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.

Basic Probability Rules

Most of the computation we wil ldo in the class relies on the following 3 rules.

Bayes rule:

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

Law of total probability:

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

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.

Level lines:

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

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/

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.

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

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:

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

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

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

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,

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

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

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 $\mathbb{R}^d$.

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/)

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.

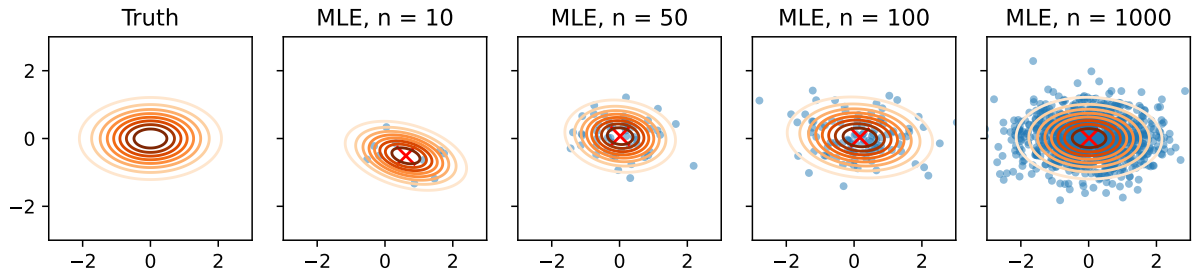

Figure 1: Illustration of MLE for various sample sizes n

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?

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?

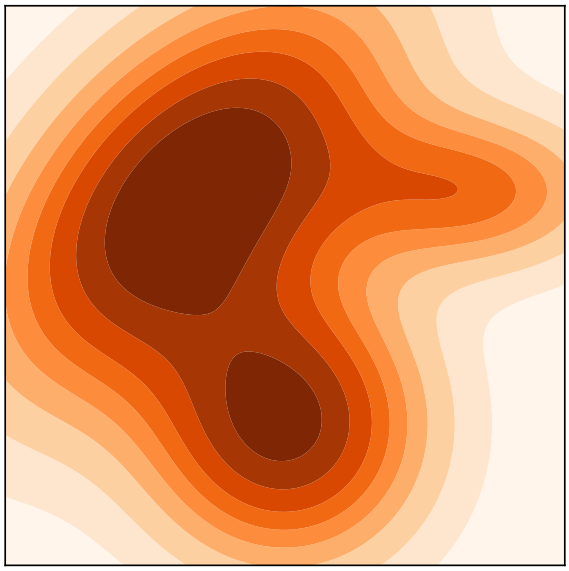

Figure 2: A mixture of Gaussians in 2D

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.

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!

Variational Auto Encoder (VAEs)

Auto Encoder (AE)

An autoencoder is a function f from $\mathbb{R}^{\mathbb{d}}$ to $\mathbb{R}^{\mathbb{d}}$, 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.

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:

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

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):

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

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.

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

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

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.

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:

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

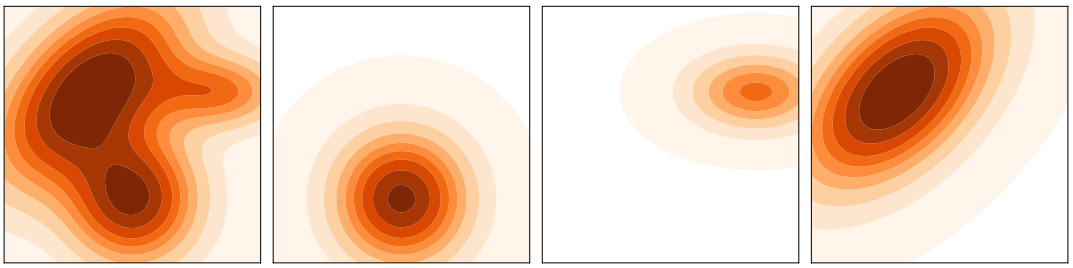

Figure 3: Mixture of Figure 2 decomposed in its 3 components.