

# Deep latent variable models and missing data imputation

**Pierre-Alexandre Mattei**

**IT University of Copenhagen**

`http://pamattei.github.io/`

`@pamattei`

*ProbAI summer school*

Joint work with **Jes Frellsen** (ITU Copenhagen)

IT UNIVERSITY OF CPH

# Overview of talk

A short introduction to deep learning

Deep latent variable models

Approximate maximum likelihood for DLVMs

Connections with nonparametric mixtures

Handling missing data in DLVMs

# But actually, what is deep learning?

Deep learning is a **general framework for function approximation**.

# But actually, what is deep learning?

Deep learning is a **general framework for function approximation**.

It uses parametric approximators called **neural networks**, which are compositions of some tunable **affine functions**  $f_1, \dots, f_L$  with a simple fixed **nonlinear function**  $\sigma$ :

$$F(\mathbf{x}) = f_1 \circ \sigma \circ f_2 \circ \dots \circ \sigma \circ f_L(\mathbf{x})$$

These functions are called **layers**. The nonlinearity  $\sigma$  is usually called the **activation function**.

# But actually, what is deep learning?

Deep learning is a **general framework for function approximation**.

It uses parametric approximators called **neural networks**, which are compositions of some tunable **affine functions**  $f_1, \dots, f_L$  with a simple fixed **nonlinear function**  $\sigma$ :

$$F(\mathbf{x}) = f_1 \circ \sigma \circ f_2 \circ \dots \circ \sigma \circ f_L(\mathbf{x})$$

These functions are called **layers**. The nonlinearity  $\sigma$  is usually called the **activation function**.

The derivatives of  $F$  with respect to the tunable parameters can be computed using the chain rule via the **backpropagation algorithm**.

# A glimpse at the zoology of layers

The simplest kind of affine layer is called a **fully connected layer**:

$$f_l(\mathbf{x}) = \mathbf{W}_l \mathbf{x} + \mathbf{b}_l,$$

where  $\mathbf{W}_l$  and  $\mathbf{b}_l$  are tunable parameters.

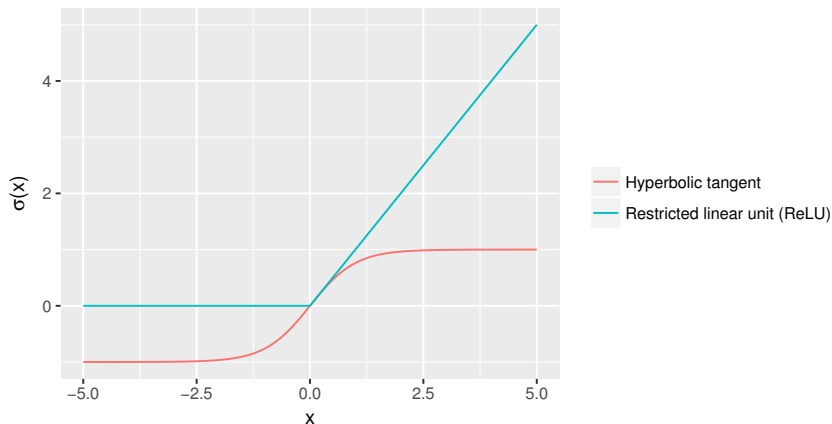
# A glimpse at the zoology of layers

The simplest kind of affine layer is called a **fully connected layer**:

$$f_l(\mathbf{x}) = \mathbf{W}_l \mathbf{x} + \mathbf{b}_l,$$

where  $\mathbf{W}_l$  and  $\mathbf{b}_l$  are tunable parameters.

The activation function  $\sigma$  is usually a **univariate fixed function** applied elementwise. Here are two popular choices:



# Why is it convenient to compose affine functions?

- Neural nets are powerful approximators: **any continuous function can be arbitrarily well approximated** on a compact using a three-layer fully connected network  $F = f_1 \circ \sigma \circ f_2$  (**universal approximation theorem**, Leshno et al., 1993). The conditions are that  $\sigma$  is not a polynomial and that **the network can be arbitrarily wide**.



# Why is it convenient to compose affine functions?

- Neural nets are powerful approximators: **any continuous function can be arbitrarily well approximated** on a compact using a three-layer fully connected network  $F = f_1 \circ \sigma \circ f_2$  (**universal approximation theorem**, Leshno et al., 1993). The conditions are that  $\sigma$  is not a polynomial and that **the network can be arbitrarily wide**.
- There are similar results for **very thin but arbitrarily deep networks** (Lin & Jegelka, 2018).

# Why is it convenient to compose affine functions?

- Neural nets are powerful approximators: **any continuous function can be arbitrarily well approximated** on a compact using a three-layer fully connected network  $F = f_1 \circ \sigma \circ f_2$  (**universal approximation theorem**, Leshno et al., 1993). The conditions are that  $\sigma$  is not a polynomial and that **the network can be arbitrarily wide**.
- There are similar results for **very thin but arbitrarily deep networks** (Lin & Jegelka, 2018).
- Some **prior knowledge** can be distilled into the **architecture** (i.e. the type of affine functions/activations) of the network. For example, **convolutional neural networks** (CNNs, LeCun et al., 1989) leverage the fact that local information plays an important role in images/sound/sequence data. In that case, the affine functions are convolution operators with some learnt filters.

# Why is it convenient to compose affine functions?

- Often, this prior knowledge can be based on **known symmetries**, leading to deep architectures that are **equivariant or invariant to the action of some group** (see e.g. the work of Taco Cohen or Stéphane Mallat). This is useful when dealing with images, sound, molecules...

# Why is it convenient to compose affine functions?

- Often, this prior knowledge can be based on **known symmetries**, leading to deep architectures that are **equivariant or invariant to the action of some group** (see e.g. the work of Taco Cohen or Stéphane Mallat). This is useful when dealing with images, sound, molecules...
- The layers can capture **hierarchical representations** of the data, sometimes (almost) explicitly (e.g. the capsules of Hinton et al., 2018).

# Why is it convenient to compose affine functions?

- Often, this prior knowledge can be based on **known symmetries**, leading to deep architectures that are **equivariant or invariant to the action of some group** (see e.g. the work of Taco Cohen or Stéphane Mallat). This is useful when dealing with images, sound, molecules...
- The layers can capture **hierarchical representations** of the data, sometimes (almost) explicitly (e.g. the capsules of Hinton et al., 2018).
- When the neural network parametrises a regression function, empirical evidence shows that **adding more layers leads to better out-of-sample behaviour**. Roughly, this means that adding more layers is a way of increasing the complexity of statistical models without paying a large overfitting price: there is a **regularisation-by-depth effect**.

# A simple example: nonlinear regression with a multilayer perceptron (MLP)

We want to perform regression on a data set

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n) \in \mathbb{R}^p \times \mathbb{R}.$$

# A simple example: nonlinear regression with a multilayer perceptron (MLP)

We want to perform regression on a data set

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n) \in \mathbb{R}^p \times \mathbb{R}.$$

We can model the regression function using a **multilayer perceptron (MLP)**: two connected layers with an hyperbolic tangent in-between:

$$\forall i \leq n, y_i = F(\mathbf{x}_i) + \varepsilon_i = \mathbf{W}_1 \tanh(\mathbf{W}_0 \mathbf{x}_i + \mathbf{b}_0) + \mathbf{b}_1 + \varepsilon_i.$$

The coordinates of the intermediate representation  $\mathbf{W}_0 \mathbf{x}_i + \mathbf{b}_0$  are called **hidden units**.

# A simple example: nonlinear regression with a multilayer perceptron (MLP)

We want to perform regression on a data set

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n) \in \mathbb{R}^p \times \mathbb{R}.$$

We can model the regression function using a **multilayer perceptron (MLP)**: two connected layers with an hyperbolic tangent in-between:

$$\forall i \leq n, y_i = F(\mathbf{x}_i) + \varepsilon_i = \mathbf{W}_1 \tanh(\mathbf{W}_0 \mathbf{x}_i + \mathbf{b}_0) + \mathbf{b}_1 + \varepsilon_i.$$

The coordinates of the intermediate representation  $\mathbf{W}_0 \mathbf{x}_i + \mathbf{b}_0$  are called **hidden units**.

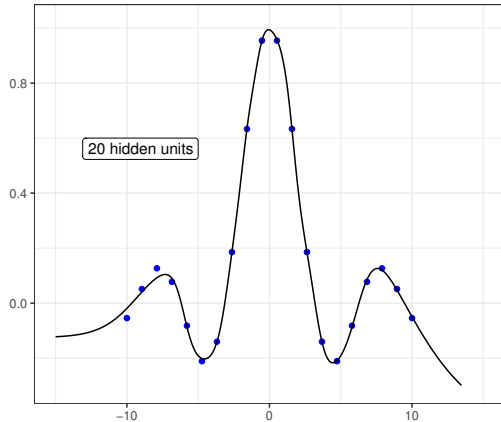
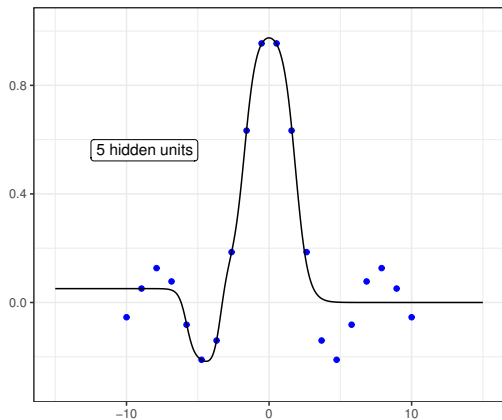
If we assume that the noise is Gaussian, then we can find the maximum likelihood estimates of  $\mathbf{W}_1, \mathbf{W}_0, \mathbf{b}_1, \mathbf{b}_0$  by **minimising the squared error using gradient descent**. Gradients are computed via **backpropagation**.



