Laboratoire de Probabilité, Statistiques et Modélisation

May 2023

Internship report notes

Master 2 - Data Science

Alexandre CHAUSSARD

Contents

1 Introduction

This part of the paper discusses the introduction to the subject.

2 Variational Auto-Encoders

2.1 Framework and optimization objective

We are interested in another kind of latent models, this time based on variational inference results to achieve a new kind of deep latent structure: the Variational Auto-Encoder (VAE). These latent models were introduced in 2013 by Kingma, better described in a more in depth paper in 2019: see [1]

Once again, we assume the observations X to be modelizable by a given distribution parameterized by θ:

$$X \sim p_\theta(x)$$

Determining θ holds to find one θ^* that would optimize a given objective, generally chosen as the maximum of likelihood. Indeed, if $\theta^* \in arg\max_\theta p_\theta(x)$, then such θ^* maximizes the density around the dense areas of the observations, which makes them highly likely to happen under such distribution p_{θ^*}. Hence, the maximum likelihood is a natural criterion:

$$\theta^* \in arg\max_\theta p_\theta(x)$$

However, such modelization does not include a latent structure. As a result, we try to enforce it by rewriting the objective as follows:

$$p_\theta(x) = \int_{\mathcal{Z}} p(x,z)dz$$

Using Bayes decomposition, we obtain the following objective:

$$p_\theta(x,z) = p_\theta(z)p_\theta(x|z)$$

Recall that the prior $p_\theta(z)$ and the a priori $p_\theta(x|z)$ are defined by the framework (ex: Bernoulli prior and Gaussian posterior gives the Gaussian mixture framework). However, the computation of the evidence $p_\theta(x)$ is generally intractable in practice, which also leads to a non-tractable posterior distribution: $p_\theta(z|x)$. As a result, not being able to compute the evidence leads to not being able to provide a gradient regarding θ, so we can not perform the backpropagation in a deep learning approach.

Note that there exist approximate inference techniques to compute the evidence and the posterior, but these are quite expensive and often yield poor convergence results.

To overcome this issue, we introduce a smart rewriting of the objective using variational inference. Indeed, let $q_\Phi(z|x) \approx p_\theta(z|x)$ to be learnt over Φ, one can write:

$$\begin{aligned}
\log p_\theta(x) &= \mathbb{E}_{q_\Phi(z|x)}[\log p_\theta(x)] \\
&= \mathbb{E}_{q_\Phi(z|x)}\left[\log \frac{p_\theta(x)}{q_\Phi(z|x)}\frac{q_\Phi(z|x)}{p_\theta(z|x)}\right] \\
&= \underbrace{\mathbb{E}_{q_\Phi(z|x)}\left[\log \frac{p_\theta(x)}{q_\Phi(z|x)}\right]}_{ELBO(q_\phi(z|x),p_\theta(x,z))} + D_{KL}\left[q_\Phi(z|x)||p_\theta(z|x)\right]
\end{aligned}$$

The first term of that decomposition is generally called the Evidence Lower BOund (ELBO), as it marks a lower bound to the evidence $\log p_\theta(x)$ since the KL divergence is a positive quantity:

$$\log p_\theta(x) \geq ELBO(q_\phi(z|x), p_\theta(x,z))$$

$q_\Phi(z|x)$ is an approximation of the true posterior $p_\theta(z|x)$ that we aim at learning in a family of distributions. For instance,

$$q_\Phi(.|x) \sim \mathcal{N}(\mu(x), \Sigma(x))$$

would be an approximation of the true posterior by a Gaussian distribution. Notice that the true posterior may very not likely be Gaussian, which creates a first complexity error in our model.

Despite being a lower bound on the true maximum likelihood objective, the ELBO is actually tractable. Indeed, as we continue the computation:

$$\begin{aligned} ELBO(q_\phi(z|x), p_\theta(x,z)) &= \mathbb{E}_{q_\Phi(z|x)}[\log p_\theta(x,z)] - \mathbb{E}_{q_\Phi(z|x)}[\log q_\Phi(z|x)] \\ &= \mathbb{E}_{q_\Phi(z|x)}[\log p_\theta(x|z)] - D_{KL}[\log q_\Phi(z|x) \| p_\theta(z)] \end{aligned}$$

Another remarkable fact, is that when maximizing the ELBO, we are actually minimizing the KL divergence between the estimated and the true posterior. Hence, one can define the ELBO as a suboptimal objective to our problem that we get to maximize to obtain (Φ^*, θ^*), the parameters of our model.

2.2 Reparameterization trick

Even though the gradient of the ELBO is well defined for θ, it is not possible to compute the differential relatively to Φ yet, as it requires samples from the approximation to the posterior $q_\Phi(z|x)$ to compute $\mathbb{E}_{q_\Phi(z|x)}[\log p_\theta(x|z)]$.

