

Variational Autoencoders

Amaires@May 2024

A Variational AutoEncoder (VAE) is an approach to generative modeling. In addition to its capability to generate new samples within the same population as existing ones, it provides a probabilistic way of describing samples in a latent space.

1 K-L Divergence

Generative modeling relies heavily on metrics of similarities between two distributions, among which the most commonly used is called the K-L divergence, short for Kullback–Leibler divergence. It is defined below for two distributions with probability density functions $p_1(x)$ and $p_2(x)$:

$$\text{KL}(p_1(x), p_2(x)) = \int p_1(x) \log \frac{p_1(x)}{p_2(x)} dx \quad (1)$$

K-L divergence has two important properties.

1. It is obvious that K-L divergence is not symmetric in terms of $p_1(x)$ and $p_2(x)$.
2. It is always non-negative, and it is 0 iff $p_1(x)$ and $p_2(x)$ are the same everywhere. To see why, we can break DL-divergence into two parts:

$$\begin{aligned} \text{KL}(p_1(x), p_2(x)) &= \int p_1(x) \log \frac{p_1(x)}{p_2(x)} dx \\ &= \int p_1(x) \log p_1(x) dx - \int p_1(x) \log p_2(x) dx \\ &= - \int p_1(x) \log p_2(x) dx - \left(- \int p_1(x) \log p_1(x) dx \right) \end{aligned} \quad (2)$$

The second term in (2), with the negative sign, is p_1 's information theoretic entropy. The first term, also with the negative sign, is the cross entropy between p_1 and p_2 . The first term is always no smaller than the second term per Gibb's inequality.

2 Intuition

The concept of autoencoders predate the VAE. An autoencoder, shown in Figure 1, consists of an encoder E_ϕ and a decoder D_θ . E_ϕ , a deep neural network parameterized by ϕ , takes a sample x from population \mathbb{X} and maps it to $z = E_\theta(x)$ in \mathbb{Z} . D_θ , another deep neural network parameterized by θ , aiming to reconstruct x , takes z as input and maps it to $\tilde{x} = D_\theta(z) = D_\theta(E_\phi(x))$. \mathbb{Z} is usually of a lower dimension than \mathbb{X} , and thus E_ϕ is considered to possess some compression capability and unsupervised feature extraction capability.

The training of an autoencoder minimizes the reconstruction loss: the expected L_2 distance between x and \tilde{x} :

$$\min_{\theta, \phi} \frac{1}{n} \sum_i \|x_i - \tilde{x}_i\|^2 = \min_{\theta, \phi} \frac{1}{n} \sum_i \|x_i - D_\theta(E_\phi(x_i))\|^2$$

Once trained, the decoder D_θ , to some extent, is already a generative model in that it can create samples in \mathbb{X} given a sample z . The distribution of z or even the range of z , however, is unknown, which prevents its effective sampling. Ideally, we'd like z to follow some

