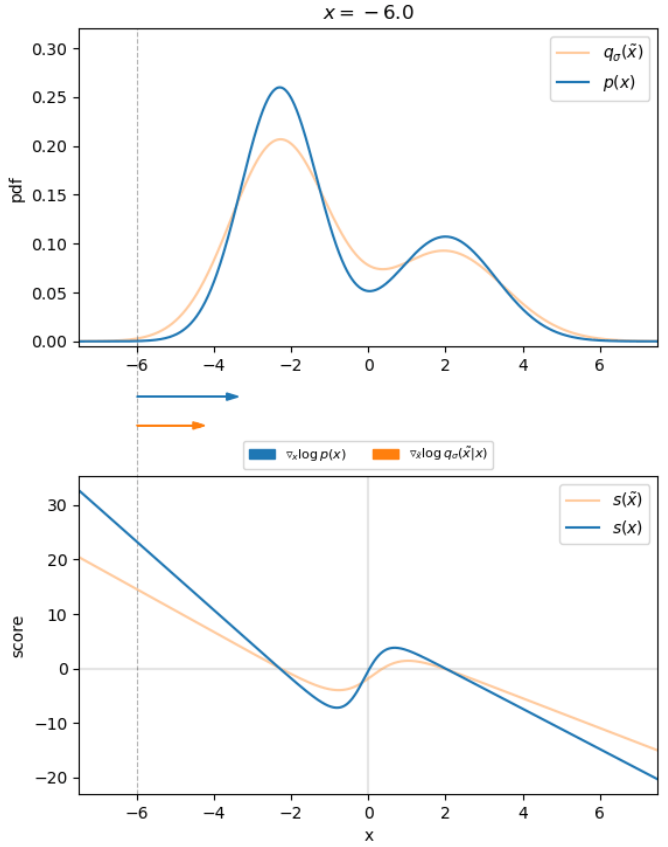


Pros: Efficient to optimize even for very high dimensional data, and useful for optimal denoising.

Con: Cannot estimate the score of clean data (noise-free).

2.2.1 Intuition

Intuitively, the gradient corresponds to the direction of moving from $\tilde{\mathbf{x}}$ back to the original $\mathbf{x}$ (i.e., denoising it), and we want our score-based model $s_\theta(\tilde{\mathbf{x}})$ to match this as best as it can.



https://johfischer.com/wp-content/uploads/2022/12/denoise-2.gif

For the **1D case**, we observe that the direction of the denoising score $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$ (→) almost perfectly matches the direction of the ground truth score $\nabla_{\mathbf{x}} \log p(\mathbf{x})$(→).

2.2.2 Why Denoising Works in Score Matching

With the denoising score matching objective we are able to circumvent the problem of not having access to the true score of our data distribution and additionally, with larger noise scales, also get signals in low density regions.

- Directly estimating the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ of the clean data distribution p_{data} is difficult because p_{data} is often unknown, complex, and discontinuous.
- Denoising score matching (DSM) simplifies this problem by adding Gaussian noise to the clean data $\mathbf{x}$, resulting in perturbed data $\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$, where q_σ is a smooth conditional Gaussian distribution.
- Instead of learning the score of p_{data}, we estimate the score of the noise-perturbed distribution $q_\sigma(\tilde{\mathbf{x}})$. The score function of the noise is analytically known:

$$\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = -\frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2}.$$

- By learning to **denoise** the noisy data $\tilde{\mathbf{x}}$ and predicting the direction back to the clean data $\mathbf{x}$, the score model $s_\theta(\tilde{\mathbf{x}})$ implicitly learns the structure of the noise.
- Importantly, learning the structure of the noise $q_\sigma(\tilde{\mathbf{x}})$ allows us to approximate the score of the clean data distribution:

$$\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) \quad \text{as} \quad \sigma \to 0.$$

- Therefore, denoising does not just recover clean data—it learns the score of the noise distribution, which bridges the gap to estimating the score of p_{data}. The score model s_θ then becomes a proxy for the true gradient of p_{data}, enabling downstream applications like sampling or density estimation.

2.3 Comparison between NSM and DSM

Aspect	**Normal Score Matching (NSM)**	**Denoising Score Matching (DSM)**
Gradient Computation	Requires the computation of **Hessian terms** $\nabla^2_{\mathbf{x}} \log p_\theta(\mathbf{x})$, which is costly for high-dimensional data.	No Hessians are needed. Only first-order gradients of the noise distribution are required, which are analytically known.
Smoothness of Target	Directly works with $p_{\text{data}}(\mathbf{x})$, which is often complex and discontinuous.	Perturbing the data smooths the distribution $q_\sigma(\tilde{\mathbf{x}})$, making the scores easier to estimate.
Training Efficiency	Computationally expensive and numerically unstable for high dimensions due to second-order derivatives.	Efficient and stable since it avoids second-order derivatives.
Dimensionality Issues	Computational cost scales poorly in high dimensions.	Handles high-dimensional data effectively with lower computational cost.

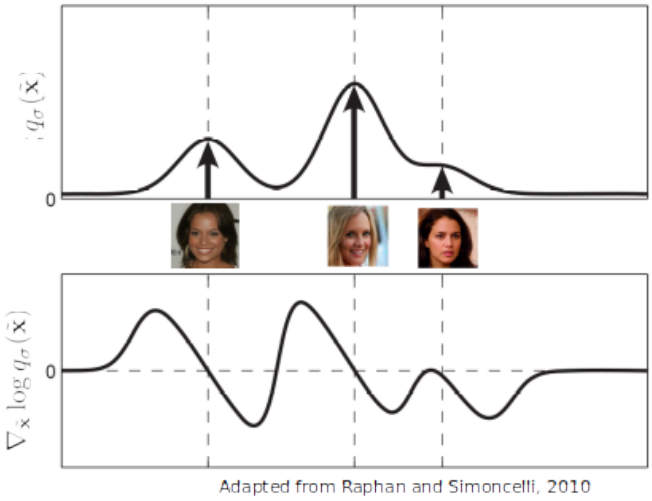
2.4 Tweedie formula

The **Tweedie formula** states that if we have a least-squares estimate x from a perturbed observation $\tilde{x}$, the score function can be estimated as:

$$\nabla_{\tilde{\mathbf{x}}} \log q(\tilde{\mathbf{x}}) = \frac{\mathbf{x} - \tilde{\mathbf{x}}}{\sigma^2},$$

where σ is the noise scale of the perturbation.

