Variational Autoencoders (VAE's)

Lecture 12

Lecturer: Haim Permuter *Scribe: Moshe Bunker*

SUMMARY

In just three years, Variational Autoencoders (VAEs),section IV,[3], [1] have emerged as one of the most popular approaches to unsupervised learning of complicated distributions. VAEs are appealing because they are built on top of standard function approximators (neural networks), and can be trained with stochastic gradient descent. VAEs have already shown promise in generating many kinds of complicated data, including handwritten digits, faces, house numbers CIFAR images, physical models of scenes, segmentation, and predicting the future from static images. This lecture introduces the intuitions behind VAEs, explains the mathematics behind them, and describes some empirical behavior. No prior knowledge of variational Bayesian methods is assumed.

INTRODUCTION

Variational Autoencoders belong to the family of generative models [1]. The generator of VAEs is able to produce meaningful outputs while navigating its continuous latent space. The possible attributes of the decoder outputs are explored through the latent vector. VAEs attempt to model the input distribution from a decodable continuous latent space. Within VAEs, the focus is on the variational inference of latent codes.

Therefore, VAEs provide a suitable framework for both learning and efficient Bayesian inference with latent variables. For example, VAEs with disentangled representations enable latent code reuse for transfer learning. In terms of structure, VAE bears a resemblance to an autoencoder. It is also made up of an encoder (also known as a recognition or inference model) and a decoder (also known as a generative model). Both VAEs and autoencoders attempt to reconstruct the input data while learning the latent

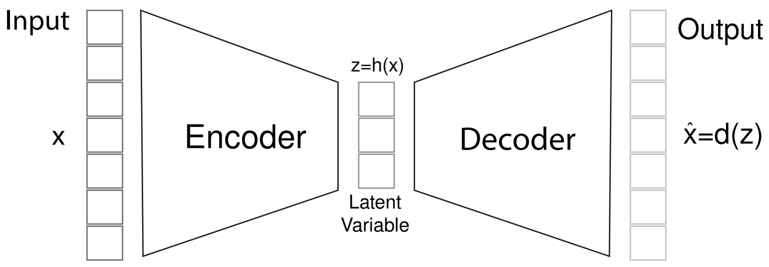

Fig. 1. Illustration of an autoencoder, dimensionality reduction principle can be seen in the diagram

vector. However, unlike autoencoders, the latent space of VAE is continuous, and the decoder itself is used as a generative model.

I. AUTO ENCODER

In its simplest form, an autoencoder [3] will learn the representation or code by trying to copy the input to output. However, using an autoencoder is not as simple as copying the input to output. Otherwise, the neural network would not be able to uncover the hidden structure in the input distribution.An autoencoder will encode the input distribution into a low-dimensional tensor, which usually takes the form of a vector. This will approximate the hidden structure that is commonly referred to as the latent representation, code, or vector. This process constitutes the encoding part. The latent vector will then be decoded by the decoder part to recover the original input.

As a result of the latent vector being a low-dimensional compressed representation of the input distribution, it should be expected that the output recovered by the decoder can only approximate the input. The dissimilarity between the input and the output can be measured by a loss function. But why would we use autoencoders? Simply put, autoencoders have practical applications both in their original form or as part of more complex neural networks. They're a key tool in understanding the advanced topics of deep learning as they give you a low-dimensional latent vector, therefore are used to **dimension reduction** [3].

Definition 1 (dimension reduction) The *dimension reduction* is the process of reducing the number of features that describe some data. This reduction is done either by selection (only some existing features are conserved) or by extraction (a reducednumber of new features are created based on the old features) and can be useful in many situations that require low dimensional data (data visualisation, data storage,heavy computation. . .). Although there exists many different methods ofdimensionality reduction, we can set a global framework that is matched by most of these methods.

Here, we should however keep two things in mind. First, an important dimensionality reduction with no reconstruction loss often comes with a price: the lack ofinterpretable and exploitable structures in the latent space (lack of regularity) [2]. Second, most of the time the final purpose of dimensionality reduction is not to onlyreduce the number of dimensions of the data but to reduce this number of dimensions while keeping the major part of the data structure information in the reducedrepresentations. For these two reasons, the dimension of the latent space and the“depth” of autoencoders (that define degree and quality of compression) have to becarefully controlled and adjusted depending on the final purpose of the dimensionalityreduction.

