Deep generative models

Pierre-Alexandre Mattei

Inria

UNIVERSITÉ CÔTE D'AZUR

3iA Côte d'Azur
Institut interdisciplinaire d'intelligence artificielle

Menu for these two last lectures

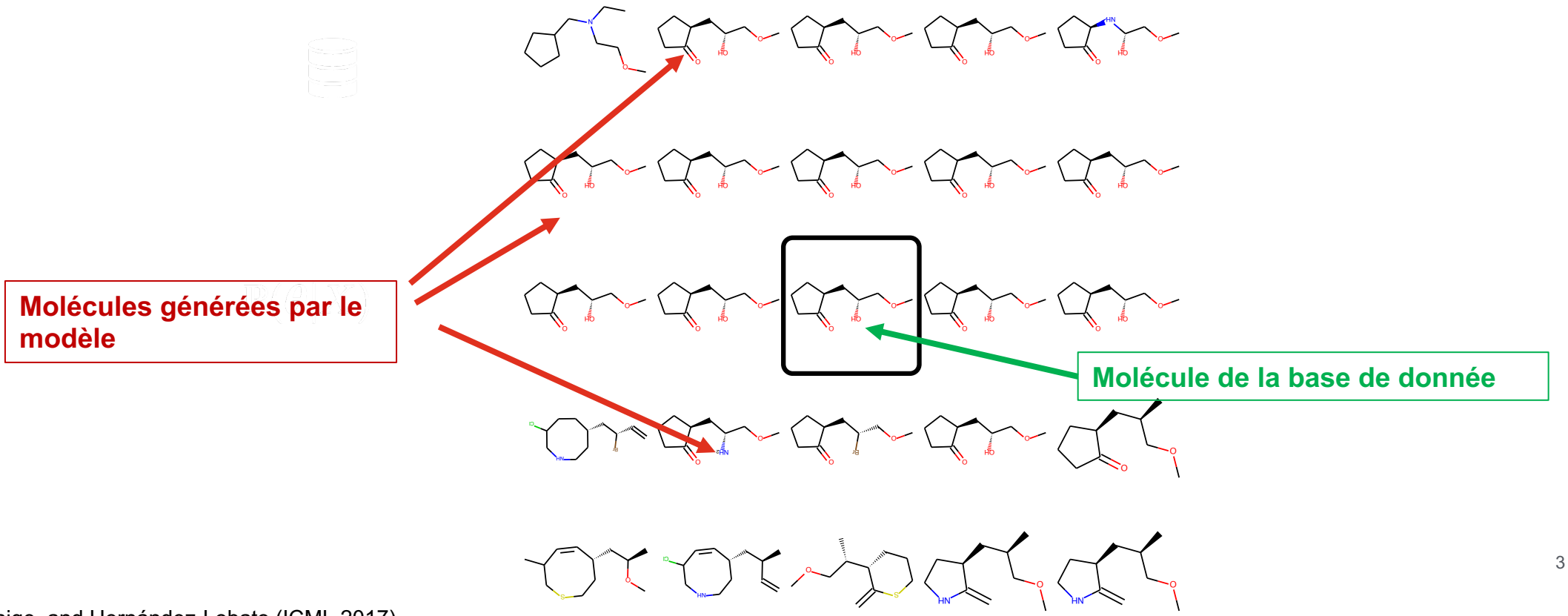
1. Deep latent variable models and variational auto-encoders

2. Diffusion models

3. Other deep generative models

Why generative models?

- We want to approximate the distribution $p_{\text{data}}(x)$ of molecules.



Kusner, Paige, and Hernández-Lobato (ICML 2017)

Generating faces…

« fake » images sampled by a VAE

Vahdat & Kautz (NeurIPS 2020)

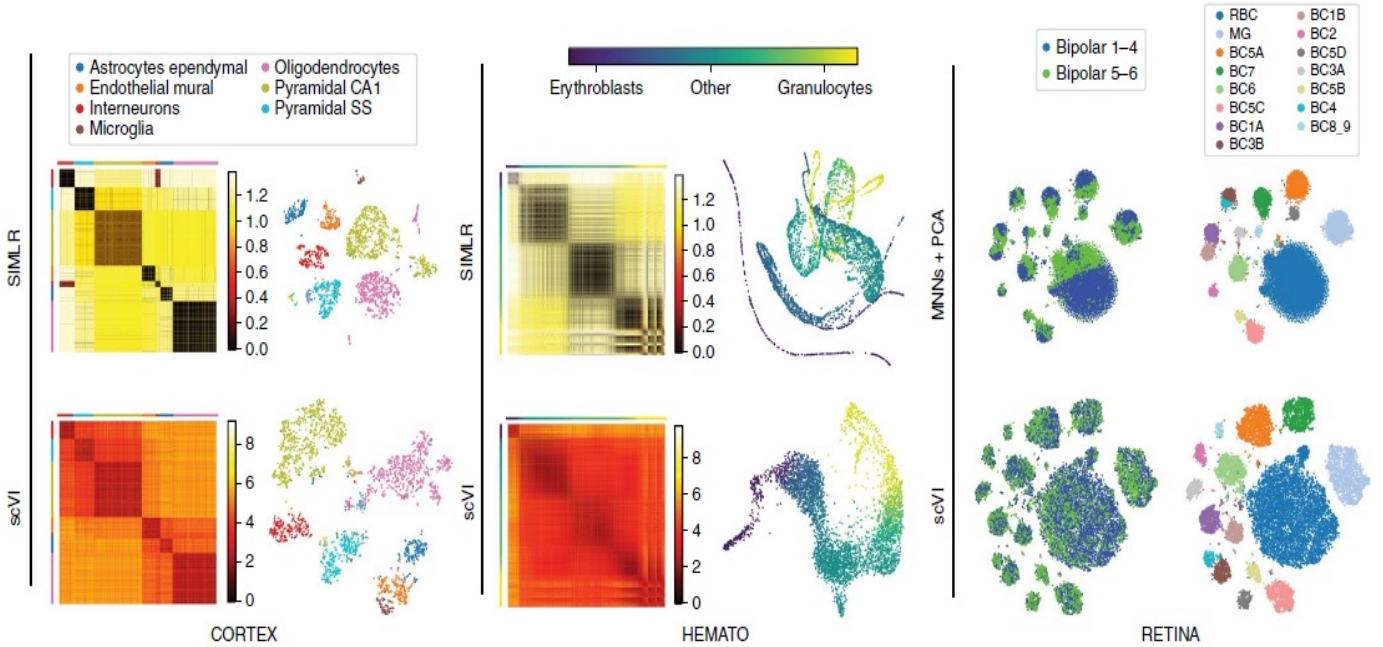
$$P(\theta|X)$$

Beyond images and molecules

nature methods

Deep generative modeling for single-cell transcriptomics

Romain Lopez[1], Jeffrey Regier[1], Michael B. Cole[2], Michael I. Jordan[1,3] and Nir Yosef[1,4,5*]



