IFT 7030,
Machine Learning for Signal Processing
Week8: Machine Learning 4, Clustering and Generative Models

Cem Subakan





UNIVERSITÉ LAVAL



Admin

- We will release Homework 2 soon.
  - On va publier le homework 2 bientot!
- I hope you have started working on your projects!.
  - J'espère que vous avez déjà commencé vos projets.
- Today: Clustering, then Deep Generative Models
  - Aujourd'hui: Clustering, puis modèles génératifs profonds

Clustering

- Let's kick things off with clustering. / On va commencer avec clustering.
- We were doing supervised learning for the past two classes. Now we will change.
  - On faisait de l'apprentissage supervisé pour les deux derniers classes.

Clustering

- Let's kick things off with clustering. / On va commencer avec clustering.
- We were doing supervised learning for the past two classes. Now we will change.
  - On faisait de l'apprentissage supervisé pour les deux derniers classes.
- What if we do not have the labels?
  - Et si on n'avait pas d'étiquettes?

Clustering

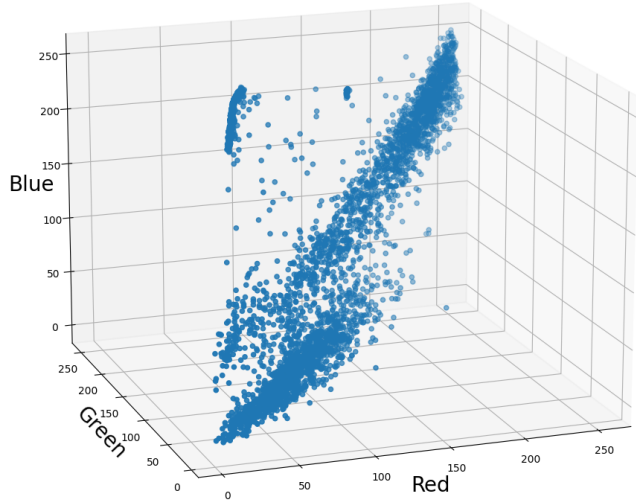
- Let's kick things off with clustering. / On va commencer avec clustering.
- We were doing supervised learning for the past two classes. Now we will change.
  - On faisait de l'apprentissage supervisé pour les deux derniers classes.
- What if we do not have the labels?
  - Et si on n'avait pas d'étiquettes?
- Today's lecture's goal / Le but d'aujourd'hui.
  - I will try to acclimate you with clustering./Je vais essayer de vous introduire des concepts de base de clustering.
  - But in a sneaky way I will introduce powerful tools from probabilistic machine learning. / Je vais vous bombarder silenciusement avec des outils des l'apprentissage automatique probabilistique.

Some motivation

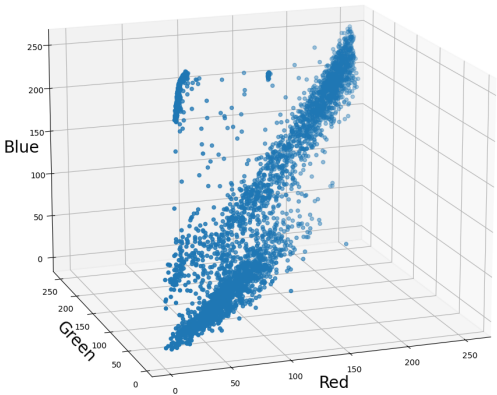


El Capitan, Yosemite National Park, California

How many clusters?

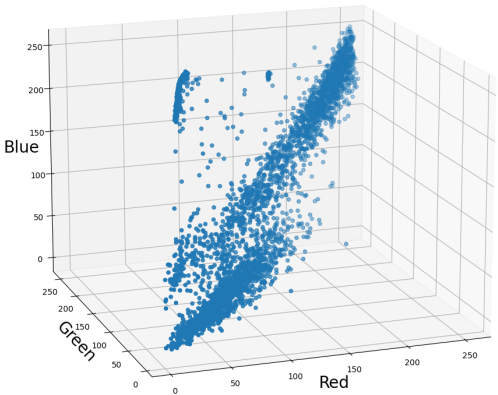


Clustering



- We see clusters, but how do we find them?
  - On voit des groupes, mais comment on les trouve?
- Can I find an algorithm for this? / Peut-on trouver un algorithme pour cela?

Clustering



- We see clusters, but how do we find them?
  - On voit des groupes, mais comment on les trouve?
- Can I find an algorithm for this? / Peut-on trouver un algorithme pour cela?
- Clustering!

Clustering

- **Clustering:** We discover clusters/groups in the data.
  - **Clustering:** On découvre des groupes dans les données.
- Fundamentally **ill defined** problem. There is often no correct solution.
  - Clustering n'est pas un problème bien défini.
- Relies on user choices.
  - Ça dépend sur les choix de l'utilisateur.

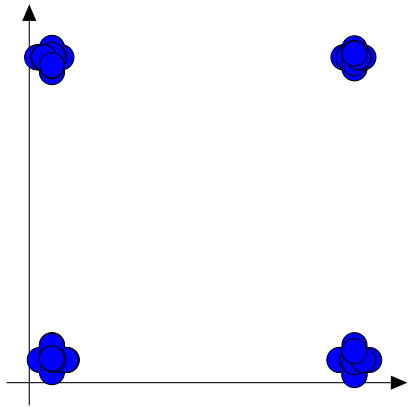
Clustering process

- **Features**
  Describe your data using features. / Quels features utilise-t-on pour répresenter les données?
- **Cluster Shapes**
  Decide what your clusters should look like / Il faut décider comment on veut que les clusters se forment.

Clustering process

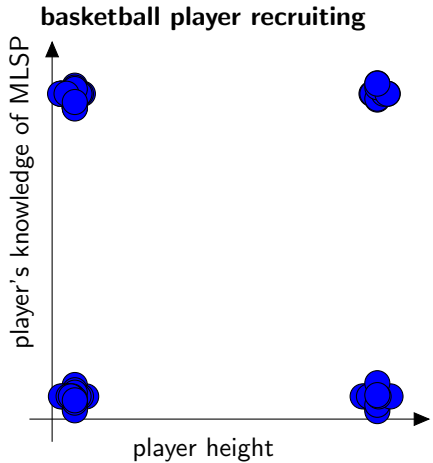
- **Features**
  Describe your data using features. / Quels features utilise-t-on pour répresenter les données?
- **Cluster Shapes**
  Decide what your clusters should look like / Il faut décider comment on veut que les clusters se forment.
- **Distance**
  Define a distance function / promixity measure.
  - Définissons une notion de distance / proximité.

Features



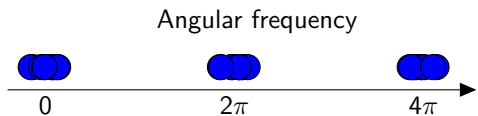
- How many clusters? / Combien de groupes?

Features

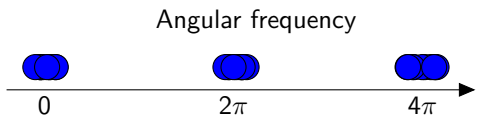


- How many clusters? / Combien de groupes?
- How many clusters? / Combien de groupes?

Use a sensible distance function

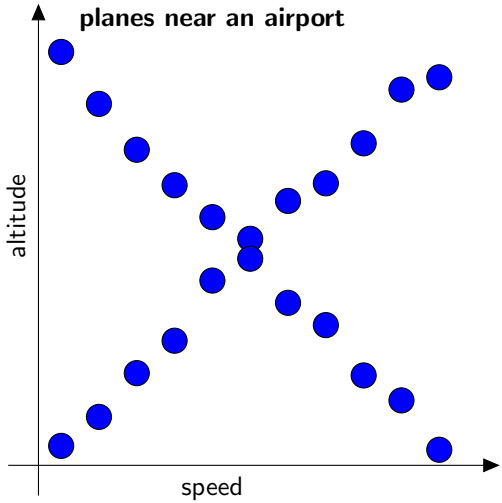


Use a sensible distance function

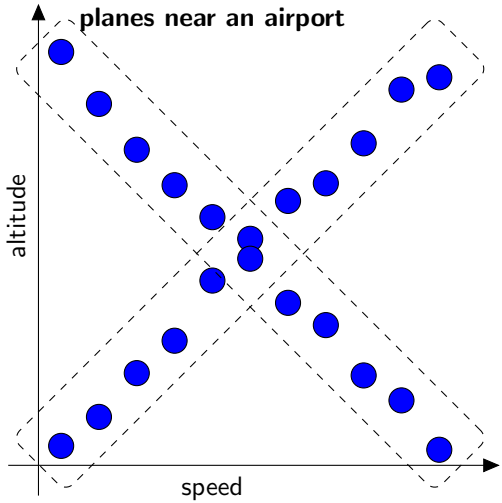


- Probably using euclidean distance is not a good idea here!
  - Très probable qu'il faut pas utiliser la distance euclidienne içi!!

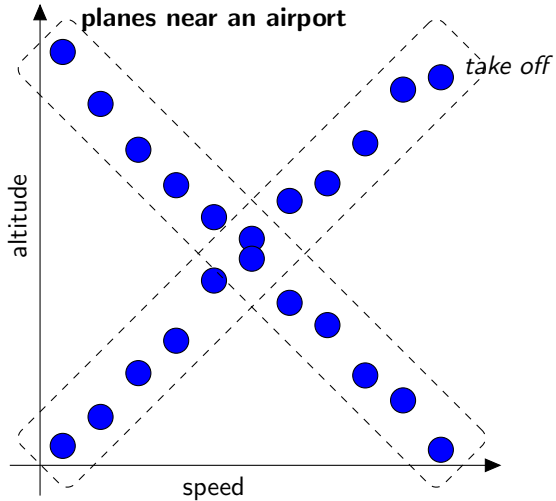
What forms clusters in your space?



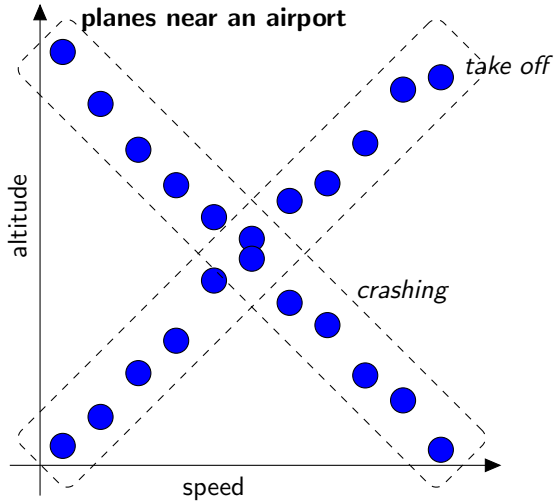
What forms clusters in your space?



What forms clusters in your space?

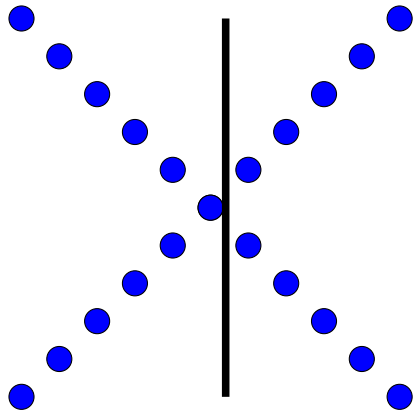


What forms clusters in your space?



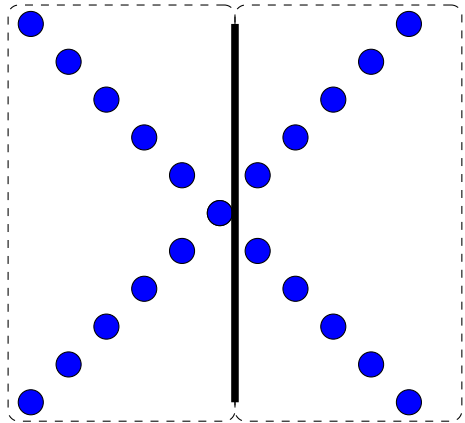
What forms clusters in your space?

ball trajectories bouncing off a wall

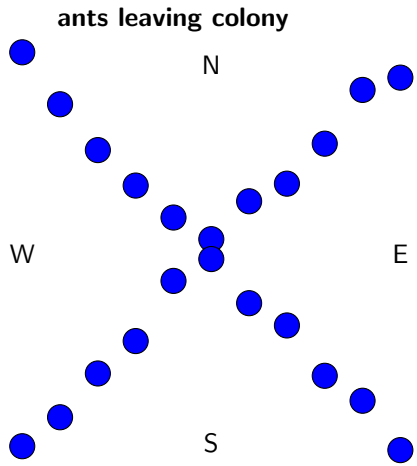


What forms clusters in your space?

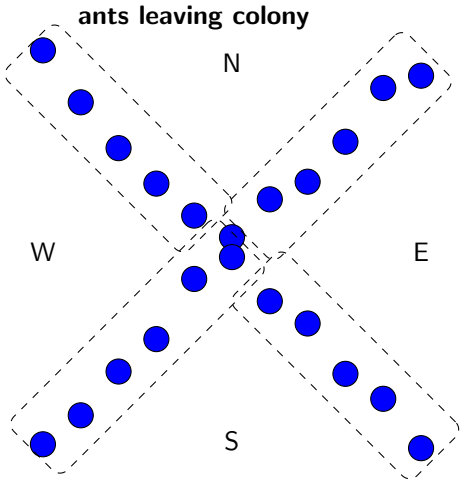
ball trajectories bouncing off a wall



What forms clusters in your space?



What forms clusters in your space?



There usually are not %100 correct answers

- Clustering is partly an art
  - Clustering de partie est un art
- You need to experiment with different things to get there
  - On a besoin d'experimenter afin d'y arriver

How to cluster

- Many approaches. Today we'll talk about
  - Centroid based approaches
    - K-means
    - Mixture Models
  - Hierarchical clustering
  - Spectral clustering

Table of Contents

Centroid based approaches
- K-means clustering
- Gaussian Mixture Model
- More Advanced GMM Learning Methods

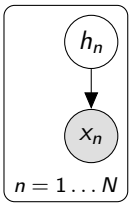
Spectral Clustering

Hierarchical Clustering

Deep Generative Models
- Variational Autoencoders
- Generative Adversarial Networks
- Diffusion Models
- Audio Diffusion Models

Gaussian Mixture Model

- Model:



$$h_n \sim \text{Discrete}(\pi)$$
$$x_n | h_n \sim \mathcal{N}(x; \mu_{h_n}, \sigma^2 I), \text{ for } n \in \{1, \dots N\}$$

- $h_n \in \{1, \dots, K\}$, cluster indicators / indicateur de groupes.
- $x_n \in \mathbb{R}^L$, observed data items / des données observées.
- $\theta = \{\mu_1, \mu_2, \dots, \mu_K\}$ parameters/cluster centers (or centroids) (les paramètres, centres de groupes).

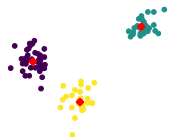


Table of Contents

Centroid based approaches
- K-means clustering
- Gaussian Mixture Model
- More Advanced GMM Learning Methods

Spectral Clustering

Hierarchical Clustering

Deep Generative Models
- Variational Autoencoders
- Generative Adversarial Networks
- Diffusion Models
- Audio Diffusion Models

Learning Variant 1 for GMM

- Find cluster indicators $\widehat{h}_{1:N}$ and parameters $\widehat{\theta}$ such that: / On trouve des indicateurs de clusters et des centroids telle que:

$$\widehat{h}_{1:N}, \widehat{\theta} = \arg\max_{h_{1:N}, \theta} p(x_{1:N} | h_{1:N}, \theta)$$

Learning Variant 1 for GMM

- Find cluster indicators $\widehat{h}_{1:N}$ and parameters $\widehat{\theta}$ such that: / On trouve des indicateurs de clusters et des centroids telle que:

$$\widehat{h}_{1:N}, \widehat{\theta} = \arg\max_{h_{1:N},\theta} p(x_{1:N}|h_{1:N}, \theta)$$

- Write down log-likelihood: / On écrit le log-likelihood:

$$\begin{aligned}
\log p(x_{1:N}, h_{1:N}|\theta) &= \log \prod_{n=1}^{N} p(x_n|h_n, \theta) p(h_n|\theta) \\
&= \log \prod_{n=1}^{N} \left(\prod_{k=1}^{K} \mathcal{N}(x_n; \mu_k, \sigma^2 I)^{[h_n=k]} \times \prod_{k=1}^{K} \pi_k^{[h_n=k]} \right) \\
&=^{+} \sum_{n=1}^{N} \left(\sum_{k=1}^{K} [h_n = k] \left(\frac{-\|x_n - \mu_k\|_2^2}{2\sigma^2} + \log \pi_k \right) \right)
\end{aligned}$$

How to learn with this objective function?

$$\mathcal{L}(\mu_{1:K}, \pi_{1:K}, h_{1:N}) = \sum_{n=1}^{N} \left(\sum_{k=1}^{K} [h_n = k] \left(\frac{\|x_n - \mu_k\|_2^2}{2\sigma^2} + \log \pi_k \right) \right)$$

- Notice that $h_{1:N}$ are discrete variables. / Notez que $h_{1:N}$ sont discrets.
- We can not directly take the gradient and optimize. / On ne peut pas juste calculer le gradient et l'optimiser.

How to learn with this objective function?

$$\mathcal{L}(\mu_{1:K}, \pi_{1:K}, h_{1:N}) = \sum_{n=1}^{N} \left(\sum_{k=1}^{K} [h_n = k] \left(\frac{\|x_n - \mu_k\|_2^2}{2\sigma^2} + \log \pi_k \right) \right)$$

- Notice that $h_{1:N}$ are discrete variables. / Notez que $h_{1:N}$ sont discrets.
- We can not directly take the gradient and optimize. / On ne peut pas juste calculer le gradient et l'optimiser.
- Any ideas? / Idées?

Learning Variant 1 for GMM

- Algorithm: Fix θ, update h. Fix h, update θ, repeat until convergence (and fix $\pi_k = 1/K$). / On alterne entre l'optimization des parametres θ et les indicateurs h.

Learning Variant 1 for GMM

- Algorithm: Fix θ, update h. Fix h, update θ, repeat until convergence (and fix $\pi_k = 1/K$). / On alterne entre l'optimization des parametres θ et les indicateurs h.
- Update $\mu_{k'}$: compute the gradient while $h_{1:N}$ is fixed: / calcule le gradient par rapport à $\mu_{k'}$ quand $h_{1:N}$ est fixé.

$$\frac{\partial \log p(x_{1:N}, h_{1:N}|\theta)}{\partial \mu_k} = \frac{\partial \sum_{n=1}^{N} \left(\sum_{k=1}^{K} [h_n = k] \left(\frac{-\|x_n - \mu_k\|_2^2}{2\sigma^2} + \log \pi_k \right) \right)}{\partial \mu_{k'}}$$

$$= \sum_{n=1}^{N} [h_n = k'] \frac{(x_n - \mu_{k'})}{\sigma^2} = \sum_{n=1}^{N} [h_n = k'] \frac{x_n}{\sigma^2} - [h_n = k'] \frac{\mu_{k'}}{\sigma^2}$$

set the gradient equal to 0 / mettez le gradient à 0, solve for / résoudre pour $\mu_{k'} \rightarrow \widehat{\mu}_{k'} = \frac{\sum_{n=1}^{N} [h_n = k'] x_n}{\sum_{n=1}^{N} [h_n = k']}$.

Learning Variant 1 for GMM

- Algorithm: Fix θ, update h. Fix h, update θ, repeat until convergence (and fix $\pi_k = 1/K$). / On alterne entre l'optimization des parametres θ et les indicateurs h.
- Update $\mu_{k'}$: compute the gradient while $h_{1:N}$ is fixed: / calcule le gradient par rapport à $\mu_{k'}$ quand $h_{1:N}$ est fixé.

$$\frac{\partial \log p(x_{1:N}, h_{1:N}|\theta)}{\partial \mu_k} = \frac{\partial \sum_{n=1}^{N} \left(\sum_{k=1}^{K} [h_n = k] \left(\frac{-\|x_n - \mu_k\|_2^2}{2\sigma^2} + \log \pi_k \right) \right)}{\partial \mu_{k'}}$$

$$= \sum_{n=1}^{N} [h_n = k'] \frac{(x_n - \mu_{k'})}{\sigma^2} = \sum_{n=1}^{N} [h_n = k'] \frac{x_n}{\sigma^2} - [h_n = k'] \frac{\mu_{k'}}{\sigma^2}$$

set the gradient equal to 0 / mettez le gradient à 0, solve for / résoudre pour $\mu_{k'} \rightarrow \widehat{\mu}_{k'} = \frac{\sum_{n=1}^{N} [h_n = k'] x_n}{\sum_{n=1}^{N} [h_n = k']}$.

- Update $h_{1:N}$ while $\mu_{k'}$ is fixed: / Mettez $h_{1:N}$ à jour quand $\mu_{k'}$ est fixé:

$$\widehat{h}_n = \arg\max_{h_n} \log p(x_n, h_n|\theta) = \arg\min_{k} \|x_n - \mu_k\|_2^2,$$

we therefore assign h_n as the index of the mean closest to x_n. / On assigne h_n au centroid plus proche.

Learning Variant 1 for GMM

- Algorithm: Fix θ, update h. Fix h, update θ, repeat until convergence (and fix $\pi_k = 1/K$). / On alterne entre l'optimization des parametres θ et les indicateurs h.
- Update $\mu_{k'}$: compute the gradient while $h_{1:N}$ is fixed: / calcule le gradient par rapport à $\mu_{k'}$ quand $h_{1:N}$ est fixé.

$$\frac{\partial \log p(x_{1:N}, h_{1:N}|\theta)}{\partial \mu_k} = \frac{\partial \sum_{n=1}^{N} \left(\sum_{k=1}^{K} [h_n = k] \left(\frac{-\|x_n - \mu_k\|_2^2}{2\sigma^2} + \log \pi_k \right) \right)}{\partial \mu_{k'}}$$

$$= \sum_{n=1}^{N} [h_n = k'] \frac{(x_n - \mu_{k'})}{\sigma^2} = \sum_{n=1}^{N} [h_n = k'] \frac{x_n}{\sigma^2} - [h_n = k'] \frac{\mu_{k'}}{\sigma^2}$$

  set the gradient equal to 0 / mettez le gradient à 0, solve for / résoudre pour $\mu_{k'} \rightarrow \widehat{\mu}_{k'} = \frac{\sum_{n=1}^{N} [h_n = k'] x_n}{\sum_{n=1}^{N} [h_n = k']}$.
- Update $h_{1:N}$ while $\mu_{k'}$ is fixed: / Mettez $h_{1:N}$ à jour quand $\mu_{k'}$ est fixé:

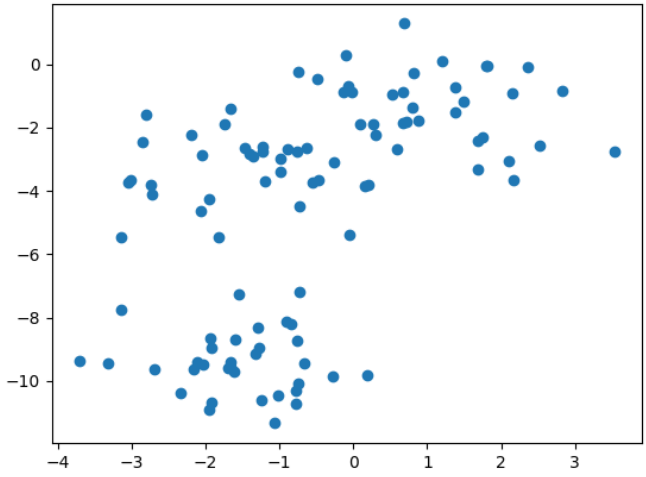
$$\widehat{h}_n = \arg\max_{h_n} \log p(x_n, h_n|\theta) = \arg\min_{k} \|x_n - \mu_k\|_2^2,$$

  we therefore assign h_n as the index of the mean closest to x_n. / On assigne h_n au centroid plus proche.
- Looks like a familiar algorithm? / Vous connaissez ça?

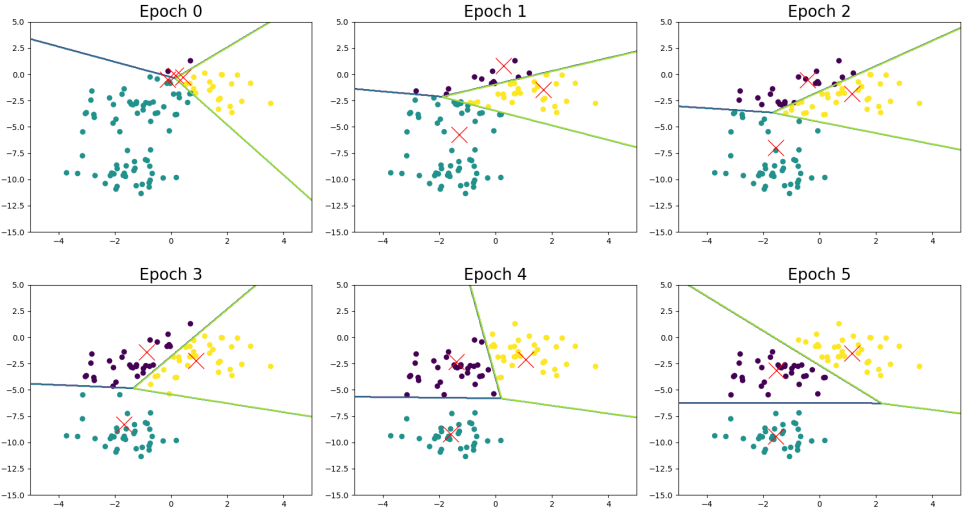
Kmeans Clustering

Randomly initialize $\mu_{1:K}$.
while Not converged **do**
  E-step:
    $\widehat{h}_n = \arg\max_{h_n} \log p(x_n, h_n|\theta) = \arg\min_k \|x_n - \mu_k\|_2^2$
  M-step:
    $\widehat{\mu}_{k'} = \frac{\sum_{n=1}^N [h_n=k']x_n}{\sum_{n=1}^N [h_n=k']}$
end while

Kmeans Example

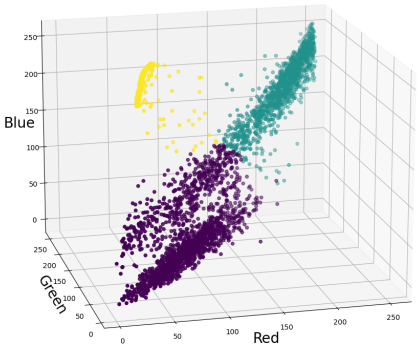


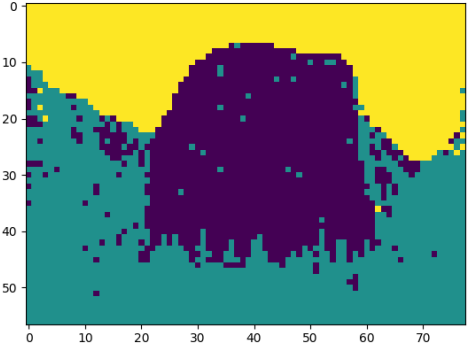
Kmeans Updates



Applying K-means on El-Capitan ($K = 3$)

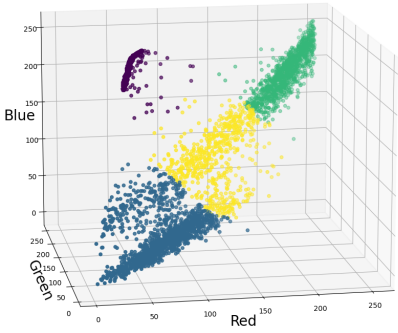


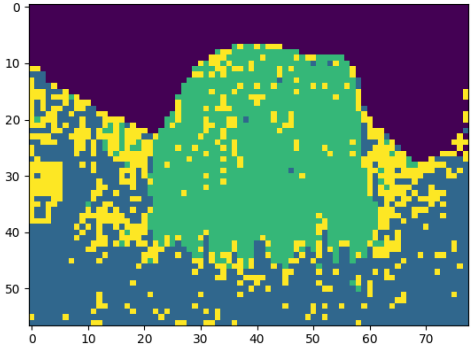




Applying K-means on El-Capitan ($K = 4$)

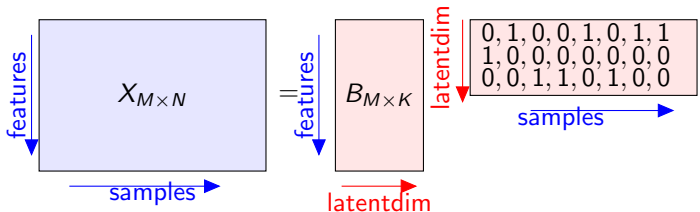






A sidenote

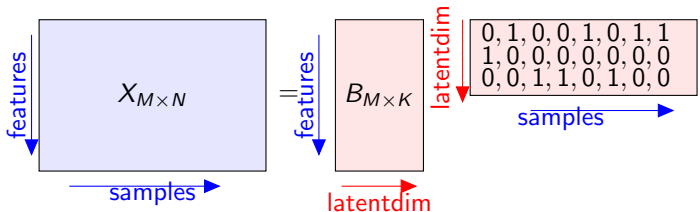
- K-means is a matrix-factorization algorithm



- B in this case has the cluster means in columns / B dans ce cas-ci a les cluster means dans ses colonnes.

