ENS Paris-Saclay
Master Mathématiques, Vision, Apprentissage (MVA)
Clément Mantoux

Research Internship Report

Inria, Aramis Lab

Investigating Bayesian Variational Methods

August 28, 2019

Referent teacher:	Stéphanie Allassonnière
Supervisor:	Stéphanie Allassonnière
Dates:	04/22/2019 - 08/31/2019
Laboratory:	CMAP at École polytechnique

Abstract

Variational Bayesian methods have met a huge success over the last years, and remain a very active research topic. However, little is known on the convergence properties of the underlying algorithm. We first focus on the conditions required to get the almost-sure convergence of the Variational Auto-Encoders algorithm. To that end, we use the mathematical arsenal from the stochastic approximation theory. In a second part, we perform numerical experiments on simple Variational Bayesian models to understand their behavior and what kind of errors they make. Then we propose the design of a new variant of Auto-Encoders based on Markov Chain Monte-Carlo methods and the use of the problem's information geometry.

Acknowledgments

First and foremost, I would like to thank my internship tutor, Pr. Stéphanie Allassonnière, for giving me the opportunity to work at INRIA. Her presence, advices and constant motivation guided my activity along my internship. I look forward to continuing my work with her during my PhD thesis.

I am grateful to INRIA which financed my internship and allowed me to work on my research projects in an ideal academical environment.

I also thank the Aramis Lab team and the Brain and Spine Institute (ICM) for giving me glimpses at the depth and richness of the mathematics behind neuroscience, biostatistics and medical imaging.

I thank the CMAP laboratory at École polytechnique for welcoming me during my internship. I learned a lot through the seminars I could attend.

For their presence and the interesting or helpful discussions we had, I also thank the other PhD students I worked with at CMAP, Vianney, Kevish, Benoît and others.

Eventually, I would like to thank all the CMAP and INRIA researchers, colleagues and staff who welcomed me and made my internship a rich experience.

Table of Contents

1 Introduction

My internship at Inria was a chance to wrap up my academic coursework by diving into the very active research topic of Variational Inference. I got to experience the dynamic walking pace of the machine learning academic community. It gave me the opportunity to work on theoretical as well as concrete applied questions. The initial topic of the internship was a broad question: can we understand how Variational Auto-Encoders methods work? In a first part of my internship, I tried to figure out what kind of mathematical results to expect on the convergence of the original numerical algorithms. This led me to explore the stochastic approximation literature. In a second part, I focused on designing variations on the original numerical procedure which would provide better approximation properties. Following this idea, I explored several directions ranging from Sequential Monte-Carlo to Hamiltonian Monte-Carlo. The current report first presents the general framework of Variational Inference and Variational Auto-Encoders. The next parts present the result of the corresponding periods of my internship, along with the related relevant results in the corresponding academic domains. A non-negligible proportion of the report is devoted to explaining currently existing techniques. I tried to keep it as concise as possible, however this structure is necessary to understand my research activity, and reflects the time I spent investigating the literature of several active research topics.

2 Understanding Variational Bayesian Methods

In this section, we present the main tools which we will be focusing on in the report. We quickly introduce Bayesian Inference and Variational Inference to expose the Variational Auto-Encoders algorithm.

2.1 Bayesian methodology

When designing a parametric statistical model, one must choose between using deterministic or random hidden parameters. The first choice corresponds to the field of *frequentist statistics* and the second one to the field of *Bayesian statistics*. Depending on the context, both methods are expected to produce similar result but may differ when is comes to the quantification of the uncertainty.

Given parameters θ and observations x, a Bayesian model [4] consists in a *prior* distribution $p(\theta)$, which reflects the information *a priori* on the value of the parameters, and a likelihood function $p(x \mid \theta)$ which gives the probability of an observation given the parameter value. The general objective in Bayesian inference is to retrieve the *posterior* distribution of θ given x using Bayes' rule:

$$p(\theta \mid x) = \frac{p(x \mid \theta)p(\theta)}{p(x)}$$

The posterior distribution is often not explicit, or hard to sample from. In those cases, one must resort to approximate the posterior distribution. Two main categories of methods exist:

- **Markov Chain Monte Carlo** (MCMC) methods produce a series of (non-independent) samples which converges ergodically toward the posterior distribution.
- **Variational Inference** looks for the best approximation of the posterior distribution among a family of simpler parametric distributions.

We will focus on the second method.

2.2 Variational Inference

2.2.1 Inference as an optimization problem

As described above, **Variational Inference** (VI) [27] is a general procedure which aims at finding the best approximation of a probability distribution among a variational family of simpler distributions.

Formally, given a posterior distribution of interest $p(\theta \mid x)$ and a family of distributions $\mathcal{Q} = \{q(\theta)\}$, VI solves the following optimization problem:

$$\inf_{q \in \mathcal{Q}} \mathrm{KL}(q(\theta) \mid\mid p(\theta \mid x))$$

where KL designates the Kullback-Leibler divergence (KLD):

$$\mathrm{KL}(q(\theta) \mid\mid p(\theta \mid x)) = \int q(\theta) \log \left(\frac{q(\theta)}{p(\theta \mid x)} \right) \mathrm{d}\theta$$

The strength of Variational Inference is to turn the initial (complex) posterior inference problem into an optimization problem, which can be addressed with a wide variety of algorithms. Recently, the need for large scale computations has led to the development of **Stochastic Variational Inference** [26], which brings the power of Stochastic Gradient Descent (SGD) to Variational Inference.

The idea of Variational Inference comes from a lower bound on the log-likelihood:

$$\begin{aligned}
\log p(x) &= \log \int p(x, \theta) \, \mathrm{d}\theta \\
&= \log \int \frac{p(x, \theta)}{q(\theta)} q(\theta) \, \mathrm{d}\theta \\
&\geq \int \log \left(\frac{p(x, \theta)}{q(\theta)} \right) q(\theta) \, \mathrm{d}\theta \quad \text{(Jensen's inequality)} \\
&\geq \log p(x) - \mathrm{KL}(q(\theta) \mid\mid p(\theta \mid x))
\end{aligned}$$

This bound is called the **Evidence Lower BOund** (ELBO). Since the term $\log p(x)$ does not depend on q, minimizing the KL divergence is equivalent to maximizing the ELBO. Many branches of VI consist in proposing variations of this bound [1, 25, 6].

2.2.2 Scaling up with amortized inference

In statistical modeling, the dimension of the hidden parameters often varies with the number of observations: typically we observe N variables x_i and want to retrieve the posterior distribution of N corresponding latent variables z_i. The independence between the experiments gives a separable likelihood:

$$p(x, z) = \prod_{i=1}^{N} p(x_i \mid z_i) p(z_i)$$

When dealing with a large number of observations, the dimension of the hidden variables thus rises accordingly. If every variational distribution $q(z_i \mid x_i)$ has to be optimized, the variational family $\{q(z \mid x) = \prod_{i=1}^{N} q(z_i \mid x_i)\}$ rapidly becomes intractably large.

This issued is addressed with the notion of amortization. **Amortized Variational Inference** [52] consists in parameterizing all the posterior distributions $q(z_i \mid x_i)$ with a common parameter θ. In other words, the variational distribution of z_i given x_i is a (known) function of x_i and θ.

Example Given a complex model $p(x_i \mid z_i)$, a common practice consists in approximating the posterior distribution $p(z_i \mid x_i)$ with a Gaussian distribution $q(z_i \mid x_i) = \mathcal{N}(\mu_i, \Sigma_i)$. In this situation, standard VI consists in finding the best means and covariances $\mu_1, ..., \mu_N, \Sigma_1, ..., \Sigma_N$ for each data point. On the contrary, Amortized VI would consist in considering the means and covariances as explicit functions of x_i and some parameter θ: $q(z_i \mid x_i) = \mathcal{N}(\mu_\theta(x_i), \Sigma_\theta(x_i))$. The function μ and Σ are often defined as neural networks.

Amortized VI speeds up dramatically the inference procedure, and allows furthermore to generalize inference to new observations by using the same common parameter θ. In turn, it introduces another source of error which, as we will see later, also needs to be taken into account.

2.3 Variational Auto-Encoders

2.3.1 Idea

Variational Auto-Encoders (VAE), also called **Variational Bayes** (VB) is a method introduced in 2013 by Kingma and Welling [31]. Since then, they have met a considerable success as a mean to use Bayesian uncertainty in auto-encoding models.

Consider a model $p(x, z, \theta)$ with latent variables (z, θ), where the hidden parameter θ is common to all observations. In this setting, VI would consist in finding an approximation of $p(z, \theta \mid x)$. The idea of VAEs is to consider θ deterministic and to learn it along with the variational distribution.

In other words, VI deals with the following optimization problem:

$$\max_{\varphi} \log p(x) - \mathrm{KL}(q_\varphi(z, \theta \mid x) \mid\mid p(z, \theta \mid x)) \tag{VI}$$

whereas VAE maximizes the following objective:

$$\max_{\theta, \varphi} \log p_\theta(x) - \mathrm{KL}(q_\varphi(z \mid x) \mid\mid p_\theta(z \mid x)) \tag{VAE}$$

Both methods maximize the ELBO, but the VAE is maximizes jointly in (θ, φ), which give their name to Variational Auto-Encoders. This method indeed outputs both a *encoding distribution* $q_\varphi(z \mid x)$, which allows to retrieve the hidden latent variable (the "code") from observation, and an *decoding distribution* $p_\theta(x \mid z)$, which gives the distribution of the observation x given the latent variable.

Remark 1. The term Variational Bayes is used when the distribution p and q are not specified. VAEs refer more specifically to neural network-related models.

2.3.2 Algorithm

It is worth noticing that performing simple maximum likelihood estimation (MLE) to find the best parameter θ would have been impossible in that scenario: the classical EM algorithm [15] requires an explicit expression of the posterior distribution $p_\theta(z \mid x)$. On the contrary, simple rewrites of the VAE's objective using the properties of the natural logarithm allows for SGD:

$$\begin{aligned} ELBO(\theta, \varphi, x) &= \log p_\theta(x) - \mathrm{KL}(q_\varphi(z \mid x) \mid\mid p_\theta(z \mid x)) \\ &= \mathbb{E}_{q_\varphi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}(q_\varphi(z \mid x) \mid\mid \ \log p_\theta(z)) \\ &= \mathbb{E}_{q_\varphi(z|x)} \left[p_\theta(x, z) - \log q(z \mid x)\right] \end{aligned}$$

This expression gives a first lead toward a SGD-based algorithm: indeed we get an expectation of known quantities. However, this expectation depends on the parameter φ hence the gradient of the expectation is not equal to the expectation of the gradient. To overcome this hurdle, the authors of [31] introduce the **reparameterization trick**. It simply consists in writing the $q_\varphi(z \mid x)$ distribution as a transformation of a non-parametric distribution. For instance, if $q_\varphi(z \mid x) = \mathcal{N}(\mu_\varphi(x), \Sigma_\varphi(x))$, one can write $z = g_\varphi(\varepsilon, x)$ with $\varepsilon \sim \mathcal{N}(0, I)$ and $g_\varphi(\varepsilon, x) = \mu_\varphi(x) + \sqrt{\Sigma_\varphi(x)}\varepsilon$. With this trick, the VAE's objective is rewritten as the following formula, which is amenable to Stochastic Gradient Descent. The numerical procedure is summarized in algorithm 1.

$$ELBO(\theta, \varphi, x) = \mathbb{E}_\varepsilon \left[p_\theta(x, g_\varphi(\varepsilon, x)) - \log q(g_\varphi(\varepsilon, x) \mid x)\right]$$

VAEs perform Bayesian inference while harnessing the power of neural networks and stochastic optimization. Their framework is very flexible and lends itself to adaptations in various fields, from image generation to shape auto-encoding [3]. They have met a considerable success and still are a very active research topic. The current research mostly focuses on improving the model's architecture [6, 25, 36, 48].

```
Data: $N$ observations $x$
while not converged do
    Sample mini-batch $x_1, ..., x_B$ ;
    Sample $\varepsilon_1, ..., \varepsilon_B \sim \mathcal{N}(0, I)$ ;
    Compute the estimate $\widehat{ELBO}(\theta, \varphi, x_{1:B}, \varepsilon_{1:B})$ ;
    Compute the gradient of $\widehat{ELBO}$ wrt. $(\theta, \varphi)$ ;
    Update $(\theta, \varphi)$ (SGD, Adam, RMSprop, ...) ;
end
Result: $\theta$ and $\varphi$
```

Algorithm 1: The original VAE algorithm from [31]

3 Theoretical study of Variational Auto-Encoders

3.1 State of the art and objectives

3.1.1 Existing theoretical results on Variational Bayes

Only a few recent articles tackle theoretical issues about Variational Bayes. Furthermore, they rather focus on the classical VI setting, where θ is a random variable to be determined by posterior inference.

In 2015, [29] proved the convergence of a proximal variant of SVI to a local minimum and gave a convergence rate (depending on the descent step sizes).

In 2018, two new results were proved on the *consistency* of the numerical algorithm, that is to say its convergence when the number of samples tends to $+\infty$. In [50], Wang and Blei prove that, when the number of sample increases, the variational posterior distribution of the latent variable θ converges normally toward the value of the true hidden parameter θ^*. This result is qualified as a *frequentist consistency* similar to the Bernstein-von-Mises theorem, as it guarantees the distributional convergence of a Bayesian algorithm, thereby proving its correctness from a frequentist point of view.

In a series of articles [11, 10], Chérief-Abdellatif proves that using a penalized version of the ELBO as a criterion for model selection results in a consistent numerical procedure: the posterior distribution in the selected model converges in expectation toward the true distribution as the number of samples tends to $+\infty$.

To the best of our knowledge, no theoretical study has yet been conducted on the proper VAE algorithm. While a great research effort is being produced in the field of VAEs, few is known about the theoretical aspect of the algorithm. This observation is more generally true in most areas where deep networks are involved: their structure makes it impossible or very difficult to prove strong results.

3.1.2 Motivation: the example of IWAEs

The original goal of my internship was to understand the overall way VAEs work. This question arises when considering the objective's structure compared with classical VI: the second one has a nice interpretation in terms of Kullback-Leibler divergence. We were thus interested to see what kind of mathematical behavior to expect when maximizing the ELBO not only in q but also in p: this is neither MLE nor VI.

The complexity of this algorithm illustrated by a recent research result [45], which we explain in this section. We can first notice that objective is structured as a tradeoff:

- The $\log p_\theta(x)$ term favors the correct maximum likelihood retrieval
- The Kullback-Leibler divergence guarantees that $q_\varphi(z \mid x)$ is close to $p_\theta(z \mid x)$.

Ideally, if the variational family is large enough, we will get a KLD close to zero and thus retrieve the correct MLE for p.

This might cause the impression that, if the ELBO gets tighter (closer to $\log p_\theta(x)$), the algorithm will improve accordingly. This idea drives Importance Weighted Auto-Encoders (IWAE, [6]), an extension of VAEs. IWAEs maximize the following ELBO:

$$\max_{\theta,\varphi} \mathcal{L}_K(\theta, \varphi, x) = \mathbb{E}_{q_\varphi(z|x)}\left[\log \frac{1}{k}\sum_{k=1}^{K} \frac{p_\theta(x, z_k)}{q_\varphi(z_k \mid x)}\right] \tag{IWAE}$$

When taking $K = 1$, we find the VAE's objective. Greater values of K simulate more samples z per observation, which are used in a way similar to importance sampling. The ELBO of IWAEs gets tighter as K increases:

$$\mathcal{L}_1 \leq \mathcal{L}_2 \leq ... \leq \mathcal{L}_K \xrightarrow[K\to+\infty]{} \log p_\theta(x)$$

The authors in [6] show that increasing K encourages the hidden variable to occupy more dimensions of the latent space, thus augmenting the model's representation power.

However, a recent study [45] shows that the intuition behind IWAE (tighter ELBO = better performance) should be questioned more thoroughly. The authors compute the order of magnitude of the signal-to-noise ratio (SNR) of the empirical ELBO's gradient. They find that the SNR of $\nabla_\theta \widehat{\mathcal{L}}_K$ increases with $O(\sqrt{K})$ (which is good), but that the SNR of $\nabla_\varphi \widehat{\mathcal{L}}_K$ is in $O(1/\sqrt{K})$. In other words, when the ELBO gets tighter, the gradient estimate in φ is essentially noise, which makes it useless for training.

3.1.3 What can we hope to get?

The example of IWAEs shows that the behavior of VAEs is not as intuitive as it may appear. Even though we obviously do not pretend nor hope to propose a complete rigorous theory, we found interesting to at least try to determine what kind of results classical theories allow to get in this context.

An important obstacle is the presence of neural networks in the problem. These functions satisfy very few hypothesis, being neither Lipschitz continuous nor convex nor bounded. Typically, it would be unreasonable to expect:

- **An explicit expression of the solution**: neural networks are complex, non invertible transformations. It is unrealistic to hope for an explicit expression of the best parameters (θ, φ). Similarly, we should not expect to get an explicit relationship between p and q at the optimum.
- **Convergence toward a global maximum**: as often in non-convex optimization, the convergence often happens only in local minima.
- **A good convergence rate**: in a non-convex setting, we can not hope for a linear convergence.

With those limitations in mind, we focused on trying to prove the convergence of VAEs toward a critical point of the objective function (and possibly a local maximum). This section explains our main attempts and what they resulted in.

3.2 First approach: a variant of SAEM algorithm?

Our first idea was to derive properties of the VAE algorithm by considering it as a variant of the SAEM algorithm, which is a variant of the EM algorithm. We briefly recall these methods.