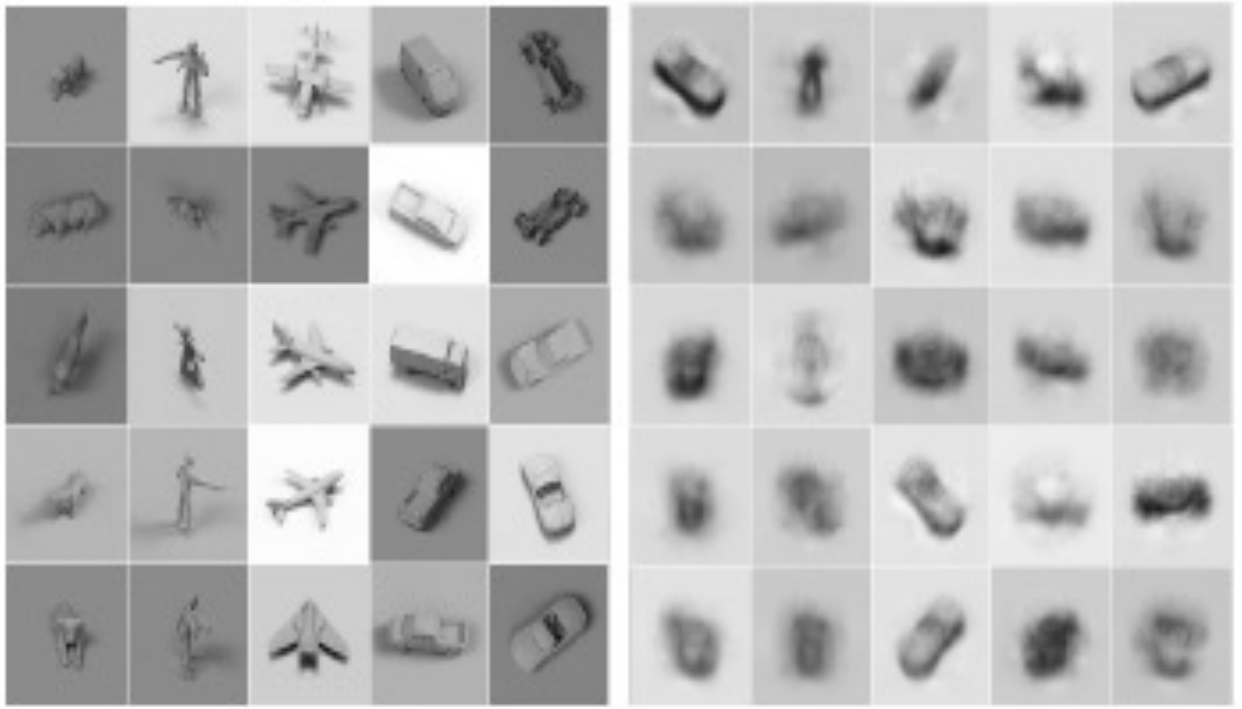
Text to image generation

- The goal is here to approximate

$$p_{\text{data}}(x|t = \text{“A head of broccoli complaining about the weather”})$$

https://openai.com/dall-e-2/

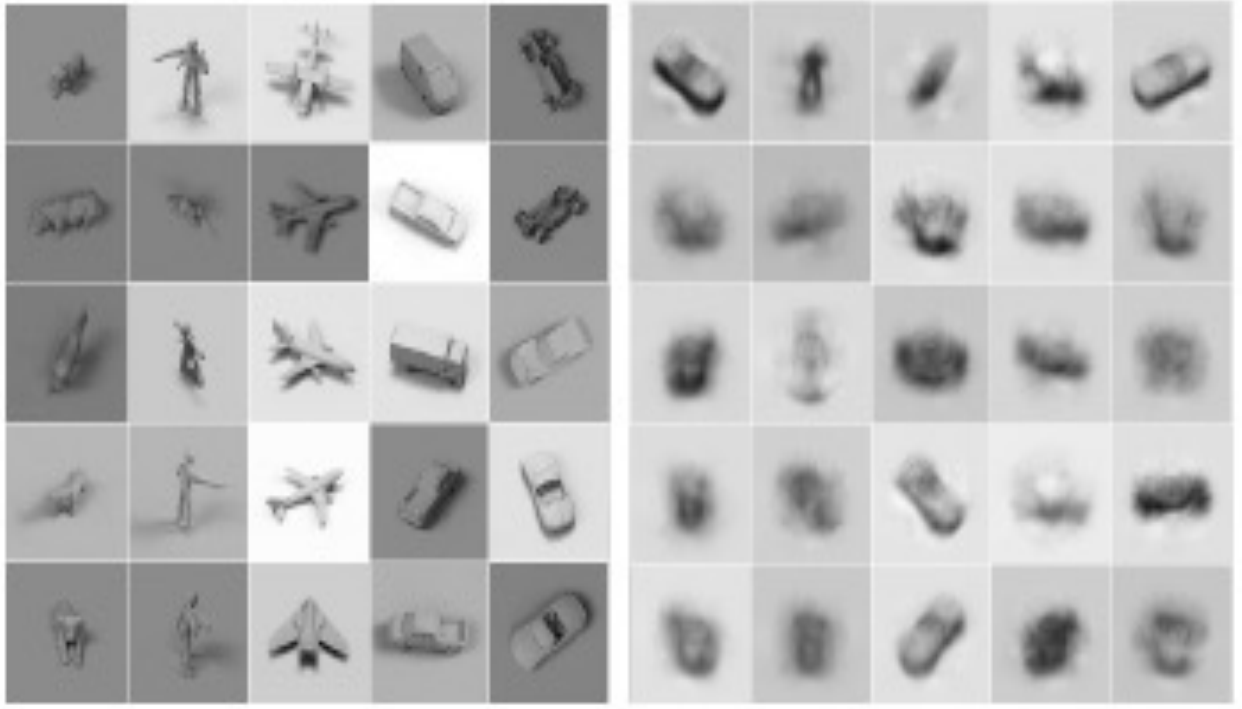
Impressive progress



Rezende, Mohamed, and Wiestra (ICML 2014)

Impressive progress

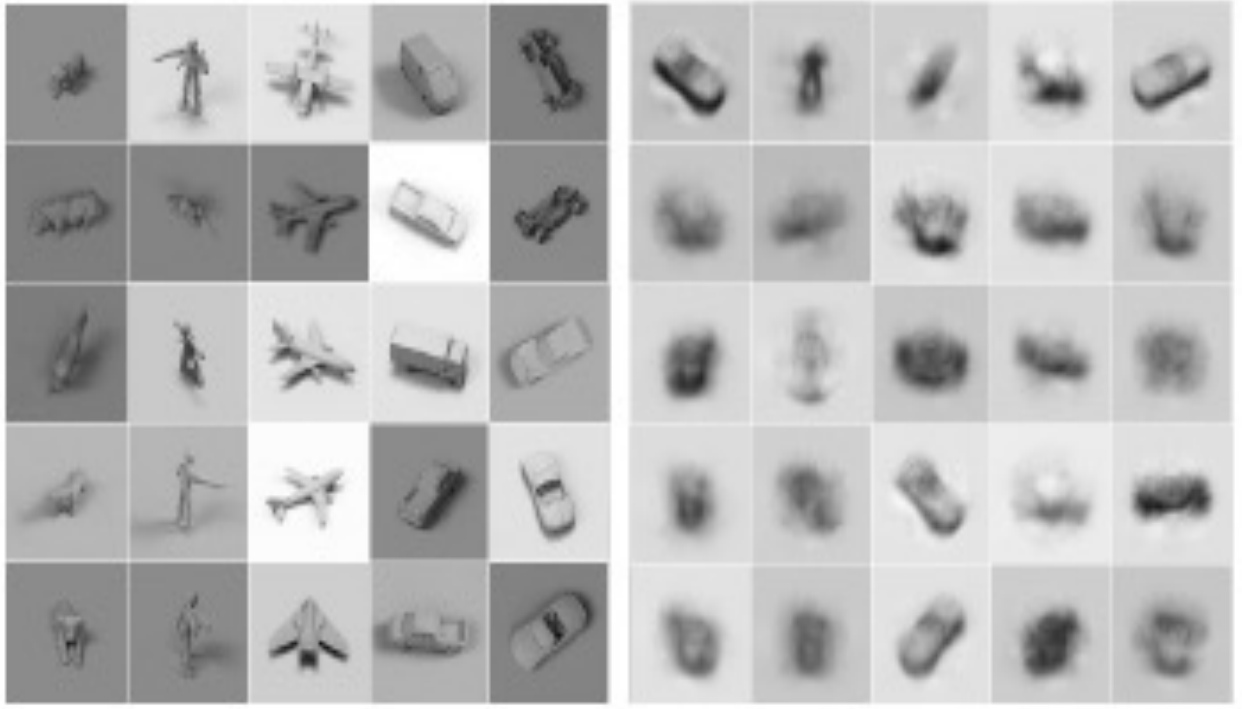
https://openai.com/dall-e-2/ (2022)



