Unsupervised Probabilistic Learning with Latent Variables

Lecture 1: Exact Expectation-Maximisation Algorithms and Variational Autoencoders

Machine Learning Summer School Africa 2023, Cape Town

Xavier Alameda-Pineda

RobotLearn team, Inria at University Grenoble-Alpes,
Jean-Kuntzman CNRS Laboratory, Multidisciplinary Institute of Artificial Intelligence

Why unsupervised learning?

Imagine a system for {person tracking, speech denoising, body pose estimation, ...} in:

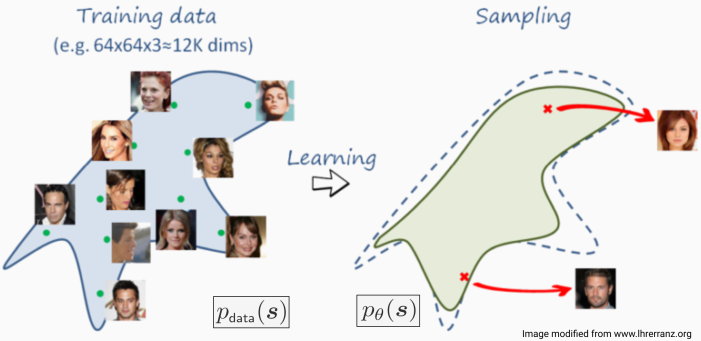


[Images under Creative Commons license]

We would like to avoid re-annotating for every new environment $\rightarrow$ Interest in unsupervised learning (and/or unsupervised domain adaptation).

And probabilistic learning?

Probabilistic generative models aim to learn a (parametric) *distribution* $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ that approximates the complex data distribution $p_{\text{data}}(\boldsymbol{x})$:



- Once learned, we can (ideally) sample new data.
- We can jointly learn them with other probabilistic models using *maximum likelihood*.

The Kullback-Leibler divergence and the ML formulation

The Kullback-Leilbler (KL) divergence between two distributions writes:

$$D_{\mathsf{KL}}(p(\boldsymbol{x})\|q(\boldsymbol{x})) = -\mathbb{E}_{p(\boldsymbol{x})}\left[\log\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right] = -\int_{\mathcal{X}} p(\boldsymbol{x})\log\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\mathrm{d}\boldsymbol{x}\begin{cases}\geq 0\\ \neq D_{\mathsf{KL}}(q(\boldsymbol{x})\|p(\boldsymbol{x}))\end{cases}$$

$D_{\mathsf{KL}}(p(\boldsymbol{x})\|q(\boldsymbol{x})) = 0 \Leftrightarrow p(\boldsymbol{x}) = q(\boldsymbol{x})$.

The Kullback-Leibler divergence and the ML formulation

The Kullback-Leilbler (KL) divergence between two distributions writes:

$$D_{\mathsf{KL}}(p(\boldsymbol{x})\|q(\boldsymbol{x})) = -\mathbb{E}_{p(\boldsymbol{x})}\left[\log\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right] = -\int_{\mathcal{X}} p(\boldsymbol{x})\log\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\mathrm{d}\boldsymbol{x}\begin{cases}\geq 0\\ \neq D_{\mathsf{KL}}(q(\boldsymbol{x})\|p(\boldsymbol{x}))\end{cases}$$

$D_{\mathsf{KL}}(p(\boldsymbol{x})\|q(\boldsymbol{x})) = 0 \Leftrightarrow p(\boldsymbol{x}) = q(\boldsymbol{x})$.

Given a training set $\{\boldsymbol{x}_i\}_{i=1}^N, \boldsymbol{x}_i \sim p_{\mathsf{data}}(\boldsymbol{x})$, ML minimizes the KL divergence:

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}}\ D_{\mathsf{KL}}\Big(p_{\mathsf{data}}(\boldsymbol{x})\ \|\ p_{\boldsymbol{\theta}}(\boldsymbol{x})\Big) = \operatorname*{argmin}_{\boldsymbol{\theta}}\ -\mathbb{E}_{p_{\mathsf{data}}}\left[\log\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x})}{p_{\mathsf{data}}(\boldsymbol{x})}\right]$$

$$= \operatorname*{argmax}_{\boldsymbol{\theta}}\ \mathbb{E}_{p_{\mathsf{data}}}\Big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})\Big] \approx \boxed{\operatorname*{argmax}_{\boldsymbol{\theta}}\ \frac{1}{N}\sum_{i=1}^{N}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i)}$$

Interest of Latent Variables

- Speech enhancement: noisy speech (observation), clean speech (latent variable)
  $\rightarrow$ Get rid of noise heard together with speech.
- Person tracking: detections (observation), person positions (latent variable)
  $\rightarrow$ Track occluded persons, filter over time.
- Representation learning: raw data (observation), representation (latent variable)
  $\rightarrow$ Compute an optimal compact representation of the raw data.

Interest of Latent Variables

- Speech enhancement: noisy speech (observation), clean speech (latent variable)
  $\rightarrow$ Get rid of noise heard together with speech.
- Person tracking: detections (observation), person positions (latent variable)
  $\rightarrow$ Track occluded persons, filter over time.
- Representation learning: raw data (observation), representation (latent variable)
  $\rightarrow$ Compute an optimal compact representation of the raw data.

Let $\boldsymbol{z}$ denote the latent variable:

$$\begin{cases} \boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}) \\ \boldsymbol{x}|\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \end{cases} \rightarrow \; p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) \mathrm{d}\boldsymbol{z} = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z}) \mathrm{d}\boldsymbol{z}$$

New samples: Draw $\boldsymbol{z}_k \sim p_{\boldsymbol{\theta}}(\boldsymbol{z})$, then draw a new sample $\boldsymbol{x}_k \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}_k)$.

