Generative modelling lecture notes

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
Bibliography 23

1 Probabilistic Thinking (10/09/2026)

Notes by Paul Bouchaud.

Goal of generative modelling: Given iid samples $(x_1, x_2, ..., x_n) \sim p(x)$ 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 wil ldo 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:

$$p(x) = \int_y p(x, y)\,\mathrm{d}y = \int_y p(y)p(x|y)\,\mathrm{d}y \tag{1}$$

1.1.3 Jensen's inequality:

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

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

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:

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

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:

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

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:

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

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

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

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,

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

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 simplifies 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 $x_1, x_2, ..., x_n \in \mathbb{R}^d$ 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.

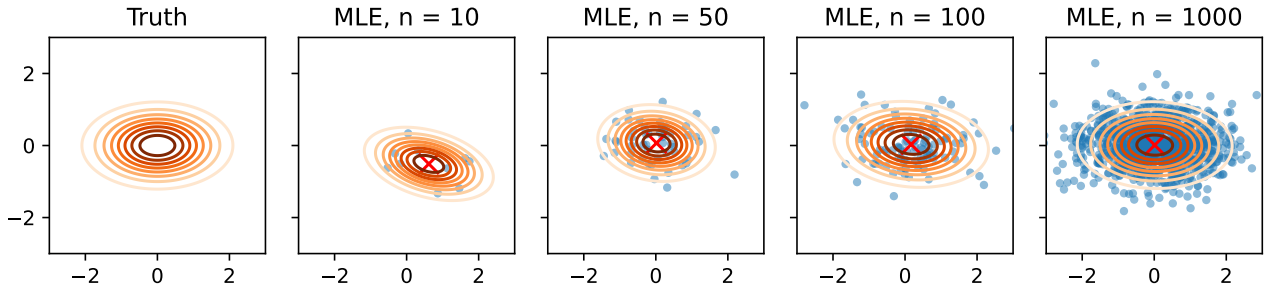

Figure 1: Illustration of MLE for various sample sizes n

From this we derive our simplest generative model:

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

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

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

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 you think there are in the mixture displayed in Figure 2? What are their means and covariances?

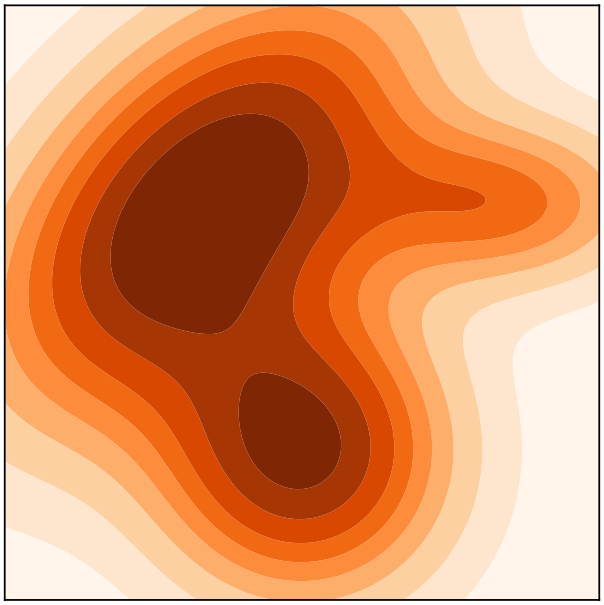

Figure 2: A mixture of Gaussians in 2D

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 $\theta = (\pi_1, ..., \pi_K, \mu_1, ..., \mu_K, \Sigma_1, ..., \Sigma_K)$ that maximizes the (log) likelihood of the data, namely:

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

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 simple 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 $[\![1, k]\!]$ 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 algorithms 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

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

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 botteneck 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:

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

where

$$\log p_\theta(x) = \log\left(\int_z p_\theta(x|z)p(z)\,\mathrm{d}z\right). \tag{20}$$

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:

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

where μ and σ_i are implemented by a neural network of aprameters φ. Here, the Gaussian density is evaluated at z, and the encoder outputs its mean $\mu_{\varphi(x)}$ and covariance 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):

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

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

$$p(x) \geq 0 \quad \wedge \quad \int_K p(x)\,\mathrm{d}x = 1. \tag{23}$$

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

$$\varphi\left(\int_K f(x)p(x)\,\mathrm{d}x\right) \leq \int_K \varphi(f(x))p(x)\,\mathrm{d}x. \tag{24}$$

Equivalently, if X has density p, then

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

2.2.2 Proof of the ELBO

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

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

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 proabilities, the Kullback–Leibler divergence.