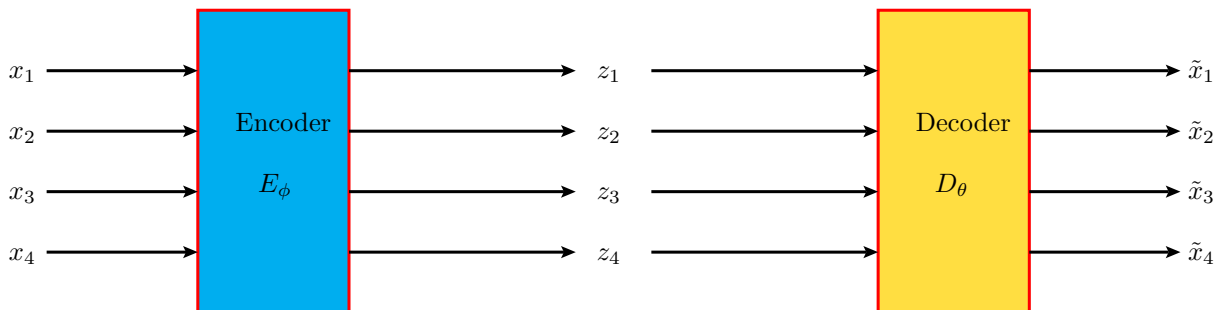


Figure 1: Autoencoder

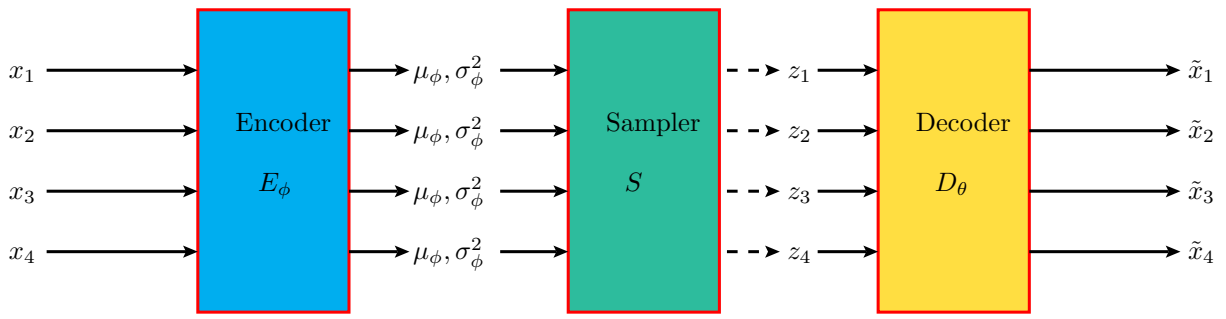


Figure 2: Variational autoencoder

simple distribution, such as $N(0, I)$, as it is easy to sample from. As summarized in Figure 2, VAE makes a few changes to the autoencoder architecture to make D_θ able to take samples from $N(0, I)$ as input and map them to \mathbb{X} .

- Instead of giving out concrete samples in \mathbb{Z} , E_ϕ outputs the parameters for the probability density function $p_\phi(z|x)$.
- $p_\theta(z|x)$ is required to be a multivariate normal distribution with independent components. That is, $p_\theta(z|x) = N(\mu_\phi(x), \sigma_\phi^2(x))$ where $\sigma_\phi^2(x)$ is a diagonal matrix.
- $\mu_\phi(x)$ is penalized for being different from 0, and $\sigma_\phi^2(x)$ for being different from I . With this penalty, $p_\theta(z|x)$ approximately follows $N(0, I)$, so does $p(z)$ as $p(z) = \int p(x)p(z|x)dx = \int p(x)N(z; 0, I)dx = N(z; 0, I)$.
- A new sampler component S is introduced which, given $\mu_\phi(x)$ and $\sigma_\phi^2(x)$, draws a sample $z \sim N(\mu_\phi(x), \sigma_\phi^2(x))$. z is then fed to the decoder D_θ , just as in a regular autoencoder.

How exactly are $\mu_\phi(x)$ and $\sigma_\phi^2(x)$ penalized? Compute the K-L divergence between $N(\mu_\phi(x), \sigma_\phi^2(x))$ and $N(0, I)$ as below:

$$\begin{aligned} \text{KL}(N(u_\phi, \sigma_\phi^2), N(0, I)) &= \int N(u_\phi, \sigma_\phi^2) \log \frac{N(u_\phi, \sigma_\phi^2)}{N(0, I)} \\ &= \frac{1}{2} \sum_{k=1}^d (\mu_{\phi,k}^2 + \sigma_{\phi,k}^2 - \log \sigma_{\phi,k}^2 - 1) \end{aligned} \quad (3)$$

In (3), d is the dimension of \mathbb{Z} . Removing the constants from (3) and estimating it with samples, our final **K-L divergence loss** is

$$\min_{\phi} \frac{1}{n} \sum_i \sum_{k=1}^d (\mu_{\phi,k}^2(x_i) + \sigma_{\phi,k}^2(x_i) - \log \sigma_{\phi,k}^2(x_i)) \quad (4)$$

The reconstruction loss for VAE, is also slightly different from that for a regular autoencoder. It can be estimated by the following equation, given a function $S(\mu, \sigma^2)$ that returns a sample from $N(\mu, \sigma^2)$.

$$\min_{\theta, \phi} \frac{1}{n} \sum_i \|x_i - \tilde{x}_i\|^2 = \min_{\theta, \phi} \frac{1}{n} \sum_i \|x_i - D_\theta(S(\mu_\phi(x_i), \sigma_\phi^2(x_i)))\|^2$$

