Deep latent variable models: variational autoencoders and extensions

Pierre-Alexandre Mattei

Inria, Université Côte d'Azur

pamattei.github.io
@pamattei
pierre-alexandre.mattei@inria.fr

Overview of today's lecture

Some bits of information geometry

Approximate inference for DLVMs

What is information geometry about?

Information geometry is a subfield of information theory that studies the **geometrical properties of sets of probability distributions**.

What is information geometry about?

Information geometry is a subfield of information theory that studies the **geometrical properties of sets of probability distributions**.

For example, we may want to answer these kinds of questions:

- what does it mean to **find the "model that fits the data best"?**
- what does it mean for a probability distribution p **to be "close" to another probability distribution** q**?** $\text{dist}(p, q) \approx 0$?
- how can we **move around the sets of probability distributions** in a (geometrically) principled way?

What is information geometry about?

Information geometry is a subfield of information theory that studies the **geometrical properties of sets of probability distributions**.

For example, we may want to answer these kinds of questions:

- what does it mean to **find the "model that fits the data best"?**
- what does it mean for a probability distribution p **to be "close" to another probability distribution** q**?** $\text{dist}(p, q) \approx 0$?
- how can we **move around the sets of probability distributions** in a (geometrically) principled way?

Today, we'll simply talk about the **first question**. There are **many ways to define "distances" over sets of probability distributions.** Today, we'll talk about the **Kullback-Leibler divergence** $\text{KL}(p||q)$.

What's a distance again?

A **distance** (aka metric) over a set $\mathcal{X}$ is a function $d : \mathcal{X}^2 \longrightarrow \mathbb{R}$ such that, for all $x, y, z \in \mathcal{X}$;

- $d(x, y) = d(y, x)$ – *symmetry*
- $d(x, y) = 0 \iff x = y$ – *identity of indiscernibles*
- $d(x, z) \geq d(x, y) + d(y, z)$ – *triangle inequality*

A consequence of these axioms in that $d(x, y) \geq 0$ (*positivity*).

What's a distance again?

A **distance** (aka metric) over a set $\mathcal{X}$ is a function $d : \mathcal{X}^2 \longrightarrow \mathbb{R}$ such that, for all $x, y, z \in \mathcal{X}$;

- $d(x, y) = d(y, x)$ – *symmetry*
- $d(x, y) = 0 \iff x = y$ – *identity of indiscernibles*
- $d(x, z) \geq d(x, y) + d(y, z)$ – *triangle inequality*

A consequence of these axioms in that $d(x, y) \geq 0$ (*positivity*).

It's generally hard to satisfy all these axioms. The **Kullback-Leibler** does not! so it's not a "true distance", that's why we call it a **divergence** (in topology, people may call it a premetric, or a pseudometric).

Definition of the KL divergence

Let p and q be two probability densities over a space $\mathcal{X}$, we define

$$\mathbf{KL}(p||q) = \mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right].$$

[1] The proof is quite short if you use Jensen's inequality

[2] If you're interested, take a look at `https://danilorezende.com/2018/07/12/short-notes-on-divergence-measures/` for example

Definition of the KL divergence

Let p and q be two probability densities over a space $\mathcal{X}$, we define

$$\mathbf{KL}(p||q) = \mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right].$$

We have the following properties:

- $\mathbf{KL}(p||q) \geq 0$ – *positivity*[1]
- $\mathbf{KL}(p||q) = 0 \iff p = q$ – *identity of indiscernibles*

but no symmetry, and no triangle inequality!

[1] The proof is quite short if you use Jensen's inequality

[2] If you're interested, take a look at `https://danilorezende.com/2018/07/12/short-notes-on-divergence-measures/` for example

Definition of the KL divergence

Let p and q be two probability densities over a space $\mathcal{X}$, we define

$$\mathbf{KL}(p||q) = \mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right].$$

We have the following properties:

- $\mathbf{KL}(p||q) \geq 0$ – *positivity*[1]
- $\mathbf{KL}(p||q) = 0 \iff p = q$ – *identity of indiscernibles*

but no symmetry, and no triangle inequality!

