Explaination of Variational Autoencoder and How to Implement It

Duc-Anh Nguyen

August 2022

1 Background

1.1 Continuous Random Variable

The continuous random variable $\mathbf{X}$ is written as a column vector

$$\mathbf{X} = [X_0, X_1, ..., X_{k-1}]^T \tag{1}$$

where k is the number of dimensions and X_i is a real value.

Let's consider two cases of $\mathbf{X}$:

- If $\mathbf{X}$ is specific, we denote $P(X)$ as the probability of $\mathbf{X}$.
- If $\mathbf{X}$ is general, we denote $P(X)$ as the distribution of $\mathbf{X}$. This distribution has it own parameters. For example, if $P(X)$ is Gaussian $N(\mu, \Sigma)$, we need to find μ and Σ. However, finding true μ and Σ is very difficult. We mitigate the difficulty by approximately $P(\mathbf{X})$.

1.2 Expected Value E

The expected value of a continuous random variable $\mathbf{X}$ is defined as:

$$E[\mathbf{X}] = \int_x P(x) \log P(x) dx \tag{2}$$

1.3 Determinant of Squared Matrix

Consider a squared matrix $\mathbf{A}^{n \times n}$, the determinant of $\mathbf{A}$ is a real number, denoted by $det(\mathbf{A})$ or $|\mathbf{A}|$. The main diagonal of $\mathbf{A}$ is from the upper left to the lower right.

If $\mathbf{A}$ is diagonal (i.e., the entries outside the main diagonal are all zero), its determinant is the product of elements of its main diagonal:

$$det(\mathbf{A}) = \prod_{i \in [0,n)} a_i^i \tag{3}$$

where a_i^i is the i-th element on the main diagonal.

Briefly, the determinant of a squared matrix could be considered as a function which converts A to a real value.

1.4 Trace of Squared Matrix

Consider a squared matrix $\mathbf{A}^{n\times n}$, the trace of $\mathbf{A}$ is a real number, denoted by $tr(\mathbf{A})$. It is defined to be the sum of elements on the main diagonal.

1.5 KL Divergence

1.5.1 Formula

Given two multivariate continuous distributions Q_1 and Q_2, we need to estimate the cost of transformation from Q_1 to Q_2. The intuition of such estimation is that if Q_1 and Q_2 are the same, the cost should be zero. If Q_1 and Q_2 are very different, the cost should be large enough.

The formula of the KL divergence for Q_1 and Q_2 is stated as follows:

$$D_{KL}[Q_1||Q_2] = \int_x Q_1(x) \cdot \log \frac{Q_1(x)}{Q_2(x)} dx \tag{4}$$

where notion $||$ is the separation between two comparable distributions.

From Equation 4, we easily see that:

$$D_{KL}[Q_1||Q_2] \neq D_{KL}[Q_2||Q_1] \tag{5}$$

We can rewrite Equation 4 in other form:

$$D_{KL}[Q_1||Q_2] = \int_x Q_1(x) \cdot (\log Q_1(x) - \log Q_2(x)) dx \tag{6}$$

$$D_{KL}[Q_1||Q_2] = \int_x Q_1(x) \cdot \log Q_1(x) - \int_x Q_1(x) \cdot \log Q_2(x) dx \tag{7}$$

$$D_{KL}[Q_1||Q_2] = E[Q_1(x)] - \int_x Q_1(x) \cdot \log Q_2(x) dx \tag{8}$$

1.5.2 Gaussian Distribution Computation

If $Q_1 = N(\mu_1, \Sigma_1)$ and $Q_2 = N(\mu_2, \Sigma_2)$, KL divergence is defined as:

$$D_{KL}[Q1||Q_2] = \frac{1}{2}\left(tr(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1) - k + \log_e \frac{|\Sigma_2|}{|\Sigma_1|} \right) \tag{9}$$

where

- k is the dimension of the vector space (i.e., the length of μ_1),

- $tr(.)$: compute the trace of a squared matrix, and
- $|.|$: compute the determinant of a squared matrix.

If $Q_1 = N(\mu_1, \sigma_1^2)$ and $Q_2 = N(0, \mathbf{I})$, KL divergence is defined as:

$$D_{KL}[Q1||Q_2] = \frac{1}{2}\left(tr(\Sigma_2^{-1}\Sigma_1) + \mu_1^2 - k + \log_e|\Sigma_2| - \log_e|\Sigma_1|\right) \tag{10}$$

$$D_{KL}[Q1||Q_2] = \frac{1}{2}\left(tr(\Sigma_2^{-1}\Sigma_1) + \mu_1^2 - k - \log_e|\Sigma_1|\right) \tag{11}$$

1.6 Sampling from Multivariate Gaussian Distribution

Given a multivariate gaussian distribution $N(\mu, \Sigma)$, we could draw a sample from this distribution:

$$\mu + \Sigma \cdot \beta \tag{12}$$

where β is sampled from $N(0, 1)$.

2 Variational Autoencoder Explanation

Given a random variable $\mathbf{X} = [X_0, X_1, ..., X_{k-1}]^T$, we need to compute the true distribution of this dataset, denoted by $P(X)$. In reality, it is very hard to find $P(\mathbf{X})$. Otherwise, we try to approximate $P(\mathbf{X})$.

