

Generative modelling lecture notes

Diffusion Models

Table of contents

1	Probabilistic Thinking (10/09/2026)	3
1.1	Basic Probability Rules	3
1.1.1	Bayes rule:	3
1.1.2	Law of total probability:	3
1.1.3	Jensen's inequality:	3
1.2	Level lines:	3
1.3	The Gaussian distribution	3
1.3.1	1D Gaussian distribution:	3
1.3.2	Multivariate Gaussian distribution:	3
1.3.2.1	Reminder on covariance	3
1.3.2.2	Multivariate Gaussian density	4
1.3.2.3	Special case: Isotropic Gaussian distribution	4
1.3.3	Another Tools for Gaussian !	4
1.4	Mixture of Gaussians (aka Gaussian Mixture Model, GMM)	5
1.4.1	The EM algorithm for a GMM	6
2	Variational Auto Encoder (VAEs)	7
2.1	Auto Encoder (AE)	7
2.2	Variational Auto Encoder (VAE)	7
2.2.1	Jensen's inequality for a probability density	8
2.2.2	Proof of the ELBO	8
2.2.3	Kullback–Leibler divergence	9
3	Generative Adversarial Networks (17/09/26)	10
3.1	GAN	10
3.1.1	Probability distribution and objective	10
3.2	Quick detour: why min–max problems are difficult	10
3.2.1	Proof	11
3.3	Solution used	11
3.3.1	Proposition: equilibrium of the GAN objective	11
3.3.2	Proof	11
3.3.3	First step	12
3.3.4	Proof	12
3.3.5	Fixed G, what is the value of $\varphi(G, D^*)$?	12
3.4	WGAN (Wasserstein GAN):	13
3.4.1	WGAN-GP: gradient penalty	13
4	Flow Matching (24/09/26)	14
4.1	Introduction: the global picture of Flow Matching	14
4.2	Generative modeling as a transport problem	15
4.2.1	Pushforward of a probability distribution	15
4.2.2	From flow of particles...	15
4.2.3	...to flow of probability distributions	17
4.3	Choosing a probability path	18
4.3.1	The transport is not unique	18
4.3.2	Conditional probability paths	18
4.3.3	Conditional velocity	19

4.4	Flow Matching	20
4.4.1	Reminder on conditional expectation	20
4.4.2	The marginal velocity field	21
4.4.3	The Flow Matching objective	22
4.4.4	Training algorithm	22
4.4.5	Sampling from the learned flow (post training)	22
5	Diffusion Models	24
5.1	Diffusion as a stochastic differential equation	24
5.1.1	The forward equation	24
5.1.2	Discretization and evolution of particles	24
5.1.3	Evolution of the density	25
5.2	Choosing a forward diffusion	27
5.2.1	Linear diffusions and transition kernels	27
5.2.2	Examples of schedules	29
5.2.3	When do schedules (α_t, σ_t) define a valid linear SDE?	30
5.3	The backward process	31
5.3.1	What happens when we reverse time?	31
5.3.2	The score appears	32
5.4	Learning the score	34
5.4.1	Denoising score matching	34
5.4.2	Gaussian corruption and noise prediction	36
5.4.3	Short parenthesis: DDPMs	36
5.5	Sampling	37
5.5.1	Euler–Maruyama	37
5.5.2	Deterministic sampling: the probability flow ODE	37
5.6	Diffusion and flow matching: two sides of the same coin	38
5.6.1	Both methods turn an easy conditional target into a hard marginal field	38
5.6.2	Tweedie’s formula: the score is a denoiser	39
5.6.3	For Gaussian source noise, score and velocity are explicitly related	40
	Bibliography	40

1 Probabilistic Thinking (10/09/2026)

Notes by Paul Bouchaud.

Goal of generative modelling: Given iid samples $(x_1, x_2, \dots, x_n) \sim p(x)$ learn to sample from $p(x)$.

Note: we write p interchangeably for the law, the probability or its density. This is not ideal, but everybody does it.

1.1 Basic Probability Rules

Most of the computation we will do in the class relies on the following 3 rules.

1.1.1 Bayes rule:

For two random variables X and Y , $p(x|y) = \frac{p(y|x)p(x)}{p(y)}$.

1.1.2 Law of total probability:

$$p(x) = \int_y p(x, y) dy = \int_y p(y)p(x|y) dy \quad (1)$$

1.1.3 Jensen's inequality:

If f is a convex function, then for any random variable X , we have:

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)] \quad (2)$$

As an example (and a way to remember in which direction to write it), for $f(x) = x^2$, we get $\mathbb{E}[X]^2 \leq \mathbb{E}[X^2]$, namely one of the proofs that the variance of X is a positive quantity.

1.2 Level lines:

Visualize level lines of a function at <https://mathurinm.github.io/levellines/>

1.3 The Gaussian distribution

This distribution is the most used and most important distribution in statistics and need to be know by heart. You can play with the visualization at <https://mathurinm.github.io/gaussian/>

1.3.1 1D Gaussian distribution:

In 1D, the Gaussian distribution density function is given by:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (3)$$

Where $\mu \in \mathbb{R}$ is the mean and $\sigma^2 > 0$ is the variance.

1.3.2 Multivariate Gaussian distribution:

For a random vector $X \in \mathbb{R}^d$ with mean $\mu = \mathbb{E}[X]$, its covariance matrix is defined by:

$$\Sigma = \mathbb{E}[(X - \mu)(X - \mu)^T] \in \mathbb{R}^{d \times d} \quad (4)$$

This matrix is positive semidefinite (all its eigenvalues are greater or equal than 0). See https://mathurinm.github.io/assets/pdf/math_background.pdf

1.3.2.1 Reminder on covariance

Assume the components of X have finite second moments, and write $\mu_i = \mathbb{E}[X_i]$. The entry in row i , column j is the covariance between X_i and X_j :

$$\begin{aligned}
\Sigma_{ij} &= \mathbb{E}[(X_i - \mu_i)(X_j - \mu_j)] \\
&= \mathbb{E}[X_i X_j] - \mu_j \mathbb{E}[X_i] - \mu_i \mathbb{E}[X_j] + \mu_i \mu_j \\
&= \mathbb{E}[X_i X_j] - \mathbb{E}[X_i] \mathbb{E}[X_j].
\end{aligned} \tag{5}$$

In particular, on the diagonal of Σ lie the variances of the coordinates of X :

$$\Sigma_{ii} = \sigma_i^2 = \mathbb{E}[X_i^2] - (\mathbb{E}[X_i])^2. \tag{6}$$

The diagonal entries are the variances of the individual components; σ_i is their standard deviation. The off-diagonal entries describe how pairs of components vary together. Equivalently,

$$\mathbb{E}[X_i X_j] = \Sigma_{ij} + \mu_i \mu_j, \quad \mathbb{E}[X_i^2] = \sigma_i^2 + \mu_i^2. \tag{7}$$

These identities hold for any random vector with finite second moments, not only Gaussian vectors.

1.3.2.2 Multivariate Gaussian density

For $\mu \in \mathbb{R}^d$ and a positive definite matrix $\Sigma \in \mathbb{R}^{d \times d}$, we say that the random variable X follows a Gaussian distribution with mean μ and covariance Σ , and we write $X \sim \mathcal{N}(\mu, \Sigma)$, if its density is

$$p(x) = p_{\mathcal{N}}(x; \mu, \Sigma) := \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right), \quad x \in \mathbb{R}^d. \tag{8}$$

The factor before the exponential normalizes the density so that it integrates to one. The quadratic expression $(x - \mu)^T \Sigma^{-1}(x - \mu)$ measures squared distance from the mean after accounting for the variances and correlations.

A covariance matrix is always symmetric and positive semidefinite. The density above requires it to be positive definite, hence invertible. A singular covariance matrix describes a degenerate Gaussian supported on a lower-dimensional subspace, which has no density with respect to volume in $\mathbb{R}^d$.

1.3.2.3 Special case: Isotropic Gaussian distribution

An isotropic Gaussian is a Gaussian that has the same variance in every direction. Its covariance matrix is $\Sigma = \sigma^2 I_d$, where I_d is the identity matrix and $\sigma > 0$. Thus,

$$\Sigma_{ii} = \sigma^2, \quad \Sigma_{ij} = 0 \quad \text{for } i \neq j. \tag{9}$$

Its components are independent Gaussians with the same variance, though their means may differ. Since $\det(\Sigma) = \sigma^{2d}$ and $\Sigma^{-1} = \frac{1}{\sigma^2} I_d$, the density plifies to

$$p(x) = \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \exp\left(-\frac{\|x - \mu\|^2}{2\sigma^2}\right). \tag{10}$$

This density depends only on the Euclidean distance from μ , so its contours are spheres (circles in two dimensions). The special case $\mu = 0$ and $\Sigma = I_d$ is the standard multivariate Gaussian $\mathcal{N}(0, I_d)$.

Passage on the vizualization of the Gaussian distribution in 2D and 3D at the following link: (<https://mathurinm.github.io/levellines/> and <https://mathurinm.github.io/gaussian/>)

1.3.3 Another Tools for Gaussian !

Let $x_1, x_2, \dots, x_n \in \mathbb{R}^d$ be independent and identically distributed observations from a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$. The maximum likelihood estimator of the mean vector μ is the sample mean:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^n x_i. \tag{11}$$

The maximum likelihood estimator for the covariance is the sample covariance $\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{\mu})(x_i - \hat{\mu})^T$

(there is a small technical detail: we need the empirical covariance to be invertible/definite for this to hold. We'll sweep this under rug, but be aware that what we have done is not 100% rigorous here)

In Figure 1, we see on the left the true distribution (via its level lines) of a Gaussian, and on the right, what we obtain with MLE for various sample sizes n . As n increases, the MLE estimated density gets closer to the true density.

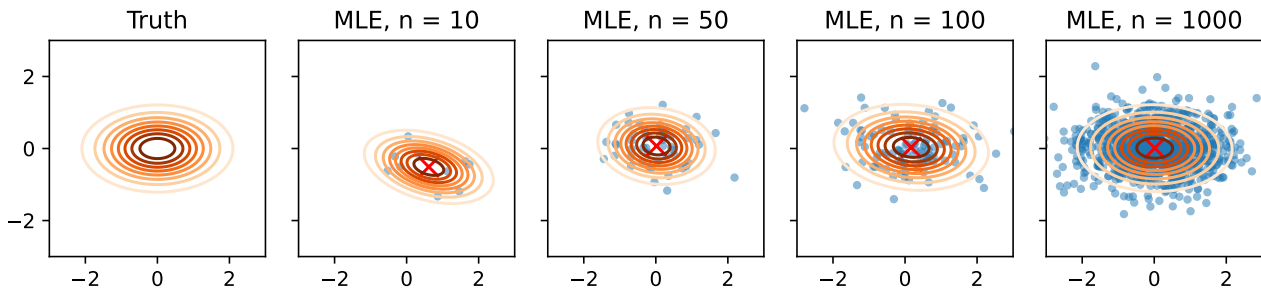


Figure 1: Illustration of MLE for various sample sizes n

From this we derive our simplest generative model:

- get your observations $x_1, \dots, x_n$
- assume that they are Gaussian
- fit a Gaussian model on your data by maximum likelihood
- generate your data by sampling from $\mathcal{N}(\hat{\mu}, \hat{\Sigma})$

Question: By the way, how do we sample from a Gaussian of given mean and covariance?

1.4 Mixture of Gaussians (aka Gaussian Mixture Model, GMM)

Definition: A mixture of $K \in \mathbb{N}$ Gaussians is a random variable whose density is

$$p(x) = \sum_{k=1}^K \pi_k p_{\mathcal{N}}(x; \mu_k, \Sigma_k) \quad (12)$$

where $\pi_k \in \mathbb{R}_+$ is the proportion coefficient of the k -th Gaussian distribution (they must sum to one), and $p_{\mathcal{N}}(x; \mu_k, \Sigma_k)$ is the density of the k -th Gaussian distribution in the mixture, which has mean μ_k and covariance matrix Σ_k .

Question: Why must the π_k be positive and sum to 1?

Example: how many Gaussians do you think there are in the mixture displayed in Figure 2? What are their means and covariances?

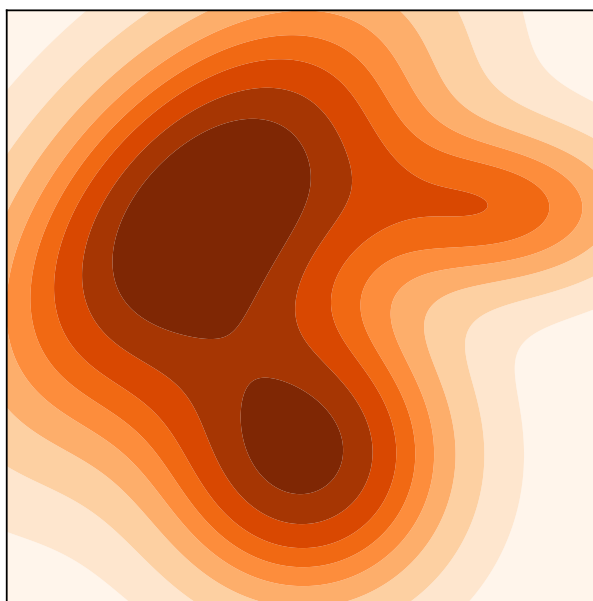


Figure 2: A mixture of Gaussians in 2D

See answer at the bottom of the document in Figure 3

We've seen how to estimate μ and Σ from samples if we believe our distribution is Gaussian: with MLE. Can we do the same if we believe the data comes from a mixture of Gaussians (with known K)?

The approach in MLE is to look for the set of parameters $\theta = (\pi_1, \dots, \pi_K, \mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K)$ that maximizes the (log) likelihood of the data, namely:

$$\arg \max \sum_{i=1}^n \log p(x_i; \theta) \quad (13)$$

For a GMM, we can write down the likelihood. But then, we cannot solve Equation 13 as we were able to do for a single Gaussian. There is a workaround to approximately solve the MLE: it is called the Expectation-Maximization algorithm (EM). It is in fact a much more generic algorithm for a wider class of problems, but we will stick to this particular case in the course.

1.4.1 The EM algorithm for a GMM

The idea is to introduce a latent variable y_i , taking values in $\llbracket 1, k \rrbracket$ for each observation x_i , which indicates which Gaussian generated the observation. Of course this variable is unknown (roughly, this is what "latent" variable means: unobserved, unavailable). If we knew it, our lives would be easier: we could just estimate μ_k and Σ_k by restricting ourselves to the set of points for which $y_i = k$.

The EM algorithm builds on this: it alternates between estimating the y_i 's, and estimating the parameters.

- If the parameters θ are known, estimating y_i (its law, actually) is easy. (Intuitively: we want to compute the density of each Gaussian at x_i , and attribute x_i to the Gaussian with highest density)

Formally, one has

$$P(y = k | x = x_i) = \frac{\pi_k p_{\mathcal{N}}(x_i; \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j p_{\mathcal{N}}(x_i; \mu_j, \Sigma_j)} \quad (14)$$

Indeed, by Bayes formula and law of total probability:

$$\begin{aligned}
p(y = k | x = x_i) &= p(x = x_i | y = k) \frac{p(y = k)}{p(x = x_i)} \\
\text{and } p(x) &= \sum_{k=1}^K p(x, y = k) \\
&= \sum_{k=1}^K p(x | y = k) p(y = k) \\
&= \sum_{k=1}^K \pi_k p_{\mathcal{N}}(x_i; \mu_k, \Sigma_k)
\end{aligned} \tag{15}$$

- If the y_i 's are known, we estimate the parameters using MLE for each “cluster” (warning: in fact the cluster attribution are “soft”, not hard), as was done for the single Gaussian case.