Optimal denoising strategy is to follow the gradient (score) : $x = \tilde{\mathbf{x}} + \sigma^2 \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})$



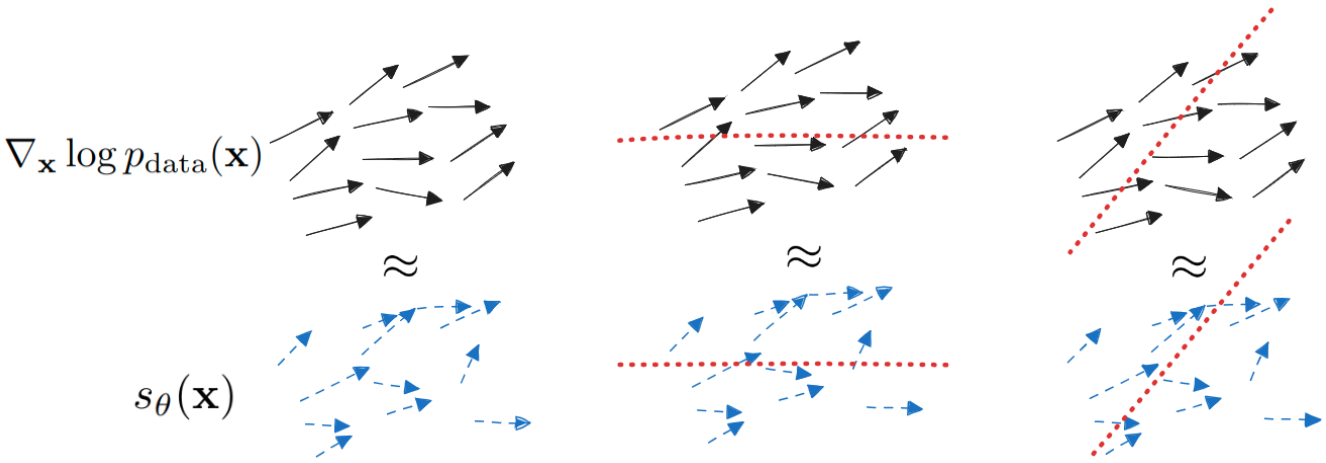
The **arrows** in the top plot indicate the direction in which the noisy samples $\tilde{\mathbf{x}}$ are pushed by the **score function** $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})$. These arrows point **toward the peaks** of the density $q_\sigma(\tilde{\mathbf{x}})$, where the probability of the clean data is highest.

In the bottom plot, the score function is visualized:

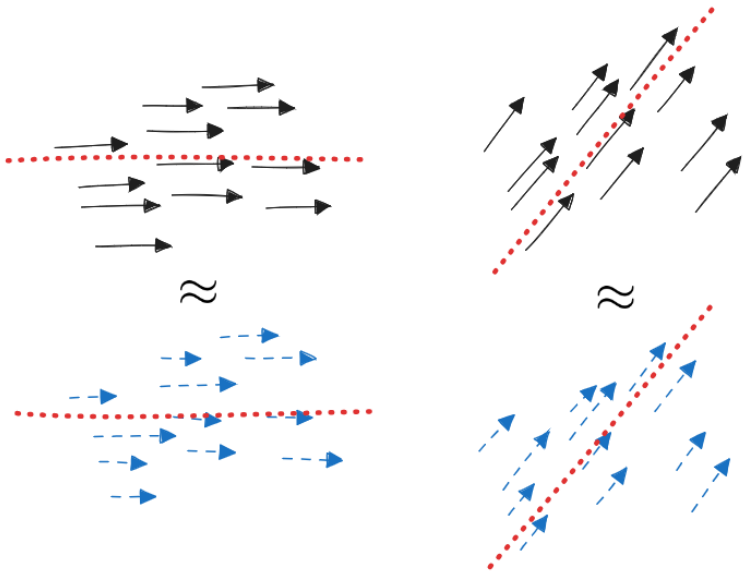
- It is **positive** when $\tilde{\mathbf{x}}$ lies to the left of a peak, pushing the sample rightward.
- It is **negative** when $\tilde{\mathbf{x}}$ lies to the right of a peak, pushing the sample leftward.
- The score becomes **zero** at the peaks, indicating no further movement.

Thus, the arrows represent the Tweedie formula's gradient-based denoising process, which pushes noisy samples toward the high-probability regions (peaks).

2.5 Sliced Score Matching



which means:



Sliced Score Matching (SSM) is an efficient alternative to standard score matching, designed to reduce computational costs in high-dimensional problems. Instead of matching the full-dimensional score, SSM projects the score function onto random directions (slices) and minimizes the score-matching loss in these lower-dimensional subspaces.

Given a random direction $\mathbf{v} \sim \mathcal{N}(0, I)$ and a score model $s_\theta(\mathbf{x})$, the sliced score matching loss is defined using the Sliced Fisher Divergence:

$$\mathcal{L}_{\text{SSM}} = \frac{1}{2}\mathbb{E}_{\mathbf{v}\sim p_{\mathbf{v}},\mathbf{x}\sim p_{\text{data}}}\left[\left(\underbrace{\mathbf{v}^T\nabla_{\mathbf{x}}\log p_{\text{data}}(\mathbf{x})}_{\text{projection along v}} - \underbrace{\mathbf{v}^T s_\theta(\mathbf{x})}_{\text{projection along v}}\right)^2\right].$$

The objective of sliced score matching is to minimize the **Sliced Fisher Divergence**, which quantifies the discrepancy between the projected gradients of the data distribution and the projected scores of the model, both along a randomly chosen direction $\mathbf{v}$.

Using **integration by parts**, the loss can be expressed in a more computationally efficient form as:

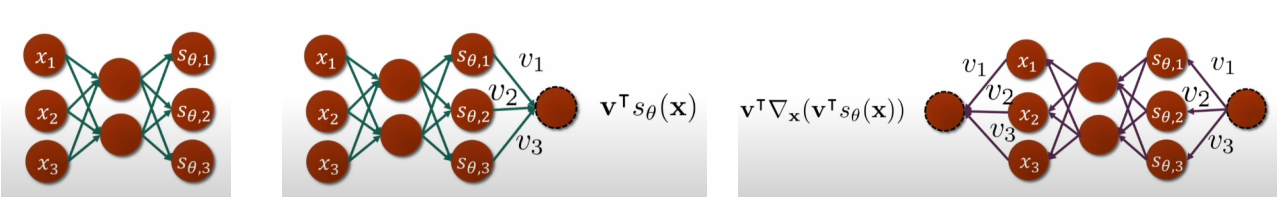
$$\mathbb{E}_{\mathbf{v}\sim p_{\mathbf{v}},\mathbf{x}\sim p_{\text{data}}}\left[\mathbf{v}^T\nabla_{\mathbf{x}}s_\theta(\mathbf{x})\mathbf{v} + \frac{1}{2}\left(\mathbf{v}^T s_\theta(\mathbf{x})\right)^2\right].$$

To compute the gradient and its projection efficiently, the Jacobian of the score model $s_\theta(\mathbf{x})$ is represented as:

$$\begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix}\begin{pmatrix} \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_1} & \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_2} & \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_3} \\ \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_1} & \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_2} & \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_3} \\ \frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_1} & \frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_2} & \frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_3} \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}.$$

Here:

- $\mathbf{v}$ represents the random direction sampled from $\mathcal{N}(0, I)$.
- $\nabla_{\mathbf{x}}s_\theta(\mathbf{x})$ is the Jacobian of the score model $s_\theta(\mathbf{x})$.
- The quadratic form $\mathbf{v}^T\nabla_{\mathbf{x}}s_\theta(\mathbf{x})\mathbf{v}$ represents the projected gradient.



