Diffusion model

for machine learning

Tran Trong Khiem

AI lab tranning

2024/08/01

Introduction

Math for machine learning :

- Complete Foundation chapter in Probabilistic Machine Learning[1]
- Probability and Statistics.
- Linear Algebra.
- Optimazation.

Generative AI:

- Gan
- VAE
- Flow-base
- Diffusion model

Generative AI

Existing **generative modeling techniques** can largely be grouped into two categories based on how they **represent probability distributions**.

1. **likelihood-based models**: which **directly learn the distribution**'s probability density (or mass) function via (approximate) maximum likelihood.(VAEs, EBMs, ...)
    - **Cons**: require strong restrictions on the model architecture to ensure a tractable normalizing constant for **likelihood computation**.
2. **implicit generative models**: where the **probability distribution** is implicitly represented by a model of its sampling process.(Gan,...)
    - **Cons**: unstable and can lead to model collapse.

Diffusion model introduces **another way** to represent **probability distributions** that circumvent several of these limitations.

Diffusion model

The **key idea** is to model the gradient of the log probability density function, score function.

- **score-based models** are not required to have a tractable normalizing constant, and can be directly learned by score matching.Better than GAN in image generation.

Denote :

- The dataset consists of i.i.d. samples $\{x_i \in \mathbb{R}^D\}_{i=1}^N$ from an **unknown data distribution** $p_{\text{data}}(x)$.
- The **score** of a probability density $p(x)$ is defined as $\nabla_x \log p(x)$.
- The score network $s_\theta : \mathbb{R}^D \to \mathbb{R}^D$, which will be **trained to approximate** the score of $p_{data}(x)$.

The **framework of score-based generative modeling**:

1. score matching
2. Langevin dynamics.

Framework of score-based generative modeling

Score matching :

- train a **score network** $s_\theta(x)$ to estimate $\nabla_x \log p_{\text{data}}(x)$ without training a model to estimate $p_{data}(x)$

Langevin dynamics

- produce samples from a probability density $p(x)$ **using only the score function** $\nabla_x \log p_{\text{data}}(x)$.

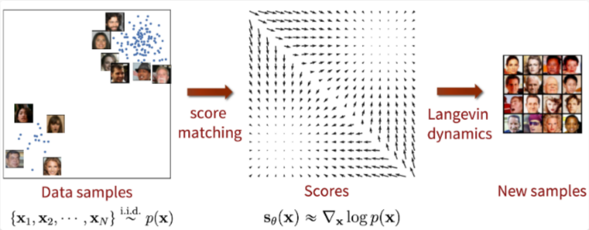


Figure 1: Score-based generative modeling with score matching + Langevin dynamics.

Score matching for score estimation

Goal: train a **score network** $s_\theta(x)$ to estimate $\nabla_x \log p_{\text{data}}(x)$.
The objective minimizes :

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

which can be shown equivalent to the following up to a constant :

$$\mathbb{E}_{p_{\text{data}}(x)}\left[\operatorname{tr}(\nabla_x s_\theta(x)) + \frac{1}{2}\|s_\theta(x)\|_2^2\right]$$

Problem: Score matching is not scalable to deep networks and high-dimensional data due to the computation of $\operatorname{tr}(\nabla_x s_\theta(x))$.
Solution: There are two popular methods for large scale score matching.

1. Denoising score matching
2. Sliced score matching

Score matching for score estimation(.cnt)

Denoising score matching:

- completely circumvents $\mathrm{tr}(\nabla_x s_\theta(x))$.
- perturbs the data point x with a prespecified noise $q_\sigma(\tilde{x} \mid x)$.
- employs score matching to estimate the score of the perturbed data.

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

- However, $s_\theta^*(x) = \nabla_x \log q_\sigma(x) \approx \nabla_x \log p_{\text{data}}(x)$ is true only when the noise is small enough such that $q_\sigma(x) \approx p_{\text{data}}(x)$.

Sliced score matching:

- uses random projections to approximate $\mathrm{tr}(\nabla_x s_\theta(x))$.
- The objective is:

$$\frac{1}{2}\left|\mathbb{E}_{p_v}\mathbb{E}_{p_{\text{data}}}\left[v\nabla_x s_\theta(x)v + \|s_\theta(x)\|_2^2\right]\right|$$

- p_v is a simple distribution of random vectors.