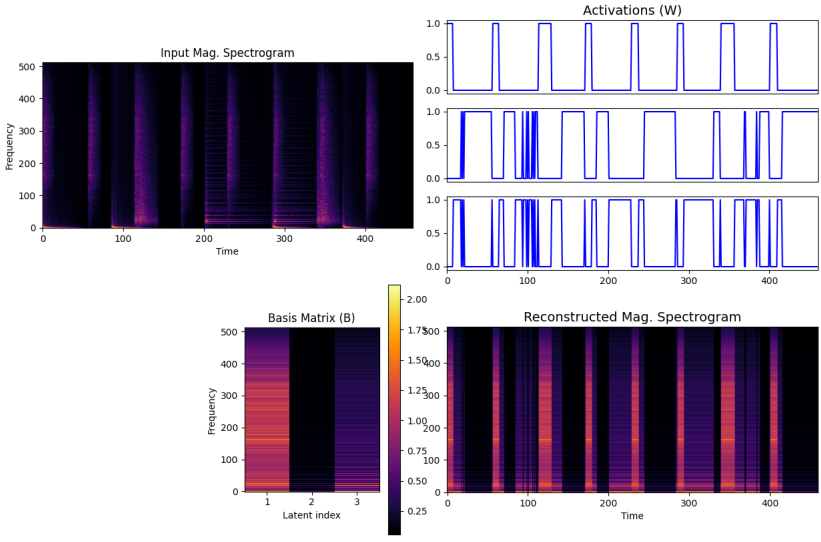
A sidenote

- K-means is a matrix-factorization algorithm

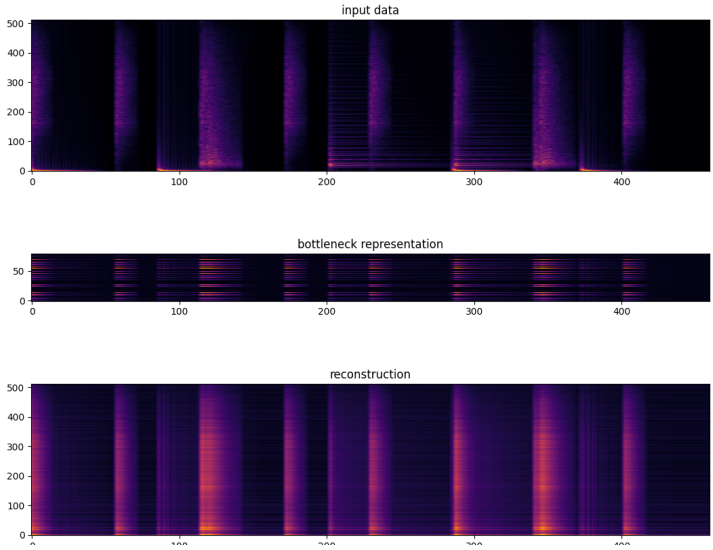


- B in this case has the cluster means in columns / B dans ce cas-ci a les cluster means dans ses colonnes.
- Kmeans is a tokenizer! (It's a buzzword these days)

Kmeans on a familiar picture



Kmeans in the latent space



Cluster the latents!

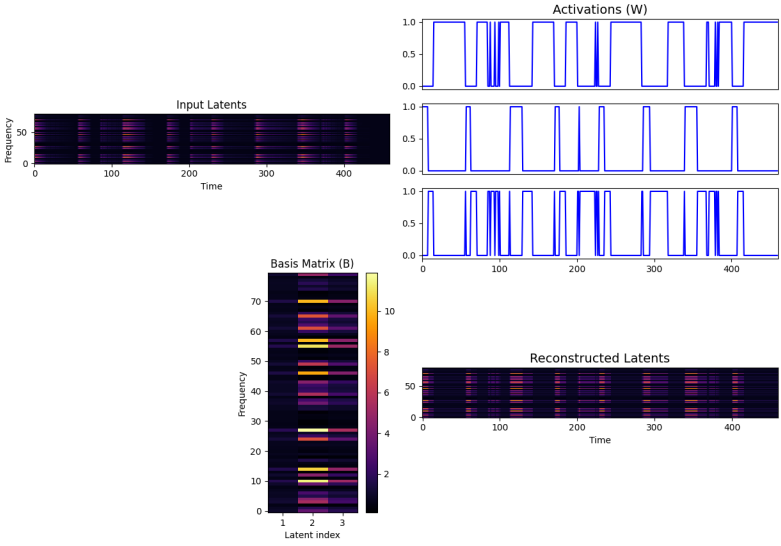


Table of Contents

Centroid based approaches
- K-means clustering
- Gaussian Mixture Model
- More Advanced GMM Learning Methods

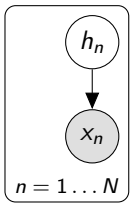
Spectral Clustering

Hierarchical Clustering

Deep Generative Models
- Variational Autoencoders
- Generative Adversarial Networks
- Diffusion Models
- Audio Diffusion Models

Extending K-means

- Kmeans is good and all, but only learns Isotropic Gaussians. / Kmeans ne peut apprendre que des Gaussiennes Isotropiques.

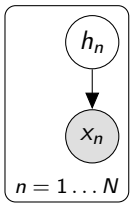


$$h_n \sim \text{Discrete}(\pi)$$
$$x_n | h_n \sim \mathcal{N}(x; \mu_{h_n}, \Sigma_{h_n}), \text{ for } n \in \{1, \dots N\}$$

- $h_n \in \{1, \dots, K\}$, cluster indicators / indicateur de groupes.
- $x_n \in \mathbb{R}^L$, observed data items / des données observées.
- $\theta = \{\mu_1, \mu_2, \dots, \mu_K, \Sigma_1, \Sigma_2, \dots, \Sigma_K, \pi_1, \pi_2, \dots, \pi_K\}$ parameters (les paramètres).

Extending K-means

- Kmeans is good and all, but only learns Isotropic Gaussians. / Kmeans ne peut apprendre que des Gaussiennes Isotropiques.



$$h_n \sim \text{Discrete}(\pi)$$
$$x_n | h_n \sim \mathcal{N}(x; \mu_{h_n}, \Sigma_{h_n}), \text{ for } n \in \{1, \dots N\}$$

- $h_n \in \{1, \dots, K\}$, cluster indicators / indicateur de groupes.
- $x_n \in \mathbb{R}^L$, observed data items / des données observées.
- $\theta = \{\mu_1, \mu_2, \dots, \mu_K, \Sigma_1, \Sigma_2, \dots, \Sigma_K, \pi_1, \pi_2, \dots, \pi_K\}$ parameters (les paramètres).
- Full Covariance matrix for each cluster, and cluster prior probabilities are estimated also.
  - Une matrice de covariance est appris pour chaque cluster, puis les probabilités a prioris de chaque cluster.

Learning Variant 2 for GMM

- In addition, we want to estimate probabilities for $h_{1:N}$. / On estimer des probabilités pour des indicateurs des clusters.
- Find cluster indicator parameters $\widehat{\theta}$ while integrating out hidden variables, such that: / On marginalise sur $h_{1:N}$

$$\widehat{\theta} = \arg\max_{\theta} p(x_{1:N}|\theta) = \arg\max_{\theta} \sum_{h_{1:N}} p(x_{1:N}, h_{1:N}|\theta)$$

Learning Variant 2 for GMM

- In addition, we want to estimate probabilities for $h_{1:N}$. / On estimer des probabilités pour des indicateurs des clusters.
- Find cluster indicator parameters $\widehat{\theta}$ while integrating out hidden variables, such that: / On marginalise sur $h_{1:N}$

$$\widehat{\theta} = \arg\max_{\theta} p(x_{1:N}|\theta) = \arg\max_{\theta} \sum_{h_{1:N}} p(x_{1:N}, h_{1:N}|\theta)$$

- Write down/Écrit log-likelihood:

$$\log p(x_{1:N}|\theta) = \log \sum_{h_{1:N}} \frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} q(h_{1:N}) = \log \mathbb{E}_q \left[\frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} \right]$$

Learning Variant 2 for GMM

- In addition, we want to estimate probabilities for $h_{1:N}$. / On estimer des probabilités pour des indicateurs des clusters.
- Find cluster indicator parameters $\widehat{\theta}$ while integrating out hidden variables, such that: / On marginalise sur $h_{1:N}$

$$\widehat{\theta} = \arg\max_{\theta} p(x_{1:N}|\theta) = \arg\max_{\theta} \sum_{h_{1:N}} p(x_{1:N}, h_{1:N}|\theta)$$

- Write down/Écrit log-likelihood:

$$\log p(x_{1:N}|\theta) = \log \sum_{h_{1:N}} \frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} q(h_{1:N}) = \log \mathbb{E}_q \left[\frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} \right]$$

$$\geq VLB := \mathbb{E}_q \left[\log \frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} \right] =^{+} \mathbb{E}_q \left[\log p(x_{1:N}, h_{1:N}|\theta) \right]$$

$$=^{+} \sum_{n=1}^{N} \left(\sum_{k=1}^{K} \mathbb{E}_q[h_n = k] \left(\frac{-\|x_n - \mu_k\|_2^2}{2\sigma^2} + \log \pi_k \right) \right)$$

Learning Variant 2 for GMM

- Algorithm: Fix θ, update q. Fix q, update θ, repeat until convergence. / On alterne entre l'optimization de q et θ.

Learning Variant 2 for GMM

- Algorithm: Fix θ, update q. Fix q, update θ, repeat until convergence. / On alterne entre l'optimization de q et θ.
- Update $\mu_{k'}$: compute the gradient while $q(h_{1:N})$ is fixed:

$$\frac{\partial VLB}{\partial \mu_{k'}} = \frac{\partial \sum_{n=1}^{N} \left(\sum_{k=1}^{K} \mathbb{E}[h_n = k] \left(\frac{-\|x_n - \mu_k\|_2^2}{2\sigma^2} + \log \pi_k \right) \right)}{\partial \mu_{k'}}$$

$$= \sum_{n=1}^{N} [h_n = k'] \frac{(x_n - \mu_{k'})}{\sigma^2} = \sum_{n=1}^{N} \mathbb{E}[h_n = k'] \frac{x_n}{\sigma^2} - \mathbb{E}[h_n = k'] \frac{\mu_{k'}}{\sigma^2}$$

set the gradient equal to 0, solve for $\mu_{k'} \rightarrow \widehat{\mu}_{k'} = \frac{\sum_{n=1}^{N} \mathbb{E}[h_n = k'] x_n}{\sum_{n=1}^{N} \mathbb{E}[h_n = k']}$.

Learning Variant 2 for GMM

- Algorithm: Fix θ, update q. Fix q, update θ, repeat until convergence. / On alterne entre l'optimization de q et θ.
- Update $\mu_{k'}$: compute the gradient while $q(h_{1:N})$ is fixed:

$$\frac{\partial VLB}{\partial \mu_{k'}} = \frac{\partial \sum_{n=1}^{N} \left(\sum_{k=1}^{K} \mathbb{E}[h_n = k] \left(\frac{-\|x_n - \mu_k\|_2^2}{2\sigma^2} + \log \pi_k \right) \right)}{\partial \mu_{k'}}$$

$$= \sum_{n=1}^{N} [h_n = k'] \frac{(x_n - \mu_{k'})}{\sigma^2} = \sum_{n=1}^{N} \mathbb{E}[h_n = k'] \frac{x_n}{\sigma^2} - \mathbb{E}[h_n = k'] \frac{\mu_{k'}}{\sigma^2}$$

set the gradient equal to 0, solve for $\mu_{k'} \rightarrow \widehat{\mu}_{k'} = \frac{\sum_{n=1}^{N} \mathbb{E}[h_n = k'] x_n}{\sum_{n=1}^{N} \mathbb{E}[h_n = k']}$.

- Update $\widehat{\sigma}_{k'} = \frac{\sum_{n=1}^{N} \mathbb{E}[h_n = k'](x_n - \mu_k)^2}{\sum_{n=1}^{N} \mathbb{E}[h_n = k']}$.
- Update $\widehat{\pi}_{k'} = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}[h_n = k']$.

Learning Variant 2 for GMM - optimal $q(h)$

- Update $q(h_{1:N})$ while θ is fixed. Notice that:

$$VLB = \mathbb{E}_q \left[\log \frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} \right] = KL(q(h) \| p(x, h|\theta)).$$

  What is the variational distribution that would minimize this divergence?

Learning Variant 2 for GMM - optimal $q(h)$

- Update $q(h_{1:N})$ while θ is fixed. Notice that:

$$VLB = \mathbb{E}_q \left[\log \frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} \right] = KL(q(h) \| p(x, h|\theta)).$$

  What is the variational distribution that would minimize this divergence?

- The derivation.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial q} =& \frac{\partial}{\partial q} \left(\int q(h) \log p(x, h|\theta) dh - \int q(h) \log q(h) dh + \lambda \left(\int q(h) dh - 1 \right) \right) \\
=& \log p(x, h) - \log q(h) - 1 + \lambda = 0 \\
\rightarrow & q(h) = \frac{p(x, h|\theta)}{\exp(1 - \lambda)} \\
\rightarrow & \exp(1 - \lambda) = p(x|\theta) \quad \rightarrow \quad q(h) = \frac{p(x, h|\theta)}{p(x|\theta)} = p(h|x, \theta)
\end{aligned}$$

Learning Variant 2 for GMM - optimal $q(h)$

- Update $q(h_{1:N})$ while θ is fixed. Notice that:

$$VLB = \mathbb{E}_q \left[\log \frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} \right] = KL(q(h) \| p(x, h|\theta)).$$

  What is the variational distribution that would minimize this divergence?

- The derivation.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial q} =& \frac{\partial}{\partial q} \left(\int q(h) \log p(x, h|\theta) dh - \int q(h) \log q(h) dh + \lambda \left(\int q(h) dh - 1 \right) \right) \\
&= \log p(x, h) - \log q(h) - 1 + \lambda = 0 \\
&\rightarrow q(h) = \frac{p(x, h|\theta)}{\exp(1 - \lambda)} \\
&\rightarrow \exp(1 - \lambda) = p(x|\theta) \quad \rightarrow \quad q(h) = \frac{p(x, h|\theta)}{p(x|\theta)} = p(h|x, \theta)
\end{aligned}$$

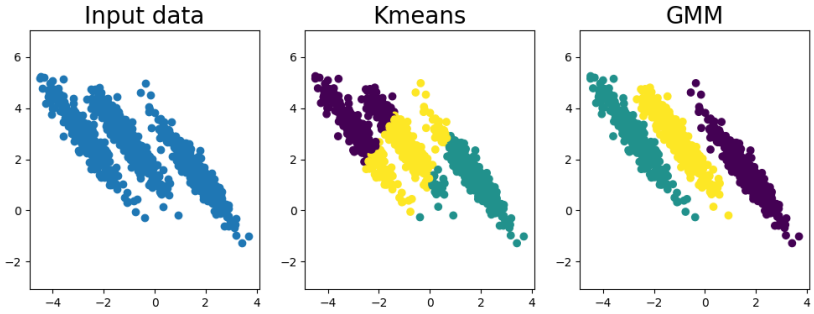
- Note that in our case $q(h) = q(h_{1:N}) = \prod_n q(h_n)$, where

$$q(h_n = k) = \frac{p(x_n, h_n = k|\theta)}{p(x_n|\theta)} = \frac{\pi_k \mathcal{N}(x_n; \mu_k, \sigma^2 I)}{\sum_{k'} \pi_{k'} \mathcal{N}(x_n; \mu_{k'}, \sigma^2 I)}$$

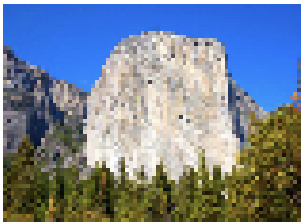
Learning Variant 2 for GMM - Summary for ICM and EM

Randomly initialize $\mu_{1:K}$.
while Not converged **do**
  E-step:
    if ICM **then**
      $\widehat{h}_n = \arg\max_{h_n} \log p(x_n, h_n|\theta) = \arg\min_k \|x_n - \mu_k\|_2^2$
    else if EM **then**
      $q(h_n = k) = \frac{\pi_k \mathcal{N}(x_n; \mu_k, \sigma^2 I)}{\sum_{k'} \pi_{k'} \mathcal{N}(x_n; \mu_{k'}, \sigma^2 I)}$
    end if
  M-step:
    if ICM **then**
      $\widehat{\mu}_{k'} = \frac{\sum_{n=1}^N [h_n = k'] x_n}{\sum_{n=1}^N [h_n = k']}$
    else if EM **then**
      $\widehat{\mu}_{k'} = \frac{\sum_{n=1}^N \mathbb{E}_q[h_n = k'] x_n}{\sum_{n=1}^N \mathbb{E}_q[h_n = k']}$
    end if
end while

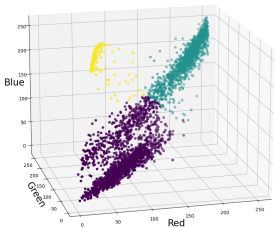
Kmeans vs GMM

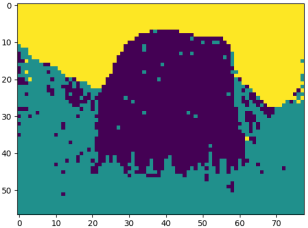


GMM on El Capitan

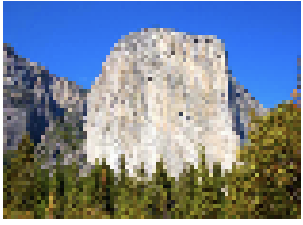


With Kmeans

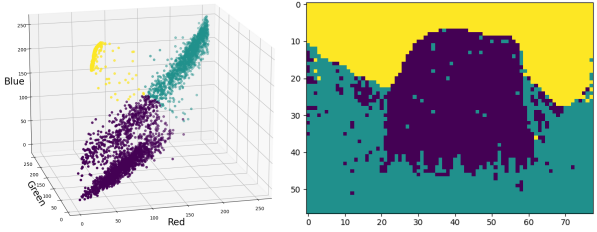




GMM on El Capitan



With Kmeans



With GMM

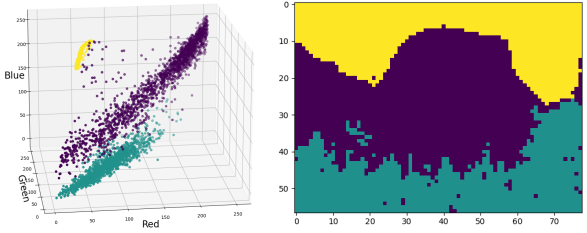


Table of Contents

Centroid based approaches
- K-means clustering
- Gaussian Mixture Model
- More Advanced GMM Learning Methods

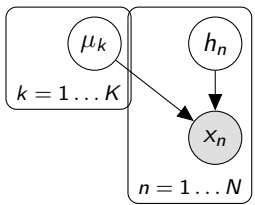
Spectral Clustering

Hierarchical Clustering

Deep Generative Models
- Variational Autoencoders
- Generative Adversarial Networks
- Diffusion Models
- Audio Diffusion Models

Learning Variant 3 for GMM - Going Full Bayesian

- Model:



$$\mu_k \sim \mathcal{N}(\mu_k; 0, \sigma_0^2 I), \text{ for } k \in \{1, \dots, K\}$$
$$h_n \sim \text{Categorical}(\pi)$$
$$x_n | h_n \sim \mathcal{N}(x; \mu_h, \sigma^2 I), \text{ for } n \in \{1, \dots, N\}$$

- $h_n \in \{1, \dots, K\}$, cluster indicators / indicateur des clusters.
- $x_n \in \mathbb{R}^L$, observed data items / données observées.
- $\theta = \{\mu_1, \mu_2, \dots, \mu_K\}$ parameters/cluster centers. But we are not treating these guys as parameters anymore. / On ne traite plus ça comme des parametres mais des distributions.

Inference for Variant 3 GMM

- Approximate the posterior distribution $p(h, \theta|x)$, with a variational distribution $\widehat{q}$ such that, / On va approximer la posterior telle que,

$$\widehat{q}(h, \theta) = \arg\min_{q} KL(q(h, \theta) \| p(x, h, \theta))$$

- We will use the mean field approximation. English: $q(h, \theta) = q_h(h) q_\theta(\theta)$.

Inference for Variant 3 GMM

- Approximate the posterior distribution $p(h, \theta|x)$, with a variational distribution $\widehat{q}$ such that, / On va approximer la posterior telle que,

$$\widehat{q}(h, \theta) = \arg\min_{q} KL(q(h, \theta) \| p(x, h, \theta))$$

- We will use the mean field approximation. English: $q(h, \theta) = q_h(h) q_\theta(\theta)$.
- Algorithm: Fix q_h, update q_θ. We can show that: (via same process as the EM case)

$$\widehat{q}_\theta(\theta) = \arg\min_{q_\theta} KL(q_h(h) q_\theta(\theta) \| p(x, h, \theta)) = \frac{1}{Z} \exp\left(\mathbb{E}_{q_h}[\log p(x, h, \theta)]\right)$$

where Z is the normalization constant (constant de normalization). Similarly,

Inference for Variant 3 GMM

- Approximate the posterior distribution $p(h, \theta|x)$, with a variational distribution $\widehat{q}$ such that, / On va approximer la posterior telle que,

$$\widehat{q}(h, \theta) = \arg\min_{q} KL(q(h, \theta) \| p(x, h, \theta))$$

- We will use the mean field approximation. English: $q(h, \theta) = q_h(h) q_\theta(\theta)$.
- Algorithm: Fix q_h, update q_θ. We can show that: (via same process as the EM case)

$$\widehat{q}_\theta(\theta) = \arg\min_{q_\theta} KL(q_h(h) q_\theta(\theta) \| p(x, h, \theta)) = \frac{1}{Z} \exp\left(\mathbb{E}_{q_h}[\log p(x, h, \theta)]\right)$$

where Z is the normalization constant (constant de normalization). Similarly,

- Fix q_θ, update q_h:

$$\widehat{q}_h(h) = \arg\min_{q_h} KL(q_h(h) q_\theta(\theta) \| p(x, h, \theta)) = \frac{1}{Z} \exp\left(\mathbb{E}_{q_\theta}[\log p(x, h, \theta)]\right)$$

Inference for Variant 3 GMM - Specifics:

$$
\begin{aligned}
\log \widehat{q}_\theta(\mu_k) &=^+ \mathbb{E}_{q_h}[\log p(x, h, \mu_k)] \\
&=^+ \sum_{n=1}^{N} \mathbb{E}[h_n = k] \frac{-(x_n^\top x_n - 2x_n^\top \mu_k + \mu_k^\top \mu_k)}{2\sigma^2} - \frac{\mu_k^\top \mu_k}{2\sigma_0^2} \\
&=^+ \frac{\sum_{n=1}^{N} \mathbb{E}[h_n = k] 2x_n^\top \mu_k - (\sum_{n=1}^{N} \mathbb{E}[h_n = k] + \sigma^2)\mu_k^\top \mu_k)}{2\sigma^2\sigma_0^2} \\
&=^+ \log \mathcal{N}\left(\mu_k; \frac{\sum_n \mathbb{E}[h_n = k] x_n}{\sum_n \mathbb{E}[h_n = k] + \sigma^2}, \frac{\sigma^2\sigma_0^2}{\sum_n \mathbb{E}[h_n = k] + \sigma^2}\right)
\end{aligned}
$$

Inference for Variant 3 GMM - Specifics:

$$
\begin{aligned}
\log \widehat{q}_\theta(\mu_k) &=^+ \mathbb{E}_{q_h}[\log p(x, h, \mu_k)] \\
&=^+ \sum_{n=1}^{N} \mathbb{E}[h_n = k] \frac{-(x_n^\top x_n - 2x_n^\top \mu_k + \mu_k^\top \mu_k)}{2\sigma^2} - \frac{\mu_k^\top \mu_k}{2\sigma_0^2} \\
&=^+ \frac{\sum_{n=1}^{N} \mathbb{E}[h_n = k] 2x_n^\top \mu_k - (\sum_{n=1}^{N} \mathbb{E}[h_n = k] + \sigma^2)\mu_k^\top \mu_k)}{2\sigma^2\sigma_0^2} \\
&=^+ \log \mathcal{N}\left(\mu_k; \frac{\sum_n \mathbb{E}[h_n = k] x_n}{\sum_n \mathbb{E}[h_n = k] + \sigma^2}, \frac{\sigma^2\sigma_0^2}{\sum_n \mathbb{E}[h_n = k] + \sigma^2}\right)
\end{aligned}
$$

Inference for Variant 3 GMM - Specifics:

$$\log \widehat{q}_h(h_n = k) = \left(\frac{\mathbb{E}[-\|x_n - \mu_k\|_2^2]}{2\sigma^2} + \log \pi_k \right)$$

$$\rightarrow \widehat{q}_h(h_n = k) = \frac{\exp\left(\frac{\mathbb{E}[-\|x_n - \mu_k\|_2^2]}{2\sigma^2} + \log \pi_k \right)}{\sum_k \exp\left(\frac{\mathbb{E}[-\|x_n - \mu_k\|_2^2]}{2\sigma^2} + \log \pi_k \right)}$$

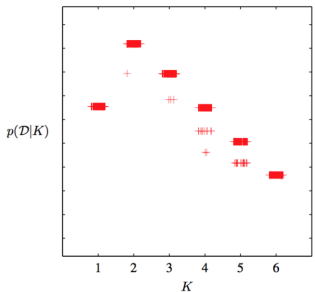
Inference for Variant 3 GMM - Why:

- Variational lower bound:

$$\int p(x, h, \theta) dh d\theta \geq \mathbb{E}_{q(h)q(\theta)}[\log p(x, h, \theta)] - \mathbb{E}_{q(h)q(\theta)}[\log q(h) + \log q(\theta)]$$

- You can use VLB to determine K: (plot taken from Bishop, 2006)

Plot of the variational lower bound $\mathcal{L}$ versus the number K of components in the Gaussian mixture model, for the Old Faithful data, showing a distinct peak at $K = 2$ components. For each value of K, the model is trained from 100 different random starts, and the results shown as '+' symbols plotted with small random horizontal perturbations so that they can be distinguished. Note that some solutions find suboptimal local maxima, but that this happens infrequently.