This formulation has one big problem. $S(\cdot, \cdot)$ is not differentiable, which makes the reconstruction loss not amenable to back-propagation based optimization. Luckily, it is easy to rewrite $S(\mu_\phi(x_i), \sigma_\phi^2(x_i))$ as $\mu_\phi(x_i) + S(0, I) \odot \sigma_\phi(x_i)$, where \odot is the element-wise product and $\sigma_\phi(x_i)$ is $\sigma_\phi^2(x_i)$'s diagonals arranged in a vector form, by leveraging the reparameterization trick for normal distributions. The final formulation for the **reconstruction loss** therefore is

$$\min_{\theta, \phi} \frac{1}{n} \sum_i \|x_i - \tilde{x}_i\|^2 = \min_{\theta, \phi} \frac{1}{n} \sum_i \|x_i - D_\theta(\mu_\phi(x_i) + S(0, I) \odot \sigma_\phi(x_i))\|^2 \quad (5)$$

The **total loss** combines the K-L divergence loss in (4) and the reconstruction loss in (5) with a weight hyperparameter λ :

$$\min_{\theta, \phi} \left(\frac{1}{n} \sum_i \|x_i - D_\theta(\mu_\phi(x_i) + S(0, I) \odot \sigma_\phi(x_i))\|^2 \right) + \lambda \frac{1}{n} \sum_i \sum_{k=1}^d (\mu_{\phi,k}^2(x_i) + \sigma_{\phi,k}^2(x_i) - \log \sigma_{\phi,k}^2(x_i)) \quad (6)$$

λ controls the relative importance between reconstructing the original samples and making sure z follows $N(0, I)$. It is likely that different data sets require different λ .

In practice, instead of outputting $\sigma_\phi^2(x_i)$, E_ϕ outputs $\log \sigma_\phi^2(x_i)$, but that is only a minor engineering detail.

3 Bayesian View

This section derives the total loss objective function through a Bayesian view.

The maximum likelihood method is often used to optimize a neural network that takes samples x_i as input and produces $p_\theta(x_i)$. Assuming each of these x_i are i.i.d samples, the likelihood of observing all of them is $p(x_1, x_2, x_3, \dots, x_n) = \prod p_\theta(x_i)$. The training objective is to maximize $\prod p_\theta(x_i)$, which is equivalent to minimizing the expected negative log odds:

$$\min_{\theta} -\frac{1}{n} \sum_i \log p_\theta(x_i)$$

Considering for now only the decoder D_θ part of VAE. It maps a sample z to \tilde{x} , but it can also be viewed as spitting out parameters for $p_\theta(x|z)$. More specifically it spits out $\mu_\theta(z)$, the parameters in $N(x; \mu_\theta(z), I)$. If the maximum likelihood method is to be used for finding the optimal θ , $p_\theta(x)$ is needed which can be calculated this way: $p_\theta(x) = \int p_\theta(x, z) dz = \int p(z) p_\theta(x|z) dz = \mathbf{E}_{z \sim p(z)} p_\theta(x|z)$. Estimating $p_\theta(x)$ this way, however, is intractable due to the number of dimensions z potentially has.

Assuming there is an effective way of sampling z that follows a distribution $p_\phi(z|x)$, which may or may not be equal to $p_\theta(z|x)$, $p_\theta(x)$ can be calculated the following way:

$$p_\theta(x) = \int p_\theta(x, z) dz = \int p_\phi(z|x) \frac{p_\theta(x, z)}{p_\phi(z|x)} dz = \mathbf{E}_{z \sim p_\phi(z|x)} \frac{p_\theta(x, z)}{p_\phi(z|x)}$$

Note that in the derivation above, the only requirement of $p_\phi(z|x)$ is to be a valid probability density function. Is there such a $p_\phi(z|x)$ that is easy to sample from? Yes, that is exactly the responsibility of VAE's encoder E_ϕ which takes in x and spits out the parameters for normal distribution $p_\phi(z|x)$: $\mu_\phi(x)$ and $\sigma_\phi^2(x)$.

With $p_\theta(x)$ estimated this way, $\log p_\theta(x)$ becomes:

$$\log p_\theta(x) = \log \mathbf{E}_{z \sim p_\phi(z|x)} \frac{p_\theta(x, z)}{p_\phi(z|x)}$$

3.1 Change of Optimization Objective

With all the derivation steps, it is still not clear how to calculate $\log p_\theta(x)$ precisely.