Interest of Latent Variables

- Speech enhancement: noisy speech (observation), clean speech (latent variable)
  $\rightarrow$ Get rid of noise heard together with speech.
- Person tracking: detections (observation), person positions (latent variable)
  $\rightarrow$ Track occluded persons, filter over time.
- Representation learning: raw data (observation), representation (latent variable)
  $\rightarrow$ Compute an optimal compact representation of the raw data.

Let $\boldsymbol{z}$ denote the latent variable:

$$\begin{cases} \boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}) \\ \boldsymbol{x}|\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \end{cases} \rightarrow \; p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) \mathrm{d}\boldsymbol{z} = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z}) \mathrm{d}\boldsymbol{z}$$

New samples: Draw $\boldsymbol{z}_k \sim p_{\boldsymbol{\theta}}(\boldsymbol{z})$, then draw a new sample $\boldsymbol{x}_k \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}_k)$.

Learning: How to estimate $\boldsymbol{\theta}$ with the ML formulation?

Lecture Organisation

- Lecture 1: Probabilistic learning with latent variables, GMM and the exact EM algorithm, probabilistic PCA and VAE.
- Lecture 2: The variational EM algoritm, mixtures of VAEs, application to audio-visual speech enhancement.
- Lecture 3: Dynamical variational autoencoders, application to speech enhancement.

Connections to other lectures:

- Matthias Bauer: from PPCA/VAE to diffusion models.
- Steve Kroon: normalising flows and their use with VAEs.

I am fully available for discussing/chatting with y'all!

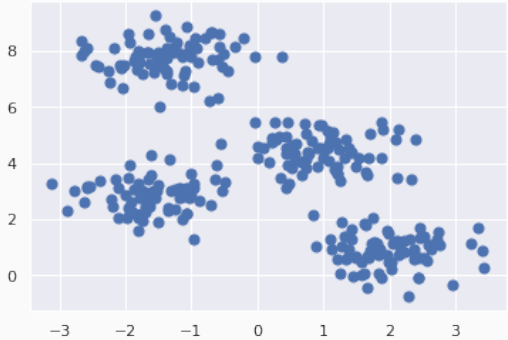
Lecture's Outline

1. Learning with Latent Variables: the Gaussian Mixture Model
2. EM for GMM and beyond
3. Probabilistic PCA and VAE
4. VAE for audio modeling

Learning with Latent Variables: the Gaussian Mixture Model

Simple example: clustering

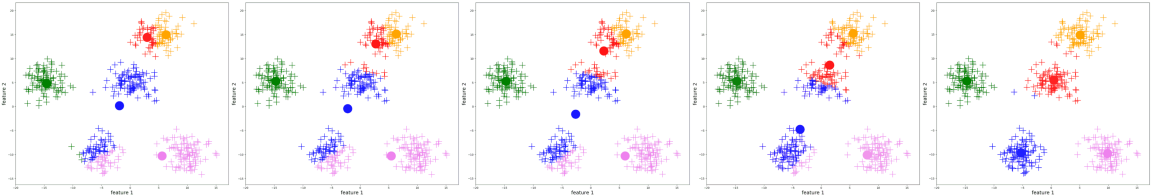
Definition: find groups of data points without labels.



Very intuitive algorithm

Very simple algorithm (called the K-means algorithm):

1. Initialise randomly K centroids.
2. Assign each data point to the closes centroid.
3. Recompute centroids from the assignments.
4. Iterate the past two steps.



[Images from `https://ai.plainenglish.io/`]

Important points of K-means

The point-to-cluster assignment variable is unknown (latent, z), and must be infered with the centroids (parameters of the model).

The assignment criterion is the Euclidean distance $\Rightarrow$ groups are spherical and equally populated a priori.

Generalising K-means: GMM

Defining the Gaussian mixture model (GMM)

- For each $\boldsymbol{x}_n$ there is a latent variable $\boldsymbol{z}_n$ taking values from 1 to K: $\boldsymbol{z}_n \in \{1, \ldots, K\}$.

Generalising K-means: GMM

Defining the Gaussian mixture model (GMM)

- For each $\boldsymbol{x}_n$ there is a latent variable $\boldsymbol{z}_n$ taking values from 1 to K: $\boldsymbol{z}_n \in \{1, \ldots, K\}$.
- Its prior probability is defined as: $p(\boldsymbol{z}_n = k) = \pi_k \geq 0$, with $\sum_{k=1}^{K} \pi_k = 1$.

Generalising K-means: GMM

Defining the Gaussian mixture model (GMM)

- For each $\boldsymbol{x}_n$ there is a latent variable $\boldsymbol{z}_n$ taking values from 1 to K: $\boldsymbol{z}_n \in \{1, \ldots, K\}$.
- Its prior probability is defined as: $p(\boldsymbol{z}_n = k) = \pi_k \geq 0$, with $\sum_{k=1}^{K} \pi_k = 1$.
- Given z_n, the data point is modeled as a multivariate Gaussian:

$$p(\boldsymbol{x}_n | \boldsymbol{z}_n = k) = \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Advantages:

1. Having $\pi_1, \ldots, \pi_K$ means that groups can be differently populated.
2. The shape of the groups is modeled by $\boldsymbol{\Sigma}_k$.
3. The parameters are: $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$.

Maximum likelihood for GMM

Let's compute $p(\boldsymbol{x}_n)$

$$p(\boldsymbol{x}_n) = \sum_{k=1}^{K} p(\boldsymbol{x}_n, \boldsymbol{z}_n = k) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

The log-likelihood:

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{X}) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

Computing directly ML by $\dfrac{\partial \mathcal{L}}{\partial \pi_k} = 0$, $\dfrac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = 0$ or $\dfrac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_k} = 0$ is very difficult.

EM for GMM and beyond

(Expected complete-data) log-likelihood

We have seen that $\log p(\boldsymbol{x})$ does not work well with derivatives. However, $\log p(\boldsymbol{x}, \boldsymbol{z})$ does!

