Implementing VAEs

Pierre-Alexandre Mattei

Inria, Université Côte d'Azur

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

Overview of today's lecture

What are DLVMs and VAEs again?

Approximate maximum likelihood for DLVMs

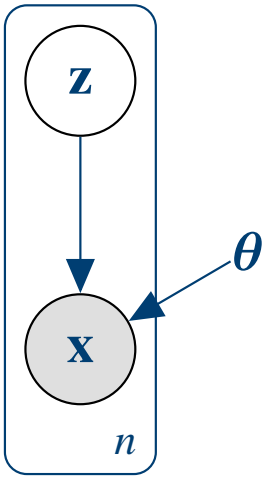
The reparametrisation trick

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

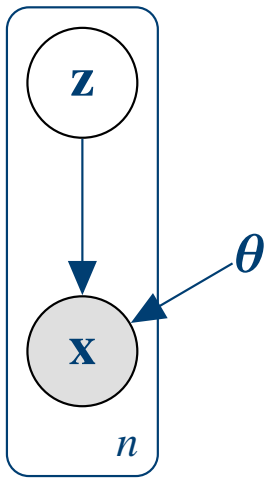
- $\mathbf{z} \in \mathbb{R}^d$ is the **latent** variable,
- $\mathbf{x} \in \mathcal{X}$ is the **observed** variable.

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:

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

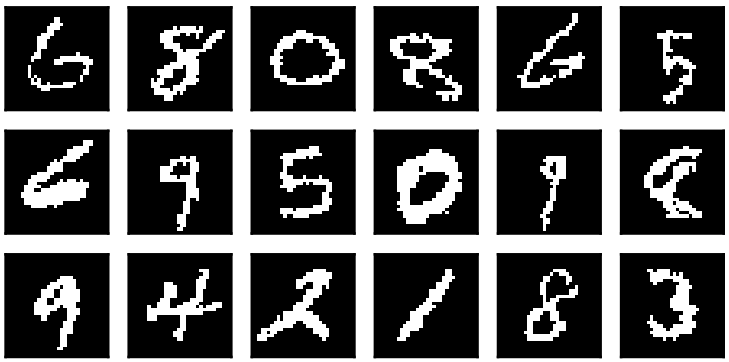


where

- $\mathbf{z} \in \mathbb{R}^d$ 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