EM algorithm Consider a model $p_\theta(x, z)$ where only x is observed. We want to retrieve the MLE estimator $\hat{\theta} = \text{argmax}_\theta\, p_\theta(x)$. The EM algorithm is an ascent algorithm which consists in two steps. At time n, we first compute a function $Q(\theta \mid \theta_n)$ such that $Q(\theta \mid \theta_n) \leq \log p_\theta(x)$ and $Q(\theta_n \mid \theta_n) = \log p_{\theta_n}(x)$. This function is defined by an expectation:

$$Q(\theta \mid \theta_n) = \mathbb{E}_{p_{\theta_n}(z|x)}\left[\log p_\theta(x, z)\right]$$

Next, we get $\theta_{n+1} = \mathrm{argmax}_\theta\, Q(\theta \mid \theta_n)$. By definition of Q, it is guaranteed that the log-likelihood is increases at each iteration. The process is summarized in algorithm 2. Since it was introduced in 1977, many theoretical results were proved on the EM algorithm. It is known that, for a wide class of problems, the algorithm converges toward a local maximum of the objective function for exponential families. The algorithm has become popular and numerous variants were developed to make it relevant in various contexts [7, 14, 28, 32, 33, 43, 44].

```
Data: Observation $x$, initial value $\theta_0$
while not converged do
    E-step: compute the expectation $Q(\theta \mid \theta_n) = \mathbb{E}_{p_{\theta_n}(z|x)}[\log p_\theta(x, z)]$ ;
    M-step: maximize $Q$ in $\theta$ to get $\theta_{n+1} = \mathrm{argmax}_\theta\, Q(\theta \mid \theta_n)$ ;
end
Result: final $\theta_N$
```

Algorithm 2: The EM algorithm [15]

SAEM algorithm The Stochastic Approximation Expectation Maximization [14] was introduced in 1999 as a mean to perform maximum likelihood when the expectation computation in the EM algorithm is intractable, but one can still generate samples from the distribution $p_{\theta_n}(z \mid x)$. The idea is to approximate the Q function using by generating one sample per step and averaging the empirical versions of Q we thus obtain:

$$Q_{n+1}(\theta) = (1 - \gamma_n)Q_n(\theta) + \gamma_n \log p_\theta(x, z_n)$$

The algorithm is particularly efficient in the setting of exponential families. In this case, we have $\log p(x, z) = \langle S(x, z), \phi(\theta)\rangle + \phi(\theta)$. We only need to save the updated value of the exhaustive statistic S and not the entire Q functions:

$$S_n = (1 - \gamma_n)S_{n-1} + \gamma_n S(x, z_n), \quad Q_n(\theta) = \langle S_n, \phi(\theta)\rangle + \psi(\theta)$$

The procedure is summarized in algorithm 3.

```
Data: Observation $x$, initial value $\theta_0$
while not converged do
    E-step:
    a) Simulate $z_n \sim p_{\theta_n}(z \mid x)$ ;
    b) Update $S_{n+1} = (1 - \gamma_n)S_n + \gamma_n S(x, z_n)$ ;
    M-step: compute $\theta_{n+1} = \mathrm{argmax}_\theta\, Q_{n+1}(\theta) = \mathrm{argmax}_\theta \langle S_n, \phi(\theta)\rangle + \psi(\theta)$ ;
end
Result: final $\theta_N$
```

Algorithm 3: The SAEM algorithm [14]

Similarity between VAE and SAEM Our initial intuition was about the resemblance between the SAEM algorithm and VAEs. Indeed, the ELBO is an expectation over $q_\varphi(z \mid x)$ which, in an ideal scenario, is supposed to be very close to $p_\theta(z \mid x)$. Hence each step of the algorithm increases the value of a function similar to the expectation of a log-likelihood over the posterior distribution. Two main differences were to consider:

- The expectation is taken over a distribution which is not the true posterior distribution
- The SAEM algorithm makes a full maximization at each time step, whereas the VAE is a gradient ascent algorithm.

The second item would probably not be an issue, since other (provably convergent) variants of the EM algorithm exist, where the maximization is not tractable and a gradient step is performed instead. On the contrary, the first item surprisingly turned out to break the analogy when it came to adapting the proof of convergence of the SAEM algorithm.

Essentially, the log-likelihood function $\ell(\theta) = \log p_\theta(x)$ verifies a nice gradient equality:

$$\begin{aligned}
\nabla_\theta \ell(\theta) &= \nabla_\theta \log p_\theta(x) \\
&= \nabla_\theta \log \int p_\theta(x,z)\,\mathrm{d}z \\
&= \frac{\int \nabla_\theta p_\theta(x,z)\,\mathrm{d}z}{\int p_\theta(x,z)\,\mathrm{d}z} \\
&= \int \frac{1}{p_\theta(x)} p_\theta(x,z) \nabla_\theta \log p_\theta(x,z)\,\mathrm{d}z \\
&= \mathbb{E}_{p_\theta(z|x)}[\nabla_\theta \log p_\theta(x,z)]
\end{aligned}$$

This equation allows the authors to smoothly overcome some technical hurdles. This formula does not hold true anymore when using the objective function of VAEs $\hat{\ell}(\theta,\varphi) = \mathbb{E}_{q_\varphi(z|x)}[\log p_\theta(x,z) - \log q_\varphi(z \mid x)]$. The cause is the absence of connection between the p and q distributions.

Despite the proof not lending itself to a direct adaptation (which would have been suspiciously easy), some important insights can be drawn from its overall structure. First, the core of the proof is to apply a powerful theorem from **stochastic approximation theory** (SA). This theory was designed to study the behavior of deterministic and stochastic approximation algorithms like the Robbins-Monro procedure, and turned out to be particularly well suited for this application. Second, the hypotheses of the theorem and their role in the proof helped me understand what could be the limitations in our VAE setting: crucially, some assumptions on the boundedness of the parameter or the growth of the functions will be required.

3.3 Second approach: Stochastic Approximation theory

Even though the apparent connection between the VAE and the SAEM ended up not working, it made it clear that the underlying theory to be used in these common frameworks is that of stochastic approximation. We first briefly present this framework, as well as some important related results. Then we consider the VAE algorithm under this angle and derive a theorem with hypotheses similar to those of the SGD algorithm.

3.3.1 Main theorem

This subsection mainly takes stock on the comprehensive review on stochastic approximation (SA) on the topic from Delyon [13]. His approach considers first deterministic algorithm and proves almost-sure results by applying deterministic theorems point-wise.

The algorithm we consider in this setting is defined by the recursive equation:

$$s_n = s_{n-1} + \gamma_n h(s_{n-1}) + \gamma_n e_n + \gamma_n r_n.$$

$s_n \in \mathcal{X}$ is the parameter which is updated at each step, γ is the step size or learning rate, e_n and r_n are two (**deterministic**) error sources. h is the update: in the VAE setting it will be the gradient of the ELBO. The goal is to find conditions under which $h(s_n) \to 0$.

First, two canonical assumptions ensure the convergence of the algorithm.

Definition 1 (Hypothesis (A))**.** The algorithm satisfies the hypothesis if the function h is **continuous** over $\mathcal{X}$ and there exists a C^1 **Lyapunov function** $V : \mathcal{X} \to \mathbb{R}$ such that:

- $F(s) = \langle \nabla V(s), h(s) \rangle \leq 0$

- For $\mathscr{S} = \{s \mid \langle \nabla V(s), h(s) \rangle = 0\}$, we have $\mathrm{Int}(V(\mathscr{S})) = \emptyset$

Definition 2 (A-stability)**.** The algorithm is **A-stable** if s_n stays in a compact set of $\mathcal{X}$, $\sum_n \gamma_n e_n$ converges and $r_n \to 0$.

These two assumption grant the convergence toward $\mathscr{S}$:

Theorem 1. *If hypothesis (A) is satisfied and the algorithm is A-stable, then $d(s_n, \mathscr{S}) \to 0$. Furthermore, if $\mathscr{S}$ is finite, then s_n converges toward an element or $\mathscr{S}$.*

Note that we don't know for sure that $h(s_n) \to 0$. We will see later that this problem can be overcome when it comes to applying the theorem. These two main conditions (existence of a Lyapunov function and boundedness of s_n) are the core SA hypotheses, and the theoretical challenge thus consists in weakening those assumptions into more flexible results.

Remark 2. The definition of the Lyapunov function may look opaque at the first glance. It has a nice interpretation in the continuous limit: when $\gamma_n = \gamma \to 0$, let aside the error terms, the variable s_n becomes a time continuous function satisfying a first-order differential equation:

$$\frac{s_n - s_{n-1}}{\gamma} = h(s_n) \xrightarrow[\gamma \to 0]{} \frac{\mathrm{d}s}{\mathrm{d}t} = h(s_t)$$

And the definition of V translates into:

$$\frac{\mathrm{d}V(s_t)}{\mathrm{d}t} = \langle V(s_t), h(s_t) \rangle \leq 0$$

Hence the Lyapunov function should be interpreted as a function which is minimized along the trajectory of the algorithm. This consideration will prove useful when it comes to exhibiting such a function.

3.3.2 Chen's reprojection method

In order to get rid of the compacity assumption, the **reprojection method** is introduced in [13]. This method consists in controlling the sequence s_n, and restarting from increasingly large random starting points. Consider an increasing sequence of compacts K_n covering $\mathcal{X}$. The method is described in algorithm 4. It is illustrated in fig. 1 with a spiral, which allows to visually understand the reprojection. This example does of course not represent a stochastic approximation process, which would tend to slow down and converge toward the set of stationary points.

```
Data: Starting point $s_0$, first compact index $c_0$
while not converged do
    Update $s_n \to s_{n+1}$ ;
    if $s_{n+1} \subset K_{c_n}$ then
        $c_{n+1} = c_n$ ;
    else
        $c_{n+1} = c_n + 1$ ;
        $s_{n+1} \sim \mathcal{U}(K_0)$ ;
    end
end
Result: $s_N$
```

Algorithm 4: Chen's reprojection method

At the first glance, it seems that reprojections heavily affect the sequence's behavior, which questions the convergence the algorithm. One thus might wonder about the purpose of the reprojections.

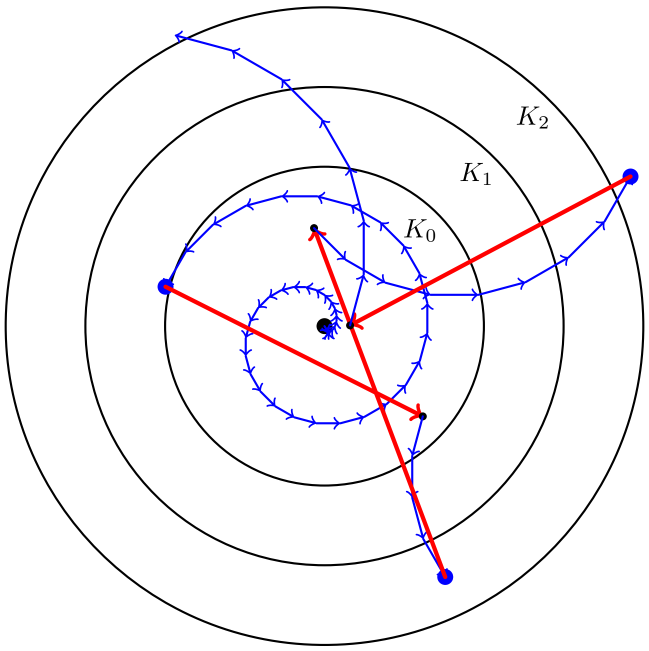

Figure 1: A spiral starting from the center and reprojected on a increasing series of circles. The reprojections are represented by red arrows.

It turns out that under certain additional hypotheses, the reprojections only happen a finite amount of times. This means that the sequence asymptotically stays in some (random) compact set, granting the boundedness we need to prove the convergence. We state below the extra hypotheses which grant the A-stability.

Definition 3 (Hypothesis (B))**.** The algorithm satisfies the hypothesis if the function h is continuous over $\mathcal{X}$ and there exists a C^1 Lyapunov function $V : \mathcal{X} \to \mathbb{R}$ and a compact K such that:

- $F(s) = \langle \nabla V(s), h(s) \rangle < 0$ if $x \notin K$
- $V(s) \to +\infty$ if $s \to \partial \mathcal{X}$ or $|x| \to +\infty$.

Theorem 2. *Consider the reprojected algorithm. If hypothesis (B) is satisfied and if, for all $M \in \mathbb{N}$,*

$$\sum_n \gamma_n e_n \mathbf{1}_{V(s_{n-1}) \leq M} \text{ converges and } \lim_n |r_n| \mathbf{1}_{V(s_{n-1}) \leq M} = 0$$

then the algorithm is A-stable (and the reprojections only occur a finite amount of times).

Remark 3. In this version of the theorem, we trade the boundedness of the sequence (s_n) against stronger properties on the Lyapunov function.

3.4 Application to VAEs

In the interest of brevity, we don't mention the theorem on stochastic approximation from [13] for the case where the error source (e_n, r_n) is random because it does not apply in our situation. Indeed it assumes that the set of solutions $\mathscr{S} = \{s \mid h(s) = 0\}$ is finite, which is not the case as we will see.
In this subsection, we will use the tools previously introduced to prove the following theorem:

Theorem 3. *Consider the VAE algorithm. We assume that p_θ and q_φ are Gaussian distributions parameterized by neural networks with upper bounded covariance matrices: $\Sigma_\theta(z) \succ \epsilon I$, $\Sigma_\varphi(x) \succ \epsilon I$. The network's activation function is supposed to be smooth and Lipschitz. Then, if the networks' parameters (θ, φ) stay bounded, the stochastic gradient descent for VAEs converges with probability one to a critical point of the ELBO.*

3.4.1 Connection between SA and VAE

In the VAE setting, our parameter is $s = (\theta, \varphi)$ and we consider the ELBO over a dataset $(x_1, \ldots, x_N)$:

$$\mathcal{L}(\theta, \varphi) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{q_\varphi(z|x_i)}[\log p_\theta(x_i, z) - \log q_\varphi(z \mid x_i)]$$

The objective is to find s such that $\nabla \mathcal{L}(\theta, \varphi) = 0$, hence we will use $h(s) = \nabla \mathcal{L}(s)$. Since h depends on the action of neural networks, which can be constant in some regions of the parameters space, it is clear that the set of critical points may be infinite and even unbounded. The traditional gradient descent algorithm writes

$$s_{n+1} = s_n + \gamma_n h(s_n) + \gamma_n e_n$$

with $e_n = \nabla \widehat{\mathcal{L}}(s_n) - h(s_n)$ the difference between the gradient of the empirical ELBO and the true gradient, and $r_n = 0$. As described in algorithm 1, the empirical ELBO is computed on a mini-batch with one sample of z using the reparameterization trick: $z = g_\varphi(\varepsilon, x), \varepsilon \sim P(\varepsilon)$. In order to be able to do concrete mathematical computations, we assume that $p_\theta(x \mid z)$ and $q_\varphi(z \mid x)$ are Gaussian distributions parameterized by neural networks. We also define a Gaussian prior for z: $p_\theta(z) = \mathcal{N}(0, I)$, which is a classical choice for continuous latent variables.

$$p_\theta(x \mid z) = \mathcal{N}_x(\mu_\theta(z), \Sigma_\theta(z)), \quad q_\varphi(z \mid x) = \mathcal{N}_x(\mu_\varphi(x), \Sigma_\varphi(x))$$

The functions μ and Σ are considered multi-layer perceptrons. Notice that our reasoning would also work for general feed-forward networks like convolutional networks. Besides, in order to get symmetric positive-definite covariance matrices, the matrix Σ is often parameterized as $\Sigma = U^t U$ with U a triangular matrix with non-zero diagonal coefficients whose coefficients are produced by a neural network. *We will restrict ourselves to the case where the covariances are diagonal matrices.* This assumption allows for significant computations alleviation. It is also often used numerically to produce faster codes [17], thus avoiding evaluation - and backpropagation - of matrix determinants. Furthermore, for theoretical convenience purpose, we assume that the activation function σ_{act} used in the networks is Lipschitz continuous, and differentiable a number of times greater than $\dim(x)$ and $\dim(z)$.
In the next section, we derive brutal yet useful bounds on neural networks and the loss function. In the two following subsections, we try two different Lyapunov functions and derive related convergence theorems.

Remark 4. A paper from Monnez [39] proves a very similar result on the almost sure convergence of SGD, but it makes strong regularity assumptions on the objective function. In our neural-network based context, those conditions are not met.

3.4.2 On the growth of neural network functions

As we went through the computations, it turned out that bounds on the growth of neural network functions could help in various ways. By using naive upper bounds, we derived a bound on neural network outputs and their gradient with respect to the network's parameters. All the results are proved in appendix A.

First we bound the growth of the neural network itself:

Proposition 1. *Let $f_\theta(x)$ a neural network with L layers with no activation on the final layer, with k-Lipschitz activation function σ_{act}. Then we have $|f_\theta(x)| \leq C(1 + \|\theta\|)^L(1 + \|x\|)$, where C is a constant depending on L, k and $\sigma_{act}(0)$.*

This bound can be used to bound the growth of the ELBO $\mathcal{L}$. It appears in the computation that, without further hypothesis, the covariance matrix can get non-invertible arbitrarily fast in any portion of the space. In order to avoid this undesired behavior, we impose that the covariance stays uniformly positive: $\exists \epsilon, \forall \theta, \varphi, z, x, \Sigma_\theta(z) \succ \epsilon I$ and $\Sigma_\varphi(x) \succ \epsilon I$. This condition can be numerically implemented very easily with the parameterization $\Sigma = U^t U$ by adding ϵI to the positive diagonal coefficients (and making them positive). We now call L the maximum number of layers in any considered neural network.

Proposition 2. *Under the previous assumptions, there exists a constant C depending on $L, k, \sigma_{act}(0)$, $N, \dim z$ and the norms $\|x_i\|$ such that $|\mathcal{L}(\theta, \varphi)| \leq C(1 + \|\theta\|)^{2L}(1 + \|\varphi\|)^{2L}$.*

Next, we derived an upper bound on the growth of the **gradient** of a neural network output with respect to its parameters:

Proposition 3. *Let f_θ a neural network with L layers without activation on the last layer, σ_{act} a smooth and k-Lipschitz activation function. Then there exists a constant C depending on L, k and d such that $|\partial_\theta[f_\theta(x)]| \leq C(1 + \|\theta\|)^{2L}(1 + \|x\|)$ and $|\partial_x[f_\theta(x)]| \leq C(1 + \|\theta\|)^L$.*

We used this bound as well as the previous ones to derive a bound on the gradient of the ELBO:

Proposition 4. *Under the previous hypotheses, there exists a constant C depending on $L, k, \sigma_{act}(0)$, $N, \dim z$ and the norms $\|x_i\|$ such that $|\nabla\mathcal{L}(\theta, \varphi)| \leq C(1 + \|\theta\|)^{4L}(1 + \|\varphi\|)^{4L}$.*

Remark 5. The bounds above are certainly not tight. For flat and shallow networks they produce very bad estimates. At a given number of parameters, the worst case is a very deep network with one neuron per layer: in this case the network's output function may have a polynomial growth with maximal degree. Perhaps smarter proof techniques would have allowed to replace the $4L$ exponent with L. However a point we make here is that, whatever the network's architecture, there can be a direction in the parameters space where the objective function and its gradient grow like a polynomial.

3.4.3 First Lyapunov function: bounded case

As for classical gradient descent, the recommended Lyapunov function for stochastic gradient descent is the objective function itself. Let us consider $V(s) = -\mathcal{L}(s)$. We get automatically $F(s) = \langle \nabla V(s), h(s) \rangle = -\|h(s)\|^2 \leq 0$, which meets the first condition of hypothesis (A). The second condition ($\text{Int}(V(\mathscr{S})) = \emptyset$) can be checked with Sard's theorem:

Theorem 4 (Sard)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a C^n function. Then the image of the set of critical points of f has Lebesgue measure zero: if $\mathscr{S} = \{s \mid \nabla f(s) = 0\}$, then $|f(\mathscr{S})| = 0$.*

If its interior was not empty, $\text{Int}(V(\mathscr{S}))$ would contain a ball with positive radius, and thus have a positive Lebesgue measure. **Therefore hypothesis (A) is satisfied**. It remains to be seen wether the algorithm is A-stable, that is to say wether s_n is bounded and $\sum_n \gamma_n e_n$ converges. Unfortunately, theorem 2 does not apply because we don't have any information on the asymptotic growth of the loss function: as it contains neural networks, it is not coercive, and can even stay constant in a given direction beyond a given point. After exploring various directions, we concluded that this problem could not be avoided, and resolved to add a strong hypothesis to our theorem. **In this section, we assume that the variable s_n is bounded almost surely.** This makes hypothesis (B) automatically satisfied. In the next section, we will be looking for a penalized version of the original optimization problem which leaves the variable bound-free.