Problem: $\boldsymbol{z}$ is not observed, thus $\sum_n \log p(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is a random variable.

(Expected complete-data) log-likelihood

We have seen that $\log p(\boldsymbol{x})$ does not work well with derivatives. However, $\log p(\boldsymbol{x}, \boldsymbol{z})$ does!

Problem: $\boldsymbol{z}$ is not observed, thus $\sum_n \log p(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is a random variable.

Given an initial value of the parameters, $\boldsymbol{\theta}^0$, let's take the expectation w.r.t. the posterior distribution $p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}^0)$ (we will justify this choice later on):

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^0) = \sum_n \mathbb{E}_{p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}^0)} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta})$$

This function is called: *expected complete-data log-likelihood*, and is the main mathematical object when working with EM algorithms.

The EM algorithm for GMM

Given $\boldsymbol{\theta}^0$, we use the *expected complete-data log-likelihood* $\mathcal{Q}$. For iteration $r = 1, \ldots, R$:

1. Expectation [E-step]:

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{r-1}) = \mathbb{E}_{p(\boldsymbol{Z}|\boldsymbol{X};\boldsymbol{\theta}^{r-1})} \log p(\boldsymbol{X}, \boldsymbol{Z}; \boldsymbol{\theta})$$

2. Maximisation [M-step]:

$$\boldsymbol{\theta}^r = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{r-1})$$

(observations $\boldsymbol{X} = \{\boldsymbol{x}_n\}_{n=1}^N$, latent variables $\boldsymbol{Z} = \{\boldsymbol{z}_n\}_{n=1}^N$, parameters $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$)

The EM algorithm for GMM

Given $\boldsymbol{\theta}^0$, we use the *expected complete-data log-likelihood* $\mathcal{Q}$. For iteration $r = 1, \ldots, R$:

1. Expectation [E-step]:

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{r-1}) = \mathbb{E}_{p(\boldsymbol{Z}|\boldsymbol{X};\boldsymbol{\theta}^{r-1})} \log p(\boldsymbol{X}, \boldsymbol{Z}; \boldsymbol{\theta})$$

2. Maximisation [M-step]:

$$\boldsymbol{\theta}^r = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{r-1})$$

(observations $\boldsymbol{X} = \{\boldsymbol{x}_n\}_{n=1}^N$, latent variables $\boldsymbol{Z} = \{\boldsymbol{z}_n\}_{n=1}^N$, parameters $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$)

We can look back to K-means:

1. Infer latent variables (assignment) given the parameters (centroids) [E-step].
2. Estimate the parameters (centroids) given the assignments [M-step].

The EM algorithm for GMM (II)

E-step The posterior distribution writes:

$$p(\boldsymbol{z} = k|\boldsymbol{x}_n; \boldsymbol{\theta}^0) = \frac{p(\boldsymbol{z}_n = k; \boldsymbol{\theta}^0)p(\boldsymbol{x}_n|\boldsymbol{z}_n = k; \boldsymbol{\theta}^0)}{\sum_\ell p(\boldsymbol{z}_n = \ell; \boldsymbol{\theta}^0)p(\boldsymbol{x}_n|\boldsymbol{z}_n = \ell; \boldsymbol{\theta}^0)} = \frac{\pi_k^0 \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k^0, \boldsymbol{\Sigma}_k^0)}{\sum_\ell \pi_\ell^0 \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_\ell^0, \boldsymbol{\Sigma}_\ell^0)} = \eta_{nk}$$

The EM algorithm for GMM (II)

E-step The posterior distribution writes:

$$p(\boldsymbol{z} = k|\boldsymbol{x}_n; \boldsymbol{\theta}^0) = \frac{p(\boldsymbol{z}_n = k; \boldsymbol{\theta}^0)p(\boldsymbol{x}_n|\boldsymbol{z}_n = k; \boldsymbol{\theta}^0)}{\sum_\ell p(\boldsymbol{z}_n = \ell; \boldsymbol{\theta}^0)p(\boldsymbol{x}_n|\boldsymbol{z}_n = \ell; \boldsymbol{\theta}^0)} = \frac{\pi_k^0 \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k^0, \boldsymbol{\Sigma}_k^0)}{\sum_\ell \pi_\ell^0 \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_\ell^0, \boldsymbol{\Sigma}_\ell^0)} = \eta_{nk}$$

The expected complete-data log-likelihood writes:

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^0) = \sum_{n=1}^{N} \sum_{k=1}^{K} \eta_{nk} \log \pi_k \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

The EM algorithm for GMM (II)

E-step The posterior distribution writes:

$$p(\boldsymbol{z} = k|\boldsymbol{x}_n; \boldsymbol{\theta}^0) = \frac{p(\boldsymbol{z}_n = k; \boldsymbol{\theta}^0)p(\boldsymbol{x}_n|\boldsymbol{z}_n = k; \boldsymbol{\theta}^0)}{\sum_\ell p(\boldsymbol{z}_n = \ell; \boldsymbol{\theta}^0)p(\boldsymbol{x}_n|\boldsymbol{z}_n = \ell; \boldsymbol{\theta}^0)} = \frac{\pi_k^0 \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k^0, \boldsymbol{\Sigma}_k^0)}{\sum_\ell \pi_\ell^0 \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_\ell^0, \boldsymbol{\Sigma}_\ell^0)} = \eta_{nk}$$

The expected complete-data log-likelihood writes:

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^0) = \sum_{n=1}^{N} \sum_{k=1}^{K} \eta_{nk} \log \pi_k \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

M-step The optimal value for $\boldsymbol{\mu}_k$ writes:

$$\boldsymbol{\mu}_k^* = \frac{1}{\sum_{n=1}^{N} \eta_{nk}} \sum_{n=1}^{N} \eta_{nk} \boldsymbol{x}_n$$

But why does the EM work?

The main mathematical object in EM is $\mathcal{Q}$ (The expected complete-data log-likelihood).

