ENSICaen

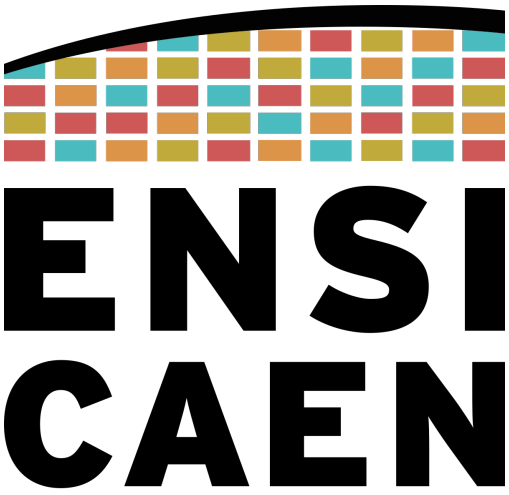
Spécialité Informatique - Majeure ISIA

Cours de Deep Avancé

Notes

Julien Rabin

julien.rabin@ensicaen.fr



Version : November 30, 2023

Avant propos

Contenu du cours

todo

À qui s'adresse ce cours ?

todo

Work in progress !

Ce document est régulièrement mis à jour : vérifier la date de version en couverture.

De nombreuses coquilles, fautes ou erreurs ont pu rester après relecture. Toutes les remarques sont donc les bienvenues, par mail ou via ce formulaire en ligne !

Bonne lecture !

Contents

1 Generative models

Sommaire

todo

1 Generation random gaussian images

Let $\mathcal{D} = \{x_i\}_{i\in[N]}$ be a collection of N datapoints, where $x_i \in \mathbb{R}^d$ is a d-dimensional vector.

Preprocessing images For datasets of color images or patches $\{u_i\}$, each item is normalized and vectorized as a pre-processing. Using tensor representation, let

$$\begin{array}{rccc} u: & [H]\times[W]\times[C] & \to & \mathbb{R} \\ & i,j,c & \mapsto & u(i,j,c) \end{array}$$

be such a color image, with $\Omega = [H]\times[W] \subset \mathbb{Z}^2$ the set of spatial coordinates, using the notation $[a] = \{1,\dots a\}$, $\forall\, a \in \mathbb{N}$. H, W and C indicate the height (number of rows), width (number of columns) and the number of channels of the image, using conventional matrix/tensor indices. Then we have the (arbitrary) 1-to-1 correspondance mapping between the matrix/tensor representation u and the vectorial representation $x = \text{vect}(u)$

$$x(\ell) = u(i,j,k)$$

with

$$\ell = WCi + Cj + k$$

and conversely

$$i = \left\lfloor \tfrac{\ell}{WC} \right\rfloor, \text{ then } j = \left\lfloor \tfrac{\ell - WCi}{C} \right\rfloor, \text{ then } k = \ell - WC - Cj$$

where $\lfloor x \rfloor = \sup\{n \in \mathbb{N}, n \le x\}$ is the *flooring* function returning the largest integer smaller or equal to x.

Matrix notation Using matrix notations and vectorial representation of datapoints x_i, the dataset can be seen as a $d \times N$ matrix :

$$X = [x_1, x_2, \dots x_N] \in \mathbb{R}^{d\times N}$$

Mean The average of datapoints $\hat{\mu} \in \mathbb{R}^d$ can be deduced from the empirical mean estimator of the population expectation μ

$$\hat{\mu} = \bar{X} = \tfrac{1}{N} X\mathbf{1} = \tfrac{1}{N}\sum_{i=1}^{N} x_i$$

where $\mathbf{1} = (1, ...) \in \mathbb{R}^N$ is a vector full of one.

Centering the data matrix writes

$$\tilde{X} = X - \bar{X}\mathbf{1}^\top$$

Using Pytorch broadcasting:

```
[Pytorch] X_center = X - X.mean(1)
```

Covariance Matrix The empirical covariance matrix writes

$$\hat{\Sigma} = \tfrac{1}{N}\tilde{X}\tilde{X}^\top$$

Distance Matrix Conversely, computing the squared Euclidean distance matrix writes

$$D_{i,j} = \|x_i - x_j\|^2 = \langle x_i - x_j\,,\, x_i - x_j \rangle = \|x_i\|^2 + \|x_j\|^2 - 2\langle x_i\,,\, x_j \rangle$$

In matrix form,

$$D = (X \odot X)\mathbf{1}\mathbf{1}^T + \mathbf{1}\mathbf{1}^T(X \odot X)^\top - 2X^\top X$$

where $(X^\top X)_{i,j} = \langle x_i\,,\, x_j \rangle$ is the scalar product between x_i and x_j

Using pytorch tensor and broadcasting, this writes

```
X_Norm2 = (X_center**2).sum(1)
D = X_Norm2 + X_Norm2.transpose(1,0) - 2 * X_center @ X_center.transpose(1,0)
```

Multivariate Gaussian $\mathcal{N}(\mu, \Sigma)$ Recall that a Multivariate random Gaussian variable X has a probability density function

$$X \in \mathbb{R}^d \sim p = \mathcal{N}(\mu, \Sigma): \qquad p(x) = \frac{1}{(\sqrt{2\pi})^d \sqrt{|\Sigma|}} e^{-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)}$$

Sampling from $\mathcal{N}(\hat{\mu}, \hat{\Sigma})$ Using the following change of variable

$$Z = f(\varepsilon) = \hat{\mu} + R\,\varepsilon \text{ where } \varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

we would like to have

$$Z \sim \mathcal{N}(\hat{\mu}, \hat{\Sigma}).$$

One can first demonstrate that Z is indeed Gaussian. Assuming that R is invertible (therefore $|R| = \det R \neq 0$), then the inverse mapping is $\varepsilon = f^{-1}(z) = R^{-1}(z - \hat{\mu})$ and the Jacobian for f and f^{-1} writes

$$\forall\, \varepsilon,\ J_f(\varepsilon) = Df(\varepsilon) = \left[\frac{\partial\, f(\varepsilon)|_i}{\partial \varepsilon_j}\right]_{(i,j)\in[d]^2} = R$$

and

$$\forall\, z,\ J_{f^{-1}}(z) = Df^{-1}(z) = R^{-1}.$$

Using the fact that the pdf of ε is $p(x) \propto \exp\left(-\frac{1}{2}\|x\|^2\right) = \exp\left(-\frac{1}{2}x^\top x\right)$

$$\begin{aligned}
\mathbb{P}(Z \in S) &= \int_{z\in S} p_Z(x)dz \\
&= \mathbb{P}(f(\varepsilon) \in S) = \mathbb{P}(\varepsilon \in f^{-1}(S)) = \int_{\varepsilon \in f^{-1}(S)} p(\varepsilon)d\varepsilon \\
&= \int_{\varepsilon \in f^{-1}(S)} p(\varepsilon)d\varepsilon = \int_{z\in S} p(f^{-1}(z))df^{-1}(z) \\
&= \int_{z\in S} p\left(R^{-1}(z-\hat{\mu})\right) \times |J_{f^{-1}}(z)|dz \\
&= |R| \int_{z\in S} p\left(R^{-1}(z-\hat{\mu})\right) dz
\end{aligned} \tag{1.1}$$

By identification,

$$p_Z(z) \propto p\left(R^{-1}(z-\hat{\mu})\right) \propto \exp\left(-\tfrac{1}{2}(z-\hat{\mu})^\top (R^{-1})^\top R^{-1}(z-\hat{\mu})\right)$$

which is indeed a gaussian pdf.

We can then show that the expectation of Z is indeed $\hat{\mu}$

$$\begin{aligned}
\mathbb{E}_Z(z) &= \mathbb{E}_\varepsilon(\hat{\mu} + R\,\varepsilon) = \mathbb{E}_\varepsilon(\hat{\mu}) + \mathbb{E}_\varepsilon(R\,\varepsilon) \\
&= \hat{\mu} + \left(\mathbb{E}_{(\varepsilon_j \sim \mathcal{N}(0,1))_{j\in[d]}}\left[\sum_{j\in[d]} R_{i,j}\varepsilon_j\right]\right)_{i\in[d]} = \hat{\mu} + \left(\sum_{j\in[d]} R_{i,j}\mathbb{E}_{\varepsilon_j \sim \mathcal{N}(0,1)}\left[\varepsilon_j\right]\right)_{i\in[d]} \\
&= \hat{\mu}
\end{aligned} \tag{1.2}$$

where $\mathbb{E}_\varepsilon(R\,\varepsilon) = \mathbf{0} \in \mathbb{R}^d$ because random variables ε_j are centered. For the expected covariance matrix, we get $RR^\top$

$$\begin{aligned} V(Z) &= \mathbb{E}_Z\left[(z - \mathbb{E}_Z(z))(z - \mathbb{E}_Z(z))^\top\right] \\ &= \mathbb{E}_\varepsilon\left[(R\,\varepsilon)(R\,\varepsilon)^\top\right] = \mathbb{E}_\varepsilon\left[R\,\varepsilon\,\varepsilon^\top R^\top\right] \\ V(Z)|_{i,j} &= \mathbb{E}_{(\varepsilon_j \sim \mathcal{N}(0,1))_{j\in[d]}}\left[\sum_{k,l} R_{i,k}\,\varepsilon_k\,\varepsilon_l R^\top_{l,j}\right] \\ &= \sum_{k,l} R_{i,k} R_{l,j} \mathbb{E}_{(\varepsilon_j \sim \mathcal{N}(0,1))_{j\in[d]}}\left[\varepsilon_k\,\varepsilon_l\right] \\ &= \sum_{k,l} R_{i,k} R_{j,l} \mathbb{1}_{\{k=l\}} = \sum_k R_{i,k} R_{j,k} = \left[R\,R^\top\right]_{i,j} \end{aligned} \tag{1.3}$$

Now, to sample from $\mathcal{N}(\hat{\mu}, \hat{\Sigma})$ using $Z = f(\varepsilon) = \hat{\mu} + R\,\varepsilon$, we need to solve for R the following equation

$$R\,R^\top = \hat{\Sigma}.$$

This is related to the computation of the square root $\sqrt{\Sigma}$. This problem has a solution which is however not unique. Indeed, is R_0 is a solution and M is an orthonormal matrix (*i.e.* $M^{-1} = M^\top$), then $R = R_0 M$ satisfies $R\,R^\top = R_0 M M^\top R_0^\top = \hat{\Sigma}$.

