Variational Auto-Encoders

VILHES Samy

2023

1 Introduction

Bayesian inference involves computing the posterior distribution, $p(z \mid x)$, given some data x. The posterior is computed using the following Bayesian formula:

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

However, in many models, the likelihood and prior distribution can be complex functions. This makes computing the posterior distribution often infeasible.
The "Auto-Encoding Variational Bayes"[1] paper by Diederik Kingma and Max Welling proposes a solution to this Bayesian inference problem. The main idea is to learn an approximation of the true posterior $p(z \mid x)$. This approximation of the posterior can be used for multiple tasks such as recognition, denoising, representing and visualisation. They introduce Variational Autoencoders (VAE), a generative model capable of approximating the posterior using variational inference and neural networks.

2 Solving the problem

2.1 Key elements

In order to better understand the problem, let's take a look at the key elements we will be using all throughout this paper.

We consider :

- A dataset X = $\{x_1, x_2, ..., x_N\}$ of N i.i.d. variables corresponding to the realisations of the variable x,

- J: the dimension of the latent space,
- $z \in \mathbb{R}^J$: a latent and continuous random variable that generates x,
- The intractable posterior: $p_\theta(z \mid x)$,
- The prior: $p_\theta(z)$ (we suppose $\sim \mathcal{N}(0, I)$),
- The likelihood: $p_\theta(x \mid z)$,
- The evidence or marginal likelihood: $p_\theta(x)$,
- The approximate distribution: $q_\phi(z \mid x)$ (we suppose $\sim \mathcal{N}(\mu, \sigma^2)$ with σ^2 a diagonal matrix).

2.2 Problem Statement

The first step is to estimate the posterior: $p_\theta(z \mid x)$

The posterior follows this formula: $p_\theta(z \mid x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)}$

Which can be rewritten as: $posterior = \frac{likelihood \times prior}{evidence}$

The issue is that the evidence is oftentimes **intractable**, meaning that finding the solution of a problem is infeasible and requires an excessive amount of ressources.

Variational Inference aims to solve this issue by approximating the posterior by a tractable distribution $q_\phi(z \mid x)$.

2.3 Kullback-Leibler Divergence

The Kullback-Leibler Divergence is a statistical method that measures the difference between two probabilistic distributions. It is defined as followed:

$$D_{KL}(p|q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx = E_{p(x)}[log(p(x) - log(q(x)]$$

This measure is not a mathematical distance because it is asymetrical and does not respect the triangle inequality. Moreover, the Kullback-Leibler divergence has the particular property of being **non-negative**.
As we said, our goal is to approach our posterior $p_\theta(z \mid x)$ by the distribution $q_\phi(z \mid x)$. So we are looking for:

$$\phi^\star = Argmin_\phi \; D_{KL}(q_\phi(z \mid x) \mid p_\theta(z \mid x))$$

2.4 Variational Lower Bound of the Margins Likelihood

The log marginal likelihood is $log(p_\theta(x))$. We also want to maximize this quantity. Problem is, we do not have an analytical form of this quantity.

Let:

$$\mathcal{L}(\theta, \phi; x) = E_{q_\phi(z|x)}[log(p_\theta(x \mid z)] - D_{KL}(q_\phi(z \mid x) \mid p_\theta(z))$$

$\mathbf{E_{q_\phi(z|x)}[log(p_\theta(x \mid z)]}$ can be seen as a **reconstruction cost** and $\mathbf{D_{KL}(q_\phi(z \mid x) \mid p_\theta(z))}$ as a **penalty**. It makes sure that $q_\phi(z \mid x)$ doesn't deviate too much from $p_\theta(z)$).

We have:

$$log(p_\theta(x)) = D_{KL}(q_\phi(z \mid x) \mid p_\theta(z \mid x)) + \mathcal{L}(\theta, \phi; x) \tag{1}$$

(We let the proof in Section 5.1)

As we said, the KL divergence is non-negative. Therefore:

$$log(p_\theta(x)) \geq \mathcal{L}(\theta, \phi; x) = E_{q_\phi(z|x)}[log(p_\theta(x \mid z)] - D_{KL}(q_\phi(z \mid x) \mid p_\theta(z))$$

The term on the left is the log-likelihood of the datapoint. The term $\mathcal{L}(\theta, \phi; x)$ is called the variational lower bound or **ELBO**. By maximizing the lower bound, we maximize the log-likelihood since it is intractable.

