Building Diffusion Model's theory from ground up

Diffusion Models, a new generative model family, have taken the world by storm after the seminal paper by Ho et al. [2020]. While diffusion models are often described as a probabilistic Markov Chains, their underlying principle is based on the decade-old theory of Stochastic Differential Equations (SDE), as found out later by Song et al. [2021]. In this article, we will go back and revisit the 'fundamental ingredients' behind the SDE formulation and show how the idea can be 'shaped' to get to the modern form of Score-based Diffusion Models. We'll start from the very definition of the 'score', how it was used in the context of generative modeling, how we achieve the necessary theoretical guarantees and how the critical design choices were made to finally arrive at the more 'principled' framework of Score-based Diffusion. Throughout this article, we provide several intuitive illustrations for ease of understanding.

AUTHORS
Ayan Das

AFFILIATIONS
University of Surrey UK, MediaTek Research UK

PUBLISHED
May 7, 2024

Contents

Introduction

Motivation

Not only generative modeling has been around for decades, few promising model families emerged and dominated the field for several years in the recent past. VAEs [1] dominated the generative modelling landscape from 2014 onwards, until GANs [2] took off in 2015-16; Normalizing Flows (NF) [3] never really made it to the mainstream generative modeling due to its restrictive architectural requirement. However, it is quite clear at this point that the magnitude of impact they made is relatively less than barely 2-3 years of Diffusion Models. It is mostly attributed to one of the seminal papers (by Jonathan Ho et al. [4]), now popularly referred to as “Denoising Diffusion Probabilistic Models” or DDPM. With the exponential explosion of works following DDPM, it is very hard, or rather unnecessary to look beyond this pivotal point.

In this article, we look back into the conceptual and theoretical ideas that were in development for a long time, even outside the field of core machine learning. We will show in a later sections that, some of the theoretical ‘pillars’ holding Diffusion Models, have their roots deep into statistical physics and other fields. A significant part of this theory was presented afresh in the ICLR paper [5] (won best paper award). Lastly, even though the ideas presented in this article are quite theoretical, we made our best attempt to convey them with intuitive explanations, diagrams and figures, thereby expanding its potential audience. To encourage further exploration, we provide all codes used in producing the figures (and experiments) of this article in this repository.

This article notes that, historically, there were two distinct roads of development that merged in order for modern diffusion models to emerge – "scalable estimation of score" and "using the score for generative modelling". The former is relatively short, while the latter traces its origin back to ~1900, if not earlier. This article explores these two paths independently – the latter one first while assuming the knowledge of the former. Rest of this introductory section is spent on defining the general modelling problem and the very notion of 'score' – the primary quantity of interest. The next section deals with how we can use score in generative modelling, assuming access to an oracle for the true score. The last section dives solely into the problem of estimating the score in a scalable manner. It is worth mentioning that, in this article, we explain only the "sufficient and necessary" concepts needed to build the diffusion model framework and hence may not directly resemble the typical formalism seen in most papers.

Generative Modeling

The problem of generative modeling, in most cases, is posed as *parametric density estimation* using a finite set of samples $\{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$ from a "true but unknown" data distribution $\boldsymbol{q_{data}(x)}$. With a suitable model family chosen as $\boldsymbol{p_\theta(x)}$, with unknown parameters $\boldsymbol{\theta}$, the problem boils down to maximizing the average (log-)likelihood (w.r.t $\boldsymbol{\theta}$) of all the samples under the model

$$\theta^* = arg\max_\theta \mathbb{E}_{x\sim q_{data}(x)}\left[\log p_\theta(x)\right] \approx arg\max_\theta \frac{1}{N}\sum_{n=1}^{N}\log p_\theta(x^{(n)})$$

It turned out however, that defining an arbitrary parametric density $\boldsymbol{p_\theta(x)}$ is not as easy as it looks. There was one aspect of $\boldsymbol{p_\theta}$ that is widely considered to be the evil behind this difficulty – *the normalizing constant* that stems from the axiom of probability

$$p_\theta(x) = \frac{\tilde{p}_\theta(x)}{\int_x \tilde{p}_\theta(x)}$$

Existing Frameworks

It was understood quite early on that any promising generative model family must have one property – *ease of sampling*, i.e. generating new data samples. Sampling was so essential to generative modeling, that the model families that followed were all geared towards effective sampling, even if it was at the expense of other not-so-important properties. It was also well understood that there was one common underlying principle most effective for crafting "sampling-centric" generative models – *transforming simple probability densities*. This formed the backbone of every single generative model family so far; be it VAEs, GANs or NFs, their generative process is a density transformation of this form

$$x = f_\theta(z),\ \text{where } z\sim\mathcal{N}(0, I)$$

that suggests to start with a simple density (often just standard normal) followed by a functional transformation $\boldsymbol{f_\theta}$, typically a neural network with parameters $\boldsymbol{\theta}$. For VAEs, the function $\boldsymbol{f_\theta}$ is the decoder; for GANs, it's the generator network and for NFs, it's the entire flow model. It is to be noted however, that the way they differ is mostly *how they are trained*, which may involve more parametric functions (e.g. VAE's encoder or GAN's discriminator) and additional machinery. This way of building generative models turned out to be an effective way of sidestepping the notorious normalizing constant.

Diffusion is no different

Diffusion Models, at its core, follow the exact same principle, but with a slightly clever design. For diffusion models, the transformation $\boldsymbol{f_\theta}$ is rather complicated. It is a sequence of invocations of a neural function (denoted as $\boldsymbol{s_\theta}$) along with some additional computation (denoted as $\boldsymbol{g(\cdot)}$)

$$x = g_1(g_2(g_3(\cdots z\cdots, s_\theta), s_\theta), s_\theta),\ \text{where } z\sim\mathcal{N}(0, I) \tag{1}$$

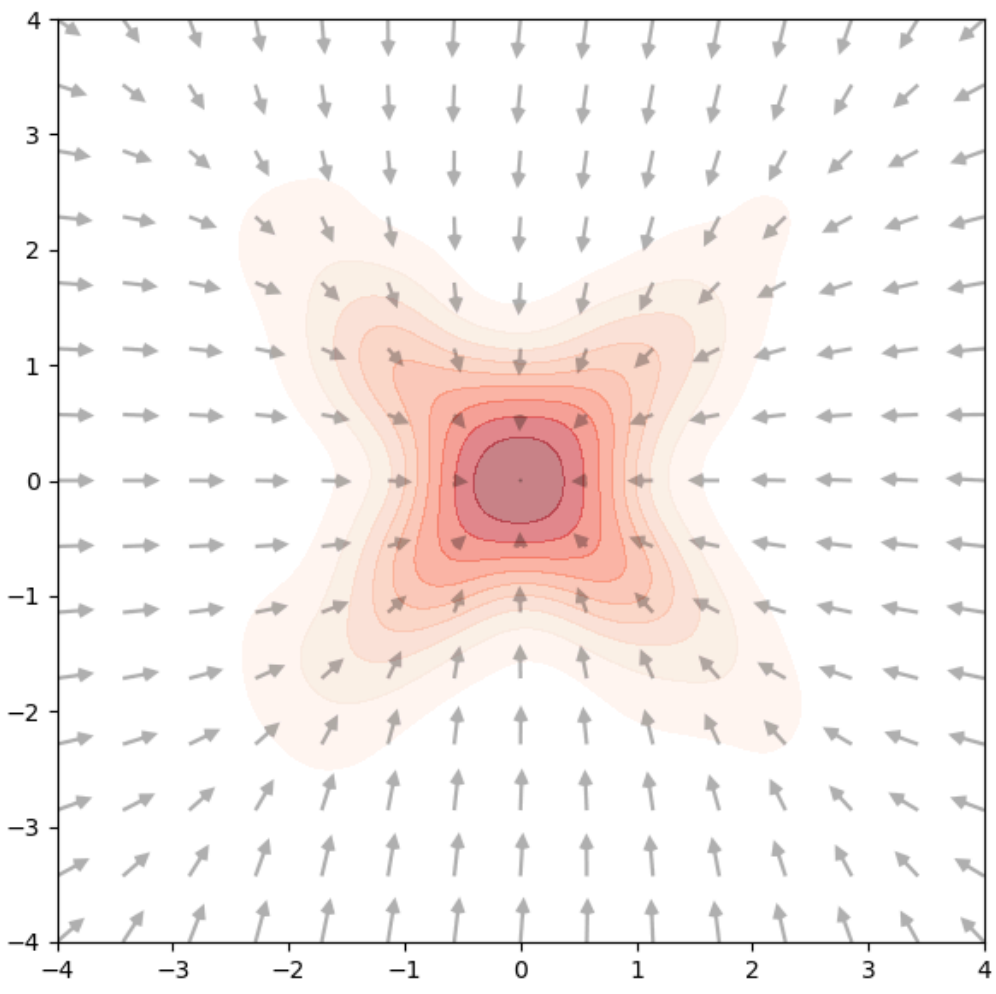
This is a big difference between Diffusion Models and other generative model families. Prior generative families tried to learn the exact transformation directly via one parametric neural function $\boldsymbol{f_\theta}$. Diffusion Models on the other hand, try to learn $\boldsymbol{s_\theta}$, a quantity very *fundamental and intrinsic* to any true data distribution $\boldsymbol{q_{data}(x)}$. The quantity in question has historically been called the "*Score*".