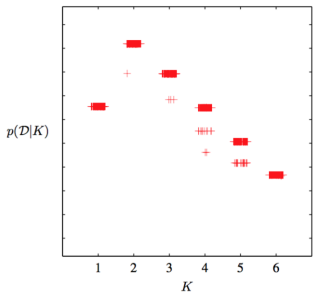
Inference for Variant 3 GMM - Why:

- Variational lower bound:

$$\int p(x, h, \theta) dh d\theta \geq \mathbb{E}_{q(h)q(\theta)}[\log p(x, h, \theta)] - \mathbb{E}_{q(h)q(\theta)}[\log q(h) + \log q(\theta)]$$

- You can use VLB to determine K: (plot taken from Bishop, 2006)

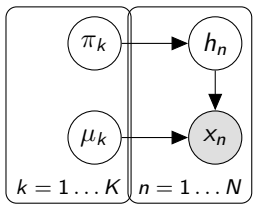
Plot of the variational lower bound $\mathcal{L}$ versus the number K of components in the Gaussian mixture model, for the Old Faithful data, showing a distinct peak at $K = 2$ components. For each value of K, the model is trained from 100 different random starts, and the results shown as '+' symbols plotted with small random horizontal perturbations so that they can be distinguished. Note that some solutions find suboptimal local maxima, but that this happens infrequently.



- But admittedly the algebra gets tiring./C'est évident que les dérivations sont fatiguants.

Variant 4 for GMM - Going Ultra Bayesian

- Model:



$$
\begin{aligned}
\pi &\sim \text{Dirichlet}(1/K, \dots, 1/K) \\
\mu_k &\sim \mathcal{N}(\mu_k; 0, \sigma_0^2 I), \text{ for } k \in \{1, \dots, K\} \\
h_n &\sim \text{Categorical}(\pi) \\
x_n | h_n &\sim \mathcal{N}(x; \mu_h, \sigma^2 I), \text{ for } n \in \{1, \dots, N\}
\end{aligned}
$$

- $h_n \in \{1, \dots, K\}$, cluster indicators / indicateurs de clusters.
- $x_n \in \mathbb{R}^L$, observed data items / données.
- $\theta = \{\mu_1, \mu_2, \dots, \mu_K\} \cup \{\pi\}$

Variant 4 for GMM - Infinite Mixture Model

- Integrate out the parameters, sample from the full conditionals / On va éliminer les parameters, et échantilloner en utilisant les full conditionals:

$$p(h_n = k|h_{-n}, x_{1:N}) \propto \int p(x_{1:N}, h_{1:N}, \pi, \mu_{1:K}) d\mu_{1:K} d\pi$$
$$\propto \frac{\alpha/K + N_k^{-n}}{\alpha + N - 1} p(x_n|\{x_m : m \neq n, h_m = k\})$$

- And, sample from these full conditionals!

Variant 4 for GMM - Infinite Mixture Model

- Integrate out the parameters, sample from the full conditionals / On va éliminer les parameters, et échantilloner en utilisant les full conditionals:

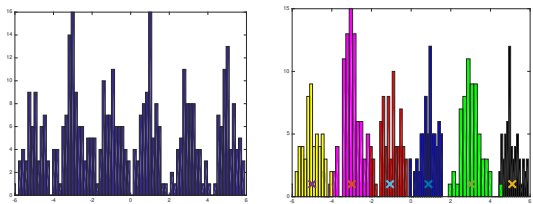
$$p(h_n = k|h_{-n}, x_{1:N}) \propto \int p(x_{1:N}, h_{1:N}, \pi, \mu_{1:K}) d\mu_{1:K} d\pi$$
$$\propto \frac{\alpha/K + N_k^{-n}}{\alpha + N - 1} p(x_n|\{x_m : m \neq n, h_m = k\})$$

- Take K to infinity:

$$p(h_n = k, k \text{ occupied}|h_{-n}, x_{1:N}) \propto \frac{N_k^{-n}}{\alpha + N - 1} p(x_n|\{x_m : m \neq n, h_m = k\})$$
$$p(h_n = k, k \text{ empty}|h_{-n}, x_{1:N}) \propto \frac{\alpha}{\alpha + N - 1} p(x_n)$$

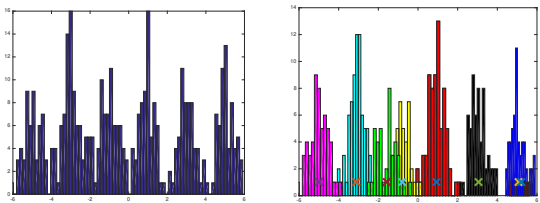
- And, sample from these full conditionals!

Collapsed Gibbs sampling in Infinite GMM



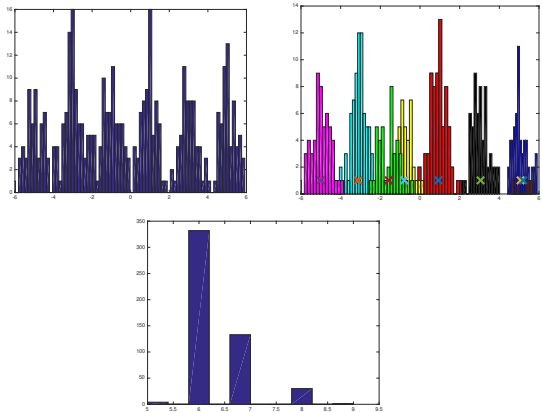
Top left: Histogram of observed data (données observés), Top right: Samples from full conditional of $h_{1:N}$, Bottom: Histogram of K (nombre de clusters)

Collapsed Gibbs sampling in Infinite GMM



Top left: Histogram of observed data (données observés), Top right: Samples from full conditional of $h_{1:N}$, Bottom: Histogram of K (nombre de clusters)

Collapsed Gibbs sampling in Infinite GMM



Top left: Histogram of observed data (données observés), Top right: Samples from full conditional of $h_{1:N}$, Bottom: Histogram of K (nombre de clusters)

What's the point of going all Bayesian then

- (Automatic) Model Selection for Unsupervised Learning / Sélection automatique du complexité du modèle

What's the point of going all Bayesian then

- (Automatic) Model Selection for Unsupervised Learning / Sélection automatique du complexité du modèle
- Model Averaging (Model plays all its cards) / On calcule un moyenne sur les modèles

What's the point of going all Bayesian then

- (Automatic) Model Selection for Unsupervised Learning / Sélection automatique du complexité du modèle
- Model Averaging (Model plays all its cards) / On calcule un moyenne sur les modèles
- Principled way of regularization / Un moyenne de regularization

What's the point of going all Bayesian then

- (Automatic) Model Selection for Unsupervised Learning / Sélection automatique du complexité du modèle
- Model Averaging (Model plays all its cards) / On calcule un moyenne sur les modèles
- Principled way of regularization / Un moyenne de regularization
- All of these 4 variants are extendable for other models. We can play with: / Ces 4 idées sont extremement puissants, on peut capturer pleins de modèles.
  - Distribution of h.
  - Impose structure on h. / Imposition d'une structure sur h (semaine prochaine)
  - We can change the conditional distribution $p(x|h, \theta)$. (Application decides) / Dépend sur l'output.
  - We can play with how we do inference and learning. / On peut changer comment on fait apprentissage.

Table of Contents

Centroid based approaches
- K-means clustering
- Gaussian Mixture Model
- More Advanced GMM Learning Methods

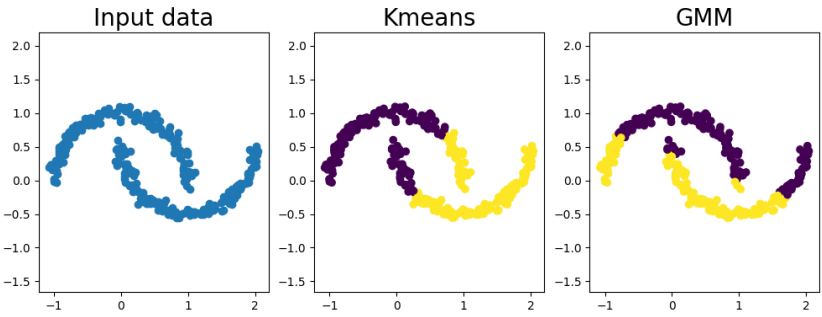
Spectral Clustering

Hierarchical Clustering

Deep Generative Models
- Variational Autoencoders
- Generative Adversarial Networks
- Diffusion Models
- Audio Diffusion Models

Failure case for Centroid Based Methods

- What if we have something like this? / Et si on avait qqch comme ça?



- Any ideas / Idées?

Remember KPCA?

- Do you remember KPCA? / Vous souvenez-vous de KPCA?
- Let's calculate a pairwise distance matrix

Spectral Clustering

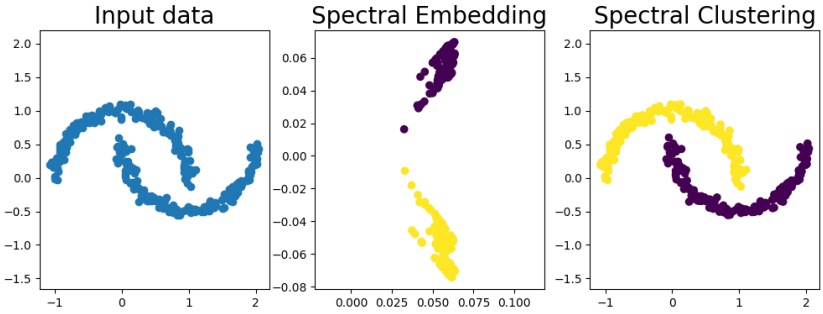
- $A_{ij} = \exp\left(-\gamma \|x_i - x_j\|_2^2\right),\ i \neq j.\ A_{ij} = 0\ i = j.$
- Compute Graph Laplacian

$$L := D^{-1/2} A D^{-1/2},\ D_{ii} = \sum_j A_{ij}$$

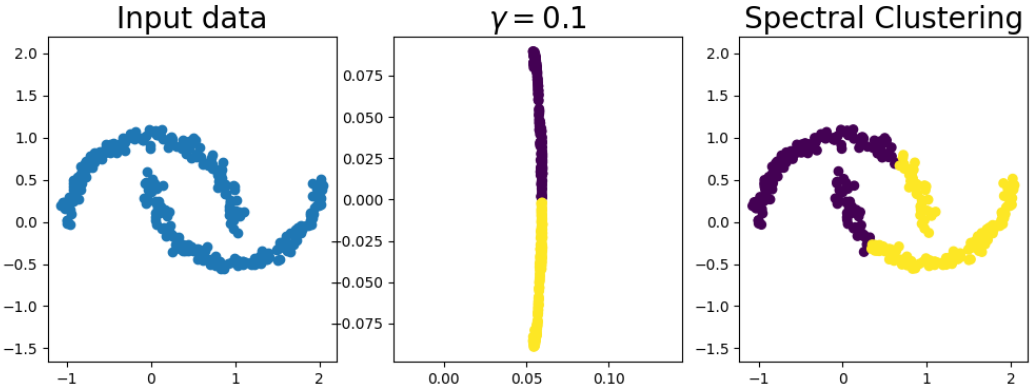
.

- K-means cluster the first k-eigenvectors of L.

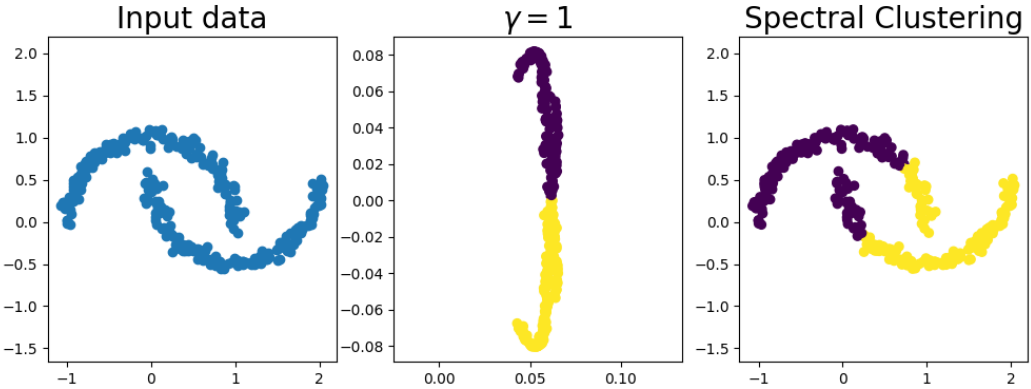
Spectral Clustering in Action



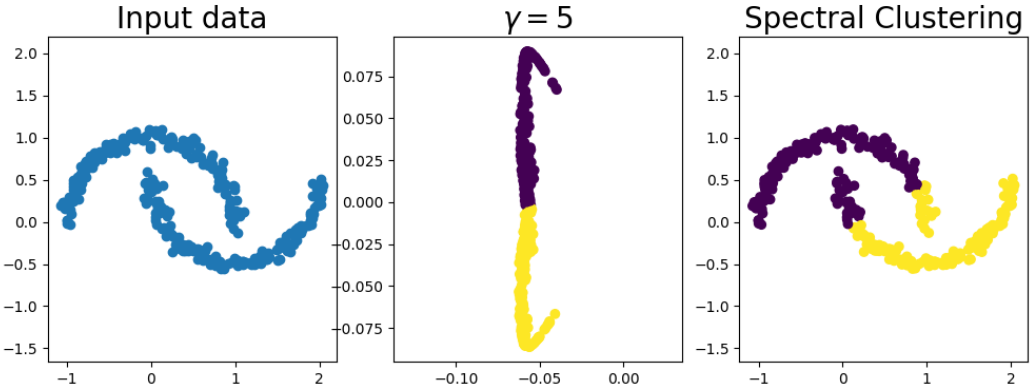
Spectral Clustering, effect of Gamma



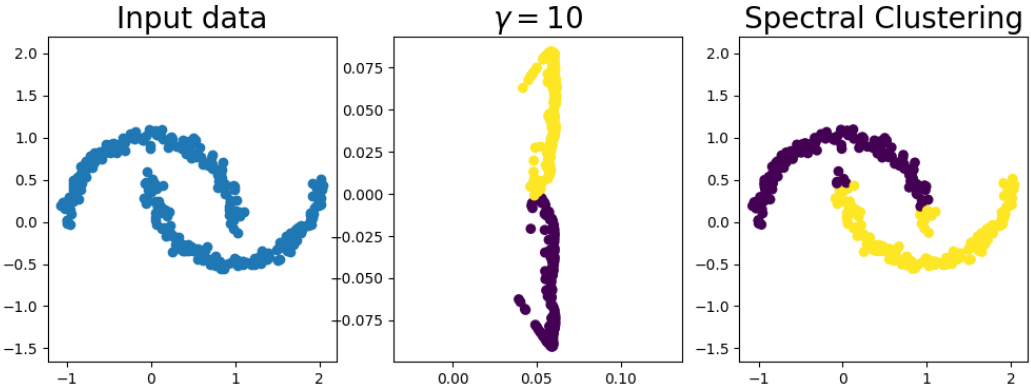
Spectral Clustering, effect of Gamma



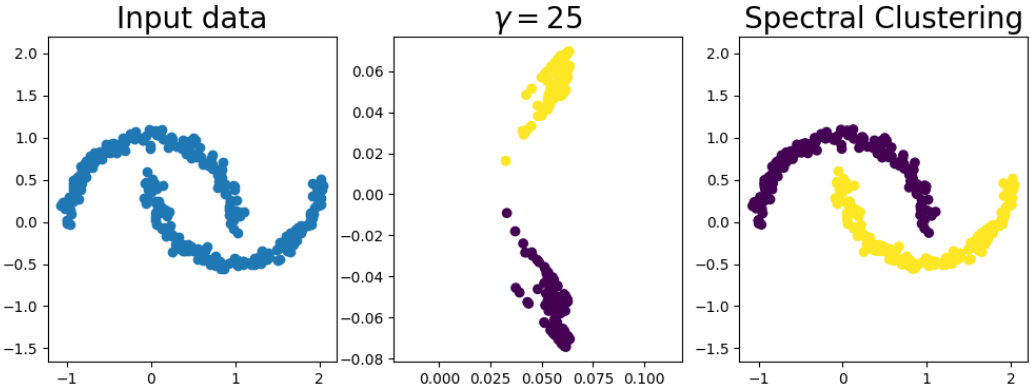
Spectral Clustering, effect of Gamma



Spectral Clustering, effect of Gamma



Spectral Clustering, effect of Gamma



Spectral Clustering, effect of Gamma

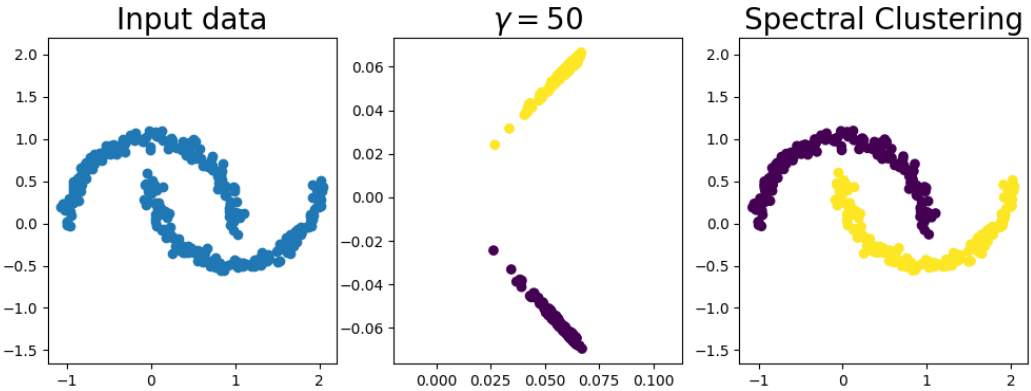


Table of Contents

Centroid based approaches
- K-means clustering
- Gaussian Mixture Model
- More Advanced GMM Learning Methods

Spectral Clustering

Hierarchical Clustering

Deep Generative Models
- Variational Autoencoders
- Generative Adversarial Networks
- Diffusion Models
- Audio Diffusion Models

Agglomerative Clustering

- Choose $R_0 = \{C_i = \{x_i\}, i = 1, \ldots N\}$. We start from each data item being a cluster. / On commence avec chaque data étant un cluster.

Agglomerative Clustering

- Choose $R_0 = \{C_i = \{x_i\}, i = 1, \ldots N\}$. We start from each data item being a cluster. / On commence avec chaque data étant un cluster.
- For $t = 1, \ldots$

Agglomerative Clustering

- Choose $R_0 = \{C_i = \{x_i\}, i = 1, \ldots N\}$. We start from each data item being a cluster. / On commence avec chaque data étant un cluster.
- For $t = 1, \ldots$
  - Among all clusters in / Parmi tous les clusters dans R_{t-1}, find cluster pair / on trouve un paire $\{C_i, C_j\}$ such that / telle que,

$$i, j = \arg\min_{i', j'} d(C_{i'}, C_{j'})$$

.

Agglomerative Clustering

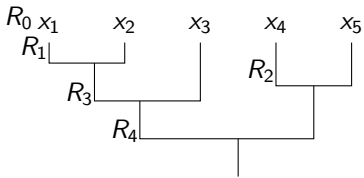
- Choose $R_0 = \{C_i = \{x_i\}, i = 1, \ldots N\}$. We start from each data item being a cluster. / On commence avec chaque data étant un cluster.
- For $t = 1, \ldots$
  - Among all clusters in / Parmi tous les clusters dans R_{t-1}, find cluster pair / on trouve un paire $\{C_i, C_j\}$ such that / telle que,

$$i, j = \arg\min_{i', j'} d(C_{i'}, C_{j'})$$

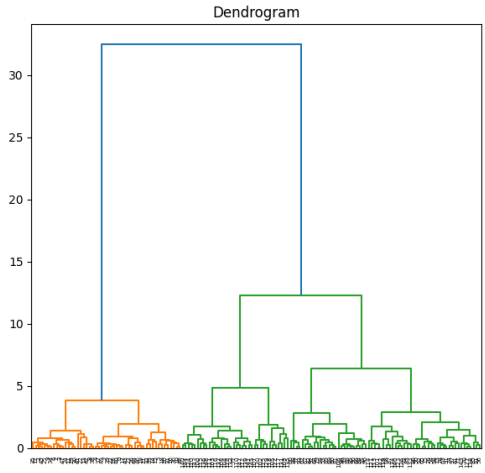
  .
  - Form new cluster pair, remove the pair / Forme un nouvelle paire de clusters, enlève l'ancien,

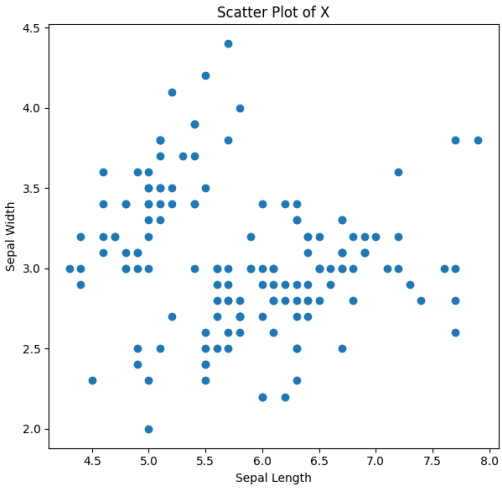
$$C_q = C_i \cup C_j, \; R_t = (R_{t-1} - C_i \cup C_j) \cup C_q$$

Dendrogram



Example Dendrogram on IRIS

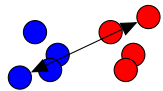




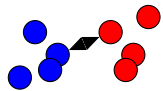
With Ward Linkage

How do we measure distances between clusters

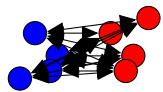
- Linkage functions
  - Complete linkage



  - Single linkage

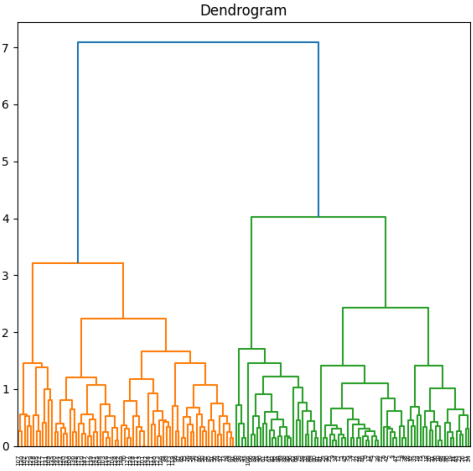


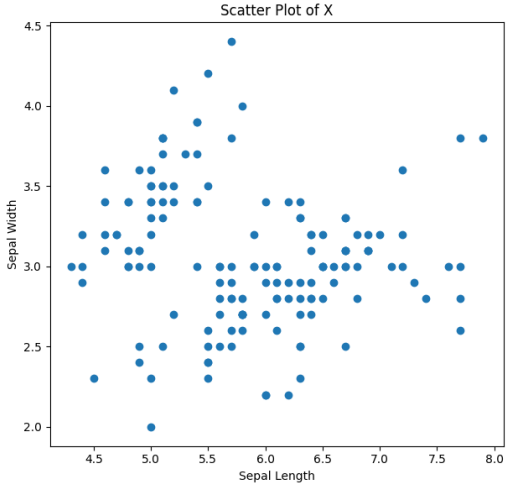
  - Average linkage



- There's more / Il y en a d'autres.