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

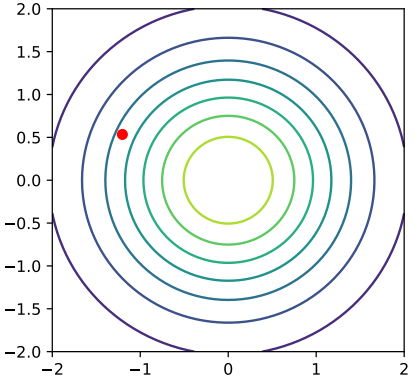
Recap: Illustrative example of a DLVM

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

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

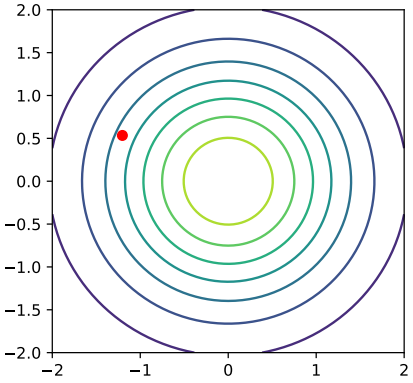
Recap: Illustrative example of a DLVM

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

Generation

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

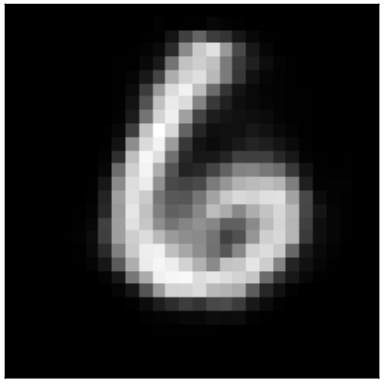
$$\mathbf{z} = (-1.2033, 0.5340)$$



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

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

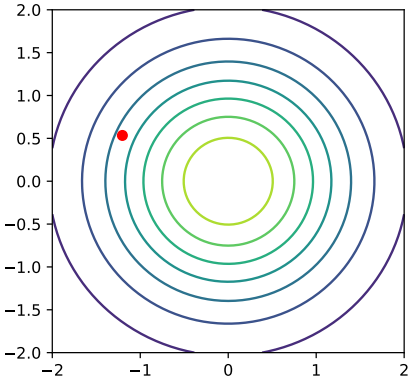
Recap: Illustrative example of a DLVM

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

Generation

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

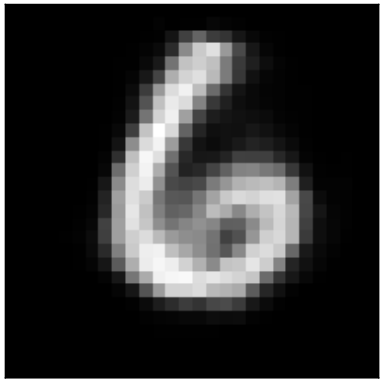
$$\mathbf{z} = (-1.2033, 0.5340)$$

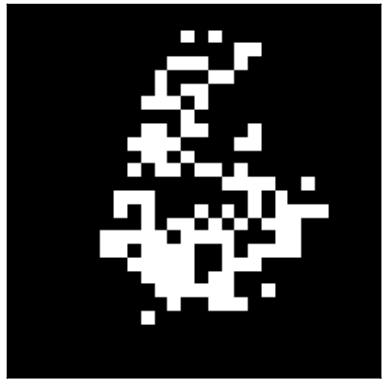


Generative model for $\mathbf{z} \in \mathbb{R}^2$ and $\mathbf{x} \in \{0,1\}^{28\times 28}$

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

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





Decoder network

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

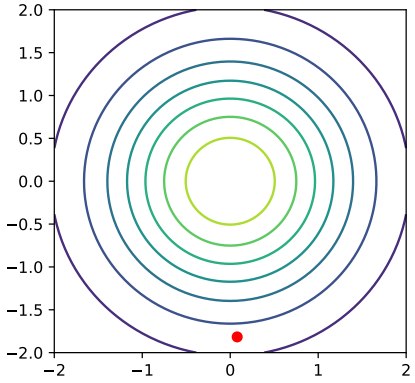
Recap: Illustrative example of a DLVM

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

Generation

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

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



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

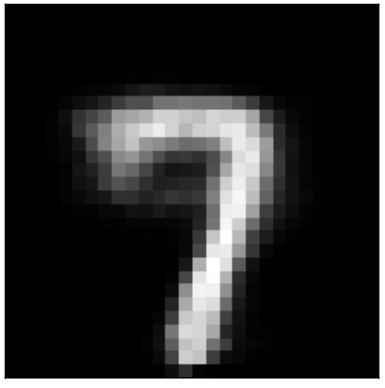
$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

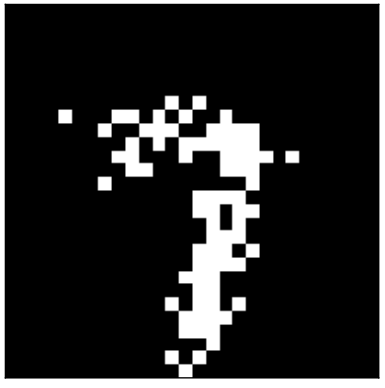
Decoder network

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

$f(z)$

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





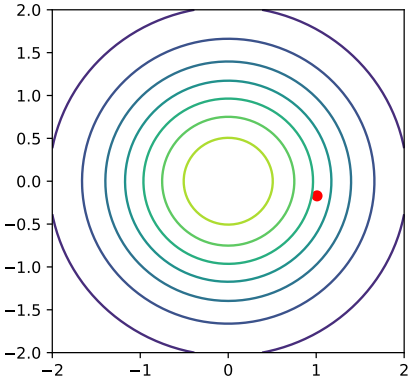
Recap: Illustrative example of a DLVM

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 $\mathbf{x} \in \{0,1\}^{28\times 28}$