A solution can be found using Cholesky decomposition of a positive definite matrix $\hat{\Sigma} \succ 0$ into a product of lower-triangular matrices L : $\hat{\Sigma} = L\,L^\top$ with time complexity about $\frac{1}{3}d^3$.

Another approach is to use PCA (Principal Component Analysis) algorithm to decompose $\hat{\Sigma}$ into an eigenvector unary base V and a diagonal matrix of positive eigenvalues S:

$$\hat{\Sigma} = VSV^\top = V\mathrm{diag}(s)V^\top = PP^\top$$

Then one can use

$$P = V\sqrt{S} = [v_1, ..., v_d] \times \mathrm{diag}(\sqrt{s_1}, ...\sqrt{s_d}) = [\sqrt{s_1}v_1, ..., \sqrt{s_d}v_d] = (\sqrt{s}^\top \mathbf{1}) \odot V$$

Computing the V and S can be achieved with the SVD algorithm which complexity is again about $O(d^3)$. Note that P can be approximated using only the main eigenvectors which may be efficiently estimated by means of the power iteration algorithm.

describes power iteration algorithm

2 Variational Auto-Encoder

Sommaire

1 Preliminaries

This section quickly covers notations and useful definitions for inference.

1.1 Random Variables and Probability Distribution

Random Variables Ω is the sample space, which represents all possible events. A random variable X is a function $X : \Omega \mapsto \mathcal{X}$ which associates to a *random* event ω a value x from a set $\mathcal{X}$. Exemples :

- draw a coin : possible events are the observed face of the coin $\Omega = \{$"head","tail"$\}$, (arbitrary) associated values are $\mathcal{X} = \{0, 1\}$
- draw a dice : possible events are faces $\Omega = \{$".",":",..."::"$\}$, associated values are $\mathcal{X} = \{1, 2, ...6\}$
- throw a dart at a target : possible events are for instance the possible hit regions $\Omega = \{$"center","sector 1",..."sector 20 associated values are the scores $\mathcal{X} = \{1, 2, ...\}$
- measure of the height of a person : an event is the individual from the population in which it is randomly selected $\Omega = \{$"John Doe",...$\}$, associated values are positive real numbers $\mathcal{X} = \mathbb{R}^+ \backslash \{0\}$
- measure of the charge of a capacitor : an event describes the electronic configuration of the capacitor, associated values are real numbers $\mathcal{X} = \mathbb{R}$

A probability distribution is a function which gives the probabilities of the outcome of a random variable. We denote by $\mathbb{P}(X \in A)$ or $\Pr(X \in A)$ the probability that X takes values in a subset A of $\mathcal{X}$: it corresponds to the measure of the random events that allows for these observed values

$$\mathbb{P}(X \in A) = \mathbb{P}(\{\omega : x = X(\omega) \in A\}) = \mathbb{P}(X^{-1}(A))$$

The probability density function p_X is a function that can be integrated to compute the probability distribution:

$$P_X(A) = \mathbb{P}(X \in A) = \int_{x \in A} p_X(x) dx$$

We denote by $X \sim P_X$ or $X \sim p_X$ a random variable following a probability distribution P_X / probability density function p_X.

Discrete probability distributions A discrete random variable takes values in a set $X \in \mathcal{V} := \{V_1, V_2, ...\}$. For instance, the face value of a dice as discussed before. The probability that a random realisation of X is equal to a given value V_i is noted as $\mathbb{P}(X = V_i) = \Pr(X = V_i) = P_i$ and verifies

$$\sum_{V \in \mathcal{V}} \mathbb{P}(X = V) = 1 \text{ and } \mathbb{P}(X = V) > 0$$

The probability density p_X of X can be written as a weighted sum of Dirac masses:

$$p_X(x) = \sum_{V \in \mathcal{V}} \mathbb{P}(X = V)\delta(x - V)$$

where the Dirac distribution $\delta(x)$ satisfies $\langle \delta, f \rangle = \int_{\mathbb{R}} \delta(x)f(x)dx = f(0)$, in particular $\int \delta = 1$.

The probability of a set

$$\mathbb{P}(X \in \mathcal{S}) = \int_{\mathcal{S}} p_X(x)dx = \int_{\mathbb{R}^d} \mathbb{1}_{\mathcal{S}}(x) \sum_{V \in \mathcal{V}} \mathbb{P}(X = V)\delta(x - V)dx = \sum_{V \in \mathcal{V}} \mathbb{P}(X = V)\mathbb{1}_{\mathcal{S}}(V) = \sum_{s \in \mathcal{S}} \mathbb{P}(X = s)$$

where $\mathbb{1}_{\mathcal{S}}(x) = \begin{cases} 1 & \text{if } x \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases}$ is the indicator function of $\mathcal{S}$.

Continuous random variable A continuous random variable $X \sim P_X$ has a probability density function p w.r.t the Lebesgue measure if

$$P_X(S) = \mathbb{P}(X \in S) = \int_S dP_X(x) = \int_S p(x)dx$$

Note that "continuous" here refers to the fact that

Normal distributions A scalar random variable X which follows a **unidimensional Gaussian** centered at $\mu \in \mathbb{R}$ ($\mathbb{E}(X) = \mu$) with standard deviation $\sigma > 0$ (variance $\mathrm{V}(X) = \mathbb{E}\left[(X - \mathbb{E}(X))^2\right] = \sigma^2$) has the following density function :

$$X \in \mathbb{R} \sim p = \mathcal{N}(\mu, \sigma^2) : \qquad p(x) = g_\sigma(x - \mu) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

In particular, for unidimensional Gaussian centered at 0 ($\mathbb{E}(X) = 0$) and normalized (variance $\mathrm{V}(X) = \mathbb{E}\left[(X - \mathbb{E}(X))^2\right] = 1$):

$$X \in \mathbb{R} \sim p = \mathcal{N}(0, 1) : \qquad p(x) = g_1(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

A multi-dimensional random variable X which follows a **multi-variate gaussian distribution** centered at $\mu \in \mathbb{R}^d$ ($\mathbb{E}(X) = \mu$) with positive-definite covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$ (variance $\mathrm{V}(X) = \mathbb{E}\left[(X - \mathbb{E}(X))(X - \mathbb{E}(X))^\top\right] = \Sigma$) such that $\forall x \in \mathbb{R}^d, x^\top \Sigma x > 0$:

$$X \in \mathbb{R}^d \sim p = \mathcal{N}(\mu, \Sigma) : \qquad p(x) = \frac{1}{(\sqrt{2\pi})^d \sqrt{|\Sigma|}} e^{-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)}$$

where

- $x^\top \in \mathbb{R}^{1 \times d}$ is the transpose representation of the vector $\mathbb{R}^{d \times 1}$
- Σ^{-1} is the (unique) inverse of the covariance matrix;
- $|\Sigma| > 0$ indicates the (positive) determinant of the covariance matrix.
- $d(x, \mu) = \sqrt{(x - \mu)^\top \Sigma^{-1}(x - \mu)} = \sqrt{\langle x - \mu, \Sigma^{-1}(x - \mu) \rangle} = \|x - \mu\|_\Sigma$ is the Mahalanobis distance between x and μ

In particular, for **separable** and isotropic multi-variate gaussian distribution centered at $\mu \in \mathbb{R}^d$ and with *diagonal covariance matrix* (meaning that components x_i are **independent** and have different means but the same variance), we have

$$X \in \mathbb{R}^d \sim p = \mathcal{N}(\mu, \sigma^2 \mathrm{Id}) : \qquad p(x) = \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} e^{-\frac{1}{2\sigma^2}\|x-\mu\|^2} = \Pi_{i=1^d} \mathcal{N}(\mu_i, \sigma^2)(x_i)$$

Bayes' Rule It can be sumarized as

"Posterior = Prior × Likelihood / Evidence"

Given two events A and B, we have that $\Pr(A,B) = \Pr(A)\Pr(B|A) = \Pr(B)\Pr(A|B)$. If $\Pr(B) \neq 0$, the Bayes theorem states that

$$\Pr(A|B) = \frac{\Pr(A)\Pr(B|A)}{\Pr(B)} \tag{2.1}$$

Additionally, denoting $\bar{A}$ the complement event A (such that $\Pr(\bar{A}) = 1 - \Pr(A)$, we have $\Pr(B) = \Pr(B|A)\Pr(A) + \Pr(B|\bar{A})\Pr(\bar{A})$ and

$$\Pr(A|B) = \frac{\Pr(A)\Pr(B|A)}{\Pr(B|A)\Pr(A) + \Pr(B|\bar{A})\Pr(\bar{A})}$$

1.2 Inference and parametric probabilistic models

The true probability distribution $P(X)$ of a random variable X is generally unknown. A fundamental problem when modelling a random phenomenon is therefore to define an appropriate approximation to this probability distribution, which is often a parametric model $P(X,\theta)$.

