Deep latent variable models

Pierre-Alexandre Mattei

IT University of Copenhagen

`http://pamattei.github.io`

@pamattei

19 avril 2018

Séminaire de statistique du CNAM

Overview of talk

A short introduction to deep learning

Deep latent variable models

On the boundedness of the likelihood of deep latent variable models

Handling missing data in deep latent variable models

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** $f_1, \ldots, f_L$ with a simple fixed **nonlinear function** σ**:**

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

These functions are called **layers.** The nonlinearity σ 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** $f_1, ..., f_L$ with a simple fixed **nonlinear function** σ**:**

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

These functions are called **layers.** The nonlinearity σ 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.

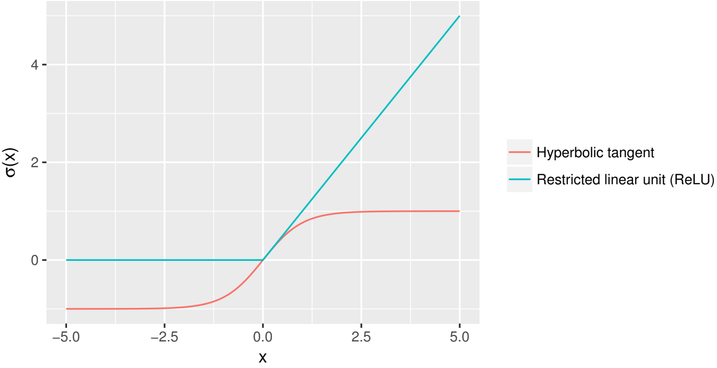
A glimpse at the zoology of layers

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 σ 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**, Cybenko, 1989, Hornik, 1991).

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**, Cybenko, 1989, Hornik, 1991).

- 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** (convnets, LeCun, 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?

- 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**, Cybenko, 1989, Hornik, 1991).

- 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** (convnets, LeCun, 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.

- 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

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

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

We want to perform regression on a data set

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

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

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

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

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

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

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

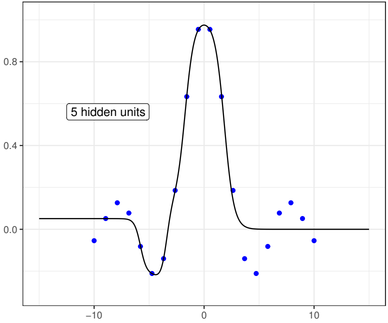
The coordinates of the intermediate representation $\mathbf{W}_0 \mathbf{x}_i + \mathbf{b}_0$ are called **hidden units.**

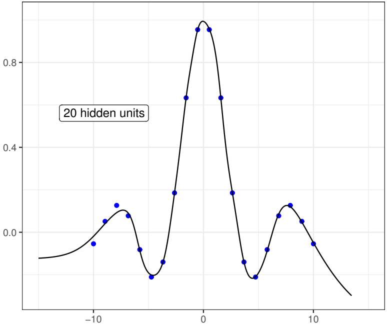
If we assume that the noise is Gaussian, then we can find the maximum likelihood estimates of $\mathbf{W}_1, \mathbf{W}_0, \mathbf{b}_1, \mathbf{b}_0$ by **minimising the squared error using gradient descent.** Gradients are computed via **backpropagation.**

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

On the boundedness of the likelihood of deep latent variable models

Handling missing data in deep latent variable models

Continuous latent variable models

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**, and Rezende et al. (2014) as **deep latent Gaussian models.**

Stochastic Backpropagation and Approximate Inference in Deep Generative Models

Danilo J. Rezende, Shakir Mohamed, Daan Wierstra
{danilor, shakir, daanw}@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-

Uria et al., 2014; Gregor et al., 2014) can be easily sampled from, 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

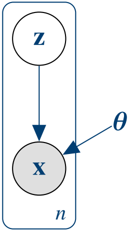
Abstract

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

Deep latent variable models (DLVMs)

(Kingma and Welling, 2014; Rezende et al., 2014; Mattei and Frellsen, 2018)

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



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

where

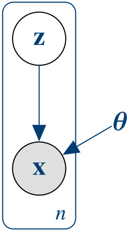
- $\mathbf{z} \in \mathbb{R}^d$ is the **latent** variable,
- $\mathbf{x} \in \mathcal{X}$ is the **observed** variable.

