CMPTG 5 - Statistical Physics of Neural Networks

Sidharth Kannan

April 2025

1 Lecture 4 - Stochastic Differential Equations and Diffusion

Over the course of the last few lectures, we have been discussing the Boltzmann machine, which is a prototype of a broader class of generative model, known as an *energy based model*. We saw how energy based models (EBMs), while simple in principle, run into issues due to the intractability of the partition function.

Now, we are going to switch tacks a little bit, to discuss another, more contemporary class of generative model, with close connections to statistical physics: the *diffusion model*. By the end of this segment of the course, you should have a good grasp of the principles of diffusion, and other flow based models, and see how they solve the problems we encountered in EBMs.

1.1 Recap

Recall that in generative modeling, our goal is to approximate the probability density function of some *data distribution*, $q(\mathbf{x})$, given a finite set of samples from it. In the case of the energy based model, we did this by assuming a particular functional form for the probability distribution,

$$\mathcal{P}(\mathbf{x}) = \frac{1}{Z} e^{-E_\theta(\mathbf{x})} \tag{1}$$

where our neural network parametrizes the *energy function*. In the case of a Boltzmann machine, the energy function took the form

$$E = -\sum_{i,j} J_{ij} m_i m_j - \sum_i h_i m_i \tag{2}$$

and our parameters are the matrix $\mathbf{J}$ and vector $\mathbf{h}$.

The way we trained this model was through *maximum likelihood training*. We said that there are a set of samples from our data distribution, and we wish to find the parameters that maximize the probability mass/density of those samples.

$$\max_\theta \prod_i p_\theta(\mathbf{x}_i) \tag{3}$$

Or, equivalently

$$\max_\theta \sum_i \log p_\theta(\mathbf{x}_i) \tag{4}$$

In order to maximize this objective, we had to perform gradient descent on the parameters, and in particular, we saw that we had to compute expectations against our model. Even though we have access to the form of the probability density function, we don't know the value of the partition function, Z, so in order to compute these expectations, we had to perform a very computationally expensive sampling procedure.

In order to solve this problem, we are going to change our perspective a little bit. We are going to give up on directly parameterizing the probability density function, and instead take the approach of reshaping a simple *prior* into the complex data distribution. This will lead us to diffusion, and then to some less well known, up and coming alternatives to diffusion models.

1.2 A Bird's Eye View on Diffusion

As a quick warm-up, consider the following toy example. Imagine that my prior is a simple, 1-dimensional, mean 0, unit variance Gaussian.

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{5}$$

and imagine that my data distribution is a mean 1, unit variance Gaussian.

$$q(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-1)^2}{2}} \tag{6}$$

If I was trying to learn this data distribution with an energy-based model, I would have to learn the energy function $E = -\frac{(x-1)^2}{2}$, somehow parameterized by a neural network. Instead, however, I could say that I will just draw samples from my prior, $p(x)$, and add 1 to them. This will yield the same distribution, but I have done this by applying a simple function to a prior. Of course, this is a trivial example, so I can write down a closed form for the transformation from the prior to the data distribution. In practice, we do not have access to the data distribution, but we can attempt instead to learn a set of transformations from our prior to the data. This is the way diffusion works.

We start with a Gaussian noise sample (a sample from a prior), and we iteratively update it to remove the noise (transform it) until it is fully denoised (we have a sample from the approximated data distribution). For an illustration, see Fig. 1.

1.2.1 Score Based Modelling

In the case of energy based models, we parametrize the data distribution as $\mathcal{P}_\theta(\mathbf{x}) = \frac{1}{Z} e^{-E_\theta(\mathbf{x})}$, and we ran into the problem that this partition function, Z, is intractable, which makes both training and inference time tasks challenging.

