Variational AutoEncoders : Theory, implementation and unanswered questions

Cédric Beaulac

December 17, 2019

1 Introduction

Latent variable models are known by statisticians as a family of models that increases the expressiveness of the observed variable distribution. The best-known model within this family is the Gaussian Mixture Model (GMM) where we assume the observed variables x comes from one of the k Normal distributions that defines the mixture. Hidden Markov Models are another example of well-known latent variable models.

The improved flexibility of such model is great, but comes at a price; it is way more complicated to maximize the likelihood of such models. One reason GMMs are so popular is because we have been able to fit these models to observed data using algorithms such as the Expectation-Maximization (EM) algorithm.

In this short document, we introduce the GMM paradigm as a simple model before introducing Variational AutoEncoders (VAE), which are even more flexible but harder to fit than GMMs. Section 2 contains the GMM small introduction and an explains the relation between the maximization of the Evidence Lower Bound (ELBO) and the EM algorithm. In section3 we introduce the theoretical motivation behind VAEs and discuss the current implementations of such models. We invite readers who want to go straight to the point to skip section 2.2. We also discuss the differences between the theory and the implementation and we try to understand the effect of such differences. Finally, in the last section we introduce the supervised VAE paradigm and once again discuss the differences between the theory and the implementation.

2 Gaussian Mixture Model

Let us introduce the Gaussian Mixture Model first and foremost. The main purpose of such model is to allow for more complex observed-data distributions. The mixture component adds an extra layer of flexibility which allows observations to belong to one of many Normal distributions.

The simplest way to define such model is to consider z to be a latent discrete variable and x to be the observed continuous variable. Simply speaking, from now on, we will always refer to z as latent variable and we define latent variable as unobserved variable that have an effect on the observed variable. We will refer to x as the set of observed or collected data.

For the Gaussian Mixture Model, z represents the component, i.e. if $z_i = k$ it means that x_i is distributed according to $N(\mu_k, \sigma_k)$. For every possible component k we have a distinct mean (μ) and standard deviation (σ), i.e. the parameters of the distribution of x depends on the latent component such that :

$$p(x|z=k) = N(\mu_k, \sigma_k)$$

As we can see, the distribution for x is now a lot more complicated as we are only able to pin down the conditional distribution given z :

$$\begin{aligned} p(x) &= \sum_z p(x,z) \\ &= \sum_z p(x|z)p(z) \end{aligned} \tag{1}$$

In equation 1 we see why $p(x)$ is more expressive but also why it is harder to fit. If we were to attempt to compute the likelihood of an observed data set using the logarithm is not enough to make the function easily differentiable :

$$\begin{aligned} p(x) &= \prod_{i=1}^{n} p(x_i) \\ &= \prod_{i=1}^{n} \sum_{j=1}^{K} p(x_i|z_i=j)p(z_i=j) \\ \Rightarrow \ln p(x) &= \sum_{i=1}^{n} \ln \sum_{j=1}^{K} p(x_i|z_i=j)p(z_i=j) \end{aligned} \tag{2}$$

The solution utilized to fit this model is the Expectation-Maximization algorithm. We will explain how this technique maximize the likelihood function later. Even though there exist intuitive formulation of the EM algorithm for GMMs we introduce the ELBO-KL decomposition of the likelihood in the next section. This might be a bit of overkill, but it will prove useful when we introduce VAEs.

2.1 The ELBO-KL decomposition

Let us demonstrate this popular likelihood decomposition. Notice that these equations hold for any distribution $q(z)$:

$$\begin{aligned}
\ln p(x) &= \ln\left(p(x,z)/p(z|x)\right)\\
&= \ln\left(p(x,z)\right) - \ln\left(p(z|x)\right)\\
&= \ln\left(p(x,z)\right) - \ln\left(p(z|x)\right) + \ln q(z) - \ln q(z)\\
&= \ln\left(\frac{p(x,z)}{q(z)}\right) - \ln\left(\frac{p(z|x)}{q(z)}\right)\\
\Rightarrow \mathbf{E}_{q(z)}[\ln p(x|\theta)] &= \mathbf{E}_{q(z)}\left[\ln\left(\frac{p(x,z)}{q(z)}\right)\right] - \mathbf{E}_{q(z)}\left[\ln\left(\frac{p(z|x)}{q(z)}\right)\right]\\
\Rightarrow \ln p(x|\theta) &= \mathbf{E}_{q(z)}\left[\ln\left(\frac{p(x|z)p(z)}{q(z)}\right)\right] - \mathbf{E}_{q(z)}\left[\ln\left(\frac{p(z|x)}{q(z)}\right)\right]\\
&= \mathcal{L}(q,p) + KL(q||p).
\end{aligned} \tag{3}$$

Notice that since the KL divergence is greater or equal than 0, then $\mathcal{L}(q)$ is a lower bound for the likelihood. It is defined as the evidence lower bound (ELBO) or as the variational lower bound. Almost all techniques for inference on graphical models are based upon the maximisation of this lower bound.

The Jenson inequality is also an easy way to observe why the ELBO is a lower bound:

$$\begin{aligned}
p(x) &= \int p(x|z)p(z)dz\\
&= \int p(x|z)p(z)\frac{q(z)}{q(z)}dz\\
&= \int \frac{p(x|z)p(z)}{q(z)}q(z)dz\\
\Rightarrow \ln p(x) &\geq \int \ln\frac{p(x|z)p(z)}{q(z)}q(z)dz\\
&= \mathbf{E}_{q(z)}\left[\ln\left(\frac{p(x|z)p(z)}{q(z)}\right)\right]
\end{aligned} \tag{4}$$

2.2 The EM algorithm : Maximization of the variational lower bound for tractable posterior.

The Expectation-Maximization algorithm is an iterative procedure that slowly increases the value of the variational lower bound with two distinct steps. To begin, we will explain the key ideas and it how the algorithm intends to maximize the likelihood and then we will see how this procedure leads to a maximization of the variational lower bound.

We've already discussed the issue of maximizing the likelihood of the observed data set in models with latent variables. Remember that :

$$\ln p_\theta(x) = \sum_{i=1}^{n} \ln \sum_{j=1}^{K} p_\theta(x_i|z_i = j) p_\theta(z_i = j)$$

and thus maximizing the likelihood is analytically impossible. This model contains $(k-1)+2k$ parameters. The $k-1$ parameters for the categorical distribution of z, the k means μ_j and the k variances σ_j^2 that correspond to the k normal components. Even though this is not a perfect notation we refer to all of these parameters as θ.