Once we have fitted a GMM, we can use it as a generative model: to sample a new point,

- pick a cluster k with probability π_k
- sample from $\mathcal{N}(\cdot; \mu_k, \Sigma_k)$

We are done with our second generative model!

2 Variational Auto Encoder (VAEs)

2.1 Auto Encoder (AE)

An autoencoder is a function f from $\mathbb{R}^d$ to $\mathbb{R}^d$, implemented by a neural network in which the layers first have a diminishing width, until you reach a bottleneck layer, and then the layer sizes increase again you get back to the original size, d . The idea is that the bottleneck layer will learn a compressed representation of the input data. This space is called the latent space.

The part until the bottleneck layer is called the encoder (it compresses the input into a small representation called the *code*), the remaining part is called the decoder.

What you want is that f is approximately the identity function so that the output is as close as possible to the input. Hence, the training loss of an autoencoder is:

$$\min_{\theta} \sum_{i=1}^n \|x_i - D_{\theta}(E_{\theta}(x_i))\|^2 \tag{16}$$

We can use an autoencoder as a generative model: we sample random codes (e.g. Gaussian, uniform) in the latent space and decode them with D_{θ} to generate new samples. However this is completely heuristic, there is no proof that the generated samples will be similar to the original data. A Variational autoencoder is a probabilistic framework to solve this.

2.2 Variational Auto Encoder (VAE)

This is where the variational autoencoder comes in. In a VAE, we two make modelling assumptions:

- We assume that an observation $x \in \mathbb{R}^d$ is generated from a latent variable $z \in \mathbb{R}^k$. We choose a standard Gaussian **prior** on z :

$$z \sim \mathcal{N}(0, I_k) \tag{17}$$

- We assume that x given z is Gaussian, of mean $D_{\theta}(z)$:

$$x|z \sim \mathcal{N}(D_{\theta}(z), I_d). \tag{18}$$

and $p(x|z)$ is called the **likelihood** (how likely is it to observe x if the latent code is z). Here, D_{θ} is still the decoder network, but it does not directly decode x : $x|z$ is random, what the decoder outputs is the mean of its distribution. Beyond the prior and the likelihood, there are two other probabilities that are interesting:

- The **posterior** $p_\theta(z|x)$ describes the possible latent values given an observation.
- The **evidence** $p_\theta(x)$ (or $\log(p_\theta(x))$) is the marginal density of that observation. Choosing the prior and the likelihood determines both, through the law of total probability and Bayes' rule.

How do we find good parameters θ ? As in the rest of this session, we want to maximize the log likelihood of the data:

$$\hat{\theta} = \arg \max_{\theta} \sum_{i=1}^n \log p_\theta(x_i), \quad (19)$$

where

$$\log p_\theta(x) = \log \left(\int_z p_\theta(x|z) p(z) dz \right). \quad (20)$$

This integral is generally *intractable*: computing it directly is too difficult in practice. This also makes the posterior $p_{\theta(z|x)}$ difficult to compute: the prior and the likelihood are computable easily (as Gaussians), they determine the values of the evidence and the posterior, but computing these last two is not possible.

We therefore introduce an **approximate posterior** $q_\varphi(z|x)$, which will somehow represent the (probabilistic) encoder. We choose it to be Gaussian:

$$q_\varphi(z|x) = \mathcal{N}(z; \mu(x), \text{diag}(\sigma_i^2(x))), \quad (21)$$

where μ and σ_i are implemented by a neural network of parameters φ . Here, the Gaussian density is evaluated at z , and the encoder outputs its mean $\mu_{\varphi(x)}$ and covariance $\text{diag}(\sigma_i^2(x))$. The parameters φ belong to the encoder, while θ belongs to the decoder.

Using Jensen's inequality, we can obtain a lower bound on the log likelihood (proof below):

$$\log p_\theta(x) \geq \mathbb{E}_{q_{\varphi(z|x)}} \left[\log \left(\frac{p_\theta(x, z)}{q_\varphi(z|x)} \right) \right] := \text{ELBO}(\theta, \varphi; x). \quad (22)$$

This is the **ELBO** (Evidence Lower Bound). Instead of maximizing the log likelihood directly, we maximize this bound over both θ and φ , summed over the observations, to learn the decoder and encoder together.

2.2.1 Jensen's inequality for a probability density

Let $K \subset \mathbb{R}^d$ and let p be a probability density on K , so that

$$p(x) \geq 0 \quad \wedge \quad \int_K p(x) dx = 1. \quad (23)$$

For every convex function $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}$ and every integrable function $f : K \rightarrow \mathbb{R}^d$, Jensen's inequality is

$$\varphi \left(\int_K f(x) p(x) dx \right) \leq \int_K \varphi(f(x)) p(x) dx. \quad (24)$$

Equivalently, if X has density p , then

$$\varphi(\mathbb{E}[f(X)]) \leq \mathbb{E}[\varphi(f(X))]. \quad (25)$$

2.2.2 Proof of the ELBO

For any approximate posterior $q_{\varphi(z|x)}$, we can rewrite the evidence as

$$\begin{aligned}
\log p_\theta(x) &= \log \left(\int_z p_\theta(x, z) \, dz \right) \\
&= \log \left(\int_z q_\varphi(z|x) \frac{p_\theta(x, z)}{q_\varphi(z|x)} \, dz \right) \\
&\geq \int_z q_\varphi(z|x) \log \left(\frac{p_\theta(x, z)}{q_\varphi(z|x)} \right) \, dz.
\end{aligned} \tag{26}$$

The last inequality is obtained by Jensen's inequality applied to $\varphi = -\log$. The right-hand side is the ELBO:

$$\mathcal{L}(\theta, \varphi; x) = \mathbb{E}_{q_\varphi(z|x)} \left[\log \left(\frac{p_\theta(x, z)}{q_\varphi(z|x)} \right) \right] \tag{27}$$

The ELBO admits a nice interpretation in terms of a well-known discrepancy measure between probabilities, the Kullback–Leibler divergence.

2.2.3 Kullback–Leibler divergence

For probability densities q and p , the KL divergence is

$$\text{KL}(q \parallel p) := \int q(z) \log \left(\frac{q(z)}{p(z)} \right) \, dz \geq 0. \tag{28}$$

Using Bayes' rule, the evidence can be decomposed as

$$\log p_\theta(x) = \text{ELBO}(\theta, \varphi; x) + \text{KL}(q_\varphi(\cdot|x) \parallel p_\theta(\cdot|x)). \tag{29}$$

Therefore,

$$\text{ELBO}(\theta, \varphi; x) = \log p_{\theta(x)} - \text{KL}(q_{\varphi(z|x)} \parallel p_{\theta(z|x)}), \tag{30}$$

which explains why the ELBO is a lower bound on the evidence.

We now only need to maximize the ELBO over both θ and φ , summed over the observations, to learn the decoder and encoder together.

Question: in its current form, the ELBO is not easy to compute (we do not know how to express $p_\theta(x, z)$ easily). Show that yet another formulation for the ELBO is:

$$\text{ELBO}(\theta, \varphi, x) = \mathbb{E}_{q_{\varphi(\cdot|x)}} [\log p_\theta(x|z)] - \text{KL}(q_{\varphi(\cdot|x)}, p(z)) \tag{31}$$

You will admit that the KL between two Gaussians admits a closed form; for the two Gaussians in the KL in Equation 31, it is:

$$\text{KL}(q_{\varphi(\cdot|x)}, p(z)) = \frac{1}{2} \sum_{i=1}^d \sigma_i^2 + \mu_i - 1 - \log(\sigma_i^2) \tag{32}$$

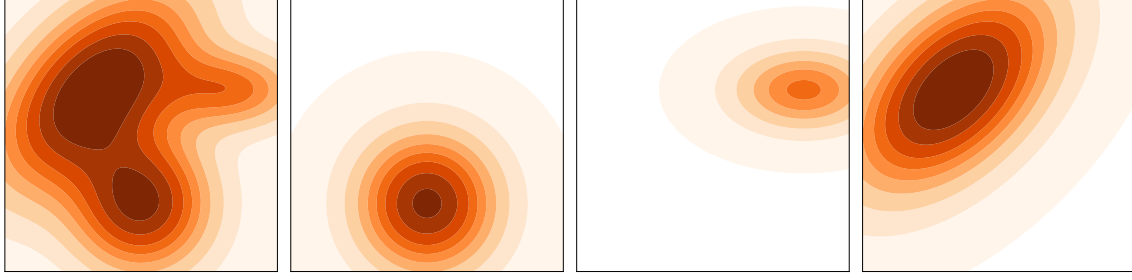


Figure 3: Mixture of Figure 2 decomposed in its 3 components.

3 Generative Adversarial Networks (17/09/26)

Goal: learn a generator G_θ whose generated distribution is close to the data distribution.

3.1 GAN

In a GAN, we learn two neural networks: the generator and the discriminator. The generator produces synthetic data, while the discriminator receives a sample and predicts whether it is real or generated.

The generator maps a latent vector to the data space, for example $G_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^m$. The discriminator maps a data point to a probability:

$$D_\varphi : \mathbb{R}^m \rightarrow \{0, 1\} \quad (33)$$

The goal is to train the generator to fool the discriminator, while the discriminator learns to distinguish real data from generated data.

3.1.1 Probability distribution and objective

For each input sample x_i , the discriminator $D(x_i)$ gives the probability that the sample is real. Let $y_i \in \{0, 1\}$ be its label, where $y_i = 1$ means real and $y_i = 0$ means generated. Therefore:

$$P(y_i = 1 \mid x_i) = D(x_i), \quad P(y_i = 0 \mid x_i) = 1 - D(x_i).$$

These two cases can be written in one expression:

$$P(y_i \mid x_i) = D(x_i)^{y_i} (1 - D(x_i))^{1-y_i}.$$

Assuming that the samples are independent, the likelihood of the labels is:

$$P(y_1, \dots, y_n \mid x_1, \dots, x_n) = \prod_{i=1}^n D(x_i)^{y_i} (1 - D(x_i))^{1-y_i}.$$

Taking the logarithm gives:

$$\log P(y_1, \dots, y_n \mid x_1, \dots, x_n) = \sum_{i=1}^n [y_i \log D(x_i) + (1 - y_i) \log(1 - D(x_i))].$$

For the GAN, real samples follow the data distribution $x \sim p_{\text{data}}$. Generated samples are obtained by first sampling $z \sim \mathcal{N}(0, I_d)$ and then computing $G(z)$. With infinitely many samples, the average log-likelihood becomes the following expectation:

$$\varphi(D, G) = E_{x \sim p_{\text{data}}} [\log(D(x))] + E_{z \sim \mathcal{N}(0, I_d)} [\log(1 - D(G(z)))].$$

The discriminator maximizes $\varphi(D, G)$: it wants $D(x)$ to be close to 1 for real samples and $D(G(z))$ to be close to 0 for generated samples. The generator minimizes $\varphi(D, G)$: it wants generated samples to be classified as real. Thus, the GAN objective is:

$$\min_G \max_D \varphi(D, G).$$

3.2 Quick detour: why min-max problems are difficult

Let us see why min-max problems can be difficult. Consider the function

$$\varphi(x, y) = xy.$$

We want to solve:

$$\min_x \max_y \varphi(x, y).$$

The variable x is minimized, so we use gradient descent for x . The variable y is maximized, so we use gradient ascent for y . Starting from (x_0, y_0) , the ultaneous update is:

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} + \eta \begin{pmatrix} -\nabla_x \varphi(x_k, y_k) \\ \nabla_y \varphi(x_k, y_k) \end{pmatrix}.$$

Since $\nabla_x \varphi(x, y) = y$ and $\nabla_y \varphi(x, y) = x$, this becomes the matrix recurrence:

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} 1 & -\eta \\ \eta & 1 \end{pmatrix} \begin{pmatrix} x_k \\ y_k \end{pmatrix}.$$

This procedure does not converge to $(0, 0)$ in general; instead, its iterates move away from the solution.

3.2.1 Proof

Consider the squared distance to the origin:

$$r_k^2 = x_k^2 + y_k^2.$$

Using the update equations:

$$r_{k+1}^2 = (x_k - \eta y_k)^2 + (y_k + \eta x_k)^2$$

$$r_{k+1}^2 = x_k^2 - 2\eta x_k y_k + \eta^2 y_k^2 + y_k^2 + 2\eta x_k y_k + \eta^2 x_k^2$$

$$r_{k+1}^2 = (1 + \eta^2)(x_k^2 + y_k^2) = (1 + \eta^2)r_k^2.$$

For every nonzero learning rate η , we have $1 + \eta^2 > 1$. Therefore:

$$r_k^2 = (1 + \eta^2)^k r_0^2.$$

If $(x_0, y_0) \neq (0, 0)$, then $r_0^2 > 0$ and r_k^2 grows with k . Consequently, the iterates diverge instead of converging to the saddle point $(0, 0)$. This shows why applying ultaneous descent to both variables is incorrect in a min-max problem: the maximizing variable must use ascent.

Solution that exists but will not be looked at today are : Optimistic Gradient and Extragradient

3.3 Solution used

3.3.1 Proposition: equilibrium of the GAN objective

Assume that the generator G^* reproduces the data distribution exactly:

$$p_{G^*} = p_{\text{data}}.$$

$$D^*(x) = \frac{1}{2} \quad \text{for all } x.$$

Then, (G^*, D^*) is a saddle point of the GAN objective. For every discriminator D and every generator G :

$$\varphi(G^*, D) \leq \varphi(G^*, D^*) \leq \varphi(G, D^*).$$

The first inequality says that D^* maximizes the objective when the generator is fixed at G^* . The second says that G^* minimizes the objective when the discriminator is fixed at D^* .