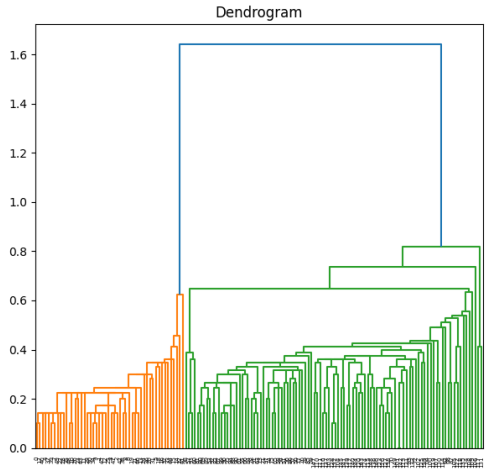
Example Dendrogram on IRIS

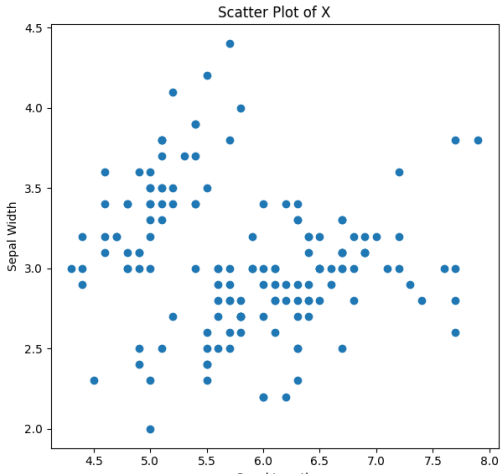




Complete Linkage

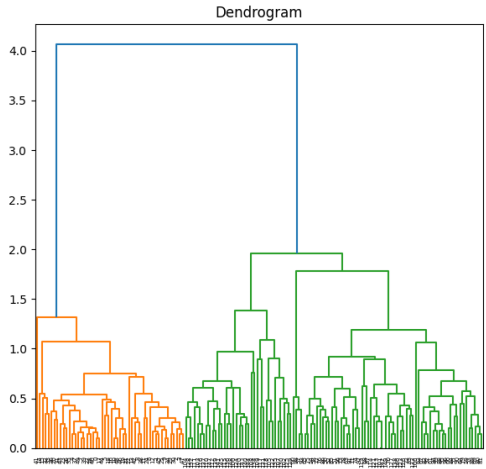
Example Dendrogram on IRIS

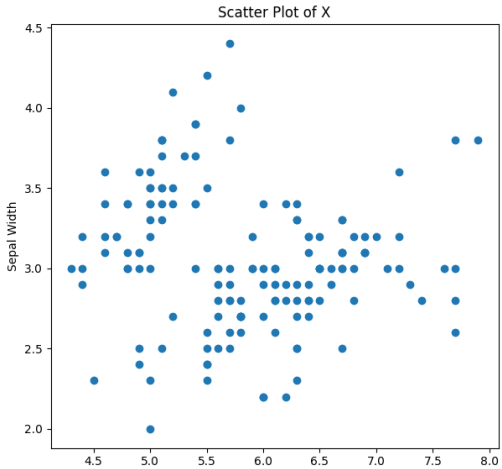




Single Linkage

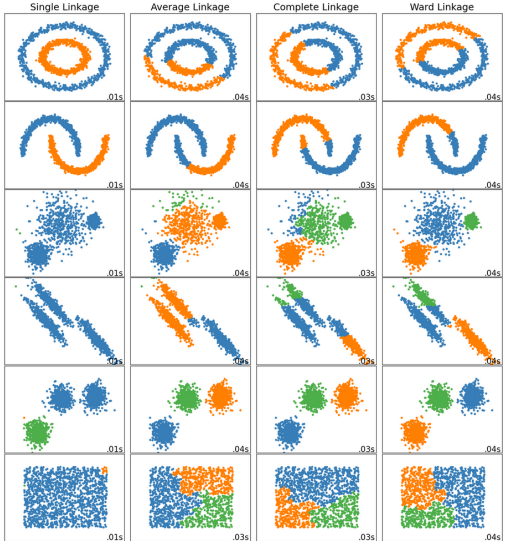
Example Dendrogram on IRIS





Average Linkage

How does linkage change things



Computational Complexity

- At level t we have $N - t$ clusters / Au niveau t, on a $N - t$ clusters.
- How many pairs of clusters do we need to deal with? Combien de pairs?

Computational Complexity

- At level t we have $N - t$ clusters / Au niveau t, on a $N - t$ clusters.
- How many pairs of clusters do we need to deal with? Combien de pairs?
  - $\binom{N-t}{2} = \frac{(N-t)(N-t-1)}{2}$.
- In total / Au totale:
  - $\sum_{t=0}^{N-1} \binom{N-t}{2} = \frac{(N-1)N(N+1)}{6}$

Computational Complexity

- At level t we have $N - t$ clusters / Au niveau t, on a $N - t$ clusters.
- How many pairs of clusters do we need to deal with? Combien de pairs?
  - $\binom{N-t}{2} = \frac{(N-t)(N-t-1)}{2}$.
- In total / Au totale:
  - $\sum_{t=0}^{N-1} \binom{N-t}{2} = \frac{(N-1)N(N+1)}{6}$
- So N^3. In the original El-Capitan Image we had $N \approx 500000$. That would give something like 2.083×10^{16}. / Dans l'image originale El-Capitan on avait $N \approx 500000$. Donc.. merci mais non merci..

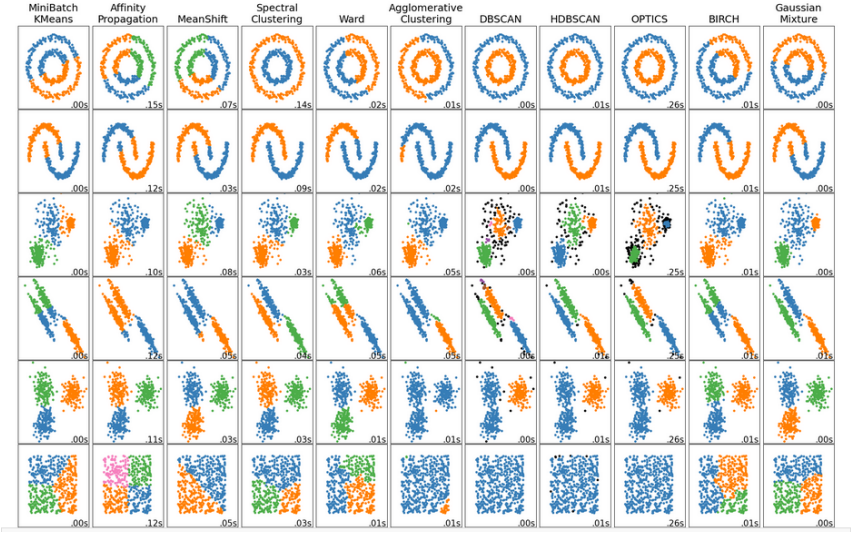
Divisive Clustering

- We start with the whole dataset forming a giant cluster. / On commence avec une seule cluster.
- Then we divide by picking out the least similar clusters / Puis on divise en choissisant les cluster les moins similaires.
- They say that divisive usually works better than agglomerative since it sees the global picture better.
  - C'est accepté que la divisive travaille mieux car il voit le paysage globale mieux.
- Agglomerative is typically faster / Agglomerative est typiquement plus rapide.

Clustering recap

- Kmeans
  - Reliable (Fiable), quick and dirty
- GMMs
  - More powerful than Kmeans but still a centroid method in heart / Plus puissant que Kmeans mais une méthode de centroids si on y pense.
- Spectral Clustering
  - The non-linear get around to find manifold-like clusters / Une cheminement alternative pour trouver des clusters qui sont comme des manifolds. Not suitable for large datasets / N'est pas approprié pour des grands datasets.
- Hiearchical Clustering
  - Gives a dendrogram, but expensive! Donne un dendrogram, est un bon utile, mais cher!

There's more!


MiniBatch KMeans
Affinity Propagation
MeanShift
Spectral Clustering
Ward
Agglomerative Clustering
DBSCAN
HDBSCAN
OPTICS
BIRCH
Gaussian Mixture

Suggested reading

- Spectral clustering: `http://ai.stanford.edu/~ang/papers/nips01-spectral.pdf`
- Bishop chapter 9.

Table of Contents

Centroid based approaches
- K-means clustering
- Gaussian Mixture Model
- More Advanced GMM Learning Methods

Spectral Clustering

Hierarchical Clustering

Deep Generative Models
- Variational Autoencoders
- Generative Adversarial Networks
- Diffusion Models
- Audio Diffusion Models

Generative models

- Goal: learn $p(x)$ from data, then **sample** new x's.
  - But: apprendre $p(x)$ à partir des données, puis **générer** de nouveaux x.

Generative models

- Goal: learn $p(x)$ from data, then **sample** new x's.
  - But: apprendre $p(x)$ à partir des données, puis **générer** de nouveaux x.
- We already have one: the GMM! Sample $h \sim \text{Discrete}(\pi)$, then $x \sim \mathcal{N}(\mu_h, \Sigma_h)$.
  - On en a déjà un: le GMM!

Generative models

- Goal: learn $p(x)$ from data, then **sample** new x's.
  - But: apprendre $p(x)$ à partir des données, puis **générer** de nouveaux x.
- We already have one: the GMM! Sample $h \sim \text{Discrete}(\pi)$, then $x \sim \mathcal{N}(\mu_h, \Sigma_h)$.
  - On en a déjà un: le GMM!
- But how many Gaussians would we need for a spectrogram, or an image?
  - Mais combien de Gaussiennes faudrait-il pour un spectrogramme, ou une image?

Generative models

- Goal: learn $p(x)$ from data, then **sample** new x's.
  - But: apprendre $p(x)$ à partir des données, puis **générer** de nouveaux x.
- We already have one: the GMM! Sample $h \sim \text{Discrete}(\pi)$, then $x \sim \mathcal{N}(\mu_h, \Sigma_h)$.
  - On en a déjà un: le GMM!
- But how many Gaussians would we need for a spectrogram, or an image?
  - Mais combien de Gaussiennes faudrait-il pour un spectrogramme, ou une image?
- Idea: keep the latent variable, replace the Gaussian by a **neural network**.
  - Idée: garder la variable latente, remplacer la Gaussienne par un **réseau de neurones**.

A map of deep generative models

	How it generates	**Training**	+	−
VAE	decode $z \sim \mathcal{N}(0, I)$	ELBO	fast, encoder	blurry samples
GAN	$G(z)$, $z \sim \mathcal{N}(0, I)$	fool a discriminator	sharp, fast	unstable
Diffusion	denoise step by step	ELBO $\rightarrow$ denoising	quality, diversity	slow sampling
Autoregressive	one element at a time	exact likelihood	quality	slow, sequential

- Autoregressive models: WaveNet, and the LLMs from last week.
  - Modèles autorégressifs: WaveNet, et les LLMs de la semaine passée.

A map of deep generative models

	How it generates	**Training**	+	−
VAE	decode $z \sim \mathcal{N}(0, I)$	ELBO	fast, encoder	blurry samples
GAN	$G(z)$, $z \sim \mathcal{N}(0, I)$	fool a discriminator	sharp, fast	unstable
Diffusion	denoise step by step	ELBO $\rightarrow$ denoising	quality, diversity	slow sampling
Autoregressive	one element at a time	exact likelihood	quality	slow, sequential

- Autoregressive models: WaveNet, and the LLMs from last week.
  - Modèles autorégressifs: WaveNet, et les LLMs de la semaine passée.
- Today: VAE $\rightarrow$ GAN $\rightarrow$ Diffusion $\rightarrow$ audio diffusion models (AudioLDM, AudioLDM 2).
  - Aujourd'hui: VAE $\rightarrow$ GAN $\rightarrow$ Diffusion $\rightarrow$ modèles de diffusion pour l'audio.

Table of Contents

Centroid based approaches
- K-means clustering
- Gaussian Mixture Model
- More Advanced GMM Learning Methods

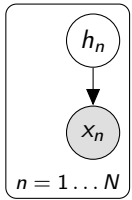
Spectral Clustering

Hierarchical Clustering

Deep Generative Models
- Variational Autoencoders
- Generative Adversarial Networks
- Diffusion Models
- Audio Diffusion Models

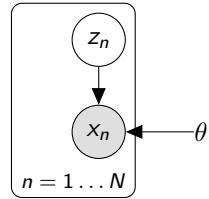
From the GMM to a deep latent variable model

GMM



$$h_n \sim \text{Discrete}(\pi)$$
$$x_n | h_n \sim \mathcal{N}(\mu_{h_n}, \Sigma_{h_n})$$

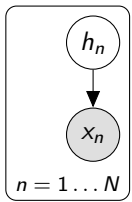
Deep latent variable model



$$z_n \sim \mathcal{N}(0, I)$$
$$x_n | z_n \sim \mathcal{N}(g_\theta(z_n), \sigma^2 I), \quad g_\theta = \text{neural net}$$

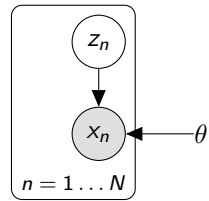
From the GMM to a deep latent variable model

GMM



$$h_n \sim \text{Discrete}(\pi)$$
$$x_n | h_n \sim \mathcal{N}(\mu_{h_n}, \Sigma_{h_n})$$

Deep latent variable model

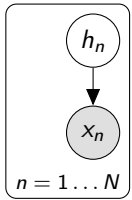


$$z_n \sim \mathcal{N}(0, I)$$
$$x_n | z_n \sim \mathcal{N}(g_\theta(z_n), \sigma^2 I), \quad g_\theta = \text{neural net}$$

- $p_\theta(x) = \int \mathcal{N}(x; g_\theta(z), \sigma^2 I)\, \mathcal{N}(z; 0, I)\, dz$: an infinite mixture of Gaussians. / Un mélange infini de Gaussiennes.

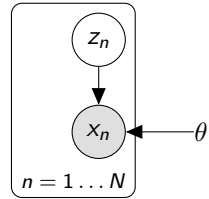
From the GMM to a deep latent variable model

GMM



$$h_n \sim \text{Discrete}(\pi)$$
$$x_n | h_n \sim \mathcal{N}(\mu_{h_n}, \Sigma_{h_n})$$

Deep latent variable model



$$z_n \sim \mathcal{N}(0, I)$$
$$x_n | z_n \sim \mathcal{N}(g_\theta(z_n), \sigma^2 I), \quad g_\theta = \text{neural net}$$

- $p_\theta(x) = \int \mathcal{N}(x; g_\theta(z), \sigma^2 I)\, \mathcal{N}(z; 0, I)\, dz$: an infinite mixture of Gaussians. / Un mélange infini de Gaussiennes.
- Generating is easy (sample z, run g_θ). Learning is hard: we cannot compute this integral. / Générer est facile, apprendre est difficile.

Why EM does not work anymore

- EM needs the posterior of the latent variable in the E-step. / EM a besoin de la postérieure.

Why EM does not work anymore

- EM needs the posterior of the latent variable in the E-step. / EM a besoin de la postérieure.
- GMM: a sum over K clusters, easy:

$$q(h_n = k) = \frac{\pi_k \mathcal{N}(x_n; \mu_k, \Sigma_k)}{\sum_{k'} \pi_{k'} \mathcal{N}(x_n; \mu_{k'}, \Sigma_{k'})}$$

Why EM does not work anymore

- EM needs the posterior of the latent variable in the E-step. / EM a besoin de la postérieure.
- GMM: a sum over K clusters, easy:

$$q(h_n = k) = \frac{\pi_k \mathcal{N}(x_n; \mu_k, \Sigma_k)}{\sum_{k'} \pi_{k'} \mathcal{N}(x_n; \mu_{k'}, \Sigma_{k'})}$$

- Deep latent variable model: an integral through a neural network, intractable:

$$p_\theta(z|x) = \frac{p_\theta(x|z)\, p(z)}{\int p_\theta(x|z')\, p(z')\, dz'}$$

  - Intégrale à travers un réseau de neurones: impossible à calculer.

Why EM does not work anymore

- EM needs the posterior of the latent variable in the E-step. / EM a besoin de la postérieure.
- GMM: a sum over K clusters, easy:

$$q(h_n = k) = \frac{\pi_k \mathcal{N}(x_n; \mu_k, \Sigma_k)}{\sum_{k'} \pi_{k'} \mathcal{N}(x_n; \mu_{k'}, \Sigma_{k'})}$$

- Deep latent variable model: an integral through a neural network, intractable:

$$p_\theta(z|x) = \frac{p_\theta(x|z)\, p(z)}{\int p_\theta(x|z')\, p(z')\, dz'}$$

  - Intégrale à travers un réseau de neurones: impossible à calculer.
- So: approximate the posterior with a distribution $q(z)$, and maximize the bound. Exactly what we did in Learning Variant 2!
  - On approxime la postérieure avec $q(z)$ et on maximise la borne, comme dans Learning Variant 2!

The ELBO (it is the VLB from the GMM)

$$
\begin{aligned}
\log p_\theta(x) &= \log \int p_\theta(x|z)\, p(z)\, dz = \log \mathbb{E}_{q(z)} \left[\frac{p_\theta(x|z)\, p(z)}{q(z)} \right] \\
&\geq \mathbb{E}_{q(z)} \left[\log \frac{p_\theta(x|z)\, p(z)}{q(z)} \right] \qquad \text{(Jensen)} \\
&= \underbrace{\mathbb{E}_{q(z)} \big[\log p_\theta(x|z) \big]}_{\text{reconstruction}} - \underbrace{KL\big(q(z) \,\|\, p(z)\big)}_{\text{stay close to the prior}} =: \text{ ELBO}
\end{aligned}
$$

The ELBO (it is the VLB from the GMM)

$$\begin{aligned}
\log p_\theta(x) &= \log \int p_\theta(x|z)\, p(z)\, dz = \log \mathbb{E}_{q(z)}\left[\frac{p_\theta(x|z)\, p(z)}{q(z)}\right] \\
&\geq \mathbb{E}_{q(z)}\left[\log \frac{p_\theta(x|z)\, p(z)}{q(z)}\right] \qquad \text{(Jensen)} \\
&= \underbrace{\mathbb{E}_{q(z)}\big[\log p_\theta(x|z)\big]}_{\text{reconstruction}} - \underbrace{KL\big(q(z)\,\|\,p(z)\big)}_{\text{stay close to the prior}} =: \text{ ELBO}
\end{aligned}$$

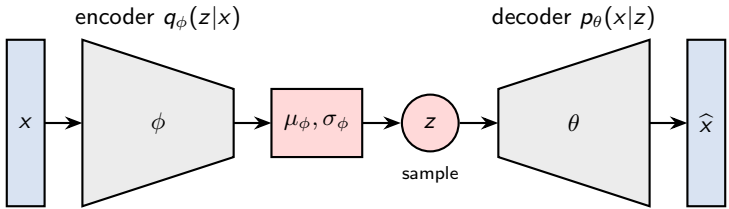
- The gap: $\log p_\theta(x) - \text{ELBO} = KL\big(q(z)\,\|\,p_\theta(z|x)\big) \geq 0$.
  - L'écart est nul quand q est la vraie postérieure (c'était l'E-step!).

The ELBO (it is the VLB from the GMM)

$$\begin{aligned}
\log p_\theta(x) &= \log \int p_\theta(x|z)\, p(z)\, dz = \log \mathbb{E}_{q(z)}\left[\frac{p_\theta(x|z)\, p(z)}{q(z)}\right] \\
&\geq \mathbb{E}_{q(z)}\left[\log \frac{p_\theta(x|z)\, p(z)}{q(z)}\right] \qquad \text{(Jensen)} \\
&= \underbrace{\mathbb{E}_{q(z)}\big[\log p_\theta(x|z)\big]}_{\text{reconstruction}} - \underbrace{KL\big(q(z)\,\|\,p(z)\big)}_{\text{stay close to the prior}} =: \text{ELBO}
\end{aligned}$$

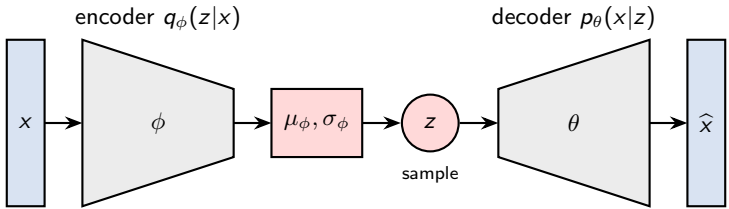
- The gap: $\log p_\theta(x) - \text{ELBO} = KL\big(q(z)\,\|\,p_\theta(z|x)\big) \geq 0$.
  - L'écart est nul quand q est la vraie postérieure (c'était l'E-step!).
- Evidence Lower BOund = VLB. Same derivation, new names. / Même dérivation, nouveaux noms.

Amortized inference: the encoder



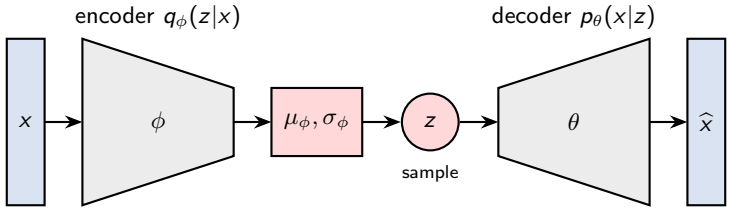
- GMM: one $q(h_n)$ per data item, recomputed in every E-step. / Un $q(h_n)$ par donnée.

Amortized inference: the encoder



- GMM: one $q(h_n)$ per data item, recomputed in every E-step. / Un $q(h_n)$ par donnée.
- VAE: one network for all items: $q_\phi(z|x) = \mathcal{N}\big(z; \mu_\phi(x), \mathrm{diag}(\sigma^2_\phi(x))\big)$.
  - Un seul réseau pour toutes les données: c'est l'inférence amortie.

Amortized inference: the encoder



- GMM: one $q(h_n)$ per data item, recomputed in every E-step. / Un $q(h_n)$ par donnée.
- VAE: one network for all items: $q_\phi(z|x) = \mathcal{N}\big(z; \mu_\phi(x), \mathrm{diag}(\sigma^2_\phi(x))\big)$.
  - Un seul réseau pour toutes les données: c'est l'inférence amortie.
- Train encoder and decoder together: $\max_{\theta,\phi} \sum_n \mathrm{ELBO}(x_n; \theta, \phi)$.
- This is the picture from week 5. Now we know where the loss comes from! / L'image de la semaine 5, avec la fonction de coût.

The two terms of the ELBO

- Reconstruction, with a Gaussian decoder: / Reconstruction, avec un décodeur Gaussien:

$$\log p_\theta(x|z) = -\frac{1}{2\sigma^2}\|x - g_\theta(z)\|_2^2 + \text{const}$$

  - The usual autoencoder loss (MSE). / La perte d'un autoencodeur classique.

The two terms of the ELBO

- Reconstruction, with a Gaussian decoder: / Reconstruction, avec un décodeur Gaussien:

$$\log p_\theta(x|z) = -\frac{1}{2\sigma^2}\|x - g_\theta(z)\|_2^2 + \text{const}$$

  - The usual autoencoder loss (MSE). / La perte d'un autoencodeur classique.
- KL between two Gaussians, in closed form: / KL entre deux Gaussiennes, en forme fermée:

$$KL\big(\mathcal{N}(\mu, \text{diag}(\sigma^2)) \,\|\, \mathcal{N}(0, I)\big) = \frac{1}{2}\sum_j \big(\mu_j^2 + \sigma_j^2 - \log \sigma_j^2 - 1\big)$$

  - A regularizer: codes stay near $\mathcal{N}(0, I)$, so the decoder knows what to do with prior samples.
  - Un régularisateur: les codes restent près de $\mathcal{N}(0, I)$, là où on va échantillonner.

The two terms of the ELBO

- Reconstruction, with a Gaussian decoder: / Reconstruction, avec un décodeur Gaussien:

$$\log p_\theta(x|z) = -\frac{1}{2\sigma^2}\|x - g_\theta(z)\|_2^2 + \text{const}$$

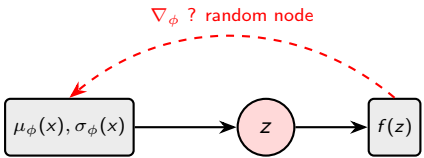
  - The usual autoencoder loss (MSE). / La perte d'un autoencodeur classique.
- KL between two Gaussians, in closed form: / KL entre deux Gaussiennes, en forme fermée:

$$KL\big(\mathcal{N}(\mu, \text{diag}(\sigma^2)) \,\|\, \mathcal{N}(0, I)\big) = \frac{1}{2}\sum_j \left(\mu_j^2 + \sigma_j^2 - \log \sigma_j^2 - 1\right)$$

  - A regularizer: codes stay near $\mathcal{N}(0, I)$, so the decoder knows what to do with prior samples.
  - Un régularisateur: les codes restent près de $\mathcal{N}(0, I)$, là où on va échantillonner.
- The decoder variance σ^2 sets the balance between the two terms (β-VAE, week 5).
  - La variance σ^2 règle l'équilibre entre les deux termes.

The reparameterization trick

- We need $\nabla_\phi \mathbb{E}_{q_\phi(z|x)}[f(z)]$, but ϕ is inside the sampling. / Comment dériver à travers un échantillonnage?

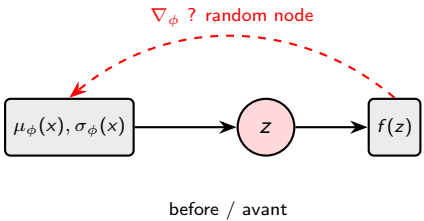


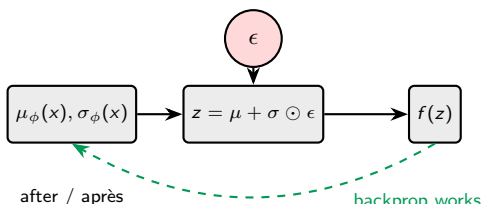
before / avant

The reparameterization trick

- We need $\nabla_\phi \mathbb{E}_{q_\phi(z|x)}[f(z)]$, but ϕ is inside the sampling. / Comment dériver à travers un échantillonnage?
- Write the sample as a deterministic function of ϕ and some fixed noise: / On sort le bruit:

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \quad \Rightarrow \quad \nabla_\phi \mathbb{E}_{q_\phi}[f(z)] = \mathbb{E}_\epsilon\left[\nabla_\phi f(\mu_\phi(x) + \sigma_\phi(x) \odot \epsilon)\right]$$

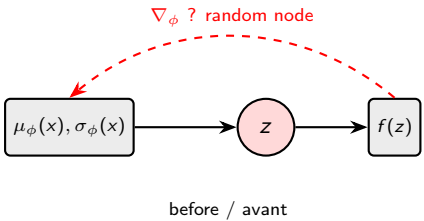




The reparameterization trick

- We need $\nabla_\phi \mathbb{E}_{q_\phi(z|x)}[f(z)]$, but ϕ is inside the sampling. / Comment dériver à travers un échantillonnage?
- Write the sample as a deterministic function of ϕ and some fixed noise: / On sort le bruit:

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \quad \Rightarrow \quad \nabla_\phi \mathbb{E}_{q_\phi}[f(z)] = \mathbb{E}_\epsilon\left[\nabla_\phi f(\mu_\phi(x) + \sigma_\phi(x) \odot \epsilon)\right]$$



- One ϵ per example is enough: SGD averages the noise.