# A simple example: nonlinear regression with a multilayer perceptron (MLP)

$$\forall i \leq n, y_i = F(\mathbf{x}_i) + \varepsilon_i = \mathbf{W}_1 \tanh(\mathbf{W}_0 \mathbf{x}_i + \mathbf{b}_0) + \mathbf{b}_1 + \varepsilon_i,$$

Let's try to recover the function  $\sin(x)/x$  using 20 samples:



# Overview of talk

A short introduction to deep learning

Deep latent variable models

Approximate maximum likelihood for DLVMs

Connections with nonparametric mixtures

Handling missing data in DLVMs

# Continuous latent variable models

Let's start with some i.i.d. data  $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathcal{X}$ .

A **generative model**  $p(\mathbf{x})$  “describes a process that is assumed to give rise to some data” (D. MacKay).

In a **continuous latent variable model** we assume that there is an **unobserved random variable**  $\mathbf{z} \in \mathbb{R}^d$ . Usually,  $d$  is smaller than the dimensionality of the data, and we can think of  $\mathbf{z}$  as a **code** summarizing multivariate data  $\mathbf{x}$ .

**A classic example: factor analysis.** The generative process is:

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

# Deep latent variable models (DLVMs)

Deep latent variable models combine the approximation abilities of deep neural networks and the statistical foundations of generative models.

Independently invented by Kingma and Welling (2014) as **variational autoencoders (VAEs)**, and Rezende, Mohamed, and Wierstra (2014) as **deep latent Gaussian models**.

---

## Stochastic Backpropagation and Approximate Inference in Deep Generative Models

---

Daniilo J. Rezende, Shakir Mohamed, Daan Wierstra  
(danilor, shakir, daanv@google.com  
Google DeepMind, London)

### Abstract

We marry ideas from deep neural networks and approximate Bayesian inference to derive a generalised class of deep, directed generative models, endowed with a new algorithm for scalable inference and learning. Our algorithm introduces a recognition model to represent an approximate posterior distribution and uses this for optimisation of a variational lower bound. We develop stochastic back-

propagation, but in most cases, efficient inference algorithms have remained elusive. These efforts, combined with the demand for accurate probabilistic inferences and fast simulation, lead us to seek generative models that are i) *deep*, since hierarchical architectures allow us to capture complex structure in the data, ii) *allow for fast sampling* of fantasy data from the inferred model, and iii) are computationally *tractable and scalable* to high-dimensional data.

---

## Auto-Encoding Variational Bayes

---

Diederik P. Kingma  
Machine Learning Group  
Universiteit van Amsterdam  
dpkingma@gmail.com

Max Welling  
Machine Learning Group  
Universiteit van Amsterdam  
welling.max@gmail.com

### Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference

Also very close to **generative adversarial networks (GANs)** (Goodfellow et al., 2014)

# Deep latent variable models (DLVMs)

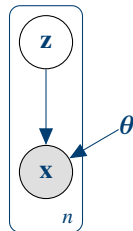
(Kingma and Welling, 2014, Rezende, Mohamed & Wierstra (2014))

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

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

where

- $\mathbf{z} \in \mathbb{R}^d$  is the **latent** variable,
- $\mathbf{x} \in \mathcal{X}$  is the **observed** variable.



# Deep latent variable models (DLVMs)

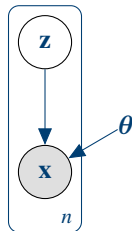
(Kingma and Welling, 2014, Rezende, Mohamed & Wierstra (2014))

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

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

where

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



# The role of the prior

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

# The role of the prior

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

More complex, **learnable priors** have also been considered. For example, in Harchaoui et al. (2018), we looked into mixtures of  $K$  Gaussians:

$$p(\mathbf{z}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

where the parameters  $\pi_1, \dots, \pi_K, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K$  are learned.

Beyond the fact that this prior is more general, it leads to a **DLVM that's fit for clustering purposes**.



# The role of the observation model

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

**Its parameters are the output of the decoder.**

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

# The role of the observation model

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

**Its parameters are the output of the decoder.**

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

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

# The role of the observation model

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

**Its parameters are the output of the decoder.**

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

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

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

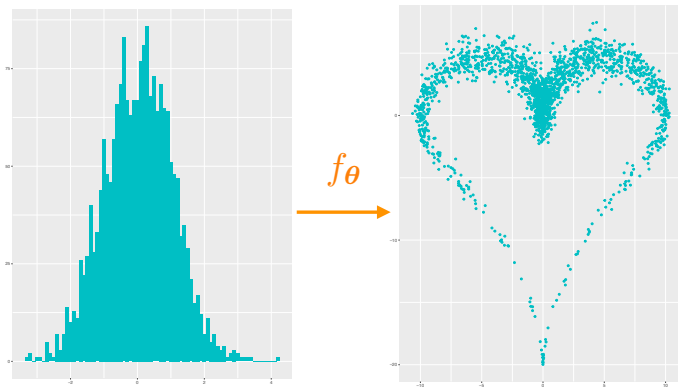
# The role of the decoder

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

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

Simple non-linear decoder ( $d = 1, p = 2$ ):  $f_{\theta}(z) = \mu_{\theta}(z), \Sigma_{\theta}(z)$  with, for all  $z \in \mathbb{R}$ ,

$$\mu_{\theta}(z) = (10 \sin(z)^3, 10 \cos(z) - 10 \cos(z)^4), \quad \Sigma_{\theta}(z) = \text{Diag} \left( \left( \frac{\sin(z)}{3z} \right)^2, \left( \frac{\sin(z)}{z} \right)^2 \right).$$



# DLVMs applications: density estimation on MNIST

Rezende, Mohamed & Wierstra (2014)

0	2	2	3	8	6	7	3	8	8
9	0	5	5	0	9	7	8	4	8
4	6	3	2	4	1	7	1	7	7
5	1	8	4	8	6	6	5	4	9
3	3	0	6	1	3	2	6	2	3
6	4	5	0	1	1	4	5	8	1
7	8	3	7	9	7	1	6	7	9
0	0	4	7	3	3	1	3	2	1
3	3	9	3	6	9	8	7	8	6
2	4	8	4	9	5	1	6	8	8

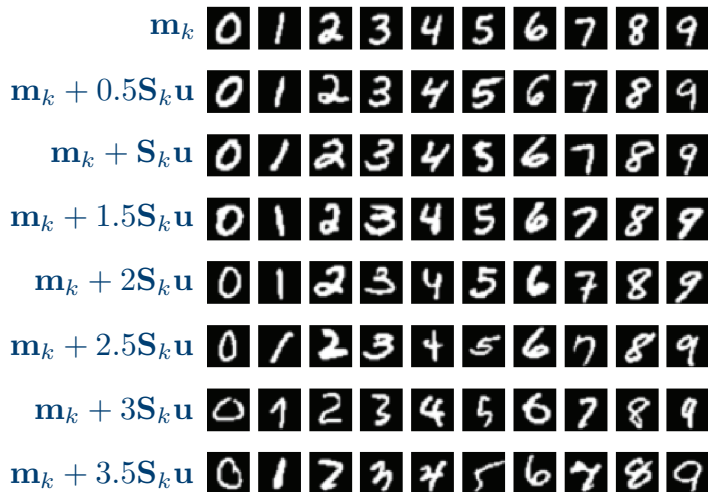
Training data

0	6	9	0	5	7	0	8	0	8
1	5	7	6	6	9	5	1	8	0
4	7	4	5	5	4	0	4	4	9
2	9	7	7	0	9	0	9	0	8
6	5	4	0	0	9	9	2	2	8
0	9	5	6	1	5	0	7	7	6
6	6	2	9	7	6	9	4	0	9
2	3	1	8	7	1	5	4	4	0
1	2	5	7	6	9	9	5	3	7
6	2	3	8	7	9	0	9	4	8

Model samples

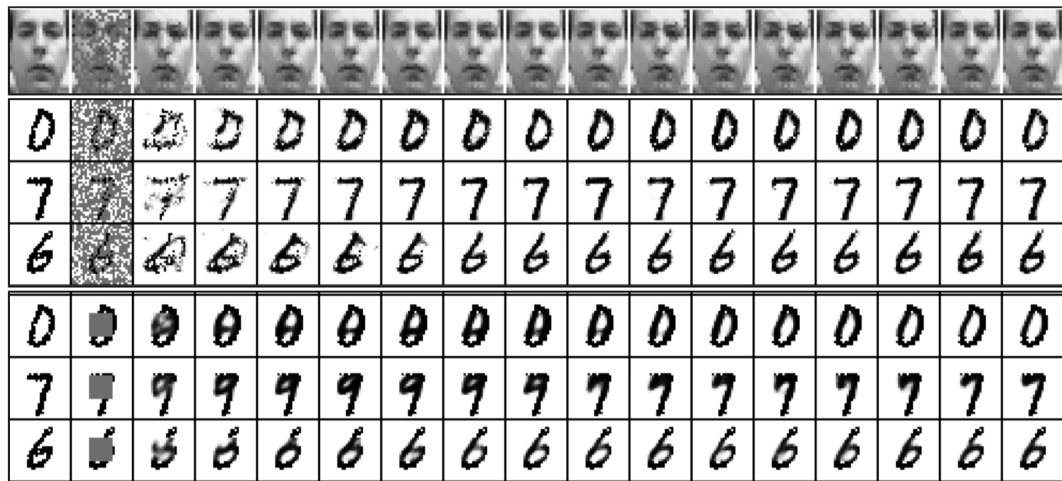
# DLVMs applications: clustering on MNIST