3.3.2 Proof

$$\varphi(G, D) = E_{x \sim p_{\text{data}}} [\log(D(x))] + E_{x \sim p_G} [\log(1 - D(x))].$$

$$p_{G^*} = p_{\text{data}}$$

$$\varphi(G^*, D) = E_{x \sim p_{\text{data}}} [\log(D(x)) + \log(1 - D(x))].$$

$$D(x)(1 - D(x)) = \frac{1}{4} - \left(D(x) - \frac{1}{2}\right)^2 \leq \frac{1}{4}$$

$$\log(D(x)) + \log(1 - D(x)) \leq \log\left(\frac{1}{4}\right) = -\log 4$$

$$D^*(x) = \frac{1}{2}$$

$$\varphi(G^*, D) \leq -\log 4 = \varphi(G^*, D^*).$$

$$\varphi(G, D^*) = E_{x \sim p_{\text{data}}} [\log(\frac{1}{2})] + E_{x \sim p_G} [\log(\frac{1}{2})]$$

$$\varphi(G, D^*) = \log(\frac{1}{2}) + \log(\frac{1}{2}) = -\log 4 = \varphi(G^*, D^*)$$

$$\varphi(G^*, D) \leq \varphi(G^*, D^*) \leq \varphi(G, D^*).$$

3.3.3 First step

For a fixed generator G and data distribution p_{data} , compute the optimal discriminator:

$$D^* = \arg \max_D \varphi(G, D).$$

Proposition: For a fixed generator G , the optimal discriminator is:

$$D^*(x) = \frac{p_{\text{data}(x)}}{p_{\text{data}(x)} + p_{G(x)}}.$$

3.3.4 Proof

For each fixed x , define:

$$f_{x(d)} = p_{\text{data}(x)} \log(d) + p_{G(x)} \log(1-d), \quad d \in (0, 1).$$

$$f_{x'}(d) = \frac{p_{\text{data}(x)}}{d} - \frac{p_{G(x)}}{1-d}.$$

At a stationary point:

$$\begin{aligned} f_{x'}(d) = 0 &\Rightarrow p_{\text{data}(x)}(1-d) = p_{G(x)}d \\ &\Rightarrow p_{\text{data}(x)} = d(p_{\text{data}(x)} + p_{G(x)}) \\ &\Rightarrow d = \frac{p_{\text{data}(x)}}{p_{\text{data}(x)} + p_{G(x)}}. \end{aligned}$$

Moreover:

$$f_{x''}(d) = -\frac{p_{\text{data}(x)}}{d^2} - \frac{p_{G(x)}}{(1-d)^2} < 0.$$

Thus, f_x is strictly concave and its stationary point is its unique maximum. Since the objective is the integral of $f_{x(D(x))}$, it is maximized pointwise:

$$D^*(x) = \frac{p_{\text{data}(x)}}{p_{\text{data}(x)} + p_{G(x)}}.$$

3.3.5 Fixed G, what is the value of $\varphi(G, D^*)$?

Recap:

For probability densities p and q :

$$\text{KL}(p \parallel q) = \int_x p(x) \log\left(\frac{p(x)}{q(x)}\right) dx = E_{x \sim p} \left[\log\left(\frac{p(x)}{q(x)}\right) \right].$$

Define the mixture density:

$$m(x) = \frac{p(x) + q(x)}{2}.$$

The Jensen–Shannon divergence is:

$$\text{JS}(p \parallel q) = \frac{1}{2} \text{KL}(p \parallel m) + \frac{1}{2} \text{KL}(q \parallel m).$$

Now let $m(x) = \frac{p_{\text{data}(x)} + p_{G(x)}}{2}$. Then:

$$\begin{aligned}
\max_D \varphi(G, D) &= \int_x p_{\text{data}(x)} \log\left(\frac{p_{\text{data}(x)}}{p_{\text{data}(x)} + p_{G(x)}}\right) dx + \int_x p_{G(x)} \log\left(\frac{p_{G(x)}}{p_{\text{data}(x)} + p_{G(x)}}\right) dx \\
&= \int_x p_{\text{data}(x)} \log\left(\frac{p_{\text{data}(x)}}{2m(x)}\right) dx + \int_x p_{G(x)} \log\left(\frac{p_{G(x)}}{2m(x)}\right) dx \\
&= \int_x p_{\text{data}(x)} \log\left(\frac{p_{\text{data}(x)}}{m(x)}\right) dx + \int_x p_{G(x)} \log\left(\frac{p_{G(x)}}{m(x)}\right) dx - \log 4 \\
&= \text{KL}(p_{\text{data}} \parallel m) + \text{KL}(p_G \parallel m) - \log 4 \\
&= 2 \text{JS}(p_{\text{data}} \parallel p_G) - \log 4.
\end{aligned}$$

So the GAN objective can be rewritten as:

$$\min_G \max_D \varphi(G, D) = 2 \min_G (\text{JS}(p_{\text{data}} \parallel p_G)) - \log 4.$$

3.4 WGAN (Wasserstein GAN):

$$\min_G W(p_{\text{data}}, p_G).$$

The Kantorovich–Rubinstein duality gives:

$$W(p, q) = \max_{f \in \text{Lip}_1} [E_{x \sim p}[f(x)] - E_{x \sim q}[f(x)]],$$

where Lip_1 is the set of 1-Lipschitz functions:

$$|f(x) - f(y)| \leq \|x - y\|.$$

Using a critic F to represent f :

$$\varphi^{\text{WGAN}(G, F)} = E_{x \sim p_{\text{data}}}[F(x)] - E_{x \sim p_G}[F(x)].$$

Equivalently, with $z \sim \mathcal{N}(0, I_d)$ and $x = G(z)$:

$$\varphi^{\text{WGAN}(G, F)} = E_{x \sim p_{\text{data}}}[F(x)] - E_{z \sim \mathcal{N}(0, I_d)}[F(G(z))].$$

Therefore, the WGAN objective is:

$$\min_G \max_{F \in \text{Lip}_1} \varphi^{\text{WGAN}(G, F)}.$$

3.4.1 WGAN-GP: gradient penalty

In practice, the 1-Lipschitz constraint is enforced using a gradient penalty. For real samples $x \sim p_{\text{data}}$ and generated samples $G(z)$, define:

$$\hat{x} = \varepsilon x + (1 - \varepsilon)G(z), \quad \varepsilon \sim \text{Uniform}(0, 1).$$

Therefore:

$$\varphi^{\text{WGAN-GP}(G, F)} = E_{x \sim p_{\text{data}}}[F(x)] - E_{z \sim \mathcal{N}(0, I_d)}[F(G(z))] - E_{\hat{x}} \left[\left(\|\nabla_{\hat{x}} F(\hat{x})\|_2 - 1 \right)^2 \right].$$

The WGAN-GP objective is:

$$\min_G \max_F \varphi^{\text{WGAN-GP}(G, F)}.$$

4 Flow Matching (24/09/26)

Some useful references:

- Flow Matching was introduced for generative modeling in (Lipman et al., 2023), (Liu et al., 2023), and (Albergo & Vanden-Eijnden, 2023).
- A visual blog post that the class follows: [Conditional Flow Matching](#) (Gagneux et al., 2025).
- Flow Matching from a mathematical perspective: (Pierret et al., 2026).

4.1 Introduction: the global picture of Flow Matching

Let p_{data} denote the unknown data distribution on $\mathbb{R}^d$. We do not have access to its density; instead, we observe i.i.d. samples

$$x^1, \dots, x^n \sim p_{\text{data}}. \quad (34)$$

The goal of generative modeling is to construct a procedure that produces new samples approximately distributed according to p_{data} .

A natural strategy is to start from a reference distribution p_0 , from which sampling is easy, for instance $p_0 = \mathcal{N}(0, I_d)$, and to construct a map $T : \mathbb{R}^d \rightarrow \mathbb{R}^d$ that transports p_0 to p_{data} . This first viewpoint is illustrated in Figure 4.

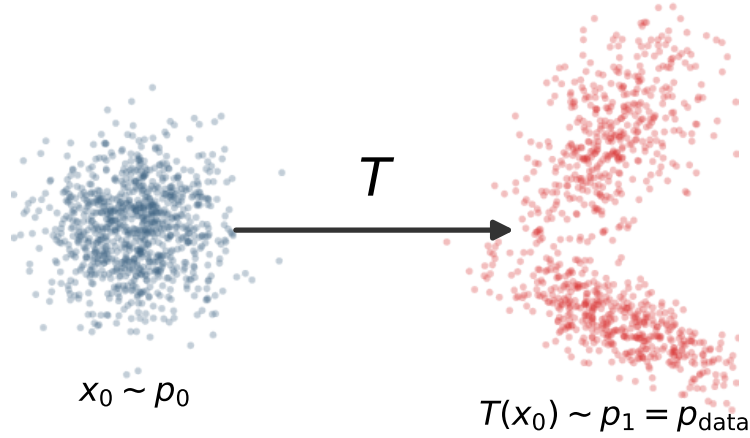


Figure 4: The generative problem can first be seen as learning a transformation from a reference source distribution to the data distribution.

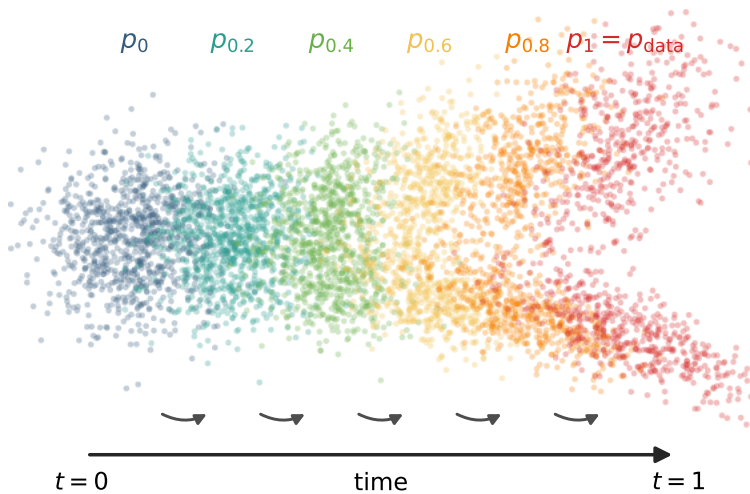


Figure 5: Flow Matching decomposes this transformation into small steps in time.

Unlike latent-variable models such as VAEs, the basic Flow Matching construction considered here works directly in the data space. Instead of learning one complicated map T in a single step, we decompose the transport into continuously many infinitesimal motions, as in Figure 5. These infinitesimal motions are described by a time-dependent velocity field.

4.2 Generative modeling as a transport problem

4.2.1 Pushforward of a probability distribution

Definition 1. Let μ be a probability measure on $\mathbb{R}^d$ and let $T : \mathbb{R}^d \rightarrow \mathbb{R}^d$ be measurable. The *pushforward* of μ by T , denoted by $T_{\#}\mu$, is the probability measure defined by

$$(T_{\#}\mu)(A) = \mu(T^{-1}(A)) \quad (35)$$

for every measurable set $A \subset \mathbb{R}^d$. Equivalently, if $X \sim \mu$, then $T(X) \sim T_{\#}\mu$.

Therefore, if $T_{\#}p_0 = p_{\text{data}}$, sampling from the data distribution is straightforward: sample $x_0 \sim p_0$ and return $T(x_0)$.

Exercise 2 (Transporting one Gaussian to another). Let $p_0 = \mathcal{N}(0, I_d)$ and $p_1 = \mathcal{N}(m, \sigma^2 I_d)$. Find a map $T : \mathbb{R}^d \rightarrow \mathbb{R}^d$ such that $T_{\#}p_0 = p_1$.

Solution. Consider $T(x) = m + \sigma x$. Indeed, if $X \sim \mathcal{N}(0, I_d)$, then

$$T(X) = m + \sigma X \sim \mathcal{N}(m, \sigma^2 I_d). \quad (36)$$

Hence $T_{\#}p_0 = p_1$.

4.2.2 From flow of particles...

For realistic data distributions, constructing a transport map directly is difficult. Instead, we build the transport progressively by moving points continuously in time. Let

$$v : [0, 1] \times \mathbb{R}^d \rightarrow \mathbb{R}^d \quad (37)$$

be a time-dependent vector field. At time t , the vector $v_{t(x)} = v(t, x)$ specifies the instantaneous direction and speed of a particle located at x . Figure 6 shows an example of such a time-dependent field in two dimensions.

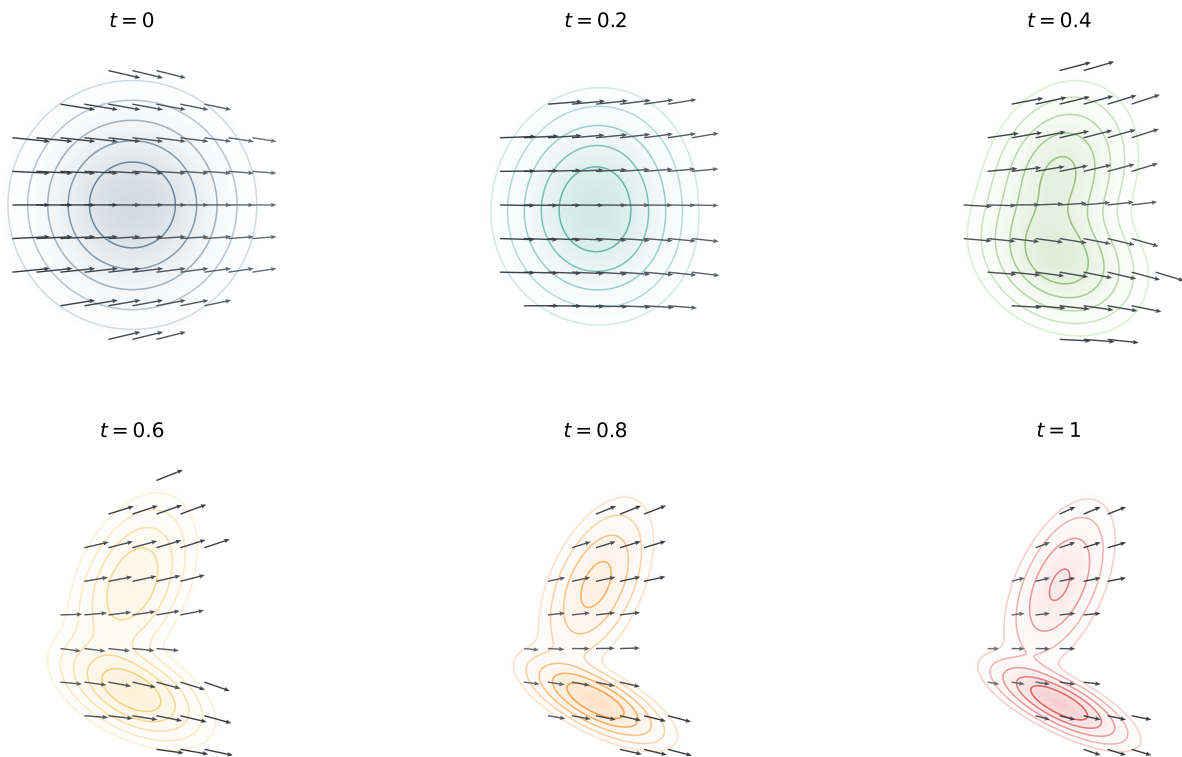


Figure 6: An example of velocity field for the Gaussian-to-GMM example.

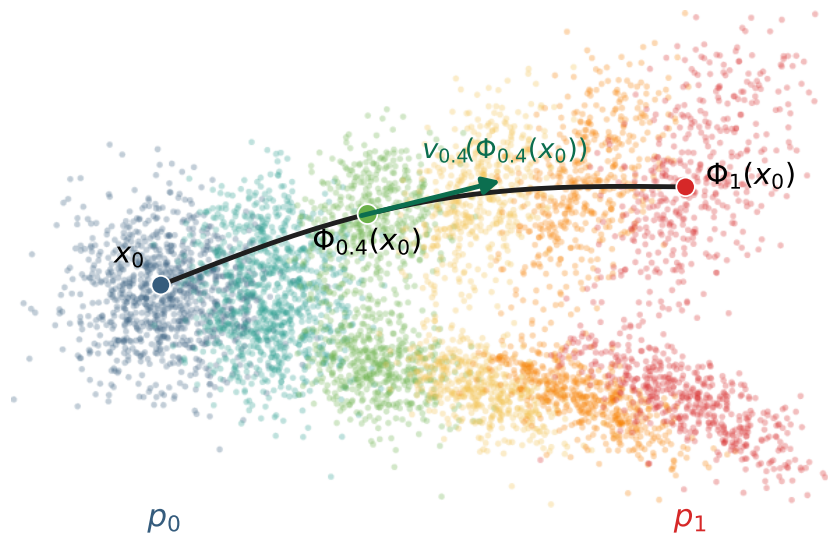


Figure 7: The flow viewpoint follows individual particles. The velocity vector is tangent to the particle trajectory.