What is the relationship with the log-likelihood? Let's take any distribution of $\boldsymbol{z}$, $q(\boldsymbol{z})$:

$$
\begin{aligned}
\log p(\boldsymbol{x}) &= \mathbb{E}_{q(\boldsymbol{z})}\Big[\log p(\boldsymbol{x})\Big] \\
&= \mathbb{E}_{q(\boldsymbol{z})}\Big[\log p(\boldsymbol{x})\frac{p(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{z})}\Big] \\
&= \underbrace{\mathbb{E}_{q(\boldsymbol{z})}\Big[\log\frac{p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z})}\Big]}_{\text{M-step - }\mathcal{Q}} + \underbrace{D_{\mathsf{KL}}\Big(q(\boldsymbol{z})\Big\|p(\boldsymbol{z}|\boldsymbol{x})\Big)}_{\text{E-step - KL}}
\end{aligned}
$$

But why does the EM work? (II)

$$\log p(\boldsymbol{x};\boldsymbol{\theta}) = \underbrace{\mathbb{E}_{q(\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z})}\right]}_{\text{M-step - } \mathcal{Q}} + \underbrace{D_{\mathsf{KL}}\Big(q(\boldsymbol{z})\Big\|p(\boldsymbol{z}|\boldsymbol{x})\Big)}_{\text{E-step - KL}}$$

Another interpretation. Given $\boldsymbol{\theta}^0$:

1. Set $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^0)$, minimise KL (thus maximize $\mathcal{Q}$) w.r.t. $q(\boldsymbol{z})$. The E-step reduces the distance between log-likelihood and $\mathcal{Q}$.
2. Maximize $\mathcal{Q}$ w.r.t. $\boldsymbol{\theta}$:
$$\mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^0) = \mathbf{E}_{q(\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})}\right].$$
The M-step pushes $\mathcal{Q}$ and therefore pushes the log-likelihood.

The Exact EM

$$\log p(\boldsymbol{x};\boldsymbol{\theta}) = \underbrace{\mathbb{E}_{q(\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z})}\right]}_{\text{M-step - } \mathcal{Q}} + \underbrace{D_{\mathsf{KL}}\Big(q(\boldsymbol{z})\Big\|p(\boldsymbol{z}|\boldsymbol{x})\Big)}_{\text{E-step - KL}}$$

Given $\boldsymbol{\theta}^0$:

1. Set $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^0)$, and compute $\mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^0)$.
2. Maximize $\mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^0)$ w.r.t. $\boldsymbol{\theta}$.

The Exact EM

$$\log p(\boldsymbol{x};\boldsymbol{\theta}) = \underbrace{\mathbb{E}_{q(\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z})}\right]}_{\text{M-step - } \mathcal{Q}} + \underbrace{D_{\mathsf{KL}}\Big(q(\boldsymbol{z})\Big\|p(\boldsymbol{z}|\boldsymbol{x})\Big)}_{\text{E-step - KL}}$$

Given $\boldsymbol{\theta}^0$:

1. Set $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^0)$, and compute $\mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^0)$.
2. Maximize $\mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^0)$ w.r.t. $\boldsymbol{\theta}$.

Several potential issues:

- The posterior distribution does not exist/is not computationally tractable.
- The expectation cannot be taken analytically.
- The maximization w.r.t. $\boldsymbol{\theta}$ does not have close-form solutions.

Probabilistic PCA and VAE

Probabilistic Principal Component Analysis (PCA)

Reduce the data dimensionality and extract a compact representation ($\boldsymbol{z}$) of each datum ($\boldsymbol{x}$).

Probabilistic version of PCA ($D \ll F$):

$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I}), \quad \boldsymbol{z} \in \mathbb{R}^D.$$

$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{A}\boldsymbol{z} + \boldsymbol{b}, \nu\boldsymbol{I}), \quad \boldsymbol{x} \in \mathbb{R}^F.$$

Probabilistic Principal Component Analysis (PCA)

Reduce the data dimensionality and extract a compact representation ($\boldsymbol{z}$) of each datum ($\boldsymbol{x}$).

Probabilistic version of PCA ($D \ll F$):

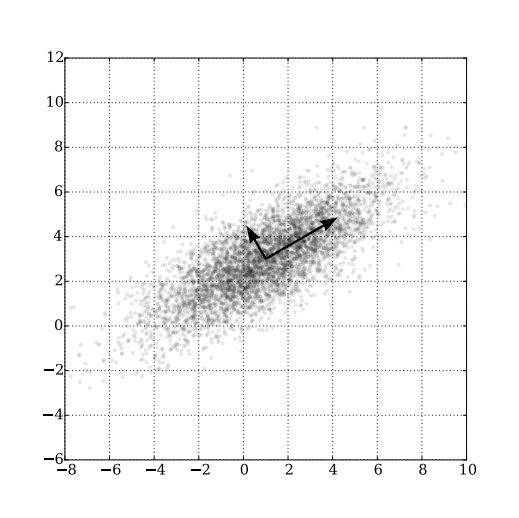
$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I}), \quad \boldsymbol{z} \in \mathbb{R}^D.$$

$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{A}\boldsymbol{z} + \boldsymbol{b}, \nu\boldsymbol{I}), \quad \boldsymbol{x} \in \mathbb{R}^F.$$

Posterior distribution $\leftrightarrow$ data projection:

$$p(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\theta}^0) = \mathcal{N}(\boldsymbol{z}_n; \boldsymbol{W}\boldsymbol{x}_n + \boldsymbol{c}, \boldsymbol{\Omega}),$$

($\boldsymbol{W}$ and $\boldsymbol{\Omega}$ depend on $\boldsymbol{\theta}^0$)



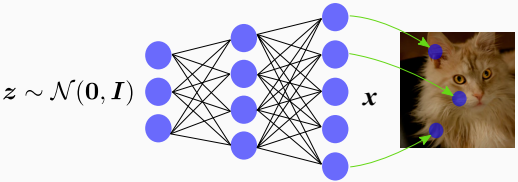
[Image from Wikimedia Commons]

Comments on PPCA - Toward VAE

- It is possible to derive an exact EM algorithm $\rightarrow$ local convergence guaranteed.
- Can only model linear dependencies $\rightarrow$ limited expressivity.