For now, let's assume that the complete data set contains both $\mathbf{x}$ and $\mathbf{z}$, then the complete log likelihood $\ln p_\theta(\mathbf{x}, \mathbf{z})$ is straight forward to maximize. Since we only observed $\mathbf{x}$, the only information we have about $\mathbf{z}$ is through the posterior distribution of z $p_\theta(\mathbf{z}|\mathbf{x})$. Therefore we cannot directly use the complete-data log likelihood $\ln p_\theta(\mathbf{x}, \mathbf{z})$ and instead we will compute the expectation of the complete log likelihood under the posterior distribution with the current set of parameters:

$$\mathbf{E}_{p_{\theta^o}(\mathbf{z}|\mathbf{x})}[\ln p_\theta(\mathbf{x}, \mathbf{z})] = \sum_{\mathbf{z}} p_{\theta^o}(\mathbf{z}|\mathbf{x}) \ln p_\theta(\mathbf{x}, \mathbf{z}) = Q(\theta, \theta^o). \tag{5}$$

Computing this expectation is the **E** step of the EM algorithm. Then, we maximize $Q(\theta, \theta^o)$this expectation with respect to θ, the **M** step. Here, θ^o stands for old θ and is the set of parameters under which we computed the posterior of the latent $p_{\theta^o}(\mathbf{z}|\mathbf{x})$. For a mixture of Gaussian, we compute this expectation is a simple manner :

$$p_{\theta^o}(z_n = k|\mathbf{x}_n) = \frac{\pi_k N_{\theta_k}(\mathbf{x}_n)}{\sum_{j=1}^{K} \pi_j N\theta_k(\mathbf{x}_n)} = \gamma(z_k) \tag{6}$$

where $\pi_k = p(z = k)$. Then optimizing $Q(\theta, \theta^o)$ is easy and leads to the following estimates :

$$\begin{aligned}
\mu_k^{new} &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n \\
\Sigma^{new} &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(\mathbf{x}_n-\mu_k^{new})(\mathbf{x}_n-\mu_k^{new})^T \\
\pi_k^{new} &= \frac{N_k}{N}
\end{aligned}$$

where $N_k = \sum_{n=1}^{N}\gamma(z_{nk})$.

Now we are going to use the ELBO-KL decomposition to demonstrate how this technique succeed at maximizing the likelihood and we will motivate the need for other techniques.

In the previous section, we've demonstrated that $\mathcal{L}(q_\varphi, p_\theta)$ is a lower bound for the log-likelihood of the observed data $\ln p_\theta(\mathbf{x})$. The ELBO is function of the parameters θ of the distribution p_θ and of the parameters φ of a distribution over the latent variables $q_\varphi(\mathbf{z})$. Let's demonstrate how every both step of the EM algorithm increase $\mathcal{L}(q_\varphi, p_\theta)$ in their own way. In the **E** step, we maximize $\mathcal{L}(q_\varphi, p_\theta)$ with respect to φ while in the **M**, we then maximize $\mathcal{L}(q_\varphi, p_\theta)$ with respect to θ.

The **E** step considers the effect of $\mathbf{z}$ through the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ under the current set of parameters, and then compute the expectation of the complete log-likelihood under that posterior distribution. We see that this step maximizes $\mathcal{L}(q_\varphi, p_\theta)$ with respect to $q_\varphi(\mathbf{z})$ by setting $q_\varphi(\mathbf{z}) = p_\theta(\mathbf{z}|\mathbf{x})$. Since $\mathcal{L}(q_\varphi, p_\theta) = \ln p_\theta(\mathbf{x}) - KL(q||p)$ setting $q_\varphi(\mathbf{z}) = p_\theta(\mathbf{z}|\mathbf{x})$ makes the KL divergence vanish which effectively maximize $\mathcal{L}(q_\varphi, p_\theta)$. This also highlights one of the main assumptions necessary to use an EM algorithm, we need to be able to compute $p_\theta(\mathbf{z}|\mathbf{x})$.

In the following **M** step, we maximize $Q(\theta, \theta^o)$ with respect to the parameters θ. Let's actually see what happens when we substitute $q_\varphi(\mathbf{z})$ by $p_{\theta^o}(\mathbf{z}|\mathbf{x})$ in the lower bound :

$$
\begin{aligned}
\mathcal{L}(q_\varphi, p_\theta) &= \sum_{\mathbf{z}} q_\varphi(\mathbf{z}) \ln\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\varphi(\mathbf{z})}\right) \\
\Rightarrow \mathcal{L}(p_{\theta^o}(\mathbf{z}|\mathbf{x}), \theta) &= \sum_{\mathbf{z}} p_{\theta^o}(\mathbf{z}|\mathbf{x}) \ln\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{p_{\theta^o}(\mathbf{z}|\mathbf{x})}\right) \\
&= \sum_{\mathbf{z}} p_{\theta^o}(\mathbf{z}|\mathbf{x}) \ln p_\theta(\mathbf{x},\mathbf{z}) - \sum_{\mathbf{Z}} p_{\theta^o}(\mathbf{z}|\mathbf{x})) \ln p_{\theta^o}(\mathbf{z}|\mathbf{x})) \\
&= Q(\theta, \theta^o) + \text{const.}
\end{aligned}
$$

Since the **M** step maximizes $Q(\theta, \theta^o)$ with respect to θ is surely maximize the ELBO in parallel since the two are equal up to a constant.

Assuming the EM algorithm is successful at fitting such latent variable model it is reasonable to ask why would we need any other techniques to optimize the parameters in latent variable models. The problem with EM is that it requires us to compute $p_{\theta^o}(\mathbf{z}|\mathbf{x})$ which may not be feasible in some cases. The propose solution in those cases is to directly maximise the ELBO in another way.