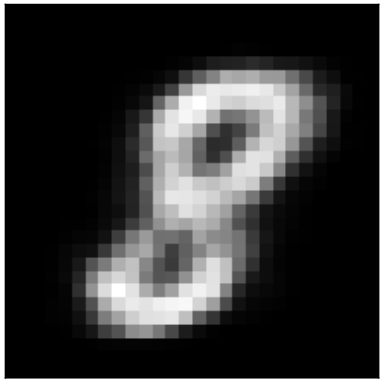
$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

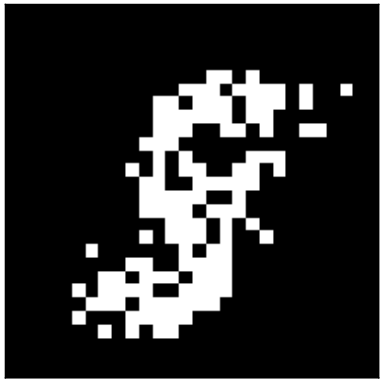
Decoder network

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

$f(z)$

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





Overview of today's lecture

What are DLVMs and VAEs again?

Approximate maximum likelihood for DLVMs

The reparametrisation trick

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:

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

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

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

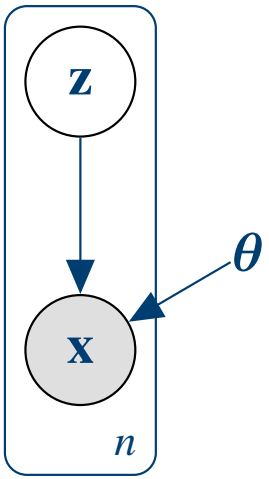
- $\mathbf{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\gamma}}(\mathbf{z}_i|\mathbf{x}_i)}[p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)]$ can be interpreted as (the opposite of) a **reconstruction error.**
- the **KL between the approximate posterior and the prior** can be interpreted as a regulariser.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

Let's summarise

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

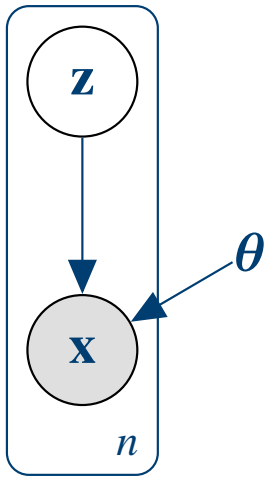
$$\begin{cases} \mathbf{z} \sim p(\mathbf{z}) & \text{(prior)} \\ \mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}) = \Phi(\mathbf{x} \mid f_{\boldsymbol{\theta}}(\mathbf{z})) & \text{(observation model)} \end{cases}$$



Let's summarise

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

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



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 $q_{\boldsymbol{\gamma}}(\mathbf{z} \mid \mathbf{x})$ of the posterior distribution $p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})$ of the codes. This approximation is based on a second neural network called the **inference network or encoder.**

Overview of today's lecture

What are DLVMs and VAEs again?

Approximate maximum likelihood for DLVMs

The reparametrisation trick

What do we still have to do?

Our objective function to maximise is:

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

What do we still have to do?

Our objective function to maximise is:

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

We do not know how to do SGD on the objective function!!

What do we still have to do?

Our objective function to maximise is:

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

We do not know how to do SGD on the objective function!!

Actually, we do not even know how to compute the objective function (because of the expectation). But that's not that big of a deal. Why?

What do we still have to do?

Our objective function to maximise is:

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

We do not know how to do SGD on the objective function!!

Actually, we do not even know how to compute the objective function (because of the expectation). But that's not that big of a deal. Why?

Because SGD only needs unbiased estimates of the gradient of the objective and not exact gradients.