Non-linear Gaussian model: $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}\Big(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{z})\Big)$

- $\boldsymbol{\mu}_{\boldsymbol{\theta}}(.), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(.)$: Non-linear functions implemented as deep nets
- Parameter estimation is challenging, but much more expressive



The non-linear Gaussian model can be optimised via the variational autoencoder (VAE).

Formalising the generative model (I)

How can we ensure that $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{z})$ is a covariance matrix?

- The covariance matrix is assumed to be diagonal:

$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{z}) = \begin{pmatrix} \nu_{\boldsymbol{\theta},1}(\boldsymbol{z}) & 0 & \cdots & 0 \\ 0 & \nu_{\boldsymbol{\theta},2}(\boldsymbol{z}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \nu_{\boldsymbol{\theta},F}(\boldsymbol{z}) \end{pmatrix}$$

  Reduces complexity and memory, but also expressivity.

- We estimate the log-variance: $\eta_{\boldsymbol{\theta},f}(\boldsymbol{z}) = \log \nu_{\boldsymbol{\theta},f}(\boldsymbol{z})$:

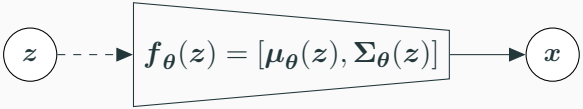
$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{z}) = \mathrm{diag}_f \left(\exp \left(\eta_{\boldsymbol{\theta},f}(\boldsymbol{z}) \right) \right)$$

  The values of $\eta_{\boldsymbol{\theta},f}(\boldsymbol{z})$ can be positive or negative.

Formalising the generative model (II)

In terms of probabilistic dependencies, they are the same as PPCA:

But we can also draw the non-lineariy:



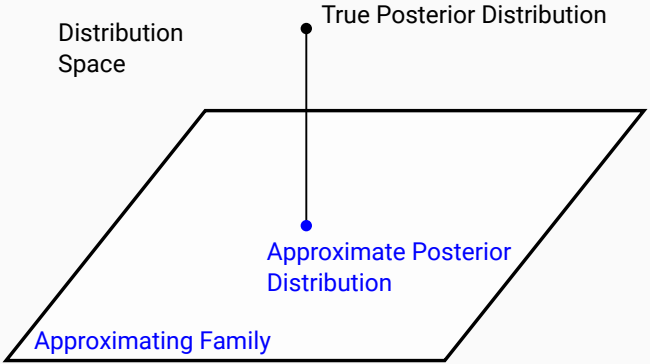
The dependency of the parameters w.r.t. $\boldsymbol{z}$ is **deterministic**.

Denoted by $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{z}) : \mathbb{R}^D \to \mathbb{R}^{2F}$, this non-linearity is implemented with a deep network, with parameters (weights and biases) $\boldsymbol{\theta}$.

The posterior distribution

Since $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^0)$ cannot be computed analytically, it needs to be approximated.

The idea is to find the **best candidate within a family of distributions**.



Approximating the posterior distribution

The posterior distribution will be approximated with **another** feed-forward network parametrised with $\boldsymbol{\phi}$:

$$p(\boldsymbol{z}|\boldsymbol{x}) \approx q(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \tilde{\boldsymbol{\mu}}_{\boldsymbol{\phi}}(\boldsymbol{x}), \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\phi}}(\boldsymbol{x}))$$

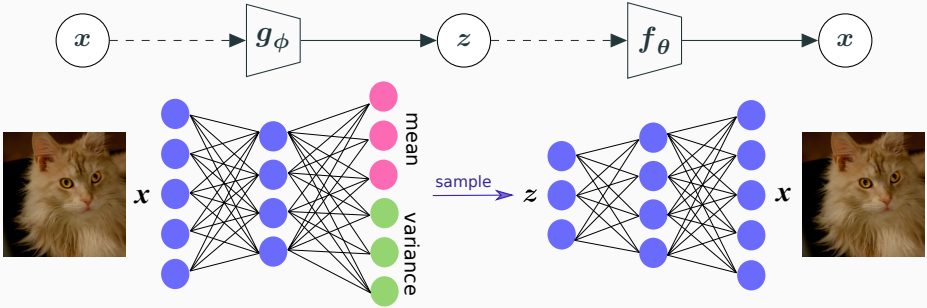
The approximating family is composed of all the distributions that can be expressed as above, for a certain value of $\boldsymbol{\phi}$.

$$\mathcal{G} = \{\boldsymbol{g}_{\boldsymbol{\phi}} : \mathbb{R}^F \to \mathbb{R}^{2D}; \boldsymbol{\phi} \in \boldsymbol{\Phi}\},$$

with $\boldsymbol{g}_{\boldsymbol{\phi}}(\boldsymbol{x}) = [\tilde{\boldsymbol{\mu}}_{\boldsymbol{\phi}}(\boldsymbol{x}), \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\phi}}(\boldsymbol{x})]$.

Overall architecture

If we "chain" the posterior (encoder) and the generative (decoder) model:



This is why we call these architectures **variational autoencoders** VAE.

But how do we optimise for the parameters $\boldsymbol{\theta}$ and ϕ?