3 Variational AutoEncoders

VAEs [5, 6] are an attempt to create an even more flexible family of latent variable models. In the GMM, we have a simple mapping from z to x $\theta : \mathcal{N}^k \to \mathcal{R} \times \mathcal{R}^+$, each component k has it's own set own set of parameters; if $z = j$ then $p(x|z) = N(\mu_j, \sigma_j)$. For our demonstration of VAEs, we assume the distribution of x is still normal but so is z. Typically, we assume $z \sim N(0, 1)$. Since z is now continuous, we could identify this model as a Gaussian mixture where we have infinitely many components. Furthermore, to allow this, the parameters are now a continuous function of the latent variable z; $\theta = [\mu_x, \sigma_x] = f_x(z)$, to use a short notation, we identify $\mu_x(z)$ as the function that takes z as input and return the parameters μ_x associated with this value and same for $\sigma_x(z)$ or simply $\theta(z) : \mathcal{R} \to \mathcal{R} \times \mathcal{R}^+$. If θ is a continuous function, it ensures that points that are alike (close to one another) in the latent space are also near in the observation space.

We define z as the latent representation of the observation x or as its code. We define $\theta(z)$

as the decoding function which takes in the code and return the parameters of the observed-data distribution. The utilize the strength from the latest machine learning development, we use a Neural Network (NN) function as decoding function. It has the benefit of allowing for a maximum amount of flexibility but in turn makes the posterior of the latent $p(z|x)$ intractable and consequently the EM algorithm cannot be used.

3.1 Maximization of the ELBO

Because it is impossible to compute the posterior distribution of the latent $p(z|x)$ we cannot compute $\mathbf{E}_{p_\theta(\mathbf{z}|\mathbf{x})}[\ln p_\theta(\mathbf{x}, \mathbf{z})]$ and thus we must find another way to maximize the likelihood function $p_\theta(x)$. The proposed solution is to replace $p_\theta(z|x)$ with an approximate distribution $q_\varphi(z|x)$ and maximize the ELBO :

$$\mathcal{L}(\varphi, \theta) = \mathbf{E}_{q_\varphi(z|x)}\left[\ln p_\theta(z) + \ln p_\theta(x|z) - \ln q_\varphi(z|x)\right]$$

Remember that the ELBO is a lower bound to the observed-data log-likelihood $\ln p_\theta(x)$. We define $q_\varphi(z|x)$ as the encoding distribution. The effect of x on the encoding distribution goes through the parameters φ. We define q_φ to be a Normal distribution and once again we explained the parameters φ as a function of x; $\varphi = [\mu_z, \sigma_z] = f_z(x)$. Once again, this function is set to be a NN that we will refer as $\varphi(x)$ from now on.

Since we cannot directly compute, $\mathbf{E}_{q_\varphi(z|x)}\left[\ln p_\theta(z) + \ln p_\theta(x|z) - \ln q_\varphi(z|x)\right]$ we will produce a Monte Carlo estimate of the ELBO. In other words, we draw from z from $q_\varphi(z|x)$ then compute $\ln p_\theta(z) + \ln p_\theta(x|z) - \ln q_\varphi(z|x)$. Then, we have to maximize this value with respect to its parameters. As we are looking at conditional distributions, the parameters are functions of the given variables, thus we will actually optimize the functions. Simply put, we optimize the parameters of the NN functions $\theta(z), \varphi(x)$.

Here is a breakdown of the procedure :

1. Process observation x through the NNs φ to produce $\varphi(x)$
2. Sample z from $q_{\varphi(x)}(z|x)$
3. Process latent sample z through the NNs θ to produce $\theta(z)$.

4. Compute $\ln p_{\theta(x)}(z) + \ln p_{\theta(z)}(x|z) - \ln q_{\varphi}(z|x)$, the ELBO Monte Carlo estimate.
5. Maximize the ELBO with respect to the weights and biases of φ and σ
6. Repeat 1-5 until convergence.

3.2 Practical uses

3.2.1 Dimensionality Reduction

VAE is a unsupervised learning model like k-means clustering, GMM or Principal Component Analysis (PCA). Just like these other techniques, dimensionality reduction is one reason to use VAE. The code z is a of much lower dimension, and given the fitted encoding function φ and decoding function θ we can easily encode large observations x into the parameters of their lower-dimension representation z and also decode this representation to get the parameters of the reconstructed observation distribution. We define it as a probabilistic dimensionality reduction rather than a deterministic one.

Dimensionality reduction is very useful for storage purpose. Besides, the lower dimensional representation can be used to group up together observations that are alike. Finally, it is quite common to apply supervised learning techniques to the latent representation itself.

3.2.2 Generator

A VAE is considered a generative model. Indeed, since a distribution is assumed for the latent variable z, $z \sim N(0,1)$ it is possible to generate new *observations* using ancestral sampling :

1. Sample z from $N(0,1)$
2. Process z through the NNs θ to get $\mu_x(z)$ and $\sigma_x(z)$.
3. Sample x from $N(\mu_x(z), \sigma_x(z))$.

3.3 Objective function

Now that we have established the theory and the algorithms we are ready to discuss the current implementations of this model. Machine learning optimization commonly defines the objective function as the function subject to the optimization. Then a gradient-based optimizer is used to maximize or minimize this objective function. For example, one could define the mean squared error (MSE) of a classifier h as the objective function: $\frac{1}{n}\sum_{i=1}^{n}(h(x_i) - y_i)^2$. Then we compute the gradient of the objective function with respect to the parameters of h. Finally, we use a gradient-based optimizer, for example stochastic gradient descent (SGD). Modern machine learning based on NN functions relies on this strategy as we are able to compute the gradient with respect to the parameters of NNs using the chain rule for derivatives.

Back in the VAE set up, we try to maximize the log-likelihood function but since it the likelihood function in impossible to compute we maximize the ELBO instead, a lower bound of the log-likelihood function. Specifically, the objective function is a Monte Carlo sample of the ELBO:

$$\ln p_\theta(z) + \ln p_\theta(x|z) - \ln q_\varphi(z|x).$$

At least that's the theory behind most implementation. To discuss further the current successful implementations, let us reorganize the terms in the ELBO:

$$\begin{aligned}
\mathcal{L}(\varphi,\theta) &= \mathbf{E}_{q_\varphi(z|x)}\left[\ln p_\theta(z) + \ln p_\theta(x|z) - \ln q_\varphi(z|x)\right] \\
&= \mathbf{E}_{q_\varphi(z|x)}\left[\ln p_\theta(x|z) - (\ln q_\varphi(z|x) - \ln p_\theta(z))\right] \\
&= \mathbf{E}_{q_\varphi(z|x)}\left[\ln p_\theta(x|z)\right] - \mathbf{E}_{q_\varphi(z|x)}\left[\ln q_\varphi(z|x) - \ln p_\theta(z)\right] \\
&= \underbrace{\mathbf{E}_{q_\varphi(z|x)}\left[\ln p_\theta(x|z)\right]}_{\text{Reconstruction error}} - \underbrace{KL\left(q_\varphi(z|x)|p_\theta(z)\right)}_{\text{Regularization term}}
\end{aligned} \tag{7}$$

It is common to perceive the ELBO with respect to these two terms. Actually we can see this as a penalized optimization problem where we want to maximize the first term and where the second term works as some kind regularization that prevents $q_\varphi(z|x)$ from drifting far away from a $N(0,1)$.