Harchaoui, Mattei, Bouveyron & Almansa (2018)



# DLVMs applications: Data imputation

Rezende, Mohamed & Wierstra (2014)



# DLVMs applications: Data imputation

Mattei & Frelsen (2019)

31/6613\13145019545303754  
31/6613\13145019545303754

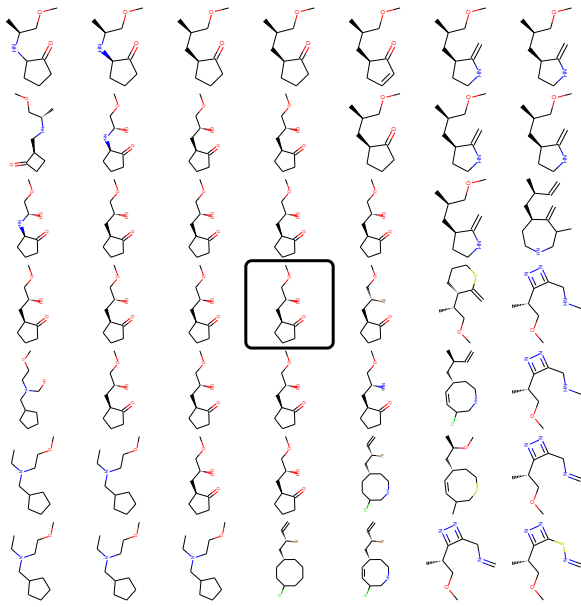
2207089887892426203882917  
2207089887892426203882917

3 9 5 5 5 5 3 3 3 5 3 5 5 3 3 3 3 5 3 5 3  
8  
1 7



# DLVMs applications: Molecular design

Kusner, Paige, and Hernández-Lobato (2017)



# DLVMs applications: Reinforcement learning

Ha & Schmidhuber (2018)

# DLVMs applications: Reinforcement learning

Ha & Schmidhuber (2018)

# Overview of talk

A short introduction to deep learning

Deep latent variable models

Approximate maximum likelihood for DLVMs

Connections with nonparametric mixtures

Handling missing data in DLVMs

# Learning DLVMs

Kingma & Welling (2014), Rezende et al. (2014)

Given a data matrix  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top \in \mathcal{X}^n$ , 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

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

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

:

# Learning DLVMs

Kingma & Welling (2014), Rezende et al. (2014)

Given a data matrix  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top \in \mathcal{X}^n$ , 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

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

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  $p_{\boldsymbol{\theta}}(\mathbf{x} | \mathbf{z})$ :

- $p_{\boldsymbol{\theta}}(\mathbf{x})$  is **intractable** rendering **MLE intractable**

# Learning DLVMs

Kingma & Welling (2014), Rezende et al. (2014)

Given a data matrix  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top \in \mathcal{X}^n$ , 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

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

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  $p_{\boldsymbol{\theta}}(\mathbf{x} | \mathbf{z})$ :

- $p_{\boldsymbol{\theta}}(\mathbf{x})$  is **intractable** rendering **MLE intractable**
- $p_{\boldsymbol{\theta}}(\mathbf{z} | \mathbf{x})$  is **intractable** rendering **EM intractable**

# Learning DLVMs

Kingma & Welling (2014), Rezende et al. (2014)

Given a data matrix  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top \in \mathcal{X}^n$ , 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

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

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  $p_{\boldsymbol{\theta}}(\mathbf{x} | \mathbf{z})$ :

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



# The solution: amortised variational inference

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)**.

# The solution: amortised variational inference

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 solution: amortised variational inference

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

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

## *Aparté:* Monte Carlo and importance sampling

Let's say we want to estimate an integral of the form

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

where  $f \geq 0$  and  $p$  is a density over a space  $\Omega$ .

# Aparté: Monte Carlo and importance sampling

Let's say we want to estimate an integral of the form

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

where  $f \geq 0$  and  $p$  is a density over a space  $\Omega$ .

**Simple Monte Carlo estimate:** We sample  $x_1, \dots, x_K \sim p$  and approximate

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

A few properties:

$$\hat{I}_K \xrightarrow{a.s.} I, \quad \mathbb{E}[\hat{I}_K] = I, \quad \mathbb{V}[\hat{I}_K] = \frac{1}{K} \mathbb{V}[f(x_1)],$$

which sounds nice, but **the variance may be very large.**

## Aparté: Monte Carlo and importance sampling

**Importance sampling** tries to improve this estimate by **sampling**  $x_1, \dots, x_K$  **from another density  $q$  rather than  $p$** . The trick is the following:

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

# Aperté: Monte Carlo and importance sampling

**Importance sampling** tries to improve this estimate by **sampling**  $x_1, \dots, x_K$  **from another density  $q$  rather than  $p$** . The trick is the following:

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

As before, the estimate is **consistent and unbiased** (under the condition that  $q$  has heavier tails than  $p$ ). **What about the variance?**

# Aperté: Monte Carlo and importance sampling

**Importance sampling** tries to improve this estimate by **sampling**  $x_1, \dots, x_K$  **from another density  $q$  rather than  $p$** . The trick is the following:

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

As before, the estimate is **consistent and unbiased** (under the condition that  $q$  has heavier tails than  $p$ ). **What about the variance?**

If we choose  $q^*(x) \propto f(x)p(x)$ , which means  $q^*(x) = f(x)p(x)/I$ , then  $\hat{I}_K^{q^*}$  **has zero variance!** But we can't do that because we don't know  $I$ ...



# Aparté: Monte Carlo and importance sampling

**Importance sampling** tries to improve this estimate by **sampling**  $x_1, \dots, x_K$  **from another density  $q$  rather than  $p$** . The trick is the following:

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

As before, the estimate is **consistent and unbiased** (under the condition that  $q$  has heavier tails than  $p$ ). **What about the variance?**

If we choose  $q^*(x) \propto f(x)p(x)$ , which means  $q^*(x) = f(x)p(x)/I$ , then  $\hat{I}_K^{q^*}$  **has zero variance!** But we can't do that because we don't know  $I$ ...

However, this still means that **importance sampling with a good  $q$  will work much better than simple MC.**

# Back to DLVMs and their likelihood

We want to approximate

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

# Back to DLVMs and their likelihood

We want to approximate

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

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

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

# Back to DLVMs and their likelihood

We want to approximate

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

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

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

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).

# Back to DLVMs and their likelihood

We want to approximate

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

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

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

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**  $q_1, \dots, q_n$  (and  $n$  is usually large in deep learning...).

# Back to DLVMs and their likelihood

We want to approximate

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

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

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

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**  $q_1, \dots, q_n$  (and  $n$  is usually large in deep learning...).

**Exercise:** What would be the optimal, zero-variance choices for  $q_1, \dots, q_n$ ?

# Amortised variational inference

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

# Amortised variational inference

**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  $\gamma$ :**

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



# Amortised variational inference

**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  $\gamma$ :**

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

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

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

# Amortised variational inference

**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  $\gamma$ :**

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

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

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

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

# Amortised variational inference

All of this leads to the following approximation of the likelihood

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

# Amortised variational inference

All of this leads to the following approximation of the likelihood

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

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

# Amortised variational inference

All of this leads to the following approximation of the likelihood

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

Rather than maximising  $\ell(\boldsymbol{\theta})$ , **we'll maximise  $\mathcal{L}_K(\boldsymbol{\theta}, \boldsymbol{\gamma})$  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  $\ell(\boldsymbol{\theta})$**  (exercise !)
- The bounds get **tighter and tighter!**

$$\mathcal{L}_1(\boldsymbol{\theta}, \boldsymbol{\gamma}) \leq \mathcal{L}_2(\boldsymbol{\theta}, \boldsymbol{\gamma}) \leq \dots \leq \mathcal{L}_K(\boldsymbol{\theta}, \boldsymbol{\gamma}) \xrightarrow{K \rightarrow \infty} \ell(\boldsymbol{\theta}).$$

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

# What about the VAE?

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

# What about the VAE?

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

The VAE bound can be interestingly rewritten

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

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

# What about the VAE?

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

The VAE bound can be interestingly rewritten

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

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

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



# What about the VAE?

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

The VAE bound can be interestingly rewritten

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

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

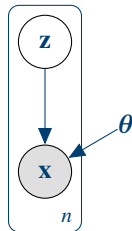
Concrete consequence: after training, **we may interpret the  $q_\gamma(\mathbf{z}_i | \mathbf{x}_i)$  as an (approachable) approximation of the (intractable)  $p_\theta(\mathbf{z}_i | \mathbf{x}_i)$ .**

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

# Let's summarise

DLVMs are flexible latent variable models that **transform low-dimensional codes  $\mathbf{z}$  into parameters of a simple observation model.**

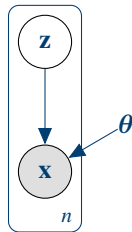
$$\begin{cases} \mathbf{z} \sim p(\mathbf{z}) & \text{(prior)} \\ \mathbf{x} \sim p_{\theta}(\mathbf{x} | \mathbf{z}) = \Phi(\mathbf{x} | f_{\theta}(\mathbf{z})) & \text{(observation model)} \end{cases}$$



# Let's summarise

DLVMs are flexible latent variable models that **transform low-dimensional codes  $\mathbf{z}$  into parameters of a simple observation model.**

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






Training is performed by **maximising a lower bound  $\mathcal{L}_K(\theta, q_{\gamma})$  of the likelihood** using stochastic gradient descent (SGD). This utilises a variational approximation  $q_{\gamma}(\mathbf{z} | \mathbf{x})$  of the posterior distribution  $p_{\theta}(\mathbf{z} | \mathbf{x})$  of the codes.

