401-4634-24L: Diffusion Models, Sampling and Stochastic Localization

Lecture 4 – Introduction to diffusion models

Lecturer: Yuansi Chen *Spring 2024*

Key concepts:

- Mixing of ULA. It was discussed in this lecture, but the notes were put into Lecture 4, as it fits better there
- Two major categories of generative modeling.
- Score function, score estimation via score matching
- Langevin with estimated score
- Diffusion models with estimated score (just the idea)

The material of this lecture is based on [SE19].

4.1 Introduction

Starting from this lecture, we move to the second setting of sampling where the target measure μ is given with a collection of N i.i.d. samples rather than its explicit form. In other words, we have $x_1, \ldots, x_N \overset{\text{i.i.d.}}{\sim} \mu$, while μ is not given explicitly. This setting is also called "generative modeling" in machine learning.

In 2021, Song and Ermon [SE19] categorized the existing generative modeling techniques into two major categories

- **Likelihood-based models**, which directly models the distribution's probability density function via (approximate) maximum likelihood. That is, we model

$$\mu(x) \propto p_\theta(x),$$

where $p_\theta(x)$ is the probability density parametrized by some parameter θ. We learn θ by maximizing the log-likelihood of the data

$$\max_\theta \sum_{i=1}^N \log p_\theta(x_i). \tag{4.1}$$

This category includes autoregressive models, normalizing flow models, energy-based models, and variational auto-encoders (VAEs). See reference in [SE19].

- **Implicit generative models**, which implicitly represent the probability distribution via a model of sampling process. That is, we say the target distribution is close to a transformation of a Gaussian
$$g(Z), Z \in \mathcal{N}(0, \mathbb{I}_m),$$
where $g : \mathbb{R}^m \to \mathbb{R}^n$ is a mapping to be learned. Then with a distance between two measures $\mathcal{D}_{\text{distributional}}(\cdot, \cdot)$, we want to find g to minimize
$$\mathcal{D}_{\text{distributional}}(g(Z), \frac{1}{N}\sum_{i=1}^{n} \delta_{x_i}(\cdot))$$
where $\frac{1}{N}\sum_{i=1}^{n} \delta_{x_i}(\cdot)$ denotes the empirical measure. The most prominent example is generative adversarial networks (GANs) [GPAM+14], where new samples are synthesized by transforming a random Gaussian vector with a neural network g. The parameters of the neural network g are learned via minimizing the adversarial loss between newly generated images and the empirical measure.

Song and Ermon pointed out that the existing models have significant limitations: "Likelihood-based models either require strong restrictions on the model architecture to ensure a tractable normalizing constant for likelihood computation, or must rely on surrogate objectives to approximate maximum likelihood training. Implicit generative models, on the other hand, often require adversarial training, which is notoriously unstable and can lead to mode collapse". The normalizing constant, instability and mode collapse are already the main computational issues the sampling from explicit density literature have been dealing with for many years.

Here we introduce another way to represent probability distributions that

- is an iterative sampling process, which does not rely on a good distance between two measures,
- models the score function, rather than the likelihood, avoiding the normalizing constant.

The key idea is to model the gradient of the log probability density function, a quantity often known as the (Stein) score function. Such score-based models are not required to have a tractable normalizing constant, and can be directly learned by score matching.

4.2 Score function, score estimation via score matching

If one tries to use likelihood models to model the target density with an unnormalized density $p_\theta(x) \propto e^{-f_\theta(x)}$, then in order to maximize the likelihood in Eq. (4.1) one needs to evaluate the normalizing constant $Z_\theta := \int e^{-f_\theta(x)}dx$, which is typically a computationally intractable quantity for general $f_\theta(x)$. An alternative is model the score function instead of the density function, in order to avoid the difficulty of evaluating normalizing constants.

Score function. The score function of a measure μ is defined as

$$x \mapsto \nabla \log \mu(x).$$

Even if we only know μ up to a normalizing constant, $\mu \propto e^{-f}$, we can compute the score function without knowing the normalizing constant

$$\nabla \log \mu(x) = \nabla \log e^{-f} = -\nabla f(x).$$

One may find the name "score function" not very intuitive, see its origin on Wikipedia.