In their implementation those two components are often modified to create better results, but little is known about the induced model optimized with those modified objective function.

3.3.1 Implementation of the objective function

In this section we will introduce two inconsistencies between the theory and the successful implementation of the VAE objective functions.

Let's first discuss the reconstruction term of the objective function :

$$\begin{aligned} \ln p_\theta(x|z) &= \ln\left(\frac{1}{\sqrt{2\pi\sigma(z)^2}}\exp\left(\frac{-(x-\mu(z))^2}{2\sigma(z)^2}\right)\right) \\ &= -\frac{1}{2}\ln\left(2\pi\sigma(z)^2\right) - \frac{(x-\mu(z))^2}{2\sigma(z)^2} \end{aligned} \tag{8}$$

Most implementations we found online do not maximize the reconstruction term of equation 8. Instead the NN θ returns an output of the same size as x and minimize the mean squared error between x and the reconstructed x. This is equivalent to maximizing the log-likelihood for a normal distribution with a fixed $\sigma = 1$:

$$-\frac{1}{2}\ln(2\pi) - \frac{(x-\mu(z))^2}{2} \propto -(x-\mu(z))^2$$

Research Question :

- Does this really produce a more useful model ?
- If yes, why ?

Based on empirical result, we assume that not optimizing $p_\theta(x|z)$ with respect to $\sigma(z)$, i.e. fixing $\sigma(z) = 1$, produce better reconstructed images. Our assumption is that it is due to the singularity problem that also affects GMM [1].

Briefly, suppose we have a GMM with 2 components and suppose that mean of component 1 μ_1 equals one of the observed data points; $\mu_1 = x_j$ for some j. The contribution to the likelihood function of that point becomes :

$$p(x_1|\mu_1, \sigma_1) = \frac{1}{\sqrt{2\pi}}\frac{1}{\sigma_1}.$$

As $\sigma_1 \to 0$ then $p(x_1|\mu_1, \sigma_1) \to \infty$ and thus having these singularities effectively maximize the likelihood. We suspect that the presence of these singularities is the cause for the poor generative performances of models where we optimize the reconstruction term with respect to $\sigma(z)$ instead of fixing $\sigma(z) = 1$.

Secondly, the β-VAE [4] implementation modifies the objective function the following way :

$$\ln p_\theta(x|z) - \beta(\ln q_\varphi(z|x) - \ln p_\theta(z)).$$

In other words, they add a parameter to increase or decrease the KL-divergence contribution of the ELBO in the objective function. Increasing the β above 0 was shown to improve the disentanglement of the latent representation [2] but it is not clear what is the resulting model. With $\beta \neq 1$, we are not maximizing the lower bound of the log-likelihood anymore.

Research questions

- What distribution are we optimizing ?
- Why is this new distribution *better* ?

From a generative model perspective, making sure $q_\varphi(z|x)$ is close to $p_\theta(z)$ is very helpful. Since we learn the decoding distribution $p_\theta(x|z)$ based on samples from $q_\varphi(z|x)$ the decoding distribution will struggle if data generated from $q_\varphi(z|x)$ and from $p_\theta(z)$ are vastly different.

3.4 Implementation : Ancestral sampling

We previously discussed how VAEs are often used as generative models. In that case, the following Ancestral Sampling scheme is used :

1. Sample z from $N(0,1)$
2. Process z through the NN θ to get $\mu_x(z)$ and $\sigma_x(z)$
3. Sample x from $N(\mu_x(z), \sigma_x(z))$,

or it should be. As a matter of fact in a desired to generate observations with high likelihood most generative models based on VAEs output $\mu_x(z)$ instead of sampling from $N(\mu_x(z), \sigma_x(z))$. So in popular implementation the following generative technique is used :

1. Sample z from $N(0,1)$
2. Process z through the NN θ to get $\mu_x(z)$ and $\sigma_x(z)$
3. Output $\mu_x(z)$.

This is not ancestral sampling and the consequences of such choice is unknown. More importantly if we combine this generative scheme with the Reconstruction Term explained in the previous section ($(x - \mu_x(z))^2$) then we truly got rid of all the probabilistic components of x. Indeed, the resulting model is totally deterministic in x given z and should no longer be considered a latent variable model.

Research question

- What is this new generative process ?
- Did we achieve our original goal now that we went back to a fully deterministic model ?

3.5 Contribution of neural networks

In the work of Kingma [5, 6] that introduces VAEs, NNs are proposed for $\theta(z)$ and $\varphi(x)$. This decision is reasonable considering most of the recent progress in Machine Learning was drive by the recent success of NNs.

NNs are continuous function, which ensures that observations that are nearby to one another are projected to lower space values that are nearby to one another as well, which is an important. Furthermore, NNs are considered universal continuous functions approximators. More precisely, Csáji [3] claims in his thesis :

> The universal approximation theorem claims that the standard multilayer feed-forward networks with a single hidden layer that contains finite number of hidden neurons, and with arbitrary activation function are universal approximators in $C(R^m)$.

It seems like NNs are an appropriate choice and we don't see any implementation problems in the current state of things.

3.6 Visualization of the different implementations

We used the MNIST data set to visualize how different implementation result into different latent space representations, generative processes and reconstructions. We have used a supervised VAE were labels were also given to the encoder and decoder. Therefore the ability to produce the right digit should not be a concerned.

