About
'Auto-Encoding Variational Bayes'

Author :

Rony Abecidan

Introduction

Usually, we naively think that Bayes methods are only useful for small datasets because they involve slow computations and are not very beneficial with big datasets. Nevertheless, things changed since deep learning methods have been introduced...

Since last few years, deep generative models are in the spotlight thanks to their inherent capacity to generate realistic contents. Among them, there is one category involving Bayesian learning called Variational Autoencoders (VAEs). These models produced impressive results with a huge amount of data and proved at the same time that it's possible to make Bayesian Learning efficient with big datasets.

Here, I am studying the pionneering paper of VAEs called "Auto-Encoding Variational Bayes" written by Diederik P. Kingma and Max Welling [1]. In this paper, the authors propose a **"stochastic variational inference and learning algorithm that scales to large dataset"** which corresponds to the first VAE model. In practice, the algorithm described in this paper can be applied to tasks of different kind such as denoising, impainting and super-resolution. In this report we will treat exclusively the generative purpose since it's the most famous application of VAEs and it's also the kind of task chosen by the authors for their experiments.

In a first time, we will give some contextual materials enabling to better understand the problems encountered in bayesian inference when we deal with large datasets. This will be also the occasion to talk about variational inference and the main limitation of classical autoencoders for a generative purpose. Then, we will present the Variational Autoencoder proposed by the authors and see to what extent it helps us to mitigate the problem met previously. At last, I propose to make some experiments enabling to assess the efficiency of the model and discuss about some variations we can make to to obtain the best results possible. All my experiments are in the repo : `https://github.com/RonyAbecidan/VAE`

1 Contextual material

1.1 Intractable posterior densities

A crucial problem we met when we do Bayesian statistics consists in approximating the most accurately possible intractable posterior densities. Concretely, if we name θ our parameter to optimize, we can write the Bayes formula as : $\underbrace{p(\theta|\mathcal{D})}_{\text{posterior}} = \frac{\overbrace{p(\mathcal{D}|\theta)p(\theta)}^{\text{likelihood x prior}}}{\underbrace{p(\mathcal{D})}_{\text{evidence}}}$

In this formula, **the bottleneck is the denominator**. Indeed, this term called **the evidence** is a normalizing factor which, in some sense, takes into account "all the values possible" for our parameter θ.

$$p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)\ \mathrm{d}\theta$$

In many cases, it's simply impossible to have this term in closed form or it involves an exponential time to compute and, this is why Bayesian inference is often tough.

In order to mitigate this problem, we discussed in our course about several strategies.

- Inverse sampling when we know the inverse c.d.f. of the posterior to sample
- Rejection sampling when we only know the numerator of the posterior density
- Gibbs sampling when we can compute easily conditioned probabilities
- MCMC when we can have access to a quantity proportional to the posterior density

If we consider now the case where we have an unknown and tricky distribution to sample with, and we are working with big datasets of observations living in a high dimensional space (such as images), we risk to meet intractable computations. Fortunately for us, Kingma and Welling have more than one trick up their sleeve. They succeed in mitigating the intractable computations using what we call the **Variational Inference** framework.

1.2 Variational Inference

In the **Variational Inference** framework, we typically consider that there is an hidden variable **z** called a **latent variable**, that could explain the data **x** we observed. For instance, if we observe our productivity over the days of a month, it could be explained by our mood each day of this month.

The variational inference consists actually in approximating the intractable posterior density $p(z|x)$ which explains how the latent variables vary, conditioned on data. In order to do so, we consider this posterior as a family member of well known densities parametrized by unknown parameters that we have to optimize.

Intuitively, when the marginal likelihood of our model is high, this is likely that it's the one that produced the data. Hence, maximizing this likelihood could help us to derive the optimal parameters to choose. Let's take the same scenario as the authors and see what's coming.

We imagine that we have a dataset $\mathcal{D} = (x_i)_{1 \leq i \leq N}$ made of N i.i.d samples of a certain distribution discrete or continuous. Since we are going to make variational inference, we consider that the data is generated by a random process involving a latent variable $z \sim p_\theta(z)$ where θ is an unknown parameter. Now, as explained before, we are going to approximate the intractable posterior $p_\theta(z|x)$ by a distribution $q_\Phi(z|x)$ belonging to some variational family parametrized by Φ. Such model is called a **recognition model**.