For each initial position $x \in \mathbb{R}^d$, consider the ODE

$$\begin{cases} \frac{d}{dt}\varphi_t(x) = v_t(\varphi_t(x)) \\ \varphi_0(x) = x. \end{cases} \quad (38)$$

For fixed x , the curve $t \mapsto \varphi_t(x)$ is the trajectory of the particle starting at x , as shown in Figure 7. For example, if the velocity is constant, $v_t(x) = a$, then $\varphi_t(x) = x + ta$.

For fixed t , the map $\varphi_t : \mathbb{R}^d \rightarrow \mathbb{R}^d$ sends each initial position to its position at time t . The family $(\varphi_t)_{t \in [0,1]}$ is the *flow* generated by v .

Thus, if we can find a velocity field v such that $(\varphi_1)_\# p_0 = p_{\text{data}}$, then we obtain a generative model: sample $x_0 \sim p_0$ and return $x_1 = \varphi_1(x_0)$. The final flow map φ_1 plays the role of the transport map T , but it is obtained indirectly by learning and integrating the velocity field.

4.2.3 ...to flow of probability distributions

So far, the flow φ_t describes the motion of individual points. We now describe the evolution of an entire probability distribution.

Let $X_0 \sim p_0$ and move the sample according to the flow, $X_t = \varphi_t(X_0)$. If p_t denotes the law of X_t , then by definition of the pushforward,

$$p_t = (\varphi_t)_\# p_0.$$

It is useful to distinguish two complementary viewpoints:

$$\frac{d}{dt} \varphi_t(x) = v_t(\varphi_t(x)) \quad (39)$$

which describes the motion of individual particles, whereas

$$p_t = (\varphi_t)_\# p_0 \quad (40)$$

describes the evolution of the whole distribution.

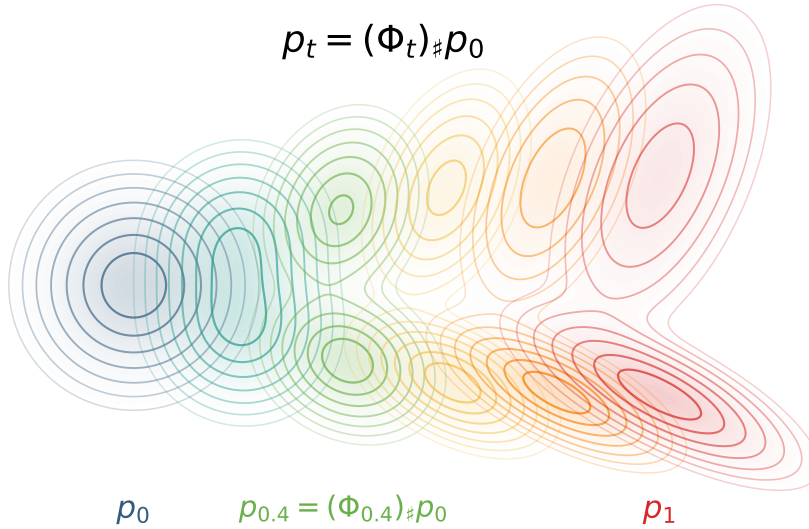


Figure 8: At the distribution level, the flow transports the whole probability measure: $p_t = (\varphi_t)_\# p_0$.

The corresponding distribution-level viewpoint is visualized in Figure 8. If particles move according to the velocity field v_t , then the density evolves according to the *continuity equation*

$$\partial_t p_t(x) + \nabla \cdot (p_t(x) v_t(x)) = 0 \quad (41)$$

This equation expresses conservation of probability mass.

Interpretation in one dimension. In one dimension, the continuity equation becomes $\partial_t p_t(x) + \partial_x (p_t(x) v_t(x)) = 0$. For an interval $[a, b]$, $\frac{d}{dt} \int_a^b p_t(x) dx = p_t(a) v_t(a) - p_t(b) v_t(b)$. The first term is the flux entering through the left endpoint and the second is the flux leaving through the right endpoint. In particular, if the outgoing flux at b is larger than the incoming flux at a , the probability mass inside $[a, b]$ decreases.

Weak formulation. For every smooth compactly supported test function $\phi \in \mathcal{C}_c^\infty(\mathbb{R}^d)$, $\frac{d}{dt} \int_{\mathbb{R}^d} \phi(x) p_t(x) dx = \int_{\mathbb{R}^d} \nabla \phi(x)^\top v_t(x) p_t(x) dx$.

The ODE and the continuity equation describe the same transport at two different levels. For example, in one dimension, if $v_t(x) = a$ is constant, then every particle follows $\varphi_t(x) = x + at$, and the density is translated according to $p_t(x) = p_0(x - at)$. A direct computation gives $\partial_t p_t + a \partial_x p_t = 0$, which is precisely the continuity equation for the constant velocity field $v_t = a$.

We will use the following classical result without proof.

Theorem (From the continuity equation to a flow - (Ambrosio et al., 2005)). Under suitable regularity assumptions, if $(p_t)_{t \in [0,1]}$ and v_t satisfy

$$\partial_t p_t + \nabla \cdot (p_t v_t) = 0, \quad p_{t=0} = p_0, \quad (42)$$

and φ_t is the flow generated by v_t , then

$$p_t = (\varphi_t)_\# p_0. \quad (43)$$

Consequence. To build a generative model, it is enough to construct a probability path $(p_t)_{t \in [0,1]}$ and a velocity field v_t such that

$$p_{t=0} = p_0, \quad p_{t=1} = p_{\text{data}}, \quad \partial_t p_t + \nabla \cdot (p_t v_t) = 0 \quad (44)$$

4.3 Choosing a probability path

4.3.1 The transport is not unique

The endpoint conditions $p_{t=0} = p_0$ and $p_{t=1} = p_{\text{data}}$ do not determine a unique transport. There are generally infinitely many probability paths and velocity fields connecting the same two distributions, as illustrated in Figure 9.

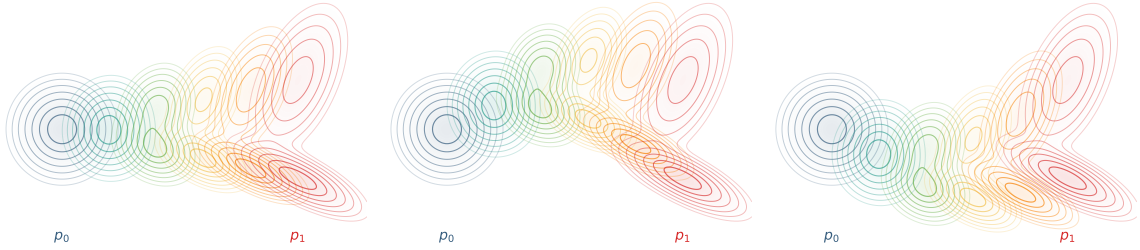


Figure 9: The same endpoint distributions can be connected by many different probability paths.

Since many valid transports exist, Flow Matching deliberately chooses a path for which the corresponding learning problem is ple.

4.3.2 Conditional probability paths

Let

$$X_0 \sim p_0 = \mathcal{N}(0, I_d), \quad X_1 \sim p_{\text{data}} \quad (45)$$

be independent and define the interpolated random variable

$$X_t = (1 - t)X_0 + tX_1, \quad t \in [0, 1].$$

Remark 3 (Non independent case). It is also possible to consider non independent variables X_0 and X_1 . In that case, all the following reasoning has to be done on the joint law p_{X_0, X_1} . This is referred to as *coupling* in the literature. We don't go over it in this class.

This choice defines a probability path $p_t = \text{Law}(X_t)$ from p_0 to p_{data} .

We first condition on the two endpoints. If $(X_0, X_1) = (x_0, x_1)$ is fixed, then

$$X_t = (1 - t)x_0 + tx_1 \quad (46)$$

is deterministic. Therefore, if we denote $p_t^{\text{cond}}(\cdot \mid x_0, x_1)$ the law of $X_t \mid (X_0, X_1) = (x_0, x_1)$, we have

$$p_t^{\text{cond}}(\cdot \mid x_0, x_1) = \delta_{(1-t)x_0 + tx_1}. \quad (47)$$

In particular, $p_0^{\text{cond}}(\cdot \mid x_0, x_1) = \delta_{x_0}$, $p_1^{\text{cond}}(\cdot \mid x_0, x_1) = \delta_{x_1}$.

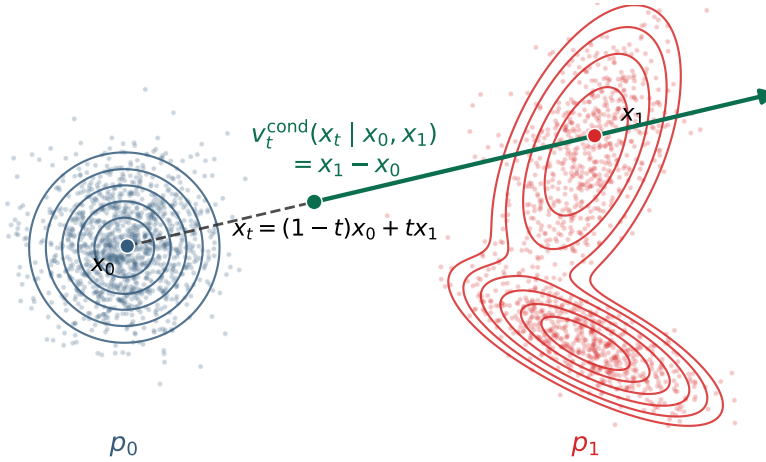


Figure 10: Conditionally on (x_0, x_1) , the interpolation follows a straight segment with constant velocity.

Figure 10 shows this conditional construction for one pair of endpoints. These straight segments are a convenient construction used to define the training probability path. They should not yet be interpreted as the trajectories of the final generative ODE.

4.3.3 Conditional velocity

For fixed endpoints (x_0, x_1) , differentiate the interpolation

$$x_t = (1 - t)x_0 + tx_1 \quad (48)$$

with respect to time: $\frac{dx_t}{dt} = x_1 - x_0$. We therefore associate with the pair (x_0, x_1) the conditional velocity field

$$v_t^{\text{cond}}(x \mid x_0, x_1) = x_1 - x_0.$$

This particular extension is constant in both space and time and satisfies

$$\frac{dx_t}{dt} = v_t^{\text{cond}}(x_t \mid x_0, x_1). \quad (49)$$

Conditionally on $(X_0, X_1) = (x_0, x_1)$, the probability path is the Dirac measure $p_t^{\text{cond}}(\cdot \mid x_0, x_1) = \delta_{x_t}$. Since it does not have a density with respect to the Lebesgue measure, the continuity equation is understood in the weak sense. Let $\phi \in \mathcal{C}_c^\infty(\mathbb{R}^d)$. Then

$$\begin{aligned}
\frac{d}{dt} \int_{\mathbb{R}^d} \phi(x) p_t^{\text{cond}}(dx \mid x_0, x_1) &= \frac{d}{dt} \phi(x_t) \\
&= \nabla \phi(x_t)^\top \frac{dx_t}{dt} \\
&= \nabla \phi(x_t)^\top (x_1 - x_0) \\
&= \int_{\mathbb{R}^d} \nabla \phi(x)^\top v_t^{\text{cond}}(x \mid x_0, x_1) p_t^{\text{cond}}(dx \mid x_0, x_1).
\end{aligned} \tag{50}$$

This is exactly the weak formulation of the continuity equation for the couple $(p_t^{\text{cond}}, v_t^{\text{cond}})$. Thus, for every pair (x_0, x_1) , the straight-line interpolation and the constant velocity $x_1 - x_0$ define a valid conditional transport from δ_{x_0} to δ_{x_1} .

Remark 4. Without conditioning on (X_0, X_1) , the random quantity $X_1 - X_0$ is not a deterministic velocity field as a function of the current position X_t alone. Indeed, the same point x at time t may arise from several different pairs (x_0, x_1) , with different velocities $x_1 - x_0$.

Remark 5 (Another possible conditioning). Instead of conditioning on both endpoints, one can condition only on the target point $X_1 = x_1$. In this case, $X_t = (1 - t)X_0 + tx_1$, $X_0 \sim \mathcal{N}(0, I_d)$, remains random conditionally on $X_1 = x_1$, and $X_t \mid X_1 = x_1 \sim \mathcal{N}(tx_1, (1 - t)^2 I_d)$. The corresponding conditional velocity field is

$$v_t^{\text{cond}}(x \mid x_1) = \frac{x_1 - x}{1 - t}. \tag{51}$$

Here we condition on both (X_0, X_1) because it makes the training construction especially easy: the interpolation is linear and its conditional velocity is $x_1 - x_0$.

4.4 Flow Matching

4.4.1 Reminder on conditional expectation

Definition 6. Let X and Y be two random variables with joint density $f_{X,Y}$. For any y such that $f_{Y(y)} > 0$, the conditional density is

$$f_{X \mid Y=y}(x) = \frac{f_{X,Y}(x, y)}{f_{Y(y)}}. \tag{52}$$

The conditional expectation of X given $Y = y$ is

$$\mathbb{E}[X \mid Y = y] = \int x f_{X \mid Y=y}(x) dx =: \phi(y). \tag{53}$$

The conditional expectation of X given Y is the random variable

$$\mathbb{E}[X \mid Y] = \phi(Y). \tag{54}$$

Proposition 7 (Least-squares characterization). Let $X \in L^2$. Then

$$\mathbb{E}[X \mid Y] \in \operatorname{argmin}_{Z \text{ } \sigma(Y)\text{-measurable}} \mathbb{E}[\|X - Z\|^2]. \tag{55}$$

Equivalently, writing $Z = g(Y)$,

$$g^* \in \operatorname{argmin}_g \mathbb{E}[\|X - g(Y)\|^2], \quad g^*(Y) = \mathbb{E}[X \mid Y]. \tag{56}$$

Pointwise, for almost every y , $\mathbb{E}[X \mid Y = y] \in \operatorname{argmin}_a \mathbb{E}[\|X - a\|^2 \mid Y = y]$.

Useful properties are

$$\mathbb{E}[\mathbb{E}[X \mid Y]] = \mathbb{E}[X] \quad (57)$$

and, for any measurable h , $\mathbb{E}[h(Y)X \mid Y] = h(Y)\mathbb{E}[X \mid Y]$.

In particular, if X is $\sigma(Y)$ -measurable, then $\mathbb{E}[X \mid Y] = X$.