Rezende, Mohamed, and Wiestra (ICML 2014)

Impressive progress

https://openai.com/dall-e-2/ (2022)



Rezende, Mohamed, and Wiestra (ICML 2014)

Brock, Donahue, and Simonyan (ICLR 2019)

1

Deep latent variable models
VAEs and extensions

Why latent-variable models?

Let's assume that we have i.i.d. data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ that live in a **high-dimensional space** $\mathcal{X}$ (for example images of newspaper articles).

Often, it is reasonable to assume that **these data essentially depend on a few important factors of variation:**

- if we have images of faces: size of nose, color of hair, glasses or not...
- if we have newspaper articles: topics of the paper, style...
- if we have molecules: physical properties, geometrical shape...

This assumption is the cornerstone of **latent variable models**, that assume that the data are governed by **unobserved random variables** $\mathbf{z}_1, \ldots, \mathbf{z}_n$ **that live in a low dimensional space** $\mathcal{Z}$ (for example $\mathbf{R}^2$). We can think of $\mathbf{z}_i$ as a **code** summarizing the essential factors of the data point $\mathbf{x}_i$.

Linear latent-variable models

In this slide, we assume that the data are continuous (i.e. $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$).

Factor analysis is probably one of the oldest latent variable models (studied since at least the 1940s). The generative process is:

- $\mathbf{z}_i \sim \mathcal{N}(0, \mathbf{I}_d)$,
- $\mathbf{x}_i | \mathbf{z}_i \sim \mathcal{N}(\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \boldsymbol{\Psi})$.

Probabilistic PCA (PPCA, Tipping and Bishop, JRSSB, 1999) is a a slightly less general model

- $\mathbf{z}_i \sim \mathcal{N}(0, \mathbf{I}_d)$,
- $\mathbf{x}_i | \mathbf{z}_i \sim \mathcal{N}(\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \sigma^2 \boldsymbol{I}_p)$.

Linear latent-variable models

In this slide, we assume that the data are continuous (i.e. $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$).

Factor analysis is probably one of the oldest latent variable models (studied since at least the 1940s). The generative process is:

- $\mathbf{z}_i \sim \mathcal{N}(0, \mathbf{I}_d)$,
- $\mathbf{x}_i | \mathbf{z}_i \sim \mathcal{N}(\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \boldsymbol{\Psi})$.

Probabilistic PCA (PPCA, Tipping and Bishop, JRSSB, 1999) is a a slightly less general model

- $\mathbf{z}_i \sim \mathcal{N}(0, \mathbf{I}_d)$,
- $\mathbf{x}_i | \mathbf{z}_i \sim \mathcal{N}(\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \sigma^2 \boldsymbol{I}_p)$.

- For such models, performing maximum likelihood is not too hard:
  - EM algorithm for FA
  - Closed-form MLE for PPCA (based on the SVD of the data matrix)

From linear (« shallow ») to deep variable models

How do we transform this kind of model

- $\mathbf{z}_i \sim \mathcal{N}(0, \mathbf{I}_d)$,
- $\mathbf{x}_i | \mathbf{z}_i \sim \mathcal{N}(\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \sigma^2 \boldsymbol{I}_p)$.

...into "something deep"?

$\mathbb{P}(\theta|X)$

From linear (« shallow ») to deep variable models

How do we transform this kind of model

- $\mathbf{z}_i \sim \mathcal{N}(0, \mathbf{I}_d)$,
- $\mathbf{x}_i | \mathbf{z}_i \sim \mathcal{N}(\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \sigma^2 \boldsymbol{I}_p)$.

...into "something deep"?

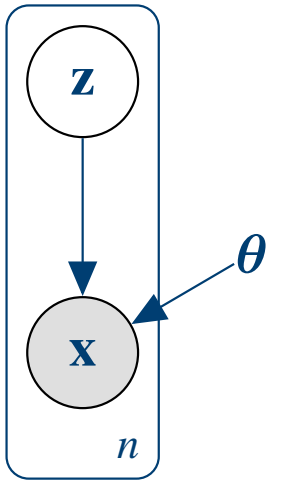
We can replace the affine function $\mathbf{z} \mapsto \mathbf{W}\mathbf{z} + \mu$ by a neural net!

That's the key idea of **deep latent variable models (DLVMs)**, present in particular in both **variational autoencoders (VAEs)**, invented independently by Kingma and Welling (ICLR 2014) and Rezende, Mohamed & Wierstra (ICML 2014), and **generative adversarial networks (GANs)** (Goodfellow et al., NeurIPS 2014).

Deep latent variable models

Assume that $(\mathbf{x}_i, \mathbf{z}_i)_{i \leq n}$ are i.i.d. random variables driven by the model:

$$\begin{cases} \mathbf{z} \sim p(\mathbf{z}) & \text{(prior)} \\ \mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}) & \text{(observation model)} \end{cases}$$



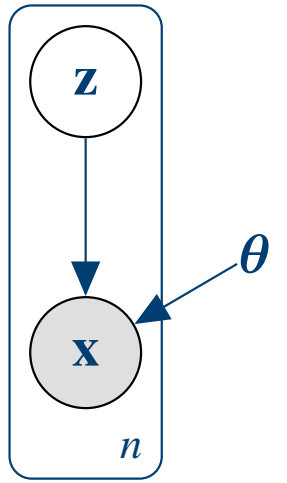
where

- $\mathbf{z} \in \mathbb{R}^d$ is the **latent** variable,
- $\mathbf{x} \in \mathcal{X}$ is the **observed** variable.
  - the function $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to H$ is a **(deep) neural network** called the **decoder**
  - $(\Phi(\cdot \mid \boldsymbol{\eta}))_{\boldsymbol{\eta} \in H}$ is a parametric family called the **observation model**, usually **very simple**: unimodal and fully factorised (e.g. multivariate Gaussians or products of multinomials)

Deep latent variable models

Assume that $(\mathbf{x}_i, \mathbf{z}_i)_{i \leq n}$ are i.i.d. random variables driven by the model:

$$\begin{cases} \mathbf{z} \sim p(\mathbf{z}) & \text{(prior)} \\ \mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}) = \Phi(\mathbf{x} \mid f_{\boldsymbol{\theta}}(\mathbf{z})) & \text{(observation model)} \end{cases}$$



where

- $\mathbf{z} \in \mathbb{R}^d$ is the **latent** variable,
- $\mathbf{x} \in \mathcal{X}$ is the **observed** variable.
  - the function $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to H$ is a **(deep) neural network** called the **decoder**
  - $(\Phi(\cdot \mid \boldsymbol{\eta}))_{\boldsymbol{\eta} \in H}$ is a parametric family called the **observation model**, usually **very simple**: unimodal and fully factorised (e.g. multivariate Gaussians or products of multinomials)

Deep latent variable models: the role of the prior

As in regular factor analysis, the prior distribution of the latent variable is often an **isotropic Gaussian** $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}_d, \mathbf{I}_d)$.