2.2.3 Kullback–Leibler divergence

For probability densities q and p, the KL divergence is

$$
\mathrm{KL}(q \parallel p) \coloneqq \int q(z)\log\left(\frac{q(z)}{p(z)}\right)\mathrm{d}z \geq 0. \tag{28}
$$

Using Bayes' rule, the evidence can be decomposed as

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

Therefore,

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

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:

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

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:

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

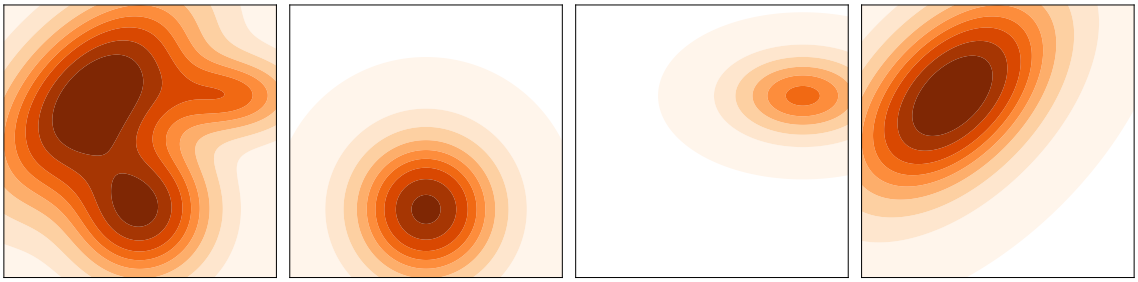

Figure 3: Mixture of Figure 2 decomposed in its 3 components.

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 $G_\theta : \mathbb{R}^d \to \mathbb{R}^m$. The discriminator maps a data point to a probability:

$$D_\varphi : \mathbb{R}^m \to \{0, 1\} \tag{33}$$

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:

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

Taking the logarithm gives:

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

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 simultaneous update is:
$\binom{x_{k+1}}{y_{k+1}} = \binom{x_k}{y_k} + \eta\binom{-\nabla_x \varphi(x_k, y_k)}{\nabla_y \varphi(x_k, y_k)}.$
Since $\nabla_x \varphi(x, y) = y$ and $\nabla_y \varphi(x, y) = x$, this becomes the matrix recurrence:
$\binom{x_{k+1}}{y_{k+1}} = \begin{pmatrix} 1 & -\eta \\ \eta & 1 \end{pmatrix}\binom{x_k}{y_k}.$
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 simultaneous 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

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

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

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

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

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

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

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

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

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

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

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:

$$f_{x'}(d) = 0 \Rightarrow p_{\text{data}(x)}(1 - d) = p_{G(x)} d$$
$$\Rightarrow p_{\text{data}(x)} = d\left(p_{\text{data}(x)} + p_{G(x)}\right)$$
$$\Rightarrow d = \frac{p_{\text{data}(x)}}{p_{\text{data}(x)} + p_{G(x)}}.$$

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:

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

Define the mixture density:

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

The Jensen–Shannon divergence is:

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

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

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

So the GAN objective can be rewritten as:

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

3.4 WGAN (Wasserstein GAN):

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

The Kantorovich–Rubinstein duality gives:

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

where Lip_1 is the set of 1-Lipschitz functions:

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

Using a critic F to represent f:

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

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

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

Therefore, the WGAN objective is:

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

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:

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

Therefore:

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

The WGAN-GP objective is:

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

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

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

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

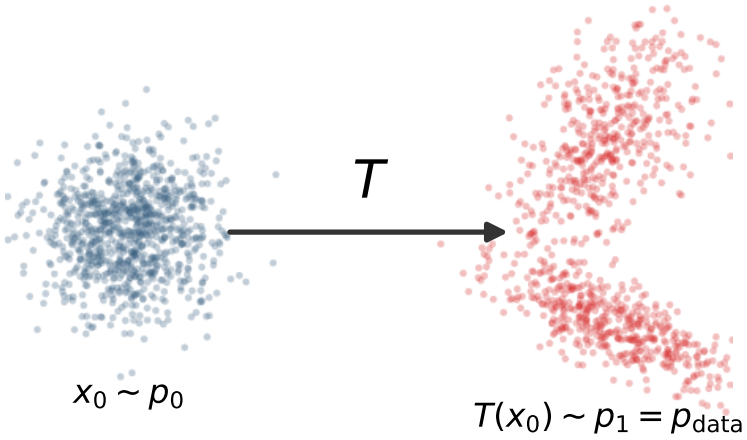

Figure 4: The generative problem can first be seen as learning a transformation from a simple source distribution to the data distribution.

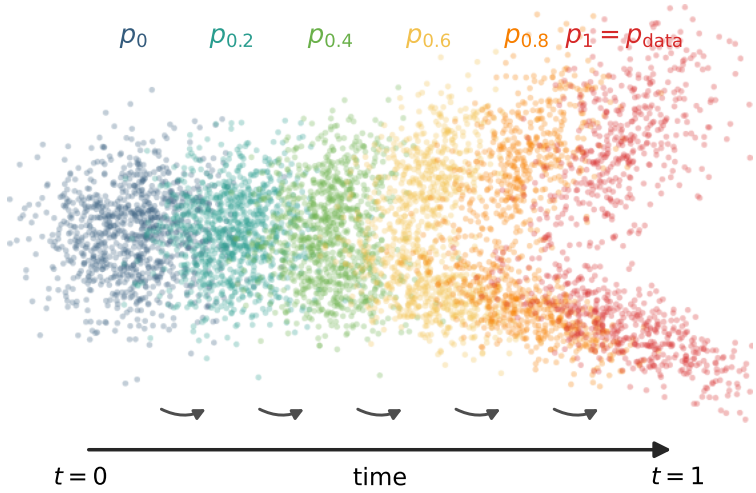

Figure 5: Flow Matching decomposes this transformation into small steps in time.

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 $T : \mathbb{R}^d \to \mathbb{R}^d$ be measurable. The *pushforward* of μ by T, denoted by $T_{\#}\mu$, is the probability measure defined by

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

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 $T : \mathbb{R}^d \to \mathbb{R}^d$ such that $T_{\#}p_0 = p_1$.

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

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

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

$$v : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d \tag{37}$$

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.