The 'Score'

The term 'Score' is simply defined as the *gradient of the log-density of a distribution*, i.e. $\nabla\log p(\cdot)$. In statistics, it is also known (but not very popular) as the 'Informant'. One might argue that 'Score' is rather a strange name for such a quantity. It so happened that the origin of this term can be traced [1] to a 1935 paper [6] by Ronald Fisher, where he used the term in a very generic sense in order to "rank" some quantities. In the context of diffusion models however, we stick to the modern definition of score. The *true score* of our data distribution is therefore defined as the gradient of the log of *true density* of data, w.r.t the data variable

$$\nabla_x \log q_{data}(x) \triangleq s(x) \tag{2}$$

The quantity in Eq.(2) is unknown, just like the true data density $\boldsymbol{q_{data}(x)}$. It does have a meaning though: the "*true score*" refers to the *direction of steepest increase* in log-likelihood at any given point in the data space. See the gray arrows in the figure below.



Simply, at a point x, it tell us the best direction to step into (with little step-size δ) if we would like to see a point x' with slightly higher likelihood

$$x' = x + \delta \cdot \nabla_x \log q_{data}(x)|_{x=x} \tag{3}$$

Please note that this stems just from the definition of the gradient operator ∇ in score. If you are familiar with gradient descent, you may find conceptual resemblance.

Now, there are two burning questions here:

1. Considering we have access to the true score, is Eq.(3) enough to define a generative process with appropriate convergence guarantee ?
2. How do we actually get the true score ?

The following two sections answer these questions respectively. Luckily, as we now understand that these two questions are somewhat decoupled, that they can be studied independently. The first section analyzes the first question, *assuming* we have access to the true score $\nabla_x \log q_{data}(x)$. The second section explores how to get the true score, or rather, an approximation of it.

Generative Modeling with Scores

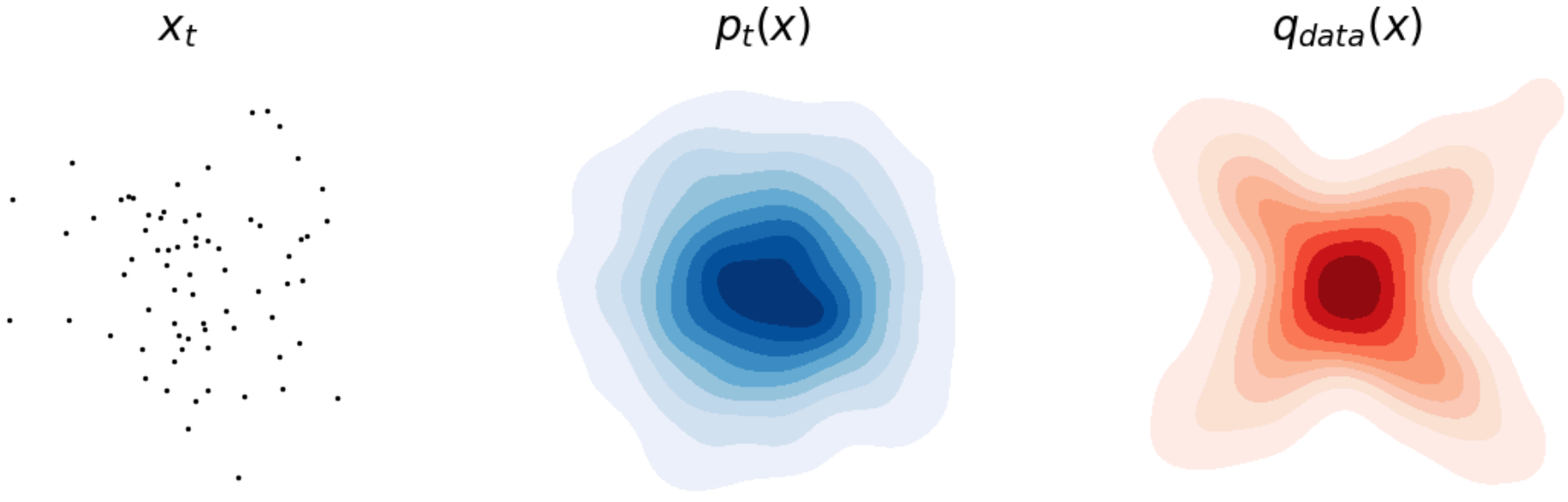
As explained before, we would like to sample from the true data distribution $q_{data}(x)$ but all we have access to (we assume) is its score $s(x)$ as defined in Eq.(2). One may define a naive generative process as the iterative application of Eq.(3). Intuitively, it is very similar to gradient descent, where we greedily climb the log-density surface to attain a local maxima. If so, we can already see a possible instance of the general structure of Diffusion's generative process as hinted in Eq.(1), with $g(\cdot)$ being

$$g(z, s(\cdot)) = z + \delta \cdot s(z) = z + \delta \cdot \nabla_x \log q_{data}(x)$$

With a little reshuffling of Eq.(3) and considering $\delta \to 0$, one can immediately reveal the underlying ODE [2] that describes the infinitesimal change

$$dx = \nabla_x \log q_{data}(x) dt \tag{4}$$

BUT, please note that this is only an intuitive attempt and is entirely based on the definition of score. It possesses **absolutely no guarantee** that this process can converge to samples from the true data distribution. In fact, this process is **greedy**, i.e. it only seeks to go uphill, converging exactly at the *modes* [3] . You can see the below figure that shows the samples x subjected to the process in Eq.(4) and its density $p_t(x)$ evolving over time. The density in red is the target density whose score (we assume we know it) is being used.



In this case, at $t = \infty$, all samples will converge to the state with *the highest* likelihood (i.e. exactly a the center). This isn't really desirable as it doesn't "explore" at all. Just like any other sampling algorithm, we need noise injection !