3.6.1 Probabilistic decoder against Deterministic decoder

It is difficult to approach all of these modification to the theory separately but we will do our best. The first comparison we will illustrate the difference between a probabilistic decoder, where both μ and σ are learned to one where only μ is fitted, in other words where we minimize the reconstruction error. We used a β-VAE with a moderately large β, the selection of β will be illustrated later in this section

The theoretical proposition for a latent variable model leads the following :

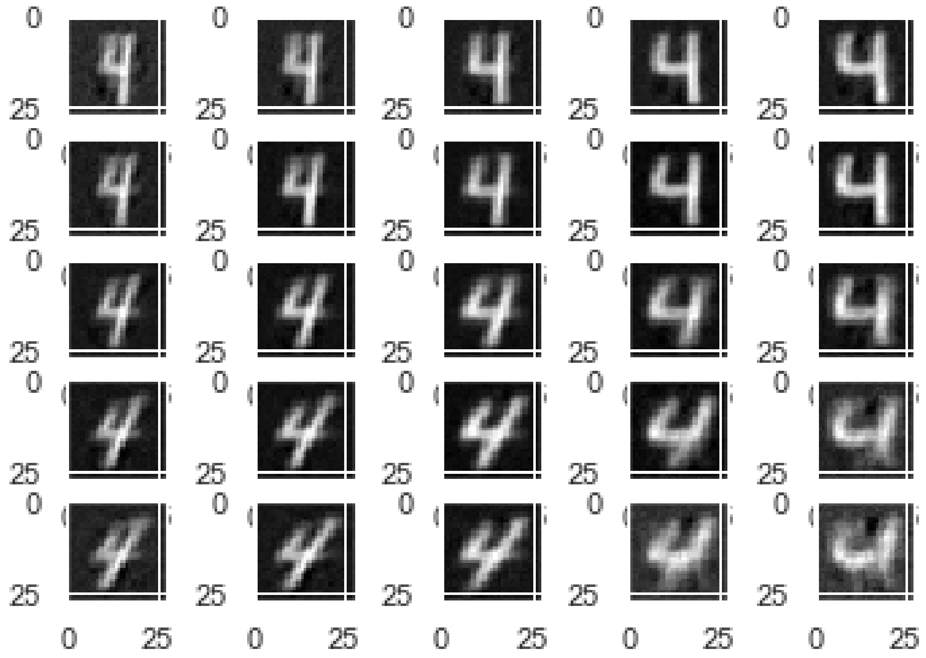


Figure 1: Latent representation for μ given the digit is four.

Based on this we can generate x from $p(z)$ and process that through $\mu_x(z)$ and $\sigma_x(z)$. Here is what $\mu(z)$ looks like for 10 samples of z :

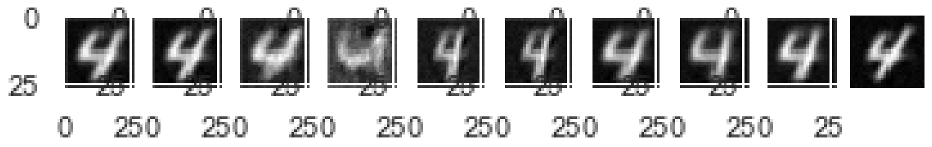


Figure 2: Samples of z from $p(z)$ processed through the NN μ_x

But this is not the ancestral sampling process proposed for VAEs, as a matter of fact, the images above are not samples from $p(x|z)$ they are $\mu_x(z)$. If we sample from $p(x|z)$ we get images like these :

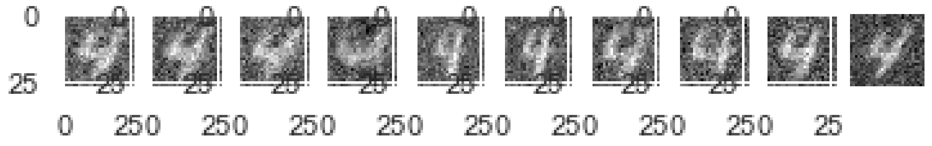


Figure 3: Samples from $p(x|z)$

Multiple things might explain why these images are horrible. For one, since pixels are drawn independently they are not correlated with one another. Perhaps a fully parametrize covariance structure for $p(x|z)$ would fix this problem. Also since the domain of a Normal variable is $(-\infty, \infty)$ when we sample from $N(\mu_x(z), \sigma_x(z))$ we have values higher than 1 and lower than 0 which do no belong in the original domain of those images.

We can also learn a different σ parameter for every individual pixels which leads to these kinds of images :

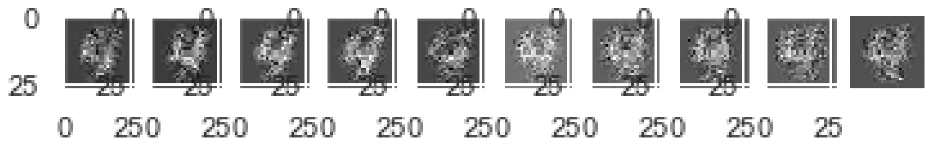


Figure 4: Samples from $p(x|z)$

where what surround the picture is clear since there is little variance observed in these parts but where the location of the digits is even more blurry since these pixels have high variance.

Outputing $\mu(z)$ does not produce pretty images because μ has high likelihood in a Normal distribution but because output *mu* for every pixels create a perfectly correlated image in Distribution, the distribution for all pixel value being exactly 0.5. To Illusatrate this we sample z from $p(z)$ and we produced images by outputing 5 images for every z : $(\mu_x(z) - 2\sigma_x(z), \mu_x(z) - \sigma_x(z), \mu_x(z), \mu_x(z) + \sigma_x(z), \mu_x(z) + 2\sigma_x(z))$:

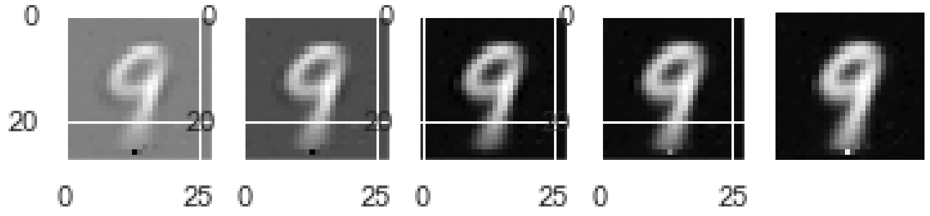


Figure 5: $(\mu_x(z) - 2\sigma_x(z), \mu_x(z) - \sigma_x(z), \mu_x(z), \mu_x(z) + \sigma_x(z), \mu_x(z) + 2\sigma_x(z))$