The KL divergence has many interesting properties that we don't have the time to see today, and it's not the only interesting "distance" over probabilities![2]

[1] The proof is quite short if you use Jensen's inequality

[2] If you're interested, take a look at `https://danilorezende.com/2018/07/12/short-notes-on-divergence-measures/` for example

Example: how far away are two Gaussians

Usually, computing the KL in closed-form is quite hard. But it is sometimes possible, like for p**-variate Gaussians:**

$$\mathbf{KL}(\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)||\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)) = \frac{1}{2}\left(\mathbf{tr}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) + \log\frac{\det\boldsymbol{\Sigma}_1}{\det\boldsymbol{\Sigma}_0} - p\right).$$

Example: how far away are two Gaussians

Usually, computing the KL in closed-form is quite hard. But it is sometimes possible, like for p**-variate Gaussians:**

$$\mathbf{KL}(\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)||\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)) = \frac{1}{2}\left(\mathbf{tr}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) + \log\frac{\det\boldsymbol{\Sigma}_1}{\det\boldsymbol{\Sigma}_0} - p\right).$$

This formula can be useful for **coding variational autoencoders** (among other things).

Example: how far away are two Gaussians

Usually, computing the KL in closed-form is quite hard. But it is sometimes possible, like for p**-variate Gaussians:**

$$\mathbf{KL}(\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)||\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)) = \frac{1}{2}\left(\mathbf{tr}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) + \log\frac{\det\boldsymbol{\Sigma}_1}{\det\boldsymbol{\Sigma}_0} - p\right).$$

This formula can be useful for **coding variational autoencoders** (among other things).

One can see that it is not symmetric, and "not just the Euclidean distance between the means and covariances".

Application: why do we do maximum likelihood?

We have some i.i.d. data $x_1, ..., x_n \in \mathcal{X}$ from a probability distribution p_{true}.

Application: why do we do maximum likelihood?

We have some i.i.d. data $x_1, ..., x_n \in \mathcal{X}$ from a probability distribution p_{true}.

We don't know p_{true}, but want to **approximate it using the data and a family of candidate probability distributions $(p_\theta)_{\theta \in \Theta}$.**

Application: why do we do maximum likelihood?

We have some i.i.d. data $x_1, ..., x_n \in \mathcal{X}$ from a probability distribution p_{true}.

We don't know p_{true}, but want to **approximate it using the data and a family of candidate probability distributions $(p_\theta)_{\theta \in \Theta}$.**

How can we choose $\hat{\theta}$?

Application: why do we do maximum likelihood?

We have some i.i.d. data $x_1, ..., x_n \in \mathcal{X}$ from a probability distribution p_{true}.

We don't know p_{true}, but want to **approximate it using the data and a family of candidate probability distributions $(p_\theta)_{\theta \in \Theta}$.**

How can we choose $\hat{\theta}$?

It makes sense to choose θ in order to be "as close to the truth as possible". If we measure "closeness" using the KL, this means that we want to find

$$\hat{\theta} \in \text{argmin}_{\theta \in \Theta} \text{KL}(p_{\text{true}}, p_\theta).$$

Application: why do we do maximum likelihood?

Let's play a bit with that KL we want to minimise:

$$\mathrm{KL}(p_{\text{true}}, p_\theta) = \mathbb{E}_{x \sim p_{\text{true}}} \left[\log \frac{p_{\text{true}}(x)}{p_\theta(x)} \right]$$

Application: why do we do maximum likelihood?