Langevin Equation and Brownian Motion

Turned out that this problem was explored long ago [7] in molecular dynamics by french physicist Paul Langevin in the context of analyzing movements of particles suspended in a fluid. He described the overall dynamics of particles, i.e how the position of the particle changes over time t when in a *potential energy* field $U(x)$

$$dx = -\nabla_x U(x)dt + \sqrt{2}dB_t \tag{5}$$

The term dB_t is called "Brownian Motion" and is effectively the source of noise – we will talk about this later in this subsection. Energy is considered "bad", i.e. particles do not want to stay in a state with high energy. So they try to go downhill and settle in low-energy states using the gradient of the energy surface. The langevin equation (i.e. Eq.(5)) happened to provide sufficient "exploration" abilities so that the particles visit states with probability $\propto e^{-U(x)}$. This suggests that we can treat "negative energy" as log-likelihood

$$q_{data}(x) \propto e^{-U(x)} \implies \log q_{data}(x) = -U(x) + C \implies \nabla_x \log q_{data}(x) = -\nabla_x U(x)$$

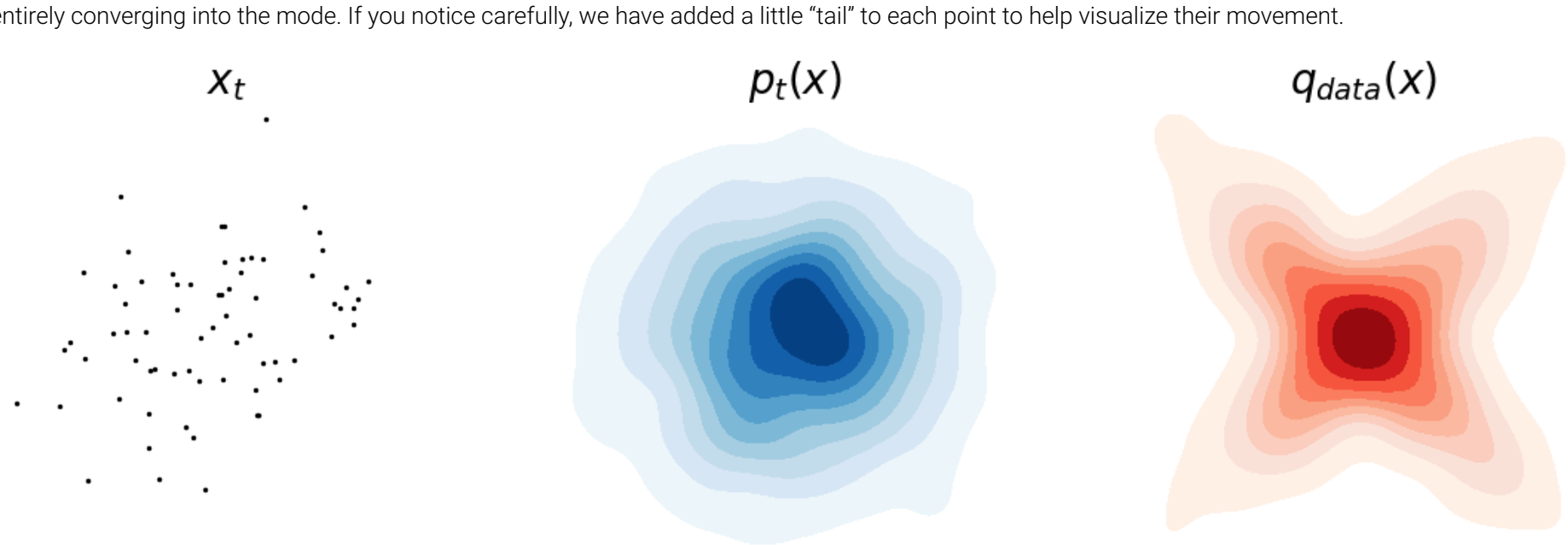
By using the above substitution into the langevin equation, we can move out of physics and continue with out ML perspective

$$dx = \nabla_x \log q_{data}(x)dt + \sqrt{2}dB_t \tag{6}$$

Note that this isn't very different from our "intuitive" and greedy process in Eq.(4), except for the noise term dB_t and a strange $\sqrt{2}$. But this makes a difference! The brownian motion is an old construct from particle physics to describe random motion of particles in fluid/gas. It is simply a gaussian noise with infinitesimally small variance [4]

$$dB_t = \mathcal{N}(0, dt) \implies dB_t = \sqrt{dt} \cdot z, \text{ where } z \sim \mathcal{N}(0, I)$$

With that, we can simulate our new langevin equation *with noise* (i.e. Eq.(6)) just like the noiseless case. You can see now that the noise is keeping the process from entirely converging into the mode. If you notice carefully, we have added a little "tail" to each point to help visualize their movement.



Fokker-Planck Equation

The simulation is convincing; but it'd be even better if we can *theoretically verify* that the process in Eq.(6) indeed converges to $q_{data}(x)$. The key to this proof is figuring out $p_t(x)$ and making sure that it stabilizes as $t \to \infty$, i.e. $p_\infty(x) = q_{data}(x)$. It turned out that a stochastic process of the form $dx = \mu_t(x)dt + \sigma_t(x)dB_t$, acting on a random variable x, induces a time-varying distribution that can be described by this ODE

$$\frac{\partial}{\partial t}p_t(x) = -\frac{\partial}{\partial x}\Big[p_t(x)\mu_t(x)\Big] + \frac{1}{2}\frac{\partial^2}{\partial x^2}\Big[p_t(x)\sigma_t^2(x)\Big] \tag{7}$$

This is a well celebrated result know as the "Fokker-Planck equation" that even predates the Langevin Equation. So, the solution of this ODE is exactly what we are seeing in the above figure (middle). One can easily verify the convergence of Eq.(6) by first observing $\mu_t(x) = \nabla_x \log q_{data}(x), \sigma_t(x) = \sqrt{2}$ and then using $\frac{\partial}{\partial t}p_\infty(x) = \frac{\partial}{\partial t}q_{data}(x) = 0$.

$$\begin{aligned}
\frac{\partial}{\partial t}p_\infty(x) =& -\frac{\partial}{\partial x}\Big[p_\infty(x)\nabla_x \log q_{data}(x)\Big] + \frac{(\sqrt{2})^2}{2}\frac{\partial^2}{\partial x^2}\Big[p_\infty(x)\Big] \\
\frac{\partial}{\partial t}q_{data}(x) =& -\frac{\partial}{\partial x}\Big[q_{data}(x)\nabla_x \log q_{data}(x)\Big] + \frac{(\sqrt{2})^2}{2}\frac{\partial^2}{\partial x^2}\Big[q_{data}(x)\Big] \\
0 \text{ (LHS)} =& -\frac{\partial}{\partial x}\Big[\nabla_x q_{data}(x)\Big] + \frac{\partial}{\partial x}\Big[\nabla_x q_{data}(x)\Big] = 0 \text{ (RHS)}
\end{aligned}$$

The LHS holds due to the fact that after a long time (i.e. $t = \infty$) the distribution stabilizes [5]. Please also note that the proof above is for the 1 dimensional case and included for illustrative purpose only – the general case is slightly more complicated.

So, we're all good. Eq.(6) is a provable way of sampling given we have access to the true score. In fact, the very work [8] (by Song et al.) that immediately precedes DDPM, used exactly Eq.(6) in its discrete form

$$x_{t+\delta} = x_t + \delta \cdot \nabla_x \log q_{data}(x) + \sqrt{2\delta} \cdot z \tag{8}$$