Kingma & Welling, ICLR 2014; Rezende et al., ICML 2014

A VAE in PyTorch

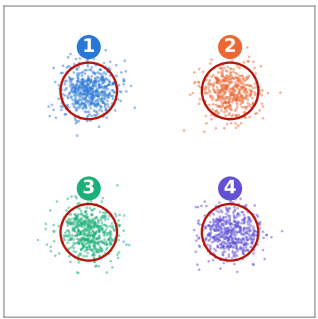
```
class VAE(nn.Module):
    def __init__(self, d_x, d_z, d_h=256):
        super().__init__()
        self.enc = nn.Sequential(nn.Linear(d_x, d_h), nn.ReLU(),
                                 nn.Linear(d_h, 2 * d_z))
        self.dec = nn.Sequential(nn.Linear(d_z, d_h), nn.ReLU(),
                                 nn.Linear(d_h, d_x))

    def forward(self, x):                        # (B, d_x)
        mu, logvar = self.enc(x).chunk(2, -1)  # (B, d_z) each
        eps = torch.randn_like(mu)
        z = mu + torch.exp(0.5 * logvar) * eps # reparam. trick
        x_hat = self.dec(z)
        rec = ((x - x_hat) ** 2).sum(-1)       # -log p(x|z)
        kl = 0.5 * (mu**2 + logvar.exp() - logvar - 1).sum(-1)
        return (rec + kl).mean()               # -ELBO

# generation: decode samples from the prior
x_new = model.dec(torch.randn(16, d_z))
```

- Predict $\log \sigma^2$, not σ: any real number is fine. / On prédit $\log \sigma^2$.
- The loss is $-$ELBO: minimize it with the training loop of week 7.
  - La perte est $-$ELBO.
- Generation only needs the decoder. / Pour générer, seul le décodeur sert.

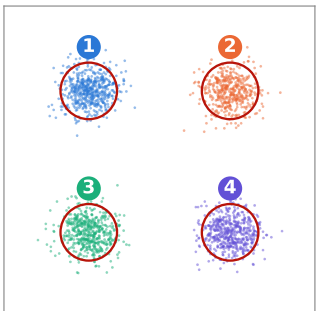
A toy problem: 4 Gaussians on a grid



red circles: 2 std of each Gaussian

- $p_{\text{data}}(x) = \frac{1}{4}\sum_{k=1}^{4} \mathcal{N}(x; \mu_k, 0.2^2 I)$, $\mu_k \in \{-1, 1\}^2$: a GMM from the first half!
  - Un GMM de la première partie. On connaît la vraie densité: on peut voir ce que chaque modèle apprend.

A toy problem: 4 Gaussians on a grid

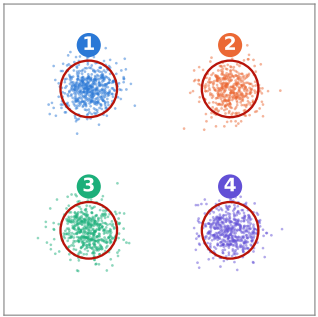


red circles: 2 std of each Gaussian

- $p_{\text{data}}(x) = \frac{1}{4}\sum_{k=1}^{4} \mathcal{N}(x; \mu_k, 0.2^2 I)$, $\mu_k \in \{-1, 1\}^2$: a GMM from the first half!
  - Un GMM de la première partie. On connaît la vraie densité: on peut voir ce que chaque modèle apprend.
- Same network for every model, only the objective changes: / Même réseau, seul l'objectif change:
  - `mlp`: $d_{\text{in}} \to 128 \to 128 \to 128 \to d_{\text{out}}$, SiLU, about 34k parameters.

	Networks	**Objective**
VAE	encoder `mlp(2, 4)`, decoder `mlp(2, 2)`	$-$ELBO
GAN	discriminator `mlp(2, 1)`, generator `mlp(2, 2)`	minimax
Diffusion	denoiser `mlp(2 + 16, 2)`: $[x_t, \text{emb}(t)]$	$\|\epsilon - \epsilon_\theta\|^2$

A toy problem: 4 Gaussians on a grid



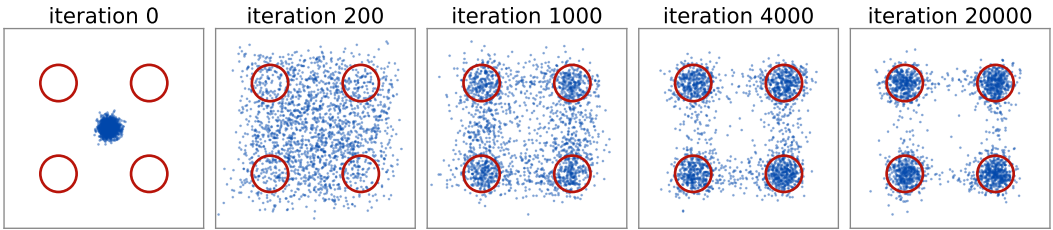
red circles: 2 std of each Gaussian

- $p_{\text{data}}(x) = \frac{1}{4}\sum_{k=1}^{4} \mathcal{N}(x; \mu_k, 0.2^2 I)$, $\mu_k \in \{-1, 1\}^2$: a GMM from the first half!
  - Un GMM de la première partie. On connaît la vraie densité: on peut voir ce que chaque modèle apprend.
- Same network for every model, only the objective changes: / Même réseau, seul l'objectif change:
  - `mlp`: $d_{\text{in}} \to 128 \to 128 \to 128 \to d_{\text{out}}$, SiLU, about 34k parameters.

	Networks	**Objective**
VAE	encoder `mlp(2, 4)`, decoder `mlp(2, 2)`	$-$ELBO
GAN	discriminator `mlp(2, 1)`, generator `mlp(2, 2)`	minimax
Diffusion	denoiser `mlp(2 + 16, 2)`: $[x_t, \text{emb}(t)]$	$\|\epsilon - \epsilon_\theta\|^2$

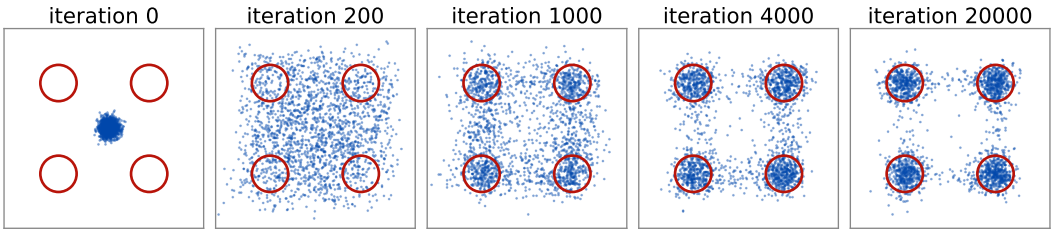
- Adam, batch 512, 20 000 steps (GAN: smaller learning rate, as usual for GANs). 7 minutes for all three on a 2-core CPU. Code: `toy_generative.py`.

Toy problem: the VAE



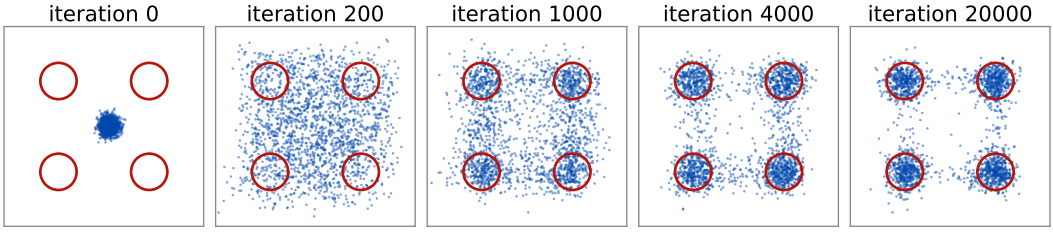
- Samples $x \sim p_\theta(x)$ during training: $z \sim \mathcal{N}(0, I)$, then $x = g_\theta(z) + \sigma\epsilon$. Red circles: 2 std of the true Gaussians.
  - Échantillons pendant l'entraînement. Cercles rouges: 2 écarts-types des vraies Gaussiennes.

Toy problem: the VAE



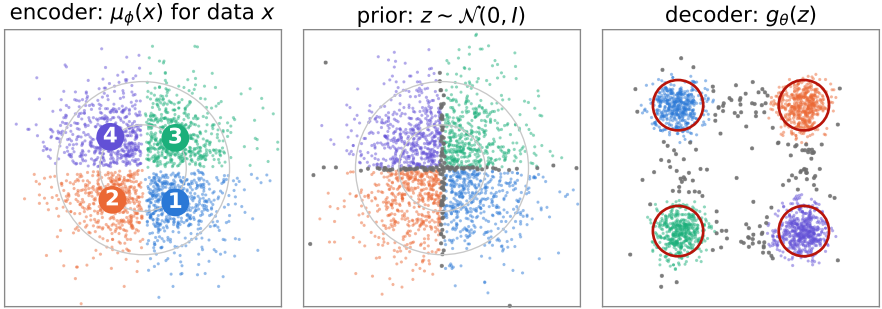
- Samples $x \sim p_\theta(x)$ during training: $z \sim \mathcal{N}(0, I)$, then $x = g_\theta(z) + \sigma\epsilon$. Red circles: 2 std of the true Gaussians.
  - Échantillons pendant l'entraînement. Cercles rouges: 2 écarts-types des vraies Gaussiennes.
- The code of the previous slide, with $d_z = 2$ and a learned decoder variance σ^2, started at 0.1^2.
  - Started at $\sigma = 1$, the KL term wins early and the decoder ignores z: posterior collapse! (our first try)
  - Avec $\sigma = 1$ au départ, le décodeur ignore z: notre premier essai s'est effondré!

Toy problem: the VAE



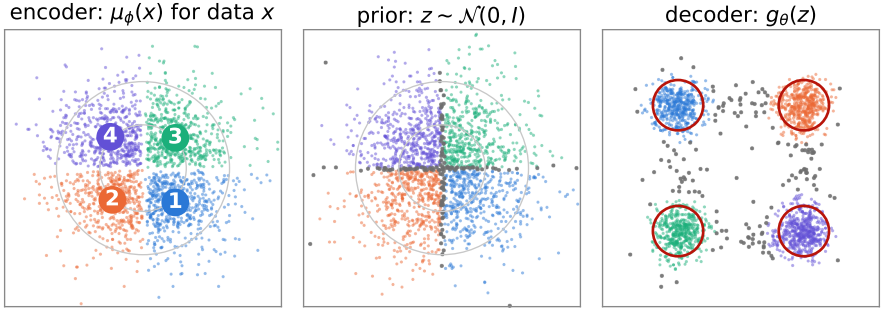
- Samples $x \sim p_\theta(x)$ during training: $z \sim \mathcal{N}(0, I)$, then $x = g_\theta(z) + \sigma\epsilon$. Red circles: 2 std of the true Gaussians.
  - Échantillons pendant l'entraînement. Cercles rouges: 2 écarts-types des vraies Gaussiennes.
- The code of the previous slide, with $d_z = 2$ and a learned decoder variance σ^2, started at 0.1^2.
  - Started at $\sigma = 1$, the KL term wins early and the decoder ignores z: posterior collapse! (our first try)
  - Avec $\sigma = 1$ au départ, le décodeur ignore z: notre premier essai s'est effondré!
- All 4 modes are found, but they are connected by **bridges** of samples. Why?
  - Les 4 modes sont trouvés, mais reliés par des ponts. Pourquoi?

Toy problem: inside the VAE



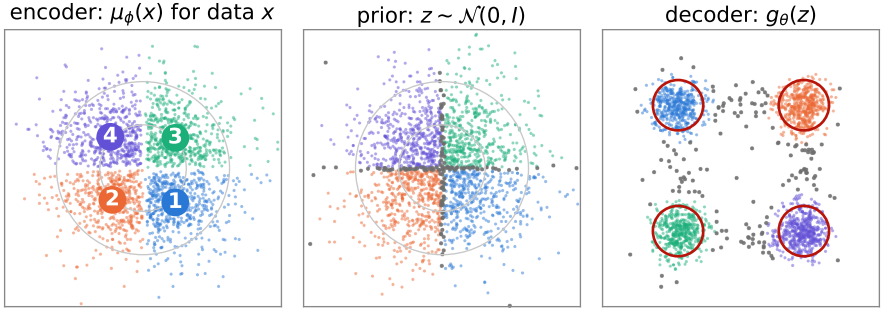
- The encoder sends each Gaussian to its own region of z-space. / Chaque Gaussienne a sa région de l'espace latent.

Toy problem: inside the VAE



- The encoder sends each Gaussian to its own region of z-space. / Chaque Gaussienne a sa région de l'espace latent.
- But $\mathcal{N}(0, I)$ is one connected blob, and g_θ is continuous: the borders between the regions (gray) are decoded **between** the modes.
  - Le prior est d'un seul morceau et le décodeur est continu: les frontières sont décodées entre les modes.

Toy problem: inside the VAE



- The encoder sends each Gaussian to its own region of z-space. / Chaque Gaussienne a sa région de l'espace latent.
- But $\mathcal{N}(0, I)$ is one connected blob, and g_θ is continuous: the borders between the regions (gray) are decoded **between** the modes.
  - Le prior est d'un seul morceau et le décodeur est continu: les frontières sont décodées entre les modes.
- In images and audio: blurry averages of plausible outputs. / Le même effet donne le flou.

What does a VAE give us?

- + Generation: $z \sim \mathcal{N}(0, I)$, then decode. / Génération.
- + A smooth latent space: interpolate between two sounds by interpolating their z's.
  - Un espace latent lisse: on peut interpoler entre deux sons.
- + Compression: $x \to z$. We will reuse this for latent diffusion! / Compression, réutilisée plus tard!

What does a VAE give us?

- + Generation: $z \sim \mathcal{N}(0, I)$, then decode. / Génération.
- + A smooth latent space: interpolate between two sounds by interpolating their z's.
  - Un espace latent lisse: on peut interpoler entre deux sons.
- + Compression: $x \to z$. We will reuse this for latent diffusion! / Compression, réutilisée plus tard!
- – Blurry samples: with the MSE, the decoder outputs the average of all plausible x's.
  - Échantillons flous: le décodeur donne la moyenne des x plausibles.
- – Posterior collapse: a very powerful decoder can ignore z ($KL \to 0$).
  - Un décodeur trop puissant peut ignorer z.

VAEs and audio

- VQ-VAE (van den Oord et al., 2017): a discrete latent, using a codebook of vectors.
  - Each z is replaced by its nearest codebook vector: K-means inside a network! ("K-means is a tokenizer")
  - Chaque z est remplacé par le vecteur le plus proche du codebook: du K-means dans un réseau!

VAEs and audio

- VQ-VAE (van den Oord et al., 2017): a discrete latent, using a codebook of vectors.
  - Each z is replaced by its nearest codebook vector: K-means inside a network! ("K-means is a tokenizer")
  - Chaque z est remplacé par le vecteur le plus proche du codebook: du K-means dans un réseau!
- Neural audio codecs: SoundStream (Zeghidour et al., 2021), EnCodec (Défossez et al., 2022).
  - Encoder $\rightarrow$ residual vector quantization $\rightarrow$ decoder, trained with reconstruction + **adversarial** losses.
  - Their tokens are what audio language models generate (AudioLM, MusicGen).
  - Leurs jetons sont générés par des modèles de langage audio.

VAEs and audio

- VQ-VAE (van den Oord et al., 2017): a discrete latent, using a codebook of vectors.
  - Each z is replaced by its nearest codebook vector: K-means inside a network! ("K-means is a tokenizer")
  - Chaque z est remplacé par le vecteur le plus proche du codebook: du K-means dans un réseau!
- Neural audio codecs: SoundStream (Zeghidour et al., 2021), EnCodec (Défossez et al., 2022).
  - Encoder $\rightarrow$ residual vector quantization $\rightarrow$ decoder, trained with reconstruction + **adversarial** losses.
  - Their tokens are what audio language models generate (AudioLM, MusicGen).
  - Leurs jetons sont générés par des modèles de langage audio.
- A VAE on mel-spectrograms: the compressor inside AudioLDM (end of today). / Le compresseur d'AudioLDM.

Table of Contents

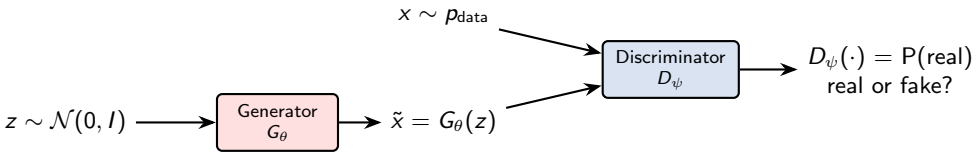
Centroid based approaches
- K-means clustering
- Gaussian Mixture Model
- More Advanced GMM Learning Methods

Spectral Clustering

Hierarchical Clustering

Deep Generative Models
- Variational Autoencoders
- Generative Adversarial Networks
- Diffusion Models
- Audio Diffusion Models

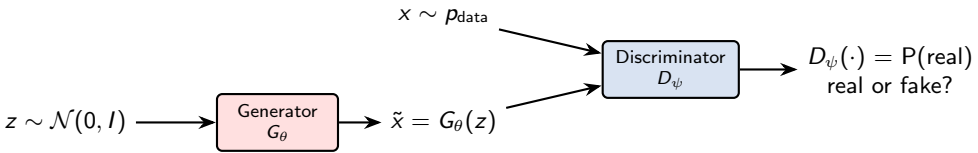
Generative Adversarial Networks (GANs)



$$\min_G \max_D \; \mathbb{E}_{x \sim p_{\text{data}}} \big[\log D(x) \big] + \mathbb{E}_{z \sim \mathcal{N}(0,I)} \big[\log(1 - D(G(z))) \big]$$

- D is a classifier (week 6!): real vs. generated. / D est un classificateur: vrai ou généré.

Generative Adversarial Networks (GANs)



$$\min_G \max_D \; \mathbb{E}_{x \sim p_{\text{data}}} \big[\log D(x) \big] + \mathbb{E}_{z \sim \mathcal{N}(0,I)} \big[\log(1 - D(G(z))) \big]$$

- D is a classifier (week 6!): real vs. generated. / D est un classificateur: vrai ou généré.
- G is trained to fool D. No encoder, no likelihood, just a game. Goodfellow et al., NeurIPS 2014
  - G apprend à tromper D. Pas d'encodeur, pas de vraisemblance, juste un jeu.

What do GANs optimize?

- For a fixed G, the best discriminator is / Pour G fixé, le meilleur discriminateur:

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

What do GANs optimize?

- For a fixed G, the best discriminator is / Pour G fixé, le meilleur discriminateur:

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

- Plug it back in: / On le remplace dans l'objectif:

$$\max_D V(G, D) = 2\, JSD\big(p_{\text{data}} \,\|\, p_G\big) - \log 4 \quad \Rightarrow \quad \text{minimized when } p_G = p_{\text{data}}$$

What do GANs optimize?

- For a fixed G, the best discriminator is / Pour G fixé, le meilleur discriminateur:

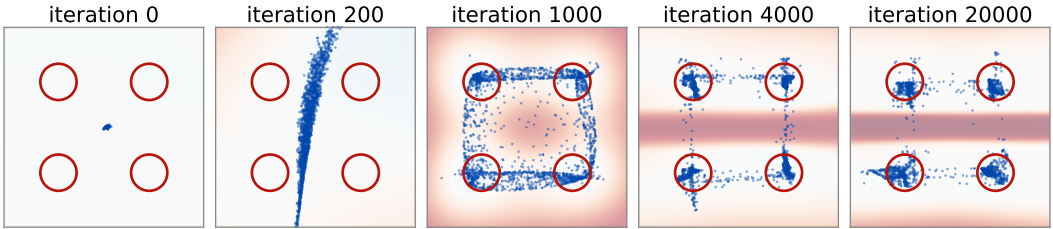
$$D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}$$

- Plug it back in: / On le remplace dans l'objectif:

$$\max_D V(G, D) = 2\, JSD\big(p_{\text{data}} \,\|\, p_G\big) - \log 4 \quad \Rightarrow \quad \text{minimized when } p_G = p_{\text{data}}$$

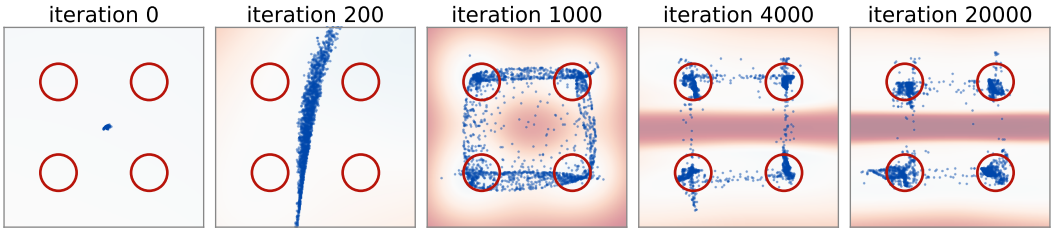
- + Sharp samples, one network call to generate. / Échantillons nets, génération rapide.
- − Unstable training (a game, not a minimization), mode collapse (G only produces a few kinds of samples).
  - Entraînement instable, effondrement des modes.
  - Fixes: Wasserstein GAN, spectral normalization, ...

Toy problem: the GAN



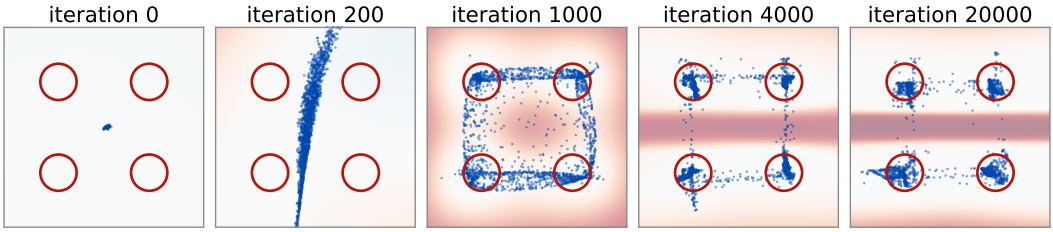
- G_θ, D_ψ: the networks of the VAE decoder and encoder. Background: $D_\psi(x)$, blue = "real", red = "fake".
  - Les réseaux du VAE. En fond: $D_\psi(x)$. G pousse ses échantillons vers le bleu.

Toy problem: the GAN



- G_θ, D_ψ: the networks of the VAE decoder and encoder. Background: $D_\psi(x)$, blue = "real", red = "fake".
  - Les réseaux du VAE. En fond: $D_\psi(x)$. G pousse ses échantillons vers le bleu.
- All 4 modes, but odd shapes (corners, streaks): G only has to fool *this* D, and D keeps changing.
  - Les 4 modes, mais des formes étranges: G doit seulement tromper *ce* D, qui change sans cesse.

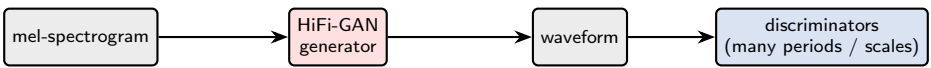
Toy problem: the GAN



- G_θ, D_ψ: the networks of the VAE decoder and encoder. Background: $D_\psi(x)$, blue = "real", red = "fake".
  - Les réseaux du VAE. En fond: $D_\psi(x)$. G pousse ses échantillons vers le bleu.
- All 4 modes, but odd shapes (corners, streaks): G only has to fool *this* D, and D keeps changing.
  - Les 4 modes, mais des formes étranges: G doit seulement tromper *ce* D, qui change sans cesse.
- Near the data, $D(x) \approx \frac{1}{2}$ (pale): it can no longer tell, as $D^* = \frac{p_{\text{data}}}{p_{\text{data}} + p_G}$ predicts.

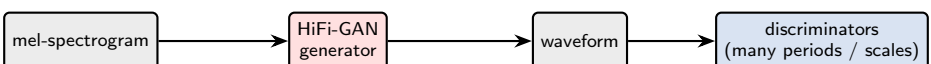
GANs in audio

- WaveGAN (Donahue et al., ICLR 2019): generating raw waveforms.



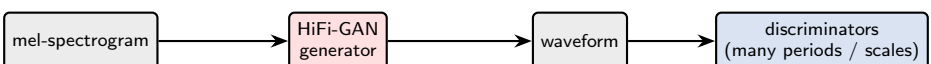
GANs in audio

- WaveGAN (Donahue et al., ICLR 2019): generating raw waveforms.
- GAN **vocoders**: mel-spectrogram $\rightarrow$ waveform, much faster than real time.
  - MelGAN (Kumar et al., NeurIPS 2019), HiFi-GAN (Kong et al., NeurIPS 2020).
  - Des vocodeurs: du mel-spectrogramme à la forme d'onde.



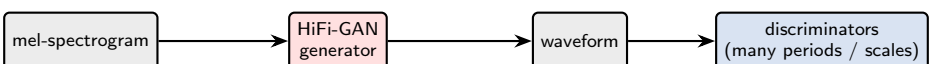
GANs in audio

- WaveGAN (Donahue et al., ICLR 2019): generating raw waveforms.
- GAN **vocoders**: mel-spectrogram $\rightarrow$ waveform, much faster than real time.
  - MelGAN (Kumar et al., NeurIPS 2019), HiFi-GAN (Kong et al., NeurIPS 2020).
  - Des vocodeurs: du mel-spectrogramme à la forme d'onde.
- Adversarial losses inside neural codecs (SoundStream, EnCodec): sharper, more natural audio.



GANs in audio

- WaveGAN (Donahue et al., ICLR 2019): generating raw waveforms.
- GAN **vocoders**: mel-spectrogram → waveform, much faster than real time.
  - MelGAN (Kumar et al., NeurIPS 2019), HiFi-GAN (Kong et al., NeurIPS 2020).
  - Des vocodeurs: du mel-spectrogramme à la forme d'onde.
- Adversarial losses inside neural codecs (SoundStream, EnCodec): sharper, more natural audio.



- Remember HiFi-GAN: it is the last block of AudioLDM. / Retenez HiFi-GAN: c'est le dernier bloc d'AudioLDM.

Table of Contents

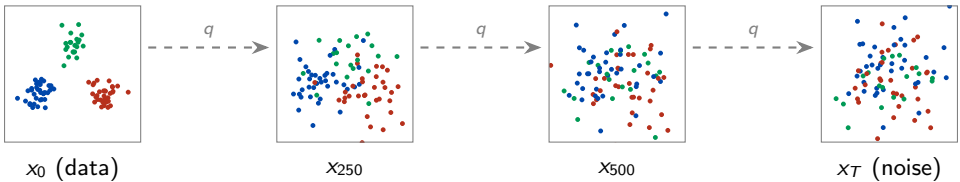
Centroid based approaches
- K-means clustering
- Gaussian Mixture Model
- More Advanced GMM Learning Methods

Spectral Clustering

Hierarchical Clustering

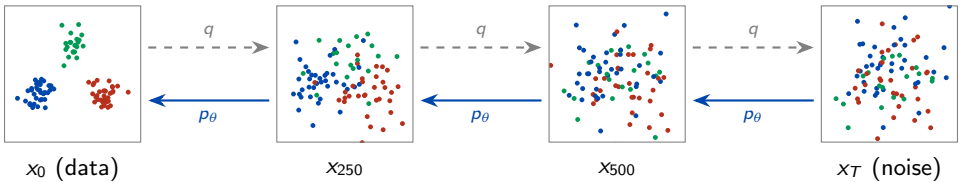
Deep Generative Models
- Variational Autoencoders
- Generative Adversarial Networks
- Diffusion Models
- Audio Diffusion Models

Diffusion models: the idea



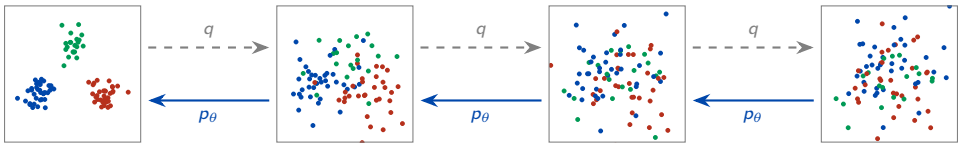
- Destroying structure is easy: add a little Gaussian noise, T times (q, fixed).
  - Détruire la structure est facile: ajouter un peu de bruit, T fois.