(a) Forward path (b) Projected forward path (c) Projected backward path

Steps:

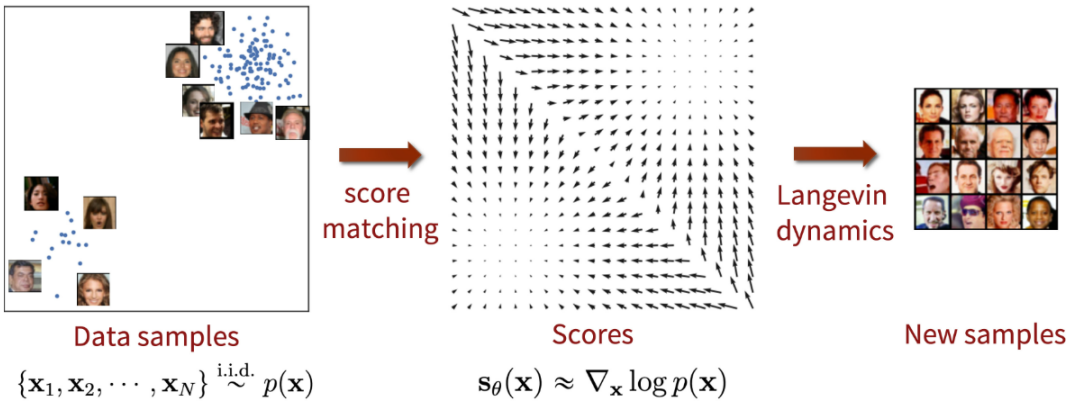
- Sample a minibatch of datapoints $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- Sample a minibatch of projection directions $\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n\} \sim p_\mathbf{v}$
- Estimate the sliced score matching loss with empirical means:

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

Notes:

- The projection distribution is typically Gaussian or Rademacher.
- Stochastic gradient descent is used for optimization.
- Using more projections per datapoint can boost performance.

2.6 Data Generation with Score based models



How do we generagte samples if we dont have access to likelihood and there is no autoregressive generation?

2.6.1 Langevin Dynamics

Once we have trained a score-based model $s_\theta(\mathbf{x}) \approx \nabla_\mathbf{x} \log p(\mathbf{x})$, we can use an iterative procedure called **Langevin dynamics** to draw samples from it.

Procedure

Langevin dynamics provides an MCMC procedure to sample from a distribution $p(\mathbf{x})$ using only its score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. It initializes the chain from an arbitrary prior distribution:

$$\mathbf{x}_0 \sim \pi(\mathbf{x}),$$

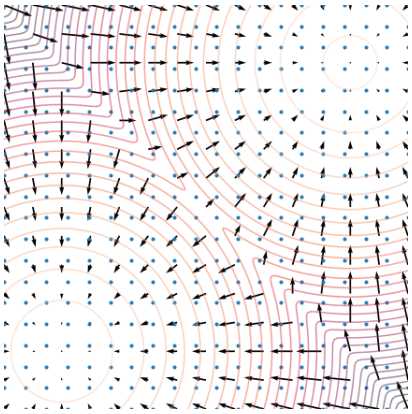
and iterates the following:

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}_i) + \sqrt{2\epsilon}\mathbf{z}_i, \quad i = 0, 1, \ldots, K,$$

where $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Convergence

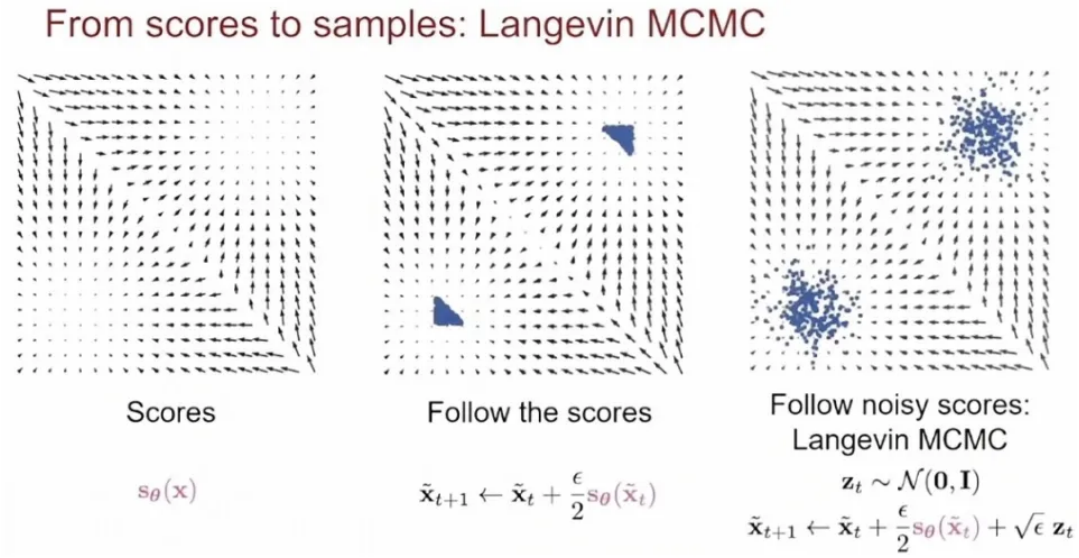
- As $\epsilon \to 0$ and $K \to \infty$, $\mathbf{x}_K$ converges to a sample from $p(\mathbf{x})$ under some regularity conditions.
- In practice, the error is negligible when ϵ is sufficiently small and K is sufficiently large.



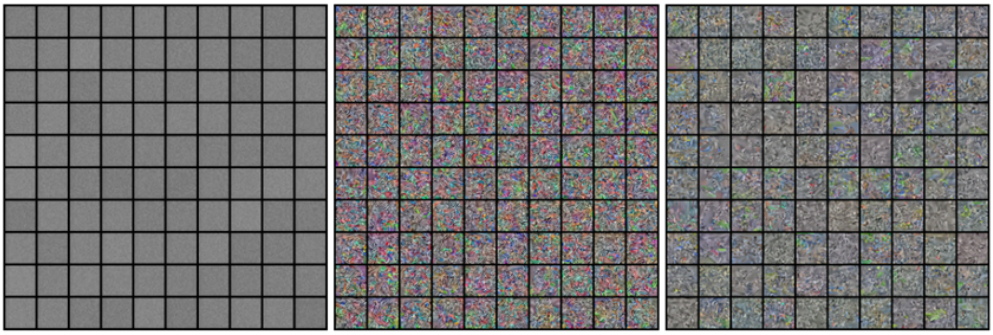
visualisation link

Key Insight

Note that Langevin dynamics accesses $p(\mathbf{x})$ only through $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. Since $s_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$, we can produce samples from our score-based model $s_\theta(\mathbf{x})$ by plugging it into the equation above.