4.4.2 The marginal velocity field

For a random pair (X_0, X_1) , define the random velocity along its conditional interpolation by

$$U_t := v_t^{\text{cond}}(X_t \mid X_0, X_1). \quad (58)$$

The quantity U_t is random because the endpoints (X_0, X_1) are random. Moreover, it is generally not determined by the current position X_t alone.

To obtain a deterministic velocity field depending only on the current position, we average the possible conditional velocities among trajectories that pass through the same point x :

$$v_t^*(x) = \mathbb{E}[U_t \mid X_t = x].$$

Equivalently, the expectation is taken over the random endpoints (X_0, X_1) under their conditional law given $X_t = x$.

For the linear interpolation $X_t = (1 - t)X_0 + tX_1$,

$$U_t = X_1 - X_0 \quad (59)$$

, so

$$v_t^*(x) = \mathbb{E}[X_1 - X_0 \mid X_t = x].$$

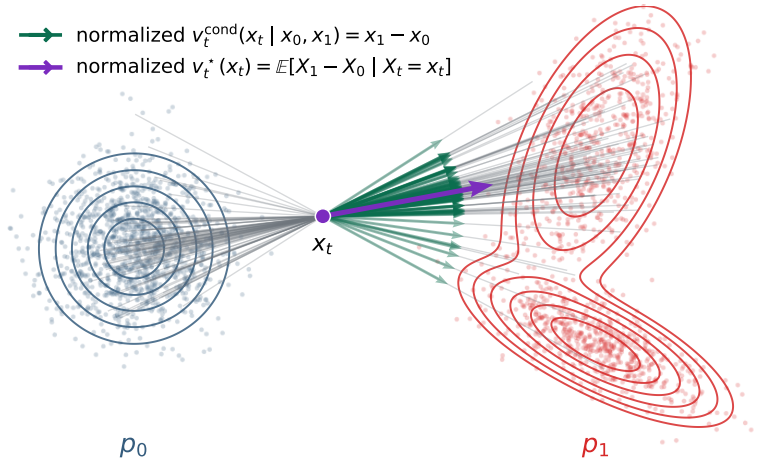


Figure 11: Several conditional trajectories can pass through the same point with different velocities. The marginal field averages these velocities conditionally on the current position.

Figure 11 illustrates why the conditional velocity must be averaged: several endpoint pairs can pass through the same current point with different conditional velocities.

Fact. The pair (p_t, v_t^*) defines a valid transport:

$$\partial_t p_t + \nabla \cdot (p_t v_t^*) = 0.$$

Thus v_t^* is a velocity field whose flow has the prescribed marginals $p_t = \text{Law}(X_t)$.

4.4.3 The Flow Matching objective

The marginal velocity v_t^* is generally not available in closed form. However, by the least-squares characterization of conditional expectation, for every fixed t ,

$$v_t^* = \operatorname{argmin}_v \mathbb{E}_{t, X_0, X_1} \left[\|v(X_t) - v_t^{\text{cond}}(X_t \mid X_0, X_1)\|^2 \right] \quad (60)$$

where the expectation is taken on $X_0 \sim p_0$, $X_1 \sim p_1$, and $t \sim \mathcal{U}([0, 1])$. We approximate the time-dependent velocity field by a neural network $v_\theta(x, t)$ and minimize the Conditional Flow Matching objective

$$\mathcal{L}_{\text{CFM}(\theta)} = \mathbb{E}_{t, X_0, X_1} \left[\|v_\theta(X_t, t) - v_t^{\text{cond}}(X_t \mid X_0, X_1)\|^2 \right] \quad (61)$$

For the linear interpolation $X_t = (1 - t)X_0 + tX_1$, the conditional velocity is $X_1 - X_0$. Hence

$$\mathcal{L}_{\text{FM}(\theta)} = \mathbb{E}_{t, X_0, X_1} \left[\|v_\theta(X_t, t) - (X_1 - X_0)\|^2 \right], \quad X_t = (1 - t)X_0 + tX_1 \quad (62)$$

The important point is that the target $X_1 - X_0$ is available for every sampled training pair, even though the marginal velocity $v_t^*(x)$ itself is unknown.

4.4.4 Training algorithm

A stochastic-gradient step can be implemented as follows:

1. Sample a data point x_1 from the training dataset.
2. Sample $x_0 \sim \mathcal{N}(0, I_d)$.
3. Sample $t \sim \mathcal{U}([0, 1])$.
4. Construct $x_t = (1 - t)x_0 + tx_1$.
5. Minimize the sample loss $\|v_\theta(x_t, t) - (x_1 - x_0)\|^2$.

Training therefore requires only samples from p_0 and from the dataset. Neither the density p_{data} nor the marginal velocity v_t^* needs to be evaluated. Moreover, the generative ODE does not need to be numerically solved during training.

4.4.5 Sampling from the learned flow (post training)

At generation time, the target endpoint X_1 is unknown. We sample only an initial point from p_0 and evolve it according to the learned *marginal* velocity field.

To generate a sample, first draw $z_0 \sim p_0$ and solve

$$\frac{d}{dt} z_t = v_\theta(z_t, t), \quad t \in [0, 1] \quad (63)$$

. With the explicit Euler scheme, $t_k = \frac{k}{K}$ and $\Delta t = \frac{1}{K}$,

$$z_{k+1} = z_k + \Delta t v_\theta(z_k, t_k).$$

If v_θ approximates v^* well and the numerical discretization is sufficiently fine, the final point z_K is approximately distributed according to p_{data} .

Remark 8. The straight training interpolation $t \mapsto (1 - t)x_0 + tx_1$ and a generative trajectory of the learned ODE are generally different curves. In the ideal case $v_\theta = v^*$, however, both constructions induce the same marginal distributions $(p_t)_{t \in [0, 1]}$.

This distinction is visible in the two-moons experiment of Figure 12; the corresponding marginals are compared in Figure 13.

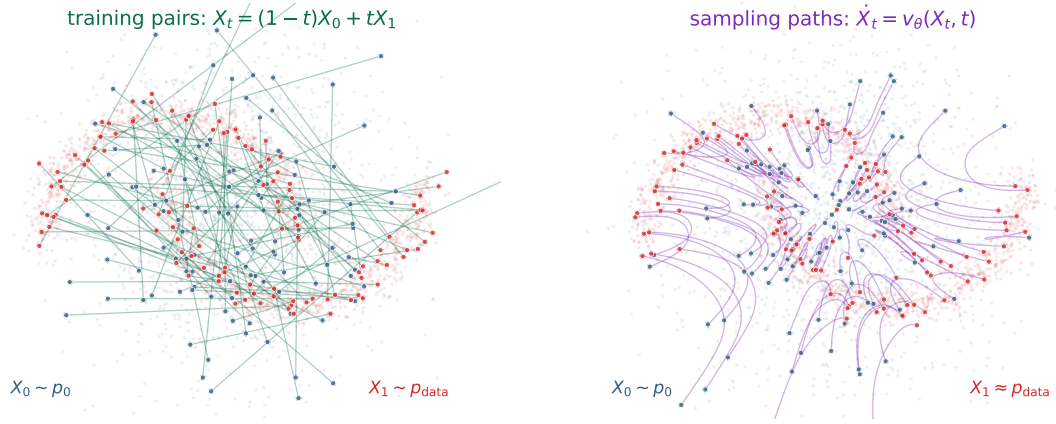


Figure 12: The conditional straight lines used during training are not the particle trajectories of the generative ODE. Flow Matching learns a marginal velocity field whose flow reproduces the same probability path.

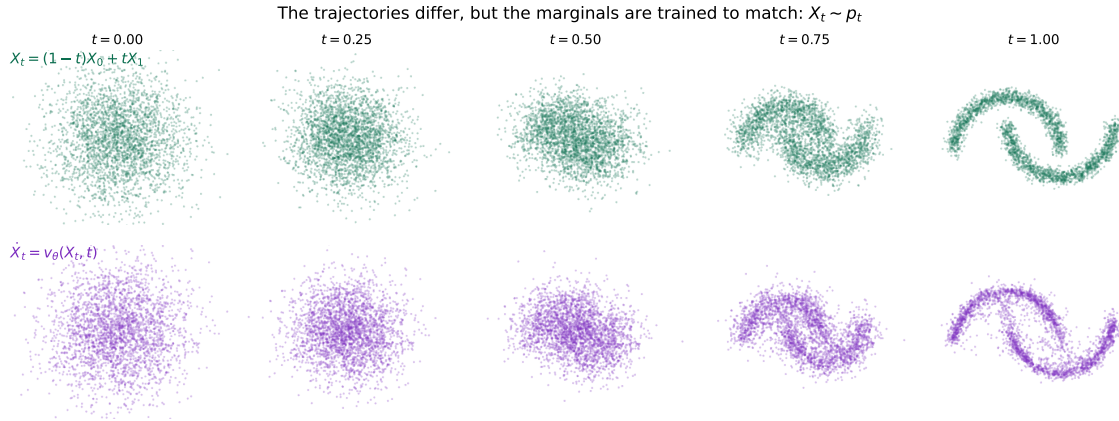


Figure 13: Although the particle trajectories differ, the prescribed interpolation and the learned generative ODE are trained to have matching marginals along time.

5 Diffusion Models

Some references.

- Seminal paper (DDPM) (Ho et al., 2020)
- Denoising score matching (i.e. the continuous point of view on diffusion) (Song et al., 2021)
- Yang Song's blog post: <https://yang-song.net/blog/2021/score/> (Song & Ermon, 2019)

Goal. Given independent samples $x_1, \dots, x_n \sim p_{\text{data}}$, we would like to generate new samples from the same unknown distribution. The basic idea of diffusion models is to define an easy stochastic process that progressively destroys the structure of the data,

$$\text{data} \rightarrow \text{noise}, \quad (64)$$

and then learn how to reverse this process.

Throughout the course we use the standard diffusion convention

$$X_0 \sim p_{\text{data}}, \quad X_T \approx p_{\text{prior}}, \quad (65)$$

so the forward process goes from data at $t = 0$ to noise at $t = T$, while generation runs from T back to 0.

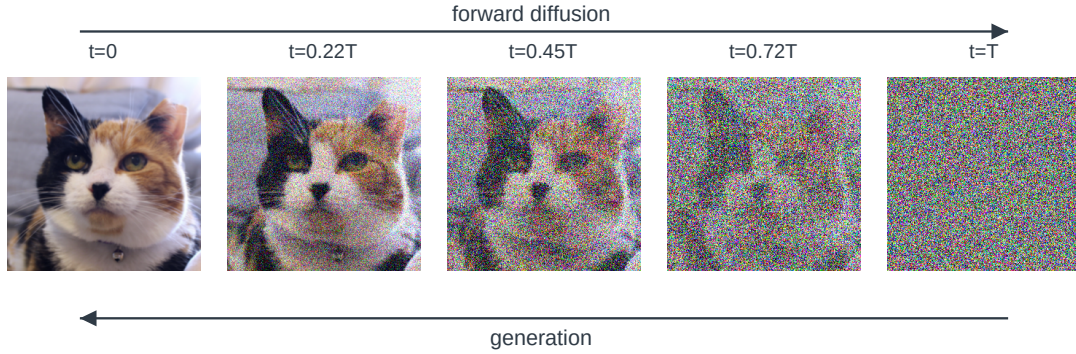


Figure 14: The forward noising idea applied to an image: the cat is progressively corrupted until only Gaussian noise remains, while generation follows the reverse direction.

5.1 Diffusion as a stochastic differential equation

5.1.1 The forward equation

Let $(X_t)_{t \in [0, T]}$ be an $\mathbb{R}^d$ -valued stochastic process. A diffusion process is described by the stochastic differential equation

$$dX_t = f(X_t, t) dt + g(t) dW_t, \quad t \in [0, T]. \quad (66)$$

where $f: \mathbb{R}^d \times [0, T] \rightarrow \mathbb{R}^d$ is the *drift*, and $g: [0, T] \rightarrow \mathbb{R}_+$ is the *diffusion coefficient*. The process $(W_t)_{t \in [0, T]}$ is a standard Brownian motion in $\mathbb{R}^d$: $W_0 = 0$, its increments are independent, and for $0 \leq s < t \leq T$,

$$W_t - W_s \sim \mathcal{N}(0, (t - s)I_d). \quad (67)$$

Goal of the section: Understand the meaning of the SDE and the physical behavior it describes.

5.1.2 Discretization and evolution of particles

Over a small step Δt ,

$$W_{t+\Delta t} - W_t \sim \sqrt{\Delta t} \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I_d). \quad (68)$$

The simplest discretization of Equation 66 is Euler–Maruyama:

$$X_{t+\Delta t} \approx X_t + f(X_t, t)\Delta t + g(t)\sqrt{\Delta t}\varepsilon. \quad (69)$$

Compared with an ODE, even if X_t is known, the next state remains random. See Figure Figure 15.

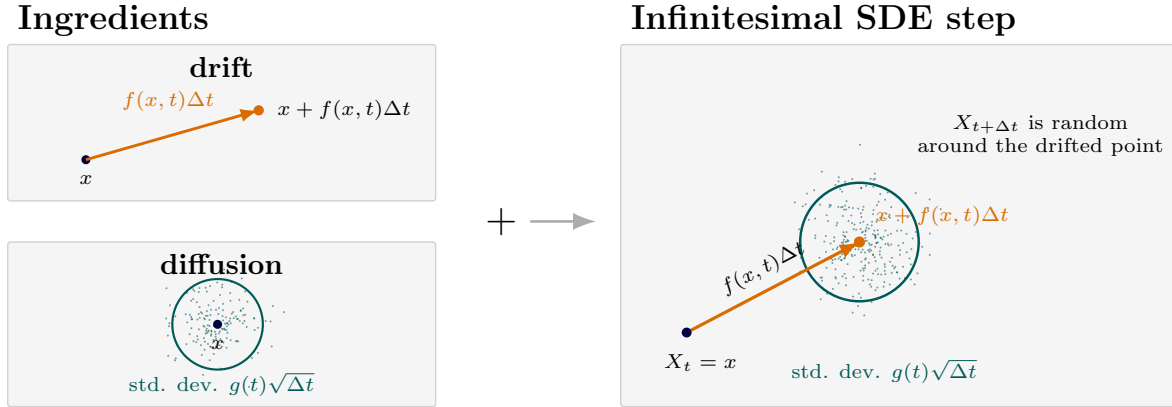


Figure 15: One infinitesimal SDE step. The drift and diffusion effects are shown separately on the left; adding them gives a Gaussian cloud centered at $x + f(x, t)\Delta t$ with standard deviation $g(t)\sqrt{\Delta t}$.

Remark. Notice the different scalings over one small step: the drift displacement is of order Δt , while the Brownian displacement is of order $\sqrt{\Delta t}$. Thus sample paths are locally rough: at very small scale, the random fluctuation dominates the deterministic displacement. The drift nevertheless has a systematic effect on the evolution of the distribution over finite time intervals.

5.1.3 Evolution of the density

For each time $t \in [0, T]$, we denote by p_t the density of X_t . Thus, the forward SDE defines a family of distributions

$$(p_t)_{t \in [0, T]}, \quad (70)$$

which forms a probability path from the data distribution toward a simple noise distribution.