In order to maximize $\mathcal{L}(\theta, \phi; x)$, we will just minimize the last term $D_{KL}(q_\phi(z \mid x) \mid p_\theta(z))$.

We supposed that $p_\theta(z_i) \sim \mathcal{N}(0, I)$ and $q_\phi(z_i \mid x_i) \sim \mathcal{N}(\mu_i, \sigma_i^2)$. Then we have an analytical solution of this quantity:

$$D_{KL}(q_\phi(z \mid x) \mid p_\theta(z)) = -\frac{1}{2} \sum_{j=1}^{J} 1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2 \tag{2}$$

(We let the proof in Section 5.2)

2.5 The reparameterization Trick

Although the reparameterization trick has been around for some time, it regained attention with the rise of VAEs. Backpropagation can not flow through a random node, and that is a reccurent issue VAEs face. This trick consists in reformulating the samples generated from these stochastic layers as a linear combination of two vectors: a parameter vector that represents the parameters of the stochastic distribution, and a vector of i.i.d. random variables. This allows us to compute the gradient and let the backpropagation flow in order to train models.

Instead of having $z \sim q_\phi(z \mid x)$, z wil be the result of a certain function g : $z = g(\phi, x, \epsilon)$
with $\epsilon \sim p(\epsilon)$

3 Presentation of the Variational Autoencoder

3.1 Introduction

To better understand Variational Autoencoders, a good grasp of Autoencoders is necessary. In short, autoencoders are used to learn a lower-dimension representation of the input data. Often used for compression tasks and dimension reduction, autoencoders compress the data into a representation we refer to as latent space. What happens is that the encoder extracts features/attributes from the input image.

It works in a similar way as our brain. When looking at a face, our brain automatically captures the essential features such as the eyes or nose. AEs do the same thing but these features aren't so explicit, which is why we often refer to latent variables as invisible or unobserved. We can of course infer what each variable represents in the image (in some cases), but more often than not, we won't have a variable that dictates a specific attribute.

Once the features are represented in the latent space (we can think of it as a vector whose elements are features the encoder has captured), it is sent to the decoder which will try and reconstruct the original image just from this vector of features. This results in a slightly different image than the one inputted, mostly due to the fact that the encoder leaves out a lot of information in the process.

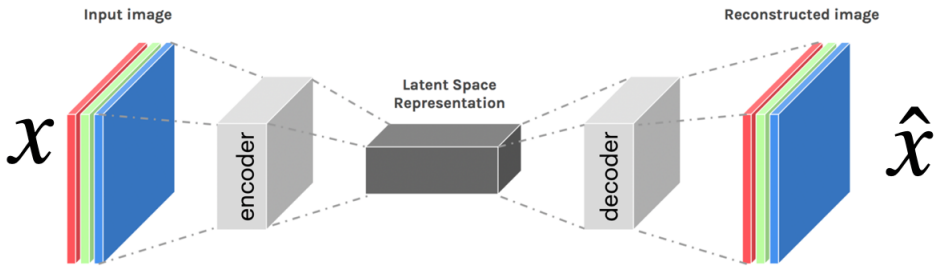


Figure 1: Autoencoder schema

This is a basic schema of an autoencoder. The input image is sent to the encoder which will extract a set number of features, depending on the latent space dimension we want. The decoder then recreates an image based on its representation in the latent space. The input and output image have the same dimensions.

Although Variational Autoencoders follow a similar schema to autoencoders, some changes are made to completely alter its purpose. By combining autoencoder and variational inference discussed previously, VAEs try to learn a compact representation of the data in a continuous space.

What makes VAEs stand out from the autoencoders however, is the generative aspect. VAEs are generative models that learn to generate new data samples similar to the input data.