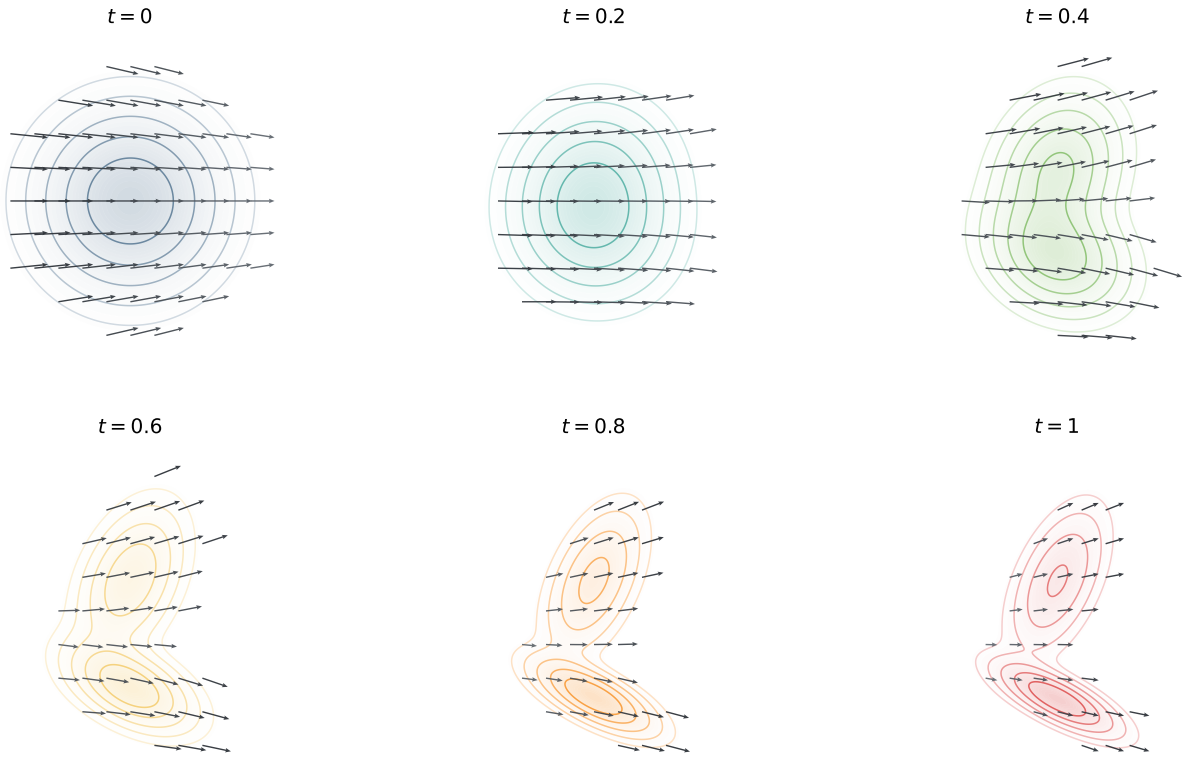

Figure 6: An example of velocity field for the Gaussian-to-GMM example.

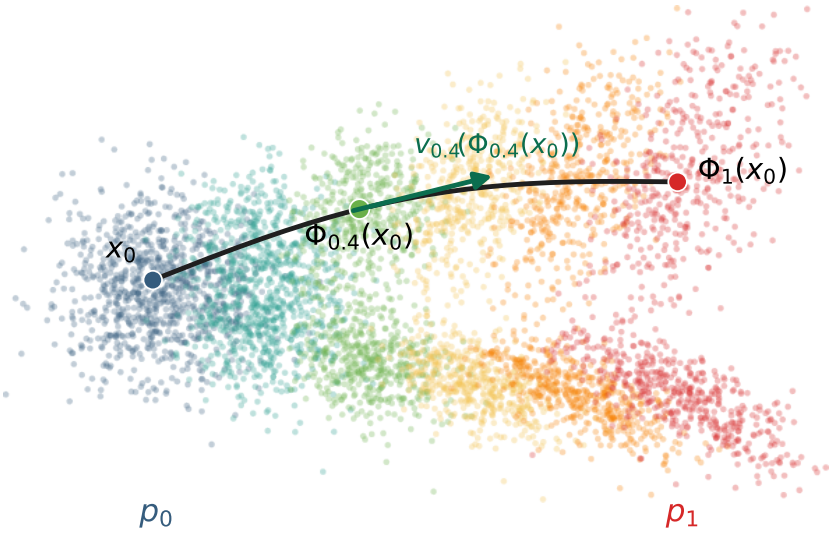

Figure 7: The flow viewpoint follows individual particles. The velocity vector is tangent to the particle trajectory.