The reparametrisation trick

The reparametrisation trick is a **general recipe to compute expressions of the form**

$$\nabla_\phi \mathbb{E}_{\mathbf{z}\sim q_\phi}[f(\mathbf{z})].$$

The reparametrisation trick

The reparametrisation trick is a **general recipe to compute expressions of the form**

$$\nabla_\phi \mathbb{E}_{\mathbf{z}\sim q_\phi}[f(\mathbf{z})].$$

There are many variants of it. Here, we will only the simplest one where $q_\phi = N(\mathbf{m}, \sigma^2\mathbf{I})$, where $\mathbf{m}, \boldsymbol{\sigma} \in \mathbb{R}^d$.

The reparametrisation trick

The reparametrisation trick is a **general recipe to compute expressions of the form**

$$\nabla_\phi \mathbb{E}_{\mathbf{z}\sim q_\phi}[f(\mathbf{z})].$$

There are many variants of it. Here, we will only the simplest one where $q_\phi = N(\mathbf{m}, \sigma^2\mathbf{I})$, where $\mathbf{m}, \boldsymbol{\sigma} \in \mathbb{R}^d$.

Key idea: It is equivalent to sample $\mathbf{z} \sim N(\mathbf{m}, \sigma^2\mathbf{I})$ and

$$\varepsilon \sim \mathcal{N}(0, \mathbf{I}), \quad \mathbf{z} = \mathbf{m} + \boldsymbol{\sigma} \odot \varepsilon.$$

The reparametrisation trick

The reparametrisation trick is a **general recipe to compute expressions of the form**

$$\nabla_\phi \mathbb{E}_{\mathbf{z}\sim q_\phi}[f(\mathbf{z})].$$

There are many variants of it. Here, we will only the simplest one where $q_\phi = N(\mathbf{m}, \sigma^2\mathbf{I})$, where $\mathbf{m}, \boldsymbol{\sigma} \in \mathbb{R}^d$.

Key idea: It is equivalent to sample $\mathbf{z} \sim N(\mathbf{m}, \sigma^2\mathbf{I})$ and

$$\varepsilon \sim \mathcal{N}(0, \mathbf{I}), \quad \mathbf{z} = \mathbf{m} + \boldsymbol{\sigma} \odot \varepsilon.$$

Therefore,

$$\nabla_\phi \mathbb{E}_{\mathbf{z}\sim q_\phi}[f(\mathbf{z})] = \nabla_\phi \mathbb{E}_{\varepsilon\sim \mathcal{N}(0,\mathbf{I})}[f(\mathbf{m} + \boldsymbol{\sigma} \odot \varepsilon)]$$

The reparametrisation trick

This leads to

$$\nabla_\phi \mathbb{E}_{\mathbf{z}\sim q_\phi}[f(\mathbf{z})] = \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,\mathbf{I})}[\nabla_\phi f(\mathbf{m} + \boldsymbol{\sigma} \odot \varepsilon)],$$

so we can have **unbiased estimates of the gradient** by sampling $\varepsilon \sim \mathcal{N}(0, \mathbf{I})$, and then computing $\nabla_\phi f(\mathbf{m} + \boldsymbol{\sigma} \odot \varepsilon)$.

[1] http://blog.shakirm.com/2015/10/machine-learning-trick-of-the-day-4-reparameterisation-tricks/

The reparametrisation trick

This leads to

$$\nabla_\phi \mathbb{E}_{\mathbf{z}\sim q_\phi}[f(\mathbf{z})] = \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,\mathbf{I})}[\nabla_\phi f(\mathbf{m} + \boldsymbol{\sigma} \odot \varepsilon)],$$

so we can have **unbiased estimates of the gradient** by sampling $\varepsilon \sim \mathcal{N}(0, \mathbf{I})$, and then computing $\nabla_\phi f(\mathbf{m} + \boldsymbol{\sigma} \odot \varepsilon)$.

This can be used for many more distributions! An easy-to-read source is provided in S. Mohamed's blog post[1].

[1] http://blog.shakirm.com/2015/10/machine-learning-trick-of-the-day-4-reparameterisation-tricks/