# A few (semi-open) questions about maximum likelihood for DLVMs


- **We do not maximise the actual likelihood but a lower-bound: does it make sense?** Is the bound tight?
- **How to perform SGD on such lower bounds?**
- **Was it sensible to maximise the likelihood in the first place?**

A few very recent papers tried to address these questions, and contained quite related results:

-  Dai, Wang, Aston, Hua, and Wipf, *Connections with Robust PCA and the Role of Emergent Sparsity in Variational Autoencoder Models*, JMLR 2018
-  Mattei & Frellsen, *Leveraging the Exact Likelihood of Deep Latent Variable Models*, NeurIPS 2018
-  Rezende & Viola, *Taming VAEs*, arXiv:1810.00597, 2018

# Does it make sense to maximise lower bounds?

The bounds can be viewed as **regularised versions of the likelihood**. In some very simple settings, Dai et al. (2018) show that this regularisation is directly linked to **robust PCA** (à la Candès et al., 2009).

 Dai, Wang, Aston, Hua, and Wipf, *Connections with Robust PCA and the Role of Emergent Sparsity in Variational Autoencoder Models*, JMLR 2018

# How to perform SGD on such lower bounds?

The usual bound is

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

Low-variance estimates of gradients of  $\mathcal{L}(\boldsymbol{\theta}, q_\gamma)$  can be found using **reparametrisation tricks** for a wide variety of choices for the variational family. A nice overview of those tricks is provided in this paper:



Figurnov, Mohamed, and Mnih, *Implicit Reparameterization Gradients*, NeurIPS 2018

# Was it sensible to maximise the likelihood in the first place?

If we see the prior as a mixing distribution, **DLVMs are continuous mixtures of distribution from the observation model:**

$$p_{\theta}(\mathbf{x}) = \int_{\mathbb{R}^d} p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} = \int_{\mathbb{R}^d} \Phi(\mathbf{x}|f_{\theta}(\mathbf{z}))p(\mathbf{z})d\mathbf{z}.$$

# Was it sensible to maximise the likelihood in the first place?

If we see the prior as a mixing distribution, **DLVMs are continuous mixtures of distribution from the observation model:**

$$p_{\theta}(\mathbf{x}) = \int_{\mathbb{R}^d} p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} = \int_{\mathbb{R}^d} \Phi(\mathbf{x}|f_{\theta}(\mathbf{z}))p(\mathbf{z})d\mathbf{z}.$$

But maximum likelihood for **finite Gaussian mixtures** is **ill-posed**: the likelihood function is unbounded and the parameters with infinite likelihood are pretty terrible.

*“Mixtures, like tequila, are inherently evil and should be avoided at all costs” – Larry Wasserman*

Hence the question: **Is maximum likelihood well-posed for DLVMs?**

**No.** We showed that the problem was pretty much the same as with finite mixtures.



# On the boundedness of the likelihood of DLVMs

Mattei & Frelsen (2018)

Consider a DLVM with a  **$p$ -variate Gaussian observation model** where

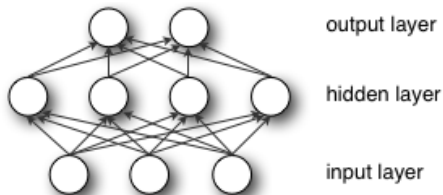
$$\ell(\theta) = \sum_{i=1}^n \log \int_{\mathbb{R}^d} \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_{\theta}(\mathbf{z}), \boldsymbol{\Sigma}_{\theta}(\mathbf{z})) p(\mathbf{z}) \mathbf{z}.$$

Like Kingma and Welling (2014), consider a **MLP decoder** with  $h \in \mathbb{N}^*$  **hidden units** of the form

$$\boldsymbol{\mu}_{\theta}(\mathbf{z}) = \mathbf{V} \tanh(\mathbf{W}\mathbf{z} + \mathbf{a}) + \mathbf{b}$$

$$\boldsymbol{\Sigma}_{\theta}(\mathbf{z}) = \exp(\boldsymbol{\alpha}^{\top} \tanh(\mathbf{W}\mathbf{z} + \mathbf{a}) + \beta) \mathbf{I}_p$$

where  $\theta = (\mathbf{W}, \mathbf{a}, \mathbf{V}, \mathbf{b}, \boldsymbol{\alpha}, \beta)$ .



# On the boundedness of the likelihood of DLVMs

Mattei & Frelsen (2018)

Consider a DLVM with a  $p$ -variate **Gaussian observation model** where

$$\ell(\theta) = \sum_{i=1}^n \log \int_{\mathbb{R}^d} \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_\theta(\mathbf{z}), \boldsymbol{\Sigma}_\theta(\mathbf{z})) p(\mathbf{z}) \mathbf{z}.$$

Like Kingma and Welling (2014), consider a **MLP decoder** with  $h \in \mathbb{N}^*$  **hidden units** of the form

$$\boldsymbol{\mu}_\theta(\mathbf{z}) = \mathbf{V} \tanh(\mathbf{W}\mathbf{z} + \mathbf{a}) + \mathbf{b}$$

$$\boldsymbol{\Sigma}_\theta(\mathbf{z}) = \exp(\boldsymbol{\alpha}^\top \tanh(\mathbf{W}\mathbf{z} + \mathbf{a}) + \beta) \mathbf{I}_p$$

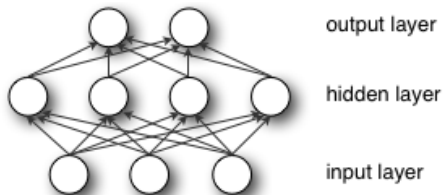
where  $\theta = (\mathbf{W}, \mathbf{a}, \mathbf{V}, \mathbf{b}, \boldsymbol{\alpha}, \beta)$ .

Now consider a subfamily with  $h = 1$  and

$$\boldsymbol{\theta}_k^{(i, \mathbf{w})} = (\alpha_k \mathbf{w}^\top, 0, 0, \mathbf{x}_i, \alpha_k, -\alpha_k),$$

where  $(\alpha_k)_{k \geq 1}$  is a nonnegative real sequence

$$\alpha_k \rightarrow \infty \text{ as } k \rightarrow \infty.$$



# On the boundedness of the likelihood of DLVMs

Mattei & Frelsen (2018)

Consider a DLVM with a  $p$ -variate **Gaussian observation model** where

$$\ell(\theta) = \sum_{i=1}^n \log \int_{\mathbb{R}^d} \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_{\theta}(\mathbf{z}), \boldsymbol{\Sigma}_{\theta}(\mathbf{z})) p(\mathbf{z}) \mathbf{z}.$$

Like Kingma and Welling (2014), consider a **MLP decoder** with  $h \in \mathbb{N}^*$  **hidden units** of the form

$$\boldsymbol{\mu}_{\theta}(\mathbf{z}) = \mathbf{V} \tanh(\mathbf{W}\mathbf{z} + \mathbf{a}) + \mathbf{b}$$

$$\boldsymbol{\Sigma}_{\theta}(\mathbf{z}) = \exp(\boldsymbol{\alpha}^{\top} \tanh(\mathbf{W}\mathbf{z} + \mathbf{a}) + \beta) \mathbf{I}_p$$

$$\boldsymbol{\mu}_{\theta_k^{(i,w)}}(\mathbf{z}) = \mathbf{x}_i$$

$$\boldsymbol{\Sigma}_{\theta_k^{(i,w)}}(\mathbf{z}) = \exp(\alpha_k \tanh(\alpha_k \mathbf{w}^{\top} \mathbf{z}) - \alpha_k) \mathbf{I}_p$$

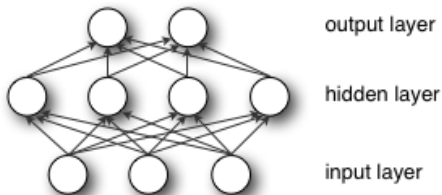
where  $\theta = (\mathbf{W}, \mathbf{a}, \mathbf{V}, \mathbf{b}, \boldsymbol{\alpha}, \beta)$ .

Now consider a subfamily with  $h = 1$  and

$$\theta_k^{(i,w)} = (\alpha_k \mathbf{w}^{\top}, 0, 0, \mathbf{x}_i, \alpha_k, -\alpha_k),$$

where  $(\alpha_k)_{k \geq 1}$  is a nonnegative real sequence

$$\alpha_k \rightarrow \infty \text{ as } k \rightarrow \infty.$$



# ML is ill-posed for a general Gaussian DLVM

Mattei & Frelsen (2018)

## Theorem

*For all  $i \in \{1, \dots, n\}$  and  $\mathbf{w} \in \mathbb{R}^d \setminus \{0\}$ , we have that  $\lim_{k \rightarrow \infty} \ell(\boldsymbol{\theta}_k^{(i, \mathbf{w})}) = \infty$ .*

**Proof main idea:** the contribution  $\log p_{\boldsymbol{\theta}_k^{(i, \mathbf{w})}}(\mathbf{x}_i)$  of the  $i$ -th observation explodes while all other contributions remain bounded below.