Theorem 9 (Fokker–Planck equation). Under suitable regularity assumptions, the density p_t satisfies

$$\partial_t p_t(x) = -\nabla \cdot (f(x, t)p_t(x)) + \frac{1}{2}g(t)^2 \Delta p_t(x). \quad (71)$$

Conversely, a probability density solving Equation 71 can be realized as the marginal densities of a process satisfying the corresponding SDE.

Two important special cases help interpret this equation. If there is no diffusion term, that is, if $g = 0$, then

$$\partial_t p_t(x) = -\nabla \cdot (f(x, t)p_t(x)), \quad (72)$$

which is the continuity equation: the density is transported by the drift.

If there is no drift term, that is, if $f = 0$, then

$$\partial_t p_t(x) = \frac{1}{2}g(t)^2 \Delta p_t(x), \quad (73)$$

which is a heat equation: the density is progressively smoothed and spread out by the Brownian noise. In the general case, the drift transports probability mass, while the Brownian term simultaneously spreads it. **Ultimately, the distribution collapses pointwise to zero in this example (it's not always the case as we will see).**

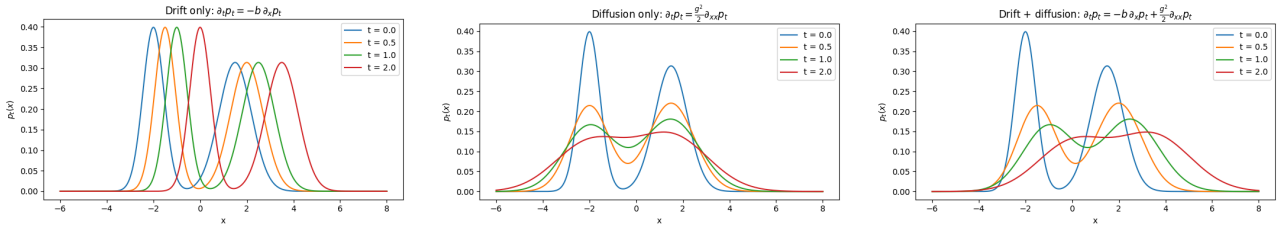


Figure 16: Illustration of the Fokker–Planck equation in one dimension. Left: when $g = 0$, the density is transported by the drift. Middle: when $f = 0$, the density evolves according to a heat equation and spreads out over time. Right: in the general case, drift and diffusion act simultaneously, so the density is both transported and smoothed.

Exercise 10 (Drift, diffusion, and the evolution of a density). Consider the one-dimensional SDE

$$dX_t = b dt + g dW_t, \quad (74)$$

where $b \in \mathbb{R}$ and $g \geq 0$ are constants, and assume $X_0 \sim p_0$.

1. Show that

$$X_t = X_0 + bt + gW_t. \quad (75)$$

2. Using $W_t \sim \mathcal{N}(0, t)$, show that, conditionally on $X_0 = x_0$,

$$X_t \mid X_0 = x_0 \sim \mathcal{N}(x_0 + bt, g^2 t). \quad (76)$$

Deduce that the density of X_t is

$$p_t(x) = \int_{\mathbb{R}} p_0(x_0) \frac{1}{\sqrt{2\pi g^2 t}} \exp\left(-\frac{(x - x_0 - bt)^2}{2g^2 t}\right) dx_0. \quad (77)$$

Equivalently,

$$p_t(x) = (p_0 * \mathcal{N}(0, g^2 t))(x - bt). \quad (78)$$

3. Interpret this formula in the two particular cases: $g = 0$; and $b = 0$.

Solution. Integrating the SDE gives

$$X_t - X_0 = \int_0^t b ds + \int_0^t g dW_s, \quad (79)$$

hence

$$X_t = X_0 + bt + gW_t. \quad (80)$$

Since

$$W_t \sim \mathcal{N}(0, t), \quad (81)$$

we have, conditionally on $X_0 = x_0$,

$$X_t = x_0 + bt + gW_t \sim \mathcal{N}(x_0 + bt, g^2 t). \quad (82)$$

Therefore, by the law of total probability,

$$\begin{aligned}
p_t(x) &= \int_{\mathbb{R}} p(x \mid x_0) p_0(x_0) dx_0 \\
&= \int_{\mathbb{R}} p_0(x_0) \frac{1}{\sqrt{2\pi g^2 t}} \exp\left(-\frac{(x - x_0 - bt)^2}{2g^2 t}\right) dx_0.
\end{aligned} \tag{83}$$

This can be written as

$$p_t(x) = (p_0 * \mathcal{N}(0, g^2 t))(x - bt). \tag{84}$$

Thus the two effects are clearly separated.

If $g = 0$,

$$X_t = X_0 + bt, \tag{85}$$

so

$$p_t(x) = p_0(x - bt). \tag{86}$$

The density is simply translated by the drift.

If $b = 0$,

$$X_t = X_0 + gW_t, \tag{87}$$

and

$$p_t = p_0 * \mathcal{N}(0, g^2 t). \tag{88}$$

The density is progressively smoothed by a Gaussian kernel. This is exactly the solution of the heat equation

$$\partial_t p_t = \frac{g^2}{2} \partial_{xx} p_t. \tag{89}$$

When both b and g are nonzero, the density is simultaneously translated and smoothed.

5.2 Choosing a forward diffusion

In diffusion models, f and g are chosen carefully so that the forward process approaches a prescribed noise distribution, often $\mathcal{N}(0, I_d)$.

5.2.1 Linear diffusions and transition kernels

A particularly convenient choice is a drift that is linear in x :

$$f(x, t) = a(t)x, \tag{90}$$

where $a : [0, T] \rightarrow \mathbb{R}$. The forward SDE becomes

$$dX_t = a(t)X_t dt + g(t) dW_t. \tag{91}$$

Stochastic calculus toolbox. We will only need the following two rules.

1. Product rule with a deterministic function. If $h(t)$ is deterministic, then

$$d(h(t)X_t) = h(t) dX_t + \dot{h}(t)X_t dt. \tag{92}$$

2. Gaussian stochastic integrals. If $h : [0, t] \rightarrow \mathbb{R}$ is deterministic, then

$$\int_0^t h(s) dW_s \sim \mathcal{N}\left(0, \left[\int_0^t h(s)^2 ds\right] I_d\right). \quad (93)$$

We use these rules as a toolbox; no knowledge of stochastic calculus is required for the rest of the course.

Proposition 11 (Conditional law of a linear diffusion). Consider the linear SDE

$$dX_t = a(t)X_t dt + g(t) dW_t. \quad (94)$$

Then, conditionally on $X_0 = x_0$,

$$X_t \mid X_0 = x_0 \sim \mathcal{N}(\alpha_t x_0, \sigma_t^2 I_d), \quad (95)$$

where

$$\sigma_t^2 = \alpha_t^2 \int_0^t g \frac{(s)^2}{\alpha_s^2} ds, \quad (96)$$

and

$$\alpha_t = \exp\left(\int_0^t a(s) ds\right), \quad \dot{\alpha}_t = a(t)\alpha_t, \quad \alpha_0 = 1. \quad (97)$$

Proof. As for an ordinary linear differential equation, we use an integrating factor and consider $\frac{X_t}{\alpha_t}$. Using Rule 1,

$$\begin{aligned} d\left(\frac{X_t}{\alpha_t}\right) &= \frac{1}{\alpha_t} dX_t - \frac{\dot{\alpha}_t}{\alpha_t^2} X_t dt \\ &= g \frac{t}{\alpha_t} dW_t, \end{aligned} \quad (98)$$

since $\dot{\alpha}_t = a(t)\alpha_t$ and the two drift terms cancel.

Integrating from 0 to t gives

$$X_t = \alpha_t X_0 + \alpha_t \int_0^t g \frac{s}{\alpha_s} dW_s. \quad (99)$$

By Rule 2,

$$\int_0^t g \frac{s}{\alpha_s} dW_s \sim \mathcal{N}\left(0, \left[\int_0^t g \frac{(s)^2}{\alpha_s^2} ds\right] I_d\right). \quad (100)$$

Hence, conditionally on $X_0 = x_0$, the random variable X_t is Gaussian with mean $\alpha_t x_0$ and covariance $\sigma_t^2 I_d$, where σ_t^2 is given by Equation 96.

The corresponding conditional density

$$p_{X_t \mid X_0=x_0}(x) \quad (101)$$

is called the *transition kernel* from time 0 to time t .

The marginal density at time t is therefore obtained by averaging over the initial distribution:

$$p_t(x) = \int p_{X_t | X_0=x_0}(x) p_0(x_0) dx_0. \quad (102)$$

Important consequence. To sample directly from the marginal distribution p_t , sample independently

$$X_0 \sim p_0, \quad \varepsilon \sim \mathcal{N}(0, I_d), \quad (103)$$

and define

$$X_t = \alpha_t X_0 + \sigma_t \varepsilon. \quad (104)$$

Then

$$X_t \sim p_t. \quad (105)$$

Thus, to generate a sample at any noise level t , we only need to sample a data point X_0 and Gaussian noise ε ; there is no need to numerically simulate all the intermediate steps of the forward SDE.

5.2.2 Examples of schedules

Now that we have an expression of p_t , we can choose the schedules α_t and σ_t in

$$X_t = \alpha_t X_0 + \sigma_t \varepsilon \quad (106)$$

so that when t becomes large, $p_t \approx \mathcal{N}(0, I_d)$, or at least a Gaussian.

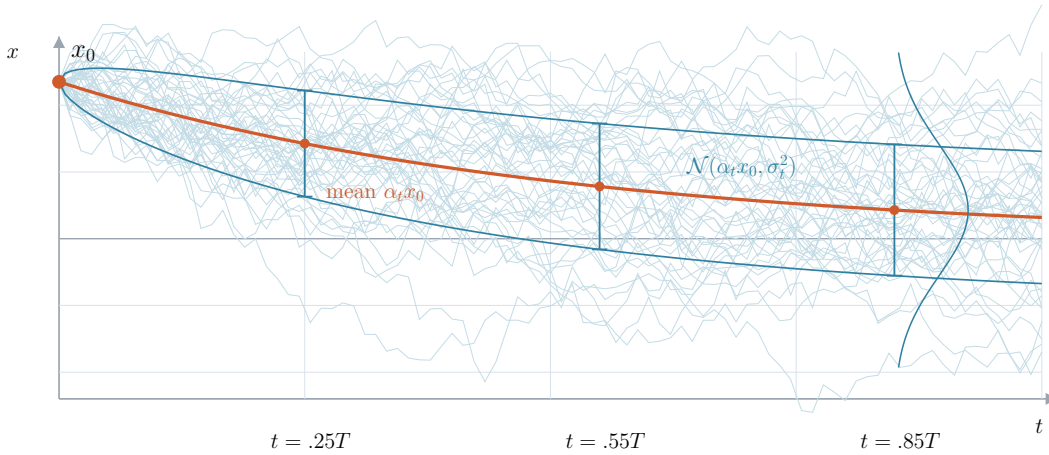


Figure 17: Conditional corruption in a one-dimensional toy Gaussian example, using the VP schedule $\alpha_t = \exp(-\beta \frac{t}{2})$ and $\sigma_t^2 = 1 - \exp(-\beta t)$ with $\beta = 4$ and $T = 1$. Many forward trajectories start from the same x_0 ; at time t , their conditional distribution is centered at $\alpha_t x_0$ with standard deviation σ_t .

Variance Exploding (VE). This corresponds to choosing $\alpha_t = 1$. Then $a(t) = 0$ and $g(t)^2 = \frac{d}{dt} \sigma_t^2$, so

$$dX_t = \sqrt{\frac{d}{dt} \sigma_t^2} dW_t. \quad (107)$$

A common choice is an exponentially increasing noise schedule:

$$\sigma_t^2 = \sigma_{\min}^2 \left[\left(\frac{\sigma_{\max}}{\sigma_{\min}} \right)^{2 \frac{t}{T}} - 1 \right], \quad \sigma_0 = 0. \quad (108)$$

Then

$$p_T = p_{\text{data}} * \mathcal{N}(0, \sigma_T^2 I_d). \quad (109)$$

For $\sigma_{\max} \gg \sigma_{\min}$, we have $\sigma_T^2 \approx \sigma_{\max}^2$, so p_T is well approximated by a wide Gaussian.

Variance Preserving (VP). A VP schedule imposes

$$\alpha_t^2 + \sigma_t^2 = 1, \quad (110)$$

with $\alpha_0 = 1$ and $\sigma_0 = 0$. A common parametrization is

$$\alpha_t = \exp\left(-\frac{1}{2} \int_0^t \beta(s) ds\right), \quad \sigma_t^2 = 1 - \alpha_t^2, \quad (111)$$

where $\beta(t) \geq 0$ controls how fast the signal is replaced by noise. The schedule is chosen so that $\alpha_T \approx 0$ and $\sigma_T \approx 1$, hence

$$X_T = \alpha_T X_0 + \sigma_T \varepsilon \approx \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I_d), \quad (112)$$

and therefore $p_T \approx \mathcal{N}(0, I_d)$.

The corresponding SDE is

$$dX_t = -\frac{1}{2}\beta(t)X_t dt + \sqrt{\beta(t)} dW_t. \quad (113)$$

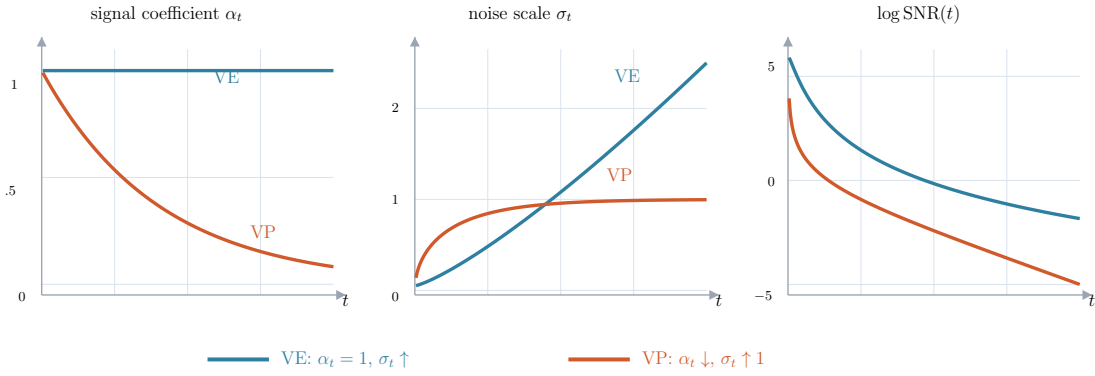


Figure 18: Comparison of illustrative VE and VP schedules. For VE, $\alpha_t = 1$ and $\sigma_t = 0.05 + (2.5 - 0.05)t^{1.25}$. For VP, a constant $\beta = 5$ gives $\alpha_t = \exp(-5\frac{t}{2})$ and $\sigma_t = (1 - \exp(-5t))^{\frac{1}{2}}$. In both cases, the signal-to-noise ratio (SNR) (t) = $\frac{\alpha_t^2}{\sigma_t^2}$ decreases along the forward process.

