Variational Autoencoders

Matteo Lugli

January 1, 2024

1 Introduction

This topic will be quite dense in terms of statistical concepts. I will try to write a summary that is a simple as possible, and I will also link the main resources that I used at the end of the document (see 6). If you are reading this, you should be familiar with deep learning, Bayesan inference and autoencoders.

In section 2 I will introduce generative modeling related to the concept of latent variables. In section 3 a framework to train variational autoencoders is provided, together with the explanation. In chapter 4 I explain how you can use a trained VAE to perform different tasks. In chapter 5 I give a brief introduction to the reparametrization trick, which is crucial to make these kind of models trainable with gradient descent.

2 Latent variables models

The classic autoencoder structure used in (for example) denoising autoencoders is useful to implicitly learn the structure of the training data. The problem is that they do not provide a *explicit* representation of that distribution. If we want to generate new data, we need a to have a way to work with a distribution that it's known and

from which we can sample!
VAEs are also called *Latent Variable Models*, because they make the assumpion that the data can be represented through a combination of **latent variables**, denotes as **z**. If we have dataset made of images, latent variables might be the orientation of the subject, the color of the landscape, ecc... thanks to this assumption, instead of directly sampling from $p(x)$, we sample a set of latent variables from $p(z)$, then we use the produced z to sample from another distribution, $p(x|z)$.
The final objective is to maximize the likelihood of our data under the entire generative process:

$$p(x) = \int p(x|z)p(z)dz \tag{1}$$

this framework is usually called **maximum likelihood**. The problem is that computing that integral is too hard: theoretically we would need to take a infinite number of z, which is obviously impossible. What we might do is using some sort of Monte Carlo method, where we sample many z and only use those. With real world data this method is really uneffective and requires to many samples.
In practice, $p(x|z)$ will be zero for most of the combinations of the latent variables (among all the possible combinations), and will be useless when computing the integral. One of the core ideas behind variational autoencoders is trying to understand how the latent variables are distributed in our dataset, so that we can sample the z from that distribution instead. This way we will compute $p(x)$ with the z that are more likely to have produced x. Let's write the **posterior** that we want to compute, that describes how the latent variables are distributed given the dataset:

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} \tag{2}$$

as we can see this is still intractable because we find $p(x)$ at the denominator. In **variational inference** we try to build a *surrogate* distribution $q(z|x)$ that approximates the *true posterior* p(z|x) by using a much simpler structure (a gaussian). Ideally, we want the surrogate to be as similar as possible to the true posterior, so we want that the KL divergence to be small[1]. Let's write the formula of the distance:

$$D_{KL}(q(z|x), p(z|x)) = E_q\Big[\log q(z|x), \log p(z|x)\Big] \tag{3}$$

$$= E_q\Big[\log q(z|x), \log \frac{p(x|z)p(z)}{p(x)}\Big] \tag{4}$$

$$= E_q\Big[\log q(z|x)\Big] - E_q\Big[\log p(x,z)\Big] + E_q\Big[p(x)\Big] \tag{5}$$

$$= E_q\Big[\log q(z|x)\Big] - E_q\Big[\log p(x,z)\Big] + p(x) \tag{6}$$

From 5 to 6 we removed the expectation on $\log p(x)$ because it is independent from z; let's bring $\log p(x)$ to the left hand side and rewrite the rest:

$$\log p(x) = \underbrace{E_{z\sim q}[\log p(x,z)] - E_{z\sim q}[\log q(z|x)]}_{\textbf{ELBO}} + D_{KL}\Big(q(z|x), p(z|x)\Big) \tag{7}$$

In equation 7 we grouped the first 2 terms of the right hand side under the name of *ELBO*. This term is a **lower bound** of the left hand side, which is what we wanted to aproximate at the beginning. By rewriting it we will see that it is a term that we can compute and work with instead of $p(x)$. For now, notice that if we find a way to maximize it, at the same time we will minimize the KL divergence between the proxy and the posterior: the left hand side is fixed and does not depend on any parameter, and the KL term is always positive because of Jensen inequalities.

Now that we know that by maximizing the ELBO we also make $q(z|x)$ similar to $p(z|x)$, let's re-write the formula to analyze it better:

$$ELBO = E\Big[\log\left(p(x|z)p(z)\right)\Big] - E\Big[\log q(z|x)\Big] \tag{8}$$

$$= E\Big[\log p(x|z)\Big] + E\Big[\log p(z)\Big] - E\Big[\log q(z|x)\Big] \tag{9}$$

$$= E\Big[\log p(x|z)\Big] - D_{KL}(\log q(z|x), p(z)) \tag{10}$$

$$ELBO = \underbrace{E_{z\sim q(z|x)}\Big[\log p(x|z)\Big]}_{\text{Reconstruction term}} - \underbrace{D_{KL}(\log q(z|x), p(z))}_{\text{Regularization term}} \tag{11}$$