# ML is ill-posed for a general Gaussian DLVM

Mattei & Frelsen (2018)

## Theorem

*For all  $i \in \{1, \dots, n\}$  and  $\mathbf{w} \in \mathbb{R}^d \setminus \{0\}$ , we have that  $\lim_{k \rightarrow \infty} \ell(\boldsymbol{\theta}_k^{(i, \mathbf{w})}) = \infty$ .*

**Proof main idea:** the contribution  $\log p_{\boldsymbol{\theta}_k^{(i, \mathbf{w})}}(\mathbf{x}_i)$  of the  $i$ -th observation explodes while all other contributions remain bounded below.

Do these infinite suprema lead to useful generative models?

## Proposition

*For all  $k \in \mathbb{N}^*$ ,  $i \in \{1, \dots, n\}$  and  $\mathbf{w} \in \mathbb{R}^d \setminus \{0\}$ , the distribution  $p_{\boldsymbol{\theta}_k^{(i, \mathbf{w})}}(\mathbf{x}_i)$  is spherically symmetric and unimodal around  $\mathbf{x}_i$ .*

**No, because of the constant mean function.**

# ML is ill-posed for a general Gaussian DLVM

Mattei & Frellsen (2018)

## Theorem

*For all  $i \in \{1, \dots, n\}$  and  $\mathbf{w} \in \mathbb{R}^d \setminus \{0\}$ , we have that  $\lim_{k \rightarrow \infty} \ell \left( \boldsymbol{\theta}_k^{(i, \mathbf{w})} \right) = \infty$ .*

**Proof main idea:** the contribution  $\log p_{\boldsymbol{\theta}_k^{(i, \mathbf{w})}}(\mathbf{x}_i)$  of the  $i$ -th observation explodes while all other contributions remain bounded below.

Do these infinite suprema lead to useful generative models?

## Proposition

*For all  $k \in \mathbb{N}^*$ ,  $i \in \{1, \dots, n\}$  and  $\mathbf{w} \in \mathbb{R}^d \setminus \{0\}$ , the distribution  $p_{\boldsymbol{\theta}_k^{(i, \mathbf{w})}}(\mathbf{x}_i)$  is spherically symmetric and unimodal around  $\mathbf{x}_i$ .*

**No, because of the constant mean function.**

## What about other parametrisations?

- The used MLP is a subfamily.
- Universal approximation abilities of neural networks.

# Tackling the unboundedness of the likelihood

Mattei & Frelsen (2018)

## Proposition

Let  $\xi > 0$ . If the parametrisation of the decoder is such that the image of  $\Sigma_\theta$  is included in

$$S_p^\xi = \{\mathbf{A} \in S_p^+ \mid \min(\text{Sp } \mathbf{A}) \geq \xi\}$$

for all  $\theta$ , then the log-likelihood function is upper bounded by  $-np \log \sqrt{\pi\xi}$

**Note:** Such constraints can be implemented by added  $\xi \mathbf{I}_p$  to  $\Sigma_\theta(\mathbf{z})$ .

# Discrete DLVMs do not suffer from unbounded likelihood

When  $\mathcal{X} = \{0, 1\}^p$ , Bernoulli DLVMs assume that  $(\Phi(\cdot|\boldsymbol{\eta}))_{\boldsymbol{\eta} \in H}$  is the family of  $p$ -variate multivariate Bernoulli distribution (that is, the family of products of  $p$  univariate Bernoulli distributions). In this case, maximum likelihood is well-posed.

## Proposition

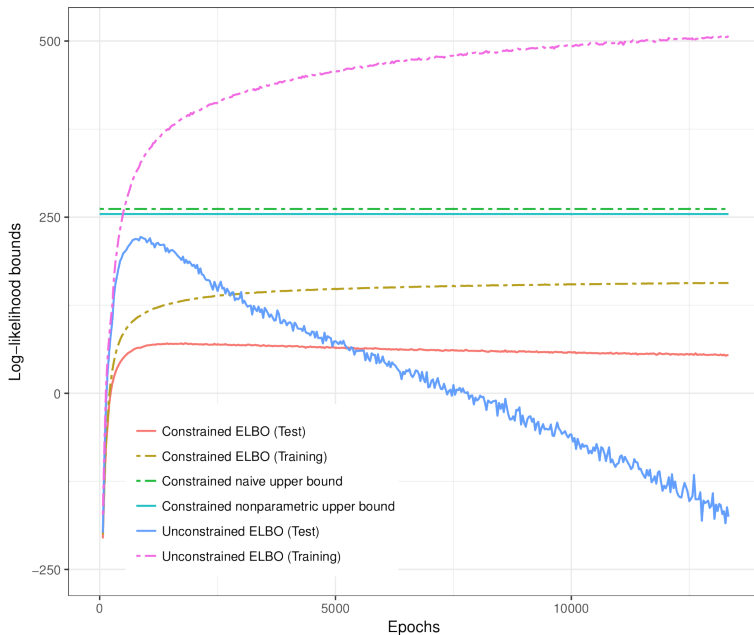
*Given any possible parametrisation, the log-likelihood function of a deep latent model a Bernoulli observation model is everywhere negative.*

## Take-home message

Always regularise when you deal with continuous data!



# Unboundedness for a DLVM with Gaussian observation model for (Brendan) Frey faces



# Overview of talk

A short introduction to deep learning

Deep latent variable models

Approximate maximum likelihood for DLVMs

Connections with nonparametric mixtures

Handling missing data in DLVMs

# Towards data-dependent likelihood upper bounds

When maximum likelihood is well posed, we were able to derive **very simple, data-independent upper bound** of the true likelihood.

# Towards data-dependent likelihood upper bounds

When maximum likelihood is well posed, we were able to derive **very simple, data-independent upper bound** of the true likelihood.

**Is it possible to derive tighter, data-dependent bounds?** This would allow us to sandwich the exact likelihood between a lower bound (the training objective), and an upper bound.

# Towards data-dependent likelihood upper bounds

Mattei & Frellsen (2018)

We can interpret DLVM as **parsimonious submodel** of a **nonparametric mixture** model

$$p_G(\mathbf{x}) = \int_H \Phi(\mathbf{x}|\eta)G(\eta) \quad \ell(G) = \sum_{i=1}^n \log p_G(\mathbf{x}_i).$$

- The model parameter is the **mixing distribution**  $G \in \mathcal{P}$ , where  $\mathcal{P}$  is the set of all probability measures over parameter space  $H$ .
- This is a DLVM, when  $G$  is generatively defined by:  $\mathbf{z} \sim p(\mathbf{z}); \eta = f_\theta(\mathbf{z})$ .
- This is a finite mixture model, when  $G$  has a finite support.

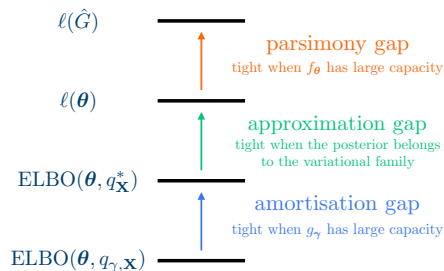
This generalisation **bridges the gap between finite mixtures and DLVMs**.

# Towards data-dependent likelihood upper bounds

Mattei & Frellsen (2018), Cremer et al. (2018)

This gives us an **immediate upper bound on the likelihood for any decoder  $f_\theta$** :

$$\ell(\theta) \leq \max_{G \in \mathcal{P}} \ell(G)$$



# Towards data-dependent likelihood upper bounds

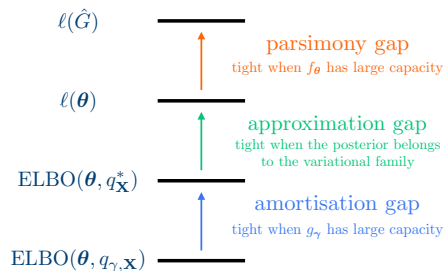
Mattei & Frelsen (2018), Cremer et al. (2018)

This gives us an **immediate upper bound on the likelihood for any decoder  $f_\theta$** :

$$\ell(\theta) \leq \max_{G \in \mathcal{P}} \ell(G)$$

## Theorem

Assume that  $(\Phi(\cdot | \eta))_{\eta \in H}$  is the family of multivariate Bernoulli distributions or Gaussian distributions with the spectral constraint. **The likelihood of the nonparametric mixture model is maximised for a finite mixture model of  $k \leq n$  distributions from the family  $(\Phi(\cdot | \eta))_{\eta \in H}$ .**



# Towards data-dependent likelihood upper bounds

Mattei & Frellsen (2018)

## Theorem (Tightness of the parsimony gap)

Assume that

1.  $(\Phi(\cdot|\boldsymbol{\eta}))_{\boldsymbol{\eta} \in H}$  is a constrained Gaussian or a Bernoulli observation model.
2. The decoder has universal approximation abilities : for any compact  $C \in \mathbb{R}^d$  and continuous function  $f : C \rightarrow H$ , for all  $\varepsilon > 0$ , there exists  $\theta$  such that  $\|f - f_{\theta}\|_{\infty} < \varepsilon$ .

Then, for all  $\varepsilon > 0$ , there exists  $\theta \in \Theta$  such that  $\ell(\hat{G}) \geq \ell(\theta) \geq \ell(\hat{G}) - \varepsilon$ .

**Proof main idea:** split the code space into a compact set made of several parts that will represent the mixture components, and an unbounded set of very small prior mass. The universal approximation property is finally used for this compact set.