Note that **this prior is not a prior in the Bayesian sense** (i.e., about parameter uncertainty).

$\mathbb{P}(\theta|X)$

Deep latent variable models: the role of the observation model

The observation model $(\Phi(\cdot \mid \boldsymbol{\eta}))_{\boldsymbol{\eta} \in H}$ usually **very simple**: unimodal and fully factorised (e.g. multivariate Gaussians or products of multinomials)

Its parameters are the output of the decoder.

$$\begin{cases} \mathbf{z} \sim p(\mathbf{z}) & \text{(prior)} \\ \mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{z})) & \text{(Gaussian observation model)} \end{cases}$$

$$\begin{cases} \mathbf{z} \sim p(\mathbf{z}) & \text{(prior)} \\ \mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}) = \mathcal{B}(\mathbf{x} \mid \boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{z})) & \text{(Bernoulli observation model)} \end{cases}$$

$$\begin{cases} \mathbf{z} \sim p(\mathbf{z}) & \text{(prior)} \\ \mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}) = \mathrm{St}(\mathbf{x} \mid \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{z}), \boldsymbol{\nu}_{\boldsymbol{\theta}}(\mathbf{z})) & \text{(Student's t observation model)} \end{cases}$$

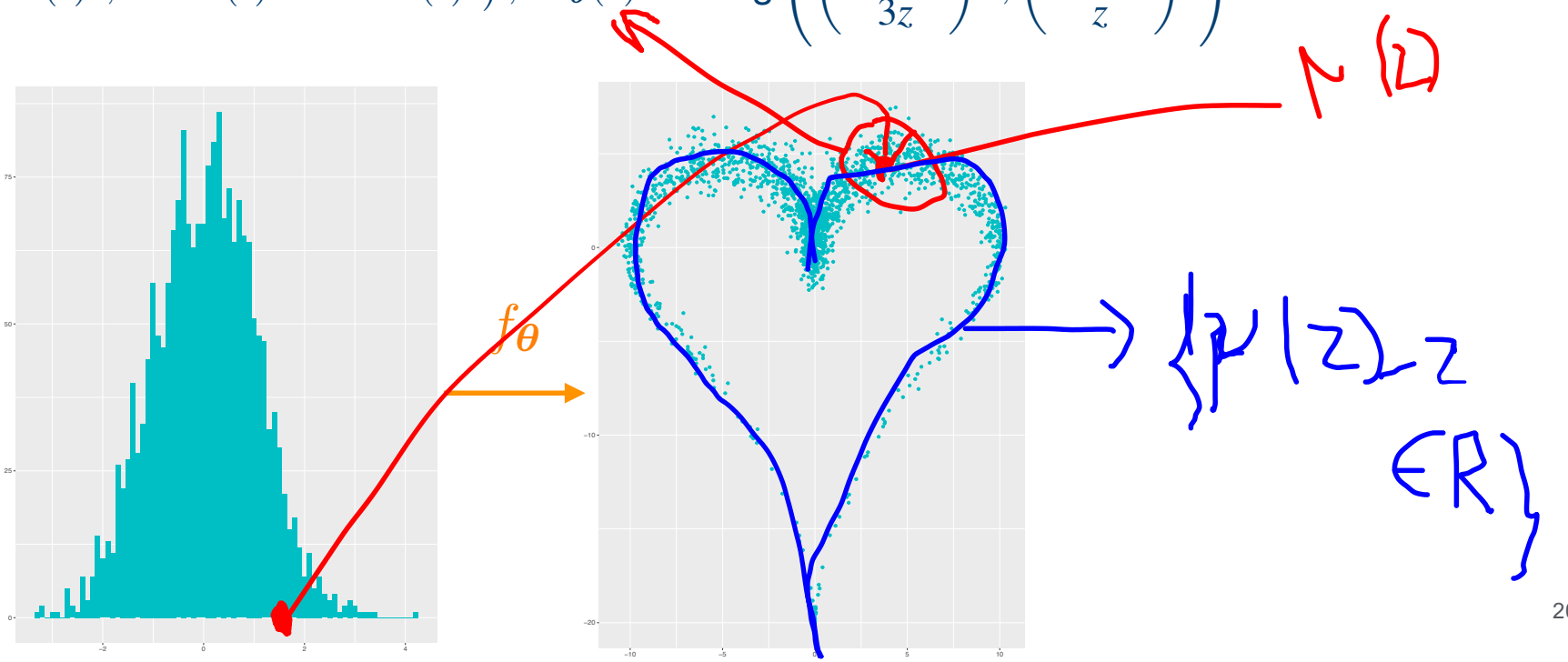
Deep latent variable models: the role of the decoder

The role of the **decoder** $f_\theta : \mathbb{R}^d \to H$ is:

- to transform $\mathbf{z}$ (**the code**) into parameters $\eta = f_{\boldsymbol{\theta}}(\mathbf{z})$ of the observation model $\Phi(\cdot \mid \eta)$.
- The weights $\boldsymbol{\theta}$ of the **decoder** are learned.

Simple non-linear decoder ($d = 1, p = 2$): $f_{\boldsymbol{\theta}}(z) = \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{z})$ with, for all $z \in \mathbb{R}$,

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(z) = \left(10\sin(z)^3, 10\cos(z) - 10\cos(z)^4\right), \ \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{z}) = \text{Diag}\left(\left(\frac{\sin(z)}{3z}\right)^2, \left(\frac{\sin(z)}{z}\right)^2\right).$$



Illustrative example of a DLVM for binary images

Training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ binary MNIST

Illustrative example of a DLVM for binary images

Generative model for $\mathbf{z} \in \mathbb{R}^2$ and $\mathbf{x} \in \{0, 1\}^{28 \times 28}$

$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

Decoder network

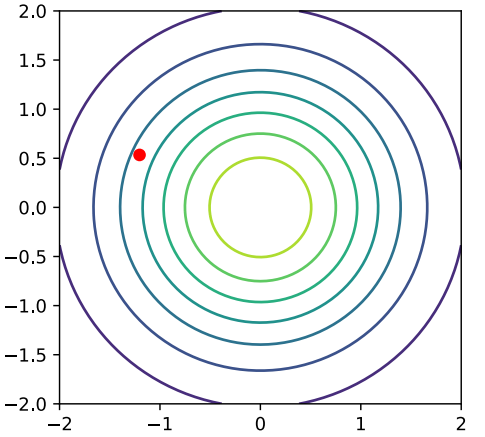
$$f(\mathbf{z}) = \text{Sigmoid}(\mathbf{V} \tanh(\mathbf{W}\mathbf{z} + \mathbf{b}) + \boldsymbol{\beta})$$

$$\mathbb{P}(\theta|X)$$

Generation

$$\mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

$$\mathbf{z} = (-1.2033, 0.5340)$$



Illustrative example of a DLVM for binary images

Generative model for $\mathbf{z} \in \mathbb{R}^2$ and $\mathbf{x} \in \{0,1\}^{28\times 28}$