Deep latent variable models (DLVMs)

(Kingma and Welling, 2014; Rezende et al., 2014; Mattei and Frellsen, 2018)

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

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



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**, and
- $(\Phi(\cdot \mid \eta))_{\eta \in H}$ is a parametric family of **output densities**,

  e.g. multivariate Gaussians or products of multinomials.

The role of the decoder

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

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

An illustrative example of a simple non-linear decoder is

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_2) \quad \text{and} \quad f(\mathbf{z}) = \mathbf{z}/10 + \mathbf{z}/\|\mathbf{z}\|.$$

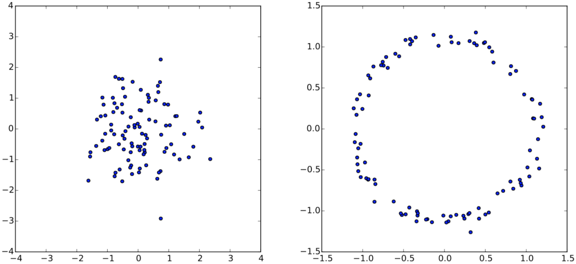
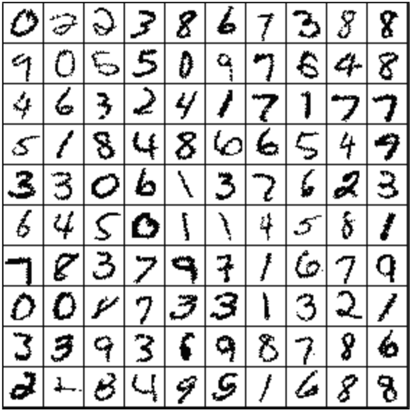


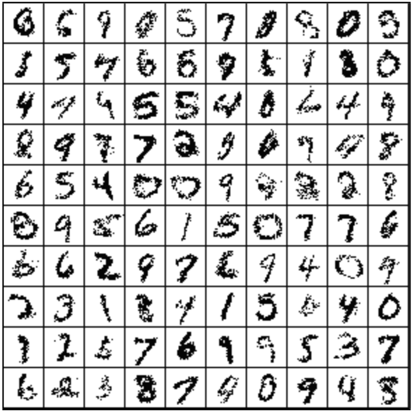
Image from Doersch (2016)

DLVMs applications: density estimation on MNIST

(Rezende et al., 2014)



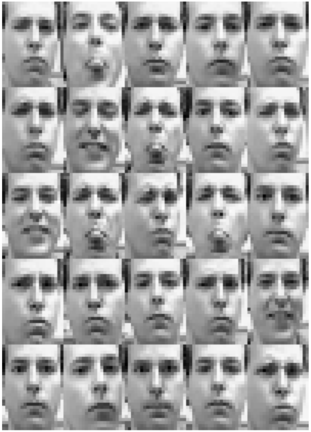
Training data



Model samples

DLVMs applications: density estimation on (Brendan) Frey faces

(Rezende et al., 2014)

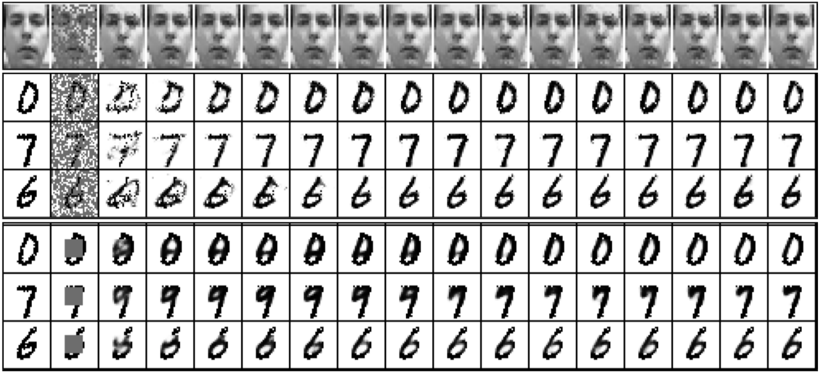


Training data

Model samples

DLVMs applications: Data imputation

(Rezende et al., 2014)