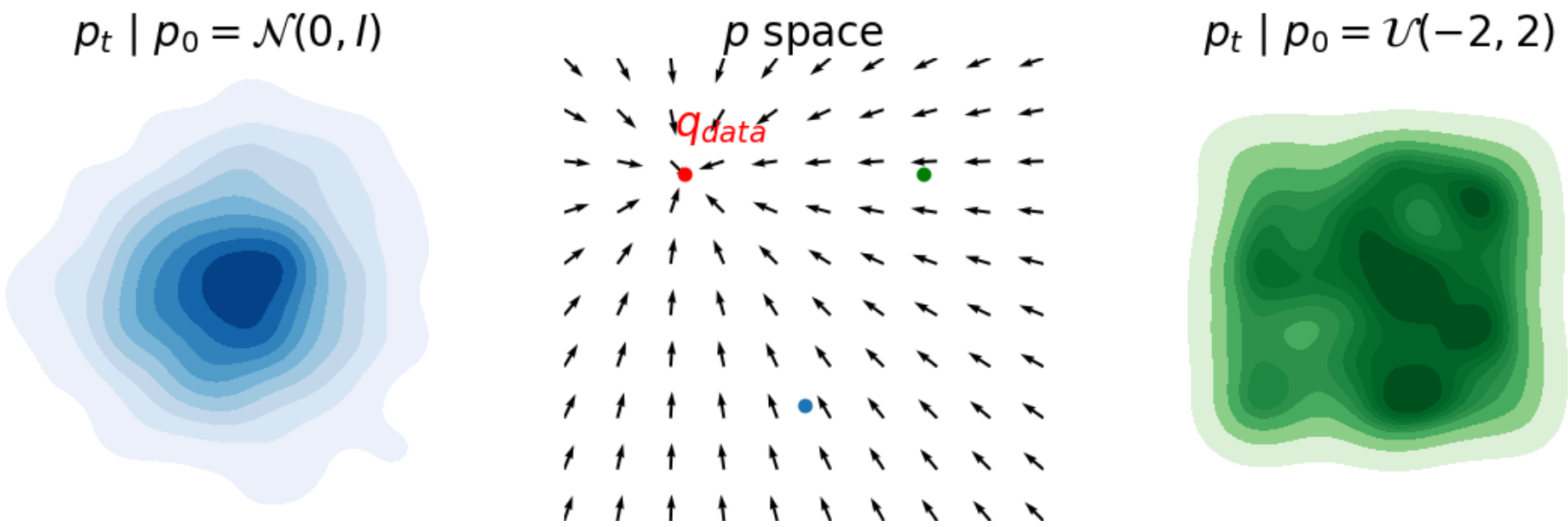
where δ (a small constant) is used as a practical proxy for the theoretical dt.

If you are already familiar with Diffusion Models, specifically their reverse process, you might be scratching your head. That is because, the generative process in Eq. (6) isn't quite same as what modern diffusion models do. We need to cross a few more hurdles before we get there.

A probability path

More than just a proof, the Fokker-Planck ODE provides us with a key insight – i.e. gradually transforming one distribution into another is equivalent to traveling (over time) on a "path" in the *space of probability distributions*. Imagine a space of all possible probability distributions p [6]. The Fokker-Planck ODE for Eq.(6), therefore, represents a specific dynamics on this probability space whose solution trajectory p_t ends at q_{data} at $t = \infty$.

Speaking of ODEs, there is something we haven't talked about yet – the initial distribution at $t = 0$, i.e. p_0. In the simulation above, I quietly used a standard normal $\mathcal{N}(0, I)$ as starting distribution [7] without ever discussing it. Turns out that our Fokker-Planck ODE does not have any specific requirement for p_0, i.e. it always converges to $p_\infty = q_{data}$ no matter where you start. Here's an illustration that shows two different starting distributions p_0 and both of their "paths" over time, i.e. p_t in probability space ultimately converges to q_{data}.



So theoretically, given the score function $\nabla_x \log q_{data}(x)$ of a target distribution $q_{data}(x)$, one can "travel to" it from *any* distribution. However, keeping in mind our need for *sampling*, it's best to choose an initial distribution that is sampling-friendly. Strictly speaking, there are couple of reasonable choices, but the diffusion model community ended up with the *Isotropic Gaussian* (i.e. $\mathcal{N}(0, I)$). This is not only due to its goodwill across machine learning and statistics, but also the fact that in the context of SDEs with Brownian motions [8], Gaussians arise quite naturally.

Estimating the "score" is hard

So far what we've talked about, is just the *generative process* or as diffusion model literature calls it, the "reverse process". But we haven't really talked about the "forward process" yet, in case you are familiar with it. The forward process, in simple terms, is an *ahead-of-time description* of the "probability path" that reverse process intends to take. But the question is, why do we need to know the path ahead of time – the reverse process seems quite spontaneous [9], no ? Sadly, it can't be answered with theory alone.

The problem lies in Eq.(6) – let's write it again with a little more verbosity

$$dx_t = \nabla_x \log q_{data}(x)|_{x=x_t}\, dt + \sqrt{2} dB_t \tag{9}$$

Even though we wished to estimate $\nabla_x \log q_{data}(x)|_{x=x_t}$ with neural network $s_\theta(x = x_t)$, this turned out to be **extremely hard** in practice [8] . It was understood that one neural network is not enough to capture the richness of the score function at all values of x. There were two options before the us – one, make the neural network expressive enough, or second, learn the network **only where it's needed**. The community settled on the second one because it was easier to solve.

So, what some of the pioneering works did, is first fixing a path [10] and then learning the score only *on that path*. It's all about specializing the neural network $s_\theta(x_t, t)$ over $t \in [0, \infty]$. The neural score estimator is capable of producing the right score if we provide the time t, which we can of course. We will see in the next section that, to learn a score of any distribution, we need samples from it. This begs the question: how do we get samples x_t (for all t) for training purpose ? It certainly can't be with Eq.(6) since it requires the score. The answer is, we need to run this process in the other way – this is what Diffusion Models call the "Forward Process".

The "forward process"

Going *the other way* requires us to run a simulation to go from $q_{data}(x)$ at $t = 0$ to $t = \infty$, just the opposite of the animation above. Recall that we already saw how to do this. To go to any distribution at $t = \infty$, all you need is its score and the langevin equation. So how about we start from $q_0 = q_{data}(x)$ this time [11] and run the langevin simulation again with a *known* end target $q_\infty = \mathcal{N}(0, I)$?

$$\begin{aligned} dx &= \nabla_x \log \mathcal{N}(0, I) dt + \sqrt{2} dB_t && (10)\\ &= -x dt + \sqrt{2dt} z && (11) \end{aligned}$$

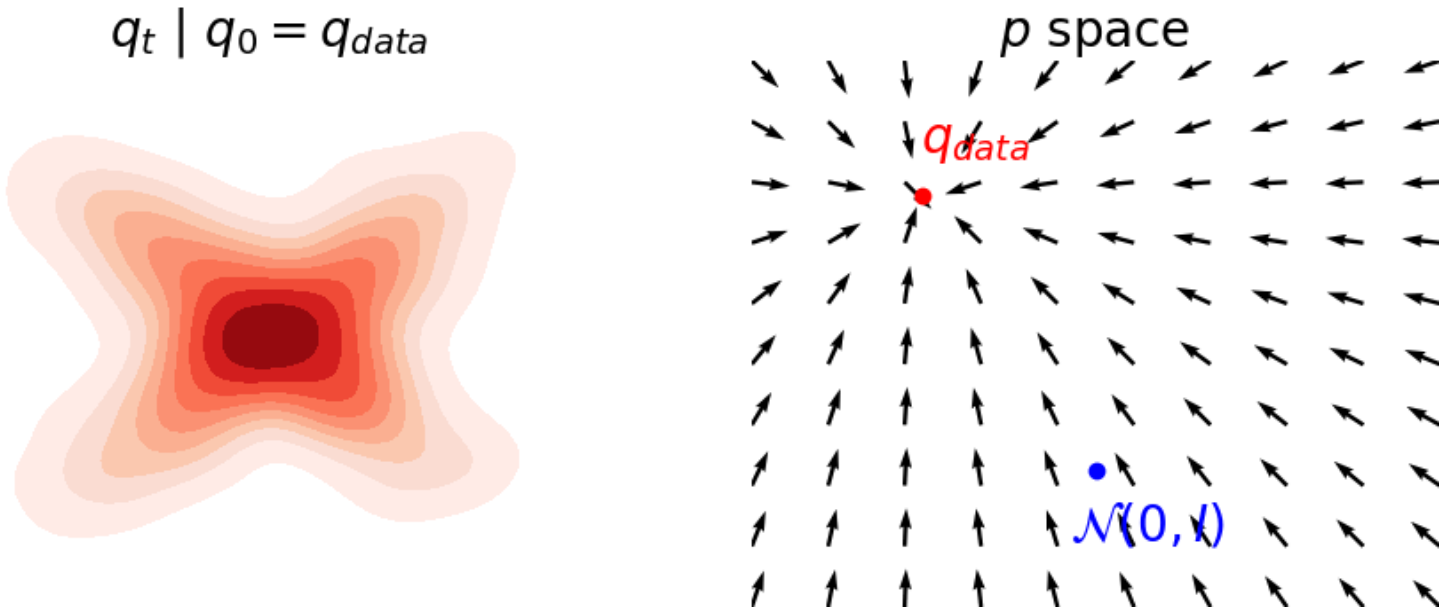
It is interesting to note that due to the target distribution being known in its closed form, we do not see any awkward scores dangling around. The score of $\mathcal{N}(0, I)$ is simply $-x$ [12] . The discretized version of Eq.(11), i.e.

$$\begin{aligned} x_{t+dt} &= x_t - x_t \cdot dt + \sqrt{2dt}\, z \\ &= (1 - dt)x_t + \sqrt{2dt}\, z \end{aligned}$$

.. may resemble DDPM's [4] forward process [13] .