The autoencoder and the variational autoencoder both use an encoder in order to compress the data into the latent space and a decoder, to construct the output. The difference between these two models resides in the output. While the decoder from the AE tries to minimize the loss reconstruction (the goal is to have the same output as the input), the VAE tries to generate new data that follows the same distribution as the input. By taking a probabilistic approach, instead of mapping the input data to a specific point in the latent space (autoencoder), the encoder of VAEs map the input data to a distribution of latent variables. Since the latent variables follow a prior distribution, VAEs learn a smooth latent space that can easily sampled in order to generate new data samples.

3.2 The architecture

As we said in the previous sections, a VAE is composed of an encoder and a decoder. They are both neural networks.

3.2.1 The encoder

The encoder is a neural network that aims to reduce the dimensions of the input data. It plays the role of $q_\phi(z \mid x)$. We made earlier the assumption that this distribution
$q_\phi(z \mid x^{(i)}) \sim \mathcal{N}(\mu^{(i)}, {\sigma^{(i)}}^2)$. By doing so, the output of the encoder is stochastic.

3.2.2 The reparameterization trick

From now on, this is where the reparameterization trick becomes useful. Instead of having only one output from the encoder representing $q_\phi(z \mid x)$, it will return two parameters μ *and* σ. Reminder: $z = g(\phi, x, \epsilon)$. With our two parameters μ *and* σ, we can now state that $z = \mu + \sigma \odot \epsilon$, with $\epsilon \sim \mathcal{N}(0, I)$
Thanks to this, the backpropagation can flow and the weights of the neural networks can be updated to minimize the loss function.

3.2.3 The decoder

The input of this neural network is the latent vector z. Its goal is to generate, from this latent represention, a new output very similar to the input of the encoder.

3.2.4 The complete architecture

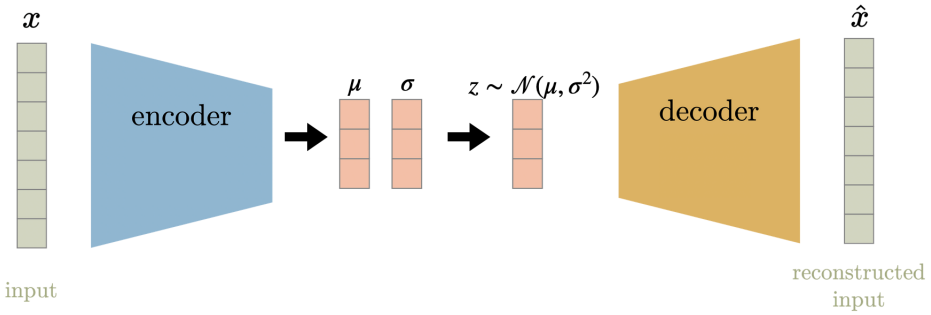


Figure 2: Variational Auto-Encoders schema

3.2.5 The loss function

We want to maximize the log-likelihood which is intractable. To do so, we maximize the variational lower bound $\mathcal{L}(\theta, \phi, x)$. In order to maximize it, we juste have to minimize its opposite (i.e. minimize $-\mathcal{L}(\theta, \phi, x)$). We have an analytical expression of the regularizer term that only needs the output of the encoder μ and σ (seen in 3). For the reconstruction term we can choose a known loss, such as the MSE-Loss or Binary Cross Entropy Loss.
The loss we used in all of our models is:

$$\mathcal{L} = -\frac{1}{2}\sum_{j=1}^{J} 1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2 + \text{BCELoss}(x_{\text{input}}, x_{\text{reconstructed}})$$

3.2.6 Summary

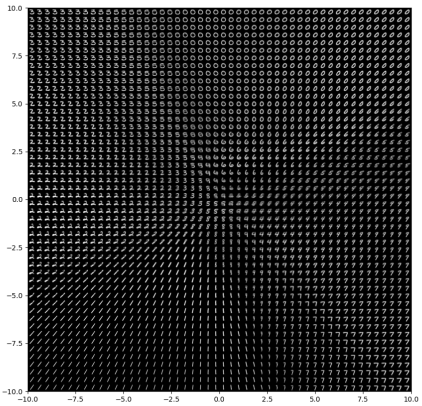
During the training, we feed the VAE the input data. First, they go through the encoder that will return μ and σ (which have the same dimension as the latent space). With the reparameterization trick, we obtain our latent vector z. We give the decoder this vector and obtain the new data very similar to the input.The weights of these two networks are updated using backpropagation to minimize the loss function allowing the network to learn an efficient representation of the input data.