Learning DLVMs

(Kingma and Welling, 2014; Rezende et al., 2014; Mattei and Frellsen, 2018)

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

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

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

:

Learning DLVMs

(Kingma and Welling, 2014; Rezende et al., 2014; Mattei and Frellsen, 2018)

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

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

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

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**

Learning DLVMs

(Kingma and Welling, 2014; Rezende et al., 2014; Mattei and Frellsen, 2018)

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

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

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

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**

Learning DLVMs

(Kingma and Welling, 2014; Rezende et al., 2014; Mattei and Frellsen, 2018)

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

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

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

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.

Variational Inference (VI)

(Kingma and Welling, 2014; Rezende et al., 2014; Blei et al., 2017)

VI approximatively maximises the log-likelihood by maximising the **evidence lower bound**

$$\mathrm{ELBO}(\boldsymbol{\theta}, q) = \mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{Z})}{q(\mathbf{X})}\right] = \ell(\boldsymbol{\theta}) - \mathrm{KL}(q \,||\, p_{\boldsymbol{\theta}}(\cdot \mid \mathbf{X})) \leq \ell(\boldsymbol{\theta})$$

wrt. $(\boldsymbol{\theta}, q)$, where

- $q \in \mathcal{D}$ **is variational distribution** for a family of distributions $\mathcal{D}$ over the space of codes $\mathbb{R}^{n\times d}$,
- $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^\intercal \in \mathcal{X}^n$ and $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n)^\intercal \in \mathbb{R}^{n\times d}$.

Variational Inference (VI)

(Kingma and Welling, 2014; Rezende et al., 2014; Blei et al., 2017)

VI approximatively maximises the log-likelihood by maximising the **evidence lower bound**

$$\mathrm{ELBO}(\boldsymbol{\theta}, q) = \mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{Z})}{q(\mathbf{X})}\right] = \ell(\boldsymbol{\theta}) - \mathrm{KL}(q \,||\, p_{\boldsymbol{\theta}}(\cdot \mid \mathbf{X})) \leq \ell(\boldsymbol{\theta})$$

wrt. $(\boldsymbol{\theta}, q)$, where

- $q \in \mathcal{D}$ **is variational distribution** for a family of distributions $\mathcal{D}$ over the space of codes $\mathbb{R}^{n\times d}$,
- $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^\intercal \in \mathcal{X}^n$ and $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n)^\intercal \in \mathbb{R}^{n\times d}$.

However, computing a distribution over $\mathbb{R}^{n\times d}$ is to costly for large datasets.

Amortised Variational Inference (AVI)

(Kingma and Welling, 2014; Rezende et al., 2014; Gershman and Goodman, 2014)

Amortised inference scales up VI by learning a function g that **transform each data point into the parameters of the approximate posterior**

$$q_{\gamma,\mathbf{x}}(\mathbf{Z}) = \prod_{i=1}^{n} \Psi(\mathbf{z}_i \mid g_{\boldsymbol{\gamma}}(\mathbf{x}_i)),$$

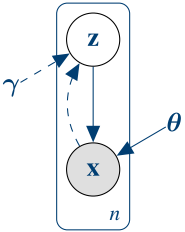
where

- $(\Psi(\cdot|\kappa))_{\kappa \in K}$ is is a parametric family of distributions over $\mathbb{R}^d$ (usually Gaussians),
- $g_{\boldsymbol{\gamma}} : \mathcal{X} \to K$ is a neural net called the **inference network**.

Inference for DLVMs solves the optimisation problem

$$\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}, \boldsymbol{\gamma} \in \boldsymbol{\Gamma}} \mathrm{ELBO}(\boldsymbol{\theta}, q_{\boldsymbol{\gamma},\mathbf{x}}),$$

DLVM with AVI is denote a **variational autoencoder (VAE).**



Our contributions

(Mattei and Frellsen, 2018)

On the boundedness of the likelihood of deep latent variable models:

- We show that **maximum likelihood is ill-posed** for DLVMs with **Gaussian outputs**.
- We propose how to **tackle this problem** using constraints.
- We show that **maximum likelihood is well-posed** for DLVMs with **discrete outputs**.
- We provide a way of **finding an upper bound of the likelihood**.