Learning - ELBO

If we recall the formulation for the EM:

$$\log p(\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})}\right] + D_{\mathsf{KL}}\Big(q(\boldsymbol{z}|\boldsymbol{x})\Big\|p(\boldsymbol{z}|\boldsymbol{x})\Big)$$

Learning - ELBO

If we recall the formulation for the EM:

$$\log p(\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})}\right] + D_{\mathsf{KL}}\Big(q(\boldsymbol{z}|\boldsymbol{x})\Big\|p(\boldsymbol{z}|\boldsymbol{x})\Big)$$

Problem: the second term cannot be computed! But it's positive:

$$\log p(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi}) \geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right]$$

$$\log p(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi}) \geq \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\Big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\Big]}_{\text{Reconstruction}} - \underbrace{D_{\mathsf{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\Big\|p(\boldsymbol{z})\Big)}_{\text{Regularisation}}$$

This is known as **Evidence Lower-BOund or ELBO**: $\mathcal{L}_{\mathsf{ELBO}}(\boldsymbol{\theta},\boldsymbol{\phi})$.

Be VERY careful with these expressions: They look alike, but they are NOT the same.

Learning - Sampling

But we still have one problem:

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\Big\{\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\Big\}}_{\text{Reconstruction}} - \underbrace{D_{\text{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\Big\|p(\boldsymbol{z})\Big)}_{\text{Regularisation}}$$

To compute the "reconstruction" term we need to take the expectation w.r.t. $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, but recall that:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{z})).$$

Due to the non-linearity, we cannot compute the reconstruction term in closed form $\rightarrow$ we sample R points $\hat{\boldsymbol{z}}^{(1)}, \ldots, \hat{\boldsymbol{z}}^{(R)}$ from $q_{\boldsymbol{\phi}}$:

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \underbrace{\frac{1}{R}\sum_{r=1}^{R} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\hat{\boldsymbol{z}}^{(r)})}_{\text{Reconstruction}} - \underbrace{D_{\text{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\Big\|p(\boldsymbol{z})\Big)}_{\text{Regularisation}}$$

Learning - Gradient ascent?

Let's go back to the architecture:

Since there is no closed-form solution for the parameters (due to the non-linearity), we will learn the parameters using stochastic gradient ascent (to maximise the ELBO).

We assume that $\boldsymbol{g}_{\boldsymbol{\phi}}$ and $\boldsymbol{f}_{\boldsymbol{\theta}}$ are differentiable (we can compute the gradient).

The sampling operation ($\hat{\boldsymbol{z}}^{(r)}$) from $q_{\boldsymbol{\phi}}$ is NOT differentiable w.r.t. $\boldsymbol{\phi}$.

Learning - Reparametrisation trick

Instead of sampling directly from the posterior ($\hat{\boldsymbol{z}}^{(r)} \sim q_{\boldsymbol{\phi}} = \mathcal{N}(\tilde{\boldsymbol{\mu}}_{\boldsymbol{\phi}}, \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\phi}})$) we sample as follows:

$$\bar{\boldsymbol{z}}^{(r)} = \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\phi}}^{1/2} \bar{\boldsymbol{\epsilon}}^{(r)} + \tilde{\boldsymbol{\mu}}_{\boldsymbol{\phi}} \quad \text{with} \quad \bar{\boldsymbol{\epsilon}}^{(r)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

$\hat{\boldsymbol{z}}^{(r)}$ and $\bar{\boldsymbol{z}}^{(r)}$ follow the same distribution, BUT $\bar{\boldsymbol{z}}^{(r)}$ is differentiable w.r.t. $\boldsymbol{\phi}$!!!

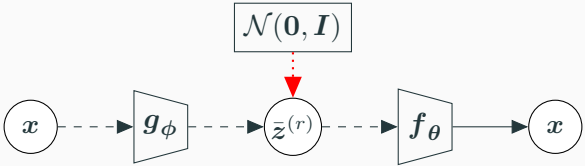
Learning - Reparametrisation trick

Instead of sampling directly from the posterior ($\hat{\boldsymbol{z}}^{(r)} \sim q_{\boldsymbol{\phi}} = \mathcal{N}(\tilde{\boldsymbol{\mu}}_{\boldsymbol{\phi}}, \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\phi}})$) we sample as follows:

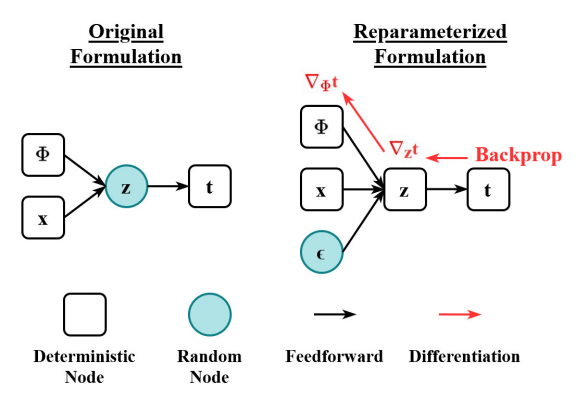
$$\bar{\boldsymbol{z}}^{(r)} = \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\phi}}^{1/2} \bar{\boldsymbol{\epsilon}}^{(r)} + \tilde{\boldsymbol{\mu}}_{\boldsymbol{\phi}} \quad \text{with} \quad \bar{\boldsymbol{\epsilon}}^{(r)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

$\hat{\boldsymbol{z}}^{(r)}$ and $\bar{\boldsymbol{z}}^{(r)}$ follow the same distribution, BUT $\bar{\boldsymbol{z}}^{(r)}$ is differentiable w.r.t. $\boldsymbol{\phi}$!!!

This is called the **reparametrisation trick** and can be used with other distributions.



Learning - Reparametrisation trick (II)



[Image from Wikimedia Commons]

Training VAE

Implementation note: Most deep learning libraries implement stochastic gradient DESCENT algorithms, but we would like to maximize the ELBO $\rightarrow$ we need to change the sign.

Training VAE

Implementation note: Most deep learning libraries implement stochastic gradient DESCENT algorithms, but we would like to maximize the ELBO $\rightarrow$ we need to change the sign.