Since we assume that the variable s_n is bounded, we only need to prove that the series $\sum_n \gamma_n e_n \mathbf{1}_{V(s_{n-1})}$ converges to apply theorem 2. With this in mind, we will use a martingale theorem from Hall and Heyde [24] (theorem 2.15 p.33):

Theorem 5. *Let $S_n = \sum_{i=1}^n X_i$ a square integrable $\mathscr{F}$-martingale with mean zero. Then S_n converges almost surely to a finite limit on the set $\{\sum_{n=0}^{\infty} \mathbb{E}[X_{n+1}^2 \mid \mathscr{F}_n] < \infty\}$.*

Let $\mathscr{F}$ be the filtration defined by the algorithm. Let $M \geq 0$, $X_n = \gamma_n e_n \mathbf{1}_{V(s_{n-1})}$ and $S_n = \sum_{i=1}^n X_i$. We have the following lemma, which proves that S_n is a zero-mean, square integrable $\mathscr{F}$-martingale.

Lemma 1. *For all $n \in \mathbb{N}$, $\mathbb{E}[e_n | \mathscr{F}_n] = 0$, and the variance $\mathrm{Var}(e_n | \mathscr{F}_n)$ is finite.*

Another lemma proves that the convergence condition for theorem 5 is met almost surely:

Lemma 2. *Assume that $\sum_n \gamma_n^2 < +\infty$. Then $\sum_{n=0}^{\infty} \mathbb{E}[X_{n+1}^2 \mid \mathscr{F}_n] < \infty$ with probability one.*

Both proofs are detailed in appendix A. As a consequence, we can apply theorem 2, which states that the reprojected algorithm is A-stable. Since hypothesis (A) is also satisfied, theorem 1 applies and the algorithm converges almost-surely toward a critical point of the objective function.

3.4.4 Second Lyapunov function: unbounded case?

We now seek a theoretical result which would get rid of the boundedness constraint on s. This requires to find a Lyapunov function which satisfies hypothesis (B) over the whole set of parameters. We already have a function $V(s) = \mathcal{L}(s)$ which satisfies $F(s) = \langle \nabla V(s), h(s) \rangle \leq 0$, but we can not guarantee that $F(s)$ is strictly negative beyond some compact set. This is in fact even false, since neural networks may have unbounded plateaus, hence unbounded critical areas. For the same reason, we can not say that $V(s)$ is coercive.
We shall thus consider another Lyapunov function. However, V can not be any function and the relation ship $\langle \nabla V(s), h(s) \rangle$ is very constraining. We did not find any clear way to derive other Lyapunov functions. In turn, we tried to modify the original objective function by adding a regularization term. We first tried to add a L^2 penalization: since the ELBO is upper bounded (as we will see soon) and the function inside of the expectation has a neural network-like structure, there is hope that, outside of some compact set, the gradient of the penalized version would never vanish. However we could not find a proper way to prove this property, even though we believe it is true.

The second option we found is far less attractive: it consists in adding a penalty so large that its gradient dominates the upper bound 4 on the ELBO's gradient. This translates into the following penalized problem:

$$\min_{\theta, \varphi} \mathcal{K}(\theta, \varphi) = -\mathcal{L}(\theta, \varphi) + \lambda (1 + \|\theta\|)^{4L+1} (1 + \|\varphi\|)^{4L+1} \tag{VAE-P}$$

Since we assumed that the covariance are lower bounded, the ELBO $\mathcal{L}$ is upper bounded:

$$\begin{aligned}
\mathcal{L}(\theta, \varphi) &= \log p_\theta(x) - \mathrm{KL}(q_\varphi(z \mid x) \mid\mid p_\theta(z \mid x)) \\
&\leq \log p_\theta(x) = \log \int p_\theta(x \mid z) p_\theta(z) \mathrm{d}z \\
&\leq \log \int \frac{1}{\sqrt{(2\pi)^D |\Sigma_\theta(z)|}} \exp\left(-\frac{1}{2} (x - \mu_\theta(z))^t \Sigma_\theta(z)^{-1} (x - \mu_\theta(z)) \right) p_\theta(z) \mathrm{d}z \\
&\leq \log \int \frac{1}{\sqrt{(2\pi)^D |\Sigma_\theta(z)|}} .1. p_\theta(z) \mathrm{d}z \\
&\leq -\frac{D}{2} \log(2\pi\epsilon)
\end{aligned}$$

Therefore $\mathcal{K}(\theta, \varphi) \geq \frac{D}{2}\log(2\pi\epsilon) + \alpha\|\theta\|^2 + \beta\|\varphi\|^2$ is coercive. Furthermore, by definition, its gradient can not be zero since the regularization part has a gradient which dominates the ELBO's gradient (see proposition 4). Thus, taking $V(s) = -\mathcal{K}(s)$ grants that $\langle \nabla V(s), h(s) \rangle < 0$, and hypothesis (B) is thereby satisfied. Sard's theorem also grants that hypothesis (A) is satisfied, and lemma 2 from previous subsection still applies: hence we can apply theorem 2, and the algorithm is A-stable almost surely. Theorem 1 allows to conclude that the algorithm converges almost surely toward a critical point.
This result is not really what we hoped for, since the penalty is extremely strong as soon as the network's depth increases. The technical hurdle mentioned two paragraphs above keep us from being able to state a stronger result on L^2 penalties. Notice that the penalty can be chosen with an arbitrarily small coefficient λ and that we may decide to apply it only after a certain norm threshold:

$$\mathcal{K}(\theta, \varphi) = -\mathcal{L}(\theta, \varphi) + \lambda(1 + \mathbf{1}_{\|\theta\|>T}(\|\theta\| - T))^{4L+1}(1 + \mathbf{1}_{\|\varphi\|>T}(\|\varphi\| - T))^{4L+1}$$

In that regard, this result is very similar to the previous one, since we constrain the parameters to stay in a bounded set, this time by applying a strong penalty when overstepping the domain's boundaries.

Remark 6. In this section, we had to use a very strong L^p penalty to get the a.s. convergence. We had to apply a strong correction because we could not prove that, for smaller penalties, the objective function has no critical point beyond a certain compact. However we believe that an L^2 penalty should be sufficient: in proposition 5, we make two additional assumptions. We assume that the activation function is piecewise polynomial, and we restrict the support of the reparameterization variable ε to a ball $B(0, R)$. Under these extra assumptions, proposition 5 grants that, for every parameter s with unit norm, there exists a threshold T_s beyond which, if $t > T_s$, the L^2-penalized VAE objective has no critical point. This is *not* equivalent to the condition we desire, because the threshold T_s is hard to bound over the unit sphere: in some directions, the threshold may diverge. We believe that the threshold is actually bounded, but could not prove it.

Proposition 5. *Suppose that $q_\varphi(z \mid x)$ is drawn from $g_\varphi(x, \varepsilon)$ with $\varepsilon \sim \mathcal{N}_{<R}(0, I)$ a truncated normal. Suppose moreover that the activation function σ_{act} is (finitely) piecewise polynomial. Then, in every direction, the L^2 penalized VAE objective has no critical point beyond a certain distance (dependent on the direction) to the origin.*

3.4.5 Summary

Overall, we proved two results, which both prove that the VAE algorithm almost surely converges toward a critical point. Since the second result could be considered an overall way to bound the variable s, which is a hypothesis for our first result, we do not mention it here.

Theorem 3. *Consider the VAE algorithm. We assume that p_θ and q_φ are Gaussian distributions parameterized by neural networks with upper bounded covariance matrices: $\Sigma_\theta(z) \succ \epsilon I$, $\Sigma_\varphi(x) \succ \epsilon I$. The network's activation function is supposed to be smooth and Lipschitz. Then, if the networks' parameters (θ, φ) stay bounded, the stochastic gradient descent for VAEs converges with probability one to a critical point of the ELBO.*

The main restrictions are the covariance lower bounds and the boundedness of the parameters. The first one is natural and may help with the numerical stability of the algorithm. The second one is present by theoretical necessity. Note that since in real life the parameters almost always never diverge when training neural networks, this result still has some value as a theoretical insight: it confirms that in most real-life cases, the algorithm actually converges toward a point of interest.

4 Understanding the inference gap

In a second part of my internship, we made some numerical experiments and tried to get a concrete understanding of the numerical procedure of VAEs.

We started by implementing the VAE algorithm on simple examples, namely a Gaussian mixture model and the MNIST dataset. The objective was to understand what solutions were produced by the algorithm in an situation, and track the evolution of the different terms in the ELBO.

4.1 Gaussian mixture

4.1.1 Model and variational family

The Gaussian Mixture Model (GMM) is a generative latent variable model: we observe the superposition of samples from $K > 1$ Gaussian densities, but we don't know the mean and covariances of the distributions, nor which point belongs to which distribution. The inference consists in retrieving this information. More formally, the model is defined by K Gaussian distributions $\mathcal{N}(\mu_1, \Sigma_1), ..., \mathcal{N}(\mu_K, \Sigma_K)$. The observation x is generated along with a latent integer z which indicates the index of the related mode. z is drawn according to a multinomial distribution with parameter π. The variable θ regroups the means and covariances of the clusters. The model is summarized below:

$$\begin{cases} p_\theta(z) = \mathcal{M}(\pi_1, ..., \pi_K) \\ p_\theta(x \mid z) = \mathcal{N}(\mu_z, \Sigma_z) \end{cases}$$

Performing inference for a GMM is a classical task, at which the EM algorithm is far more suitable than our method, which could be considered as a relaxation of the EM algorithm. But we found interesting to apply the Variational Bayes algorithm in this simple setting so as to make sure to get interpretable results.

Variational distribution As we work in the VB setting, we need to specify a variational family for the decoding encoding distribution. Note that, in the GMM, we have an explicit expression of the posterior density $p_\theta(z \mid x)$, but we voluntarily choose to ignore this expression. Thus we choose a deliberately wrong family of distributions: we define a model q_φ with an observation x and a discrete latent variable z, with z following a multinomial distribution. However, the distribution of x is not Gaussian: the angle direction around the mean is uniform,but the radius follows a Laplace distribution instead of a normal distribution.

$$q_\varphi(x \mid z) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_z^\varphi|}} \exp\left(-\sqrt{(x - \mu_z^\varphi)^t \left(\Sigma_z^\varphi\right)^{-1} (x - \mu_z^\varphi)}\right)$$

We make this choice so as to ensure that the parameters of the distribution p are uncorrelated from those of q. Therefore, just like p, the variational distribution q has a parameter π^q for the prior distribution of z, and a parameter for each mean and covariance. We compute the explicit value of $q_\varphi(z \mid x)$ using Bayes rule.

Numerical implementation with Pytorch We used the open source library Pytorch to perform automatic differentiation and effortlessly compute the gradients we needed. However, while implementing the algorithm, we came upon a problem with the reparameterization trick.

4.1.2 Differentiating through discrete latent variables

A note on the reparametrization trick Remember that the practical ELBO maximization relies on the reparameterization trick, which allows to rewrite the ELBO from $\mathbb{E}_{q_\varphi(z|x)}[\dots]$ (hard to differentiate) to $\mathbb{E}_{\epsilon \sim P(\epsilon)}[\dots]$ (we can put the gradient in the expectation). It consists in writing $z = g_\varphi(\epsilon, x)$ to describe the distribution $q_\varphi(z \mid x)$. However, in order for the reparameterization trick to work, the function g obviously needs to be differentiable. Unfortunately, for GMMs the latent variable is discrete, so that the trick will not work that easily.

Option 1: the REINFORCE rule The first idea we came up with was to find another way to compute the ELBO's gradient. Namely, we directly computed the derivative of the expectation:

$$\begin{aligned}
\mathbb{E}_z[f(\theta,\varphi,z)] &= \int f(\theta,\varphi,z)q_\varphi(z)\mathrm{d}z \\
\partial_\varphi\mathbb{E}_z[f(\theta,\varphi,z)] &= \int \partial_\varphi f(\theta,\varphi,z)q_\varphi(z) + f(\theta,\varphi,z)\partial_\varphi q_\varphi(z)\mathrm{d}z \\
&= \mathbb{E}_z[\partial_\varphi f(\theta,\varphi,z) + f(\theta,\varphi,z)\partial_\varphi \log q_\varphi(z)]
\end{aligned}$$

This equation yields a new expectation, which is interesting because it provides an unbiased estimator for the gradient and does not need the reparameterization trick. This estimator is the same as the one used for the REINFORCE algorithm in Reinforcement Learning. We tried this solution and implemented it, but the algorithm would not converge because of the high variance of the gradient estimate.

Option 2: the Gumbel-Softmax distribution We found that a specific method was developed to deal with this situation. It was proposed in 2016 simultaneously by [20] and [38]. We adopt the formalism from the first article, for it had the most influence in the posterior litterature. The first idea of the Gumbel-Softmax (GS) distribution is to consider a one-hot encoding of the latent variable. That is to say, we convert $z = k$ to e_k the k-th vector of the canonical base of $\mathbb{R}^K$. The next step is to realize that all the new possible values of z are the extremal points of the simplex of $\mathbb{R}^K$, i.e. $\Delta_K = \{y \in \mathbb{R}_+^K \mid \sum_{i=1}^K y_i = 1\}$. The Gumbel-Softmax with temperature T is a relaxation of this distribution over the simplex: the new variable z will be a point in Δ_K.

Suppose we want to sample from the GS approximation of a multinomial distribution $(\pi_1, ..., \pi_K)$. The procedure is as follows:

```
**Data:** probabilities $\pi$, temperature $T$
Generate $g_1, \dots, g_K \sim \text{Gumbel}(0,1)$ ;
Compute $a_k = g_k + \log \pi_k$ for $1 \le k \le K$ ;
Compute $y_k = \text{Softmax}(a)_k = \frac{\exp(a_k/T)}{\sum_{i=1}^K \exp(a_i/T)}$ for $1 \le k \le K$ ;
**Result:** $y$
```

Algorithm 5: Sampling from the Gumbel Softmax distribution

The temperature parameter T controls the smoothness of the relaxation. As shown in figure 2, a small temperature yields a distribution very close to the original categorical distribution, wheras a bigger temperature tends to produce more uniform samples. There is a numerical tradeoff between both extremes: $T \to 0$ produces good samples but the related gradient estimator has a very large variance, whereas larger T will have a better gradient variance and worse correspondence with the original discrete distribution.

The main advantage of the GS distribution is to produce fully differentiable samples: the samples write indeed $z = F(\pi_1, ..., \pi_k, g_1, ..., g_K)$ with $g_k \sim \text{Gumbel}(0,1)$. They also allow to adjust the smoothing factor; in our experiments, we arbitrarily set this factor to $T = 0.5$, which we empirically found out to be a good compromise between both extremes.

Finally, in the VAE sampling step, we replace sampling the discrete variable $z \in \{1, ..., K\}$ by sampling from the GS distribution $(z_1, ..., z_K)$. The replacement is executed as follows:

$$f(z) = \sum_{k=1}^{K} \mathbf{1}_{z=k} f(k) \quad \Longrightarrow \quad \sum_{k=1}^{K} z_k f(k)$$

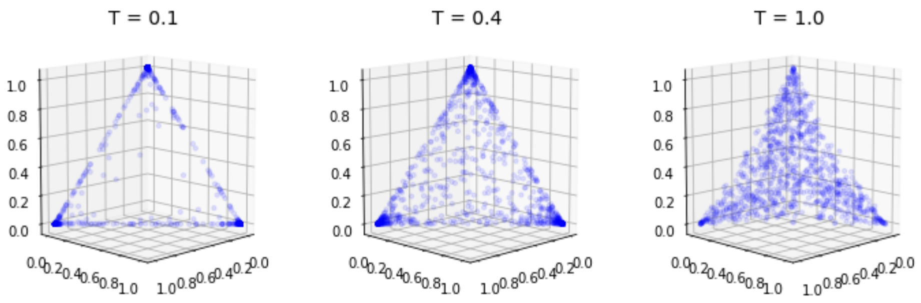

Figure 2: 1000 samples from the Gumbel-Softmax distribution with temperatures $T \in \{0.1, 0.4, 1\}$

Remark 7. To a lesser extent, the parametrization of π and Σ also required some work: we need to insure the constraints $\pi \in \Delta_K$ or $\Sigma \in S_n^{++}(\mathbb{R})$. We chose to express π as a softmax of auxiliary variables: $\pi = \text{Softmax}(\xi)$. In order to lighten the computations, we used diagonal covariance matrices, and constrained the diagonal parameters to be positive by reparameterizing them as $\Sigma_{ii} = \log(\exp(\lambda_i) + 1)$. Finally we performed SGD on ξ and λ instead of π and Σ.

4.1.3 Numerical results

We ran the algorithm in dimension two for 1000 steps with learning rate 0.1 with various number of clusters and cluster shapes. The algorithm turned out to work poorly with a large number of widely spread clusters. This is the consequence of using SGD in a highly non convex optimization problem with a relatively small number of parameters. We focus on cases with a small number of clusters ($K \leq 5$) and present here results with $K = 3$, which give the most interpretable results.

Interpreting the position of the means and covariances The algorithm's output after 1000 iterations is displayed in figure 3. Several comments can be made on this result. First, the clusters are correctly identified: no cluster contains points from two different distributions. Because of that, the means of the p_θ model are correctly identified and match the original means almost perfectly. Notice that the covariance matrices of p are not well adjusted: this is due to their very slow convergence (had we waited 20000 iterations that they would be in the right place).

However, when it comes to the parameters of the q distribution, the convergence seems much worse *a priori*: the means do not adjust at all with the real cluster centers. Furthermore, they all seem to be offset outward. In fact, this result can be easily interpreted. Remember that, when maximizing the ELBO VAE, we only take into account the mathematical expression $\text{KL}(q_\varphi(z \mid x) \mid\mid p_\theta(z \mid x))$ for q. This particularly means that we do not care about maximizing the likelihood of the observation $q_\varphi(x)$, which would require a more precise estimation of the means (comparatively, the means for p are better because the likelihood $\log p_\theta(x)$ is taken into account into the ELBO). The only criterion that matters is how likely the point classification is. Moreover, the likelihood of a given class evolves according to fuzzy decision boundaries, and the choice of a slightly outward offset guarantees that the uncertainty between the classes is reduced. Figure 4 shows that, if on the contrary the means of q_φ were offset inward the probabilities would tend to get closer one class from another.