To do so, one generally follow the following steps :

1. collect random N data samples x_i : this is the sampling process. All samples must be collected **independantly**, from the same population, so that they can be seen as observation of N **independent and identically distributed** (**iid**) random variables : $X_i \sim P_X$
2. choose an appropriate parametric model $P(X,\theta)$ do describe the data, given the observation and some prior knowledge (for instance, χ^2 for positive residual errors, gaussian when dealing with sums of iid measurements, a poisson distribution when counting disintegration of radioactive particles, a mixture of gaussian when collecting samples from two populations, etc)
3. choose the best parameters θ to fit the collected data (maximising likelihood or posterior probability for instance).

This process of inferring properties of the true distribution $P(X)$ from random samples is called the statistical **inference**. Other related problems include hypothesis testing and parametric inference.

1.3 Kullback-Leibler Divergence

The Kullback-Leibler divergence is a mean to compare two probability distributions.

Definition The KL divergence of P with respect to Q for two distributions P and Q on a measurable space $\mathcal{X} \subset \mathbb{R}^{\lceil}$

$$\mathrm{KL}(P||Q) = \begin{cases} \displaystyle\int_{\mathcal{X}} \log\left(\frac{P(dx)}{Q(dx)}\right) dP(x) & \text{if } P \text{ is absolutely continuous w.r.t } Q \\ +\infty & \text{otherwise} \end{cases} \tag{2.2}$$

(P is absolutely continuous w.r.t Q if $Q(A) = 0 \Rightarrow P(A) = 0$) and $\frac{P(dx)}{Q(dx)}$ is the Radon-Nykodym derivative of P with respect to Q.

When P and Q have respective densities p and q with respect to the Lebesgue measure, then

$$\mathrm{KL}(P||Q) = \mathrm{KL}(p||q) = \int_{\mathbb{R}^d} \log\left(\frac{p(x)}{q(x)}\right) p(x)dx = \mathbb{E}_{X\sim p}\left[\log\left(\frac{p(X)}{q(X)}\right)\right] \tag{2.3}$$

Properties This is not a distance since for instance there is no symmetry :

$$\mathrm{KL}(P||Q) \neq \mathrm{KL}(Q||P) \tag{2.4}$$

However, it verifies some interesting properties :

$$\mathrm{KL}(P||Q) \geq 0 \quad \text{(non-negativity)} \tag{2.5}$$

$$\mathrm{KL}(P||Q) = 0 \Rightarrow P = Q \quad \text{(equality)} \tag{2.6}$$

$$\mathrm{KL}(P||Q) \text{ is convex in } (P,Q). \quad \text{(convexity)} \tag{2.7}$$

A symmetric version of the KL divergence exists : the Jensen-Shannon divergence

$$\mathrm{JS}(P,Q) = \mathrm{KL}\left(P||\tfrac{P+Q}{2}\right) + \mathrm{KL}\left(Q||\tfrac{P+Q}{2}\right) \tag{2.8}$$

The triangular inequality is yet not verified.

Kullback-Leibler Divergence between Gaussians Let us consider first the unidimensional case $d = 1$ for simplicity where $P = \mathcal{N}(\mu, \sigma^2)$ and $Q = \mathcal{N}(0,1)$ (*i.e.* with $\mu \in \mathbb{R}$, and $\sigma > 0$)

$$\mathrm{KL}\left(\mathcal{N}(\mu,\sigma^2)\,||\,\mathcal{N}(0,1)\right) = \tfrac{1}{2}\left(\mu^2 + \sigma^2 - \log(\sigma^2) - 1\right) \tag{2.9}$$

and

$$\mathrm{KL}\left(\mathcal{N}(0,1)\,||\,\mathcal{N}(\mu,\sigma^2)\right) = \tfrac{1}{2}\left(\frac{\mu^2+1}{\sigma^2} + \log(\sigma^2) - 1\right) \tag{2.10}$$

In general

$$\mathrm{KL}\left(\mathcal{N}(\mu_1,\sigma_1)\,||\,\mathcal{N}(\mu_2,\sigma_2^2)\right) = \tfrac{1}{2}\left(\frac{(\mu_1-\mu_2)^2+\sigma_1^2}{\sigma_2^2} + \log\frac{\sigma_2^2}{\sigma_1^2} - 1\right) \tag{2.11}$$

Proof.

$$\begin{aligned}
\mathrm{KL}\left(\mathcal{N}(\mu,\sigma^2)\,||\,\mathcal{N}(0,1)\right) &= \mathbb{E}_{\mathcal{N}(\mu,\sigma^2)} \log\left[\frac{\mathcal{N}(\mu,\sigma^2)}{\mathcal{N}(0,1)}\right] \\
&= \mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)} \log\left[\frac{\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2\sigma^2}(z-\mu)^2}}{\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}z^2}}\right] \\
&= \mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)}\left[-\log(\sigma) - \tfrac{1}{2\sigma^2}(z-\mu)^2 + \tfrac{1}{2}z^2\right] \\
&= -\log(\sigma) \times \mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)}[1] - \tfrac{1}{2\sigma^2} \times \mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)}\left[(z-\mu)^2\right] + \tfrac{1}{2} \times \mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)}\left[z^2\right] \\
&= -\log(\sigma) - \tfrac{1}{2\sigma^2} \times \sigma^2 + \tfrac{1}{2} \times \mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)}\left[z^2 - (z-\mu)^2 + (z-\mu)^2\right] \\
&= -\log(\sigma) - \tfrac{1}{2} + \tfrac{1}{2}\mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)}\left[2z\mu - \mu^2 + (z-\mu)^2\right] \\
&= -\log(\sigma) - \tfrac{1}{2} + \tfrac{1}{2}2\mu\mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)}\left[z\right] - tfrac12\mu^2\mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)}\left[1\right] + tfrac12\mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)}\left[(z-\mu)^2\right] \\
&= -\log(\sigma) - \tfrac{1}{2} + \mu^2 - \tfrac{1}{2}\mu^2 + \tfrac{1}{2}\sigma^2 \\
&= -\frac{1}{2} - \log(\sigma) + \tfrac{1}{2}(\mu^2+\sigma^2)
\end{aligned}$$

□

Now in the case of multi-variate but separable gaussians $\mathcal{N}(\mu, \mathrm{Id}\,\sigma^2)$, where $\mu = (\mu_i)_{i\in[d]}, \sigma^2 = (\sigma_i^2)_{i\in[d]} \in \mathbb{R}^d$ and $\mathrm{Id}\,\sigma^2$ is a diagonal covariance matrix, we have a similar result :

$$\begin{aligned}
\mathrm{KL}\left(\mathcal{N}(\mu,\mathrm{Id}\sigma^2)\,||\,\mathcal{N}(\mathbf{0},\mathrm{Id})\right) &= \tfrac{1}{2}\left(\|\mu\|^2 + \|\sigma\|^2 - \langle\,\mathbf{1}\,,\log\sigma^2\,\rangle - d\right) \\
&= \tfrac{1}{2}\left(\sum_{i=1}^{d}\mu_i^2 + \sigma_i^2 - \log(\sigma_i^2) - 1\right)
\end{aligned} \tag{2.12}$$

Proof.

$$\begin{aligned}
\mathrm{KL}\left(\mathcal{N}(\mu, \mathrm{Id}\,\sigma^2)\,||\,\mathcal{N}(\mathbf{0}, 1)\right) &= \mathbb{E}_{\mathcal{N}(\mu,\mathrm{Id}\,\sigma^2)} \log\left[\frac{\mathcal{N}(\mu,\sigma^2)}{\mathcal{N}(\mathbf{0},\mathrm{Id}\,)}\right] \\
&= \mathbb{E}_{z\sim\mathcal{N}(\mu,\mathrm{Id}\,\sigma^2)} \log\left[\frac{\frac{1}{\sqrt{\Pi_i 2\pi\sigma_i^2}} e^{-\sum_i \frac{1}{2\sigma_i^2}(z_i-\mu_i)^2}}{\frac{1}{\sqrt{2\pi}^d} e^{-\frac{1}{2}\|z\|^2}}\right] \\
&= \mathbb{E}_{z\sim\mathcal{N}(\mu,\mathrm{Id}\,\sigma^2)} \left[\sum_i -\log(\sigma_i) - \tfrac{1}{2\sigma_i^2}(z_i-\mu_i)^2 + \tfrac{1}{2}z_i^2\right] \\
&= \sum_i \mathbb{E}_{z_i\sim\mathcal{N}(\mu_i,\sigma_i^2)} \left[-\log(\sigma_i) - \tfrac{1}{2\sigma_i^2}(z_i-\mu_i)^2 + \tfrac{1}{2}z_i^2\right] \\
&= \sum_{i=1}^{d} -\frac{1}{2} - \frac{1}{2}\log(\sigma_i^2) + \tfrac{1}{2}(\mu_i^2+\sigma_i^2)
\end{aligned}$$

□

1.4 Evidence Lower Bound (ELBO)

Let us consider here a joint probability distribution $P_{X,Z}(x,z)$ of two random variables, where x is a measurement (a sample/observation from a random variable X) and z is a hidden state/factor, from a random variable Z we cannot necessarily observe. For instance,

- Z describes the age of a tree and X the height;
- Z describes the consumption of a substance (say tobacco) and X the symptoms.

The two random variables X and Z are not independent, so that $P_{X,Z}(x,z) \neq P_X(x)P_Z(z)$. Rather, we can write

$$P_{X,Z}(x,z) = P_Z(z)P_{X|Z}(x|z)$$

where $P_{X|Z}(x|z)$ is the conditional probability distribution of X **knowing** Z.