Since, sampling is not a differentiable operation, we make use of the change of variable formula, so that for a bijective transformation $z = \phi_x(\epsilon)$, we get:

$$p(z) = p(\epsilon) \det \left| \frac{\partial \epsilon}{\partial z} \right|$$

Hence, if we take ϵ a random variable of density $p(\epsilon)$ that does not depend on θ, Φ nor x, so that $z = \phi_x(\epsilon)$, for any L_1 function f,

$$\mathbb{E}_{q_\Phi(z|x)}[f(z)] = \mathbb{E}_{p(\epsilon)}[f(z)]$$

As a result, the samples are not obtained through q_Φ anymore but through $p(\epsilon)$, so that can safely perform derivation of the ELBO relatively to Φ and backpropagate our gradient through the network.

2.3 Architecture

The vanilla architecture of the VAE is described by the following illustration:

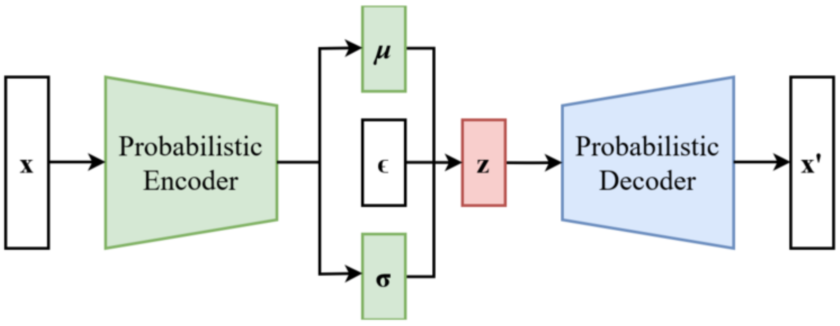

Figure 1: Illustration of a VAE with Gaussian prior (wikipedia)

The first part is generally called the encoder, as it turns a sample x into its latent representation z by modelizing the posterior $q_\Phi(z|x)$. The second part is then called the decoder, as it throws a latent representation in the sample space. The latest can even serve as a generative architecture, as one can sample from the latent space through $q_\Phi(z|x)$, and decode it to obtain a new sample.

As we can see more clearly in that illustration, we can see that Φ and θ are trained jointly through the ELBO, both serving for one part of the VAE at a time.

The training procedure is straightforward: the entry is a sample x and the output objective is the same sample x. We aim at train the VAE for learning the data space and its latent representation by learning how to reconstruct the samples through it.

2.4 Limits for our problem

As we have seen through the reparameterization trick, training a VAE architecture requires to be able to backpropagate the gradient of the ELBO at each step. We namely had to perform the reparameterization trick to circumvent the randomness operation which is not differentiable. As a result, learning a discrete posterior is not possible with such architecture, since we would have to perform a projection of the output of the encoder on a discrete space, which is not a differentiable operation.

Yet, learning discrete representation of our data seems much more natural than continuous latent ones. As we tend to categorize things as much as we can, describing behaviors with words for instance. Furthermore, a discrete representation facilitates the interpretation of the latent space by ordering data distribution in simple bins.

The next architecture, called the VQ-VAE, stands as a first fork option to the VAE with discrete posterior.

3 VQ-VAE

3.1 Quick overview

Introduced in [3], VQ-VAE architecture provides a framework to compute discrete posterior distributions $q_\Phi(z|x)$. To compare that model with the VAE, we start by introducing the architecture of the model for which an illustration is given below:

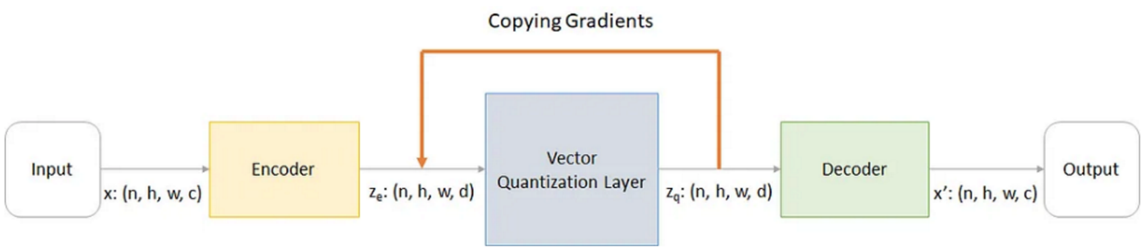

Figure 2: VQ-VAE architecture (source: Medium)

As we can see on figure 2, the major difference with the vanilla VAE architecture lies in the vector quantization step which enables to project the output of the encoder denoted by $z_e(x)$ onto a discrete embedding dictionary $(e_1, \ldots, e_K)$ by a simple distance argument:

$$k = arg\min_j \|z_q(x) - e_j\|_2, \quad z_q(x) = e_k$$

The projection of $z_e(x)$ on that discrete dictionary is denoted by $z_q(x)$, and serves as the input of the decoder. For further visual representation of the vector quantization layer, an illustration is given below.