Score estimation. To estimate the score function, it is natural to parametrize the score function by a parameter θ via $s_\theta(x)$, and try to minimize its distance to the true score function over data sampled from the target measure:

$$\min_\theta \mathbb{E}_{X\sim\mu} \left\| \nabla \log \mu(X) - s_\theta(X) \right\|_2^2 . \tag{4.2}$$

The above objective is in fact the **Fisher divergence**, between the target measure and the measure induced by the score s_θ, defined as

$$\mathcal{F}(q,p) := \mathbb{E}_{X\sim q} \left\| \nabla \log q(X) - \nabla \log p(X) \right\|_2^2 .$$

$\mathcal{F}$ is not a proper metric. However, it has other desirable properties of a discrepancy measure, i.e., non-negative and equal zero if and only if the two densities are equal q-almost everywhere. In general, it is difficult to evaluate the objective function in Eq. (4.2) when μ is represented via N samples. $\nabla \log \mu(x)$ is difficult to evaluate with only data samples. For this, a family of methods called **score matching** [HD05] is introduced.

Score matching. Score matching [HD05] is a way to transform the objective in Eq. (4.2) so that it can be evaluated via only data samples and the score function estimator $s_\theta(\cdot)$. We can rewrite the objective as follows

$$\begin{aligned}&\mathbb{E}_{X\sim\mu} \left\| \nabla \log \mu(X) - s_\theta(X) \right\|_2^2 \\ =& \underbrace{\mathbb{E} \left\| \nabla \log \mu(X) \right\|_2^2}_{\text{does not depend on } \theta} - 2\mathbb{E} \left\langle s_\theta(X), \nabla \log \mu(X) \right\rangle + \mathbb{E} \left\| s_\theta(X) \right\|_2^2 .\end{aligned}$$

The first term does not depend on θ. We rewrite the second term via integration by parts

$$\begin{aligned}
&-\mathbb{E}\left\langle s_\theta(X), \nabla \log \mu(X)\right\rangle \\
&= -\int \left\langle s_\theta(x), \nabla \log \mu(x)\right\rangle \mu(x)dx \\
&\overset{(i)}{=} \int \nabla \cdot s_\theta(x)\mu(x)dx \\
&= \mathbb{E}\nabla \cdot s_\theta(X),
\end{aligned}$$

where $\nabla\cdot$ is the divergence operator $\nabla \cdot F = \sum_{i=1}^n \frac{\partial F_i}{\partial x_i}$. (i) uses integration by parts where $\nabla \log \mu(x)\mu(x)$ integrates to $\mu(x)$, and the boundary term is 0 as long as μ has fast decay as $\|x\|_2 \to \infty$. Finally, the objective (4.2) becomes

$$\min_\theta \mathbb{E}_{X\sim\mu}\left[\|s_\theta(X)\|_2^2 + 2\nabla \cdot s_\theta(X)\right]. \tag{4.3}$$

The above objective can be evaluated by replacing $\mathbb{E}_{X\sim\mu}$ by the expectation over the empirical measure

$$\min_\theta \frac{1}{N}\sum_{i=1}^N \left[\|s_\theta(x_i)\|_2^2 + 2\nabla \cdot s_\theta(x_i)\right].$$

By doing so, we avoided the knowledge of $\nabla \log \mu(\cdot)$. There is one more undesirable feature of the above formulation: computing the divergence term $\nabla \cdot s_\theta(\cdot)$ might not be efficient for high dimensional data. Denoising score matching is introduced to avoid the computation of $\nabla \cdot s_\theta(\cdot)$.

Denoising score matching. Remark that the divergence term $\nabla \cdot s_\theta(\cdot)$ appears because we used integration by parts to make $\nabla \log \mu$ disappear. It is possible to make the divergence term disappear again via another integration by parts trick. Consider the slightly perturbed measure $\mu_\sigma := \mu * \mathcal{N}(0, \sigma^2\mathbb{I}_n)$, $\sigma > 0$, where $*$ denotes convolution. The score function of the perturbed measure μ_σ can be estimated via the score matching objective in Eq. (4.3). To get rid of the divergence term $\nabla \cdot s_\theta$, observe that

$$\begin{aligned}
&\mathbb{E}_{X\sim\mu_\sigma}\left[\nabla \cdot s_\theta(X)\right] \\
&= \int\int \nabla \cdot s_\theta(x+\sigma z)\mu(x)\gamma(z)dxdz \\
&= \int \left[\int \nabla \cdot s_\theta(x+\sigma z)\gamma(z)dz\right]\mu(x)dx \\
&\overset{(i)}{=} \int \left[\int \left\langle \frac{1}{\sigma}s_\theta(x+\sigma z), z\right\rangle \gamma(z)dz\right]\mu(x)dx \\
&= \mathbb{E}_{X\sim\mu, Z\sim\mathcal{N}(0,\sigma^2\mathbb{I}_n)}\frac{1}{\sigma}\left\langle s_\theta(X+\sigma Z), Z\right\rangle,
\end{aligned}$$