> *NOTE: A little subtlety here that we only fixed the end point of the forward process, but not the exact path. It seems that running the langevin equation in the forward direction chose one path on its own. Turns out that this is the "isotropic path" where all dimensions of the variable x evolves in time the exact same way. Some works [9] [10] recently uncovered non-isotropic diffusion, where it is indeed possible to travel on other paths. But this is outside the scope of this article.*

We can simulate the above equation just like we did in the reverse process, in order to get samples $x_t \sim q_t$. Below we show simulation of the forward process



While it is true that the reverse process in inherently sequential due to the arbitrary nature of the score, the forward process (in Eq.(11)) is entirely known and hence can be exploited for easing the sequentiality. We can see a way out if we try to simplify [14] the expression for x_{t+2dt} using x_{t+dt}

$$\begin{aligned} x_{t+2dt} &= (1 - dt)x_{t+dt} + \sqrt{2dt}\, z_2 \\ &= (1 - dt)\Big[(1 - dt)x_t + \sqrt{2dt}\, z_1\Big] + \sqrt{2dt}\, z_2 \\ &= (1 - 2dt)x_t + \sqrt{2dt(1 - dt)^2 + 2dt}\, z_{12} \\ &= (1 - 2 \cdot dt)x_t + \sqrt{2 \cdot 2dt}\, z_{12} \\ \implies x_{t+2dt} &\sim \mathcal{N}((1 - 2 \cdot dt)x_t, 2 \cdot 2dt I) \end{aligned}$$

The above simplification suggests that we can jump to any time t, without going through the entire sequence, in order to sample $x_t \sim q_t$. In fact, $q_t(x_t|x_0)$ is gaussian ! This result opens up an interesting interpretation – generating $x_0 \sim q(x_0|x_t)$ can be interpreted as solving a "gaussian inverse problems", which we explore in a later section.

All good for now, but there is one more thing we need to deal with.

Finite time & the "schedule"

What we discussed so far, i.e. the forward and reverse process, require infinite time to reach its end state. This is a direct consequence of using the langevin equation. That, of course, is unacceptable in practice. But it so happened that there exists quite an elegant fix, which is well known to mathematics – we simply *re-define what time means*. We may choose a re-parameterization of time as, for example, $t' = \mathcal{T}(t) = 1 - e^{-t} \in [0, 1]$ [15]. Plugging $dt = \mathcal{T}'(t)^{-1}dt' = e^t dt'$ [16] into the forward equation brings us even closer to DDPM's forward process

$$x_{t'+dt'} = (1 - e^t dt')x_t + \sqrt{2e^t dt'}\, z$$

This suggests that in the world where time runs from $t' = 0 \to 1$, we need to *escalate* the forward process by replacing dt with $e^t dt'$. The quantity $\mathcal{T}'(t)^{-1}dt' = e^t dt'$ is analogous to what diffusion models [4] [11] call a "schedule". Recall that DDPM uses a small but increasing [17] "schedule" β_t.

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) \coloneqq \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

Of course, our choice of the exact value of end time (i.e. $t' = 1$) and the re-parameterization $\mathcal{T}$ are somewhat arbitrary. Different choices of $\mathcal{T}$, and consequently $\mathcal{T}'(t)^{-1}dt'$ lead to different schedules (e.g. linear, cosine etc.).

> *NOTE: Choosing a different schedule does not mean the process takes a different path on the probability space, it simply changes its speed of movement over time towards the end state.*

SUMMARY

To summarize, in this section, we started with the definition of 'score' and arrived at a stochastic process (thanks to an old result by Langevin) that, at infinite time, converges to the density associated with the score. We saw that this process is provably correct and can be interpreted as a "path" on the probability space. We argued that due to the difficulty of score estimation everywhere along the path, we need samples at the intermediate time t in order to specialize the score estimates. To do that, we had to travel backwards on the path, which can be done in closed form. We also saw how this process, even though theoretically takes infinite time, can be shrunk down to a finite interval, opening up a design choice known as "schedules".

Estimating the Score

The last chapter, while explaining the "sampling" part of score-based diffusion models, assumed that we have access to the true score $\nabla_x \log q_{data}(x)$ via some oracle. That is, of course, untrue in practice. In fact, accessing the true score for any arbitrary distribution is just not possible [18]. So the way forward, as mentioned before, is to estimate/learn it with a parametric neural network $s_\theta(x)$. Recall however, that all we have access to is samples from $q_{data}(x)$.

If curious enough, one may question how realistic it is to estimate the score $\nabla_x \log q_{data}(x)$, while we can NOT usually estimate the density $q_{data}(x)$ itself ? After all, it is a quantity derived from the density ! The answer becomes clear once you make the *normalization constant* explicit

$$\begin{aligned}\nabla_x \log q_{data}(x) &= \nabla_x \log \frac{\tilde{q}_{data}(x)}{\int_x \tilde{q}_{data}(x)dx} \\ &= \nabla_x \log \tilde{q}_{data}(x) - \nabla_x \log \int_x \tilde{q}_{data}(x)dx \\ &= \nabla_x \log \tilde{q}_{data}(x)\end{aligned}$$

The part in red is zero due to not having dependence on x. So, the score, very cleverly **sidesteps the normalization constant**. This is the reason score estimation gained momentum in the research community.

Implicit Score Matching

The first notable attempt of this problem was by Aapo Hyvärinen [12] back in 2005. His idea was simply to start from a loss function that, when minimized, leads to an estimator of the true score

$$J(\theta) = \frac{1}{2}\mathbb{E}_{x\sim q_{data}(x)}\Big[||s_\theta(x) - \nabla_x \log q_{data}(x)||^2\Big] \tag{12}$$

It is simply an L_2 loss between a parametric model and the true score, weighted by the probability of individual states (hence the expectation). But of course, it is not computable in this form as it contains the true score. Hyvärinen's contribution was to simply show that, theoretically, the minimization problem is equivalent when the loss function is

$$J_{\mathrm{I}}(\theta) = \mathbb{E}_{x\sim q_{data}(x)}\Big[\mathrm{Tr}(\nabla_x s_\theta(x)) + \frac{1}{2}||s_\theta(x)||^2\Big] \tag{13}$$

In the literature, this is known as the "*Implicit Score Matching*". The derivation is relatively simple and only involves algebraic manipulations – please see Appendix A of [12] . The remarkable nature of this result stems from the fact that J_{I} no longer contains the true score. The only dependency on q_{data} is via the expectation, which can be approximated by sample average over our dataset.

But the key challenge with Implicit Score Matching was the $\mathbf{Tr}(\nabla_x \boldsymbol{s}_\theta(\boldsymbol{x}))$ term, i.e. the trace of the hessian of the neural score model, which is costly to compute. This prompted several follow-up works for the race towards scalable score matching, one of which (namely De-noising score matching) is used in Diffusion Models till this day.