In order to train our energy based model, we employed the maximum-likelihood objective, in which we attempt to choose parameters such that the probability of the data is maximized.

$$\max_\theta \sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i) \tag{7}$$

However, this required that we have access to the normalized probability distribution, p_θ, or an approximation of this. To circumvent this, instead of having our model parametrize the probability density function, we can have it parametrize something called the *score function*, defined as

$$\mathbf{s}(x) = \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) \tag{8}$$

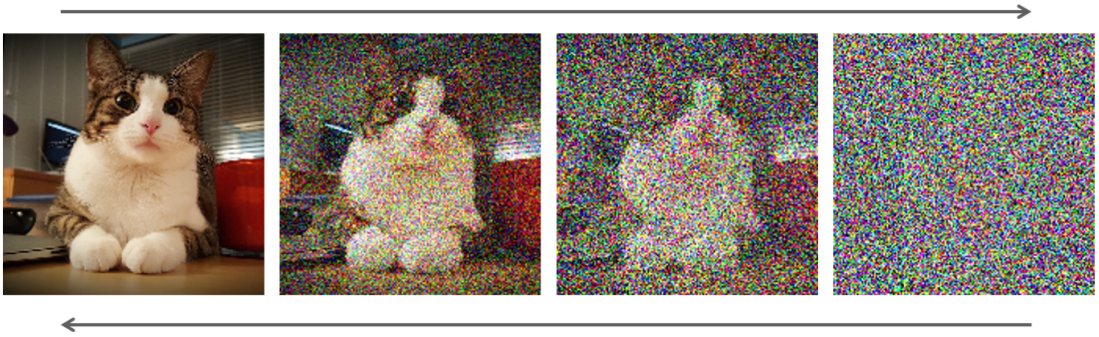


Figure 1: A sample of the diffusion process.

To see why this helps, let's go back to the example of our energy based model:

$$\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} \log \frac{1}{Z} e^{-E_\theta(\mathbf{x})} = -\nabla_{\mathbf{x}} E_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log Z \tag{9}$$

but since the partition function is a constant, the second term drops out, and our log-likelihood no longer depends on it! So, we see why the score function is useful. Now two questions remain:

1. How do we train a model to estimate the score?
2. Once our model closely approximates the score function, how do we actually use that score function to generate new samples?

I will skip answering the first question in detail, but the main idea is that we attempt to minimize the *Fisher divergence* between the model output and the score function of our data.

$$\mathcal{F}(p_{data}, \mathbf{s}_\theta) = \mathbb{E}\left[||\nabla_{\mathbf{x}} \log p_{data} - \mathbf{s}_\theta(\mathbf{x})||_2^2\right] \tag{10}$$

This equation still requires knowledge of the data distribution, but there exists a family of methods known as score matching that allow us to construct an objective that relies purely on the model score and the data samples. The second question is more relevant to us. The approach we will take is called *Langevin dynamics*.

1.2.2 Langevin dynamics

To understand the physics of what is going on, let's revisit the score function. We saw that the score function is the *spatial gradient of our energy function*. Physically, that tells us that it acts as some kind of force. To see this analogy more closely, let's talk about Langevin dynamics.

Langevin dynamics is a set of approaches taken in physics to model the motion of molecular systems. We know that the particle will move from high potential to low potential, but as it attempts to do this, it will be buffeted by collisions with the many other particles surrounding it. We can model this using something called the *Langevin equation*.

$$m\frac{d^2\mathbf{x}}{dt^2} = -\nabla U(x) - m\gamma\frac{dx}{dt} + \sqrt{2m\gamma k_B T}\epsilon \tag{11}$$

I've been a little schematic here with my notation. Let's break it down. The left hand side of the equation is the force on the particle. The right hand side has three terms, the first of which is the negative gradient of the potential, the second of which is a damping term. The third term is stochastic; ϵ is a Gaussian random variable, meant to represent the random forces due to collisions.