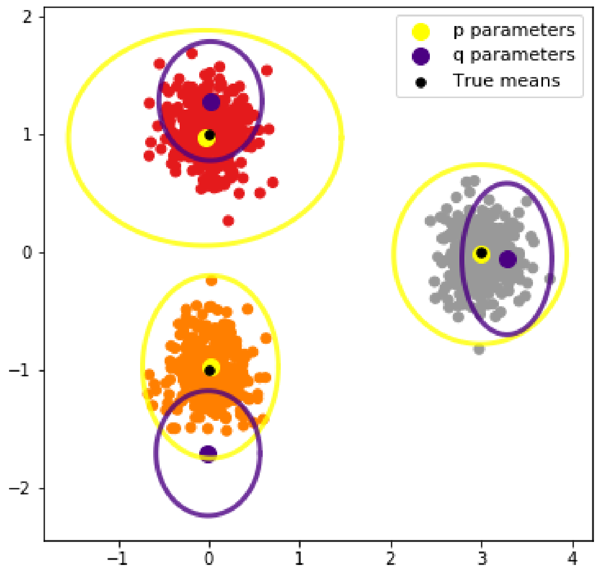

Figure 3: VAE algorithm applied to a mixture of 3 Gaussian distributions. The black dots represent the true cluster means. The yellow dots and ellipses are the mean and 95% Gaussian confidence ellipses of the covariance matrices of the encoder p_θ. The green dots and ellipses are the mean and 95% Laplace confidence ellipses of the decoder q_φ.

Time evolution of the objective components We track along the simulation the terms in the ELBO: first the $\log p_\theta(x)$ term, for which we have an explicit expression in this model. However, we do not have access to the explicit expression of the Kullback-Leibler divergence $\mathrm{KL}(q_\varphi(z \mid x) \mid\mid p_\theta(z \mid x))$. We estimate it by sampling from q at each step. The result is displayed in figure 5. As we can see, the KL divergence diminishes while the log-likelihood of p increases, which is reassuring: the VB algorithm does what its is meant to do in this setting. Notice that the KL divergence does not seem converge to zero, even though it might would be coherent (all the points are correctly classified). The difference come from two factors: the covariances of p have not converged yet, and the q mode are Laplacian instead of Gaussian. The size of the p covariances mainly contributes to changing the confidence of p for its classification probabilities $p_\theta(z \mid x)$. Apart from these points, the two posterior distributions are relatively similar after the algorithm has converged: this is coherent since the inference problem is simple and the variational family approximates the true posterior distribution well.

4.2 Hand-written digits

4.2.1 Model and variational famiy

We now consider the classical yet more complex case of the MNIST dataset. The MNIST dataset [35] is a set of 60000 training images and 10000 test images representing hand-written digits. This example was originally used the paper introducing VAEs [31]. We will use the same model with a slightly more modern neural network architecture. The observations x we are given are 28×28 gray scale images. The objective is to find a latent observation of an image in a much lower dimension space, for example $d = 16$. The model, given the latent variable z, of an image x, is to consider each pixel an independent Bernoulli variable whose parameter is a function of z. This set of probabilities is produced by a neural network $f_\theta(z)$. The variational distribution of z given x is a Gaussian distribution, whose mean and covariance are parameterized by neural networks μ_φ, Σ_φ. The prior on z is a normal Gaussian. The

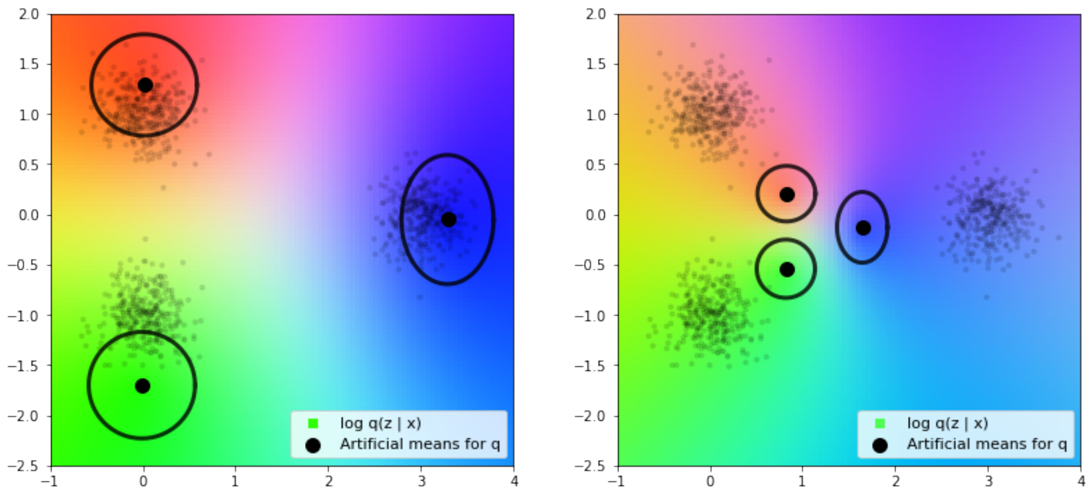

Figure 4: Representation of the log-probability $\log q_\varphi(z \mid x)$ with red, green and blue channels corresponding to the log-likelihood of each class. On the left, q_φ is parameterized by the means optained through the optimization process. On the right, we compute the same log-probability with artificially closer centers. We can see that, when the centers get too close, the outer contrast between the classes diminishes, which explains the optimized positioning of the q means.

VAE model is summed up by the following equations:

$$\begin{cases} p_\theta(z) = \mathcal{N}(0, I) \\ p_\theta(x_{ij} \mid z) = \mathrm{Ber}(f_\theta(z)_{ij}) \\ q_\varphi(z \mid x) = \mathcal{N}(\mu_\varphi(x), \Sigma_\varphi(x)) \end{cases}$$

In this model, the reparameterization trick applies without any problem, because the variational distribution is Gaussian: we use $z = g_\varphi(\varepsilon, x)$ with $\varepsilon \sim \mathcal{N}(0, I)$ and $g_\varphi(\varepsilon, x) = \mu_\varphi(x) + \sqrt{\Sigma_\varphi(x)}\varepsilon$. As usually for this example, we take Σ_φ diagonal and parameterize is with $\Sigma_\varphi[i, i] = \exp(\lambda_i)$ so as to enforce the positivity constraint. Again, the model is implemented with Pytorch, whose automatic backpropagation makes training possible. At training time, the images x (whose coordinate are floats between 0 and 1) are randomly binarized according to the value of each pixel. The networks we use have one hidden layers and 400 hidden units with ReLU and sigmoid activations. The model is trained using Adam optimizer [30] with learning rate 10^{-3}, using a batch size 128 over 10 epochs total.

4.2.2 Numerical results

As we can see in figure 6, the VAE correctly reproduces sample digits from each class, while also preserving the intra-class shape variability (i.e. the diversity of shapes corresponding to each digit). Since the reconstructed object is much more complex than the cluster means of the GMM, it is however hard to assess the quality of the convergence at the first glance. As for GMM, we tracked the evolution of $\log p_\theta(x)$ and $\mathrm{KL}(q_\varphi(z \mid x) \mid\mid p_\theta(z \mid x))$ across the epochs. Contrarily to the GMM, we do not have access to explicit expressions of these quantities. Therefore we resorted to estimate them using Importance Sampling (IS) at each step. Mainly, the problem lies in the expressions $\log p_\theta(x)$ and

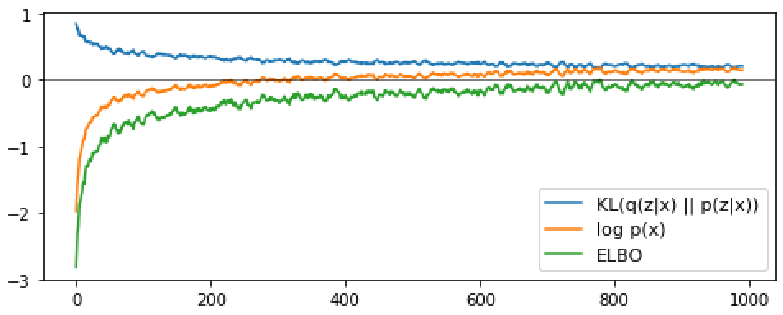

Figure 5: Time evolution of the ELBO, $\log p_\theta(x)$ and $\mathrm{KL}(q_\varphi(z \mid x) \mid\mid p_\theta(z \mid x))$ during the optimization of Variational Bayes for GMMs.

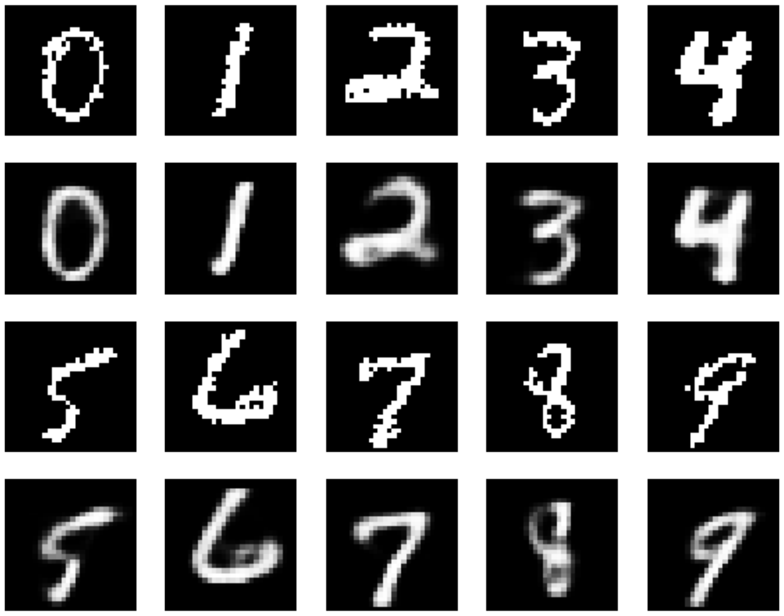

Figure 6: On the first and third row, a binarized sample x from each digit class. On the second and fourth row, the images of corresponding reconstructed probabilities $f_\theta(\mu_\varphi(x))$.

$\log p_\theta(z \mid x)$. Using Importance Sampling on the first quantity gives:

$$p_\theta(x) = \int p_\theta(x, z)\,\mathrm{d}z = \int \frac{p_\theta(x, z)}{q_\varphi(z \mid x)} q_\varphi(z \mid x)\,\mathrm{d}z$$
$$= \mathbb{E}_{q_\varphi(z|x)}\left[\frac{p_\theta(x, z)}{q_\varphi(z \mid x)}\right] \simeq \sum_{k=1}^{K} \frac{p_\theta(x, z_k)}{q_\varphi(z_k \mid x)}$$

with $z_1, ..., z_K \overset{\text{i.i.d}}{\sim} q_\varphi(z \mid x)$. IS is interesting in this situation because the distribution $q_\varphi(z \mid x)$ is precisely designed to be as close as possible from $p_\theta(z \mid x)$, which is an important for the performance of the IS. Similarly, we estimate the Kullback-Leibler divergence using Monte-Carlo estimation and by using $\log p_\theta(z \mid x) = \log p_\theta(x \mid z) + \log p_\theta(z) - \log p_\theta(x)$ and our previous approximation of $\log p_\theta(x)$. Empirically, the variance of these estimations appears to be small with respect to their order of magnitude. The evolution we observed in figure 7.

Like for the GMM, we observe that the log-likelihood of p increases and the KL divergence diminishes with time. However, unlike GMM, the divergence does not converge to zero and stagnates long before the likelihood stops rising. This is understandably a consequence of the **amortization gap** that we mentioned earlier. In our case, this gap consists is the error between the best Gaussian distribution approximating $p_\theta(z \mid x)$ and the distribution $\mathcal{N}(\mu_\varphi(x), \Sigma_\varphi(x))$ produced by the encoder q. When the structure of the data distribution gets more and more complex, the amortization will logically perform worse. In our GMM example, there was no amortization because each cluster had its own parameters. On the contrary, if we did not use amortization for MNIST pictures, we would have to learn an individual Gaussian distribution for each one of the 70000 (training + testing) pictures in the dataset.

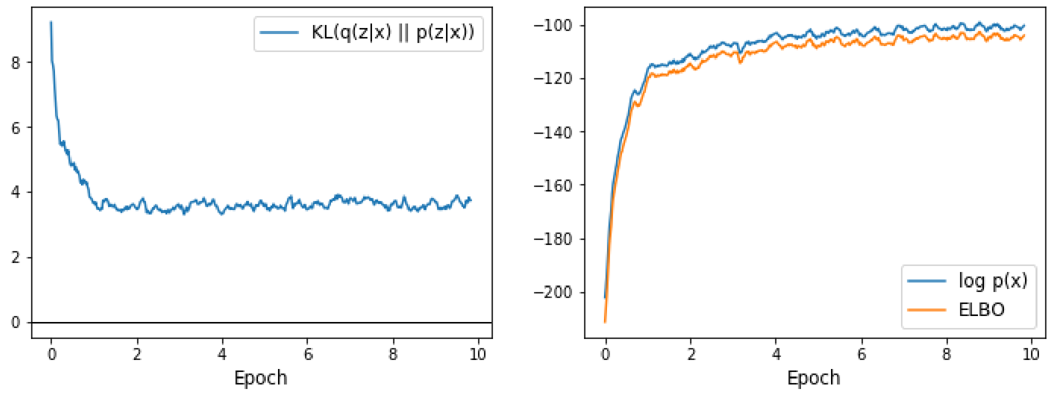

Figure 7: Evolution across the training epochs of $\log p_\theta(x)$, $\mathrm{KL}(q_\varphi(z \mid x) \mid\mid p_\theta(z \mid x))$ and the ELBO for the VAE algorithm on the MNIST dataset.

5 Extending the VAE algorithm

Our experiments with the MNIST dataset gave us a reliable code base for various experiments and extensions of the VAE algorithm. We first tried to make a better use of old samples, then to include an MCMC dynamic in the encoding distribution.

5.1 Using previous samples

More accurate Monte-Carlo Our initial idea was to re-use the samples of the q_φ distribution generated in the previous steps of the algorithm when computing the estimate of the ELBO. This would have allowed to reduce the variance of the estimate, in a similar fashion to the SAEM algorithm, which aggregates the objective functions of all previous steps. However this approach is incompatible with the reparameterization trick at its core: how should we use the sample from last state ? It does not make any sense to keep only the value of the old ε because the center of the Gaussian distribution changes at each step. and ε is only an offset relative to this center. We did not consider trying other differentiation techniques like the REINFORCE rule, for we wanted to keep the algorithm as light as possible. The REINFORCE rule needs a manual backpropagation (it is not implemented in any deep learning framework), and thus requires to be adapted for each new model.

Insights from Sequential Monte Carlo Another idea we had was to give up on the parametric representation of the variational distribution: instead, we would maintain for each observation x a collection of K samples z_1, ..., z_k. The samples would be generated by a parametric distribution q_φ whose parameters would have been learned along the process. In order to make the samples represent the distribution, we considered mechanisms inspired from Sequential Monte-Carlo [18] (SMC, aka Particle Filtering or PF). SMC resolves the filtering problem (determine the distribution $(z_k \mid x_1, ..., x_k)$ where (x_k) is a hidden Markov chain and the distribution of $(z_k \mid x_k)$ is known). SMC proceeds by keeping a list of particles $(z^1, ..., z^P)$ at step k, and update them and their weight at step $k+1$. The weight of each particle is computed so as to guarantee that, even though the sample is discrete, is represents the desired distribution as precisely as possible (in a way similar to Importance Sampling). Our idea was, similarly, to attribute a weight to each sample so as to keep useful sample along iterations. This method would have the advantage of keeping very old particles which may be far from the cluster center $\mu_\varphi(x)$ but still be relevant to approximate $p_\theta(z \mid x)$. Regularly updating the probabilities would have allowed to keep the particle set balanced. Furthermore, recent adaptations [34, 40, 2] of the SMC framework to the VAE setting opened encouraging avenues of interactions between both fields.

The SMC variant idea also turned out to be impracticable. First, the method raised issues with the reparameterization trick as in last paragraph. Furthermore, keeping a representative number of particles per observation adds a non-negligible cost in terms of memory, as soon as the dataset exceeds a few dozen thousands entries (e.g. the MNIST dataset). Finally, it turned out to be very delicate to equilibrate the particle weights: in various tests, the coefficients would often collapse and concentrate into a restricted number of particles.

Trying to improve only the sampling is a bad idea More generally, after various attempts and many variants experiments, we realized that improving the sampling procedure without changing the distribution q_φ was probably not a good idea, for several reasons:

- First, as we mentioned several times already, making a better use of previous samples makes the reparameterization trick impracticable.
- Next, a more heuristic argument. Our previous explorations were aiming at making samples closer to the desired distribution $p_\theta(z \mid x)$. However improving the sampling quality without changing q_φ is equivalent to maximizing an expectation which is different from the ELBO. For instance if we correct each sample of q_φ with an Importance Sampling weight targeting $p_\theta(z \mid x)$, we will in fact be maximizing:

$$\mathbb{E}_{p_\theta(z|x)}\left[\log p_\theta(x,z) - \log q_\varphi(z \mid x)\right] = \log p_\theta(x) + \mathrm{KL}(p_\theta(z \mid x) \mid\mid q_\varphi(z \mid x))$$

  which is not even a lower bound on $\log p_\theta(x)$ anymore and favors bad distributions q_φ.
- The problem above could be solved if we considered that our variational distribution is not q_φ but the samples we derived from it, as well as their related weights. However, such a distribution would be a sum of Dirac masses and it would not make sense to compute its KL divergence with $p_\theta(z \mid x)$, which has a continuous density: discrete distributions are not amenable to the VAE framework.
- Finally, consider the following question. What would such a variant reduce: the approximation gap or the amortization gap? Using Importance Sampling weights or adding more samples will not change the location of the distribution represented by q_φ, but it will change the empirical histogram built on samples. Therefore it will have more impact on the shape of the distribution (approximation) than on its location (amortization). Improving the sampling is thus expected to reduce the approximation error, but not necessarily the amortization error. Besides, a recent survey [12] highlights that the approximation error actually represents approximately 4% of the total error, the remaining 96% being the amortization error (these results are obtained for the standard VAE experiment on the MNIST dataset). This result is as a matter of fact not so surprising: for example having a distribution placed in the right location but with an over-skewed tail will always be better than having a distribution with the right profile but located out of position - which is what happens when amortization is used and generalizes poorly.

Remark 8. While we show in this section that designing new methods to improve the sampling procedure in VAEs seems to be a bad idea, the estimation quality of the expectation can always be improved. In particular, a recent paper [5] shows that using Quasi-Monte-Carlo sampling instead of classical Monte-Carlo in Variational Inference leads to substantial improvements in terms of convergence speed and quality.

5.2 Adding MCMC to the variational encoder

Following the conclusions of the above subsection, we focused on a different, yet since recently thriving, field of the VI literature. Instead of changing the overall structure of the algorithm, the researchers propose to tweak the encoding distribution $q_\varphi(z \mid x)$ by adding to it a fixed number of MCMC steps targeting $p_\theta(z \mid x)$. In other words, the sample z at a given step is computed as follows: 1) simulate $z_0 \sim q_\varphi(z \mid x)$; 2) simulate $(z_1 \mid z_0, x), ..., (z_K \mid z_{K-1}, x)$ from some Markov chain targeting $p_\theta(z \mid x)$; 3) use z_K in the SGD procedure. Contrarily to the subsection above, adding MCMC steps would directly act on the amortization error by displacing the latent variable in the right direction. However, it also adds new technical challenges like differentiating through the MCMC steps.