Handling missing data in deep latent variable models:

- For missing data at test time, we show how to **draw samples according to the exact conditional distribution of the missing data**.

Overview of talk

A short introduction to deep learning

Deep latent variable models

On the boundedness of the likelihood of deep latent variable models

Handling missing data in deep latent variable models

On the boundedness of the likelihood of DLVMs

(Mattei and Frellsen, 2018)

If we see the prior as a mixing distribution, **DLVMs are continuous mixtures of the output distribution.** But ML 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

On the boundedness of the likelihood of DLVMs

(Mattei and Frellsen, 2018)

If we see the prior as a mixing distribution, **DLVMs are continuous mixtures of the output distribution.** But ML 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 questions:

- **Is the likelihood function of DLVMs with Gaussian outputs bounded above?**
- Do we really want a very tight ELBO?

On the boundedness of the likelihood of DLVMs

(Mattei and Frellsen, 2018)

Consider a DLVM with p**-variate Gaussian outputs** where

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

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

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

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

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

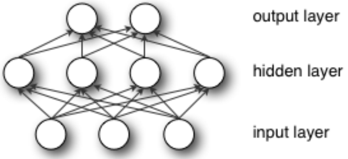


Image from http://deeplearning.net/tutorial/mlp.html

On the boundedness of the likelihood of DLVMs

(Mattei and Frellsen, 2018)

Consider a DLVM with p-**variate Gaussian outputs** where

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

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

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

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

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

Now consider a subfamily with $h = 1$ and

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

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

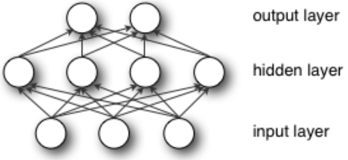


Image from http://deeplearning.net/tutorial/mlp.html

On the boundedness of the likelihood of DLVMs

(Mattei and Frellsen, 2018)

Consider a DLVM with p**-variate Gaussian outputs** where

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

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

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}) = \mathbf{V}\tanh(\mathbf{W}\mathbf{z} + \mathbf{a}) + \mathbf{b} \qquad \boldsymbol{\mu}_{\boldsymbol{\theta}_k^{(i,\mathbf{w})}}(\mathbf{z}) = \mathbf{x}_i$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{z}) = \exp(\boldsymbol{\alpha}^\intercal \tanh(\mathbf{W}\mathbf{z} + \mathbf{a}) + \beta)\mathbf{I}_p \qquad \boldsymbol{\Sigma}_{\boldsymbol{\theta}_k^{(i,\mathbf{w})}}(\mathbf{z}) = \exp(\alpha_k \tanh(\alpha_k \mathbf{w}^\intercal \mathbf{z}) - \alpha_k)\mathbf{I}_p$$

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

Now consider a subfamily with $h = 1$ and

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

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

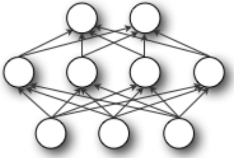


Image from http://deeplearning.net/tutorial/mlp.html

On the boundedness of the likelihood of DLVMs

(Mattei and Frellsen, 2018)

Theorem

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

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.

On the boundedness of the likelihood of DLVMs

(Mattei and Frellsen, 2018)

Theorem

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

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

No, because of the constant mean function.

On the boundedness of the likelihood of DLVMs

(Mattei and Frellsen, 2018)

Theorem

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

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 $k \in \mathbb{N}^*$, $i \in \{1, \ldots, n\}$ and $\mathbf{w} \in \mathbb{R}^d \setminus \{0\}$, the distribution $p_{\boldsymbol{\theta}_k^{(i,\mathbf{w})}}(\mathbf{x}_i)$ is 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 and Frellsen, 2018)

Proposition

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

$$S_p^\xi = \{\mathbf{A} \in S_p^+ \,|\, \min(\mathrm{Sp}\,\mathbf{A}) \geq \xi\}$$

for all $\boldsymbol{\theta}$, 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 $\boldsymbol{\Sigma_\theta}(\mathbf{z})$.

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 with Bernoulli outputs is everywhere negative.