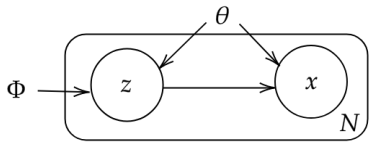


Figure 1: Graphical illustration of the problem scenario

Let's compute now the marginal log likelihood of our model : $\log p_\theta(x_1, \cdots, x_N) = \sum_{i=1}^{N} \log p_\theta(x_i)$ since the observations are i.i.d.

For each i, we have :

$$\begin{aligned}
\log p_\theta(x_i) &= \log[p_\theta(x_i)] \underbrace{\int q_\Phi(z|x_i)\ \mathrm{d}z}_{=1} = \int q_\Phi(z|x_i) \log[p_\theta(x_i)]\ \mathrm{d}z \\
&= \int q_\Phi(z|x_i) \log[p_\theta(x_i)]\ \mathrm{d}z + \int q_\Phi(z|x_i) \log[\frac{q_\Phi(z|x_i)}{p_\theta(x_i, z)}]\ \mathrm{d}z - \int q_\Phi(z|x_i) \log[\frac{q_\Phi(z|x_i)}{p_\theta(x_i, z)}]\ \mathrm{d}z \\
&= \int q_\Phi(z|x_i) \log[p_\theta(x_i) \frac{q_\Phi(z|x_i)}{p_\theta(x_i, z)}]\ \mathrm{d}z - \int q_\Phi(z|x_i) \log[\frac{q_\Phi(z|x_i)}{p_\theta(x_i, z)}]\ \mathrm{d}z \\
&= \underbrace{\int q_\Phi(z|x_i) \log[\frac{q_\Phi(z|x_i)}{p_\theta(z|x_i)}]\ \mathrm{d}z}_{\mathsf{KL}(q_\Phi(z|x_i)||p_\theta(z|x_i))} \underbrace{- \int q_\Phi(z|x_i) \log(\frac{q_\Phi(z|x_i)}{p_\theta(x_i, z)})\ \mathrm{d}z}_{:=\mathcal{L}(\theta, \Phi, x_i)} \\
&= \mathsf{KL}(q_\Phi(z|x_i)||p_\theta(z|x_i)) + \mathcal{L}(\theta, \Phi, x_i)
\end{aligned}$$

Thanks to that result we notice that $\log(p_\theta(x_i)) \geq \mathcal{L}(\theta, \Phi, x_i)$ since the KL is positive by the Jensen Inequality. Hence, maximizing $\sum_i \mathcal{L}(\theta, \Phi, x_i)$ imply maximizing the marginal likelihood and this is precisely the idea that is exploited in variational inference. Moreover, it's possible to rewrite $\mathcal{L}$ in a more interesting form :

$$\begin{aligned}
\mathcal{L}(\theta, \Phi, x_i) &= -\int q_\Phi(z|x_i) \log(\frac{q_\Phi(z|x_i)}{p_\theta(x_i, z)})\mathrm{d}z \\
&= -\int q_\Phi(z|x_i) \log[\frac{q_\Phi(z|x_i)}{p_\theta(z) p_\theta(x_i|z)}]\mathrm{d}z \\
&= -\mathsf{KL}(q_\Phi(z|x_i)||p_\theta(z)) + \mathbb{E}_{q_\Phi(z|x_i)}[\log[p_\theta(x_i|z)]]
\end{aligned}$$

We will see later how this **variational lower bound** can be interpreted. Now we are familiar with the notion of variational inference, let's talk about autoencoders.

1.3 A brief presentation of autoencoders

When we work with observations living in a very large dimensional space, it could be interesting to project it into a space of low dimension before starting any further computations. We can see that idea as linking a large vector to a narrow one while keeping as much as possible the information contained in the original one. This is exactly the idea of autoencoders.