Given the probability distribution P_Z and $P_{X,Z}$, it is worth noting that computing the marginal distribution requires $P_X(x)$ to consider every possible state for the hidden variable Z :

$$P_X(x) = \int_{z\in\mathcal{Z}} P_{X,Z}(x,z)dz = \int_{z\in\mathcal{Z}} P_Z(z)P_{X|Z}(x|z)dz = \mathbb{E}_{Z\sim P_Z} P_{X|Z}(x|z)$$

This integral is generally difficult to compute, and requires approximation techniques such as Monte-Carlo sampling.

Inference Now, assuming that we have only some prior knowledge of P_Z and $P_{X,Z}$ (for instance some parametric model that we denote as $p_\theta(x,z)$), the question of model inference (which is discussed in depth in the next section) is again related to the computation of the model marginal likelihood that we denote for simplicity $p_\theta(x)$, that is also called **evidence**.

Evidence Lower BOund ELBO (which stands for Evidence Lower BOund) is an inequality that is useful in inference to avoid the need to compute directly the likelihood $p_\theta(x)$. The following inequality provides a bound to maximise the marginal (log) likelihood of a parametric model p_θ (a.k.a model evidence) w.r.t to the model's parameters θ

$$\text{(ELBO)} \qquad \log p_\theta(x) \geq \mathbb{E}_{z\sim q}\left[\log\left(\frac{p_\theta(x,z)}{q(z)}\right)\right] \tag{2.13}$$

where $q(z)$ can be *any* probability distribution. However, as detailed in the next part the gap in this inequality is the KL divergence between $q(z)$ and the posterior distribution $p_\theta(z|x)$.

Proof For simplicity, let us consider **any probability distribution** Q with a density $q(z)$ (thus satisfying $\int dQ = \int q = 1$) we can write

$$\begin{aligned}
\log p_\theta(x) &= \int_{z\in\mathbb{R}^\ell} q(z)\log p_\theta(x)dz = \mathbb{E}_{z\sim q}\left[\log p_\theta(x)\right] \\
&= \int_{z\in\mathbb{R}^\ell} q(z)\log\left(\frac{p_\theta(x)p_\theta(z|x)}{p_\theta(z|x)}\right)dz = \int_{z\in\mathbb{R}^\ell} q(z)\log\left(\frac{p_\theta(x,z)}{p_\theta(z|x)}\right)dz \\
&= \int_{z\in\mathbb{R}^\ell} q(z)\log\left(\frac{p_\theta(x,z)}{p_\theta(z|x)}\frac{q(z)}{q(z)}\right)dz \\
&= \int_{z\in\mathbb{R}^\ell} q(z)\log\left(\frac{p_\theta(x,z)}{q(z)}\right)dz + \int_{z\in\mathbb{R}^\ell} q(z)\log\left(\frac{q(z)}{p_\theta(z|x)}\right)dz \\
&= \mathbb{E}_{z\sim q}\left[\log\left(\frac{p_\theta(x,z)}{q(z)}\right)\right] + \underbrace{\mathrm{KL}\left(q(z)||p_\theta(z|x)\right)}_{\geq 0} \\
&\geq \mathbb{E}_{z\sim q}\left[\log\left(\frac{p_\theta(x,z)}{q(z)}\right)\right]
\end{aligned} \tag{2.14}$$

2 Principle of variational inference with latent models

Let consider a dataset $\mathcal{D} = \{x_1, ...x_n\} = \{x_i\}_{i\in[N]}$ composed of N independent and identically distributed (*iid*) samples $x_i \in \mathbb{R}^d$. The notation $[N]$ indicates the set $\{1, 2, ...N\}$. These datapoints/measurements are sampled from an (unknown) probability distribution often referred to as P (or μ), such that

$$\Pr(X \in A) = P(A) = \int_A dP.$$

When P has a density p w.r.t the Lebesgue measure on $\mathbb{R}^d$, we have $P(A) = \int_A dP = \int_{x\in A} p(x)dx$ and we may denote for simplicity $X \sim p(x)$.

The objective of a generative model is to sample new random realisations from $X \sim P$.

If P is known, one could assess the likelihood of a given datapoint x by computing $P(x)$. A major practical problem is that computing such probability is often intractable. A second critical problem is that P is more than often unknown. Last, it may be difficult to randomly sample from P directly.

2.1 Maximum Likelihood Inference

Variational inference consists in using a **surrogate** likelihood function $p_\theta(x) = \Pr(X = x|\theta)$ to infer the probability $P(x)$. This function $p_\theta(x)$ is a parametric model with parameters represented by the vector θ. We want to choose good parameters θ such that p_θ is close in some sense to P.

The model is learned from the dataset. This can be done by *maximum likelihood estimation* (MLE), that is choosing the optimal parameters θ as the one maximising the likelihood $\mathcal{L}_\mathcal{D}(\theta)$ of datapoints, *i.e.*

$$\text{(MLE)} \qquad \max_\theta \left\{\mathcal{L}_\mathcal{D}(\theta) := p_\theta(\mathcal{D}) := p_\theta(x_1, ...x_N)\right\} = \max_\theta p_\theta(x_1) \times p_\theta(x_2) \times ... \times p_\theta(x_N) \tag{2.15}$$

where the product comes from the hypothesis that samples are independently drawn from the same distribution P.

[mettre illustration 1D avec modèle gaussien ou uniforme]

Using the fact that the natural logarithmic function log is an increasing function, we have the equivalent (and more conventional) minimisation problem when considering the **negative log-likelihood** (or NNL) of the dataset $-\log p_\theta(\mathcal{D})$:

$$\text{(NNL-minimization)} \qquad \min_\theta \left\{-\frac{1}{N}\log\mathcal{L}_\mathcal{D}(\theta) = -\frac{1}{N}\log p_\theta(\mathcal{D}) = -\frac{1}{N}\sum_{i=1}^N \log p_\theta(x_i)\right\} \tag{2.16}$$

Exemple of the gaussian distribution Assuming 1D samples ($d = 1$), and using a normal distribution $p_\theta(x) = \mathcal{N}(\mu, \sigma^2)(x) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ we have the following optimization problem

$$\min_{\mu,\sigma} \frac{1}{N}\sum_{i=1}^N \left(\log\sqrt{2\pi} + \log\sigma + \frac{(x_i-\mu)^2}{2\sigma^2}\right)$$

Solving for parameter μ, then σ gives the empirical estimators :

$$\mu^* = \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad (\sigma^2)^* = \hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

These estimators are both consistent, meaning that they converge to the true parameter provided infinite sample ($N \to \infty$). Note however that contrary to $\bar{x}$, $\hat{\sigma}^2$ is a **biased** estimator. Indeed in average for a finite sample ($N < \infty$) there is a discrepancy with the parameter. We can show that in $\mathbb{E}_X(\hat{\sigma}^2(X) - \sigma^2) = -\frac{1}{N}\sigma^2 < 0$. A simple correction yields the following unbiaised estimator : $\frac{N}{N-1}\hat{\sigma}^2(x)$ for which $\mathbb{E}_X(\frac{N}{N-1}\hat{\sigma}^2(X)) = \sigma^2$).

Stochastic optimization When considering more complex model, such as deep neural network, a standard approach is to resort to gradient descent techniques to optimise parameters θ, *i.e.*

$$\nabla_\theta \log \mathcal{L}_{\mathcal{D}}(\theta) = \nabla_\theta \log p_\theta(\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log p_\theta(x_i) \approx \mathbb{E}\left(\nabla_\theta \log p_\theta(X)\right) \tag{2.17}$$

When the dataset is too large (large number of datapoints N, in large dimension d), one may have to use mini-batch stochastic optimisation by randomly and uniformly sampling a batch $\mathcal{B}$ subset of the dataset $\mathcal{D}$

$$\nabla_\theta \log p_\theta \approx \frac{1}{|\mathcal{B}|}\sum_{b \in \mathcal{B} \subset \mathcal{D}} \nabla_\theta \log p_\theta(b) \tag{2.18}$$

Other techniques may be employed, such as Expectation-Maximisation for instance.

Bayesian Inference For the problem of parametric inference of a given model (for instance, considering a particular parametric family of probability distribution, such as a mixture of K multivariate gaussians) with $X \sim P$ and parameters θ, one can derive from the Bayes theorem

$$\Pr(\theta|X) = \frac{\Pr(\theta)\Pr(X|\theta)}{\Pr(X)} = \frac{\Pr(\theta)\Pr(X|\theta)}{\int_{\theta\in\Theta}\Pr(\theta)\Pr(X|\theta)d\theta} \propto \Pr(\theta)\Pr(X|\theta) \tag{2.19}$$

where

- Θ is the parameter space
- $\Pr(\theta)$ is the **prior** distribution over parameters
- $\Pr(X|\theta)$ is the **likelihood** of the data given the parameters
- $\Pr(X)$ is the **model evidence** or marginal likelihood, that is the probability over all possible choice of parameters, often intractable
- $\Pr(\theta|X)$ is the posterior distribution of parameters

Rather than maximizing directly the likelihood, **maximum a posteriori inference** (MAP) consists in maximizing $\Pr(\theta|X)$ the posterior distribution of parameters. This can be achieved by addressing the following problem

$$\text{(MAP)} \qquad \max_\theta \Pr(\theta)\Pr(X|\theta) \tag{2.20}$$

for a specific prior distribution of parameters $\Pr(\theta)$.

Latent variable models A latent variable model takes into account hidden factors, or latent variables denoted as $z \in \mathbb{R}^\ell$. For instance, to generate natural outdoor scene, z may represent the season, the hour of the day, the location, so forth and so on.

Using conditional-distribution, one has : $\Pr(X = x, Z = z) = \Pr(Z = z)\Pr(X = x|Z = z)$.

With a surrogate parametric probability function, a latent variable model formulates now as follows : $p_\theta(x, z)$.