Given $\log(\cdot)$ is a concave function, Jensen's inequality states that $\log \mathbf{E}(x) \geq \mathbf{E}(\log(x))$. We thus have:

$$\log p_\theta(x) = \log \mathbf{E}_{z \sim p_\phi(z|x)} \frac{p_\theta(x, z)}{p_\phi(z|x)} \geq \mathbf{E}_{z \sim p_\phi(z|x)} \log \frac{p_\theta(x, z)}{p_\phi(z|x)}$$

It is now possible to estimate the right hand side of the inequality, broadly known as the **Evidence Lower BOund** (ELBO), which can be rewritten further:

$$\begin{aligned} \text{ELBO} &= \mathbf{E}_{z \sim p_\phi(z|x)} \log \frac{p_\theta(x, z)}{p_\phi(z|x)} \\ &= \mathbf{E}_{z \sim p_\phi(z|x)} \log \frac{p_\theta(x|z)p(z)}{p_\phi(z|x)} \\ &= \mathbf{E}_{z \sim p_\phi(z|x)} [\log p_\theta(x|z) + \log p(z) - \log p_\phi(z|x)] \\ &= \mathbf{E}_{z \sim p_\phi(z|x)} \log p_\theta(x|z) - \text{KL}(p_\phi(z|x), p(z)) \end{aligned} \tag{7}$$

Since $p(z) = N(0, I)$, the second term in (7), as already calculated by (3) in Section 2, is:

$$\text{KL}(p_\phi(z|x), p(z)) = \frac{1}{2} \sum_{k=1}^d (\mu_{\phi,k}^2 + \sigma_{\phi,k}^2 - \log \sigma_{\phi,k}^2 - 1)$$

At the beginning of this section, $p_\theta(x|z)$ is already required to take the form of $N(\mu_\theta(z), I)$ which means:

$$\log p_\theta(x|z) = \log(2\pi)^{-\frac{d}{2}} \exp(-\frac{1}{2} \|x_i - \mu_\theta(z)\|^2) = \text{const} - \frac{1}{2} \|x_i - \mu_\theta(z)\|^2$$

If in the process of estimating $\mathbf{E}_{z \sim p_\phi(z|x)} \log p_\theta(x|z)$, only one single sample z drawn which is equal to $\mu_\phi(x_i) + S(0, I) \odot \sigma_\phi(x_i)$, the first term in (7) gives

$$\mathbf{E}_{z \sim p_\phi(z|x)} \log p_\theta(x|z) \approx \text{const} - \frac{1}{2} \|x_i - \mu_\theta(\mu_\phi(x_i) + S(0, I) \odot \sigma_\phi(x_i))\|^2$$

Putting the maximum likelihood and the two terms of the ELBO together, we've arrived at:

$$\begin{aligned}
\min_{\theta} -\frac{1}{n} \sum_i \log p_{\theta}(x_i) &\leq \min_{\theta, \phi} -\frac{1}{n} \sum_i \text{ELBO} \\
&= \min_{\theta, \phi} -\frac{1}{n} \sum_i (\text{const} - \frac{1}{2} \|x_i - \mu_{\theta}(\mu_{\phi}(x_i) + S(0, I) \odot \sigma_{\phi}(x_i))\|^2 + \text{const} - \frac{1}{2} \sum_{k=1}^d (\mu_{\phi, k}^2 + \sigma_{\phi, k}^2 - \log \sigma_{\phi, k}^2)) \\
&= \text{const} + \frac{1}{2} \cdot \frac{1}{n} \sum_i (\|x_i - \mu_{\theta}(\mu_{\phi}(x_i) + S(0, I) \odot \sigma_{\phi}(x_i))\|^2 + \sum_{k=1}^d (\mu_{\phi, k}^2 + \sigma_{\phi, k}^2 - \log \sigma_{\phi, k}^2))
\end{aligned}$$

Removing the constants and $\frac{1}{2}$, our final optimization objective shown above is identical to (6) in Section 2, keeping in mind that

- μ_{θ} and D_{θ} are the same function, and
- with this theoretical foundation, the need for a λ is also eliminated.

3.2 The ELBO gap

Since the optimization objective is changed from the log likelihoods to the ELBO, it is helpful to understand the gap between the two.

$$\begin{aligned}
\log p_{\theta}(x) - \mathbf{E}_{z \sim p_{\phi}(z|x)} \log \frac{p_{\theta}(x, z)}{p_{\phi}(z|x)} &= \mathbf{E}_{z \sim p_{\phi}(z|x)} (\log p_{\theta}(x) - \log \frac{p_{\theta}(x, z)}{p_{\phi}(z|x)}) \\
&= \mathbf{E}_{z \sim p_{\phi}(z|x)} \log \frac{p_{\theta}(x) p_{\phi}(z|x)}{p_{\theta}(x, z)} \\
&= \mathbf{E}_{z \sim p_{\phi}(z|x)} \log \frac{p_{\phi}(z|x)}{p_{\theta}(z|x)} \\
&= \text{KL}(p_{\phi}(z|x), p_{\theta}(z|x)) \\
&\geq 0
\end{aligned}$$

The gap is 0 when $p_{\phi}(z|x)$ and $p_{\theta}(z|x)$ are identical.