More precisely we can see an autoencoder as a combination of two neural networks :

- An encoder which projects an input $x \in \mathcal{X}$ in a low dimensional latent space. This enables to obtain what we call a **code** : $z = \Phi(x)$
- A decoder which decodes z enabling to projecting it back to the original space $\mathcal{X}$: $x' = \Psi(z)$

Now in order to have a relevant projection, we have to precise what is our loss function. Here, we generally take the **reconstruction error** : $||x - x'||^2$. Having a combination of neural networks minimizing this loss should intuitively lead to a relevant projection. Once the autoencoder is trained, we could naively think that it could be efficient for a generation task. Indeed, since we have a way to decode

points from the latent space, maybe we can simply sample points in this latent space and decode them. Unfortunately, doing such a thing is likely to fail in most cases.

Indeed when we sample a point from the latent space, there is no absolute guarantee that this point will be decoded in a meaningful point in the input space. This comes from the reconstruction loss which forces the autoencoder to correctly decode the latent codes issued from the training set without ensuring a meaningful decoding for all the other points. In other terms, the latent space is likely to be not enough "smooth" in the sense that if $x \sim x' \in \mathcal{X}$, $z_x \not\approx z_{x'}$. This is an overfitting behaviour intimely linked to the objective we have considered.

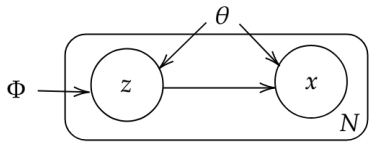
We can now present the **Variational autoencoder** which is simply an extension to the Autoencoder able to reduce the overfitting phenomenon using Variational inference.

2 The Variational Auto-Encoder

The Variational autoencoder is the model presented in this article. It corresponds to a special case of variational inference where the recognition model q_Φ is obtained thanks to a neural network. In this section, we are going to present this model, see how is it trained and exhibit its advantages.

2.1 A first presentation of the model

As said in the introduction, we will restrict ourself here to the case of a generative model, i.e. we are looking for a model able to learn and reproduce the data distribution. For that, we are going to consider again the following variational inference framework.



Now, we could propose that the recognition model follows a law whose parameters are determined by a neural network :

$$\boxed{q_\Phi(z|x_i) \sim \mathcal{L}(f(x_i, \Phi)) \text{ with f determined by a neural network}}$$

We could also consider to apply the same idea for the reconstruction model $p(x|z)$ with an other law (potentially the same one) :

$$\boxed{p_\theta(x|z) \sim \mathcal{L}'(g(z, \theta)) \text{ with g determined by a neural network}}$$

Doing so we end up with what we called the **Variational autoencoder** (also called VAE for short). The term "autoencoder" is not innocuous. In fact, we could see our recognition model as an encoder from our data distribution to our latent space and, in the same fashion, we could consider our reconstruction model as a decoder from the latent space to the data distribution. Furthermore, the structure of a VAE is very similar to an autoencoder.

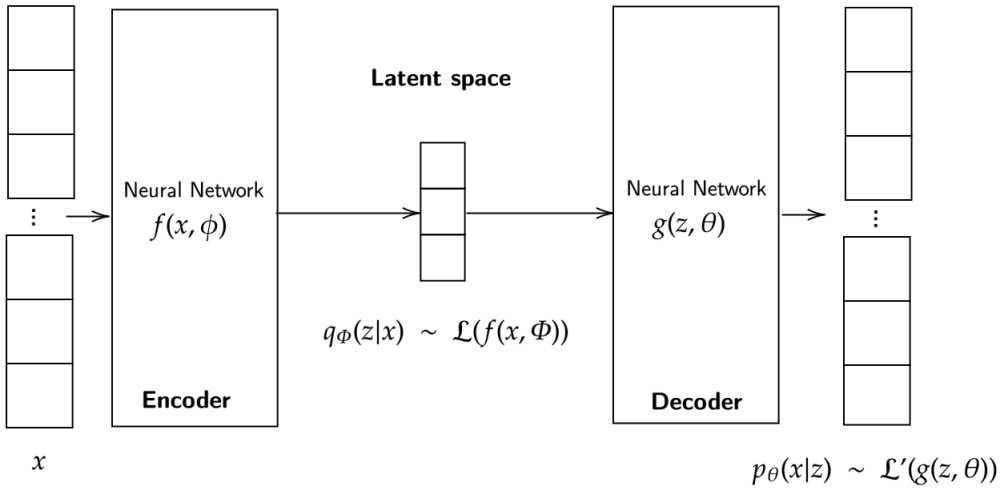


Figure 2: Traditional architecture of a Variational AutoEncoder

The big question is how are we going to optimize Φ and θ ?. As we have seen before, a way to do it could simply consists in maximizing the sum of the variational lower bound we have exhibited before :

$$\mathcal{L}(\theta, \Phi, x_i) = \mathbb{E}_{q_\Phi(z|x_i)}[\log[p_\theta(x_i|z)] - \mathrm{KL}(q_\Phi(z|x_i)||p_\theta(z))$$