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

$$\begin{cases} \frac{\mathrm{d}}{\mathrm{d}t}\varphi_t(x) = v_t(\varphi_t(x)) \\ \varphi_0(x) = x. \end{cases} \tag{38}$$

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 $\varphi_t : \mathbb{R}^d \to \mathbb{R}^d$ 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,

$$\boxed{p_t = (\varphi_t)_{\#}p_0.}$$

It is useful to distinguish two complementary viewpoints:

$$\frac{\mathrm{d}}{\mathrm{d}t}\varphi_t(x) = v_t(\varphi_t(x)) \tag{39}$$

which describes the motion of individual particles, whereas

$$p_t = (\varphi_t)_{\#}p_0 \tag{40}$$

describes the evolution of the whole distribution.

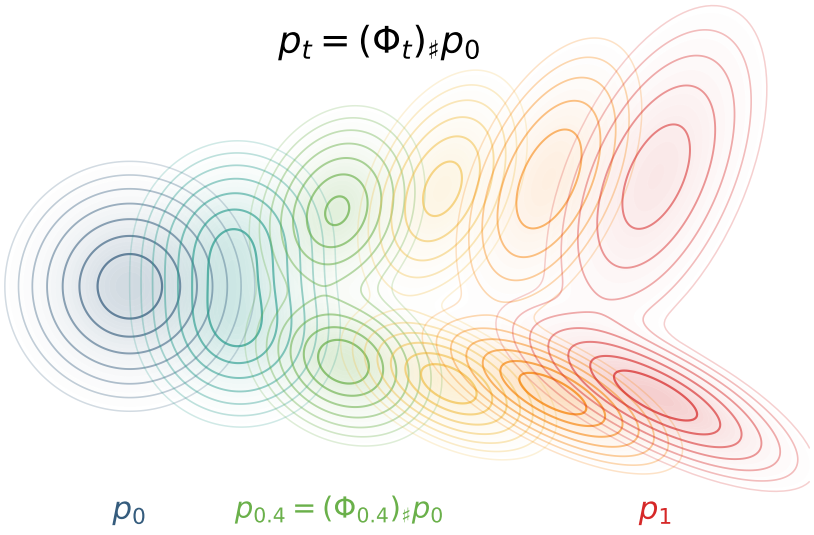

Figure 8: At the distribution level, the flow transports the whole probability measure: $p_t = (\varphi_t)_{\#}p_0$.

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*

$$\partial_t p_t(x) + \nabla \cdot (p_t(x)v_t(x)) = 0 \tag{41}$$

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 $[a, b]$, $\frac{\mathrm{d}}{\mathrm{d}t}\int_a^b p_t(x)\mathrm{d}x = p_t(a)v_t(a) - p_t(b)v_t(b)$. 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 $\phi \in \mathcal{C}_c^\infty(\mathbb{R}^d)$, $\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{R}^d}\phi(x)p_t(x)\mathrm{d}x = \int_{\mathbb{R}^d}\nabla\phi(x)^\top v_t(x)p_t(x)\mathrm{d}x$.

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
>
> $$\partial_t p_t + \nabla\cdot(p_t v_t) = 0, \quad p_{t=0} = p_0, \tag{42}$$
>
> and φ_t is the flow generated by v_t, then
>
> $$p_t = (\varphi_t)_{\#}p_0. \tag{43}$$

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

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

.

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.

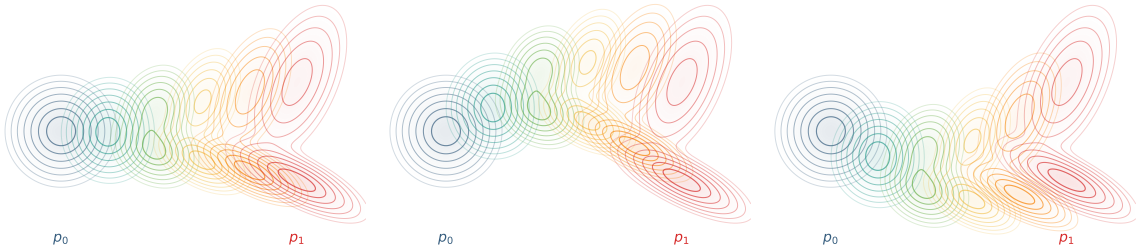

Figure 9: The same endpoint distributions can be connected by many different probability paths.