$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

Decoder network

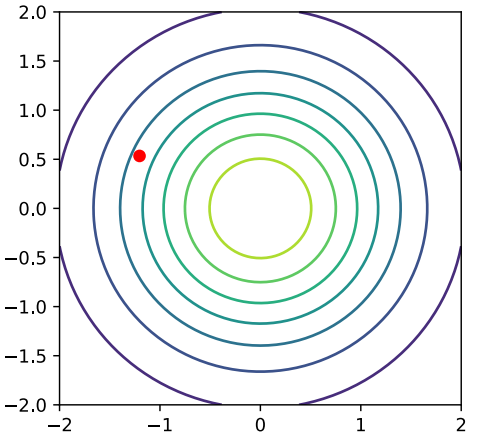
$$f(\mathbf{z}) = \text{Sigmoid}(\mathbf{V}\tanh(\mathbf{W}\mathbf{z} + \mathbf{b}) + \boldsymbol{\beta})$$

$\mathbb{P}(\theta|X)$

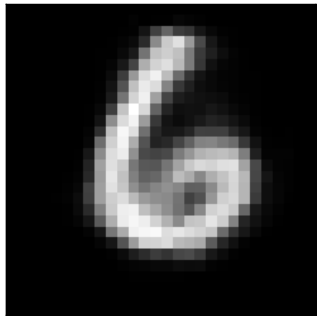
Generation

$$\mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

$$\mathbf{z} = (-1.2033, 0.5340)$$



$f(\mathbf{z})$



Illustrative example of a DLVM for binary images

Generative model for $\mathbf{z} \in \mathbb{R}^2$ and $\mathbf{x} \in \{0,1\}^{28\times 28}$

$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

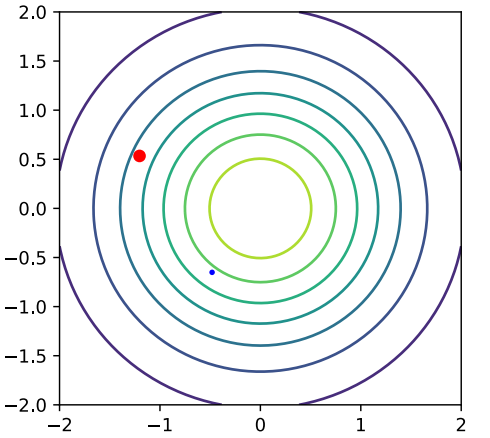
Decoder network

$$f(\mathbf{z}) = \text{Sigmoid}(\mathbf{V}\tanh(\mathbf{W}\mathbf{z} + \mathbf{b}) + \boldsymbol{\beta})$$

Generation

$$\mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

$$\mathbf{z} = (-1.2033, 0.5340)$$



$f(\mathbf{z})$

$x^{j,k} \sim \text{Bern}(f^{j,k}(\mathbf{z}))$

Illustrative example of a DLVM for binary images

Generative model for $\mathbf{z} \in \mathbb{R}^2$ and $\mathbf{x} \in \{0,1\}^{28\times 28}$

$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

Decoder network

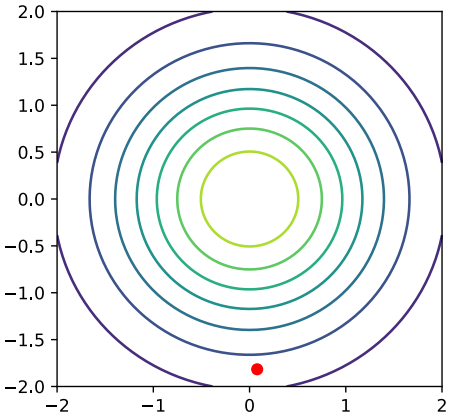
$$f(\mathbf{z}) = \text{Sigmoid}(\mathbf{V}\tanh(\mathbf{W}\mathbf{z} + \mathbf{b}) + \boldsymbol{\beta})$$