Diffusion models: the idea



- Destroying structure is easy: add a little Gaussian noise, T times (q, fixed).
  - Détruire la structure est facile: ajouter un peu de bruit, T fois.
- Learn to undo one small step at a time (p_θ). Then: start from pure noise and denoise T times.
  - Apprendre à défaire chaque petite étape, puis partir du bruit et débruiter T fois.

Diffusion models: the idea



- Destroying structure is easy: add a little Gaussian noise, T times (q, fixed).
  - Détruire la structure est facile: ajouter un peu de bruit, T fois.
- Learn to undo one small step at a time (p_θ). Then: start from pure noise and denoise T times.
  - Apprendre à défaire chaque petite étape, puis partir du bruit et débruiter T fois.
- Sohl-Dickstein et al., ICML 2015; Ho, Jain & Abbeel (DDPM), NeurIPS 2020; Song et al., ICLR 2021

The forward process: adding noise

- One step: / Une étape:

$$q(x_t|x_{t-1}) = \mathcal{N}\big(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\big), \quad t = 1, \ldots, T$$

  - β_t small and increasing (DDPM: from 10^{-4} to 0.02, with $T = 1000$). Nothing to learn!
  - β_t petit et croissant. Rien à apprendre!

The forward process: adding noise

- One step: / Une étape:

$$q(x_t|x_{t-1}) = \mathcal{N}\big(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\big), \quad t = 1, \ldots, T$$

  - β_t small and increasing (DDPM: from 10^{-4} to 0.02, with $T = 1000$). Nothing to learn!
  - β_t petit et croissant. Rien à apprendre!
- Why $\sqrt{1-\beta_t}$? If $\mathrm{Var}(x_{t-1}) = 1$, then $\mathrm{Var}(x_t) = (1-\beta_t) + \beta_t = 1$: the variance stays fixed.
  - La variance est préservée: le signal n'explose pas.

The forward process: adding noise

- One step: / Une étape:

$$q(x_t|x_{t-1}) = \mathcal{N}\big(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\big), \quad t = 1, \ldots, T$$

  - β_t small and increasing (DDPM: from 10^{-4} to 0.02, with $T = 1000$). Nothing to learn!
  - β_t petit et croissant. Rien à apprendre!
- Why $\sqrt{1-\beta_t}$? If $\mathrm{Var}(x_{t-1}) = 1$, then $\mathrm{Var}(x_t) = (1-\beta_t) + \beta_t = 1$: the variance stays fixed.
  - La variance est préservée: le signal n'explose pas.
- Jump to any t in one shot. With $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$: / On saute directement à t:

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \qquad \Leftrightarrow \qquad q(x_t|x_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t}\, x_0, (1-\bar{\alpha}_t) I\big)$$

Why the one-shot formula holds

$$x_t = \sqrt{\alpha_t}\, x_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_t$$

Why the one-shot formula holds

$$
\begin{aligned}
x_t &= \sqrt{\alpha_t}\, x_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_t \\
&= \sqrt{\alpha_t}\big(\sqrt{\alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_{t-1}}\, \epsilon_{t-1}\big) + \sqrt{1-\alpha_t}\, \epsilon_t
\end{aligned}
$$

Why the one-shot formula holds

$$
\begin{aligned}
x_t &= \sqrt{\alpha_t}\, x_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_t \\
&= \sqrt{\alpha_t}\big(\sqrt{\alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_{t-1}}\, \epsilon_{t-1}\big) + \sqrt{1-\alpha_t}\, \epsilon_t \\
&= \sqrt{\alpha_t \alpha_{t-1}}\, x_{t-2} + \underbrace{\sqrt{\alpha_t(1-\alpha_{t-1})}\, \epsilon_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_t}_{\text{sum of independent Gaussians}}
\end{aligned}
$$

Why the one-shot formula holds

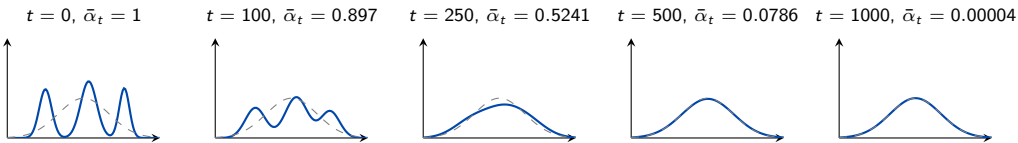
$$
\begin{aligned}
x_t &= \sqrt{\alpha_t}\, x_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_t \\
&= \sqrt{\alpha_t}\big(\sqrt{\alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_{t-1}}\, \epsilon_{t-1}\big) + \sqrt{1-\alpha_t}\, \epsilon_t \\
&= \sqrt{\alpha_t \alpha_{t-1}}\, x_{t-2} + \underbrace{\sqrt{\alpha_t(1-\alpha_{t-1})}\, \epsilon_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_t}_{\text{sum of independent Gaussians}} \\
&= \sqrt{\alpha_t \alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_t \alpha_{t-1}}\, \bar{\epsilon} \qquad \text{variance: } \alpha_t - \alpha_t \alpha_{t-1} + 1 - \alpha_t
\end{aligned}
$$

Why the one-shot formula holds

$$
\begin{aligned}
x_t &= \sqrt{\alpha_t}\, x_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_t \\
&= \sqrt{\alpha_t}\big(\sqrt{\alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_{t-1}}\, \epsilon_{t-1}\big) + \sqrt{1-\alpha_t}\, \epsilon_t \\
&= \sqrt{\alpha_t \alpha_{t-1}}\, x_{t-2} + \underbrace{\sqrt{\alpha_t(1-\alpha_{t-1})}\, \epsilon_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_t}_{\text{sum of independent Gaussians}} \\
&= \sqrt{\alpha_t \alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\, \bar{\epsilon} \qquad \text{variance: } \alpha_t - \alpha_t\alpha_{t-1} + 1 - \alpha_t \\
&= \cdots = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon
\end{aligned}
$$

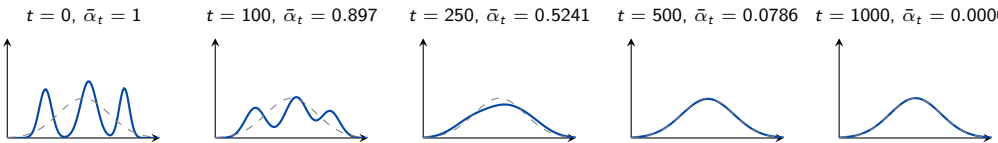
- With $T = 1000$: $\bar{\alpha}_T \approx 4 \times 10^{-5}$, so $x_T \approx \mathcal{N}(0, I)$ whatever x_0 was.
  - À la fin, x_T est du bruit, peu importe x_0.

Forward diffusion of a 1D GMM



- If x_0 is a GMM, every x_t is a GMM: component $\mathcal{N}(\mu_k, s_k^2)$ becomes $\mathcal{N}\big(\sqrt{\bar{\alpha}_t}\mu_k, \bar{\alpha}_t s_k^2 + 1 - \bar{\alpha}_t\big)$.
  - Si x_0 est un GMM, chaque x_t l'est aussi: les moyennes rétrécissent, les variances vont vers 1.

Forward diffusion of a 1D GMM



- If x_0 is a GMM, every x_t is a GMM: component $\mathcal{N}(\mu_k, s_k^2)$ becomes $\mathcal{N}\left(\sqrt{\bar{\alpha}_t}\mu_k, \bar{\alpha}_t s_k^2 + 1 - \bar{\alpha}_t\right)$.
  - Si x_0 est un GMM, chaque x_t l'est aussi: les moyennes rétrécissent, les variances vont vers 1.
- The clusters melt into $\mathcal{N}(0, 1)$ (dashed). Linear schedule of DDPM, $T = 1000$.
  - Les clusters fondent dans $\mathcal{N}(0, 1)$ (pointillé).

The reverse process: removing noise

- Start from noise and go backwards with a learned Gaussian step: / On part du bruit et on recule:

$$p(x_T) = \mathcal{N}(0, I), \qquad p_\theta(x_{t-1}|x_t) = \mathcal{N}\big(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I\big)$$

The reverse process: removing noise

- Start from noise and go backwards with a learned Gaussian step: / On part du bruit et on recule:

$$p(x_T) = \mathcal{N}(0, I), \qquad p_\theta(x_{t-1}|x_t) = \mathcal{N}\big(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I\big)$$

- One network $\mu_\theta(x_t, t)$ for all steps: t is an input. / Un seul réseau, t est une entrée.

The reverse process: removing noise

- Start from noise and go backwards with a learned Gaussian step: / On part du bruit et on recule:

$$p(x_T) = \mathcal{N}(0, I), \qquad p_\theta(x_{t-1}|x_t) = \mathcal{N}\big(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I\big)$$

- One network $\mu_\theta(x_t, t)$ for all steps: t is an input. / Un seul réseau, t est une entrée.
- Why is a Gaussian step good enough? When β_t is small, the reverse of a small Gaussian step is also close to Gaussian.
  - Quand β_t est petit, l'inverse d'une petite étape Gaussienne est presque Gaussien.

The reverse process: removing noise

- Start from noise and go backwards with a learned Gaussian step: / On part du bruit et on recule:

$$p(x_T) = \mathcal{N}(0, I), \qquad p_\theta(x_{t-1}|x_t) = \mathcal{N}\big(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I\big)$$

- One network $\mu_\theta(x_t, t)$ for all steps: t is an input. / Un seul réseau, t est une entrée.
- Why is a Gaussian step good enough? When β_t is small, the reverse of a small Gaussian step is also close to Gaussian.
  - Quand β_t est petit, l'inverse d'une petite étape Gaussienne est presque Gaussien.
- The model: / Le modèle:

$$p_\theta(x_0) = \int p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)\, dx_{1:T}$$

  - A latent variable model, with latents $x_{1:T}$! / Un modèle à variables latentes!

A diffusion model is a (very deep) VAE

	GMM	**VAE**	**Diffusion**
Latent	h (discrete)	z (small)	$x_1, \ldots, x_T$ (same size as x)
Approx. posterior	exact (E-step)	encoder $q_\phi(z\|x)$, learned	forward process $q(x_{1:T}\|x_0)$, **fixed**
Decoder	$\mathcal{N}(\mu_h, \Sigma_h)$	$g_\theta(z)$	$p_\theta(x_{t-1}\|x_t)$, T times
Objective	VLB	ELBO	ELBO

A diffusion model is a (very deep) VAE

	GMM	**VAE**	**Diffusion**
Latent	h (discrete)	z (small)	$x_1, \ldots, x_T$ (same size as x)
Approx. posterior	exact (E-step)	encoder $q_\phi(z\|x)$, learned	forward process $q(x_{1:T}\|x_0)$, **fixed**
Decoder	$\mathcal{N}(\mu_h, \Sigma_h)$	$g_\theta(z)$	$p_\theta(x_{t-1}\|x_t)$, T times
Objective	VLB	ELBO	ELBO

- The same bound, for the third time today: / La même borne, pour la troisième fois aujourd'hui:

$$\log p_\theta(x_0) \geq \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] = \text{ELBO}$$

A diffusion model is a (very deep) VAE

	GMM	VAE	Diffusion
Latent	h (discrete)	z (small)	$x_1, \dots, x_T$ (same size as x)
Approx. posterior	exact (E-step)	encoder $q_\phi(z\|x)$, learned	forward process $q(x_{1:T}\|x_0)$, **fixed**
Decoder	$\mathcal{N}(\mu_h, \Sigma_h)$	$g_\theta(z)$	$p_\theta(x_{t-1}\|x_t)$, T times
Objective	VLB	ELBO	ELBO

- The same bound, for the third time today: / La même borne, pour la troisième fois aujourd'hui:

$$\log p_\theta(x_0) \geq \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] = \text{ELBO}$$

- The encoder is fixed, so only the decoder is learned: no posterior collapse.
  - L'encodeur est fixe, seul le décodeur est appris.

Breaking down the ELBO

$$-\text{ELBO} = \mathbb{E}_q\Big[\underbrace{KL\big(q(x_T|x_0)\,\|\,p(x_T)\big)}_{L_T\text{: no parameters}} + \sum_{t=2}^{T}\underbrace{KL\big(q(x_{t-1}|x_t,x_0)\,\|\,p_\theta(x_{t-1}|x_t)\big)}_{L_{t-1}} \underbrace{-\log p_\theta(x_0|x_1)}_{L_0}\Big]$$

Breaking down the ELBO

$$-\text{ELBO} = \mathbb{E}_q\Big[\underbrace{KL\big(q(x_T|x_0)\,\|\,p(x_T)\big)}_{L_T\text{: no parameters}} + \sum_{t=2}^{T}\underbrace{KL\big(q(x_{t-1}|x_t,x_0)\,\|\,p_\theta(x_{t-1}|x_t)\big)}_{L_{t-1}} \underbrace{-\log p_\theta(x_0|x_1)}_{L_0}\Big]$$

- How? Bayes' rule on each forward step (the Markov property lets us add x_0): / Bayes sur chaque étape:

$$q(x_t|x_{t-1}) = q(x_t|x_{t-1},x_0) = \frac{q(x_{t-1}|x_t,x_0)\,q(x_t|x_0)}{q(x_{t-1}|x_0)}$$

  and the $q(x_t|x_0)$ terms cancel in a telescoping product. / Les termes $q(x_t|x_0)$ se simplifient.

Breaking down the ELBO

$$-\text{ELBO} = \mathbb{E}_q\Big[\underbrace{KL\big(q(x_T|x_0)\,\|\,p(x_T)\big)}_{L_T\text{: no parameters}} + \sum_{t=2}^{T}\underbrace{KL\big(q(x_{t-1}|x_t,x_0)\,\|\,p_\theta(x_{t-1}|x_t)\big)}_{L_{t-1}} \underbrace{-\log p_\theta(x_0|x_1)}_{L_0}\Big]$$

- How? Bayes' rule on each forward step (the Markov property lets us add x_0): / Bayes sur chaque étape:

$$q(x_t|x_{t-1}) = q(x_t|x_{t-1},x_0) = \frac{q(x_{t-1}|x_t,x_0)\,q(x_t|x_0)}{q(x_{t-1}|x_0)}$$

  and the $q(x_t|x_0)$ terms cancel in a telescoping product. / Les termes $q(x_t|x_0)$ se simplifient.
- Each L_{t-1}: the learned reverse step should match the forward process run backwards, **given** x_0.
  - L'étape inverse apprise doit ressembler au processus avant renversé, sachant x_0.

The forward process, run backwards

- Given x_0, all three terms are Gaussian, so the backward step is Gaussian too: / Tout est Gaussien:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}\big(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I\big)$$

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\, x_0 + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\, x_t, \qquad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\,\beta_t$$

The forward process, run backwards

- Given x_0, all three terms are Gaussian, so the backward step is Gaussian too: / Tout est Gaussien:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}\big(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I\big)$$

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\, x_0 + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\, x_t, \qquad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\,\beta_t$$

- Same algebra as $\widehat{q}_\theta(\mu_k)$ in Variant 3: multiply Gaussians, complete the square.
  - Même algèbre que pour Variant 3: on complète le carré.

The forward process, run backwards

- Given x_0, all three terms are Gaussian, so the backward step is Gaussian too: / Tout est Gaussien:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}\big(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I\big)$$

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\, x_0 + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\, x_t, \qquad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\,\beta_t$$

- Same algebra as $\widehat{q}_\theta(\mu_k)$ in Variant 3: multiply Gaussians, complete the square.
  - Même algèbre que pour Variant 3: on complète le carré.
- $\tilde{\mu}_t$ is a weighted average of x_0 (where we go) and x_t (where we are).
  - Une moyenne pondérée entre x_0 (où on va) et x_t (où on est).

Predict the noise

- Two Gaussians with fixed variance: $L_{t-1} = \frac{1}{2\sigma_t^2} \left\| \tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t) \right\|^2 + \text{const}$

Predict the noise

- Two Gaussians with fixed variance: $L_{t-1} = \frac{1}{2\sigma_t^2} \left\| \tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t) \right\|^2 + \text{const}$
- Use $x_0 = \left(x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon\right)/\sqrt{\bar{\alpha}_t}$ to remove x_0: / On élimine x_0:
  $\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon\right)$

Predict the noise

- Two Gaussians with fixed variance: $L_{t-1} = \frac{1}{2\sigma_t^2} \left\| \tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t) \right\|^2 + \text{const}$
- Use $x_0 = \left(x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon\right)/\sqrt{\bar{\alpha}_t}$ to remove x_0: / On élimine x_0:
  $\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon\right)$
- x_t is the input, only ϵ is unknown. So the network predicts the noise: / Le réseau prédit le bruit: $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(x_t, t)\right)$

Predict the noise

- Two Gaussians with fixed variance: $L_{t-1} = \frac{1}{2\sigma_t^2} \left\| \tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t) \right\|^2 + \text{const}$
- Use $x_0 = \left(x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon\right)/\sqrt{\bar{\alpha}_t}$ to remove x_0: / On élimine x_0:
  $\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon\right)$
- x_t is the input, only ϵ is unknown. So the network predicts the noise: / Le réseau prédit le bruit: $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(x_t, t)\right)$
- The loss becomes: / La perte devient:

$$L_{t-1} = \frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)} \left\| \epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\,x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon,\ t\right) \right\|^2$$

Predict the noise

- Two Gaussians with fixed variance: $L_{t-1} = \frac{1}{2\sigma_t^2} \left\| \tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t) \right\|^2 + \text{const}$
- Use $x_0 = \left(x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon\right)/\sqrt{\bar{\alpha}_t}$ to remove x_0: / On élimine x_0:
  $\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon \right)$
- x_t is the input, only ϵ is unknown. So the network predicts the noise: / Le réseau prédit le bruit: $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(x_t, t) \right)$
- The loss becomes: / La perte devient:

$$L_{t-1} = \frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)} \left\| \epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\,x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon,\ t\right) \right\|^2$$

- Ho et al. drop the weight: $L_{\text{simple}} = \mathbb{E}_{t, x_0, \epsilon} \left\| \epsilon - \epsilon_\theta(x_t, t) \right\|^2$. A **denoiser**! / Un débruiteur!

Training and sampling (DDPM)

Training / Entraînement

repeat
$x_0 \sim$ data, $\quad t \sim \text{Uniform}\{1, \dots, T\}$
$\epsilon \sim \mathcal{N}(0, I)$
$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon$
Gradient step on $\nabla_\theta \left\| \epsilon - \epsilon_\theta(x_t, t) \right\|^2$
until converged

- Training: a random t, add noise, predict the noise. Cheap: one network call per example.
  - Entraînement: un t au hasard, ajouter du bruit, prédire le bruit.

Training and sampling (DDPM)

Training / Entraînement

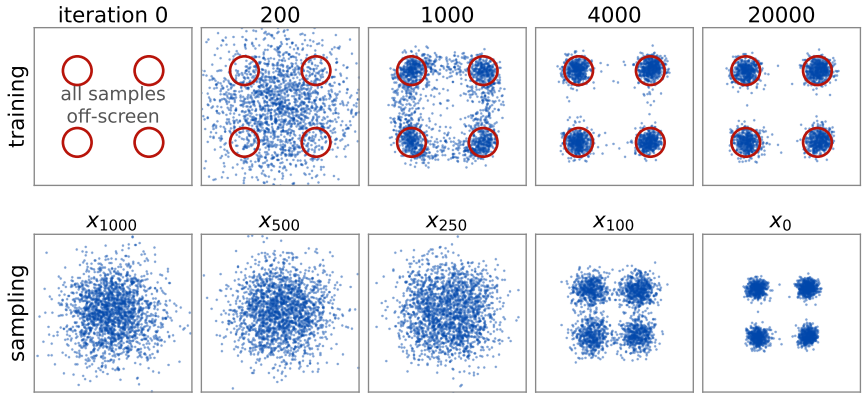
repeat
  $x_0 \sim$ data, $t \sim \text{Uniform}\{1, \dots, T\}$
  $\epsilon \sim \mathcal{N}(0, I)$
  $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon$
  Gradient step on $\nabla_\theta \left\| \epsilon - \epsilon_\theta(x_t, t) \right\|^2$
until converged

Sampling / Génération

$x_T \sim \mathcal{N}(0, I)$
for $t = T, \dots, 1$ **do**
  $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z = 0$
  $x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t z$
end for
return x_0

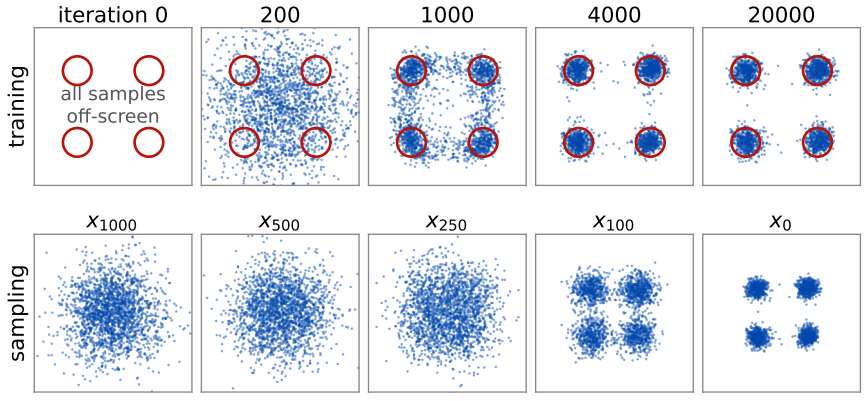
- Training: a random t, add noise, predict the noise. Cheap: one network call per example.
  - Entraînement: un t au hasard, ajouter du bruit, prédire le bruit.
- Sampling: $T = 1000$ network calls, each removes a little noise. $\sigma_t^2 = \beta_t$ works well.
  - Génération: T appels du réseau. C'est lent!

Toy problem: the diffusion model



- Top: samples during training (1000 denoising steps each). Bottom: one sampling run, from noise to data.
  - En haut: pendant l'entraînement. En bas: une génération, du bruit aux données.

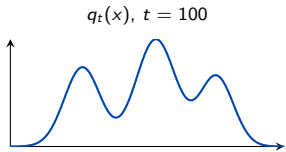
Toy problem: the diffusion model



- Top: samples during training (1000 denoising steps each). Bottom: one sampling run, from noise to data.
  - En haut: pendant l'entraînement. En bas: une génération, du bruit aux données.
- ϵ_θ: the same MLP, input $[x_t, \text{emb}(t)]$. The two algorithms of the previous slide, $T = 1000$.

Another view: the network learns the score

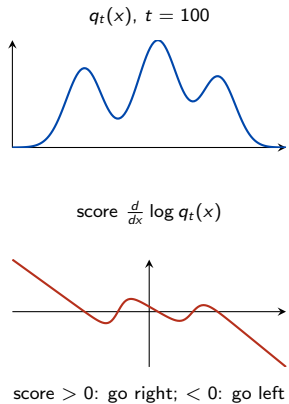
- Score: $\nabla_x \log q_t(x)$, points towards higher density. / Le score pointe vers les régions denses.



Another view: the network learns the score

- Score: $\nabla_x \log q_t(x)$, points towards higher density. / Le score pointe vers les régions denses.
- With $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon$ (Tweedie):

$$\nabla_{x_t} \log q_t(x_t) = -\frac{\mathbb{E}[\epsilon \mid x_t]}{\sqrt{1-\bar{\alpha}_t}}$$

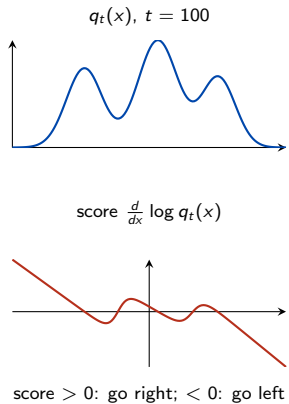


Another view: the network learns the score

- Score: $\nabla_x \log q_t(x)$, points towards higher density. / Le score pointe vers les régions denses.
- With $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon$ (Tweedie):

$$\nabla_{x_t} \log q_t(x_t) = -\frac{\mathbb{E}[\epsilon \mid x_t]}{\sqrt{1-\bar{\alpha}_t}}$$

- So $\epsilon_\theta(x_t, t) \approx -\sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log q_t(x_t)$: predicting the noise = estimating the score.
  - Prédire le bruit = estimer le score (Vincent, 2011; Song & Ermon, 2019).