Sampling with Langevin dynamics

Goal: produce samples from a probability density $p(x)$ using only the score function $\nabla_x \log p(x)$.

- Given a fixed step size $\epsilon > 0$, and an initial value $\tilde{x}_0 \sim \pi(x)$
- π is a prior distribution.
- Langevin method recursively computes the following :

$$\tilde{x}_t = \tilde{x}_{t-1} + \frac{\epsilon}{2} \nabla_x \log p(\tilde{x}_{t-1}) + \epsilon z_t,$$

  - $z_t \sim \mathcal{N}(0, I)$
  - The distribution of $\tilde{x}_T$ equals $p(x)$ when $\epsilon \to 0$ and $T \to \infty$,
  - In practice, ϵ is small and T is large.

Challenges of score-based generative modeling

Inaccurate score estimation with score matching:

- In score matching, we minimize :

$$\mathbb{E}_{p_{\text{data}}}\left[\|s_\theta(x) - \nabla_x \log p_{\text{data}}(x)\|_2^2\right] = \int p(x)\left[\|s_\theta(x) - \nabla_x \log p_{\text{data}}(x)\|_2^2\right] dx$$

- Since square error weighted by $p(x)$, they are largely ignored in low density regions where $p(x)$ is small.

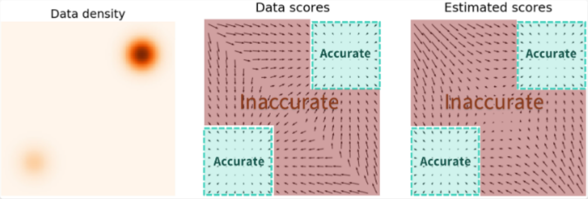


Figure 2: Estimated scores are only accurate in high density regions

How to bypass the inaccurate score estimation in regions of low data density?

Observation : **perturbing data** with random Gaussian noise makes the data distribution more amenable to score-based generative modeling.

- **large Gaussian noise** has the effect of filling low density regions in the original distribution.

Upon intuition is the key idea for Noise Conditional Score Networks(NCSN):

1. perturbing the data using various levels of noise.
2. simultaneously estimating scores corresponding to all noise levels by training a single conditional score network.

Noise Conditional Score Networks

Problem : How to choose an appropriate noise scale for the perturbation process?

- Larger noise over-corrupts the data and alters it significantly from the original distribution.
- Smaller noise, on the other hand, causes less corruption of the original data.

Solution: Use multiple scales of noise perturbations simultaneously.
Denote:

- $\{\sigma_i\}_{i=1}^{L}$ be a positive sequence geometric decending sequence.
- $q_\sigma(x) = \int p_{\text{data}}(t)\mathcal{N}(x \mid t, \sigma^2 I)\, dt$ the **perturbed data distribution**.
- $s_\theta(x, \sigma)$ is a Noise Conditional Score Network (NCSN).
- train model to jointly estimate the scores of **all perturbed data distributions** :

$$\forall \sigma \in \{\sigma_i\}_{i=1}^{L} : s_\theta(x, \sigma) \approx \nabla_x \log q_\sigma(x)$$

Learning NCSNs via score matching

Adapt **denoising score matching** for learning NCSNs.

- choose the noise distribution to be $q_\sigma(\tilde{x} \mid x) = \mathcal{N}(\tilde{x} \mid x, \sigma^2 I)$
- therefore $\nabla_{\tilde{x}} \log q_\sigma(\tilde{x} \mid x) = -\frac{\tilde{x}-x}{\sigma^2}$
- For a given σ, the denoising score matching objective is :

$$\mathcal{L}(\theta; \sigma) = \frac{1}{2} \mathbb{E}_{p_{\text{data}}(x)} \mathbb{E}_{\tilde{x} \sim \mathcal{N}(x, \sigma^2 I)} \left[\left\| s_\theta(\tilde{x}, \sigma) + \frac{\tilde{x} - x}{\sigma^2} \right\|_2^2 \right].$$

- We combine for all $\sigma \in \{\sigma_i\}_{i=1}^L$ to get one unified objective :

$$\mathcal{L}(\theta; \{\sigma_i\}_{i=1}^L) = \frac{1}{L} \sum_{i=1}^{L} \lambda(\sigma_i) \mathcal{L}(\theta; \sigma_i)$$

NCSN inference via annealed Langevin dynamics

- propose a sampling approach— annealed Langevin dynamics

Algorithm 1 Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T$.