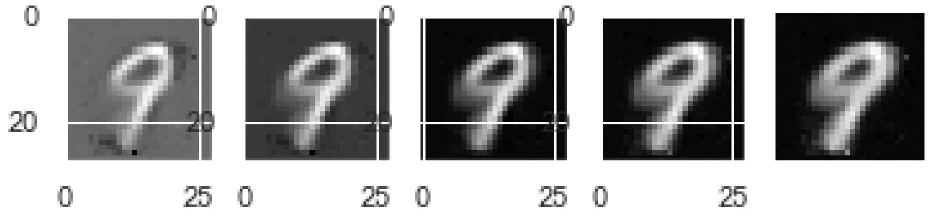


Figure 6: $(\mu_x(z) - 2\sigma_x(z), \mu_x(z) - \sigma_x(z), \mu_x(z), \mu_x(z) + \sigma_x(z), \mu_x(z) + 2\sigma_x(z))$

Let us compare these with a simpler model where we do not learn σ_x and where the images generated are simply $\mu_x(z)$, in other words $x = \mu_x(z)$ is a deterministic given z.

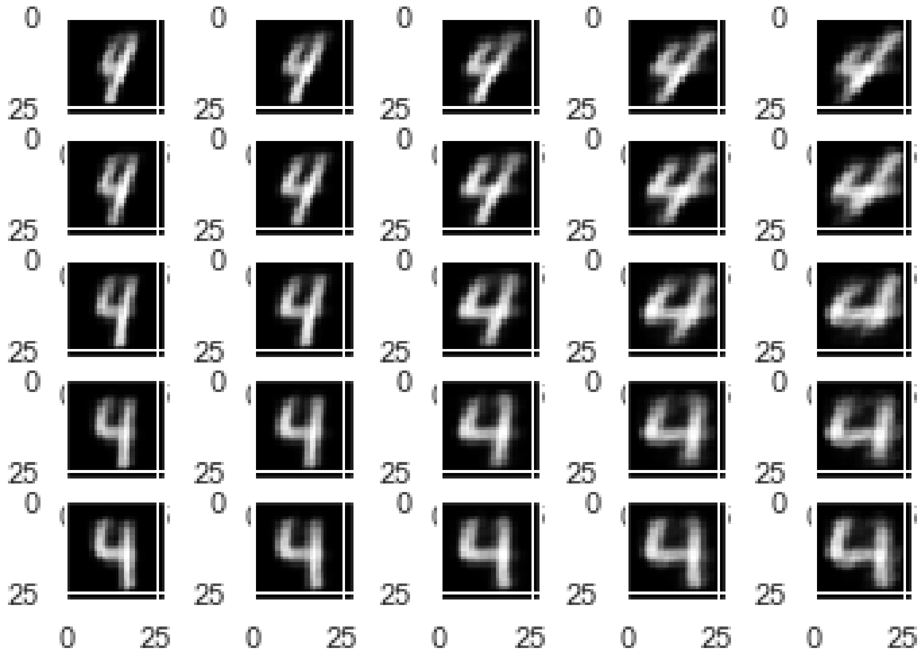


Figure 7: Latent representation for μ given the digit is four.

Since we did not learn σ, we cannot draw from $p(x|z)$ and automatically output $\mu_x(z)$ when generating images, which results in :

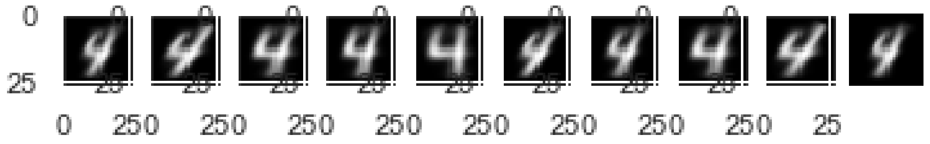
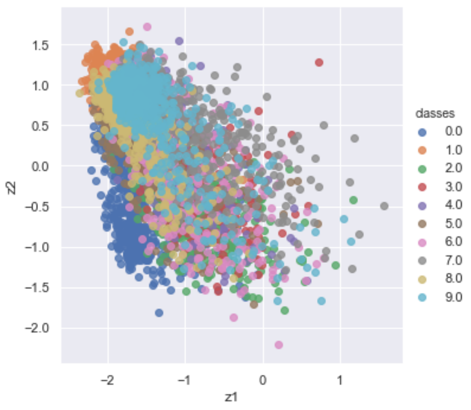


Figure 8: Samples of z from $p(z)$ processed through the NN μ_x

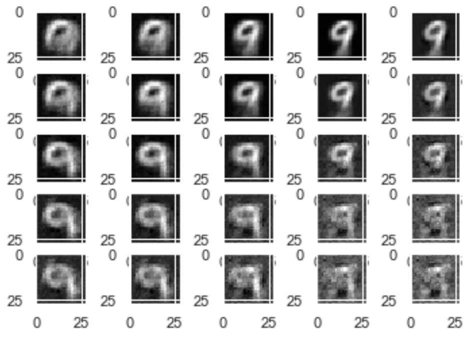
These images look good, but they were not generated from the proposed Ancestral Sampling technique described in the literature and they are not the result of a latent variable model but a single variable model (z) process through a NN. This new generative process resembles more GAN than a latent variable model. Samples from that model do look better at eyesight but are not the result of a successful implementation of the model proposed in the literature.