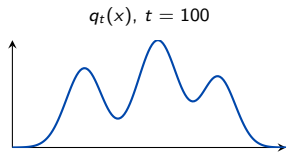


Another view: the network learns the score

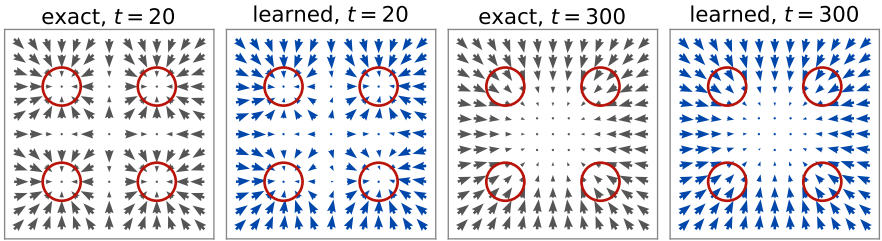
- Score: $\nabla_x \log q_t(x)$, points towards higher density. / Le score pointe vers les régions denses.
- With $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon$ (Tweedie):

$$\nabla_{x_t} \log q_t(x_t) = -\frac{\mathbb{E}[\epsilon \mid x_t]}{\sqrt{1-\bar{\alpha}_t}}$$

- So $\epsilon_\theta(x_t, t) \approx -\sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log q_t(x_t)$: predicting the noise = estimating the score.
  - Prédire le bruit = estimer le score (Vincent, 2011; Song & Ermon, 2019).
- Sampling $\approx$ follow the score, plus a bit of noise (Langevin): $x \leftarrow x + \eta \nabla_x \log p(x) + \sqrt{2\eta}\, z$



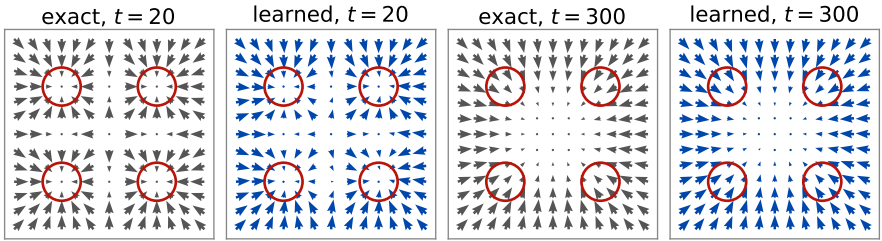
Toy problem: the learned score



- Our x_t is a GMM, so its score is known exactly. With $v_t = \bar{\alpha}_t\, 0.2^2 + 1 - \bar{\alpha}_t$:

$$\nabla_x \log q_t(x) = \sum_k \underbrace{q_t(k \mid x)}_{\text{responsibility (E-step!)}} \frac{\sqrt{\bar{\alpha}_t}\,\mu_k - x}{v_t}$$

Toy problem: the learned score

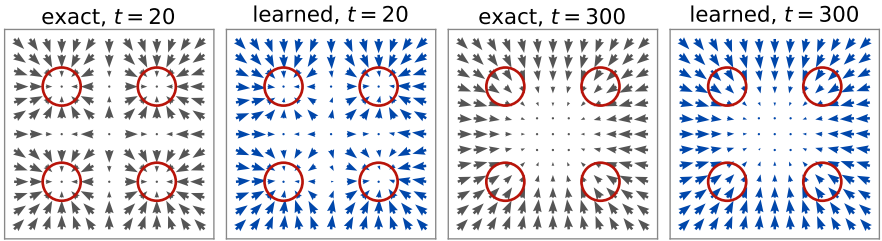


- Our x_t is a GMM, so its score is known exactly. With $v_t = \bar{\alpha}_t\, 0.2^2 + 1 - \bar{\alpha}_t$:

$$\nabla_x \log q_t(x) = \sum_k \underbrace{q_t(k \mid x)}_{\text{responsibility (E-step!)}} \frac{\sqrt{\bar{\alpha}_t}\,\mu_k - x}{v_t}$$

- Learned: $-\epsilon_\theta(x, t)/\sqrt{1 - \bar{\alpha}_t}$. The network has never seen $q_t(x)$, only noisy samples. / Le réseau n'a vu que des échantillons bruités.

Toy problem: the learned score

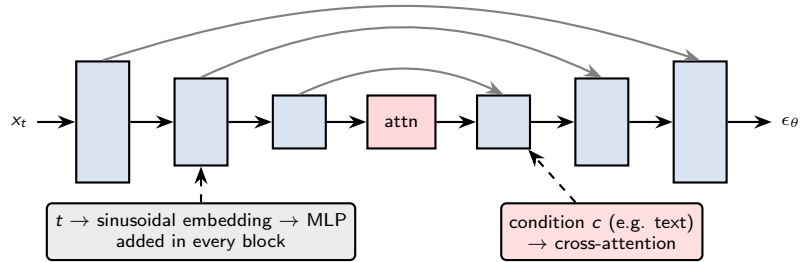


- Our x_t is a GMM, so its score is known exactly. With $v_t = \bar{\alpha}_t\, 0.2^2 + 1 - \bar{\alpha}_t$:

$$\nabla_x \log q_t(x) = \sum_k \underbrace{q_t(k \mid x)}_{\text{responsibility (E-step!)}} \frac{\sqrt{\bar{\alpha}_t}\,\mu_k - x}{v_t}$$

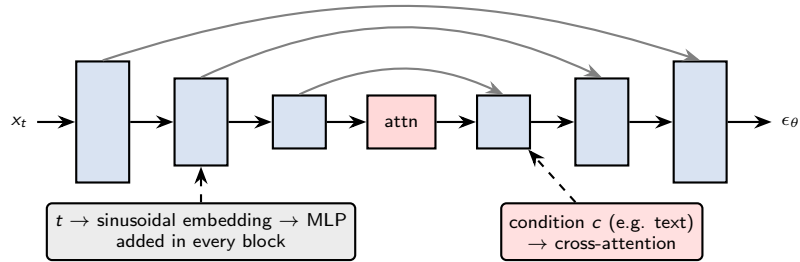
- Learned: $-\epsilon_\theta(x, t)/\sqrt{1 - \bar{\alpha}_t}$. The network has never seen $q_t(x)$, only noisy samples. / Le réseau n'a vu que des échantillons bruités.
- Relative error: 5% at $t = 300$, 22% at $t = 20$. Little noise: almost no x_t between the modes, so little to learn from there.
  - Peu de bruit: presque aucun x_t entre les modes, donc peu d'exemples pour apprendre.

The denoising network $\epsilon_\theta(x_t, t)$



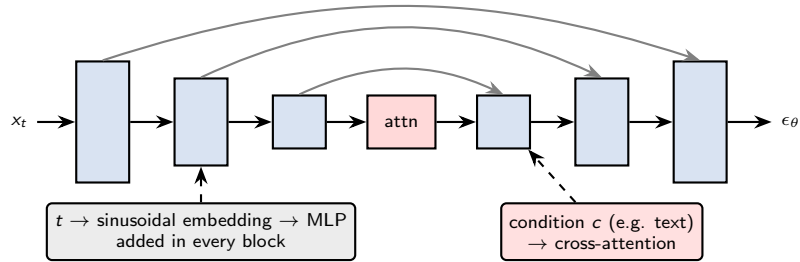
- A U-Net: a conv encoder-decoder with skip connections (gray), like week 7's denoiser. / Comme le débruiteur de la semaine 7.

The denoising network $\epsilon_\theta(x_t, t)$



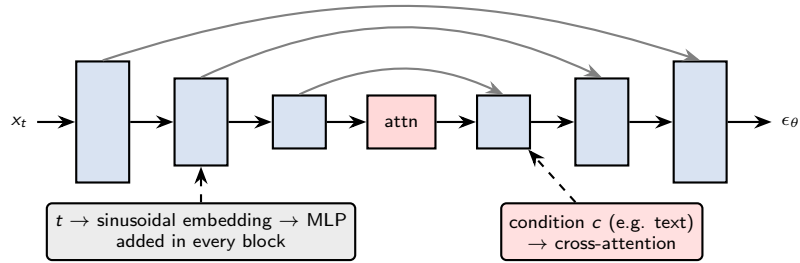
- A U-Net: a conv encoder-decoder with skip connections (gray), like week 7's denoiser. / Comme le débruiteur de la semaine 7.
- Time t: sinusoidal embedding, like positions in a transformer. / Comme les positions dans un transformeur.

The denoising network $\epsilon_\theta(x_t, t)$



- A U-Net: a conv encoder-decoder with skip connections (gray), like week 7's denoiser. / Comme le débruiteur de la semaine 7.
- Time t: sinusoidal embedding, like positions in a transformer. / Comme les positions dans un transformeur.
- A condition c enters through cross-attention: queries from the features, keys and values from c. / Attention croisée.

The denoising network $\epsilon_\theta(x_t, t)$



- A U-Net: a conv encoder-decoder with skip connections (gray), like week 7's denoiser. / Comme le débruiteur de la semaine 7.
- Time t: sinusoidal embedding, like positions in a transformer. / Comme les positions dans un transformeur.
- A condition c enters through cross-attention: queries from the features, keys and values from c. / Attention croisée.
- Transformers can also replace the U-Net (DiT, Peebles & Xie, 2023). / Un transformeur peut remplacer le U-Net.

Faster sampling

- DDPM: $T = 1000$ network calls per sample. VAEs and GANs need one. / Trop lent!

Faster sampling

- DDPM: $T = 1000$ network calls per sample. VAEs and GANs need one. / Trop lent!
- DDIM (Song, Meng & Ermon, ICLR 2021): same trained ϵ_θ, deterministic, skips steps (e.g. 50):

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}, \qquad x_{t'} = \sqrt{\bar{\alpha}_{t'}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t'}}\,\epsilon_\theta(x_t, t), \quad t' < t$$

  - Predict the clean $\hat{x}_0$, then re-noise it to a lower noise level. / Prédire $\hat{x}_0$, puis le re-bruiter moins.

Faster sampling

- DDPM: $T = 1000$ network calls per sample. VAEs and GANs need one. / Trop lent!
- DDIM (Song, Meng & Ermon, ICLR 2021): same trained ϵ_θ, deterministic, skips steps (e.g. 50):

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}, \qquad x_{t'} = \sqrt{\bar{\alpha}_{t'}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t'}}\,\epsilon_\theta(x_t, t), \quad t' < t$$

  - Predict the clean $\hat{x}_0$, then re-noise it to a lower noise level. / Prédire $\hat{x}_0$, puis le re-bruiter moins.
- Continuous time: sampling is solving an ODE / SDE (Song et al., ICLR 2021) $\rightarrow$ better solvers (DPM-Solver: 10–20 steps).
  - En temps continu: résoudre une équation différentielle.

Faster sampling

- DDPM: $T = 1000$ network calls per sample. VAEs and GANs need one. / Trop lent!
- DDIM (Song, Meng & Ermon, ICLR 2021): same trained ϵ_θ, deterministic, skips steps (e.g. 50):

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}, \qquad x_{t'} = \sqrt{\bar{\alpha}_{t'}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t'}}\,\epsilon_\theta(x_t, t), \quad t' < t$$

  - Predict the clean $\hat{x}_0$, then re-noise it to a lower noise level. / Prédire $\hat{x}_0$, puis le re-bruiter moins.
- Continuous time: sampling is solving an ODE / SDE (Song et al., ICLR 2021) $\rightarrow$ better solvers (DPM-Solver: 10–20 steps).
  - En temps continu: résoudre une équation différentielle.
- Distillation: consistency models, 1 to a few steps. / Distillation: une ou quelques étapes.

Conditioning: classifier-free guidance

- We want $p(x|c)$, with c = text, class, ...: use $\epsilon_\theta(x_t, t, c)$. / On veut générer sachant c.

Conditioning: classifier-free guidance

- We want $p(x|c)$, with c = text, class, ...: use $\epsilon_\theta(x_t, t, c)$. / On veut générer sachant c.
- Training: replace c by $\varnothing$ ("no condition") about 10% of the time. The same network learns $p(x)$ and $p(x|c)$.
  - Entraînement: on enlève c 10% du temps.

Conditioning: classifier-free guidance

- We want $p(x|c)$, with c = text, class, ...: use $\epsilon_\theta(x_t, t, c)$. / On veut générer sachant c.
- Training: replace c by $\varnothing$ ("no condition") about 10% of the time. The same network learns $p(x)$ and $p(x|c)$.
  - Entraînement: on enlève c 10% du temps.
- Sampling: push away from the unconditional prediction, with a guidance scale w: / Génération:

$$\tilde{\epsilon} = \epsilon_\theta(x_t, t, \varnothing) + w\big(\epsilon_\theta(x_t, t, c) - \epsilon_\theta(x_t, t, \varnothing)\big)$$

  - $w = 1$: plain conditional model. $w > 1$: follows c more closely, less diversity.
  - $w > 1$: plus fidèle à c, moins de diversité.

Conditioning: classifier-free guidance

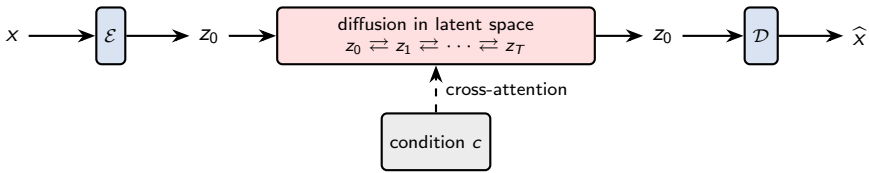
- We want $p(x|c)$, with c = text, class, ...: use $\epsilon_\theta(x_t, t, c)$. / On veut générer sachant c.
- Training: replace c by $\varnothing$ ("no condition") about 10% of the time. The same network learns $p(x)$ and $p(x|c)$.
  - Entraînement: on enlève c 10% du temps.
- Sampling: push away from the unconditional prediction, with a guidance scale w: / Génération:

$$\tilde{\epsilon} = \epsilon_\theta(x_t, t, \varnothing) + w\big(\epsilon_\theta(x_t, t, c) - \epsilon_\theta(x_t, t, \varnothing)\big)$$

  - $w = 1$: plain conditional model. $w > 1$: follows c more closely, less diversity.
  - $w > 1$: plus fidèle à c, moins de diversité.
- Why? Score view: $\nabla \log p(x|c) = \nabla \log p(x) + \nabla \log p(c|x)$; guidance multiplies the "implicit classifier" term $\nabla \log p(c|x)$ by w. Ho & Salimans, 2021

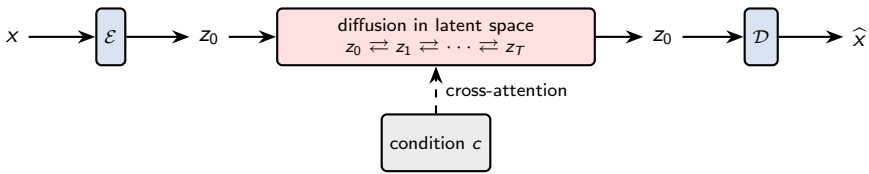
Latent diffusion

- Diffusion on raw data is expensive: every step runs the network on the full x (10 s of audio at 16 kHz = 160 000 samples).
  - La diffusion sur les données brutes coûte cher.



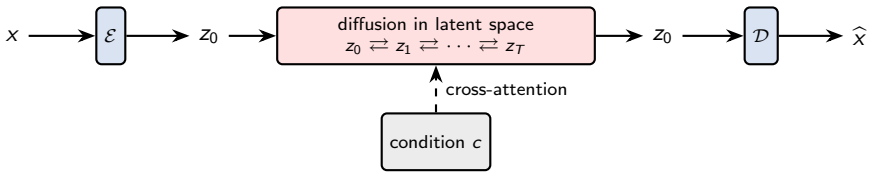
Latent diffusion

- Diffusion on raw data is expensive: every step runs the network on the full x (10 s of audio at 16 kHz = 160 000 samples).
  - La diffusion sur les données brutes coûte cher.
- Latent diffusion (Rombach et al., CVPR 2022, "Stable Diffusion"): / Diffusion latente:
  - Train an autoencoder $z = \mathcal{E}(x)$, $x \approx \mathcal{D}(z)$: KL-regularized like a VAE, + adversarial loss for sharpness.
  - Run the diffusion model on z instead of x. Generate z_0, then decode $\mathcal{D}(z_0)$.



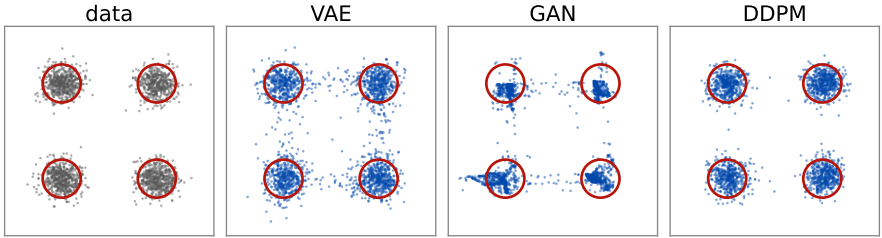
Latent diffusion

- Diffusion on raw data is expensive: every step runs the network on the full x (10 s of audio at 16 kHz = 160 000 samples).
  - La diffusion sur les données brutes coûte cher.
- Latent diffusion (Rombach et al., CVPR 2022, "Stable Diffusion"): / Diffusion latente:
  - Train an autoencoder $z = \mathcal{E}(x)$, $x \approx \mathcal{D}(z)$: KL-regularized like a VAE, + adversarial loss for sharpness.
  - Run the diffusion model on z instead of x. Generate z_0, then decode $\mathcal{D}(z_0)$.



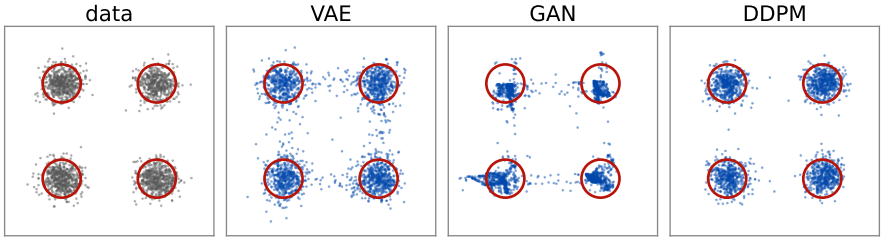
- Everything from today in one model: VAE (compression) + diffusion (generation) + GAN (sharp decoder).
  - Tout ce qu'on a vu aujourd'hui dans un seul modèle!

Toy problem: same network, three objectives



	VAE	GAN	Diffusion
Objective	$-$ELBO	minimax game	$\|\epsilon - \epsilon_\theta(x_t, t)\|^2$
Parameters (all `mlp`)	34k + 34k	34k + 34k	36k
Network calls per sample	1	1	1000
Samples between the modes (data: 1.4%)	7.8%	3.8%	1.4%
What goes wrong	bridges between modes	distorted shapes	slow sampling

Toy problem: same network, three objectives



	VAE	GAN	Diffusion
Objective	$-$ELBO	minimax game	$\|\epsilon - \epsilon_\theta(x_t, t)\|^2$
Parameters (all `mlp`)	34k + 34k	34k + 34k	36k
Network calls per sample	1	1	1000
Samples between the modes (data: 1.4%)	7.8%	3.8%	1.4%
What goes wrong	bridges between modes	distorted shapes	slow sampling

- Same network, same data: the objective alone decides what is learned. / Seul l'objectif change le résultat.

From toy to real: faces (CelebA-HQ, 64×64)

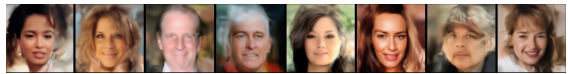
VAE ($\beta = 1$)
FID 67.2



VAE ($\beta = 4$)
FID 49.5



VAE ($\beta = 16$)



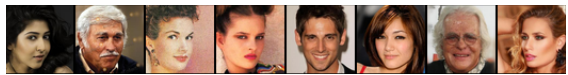
GAN (R1, $\gamma = 1$)
FID 3.70



Diffusion
FID 2.85



Real data
FID 1.65 (half vs. half)



- Not cherry-picked: the first 8 samples of each. / Pas choisis à la main.

From toy to real: faces (CelebA-HQ, 64×64)

VAE ($\beta = 1$)
FID 67.2

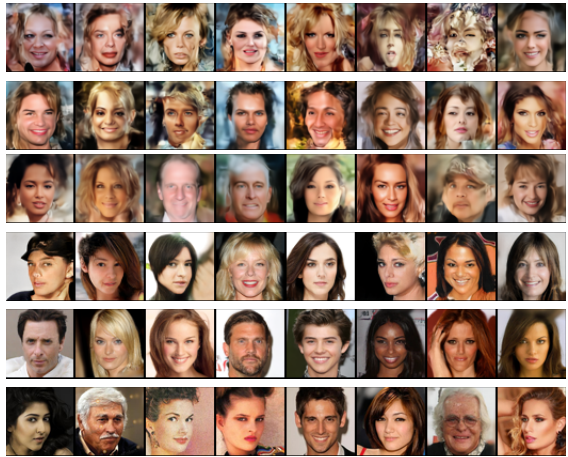
VAE ($\beta = 4$)
FID 49.5

VAE ($\beta = 16$)

GAN (R1, $\gamma = 1$)
FID 3.70

Diffusion
FID 2.85

Real data
FID 1.65 (half vs. half)



- Not cherry-picked: the first 8 samples of each. / Pas choisis à la main.
- VAE: blurrier but cleaner as β grows. GAN and diffusion: almost like the real ones.
  - VAE: plus flou quand β grandit. GAN et diffusion: presque comme les vraies.

Faces: the numbers

Model	FID-50k ↓	Precision ↑	Recall ↑	Coverage ↑	Passes / sample	ms / image
Diffusion	**2.85**	**0.76**	**0.56**	**0.97**	79	72
GAN (R1, $\gamma = 1$)	3.70	0.72	0.39	0.95	1	0.9
GAN (R1, $\gamma = 0.1$)	5.75	0.73	0.20	0.91	1	0.9
GAN (R1, $\gamma = 5$)	5.96	0.71	0.20	0.90	1	0.9
VAE ($\beta = 1$)	67.2	0.46	0.05	0.18	1	0.9
VAE ($\beta = 4$)	49.5	0.53	0.08	0.27	1	0.9
Real data, half vs. half	1.65	0.71	0.70	0.97		

- FID (Heusel et al., 2017): fit a Gaussian to the Inception features of real and of generated images, measure the distance between the two Gaussians. Lower is better.
  - Distance entre deux Gaussiennes ajustées aux caractéristiques. Même idée que la FAD pour l'audio.

Faces: the numbers

Model	FID-50k ↓	Precision ↑	Recall ↑	Coverage ↑	Passes / sample	ms / image
Diffusion	**2.85**	**0.76**	**0.56**	**0.97**	79	72
GAN (R1, $\gamma = 1$)	3.70	0.72	0.39	0.95	1	0.9
GAN (R1, $\gamma = 0.1$)	5.75	0.73	0.20	0.91	1	0.9
GAN (R1, $\gamma = 5$)	5.96	0.71	0.20	0.90	1	0.9
VAE ($\beta = 1$)	67.2	0.46	0.05	0.18	1	0.9
VAE ($\beta = 4$)	49.5	0.53	0.08	0.27	1	0.9
Real data, half vs. half	1.65	0.71	0.70	0.97		

- FID (Heusel et al., 2017): fit a Gaussian to the Inception features of real and of generated images, measure the distance between the two Gaussians. Lower is better.
  - Distance entre deux Gaussiennes ajustées aux caractéristiques. Même idée que la FAD pour l'audio.
- Precision: are the samples realistic? Recall: is all of the data covered? (Kynkäänniemi et al., 2019; coverage: Naeem et al., 2020)
  - Précision: réalisme. Rappel: diversité.

Faces: the numbers

Model	FID-50k ↓	Precision ↑	Recall ↑	Coverage ↑	Passes / sample	ms / image
Diffusion	**2.85**	**0.76**	**0.56**	**0.97**	79	72
GAN (R1, $\gamma = 1$)	3.70	0.72	0.39	0.95	1	0.9
GAN (R1, $\gamma = 0.1$)	5.75	0.73	0.20	0.91	1	0.9
GAN (R1, $\gamma = 5$)	5.96	0.71	0.20	0.90	1	0.9
VAE ($\beta = 1$)	67.2	0.46	0.05	0.18	1	0.9
VAE ($\beta = 4$)	49.5	0.53	0.08	0.27	1	0.9
Real data, half vs. half	1.65	0.71	0.70	0.97		

- FID (Heusel et al., 2017): fit a Gaussian to the Inception features of real and of generated images, measure the distance between the two Gaussians. Lower is better.
  - Distance entre deux Gaussiennes ajustées aux caractéristiques. Même idée que la FAD pour l'audio.
- Precision: are the samples realistic? Recall: is all of the data covered? (Kynkäänniemi et al., 2019; coverage: Naeem et al., 2020)
  - Précision: réalisme. Rappel: diversité.
- Diffusion is best everywhere, but 80× slower. The GAN is less diverse (recall) and depends on R1, a gradient penalty on D with weight γ.
  - La diffusion gagne partout, mais 80 fois plus lente. Le GAN dépend de sa régularisation.

Faces: why is the VAE behind?

	FID of samples $\downarrow$ $z \sim \mathcal{N}(0, I)$, decode	**FID of reconstructions** $\downarrow$ real x, encode, decode
VAE ($\beta = 1$)	67.2	13.0
VAE ($\beta = 4$)	49.5	13.6

- Reconstructions are much better than samples: the decoder can draw faces, when it gets a z from the encoder.
  - Le décodeur sait dessiner des visages, si le z vient de l'encodeur.

Faces: why is the VAE behind?

	FID of samples ↓ $z \sim \mathcal{N}(0, I)$, decode	**FID of reconstructions ↓** real x, encode, decode
VAE ($\beta = 1$)	67.2	13.0
VAE ($\beta = 4$)	49.5	13.6

- Reconstructions are much better than samples: the decoder can draw faces, when it gets a z from the encoder.
  - Le décodeur sait dessiner des visages, si le z vient de l'encodeur.
- Samples: many $z \sim \mathcal{N}(0, I)$ fall between the regions the encoder uses, and decode to mixed faces (the toy bridges again).
  - Les z tirés du prior tombent entre les régions de l'encodeur: les ponts du problème jouet.

Faces: why is the VAE behind?

	FID of samples $\downarrow$ $z \sim \mathcal{N}(0, I)$, decode	**FID of reconstructions** $\downarrow$ real x, encode, decode
VAE ($\beta = 1$)	67.2	13.0
VAE ($\beta = 4$)	49.5	13.6

- Reconstructions are much better than samples: the decoder can draw faces, when it gets a z from the encoder.
  - Le décodeur sait dessiner des visages, si le z vient de l'encodeur.
- Samples: many $z \sim \mathcal{N}(0, I)$ fall between the regions the encoder uses, and decode to mixed faces (the toy bridges again).
  - Les z tirés du prior tombent entre les régions de l'encodeur: les ponts du problème jouet.
- Larger β: the KL pulls $q_\phi(z|x)$ toward $\mathcal{N}(0, I)$, so fewer gaps. Samples improve ($67.2 \rightarrow 49.5$), reconstructions barely change ($13.0 \rightarrow 13.6$).
  - β plus grand: moins de trous, de meilleurs échantillons, presque les mêmes reconstructions.