In many cases, the inertia term on the left is negligible compared to the damping term, and so we have that

$$m\gamma\frac{dx}{dt} = -\nabla U(x) + \sqrt{2m\gamma k_B T}\epsilon \tag{12}$$

Which is to say that if we want to know what the velocity of our particle is, it should be such that the particle follows the negative gradient of the potential, plus some stochastic term.

This leads to the Langevin algorithm for generative modeling. We start with a random sample from some easy to sample from distribution, and we have access to our score model. We update our sample using the following equation:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \eta\nabla_{\mathbf{x}}\log p(\mathbf{x}) + \sqrt{2\eta}\boldsymbol{\epsilon} \tag{13}$$

Note that since we have a model that parametrizes $\nabla_{\mathbf{x}}\log p(\mathbf{x})$, we don't have to have access to the true probability density.

In principle, this works. In practice, the model that we train to estimate the score will do so poorly in regions of low probability density. This is because our Fisher divergence equation is an expectation against $\mathbf{x}$, and so the terms in the summation are weighted by $p(\mathbf{x})$. This means that low probability regions don't contribute very much to the loss, and so are the last to be minimized.

Now, this is a problem for our algorithm, since we start by sampling randomly from some prior. This sample is likely not very close to a high probability region or our data distribution, and so the score estimate will be poorly, leading to poor performance overall. How do we solve this?

Well, it turns out that if we just add random noise to our training samples, we will get better performance in regions of low probability. This is because the noisy samples are further from the data distribution, and so we are essentially augmenting our training set with samples unlike those in the data distribution. The natural question arises: How much noise is enough noise? Too much noise and our dataset doesn't look like the data distribution anymore. Too little noise, and we have the same problem as before. We can start by adding very little noise, and then add a bit more, and a bit more, and so on. The loss function we use will then be a weighted sum over the Fisher divergences at each noise scale. This is what is termed the *forward process*.

To perform inference, now, we run our Langevin dynamics algorithm, first at the highest noise scale, then at the next, and so on, until zero noise. This is the *reverse process*. Note that if we

take the number of noise scales to infinity, we essentially get a continuous time process, modeled by a stochastic differential equation. Here, our score estimator would take the noise scale as input parameter (often denoted as time).

This is the *diffusion model*. There are many equivalent formalisms for and variants of diffusion models. These include denoising diffusion probabilistic models (DDPMs), score based diffusion models, etc. All of these formalisms are provably equivalent. However, for the sake of pedagogical clarity, we will stick to one: the stochastic differential equation (SDE) formalism, set forth by Song et. al., which is what happens when the number of noise scales goes to infinity.

1.3 Mathematical Preliminaries for Diffusion

Before we dive deep into the SDE formalism of diffusion, it would be prudent to review some math.

1.3.1 Ordinary and Partial Differential Equations

Here, we will very briefly review the conceptual foundations of ordinary and partial differential equations. This should all be familiar to you already, but I will take this as an opportunity to refresh your memory and introduce the notation that I will be using.

An *ordinary differential equation* (ODE) is an algebraic equation involving derivatives. An n-th order ODE is one that involves the n-th derivative of some unknown function, for which we are solving. For example, we have the general form for a first order ODE,

$$\frac{d}{dx}y(x) + g(x)y(x) = 0 \tag{14}$$

In this differential equation, $g(x)$ is a known function, and $y(x)$ is the function we are solving for. It is common to omit the arguments of the function we are solving for. That is

$$\frac{dy}{dx} + g(x) \cdot y = 0 \tag{15}$$

Newton's second law is an example of a *second-order* ODE:

$$F = ma \rightarrow F = m\frac{d^2x}{dt^2} \tag{16}$$

A *partial differential equation* is one that involves derivatives with respect to multiple different variables. For example, Laplace's equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = 0 \tag{17}$$

Now, let's go back to equation 15, and just recap the intuitive picture we should have of it. While this may all be recap, I harp on it because this intuitive picture will be instrumental in making the leap to stochastic differential equations. Eq. 15 is just saying that for some small step in x, the amount that y changes is proportional to the current value of y, times some known function $g(x)$. So, if we knew the current value of (x, y) (our initial condition), we could present the following discretization of the differential equation:

$$\frac{\Delta y}{\Delta x} = -g(x) \cdot y \tag{18}$$

$$y^{(x+1)} = y^{(x)} \cdot (1 - \Delta x \cdot g(x)) \tag{19}$$

and numerically solve our equation this way. Note that while the solution to a typical arithmetic equation is a number, the solution to a differential equation is a function.