The technique described above was first introduced in 2014 by Salimans *et al.* in [48]. As we just said, the main idea is to generate a fixed size Markov chain from the original variational distribution sample z_0. This procedure introduces a collection of auxiliary variables $y = (z_0, z_1, ..., z_{K-1})$ whose likelihood also needs to be taken into account. Their distribution is determined by $q_\varphi(z_{k+1} \mid z_k)$. Integrating these new variables in the variational lower bound yields the new ELBO:

$$\mathcal{L}_{aux} = \mathbb{E}_{q_\varphi(y, z_K \mid x)} \left[\log[p_\theta(x, z_K) r(y \mid z_K, x)] - \log q_\varphi(y, z_K \mid x) \right]$$

Ideally, we would like to use $r(y \mid z_K, x) = q_\varphi(y \mid z_K, x)$. When using this distribution for r, we find that the new bound $\mathcal{L}_{aux}$ is equal to the original ELBO $\mathcal{L}$. However, this expression requires an explicit formula for $q_\varphi(z_k \mid z_{k+1})$, which may not be available. Therefore we may have no other choice but to introduce another parameterization r_ϑ to approximate the reverse kernel of the Markov chain. The objective now thus consists in optimizing at the same time:

- the original model $\log p_\theta(x \mid z)$,
- the decoding first variable $\log q_\varphi(z_0 \mid x)$,
- possibly: the Markov transition kernel $\log q_\varphi(z_{k+1} \mid z_k)$,
- possibly: the approximate reverse transition kernel $r_\vartheta(z_k \mid z_{k+1})$.

The MCMC-VI algorithm now just consists in performing SGD on this new objective. Understandably, this new parameterization can lead to substantial computational costs. The authors of [48] solve this problem by choosing to use Hamiltonian Monte Carlo as a MCMC targeting $p_\theta(z \mid x)$. We briefly introduce this technique to highlight its advantages in our setting.

Hamiltonian Monte Carlo Initially introduced in 1987 by physicians [19], Hamiltonian Monte Carlo (HMC) is a MCMC method which, like Metropolis-Adjusted Langevin Algorithm, relies on a time-continuous dynamics. Hamiltonian dynamics is a subfield of physics and optimal control where the quantity of interest z is considered a particle with position equal to that quantity. The particle also has a momentum v, which is a new auxiliary variable with same dimension as z. The particle's trajectory is guided by its momentum $v = \dot{z}$ and a potential $U(z)$ which depends only on the position. The system is defined using a *Hamiltonian* $\mathcal{H}$:

$$\mathcal{H}(z, v) = U(z) + \frac{1}{2} \|v\|^2$$

The *Hamiltonian equations* determine the evolution of the system over time:

$$\begin{cases} \dot{z} = \nabla_v \mathcal{H} \\ \dot{v} = -\nabla_z \mathcal{H} \end{cases}$$

The Hamiltonian system defines a revertible flow $F_t(z)$. It can be proved that this flow is *volume-preserving*, and that $\det J_{F_t}(z) = 1$. Hamiltonian dynamics behave in an intuitive way: the Hamiltonian represents the energy, which is preserved across the trajectory. The curve $z(t)$ looks like a ball falling onto the landscape of U. Numerically, the Hamiltonian system is most of time simulated using the *Leapfrog algorithm*, which guarantees that the Hamiltonian remains constant and stabilizes the trajectory. The equations of the Leapfrog integrator from t to $t + \epsilon$ are given by:

$$\begin{cases} v(t+\epsilon/2) & = v(t) - (\epsilon/2)\nabla_z \mathcal{H}(z(t)) \\ z(t+\epsilon) & = z(t) + \epsilon \nabla_v \mathcal{H}(v(t+\epsilon/2)) \\ v(t+\epsilon) & = v(t+\epsilon/2) - (\epsilon/2)\nabla_z \mathcal{H}(z(t+\epsilon)) \end{cases} \tag{LF}$$

The initial idea of HMC [42] is to use $U(z) = -\log \pi(z)$, where π is the distribution we want to sample from: the variable z should indeed fall into the areas where the likelihood is high. However, such a technique would greatly depend on the initial position and momentum: if the point starts far away from the regions with high probability, it will acquire a very large momentum by the time it reaches the likely regions, and the momentum will thus propel it outward again, so that we can not hope to get an ergodic convergence. The main problem is thus that the Hamiltonian remains constant across the trajectory. This problem is solved by stopping the trajectory every τ seconds to choose a new random momentum from a Gaussian distribution: by doing this we ensure that, when the point falls close to a high probability region, its old (possibly very high) momentum will be forgotten and replaced by a lower value, so that the particle will be much more likely to stay in the high probability region and explore it. After each piece of trajectory, the new position is accepted or rejected with a Metropolis step which corrects the error in the numerical integration. The acceptance probability is: $\exp(-\mathcal{H}(z_{k+1}, -v_{k+1}) + \mathcal{H}(z_k, v_k)$. If the numerical integrator preserved the Hamiltonian and was revertible, the acceptance would always be 1. The HMC procedure is summarized in algorithm 6. It can be proved that the variable z_k converges in distribution toward the target π (the Markov transition kernel defined by the algorithm satisfies the detailed balance equation for the distribution π).

```
Data: target distribution $\pi(z)$, momentum variance $\sigma^2$, leapfrog size $\eta$ and initial position $z_0$
while not converged do
    Sample a momentum $v_k \sim \mathcal{N}(0, \sigma^2 I)$ ;
    Compute $(z_{k+1}, v_{k+1})$ with $T$ leapfrog steps from $(z_k, v_k)$ ;
    Accept / reject $(z_{k+1}, v_{k+1})$ with probability
      $\alpha = \min(1, \exp(-\mathcal{H}(z_{k+1}, -v_{k+1}) + \mathcal{H}(z_k, v_k)))$ ;
end
Result: Sample $z_1, ..., z_K$
```

Algorithm 6: The Hamiltonian Monte-Carlo sampling algorithm

Hamiltonian VI As described above, HMC seems to be an interesting choice to append to the VI algorithm. It requires indeed very few parameters (only the variance of the momentum σ^2 needs to be tuned). Moreover, when performing only one series of leapfrog steps (i.e. without Metropolis acceptance step), the Hamiltonian flow $z \mapsto F_T(z)$ is fully deterministic and differentiable, and the reparameterization trick thus work. Furthermore, the volume-preserving property of the flow allows for interesting simplifications in the computations: suppose we have a random variable z with distribution q and another variable $z' = F_T(z)$. Then the distribution of z' is given by:

$$\log q(z') = \log q(z) - \log |\det J_{F_T}(z)| = \log q(z)$$

so that the new distribution requires no more computational effort. Notice that, in the VI setting, the posterior distribution $p_\theta(z \mid x)$ is not known. Therefore the potential used in the Hamiltonian computation is $U(z) = \log p_\theta(x, z)$. Since $\log p_\theta(x)$ is constant along the trajectory, using this potential is rigorously equivalent to using $\log p_\theta(z \mid x)$.

The Hamiltonian Variational Inference algorithm proposed in [48] adds a new step after the initial variational posterior sampling: the true variational posterior is obtained after applying a fixed number of HMC leapfrogs. Notice that the original paper introducing HVI does not use the Metropolis rejection step. A posterior paper [51] fixes this problem and proposes a differentiable way to account for the rejection step. In their experiments, the authors of both [48] and [51] show that the new variational posterior distribution performs better and yield higher ELBOs on the MNIST dataset as the number of leapfrog steps increases.

In 2018, the Hamiltonian VI was adapted to the VAE framework [9] (i.e. θ is not a random variable anymore but a parameter to be optimized). The approach is very similar to that of [48] (in particular, no Metropolis step is used). The main difference lies in the sampling step: instead of performing a fixed number of regular HMC leapfrogs, a tempering step is added after each step. After each numerical integrator step k, the momentum is artificially decreased by a factor $\alpha_k = \beta_k \alpha_{k-1}$. The idea is to produce an effect similar to that of Annealed Importance Sampling [41]. The authors of [9] propose two tempering schemes: a quadratic scheme and a free scheme. In the free scheme, the parameter β is learned along with θ and φ. In the quadratic scheme, only β_0 is learned, and the next coefficients are determined by the formula:

$$\beta_k = \left(\left(1 - \frac{1}{\beta_0}\right)\frac{k^2}{K^2} + \frac{1}{\beta_0}\right)^{-1}$$

The resulting sampling procedure is included in the standard VAE algorithm, thus giving the Hamiltonian Variational Auto-Encoder (HVAE). As with HVI, numerical experiments, the authors show that increasing the number of leapfrog steps improves the ELBO and thus the performance of the algorithm. HVAE are the focus of the remaining part of my internship, and we will discuss some possible improvements in the next subsection.

For the sake of completeness of the start of the art, we now quickly present a couple of similar approaches which were proposed in 2019. These papers illustrate the pace at which the field of VI and VAEs is developing (some of them were published during my internship).

- First, Ruiz and Titsias [47] proposes to perform HVI with an improved ELBO, which uses a different KL divergence (which takes into account not only $q_\varphi(z_K \mid x)$ but also $q_\varphi(z_0 \mid x)$). They show that this algorithm performs slightly better than the VAE algorithm.

- In [16], Ding and Freedman propose an variant of IWAEs (Importance Weighted Auto-Encoders, described in section 3.1.2) based on Annealed Importance Sampling. Just like Salimans *et al.*, they add HMC steps to the variational posterior approximation. However the MCMC target is not $p_\theta(z \mid x)$ but a combination of $p_\theta(z \mid x)$ and $q_\varphi(z \mid x)$: they use the unnormalized distribution $f_n(z) = q_\varphi(z \mid x)^{1-\beta_n} p_\theta(x, z)^{\beta_n}$, with n the iteration number and $0 = \beta_0 < ... < \beta_N = 1$. This improved version of IWAE, called AIWAE, is shown to perform better than IWAE. In particular, they improve the way the latent space is used by the encoder and the decoder, which was one of the original advantages of IWAEs.

- Vahdat *et al.* [49] propose an inference procedure in the MCMC-VI framework for undirected graphical models. In this situation, one can use Gibbs sampling as a MCMC, which produces samples which are much faster to differentiate than HMC (which requires to backpropagate the gradient through all the leapfrog steps). The authors show that this method outperforms previous algorithms for training VAEs on posteriors with a directed graphical model structure.

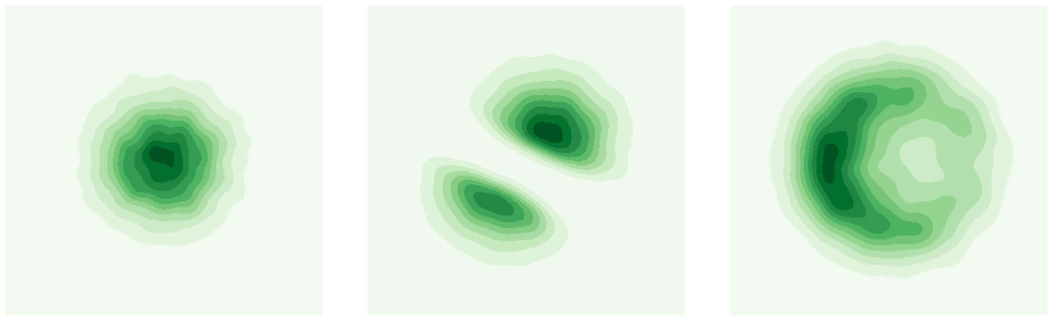

Figure 8: On the left, the histogram of 10000 Gaussian samples. In the middle, the same samples after a planar flow transformation. On the right, the samples after a radial (outward) flow transformation.

Another approach whose popularity has been growing recently is that of Normalizing Flows (NF). Even though it does not exactly consist in a series of MCMC steps, the resemblance creates an interesting parallel between NF and MCMC-VI, and both are often compared in the literature.

Normalizing Flows are essentially smooth, invertible, parameterized transformations. In 2015, Rezende *et al.* [46] proposed a new kind of variational posterior distribution. Until there, much effort was spent on studying simple distribution with sophisticated parameterizations (e.g. Gaussian distributions with mean and covariances determined by neural networks). On the contrary, the authors study a posterior distribution which is a parameterized function of a standard Gaussian variable $z_0 \sim \mathcal{N}(0, I)$. The parameterized function is a composition simple invertible transformations: $z_K = f_x^K \circ ... \circ f_x^1(z_0)$. The probability density of the final posterior z_K can be computed explicitly:

$$\log q_\varphi(z_K \mid x) = \log q(z_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_x^k}{\partial z_{k-1}} \right|$$

which makes backpropagation tractable. The functions f_x^x are simple transformations, like planar flows or radial flows (see figure 8 for simple examples). Their parameters, in turn, are computed from x using neural networks. When applied to VI, Normalizing Flows exhibit overall similar performances as MCMC-VI methods at a given total parameters number. They also show that NFs are able to handle naturally multi-modal posterior distribution (as can be seen with our example for the planar flow). Like MCMC-VI, NFs is a rapidly growing field: a workshop on Invertible Neural Network and NFs was organized this year at ICML (`https://invertibleworkshop.github.io/`), and new extensions of the framework are regularly proposed [37].

5.3 Ongoing research

5.3.1 A Riemannian Manifold HVAE ?

As we investigated throughout the literature, we were interested in the idea of HVAE and HVI. It had become relatively clear that including MCMC inside the posterior distribution and not outside (as we were trying to do in the last subsection) was the right idea. We thus wondered wether we could think of an relevant way to improve the performance of HVAEs. My teaching advisor advised me to try to connect HVI with the existing work on Riemannian HMC.

Riemannian Manifold Hamiltonian Monte Carlo (RHMC) was first introduced by Girolami *et al.* [23, 22]. It is a technique which allows to perform HMC on Riemannian manifold, or just to perform a smarter form of HMC. In the Hamiltonian of HMC $\mathcal{H}(z, v) = -\log p(z) + \frac{1}{2}\|v\|^2$, the choice of the L^2 norm for v could seem arbitrary. A classical variant consists in introducing a symmetric positive definite *matrix mass* M which changes the norm used for v: $v^T v$ is replaced by $v^T M^{-1} v$, and v is now drawn from the distribution $\mathcal{N}(0, M)$. Another term $\log((2\pi)^D |M|)$ also needs to be added to the Hamiltonian, so that is represents the complete log-likelihood $p(z, v)$. We thus get:

$$\mathcal{H}(z, v) = -\log p(z, v) = -\log p(z) - \log p(v) = -\log p(z) + \log((2\pi)^d |M|) + v^T M^{-1} v$$

Changing the matrix M typically acts on the geometry of the variable's space, widening or shrinking the distances along the eigenvectors. It can be interpreted as changing the inner product (the *Riemannian metric*) of the tangent space of the space considered as a manifold. From that perspective, it is also possible to make the metric space-dependent, i.e. to replace M with $M(z)$. Girolami *et al.* propose an optimal design for $M(z)$ adapted to the geometry of the space of parameters as well as a numerical method to simulate the related Hamiltonian dynamics. The authors explain that, if the variable of interest z is the parameter of a model $p(x \mid z)$, then the optimal choice for the metric is the *Fisher information* of this parameter $\mathbb{E}[\nabla_z \log p(x|z) \nabla_z \log p(x|z)^T]$.

Riemannian HVAE The idea we have thus consists in adding a Riemannian metric to the latent space and use it when performing HMC steps after drawing the initial point from the Gaussian distribution $\mathcal{N}(\mu_\varphi(x), \Sigma_\varphi(x))$. Three general options are being considered:

- Using a constant metric M_0. This approach should not be expected to produce significant improvements since neural networks can easily mimic this behavior by making linear transformations of their own.
- Using the Fisher information $I(z)$ related to the model $p_\theta(x \mid z)$. Even though this quantity is not known, we could consider using the observation x which was used by μ_φ to generate z.
- Learning the metric $M(z)$ using a neural network. This option would however significantly increase the number of parameters. We hope to be able to compare the metric which is learned with the estimate of the Fisher information.

Since the HVAE algorithm already has a clean Python implementation using TensorFlow [8], it is possible to design an upgrade implementing those new features without restarting the code from scratch.

5.3.2 Experiments with RHMC

Before implementing the RHVAE, we first wanted to get an intuitive grasp on the way RHMC works in practice. Moreover, the RHMC numerical scheme is a lot more complicated than HMC's simple leapfrog method, so that a first implementation was also useful as a reference for implementing the RHVAE.

Numerical implementation We tried successively two numerical schemes for the RHMC, respectively proposed in [23] and [22]. Both methods aim at correcting the leapfrog method when adding a Riemannian metric: the classical numerical scheme does not preserve the Hamiltonian anymore. The first method, proposed in the original paper introducing RHMC, designs an entirely new *symplectic integrator* based on the new differential equations. Symplectic integrators take stock on the fact that, for an ODE $\dot{x} = (A + B)x$ (where A, B are possibly non-linear yet simple operators), we have $x_t = \exp(t(A + B))x_0$. Since the exponential can not be factorized easily, an approximation is used for small time steps: $\exp(\epsilon(A + B)) \simeq \exp(\epsilon a_1 A) \exp(\epsilon b_1 B) ... \exp(\epsilon a_S A) \exp(\epsilon b_S B) + O(\epsilon^S)$, where $a_1 + ... + a_S = 1$ and $b_1 + ... + b_S = 1$. This method is applied to the Hamiltonian system in [23] with a bigger number of linear operators. The implementation of this method worked successfully, but when it came to include it into the RHVAE code the method would not scale up, thus taking a considerable amount of time per batch. We tried to see wether explicit computations could lead to reduce the number of approximation steps, but we did not get any easily usable formula.

Therefore we turned ourselves toward another implementation proposed in the more recent article [22], which introduces the same RHMC method but proposes a different numerical scheme:

$$\begin{cases} v(t+\epsilon/2) &= v(t) - (\epsilon/2)\nabla_z \mathcal{H}(z(t), v(t+\epsilon/2)) \\ z(t+\epsilon) &= z(t) + (\epsilon/2)\left(\nabla_v \mathcal{H}(z(t), v(t+\epsilon/2)) + \nabla_v \mathcal{H}(z(t+\epsilon), v(t+\epsilon/2))\right) \\ v(t+\epsilon) &= v(t+\epsilon/2) - (\epsilon/2)\nabla_z \mathcal{H}(z(t+\epsilon), v(t+\epsilon/2)) \end{cases} \tag{GLF}$$

This scheme, called the Generalized Leapfrog, is much more similar to the original Leapfrog integrator (LF). The main difference is that this new scheme is implicit: for instance the expression giving $v(t+\epsilon/2)$ contains itself $v(t+\epsilon/2)$. We turned these equation into a numerical algorithm by solving the implicit equations using a fixed point procedure, i.e. computing $v_{k+1} = f(v_k)$ until convergence $v_k = f(v_k)$. In practice, only a few (3-4) iterations suffice to get a very accurate result and a Hamiltonian-preserving scheme.