With a slight abuse of notations, we will write similarly :

$$p_\theta(x, z) = p_\theta(z)p_\theta(x|z)$$

where we have now the following terms :

- $p_\theta(x, z)$ is the latent variable model;
- $p_\theta(x|z)$ is the conditional distribution of a sample x knowing the hidden factor z : this is the ***likelihood*** of x given the latent observation z (for instance, knowing that a person has a moustache, this is the probability distribution of face of a person with such attributes);
- $p_\theta(z|x)$ is the conditional distribution of a latent variable z given the sample x : this is the ***posterior*** distribution of latent variable given a sample x (for instance, given a portrait of a bald person, this is the probability distribution of attributes such as age or gender);
- $p_\theta(z) = \int_{x\in\mathbb{R}^d} p_\theta(x, z)dx$ is the marginal distribution for z; it indicates the ***prior*** distribution of hidden variables; for instance in face generation, if $z \in \{0, 1\}$ accounts for a binary attribute (such as gender, wearing glasses, etc), $p_\theta(Z = 1) = p$ and $p_\theta(Z = 0) = 1 - p$ is a Bernouilli distribution and the parameter p is one of the variable in θ;
- $p_\theta(x) = \int_{z\in\mathbb{R}^\ell} p_\theta(x, z)dz$: this represents the marginal distribution for the variable x. When considering datapoints, this is the **model evidence** (or marginal likelihood) for x.

Sampling latent variable models Using the relationship $p_\theta(x, z) = p_\theta(z)p_\theta(x|z)$, sampling the model can be achieved as follows

1. sample a latent variable $Z \sim p_\theta(z)$
2. sample a variable $X \sim p_\theta(X|z)$ depending on $Z = z$

Optimising latent variable models Given a dataset $\mathcal{D}$, the "best" latent variable models (for a given parametric family) can be defined as maximizing the model **evidence**, that is the marginal distribution of datapoints x, i.e. for $x \in \mathcal{D}$

$$p_\theta(x) = \int_{z\in\mathbb{R}^\ell} p_\theta(x, z)dz = \int_{z\in\mathbb{R}^\ell} p_\theta(z)p_\theta(x|z)dz = \mathbb{E}_{Z\sim p_\theta(Z)}\left(p_\theta(x|Z)\right) \tag{2.21}$$

which can be interpreted as the average/expected likelihood of x with respect to all (plausible) latent configurations.

Note that optimising the posterior probability now writes:

$$p_\theta(z|x) = \frac{p_\theta(z)p_\theta(x|z)}{p_\theta(x)}$$

where one cannot discard the denominator (model evidence) which also depends on θ.

Example of the GMM Let us use again the example of GMM (Gaussian Mixture Model). One first choose a model (let say, fixing the number of components K) and then we have

- $p_\theta(z) = \sum_{k=1}^{K} \pi_k \delta_k(z) = \sum_{k=1}^{K} \pi_k \delta(z - k)$ the probability distribution of the latent variable z : it has a discrete probability Pr(Z=k) $= \pi_k$ if $k \in [K]$, such that $\sum_{k=1}^{K} \pi_k = 1$ and $\pi_k > 0$
- the conditional probability has a density $p_\theta(x|z = k) = \mathcal{N}(\mu_k, \Sigma_k)(x)$
- the evidence for a given sample x writes

$$p_\theta(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \Sigma_k)(x) \tag{2.22}$$

- the set of parameters θ is composed of $\pi = (\pi_k)_{k\in[K]}$, $\mu = (\mu_k)_{k\in[K]}$ and $\Sigma = (\Sigma_k)_{k\in[K]}$

The optimal parameters can be again defined as maximising the model evidence (marginal likelihood). To do so, we first write the model evidence for the dataset composed of iid samples

$$p_\theta(\mathcal{D}) = \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \Sigma_k)(x_i)$$

Now, trying direct minimization of the NLL with gradient descent techniques :

$$-\log p_\theta(\mathcal{D}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \Sigma_k)(x_i) \right)$$

is more difficult than before (1D case) : first because of the sum inside the log, and second because matrices Σ_k are subjected to be symmetric and positive definite covariance matrices.

As a result, the optimal parameters maximising the model evidence (marginal likelihood) are usually approached by an EM algorithm (Expectation-Maximisation), without guaranty of optimality.

Expectation-Maximisation (EM) Using the lower bound on model evidence (ELBO (2.13)), the principle of EM algorithm is to update the model's parameters θ by optimising the right hand side of the inequality for a specific choice of probability distribution $q(z)$:

$$\log p_\theta(x) \geq L(\theta, x) := \mathbb{E}_{z\sim q}\left[\log \left(\frac{p_\theta(x,z)}{q(z)} \right) \right]$$

Let consider a sequence $(\theta_t)_{t\in\mathbb{N}}$ of the model's parameters where t indicates the time steps. At time $t+1$, using $q(z) = p_{\theta_t}(z|x)$, one has

$$\begin{aligned} L(\theta, x) &= \mathbb{E}_{z\sim p_{\theta_t}(z|x)}\left[\log \left(\frac{p_\theta(z) p_\theta(x|z)}{p_{\theta_t}(z|x)} \right) \right] = \int_{z\in\mathbb{R}^\ell} p_{\theta_t}(z|x) \log \left(\frac{p_\theta(z) p_\theta(x|z)}{p_{\theta_t}(z|x)} \right) dz \\ &= \int_{z\in\mathbb{R}^\ell} p_{\theta_t}(z|x) \log \left(p_\theta(z) p_\theta(x|z) \right) dz - \int_{z\in\mathbb{R}^\ell} p_{\theta_t}(z|x) \log p_{\theta_t}(z|x) dz \\ &= E(\theta, \theta_t, x) + H(\theta_t, x) \end{aligned} \tag{2.23}$$

Observe that the last term is the entropy of $\log p_{\theta_t}(z|x)$, which is constant w.r.t the variable θ. This means that only the first term (the expectation of the latent model denoted to as $E(\theta, \theta_t, x)$ has to be maximized w.r.t θ at step $t+1$.

It is important to add that, when considering the likelihood of the full dataset, one has to consider the set of iid datapoints $x = (x_i)_{i\in[N]}$. Enforcing that the latent variables are iid as well (meaning that $q(z) = \Pi_{i\in[N]} q(z_i)$), this boils down to consider **the sum of expectations** w.r.t corresponding latent variables z_i, that is

$$E(\theta, \theta_t) = \sum_{x_i \in \mathcal{D}} \mathbb{E}_{z_i \sim p_{\theta_t}(z_i|x_i)} \log \left(p_\theta(z_i) p_\theta(x_i|z_i) \right)$$

The EM algorithm works as follows : at step $t+1$, considering parameters θ_t estimated at the previous step t

- **E step:** Compute the **expectation** $E(\theta, \theta_t)$. This requires to compute the posterior probability $p_{\theta_t}(z_i|x_i)$ for the previous model (at step t)

- **M step:** **maximization** of the objective function $E(\theta, \theta_t)$ w.r.t θ:

$$\theta_{t+1} = \max_\theta E(\theta, \theta_t)$$

Finally, from the computation of the ELBO inequality (2.14) we have that

$$\log p_\theta(x) = \mathbb{E}_{z\sim p_{\theta_t}(z|x)}\left[\log \left(\frac{p_\theta(x,z)}{p_{\theta_t}(z|x)} \right) \right] + \mathrm{KL}\left(p_{\theta_t}(z|x) || p_\theta(z|x) \right)$$

This shows that the EM algorithm performs an alternate minimization of the marginal model likelihood, between the update of parameters (M step) and the update of the surrogate function $q(z)$ for $p_\theta(z|x)$ (E-step) (θ being fixed to θ_t).

Application to gaussian model mixtures The EM algorithm is particularily interesting when dealing with mixture models, where the latent variables are discrete, in such a way that the computation of the posterior probabilities $p_{\theta_t}(z_i|x_i)$ are tractable. In the general case however, this is not the case.

For a GMM with K components as described in (2.22), using Bayes's rule (2.1):

$$p_{\theta_t}(z_i = k|x_i) = \frac{p_{\theta_t}(x_i|z_i = k)p_{\theta_t}(z_i = k)}{p_{\theta_t}(x_i)}$$

Now, remember that parameters θ_t are fixed at step $t+1$ so we can simply write

$$p_{\theta_t}(z_i = k|x_i) = \tfrac{1}{C} p_{\theta_t}(x_i|z_i = k)p_{\theta_t}(z_i = k)$$

where C is a normalisation constant such that $\sum_{k=1}^{K} p_{\theta_t}(z_i = k|x_i) = 1$. This yields

$$p_{\theta_t}(z_i = k|x_i) = \frac{\mathcal{N}(\mu_k^t, \Sigma_k^t)(x_i)\pi_k^t}{\sum_{k=1}^{K} \mathcal{N}(\mu_k^t, \Sigma_k^t)(x_i)\pi_k^t},$$

where π_k^t and (μ_k^t, Σ_k^t) indicate respectively the mixture and the gaussian parameters for the k-th component **at time step** t. This corresponds to the softmax of the log probabilities $\log p_{\theta_t}(x_i, z_i = k)$.