$\mathbb{P}(\theta|X)$

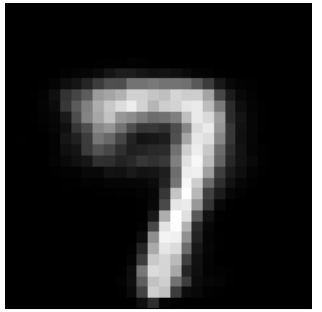
Generation

$$\mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

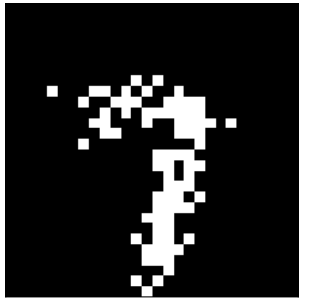
$$\mathbf{z} = (0.0791, -1.8165)$$



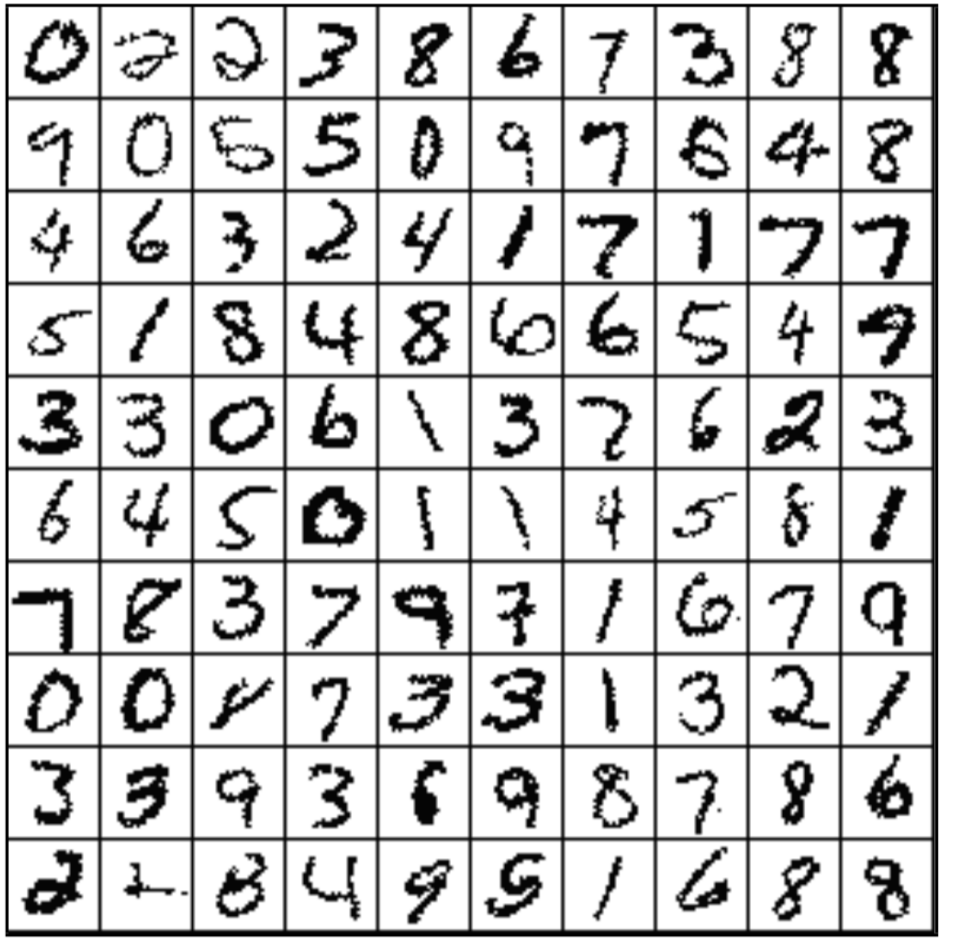
$f(z)$



$x^{j,k} \sim \text{Bern}(f^{j,k}(z))$



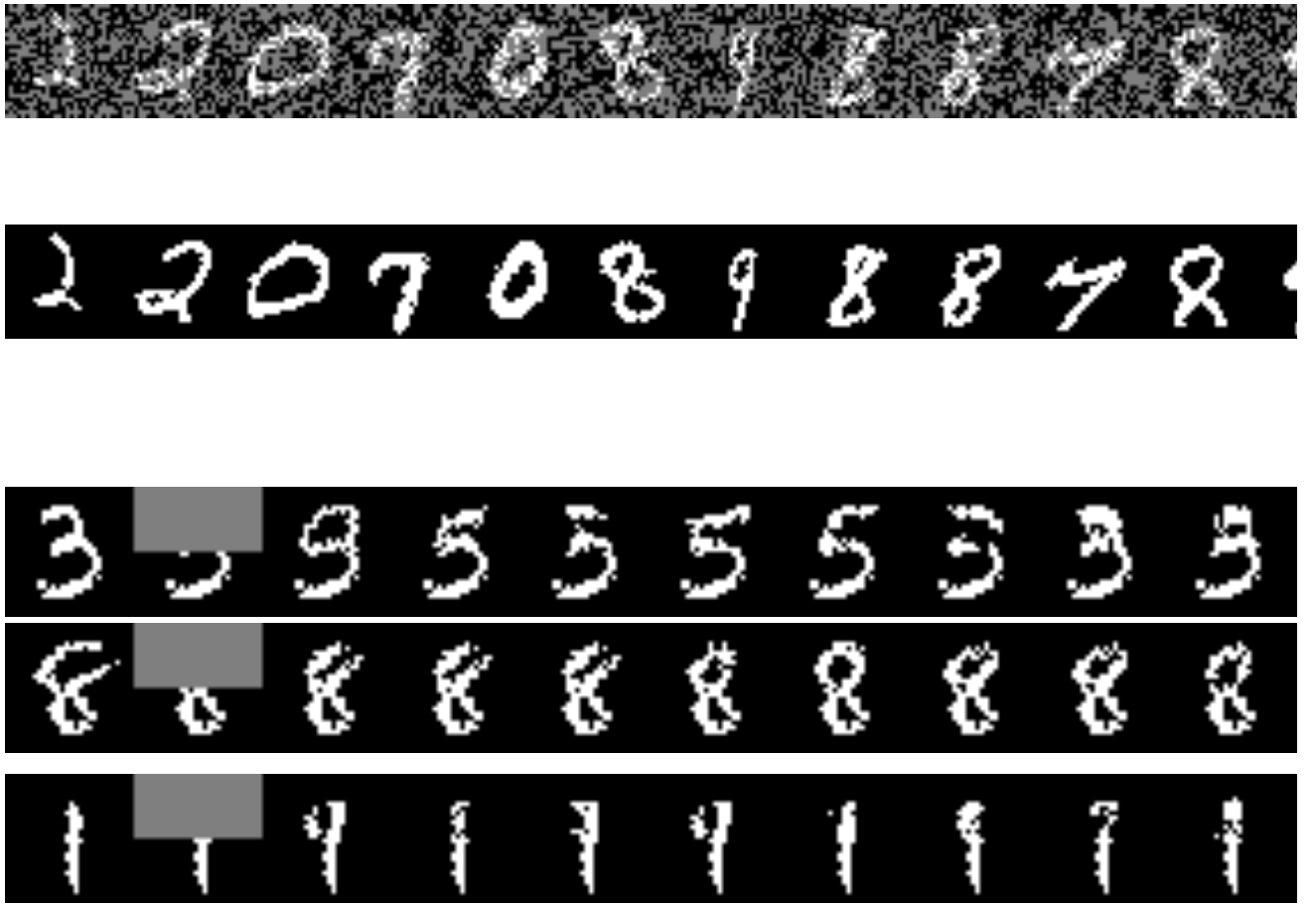
DLVM from one of the seminal VAE papers (Rezende et al., ICML'14)



Training data

Model samples

DLVM for missing data imputation (Mattei & Frellsen, ICML'19)



Maximum likelihood for DLVM

Given a data matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^\intercal \in \mathcal{X}^n$, the **log-likelihood function** for a DLVM is

$$\ell(\boldsymbol{\theta}) = \log p_{\boldsymbol{\theta}}(\mathbf{X}) = \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i),$$

where

$$p_{\boldsymbol{\theta}}(\mathbf{x}_i) = \int_{\mathbb{R}^d} p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}) p(\mathbf{z}) \mathbf{z}.$$

We would like to find a **MLE** $\hat{\boldsymbol{\theta}} \in \operatorname{argmax}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta})$.

Maximum likelihood for DLVM

Given a data matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^\intercal \in \mathcal{X}^n$, the **log-likelihood function** for a DLVM is

$$\ell(\boldsymbol{\theta}) = \log p_{\boldsymbol{\theta}}(\mathbf{X}) = \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i),$$

where

$$p_{\boldsymbol{\theta}}(\mathbf{x}_i) = \int_{\mathbb{R}^d} p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}) p(\mathbf{z}) \mathbf{z}.$$

We would like to find a **MLE** $\hat{\boldsymbol{\theta}} \in \operatorname{argmax}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta})$.

However, even with a simple output density $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$:

- $p_{\boldsymbol{\theta}}(\mathbf{x})$ is **intractable** rendering **MLE intractable**
- $p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})$ is **intractable** rendering **EM intractable**
- **stochastic EM is not scalable** to large n and moderate d.