# Towards data-dependent likelihood upper bounds

Mattei & Frellsen (2018)

## Theorem (Tightness of the parsimony gap)

Assume that

1.  $(\Phi(\cdot|\boldsymbol{\eta}))_{\boldsymbol{\eta} \in H}$  is a constrained Gaussian or a Bernoulli observation model.
2. The decoder has universal approximation abilities : for any compact  $C \in \mathbb{R}^d$  and continuous function  $f : C \rightarrow H$ , for all  $\varepsilon > 0$ , there exists  $\theta$  such that  $\|f - f_\theta\|_\infty < \varepsilon$ .

Then, for all  $\varepsilon > 0$ , there exists  $\theta \in \Theta$  such that  $\ell(\hat{G}) \geq \ell(\theta) \geq \ell(\hat{G}) - \varepsilon$ .

**Proof main idea:** split the code space into a compact set made of several parts that will represent the mixture components, and an unbounded set of very small prior mass. The universal approximation property is finally used for this compact set.

This means that **the nonparametric upper bound  $\hat{G}$  characterises the large capacity limit of the decoder.**

## Take-home message

A DLVM with a very large decoder will behave like a big finite mixture.

# Overview of talk

A short introduction to deep learning

Deep latent variable models

Approximate maximum likelihood for DLVMs

Connections with nonparametric mixtures

Handling missing data in DLVMs

# Data imputation with variational autoencoders

Rezende et al. (2014), Mattei & Frellsen (2018)

**After training a couple encoder/decoder**, we consider a **new data point**  $\mathbf{x} = (\mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}})$ .

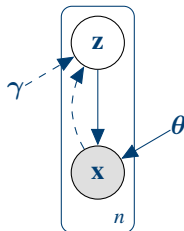
In principle we can **impute**  $\mathbf{x}^{\text{miss}}$  using

$$p_{\theta}(\mathbf{x}^{\text{miss}} | \mathbf{x}^{\text{obs}}) = \int_{\mathbb{R}^d} \Phi(\mathbf{x}^{\text{miss}} | \mathbf{x}^{\text{obs}}, f_{\theta}(\mathbf{z})) p(\mathbf{z} | \mathbf{x}^{\text{obs}}) d\mathbf{z}.$$

Since this integral is intractable, Rezende et al. (2014) suggested using **pseudo-Gibbs sampling**, by forming a Markov chain

$(\mathbf{z}_t, \hat{\mathbf{x}}_t^{\text{miss}})_{t \geq 1}$

- $\mathbf{z}_t \sim \Psi(\mathbf{z}_t | g_{\gamma}(\mathbf{x}^{\text{obs}}, \hat{\mathbf{x}}_{t-1}^{\text{miss}}))$
- $\hat{\mathbf{x}}_t^{\text{miss}} \sim \Phi(\mathbf{x}^{\text{miss}} | \mathbf{x}^{\text{obs}}, f_{\theta}(\mathbf{z}_t)) p(\mathbf{z}_t)$



# Data imputation with variational autoencoders

Rezende et al. (2014), Mattei & Frellsen (2018)

**After training a couple encoder/decoder**, we consider a **new data point**  $\mathbf{x} = (\mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}})$ .

In principle we can **impute**  $\mathbf{x}^{\text{miss}}$  using

$$p_{\theta}(\mathbf{x}^{\text{miss}} | \mathbf{x}^{\text{obs}}) = \int_{\mathbb{R}^d} \Phi(\mathbf{x}^{\text{miss}} | \mathbf{x}^{\text{obs}}, f_{\theta}(\mathbf{z})) p(\mathbf{z} | \mathbf{x}^{\text{obs}}) d\mathbf{z}.$$

Since this integral is intractable, Rezende et al. (2014) suggested using **pseudo-Gibbs sampling**, by forming a Markov chain

$(\mathbf{z}_t, \hat{\mathbf{x}}_t^{\text{miss}})_{t \geq 1}$

- $\mathbf{z}_t \sim \Psi(\mathbf{z}_t | g_{\gamma}(\mathbf{x}^{\text{obs}}, \hat{\mathbf{x}}_{t-1}^{\text{miss}}))$
- $\hat{\mathbf{x}}_t^{\text{miss}} \sim \Phi(\mathbf{x}^{\text{miss}} | \mathbf{x}^{\text{obs}}, f_{\theta}(\mathbf{z}_t)) p(\mathbf{z}_t)$

## We propose Metropolis-within-Gibbs:

---

**Algorithm 1** Metropolis-within-Gibbs sampler for missing data imputation using a trained VAE

---

**Inputs:** Observed data  $\mathbf{x}^{\text{obs}}$ , trained VAE  $(f_{\theta}, g_{\gamma})$ , number of iterations  $T$

**Initialise**  $(\mathbf{z}_0, \hat{\mathbf{x}}_0^{\text{miss}})$

**for**  $t = 1$  **to**  $T$  **do**

$\tilde{\mathbf{z}}_t \sim \Psi(\mathbf{z}_t | g_{\gamma}(\mathbf{x}^{\text{obs}}, \hat{\mathbf{x}}_{t-1}^{\text{miss}}))$

$\rho_t = \frac{\Phi(\mathbf{x}^{\text{obs}}, \hat{\mathbf{x}}_{t-1}^{\text{miss}} | f_{\theta}(\tilde{\mathbf{z}}_t)) p(\tilde{\mathbf{z}}_t)}{\Phi(\mathbf{x}^{\text{obs}}, \hat{\mathbf{x}}_{t-1}^{\text{miss}} | f_{\theta}(\mathbf{z}_{t-1})) p(\mathbf{z}_{t-1})} \frac{\Psi(\mathbf{z}_{t-1} | g_{\gamma}(\mathbf{x}^{\text{obs}}, \hat{\mathbf{x}}_{t-1}^{\text{miss}}))}{\Psi(\tilde{\mathbf{z}}_t | g_{\gamma}(\mathbf{x}^{\text{obs}}, \hat{\mathbf{x}}_{t-1}^{\text{miss}}))}$

$\mathbf{z}_t = \begin{cases} \tilde{\mathbf{z}}_t & \text{with probability } \rho_t \\ \mathbf{z}_{t-1} & \text{with probability } 1 - \rho_t \end{cases}$

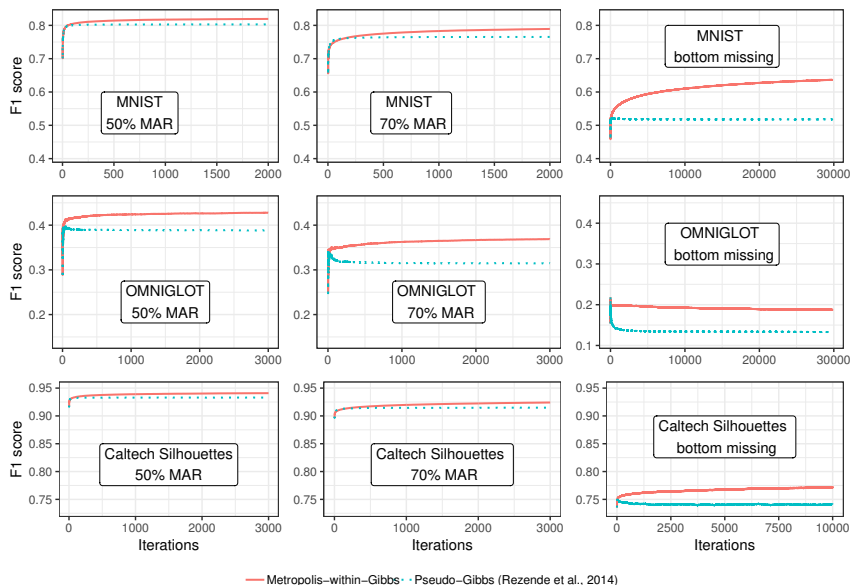
$\hat{\mathbf{x}}_t^{\text{miss}} \sim \Phi(\mathbf{x}^{\text{miss}} | \mathbf{x}^{\text{obs}}, f_{\theta}(\mathbf{z}_t))$

**end for**

---

# Comparing pseudo-Gibbs and Metropolis-within-Gibbs

Mattei & Frelsen (2018)



# What happens if the training set is incomplete?

Assume that some of the training are **missing-at-random** (MAR).

We can then split each sample  $i \in \{1, \dots, n\}$  into

- the **observed features**  $\mathbf{x}_i^o$  and
- the **missing features**  $\mathbf{x}_i^m$ .

Under the MAR assumption, the relevant quantity to **maximise is the likelihood of the observed data** equal to

$$\ell^o(\boldsymbol{\theta}) = \sum_{i=1}^n \log p_{\boldsymbol{\theta}}(\mathbf{x}_i^o) = \sum_{i=1}^n \log \int p_{\boldsymbol{\theta}}(\mathbf{x}_i^o | \mathbf{z}) p(\mathbf{z}) d\mathbf{z}.$$

Direct MLE is intractable, but we can derive tractable tight lower bounds of  $\ell(\boldsymbol{\theta})$ .

# The missing data importance-weighted autoencoder (MIWAE) bound

Mattei & Frelsen (2019)

For the case of **missing data**, we propose the variational distribution

$$q_{\gamma}(\mathbf{z}|\mathbf{x}^o) = \Psi(\mathbf{z}|g_{\gamma}(\iota(\mathbf{x}^o))),$$

where:

- The set  $(\Psi(\cdot|\kappa))_{\kappa \in \mathcal{K}}$  is the *variational family*.
- The function  $g_{\gamma} : \mathcal{X} \rightarrow \mathcal{K}$  is the *encoder*.
- $\iota$  is an **imputation function chosen beforehand** that transforms  $\mathbf{x}^o$  into a complete input vector  $\iota(\mathbf{x}^o) \in \mathcal{X}$  such that  $\iota(\mathbf{x}^o)^o = \mathbf{x}^o$ .