Towards data-dependent likelihood upper bounds

(Mattei and Frellsen, 2018)

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

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

- 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: $\mathbf{z} \sim p(\mathbf{z}); \eta = f_{\boldsymbol{\theta}}(\mathbf{z})$.
- This is a finite mixture model, when G is has a finite support.

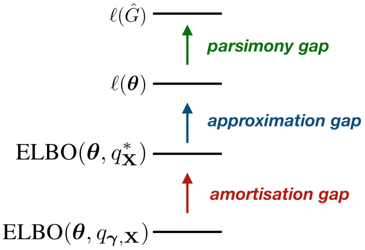
This generalisation **bridges the gap between finite mixtures and DLVMs.**

Towards data-dependent likelihood upper bounds

(Mattei and Frellsen, 2018; Cremer et al., 2018)

This gives us an **immediate upper bound on the likelihood for any decoder f_θ**:

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



Towards data-dependent likelihood upper bounds

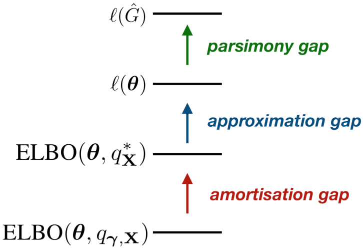
(Mattei and Frellsen, 2018; Cremer et al., 2018)

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

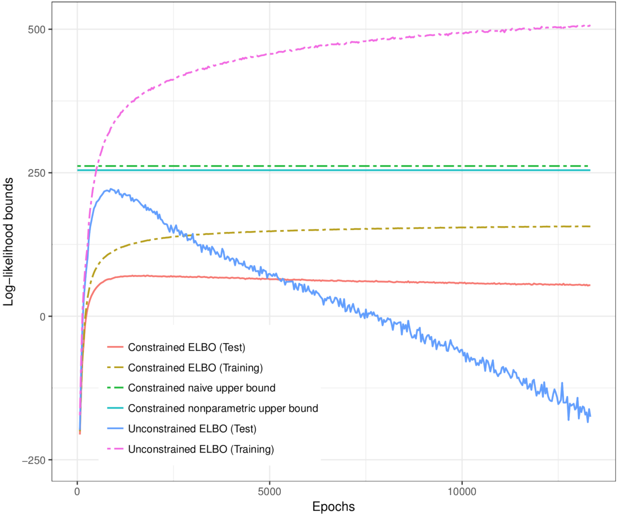
$$\ell(\boldsymbol{\theta}) \leq \max_{G \in \mathcal{P}} \ell(G)$$

Theorem

Assume that $(\Phi(\cdot \mid \eta))_{\eta \in H}$ 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 $(\Phi(\cdot \mid \eta))_{\eta \in H}$.



Unboundedness for a DLVM with Gaussian outputs (Frey faces)



Overview of talk

A short introduction to deep learning

Deep latent variable models

On the boundedness of the likelihood of deep latent variable models

Handling missing data in deep latent variable models

Data imputation with variational autoencoders

(rezende2014; Mattei and Frellsen, 2018)

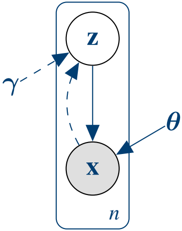
After training a couple encoder/decoder, we consider a **new data point** $\mathbf{x} = (\mathbf{x}^{\mathbf{obs}}, \mathbf{x}^{\mathbf{miss}})$.

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

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

Since (31) is intractable, Rezende et al. (2014) suggested using **pseudo-Gibbs sampling**, by forming a Markov chain $(\mathbf{z}_t, \hat{\mathbf{x}}_t^{\text{miss}})_{t \geq 1}$

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



Data imputation with variational autoencoders

(rezende2014; Mattei and 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

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

Since (31) is intractable, Rezende et al. (2014) suggested using **pseudo-Gibbs sampling**, by forming a Markov chain $(\mathbf{z}_t, \hat{\mathbf{x}}_t^{\text{miss}})_{t \geq 1}$

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

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 $(f_{\boldsymbol{\theta}}, g_{\boldsymbol{\gamma}})$, number of iterations T