Using this posterior probability for model at step t, we can compute the expectation:

$$\begin{aligned} E(\theta, \theta_t) &= \sum_{x_i \in \mathcal{D}} \mathbb{E}_{z_i \sim p_{\theta_t}(z_i|x_i)} \log\left(p_\theta(z_i)p_\theta(x_i|z_i)\right) \\ &= \sum_{x_i \in \mathcal{D}} \sum_{k \in [K]} \frac{\mathcal{N}(\mu_k^t, \Sigma_k^t)(x_i)\pi_k^t}{\sum_{k=1}^{K} \mathcal{N}(\mu_k, \Sigma_k)(x_i)\pi_k} \log\left(\pi_k \mathcal{N}(\mu_k, \Sigma_k)(x_i)\right) \end{aligned} \tag{2.24}$$

Remember that inside the log parameters are not fixed and need to be optimized for the next step $t+1$. Denoting $\alpha_{i,k}^t = p_{\theta_t}(z_i = k|x_i)$ the fixed normalized weights in this sum, we have therefore

$$E(\theta, \theta_t) = \sum_{i \in [N]} \sum_{k \in [K]} \alpha_{i,k}^t \left(\log \pi_k - \frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x_i - \mu_k)^\top \Sigma_k^{-1}(x_i - \mu_k) \right) \tag{2.25}$$

The M-step corresponds to solving

$$\max_{\{\pi_k, \mu_k, \Sigma_k\}_{k \in [K]}} E(\theta, \theta_t)$$

For parameters π_k, the solution is straight-forward

$$\forall k \in [K], \quad \pi_k^{(t+1)} \propto \sum_{i \in [N]} \alpha_{i,k}^t$$

since $\forall i \in [N]$, $\sum_k \alpha_{i,k}^t = 1 = \sum_k \pi_k^{(t+1)}$ we have that

$$\forall k \in [K], \quad \pi_k^{(t+1)} = \tfrac{1}{N} \sum_{i \in [N]} \alpha_{i,k}^t = \tfrac{1}{N} \sum_{x_i \in \mathcal{D}} \frac{\pi_k^t \mathcal{N}(\mu_k^t, \Sigma_k^t)(x_i)}{\sum_{j=1}^{K} \pi_j^t \mathcal{N}(\mu_j^t, \Sigma_j^t)(x_i)}$$

More computations for each gaussian components' parameters yield

$$\mu_k{}^{(t+1)} = \frac{\sum_{i=1}^{N} \alpha_{i,k}^t \mathbf{x}_i}{\sum_{j=1}^{N} \alpha_{j,k}^t}$$

and

$$\Sigma_k^{(t+1)} = \frac{\sum_{i=1}^{N} \alpha_{i,k}^t (\mathbf{x}_i - \mu_k^{(t+1)})(\mathbf{x}_i - \mu_k^{(t+1)})^\top}{\sum_{j=1}^{N} \alpha_{j,k}^t}.$$

Proof. Optimising for mean parameters $(\mu_k)_{k \in [K]}$ in (2.25) is equivalent to minimising for each k the following quadratic objective function (covariance matrix Σ_k being fixed) :

$$\min_{\mu} \; f_k(\mu) := \sum_{i \in [N]} \alpha_{i,k}^t \left(\frac{1}{2}(x_i - \mu)^\top \Sigma_k^{-1}(x_i - \mu) \right) = \frac{1}{2} \sum_{i \in [N]} \alpha_{i,k}^t \langle\, x_i - \mu \,,\, \Sigma_k^{-1}(x_i - \mu) \,\rangle.$$

Using the fact that $\nabla_x \langle\, x\,,\, Ax\,\rangle = (A + A^\top)x$, Σ_k is a symmetric and invertible matrix, the gradient writes

$$\nabla_\mu f_k(\mu) = \sum_{i\in[N]} \alpha_{i,k}^t \Sigma_k^{-1}(\mu - x_i) = \Sigma_k^{-1}\left[(\sum_{i\in[N]} \alpha_{i,k}^t)\mu - \sum_{i\in[N]} \alpha_{i,k}^t x_i \right].$$

Optimality condition $\nabla_\mu f_k(\mu) = \vec{0}$ and the fact that $\ker(\Sigma_k) = \{\vec{0}\}$ gives the desired result.

Optimising for covariance parameters $(\Sigma_k)_{k\in[K]}$, is equivalent to minimising for each k the following objective function (using the above expression for μ_k) :

$$\min_S\ g_k(S) := \sum_{i\in[N]} \alpha_{i,k}^t \left(y_{i,k}^\top S^{-1} y_{i,k} + \log|S| \right) = \sum_{i\in[N]} \alpha_{i,k}^t \langle\, y_{i,k}\,,\, S^{-1} y_{i,k}\,\rangle + \sum_{i\in[N]} \alpha_{i,k}^t \log|S|.$$

where $y_{i,k} = x_i - \mu_k$ and $|S|$ indicates the determinant of matrix S. Remember that the multivariate gaussian model assumes that S^{-1} exists, that is $|S| \neq 0$. Here are some useful properties of matrix derivation for the determinant and the dot product with a symmetric and invertible matrix $S = S^\top$ (see *e.g.* the matrix cookbook)

$$\frac{\partial |S|}{\partial S} = |S|(S^{-1})^\top = |S|S^{-1}$$
$$\frac{\partial a^T S^{-1} b}{\partial S} = -(S^{-1})^\top a b^\top (S^{-1})^\top = -S^{-1} a b^\top S^{-1}$$

Then we have $\dfrac{\partial}{\partial S}\log|S| = S^{-1}$, which results in

$$\begin{aligned}\frac{\partial}{\partial S} g_k(S) &= \sum_{i\in[N]} \alpha_{i,k}^t (S^{-1} - S^{-1} y_{i,k} y_{i,k}^\top S^{-1}) = ((\sum_{i\in[N]} \alpha_{i,k}^t)\mathrm{Id} - S^{-1} \sum_{i\in[N]} \alpha_{i,k}^t y_{i,k} y_{i,k}^\top) S^{-1} \\ &= c\left(\mathrm{Id} - S^{-1} C\right) S^{-1}\end{aligned}$$

where Id is the identity matrix, $c = \sum_{i\in[N]} \alpha_{i,k}^t$ and $C = \dfrac{1}{c}\sum_{i\in[N]} \alpha_{i,k}^t y_{i,k} y_{i,k}^\top$ is the weighted average of empirical covariance matrices. Optimality condition $\dfrac{\partial}{\partial S} g_k(S) = \mathbf{0}$ (the null matrix) shows that $S = C$. □

EM and K-means The K-means algorithm (a.k.a Lloyd's algorithm) is closely related to the EM algorithm described above, as it performs an alternate optimization to achieve **hard-assignment** clustering. In such a context, the EM algorithm can be seen as a soft assignment clustering variant of the K-means.

In the simplest form, the K-means aims at solving the following clustering problem

$$\min_{A\in\{0,1\}^{K\times N}, \mu=(\mu_k\in\mathbb{R}^d)_{k\in[K]}} L(A,\mu) := \sum_{x_i\in\mathcal{D}} \sum_{k\in[K]} A_{k,i}|x_i - \mu_k|^2$$

where A is a binary assignment matrix (compared to the soft assignment matrix $\alpha \in [0,1]^{K\times N}$ such that each row sum to one : $\alpha\mathbf{1} = \mathbf{1}$). The algorithm then writes as follows :

- **Assignement :** μ being fixed, $\min_A L(A,\mu)$ corresponds to the following update

$$A_{k,i} = \begin{cases} 1 & \text{if } |x_i - \mu_k| \leq |x_i - \mu_j|\ \forall j \neq k \\ 0 & \text{otherwise} \end{cases}$$

  In the unlikely event that some x_i is equidistant to several centroids μ_k, a random cluster can be chosen arbitrarily.

- **Moving average :** $A_{k,i}$ being fixed, update the centroids using $\min_\mu L(A,\mu)$

$$\mu_k = \frac{\sum_{x_i\in\mathcal{D}} A_{k,i} x_i}{\sum_{x_i\in\mathcal{D}} A_{k,i}}$$

  In vectorial form $\mu = \dfrac{AX}{A\mathbf{1}} \in \mathbb{R}^{K\times d}$ where $X = [x_i \in \mathbb{R}^d]_{i\in[N]} \in \mathbb{R}^{N\times d}$ is a matrix where each row is a datapoint.

It is worth underlying that this algorithm does not guaranty an optimal solution as the problem is **NP-hard** (*i.e.* at least as difficult as any NP problem (which are solvable in polynomial time using on a nondeterministic Turing machine and verifiable in polynomial time using on a deterministic Turing machine)).

Deep latent models When considering deep models, the parameters for the parametric model are encoded by neural networks. For instance, using a multivariable Gaussian density function and two neural networks μ_θ and Σ_θ encoding respectively the mean and covariance matrix, we can define :

$$p_\theta(x|z) = \mathcal{N}(\mu_\theta(z), \Sigma_\theta(z))(x).$$

This is analogue to the GMM discussed just before, except that now the latent variable is not discrete anymore. We have an "infinite" mixture of gaussians which allows for a richer model (while removing the need to set up the number of components K).

However, as a result, both the evidence and the posterior probability becomes **intractable**, even in simple settings.

2.2 Variational Inference

Motivation In the previous paragraph devoted to the EM algorithm, we have seen that we can take advantage of the ELBO inequality to maximize a lower bound on the marginal likelihood. From the computation of the ELBO inequality (2.14) recall that for any density q, one has:

$$\log p_\theta(x) = \mathbb{E}_{z\sim q}\left[\log\left(\frac{p_\theta(x,z)}{q(z)}\right)\right] + \mathrm{KL}\left(q(z)\,||\,p_\theta(z|x)\right)$$

The EM approach requires to compute the expectation of the model total likelihood $p_\theta(x,z)$ with respect to the posterior distribution of hidden variables $q(z) = p_{\theta_t}(z|x)$ (for the model θ_t fixed at the previous time step t). We have managed to derive the computations for the gaussian mixture model, but this is not possible in general (large dataset, high dimension, etc.).