# The missing data importance-weighted autoencoder (MIWAE) bound

Mattei & Frellsen (2019)

For the case of **missing data**, we propose the variational distribution

$$q_\gamma(\mathbf{z}|\mathbf{x}^o) = \Psi(\mathbf{z}|g_\gamma(\iota(\mathbf{x}^o))),$$

where:

- The set  $(\Psi(\cdot|\kappa))_{\kappa \in \mathcal{K}}$  is the *variational family*.
- The function  $g_\gamma : \mathcal{X} \rightarrow \mathcal{K}$  is the *encoder*.
- $\iota$  is an **imputation function chosen beforehand** that transforms  $\mathbf{x}^o$  into a complete input vector  $\iota(\mathbf{x}^o) \in \mathcal{X}$  such that  $\iota(\mathbf{x}^o)^o = \mathbf{x}^o$ .

Following Burda et al. (2016), we can use the distribution  $q_\gamma$  to build lower bounds of  $\ell^o(\theta)$

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

Like before, we will have

$$\mathcal{L}_1^o(\theta, \gamma) \leq \mathcal{L}_2^o(\theta, \gamma) \leq \dots \leq \mathcal{L}_K^o(\theta, \gamma) \xrightarrow{K \rightarrow \infty} \ell(\theta).$$



# The missing data importance-weighted autoencoder (MIWAE) bound

Mattei & Frellsen (2019)

## Imputation function

When  $K \rightarrow \infty$ , the bound is tight for any imputation function  $\iota$

We showed that zero-imputation works fine! It is also possible to use more complex, learnable imputations.

the variational distribution

$$\Psi(\mathbf{z} | g_\gamma(\iota(\mathbf{x}^0))),$$

family.

forehand that transforms  $\mathbf{x}^0$  into a complete

input vector  $\iota(\mathbf{x}^0) \in \mathcal{X}$  such that  $\iota(\mathbf{x}^0)^0 = \mathbf{x}^0$ .

Following Burda et al. (2016), we can use the distribution  $q_\gamma$  to build lower bounds of  $\ell^0(\theta)$

$$\mathcal{L}_K^0(\theta, \gamma) = \sum_{i=1}^n \mathbb{E}_{\mathbf{z}_{i1}, \dots, \mathbf{z}_{iK} \sim q_\gamma(\mathbf{z} | \mathbf{x}_i^0)} \left[ \log \frac{1}{K} \sum_{k=1}^K \frac{p_\theta(\mathbf{x}_i^0 | \mathbf{z}_{ik}) p(\mathbf{z}_{ik})}{q_\gamma(\mathbf{z}_{ik} | \mathbf{x}_i^0)} \right].$$

Like before, we will have

$$\mathcal{L}_1^0(\theta, \gamma) \leq \mathcal{L}_2^0(\theta, \gamma) \leq \dots \leq \mathcal{L}_K^0(\theta, \gamma) \xrightarrow{K \rightarrow \infty} \ell(\theta).$$

# The missing data importance-weighted autoencoder (MIWAE) bound

Mattei & Frellsen (2019)

## Imputation function

When  $K \rightarrow \infty$ , the bound is tight for any imputation function  $\iota$ .

We showed that zero-imputation works fine! It is also possible to use more complex, learnable imputations.

input vector  $\iota(\mathbf{x}^o) \in \mathcal{X}$  such that  $\iota(\mathbf{x}^o)^o = \mathbf{x}^o$ .

## MVAE bound

When  $K = 1$ , the bound resembles the VAE bound and we call it MVAE.

This bound was independently derived independently by Nazabal et al. (2018) in concurrent work.

Following Burda et al. (2016), we can use the distribution  $q_\gamma$  to build lower bounds of  $\ell^o(\theta)$

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

Like before, we will have

$$\mathcal{L}_1^o(\theta, \gamma) \leq \mathcal{L}_2^o(\theta, \gamma) \leq \dots \leq \mathcal{L}_K^o(\theta, \gamma) \xrightarrow{K \rightarrow \infty} \ell(\theta).$$

# Imputation with MIWAE

Mattei & Frelsen (2019)

For the **single imputation problem** an optimal decision-theoretic choice is

$$\hat{\mathbf{x}}^m = \mathbb{E}[\mathbf{x}^m | \mathbf{x}^o] = \int \mathbf{x}^m p_{\theta}(\mathbf{x}^m | \mathbf{x}^o) d\mathbf{x}^m = \iint \mathbf{x}^m p_{\theta}(\mathbf{x}^m | \mathbf{x}^o, \mathbf{z}) p_{\theta}(\mathbf{z} | \mathbf{x}^o) d\mathbf{z} d\mathbf{x}^m,$$

**Exercise:** Derive an estimate of  $\mathbb{E}[\mathbf{x}^m | \mathbf{x}^o]$  using importance sampling.

# Imputation with MIWAE

Mattei & Frelsen (2019)

For the **single imputation problem** an optimal decision-theoretic choice is

$$\hat{\mathbf{x}}^m = \mathbb{E}[\mathbf{x}^m | \mathbf{x}^o] = \int \mathbf{x}^m p_{\theta}(\mathbf{x}^m | \mathbf{x}^o) d\mathbf{x}^m = \iint \mathbf{x}^m p_{\theta}(\mathbf{x}^m | \mathbf{x}^o, \mathbf{z}) p_{\theta}(\mathbf{z} | \mathbf{x}^o) d\mathbf{z} d\mathbf{x}^m,$$

**Exercise:** Derive an estimate of  $\mathbb{E}[\mathbf{x}^m | \mathbf{x}^o]$  using importance sampling.

**Hint:** There is a **self-normalised version of importance sampling** for cases where we only know  $p$  up to a constant:

$$\int_{\Omega} f(x) p(x) dx \approx \sum_{k=1}^K w_k f(x_k),$$

where  $x_1, \dots, x_K \sim q$  and

$$w_k = \frac{r_k}{r_1 + \dots + r_K}, \text{ with } r_k = \frac{\tilde{p}(x_k)}{q(x_k)}.$$

# Imputation with MIWAE

Mattei & Frelsen (2019)

For the **single imputation problem** an optimal decision-theoretic choice is

$$\hat{\mathbf{x}}^m = \mathbb{E}[\mathbf{x}^m | \mathbf{x}^o] = \int \mathbf{x}^m p_{\theta}(\mathbf{x}^m | \mathbf{x}^o) d\mathbf{x}^m = \iint \mathbf{x}^m p_{\theta}(\mathbf{x}^m | \mathbf{x}^o, \mathbf{z}) p_{\theta}(\mathbf{z} | \mathbf{x}^o) d\mathbf{z} d\mathbf{x}^m,$$

# Imputation with MIWAE

Mattei & Frelsen (2019)

For the **single imputation problem** an optimal decision-theoretic choice is

$$\hat{\mathbf{x}}^m = \mathbb{E}[\mathbf{x}^m | \mathbf{x}^o] = \int \mathbf{x}^m p_{\theta}(\mathbf{x}^m | \mathbf{x}^o) d\mathbf{x}^m = \iint \mathbf{x}^m p_{\theta}(\mathbf{x}^m | \mathbf{x}^o, \mathbf{z}) p_{\theta}(\mathbf{z} | \mathbf{x}^o) d\mathbf{z} d\mathbf{x}^m,$$

This is intractable, but can be estimated using **self-normalised importance sampling** with the proposal distribution  $p_{\theta}(\mathbf{x}^m | \mathbf{x}^o, \mathbf{z}) q_{\gamma}(\mathbf{z} | \mathbf{x}^o)$ , leading to the estimate

$$\mathbb{E}[\mathbf{x}^m | \mathbf{x}^o] \approx \sum_{l=1}^L w_l \mathbf{x}_{(l)}^m,$$

where  $(\mathbf{x}_{(1)}^m, \mathbf{z}_{(1)}), \dots, (\mathbf{x}_{(L)}^m, \mathbf{z}_{(L)})$  are i.i.d. samples from  $p_{\theta}(\mathbf{x}^m | \mathbf{x}^o, \mathbf{z}) q_{\gamma}(\mathbf{z} | \mathbf{x}^o)$  and

$$w_l = \frac{r_l}{r_1 + \dots + r_L}, \text{ with } r_l = \frac{p_{\theta}(\mathbf{x}^o | \mathbf{z}_{(l)}) p(\mathbf{z}_{(l)})}{q_{\gamma}(\mathbf{z}_{(l)} | \mathbf{x}^o)}.$$

Here, we leverage the fact that  $q_{\gamma}(\mathbf{z} | \mathbf{x}^o)$  is a good approximation of  $p_{\theta}(\mathbf{z} | \mathbf{x}^o)$ .