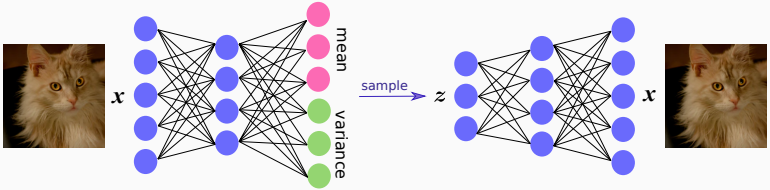
Posterior collapse: common phenomenon when the posterior q_ϕ gets to close to the standard prior. It can happen for various dimensions of $\boldsymbol{z}$ $\rightarrow$ the VAE stops learning.

- The KL term dominates the ELBO $\rightarrow$ weight the KL term with $\beta < 1$.
- The data can be reduced to less dimensions than D $\rightarrow$ decrease D.

VAE: Summary

Generative model. Prior: $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I})$ and decoder: $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}\Big(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{z})\Big)$.

Inference model (encoder): $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \approx q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\Big(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \boldsymbol{\Sigma}_{\boldsymbol{\phi}}(\boldsymbol{x})\Big)$



Training criterion (maximise the evidence lower bound):

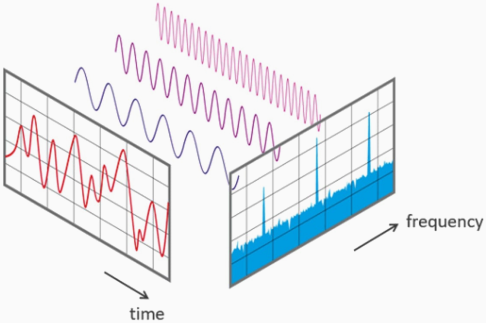
$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \underbrace{\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\hat{\boldsymbol{z}})}_{\text{Reconstruction}} - \underbrace{D_{\text{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \Big\| p(\boldsymbol{z})\Big)}_{\text{Regularisation}}$$

where $\hat{\boldsymbol{z}} \sim q_{\boldsymbol{\phi}}$ is sampled using the reparametrisation trick.

VAE for audio modeling

Representing audio: time vs. frequency

Fourier domain decomposes the signal in frequencies:

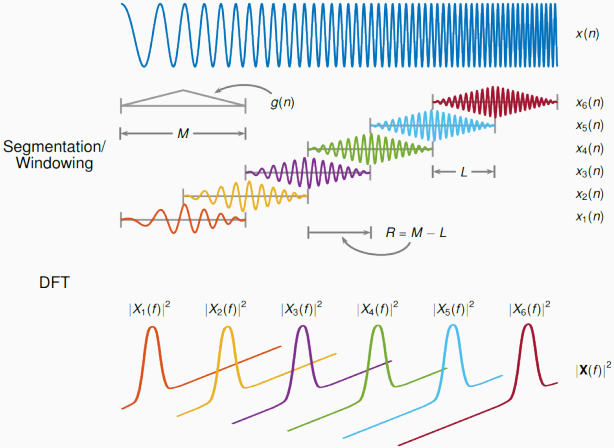


[Image from `https://dev.to/trekhleb/`]

Problem: we have to choose either time **or** frequency.

The short-time Fourier transform (STFT)

STFT: segment the input signal, and apply DFT to each segment.



[Image from Mathworks]

Let's see it!

- Sine frequency sweep (pure sine of increasing frequency).
- Leyenda (piano, harmonics).
- O mio babbino caro (opera, vibrato).
- Highway to hell (rock&roll, distortion).

Defining the VAE to model spectrograms

Each observation $\boldsymbol{x}$ will be an F-dimensional **complex** vector: $\boldsymbol{x} \in \mathbb{C}^F$.
The low-dimensional latent variable will be **real** of dimension D: $\boldsymbol{z} \in \mathbb{R}^D$.

The model:

- Prior: $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \mathbf{0}, \boldsymbol{I})$.
- Decoder: $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}_c(\boldsymbol{x}; \mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{z}))$, complex Gaussian distribution, see next.
- Posterior: $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \approx q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\Big(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \boldsymbol{\Sigma}_{\boldsymbol{\phi}}(\boldsymbol{x})\Big)$

We need to compute the reconstruction and the regularization terms.

Let's recall that the covariance matrices $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{z})$ and $\boldsymbol{\Sigma}_{\boldsymbol{\phi}}(\boldsymbol{x})$ are diagonal log-variance matrices.

ELBO for spectrogram VAEs

The centered 1D complex-normal writes:

$$\mathcal{N}_c(x; 0, \nu) = \frac{1}{\pi\nu} \exp\left(-\frac{|x|^2}{\nu}\right).$$

The 1D complex-normal for the f-th entry of $\boldsymbol{x}$ as a function of the log-var writes:

$$\log \mathcal{N}_c(x_f; 0, \exp \eta_{\boldsymbol{\theta},f}(\boldsymbol{z})) = -\log(\pi) - \eta_{\boldsymbol{\theta},f}(\boldsymbol{z}) - \frac{|x_f|^2}{\exp(\eta_{\boldsymbol{\theta},f}(\boldsymbol{z}))}.$$

For a given sample $\hat{\boldsymbol{z}}$, the reconstruction term writes:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\hat{\boldsymbol{z}}) = \sum_f \log p_{\boldsymbol{\theta}}(x_f|\hat{\boldsymbol{z}}) = -F\log(\pi) - \sum_f \left(\eta_{\boldsymbol{\theta},f}(\hat{\boldsymbol{z}}) + \frac{|x_f|^2}{\exp(\eta_{\boldsymbol{\theta},f}(\hat{\boldsymbol{z}}))}\right).$$

ELBO for spectrogram VAEs (II)

The KL between two D-dimensional real Gaussian distributions writes:

$$D_{\mathrm{KL}}(\mathcal{N}_0 \| \mathcal{N}_1) = \frac{1}{2}\left(\mathrm{Tr}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0) - D + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) + \log\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_0|}\right)$$