Initialize $\tilde{\mathbf{x}}_0$
for $i \leftarrow 1$ to L **do**
  $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ ▷ α_i is the step size.
  for $t \leftarrow 1$ to T **do**
    Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
    $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\, \mathbf{z}_t$
  end for
  $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
end for
return $\tilde{\mathbf{x}}_T$

Figure 3: Annealed Langevin dynamics.

Denoising Diffusion Probabilistic Models

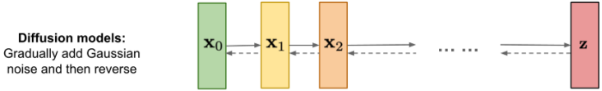


Figure 4: Diffusion model

Forward diffusion process:

- add small amount of Gaussian noise to the sample in T
- producing a sequence of noisy samples $x_1, x_2 \cdots x_T$
- converts any complex data distribution into a simple, tractable, distribution.

Reverse diffusion process:

- Learn a reveral of forward diffusion process.

Foward process

Gradually adds Gaussian noise to the data according to a variance schedule $\beta_1, \ldots, \beta_T$:

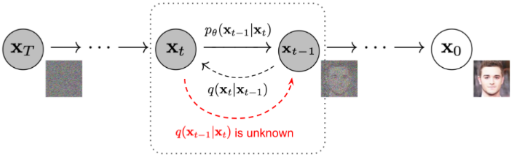
$$q(x_{1:T} \mid x_0) := \prod_{t=1}^{T} q(x_t \mid x_{t-1}), \quad q(x_t \mid x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t I)$$

Nice property: We can sample x_t at timestep t as :

$$x_t = \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t}\epsilon_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

- $\epsilon_t \sim \mathcal{N}(0, I)$
- $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_t$ and $\alpha_t = 1 - \beta_t$
- Thus we have : $q(x_t|x_0) = \mathcal{N}(x_t, \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t I))$

Reverse diffusion process



Goal: Learn to reverse the forward process and sample from $q(x_{t-1}|x_t)$.

- Use $p_\theta(x_{t-1}|x_t)$ to approximate $q(x_{t-1}|x_t)$.
- The reverse conditional probability is tractable when conditioned on x_0:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}, \tilde{\mu}(x_t, x_0), \tilde{\beta}_t I)$$

- $\tilde{\beta}_t = \frac{1-\bar{a}_{t-1}}{1-\bar{a}_t}\beta_t$ and $\tilde{\mu}(x_t, x_0) = \sqrt{\alpha_t}\left(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)x_t + \left(\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\right)x_0$

Reverse diffusion process(.cnt)

Training is performed by optimizing the usual variational bound on negative log likelihood:

$$\mathbb{E}_q\left[-\log p_\theta(x_0)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)}\right]$$
$$= \mathbb{E}_q\left[-\log p(x_T) - \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1})}\right] \qquad =: L.$$

Loss function can rewrite as :

$$\mathbb{E}_q\left[D_{KL}\left(q(x_T|x_0)||p(x_T)\right) + \sum_{t>1} D_{KL}\left(q(x_{t-1}|x_t,x_0)||p_\theta(x_{t-1}|x_t)\right) - \log p_\theta(x_0|x_1)\right] \tag{1}$$

Label each component in the variational lower bound loss separately:

- $L_{VLB} = \sum_{t=0}^{T} L_t$

Reverse diffusion process(.cnt)

The loss term L_t is parameterized and simplified to minimize :

$$L_t^{simple} = \mathbb{E}_{t\sim[1,T],x_0,\epsilon_t}\left[||\epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t t)||^2\right]$$

Algorithm 1 Training

repeat
 $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
 $t \sim \text{Uniform}(\{1,\ldots,T\})$
 $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
 Take gradient descent step on
 $\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$
until converged

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
for $t = T,\ldots,1$ **do**
 $\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
 $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z}$
end for
return $\mathbf{x}_0$

Figure 5: Traing process.

References

1. Weng, Lilian. (Jul 2021). What are diffusion models? Lil'Log.
2. Jonathan Ho,Ajay Jain,Pieter Abbeel, Denoising Diffusion Probabilistic Models
3. Yang Song,Stefano Ermon, Generative Modeling by Estimating Gradients of the Data Distribution
4. Generative Modeling by Estimating Gradients of the Data Distribution