Variational Bayes Variational inference or **Variational Bayes** (VB) consists in using a **surrogate** and differentiable likelihood function $q(z) \approx p_\theta(z|x)$ to infer the posterior probability on latent variables. In Variational Auto-Encoders (VAEs) that are presented in the next section, a neural network (the encoder) is used to parametrize this function, that we denote from now $q_\phi(z|x) \approx p_\theta(z|x)$. Fo a single datapoint x, the model likelihood writes

$$\log p_\theta(x) = \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log\left(\frac{p_\theta(x,z)}{q_\phi(z|x)}\right)\right] + \mathrm{KL}\left(q_\phi(z|x)\,||\,p_\theta(z|x)\right)$$

Again, using the fact that the KL divergence is non-negative, the objective function of VB is to minimise the rhs (ELBO)

$$\begin{aligned}\min_{\theta,\phi}\mathcal{L}(\theta,\phi) :=& -\mathbb{E}_{x\sim P}\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log\left(\frac{p_\theta(x,z)}{q_\phi(z|x)}\right)\right] \\ =& -\mathbb{E}_{x\sim P}\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log\left(p_\theta(x,z)\right)\right] + \mathbb{E}_{x\sim P}\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log\left(q_\phi(z|x)\right)\right]\end{aligned} \tag{2.26}$$

where P is the data distribution.

Optimising the model's parameters θ When the posterior distribution surrogate $q_\phi(z|x)$ is fixed, optimising the objective loss function $\mathcal{L}(\theta,\phi)$ boils down to the following problem

$$\min_\theta f(\theta) = -\mathbb{E}_{x\sim P}\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log\left(p_\theta(x,z)\right)\right]$$

Recall that the joint likelihood $p_\theta(x,z) = p_\theta(z) \times p_\theta(x|z)$ can be expressed from the prior $p_\theta(z)$ and the conditional probability $p_\theta(x|z)$. This problem can therefore be addressed using gradient descent :

$$\nabla_\theta f(\theta) = -\mathbb{E}_{x\sim P}\mathbb{E}_{z\sim q_\phi(z|x)}\nabla_\theta\left[\log p_\theta(z) + \log p_\theta(x|z)\right]$$

However exact computation of the gradient is still intractable, as P is unknown. Using a few datapoint samples, one can turn to stochastic mini-batch optimization to approximate the expectation with an empirical mean : sampling a random batch of data $\mathcal{B} = \{x_{\sigma(1)}, ...x_{\sigma(B)}\} \subset \mathcal{D}$ ($\sigma : [N] \mapsto [N]$ indicates a random mapping between indices) this corresponds to :

$$\nabla_\theta f(\theta) \approx -\tfrac{1}{B}\sum_{x_b\in\mathcal{B}}\mathbb{E}_{z\sim q_\phi(z|x_b)}\nabla_\theta\left[\log p_\theta(z) + \log p_\theta(x_b|z)\right]$$

We can use the same strategy to approximate the inner expectation w.r.t the latent distribution $q_\phi(z|x_b)$. To do so, we need to get sample from the conditional distribution for each data sample in the batch. This is the role of the encoder in the VAE. Denoting $z_b \sim q_\phi(z|x_b)$ a random latent variable conditionally to x_b, we have :

$$\nabla_\theta f(\theta) \approx -\frac{1}{B} \sum_{x_b \in \mathcal{B}} \nabla_\theta \left[\log p_\theta(z_b) + \log p_\theta(x_b|z_b)\right] \text{ where } z_b \sim q_\phi(.|x_b) \text{ and } x_b \sim P$$

Assuming that the joint distribution $p_\theta(x,z)$ is differentiable, stochastic gradient descent techniques can be used to optimize parameters θ. As we will see in the next section devoted to variational auto-encoders, in practice the prior distribution is quite simple (*i.e.* with fixed parameters, such as uniform $\mathcal{U}([0,1]^\ell)$ or a white gaussian $\mathcal{N}(\mathbf{0}, \mathrm{Id})$) The conditional distribution $p_\theta(x|z)$ is

Optimising the posterior distribution's parameters ϕ with the reparametrization trick The model's parameters θ being fixed, the optimization of the posterior distribution corresponds to

$$\nabla_\phi \mathcal{L}(\theta,\phi) = -\mathbb{E}_{x\sim P} \nabla_\phi \mathbb{E}_{z\sim q_\phi(z|x)} \left[\log p_\theta(z) + \log p_\theta(x|z)\right] + \mathbb{E}_{x\sim P} \nabla_\phi \mathbb{E}_{z\sim q_\phi(z|x)} \left[\log\left(q_\phi(z|x)\right)\right]$$

Now the computation of the is more difficult since the expectation w.r.t the latent variable depends on parameters ϕ.

To makes the computation tractable, one can make use of the so-called "re-parametrization trick", which consists in using a change of (random) variable in such a way that the expectation does not depends on ϕ. Assuming that it exists a differentiable function $z = h_\phi(\varepsilon, x)$ depending on an auxiliary random variable ε and a fixed distribution $q_0(\varepsilon)$ such that

$$\varepsilon \sim q_0(\varepsilon) \Rightarrow z = h_\phi(\varepsilon, x) \sim q_\phi(z|x)$$

then the gradient can be computed from

$$\nabla_\phi \mathcal{L}(\theta,\phi) = -\mathbb{E}_{x\sim P} \nabla_\phi \mathbb{E}_{\varepsilon\sim q_0(\varepsilon)} \left[\log p_\theta(h_\phi(\varepsilon,x)) + \log p_\theta(x|h_\phi(\varepsilon,x))\right] + \mathbb{E}_{x\sim P} \nabla_\phi \mathbb{E}_{\varepsilon\sim q_0(\varepsilon)} \left[\log\left(q_\phi(h_\phi(\varepsilon,x)|x)\right)\right]$$

Relying again on mini-batch approximation to approximate the expectations, we can sample a random batch of data $\mathcal{B} = \{x_{\sigma(1)}, ...x_{\sigma(B)}\} \subset \mathcal{D}$ and a batch realisation of the auxiliary random variable $\varepsilon_1, ...\varepsilon_B$, we have

$$\nabla_\phi \mathcal{L}(\theta,\phi) \approx -\frac{1}{B} \sum_{x_b \in B} \nabla_\phi \mathbb{E}_{\varepsilon\sim q_0(\varepsilon)} \left[\log p_\theta(h_\phi(\varepsilon,x)) + \log p_\theta(x|h_\phi(\varepsilon,x))\right] + \mathbb{E}_{x\sim P} \nabla_\phi \mathbb{E}_{\varepsilon\sim q_0(\varepsilon)} \left[\log\left(q_\phi(h_\phi(\varepsilon,x)|x)\right)\right]$$

3 Variational Auto-Encoders

In "Auto-Encoding Variational Bayes" by Kingma and Welling **mettre citation**, a generative latent model $p_\theta(x|z)$ is defined which aims at maximizing the model evidence $p_\theta(x)$ on a training dataset. A generative latent model consists in first sampling a latent code from a prior latent distribution $p_Z(z)$ (often uniform or normalised gaussian distribution in a small latent dimension $\ell \ll d$) and second using a feed-forward neural network $\hat{x} = G_\theta(z) \in \mathbb{R}^d$ which generates fake samples $\hat{x}$. The goal is to maximize $p_\theta(x) = G_\theta \sharp p_Z(x)$ the likelihood of true samples x.

To achieve this task, the author propose to optimize the lower bound of the model evidence using variational bayes inference as described in the previous section. More specifically they introduce an auto-encoder neural network whose parameters are optimized with stochastic gradient descent.

In the resulting "Variational Auto-Encoders" (VAEs) architecture, the encoder E_ϕ and the decoder G_θ are used to parametrize the model conditional probability distribution and the surrogate posterior distribution. Building upon the results from the previous section devoted to Variational Bayes inference we have the following results.

Parametrizing the posterior with the encoder The encoder $E_\phi : \mathbb{R}^d \mapsto \mathbb{R}^{d'}$ implicitly parametrises the posterior distribution $q_\phi(z|x)$ with a deep neural network.

Recall that we have to sample latent codes from $q_\phi(z|x)$ to compute the lower bound with mini-batch, so we need to consider simple parametric models.

Typically, a Gaussian Multi-Variate distribution $q_\phi(z|x) = \mathcal{N}(\mu(x), \Sigma(x))(z)$ is used, and consider the separable case where $\Sigma(x) = \mathrm{diag}(\sigma^2(x)) = \mathrm{Id}\ \sigma^2_\phi(x)$ is a diagonal matrix whose diagonal is $\sigma^2(x)$. In such a case, the mean and the covariance of $q_\phi(z|x)$ are the output of the encoding neural network (so that output dimension is twice the latent dimension $d' = 2\ell$) :

$$(\mu_\phi(x), \sigma^2_\phi(x)) = E_\phi(x)$$

To sample $Z \sim q_\phi(z|x)$ from the posterior distribution, one can use the following change of variable (see the "reparametrization trick" in the previous section)

$$Z = \mu_\phi(x) + \sigma_\phi(x) \odot \varepsilon \sim q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \mathrm{Id}\ \sigma^2_\phi(x))(z) \quad \text{with } \varepsilon \sim \mathcal{N}(\mathbf{0}, \mathrm{Id})$$

where $\odot$ indicates the component-wise multiplication of two vectors. Note that this is a separable joint distribution (the covariance is diagonal) meaning that the latent variables components are therefore independant. However, the joint distribution is not isotropic, since each component has its own variance.

Parametrizing the latent model with the decoder The decoder is defined from the generative feed-forward model $G_\theta : \mathbb{R}^\ell \mapsto \mathbb{R}^d$ which maps latent variables z to samples $\hat{x} = G_\theta(z)$. Now, we need to defined parametric probability function of the model distribution $p_\theta(x, z) = p_\theta(z)p_\theta(x|z)$ to compute the loss function on real samples x.