II. Variational Autoencoders idea

In a generative model, we’re often interested in approximating the true distribution of our inputs using neural networks:

$$z = f(x) \tag{1}$$

In machine learning, to perform a certain level of inference, we’re interested in finding $p_\theta(x, z)$,a joint distribution between inputs - x and latent variables - z, when θ represents the parameters determined during training, The latent variables are not part of the dataset but instead encode certain properties observable from inputs. $p_\theta(x, z)$ is practically a distribution of input data points and their attributes. $p_\theta(x)$ can be computed from the marginal distribution:

$$p_\theta(x) = \int p_\theta(x, z) dz \tag{2}$$

In other words, considering all of the possible attributes, we end up with the distribution that describes the inputs. The problem is that Equation 2 is intractable. The equation does not have an analytic form or an efficient estimator. It cannot be differentiated with respect to its parameters. Therefore, optimization by a neural networkis not feasible. Using Bayes' theorem, we can find an alternative expression for Equation 2:

$$p_\theta(x) = \int p_\theta(x \mid z) p(z) dz \tag{3}$$

When $p(z)$ is a prior distribution over z.

In practice, if we try to build a neural network to approximate $p_\theta(x \mid z)$ without a suitable loss function, it will just ignore z and arrive at a trivial solution, $p_\theta(x \mid z) = p_\theta(x)$. Therefore, Equation 3 does not provide us with a good estimate of $p_\theta(x)$. Alternatively, Equation 2 can also be expressed as:

$$p_\theta(x) = \int p_\theta(z \mid x) p(x) dz \tag{4}$$

However, $p_\theta(z \mid x)$ is also intractable. The goal of a VAE is to find a tractable distribution that closely estimates $p_\theta(z \mid x)$ an estimate of the conditional distribution of the latent attributes, z, given the input, x (later in the lecture we will define a normal distribution).

III. Variational iference

We want to compute $p(z^m \mid x^n)$, but it is very difficult to compute.

$$p(z^m|x^n) = \frac{(z^m, x^n)}{p(x^n)} = \frac{p(z^m)P(x^n|z^n)}{\int P(z^m)p(z^n|z^m)dz^n} \tag{5}$$

However, we can use the inference network to compute an approximate posterior, $q(z^m \mid x^n)$. In this section, we discuss variational inference, which is another optimization-based approach to posterior inference, but which has much more modeling flexibility (and thus can give a much more accurate approximation).variational inference attempts to approximate an intractable probability distribution, such as $p(z^m|x^n)$, with one that is tractable, $q(z^m)$, so as to minimize some discrepancy $\mathcal{D}$ between the distributions:

$$\arg \min_{q(z^m) \in Q_\theta} D\big(q(z^m) \| p(z^m|x^n)\big) \tag{6}$$

where Q is some tractable family of distributions (e.g., multivariate Gaussian). we define $\mathcal{D}$ to be the divergence, then we can derive a lower bound to the log marginal likelihood:

$$\begin{aligned} &= \arg\min\ E_{q(z^m)}[\log q(z^m)] - E_q\Big[\log \frac{p(z^n, x^n)}{p(x^n)}\Big] \\ &= \arg\min\ E_q\big(\log q(z^m)\big) - E_q[\log p(z^m, x^n)] + \log p(x^n) \end{aligned} \tag{7}$$

By using the definition of **evidence lower bound** or **ELBO**:

$$ELBO = E_q[\log p(z^m, x^n)] - E_q\big[\log q(z^m)\big] \tag{8}$$

We can present the expression in Equation 8:

$$\begin{aligned} ELBO &= E_q[\log p(z^m, x^n)] - E_q\big[\log q(z^m)\big] \\ &= E_q[\log p(z^m)] + E_q[\log p(x^n \mid z^m)] - E_q\big[\log q(z^m)\big] \\ &= -D\big(q(z^m) \| p(z^m)\big) + E_q[\log p(x^n \mid z^m)] \end{aligned} \tag{9}$$

Therefore, we will return to Equation 6 and place the development we made in Equation 8 We get that the minimum condition becomes the maximum on the expression:

$$max\Bigg[E_q[\log p(x^n \mid z^m)] - D\big(q(z^m) \| p(z^m)\Bigg] \tag{10}$$

IV. Variational Auto encoders

Let's now make the assumption that $p(z)$ is a standard Gaussian distribution and that $p(x|z)$ is a Gaussian distribution whose mean is defined by a deterministic function f of the variable of z and whose covariance matrix has the form of a positive constant c that multiplies the identity matrix I. The function f is left unspecified for the moment and that will be chosen later. Thus, we have

$$\begin{aligned} &p(x \mid z) \sim \mathcal{N}(f(z), cI), c > 0 \\ &p(z) \sim \mathcal{N}(0, I) \end{aligned} \tag{11}$$