Maximum likelihood for DLVMs via importance sampling

A general and scalable framework to tackle these issues was proposed by Kingma & Welling (2014), Rezende et al. (2014), leading to the **variational autoencoder (VAE).**

Here, **I am going to derive this approach in a slightly different manner**, largely inspired by the following paper:

Burda, Grosse & Salakhutdinov (2016), *Importance weighted autoencoders*, ICLR 2016

The main idea is to use **Monte Carlo techniques** to approximate the intractable integrals

$$p_{\boldsymbol{\theta}}(\mathbf{x}_i) = \int_{\mathbb{R}^d} p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}.$$

Beyond simple MC: importance sampling

- Short Monte Carlo course! Our goal is to approximate an integral

$$I = \int_{\Omega} f(x)p(x)dx$$

- We already saw the **simple MC estimate**

$$I \approx \frac{1}{K}\sum_{k=1}^{K} f(x_k) = \hat{I}_K.$$

Beyond simple MC: importance sampling

- Short Monte Carlo course! Our goal is to approximate an integral

$$I = \int_{\Omega} f(x)p(x)dx$$

- We already saw the **simple MC estimate**

$$I \approx \frac{1}{K}\sum_{k=1}^{K} f(x_k) = \hat{I}_K.$$

- This estimate has nice properties:
  - Unbiasedness: $\mathbb{E}[\hat{I}_K] = I$
  - Consistency: $\hat{I}_K \overset{a.s.}{\to} I$
  - Asymptotic normality

Beyond simple MC: importance sampling

- One way of assessing the accuracy of any unbiased estimate is by looking at its variance. **The lower the variance of an unbiased estimate, the better.**

- If the samples are iid, then the variance of simple MC will be $\mathbb{V}[\hat{I}_K] = \frac{1}{K}\mathbb{V}[f(x_1)]$

Beyond simple MC: importance sampling

- One way of assessing the accuracy of any unbiased estimate is by looking at its variance. **The lower the variance of an unbiased estimate, the better.**
- If the samples are iid, then the variance of simple MC will be $\mathbb{V}[\hat{I}_K] = \frac{1}{K}\mathbb{V}[f(x_1)]$
- So the variance gets smaller and smaller at speed *1/K*, that's good news!
- But it can still be pretty big, depending on the value of $\mathbb{V}[f(x_1)]$
- **Can we reduce the variance of the MC estimate?**

Beyond simple MC: importance sampling

- Key idea: rather than sampling from $p(x)$, we're going to sample from another density $q(x)$ that we'll call a **proposal**

$$x_1, \ldots, .x_K \sim q$$

- But the integral is an expected value with respect to *p*. **Can we turn an expected value with respect to *p* into an expected value with respect to *q*?**

Beyond simple MC: importance sampling

- Key idea: rather than sampling from $p(x)$, we're going to sample from another density $q(x)$ that we'll call a **proposal**

$$x_1, \ldots, .x_K \sim q$$

- But the integral is an expected value with respect to *p*. **Can we turn an expected value with respect to *p* into an expected value with respect to *q*? Yes!**

$$I = \int_\Omega f(x)p(x)dx$$

$$= \int_\Omega \frac{f(x)p(x)}{q(x)} q(x)dx \approx \frac{1}{K}\sum_{k=1}^{K} \frac{f(x_k)p(x_k)}{q(x_k)} = \hat{I}_K^q.$$

Beyond simple MC: importance sampling

- This new estimate $\hat{I}_K^q$ is called an **importance sampling** estimate.
- Now, is $\hat{I}_K^q$ any better than the simple MC estimate?

Beyond simple MC: importance sampling

- This new estimate $\hat{I}_K^q$ is called an **importance sampling** estimate.
- Now, is $\hat{I}_K^q$ any better than the simple MC estimate? Of course, this depends of the choice of the proposal q…
- It's clear that $\hat{I}_K^q$ will aslo be unbiased, consistent, and asymptotically normal.
- The variance will be $\mathbb{V}_{x\sim q}[f(x)p(x)/q(x)]/K$

Beyond simple MC: importance sampling

- This new estimate $\hat{I}_K^q$ is called an **importance sampling** estimate.
- Now, is $\hat{I}_K^q$ any better than the simple MC estimate? Of course, this depends of the choice of the proposal q…
- It's clear that $\hat{I}_K^q$ will aslo be unbiased, consistent, and asymptotically normal.
- The variance will be $\mathbb{V}_{x\sim q}[f(x)p(x)/q(x)]/K$
- For $q^*(x) \propto f(x)p(x)$, **the variance will be exaclty zero!**
- **That seems a bit too good to be true… What's the catch?**

The optimal importance sampling proposal

- For $q^*(x) \propto f(x)p(x)$, **the variance will be exactly zero!**
- **That seems a bit too good to be true… What's the catch?**

$$q^*(x) = \frac{f(x)p(x)}{\int f(x)p(x)dx} = \frac{f(x)p(x)}{I}$$

… and I is precisely the thing we want to compute!

The optimal importance sampling proposal

- For $q^*(x) \propto f(x)p(x)$, **the variance will be exactly zero!**
- **That seems a bit too good to be true… What's the catch?**

$$q^*(x) = \frac{f(x)p(x)}{\int f(x)p(x)dx} = \frac{f(x)p(x)}{I}$$

… and I is precisely the thing we want to compute!

- In practice, we won't be able to find this optimal proposal, but this shows that **the improvements of importance sampling can be potentially huge!**
- This simple result is therefore a motivation for looking for good proposals.

Maximum likelihood for DLVM

We want to approximate

$$p_{\boldsymbol{\theta}}(\mathbf{x}_i) = \int_{\mathbb{R}^d} p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}) p(\mathbf{z}) \mathbf{z}.$$

Idea: use importance sampling! Let $\mathbf{z}_{i1}, \ldots, \mathbf{z}_{iK}$ follow some proposal q_i:

$$\int_{\mathbb{R}^d} p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}) p(\mathbf{z}_{ik}) d\mathbf{z} \approx \frac{1}{K} \sum_{k=1}^{K} \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}_{ik}) p(\mathbf{z})}{q_i(\mathbf{z}_{ik})}$$

Let's say that we want to choose our **proposal in a parametric family** $(\Psi(\cdot|\kappa))_{\kappa \in \mathcal{K}}$ over $\mathbb{R}^d$ (e.g. Gaussians).

Maximum likelihood for DLVM

We want to approximate

$$p_{\boldsymbol{\theta}}(\mathbf{x}_i) = \int_{\mathbb{R}^d} p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}) p(\mathbf{z}) \mathbf{z}.$$

Idea: use importance sampling! Let $\mathbf{z}_{i1}, \ldots, \mathbf{z}_{iK}$ follow some proposal q_i:

$$\int_{\mathbb{R}^d} p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}) p(\mathbf{z}_{ik}) d\mathbf{z} \approx \frac{1}{K} \sum_{k=1}^{K} \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}_{ik}) p(\mathbf{z})}{q_i(\mathbf{z}_{ik})}$$

Let's say that we want to choose our **proposal in a parametric family** $(\Psi(\cdot|\kappa))_{\kappa \in \mathcal{K}}$ over $\mathbb{R}^d$ (e.g. Gaussians).

Problem: we need to choose n **proposals** $q_1, \ldots, q_n$ (and n is usually large in deep learning...).

Maximum likelihood for DLVMs: choosing proposals

$$\int_{\mathbb{R}^d} p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}) p(\mathbf{z}_{ik}) d\mathbf{z} \approx \frac{1}{K} \sum_{k=1}^{K} \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}_{ik}) p(\mathbf{z})}{q_i(\mathbf{z}_{ik})}$$

What would be the optimal, zero-variance choices for $q_1, \ldots, q_n$?