1.3.2 Stochastic Processes

A stochastic process is a sequence of random variables. That is, it is a set of random variables, indexed by some quantity, say time. The most fundamental stochastic process is the *Wiener process* (a.k.a. the Brownian motion), denoted W_t.

The Wiener process is characterized by the following properties:

1. *Normally distributed increments*: For any two consecutive time steps t_i, t_{i+1}, the *increment* $W_{t_{i+1}} - W_{t_i} \sim \mathcal{N}(0, u)$ for some variance u, proportional to the time step.

2. *Independent increments*: Any two increments are independent.

Consider a mote of dust, floating in the air. Upon close examination, we see that in addition to its general drifting around, as caused by air currents, there is also a random jiggling, as a result of its constant bombardment by air molecules. This is the prototypical example of a Brownian motion. For an example of what this trajectory might look like, see Fig. 2. Looking at it, you might also see why this topic is relevant to mathematical finance. See Fig. 3.

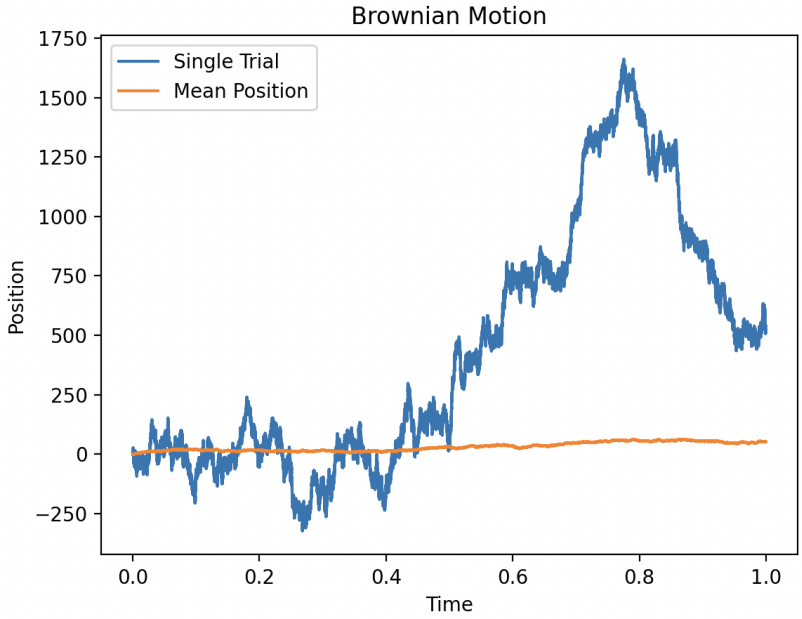


Figure 2: Simple Model of Brownian Motion. Particle starts at position 0, and at each time step, updates its position with by $\Delta x \sim \mathcal{N}(0, 1)$. Single Trial represents the trajectory of the particle given one such series of updates. Mean position is the average location of the particle at that time step, taken over 1,000 trials. Note that while on average, the particle is always at location 0, the actual value on a single trial can deviate quite a lot.