We can write $P(\mathbf{X})$ as follows: $P(\mathbf{X}) = \int_z P(\mathbf{X}|z)P(z)dz$ where

- z: the latent space (i.e., we transform $\mathbf{X}$ to a latent space z)
- $P(\mathbf{X}|z)$: the probability of $\mathbf{X}$ given z

Now, to compute $P(\mathbf{X})$, we need to compute $P(\mathbf{X}|z)$ and $P(z)$. After some operations, we finally have:

$$\log P(\mathbf{X}) - D_{KL}[Q(z|\mathbf{X}) \,||\, P(z|\mathbf{X})] = E_{z\sim Q}[\log P(\mathbf{X}|z)] + D_{KL}[P(z)||Q(z|\mathbf{X})] \tag{13}$$

where:

- Q is a standard normal random distribution (i.e., $Q = N(0, \mathbf{I})$)
- $\log P(\mathbf{X})$: we need to find parameters to maximize the distribution of $P(\mathbf{X})$, or $\log P(\mathbf{X})$
- $D_{KL}[Q(z|\mathbf{X}) \,||\, P(z|\mathbf{X})]$: error term of $\log P(\mathbf{X})$. We could ignore it when implementing the formula.
- $P(\mathbf{X}|z)$ transforms a vector in the latent space z to X (i.e., decoder)
- $Q(z|\mathbf{X})$: transforms the input X to a vector in latent space z(i.e., encoder)

- $z \sim Q$: vector z follows the distribution of Q
- $E_{z \sim Q}[\log P(\mathbf{X}|z)]$ (reconstruction loss): for many $z \sim Q$, we compute the average of $\log P(\mathbf{X}|z)$. From a specific z, we would generate a corresponding reconstruction and it should be similar to the input $\mathbf{X}$. To compare the difference between the input and the output of the variational autoencoder, we could use a distance metric such as binary cross-entropy (for each pixel we apply binary cross-entropy one time - preferable method), mean square error, etc.

  To approximately compute $\log P(\mathbf{X}|z)$, we draw z from standard normal random distribution $Q = N(0, \mathbf{I})$: $z = \mu + \Sigma \cdot \beta$ where β is sampled from $N(0, 1)$.
- $D_{KL}[P(z)||Q(z|\mathbf{X})]$ (KL loss): In other words, we compute

$$D_{KL}[P(z) \sim N(\mu, \Sigma) \;||\; N(0, \mathbf{I})]$$

3 Implementation

The implementation of the variational autoencoder is published in this repository. A variational autoencoder includes two parts:

- Encoder: convert the input $\mathbf{X}$ to a multivariate gaussian distribution in the latent space z,
- Sampling: draw a sample from the current latent space, and
- Decoder: reconstruct the input.

I designed the variational autoencoder to work on all sizes of input images such as MNIST (28×28), CIFAR-10 ($32 \times 32 \times 3$), etc.

4 Testing the Trained Variational Autoencoder

From the trained variational autoencoder, we conduct the following steps:

- Step 1: Get the decoder. Note that the input of the decoder is a vector in the latent space z, in which z follows $N(0, I)$.
- Step 2: Draw a sample of z
- Step 3: Pass z to the decoder to get the reconstruction.

For example, we conducted an experiment on MNIST. The VAE is trained on the training set with 2000 epochs. The size of dimensions in the latent space is 8. During the test phase, I generate a univariate gaussian vector of size 8 as follows: z = [0.2791389 1.21881375 -0.45776135 0.54741264 1.73804693 -1.88973165 -0.54314976 0.50457552]

The reconstruction of the decoder from the above z is as follows:

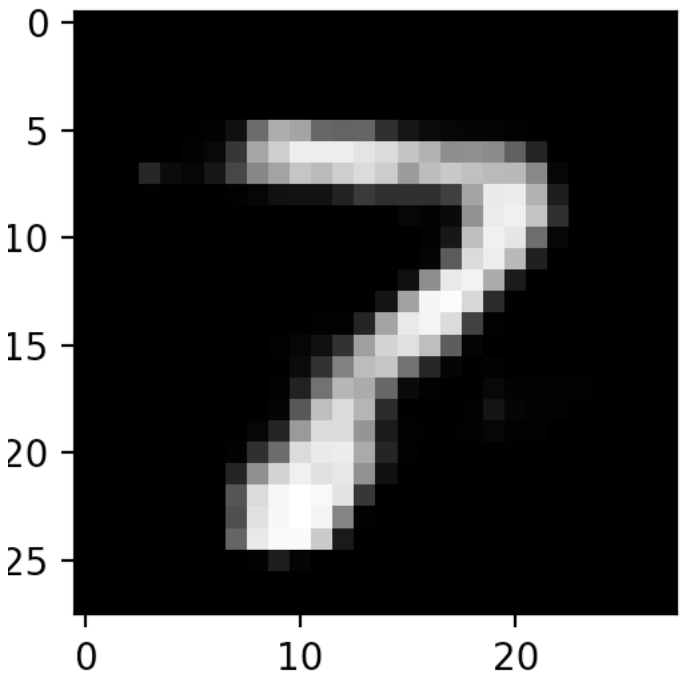

Figure 1: An example of generating image from a vector in the latent space z.