In our case: $\boldsymbol{\mu}_0 = \tilde{\boldsymbol{\mu}}_{\boldsymbol{\phi}}(\boldsymbol{x})$, $\boldsymbol{\Sigma}_0 = \mathrm{diag}_d\left(\exp\left(\tilde{\eta}_{\boldsymbol{\phi},d}(\boldsymbol{x})\right)\right)$, $\boldsymbol{\mu}_1 = \mathbf{0}$ and $\boldsymbol{\Sigma}_1 = \boldsymbol{I}$. Thus:

$$D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z})) = \frac{1}{2}\left(\sum_d \exp\left(\tilde{\eta}_{\boldsymbol{\phi},d}(\boldsymbol{x})\right) + |\tilde{\mu}_{\boldsymbol{\phi},d}(\boldsymbol{x})|^2 - \tilde{\eta}_{\boldsymbol{\phi},d}(\boldsymbol{x})\right) - \frac{D}{2}$$

ELBO for spectrogram VAEs (II)

The KL between two D-dimensional real Gaussian distributions writes:

$$D_{\mathrm{KL}}(\mathcal{N}_0 \| \mathcal{N}_1) = \frac{1}{2}\left(\mathrm{Tr}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0) - D + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) + \log\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_0|}\right)$$

In our case: $\boldsymbol{\mu}_0 = \tilde{\boldsymbol{\mu}}_{\boldsymbol{\phi}}(\boldsymbol{x})$, $\boldsymbol{\Sigma}_0 = \mathrm{diag}_d\left(\exp\left(\tilde{\eta}_{\boldsymbol{\phi},d}(\boldsymbol{x})\right)\right)$, $\boldsymbol{\mu}_1 = \mathbf{0}$ and $\boldsymbol{\Sigma}_1 = \boldsymbol{I}$. Thus:

$$D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z})) = \frac{1}{2}\left(\sum_d \exp\left(\tilde{\eta}_{\boldsymbol{\phi},d}(\boldsymbol{x})\right) + |\tilde{\mu}_{\boldsymbol{\phi},d}(\boldsymbol{x})|^2 - \tilde{\eta}_{\boldsymbol{\phi},d}(\boldsymbol{x})\right) - \frac{D}{2}$$

Therefore the ELBO (w/o constant terms) writes:

$$\mathcal{L}_{\mathsf{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = -\sum_f \left(\eta_{\boldsymbol{\theta},f}(\hat{\boldsymbol{z}}) + \frac{|x_f|^2}{\exp(\eta_{\boldsymbol{\theta},f}(\hat{\boldsymbol{z}}))}\right) - \frac{1}{2}\left(\sum_d \exp\left(\tilde{\eta}_{\boldsymbol{\phi},d}(\boldsymbol{x})\right) + |\tilde{\mu}_{\boldsymbol{\phi},d}(\boldsymbol{x})|^2 - \tilde{\eta}_{\boldsymbol{\phi},d}(\boldsymbol{x})\right),$$

which is what we give to our optimizer.

ELBO for spectrogram VAEs (II)

The KL between two D-dimensional real Gaussian distributions writes:

$$D_{\mathrm{KL}}(\mathcal{N}_0 \| \mathcal{N}_1) = \frac{1}{2}\left(\mathrm{Tr}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0) - D + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) + \log\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_0|}\right)$$

In our case: $\boldsymbol{\mu}_0 = \tilde{\boldsymbol{\mu}}_{\boldsymbol{\phi}}(\boldsymbol{x})$, $\boldsymbol{\Sigma}_0 = \mathrm{diag}_d\left(\exp\left(\tilde{\eta}_{\boldsymbol{\phi},d}(\boldsymbol{x})\right)\right)$, $\boldsymbol{\mu}_1 = \mathbf{0}$ and $\boldsymbol{\Sigma}_1 = \boldsymbol{I}$. Thus:

$$D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z})) = \frac{1}{2}\left(\sum_d \exp\left(\tilde{\eta}_{\boldsymbol{\phi},d}(\boldsymbol{x})\right) + |\tilde{\mu}_{\boldsymbol{\phi},d}(\boldsymbol{x})|^2 - \tilde{\eta}_{\boldsymbol{\phi},d}(\boldsymbol{x})\right) - \frac{D}{2}$$

Therefore the ELBO (w/o constant terms) writes:

$$\mathcal{L}_{\mathsf{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = -\sum_f \left(\eta_{\boldsymbol{\theta},f}(\hat{\boldsymbol{z}}) + \frac{|x_f|^2}{\exp(\eta_{\boldsymbol{\theta},f}(\hat{\boldsymbol{z}}))}\right) - \frac{1}{2}\left(\sum_d \exp\left(\tilde{\eta}_{\boldsymbol{\phi},d}(\boldsymbol{x})\right) + |\tilde{\mu}_{\boldsymbol{\phi},d}(\boldsymbol{x})|^2 - \tilde{\eta}_{\boldsymbol{\phi},d}(\boldsymbol{x})\right),$$

which is what we give to our optimizer. NO!!! We should give $-\mathcal{L}_{\mathsf{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\phi})$.

Thank you for your attention

- Lecture 2: The variational EM algoritm, mixtures of VAEs, application to audio-visual speech enhancement.
- Lecture 3: Dynamical variational autoencoders, application to speech enhancement.

References

1. D. P. Kingma and M. Welling, "*Auto-encoding variational Bayes*," ICLR, 2014.
2. Y. Bando et al., "*Statistical speech enhancement based on probabilistic integration of variational autoencoder and non-negative matrix factorization*," in Proc. ICASSP, 2018, pp. 716–720
3. S. Leglaive et al., "*A variance modeling framework based on variational autoencoders for speech enhancement*," in Proc. MLSP, 2018.
4. C. Bishop, Pattern Recognition and Machine Learning, Springer-Verlag Berlin, Heidelberg, 2006.