Gaussian density We first compared HMC and RHMC on a simple Gaussian density, with a covariance particularly stretched along one dimension, typically $\Sigma = \mathrm{Diag}(1, 10)$. We used various metrics to get a concrete understanding of the way RHMC works. It seems that, the smaller the metric is in a given direction (i.e. the smaller the eigenvalue along a given eigenvector), the more amplitude the points has in this direction. For instance, using a small metric ϵI_2 would produce extremely large trajectories which would never converge toward the right distribution. With this in mind, the natural choice wad to use a constant metric $G(x) = \Sigma^{-1}$, i.e. a metric which would favor the motion along the y axis. We ran the Leapfrog and Generalized Leapfrog algorithms for 25*25 steps, with one Metropolis acceptance every 25 step. The result is shown in figure 9. Both trajectories explore the entire Gaussian more, but the behaviors clearly differ: the RHMC algorithm acts at the right scale in each direction, whereas the classical HMC oscillates as a point stuck in a narrow valley. An immediate consequence is that the autocorrelation of the coordinates decreases much faster for the RHMC; mixing rate should also be better, since the space is explored more uniformly after a reduced amount of time steps.

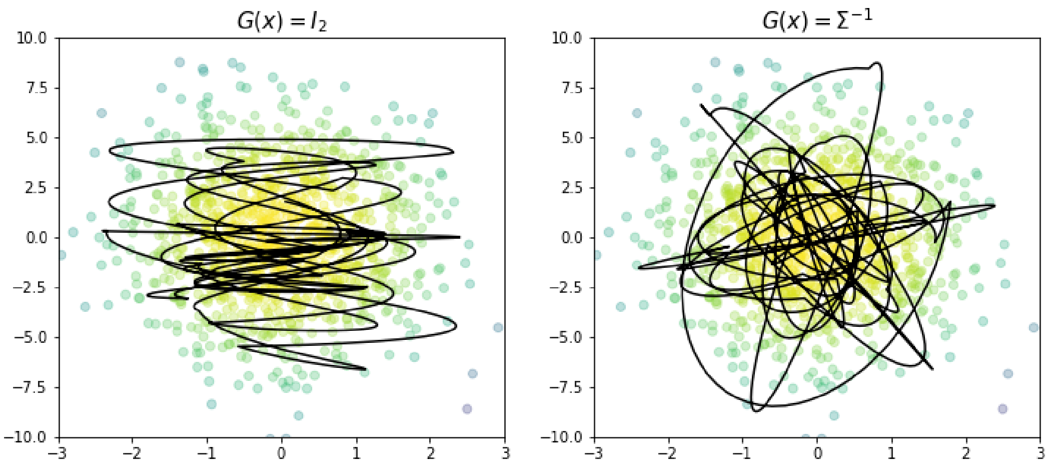

Figure 9: Classical HMC (left) vs. Riemannian HMC (right) with constant metric.

Using Fisher information Next we tried to see the impact of using the Fisher information on the performance of the MCMC. We generated a sample $x_1, ..., x_{1000}$ from a distribution $\mathcal{N}(0, \Sigma)$ with $\Sigma = \mathrm{Diag}(1^2, 7^2)$ and performed Bayesian inference on the standard deviations σ_i along each axis. Notice that we voluntarily choose a scale dissymmetry between the two components so as to evidence the misbehavior of the classical HMC. We used an Inverse Gamma prior with parameters $\alpha = 2, \beta = 1$

for σ_1 and σ_2. The complete log-probability given to the RHMC sampler was thus given by $\log p(x \mid \sigma) + \log p(\sigma)$. The Fisher information of the model $p(x \mid \sigma)$ can be computed explicitly: we have $I(\sigma) = \mathrm{Diag}(2/\sigma^2)$. We thus used $G(\sigma) = I(\sigma)$ for the RHMC in this experiment. We ran the Leapfrog and Generalized Leapfrog algorithms for 25*25 steps, with one Metropolis acceptance every 25 step. The result is displayed in figure 10. Again, we can see that the Riemannian metric accelerates a lot the MCMC's convergence. Furthermore, in the first case the distribution is not explored enough in the lower right area, which is not the case with the RHMC. This is a very understandable consequence of the Fisher metric shape $I(\sigma)_{ii} = 2/\sigma_i^2$: the diagonal coefficients decrease along each corresponding axis. As a consequence (smaller metric $\Longrightarrow$ bigger motion), the motion sensitivity is amplified when the point gets further from the origin along an axis or the other. This behavior corresponds to the profile of the likelihood function, hence it explains partly the better performance of RHMC on this specific example.

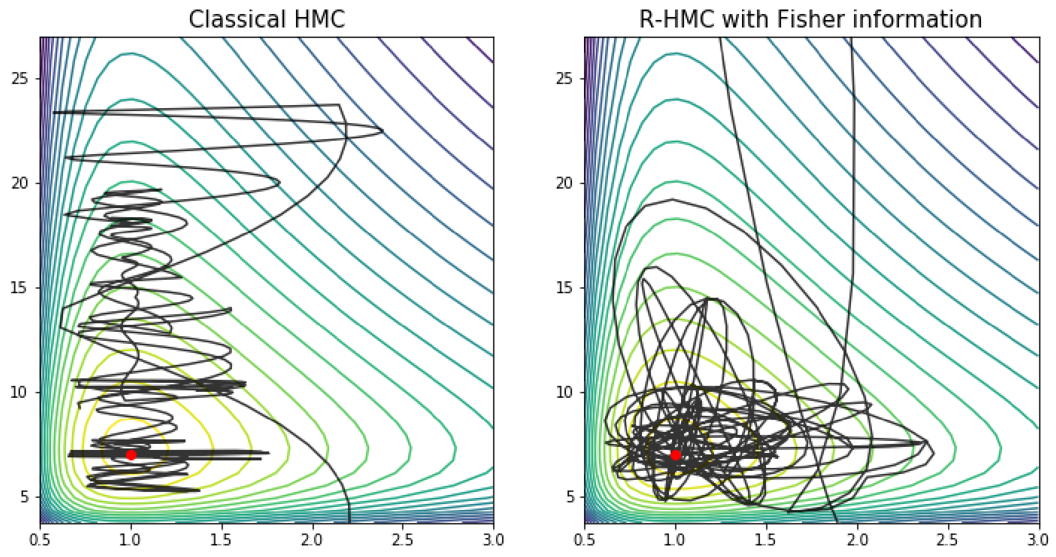

Figure 10: Classical HMC (left) vs. Riemannian HMC (right) using Fisher information for the posterior distribution of σ given x. The contour describes the complete log-likelihood $\log p(x, \sigma)$ and the red dot is the ground truth.

5.3.3 Implementation issues with RHVAE: ongoing work

For the time being, I have not finished including the Riemannian Manifold HMC into the Hamiltonian VAE algorithm. Several reasons can be accounted for:

- The original HVAE implementation was proposed in TensorFlow. In order to insure the results reproducibility and make a comparison with the original method relevant, we had to convert our Pytorch-based RHMC implementation, which takes time as the differentiation mechanisms work differently in both frameworks.

- I first spent a consequent time on trying to implement the symplectic integrator of [23] into TensorFlow. However one of the differential sub-operators scales poorly with the latent dimension, and the algorithm can not be fully vectorized, which lead to very heavy computations. I later moved on to the Generalized Leapfrog implementation proposed in [22], which should not face such issues, but I could not finish this implementation during my internship.

Nevertheless, as I am still interested to see wether adding a Riemannian metric will improve the performance of the HVAE, I plan to finish this work with my internship tutor Stéphanie Allassonnière, which will also become my thesis advisor starting from October. To be continued !

6 Conclusion

My internship allowed me to discover the wide and ever-expanding field of Variational Inference and the related algorithms. My work led me from the MVA courses up to the state of the art and its most recent developments. Variational Inference in its current state is still evolving fast, and the related computational tools, although very powerful, are not mature yet. The research community's attention is being driven increasingly toward the question of the inference gap which is crucial when using neural networks in the inference model. Ideas like MCMC-VI are starting to emerge as possible answers to this problem, but still need to prove more efficient before they can be called a true solution. We hope that our work on Riemannian manifold VAEs will help shed a piece of light on this ambitious challenge.

I was also very interested by the theoretical part of my internship. Most of the stochastic approximation tools necessary to analyze the convergence of the current algorithms exist for more than thirty years, yet little attention seems to have been devoted to analyzing their relevancy to the current ill-understood neural networks-related problems. Our work however suggests that interesting insights could be drawn from an in-depth technical analysis of the convergence hypotheses and their satisfiability.

Finally, my internship was a great opportunity to immerse myself into the world of academical research and applied mathematics. It confirmed my desire to continue my studies into a PhD with my internship advisor Stéphanie Allassonnière as well as Stanley Durrleman. The topic I will be working on is not directly related to my internship's theme, even though both are connected to the more general question of statistical inference. I will be working to try to adapt the LDDMM model's shape deformation theory to the setting of large graphs. The idea is to capture a similar kind of structure and thus to measure distances between large graphs. Such a distance could then be employed to compute means of large graphs. The application domains range in the medical field from neuroscience modeling to protein regulation networks.

A Proofs for the stochastic approximation

A.1 Growth bounds for neural networks and the ELBO

Proposition 1. *Let $f_\theta(x)$ a neural network with L layers with no activation on the final layer, with k-Lipschitz activation function σ_{act}. Then we have $|f_\theta(x)| \leq C(1+\|\theta\|)^L(1+\|x\|)$, where C is a constant depending on L, k and $\sigma_{act}(0)$.*

Proof. We proceed by induction: let $f_\theta(x) = \sigma_{act}(Wx+b)$ be a one-layer network. Then we have:

$$\begin{aligned}|f_\theta(x) - f_{\theta'}(x)| &\leq k|(W-W')x + b - b'| \\ &\leq L(1+|x|)(|W'-W| + |b'-b|) \leq k(1+\|x\|)(1+\|\theta-\theta'\|)\end{aligned}$$

The result is true by taking $\theta' = 0$. Consider now a network with $L+1$ layers: this writes $f_\theta(x) = \sigma_{act}(Wg_\theta(x)+b)$, where g_θ is a network with L layers. We also denote $C_0 = |f_0(x)|$ a constant which depends only on the network's structure. We get:

$$\begin{aligned}|f_\theta(x) - f_0(x)| &\leq L|Wg_\theta(x) + b - f_0(x)| \\ &\leq k(|W|.|g_\theta(x)| + |b| + C_0) \\ &\leq k(1+C_0)(1+|W|+|b|)(1+|g_\theta(x)|) \\ &\leq k(1+C_0)\,\|\theta\|\,(1+(1+\|\theta\|)^L(1+\|x\|)) \\ &\leq K(1+\|\theta\|)^{L+1}(1+\|x\|)\end{aligned}$$

Hence the result. □

In the proofs that follow, C will denote an arbitrary constant depending on $L, k, \sigma_{act}(0)$, $N, \dim z$ and the norms $\|x_i\|$, which is large enough for our purpose. It might change from one line to another. Note that it does not impact the final results, which are asymptotic growth bounds.

Proposition 2. *Under the previous assumptions, there exists a constant C depending on $L, k, \sigma_{act}(0)$, $N, \dim z$ and the norms $\|x_i\|$ such that $|\mathcal{L}(\theta,\varphi)| \leq C(1+\|\theta\|)^{2L}(1+\|\varphi\|)^{2L}$.*

Proof. Let us first recall the expression of a multivariate Gaussian density with mean μ and covariance Σ:

$$\mathcal{N}(\mu,\Sigma) = \frac{1}{\sqrt{2\pi}^d\sqrt{|\Sigma|}}\exp\left(-\frac{1}{2}(x-\mu)^t\Sigma^{-1}(x-\mu)\right)$$

Since the encoder is Gaussian, the reparameterization trick writes for z: $z_i = g_\varphi(\varepsilon_i, x_i) = \mu_\varphi(x_i) + \sqrt{\Sigma_\varphi(x_i)}\varepsilon_i$. We can now write the ELBO:

$$\begin{aligned}\mathcal{L}(\theta,\varphi) &= -\mathbb{E}_{q_\varphi(z|y)}\left[\frac{1}{N}\sum_{i=1}^N \log p_\theta(x_i,z_i) - \log q_\varphi(z_i \mid x_i)\right] \\ &= -\mathbb{E}_{q_\varphi(z|y)}\Bigg[-\frac{1}{2N}\sum_{i=1}^N \log\det\Sigma_\theta(z_i) + (x_i-\mu_\theta(z_i))\Sigma_\theta(z_i)^{-1}(x_i-\mu_\theta(z_i)) + \|z_i\|^2 \\ &\qquad + \frac{1}{2N}\sum_{i=1}^N \log\det\Sigma_\varphi(x_i) + (z_i-\mu_\varphi(x_i))\Sigma_\varphi(x_i)^{-1}(z_i-\mu_\varphi(x_i))\Bigg] \\ &= -\mathbb{E}_{P(\varepsilon)}\Bigg[-\frac{1}{2N}\sum_{i=1}^N \log\det\Sigma_\theta(g_\varphi(\varepsilon_i,x_i)) \\ &\qquad + (x_i-\mu_\theta(g_\varphi(\varepsilon_i,x_i)))\Sigma_\theta(g_\varphi(\varepsilon_i,x_i))^{-1}(x_i-\mu_\theta(g_\varphi(\varepsilon_i,x_i))) + \|g_\varphi(\varepsilon_i,x_i)\|^2 \\ &\qquad + \frac{1}{2N}\sum_{i=1}^N \log\det\Sigma_\varphi(x_i) + \|\varepsilon_i\|^2\Bigg]\end{aligned}$$

We bound this expression term-wise. The determinant of a matrix is a degree d polynomial in its coefficients, so we have $\epsilon < \det \Sigma \leq C(1+\|\Sigma\|)^d$, and thus $|\log \det \Sigma| \leq |\log \epsilon| + \log C + d \log(1+\|\Sigma\|) \leq C(1+\|\Sigma\|)$ (C is any constant). Furthermore, as Σ is upper bounded, its inverse is lower bounded:

$$\left\|\Sigma^{-1}\right\|_2^2 = \sum_i \frac{1}{\Sigma_{ii}^2} \leq \frac{d}{\epsilon^2}$$

Furthermore, in VAEs, $g_\varphi(\varepsilon_i, x_i) = \mu_\varphi(x_i) + \sqrt{\Sigma_\varphi(x_i)}$. Since we assumed the covariance diagonal, we get:

$$\begin{aligned}
|\mathcal{L}(\theta,\varphi)| \leq \mathbb{E}_{P(\varepsilon)} \Bigg[\frac{1}{2N} \sum_{i=1}^N & C(1+\|x_i\|)(1+\|\varphi\|)^L(1+\|\varepsilon_i\|)(1+\|\theta\|)^L \\
&+ \frac{Cd}{\epsilon^2}(\|x_i\| + \|\mu_\theta(g_\varphi(\varepsilon_i,x_i))\|)^2 + C(1+\|x_i\|)(1+\|\varphi\|)^L(1+\|\varepsilon_i\|) \\
&+ C(1+\|x_i\|)(1+\|\varphi\|)^L + \|\varepsilon_i\|^2 \Bigg] \\
\leq \frac{C}{2N}(1+\|\varphi\|)^{2L}(1+\|\theta\|)^{2L} & \mathbb{E}_{P(\varepsilon)} \left[\sum_{i=1}^N (1+\|x_i\|)^2(1+\|\varepsilon_i\|)^2 \right]
\end{aligned}$$

Since the ε_i are Gaussian, they admit a second order moment, and we have the overall data-dependent bound $|\mathcal{L}(\theta,\varphi)| \leq C(1+\|\theta\|)^{2L}(1+\|\varphi\|)^{2L}$. □

Proposition 3. *Let f_θ a neural network with L layers without activation on the last layer, σ_{act} a smooth and k-Lipschitz activation function. Then there exists a constant C depending on L, k and d such that $|\partial_\theta[f_\theta(x)]| \leq C(1+\|\theta\|)^{2L}(1+\|x\|)$ and $|\partial_x[f_\theta(x)]| \leq C(1+\|\theta\|)^L$.*

Proof. We follow the notations from François Malgouyres's course on backpropagation [21]: c_θ^h is the action of layer h, d_θ^h is the same function but using σ'_{act} instead of σ_{act}; $F_\theta^h = c_\theta^L \circ ... \circ c_\theta^{h+1}$; $f_\theta^{h-1} = c_\theta^{h-1} \circ ... \circ c_\theta^1$. The proposition at slide 36/55 writes:

$$\partial_\theta[f_\theta(x)](\theta) \cdot \theta' = \sum_{h=1}^L \partial_x[F_\theta^h(x)](f_\theta^h(x)) \cdot \mathrm{Diag}(d_\theta^h(f_\theta^{h-1}(x))) \cdot (W'_h f_\theta^{h-1}(x) + b'_h) \tag{1}$$

The term $\partial_x[F_\theta^h(x)]$ is computed using the proposition at slide 37/55:

$$\partial_x[F_\theta^h(x)](f) = \partial_x[F_\theta^{h+1}(x)](c_\theta^{h+1}(f)) \cdot \mathrm{Diag}(d_\theta^{h+1}(f)) \cdot W_{h+1} \tag{2}$$

We recall that σ_{act} is assumed to be k-Lipschitz. The formula above thus gives by induction an easy upper bound:

$$|\partial_x[F_\theta^h(x)](f)| \leq k^{L-h} \|\theta\|^{L-h} \leq C(1+\|\theta\|)^L \tag{3}$$

This equation yields the gradient in x when $h=0$. Notice that we should not be surprised that the Jacobian in x is upper bounded by a constant (see proposition 1). We now plug this upper bound into equation 1 and we get:

$$\begin{aligned}
|\partial_\theta[f_\theta(x)](\theta) \cdot \theta'| &\leq \sum_{h=1}^L C(1+\|\theta\|)^L k(|W'_h|.|f_\theta^{h-1}(x)| + |b'_h|) \\
&\leq C(1+\|\theta\|)^L \sum_{h=1}^L (1+|f_\theta^{h-1}(x)|)(|W'_h| + |b'_h|)
\end{aligned}$$

And proposition 1 gives $|f_\theta^{h-1}(x)| \leq C(1+\|\theta\|)^{h-1}(1+\|x\|)$, hence:

$$\begin{aligned}|\partial_\theta[f_\theta(x)](\theta)| &\leq C(1+\|\theta\|)^L \sum_{h=1}^{L}(1+\|\theta\|)^{h-1}(1+\|x\|) \\ &\leq C(1+\|\theta\|)^{2L}(1+\|x\|)\end{aligned}$$

□

Proposition 4. *Under the previous hypotheses, there exists a constant C depending on $L, k, \sigma_{act}(0)$, $N, \dim z$ and the norms $\|x_i\|$ such that $|\nabla\mathcal{L}(\theta,\varphi)| \leq C(1+\|\theta\|)^{4L}(1+\|\varphi\|)^{4L}$.*

Proof. We proceed in two steps. First, compute the gradient of each term in the ELBO and bound them individually. Then we aggregate these bounds into the result.