5.2.3 When do schedules (α_t, σ_t) define a valid linear SDE?

Suppose we want to start directly from

$$X_t = \alpha_t X_0 + \sigma_t \varepsilon, \quad (114)$$

with $\alpha_0 = 1$, $\sigma_0 = 0$, and $\alpha_t > 0$. If α_t and σ_t^2 are differentiable (as functions of time), then $a(t) = \frac{\dot{\alpha}_t}{\alpha_t}$. Differentiating Equation 96 gives $\frac{d}{dt}\sigma_t^2 = 2a(t)\sigma_t^2 + g(t)^2$, hence

$$g(t)^2 = \frac{d}{dt}\sigma_t^2 - 2\frac{\dot{\alpha}_t}{\alpha_t}\sigma_t^2. \quad (115)$$

Therefore the schedule must satisfy

$$\frac{d}{dt}\sigma_t^2 - 2\frac{\dot{\alpha}_t}{\alpha_t}\sigma_t^2 \geq 0. \quad (116)$$

Under standard regularity assumptions, this gives the diffusion coefficient of the corresponding linear SDE.

Remark (Flow matching schedules). Common flow matching probability paths can also be written in the form

$$X_t = \alpha_t X_0 + \sigma_t \varepsilon, \quad (117)$$

after orienting time from data to noise. For example, the straight interpolation schedule is

$$\alpha_t = 1 - t, \quad \sigma_t = t, \quad 0 \leq t < 1. \quad (118)$$

Then Equation 115 gives

$$g(t)^2 = 2t + 2\frac{t^2}{1-t} = 2\frac{t}{1-t} \geq 0. \quad (119)$$

Thus this schedule satisfies the compatibility condition on every interval $[0, T]$ with $T < 1$.

However, the corresponding SDE becomes singular as $t \rightarrow 1$, since

$$\frac{\dot{\alpha}_t}{\alpha_t} = -\frac{1}{1-t}, \quad g(t)^2 = 2\frac{t}{1-t}, \quad (120)$$

both diverge. Therefore the standard regularity assumptions (that we don't detail here) ensuring existence of a well-behaved diffusion on the closed interval $[0, 1]$ are no longer satisfied.

5.3 The backward process

We have designed a forward process

$$X_0 \sim p_{\text{data}} \rightarrow X_T \approx p_{\text{prior}}. \quad (121)$$

For generation, we need to run this process in the opposite direction.

The question is therefore:

Given the forward SDE, what stochastic process transforms p_T back into p_0 ?

5.3.1 What happens when we reverse time?

Define a new process

$$Y_s = X_{T-s}, \quad s \in [0, T]. \quad (122)$$

Its density is

$$q_s(x) = p_{T-s}(x). \quad (123)$$

Differentiating with respect to s ,

$$\partial_s q_s = -\partial_t p_t \big|_{t=T-s}. \quad (124)$$

Using the Fokker–Planck equation,

$$\partial_s q_s = \nabla \cdot (f q_s) - \frac{1}{2} g^2 \Delta q_s, \quad (125)$$

where f and g are evaluated at time $T - s$.

At first sight, this is problematic: the diffusion term now has the wrong sign to be the Fokker–Planck equation of an SDE.

However, we can rewrite part of it as a transport term.

Since

$$\nabla q_s = q_s \nabla \log q_s, \quad (126)$$

we have

$$\Delta q_s = \nabla \cdot (q_s \nabla \log q_s). \quad (127)$$

We can therefore write

$$\begin{aligned} \partial_s q_s &= \nabla \cdot (f q_s) - g^2 \Delta q_s + \frac{1}{2} g^2 \Delta q_s \\ &= -\nabla \cdot [(-f + g^2 \nabla \log q_s) q_s] + \frac{1}{2} g^2 \Delta q_s. \end{aligned} \quad (128)$$

But Equation 128 now has exactly the form of a Fokker–Planck equation.

Hence the reversed process should have drift

$$-f + g^2 \nabla \log q_s. \quad (129)$$

A new quantity has appeared naturally:

$$\nabla \log q_s. \quad (130)$$

Theorem 12 (Reverse-time SDE). Under suitable regularity assumptions, if

$$dX_t = f(X_t, t) dt + g(t) dW_t, \quad (131)$$

then the reversed process

$$Y_s = X_{T-s} \quad (132)$$

satisfies

$$dY_s = [-f(Y_s, T-s) + g(T-s)^2 \nabla \log p_{T-s}(Y_s)] ds + g(T-s) d\bar{W}_s, \quad (133)$$

where $\bar{W}_s$ denotes a Brownian motion (a different one from the one of the forward SDE)

Equivalently, using the usual diffusion notation and keeping the variable t while integrating from T down to 0,

$$dX_t = [f(X_t, t) - g(t)^2 \nabla \log p_t(X_t)] dt + g(t) d\bar{W}_t, \quad t : T \rightarrow 0. \quad (134)$$

5.3.2 The score appears

For a probability density p , the vector field

$$s(x) = \nabla_x \log p(x) \quad (135)$$

is called its *score*.

For the intermediate distributions of the diffusion, we write

$$s_t(x) = \nabla_x \log p_t(x). \quad (136)$$

The score points toward directions in which the log-density increases.

For example, if

$$p(x) = \mathcal{N}(\mu, \sigma^2 I), \quad (137)$$

then

$$\nabla \log p(x) = -\frac{x - \mu}{\sigma^2}. \quad (138)$$

The score points toward the center of the Gaussian.

Score of a Gaussian mixture Consider a Gaussian mixture

$$p(x) = \sum_{k=1}^K \pi_k p_k(x), \quad (139)$$

where p_k denotes the density of the k -th Gaussian component. Then

$$\begin{aligned} \nabla \log p(x) &= \frac{\sum_k \pi_k \nabla p_k(x)}{\sum_j \pi_j p_j(x)} \\ &= \sum_{k=1}^K \frac{\pi_k p_k(x)}{\sum_j \pi_j p_j(x)} \nabla \log p_k(x). \end{aligned} \quad (140)$$

By Bayes' rule,

$$\frac{\pi_k p_k(x)}{\sum_j \pi_j p_j(x)} = \mathbb{P}(K = k \mid X = x), \quad (141)$$

so

$$\nabla \log p(x) = \sum_{k=1}^K \mathbb{P}(K = k \mid X = x) \nabla \log p_k(x). \quad (142)$$

Thus, the score of the mixture is a posterior-weighted average of the scores of its components. Near one mode, the corresponding component typically receives most of the weight, while between modes several components may contribute.

This is a first example of an idea that will appear again in denoising score matching: a marginal score can be obtained by averaging conditional scores.

What did we learn?

To reverse the diffusion, we need to know the score $s_t(x) = \nabla \log p_t(x)$ of every intermediate noisy distribution.

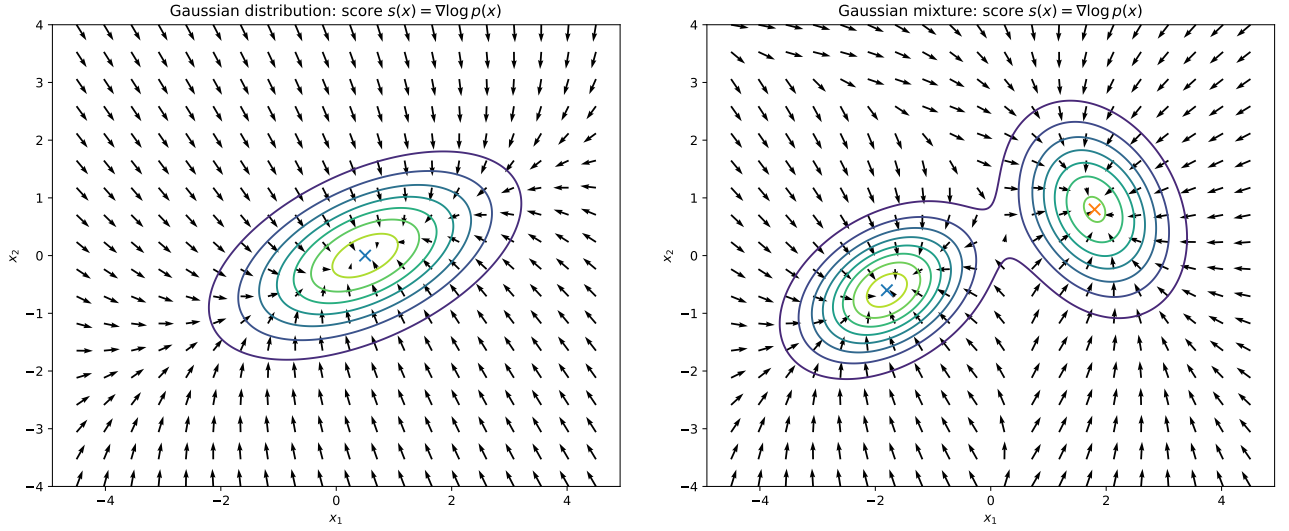


Figure 19: todo

This is the learning problem at the heart of diffusion models.

Marginals versus exact time reversal The Fokker–Planck calculation shows that the reverse SDE above has the desired reversed marginal distributions p_{T-s} . For generative modeling, this is already sufficient: starting from p_T , the dynamics end with distribution $p_0 = p_{\text{data}}$.

A stronger statement is that this SDE is the actual time reversal of the forward stochastic process, i.e. that it also reproduces its full trajectory law backwards in time. This stronger result requires a time-reversal theorem, but is not necessary if our only goal is to sample the correct marginals.

5.4 Learning the score

We train a network $s_\theta(x, t)$ to approximate $\nabla \log p_t(x)$. The ideal MSE target is unknown, but the conditional transition kernel is known.

5.4.1 Denoising score matching

From

$$p_t(x) = \int p_{X_t | X_0=x_0}(x) p_0(x_0) dx_0, \quad (143)$$

differentiate with respect to x :

$$\nabla p_t(x) = \int p_{X_t | X_0=x_0}(x) \nabla_x \log p_{X_t | X_0=x_0}(x) p_0(x_0) dx_0. \quad (144)$$

Dividing by $p_t(x)$ and using Bayes' rule gives

$$\nabla \log p_t(x) = \mathbb{E}_{X_0 | X_t=x} [\nabla_x \log p_{X_t | X_0=x_0}(x)] = \mathbb{E} [\nabla_x \log p_{X_t | X_0=x_0}(x) | X_t = x]. \quad (145)$$

For a fixed time t , consider the regression problem

$$\mathcal{L}_t(s) = \mathbb{E}_{x_0 \sim p_{\text{data}}, x_t \sim p_{X_t | X_0=x_0}} [\|s(x_t, t) - \nabla_{x_t} \log p_{X_t | X_0=x_0}(x_t)\|^2]. \quad (146)$$

The expectation is taken with respect to the joint distribution of (X_0, X_t) : first sample $X_0 \sim p_{\text{data}}$, and then, conditionally on $X_0 = x_0$, sample $X_t \sim p_{X_t | X_0=x_0}$.

Since the minimizer of

$$\mathbb{E} [\|h(X) - Y\|^2] \quad (147)$$

is

$$h^*(x) = \mathbb{E}[Y \mid X = x], \quad (148)$$

the minimizer of $\mathcal{L}_t$ satisfies

$$\begin{aligned} s^*(x, t) &= \mathbb{E}[\nabla_x \log p_{X_t \mid X_0=x_0}(x) \mid X_t = x] \\ &= \nabla \log p_t(x), \end{aligned} \quad (149)$$

where the last equality follows from Equation 145. Thus, regressing onto the known conditional score learns the unknown marginal score.

For the Gaussian corruption considered previously,

$$X_t = \alpha_t X_0 + \sigma_t \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I), \quad (150)$$

sampling

$$X_t \sim p_{X_t \mid X_0=x_0} \quad (151)$$

is equivalent to sampling

$$\varepsilon \sim \mathcal{N}(0, I) \quad (152)$$

and setting

$$X_t = \alpha_t x_0 + \sigma_t \varepsilon. \quad (153)$$

Therefore, the expectation over (X_0, X_t) can equivalently be written as an expectation over (X_0, ε) :

$$\begin{aligned} &\mathbb{E}_{x_0 \sim p_{\text{data}}, x_t \sim p_{X_t \mid X_0=x_0}} \left[\left\| s(x_t, t) - \nabla_{x_t} \log p_{X_t \mid X_0=x_0}(x_t) \right\|^2 \right] \\ &= \mathbb{E}_{x_0 \sim p_{\text{data}}, \varepsilon \sim \mathcal{N}(0, I)} \left[\left\| s(\alpha_t x_0 + \sigma_t \varepsilon, t) - \nabla_{x_t} \log p_{X_t \mid X_0=x_0}(\alpha_t x_0 + \sigma_t \varepsilon) \right\|^2 \right]. \end{aligned} \quad (154)$$

Moreover, since

$$p_{X_t \mid X_0=x_0} = \mathcal{N}(\alpha_t x_0, \sigma_t^2 I), \quad (155)$$

its score is known explicitly:

$$\nabla_{x_t} \log p_{X_t \mid X_0=x_0}(x_t) = -\frac{x_t - \alpha_t x_0}{\sigma_t^2}. \quad (156)$$

Using

$$x_t = \alpha_t x_0 + \sigma_t \varepsilon, \quad (157)$$

this becomes

$$\nabla_{x_t} \log p_{X_t \mid X_0=x_0}(x_t) = -\frac{\varepsilon}{\sigma_t}. \quad (158)$$

Finally, to learn a single network for all noise levels, we also sample t during training:

$$\mathcal{L}_{\text{DSM}}(\theta) = \mathbb{E}_{t, x_0 \sim p_{\text{data}}, \varepsilon \sim \mathcal{N}(0, I)} \left[\left\| s_\theta(\alpha_t x_0 + \sigma_t \varepsilon, t) + \frac{\varepsilon}{\sigma_t} \right\|^2 \right].$$
(159)

Thus, training can be implemented by sampling only a data point x_0 , a time t , and Gaussian noise ε . The MSE against the known *conditional* score nevertheless learns the unknown *marginal* score.

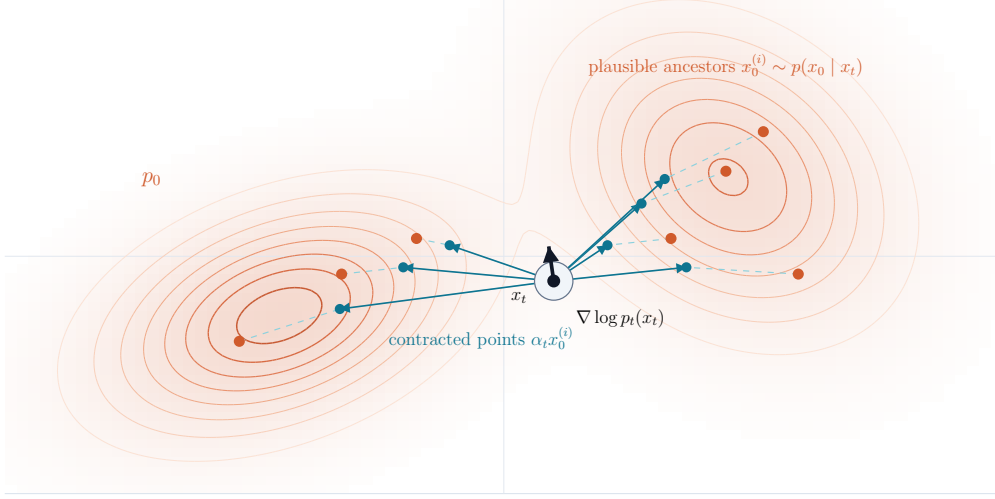


Figure 20: Thin arrows are conditional score targets up to the common factor $\frac{1}{\sigma_t^2}$. MSE regression at the fixed input x_t returns their conditional average, which is the marginal score $\nabla \log p_t(x_t)$.