3.6.2 β-VAE

The next topic we would like to illustrate is the effect of β-VAEs. As we adjust β we can force the encoder to produces encoded observation as close to the prior as possible but it directly affect the expressivity of the generative process as illustrated below :

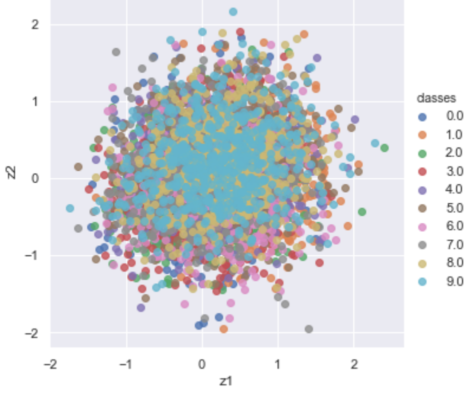


(a) Posterior distribution of the latent : $p(z|x)$

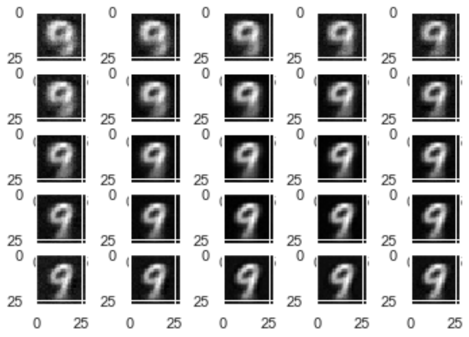


(b) Latent representation of $\mu_x(z)$ for the digit nine.

Figure 9: Small β

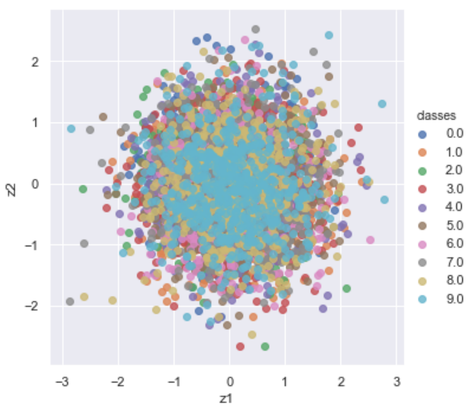


(a) Posterior distribution of the latent : $p(z|x)$

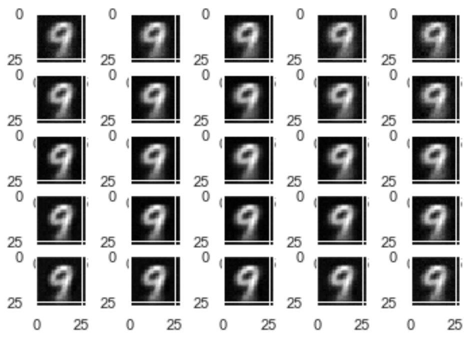


(b) Latent representation of $\mu_x(z)$ for the digit nine.

Figure 10: Medium β



(a) Posterior distribution of the latent : $p(z|x)$



(b) Latent representation of $\mu_x(z)$ for the digit nine.

Figure 11: Large β

When β is too small, the generative process can not work as illustrated in figure 9. Since the posterior $p(z|x)$ is extremely different from the prior $p(z)$ this is a problem because we use Monte Carlo estimates of the ELBO to train the model. Remember that we sample from $p(z|x)$ to do so and thus the encoder is solely trained on observations coming from $p(z|x)$. If $p(z|x)$ is too different from $p(z)$ most of the sample generate from $p(z)$ will result in value never observed by the decoder previously and will result in the fuzzy images observe in figure 9b.

On the other hand, if β gets too large then the algorithm puts too much pressure on making $p(z|x)$ Gaussian and not enough to recreate precise images. Instead $p(x|z)$ is almost the same regardless of z. Lucas & al. recently addressed the situation and defined posterior collapse properly establishing the σ parameter as the real driving force for this issue. Setting small σ parameters ensure the latent representation is thigh for μ.

3.6.3 Fixing σ

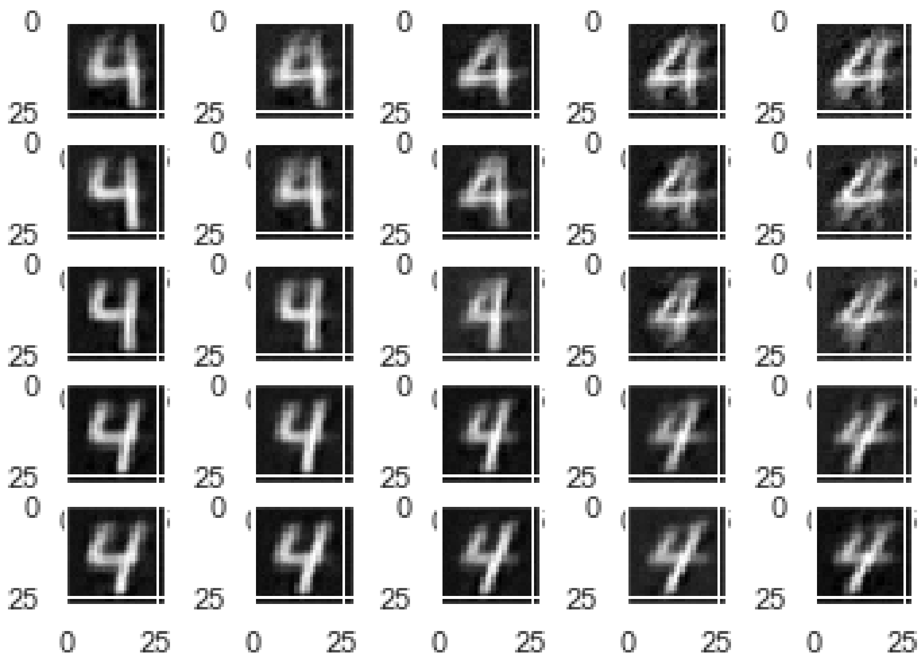


Figure 12: Latent representation for μ given the digit is four.

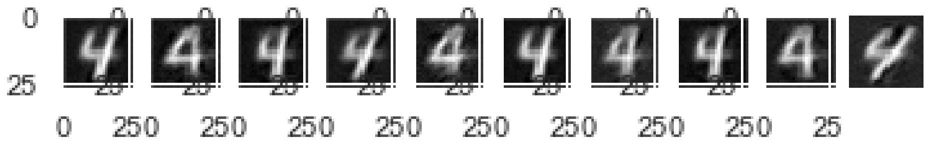


Figure 13: Samples from $p(x|z)$

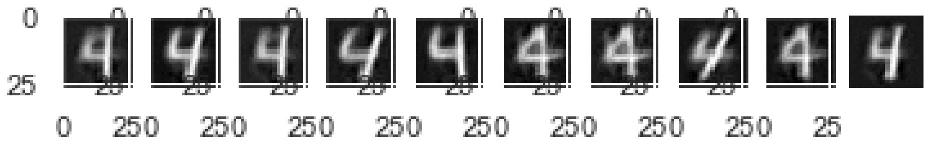


Figure 14: Samples from $p(x|z)$

3.7 Overall Observation

It seems that σ is more meant to be a parameter meant to facilitate learning, i.e. map images slightly different to a similar latent representation by allowing some observed variability given similar latent representation rather than a parameter with a dedicated generative purpose.