4 Experimentation

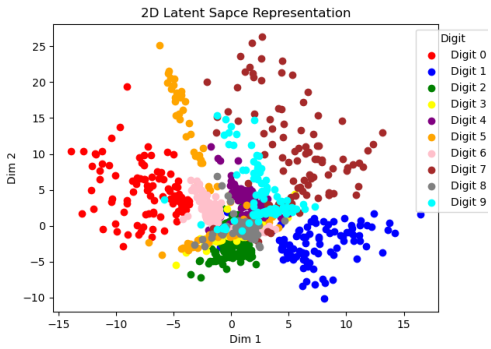
4.1 MNIST Dataset

Our experiments are conducted using the MNIST dataset, a standard benchmark for evaluating machine learning models. Both the encoder and decoder are implemented using neural networks composed of linear layers. The primary goal of these experiments is to analyze and visualize the latent space representation learned by the model. We use a 2D latent space in our experiment to simplify visualization and observe that it is sufficient for generating MNIST digits.

We focus on understanding the distribution of digits in the latent space and how effectively the model captures the underlying structure of the data. Two key visualizations are presented: the distribution of digits in the latent space and examples of generated digits from the decoder.



(a) Examples of generated digits.



(b) Latent space distribution of digits.

Figure 3: Visualization of generated digits and their latent space distribution.

Figure 3a showcases digits generated by decoding random samples from the latent space, illustrating the model's capacity to produce realistic outputs. Figure 3b visualizes the distribution of digits in the latent space, highlighting the clustering of similar digits and the separation between different classes.
The code is available in the github repo: `https://github.com/vilhess/codes/tree/main/vae`

5 Proofs

5.1 Proof of 1

Let:

$$\mathcal{L}(\theta, \phi; x) = E_{q_\phi(z|x)}[log(p_\theta(x \mid z)] - D_{KL}(q_\phi(z \mid x) \mid p_\theta(z))$$

We have:

$$
\begin{aligned}
D_{KL}(q_\phi(z \mid x) \mid p_\theta(z \mid x)) + \mathcal{L}(\theta, \phi; x) = \; & E_{q_\phi(z|x)}[log(q_\phi(z \mid x) - log(p_\theta(z \mid x)] \; + \; E_{q_\phi(z|x)}[log(p_\theta(x \mid z)] \\
& - D_{KL}(q_\phi(z \mid x) \mid p_\theta(z)) \\
= \; & E_{q_\phi(z|x)}[log(q_\phi(z \mid x) - log(p_\theta(z \mid x)] \; + \; E_{q_\phi(z|x)}[log(p_\theta(x \mid z)] \\
& - E_{q_\phi(z|x)}[log(q_\phi(z \mid x) - log(p_\theta(z)] \\
= \; & E_{q_\phi(z|x)}[-log(p_\theta(z \mid x) + log(p_\theta(x \mid z) + log(p_\theta(z)] \\
= \; & -log(p_\theta(z \mid x) + log(p_\theta(x \mid z) + log(p_\theta(z) \\
= \; & log(p_\theta(x)) - log(p_\theta(x, z)) + log(p_\theta(x, z)) - log(p_\theta(z)) + log(p_\theta(z)) \\
= \; & log(p_\theta(x))
\end{aligned}
$$

5.2 Proof of 3

We want to show:

$$
D_{KL}(q_\phi(z \mid x) \mid p_\theta(z)) = -\sum_{j=1}^{J} \frac{1}{2}\left(1 + \log(\sigma_j^2) - \sigma_j^2 - \mu_j^2\right) \tag{3}
$$

We supposed that $p_\theta(z_i) \sim \mathcal{N}(0, I)$ and $q_\phi(z_i \mid x_i) \sim \mathcal{N}(\mu_i, \sigma_i^2)$.