1.4 Stochastic Differential Equations

1.4.1 The Itô Calculus

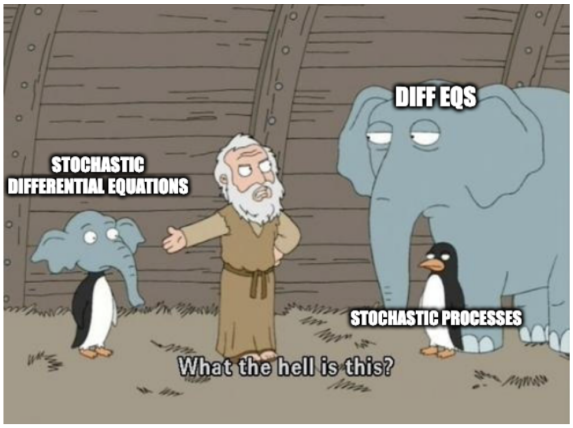
Now that we understand differential equations and stochastic processes, what do we get if we mix them? A *stochastic differential equation* is a differential equation whose solution is a stochastic process. This raises a lot of hairy questions. First of all, it is not at all clear what it means for a stochastic process to "solve" an equation. More than that, it turns out that our very basic notions of calculus, the derivative and integral, are ill-defined for stochastic processes, due to their discontinuous nature and unbounded nature.

I will, for the most part, ignore these questions, since they are not of much relevance to diffusion models. Sorry! However, to go some way in convincing you that the usual rules of calculus do not work for SDEs, and to help build some intuition for the stochastic calculus, I will offer the following example. Consider the following integral, where $W(t)$ is our standard Wiener process.

$$\int_0^{t_f} W(t) dW(t) \tag{20}$$

Ordinary calculus would tell you that this integrates to $\frac{1}{2}W(t)^2$, but this is not the case. To see this, let's go back to the basics, and divide up the interval into chunks $0 < t_1 < \cdots < t_f$, and

Figure 3: TSLA stock price over the past year. Doesn't it look kind of like our Brownian motion?



consider the sum.

$$\int_0^{t_f} W(t)dW(t) = \lim_{n\to\infty} \sum_k^n W(t_k) \cdot [W(t_{k+1}) - W(t_k)] \tag{21}$$

Now, we can rewrite this in the following way

$$= \lim_{n\to\infty} \sum_k^n \frac{1}{2} [W(t_{k+1}) - W(t_k)]^2 - \frac{1}{2} \left[W(t_{k+1})^2 - W(t_k)^2\right] \tag{22}$$

The second term just sums to $W(t)^2$, as it does in our ordinary calculus. The first is the interesting one. In ordinary calculus, the increment is approximately

$$[W(t_{k+1}) - W(t_k)] \approx W'(t_k)\Delta t \tag{23}$$

so $[W(t_{k+1}) - W(t_k)]^2$ is of order $(\Delta t)^2$, and so it drops out of the sum as $t \to 0$. This is not, however, the case in our stochastic integral. In the stochastic integral, we recognize $[W(t_{k+1}) - W(t_k)]$ as our increment, which we know to be normally distributed, with variance $\sqrt{\Delta t}$. Thus,

$$[W(t_{k+1}) - W(t_k)]^2 \sim \mathcal{O}(\Delta t) \tag{24}$$

and thus *cannot be ignored*. It turns out that since these terms are all of order Δt, we have

$$\int_0^{t_f} W(t)dW(t) = \frac{1}{2}W(t)^2 - \frac{1}{2}t \tag{25}$$

Weird, right! I refer you again to the meme at the top of the page. To make sense of this result, we might refer back to our intuition. The integral of a function is the area under its curve. If we were to integrate our mean 0 Gaussian process, we might reasonably expect that it integrates to 0, but ordinarily calculus gives us a function that would grow over time. Thus, we need to subtract off some amount to recover our expected result.

This is one particular example that we have walked through, but as a wise man once said, it only takes one counter example to topple an empire! The alternative formulation of calculus that we must use is called the Itô calculus, and as we have seen, it has slightly different rules for integration and differentiation. For a general stochastic process, or function of one, there is something called the Itô formula for how we might write its differential, but we will get to that later.