In the paper, they consider that the first term of this lower bound could be interpreted as **a negative reconstruction error** and the second one as a **regularizer**. We can understand easily this analogy using for our both models a normal law.

Doing this choice, $\log[p_\theta(x_i|z)] \quad \alpha \quad -||x_i - \mu_{p_\theta(x_i|z)}||^2$ and here we can recognize the negative reconstruction error which quantify how much the reconstruction of x_i is far from x_i on average.

Considering the -KL term, we know this term is maximized when the KL is minimized, i.e. when the recognition model and the prior distribution of z are the same. Hence, the role of this term is to make sure that the recognition model stays close to the prior distribution. To what extent now this prevent the model from overfitting ?.

If we consider that the variance of the recognition model is zero, the only meaningful term of the loss is the reconstruction loss and in fact, we are back to the autoencoder model. Now, this model often suffers from overfitting since the latent space built is not regular enough. Hence, the role of the KL term becomes clear. It enables to prevent the model from encoding points which are too far in the latent space and doing so, it produces a smoother latent space in which it could be easier to produce meaningful latent code. Now that we understand how the VAE works, let's see how are we going to train it.

2.2 How to train a VAE ?

As we've said before, the training of a VAE consists in finding the optimal parameters θ and Φ which enable to maximize the sum of the variational lower bounds $\mathcal{L}(\theta, \Phi, x_i)$. In order to do that, the authors have simply proposed to use a **mini-batch stochastic gradient descent**. For the KL term, it is often easy to deduce it in closed form given reasonable assumptions on the distributions. However, for the reconstruction term it is not as simple as it may appear at first glance. Let's call this term $R(\theta, \Phi)$ for simplicity.

For the decoder we have :

$$\begin{aligned}\nabla_\theta R(\theta, \Phi) &= \int q_\Phi(z \mid x_i) \nabla_\theta \log p_\theta(x_i \mid z) dz \\ &= \mathbb{E}_{q_\Phi(z|x_i)} \nabla_\theta \log p_\theta(x_i \mid z)\end{aligned}$$

Now, the expected value $\nabla_\theta \log p_\theta(x_i \mid z)$ is often intractable. However we can approximate this quantity simply using the Monte Carlo estimator :

$$\mathbb{E}_{q_\Phi(z|x_i)} \nabla_\theta \log p_\theta(x_i \mid z) \approx \frac{1}{L} \sum_l \nabla_\theta \log p_\theta(x_i \mid \widehat{z}_l) \text{ with } \widehat{z}_l \sim q_\Phi(z \mid x_i)$$

For the encoder, it's tricky. First, this time we can't easily obtain an expectation like before since $q_\Phi(z \mid x_i)$ depends of Φ and hence, we can't move the gradients until $\log p_\theta(x_i \mid z)$. In order to solve this problem, we can simply multiply and divide the quantity in the integral by $q_\Phi(z \mid x_i)$ (in order to have an expectation suddenly appearing).

$$\begin{aligned}\nabla_\Phi R(\theta, \Phi) &= \int \nabla_\Phi \; q_\Phi(z \mid x_i) \log p_\theta(x_i \mid z) dz \\ &= \int q_\Phi(z \mid x_i) \underbrace{\frac{\nabla_\Phi \; q_\Phi(z \mid x_i)}{q_\Phi(z \mid x_i)}}_{\nabla_\Phi \log(q_\Phi(z \mid x_i))} \log p_\theta(x_i \mid z) dz \\ &= \mathbb{E}_{q_\Phi(z|x_i)}[\nabla_\Phi \log(q_\Phi(z \mid x_i)) \log p_\theta(x_i \mid z)] \end{aligned} \tag{1}$$

We could think naively that now, we just have to consider again a Monte Carlo estimator for approximating this expectation. Nevertheless, it's not a good idea here since the variance of this estimator could be very high.