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

4.3.2 Conditional probability paths

Let

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

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 $p_t = \mathrm{Law}(X_t)$ 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

$$X_t = (1-t)x_0 + tx_1 \tag{46}$$

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

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

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

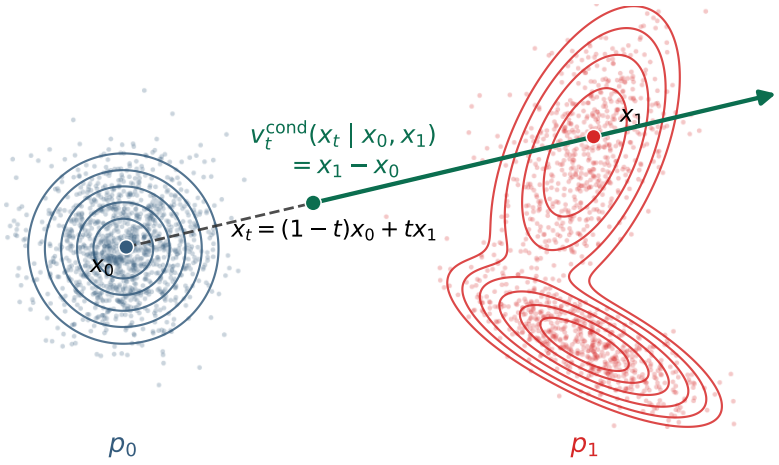

Figure 10: Conditionally on (x_0, x_1), the interpolation follows a straight segment with constant velocity.

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

$$x_t = (1-t)x_0 + tx_1 \tag{48}$$

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

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

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

$$\frac{\mathrm{d}x_t}{\mathrm{d}t} = v_t^{\mathrm{cond}}(x_t \mid x_0, x_1). \tag{49}$$

Conditionally on $(X_0, X_1) = (x_0, x_1)$, the probability path is the Dirac measure $p_t^{\mathrm{cond}}(\cdot \mid x_0, x_1) = \delta_{x_t}$. 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

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

This is exactly the weak formulation of the continuity equation for the couple $\left(p_t^{\mathrm{cond}}, v_t^{\mathrm{cond}}\right)$. 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, $X_t=(1-t)X_0+tx_1, \quad X_0\sim\mathcal{N}(0,I_d)$, remains random conditionally on $X_1=x_1$, and $X_t\mid X_1=x_1\sim\mathcal{N}(tx_1,(1-t)^2I_d)$. The corresponding conditional velocity field is

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

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 simply 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

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

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

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

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

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

> 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

$$\mathbb{E}[\mathbb{E}[X \mid Y]] = \mathbb{E}[X] \tag{57}$$

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

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

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:

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

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 $X_t = (1-t)X_0 + tX_1$,

$$U_t = X_1 - X_0 \tag{59}$$

, so

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

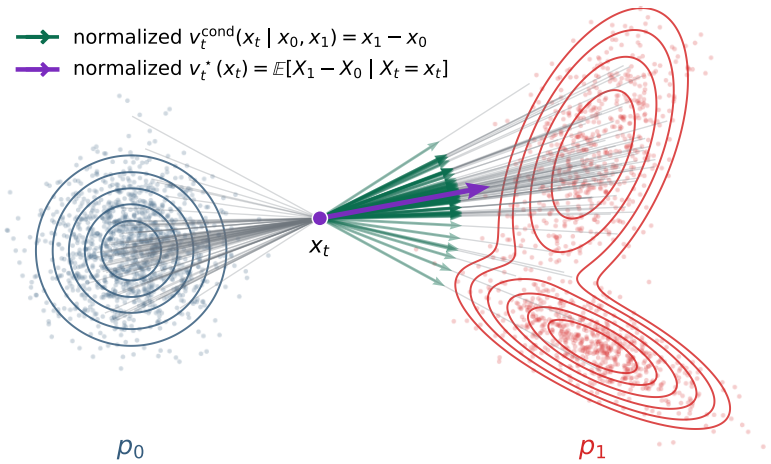

Figure 11: Several conditional trajectories can pass through the same point with different velocities. The marginal field averages these velocities conditionally on the current position.