[1]Intuitively, it measures the distance between the two distributions

- $p(z)$ is our prior distribution over the latent variables. It is set to be a gaussian: $\mathcal{N}(0, I)$;
- $q(z|x)$ is the proxy of the posterior, modeled as a normal distribution. The parameters of this distribution are the output of the **encoder network**, that taken a x value outputs the mean $\mu(x)$ and the s.t.d σ^2(x) of the distribution instead of the plain encoding of the sample.
- $p(x|z)$ is the distribution that given a set of latent variables z describes the likelihood of each x, modeled as a normal distribution as well. The parameters of this distribution are the output of the **decoder network**.

3 Training procedure

In Figure 1 I try to explain visually the forward pass that is used during the training of the variational autoencoder.

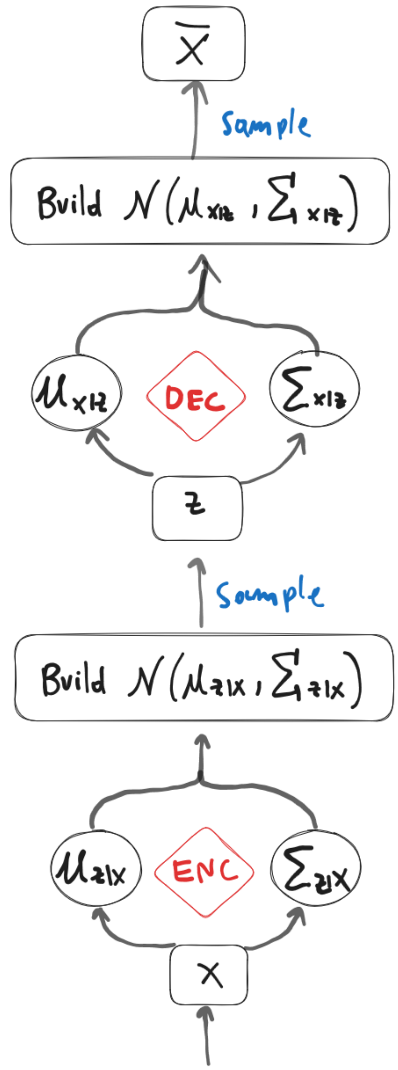

Figure 1: Training forward pass

Ideally we want $\bar{x}$ to be similar to the original sample x, so we can minimize the MSE between the input and the reconstruction: $||x - \bar{x}||^2$ by using gradient descent. All of the operations except for the sampling are derivable, so we can perform standard backpropagation to train the network. To make also the sampling derivable, we use a little trick called **reparametrization trick** (see section 5). We have now seen how we can maximize the reconstruction term by minimizing the MSE between the reconstruction and the real input. To maximize the ELBO, we also want to minimize the regularization term, that represents how much the proxy posterior diverges from the prior $p(z)$. Considering that we model both of them as normal distributions as explained in 2, the regularization term can be written in a closed form solution, and the KL divergence between two gaussians can be easily computed.

4 How to generate new data

If we want to generate variants of a sample, we can just do an entire forward pass and keep $\bar{x}$. If we want to generate data from scratch, we can sample $z \sim \mathcal{N}(0, I)$ and forward the latent variable into the decoder, that will output a new sample similar to the ones in the training dataset. One thing we can also do is calculate what is the probability of the decoder producing an object x. To do that we would need to compute $p(x)$, which we know is intractable. What we can do instead is just compute the ELBO with a forward pass, which is a **lower bound** of $p(x)$, so it is a valid approximation of what we want.

5 Reparametrization trick

As we said, we want each operation that we do for the forward pass to be derivable, so that we can backprop during training. Unfortunately, doing $z \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$ is not a derivable operation. With little effort we can make it derivable: we just sample ϵ from $\mathcal{N}(0, I)$ and do $z = \mu_{z|x} + \epsilon\Sigma_{z|x}$. This way we provide a way for the gradient to backpropagate also through the sampling operator.

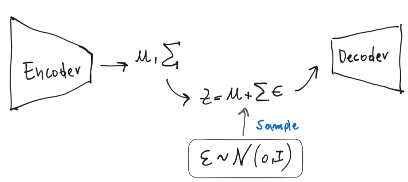

Figure 2: Reparametrization trick

6 References

[1] Doersch, Carl. "Tutorial on variational autoencoders." arXiv preprint arXiv:1606.05908 (2016).

[2] Slides from "Machine Learinng and Deep Learning" course of AI Engineering, Unimore

[3] Stanford lecture on generative models, https://www.youtube.com/watch?v=5WoItGTWV54

[4] Introduction to variational inference, https://www.youtube.com/watch?v=HxQ94L8n0vU