Let's play a bit with that KL we want to minimise:

$$\mathrm{KL}(p_{\text{true}}, p_\theta) = \mathbb{E}_{x\sim p_{\text{true}}}\left[\log \frac{p_{\text{true}}(x)}{p_\theta(x)}\right]$$

$$\mathrm{KL}(p_{\text{true}}, p_\theta) = \mathbb{E}_{x\sim p_{\text{true}}}\left[\log p_{\text{true}}(x)\right] - \mathbb{E}_{x\sim p_{\text{true}}}\left[\log p_\theta(x)\right]$$

The **first term does not depend on the parameter**, so minimising $\mathrm{KL}(p_{\text{true}}, p_\theta)$ is equivalent to minimising $-\mathbb{E}_{x\sim p_{\text{true}}}\left[\log p_\theta(x)\right]$.

Application: why do we do maximum likelihood?

Let's play a bit with that KL we want to minimise:

$$\mathrm{KL}(p_{\text{true}}, p_\theta) = \mathbb{E}_{x\sim p_{\text{true}}}\left[\log \frac{p_{\text{true}}(x)}{p_\theta(x)}\right]$$

$$\mathrm{KL}(p_{\text{true}}, p_\theta) = \mathbb{E}_{x\sim p_{\text{true}}}\left[\log p_{\text{true}}(x)\right] - \mathbb{E}_{x\sim p_{\text{true}}}\left[\log p_\theta(x)\right]$$

The **first term does not depend on the parameter**, so minimising $\mathrm{KL}(p_{\text{true}}, p_\theta)$ is equivalent to minimising $-\mathbb{E}_{x\sim p_{\text{true}}}\left[\log p_\theta(x)\right]$.

But **we don't know p_{true}!** How can we try to compute

$$-\mathbb{E}_{x\sim p_{\text{true}}}\left[\log p_\theta(x)\right]$$

anyway?

Application: why do we do maximum likelihood?

Let's use Monte Carlo!

$$-\mathbb{E}_{x\sim p_{\text{true}}}\left[\log p_\theta(x)\right] \approx \frac{-1}{n}\sum_{i=1}^{n}\log p_\theta(x_i) = \frac{-1}{n} \times \textbf{Log-likelihood},$$

so **maximum likelihood is equivalent to minimising a Monte Carlo estimate of the KL between the truth and the model.**

Application: why do we do maximum likelihood?

Let's use Monte Carlo!

$$-\mathbb{E}_{x\sim p_{\text{true}}}\left[\log p_\theta(x)\right] \approx \frac{-1}{n}\sum_{i=1}^{n}\log p_\theta(x_i) = \frac{-1}{n} \times \textbf{Log-likelihood},$$

so **maximum likelihood is equivalent to minimising a Monte Carlo estimate of the KL between the truth and the model.**

Maximum likelihood (approximatively) finds the best model with respect to the KL divergence.

Application: why do we do maximum likelihood?

Let's use Monte Carlo!

$$-\mathbb{E}_{x\sim p_{\text{true}}}\left[\log p_\theta(x)\right] \approx \frac{-1}{n}\sum_{i=1}^{n}\log p_\theta(x_i) = \frac{-1}{n} \times \textbf{Log-likelihood},$$

so **maximum likelihood is equivalent to minimising a Monte Carlo estimate of the KL between the truth and the model.**

Maximum likelihood (approximatively) finds the best model with respect to the KL divergence.

This does not means that maximum likelihood is always a good choice. There were a few important assumptions in these two slides (existence of a true model, Monte Carlo approximation...). Sometimes, maximum likelihood works very poorly (but that's another story).

Overview of today's lecture

Some bits of information geometry

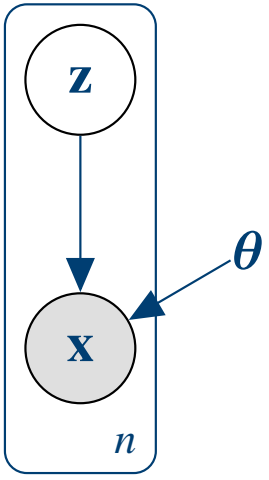
Approximate inference for DLVMs

Recap: Deep latent variable models (DLVMs)

Kingma and Welling (ICLR 2014), Rezende, Mohamed & Wierstra (ICML 2014), Goodfellow et al. (NeurIPS 2014)