where γ is the density of the standard Gaussian $\gamma(x) = (2\pi)^{-\frac{1}{n}} \exp(-\|x\|_2^2/2)$. (i) applies integration by parts which integrates the divergence term and derives $\gamma(\cdot)$. Then the score matching objective for μ_σ objective becomes

$$\min_\theta \mathbb{E}_{X\sim\mu, Z\sim\mathcal{N}(0,\sigma^2\mathbb{I}_n)} \left\| s_\theta(X+\sigma Z) + \frac{Z}{\sigma} \right\|_2^2 . \tag{4.4}$$

The above objective can easily be replaced by its empirical counterpart, as $X+\sigma Z$ can be obtained by perturbing each data point by adding a bit of Gaussian noise. To explain the objective (4.4) in words, denoising score matching reduces the estimation of the score of μ_σ to the following prediction problem: given the noisy samples $X_i+\sigma Z_i, i = 1, \dots, N$, find s_θ to predict the noise $-\frac{Z_i}{\sigma}$ in small squared error loss.

However, it should be noted that the score of μ_σ is not exactly the score of μ. The score of μ_σ provides a good estimate of μ when the noise level σ is small enough.

4.3 Langevin algorithms with estimated score

Once we have estimated the score function as explained in the previous section, we can replace the negative gradient of the target measure μ in Langevin algorithms with the estimated score. This gives us a way to approximately sample from the target measure μ. Recall the unadjusted Langevin algorithm (ULA) for sampling $\mu \propto e^{-f}$,

$$X^{k+1} = X^k - h\nabla f(X^k) + \sqrt{2h}\xi_k,$$

where $h > 0$ is step-size and ξ_k is i.i.d. standard Gaussian noise. Replacing the $-\nabla f(\cdot)$ with estimated score leads to the following.

Unadjusted Langevin algorithm with estimated score. It iteratively runs

$$X^{k+1} = X^k + hs_\theta(X^k) + \sqrt{2h}\xi_k,$$

where s_θ is an estimate of score function $\log\mu(\cdot)$. As long as the score estimate is good ($s_\theta \approx \log\mu(\cdot)$) and h is small, we expect that the law of μ_K to be close to μ for K large enough.

While it sounds a plausible idea to run unadjusted Langevin algorithm with estimated score for generative modeling, it faces two main challenges which we illustrate via a toy example.

1. Estimated score function is inaccurate in low density regions. Say the target measure in 2 dimension $\mu = \frac{1}{5}\mathcal{N}((-5,-5),\mathbb{I}_2) + \frac{4}{5}\mathcal{N}((5,5),\mathbb{I}_2)$. With 1280 data samples, [SE19] estimated the score function via score matching with a neural network and plotted the estimated score in the Figure 4.1. Unsurprisingly, the score estimation is only accurate around the two modes of μ, and it is not reliable in low density regions. This is mainly because the score estimation objective (4.2) focuses loss on where data points are present.

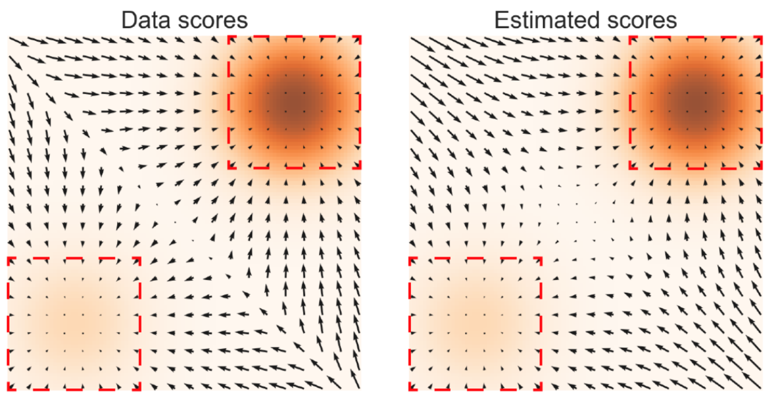

Figure 2: **Left**: $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$; **Right**: $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$. The data density $p_{\text{data}}(\mathbf{x})$ is encoded using an orange colormap: darker color implies higher density. Red rectangles highlight regions where $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) \approx \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$.