We have:

$$
p_{\mathcal{N}(0,I)}(z) = (2\pi)^{-J/2} \exp\left[-\frac{1}{2} z^T z\right]
$$

$$
p_{\mathcal{N}(\mu,\sigma^2)}(z) = (2\pi)^{-J/2} \det(\sigma^2)^{-1/2} \exp\left[-\frac{1}{2}(z-\mu)^T (\sigma^2)^{-1}(z-\mu)\right]
$$

We recall:

$$
\sigma^2 = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_J^2 \end{bmatrix}
$$

Then:

$$
\det(\sigma^2)^{-1/2} = \left(\prod_{j=1}^{J} \sigma_j^2\right)^{-1/2}
$$

So:

$$
\log p_\theta(z) = \log p_{\mathcal{N}(0,I)}(z) = -\frac{J}{2}\log(2\pi) - \frac{1}{2} z^T z
$$

$$
\log q_\phi(z \mid x) = \log p_{\mathcal{N}(\mu,\sigma^2)}(z) = -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}\log(\sigma_j^2) - \frac{1}{2}(z-\mu)^T(\sigma^2)^{-1}(z-\mu)
$$

Before stepping into the demonstration, we need another property:

If: $X \sim \mathcal{N}(\mu, \Sigma)$ and A is symmetric, then:

$$\mathbb{E}\left[X^T A X\right] = \mu^T A \mu + \operatorname{tr}(A\Sigma) \tag{4}$$

Using the previous equality 4:

$$\begin{aligned}\mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)}\left[z^T z\right] &= \mu^T\mu + \operatorname{tr}(\sigma^2) \\ &= \sum_{j=1}^{J} \mu_j^2 + \sigma_j^2\end{aligned}$$

(the trace of a diagonal matrix is the sum of the diagonal elements).

And:
let $Y = (z - \mu)$; then $Y \sim \mathcal{N}(0, \sigma^2)$, $A = (\sigma^2)^{-1}$ still symmetric.

$$\begin{aligned}\mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)}\left[(z-\mu)^T(\sigma^2)^{-1}(z-\mu)\right] &= \mathbb{E}_{Y\sim\mathcal{N}(0,\sigma^2)}\left[Y^T A Y\right] \\ &= 0^T A 0 + \operatorname{tr}(A\sigma^2) \\ &= \operatorname{tr}((\sigma^2)^{-1}\sigma^2) \\ &= \operatorname{tr}(I) \\ &= J\end{aligned}$$

$$\begin{aligned}D_{KL}(q_\phi(z \mid x) \,||\, p_\theta(z)) &= D_{KL}(\mathcal{N}(z;\mu,\sigma^2) \,||\, \mathcal{N}(z;0,I)) \\ &= \mathbb{E}_{q_\phi(z|x)}\left[\log q_\phi(z \mid x) - \log p_\theta(z)\right] \\ &= \mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)}\left[-\frac{1}{2}\sum_{j=1}^{J}\log(\sigma_j^2) - \frac{1}{2}(z-\mu)^T(\sigma^2)^{-1}(z-\mu) + \frac{1}{2}z^T z\right] \\ &= -\frac{1}{2}\sum_{j=1}^{J}\log(\sigma_j^2) - \frac{1}{2}\mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)}\left[(z-\mu)^T(\sigma^2)^{-1}(z-\mu)\right] + \frac{1}{2}\mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)}\left[z^T z\right] \\ &= -\frac{1}{2}\sum_{j=1}^{J}\log(\sigma_j^2) - \frac{1}{2}J + \frac{1}{2}\sum_{j=1}^{J}\mu_j^2 + \sigma_j^2 \\ &= -\frac{1}{2}\sum_{j=1}^{J}1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2\end{aligned}$$

References

[1] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML]. URL: https://arxiv.org/abs/1312.6114.