Recall that latent variables are drawn from a prior distribution $p_\theta(z)$, which again should be simple but also match (or more precisely be compatible with) the posterior distribution from the encoder $q_\phi(z|x)$ which is gaussian. In practice, this prior does not even use model's parameter, with typically

$$p_\theta(z) = \mathcal{N}(\mathbf{0}, \mathrm{Id})(z)$$

Thus we need to define a parametric probability distribution for $p_\theta(x|z)$. For instance, we can again rely on a Gaussian distribution and choose

$$p_\theta(x|z) = \mathcal{N}(G_\theta(z), s^2\mathrm{Id})(x)$$

where the expectation/mean is the output of the generative network for the latent code z, and s^2 is the variance of the fake samples, which is an hyper-parameter. Of course, other choice may be used and will be discussed later on.

Defining the objective loss function Recall from Variational Bayes (VB) in (2.26) that the objective loss function writes (to be minimized using the negative log likelihood upper-bound)

$$\begin{aligned}
\mathcal{L}(\theta, \phi) &= -\mathbb{E}_{x\sim P}\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log\left(p_\theta(x, z)\right)\right] + \mathbb{E}_{x\sim P}\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log\left(q_\phi(z|x)\right)\right] \\
&= -\mathbb{E}_{x\sim P}\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log\left(p_\theta(z)\right) + \log\left(p_\theta(x|z)\right)\right] + \mathbb{E}_{x\sim P}\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log\left(q_\phi(z|x)\right)\right] \\
&= \mathrm{KL}(q_\phi(z|x)\,||\,p_\theta(z)) - \mathbb{E}_{x\sim P}\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log\left(p_\theta(x|z)\right)\right]
\end{aligned} \tag{2.27}$$

This last expression has an interesting variational interpretation for the auto-encoder.

- Minimizing the first term drives the encoder to reduce the discrepancy (measured by KL divergence) between the prior distribution of latent variables $p_\theta(z)$ used to generate fake samples and the encoded latent representation $q_\phi(z|x)$ from the true data samples x. This can be interpreted as a **latent regularization** term.

- The minimization of the second term (which is a negative log likelihood) drives the generative model to increase the likelihood of generated samples from the encoded latent representation of true datapoint samples

Deriving the objective loss function More precisely, the objective loss function with the proposed gaussian distributions writes (Cst indicates constants w.r.t θ and ϕ)

$$
\begin{aligned}
\mathcal{L}(\theta,\phi) =& -\mathbb{E}_{x\sim P}\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log\left(p_\theta(z)\right)+\log\left(p_\theta(x|z)\right)\right]\\
&+\mathbb{E}_{x\sim P}\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log\left(q_\phi(z|x)\right)\right]\\
\mathcal{L}(\theta,\phi) =& -\mathbb{E}_{x\sim P}\mathbb{E}_{z\sim q_\phi(z|x)}\left[-\tfrac{1}{2}\|z\|^2-\tfrac{1}{2s^2}\|x-G_\theta(z)\|^2+\mathrm{Cst}\right]\\
&+\mathbb{E}_{x\sim P}\mathbb{E}_{z\sim q_\phi(z|x)}\left[-\tfrac{1}{2}\sum_{k\in[\ell]}\log(\sigma_{\phi,k}(x)^2)-\tfrac{1}{2\sigma_{\phi,k}(x)^2}(z_k-\mu_{\phi,k}(x))^2+\mathrm{Cst}\right]\\
\mathcal{L}(\theta,\phi) =& \mathbb{E}_{x\sim P}\,\mathbb{E}_{\substack{\varepsilon\sim\mathcal{N}(\mathbf{0},\mathrm{Id})\\ z=\mu_\phi(x)+\varepsilon\odot\sigma_\phi(x)}}\tfrac{1}{2}\left[\|z\|^2+\tfrac{1}{s^2}\|x-G_\theta(z)\|^2-\langle\,\mathbf{1}\,,\log(\sigma_\phi(x)^2)\,\rangle-\|(z-\mu_\phi(x))\oslash\sigma_{\phi,k}(x)\|^2\right]+Cst\\
\mathcal{L}(\theta,\phi) =& \tfrac{1}{2}\mathbb{E}_{x\sim P}\,\mathbb{E}_{\substack{\varepsilon\sim\mathcal{N}(\mathbf{0},\mathrm{Id})\\ z=\mu_\phi(x)+\varepsilon\odot\sigma_\phi(x)}}\left[\|z\|^2+\tfrac{1}{s^2}\|x-G_\theta(z)\|^2-\langle\,\mathbf{1}\,,\log(\sigma_\phi(x)^2)\,\rangle-\|\varepsilon\|^2\right]+Cst
\end{aligned}
\tag{2.28}
$$

where $\odot$ and $\oslash$ indicates component-wise multiplication and division operations, $\sigma_\phi(x)^2$ is a vector which components are $\sigma_{\phi,k}(x)^2$, $\langle\,x\,,\,y\,\rangle=\sum_i x_iy_i$ is the dot product. In the last equation we have explicitly used the change of variable $z=\mu_\phi(x)+\varepsilon\odot\sigma_\phi(x)$ to ensure that $z\sim q_\phi(z|x)$ by sampling $\varepsilon\sim\mathcal{N}(\mathbf{0},\mathrm{Id})$. Thanks to this we can simplify the expression. First we have for the expectation of $\|\varepsilon\|^2$:

$$
\begin{aligned}
\mathbb{E}_{\varepsilon\sim\mathcal{N}(\mathbf{0},\mathrm{Id})}\left[\|\varepsilon\|^2\right] &= \sum_{k=1}^{\ell}\mathbb{E}_{\varepsilon_k\sim\mathcal{N}(0,1)}\left[\varepsilon_k^2\right]\\
&= \sum_{k=1}^{\ell}1=\ell
\end{aligned}
$$

Then, for the expectation of $\|z\|^2$:

$$
\begin{aligned}
\mathbb{E}_{z\sim\mathcal{N}(\mu_\phi(x),\mathrm{Id}\sigma_\phi^2(x))}\left[\|z\|^2\right] &= \mathbb{E}_{\varepsilon\sim\mathcal{N}(\mathbf{0},\mathrm{Id})}\left[\|\mu_\phi(x)+\sigma_\phi(x)\odot\varepsilon\|^2\right]\\
&= \mathbb{E}_{\varepsilon\sim\mathcal{N}(\mathbf{0},\mathrm{Id})}\left[\sum_{k=1}^{\ell}(\mu_{\phi,k}(x)+\varepsilon_k\sigma_{\phi,k}(x))^2\right]\\
&= \sum_{k=1}^{\ell}\mathbb{E}_{\varepsilon_k\sim\mathcal{N}(0,1)}\left[\mu_{\phi,k}(x)^2+2\mu_{\phi,k}(x)\sigma_{\phi,k}(x)\varepsilon_k+\varepsilon_k^2\sigma_{\phi,k}(x)^2\right]\\
&= \sum_{k=1}^{\ell}\left[\mu_{\phi,k}(x)^2+0+1\times\sigma_{\phi,k}(x)^2\right]\\
&= \|\mu_\phi(x)\|^2+\|\sigma_\phi(x)\|^2
\end{aligned}
$$

Finally, we get

$$
\boxed{\mathcal{L}(\theta,\phi) =\tfrac{1}{2}\mathbb{E}_{x\sim P}\,\mathbb{E}_{\substack{\varepsilon\sim\mathcal{N}(\mathbf{0},\mathrm{Id})\\ z=\mu_\phi(x)+\varepsilon\odot\sigma_\phi(x)}}\left[\tfrac{1}{s^2}\|x-G_\theta(z)\|^2+\|\mu_\phi(x)\|^2+\|\sigma_\phi(x)\|^2-\langle\,\mathbf{1}\,,\log(\sigma_\phi(x)^2)\,\rangle\right]+Cst}
\tag{2.29}
$$

Important note : there is another way to get the same result by computing directly the Kullback-Leibler divergence between separable gaussian in (2.27) as done in Eq. (2.12).

Interpretation This relatively simple expression is quite similar to the objective loss function of a conventional auto-encoder.

Indeed, the first term $\|x - G_\theta(z)\|^2$ is an MSE loss between a data-sample and a generated sample by the decoder $G_\theta(z)$. This is the only term used to optimize the parameters θ of the decoder. However, there is a subtle difference with auto-encoder where z is direclty the encoded representation of x in the latent space : in VAEs, z is sampled *conditionally* to x using $z = \mu_\phi(x) + \varepsilon \odot \sigma_\phi(x)$ by modelling a gaussian posterior distribution $q(z|x) = \mathcal{N}(\mu_\phi(x), \mathrm{Id}\sigma_\phi(x)^2)$ which is parametrized by the encoder $E_\phi(x) = (\mu_\phi(x), \sigma_\phi(x))$.

The rest of the loss only depends on the encoder parameters ϕ. It corresponds (up to a constant) to the KL divergence of the prior distribution $p_\theta(z) = \mathcal{N}(\mathbf{0}, \mathrm{Id})$ relatively to the posterior distribution $q(z|x)$ (see the KL distance between two gaussians in Eq. (2.12)). Minimizing this term allows the encoder to match the desired latent distribution and makes it possible to sample directly from the latent distribution when the training is over !

Minimization of the objective loss function Using neural networks to encode the decoder G_θ and the encoder E_ϕ, the gradient of each terms in the loss function (2.29) can be carried out by auto-differentiation.

3.1 Practical guidelines setting for VAE

use sum rather than mean over batch to deal with the change of dimensionality between d for x and $\ell \ll d$ for z

choose scale parameter s

encoding $\log \sigma^2$ directly to avoid having to deal with the constraint $\sigma > 0$

Objective loss function with closed form

Algorithm

sampling whitout additive noise (simply use the expected mean)