5.4.2 Gaussian corruption and noise prediction

For

$$X_t = \alpha_t X_0 + \sigma_t \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I), \quad (160)$$

we have

$$p_{X_t|X_0=x_0}(x) = \mathcal{N}(\alpha_t x_0, \sigma_t^2 I), \quad (161)$$

and therefore

$$\nabla_{x_t} \log p_{X_t|X_0=x_0}(x) = -\frac{x_t - \alpha_t x_0}{\sigma_t^2} = -\frac{\varepsilon}{\sigma_t}. \quad (162)$$

If we parameterize

$$s_\theta(x, t) = -\frac{\varepsilon_\theta(x, t)}{\sigma_t}, \quad (163)$$

then, with $\lambda(t) = \sigma_t^2$, denoising score matching becomes

$$\mathcal{L}_\varepsilon(\theta) = \mathbb{E}_{t, X_0, \varepsilon} [\|\varepsilon_\theta(\alpha_t X_0 + \sigma_t \varepsilon, t) - \varepsilon\|^2]. \quad (164)$$

Training algorithm.

1. sample X_0 from the dataset;
2. sample a time t ;
3. sample $\varepsilon \sim \mathcal{N}(0, I)$;
4. set $X_t = \alpha_t X_0 + \sigma_t \varepsilon$;
5. update θ using $\|\varepsilon_\theta(X_t, t) - \varepsilon\|^2$.

5.4.3 Short parenthesis: DDPMs

The original DDPM formulation starts from the discrete Markov chain

$$q(x_k|x_{k-1}) = \mathcal{N}(\sqrt{1 - \beta_k} x_{k-1}, \beta_k I). \quad (165)$$

Let

$$|\alpha|_k = \prod_{j=1}^k (1 - \beta_j). \quad (166)$$

Then

$$q(x_k|x_0) = \mathcal{N}(\sqrt{|\alpha|_k}x_0, (1 - |\alpha|_k)I), \quad (167)$$

so

$$x_k = \sqrt{|\alpha|_k}x_0 + \sqrt{1 - |\alpha|_k}\varepsilon. \quad (168)$$

DDPMs start from a likelihood/variational viewpoint: they parameterize a reverse Markov chain and optimize a variational lower bound. Reparameterizing the reverse mean with a noise predictor leads to weighted ε -prediction losses; the commonly used simplified DDPM objective is the unweighted MSE $\mathbb{E}\|\varepsilon - \varepsilon_\theta(x_k, k)\|^2$.

Thus DDPMs and score-SDE models arrive at essentially the same core learning problem through different derivations. In continuous time, the classical DDPM forward chain corresponds to a VP diffusion.

5.5 Sampling

After training, replace the unknown score in the reverse SDE by s_θ :

$$dX_t = [f(X_t, t) - g(t)^2 s_\theta(X_t, t)] dt + g(t) d\bar{W}_t, \quad t : T \rightarrow 0. \quad (169)$$

Initialize $X_T \sim p_{\text{prior}}$ and integrate backwards.

5.5.1 Euler–Maruyama

For $\Delta t < 0$,

$$X_{t+\Delta t} \approx X_t + [f(X_t, t) - g(t)^2 s_\theta(X_t, t)] \Delta t + g(t) \sqrt{|\Delta t|} \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I). \quad (170)$$

Training can jump directly to a random time t using the transition kernel; sampling must numerically traverse a reverse trajectory.

5.5.2 Deterministic sampling: the probability flow ODE

Starting again from Fokker–Planck,

$$\partial_t p_t = -\nabla \cdot (f p_t) + \frac{1}{2} g^2 \Delta p_t. \quad (171)$$

Since $\nabla p_t = p_t \nabla \log p_t$,

$$\Delta p_t = \nabla \cdot (p_t \nabla \log p_t). \quad (172)$$

Therefore

$$\partial_t p_t = -\nabla \cdot \left[p_t \left(f - \frac{1}{2} g^2 \nabla \log p_t \right) \right]. \quad (173)$$

This is the continuity equation of the ODE

$$\frac{dX_t}{dt} = f(X_t, t) - \frac{1}{2} g(t)^2 \nabla \log p_t(x). \quad (174)$$

This is the *probability flow ODE*. It has the same one-time marginal densities (p_t) as the forward SDE. To generate, replace the score by s_θ and integrate from T to 0.

Remark. The SDE and ODE trajectories are different. The statement is only that they have the same marginal distribution at every fixed time.

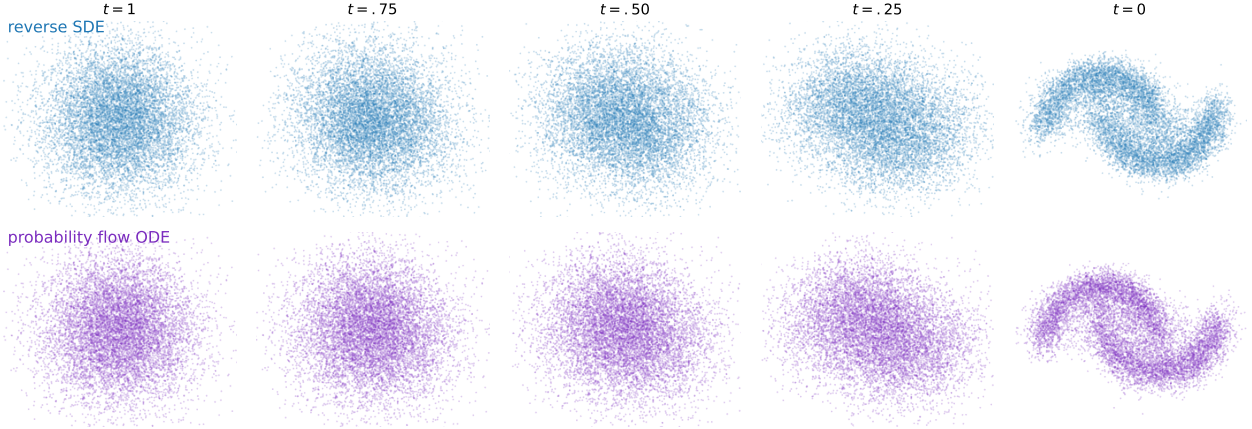


Figure 21: Reverse SDE and probability-flow ODE sampling with the same denoising score. The point clouds show the same marginal distributions at aligned times.

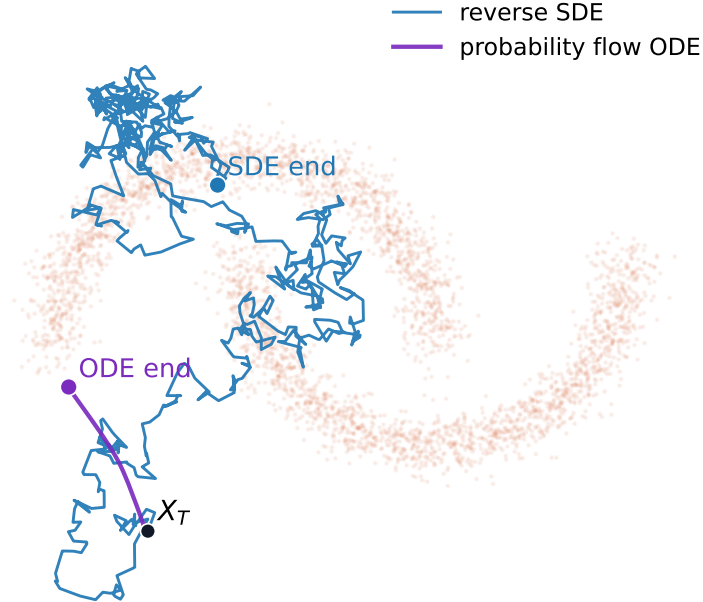


Figure 22: Starting from the same noisy point, the reverse SDE and the probability-flow ODE follow different trajectories and do not end at the same point.

5.6 Diffusion and flow matching: two sides of the same coin

Diffusion models and flow matching are usually introduced from rather different perspectives.

Diffusion. Choose a stochastic forward process that progressively transforms data into noise. Reversing this process introduces the score $\nabla \log p_t$. One then learns the score and generates samples either with the reverse-time SDE or with the probability flow ODE.

Flow matching. Choose a probability path between two distributions and directly learn a velocity field whose ODE transports samples along this path.

Despite these different viewpoints, the two approaches are closely related.

5.6.1 Both methods turn an easy conditional target into a hard marginal field

In diffusion models, the marginal score can be written as the conditional expectation

$$\nabla \log p_t(x) = \mathbb{E}[\nabla_x \log p_{X_t | X_0=x_0}(x) | X_t = x]. \quad (175)$$

The conditional score on the right-hand side is tractable because the corruption kernel $p_{X_t | X_0=x_0}$ is chosen by construction.

Conditional flow matching relies on an analogous identity:

$$v_t(x) = \mathbb{E}[v_t^{\text{cond}} | X_t = x], \quad (176)$$

where v_t^{cond} is the velocity associated with the chosen conditional probability path.

Thus, in both cases, the quantity that we ultimately need is a marginal vector field, which is difficult to compute directly. Instead, we regress onto a tractable conditional target:

$$\text{tractable conditional target} \xrightarrow{\text{MSE regression}} \text{marginal vector field}. \quad (177)$$

5.6.2 Tweedie's formula: the score is a denoiser

A second connection comes from the interpretation of the score as a denoising function. Let

$$Y = X + \sigma\varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I). \quad (178)$$

Then

$$p_Y(y) = \int p_X(x) \varphi_\sigma(y - x) dx, \quad (179)$$

where φ_σ denotes the Gaussian density with covariance $\sigma^2 I$. Differentiating with respect to y ,

$$\begin{aligned} \nabla p_Y(y) &= -\frac{1}{\sigma^2} \int (y - x) p_X(x) \varphi_\sigma(y - x) dx \\ &= -\frac{1}{\sigma^2} \left(y p_Y(y) - \int x p_X(x) \varphi_\sigma(y - x) dx \right). \end{aligned} \quad (180)$$

By Bayes' rule,

$$\frac{p_X(x) \varphi_\sigma(y - x)}{p_Y(y)} = p_{X | Y=y}(x). \quad (181)$$

Dividing by $p_Y(y)$ therefore gives

$$\nabla \log p_Y(y) = \frac{\mathbb{E}[X | Y = y] - y}{\sigma^2}. \quad (182)$$

Hence Tweedie's formula:

$$\mathbb{E}[X | Y = y] = y + \sigma^2 \nabla \log p_Y(y). \quad (183)$$

For the Gaussian corruption

$$X_t = \alpha_t X_0 + \sigma_t \varepsilon, \quad (184)$$

the same identity gives

$$\mathbb{E}[X_0 | X_t = x] = \frac{x + \sigma_t^2 \nabla \log p_t(x)}{\alpha_t}. \quad (185)$$

Therefore, knowing the score is equivalent to knowing the MMSE denoiser. It is also equivalent to predicting the Gaussian noise, since

$$\mathbb{E}[\varepsilon \mid X_t = x] = -\sigma_t \nabla \log p_t(x). \quad (186)$$

Thus score prediction, noise prediction, and denoising are simply different parametrizations of the same conditional information.

5.6.3 For Gaussian source noise, score and velocity are explicitly related

Consider the Gaussian probability path

$$X_t = \alpha_t X_0 + \sigma_t \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I). \quad (187)$$

The corresponding conditional velocity is obtained by differentiating the path:

$$v_t^{\text{cond}} = \dot{\alpha}_t X_0 + \dot{\sigma}_t \varepsilon. \quad (188)$$

Therefore, the marginal velocity field learned by flow matching is

$$v_t(x) = \mathbb{E}[\dot{\alpha}_t X_0 + \dot{\sigma}_t \varepsilon \mid X_t = x]. \quad (189)$$

From Tweedie's formula,

$$\mathbb{E}[X_0 \mid X_t = x] = \frac{x + \sigma_t^2 \nabla \log p_t(x)}{\alpha_t}, \quad (190)$$

and, as seen above,

$$\mathbb{E}[\varepsilon \mid X_t = x] = -\sigma_t \nabla \log p_t(x). \quad (191)$$

Substituting these two identities gives

$$\begin{aligned} v_t(x) &= \dot{\alpha}_t \mathbb{E}[X_0 \mid X_t = x] + \dot{\sigma}_t \mathbb{E}[\varepsilon \mid X_t = x] \\ &= \frac{\dot{\alpha}_t}{\alpha_t} x + \left(\frac{\dot{\alpha}_t}{\alpha_t} \sigma_t^2 - \sigma_t \cdot (\sigma)_t \right) \nabla \log p_t(x). \end{aligned} \quad (192)$$

Hence

$$v_t(x) = \frac{\dot{\alpha}_t}{\alpha_t} x + \left(\frac{\dot{\alpha}_t}{\alpha_t} \sigma_t^2 - \sigma_t \cdot (\sigma)_t \right) \nabla \log p_t(x). \quad (193)$$

If the same Gaussian probability path is generated by the linear diffusion

$$dX_t = a(t)X_t dt + g(t) dW_t, \quad (194)$$

then $a(t) = \frac{\dot{\alpha}_t}{\alpha_t}$ and $g(t)^2 = 2\sigma_t \cdot (\sigma)_t - 2\frac{\dot{\alpha}_t}{\alpha_t}\sigma_t^2$. Consequently, Equation 193 becomes

$$v_t(x) = a(t)x - \frac{1}{2}g(t)^2 \nabla \log p_t(x). \quad (195)$$

This is exactly the velocity field of the probability flow ODE. Thus, for the same Gaussian probability path, flow matching and diffusion lead to the same deterministic transport field: flow matching learns it directly, while diffusion obtains it from the score.

Bibliography

Albergo, M. S., & Vanden-Eijnden, E. (2023,). Building normalizing flows with stochastic interpolants. *International Conference on Learning Representations (ICLR)*.

- Ambrosio, L., Gigli, N., & Savaré, G. (2005). *Gradient flows: in metric spaces and in the space of probability measures*. Springer.
- Gagneux, A., Martin, S. T., Emonet, R., Bertrand, Q., & Massias, M. (2025,). A visual dive into conditional flow matching. *The Fourth Blogpost Track at ICLR 2025*.
- Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (Neurips)*, 33, 6840–6851.
- Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., & Le, M. (2023,). Flow matching for generative modeling. *International Conference on Learning Representations (ICLR)*.
- Liu, X., Gong, C., & Liu, Q. (2023,). Flow straight and fast: Learning to generate and transfer data with rectified flow. *International Conference on Learning Representations (ICLR)*.
- Pierret, E., Tosel, V., Delon, J., & Newson, A. (2026,). *Flow Matching for Applied Mathematicians*.
- Song, Y., & Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32.
- Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., & Poole, B. (2021,). Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations (ICLR)*. <https://openreview.net/forum?id=PXTIG12RRHS>