Now would probably be a good time to take a look at our first stochastic differential equation.

$$dX(t) = f(X_t, t)dt + g(t)dW_t \tag{26}$$

On the left side, we have the differential of our stochastic process, which is essentially, how much does it change? This rate of change is the sum of two terms:

1. A *deterministic* drift, which is function of its current value and the current time
2. A stochastic drift, which is a determined by a function g and a Brownian motion.

We will also specify some initial condition $X_0 \sim p_0$. Now, for the purposes of diffusion, we are not particularly interested in how to solve general continuous time stochastic differential equations. Rather, we care more about understanding their discrete time analogues.

We can discretize our SDE as follows:

$$X_{t+\Delta t} \approx X_t + \Delta t \cdot f(X_t, t) + g(t)\sqrt{s}(W_{t+\Delta t} - W_t) \tag{27}$$

Or, the conditional distribution of $X_{t+\Delta t}|X_t \sim \mathcal{N}\left(X_t + \Delta t \cdot f(X_t, t), \Delta t \cdot g^2(t)\right)$. This tells us that the deterministic drift function tells us about the change to the mean, and the random drift tells us the change in the standard deviation.

To bring this all back to diffusion models for a moment, what is going to end up happening is that we will choose particular functions, f, g, which will give us a stochastic differential equation for the forward process. The task of our model will then be to *invert* this differential equation, or run it in reverse time, to get back the original sample.

1.4.2 SDEs with Affine Drift Coefficients

Okay, going back to the forward process, we have a stochastic differential equation now, and a numerical scheme to approximate it. The next question becomes: Can we say anything about the distribution of X_t? What is its expectation? What is its variance?

It is hard to solve arbitrary differential equations. It is even harder to solve arbitrary stochastic differential equations. It is possible, however, to make analytic progress, when we make certain assumptions about the form of our equation.

For the purposes of diffusion models, the only SDEs that we care about are those with "affine" drift coefficients. That is, drift coefficients of the form

$$f(x, t) = a(t)x + b(t) \tag{28}$$

$g(t)$ already satisfies this form, since we can set $a(t) = 0, b(t) = g(t)$.

Then, we have that our SDE discretization (Eq. 27 has the form

$$X_{t+\Delta t} - X_t \approx \Delta t \cdot a(t)X_t + \Delta t \cdot b(t) + g(t)\sqrt{\Delta t}\epsilon \tag{29}$$

where $\epsilon \sim \mathcal{N}(0, 1)$.

Let's try to use this to compute $\mathbb{E}[X_t|X_0]$. Taking the expectation on both sides of 29, we have that

$$\mathbb{E}[X_{t+\Delta t}|X_0] - \mathbb{E}[X_t|X_0] \approx \Delta t \cdot a(t)\mathbb{E}[X_t|X_0] + \Delta t \cdot b(t) + g(t)\sqrt{\Delta t}\mathbb{E}[\epsilon] \tag{30}$$

ϵ is defined to be a mean 0 Gaussian, so its expectation is 0.

$$= \Delta t \cdot a(t)\mathbb{E}[X_t|X_0] + \Delta t \cdot b(t) \tag{31}$$

Dividing both sides by Δt and taking the limit as $\Delta t \to 0$, we see that $\mathbb{E}[X_t|X_0]$ satisfies the differential equation

$$\frac{d}{dt}\mathbb{E}[X_t|X_0] = a(t)\mathbb{E}[X_t|X_0] + b(t) \tag{32}$$

Which we recognize to be solved by an exponential.

$$\mathbb{E}[X_t|X_0] = \left(X_0 + \int_0^t b(s) \exp\left[-\int_0^s a(r)dr\right] ds\right) \exp\left(\int_0^t a(s)ds\right) \tag{33}$$