Figure 4.1: Figure 2 in [SE19] for score estimation in toy data

2. Langevin algorithms mix slowly when faced a bottleneck. As we have seen in previous lectures, when two modes of the data distribution are separated by low density regions, it creates a bottleneck for the mixing of Langevin algorithm. When a Langevin algorithm is initialized at one mode, it has a hard time going to the other mode. Even if we can initialize at two modes together, it will not be able to correctly recover the relative weights of these two modes in reasonable time in high dimension.

4.4 Annealed Langevin algorithms

The two challenges that Langevin algorithm with estimated score function faces have different nature: the first score estimation in low density region problem is a statistical estimation problem; while the second problem is a computational problem. However, in the toy example, they are both cause by the low density regions of μ. A key observation here is that once we smooth μ by convolving it with large Gaussian noise, $\mu_\sigma :=$

$\mu * \mathcal{N}(0, \sigma^2 \mathbb{I}_n)$, then both problems are gone for the problem of sampling μ_σ, as illustrated in Figure 4.2. However, to sample from μ, it is not enough to say that we can sample from μ_σ with large enough noise σ. [SE19] proposed to improve Langevin algorithms with estimated score by

1. perturbing the data using various levels of noise
2. estimating score and run Langevin algorithms, at all noise levels.

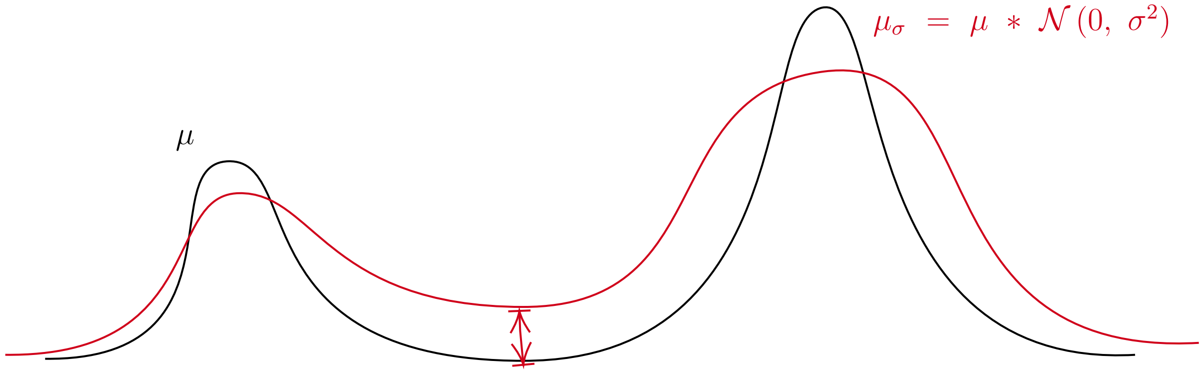

Figure 4.2. Once we convolve μ with a Gaussian, for μ_σ, the density in the highlighted region becomes larger. What was a bottleneck for Langevin algorithms is no longer one.

Inspired by the idea of annealing [KGJV83], the annealed Langevin algorithm is proposed. Set L levels of noise from large to small, $\sigma_1, \ldots, \sigma_L$, and $h_L > 0$

- For i from 1 to L run
  - Set step-size $h_i = h_L \sigma_i^2 / \sigma_L^2$.
  - Run unadjusted Langevin algorithm for K steps

$$X^{k+1} = X^k + h_i s_\theta(X^k, \sigma_i) + \sqrt{2h_i} Z^k,$$

where Z^k is independent standard Gaussian noise, and $s_\theta(X^k, \sigma_i)$ is the estimated score function for $\mu * \mathcal{N}(0, \sigma_i^2 \mathbb{I}_n)$.

Intuitively, for small i (corresponding to large noise), the score estimation is easy everywhere, the Langevin algorithm can go across modes easily and mixes fast for $\mu * \mathcal{N}(0, \sigma_i^2 \mathbb{I}_n)$. As i increases, the target measure $\mu * \mathcal{N}(0, \sigma_i^2 \mathbb{I}_n)$ gets closer to the true target measure μ. For large L and $\sigma_L \approx 0$, we expect to sample $\mu * \mathcal{N}(0, \sigma_L^2 \mathbb{I}_n)$ which is approximately to μ.

4.5 Diffusion models

Does the annealed Langevin algorithm converges to the target measure as our intuition suggest? How to set the noise levels $\sigma_1, \dots, \sigma_L$ in practice?

To answer the above questions, we first study the convergence property of the continuous analogue of the annealed Langevin algorithm by taking the limit $L \to \infty$ and the difference between $\sigma_{j+1} - \sigma_j \to 0$. This leads to the study of diffusion models.

Bibliography

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[HD05] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[KGJV83] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[SE19] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.