Step 1. We compute the derivatives using chain rule and apply propositions 1 and 3 on each component.

- $g_\varphi(\varepsilon_i, x_i)_j = \mu_\varphi(x_i)_j + \sqrt{\Sigma_\varphi(x_i)_j}\varepsilon_{ij}$. Hence the derivative:

$$\partial_\varphi g_\varphi(\varepsilon_i, x_i)_j = \partial_\varphi \mu_\varphi(x_i) + \frac{\varepsilon_{ij}}{2\sqrt{\Sigma_\varphi(x_i)_j}}\partial_\varphi \Sigma_\varphi(x_i)_j$$

  Since we assumed $\Sigma_\varphi \succ \epsilon I$, we get the upper bound:

$$|\partial_\varphi g_\varphi(\varepsilon_i, x_i)| \leq |\partial_\varphi \mu_\varphi(x_i)| + \frac{1}{2\sqrt{\epsilon}}\|\varepsilon_i\|.|\partial_\varphi \Sigma_\varphi(x_i)| \leq C(1+\|\varepsilon_i\|)(1+\|\varphi\|)^{2L}(1+\|x_i\|)$$

- $\log\det\Sigma_\varphi(x_i) = \sum_j \log\sigma_\varphi^j(x_i)$. Hence the derivative:

$$\begin{aligned}|\partial_\varphi \log\det\Sigma_\varphi(x_i)| = \left|\sum_j \frac{1}{\sigma_\varphi^j(x_i)}\nabla_\varphi \sigma_\varphi^j(x_i)\right| &\leq \frac{d}{\epsilon}|\partial_\varphi \Sigma_\varphi(x_i)| \\ &\leq C(1+\|\varphi\|)^{2L}(1+\|x_i\|)\end{aligned}$$

- $\log\det\Sigma_\theta(g_\varphi(\varepsilon_i, x_i)) = \sum_j \log\sigma_\theta^j(g_\varphi(\varepsilon_i, x_i))$, hence:

$$\begin{aligned}|\partial_\theta \log\det\Sigma_\theta(g_\varphi(\varepsilon_i, x_i))| \leq \frac{d}{\epsilon}|\partial_\theta \Sigma_\theta(g_\varphi(\varepsilon_i, x_i))| &\leq C(1+\|\theta\|)^{2L}(1+\|g_\varphi(\varepsilon_i, x_i)\|) \\ &\leq C(1+\|\theta\|)^{2L}(1+\|x_i\|)(1+\|\varphi\|)^L(1+\|\varepsilon_i\|)\end{aligned}$$

- $\nabla_\varphi \log\det\Sigma_\theta(g_\varphi(\varepsilon_i, x_i)) = \nabla_x[\log\det\Sigma_\theta(x)]\partial_\varphi g_\varphi(\varepsilon_i, x_i)$, hence:

$$|\nabla_\varphi \log\det\Sigma_\theta(g_\varphi(\varepsilon_i, x_i))| \leq \frac{d}{\epsilon}|\partial_x \Sigma_\theta(x)|.|\partial_\varphi g_\varphi(\varepsilon_i, x_i)| \leq C(1+\|\theta\|)^L(1+\|\varepsilon_i\|)(1+\|\varphi\|)^{2L}(1+\|x_i\|)$$

- $\nabla_\varphi \|g_\varphi(\varepsilon_i, x_i)\|^2 = 2[\partial_\varphi g_\varphi(\varepsilon_i, x_i)]^t g_\varphi(\varepsilon_i, x_i)$, hence:

$$\begin{aligned}|\nabla_\varphi \|g_\varphi(\varepsilon_i, x_i)\|^2| &\leq C(1+\|\varepsilon_i\|)(1+\|\varphi\|)^{2L}(1+\|x_i\|).(1+\|x_i\|)(1+\|\varphi\|)^L(1+\|\varepsilon_i\|) \\ &\leq C(1+\|\varepsilon_i\|)^2(1+\|\varphi\|)^{3L}(1+\|x_i\|)^2\end{aligned}$$

- $(x_i - \mu_\theta(g_\varphi(\varepsilon_i, x_i)))^t \Sigma_\theta(g_\varphi(\varepsilon_i, x_i))^{-1}(x_i - \mu_\theta(g_\varphi(\varepsilon_i, x_i))) = \sum_j \dfrac{1}{\sigma_\theta^j(g_\varphi(\varepsilon_i, x_i))}(x_i - \mu_\theta(g_\varphi(\varepsilon_i, x_i)))_j^2$.

  Which gives in θ (remark: each coordinate of a network's output can be considered a network by itself):

$$
\begin{aligned}
\nabla_\theta[...] &= \sum_j \frac{1}{\sigma_\theta^j(g_\varphi(\varepsilon_i, x_i))}\nabla_\theta(x_i - \mu_\theta(g_\varphi(\varepsilon_i, x_i)))_j^2 - \frac{\nabla_\theta \sigma_\theta^j(g_\varphi(\varepsilon_i, x_i))}{\sigma_\theta^j(g_\varphi(\varepsilon_i, x_i))^2}(x_i - \mu_\theta(g_\varphi(\varepsilon_i, x_i)))_j^2 \\
|\nabla_\theta[...]| &\leq \sum_j \frac{1}{\epsilon}|\nabla_\theta(x_i - \mu_\theta(g_\varphi(\varepsilon_i, x_i)))_j^2| + \frac{1}{\epsilon^2}|\nabla_\theta \sigma_\theta^j(g_\varphi(\varepsilon_i, x_i))|(x_i - \mu_\theta(g_\varphi(\varepsilon_i, x_i)))_j^2 \\
&\leq C\sum_j |(\mu_\theta(g_\varphi(\varepsilon_i, x_i)) - x_i)_j|.|\nabla_\theta \mu_\theta(g_\varphi(\varepsilon_i, x_i))_j| \\
&\qquad + (1+\|\theta\|)^{2L}(1+\|g_\varphi(\varepsilon_i, x_i)\|)(1+\|x_i\|)^2(1+\|\mu_\theta(g_\varphi(\varepsilon_i, x_i))\|^2) \\
&\leq C(1+\|\mu_\theta(g_\varphi(\varepsilon_i, x_i))\|)(1+\|x_i\|)\sum_j |\nabla_\theta \mu_\theta(g_\varphi(\varepsilon_i, x_i))_j| \\
&\quad + C(1+\|\theta\|)^{2L}(1+\|g_\varphi(\varepsilon_i, x_i)\|)(1+\|x_i\|)^2(1+\|\mu_\theta(g_\varphi(\varepsilon_i, x_i))\|^2) \\
&\leq C(1+\|\theta\|)^L(1+\|\varphi\|)^L(1+\|\varepsilon_i\|)(1+\|x_i\|)^2 \times d(1+\|\theta\|)^{2L}(1+\|\varphi\|)^L(1+\|\varepsilon_i\|)(1+\|x_i\|) \\
&\quad + C(1+\|\theta\|)^{2L}(1+\|\varphi\|)^L(1+\|\varepsilon_i\|)(1+\|x_i\|)^3 \times ((1+\|\theta\|)^L(1+\|\varphi\|)^L(1+\|\varepsilon_i\|)(1+\|x_i\|))^2 \\
&\leq C(1+\|\theta\|)^{4L}(1+\|\varphi\|)^{3L}(1+\|x_i\|)^5(1+\|\varepsilon_i\|)^3
\end{aligned}
$$

  Similarly, we get with φ:

$$
\begin{aligned}
|\nabla_\varphi[...]| &\leq \sum_j \frac{1}{\epsilon}|\nabla_\varphi(x_i - \mu_\theta(g_\varphi(\varepsilon_i, x_i)))_j^2| + \frac{1}{\epsilon^2}|\nabla_\varphi \sigma_\theta^j(g_\varphi(\varepsilon_i, x_i))|(x_i - \mu_\theta(g_\varphi(\varepsilon_i, x_i)))_j^2 \\
&\leq C\sum_j |(\mu_\theta(g_\varphi(\varepsilon_i, x_i)) - x_i)_j|.|\nabla_\varphi \mu_\theta(g_\varphi(\varepsilon_i, x_i))_j| \\
&\qquad + C|\nabla_x[\sigma_\theta^j(x)](g_\varphi(\varepsilon_i, x_i))|.|\partial_\varphi g_\varphi(\varepsilon_i, x_i)|(1+\|x_i\|)^2(1+\|\mu_\theta(g_\varphi(\varepsilon_i, x_i))\|^2) \\
&\leq C(1+\|\theta\|)^L(1+\|\varphi\|)^L(1+\|\varepsilon_i\|)(1+\|x_i\|)^2\sum_j |\nabla_x[\mu_\theta^j(x)](g_\varphi(\varepsilon_i, x_i))|.|\partial_\varphi g_\varphi(\varepsilon_i, x_i)| \\
&\qquad + C(1+\|\theta\|)^L \times (1+\|\varepsilon_i\|)(1+\|\varphi\|)^{2L}(1+\|x_i\|)^3 \times (1+\|\mu_\theta(g_\varphi(\varepsilon_i, x_i))\|^2) \\
&\leq C(1+\|\theta\|)^L(1+\|\varphi\|)^L(1+\|\varepsilon_i\|)(1+\|x_i\|)^2 \times (1+\|\theta\|)^L \times (1+\|\varepsilon_i\|)(1+\|\varphi\|)^{2L}(1+\|x_i\|) \\
&\qquad + C(1+\|\theta\|)^L \times (1+\|\varepsilon_i\|)(1+\|\varphi\|)^{2L}(1+\|x_i\|)^3 \times ((1+\|\theta\|)^L(1+\|\varphi\|)^L(1+\|\varepsilon_i\|)(1+\|x_i\|))^2 \\
&\leq C(1+\|\theta\|)^{3L}(1+\|\varphi\|)^{4L}(1+\|x_i\|)^5(1+\|\varepsilon_i\|)^3
\end{aligned}
$$

Step 2. We aggregate all these bounds into the ELBO. We recall the developed expression of $\mathcal{L}$:

$$
\begin{aligned}
\mathcal{L}(\theta,\varphi) = -\mathbb{E}_{P(\varepsilon)}\Bigg[&-\frac{1}{2N}\sum_{i=1}^N \big(\log\det\Sigma_\theta(g_\varphi(\varepsilon_i, x_i))) + (x_i - \mu_\theta(g_\varphi(\varepsilon_i, x_i)))\Sigma_\theta(g_\varphi(\varepsilon_i, x_i))^{-1}(x_i - \mu_\theta(g_\varphi(\varepsilon_i, x_i))) \\
&+ \|g_\varphi(\varepsilon_i, x_i)\|^2\big) + \frac{1}{2N}\sum_{i=1}^N \log\det\Sigma_\varphi(x_i) + \|\varepsilon_i\|^2\Bigg]
\end{aligned}
$$

We can now bound the complete ELBO gradient:

$$
\begin{aligned}
|\nabla \mathcal{L}(\theta, \varphi)| &= |\nabla_\theta \mathcal{L}(\theta, \varphi)| + |\nabla_\varphi \mathcal{L}(\theta, \varphi)| \\
&\leq \frac{1}{2N}\mathbb{E}_\varepsilon \left[\sum_{i=1}^{N} |\nabla_\theta \log\det \Sigma_\theta(g_\varphi(\varepsilon_i, x_i)))| + |\nabla_\varphi \log\det \Sigma_\theta(g_\varphi(\varepsilon_i, x_i)))| \right. \\
&\qquad \left. + |\nabla_\theta[...]| + |\nabla_\varphi[...]| + \nabla_\varphi \left\| g_\varphi(\varepsilon_i, x_i) \right\|^2 + |\nabla_\varphi \log\det \Sigma_\varphi(x_i)| \right] \\
&\leq \frac{C}{2N}\mathbb{E}_\varepsilon \left[\sum_{i=1}^{N} (1+\|\theta\|)^{2L}(1+\|x_i\|)(1+\|\varphi\|)^{L}(1+\|\varepsilon_i\|) + (1+\|\theta\|)^{L}(1+\|\varepsilon_i\|)(1+\|\varphi\|)^{2L}(1+\|x_i\|) \right. \\
&\qquad + (1+\|\theta\|)^{4L}(1+\|\varphi\|)^{3L}(1+\|x_i\|)^5(1+\|\varepsilon_i\|)^3 + (1+\|\theta\|)^{3L}(1+\|\varphi\|)^{4L}(1+\|x_i\|)^5(1+\|\varepsilon_i\|)^3 \\
&\qquad \left. + (1+\|\varepsilon_i\|)^2(1+\|\varphi\|)^{3L}(1+\|x_i\|)^2 + (1+\|\varphi\|)^{2L}(1+\|x_i\|) \right] \\
&\leq \frac{C}{2N}(1+\|\theta\|)^{4L}(1+\|\varphi\|)^{4L} \sum_{i=1}^{N} (1+\|x_i\|)^5 \mathbb{E}_\varepsilon \left[(1+\|\varepsilon_1\|)^3 \right]
\end{aligned}
$$

We finally get $|\nabla \mathcal{L}(\theta, \varphi)| \leq C(1+\|\theta\|)^{4L}(1+\|\varphi\|)^{4L}$. □

A.2 Miscellaneous lemmas and propositions

Lemma 1. *For all $n \in \mathbb{N}$, $\mathbb{E}[e_n|\mathscr{F}_n] = 0$, and the variance $\mathrm{Var}(e_n|\mathscr{F}_n)$ is finite.*

Proof. Let us recall the definition of e_n: $e_n = \nabla\widehat{\mathcal{L}}(s_n) - h(s_n)$, where $h(s)$ is the gradient of the ELBO and $\widehat{\mathcal{L}}$ its estimate on a mini-batch. Let B_n be the random batch selected at step n. Let $\varepsilon_1, ..., \varepsilon_N$ be independent samples of $P(\varepsilon)$. Then $\widehat{\mathcal{L}}$ write:

$$
\widehat{\mathcal{L}}(s) = \frac{1}{|B|}\sum_{i\in B} f(s, x_i, \varepsilon_i)
$$

with $f(s, x, \varepsilon) = \log p_\theta(x, g_\varphi(\varepsilon, x)) - \log q_\varphi(g_\varphi(\varepsilon, x) \mid x)$. Therefore we have:

$$
\begin{aligned}
\mathbb{E}[\nabla\widehat{\mathcal{L}}(s_n) \mid s_n] &= \mathbb{E}\left[\frac{1}{|B|}\sum_{i\in B}\nabla f(s_n, x_i, \varepsilon_i) \mid s_n\right] \\
&= \frac{1}{|B|}\sum_{i=1}^{N}\mathbb{E}\left[\mathbb{E}[\delta_{i\in B}\nabla f(s_n, x_i, \varepsilon_i) \mid s_n] \mid s_n\right] \\
&= \frac{1}{|B|}\sum_{i=1}^{N}\mathbb{E}\left[\frac{|B|}{N}\nabla f(s_n, x_i, \varepsilon_i) \mid s_n\right] \\
&= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[\nabla f(s_n, x_i, \varepsilon_i) \mid s_n\right]
\end{aligned}
$$

Since f is a smooth function and the expectation is on a Gaussian distribution, the expectation can be interchanged with the gradient. We thus get:

$$
\mathbb{E}[\nabla\widehat{\mathcal{L}}(s_n) \mid s_n] = \frac{1}{N}\sum_{i=1}^{N}\nabla\mathbb{E}\left[f(s_n, x_i, \varepsilon_i) \mid s_n\right] = h(s)
$$

Hence $\mathbb{E}[e_n|s_n] = 0$. We now want to bound the conditional variance of e_n. Let us first deal with the mini-batch. After some tedious computations, we find that the variance of the mean of a mini-batch of variables $X_1, ..., X_N$ can be expressed as

$$\mathrm{Var}\left(\frac{1}{|B|}\sum_{i\in B} X_i\right) = \frac{1}{|B|N}\left(\sum_{i=1}^{N}\mathbb{E}[X_i^2] - \frac{|B|-1}{N-1}\mathbb{E}[X_i]^2\right) - \frac{N-|B|}{|B|(N-1)}\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} X_i\right]^2$$

Although we don't include the full computation, a suspicious reader could use a sanity check. One can easily verify that, for i.i.d. variables, this formula leads to $\mathrm{Var}\left(\frac{1}{|B|}\sum_{i\in B} X_i\right) = \frac{1}{|B|}\mathrm{Var}(X_1)$, which is the correct result. We led other (no less tedious) computations on this variance expression to compute its derivative in $|B|$. We find that the variance decreases with the batch size, which is coherent. Finally, it comes that the variance when the batch size is one is bounded by $\sum_{i=1}^{N}\mathrm{Var}(X_i)$. Therefore, if the variance of each sample is finite, the variance of the mini-batch is finite.