We can see it very easily at the beginning of the training. Indeed, at that moment $p_\theta(x_i \mid z)$ will be very low since the model hasn't learn yet how to reproduce faithfully the data distribution. Thus, this will apply that $|\log p_\theta(x_i \mid z)|$ will be very high inducing a very high variance for our estimator. Hence, we will need to sample an incredibly high amount of points so that we can approximate our expectation reasonably i.e. this estimator is intractable. Fortunately for us, the authors have proposed a clever trick to override this problem.

From now, we will consider that we are working for instance with normal laws as we usually do when we work with VAEs since gaussians are easy to manipulate and prove to be effective for many applications. In that case, instead of sampling directly $\widehat{z}_l \sim q_\Phi(z \mid x_i) = \mathcal{N}(\mu, \Sigma)$ we could naturally sample $\epsilon \sim p(\epsilon) = \mathcal{N}(0, I)$ and consider then $z = \epsilon \odot \Sigma + \mu$. This is called the **reparametrization trick**.

Remark : As the authors precised, this trick can be applied when we know the inverse of the c.d.f. doing an inverse sampling, or when we have a 'location-scale' law that we could rewrite using a reference as the Gaussian law or when we can express the density as a composition of interesting densities.

Thanks to that trick, we can now write :

$$\begin{aligned}\nabla_\Phi R(\theta, \Phi) &= \int \nabla_\Phi \; q_\Phi(z \mid x_i) \log p_\theta(x_i \mid z) dz \\ &= \int p(\epsilon) \nabla_\Phi \log p_\theta(x_i \mid z = \epsilon \odot \Sigma + \mu) d\epsilon \\ &= \mathbb{E}_{p(\epsilon)} \nabla_\Phi \log p_\theta(x_i \mid z = \epsilon \odot \Sigma + \mu) \end{aligned} \tag{2}$$

Now, in practice, the Monte-Carlo estimator of (2) seems to often have a lower variance than the one of (1) and become tractable. Hence, that's the one we used in the VAE :

$$\mathbb{E}_{p(\epsilon)} \nabla_\Phi \log p_\theta(x_i \mid z = \epsilon \odot \Sigma + \mu) \approx \frac{1}{L} \sum_l \nabla_\Phi \log p_\theta(x_i \mid \widehat{z}_l = \epsilon_l \odot \Sigma_l + \mu_l) \text{ with } \epsilon_l \sim \mathcal{N}(0, I)$$

However, after a few search, it seems that it's not always true and the authors have forgot to mention that. In some cases, it even appears that it's not the case as you could see in [2] section 3.1.2.

3 A multi-armed VAE

In this part, I am going to present a strategy I have designed enabling to save times on the construction of a VAE for a particular problem.

3.1 A striking observation

In the paper studied, they tested the VAE only with very simple datasets such as the MNIST and the Frey Faces Datasets. These datasets do not present many varieties and can be handled easily with simple architectures such as Multi-Layers Perceptrons. Hence, we could think that the model is only good for simple cases but in reality it can handle much more challenging problems.

In practice, when we met a complex dataset, designing a generation model is very difficult. Different architectures lead to different results and take generally a long time to train. For finding a good model, the process if often iterative. We test a simple one at first, we see the results through a certain number of training epochs. Then, based on these results, we adjust the model adding or removing complexity according to our observations. The problem here is that this process can take a long time before achieving the perfect architecture for the specific problem we consider. I have personally observed that problem during my research project where I tried to generate faces with a GAN architecture. Now, is there a solution to help saving us time in this long and fastidious search of the best model ? I thought about that and I am going to describe a strategy that comes to my mind while doing my RL "mini" project.

3.2 The Multi-armed Bandit VAE

When we are constructing a VAE we have to make a choice for the architectures of the encoder and the decoder. However, there is infinitely many possibilities and no way to know in advance what will be the best structure for these two neural networks. Hence, I have imagined a situation that can potentially help us to find a good architecture for a problem in a reasonable time.