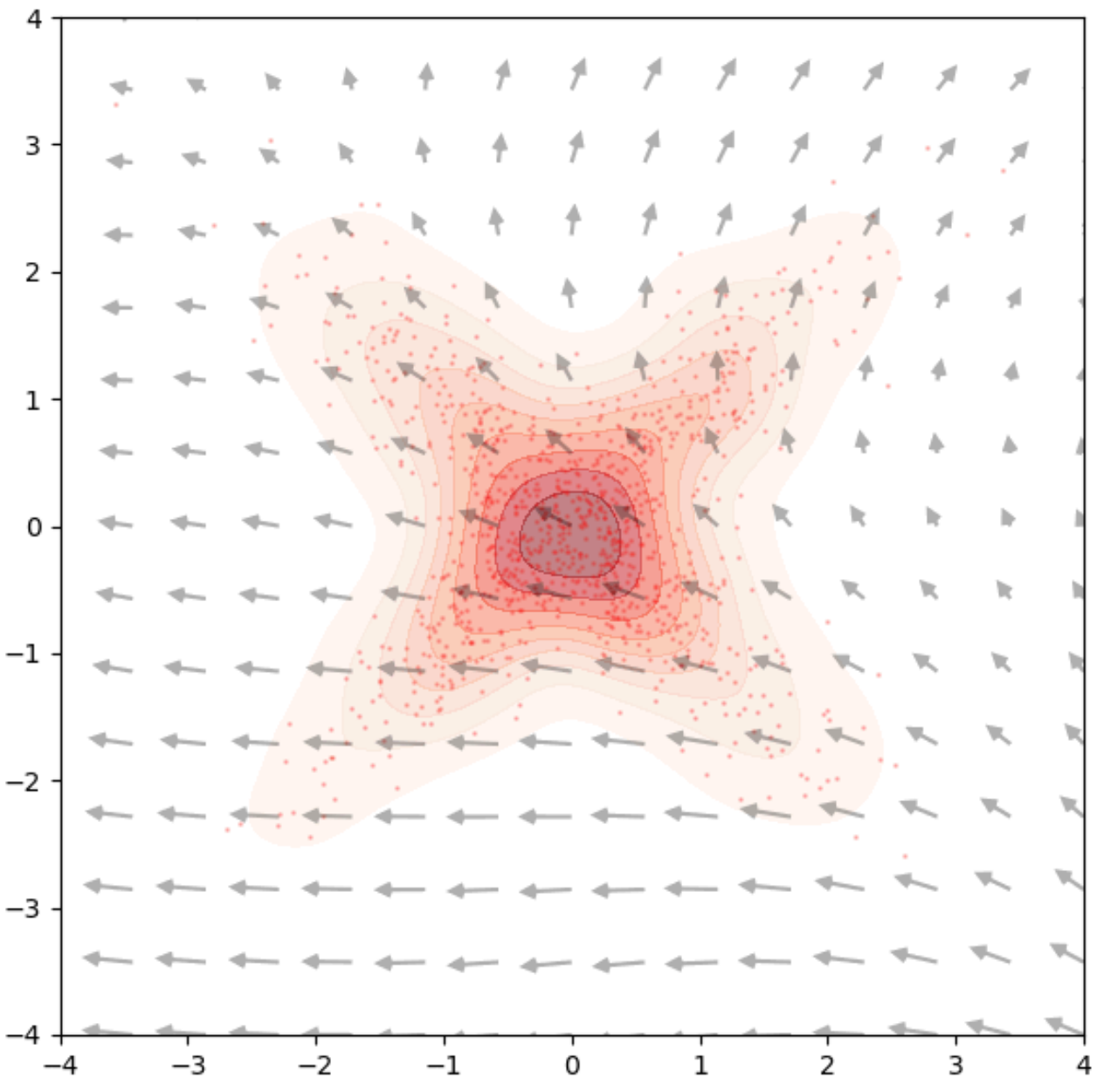
For the sake of completeness, I would like to mention the work of Yang Song et al. [13] around 2019, that proposed an engineering trick to alleviate the hessian computation. They simply used the "Hutchinson Trace estimator" [19] to replace the $\mathbf{Tr}(\cdot)$ in Eq.$(\mathbf{13})$, which eased the computation a bit. This approach however, did not end up being used in practice.

Denoising Score Matching

The most valuable contribution came from Vincent Pascal in 2011, when he showed [14] that the score matching problem has yet another equivalent objective, which was called "Denoising" score matching

$$J_{\mathrm{D}}(\theta) = \mathbb{E}_{x\sim q_{data}(x),\epsilon\sim\mathcal{N}(0,I)}\left[\frac{1}{2}\left\| \boldsymbol{s}_\theta(\underbrace{x+\sigma\epsilon}_{\tilde{x}}) - (-\frac{\epsilon}{\sigma})\right\|^2\right] \tag{14}$$

We deliberately wrote it in a way that exposes its widely accepted interpretation. Denoising score matching simply adds some *known* noise $\boldsymbol{\sigma\epsilon}$ to the datapoints $\boldsymbol{x}$ and learns (in mean squeared sense), from the "noisy" point $\tilde{\boldsymbol{x}}$, the direction of comeback, i.e. $(-\boldsymbol{\epsilon})$, scaled by $\frac{1}{\sigma}$. In a way, it acts like a "de-noiser", hence the name. It is theoretically guaranteed [14] that J_{D} leads to an unbiased estimate of the true score. Below we show a visualization of the score estimate as it learns from data.



A little algebraic manipulation of Eq.$(\mathbf{14})$, demonstrated by Ho et al. [4], leads to an equivalent form which turned out to be training friendly.

$$J_{\mathrm{D}}(\theta) = \mathbb{E}_{x\sim q_{data}(x),\epsilon\sim\mathcal{N}(0,I)}\left[\frac{1}{2\sigma^2}\left\| -\sigma\boldsymbol{s}_\theta(\tilde{x}) - \epsilon\right\|^2\right] \tag{15}$$

$$= \mathbb{E}_{x\sim q_{data}(x),\epsilon\sim\mathcal{N}(0,I)}\left[\frac{1}{2\sigma^2}\left\| \boldsymbol{\epsilon}_\theta(\tilde{x}) - \epsilon\right\|^2\right] \tag{16}$$

We simply change the *interpretation* of what the network learns. In this form, the "noise estimator" network learns *just* the original pure gaussian noise vector $\boldsymbol{\epsilon}$ that was added while crafting the noisy sample. So, from a noisy sample, the network $\boldsymbol{\epsilon_\theta}$ learns roughly an unit variance direction that points towards the clean sample.

There is yet another re-interpretation of Eq.(14) that leads to a slightly different perspective

$$J_{\mathrm{D}}(\theta) = \mathbb{E}_{x\sim q_{data}(x),\epsilon\sim\mathcal{N}(0,I)}\left[\frac{1}{2\sigma^4}\left|\left|\tilde{x} + \sigma^2 s_\theta(\tilde{x}) - (\underbrace{\tilde{x} - \sigma\epsilon}_{x})\right|\right|^2\right] \tag{17}$$

$$= \mathbb{E}_{x\sim q_{data}(x),\epsilon\sim\mathcal{N}(0,I)}\left[\frac{1}{2\sigma^4}||x_\theta(\tilde{x}) - x||^2\right] \tag{18}$$

Eq.(18) shows, that instead of the noise direction towards clean sample, we can also have the clean sample directly as a learning target. This is like doing "denoising" in its true sense. We will get back to this in the next subsection.

Probing the learning objective

If you are still puzzled about how Eq.(16) is related to learning the score, there is a way to probe exactly what the network is learning at an arbitrary input point $\tilde{\boldsymbol{x}}$. We note that the clean sample $\boldsymbol{x}$ and the noisy sample $\tilde{\boldsymbol{x}}$ come from a joint distribution that factorizes

$$q(x, \tilde{x}) = q(\tilde{x}|x)q_{data}(x) = \mathcal{N}(\tilde{x}; x, \sigma I)q_{data}(x).$$

We then factorize this joint in a slightly different way, i.e.