4 Joint Distribution View

Section 3's derivation starts from the maximum likelihood objective, and then switches to maximizing the ELBO. This section provides a simpler joint distribution approach to derive the ELBO objective directly, inspired by Jianlin Su at <http://kexue.fm>.

In VAE, once trained, the decoder D_{θ} can be used as an independent generative model, without the encoder. The encoder can also be used without the decoder as a discriminative model. The training process is what links both components together. It is reasonable to require them to work on the same distribution about both x and z . That is, our objective is to minimize the K-L divergence between $p_{\theta}(x, z)$ and $p_{\phi}(x, z)$:

$$\begin{aligned}
\min_{\phi, \theta} \text{KL}(p_\phi(x, z), p_\theta(x, z)) &= \min_{\phi, \theta} \int \int p_\phi(x, z) \log \frac{p_\phi(x, z)}{p_\theta(x, z)} dz dx \\
&= \min_{\phi, \theta} \int \int [p(x) p_\phi(z|x) \log \frac{p(x) p_\phi(z|x)}{p_\theta(x, z)} dz] dx \\
&= \min_{\phi, \theta} \int p(x) \left[\int p_\phi(z|x) \log \frac{p(x) p_\phi(z|x)}{p_\theta(x, z)} dz \right] dx \\
&= \min_{\phi, \theta} E_{x \sim p(x)} \left[\int p_\phi(z|x) \log \frac{p(x) p_\phi(z|x)}{p_\theta(x, z)} dz \right] \\
&= \min_{\phi, \theta} \left[E_{x \sim p(x)} \int p_\phi(z|x) \log p(x) dz + E_{x \sim p(x)} \int p_\phi(z|x) \log \frac{p_\phi(z|x)}{p_\theta(x, z)} dz \right] \\
&= \min_{\phi, \theta} \left[E_{x \sim p(x)} \log p(x) + E_{x \sim p(x)} \int p_\phi(z|x) \log \frac{p_\phi(z|x)}{p_\theta(x, z)} dz \right] \\
&= \min_{\phi, \theta} [\text{const} + E_{x \sim p(x)} \int p_\phi(z|x) \log \frac{p_\phi(z|x)}{p_\theta(x, z)} dz] \\
&= \text{const} + \min_{\phi, \theta} E_{x \sim p(x)} \int p_\phi(z|x) \log \frac{p_\phi(z|x)}{p_\theta(x, z)} dz \\
&= \text{const} + \min_{\phi, \theta} E_{x \sim p(x)} \int p_\phi(z|x) \log \frac{p_\phi(z|x)}{p_\theta(x, z)} dz \\
&= \text{const} + E_{x \sim p(x)} \int p_\phi(z|x) \log \frac{p_\phi(z|x)}{p_\theta(x|z)p(z)} dz \\
&= \text{const} + E_{x \sim p(x)} [-E_{z \sim p(z|x)} \log p_\theta(x|z) + E_{z \sim p(z|x)} \log \frac{p_\phi(z|x)}{p(z)}] \\
&= \text{const} + E_{x \sim p(x)} [-E_{z \sim p(z|x)} \log p_\theta(x|z) + \text{KL}(p_\phi(z|x), p(z))] \\
&= \text{const} + E_{x \sim p(x)} [-\text{ELBO}]
\end{aligned}$$

The definition of the ELBO in Section 3 can be used to verify this.

5 Latent Space

In VAE's training process, $p(x)$ and $p(z) = N(0, I)$ are given, VAE learns $p_\phi(z|x)$ and $p_\theta(x|z)$ simultaneously. Note however that $p_\theta(x)$ is never directly optimized to match $p(x)$. This could be one major reason why VAE is not known to generate very realistic images.

VAE's encoder, on the other hand, is a very reasonable feature extraction tool. Suppose there are a bunch of sample human face pictures labelled with whether the person has large eyes or not. Denote these samples by (x, y) where x is the image, and $y = 1$ if the person has large eyes and 0 otherwise. A vector e in \mathbb{Z} calculated the following way probably captures the latent representation of large eyes.

$$e = E_{x \sim p(x|y=0)} \mu_\phi(x) - E_{x \sim p(x|y=0)} \mu_\phi(x)$$

Given any human face picture x , $\mu_\theta(x + \lambda e)$ should generate a variation of x that has big or small eyes as λ varies.