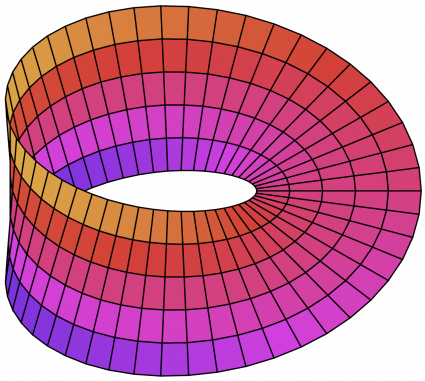


unfortunately the results generated by this approach will stuck in some local minima and wont be useful.



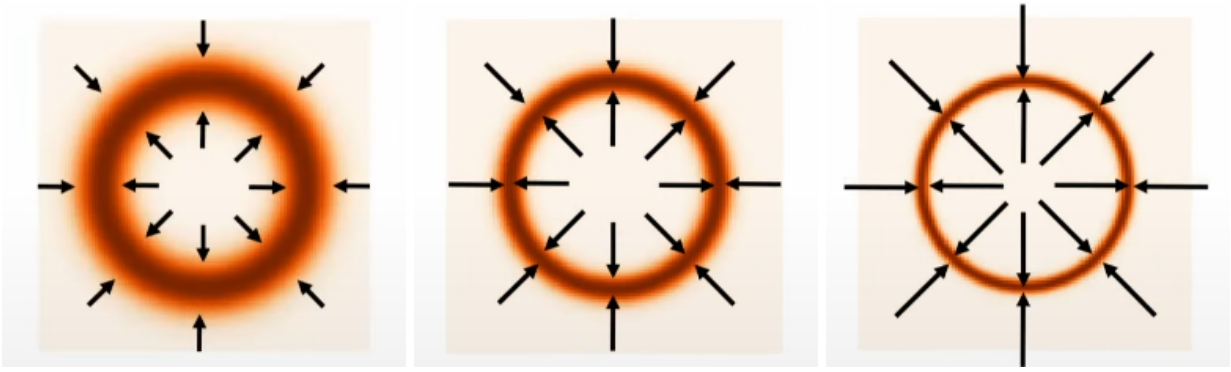
2.7 Pitfalls:

2.7.1 Pitfall 1: Manifold hypothesis



Manifold Hypothesis: The manifold hypothesis states that high-dimensional data (e.g., images, text, or audio) often lies on a lower-dimensional submanifold embedded in the high-dimensional space. For example, while an image has thousands of pixels (high-dimensional), the space of all realistic images occupies a much smaller subspace (low-dimensional).

Data tends to lie on a manifold which means the score might not be defined as our density is concentrated on a ring, as we make the ring thinner and thinner the magnitude of the gradient $\nabla_{\mathbf{x}} \log p_{data}(\mathbf{x})$ gets bigger and bigger and at some points becomes undefined.



The score function is defined as the gradient of the log-likelihood of the data distribution:

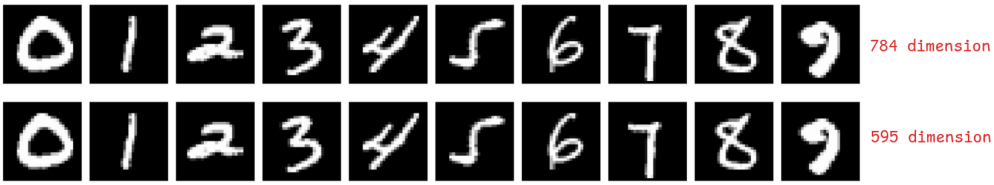
$$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}).$$

When data lies on a low-dimensional manifold:

- The probability density $p_{\text{data}}(\mathbf{x})$ is **concentrated** on or near the manifold.
- At points **far from the manifold**, the score function $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ becomes very large because the log-density drops off sharply.
- At points **on the manifold**, the score function may become **undefined or inconsistent**, especially near boundaries or in regions where the data is sparse.

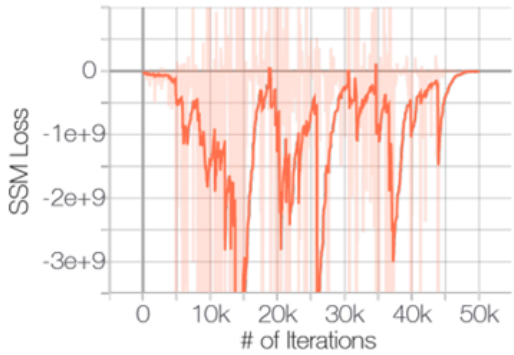
For example fitting linear manifold PCA on:

MNIST:



CIFAR10:

results to almost no difference and the score matching loss is really bumpy and doesn't quite train.



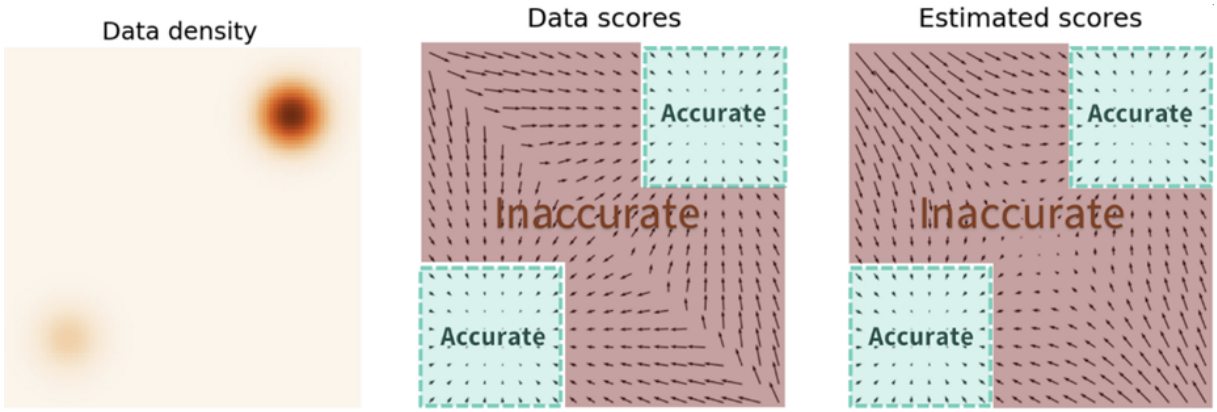
Score matching loss becomes noisy or unstable because the gradients explode in off-manifold regions. Learning the score function requires careful regularization and noise injection (e.g., through denoising score matching) to smooth the gradients and stabilize training.