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 $(p_t, v_t^\star)$ defines a valid transport:

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

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

4.4.3 The Flow Matching objective

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

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

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

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

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

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

.

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

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

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

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

If v_θ approximates $v^\star$ 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 $v_\theta = v^\star$, 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.

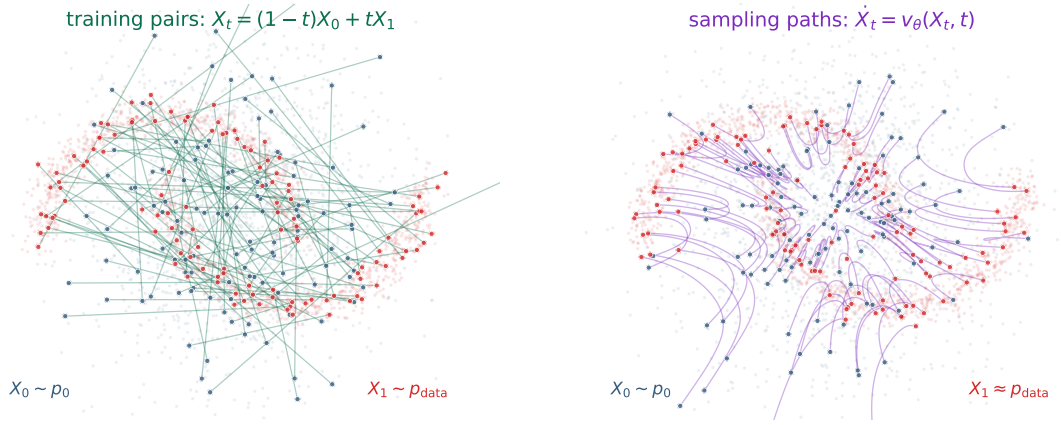

Figure 12: The conditional straight lines used during training are not the particle trajectories of the generative ODE. Flow Matching learns a marginal velocity field whose flow reproduces the same probability path.

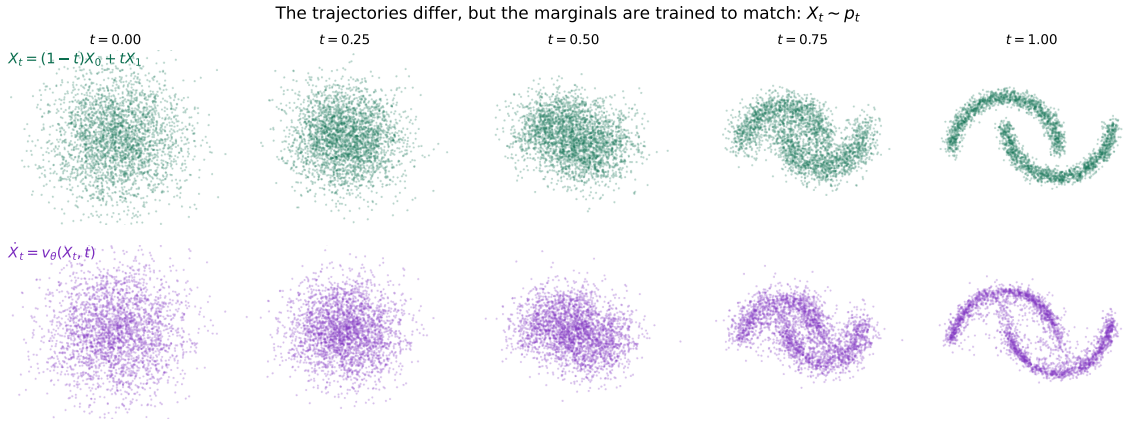

Figure 13: Although the particle trajectories differ, the prescribed interpolation and the learned generative ODE are trained to have matching marginals along time.

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*.

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*.