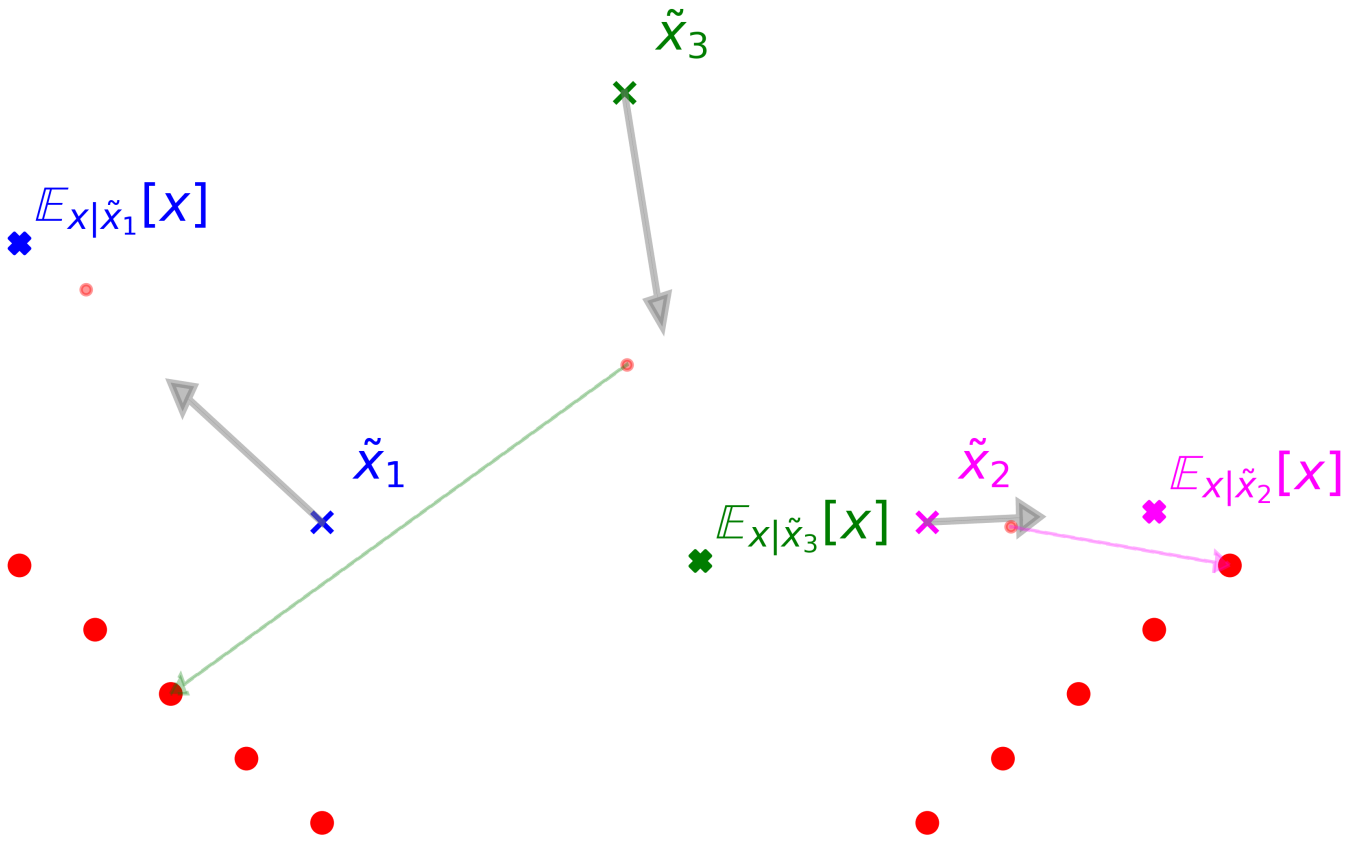
$$q(x, \tilde{x}) = q(x|\tilde{x})q(\tilde{x})$$

where $q(\boldsymbol{x}|\tilde{\boldsymbol{x}})$ can be thought of as a distribution of all clean samples which could've led to the given $\tilde{\boldsymbol{x}}$. Eq.(16) can therefore be written as

$$\begin{aligned}
J_{\mathrm{D}}(\theta) &= \mathbb{E}_{(x,\tilde{x})\sim q(x,\tilde{x})}\left[\frac{1}{2\sigma^2}||\epsilon_\theta(\tilde{x}) - \epsilon||^2\right] \\
&= \mathbb{E}_{\tilde{x}\sim q(\tilde{x}),x\sim q(x|\tilde{x})}\left[\frac{1}{2\sigma^2}\left|\left|\epsilon_\theta(\tilde{x}) - \frac{\tilde{x} - x}{\sigma}\right|\right|^2\right] \\
&= \mathbb{E}_{\tilde{x}\sim q(\tilde{x})}\left[\frac{1}{2\sigma^2}\left|\left|\epsilon_\theta(\tilde{x}) - \frac{\tilde{x} - \mathbb{E}_{x\sim q(x|\tilde{x})}[x]}{\sigma}\right|\right|^2\right]
\end{aligned}$$

In the last step, the expectation $\mathbb{E}_{q(x|\tilde{x})}\left[\cdot\right]$ was pushed inside, up until the only quantity that involves $\boldsymbol{x}$. Looking at it, you may realize that the network $\boldsymbol{\epsilon_\theta}$, given an input $\tilde{\boldsymbol{x}}$, learns the *average noise direction* that leads to the given input point $\tilde{\boldsymbol{x}}$. It also exposes the quantity $\mathbb{E}_{x\sim q(x|\tilde{x})}[\boldsymbol{x}]$, which is the *average clean sample* that led to the given $\tilde{\boldsymbol{x}}$.

Below we visualize this process with a toy example, followed by a short explanation.



Explanation: We have 10 data points $x \sim q_{data}(x)$ in two clusters (big red dots) and we run the learning process by generating noisy samples $\tilde{x} \sim q(\tilde{x})$ (small red dots). Instead of learning a neural mapping over the entire space, we learn a tabular map with only three chosen input points $\tilde{x}_1, \tilde{x}_2, \tilde{x}_3$ (blue, magenta and green cross). Every time we sample one of those [20] three chosen input points, we note which input data point it came from (shown by connecting a dotted line of same color) and maintain a running average (bold cross of same color) of them, i.e. which is nothing but $\mathbb{E}_{x \sim q(x|\tilde{x})}[x]$. We also show the average noise direction at each $\tilde{x}$, i.e. $\frac{\tilde{x} - \mathbb{E}_{x \sim q(x|\tilde{x})}[x]}{\sigma}$, with gray arrows. The gray arrows, as the training progresses, start to resemble the score estimate of the data.

Denoising as inverse problem

A similar treatment, when applied on Eq.(18), yields the following

$$
\begin{aligned}
J_{\mathrm{D}}(\theta) &= \mathbb{E}_{(x,\tilde{x}) \sim q(x,\tilde{x})} \left[\frac{1}{2\sigma^4} ||x_\theta(\tilde{x}) - x||^2 \right] \\
&= \mathbb{E}_{\tilde{x} \sim q(\tilde{x})} \left[\frac{1}{2\sigma^4} \left\| \tilde{x} + \sigma^2 s_\theta(\tilde{x}) - \mathbb{E}_{x \sim q(x|\tilde{x})}[x] \right\|^2 \right]
\end{aligned}
$$

Notice that I brought back the original form of $x_\theta(\cdot)$ that involves the score. If we had the true score instead of an learned estimate, we would have

$$
\mathbb{E}_{x \sim q(x|\tilde{x})}[x] = \tilde{x} + \sigma^2 \nabla_{\tilde{x}} \log p(\tilde{x})
$$

In “Inverse problem” and Bayesian literature, this is a very well celebrated result named “*Tweedie's Formula*”, first published by Robbins [15] but credited to statistician Maurice Tweedie. This theorem is applied in the context of bayesian posterior estimation of a “true” quantity x which we only observe through a (gaussian) noisy measurement $\tilde{x}$. Tweedie's formula tells us that the *posterior mean* of the inverse problem $q(x|\tilde{x})$ can be computed without ever knowing the actually density, as long as we have access to the score at the noisy measurement.