2.7.2 Pitfall 2: Challenge in low data density regions

The key challenge is the fact that the estimated score functions are inaccurate in **low-density regions**, where few data points are available for computing the score matching objective. This is expected as score matching minimizes the Fisher divergence:

$$\mathbb{E}_{p(\mathbf{x})} \left[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - s_\theta(\mathbf{x})\|_2^2 \right].$$

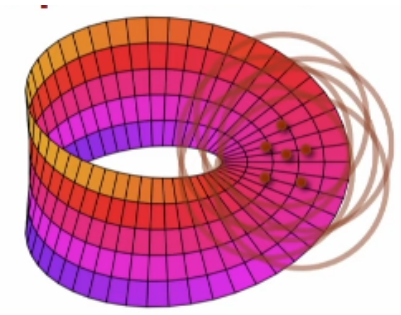
Since the ℓ_2 differences between the true data score function and the score-based model are weighted by $p(\mathbf{x})$, they are largely ignored in low-density regions where $p(\mathbf{x})$ is small. This behavior can lead to subpar results, as shown in the figure below:



When sampling with Langevin dynamics, the initial sample is highly likely to be in **low-density regions** when the data resides in a high-dimensional space. Therefore, having an inaccurate score-based model will derail Langevin dynamics from the very beginning of the procedure, preventing it from generating high-quality samples that are representative of the data.

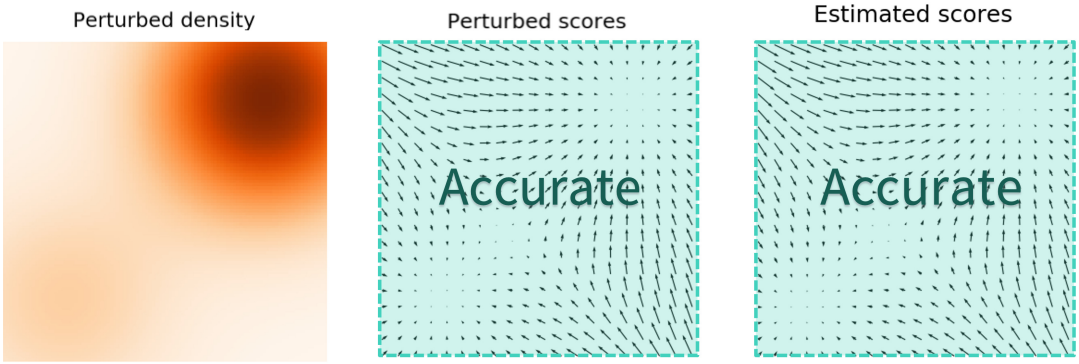
2.8 Solution to Pitfalls:

Adding noise via Gaussian perturbation which leads to having some data points falling outside of the manifold.



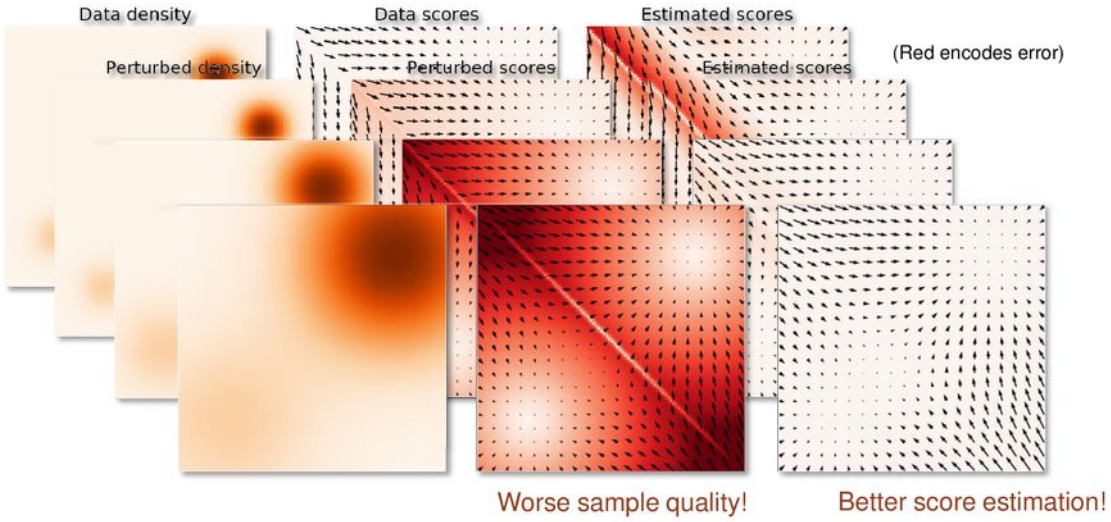
Manifold + Noise

When the noise magnitude is sufficiently large, it can populate low data density regions to improve the accuracy of estimated scores. Concretely, here is what happens when we perturb a mixture of two Gaussians with additional Gaussian noise.



- **Score Field based on different noise level**:

- The arrows represent the score function $\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})$, which points towards regions of high data density.
- At low noise levels (σ_1), the score field is highly concentrated and sharp, accurately capturing the data distribution but potentially noisy in regions with few data points.
- At higher noise levels (σ_3), the score field becomes smoother, covering larger regions of the space. However, the fine details of the true data distribution are lost.



- **Trade-off Between Noise and Convergence**:
  - **Low Noise (σ_1)**: While the score field is sharper and more accurate, the optimization can become unstable in low-density regions due to large gradients.
  - **Medium Noise (σ_2)**: A moderate noise level balances smoothness and accuracy, enabling stable convergence during score matching and sampling.
  - **High Noise (σ_3)**: At high noise levels, the score field is overly smooth. While this ensures stability, the model may fail to accurately recover fine details of the data distribution.
- **Implications for Sampling Quality**:
  - Langevin dynamics sampling relies on the accuracy of the score field. If the noise level is too high, the samples will lack detail and may converge slowly.