# Imputation with MIWAE

Mattei & Frelsen (2019)

For the **single imputation problem** an optimal decision-theoretic choice is

$$\hat{\mathbf{x}}^m = \mathbb{E}[\mathbf{x}^m | \mathbf{x}^o] = \int \mathbf{x}^m p_{\theta}(\mathbf{x}^m | \mathbf{x}^o) d\mathbf{x}^m = \iint \mathbf{x}^m p_{\theta}(\mathbf{x}^m | \mathbf{x}^o, \mathbf{z}) p_{\theta}(\mathbf{z} | \mathbf{x}^o) d\mathbf{z} d\mathbf{x}^m,$$

This is intractable, but can be estimated using **self-normalised importance sampling** with the proposal distribution  $p_{\theta}(\mathbf{x}^m | \mathbf{x}^o, \mathbf{z}) q_{\gamma}(\mathbf{z} | \mathbf{x}^o)$ , leading to the estimate

$$\mathbb{E}[\mathbf{x}^m | \mathbf{x}^o] \approx \sum_{l=1}^L w_l \mathbf{x}_{(l)}^m,$$

where  $(\mathbf{x}_{(1)}^m, \mathbf{z}_{(1)}), \dots, (\mathbf{x}_{(L)}^m, \mathbf{z}_{(L)})$  are i.i.d. samples from  $p_{\theta}(\mathbf{x}^m | \mathbf{x}^o, \mathbf{z}) q_{\gamma}(\mathbf{z} | \mathbf{x}^o)$  and

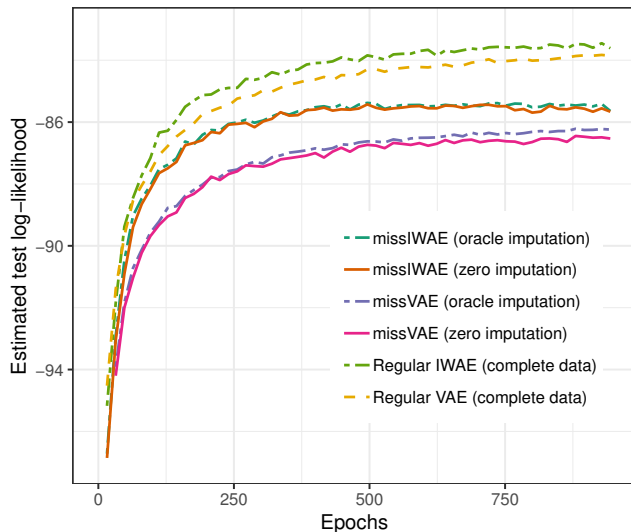
$$w_l = \frac{r_l}{r_1 + \dots + r_L}, \text{ with } r_l = \frac{p_{\theta}(\mathbf{x}^o | \mathbf{z}_{(l)}) p(\mathbf{z}_{(l)})}{q_{\gamma}(\mathbf{z}_{(l)} | \mathbf{x}^o)}.$$

Here, we leverage the fact that  $q_{\gamma}(\mathbf{z} | \mathbf{x}^o)$  is a good approximation of  $p_{\theta}(\mathbf{z} | \mathbf{x}^o)$ .

**Multiple imputation**, i.e. sampling from  $p_{\theta}(\mathbf{x}^m | \mathbf{x}^o)$ , can be done using **sampling importance resampling** according to the weights  $w_l$  for large  $L$ .

# Convolutional MIWAE on binary MNIST (50% MCAR pixels)

Mattei & Frelsen (2019)



Estimated test log-likelihood of various models trained on binary MNIST as a function of the number of training epochs. The MIWAE model was trained using  $K = 50$  importance weights.



# Imputation on binary MNIST

Mattei & Frelsen (2019)

Single imputations (50% MCAR pixels):



# Imputation on binary MNIST

Mattei & Frellsen (2019)

Single imputations (50% MCAR pixels):



Multiple imputations (non-MCAR but MAR scenario):



Random incomplete samples from the MNIST training data set, and the imputations obtained by MIWAE (trained with  $K = 50$  importance weights, and imputed with  $L = 10\,000$  importance weights)

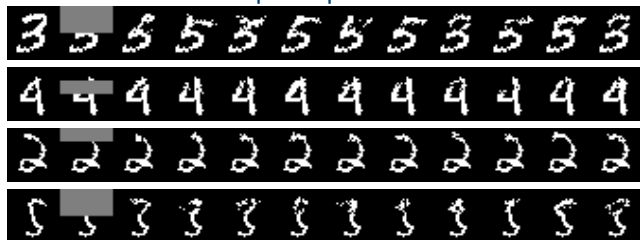
# Another MAR but not MCAR experiment

Mattei & Frelsen (2019)

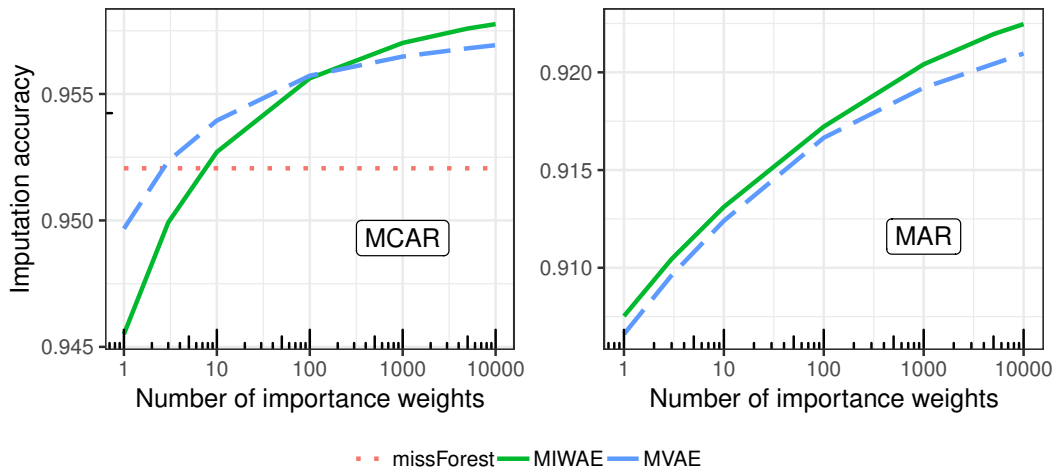
Single imputations :



Multiple imputations :



# Convolutional MIWAE on binary MNIST (50% MCAR pixels and a MAR experiment)



# Classification of binary and incomplete MNIST (50% MCAR pixels)

Mattei & Frelsen (2019)

To evaluate **multiple imputation**, we consider the task of classifying the incomplete binary MNIST data set.

	<i>Test accuracy</i>	<i>Test cross-entropy</i>
Zero imp.	0.9739 (0.0018)	0.1003 (0.0092)
missForest imp.	0.9805 (0.0018)	0.0645 (0.0066)
MIWAE single imp.	0.9847 (0.0009)	0.0510 (0.0035)
MIWAE multiple imp.	<b>0.9870 (0.0003)</b>	<b>0.0396 (0.0003)</b>
Complete data	0.9866 (0.0007)	0.0464 (0.0026)

Test accuracy and cross-entropy obtained by training a convolutional network using the imputed versions of the static binarisation of MNIST. The numbers are the mean of 10 repeated trainings with different seeds and standard deviations are shown in brackets.

# Single imputation of UCI data sets (50% MCAR)

For all data sets **we train DLVMs with the same general properties:**

- Both encoder and decoder are multi-layer perceptrons with 3 hidden layers (128 hidden units) and tanh activations.
- Products of Student's  $t$  for both the variational family and the observation model
- Same number of gradient steps (500 000) for all data sets, and no regularisation.

	<i>Banknote</i>	<i>Breast</i>	<i>Concrete</i>	<i>Red</i>	<i>White</i>	<i>Yeast</i>
MIWAE	<b>0.446 (0.038)</b>	<b>0.280 (0.021)</b>	<b>0.501 (0.040)</b>	<b>0.643 (0.026)</b>	<b>0.735 (0.033)</b>	<b>0.964(0.057)</b>
MVAE	0.593 (0.059)	0.318 (0.018)	0.587(0.026)	0.686 (0.120)	0.782 (0.018)	0.997 (0.064)
missForest	0.676 (0.040)	0.291 (0.026)	0.510 (0.11)	0.697 (0.050)	0.798 (0.019)	1.41 (0.02)
PCA	0.682 (0.016)	0.729 (0.068)	0.938 (0.033)	0.890 (0.033)	0.865 (0.024)	1.05(0.061)
$k$ NN	0.744 (0.033)	0.831 (0.029)	0.962(0.034)	0.981 (0.037)	0.929 (0.025)	1.17 (0.048)
Mean	1.02 (0.032)	1.00 (0.04)	1.01 (0.035)	1.00 (0.03)	1.00 (0.02)	1.06 (0.052)

Mean-squared error for single imputation for various continuous UCI data sets (mean and standard deviations over 5 randomly generated incomplete data sets).

Let's play a bit with MIWAE and impute some data sets!

## Take-home message

- DLVMs are flexible generative models.
- Showed how to **train DLVM with missing data**
- Obtained **state-of-the-art performance in missing data imputation**

## Advertisement

**Postdoc and PhD positions** on deep generative models, and applications for **heterogeneous data, and network data**, starting next spring, at **INRIA Nice** in the south of France. Co-supervised by Charles Bouveyron (INRIA Nice and Université Cote d'Azur) and Pierre Latouche (Université de Paris).



## Take-home message

- DLVMs are flexible generative models.
- Showed how to **train DLVM with missing data**
- Obtained **state-of-the-art performance in missing data imputation**

## Advertisement

**Postdoc and PhD positions** on deep generative models, and applications for **heterogeneous data, and network data**, starting next spring, at **INRIA Nice** in the south of France. Co-supervised by Charles Bouveyron (INRIA Nice and Université Cote d'Azur) and Pierre Latouche (Université de Paris).

**Thank you for your attention**