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

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



where

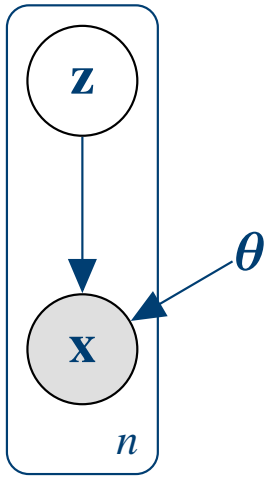
- $\mathbf{z} \in \mathbb{R}^d$ is the **latent** variable,
- $\mathbf{x} \in \mathcal{X}$ is the **observed** variable.

Recap: Deep latent variable models (DLVMs)

Kingma and Welling (ICLR 2014), Rezende, Mohamed & Wierstra (ICML 2014), Goodfellow et al. (NeurIPS 2014)

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

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

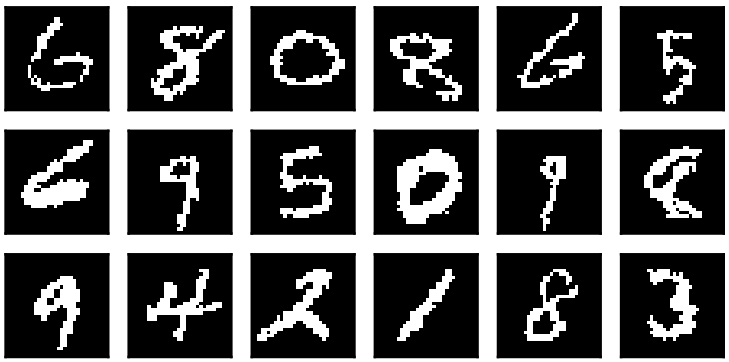


where

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

Recap: Illustrative example of a DLVM

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



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

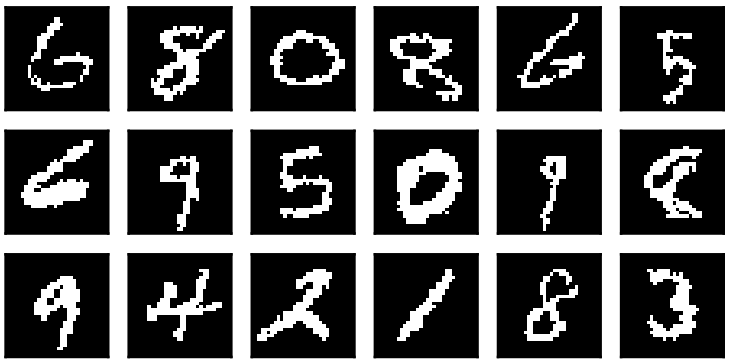
$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

Decoder network

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

Recap: Illustrative example of a DLVM

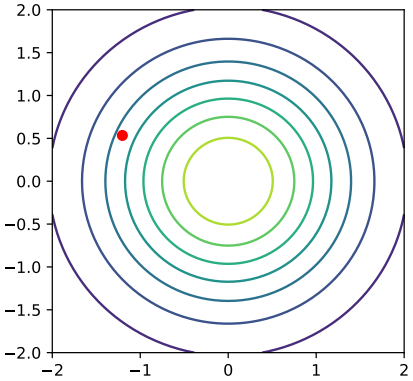
Training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ binary MNIST



Generation

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

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



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

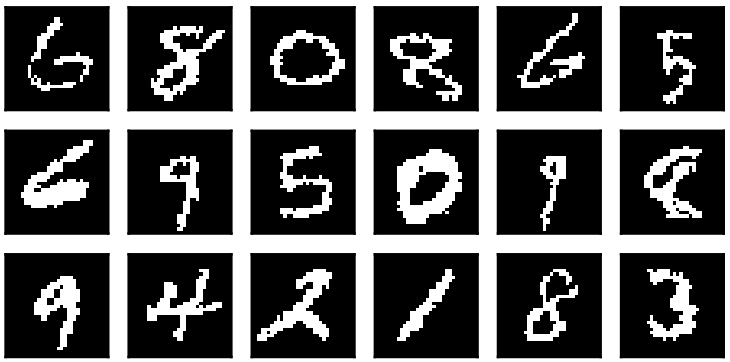
$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