What we want to generate

What model generates

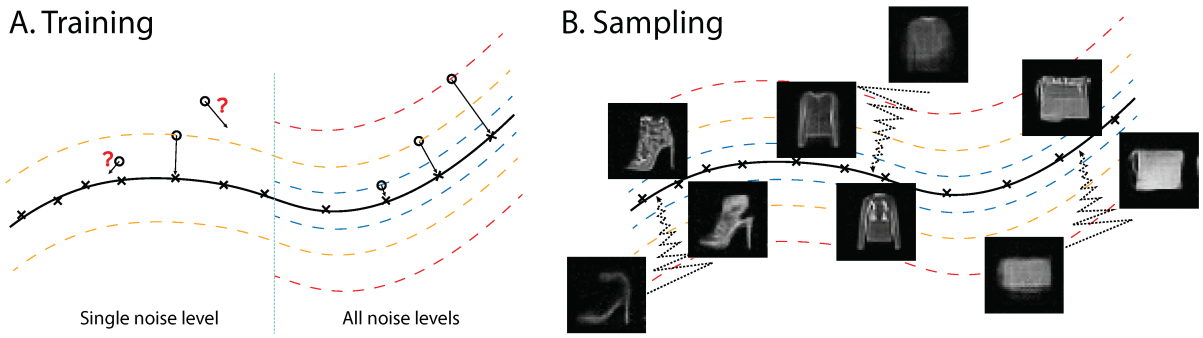
  - Conversely, with too little noise, the score estimation can become unstable, preventing convergence and resulting in poor sampling quality.

3 Noise Conditional Score-Based Model

To achieve the best of both worlds, we use multiple scales of Gaussian noise perturbations simultaneously. Suppose we always perturb the data with isotropic Gaussian noise. Let the noise levels be given by a sequence $\sigma_1 < \sigma_2 < \cdots < \sigma_L$. Specifically:

$$p_{\sigma_i}(\mathbf{x}) = \int p(\mathbf{y})\mathcal{N}(\mathbf{x};\mathbf{y},\sigma_i^2\mathbf{I})\,d\mathbf{y},$$

where $p(\mathbf{y})$ is the data distribution.



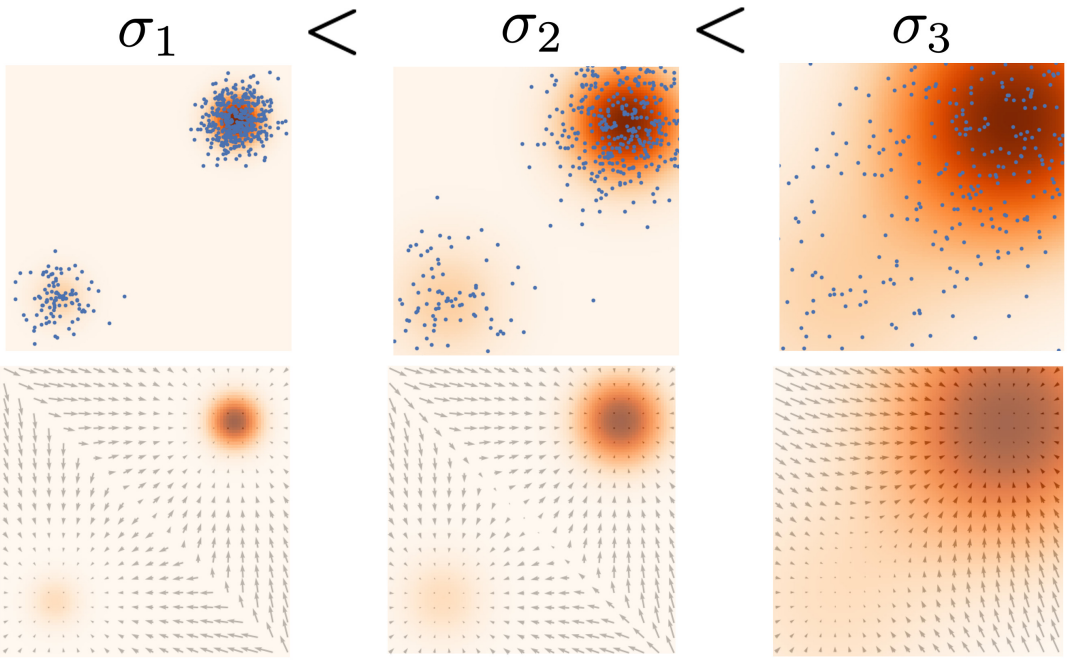
- **(A) Training:** During training, the derivative of the log-likelihood is forced to point toward the data manifold, establishing the energy difference between points within the manifold and those outside it. Since energy is defined as the negative log-likelihood, the energy is higher for points further away from the data manifold. This ensures that the gradient points towards the manifold, guiding the model to learn the underlying data distribution.
- **(B) Sampling:** During annealed Langevin sampling, the sample transitions from outside the data manifold to the data manifold. This process is guided by the learned energy function, which encourages movement toward regions of lower energy. Single-step denoised samples are visualized during the sampling of an energy function trained using Multiscale Denoising-Score Matching (MDSM) on the Fashion MNIST dataset.

During Training:

Training the Score Function: To estimate the score function $\nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x})$ of the noise-perturbed distributions $p_{\sigma_i}(\mathbf{x})$, we train a *Noise Conditional Score-Based Model* $\mathbf{s}_\theta(\mathbf{x}, \sigma_i)$. The training objective is defined as the weighted sum of Fisher divergences across all noise scales:

$$\sum_{i=1}^{L} \lambda(i)\mathbb{E}_{p_{\sigma_i}(\mathbf{x})}\left[\| \overbrace{\nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x})}^{\text{true score function}} - \underbrace{\mathbf{s}_\theta(\mathbf{x}, \sigma_i)}_{\text{noise Conditional model}} \|_2^2\right],$$

where $\lambda(i)$ is a positive weighting function, often chosen as $\lambda(i) = \sigma_i^2$. The objective ensures that $\mathbf{s}_\theta(\mathbf{x}, \sigma_i)$ approximates the true score functions $\nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x})$ across all noise scales.



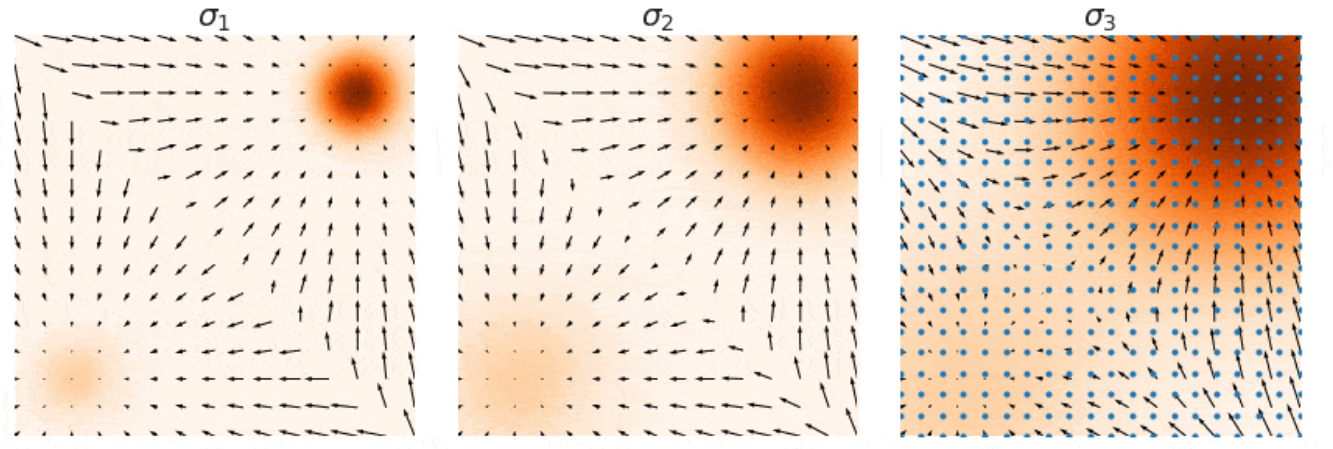
During Generation:

3.1 Annealed Langevin Dynamics

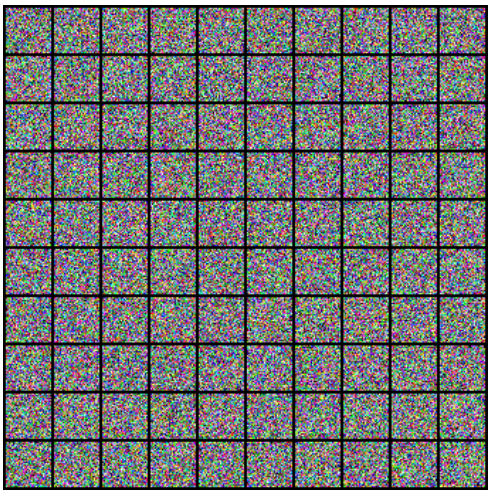
After training the noise-conditional score-based model $\mathbf{s}_\theta(\mathbf{x}, \sigma_i)$, we can sample from the learned data distribution using *Annealed Langevin Dynamics*. The process involves iteratively denoising a sample by descending along the learned score field for decreasing noise levels $\sigma_L, \ldots, \sigma_1$. The update step at each noise scale σ_i is:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \epsilon \mathbf{s}_\theta(\mathbf{x}_t, \sigma_i) + \sqrt{2\epsilon}\mathbf{z}_t, \quad \mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I}),$$

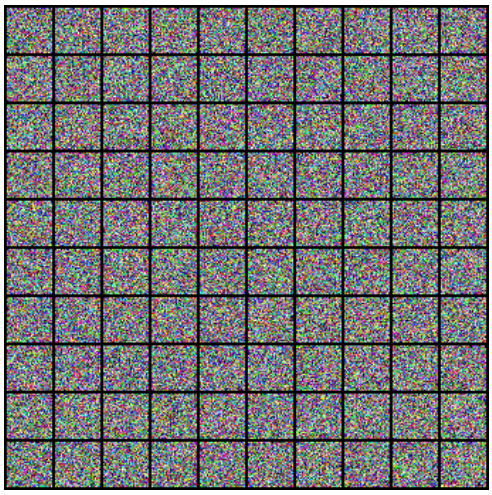
where ϵ is the step size and $\mathbf{z}_t$ is Gaussian noise added to ensure proper sampling.



Link to Animation

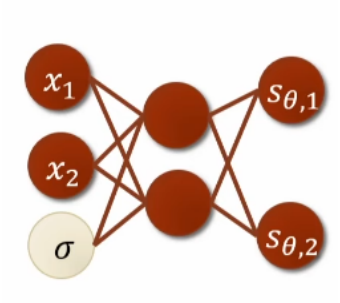


(a) Link to Animation



(b) Link to Animation

Intuition: Starting from random noise, the sample progressively transitions through noise scales $\sigma_L > \sigma_{L-1} > \cdots > \sigma_1$, following the learned score functions $\mathbf{s}_\theta(\mathbf{x}, \sigma_i)$ at each level. As the noise scale anneals (decreases), the sample moves closer to the data manifold.



We can approach this problem in two ways: by training separate models for each noise level or by using a single conditional neural network (NN) that jointly learns the score function across all noise levels. The conditional model incorporates a weighting parameter $\lambda(\sigma_i)$, which adapts the importance of accurately estimating the scores at different noise scales. Specifically, for larger noise levels, the emphasis is reduced, while smaller noise levels are prioritized to ensure higher precision near the data manifold.

3.1.1 Training Noise Conditional Score Networks

- **Which score matching loss?**
  - Sliced score matching?
  - Denoising score matching?
- **Denoising score matching** is naturally suitable, since the goal is to estimate the score of perturbed data distributions.
- **Weighted combination of denoising score matching losses:**

$$\frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)\mathbb{E}_{q_{\sigma_i}(\mathbf{x})}\left[\|\nabla_{\mathbf{x}}\log q_{\sigma_i}(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x},\sigma_i)\|_2^2\right]$$

$$= \frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)\mathbb{E}_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}\left[\|\nabla_{\tilde{\mathbf{x}}}\log q_{\sigma_i}(\tilde{\mathbf{x}}\mid\mathbf{x}) - \mathbf{s}_\theta(\tilde{\mathbf{x}},\sigma_i)\|_2^2\right] + \text{const.}$$

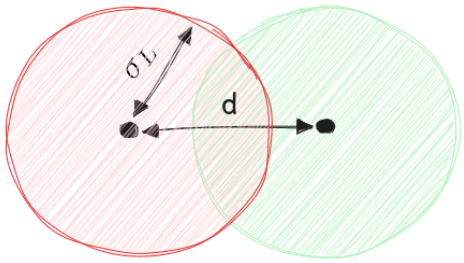
$$= \frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)\mathbb{E}_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}\left[\left\|\mathbf{s}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\frac{\mathbf{z}}{\sigma_i}\right\|_2^2\right]+\text{const.}$$

This process ensures that the samples are gradually refined to match the true data distribution.

3.2 Choosing Noise Scales

- **Maximum noise scale**

$$\sigma_L \approx \text{maximum pairwise distance between datapoints.}$$



The key idea is to ensure that the noise level is sufficiently large so that it becomes feasible to transition between any two data points in the dataset, such as $\mathbf{x}_1$ and $\mathbf{x}_2$. Specifically, if you start from $\mathbf{x}_1$ and add a sufficiently large amount of noise, there should be a reasonable probability of reaching $\mathbf{x}_2$. Conversely, the same holds true when starting from $\mathbf{x}_2$ and transitioning to $\mathbf{x}_1$. This ensures that the noise effectively connects different regions of the data space.