Maximum likelihood for DLVMs: choosing proposals

$$\int_{\mathbb{R}^d} p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}) p(\mathbf{z}_{ik}) d\mathbf{z} \approx \frac{1}{K} \sum_{k=1}^{K} \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}_{ik}) p(\mathbf{z})}{q_i(\mathbf{z}_{ik})}$$

What would be the optimal, zero-variance choices for $q_1, \ldots, q_n$?

$$q_i(z) = p(z|x_i)$$

Maximum likelihood for DLVM

A solution: Amortised variational inference, **all the q_i will be defined together via a neural net!**

Rationale: q_i needs to depends on $\mathbf{x}_i$, so we'll define it as a **conditional distribution parametrised by γ:**

$$q_i(\mathbf{z}) = q_{\boldsymbol{\gamma}}(\mathbf{z}|\mathbf{x}_i).$$

How to parametrise this conditional distribution? The key idea is that **its parameters are the output of a neural net $g_{\boldsymbol{\gamma}}$:**

$$q_{\boldsymbol{\gamma}}(\mathbf{z}|\mathbf{x}_i) = \Psi(\mathbf{z}|g_{\boldsymbol{\gamma}}(\mathbf{x}_i)).$$

This neural net is called the **inference network** or **encoder**.

Maximum likelihood for DLVM

All of this leads to the following approximation of the likelihood

$$\ell(\boldsymbol{\theta}) \approx \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}_{i1},\ldots,\mathbf{z}_{iK} \sim q_{\boldsymbol{\gamma}}(\mathbf{z}|\mathbf{x}_i)} \left[\log \frac{1}{K} \sum_{k=1}^{K} \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_{ik}) p(\mathbf{z}_{ik})}{q_{\boldsymbol{\gamma}}(\mathbf{z}_{ik}|\mathbf{x}_i)} \right] = \mathcal{L}_K(\boldsymbol{\theta}, \boldsymbol{\gamma}).$$

Rather than maximising $\ell(\boldsymbol{\theta})$, **we'll maximise $\mathcal{L}_K(\theta, \gamma)$ using SGD** and the reparametrisation trick. But does it make sense to do that?

$$\mathbb{P}(\theta|X)$$

Maximum likelihood for DLVM

All of this leads to the following approximation of the likelihood

$$\ell(\boldsymbol{\theta}) \approx \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}_{i1},\ldots,\mathbf{z}_{iK}\sim q_{\boldsymbol{\gamma}}(\mathbf{z}|\mathbf{x}_i)} \left[\log \frac{1}{K} \sum_{k=1}^{K} \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_{ik}) p(\mathbf{z}_{ik})}{q_{\boldsymbol{\gamma}}(\mathbf{z}_{ik}|\mathbf{x}_i)} \right] = \mathcal{L}_K(\boldsymbol{\theta}, \boldsymbol{\gamma}).$$

Rather than maximising $\ell(\boldsymbol{\theta})$, **we'll maximise $\mathcal{L}_K(\theta, \gamma)$ using SGD** and the reparametrisation trick. But does it make sense to do that?

It does make sense! For several reasons:

- $\mathcal{L}_K(\boldsymbol{\theta}, \boldsymbol{\gamma})$ is a **lower bound of $\ell(\theta)$** (exercise !)
- The bounds get **tighter and tighter!**

$$\mathcal{L}_1(\boldsymbol{\theta}, \boldsymbol{\gamma}) \leq \mathcal{L}_2(\boldsymbol{\theta}, \boldsymbol{\gamma}) \leq \ldots \leq \mathcal{L}_K(\boldsymbol{\theta}, \boldsymbol{\gamma}) \xrightarrow[K\to\infty]{} \ell(\boldsymbol{\theta}).$$

$\mathcal{L}_K(\boldsymbol{\theta}, \boldsymbol{\gamma})$ is called the **importance weighted autoencoder (IWAE)** bound, and was introduced by Burda et al. (2016).

What about VAEs?

The VAE bound of Kingma & Welling (2014) and Rezende et al. (2014) is actually $\mathcal{L}_1(\boldsymbol{\theta}, \boldsymbol{\gamma})$, which is the loosest bound!

The VAE bound can be interestingly rewritten

$$\mathcal{L}_1(\boldsymbol{\theta}, \boldsymbol{\gamma}) = \ell(\boldsymbol{\theta}) - \mathrm{KL}\left(\prod_{i=1}^{n} q_{\boldsymbol{\gamma}}(\mathbf{z}_i|\mathbf{x}_i) \middle|\middle| \prod_{i=1}^{n} p_{\boldsymbol{\theta}}(\mathbf{z}_i|\mathbf{x}_i) \right).$$

which means that, for a given $\boldsymbol{\theta}$, **the optimal $q_{\boldsymbol{\gamma}}(\mathbf{z}_i|\mathbf{x}_i)$ will be as close as possible (in a KL sense) to the true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}_i|\mathbf{x}_i)$.**

Concrete consequence: after training, **we may interpret the $q_{\boldsymbol{\gamma}}(\mathbf{z}_i|\mathbf{x}_i)$ as an (approachable) approximation of the (intractable) $p_{\boldsymbol{\theta}}(\mathbf{z}_i|\mathbf{x}_i)$.**

What about VAEs?

Is it still true when $K > 1$? **Kind of, but it gets more complicated.** Domke & Sheldon (2019) showed that, when $K \to \infty$, the the "closeness" is no longer in KL sense but in the sense of the χ divergence.

$$\mathrm{P}(\theta|X)$$

What about VAEs?

The VAE bound of Kingma & Welling (2014) and Rezende et al. (2014) is actually $\mathcal{L}_1(\boldsymbol{\theta}, \boldsymbol{\gamma})$, which is the loosest bound!

The VAE bound can be interestingly rewritten

$$\mathcal{L}_1(\boldsymbol{\theta}, \boldsymbol{\gamma}) = \ell(\boldsymbol{\theta}) - \mathrm{KL}\left(\prod_{i=1}^{n} q_{\boldsymbol{\gamma}}(\mathbf{z}_i|\mathbf{x}_i) \middle|\middle| \prod_{i=1}^{n} p_{\boldsymbol{\theta}}(\mathbf{z}_i|\mathbf{x}_i) \right).$$

which means that, for a given $\boldsymbol{\theta}$, **the optimal $q_{\boldsymbol{\gamma}}(\mathbf{z}_i|\mathbf{x}_i)$ will be as close as possible (in a KL sense) to the true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}_i|\mathbf{x}_i)$.**

Concrete consequence: after training, **we may interpret the $q_{\boldsymbol{\gamma}}(\mathbf{z}_i|\mathbf{x}_i)$ as an (approachable) approximation of the (intractable) $p_{\boldsymbol{\theta}}(\mathbf{z}_i|\mathbf{x}_i)$.**

What about VAEs?

The VAE bound can be also rewritten

$$\mathcal{L}_1(\boldsymbol{\theta}, \boldsymbol{\gamma}) = \mathbb{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\gamma}}(\mathbf{z}_i|\mathbf{x}_i)}[p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)] - \mathrm{KL}\left(\prod_{i=1}^{n} q_{\boldsymbol{\gamma}}(\mathbf{z}_i|\mathbf{x}_i) \middle|\middle| \prod_{i=1}^{n} p_{\boldsymbol{\theta}}(\mathbf{z}_i)\right).$$

- $\mathbf{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\gamma}}(\mathbf{z}_i|\mathbf{x}_i)}[p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)]$ can be interpreted as (the opposite of) a **reconstruction error.**
- the **KL between the approximate posterior and the prior** can be interpreted as a regulariser. If $q_{\boldsymbol{\gamma}}(\mathbf{z}|\mathbf{x})$ is Gaussian, then this can be computed in **closed-form.**

This version of the bound resembles the opposite of the loss of a **KL-regularised autoencoder**, hence the name **variational autoencoder (VAE)**, coined by Kingma and Welling (ICLR 2014).