Decoder network

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

Recap: Illustrative example of a DLVM

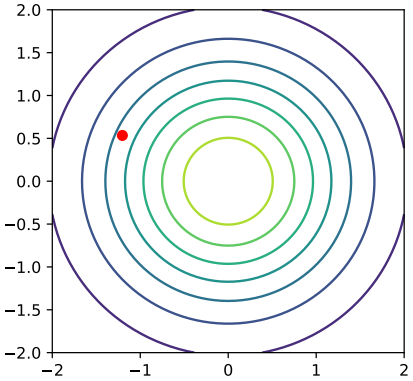
Training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ binary MNIST



Generation

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

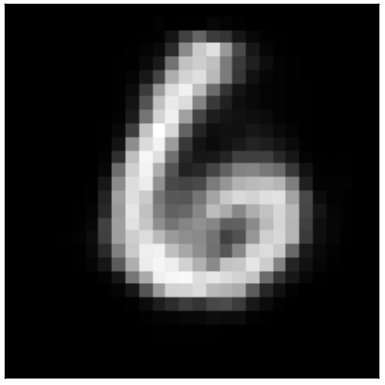
$$\mathbf{z} = (-1.2033, 0.5340)$$



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

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

$f(\mathbf{z})$

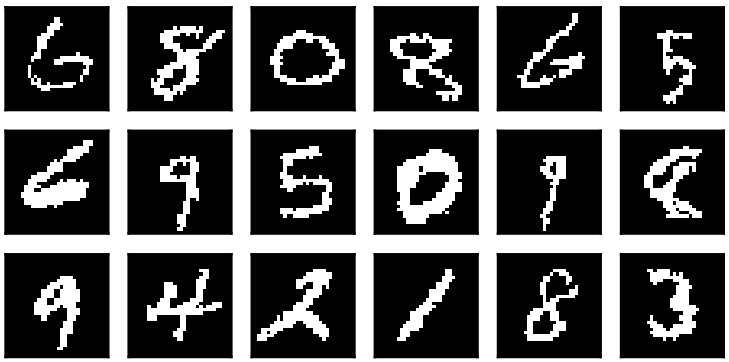


Decoder network

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

Recap: Illustrative example of a DLVM

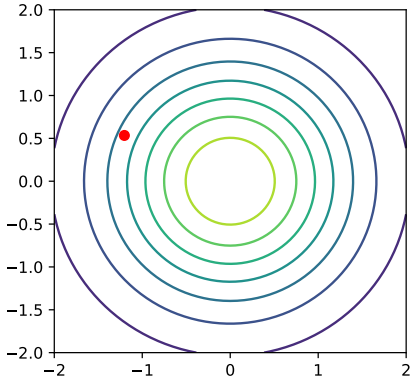
Training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ binary MNIST



Generation

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

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



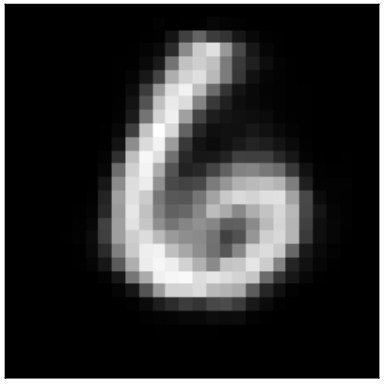
Generative model for $\mathbf{z} \in \mathbb{R}^2$ and $\mathbf{x} \in \{0,1\}^{28\times 28}$