SUMMARY

In this section, we explored the problem of scalable score matching. We looked at the notable attempts in the literature and learned that score can be estimated from samples only. We also looked at several interpretations of the learning objective and the connections they expose.

Last few bits

INCORPORATING TIME

In the last section, we expressed and explained everything in terms of one known noise level σ and the noisy sample $\tilde{x}$. We did so to avoid cluttering of multiple concepts that aren't necessary to explain each other. In a previous section however, we learned that the score must be estimated along every timestep of the forward process. By simply augmenting Eq.(14) with an additional time variable $t \in \mathcal{U}[0, 1]$ is sufficient to induce the time dependency in the score matching problem

$$J_{\mathrm{D}}(\theta) = \mathbb{E}_{x_0, \epsilon, t \sim \mathcal{U}[0,1], x_t \sim q_t(x_t|x_0)} \left[\frac{1}{2} \quad s_\theta(x_t, t) - (-\frac{\epsilon}{\sigma_t}) \quad ^2 \right] \tag{19}$$

.. where $q_t(x_t|x_0)$ is defined in a previous section and σ_t is the standard deviation of it.

WE TOOK AN DIFFERENT APPROACH

We would like to highlight that, in this article, we first explored the reverse process and then showed why the forward process emerges out of necessity. Typical diffusion models papers start from a forward process specification of the form

$$dx_t = f(t)x_t dt + g(t)dB_t$$

.. and then use Anderson's SDE reversal [16] to explain the reverse process, which also involves the score

$$dx_t = \left[f(t)x_t - g(t)^2 \underbrace{\nabla_{x_t} \log q_t(x_t)}_{s_\theta(x_t, t)} \right] dt + g(t)dB_t$$

We argue that our approach is more "organic" in the sense that it builds up the theory *chronologically*, exploring the exact path the community went through over time.

CONCLUSION

In this article, we dived deep into the theoretical fundamentals of Diffusion Models, which are often ignored by practitioners. We started from the 'heart' of diffusion models, i.e. scores, and built the concepts up almost chronologically. We hope this article will serve as a conceptual guide toward understanding diffusion models from the score SDE perspective. We intentionally avoid the 'probabilistic markov model' view of diffusion since more and more works have been seen to embrace the SDE formalism.

Footnotes

1. Thanks to this StackOverflow answer by @ben [↩]
2. Ordinary Differential Equations, or ODEs describe how a process evolves over time by its infinitesimal change. [↩]
3. Local maxima of probability density [↩]
4. In practice, the smaller step you take, the small noise you get. [↩]
5. It's called a "stationary or equilibrium distribution" [↩]
6. While each distribution vary in space (i.e. x) too, let's hide it for now and imagine them to be just a vectors. [↩]
7. You can notice this if you carefully see the first few frames of the animation. [↩]
8. Remember, they are infinitesimal gaussian noises. [↩]
9. In the sense that, given a score function, it just travels to the correct target distribution on its own. [↩]
10. On probability space, like we showed above [↩]
11. Do you remember that starting point doesn't matter ! [↩]
12. We encourage the reader to verify this on their own as an exercise. [↩]
13. Hint: compare dt with DDPM's β_t. [↩]
14. We use the standard assumption of $dt^2 = 0$. [↩]
15. You can see $t = 0 \implies t' = 0$ and $t = \infty \implies t' = 1$. Hence we converted the range $[0, \infty]$ to $[0, 1]$. [↩]
16. One can easily see that $t' = 1 - e^{-t} \implies dt' = e^{-t}dt \implies dt = e^t dt'$. [↩]
17. $e^t dt'$ is small because of dt', while increasing because of e^t. [↩]
18. We can only have access to the true score for distributions with closed-form, e.g. Gaussian. [↩]
19. A stochastic way of computing trace: $\mathrm{Tr}(M) = \mathbb{E}_{v \sim p_v}\left[v^T M v\right]$, where p_v can be a lot of distributions, most notably $\mathcal{N}(0, I)$. [↩]
20. Practically it's impossible to randomly sample a specific point. So we assume a little ball around each point. [↩]

References

1. **Auto-Encoding Variational Bayes**
   Kingma, D.P. and Welling, M., 2014. ICLR.

2. **Generative Adversarial Nets**
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. NeurIPS.

3. **Variational Inference with Normalizing Flows**
Rezende, D.J. and Mohamed, S., 2015. ICML.

4. **Denoising Diffusion Probabilistic Models**
Ho, J., Jain, A. and Abbeel, P., 2020. NeurIPS.

5. **Score-Based Generative Modeling through Stochastic Differential Equations** [link]
Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S. and Poole, B., 2021. International Conference on Learning Representations.

6. **The detection of linkage with “dominant” abnormalities**
Fisher, R.A., 1935. Annals of Eugenics, Vol 6(2), pp. 187--201. Wiley Online Library.

7. **Paul langevin’s 1908 paper “on the theory of brownian motion”[“sur la theorie du mouvement brownien,” cr acad. sci.(paris) 146, 530--533 (1908)]**
Lemons, D.S. and Gythiel, A., 1997. American Journal of Physics, Vol 65(11), pp. 1079--1081. American Association of Physics Teachers.

8. **Generative modeling by estimating gradients of the data distribution**
Song, Y. and Ermon, S., 2019. Advances in neural information processing systems, Vol 32.

9. **Image generation with shortest path diffusion**
Das, A., Fotiadis, S., Batra, A., Nabiei, F., Liao, F., Vakili, S., Shiu, D. and Bernacchia, A., 2023. International Conference on Machine Learning.

10. **Blurring Diffusion Models** [link]
Hoogeboom, E. and Salimans, T., 2023. The Eleventh International Conference on Learning Representations .

11. **Deep Unsupervised Learning using Nonequilibrium Thermodynamics**
Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N. and Ganguli, S., 2015. Proceedings of the 32nd International Conference on Machine Learning, Vol 37.

12. **Estimation of Non-Normalized Statistical Models by Score Matching**
Hyvärinen, A., 2005. Journal of Machine Learning Research.

13. **Sliced score matching: A scalable approach to density and score estimation**
Song, Y., Garg, S., Shi, J. and Ermon, S., 2020. Uncertainty in Artificial Intelligence, pp. 574--584.

14. **A connection between score matching and denoising autoencoders**
Vincent, P., 2011. Neural computation. MIT Press.

15. **An empirical Bayes approach to statistics**
Robbins, H.E., 1992. Breakthroughs in Statistics: Foundations and basic theory, pp. 388--394. Springer.

16. **Reverse-time diffusion equation models**
Anderson, B.D., 1982. Stochastic Processes and their Applications, Vol 12.

For attribution in academic contexts, please cite this work as

```
Das, "Building Diffusion Model's theory from ground up", ICLR Blogposts, 2024.
```

BibTeX citation

```
@inproceedings{das2024buildingdiffusionmodels,
  author = {Das, Ayan},
  title = {Building Diffusion Model's theory from ground up},
  abstract = {Diffusion Models, a new generative model family, have taken the world by storm after the seminal paper by Ho et al. [2020]. While diffusion models are often described as a probabilistic Markov Chains, their underlying principle is based on the decade-old theory of Stochastic Differential Equations (SDE), as found out later by Song et al. [2021]. In this article, we will go back and revisit the 'fundamental ingredients' behind the SDE formulation and show how the idea can be 'shaped' to get to the modern form of Score-based Diffusion Models. We'll start from the very definition of the 'score', how it was used in the context of generative modeling, how we achieve the necessary theoretical guarantees and how the critical design choices were made to finally arrive at the more 'principled' framework of Score-based Diffusion. Throughout this article, we provide several intuitive illustrations for ease of understanding.},
  booktitle = {ICLR Blogposts 2024},
  year = {2024},
  date = {May 7, 2024},
  note = {https://iclr-blogposts.github.io/2024/blog/diffusion-theory-from-scratch/},
  url  = {https://iclr-blogposts.github.io/2024/blog/diffusion-theory-from-scratch/}
}
```