Here we are going to approximate $p(z|x)$ by a Gaussian distribution $q_x(z)$ whose mean and covariance are defined by two functions, g and h, of the parameter x. These two

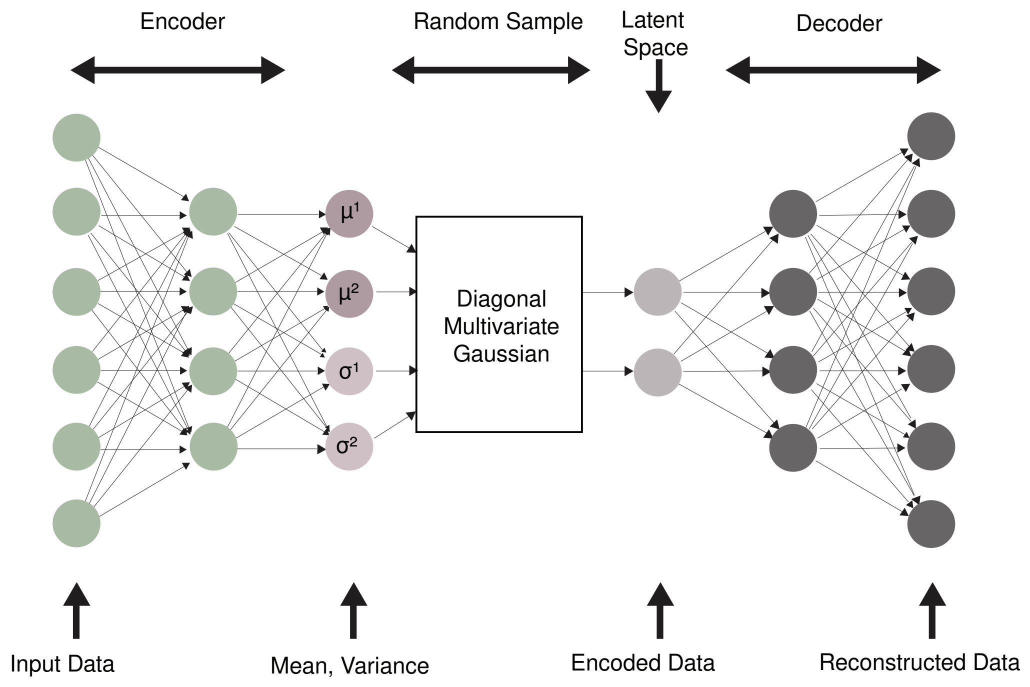

Fig. 2. Schematic illustration of a VAE

functions are supposed to belong, respectively, to the families of functions G and H that will be specified later but that are supposed to be parametrised. Thus we can denote

$$q(z \mid x) \sim \mathcal{N}(g(x), h(x)) \tag{12}$$

So, we have defined this way a family of candidates for variational inference and need now to find the best approximation among this family by optimising the functions g and h (in fact, their parameters) to minimise the divergence between the approximation and the target $p(z|x)$. In other words, we are looking for the optimal $g*$ and $h*$ such that:

$$\begin{aligned}
&(g^*, h^*) = arg\,min_{g,h}\, D\big(q(z)\|p(z|x)\big) \\
&= arg\,min_{g,h}\, E_q\big(\log q(z)\big) - E_q\Big[\log \frac{p(x|z)p(z)}{p(x)}\Big] \\
&= arg\,min_{g,h}\, E_q\big(\log q(z)\big) - E_q\Big[\log p(x,z)\Big] \\
&= arg\,min_{g,h}\, E_q\big(\log q(z)\big) - E_q\Big[\log p(z)\Big] - E_q\Big[\log p(x|z)\Big] \\
&= arg\,min_{g,h}\, D\big(q\|p\big) - E_q\Big[\log p(x|z)\Big] \\
&= arg\,min_{g,h}[-ELBO] \\
&= arg\,min_{g,h}\, E_q\Big[\frac{(x-f(z))^2}{2c}\Big] + D\big(\mathcal{N}(g(x),h(x))\|N(0,I)\big)
\end{aligned} \tag{13}$$

Given the encoder and decoder models, there is one more problem to solve before we can build and train a VAE, the stochastic sampling block, which generates the latent attributes. In the next section, we will discuss this issue and how to resolve it using the reparameterization trick.