We now need to bound the individual variance terms $\nabla f(s_n, x_i, \varepsilon_i)$. The proofs of propositions 2 and 4 can be easily adapted into bounds for their empirical equivalent: the growth function of f and ∇f can be bounded by some power of $\|s_n\| \,.\, \|x_i\| \,.\, \|\varepsilon_i\|$. Hence, since ε is Gaussian, $\nabla f(s_n, x_i, \varepsilon_i)$ is bounded by some power of $\|s_n\| \,.\, \|x_i\|$, which is finite. □

Lemma 2. *Assume that $\sum_n \gamma_n^2 < +\infty$. Then $\sum_{n=0}^{\infty}\mathbb{E}[X_{n+1}^2 \mid \mathscr{F}_n] < \infty$ with probability one.*

Proof. We recall that $S_n = \sum_{t=1}^{n} X_t$ is a martingale, with $X_t = \gamma_t e_t \mathbf{1}_{V(s_{t-1})\le M}$. We have further:

$$\begin{aligned}\sum_{t=1}^{n}\mathbb{E}[\|X_t\|^2 \mid \mathscr{F}_{t-1}] &= \sum_{t=1}^{n}\gamma_t^2\mathbf{1}_{V(s_{t-1})\le M}\mathbb{E}[\|e_t\|^2 \mid \mathscr{F}_{t-1}] \\ &= \sum_{t=1}^{n}\gamma_t^2\mathbf{1}_{V(s_{t-1})\le M}\mathbb{E}\left[\left\|\nabla\widehat{\mathcal{L}}(s_t) - \nabla\mathcal{L}(s_t)\right\|^2 \mid \mathscr{F}_{t-1}\right]\end{aligned}$$

Since $\widehat{\mathcal{L}}$ and $\mathcal{L}$ have a gradient with asymptotically polynomial gradient, at a given M there exists U such that

$$V(s) \le M \implies \|s\| \le U$$

Now we can use the bound from proposition 4 (which easily extends to $\widehat{\mathcal{L}}$). When $V(s) \le M$ we get:

$$\begin{aligned}\left\|\nabla\widehat{\mathcal{L}}(s_t) - \nabla\mathcal{L}(s_t)\right\|^2 &\le (1+\|\theta_t\|)^{8L}(1+\|\varphi_t\|)^{8L}\left(\sum_{i=1}^{N}(1+\|x_i\|)^5(\mathbb{E}_\varepsilon\left[(1+\|\varepsilon_1\|)^3\right] + (1+\|\varepsilon_1\|)^3)\right)^2 \\ &\le C\,\|s_t\|^{16}\left(\sum_{i=1}^{N}...\right)^2 \le CU^{16}\left(\sum_{i=1}^{N}...\right)^2\end{aligned}$$

The sum is finite and only depends on ε and x. We finally get a bound which is uniform in s. This gives for S_n:

$$\sum_{t=1}^{n}\mathbb{E}[\|X_t\|^2 \mid \mathscr{F}_{t-1}] \le \sum_{t=1}^{n}\gamma_t^2\mathbf{1}_{V(s_{t-1})\le M}CU^{16}\mathbb{E}\left[\left(\sum_{i=1}^{N}...\right)^2\right] < +\infty$$

because the sum $\sum_t \gamma_t^2$ is finite. □

Proposition 5. *Suppose that $q_\varphi(z \mid x)$ is drawn from $g_\varphi(x, \varepsilon)$ with $\varepsilon \sim \mathcal{N}_{<R}(0, I)$ a truncated normal. Suppose moreover that the activation function σ_{act} is (finitely) piecewise polynomial. Then, in every direction, the L^2 penalized VAE objective has no critical point beyond a certain distance (dependent on the direction) to the origin.*

Proof. The penalized objective is $\mathcal{K}(s) = -\mathcal{L}(s) + \lambda \|s\|^2$ with the summary variable $s = (\theta, \varphi)$. A critical point thus verifies $\nabla\mathcal{L}(s) = \lambda s$. All we need to prove is that this equality can not happen beyond a certain point. The intuition is that, since the ELBO is positive and has a behavior somewhat similar to that of a polynomial, it will either decrease or stay constant at infinity. Let us first consider a **fixed** $s = (\theta, \varphi)$ with norm 1. We want to prove that there exists T_s such that $t > T_s \implies \nabla\mathcal{L}(ts) \neq \lambda ts$.

It follows from the course of François Malgouyres [21] that the output of a neural network as a function of the parameters is piecewise polynomial with a finite number of pieces. The pieces are determined by a set of polynomial inequalities linking θ and ε or φ and ε. We now refer the reader to the step 1 in the proof of proposition 4, where we computed separately the gradients of all components in the ELBO. Bearing in mind the the structure of the neural network functions, we see that each term we get is written under what we will call a **piecewise** square-root-rational function: that is to say, a function that is piecewise a quotient of functions which are expressed as sum of powers and square roots of polynomials. We thus define $m_t(s, \varepsilon)$ the current piece where the considered point (ts, ε) currently lies. We can now express the fact that, for each (ε, s), the point (ts, ε) ends in an area where all polynomial have constant sign: it is equivalent to say that there exist $m(s, \varepsilon)$ such that $m_t(s, \varepsilon) \xrightarrow[t\to+\infty]{} m(s, \varepsilon)$.

Square-root-polynomials have a simple asymptotic behavior when $t \to \infty$: each polynomial term has a dominant monomial. The square root divides the power of a dominant monomial by two, and the quotient is equivalent to the quotient of dominating powers. On a given piece m, the resulting final quotient F^m thus is equivalent to $t^{h_m} Q_m(\varepsilon, \theta, \varphi)$ on piece A^m, where Q_m is a continuous function (note that the exponents h_m may depend on s, which is fixed here). If necessary, the M pieces can be further split into smaller pieces, over which the dominating coefficient $Q_m(\varepsilon, \theta, \varphi)$ is not zero. This argument, whose relevance may not appear clearly here, is detailed on an example in remark 9. Over these pieces, Q_m is a continuous function of (θ, φ). The remainder can be written as $t^{h_m - 1/2} R_m(t, \theta, \varphi, \varepsilon)$ with R_m a continuous bounded function. Therefore one can write:

$$\begin{aligned}
\nabla\mathcal{L}(ts) &= \int_{B(0,R)} \sum_{i=1}^{M} \mathbf{1}_{m=m_t(s,\varepsilon)} F^m(ts, \varepsilon) \mathrm{d}P(\varepsilon) \\
&= \sum_{i=1}^{M} \int_{B(0,R)} \mathbf{1}_{m=m(s,\varepsilon)} F^m(ts, \varepsilon) + (\mathbf{1}_{m=m_t(s,\varepsilon)} - \mathbf{1}_{m=m(s,\varepsilon)}) F^m(ts, \varepsilon) \mathrm{d}P(\varepsilon) \\
&= \sum_{i=1}^{M} t^{h_m} \int_{B(0,R)} Q_m(\varepsilon, \theta, \varphi) \mathbf{1}_{m=m(s,\varepsilon)} \mathrm{d}P(\varepsilon) + t^{h_m - 1/2} \int_{B(0,R)} R_m(t, \varepsilon, \theta, \varphi) \mathbf{1}_{m=m(s,\varepsilon)} \mathrm{d}P(\varepsilon) \\
&\quad + t^{h_m} \int_{B(0,R)} Q_m(\varepsilon, \theta, \varphi)(\mathbf{1}_{m=m_t(s,\varepsilon)} - \mathbf{1}_{m=m(s,\varepsilon)}) \mathrm{d}P(\varepsilon) \\
&\quad + t^{h_m - 1/2} \int_{B(0,R)} R_m(t, \varepsilon, \theta, \varphi)(\mathbf{1}_{m=m_t(s,\varepsilon)} - \mathbf{1}_{m=m(s,\varepsilon)}) \mathrm{d}P(\varepsilon)
\end{aligned}$$

The last equation gives term by term:

- $I_m(s) = \int_{B(0,R)} Q_m(\varepsilon, \theta, \varphi) \mathbf{1}_{m=m(s,\varepsilon)} \mathrm{d}P(\varepsilon)$: this integral only depends on s.
- $\int_{B(0,R)} R_m(t, \varepsilon, \theta, \varphi) \mathbf{1}_{m=m(s,\varepsilon)} \mathrm{d}P(\varepsilon)$: this integral is bounded uniformly in t.
- $\int_{B(0,R)} Q_m(\varepsilon, \theta, \varphi)(\mathbf{1}_{m=m_t(s,\varepsilon)} - \mathbf{1}_{m=m(s,\varepsilon)}) \mathrm{d}P(\varepsilon)$: the dominated convergence theorem grants that this integral converges to zero when $t \to \infty$.x
- $\int_{B(0,R)} R_m(t, \varepsilon, \theta, \varphi)(\mathbf{1}_{m=m_t(s,\varepsilon)} - \mathbf{1}_{m=m(s,\varepsilon)}) \mathrm{d}P(\varepsilon)$: similarly, this integral converges to zero.

We can thus regroup the three last terms into a function $t^{h_m} g_m(t,s)$ such that $g_m(t,s) \to 0$ when $t \to +\infty$. This writes:

$$\nabla \mathcal{L}(ts) = \sum_{m=1}^{M} t^{h_m} (I_m(s) + g_m(t,s))$$

And, when $t \to +\infty$, only the terms with the highest h_m and non zero $I_m(s)$ dominate. Notice that if all $I_m(s)$ are null then by definition F^m converges to a constant. In particular, there exists an overall coefficient h, a continuous function $I(s)$ and a continuous function $g(t,s)$ such that $g(t,s) = O(1/\sqrt{t})$ when $t \to \infty$ and:

$$\langle \nabla \mathcal{L}(ts), s \rangle = t^h (\langle I(s), s\rangle + g(t,s)) \sim t^h \langle I(s), s \rangle$$

If $h \leq 0$, the ELBO's gradient does not grow at infinity, and cannot be equal to λts. If $h > 0$, three cases arise:

- $\langle I(s), s \rangle > 0$: in this case, the theorems on equivalents of divergent series and integrals grant that

  $$\mathcal{L}(st) - \mathcal{L} = \int_0^t \langle \nabla \mathcal{L}(st), s \rangle \mathrm{d}t \sim \int_0^t t^h \langle I(s), s \rangle \mathrm{d}t \to +\infty$$

  But we know on the other hand that $\mathcal{L}(s)$ is upper bounded (a consequence of the hypothesis $\Sigma_\theta \succ \epsilon I$), which is contradictory. Hence this first case does not happen.

- $\langle I(s), s \rangle = 0$: since $I(s) \neq 0$, we get at infinity that $\langle \nabla \mathcal{L}(ts), s \rangle / \left\| t^h I(s) \right\| \to 0$. Since $\left\| t^h I(s) \right\| \leq \left\| \nabla \mathcal{L}(ts) \right\|$, this gives $\langle \nabla \mathcal{L}(ts), s \rangle / \left\| \nabla \mathcal{L}(ts) \right\| \to 0$. Therefore the equality is never achieved in the Cauchy-Schwartz inequality, hence the vectors $\nabla \mathcal{L}(ts)$ and s are not positively colinear, and we cannot have $\nabla \mathcal{L}(ts) = \lambda ts$.

- $\langle I(s), s \rangle < 0$: in this situation, we have when $t \to +\infty$ that $\langle \nabla \mathcal{L}(ts), s \rangle < 0$. Thus we cannot have $\nabla \mathcal{L}(ts) = \lambda ts$ (otherwise the inner product would be positive).

We have just proved that, in a fixed direction s ($\|s\| = 1$), there exist a threshold $T(s)$ such that $t \geq T(s) \implies \nabla \mathcal{L}(ts) \neq \lambda ts$. It remains to be seen wether T can be bounded on the unit ball. The main problem is that, despite Q_m and R_m being continuous, we have no information as of wether $I(s)$ and $g(t,s)$ are continuous.

□

Remark 9. In this remark, we use an example to justify the continuity of Q_m and R_m as well as the need to split the domains depending on the dominant degree. A rigorous proof would be tedious for the general case, but the idea is simple. If P denotes any component-wise polynomial function, the functions F^m we consider are (at most) of the form $P(P(s) + \sqrt{P(s)}\varepsilon, s)/P(P(s) + \sqrt{P(s)}\varepsilon, s)$. For example, imagine we have the following function on a given piece:

$$F(t(\theta, \varphi), \varepsilon) = \frac{t^4 \theta_1^3 \theta_2 + t\theta_1 (t\varphi_1 + \varepsilon\sqrt{t\varphi_1 + t^3 \varphi_2^3})^2}{t\theta_1 + t^2\theta_2^2 + t^2\theta_2^2 (t\varphi_1 + \varepsilon\sqrt{t^2\varphi_1\varphi_2 + t\varphi_2})}$$

The dominant term is easy to extract:

$$F(t(\theta, \varphi), \varepsilon) = \frac{t^4}{t^3} \frac{\theta_1^3\theta_2 + \theta_1(1/t^{3/2}\varphi_1 + \varepsilon\sqrt{\varphi_1/t^2 + \varphi_2^3})^2}{\theta_1/t^2 + \theta_2^2/t + \theta_2^2(\varphi_1 + \varepsilon\sqrt{\varphi_1\varphi_2 + \varphi_2/t})}$$

We will thus use $Q(\varepsilon, \theta, \varphi) = \frac{\theta_1^3\theta_2 + \theta_1\varphi_2^3}{\theta_2^2 + (\varphi_1 + \varepsilon\sqrt{\varphi_1\varphi_2})}$. With this notation, we have:

$$F(t(\theta, \varphi), \varepsilon) = tQ(\varepsilon, \theta, \varphi) + R(t, \varepsilon, \theta, \varphi)$$

Where R is the remainder term. The functions Q and R are clearly continuous. And, with rational fractions, the remainder is an order of magnitude lower than the dominant term. Here with the introduction of a square root, the order of magnitude becomes half an order of magnitude: we can not *a priori* say better than $R = O(t^{1-1/2}) = O(\sqrt{t})$.

References

[1] Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A. Saurous, and Kevin Murphy. Fixing a Broken ELBO. In *PMLR*, pages 159–168.

[2] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, pages 269–342.

[3] Alexandre Bône, Maxime Louis, Olivier Colliot, and Stanley Durrleman. Learning Low-Dimensional Representations of Shape Data Sets with Diffeomorphic Autoencoders. In Albert C. S. Chung, James C. Gee, Paul A. Yushkevich, and Siqi Bao, editors, *Information Processing in Medical Imaging*, Lecture Notes in Computer Science, pages 195–207. Springer International Publishing, 2019.

[4] William M Bolstad and James M Curran. *Introduction to Bayesian statistics*. John Wiley & Sons, 2016.

[5] Alexander Buchholz, Florian Wenzel, and Stephan Mandt. Quasi-Monte Carlo Variational Inference. In *International Conference on Machine Learning*, pages 667–676, 2018.

[6] Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance Weighted Autoencoders. *ICLR*, 2016. arXiv: 1509.00519.

[7] O. Cappe. Online sequential Monte Carlo EM algorithm. In *2009 IEEE/SP 15th Workshop on Statistical Signal Processing*, pages 37–40.

[8] Anthony Caterini. Code for the Hamiltonian Variational Auto-Encoder from the proceedings of NeurIPS 2018: anthonycaterini/hvae-nips, June 2019. original-date: 2018-10-22T15:46:01Z.

[9] Anthony L Caterini, Arnaud Doucet, and Dino Sejdinovic. Hamiltonian Variational Auto-Encoder. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8167–8177. Curran Associates, Inc.

[10] Badr-Eddine Cherief Abdellatif. Consistency of ELBO maximization for model selection. In *Symposium on Advances in Approximate Bayesian Inference*, pages 11–31, January 2019.

[11] Badr-Eddine Chérief Abdellatif and Pierre Alquier. Consistency of variational Bayes inference for estimation and model selection in mixtures. *Electronic Journal of Statistics*, 12(2):2995–3035, 2018.

[12] Chris Cremer, Xuechen Li, and David Duvenaud. Inference Suboptimality in Variational Autoencoders. In *International Conference on Machine Learning*, pages 1078–1086, July 2018.

[13] Bernard Delyon. Stochastic approximation with decreasing gain : Convergence and asymptotic theory. July 2000.

[14] Bernard Delyon, Marc Lavielle, and Eric Moulines. Convergence of a stochastic approximation version of the EM algorithm. *The Annals of Statistics*, 27(1):94–128.

[15] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

[16] Xinqiang Ding and David J. Freedman. Improving Importance Weighted Auto-Encoders with Annealed Importance Sampling. *arXiv:1906.04904 [cs, stat]*, June 2019. arXiv: 1906.04904.

[17] Carl Doersch. Tutorial on Variational Autoencoders. *arXiv:1606.05908 [cs, stat]*.

[18] Randal Douc, Olivier Cappé, and Eric Moulines. Comparison of resampling schemes for particle filtering. *Proc. Image and Signal Processing and Analysis (ISPA), Nanjing Sept. 2005*, pages 64–69, 2005.

[19] Simon Duane, A. D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, September 1987.

[20] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[21] Sébastien Gerchinovitz, François Malgouyres, Edouard Pauwels, and Nicolas Thome. Fondements théoriques du Deep Learning, January 2019.

[22] Mark Girolami and Ben Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214.

[23] Mark Girolami, Ben Calderhead, and Siu Chin. Riemannian Manifold Hamiltonian Monte Carlo. July 2009.

[24] Peter Hall and Christopher C Heyde. *Martingale limit theory and its application.* Academic press, 1980.

[25] Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *ICLR*, 2017.

[26] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[27] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, November 1999.

[28] Belhal Karimi, Marc Lavielle, and Eric Moulines. On the Convergence Properties of the Mini-Batch EM and MCEM Algorithms. *CMAP*.

[29] Mohammad Emtiyaz Khan, Reza Babanezhad, Wu Lin, Mark Schmidt, and Masashi Sugiyama. Convergence of proximal-gradient stochastic variational inference under non-decreasing step-size sequence. *arXiv preprint arXiv:1511.00146*, 2015.

[30] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014. arXiv: 1412.6980.

[31] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat].*

[32] Estelle Kuhn and Marc Lavielle. Coupling a stochastic approximation version of EM with an MCMC procedure. *ESAIM: Probability and Statistics*, 8:115–131.

[33] Kenneth Lange. A Gradient Algorithm Locally Equivalent to the Em Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(2):425–437.

[34] Tuan Anh Le, Maximilian Igl, Tom Rainforth, Tom Jin, and Frank Wood. Auto-Encoding Sequential Monte Carlo. *arXiv:1705.10306 [stat].*

[35] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[36] Romain Lopez, Jeffrey Regier, Michael I Jordan, and Nir Yosef. Information Constraints on Auto-Encoding Variational Bayes. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6114–6125. Curran Associates, Inc.

[37] Christos Louizos and Max Welling. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 2218–2227. JMLR.org. event-place: Sydney, NSW, Australia.

[38] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *arXiv:1611.00712 [cs, stat]*.

[39] Jean-Marie Monnez. Almost sure convergence of stochastic gradient processes with matrix step sizes. *Statistics and Probability Letters*, 76:531–536.

[40] Christian A. Naesseth, Scott W. Linderman, Rajesh Ranganath, and David M. Blei. Variational Sequential Monte Carlo. *arXiv:1705.11140 [stat]*.

[41] Radford M. Neal. Annealed Importance Sampling. *arXiv:physics/9803008*.

[42] Radford M Neal. Mcmc using hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*, pages 139–188. Chapman and Hall/CRC, 2011.

[43] Ronald C. Neath. *On Convergence Properties of the Monte Carlo EM Algorithm.* Institute of Mathematical Statistics, 2013.

[44] Jimmy Olsson and Johan Westerborn. An efficient particle-based online EM algorithm for general state-space models. *IFAC-PapersOnLine*, 48(28):963–968, January 2015.

[45] Tom Rainforth, Adam Kosiorek, Tuan Anh Le, Chris Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh. Tighter Variational Bounds are Not Necessarily Better. In *PMLR*, pages 4277–4285.

[46] Danilo Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *International Conference on Machine Learning*, pages 1530–1538, June 2015.

[47] Francisco Ruiz and Michalis Titsias. A Contrastive Divergence for Combining Variational Inference and MCMC. In *International Conference on Machine Learning*, pages 5537–5545.

[48] Tim Salimans, Diederik Kingma, and Max Welling. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. In *International Conference on Machine Learning*, pages 1218–1226, June 2015.

[49] Arash Vahdat, Evgeny Andriyash, and William G. Macready. Learning Undirected Posteriors by Backpropagation through MCMC Updates. *arXiv:1901.03440 [cs, stat]*.

[50] Yixin Wang and David M. Blei. Frequentist Consistency of Variational Bayes. *Journal of the American Statistical Association*, 0(0):1–15.

[51] Christopher Wolf, Maximilian Karl, and Patrick van der Smagt. Variational Inference with Hamiltonian Monte Carlo.

[52] C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt. Advances in Variational Inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008–2026, August 2019.