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

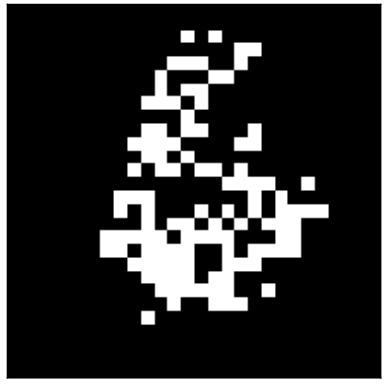
Decoder network

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

$f(\mathbf{z})$

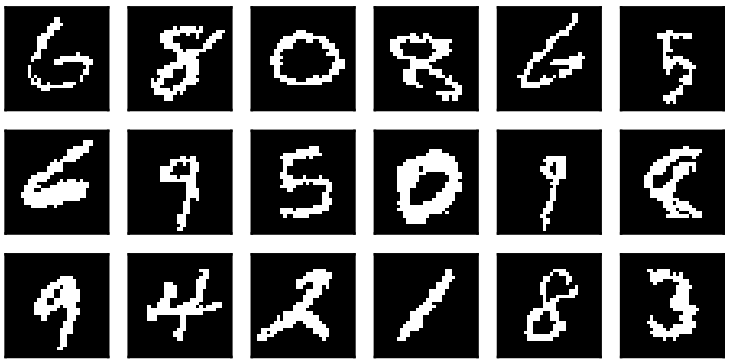


$x^{j,k} \sim \text{Bern}(f^{j,k}(\mathbf{z}))$



Recap: Illustrative example of a DLVM

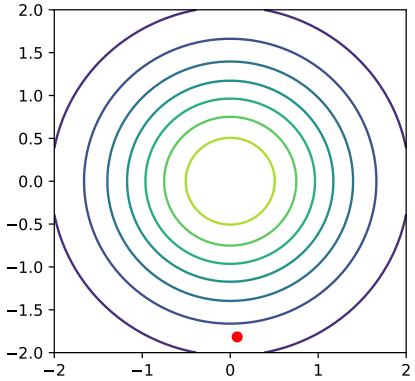
Training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ binary MNIST



Generation

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

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



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

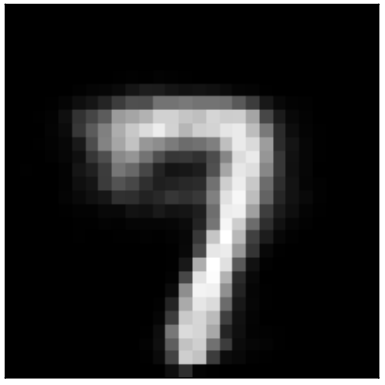
$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

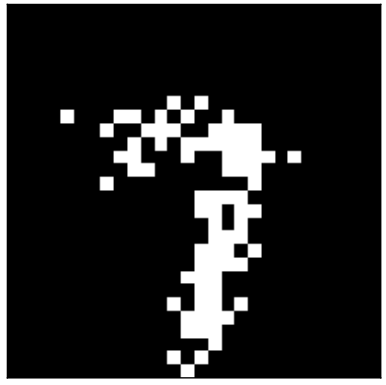
Decoder network

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

$f(z)$

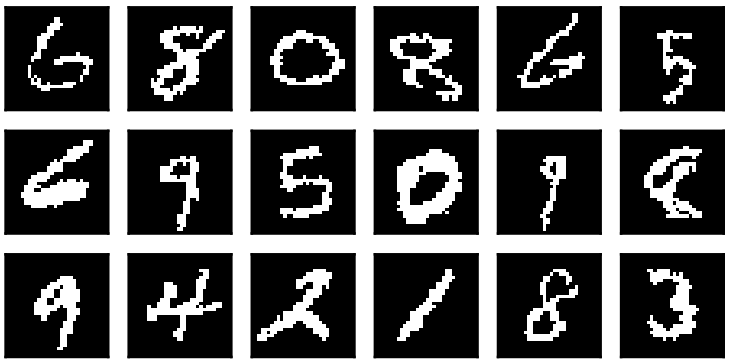
$x^{j,k} \sim \text{Bern}(f^{j,k}(z))$





Recap: Illustrative example of a DLVM

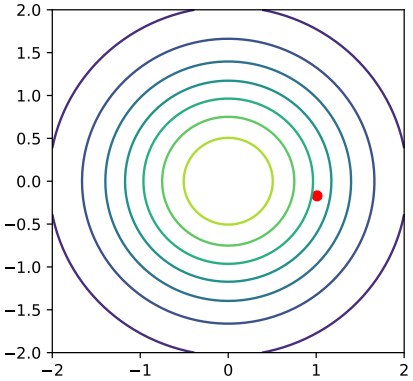
Training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ binary MNIST