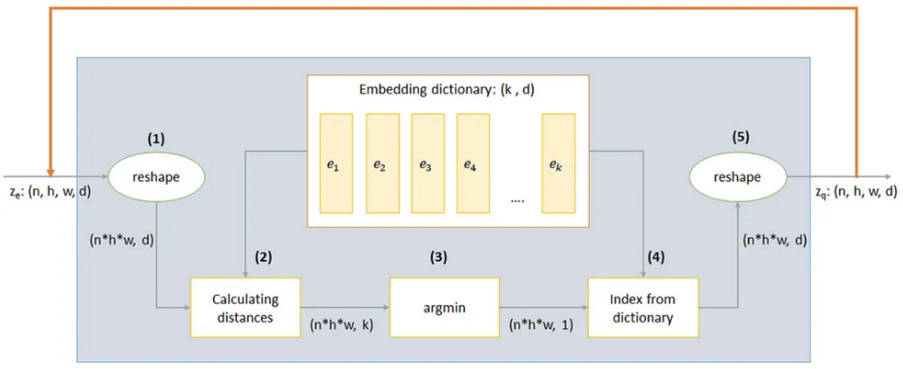

Figure 3: Architecture of the VQ-VAE: quantization layer (source: Medium)

Looking at the previous figures, we can grasp the challenge of backpropagation in such model with discrete prior and posterior. In the next section, we enter in the mathematical definition of the objective and how to train this architecture.

3.2 Framework and optimization objective

Contrary to the vanilla VAE, we have the following categorical distributions assumption:

- The prior $p_\theta(z)$ is categorical. In the original paper, it is taken as uniform supported in $\{1, \dots, K\}$ during the training. When the training is over, it is fit to an autoregressive distribution through a PixelCNN (see [2]). It is left as an exploration research field to be able to learn the prior while training the model.
- The posterior $q_\Phi(z|x)$ is categorical and set to the following:

$$q_\Phi(k|x) = \mathbb{1}_{\{k = arg\min_j \|z_q(x) - e_j\|_2\}}$$

This modelization of the posterior enables to obtain a discrete latent space, but it does not allow to differentiate regarding Φ. As a result, the authors suggest two possible strategies:

- *Straight-through*: propagate the gradient through the discrete part (vector quantization layer) without changing it. The intuition is that the gradient propagated from the encoder contains sufficient information to update the encoder accordingly, but this is just intuition.
- *Subgradient*: compute the subgradient of the quantization layer (unexplored yet)

The optimization objective of the VQ-VAE is based on the ELBO, that one can compute as follow:

$$\begin{aligned} ELBO(q_\Phi(z|x), p_\theta(z|x)) &= \mathbb{E}_{q_\Phi(z|x)}[\log p_\theta(x, z)] - \mathbb{E}_{q_\Phi(z|x)}[\log q_\Phi(z|x)] \\ &= \mathbb{E}_{q_\Phi(z|x)}[\log p_\theta(x|z)] - D_{KL}[q_\Phi(z|X) \| p_\theta(z)] \end{aligned}$$

Notice then that:

- Since $q_\Phi(k|x) = \mathbb{1}_{\{k = arg\min_j \|z_q(x) - e_j\|_2\}}$, we have:

$$\mathbb{E}_{q_\Phi(z|x)}[\log p_\theta(x|z)] = \log p_\theta(x|z_q(x))$$

- Since $Z \sim \mathbb{U}(\{1, ..., K\})$, $\mathbb{P}(Z = k) = \frac{1}{K}$. Also, notice that $q_\Phi(z_q(x)|x) = 1$ by definition of $q_\Phi(z|x)$ and $z_q(x)$. Combining those results, we obtain:

$$\begin{aligned} D_{KL}[q_\Phi(z|x) \| p_\theta(z)] &= \mathbb{E}_{q_\Phi(z|x)}\left[\log \frac{q_\Phi(z|x)}{p_\theta(z)}\right] \\ &= \log \frac{q_\Phi(z_q(x)|x)}{p_\theta(z_q(x))} \\ &= \log K \end{aligned}$$

  Therefore, this KL divergence does not impact the optimization objective as it does not depend on (θ, Φ).

Computing the previous quantities, we would obtain the following suboptimal objective for the VQ-VAE:

$$ELBO(\Phi, \theta) = log p_\theta(x|z_q(x))$$

However, such objective does not enable to learn the dictionary since we use a straight-through approach over the quantization layer. To update the dictionary, the authors suggest to add the following term to the loss:

$$\|sg[z_e(x)] - e\|_2^2$$

Where sg denotes the stop-gradient operator, meaning we do not consider any gradient after the given operation.