Table of Contents

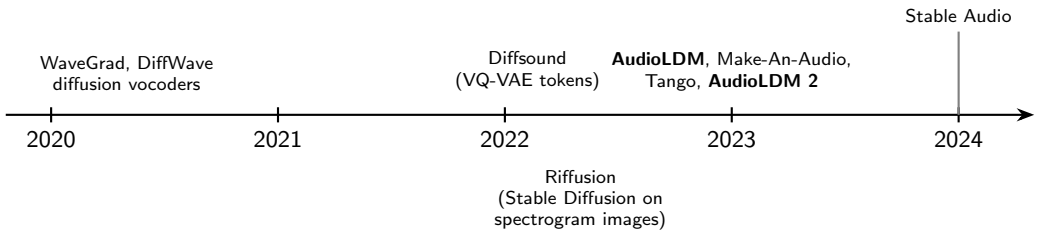
Centroid based approaches
- K-means clustering
- Gaussian Mixture Model
- More Advanced GMM Learning Methods

Spectral Clustering

Hierarchical Clustering

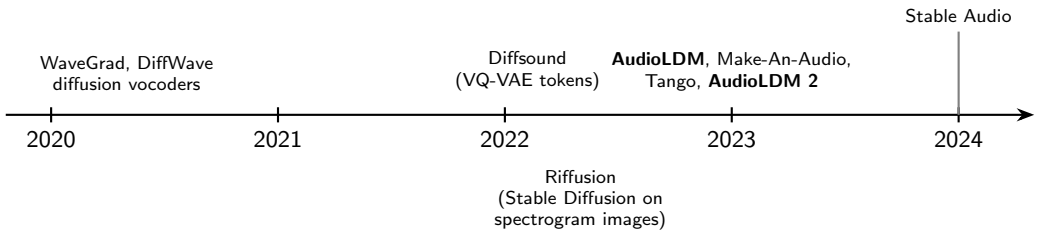
Deep Generative Models
- Variational Autoencoders
- Generative Adversarial Networks
- Diffusion Models
- Audio Diffusion Models

Diffusion models for audio: a short timeline



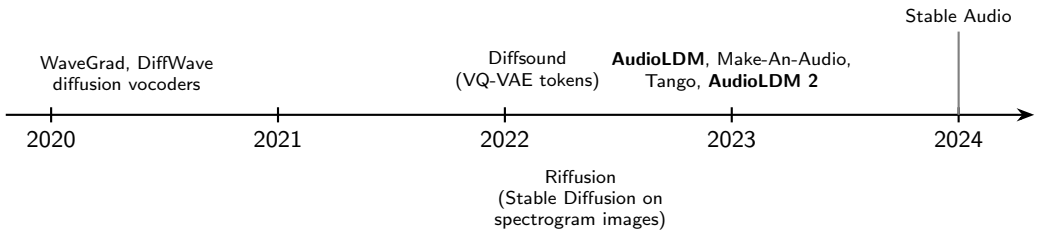
- First: diffusion as a vocoder (mel $\rightarrow$ waveform), as an alternative to GAN vocoders. / D'abord comme vocodeur.
  - WaveGrad (Chen et al., ICLR 2021), DiffWave (Kong et al., ICLR 2021)

Diffusion models for audio: a short timeline



- First: diffusion as a vocoder (mel → waveform), as an alternative to GAN vocoders. / D'abord comme vocodeur.
  - WaveGrad (Chen et al., ICLR 2021), DiffWave (Kong et al., ICLR 2021)
- Then: text-to-audio, mostly with **latent** diffusion on mel-spectrograms. / Puis: du texte vers l'audio.

Diffusion models for audio: a short timeline



- First: diffusion as a vocoder (mel → waveform), as an alternative to GAN vocoders. / D'abord comme vocodeur.
  - WaveGrad (Chen et al., ICLR 2021), DiffWave (Kong et al., ICLR 2021)
- Then: text-to-audio, mostly with **latent** diffusion on mel-spectrograms. / Puis: du texte vers l'audio.
- Today's focus: AudioLDM (Liu et al., ICML 2023) and AudioLDM 2 (Liu et al., IEEE/ACM TASLP 2024).

Text-to-audio: what is hard?

- Few paired data: text-audio datasets (AudioCaps, Clotho) have tens of thousands of clips. Image-text datasets have billions of pairs.
  - Peu de données appariées texte-audio.

Text-to-audio: what is hard?

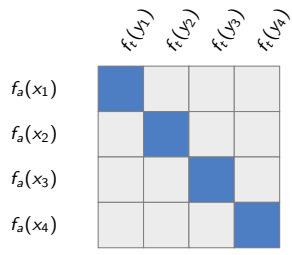
- Few paired data: text-audio datasets (AudioCaps, Clotho) have tens of thousands of clips. Image-text datasets have billions of pairs.
  - Peu de données appariées texte-audio.
- Audio is long: 10 s at 16 kHz is 160 000 samples $\rightarrow$ work in a compressed latent space.
  - L'audio est long: on travaille dans un espace latent compressé.

Text-to-audio: what is hard?

- Few paired data: text-audio datasets (AudioCaps, Clotho) have tens of thousands of clips. Image-text datasets have billions of pairs.
  - Peu de données appariées texte-audio.
- Audio is long: 10 s at 16 kHz is 160 000 samples $\rightarrow$ work in a compressed latent space.
  - L'audio est long: on travaille dans un espace latent compressé.
- Captions are vague: "a dog barks while cars pass by". When? How loud? In which room? Many valid answers.
  - Les descriptions sont vagues: beaucoup de réponses valides. Good for a generative model!

CLAP: one space for text and audio

- Contrastive Language-Audio Pretraining (Wu et al., ICASSP 2023), like CLIP for images.
- Audio encoder f_a (HTS-AT) and text encoder f_t (RoBERTa), normalized embeddings, $s_{ij} = f_a(x_i)^\top f_t(y_j)$.



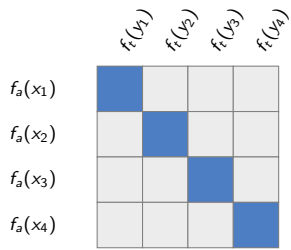
similarities s_{ij}:
high on the diagonal

CLAP: one space for text and audio

- Contrastive Language-Audio Pretraining (Wu et al., ICASSP 2023), like CLIP for images.
- Audio encoder f_a (HTS-AT) and text encoder f_t (RoBERTa), normalized embeddings, $s_{ij} = f_a(x_i)^\top f_t(y_j)$.
- Over a batch of N (audio, caption) pairs: / Sur un batch de N paires:

$$\mathcal{L} = -\frac{1}{2N} \sum_i \left(\log \frac{e^{s_{ii}/\tau}}{\sum_j e^{s_{ij}/\tau}} + \log \frac{e^{s_{ii}/\tau}}{\sum_j e^{s_{ji}/\tau}} \right)$$

  - Matched pairs close, the others far: "which caption goes with this clip?"



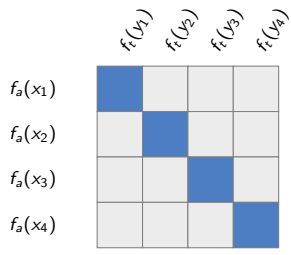
similarities s_{ij}:
high on the diagonal

CLAP: one space for text and audio

- Contrastive Language-Audio Pretraining (Wu et al., ICASSP 2023), like CLIP for images.
- Audio encoder f_a (HTS-AT) and text encoder f_t (RoBERTa), normalized embeddings, $s_{ij} = f_a(x_i)^\top f_t(y_j)$.
- Over a batch of N (audio, caption) pairs: / Sur un batch de N paires:

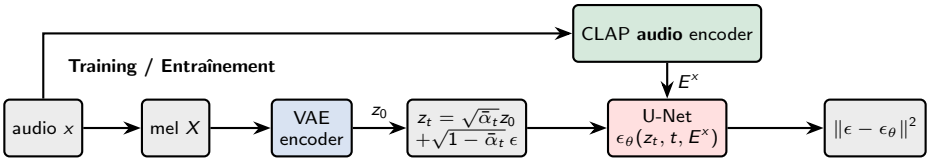
$$\mathcal{L} = -\frac{1}{2N} \sum_i \left(\log \frac{e^{s_{ii}/\tau}}{\sum_j e^{s_{ij}/\tau}} + \log \frac{e^{s_{ii}/\tau}}{\sum_j e^{s_{ji}/\tau}} \right)$$

  - Matched pairs close, the others far: "which caption goes with this clip?"
- Result: $f_t(\text{caption}) \approx f_a(\text{audio})$, interchangeable. AudioLDM's key trick! / Interchangeables!

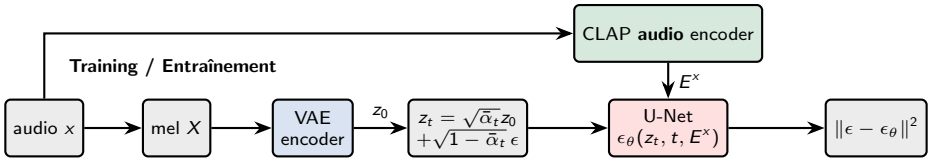


similarities s_{ij}:
high on the diagonal

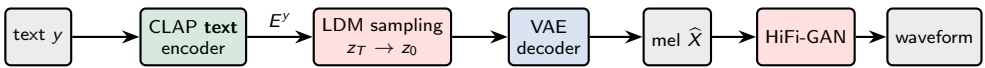
AudioLDM: the pipeline



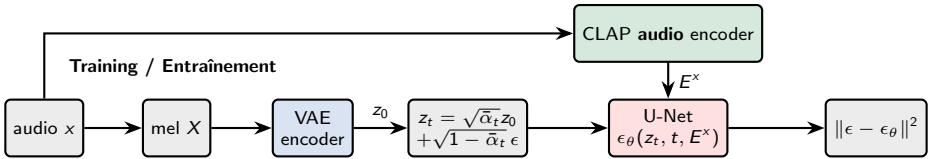
AudioLDM: the pipeline



Generation / Génération



AudioLDM: the pipeline



- The VAE compresses the mel-spectrogram $X \in \mathbb{R}^{T \times F}$ into $z \in \mathbb{R}^{C \times T/r \times F/r}$ (default $r = 4$); 16 kHz audio.
  - Le VAE compresse le mel-spectrogramme; la diffusion se fait sur z.

AudioLDM: train with audio, generate with text

- Training: condition on the CLAP **audio** embedding E^x of the clip itself.
  - Entraînement: on conditionne sur l'embedding **audio** du clip lui-même.

AudioLDM: train with audio, generate with text

- Training: condition on the CLAP **audio** embedding E^x of the clip itself.
  - Entraînement: on conditionne sur l'embedding **audio** du clip lui-même.
- Generation: swap in the CLAP **text** embedding E^y of the prompt. Same space, thanks to the contrastive training!
  - Génération: on le remplace par l'embedding **texte**. Même espace grâce à CLAP!

AudioLDM: train with audio, generate with text

- Training: condition on the CLAP **audio** embedding E^x of the clip itself.
  - Entraînement: on conditionne sur l'embedding **audio** du clip lui-même.
- Generation: swap in the CLAP **text** embedding E^y of the prompt. Same space, thanks to the contrastive training!
  - Génération: on le remplace par l'embedding **texte**. Même espace grâce à CLAP!
- So the diffusion model needs **no captions**: it is trained on audio only.
  - AudioSet, AudioCaps, Freesound, BBC Sound Effects.
  - Le modèle de diffusion n'a pas besoin de descriptions textuelles!

AudioLDM: train with audio, generate with text

- Training: condition on the CLAP **audio** embedding E^x of the clip itself.
  - Entraînement: on conditionne sur l'embedding **audio** du clip lui-même.
- Generation: swap in the CLAP **text** embedding E^y of the prompt. Same space, thanks to the contrastive training!
  - Génération: on le remplace par l'embedding **texte**. Même espace grâce à CLAP!
- So the diffusion model needs **no captions**: it is trained on audio only.
  - AudioSet, AudioCaps, Freesound, BBC Sound Effects.
  - Le modèle de diffusion n'a pas besoin de descriptions textuelles!
- Classifier-free guidance on the condition, to follow the text more closely.
  - Guidage sans classificateur, pour mieux suivre le texte.

AudioLDM: train with audio, generate with text

- Training: condition on the CLAP **audio** embedding E^x of the clip itself.
  - Entraînement: on conditionne sur l'embedding **audio** du clip lui-même.
- Generation: swap in the CLAP **text** embedding E^y of the prompt. Same space, thanks to the contrastive training!
  - Génération: on le remplace par l'embedding **texte**. Même espace grâce à CLAP!
- So the diffusion model needs **no captions**: it is trained on audio only.
  - AudioSet, AudioCaps, Freesound, BBC Sound Effects.
  - Le modèle de diffusion n'a pas besoin de descriptions textuelles!
- Classifier-free guidance on the condition, to follow the text more closely.
  - Guidage sans classificateur, pour mieux suivre le texte.
- Self-supervised in disguise: the model learns to regenerate a clip from its own embedding. / De l'auto-supervision déguisée.

AudioLDM: zero-shot audio manipulation

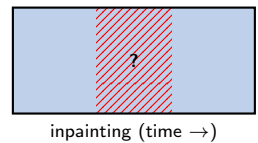
- No fine-tuning on these tasks: just change how we sample. / Sans réentraînement.

AudioLDM: zero-shot audio manipulation

- No fine-tuning on these tasks: just change how we sample. / Sans réentraînement.
- **Style transfer**: noise a source clip up to a step $t_0 < T$, then denoise with a new text.
  - t_0 sets how much of the source we keep. / t_0 règle ce qu'on garde de la source.

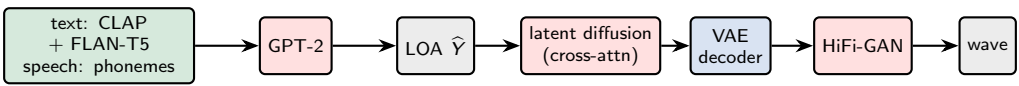
AudioLDM: zero-shot audio manipulation

- No fine-tuning on these tasks: just change how we sample. / Sans réentraînement.
- **Style transfer**: noise a source clip up to a step $t_0 < T$, then denoise with a new text.
  - t_0 sets how much of the source we keep. / t_0 règle ce qu'on garde de la source.
- **Inpainting**: a time span is missing. **Super-resolution**: the high frequencies are missing.
  - At every step, keep the known part from the (noised) observed audio, generate the rest.
  - À chaque étape, on garde la partie connue et on génère le reste.



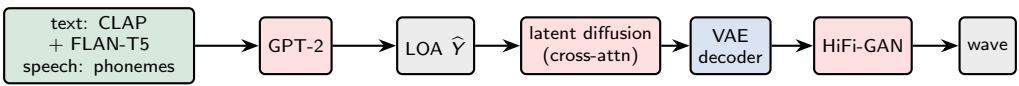
AudioLDM 2: one framework for sounds, music and speech

- Before: different models for speech, music and sound effects. / Avant: un modèle par type d'audio.



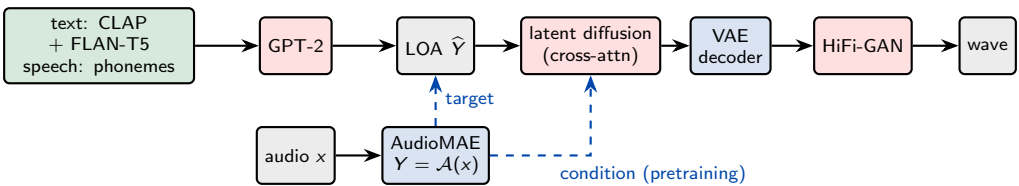
AudioLDM 2: one framework for sounds, music and speech

- Before: different models for speech, music and sound effects. / Avant: un modèle par type d'audio.
- Idea: a shared intermediate representation, the "**language of audio**" (LOA): features of AudioMAE, a self-supervised audio encoder.
  - Une représentation commune: le "langage de l'audio", les features d'AudioMAE.



AudioLDM 2: one framework for sounds, music and speech

- Before: different models for speech, music and sound effects. / Avant: un modèle par type d'audio.
- Idea: a shared intermediate representation, the "**language of audio**" (LOA): features of AudioMAE, a self-supervised audio encoder.
  - Une représentation commune: le "langage de l'audio", les features d'AudioMAE.



- Two stages: condition → LOA (GPT-2), then LOA → audio (latent diffusion, same VAE + HiFi-GAN as before).
  - Deux étapes: condition → LOA, puis LOA → audio.

AudioLDM 2: how it is trained

- The latent diffusion model is trained on **all** audio, labelled or not: it is conditioned on the AudioMAE features $Y = \mathcal{A}(x)$ of the same clip.
  - Same trick as AudioLDM with CLAP: self-supervised. / Même astuce que AudioLDM: auto-supervisé.

AudioLDM 2: how it is trained

- The latent diffusion model is trained on **all** audio, labelled or not: it is conditioned on the AudioMAE features $Y = \mathcal{A}(x)$ of the same clip.
  - Same trick as AudioLDM with CLAP: self-supervised. / Même astuce que AudioLDM: auto-supervisé.
- The LOA: average of the last 16 AudioMAE layers, pooled over time and frequency (pooling size $\lambda = 8$ for sounds and music, $\lambda = 1$ for speech).
  - Le LOA: moyenne des 16 dernières couches d'AudioMAE, puis un pooling temps-fréquence.

AudioLDM 2: how it is trained

- The latent diffusion model is trained on **all** audio, labelled or not: it is conditioned on the AudioMAE features $Y = \mathcal{A}(x)$ of the same clip.
  - Same trick as AudioLDM with CLAP: self-supervised. / Même astuce que AudioLDM: auto-supervisé.
- The LOA: average of the last 16 AudioMAE layers, pooled over time and frequency (pooling size $\lambda = 8$ for sounds and music, $\lambda = 1$ for speech).
  - Le LOA: moyenne des 16 dernières couches d'AudioMAE, puis un pooling temps-fréquence.
- GPT-2 learns condition $\rightarrow$ LOA on the audio that has text or phonemes. It generates the LOA autoregressively as **continuous** vectors (MSE loss, no tokens).
  - GPT-2 génère le LOA vecteur par vecteur, sans le discrétiser.

AudioLDM 2: how it is trained

- The latent diffusion model is trained on **all** audio, labelled or not: it is conditioned on the AudioMAE features $Y = \mathcal{A}(x)$ of the same clip.
  - Same trick as AudioLDM with CLAP: self-supervised. / Même astuce que AudioLDM: auto-supervisé.
- The LOA: average of the last 16 AudioMAE layers, pooled over time and frequency (pooling size $\lambda = 8$ for sounds and music, $\lambda = 1$ for speech).
  - Le LOA: moyenne des 16 dernières couches d'AudioMAE, puis un pooling temps-fréquence.
- GPT-2 learns condition $\rightarrow$ LOA on the audio that has text or phonemes. It generates the LOA autoregressively as **continuous** vectors (MSE loss, no tokens).
  - GPT-2 génère le LOA vecteur par vecteur, sans le discrétiser.
- Then fine-tune both stages together. The diffusion model also cross-attends to FLAN-T5 (text tasks).
  - Puis on ajuste les deux étapes ensemble.

AudioLDM 2: how it is trained

- The latent diffusion model is trained on **all** audio, labelled or not: it is conditioned on the AudioMAE features $Y = \mathcal{A}(x)$ of the same clip.
  - Same trick as AudioLDM with CLAP: self-supervised. / Même astuce que AudioLDM: auto-supervisé.
- The LOA: average of the last 16 AudioMAE layers, pooled over time and frequency (pooling size $\lambda = 8$ for sounds and music, $\lambda = 1$ for speech).
  - Le LOA: moyenne des 16 dernières couches d'AudioMAE, puis un pooling temps-fréquence.
- GPT-2 learns condition $\rightarrow$ LOA on the audio that has text or phonemes. It generates the LOA autoregressively as **continuous** vectors (MSE loss, no tokens).
  - GPT-2 génère le LOA vecteur par vecteur, sans le discrétiser.
- Then fine-tune both stages together. The diffusion model also cross-attends to FLAN-T5 (text tasks).
  - Puis on ajuste les deux étapes ensemble.
- Evaluated on text-to-audio (AudioCaps), text-to-music (MusicCaps), text-to-speech (LJSpeech).

AudioLDM vs. AudioLDM 2

	AudioLDM	**AudioLDM 2**
Diffusion model conditioned on	one CLAP embedding	LOA sequence (+ FLAN-T5), cross-attention
Text → condition	swap CLAP audio for CLAP text	GPT-2 predicts the LOA
Self-supervised trick	CLAP audio embedding of x	AudioMAE features of x
Audio types	general audio (mostly sound effects)	sounds, music, speech
Shared	mel VAE + latent diffusion + HiFi-GAN vocoder	

AudioLDM vs. AudioLDM 2

	AudioLDM	**AudioLDM 2**
Diffusion model conditioned on	one CLAP embedding	LOA sequence (+ FLAN-T5), cross-attention
Text → condition	swap CLAP audio for CLAP text	GPT-2 predicts the LOA
Self-supervised trick	CLAP audio embedding of x	AudioMAE features of x
Audio types	general audio (mostly sound effects)	sounds, music, speech
Shared	mel VAE + latent diffusion + HiFi-GAN vocoder	

- Common idea: train the generator on audio alone, by conditioning on a representation computed from the audio itself.
  - Idée commune: entraîner sur l'audio seul, en conditionnant sur une représentation de l'audio lui-même.

Try it yourself

```
import scipy, torch
from diffusers import AudioLDM2Pipeline

pipe = AudioLDM2Pipeline.from_pretrained(
    "cvssp/audioldm2", torch_dtype=torch.float16).to("cuda")

audio = pipe(
    "The sound of a hammer hitting a wooden surface.",
    negative_prompt="Low quality.",
    num_inference_steps=200,       # denoising steps
    audio_length_in_s=10.0,
    num_waveforms_per_prompt=3,    # ranked by CLAP score
).audios

scipy.io.wavfile.write("out.wav", rate=16000, data=audio[0])
```

- `guidance_scale` (default 3.5) is w in classifier-free guidance.
- `negative_prompt` replaces $\varnothing$ in the guidance: steer **away** from it.
  - On s'éloigne du `negative_prompt`.

Demos / Démos: `https://audioldm.github.io` `https://audioldm.github.io/audioldm2`

How do we evaluate generated audio?

- Fréchet Audio Distance (Kilgour et al., 2019): embed real and generated clips with a pretrained classifier (VGGish), fit a Gaussian to each set:

$$\text{FAD} = \|\mu_r - \mu_g\|_2^2 + \text{Tr}\big(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\big)$$

  - Gaussians again! The distance between the two fitted Gaussians. / Encore des Gaussiennes!

How do we evaluate generated audio?

- Fréchet Audio Distance (Kilgour et al., 2019): embed real and generated clips with a pretrained classifier (VGGish), fit a Gaussian to each set:

$$\mathrm{FAD} = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}\big(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\big)$$

  - Gaussians again! The distance between the two fitted Gaussians. / Encore des Gaussiennes!
- KL between the class predictions (PANNs) of the generated clip and of the reference clip.

How do we evaluate generated audio?

- Fréchet Audio Distance (Kilgour et al., 2019): embed real and generated clips with a pretrained classifier (VGGish), fit a Gaussian to each set:

$$\mathrm{FAD} = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}\big(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\big)$$

  - Gaussians again! The distance between the two fitted Gaussians. / Encore des Gaussiennes!
- KL between the class predictions (PANNs) of the generated clip and of the reference clip.
- CLAP score: how well the audio matches the text. / Accord entre l'audio et le texte.

How do we evaluate generated audio?

- Fréchet Audio Distance (Kilgour et al., 2019): embed real and generated clips with a pretrained classifier (VGGish), fit a Gaussian to each set:

$$\mathrm{FAD} = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}\big(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\big)$$

  - Gaussians again! The distance between the two fitted Gaussians. / Encore des Gaussiennes!
- KL between the class predictions (PANNs) of the generated clip and of the reference clip.
- CLAP score: how well the audio matches the text. / Accord entre l'audio et le texte.
- Listening tests: overall quality (OVL) and relevance to the text (REL). Still the most reliable.
  - Tests d'écoute: toujours les plus fiables.

Beyond AudioLDM

- Tango (Ghosal et al., 2023): an instruction-tuned LLM (FLAN-T5) as the text encoder, trained on AudioCaps only.

Beyond AudioLDM

- Tango (Ghosal et al., 2023): an instruction-tuned LLM (FLAN-T5) as the text encoder, trained on AudioCaps only.
- Make-An-Audio (Huang et al., ICML 2023): creates extra captions for unlabelled audio, then latent diffusion.

Beyond AudioLDM

- Tango (Ghosal et al., 2023): an instruction-tuned LLM (FLAN-T5) as the text encoder, trained on AudioCaps only.
- Make-An-Audio (Huang et al., ICML 2023): creates extra captions for unlabelled audio, then latent diffusion.
- Stable Audio (Evans et al., ICML 2024): latent diffusion on a **waveform** autoencoder (no mel, no vocoder), 44.1 kHz stereo, conditioned on the start time and total length.
  - Diffusion latente directement sur la forme d'onde, avec une durée contrôlée.

Beyond AudioLDM

- Tango (Ghosal et al., 2023): an instruction-tuned LLM (FLAN-T5) as the text encoder, trained on AudioCaps only.
- Make-An-Audio (Huang et al., ICML 2023): creates extra captions for unlabelled audio, then latent diffusion.
- Stable Audio (Evans et al., ICML 2024): latent diffusion on a **waveform** autoencoder (no mel, no vocoder), 44.1 kHz stereo, conditioned on the start time and total length.
  - Diffusion latente directement sur la forme d'onde, avec une durée contrôlée.
- Open problems: long structure (a whole song), precise control (timing of events), evaluation, rights on the training data.
  - Problèmes ouverts: structure longue, contrôle précis, évaluation, droits sur les données.

Generative models recap

- VAE: latent variable model + encoder, trained with the ELBO. Fast, but blurry.
  - Modèle à variable latente + encodeur. Rapide, mais flou.
- GAN: a generator against a discriminator. Sharp and fast, but unstable. The best vocoders (HiFi-GAN).
  - Un générateur contre un discriminateur. Net et rapide, mais instable.
- Diffusion: fixed noising, learned denoising. The ELBO reduces to "predict the noise". Best quality, slow $\rightarrow$ DDIM, latent diffusion.
  - Bruitage fixe, débruitage appris. Meilleure qualité, mais lent.
- AudioLDM / AudioLDM 2: mel VAE + latent diffusion (CLAP / LOA conditioning) + HiFi-GAN.
- And the ELBO from the GMM was behind all of it! / Et la borne du GMM était derrière tout ça!

Suggested reading (generative models)

- Kingma & Welling, Auto-Encoding Variational Bayes, ICLR 2014.
- Goodfellow et al., Generative Adversarial Nets, NeurIPS 2014.
- Ho, Jain & Abbeel, Denoising Diffusion Probabilistic Models, NeurIPS 2020.
- Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021.
- Rombach et al., High-Resolution Image Synthesis with Latent Diffusion Models, CVPR 2022.
- Liu et al., AudioLDM: Text-to-Audio Generation with Latent Diffusion Models, ICML 2023.
- Liu et al., AudioLDM 2: Learning Holistic Audio Generation with Self-supervised Pretraining, IEEE/ACM TASLP 2024.
- Tutorial: C. Luo, Understanding Diffusion Models: A Unified Perspective, arXiv:2208.11970.
- Blog: L. Weng, What are Diffusion Models?
  `https://lilianweng.github.io/posts/2021-07-11-diffusion-models/`

Next week/class

- Well, next week we are off as it is the reading week.
- But after that, in the next class we will add connections between h_n's!