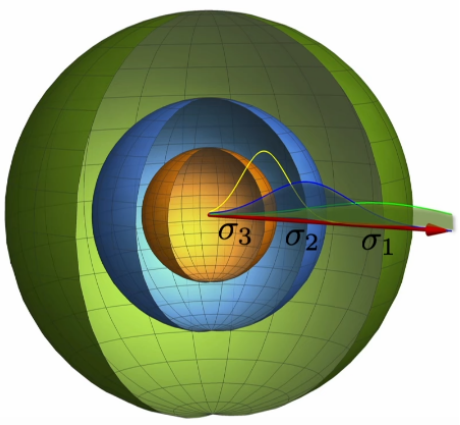
- **Minimum noise scale**

$$\sigma_1 \approx \text{minimum pairwise distance between datapoints.}$$

The noise level should be sufficiently small so that the structure of the data is preserved, ensuring that the noise in final samples is negligible.

- **Key intuition:** Adjacent noise scales should have sufficient overlap to facilitate transitioning across noise scales in annealed Langevin dynamics.
- A geometric progression with sufficient length.

$$\sigma_1 > \sigma_2 > \sigma_3 > \cdots > \sigma_{L-1} > \sigma_L$$
$$\frac{\sigma_1}{\sigma_2} = \frac{\sigma_2}{\sigma_3} = \cdots = \frac{\sigma_{L-1}}{\sigma_L}$$



3.3 Choosing the Weighting Function

- **Weighted combination of denoising score matching losses:**

$$\frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)\mathbb{E}_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}\left[\left\|\mathbf{s}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\frac{\mathbf{z}}{\sigma_i}\right\|_2^2\right]$$

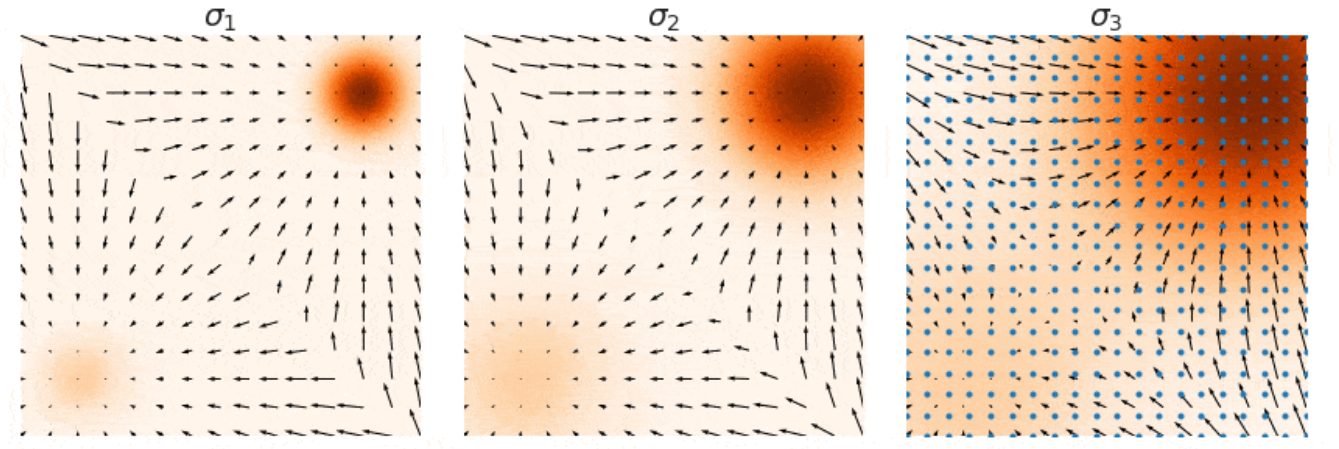
- **How to choose the weighting function** $\lambda:\mathbb{R}_{>0}\to\mathbb{R}_{>0}$?
- **Goal:** Balancing different score matching losses in the sum $\implies \lambda(\sigma_i)=\sigma_i^2$.

$$\frac{1}{L}\sum_{i=1}^{L}\sigma_i^2\mathbb{E}_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}\left[\left\|\mathbf{s}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\frac{\mathbf{z}}{\sigma_i}\right\|_2^2\right]$$

$$=\frac{1}{L}\sum_{i=1}^{L}\mathbb{E}_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}\left[\|\sigma_i\mathbf{s}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\mathbf{z}\|_2^2\right]$$

$$=\frac{1}{L}\sum_{i=1}^{L}\mathbb{E}_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}\left[\|\boldsymbol{\epsilon}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\mathbf{z}\|_2^2\right]\quad[\boldsymbol{\epsilon}_\theta(\cdot,\sigma_i)=-\sigma_i\mathbf{s}_\theta(\cdot,\sigma_i)]$$

3.4 Training Noise Conditional Score Networks



Link to Animation

- Sample a mini-batch of datapoints:

$$\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n\}\sim p_{\text{data}}$$

- Sample a mini-batch of noise scale indices:

$$\{i_1,i_2,\ldots,i_n\}\sim\mathcal{U}(\{1,2,\ldots,L\}),$$

where i_k is the noise scale index for datapoint $\mathbf{x}_k$, chosen uniformly at random from $\{1,2,\ldots,L\}$.

- Sample a mini-batch of Gaussian noise:

$$\{\mathbf{z}_1,\mathbf{z}_2,\ldots,\mathbf{z}_n\}\sim\mathcal{N}(0,\mathbf{I}).$$

- Perturb each datapoint with its corresponding noise scale: For each $k\in\{1,\ldots,n\}$, compute:

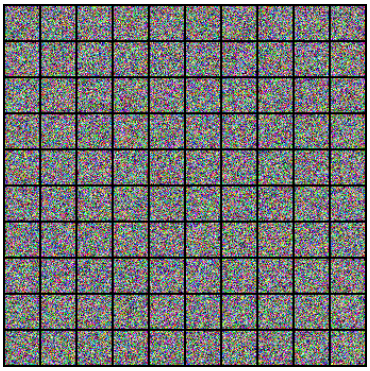
$$\tilde{\mathbf{x}}_k=\mathbf{x}_k+\sigma_{i_k}\mathbf{z}_k,$$

where σ_{i_k} is the noise level associated with index i_k.

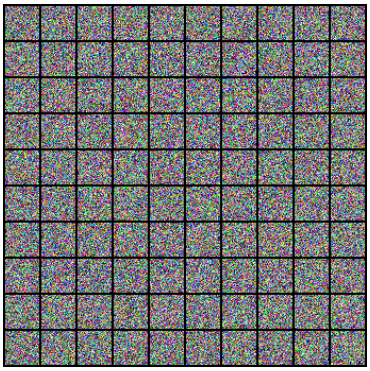
- Estimate the weighted mixture of score matching losses:

$$\frac{1}{n}\sum_{k=1}^{n}\|\sigma_{i_k}\mathbf{s}_\theta(\tilde{\mathbf{x}}_k,\sigma_{i_k})+\mathbf{z}_k\|_2^2\,.$$

- Optimize using stochastic gradient descent.



(a) Link to Animation



(b) Link to Animation

References

[1] Stefano Ermon. *CS236*.

[2] Yang Song Generative Modeling by Estimating Gradients of the Data Distribution