Finally, to prevent embedding space over expansion, the authors suggest the addition of a commitment loss parameterized by $\beta > 0$:

$$\beta \|z_e(x) - sg[e]\|_2^2$$

Indeed, the embedding space was unconstrained so far, so its dimension could grow arbitrarily. Intuitively, adding that term forces the model to commit to a given embedding.

Overall, we obtain this final objective to maximize:

$$\mathcal{L}(\theta, \Phi) = \log p_\theta(x|z_q(x)) + \|sg[z_e(x)] - e\|_2^2 + \beta \|z_e(x) - sg[e]\|_2^2$$

3.3 Discussion over some limiting aspects

Even though the previous objective seems natural, we only rigorously justified the first term thanks to the ELBO. Indeed, the dictionary update loss as well as the commitment loss keep dropping out from nowhere, while they seem necessary to ensure the training of our model.

Furthermore, we did not really deal with the non differentiability of the vector quantization loss, and the straight-through estimator can definitely be criticized about what information it actually provides to the encoder to update appropriately.

Finally, the usage of the PixelCNN after the training seems very unnatural, and could significantly boost the model as the PixelCNN did show amazing performances so far.

4 PixelCNN

4.1 Overview

Introducing the VQ-VAE, we have seen that an under-table tool that was being used is the Pixel CNN, first introduced in [2]. In the context of the VQ-VAE, it plays a major role after training by learning the prior $p(z)$, that was set to a discrete uniform previously. This makes a drastic difference with the VAE, as we are not setting the prior ourselves as it's being learnt and modeled by the PixelCNN in this process.

Indeed, the PixelRNN/PixelCNN architectures are sequential deep neural networks that aims at modeling the distribution of a data space in an autoregressive fashion. Their main characteristic is that they take advantage of the structure of an image to learn the data space distribution.

- *PixelRNN*: Bi-directional recurrent networks with variable smart directions are used to model the spatial dependencies between pixels. (3 architectures are presented in the original paper, but our focus will be on the PixelCNN here).
- *PixelCNN*: the dependencies between the pixels is modeled through stacking of masked convolution layers (it's faster since the receptive field is bounded by the size of the convolution), no pooling layer is used. The masking ensures that we keep an autoregressive estimation of a new pixel, without seeing the future pixels.

The following figure illustrates the masked convolution technique.

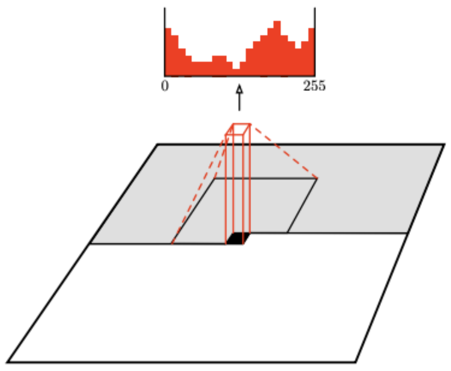

Figure 4: Masked convolution on an image (largest square), the big square represents the receptive field of the black pixel, the white area is masked

Hence, given x an image represented by its pixels $x = (x_1, \ldots, x_n)$, as usual for these distribution modeling problems, we aim at finding $\theta^* = arg\max_\theta p_\theta(x)$ in a family of distributions parameterized by θ: the maximum of likelihood. Contrary to the usual independent framework, we consider a local dependency between pixels given by the receptive field of our convolution. This local dependency is limited to the already seen pixels only as well, thanks to the masked convolution.

Furthermore, rather than using continuous outputs, these architecture use a softmax layer to determine the pixel of a given generation, leading to a discrete prior rather than a continuous one, which is required for the VQ-VAE for instance. As a result, the distribution of a pixel conditionally to the ones in its receptive field is given by a multinomial in $\{0, \dots, 256\}$.

4.2 Usage in the VQ-VAE

Once we have trained the VQ-VAE, the original paper states that we can replace the uniform prior on $p(z)$ by a PixelCNN to model the prior. To perform the training of the PixelCNN, we turn the samples x in their latent representations z, and train the PixelCNN over the latent representations z. This way, we have created dependency between the z in the latent feature mapping, and we obtain a prior over their distribution modeled by the PixelCNN.

5 Bibliography

References

[1] Diederik P Kingma, Max Welling, et al. "An introduction to variational autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392.

[2] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel Recurrent Neural Networks". In: *CoRR* abs/1601.06759 (2016). arXiv: `1601.06759`. URL: `http://arxiv.org/abs/1601.06759`.

[3] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. "Neural Discrete Representation Learning". In: *CoRR* abs/1711.00937 (2017). arXiv: `1711.00937`. URL: `http://arxiv.org/abs/1711.00937`.