Generation

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

$$\mathbf{z} = (1.0097, -0.1711)$$



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

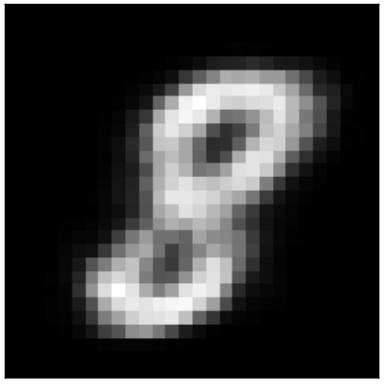
$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

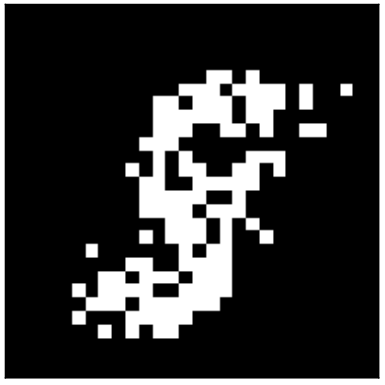
Decoder network

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

$f(z)$

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





Learning DLVMs

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

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

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

where

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

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

:

Learning DLVMs

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

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

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

where

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

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

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

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

Learning DLVMs

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

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

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

where

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

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

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

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

Learning DLVMs

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

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

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

where

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

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

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

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

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

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

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 Ω.

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 Ω.

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

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

A few properties:

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

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** $x_1, \ldots, .x_K$ **from another density** q **rather than** p**.** The trick is the following:

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

Aparté: Monte Carlo and importance sampling

Importance sampling tries to improve this estimate by **sampling** $x_1, \ldots, .x_K$ **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?**

Aparté: Monte Carlo and importance sampling

Importance sampling tries to improve this estimate by **sampling** $x_1, \ldots, .x_K$ **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** $x_1, \ldots, .x_K$ **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

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

Back to DLVMs and their likelihood

We want to approximate

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

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

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

Back to DLVMs and their likelihood

We want to approximate

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

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

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

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

Back to DLVMs and their likelihood

We want to approximate

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

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

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

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

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

Back to DLVMs and their likelihood

We want to approximate

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

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

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

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

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

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

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 γ:**

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

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 γ:**

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

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

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

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 γ:**

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

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

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

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

Amortised variational inference

All of this leads to the following approximation of the likelihood

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

Amortised variational inference

All of this leads to the following approximation of the likelihood

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

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

Amortised variational inference

All of this leads to the following approximation of the likelihood

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

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

It does make sense! For several reasons:

- $\mathcal{L}_K(\boldsymbol{\theta}, \boldsymbol{\gamma})$ is a **lower bound of $\ell(\theta)$** (exercise !). Which means that we know that the likelihood is at least as big as $\mathcal{L}_K(\boldsymbol{\theta}, \boldsymbol{\gamma})$.
- The bounds get **tighter and tighter!**

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

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

What about the VAE?

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

What about the VAE?

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

The VAE bound can be interestingly rewritten

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

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

What about the VAE?

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

The VAE bound can be interestingly rewritten

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

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

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

What about the VAE?

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

The VAE bound can be interestingly rewritten

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

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

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

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

Playing with the VAE bound

The VAE bound can be also rewritten

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

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

Playing with the VAE bound

The VAE bound can be also rewritten

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

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

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