If we want $\mathbb{E}[X_t]$, we can just take the expectation against X_0 to get

$$\mathbb{E}[X_t] = \left(\mathbb{E}[X_0] + \int_0^t b(s) \exp\left[-\int_0^s a(r)dr\right] ds\right) \exp\left(\int_0^t a(s)ds\right) \tag{34}$$

We can carry out a similar exercise for the variance, to get that

$$\text{Var}(X_t) = \mathbb{E}[X_t^2] - \mathbb{E}[X_t]^2 = \left(\text{Var}(X_0) + \int_0^t g^2(s) \exp\left[-2\int_0^s a(r)dr\right] ds\right) \int_0^t 2a(s)ds \tag{35}$$

These are some rather nasty equations, admittedly. But there is something important to be gotten from them, which is that the distribution of $X_t|X_0$ is also *Gaussian*, with the mean given by $\mathbb{E}[X_t|X_0]$ and variance $\text{Var}(X_t|X_0)$. This will be critical for training our model.

Now, let's see how these formulas apply to our diffusion models. Starting with the discrete time case, we know that the forward process adds noise to our sample in N discrete stages. The noise level as a function of time is given by a noise scheduling function, $\beta(t)$, with the following properties:

1. $\beta(0) = 0$
2. $\beta'(t) \geq 0$
3. $\lim_{t\to\infty} \beta(t) = \infty$

and the forward process is written

$$X_{t_i+1} = \sqrt{1 - (\beta(t_{i+1}) - \beta(t_i))}X_i + \sqrt{\beta(t_{i+1}) - \beta(t_i)}\epsilon \tag{36}$$

So this gives us the conditional distribution,

$$q(X_{t_{i+1}}|X_{t_i}) \sim \mathcal{N}\left(\sqrt{1 - (\beta(t_{i+1}) - \beta(t_i))}X_{t_i}, \beta(t_{i+1}) - \beta(t_i)\right) \tag{37}$$

What is the stochastic differential equation corresponding to this? We can figure this out by simply deriving the expressions for our drift coefficients.

$$f(x,t) = \lim_{h\to 0} \frac{\mathbb{E}[X_{t+h} - X_t|X_t = x]}{h} \tag{38}$$

Substituting in the distributions for X_t, X_{t+h}, we have

$$\lim_{h\to 0} \frac{x\sqrt{1 - (\beta(t+h) - \beta(t))} - x}{h} = x \lim_{h\to 0} \frac{\sqrt{1 - (\beta(t+h) - \beta(t))} - 1}{h}$$

$$= x\frac{d}{dh}\sqrt{1 - \beta(t+h) + \beta(t)}\Big|_{h=0} = -x\frac{1}{2}\beta'(t)$$

A similar argument for $g(t)$ yields that $g(t) = \sqrt{\beta'(t)}$. So, the continuous time limit of the forward process of our diffusion model is

$$dX_t = -\frac{1}{2}\beta'(t)X_t dt + \sqrt{\beta'(t)}dW_t \tag{39}$$

What is the distribution of X_t in the limit of large t? Well, we just computed some formulas to give us the mean and variance of the solution to an SDE. Using Eqs. 28, 29, we can show (as you will on the homework) that

$$E[X_t|X_0] = X_0 \exp\left(\int_0^t -\frac{1}{2}\beta'(s)\,ds\right) = X_0 \exp\left(-\frac{1}{2}\beta(t)\right)$$

$$E[X_t] = E[X_0] \exp\left(-\frac{1}{2}\beta(t)\right)$$

The variance is given by

$$\text{Var}([X_t|X_0]) = 1 - \exp(-\beta(t))$$

And so,

$$\text{Var}([X_t]) = 1 + (\text{Var}([X_0]) - 1)\exp(-\beta(t))$$

And so, since we know that β is monotonically increasing, we see that the mean decays exponentially to 0, and if the variance of the training sample is 1, the variance stays at 1 for all time steps. This is called the *variance-preserving* model (VP-Diffusion). The key point here is that the diffusion process will invariably transform our sample into a sample from a Gaussian.