It still it unclear if this added variability on observed variable is actually useful from a generative perspective.

4 Probabilistic Principal Component Analysis

In section 2, we have introduced GMM as a simple latent variable model we can refer too and compare newer models such as VAEs. GMM was considered a well-known technique and we included in here to facilitate the explanation of VAEs motivation and structure.

In the section we discussed a *bridging model* that really stands between GMMs and VAEs in terms of structure; Probabilistic Principal Component Analysis (pPCA) [8, 7]. Our goal is to use pPCA to provide some intuition on the more complex VAE model. We also expect some differences in their ability to fit data, the difference in the resulting fitted models is cause by their theoretical difference and we hope that our understand of pPCA will help us shed some light in the VAE theory and implementation.

pPCA is a continuous latent variable model similar to VAEs but where the NN functions are replace with a simple linear function. Once again a prior distribution is assumed for the latent and a resulting conditional distribution of observations :

$$\mathbf{z} \sim N(0, I)$$

$$\mathbf{x}|\mathbf{z} \sim N(\mathbf{W}\mathbf{z} + \mu, \sigma^2 I)$$

To bring back notation used in the previous section, μ_x is actually a function of the latent variable : $\mu_x(z) = \mathbf{W}\mathbf{z} + \mu$. From a generative perspective we can perceive a D-dimensional observed variable x being defined as a linear transformation of an M-dimensional latent variable z plus an additional Gaussian noise :

$$x = \mathbf{W}\mathbf{z} + \mu + \varepsilon$$

where z is an M-dimensional standard Normal variable, μ is a D-dimensional vector of constant and ε is a D-dimensional zero-mean Normal variable with covariance $\sigma^2 I$. This corresponds to a linear-Gaussian model for which $p(x)$ is computable and for which we can compute likelihood and maximize it.

Nonetheless, since we have approached fitting GMM with the EM algorithm and later explained the parallel between EM and ELBO maximisation we will introduce the EM solutions for pPCA but remember that an exact maximum likelihood solution also exists.

4.1 EM solution for pPCA

To begin, once again, let us compute the expectation of the complete log-likelihood :

$$\begin{aligned}\mathbf{E}[\ln p(\mathbf{x}, \mathbf{z}|\mu, \mathbf{W}, \sigma^2)] = -\sum_{n=1}^{N} \Bigg(& \frac{D}{2}\ln(2\pi\sigma^2) + \frac{1}{2}\mathrm{Tr}(\mathbf{E}[\mathbf{z_n}\mathbf{z_n^T}]) \\ & + \frac{1}{2\sigma^2}||\mathbf{x_n} - \mu||^2 - \frac{1}{\sigma^2}\mathbf{E}[\mathbf{z_n}]^T\mathbf{W^T}(\mathbf{x_n} - \mu) \\ & + \frac{1}{2\sigma^2}\mathrm{Tr}(\mathbf{E}[\mathbf{z_n}\mathbf{z_n^T}]\mathbf{W^t}\mathbf{W})\Bigg)\end{aligned} \tag{9}$$

We set μ to its maximum-likelihood value $\mu = \bar{x}$ and the EM algorithm is strictly used to determine the value for the two other set of parameters $\mathbf{W}$ and σ. The **E**-step consist of using old parameters value to compute $\mathbf{E}[\mathbf{z_n}]$ and $\mathbf{E}[\mathbf{z_n}\mathbf{z_n^T}]$:

$$\mathbf{E}[\mathbf{z_n}] = \mathbf{M^{-1}}\mathbf{W^T}(\mathbf{x_n} - \mathbf{x})$$
$$\mathbf{E}[\mathbf{z_n}\mathbf{z_n^T}] = \sigma^2\mathbf{M^{-1}} + \mathbf{E}[\mathbf{z_n}]\mathbf{E}[\mathbf{z_n}]^T$$

where

$$\mathbf{M} = \mathbf{W^T}\mathbf{W} + \sigma^2 I$$

Then we replace these values in equation 9 and we obtain the maximum likelihood solutions for W and σ^2 for the current iteration :

$$\begin{aligned}
\mathbf{W}_{\text{new}} &= \left[\sum_{n=1}^{N}(\mathbf{x}_n - \bar{x})\mathbf{E}[\mathbf{z_n}]^T\right]\left[\sum_{n=1}^{N}\mathbf{E}[\mathbf{z_n}\mathbf{z_n^T}]\right] \\
\sigma_{new}^2 &= \frac{1}{ND}\sum_{n=1}^{N}\Big(||\mathbf{x_n} - \mu||^2 - 2\mathbf{E}[\mathbf{z_n}]^T\mathbf{W^T}_{\text{new}}(\mathbf{x_n} - \mathbf{x}) \\
&\quad + \text{Tr}(\mathbf{E}[\mathbf{z_n}\mathbf{z_n^T}]\mathbf{W}_{\text{new}}^{\mathbf{T}}\mathbf{W}_{\text{new}}\Big)
\end{aligned}$$

5 Conclusion

The VAE model as define in the literature [5, 6] is built upon a rigorous theory and the described model is both innovative and a big contribution to the fields of Machine Learning and Statistics. It extends latent variable models to allow for more flexible functions between the latent and the observations space and has empirically performed well on some problems.

But there seems to be a gap between the theory and popular implementations. It is not clear if this gap is intended and if the effect of this gap and the resulting model have been studied. In this small report, we have highlighted some of these differences and discussed some potential research avenues.

References

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[2] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[3] Balázs Csanád Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.

[4] Irina Higgins, Loic 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. *ICLR*, 2(5):6, 2017.

[5] D. P Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, December 2013.

[6] Diederik P. Kingma. *Variational Inference & Deep Learning : A New Synthesis*. PhD thesis, Universiteit van Armsterdam, 10 2017.

[7] Michael E Tipping and Christopher M Bishop. Mixtures of probabilistic principal component analyzers. *Neural computation*, 11(2):443–482, 1999.

[8] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.