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

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

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

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

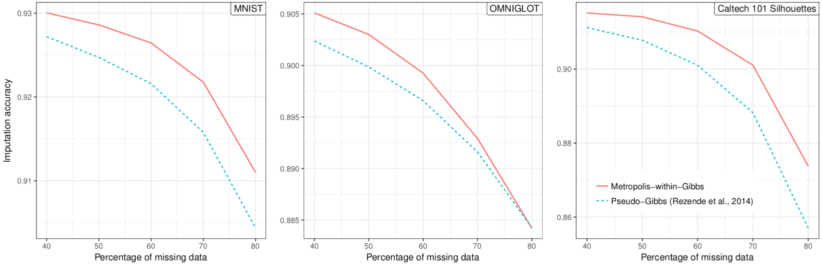
$\mathbf{z}_t = \begin{cases} \tilde{\mathbf{z}}_t & \text{with probability } \rho_t \\ \mathbf{z}_{t-1} & \text{with probability } 1 - \rho_t \end{cases}$

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

end for

Comparing pseudo-Gibbs and Metropolis-within-Gibbs

(Mattei and Frellsen, 2018)



- Network architectures from Rezende et al. (2014) with 200 hidden units and intrinsic dimension of 50.
- Both samples use the same trained VAE.
- Perform the same number of iterations (300).

Comparing pseudo-Gibbs and Metropolis-within-Gibbs

(Mattei and Frellsen, 2018)

	MNIST		OMNIGLOT		Caltech 101 Silhouettes	
Missing half	*top*	*bottom*	*top*	*bottom*	*top*	*bottom*
Pseudo-Gibbs (Rezende et al., 2014)	85.76	88.32	86.98	85.99	68.41	71.02
Metropolis-within-Gibbs	**86.83**	**89.21**	**87.09**	**87.08**	**73.32**	**73.77**

- Network architectures from Rezende et al. (2014) with 200 hidden units and intrinsic dimension of 50.
- Both samples use the same trained VAE.
- Perform the same number of iterations (500).

Summary

- DLVMs are highly flexible generative models.
- We showed that **MLE is ill-posed** for unconstrained DLVMs with Gaussian output.
- We propose how to **tackle this problem** using constraints.
- We provided an **upper bound for the likelihood** in well-posed cases.
- We showed how to **draw samples according to the exact conditional distribution** with missing data.

Summary

- DLVMs are highly flexible generative models.
- We showed that **MLE is ill-posed** for unconstrained DLVMs with Gaussian output.
- We propose how to **tackle this problem** using constraints.
- We provided an **upper bound for the likelihood** in well-posed cases.
- We showed how to **draw samples according to the exact conditional distribution** with missing data.

Thank you for your attention!

Questions?

References

Blei, D. M., A. Kucukelbir, and J. D. McAuliffe (2017). “Variational Inference: A Review for Statisticians”. In: *Journal of the American Statistical Association* 112.518, pp. 859–877.

Cremer, C., X. Li, and D. Duvenaud (2018). “Inference Suboptimality in Variational Autoencoders”. In: *arXiv:1801.03558*.

Day, N. E. (1969). “Estimating the Components of a Mixture of Normal Distributions”. In: *Biometrika* 56.3, pp. 463–474.

Doersch, C. (2016). “Tutorial on variational autoencoders”. In: *arXiv:1606.05908*.

Frellsen, J., I. Moltke, M. Thiim, K. V. Mardia, J. Ferkinghoff-Borg, and T. Hamelryck (2009). “A Probabilistic Model of RNA Conformational Space”. In: *PLOS Computational Biology* 5.6, pp. 1–11.

Gershman, S. and N. Goodman (2014). “Amortized inference in probabilistic reasoning”. In: *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 36.

Kingma, D. P. and M. Welling (2014). “Auto-encoding variational Bayes”. In: *International Conference on Learning Representations*.

MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge University Press.

Mattei, P.-A. and J. Frellsen (2018). “Leveraging the Exact Likelihood of Deep Latent Variables Models”. In: *arXiv:1802.04826*.

References

Mohamed, S. and D. Rezende (2017). *Tutorial on Deep Generative Models*. UAI 2017.

Rezende, D. J., S. Mohamed, and D. Wierstra (2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *Proceedings of the 31st International Conference on Machine Learning*, pp. 1278–1286.