V. Reparameterization trick

The left-hand side of Figure 3 below shows the VAE network. The encoder takes the input, and estimates the mean, $g(x)$, and the standard deviation, $h(x)$, of the multivariate Gaussian distribution of the latent vector, z, to reconstruct the input as $\tilde{x}$. This seems straightforward until the gradient updates happen during backpropagation. Backpropagation gradients will not pass through the stochastic Sampling block. While it's fine to have stochastic inputs for neural networks, it's not possible for the gradients to go through a stochastic layer.

The solution to this problem is to push out the Sampling process as the input, as shown on the right side of Figure 3. Then, compute the sample as:

$$z = g(x) + \epsilon * h(x) \tag{14}$$

If ϵ and $h(x)$ are expressed in vector format, then $\epsilon * h(x)$ is element-wise multiplication. Using Equation 14, it appears as if sampling is directly coming from the latent space

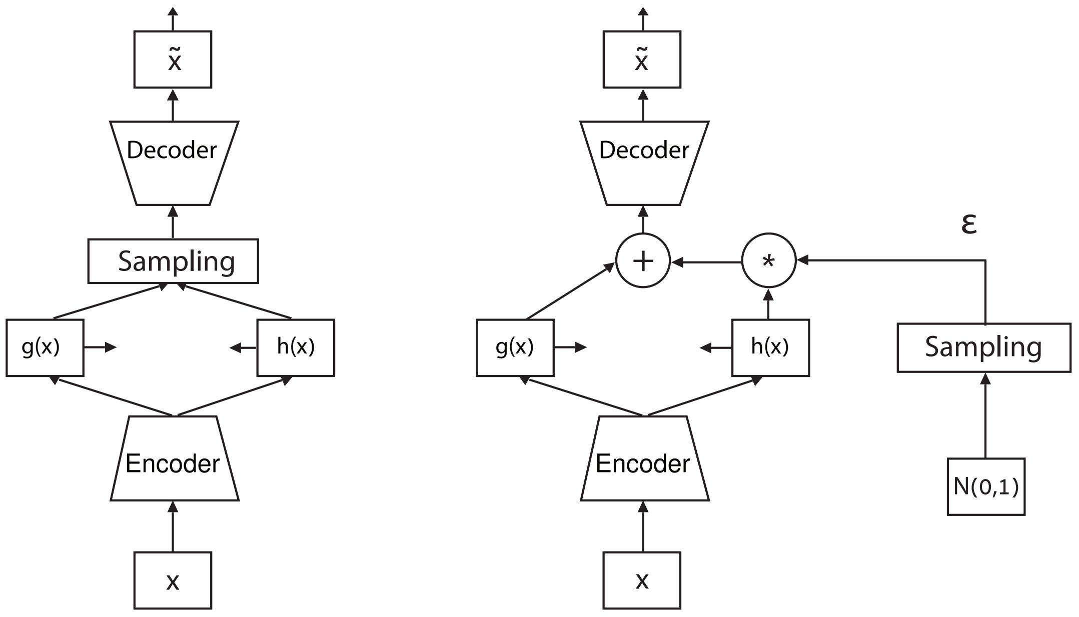

Fig. 3. on the left side A VAE network [2] without the reparameterization trick, on the right is a diagram with the reparameterization trick

as originally intended. This technique is better known as the Reparameterization trick. With Sampling now happening at the input, the VAE network can be trained using the familiar optimization algorithms, that we have learned in previous lectures.

References

[1] Carl Doersch, Carnegie Mellon/ UC Berkeley. *Tutorial on Variational Autoencoders*. Addison-Wesley, Reading, Massachusetts, 1993.

[2] *Rowel Atienzaörper*. [*Advanced Deep Learning with Keras*]. Birmingham - Mumbai, Packt Publishing Ltd,Birmingham B3 2PB, UK, 2018.

[3] Joseph Rocca - Understanding Variational Autoencoders (VAEs), sep 24, 2019. `https://towardsdatascience.com/~understanding-variational-autoencoders-vaes-f70510919f73`