To make this a little more concrete, here are some examples of noise functions that have been used in seminal papers on diffusion:

1. Score-matching with Langevin Dynamics: This paper by Song and Ermon uses the noise scheduler
$$\beta(t) = \sigma_{\text{min}}^2 \left(\frac{\sigma_{\text{min}}^2}{\sigma_{\text{max}}^2}\right)^t \tag{40}$$

2. Ho et. al introduce the DDPM (Denoising Diffusion Probabilistic Model), which uses the noise function
$$\beta(t) = \frac{1}{2}t^2\left(\beta_{\text{max}} - \beta_{\text{min}}\right) + t\beta_{\text{min}} \tag{41}$$

This concludes our discussion of the forward process. We now understand that the forward process of a diffusion model can be seen as the continuous time limit of a score-matching model, and that this gives rise to a stochastic differential equation. For various noise schedules, we see how we can calculate the corresponding SDE, and we have seen how to prove that these SDEs do in fact map a training sample to noise.

1.4.3 The Reverse Process

Now that we can map a training sample to noise, using an SDE, all that remains to create our generative model is to learn how to *reverse* that SDE. If we can do that, then we can, in principle, run the SDE backwards, and the distribution of our resulting stochastic process will be the data distribution.

Let's make this notion a little more concrete. Our forward process does the following: It starts with a sample from the training set, X_0, and maps it to a random variable $X_t \sim p_t$. Now, we are asking the question: If we started with that sample $X_t \sim p_t$, is there an SDE that runs backward in time, and would give us a random variable distributed like X_0? I will use the notation of placing a bar over a quantity to represent its time reversed analogue.

$$d\bar{X}_t = \bar{f}(X_t, t)dt + \bar{g}(t)d\bar{W}_t \tag{42}$$

If our SDE is reversed, then the Brownian motion becomes reversed as well. Instead of the increment forward in time being Gaussian distributed, it is the difference between a time step and the *previous* one that is normally distributed.

It has been shown that for every Itô SDE of this form, we can create the reverse-SDE with the following two formulae:

$$\bar{g} = g \tag{43}$$

$$\bar{f}(x, t) = f(x, t) - g^2(t)\frac{\partial}{\partial x}\log p_t(x)$$

This is the univariate case. In multiple dimensions, the formula is essentially the same. So, our time reversed SDE becomes

$$d\bar{X}_t = [f(X_t, t) - g^2(t)\frac{\partial}{\partial x}\log p_t(\bar{X}_t)]dt + g(t)d\bar{W}_t \tag{44}$$

Et voila! We can see that in order to reverse the SDE, we need access to the score function, that we introduced so many eons ago. So, since we already know we can train a model to estimate these score functions, we have the complete picture. We can just simulate this SDE in reverse, and get our generated samples.

The question then becomes how do we simulate the SDE in reverse. The simplest method is known as the *Euler-Maruyama method*, which is a straightforward generalization of Euler's method from ordinary calculus. All we must do is approximate the derivatives above to first order, yielding the discretization

$$\Delta x = \left[f(x, t) - g^2(t)\nabla_x \log p_t(x)\right]\Delta t + g(t)\sqrt{\Delta t}\epsilon \tag{45}$$

We could, instead use a higher-order solver, like stochastic Runge-Kutta, but the details here are better left for a numerical methods course.

There is one thing to note here, as we solve our SDE. Our solver is actually introducing additional structure that we don't really care about. We only care about the marginal distribution at each step, $p_t(x_t)$, not the conditional distributions $p_t(x_t|x_{t-\Delta t}, x_{t-2\Delta t}, \dots)$. We can use this to improve our model output using the *predictor-corrector* scheme. Repeat the following procedure at each step:

1. Use the discretization in Eq. 45 to update the sample. (Predictor)
2. Use a purely score based MCMC (like Langevin dynamics) to improve the sample. (Corrector)