Let's say that we have fixed our encoder and we think about different possible architectures for the decoder. You can imagine for instance that you have one sentence to translate and several tools to do it such as Google, Linguee or Deepl. Which one is the best for your sentence ? We often see parallel architectures for the decoder and the encoder but nothing prevent us from breaking it to see what will happen. Here, I suggest to imagine the following scenario :

> We want to construct a VAE for a particular generation problem (that could be easy or complex). We are pretty confident on the architecture of our encoder but not really for the decoder. Hence, we decide to design K decoders and our goal will be to find among these decoders which one is going to render the best result in the long term.

This scenario can be considered as a **Multi-Armed Bandit problem** where our arms are the decoders and our rewards will be the negative reconstruction losses of the decoders.

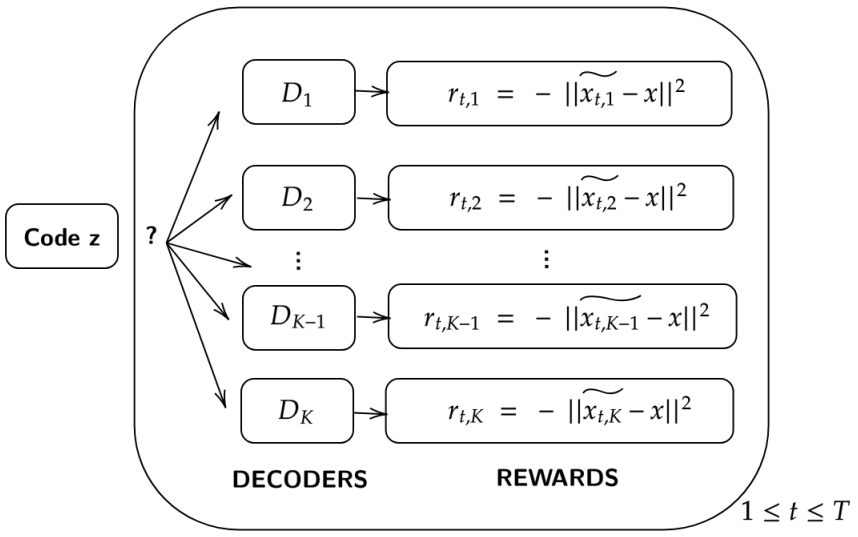
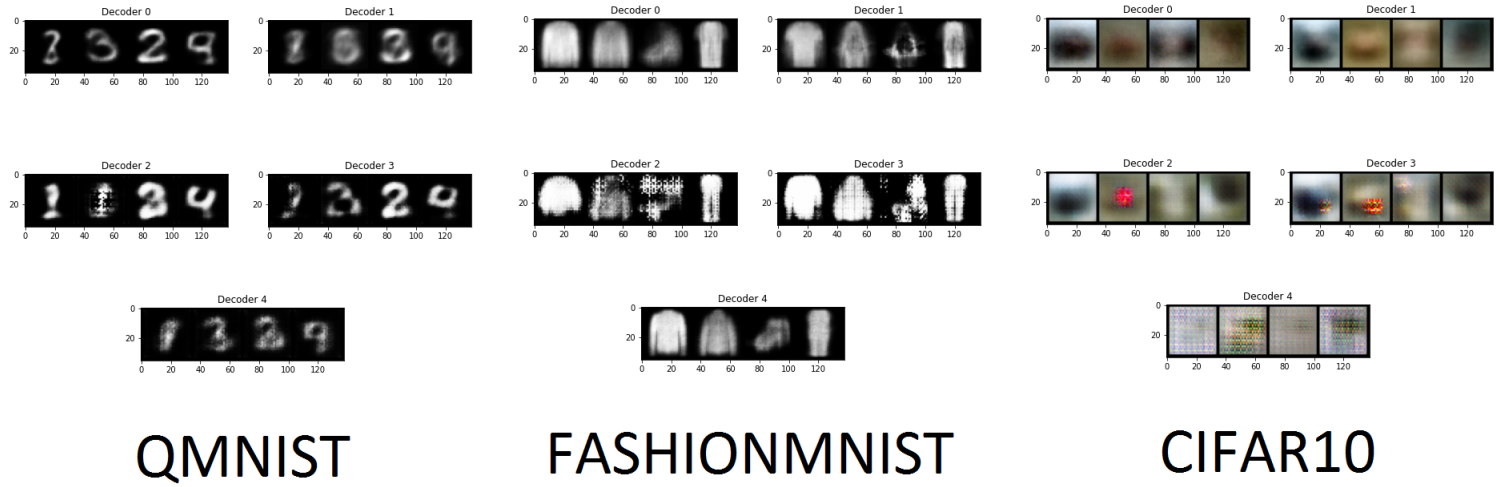


Figure 3: Illustration of the MAB scenario imagined

If we succeed in finding the best decoders in reasonable time, that will mean that we have potentially save times of trials and errors that could have been spend modifying the decoder after each training on several epochs. Hence, this goal is rather motivating.

I have tested this idea using 5 decoders and 3 datasets. All the details of the architectures are in the illustrative notebook of **my repository**.

I've chosen 3 datasets with gradually increasing complexities : The **QMNIST** dataset, the **FashionMNIST** dataset and the **CIFAR10** dataset. All the images were rescaled to the size 32x32 and pre-normalized[1]. For the recognition model, I considered a normal distribution while using a Bernouilli distribution for the pixels of the reconstructed images by the decoders. Hence, the reconstruction loss was naturally the **BCE Loss** as proposed by the authors. At last, I've chosen decoders of different natures like MLPs, deconvolutions and some combinations of a linear layer with deconvolutions. Let's observe now the results I obtained after a few epochs for the 3 datasets



As you see, the reconstruction is clearly different according to the decoders and we can easily identify towards which architectures we should go in order to obtain satisfying results. There is no one decoder which seems perfect for all the tasks and that shows that a specific architecture is needed for each dataset. Now, we can certainly improve the strategy extending this idea with multiple couple of encoders and decoders. Having the choice between K couples of encoders and decoders lead to many more possibilities but maybe more interesting results. Unfortunately, I didn't have the time to do this experiment because the computations involved are rather heavy and I am limited by Google Collab' for the usage of their GPU. We can also notice that we didn't succeed in capturing the distribution of the CIFAR10 images. This is not surprising since this dataset is rather complex containing a wide range of images presenting many varieties.

I have also thought about some ways to potentially improve the efficiency of the VAE even if I didn't have the time to test them.

At first, I think that may be interesting to change the KL loss in the cost function by an other loss measuring the dissimilarity between the encoding distribution and the prior of the latent variable. For instance, there is the Jensen-Shannon divergence or the Maximum Mean Discrepancy that could potentially help. Even if it is not a choice completely justified by the theory, intuitively, it can have the same regularizing impact on the latent space.

One other thing that could be interesting is to use a bayesian optimization technique enabling to tune the hyperparemeters of the model such as the learning rate, the latent space dimension or why not the batch size.

At last, I've noticed during my experiments that the choice of the reconstruction loss is also very important. Using a normal law for the decoder, the reconstruction loss is traditionally the **MSE** loss and it is generally observed that this kind of loss is rather adapted for RGB images. Nevertheless, we could imagine an other law for the decoder that will lead again to an other reconstruction loss maybe adapted for some particular situation. Hence, testing the reaction of a VAE using different decoding distributions could be interesting.

Conclusion and Perspectives

In this paper, the authors demonstrated that it's possible to make bayesian inference efficient with big datasets. The model presented has many applications and is rather well explained although some part of the computations are sometimes missing. Here, we have checked both theoretically and empirically the claims of the authors and we have seen through an illustrative experiment the potential of this strategy for a generation purpose. To extend this work, one could investigate about the other famous generative model using generative adversarial nets and make a comparison with the VAE. It is certainly interesting to see to what extent one model can be more efficient than an other for a particular problem.

References

[1] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: 1312.6114 [stat.ML].

[2] Yarin Gal. "Uncertainty in Deep Learning". PhD thesis. University of Cambridge, 2016.

[3] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* 112.518 (2017), 859–877. ISSN: 1537-274X. DOI: 10.1080/01621459.2017.1285773. URL: http://dx.doi.org/10.1080/01621459.2017.1285773.

[1] They are represented with only numbers between 0 and 1